Package 'psych'

June 28, 2016

Version 1.6.6

Date 2016-06-28

Title Procedures for Psychological, Psychometric, and Personality Research

Author William Revelle <revelle@northwestern.edu>

Maintainer William Revelle <revelle@northwestern.edu>

Description A general purpose toolbox for personality, psychometric theory and experimental psychology. Functions are primarily for multivariate analysis and scale construction using factor analysis, principal component analysis, cluster analysis and reliability analysis, although others provide basic descriptive statistics. Item Response Theory is done using factor analysis of tetrachoric and polychoric correlations. Functions for analyzing data at multi-levels include within and between group statistics, including correlations and factor analysis. Functions for simulating particular item and test structures are included. Several functions serve as a useful front end for structural equation modeling. Graphical displays of path diagrams, factor analysis and structural equation models are created using basic graphics. Some of the functions are written to support a book on psychometrics as well as publications in personality research. For more information, see the personality-project.org/r webpage.

License GPL (>= 2)

Imports mnormt,parallel,stats,graphics,grDevices,methods

Suggests GPArotation, sem, lavaan, Rcsdp, graph, Rgraphviz

LazyData true

URL http://personality-project.org/r/psych

http://personality-project.org/r/psych-manual.pdf

NeedsCompilation no

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2016-06-28 23:40:54

R topics documented:

`00.psych` *A package for personality, psychometric, and psychological research*

Description

Overview of the psych package.

The psych package has been developed at Northwestern University to include functions most useful for personality and psychological research. Some of the functions (e.g., `read.clipboard`, `describe`, `pairs.panels`, `error.bars`) are useful for basic data entry and descriptive analyses.

Use help(package="psych") for a list of all functions. Two vignettes are included as part of the package. The overview provides examples of using psych in many applications.

Psychometric applications include routines (fa for principal axes (fm="pa"), minimum residual (fm="minres"), maximum likelihood (fm="mle") and weighted least squares (fm="wls") factor analysis as well as functions to do Schmid Leiman transformations (schmid) to transform a hierarchical factor structure into a bifactor solution. Factor or components transformations to a target matrix include the standard Promax transformation (Promax), a transformation to a cluster target, or to any simple target matrix (target.rot) as well as the ability to call many of the GPArotation functions. Functions for determining the number of factors in a data matrix include Very Simple Structure (VSS) and Minimum Average Partial correlation (MAP). An alternative approach to factor analysis is Item Cluster Analysis (ICLUST). Reliability coefficients alpha (score.items, score.multiple.choice), beta (ICLUST) and McDonald's omega (omega and omega.graph) as well as Guttman's six estimates of internal consistency reliability (guttman) and the six measures of Intraclass correlation coefficients (ICC) discussed by Shrout and Fleiss are also available.

The scoreItems, and score.multiple.choice functions may be used to form single or multiple scales from sets of dichotomous, multilevel, or multiple choice items by specifying scoring keys.

Additional functions make for more convenient descriptions of item characteristics. Functions under development include 1 and 2 parameter Item Response measures. The tetrachoric, polychoric and irt.fa functions are used to find 2 parameter descriptions of item functioning.

A number of procedures have been developed as part of the Synthetic Aperture Personality Assessment (SAPA) project. These routines facilitate forming and analyzing composite scales equivalent to using the raw data but doing so by adding within and between cluster/scale item correlations. These functions include extracting clusters from factor loading matrices (factor2cluster), synthetically forming clusters from correlation matrices (cluster.cor), and finding multiple ((mat.regress) and partial ((partial.r) correlations from correlation matrices.

Functions to generate simulated data with particular structures include sim.circ (for circumplex structures), sim.item (for general structures) and sim.congeneric (for a specific demonstration of congeneric measurement). The functions sim.congeneric and sim.hierarchical can be used to create data sets with particular structural properties. A more general form for all of these is sim.structural for generating general structural models. These are discussed in more detail in the vignette (psych_for_sem).

Functions to apply various standard statistical tests include p.rep and its variants for testing the probability of replication, r.con for the confidence intervals of a correlation, and r.test to test single, paired, or sets of correlations.

In order to study diurnal or circadian variations in mood, it is helpful to use circular statistics. Functions to find the circular mean (circadian.mean), circular (phasic) correlations (circadian.cor) and the correlation between linear variables and circular variables (circadian.linear.cor) supplement a function to find the best fitting phase angle (cosinor) for measures taken with a fixed period (e.g., 24 hours).

The most recent development version of the package is always available for download as a *source* file from the repository at http://personality-project.org/r/src/contrib/.

Details

Two vignettes (overview.pdf) and psych_for_sem.pdf) are useful introductions to the package. They may be found as vignettes in R or may be downloaded from http://personality-project.org/r/book/overview.pdf and http://personality-project.org/r/book/psych_for_sem.pdf.

The psych package was originally a combination of multiple source files maintained at the http://personality-project.org/r repository: “useful.r", VSS.r., ICLUST.r, omega.r, etc.“useful.r" is a set of routines for easy data entry (read.clipboard), simple descriptive statistics (describe), and splom plots combined with correlations (pairs.panels, adapted from the help files of pairs). Those files have now been replaced with a single package.

The vss routines allow for testing the number of factors (vss), showing plots (VSS.plot) of goodness of fit, and basic routines for estimating the number of factors/components to extract by using the MAP's procedure, the examining the scree plot (VSS.scree) or comparing with the scree of an equivalent matrix of random numbers (VSS.parallel).

In addition, there are routines for hierarchical factor analysis using Schmid Leiman tranformations (omega, omega.graph) as well as Item Cluster analysis (ICLUST, ICLUST.graph).

The more important functions in the package are for the analysis of multivariate data, with an emphasis upon those functions useful in scale construction of item composites.

When given a set of items from a personality inventory, one goal is to combine these into higher level item composites. This leads to several questions:

1) What are the basic properties of the data? describe reports basic summary statistics (mean, sd, median, mad, range, minimum, maximum, skew, kurtosis, standard error) for vectors, columns of matrices, or data.frames. describeBy provides descriptive statistics, organized by one or more grouping variables. pairs.panels shows scatter plot matrices (SPLOMs) as well as histograms and the Pearson correlation for scales or items. error.bars will plot variable means with associated confidence intervals. error.bars will plot confidence intervals for both the x and y coordinates. corr.test will find the significance values for a matrix of correlations.

2) What is the most appropriate number of item composites to form? After finding either standard Pearson correlations, or finding tetrachoric or polychoric correlations using a wrapper (poly.mat) for John Fox's hetcor function, the dimensionality of the correlation matrix may be examined. The number of factors/components problem is a standard question of factor analysis, cluster analysis, or principal components analysis. Unfortunately, there is no agreed upon answer. The Very Simple Structure (VSS) set of procedures has been proposed as on answer to the question of the optimal number of factors. Other procedures (VSS.scree, VSS.parallel, fa.parallel, and MAP) also address this question.

3) What are the best composites to form? Although this may be answered using principal components (principal), principal axis (factor.pa) or minimum residual (factor.minres) factor analysis (all part of the fa function) and to show the results graphically (fa.diagram), it is sometimes more useful to address this question using cluster analytic techniques. Previous versions of ICLUST (e.g., Revelle, 1979) have been shown to be particularly successful at forming maximally consistent and independent item composites. Graphical output from ICLUST.graph uses the Graphviz dot language and allows one to write files suitable for Graphviz. If Rgraphviz is available, these graphs can be done in R.

Graphical organizations of cluster and factor analysis output can be done using cluster.plot which plots items by cluster/factor loadings and assigns items to that dimension with the highest loading.

4) How well does a particular item composite reflect a single construct? This is a question of reliability and general factor saturation. Multiple solutions for this problem result in (Cronbach's) alpha (alpha, score.items), (Revelle's) Beta (ICLUST), and (McDonald's) omega (both omega hierarchical and omega total). Additional reliability estimates may be found in the guttman function.

This can also be examined by applying irt.fa Item Response Theory techniques using factor analysis of the tetrachoric or polychoric correlation matrices and converting the results into the standard two parameter parameterization of item difficulty and item discrimination. Information functions for the items suggest where they are most effective.

5) For some applications, data matrices are synthetically combined from sampling different items for different people. So called Synthetic Aperture Personality Assessement (SAPA) techniques allow the formation of large correlation or covariance matrices even though no one person has taken all of the items. To analyze such data sets, it is easy to form item composites based upon the covariance matrix of the items, rather than original data set. These matrices may then be analyzed using a number of functions (e.g., cluster.cor, factor.pa, ICLUST, principal, mat.regress, and factor2cluster.

6) More typically, one has a raw data set to analyze. alpha will report several reliablity estimates as well as item-whole correlations for items forming a single scale, score.items will score data sets on multiple scales, reporting the scale scores, item-scale and scale-scale correlations, as well as coefficient alpha, alpha-1 and G6+. Using a 'keys' matrix (created by make.keys or by hand), scales can have overlapping or independent items. score.multiple.choice scores multiple choice items or converts multiple choice items to dichtomous (0/1) format for other functions.

An additional set of functions generate simulated data to meet certain structural properties. sim.anova produces data simulating a 3 way analysis of variance (ANOVA) or linear model with or with out repeated measures. sim.item creates simple structure data, sim.circ will produce circumplex structured data, sim.dichot produces circumplex or simple structured data for dichotomous items. These item structures are useful for understanding the effects of skew, differential item endorsement on factor and cluster analytic soutions. sim.structural will produce correlation matrices and data matrices to match general structural models. (See the vignette).

When examining personality items, some people like to discuss them as representing items in a two dimensional space with a circumplex structure. Tests of circumplex fit circ.tests have been developed. When representing items in a circumplex, it is convenient to view them in polar coordinates.

Additional functions for testing the difference between two independent or dependent correlation r.test, to find the phi or Yule coefficients from a two by table, or to find the confidence interval of a correlation coefficient.

Ten data sets are included: bfi represents 25 personality items thought to represent five factors of personality, iqitems has 14 multiple choice iq items. sat.act has data on self reported test scores by age and gender. galton Galton's data set of the heights of parents and their children. peas recreates the original Galton data set of the genetics of sweet peas. heights and cubits provide even more Galton data, vegetables provides the Guilford preference matrix of vegetables. cities provides airline miles between 11 US cities (demo data for multidimensional scaling).

Package:	psych
Type:	Package
Version:	1.4.3
Date:	2014–March–25
License:	GPL version 2 or newer

Index:

psych A package for personality, psychometric, and psychological research.

Useful data entry and descriptive statistics

read.clipboard	shortcut for reading from the clipboard
read.clipboard.csv	shortcut for reading comma delimited files from clipboard
read.clipboard.lower	shortcut for reading lower triangular matrices from the clipboard
read.clipboard.upper	shortcut for reading upper triangular matrices from the clipboard
describe	Basic descriptive statistics useful for psychometrics
describe.by	Find summary statistics by groups
statsBy	Find summary statistics by a grouping variable, including within and between correlation matrices.
headtail	combines the head and tail functions for showing data sets
pairs.panels	SPLOM and correlations for a data matrix
corr.test	Correlations, sample sizes, and p values for a data matrix
cor.plot	graphically show the size of correlations in a correlation matrix
multi.hist	Histograms and densities of multiple variables arranged in matrix form
skew	Calculate skew for a vector, each column of a matrix, or data.frame
kurtosi	Calculate kurtosis for a vector, each column of a matrix or dataframe
geometric.mean	Find the geometric mean of a vector or columns of a data.frame
harmonic.mean	Find the harmonic mean of a vector or columns of a data.frame
error.bars	Plot means and error bars
error.bars.by	Plot means and error bars for separate groups
error.crosses	Two way error bars
interp.median	Find the interpolated median, quartiles, or general quantiles.
rescale	Rescale data to specified mean and standard deviation
table2df	Convert a two dimensional table of counts to a matrix or data frame

Data reduction through cluster and factor analysis

fa	Combined function for principal axis, minimum residual, weighted least squares, and maximum likelihood factor analysis
factor.pa	Do a principal Axis factor analysis (deprecated)
factor.minres	Do a minimum residual factor analysis (deprecated)
factor.wls	Do a weighted least squares factor analysis (deprecated)
fa.graph	Show the results of a factor analysis or principal components analysis graphically
fa.diagram	Show the results of a factor analysis without using Rgraphviz
fa.sort	Sort a factor or principal components output
fa.extension	Apply the Dwyer extension for factor loadingss
principal	Do an eigen value decomposition to find the principal components of a matrix
fa.parallel	Scree test and Parallel analysis
fa.parallel.poly	Scree test and Parallel analysis for polychoric matrices
factor.scores	Estimate factor scores given a data matrix and factor loadings
guttman	8 different measures of reliability (6 from Guttman (1945)
`irt.fa`	Apply factor analysis to dichotomous items to get IRT parameters
`iclust`	Apply the ICLUST algorithm

ICLUST.graph	Graph the output from ICLUST using the dot language
ICLUST.rgraph	Graph the output from ICLUST using rgraphviz
kaiser	Apply kaiser normalization before rotating
polychoric	Find the polychoric correlations for items and find item thresholds
poly.mat	Find the polychoric correlations for items (uses J. Fox's hetcor)
omega	Calculate the omega estimate of factor saturation (requires the GPArotation package)
omega.graph	Draw a hierarchical or Schmid Leiman orthogonalized solution (uses Rgraphviz)
partial.r	Partial variables from a correlation matrix
predict	Predict factor/component scores for new data
schmid	Apply the Schmid Leiman transformation to a correlation matrix
score.items	Combine items into multiple scales and find alpha
score.multiple.choice	Combine items into multiple scales and find alpha and basic scale statistics
set.cor	Find Cohen's set correlation between two sets of variables
smc	Find the Squared Multiple Correlation (used for initial communality estimates)
tetrachoric	Find tetrachoric correlations and item thresholds
polyserial	Find polyserial and biserial correlations for item validity studies
mixed.cor	Form a correlation matrix from continuous, polytomous, and dichotomous items
VSS	Apply the Very Simple Structure criterion to determine the appropriate number of factors.
VSS.parallel	Do a parallel analysis to determine the number of factors for a random matrix
VSS.plot	Plot VSS output
VSS.scree	Show the scree plot of the factor/principal components
MAP	Apply the Velicer Minimum Absolute Partial criterion for number of factors

Functions for reliability analysis (some are listed above as well).

alpha	Find coefficient alpha and Guttman Lambda 6 for a scale (see also score.items)
guttman	8 different measures of reliability (6 from Guttman (1945)
omega	Calculate the omega estimates of reliability (requires the GPArotation package)
omegaSem	Calculate the omega estimates of reliability using a Confirmatory model (requires the sem package)
ICC	Intraclass correlation coefficients
score.items	Combine items into multiple scales and find alpha
glb.algebraic	The greates lower bound found by an algebraic solution (requires Rcsdp). Written by Andreas Moeltner

Procedures particularly useful for Synthetic Aperture Personality Assessment

alpha	Find coefficient alpha and Guttman Lambda 6 for a scale (see also score.items)
make.keys	Create the keys file for score.items or cluster.cor
correct.cor	Correct a correlation matrix for unreliability
count.pairwise	Count the number of complete cases when doing pair wise correlations
cluster.cor	find correlations of composite variables from larger matrix
cluster.loadings	find correlations of items with composite variables from a larger matrix
eigen.loadings	Find the loadings when doing an eigen value decomposition
fa	Do a minimal residual or principal axis factor analysis and estimate factor scores
fa.extension	Extend a factor analysis to a set of new variables
factor.pa	Do a Principal Axis factor analysis and estimate factor scores

factor2cluster	extract cluster definitions from factor loadings
factor.congruence	Factor congruence coefficient
factor.fit	How well does a factor model fit a correlation matrix
factor.model	Reproduce a correlation matrix based upon the factor model
factor.residuals	Fit = data - model
factor.rotate	“hand rotate" factors
guttman	8 different measures of reliability
mat.regress	standardized multiple regression from raw or correlation matrix input
polyserial	polyserial and biserial correlations with massive missing data
tetrachoric	Find tetrachoric correlations and item thresholds

Functions for generating simulated data sets

sim	The basic simulation functions
sim.anova	Generate 3 independent variables and 1 or more dependent variables for demonstrating ANOVA and lm designs
sim.circ	Generate a two dimensional circumplex item structure
sim.item	Generate a two dimensional simple structure with particular item characteristics
sim.congeneric	Generate a one factor congeneric reliability structure
sim.minor	Simulate nfact major and nvar/2 minor factors
sim.structural	Generate a multifactorial structural model
sim.irt	Generate data for a 1, 2, 3 or 4 parameter logistic model
sim.VSS	Generate simulated data for the factor model
phi.demo	Create artificial data matrices for teaching purposes
sim.hierarchical	Generate simulated correlation matrices with hierarchical or any structure
sim.spherical	Generate three dimensional spherical data (generalization of circumplex to 3 space)

Graphical functions (require Rgraphviz) – deprecated

structure.graph	Draw a sem or regression graph
fa.graph	Draw the factor structure from a factor or principal components analysis
omega.graph	Draw the factor structure from an omega analysis(either with or without the Schmid Leiman transformation)
ICLUST.graph	Draw the tree diagram from ICLUST

Graphical functions that do not require Rgraphviz

diagram	A general set of diagram functions.
structure.diagram	Draw a sem or regression graph
fa.diagram	Draw the factor structure from a factor or principal components analysis
omega.diagram	Draw the factor structure from an omega analysis(either with or without the Schmid Leiman transformatic
ICLUST.diagram	Draw the tree diagram from ICLUST
plot.psych	A call to plot various types of output (e.g. from irt.fa, fa, omega, iclust

cor.plot	A heat map display of correlations
spider	Spider and radar plots (circular displays of correlations)

Circular statistics (for circadian data analysis)

circadian.cor	Find the correlation with e.g., mood and time of day
circadian.linear.cor	Correlate a circular value with a linear value
circadian.mean	Find the circular mean of each column of a a data set
cosinor	Find the best fitting phase angle for a circular data set

Miscellaneous functions

comorbidity	Convert base rate and comorbity to phi, Yule and tetrachoric
df2latex	Convert a data.frame or matrix to a LaTeX table
dummy.code	Convert categorical data to dummy codes
fisherz	Apply the Fisher r to z transform
fisherz2r	Apply the Fisher z to r transform
ICC	Intraclass correlation coefficients
cortest.mat	Test for equality of two matrices (see also cortest.normal, cortest.jennrich)
cortest.bartlett	Test whether a matrix is an identity matrix
paired.r	Test for the difference of two paired or two independent correlations
r.con	Confidence intervals for correlation coefficients
r.test	Test of significance of r, differences between rs.
p.rep	The probability of replication given a p, r, t, or F
phi	Find the phi coefficient of correlation from a 2 x 2 table
phi.demo	Demonstrate the problem of phi coefficients with varying cut points
phi2poly	Given a phi coefficient, what is the polychoric correlation
phi2poly.matrix	Given a phi coefficient, what is the polychoric correlation (works on matrices)
polar	Convert 2 dimensional factor loadings to polar coordinates.
scaling.fits	Compares alternative scaling solutions and gives goodness of fits
scrub	Basic data cleaning
tetrachor	Finds tetrachoric correlations
thurstone	Thurstone Case V scaling
tr	Find the trace of a square matrix
wkappa	weighted and unweighted versions of Cohen's kappa
Yule	Find the Yule Q coefficient of correlation
Yule.inv	What is the two by two table that produces a Yule Q with set marginals?
Yule2phi	What is the phi coefficient corresponding to a Yule Q with set marginals?
Yule2tetra	Convert one or a matrix of Yule coefficients to tetrachoric coefficients.

Functions that are under development and not recommended for casual use

irt.item.diff.rasch	IRT estimate of item difficulty with assumption that theta = 0
irt.person.rasch	Item Response Theory estimates of theta (ability) using a Rasch like model

Data sets included in the psych package

bfi	represents 25 personality items thought to represent five factors of personality
Thurstone	8 different data sets with a bifactor structure
cities	The airline distances between 11 cities (used to demonstrate MDS)
epi.bfi	13 personality scales
iqitems	14 multiple choice iq items
msq	75 mood items
sat.act	Self reported ACT and SAT Verbal and Quantitative scores by age and gender
Tucker	Correlation matrix from Tucker
galton	Galton's data set of the heights of parents and their children
heights	Galton's data set of the relationship between height and forearm (cubit) length
cubits	Galton's data table of height and forearm length
peas	Galton's data set of the diameters of 700 parent and offspring sweet peas
vegetables	Guilford's preference matrix of vegetables (used for thurstone)

A debugging function that may also be used as a demonstration of psych.

test.psych	Run a test of the major functions on 5 different data sets. Primarily for development purposes. Although the output can be used as a demo of the various functions.

Note

Development versions (source code) of this package are maintained at the repository http://personality-project.org/r along with further documentation. Specify that you are downloading a source package.
Some functions require other packages. Specifically, omega and schmid require the GPArotation package, ICLUST.rgraph and fa.graph require Rgraphviz but have alternatives using the diagram functions. i.e.:

function	requires
omega	GPArotation
schmid	GPArotation
poly.mat	polychor
phi2poly	polychor
polychor.matrix	polychor
ICLUST.rgraph	Rgraphviz
fa.graph	Rgraphviz
structure.graph	Rgraphviz
glb.algebraic	Rcsdp

Author(s)

William Revelle
Department of Psychology
Northwestern University
Evanston, Illinios
http://personality-project.org/revelle.html

Maintainer: William Revelle <revelle@northwestern.edu>

References

A general guide to personality theory and research may be found at the personality-project http://personality-project.org. See also the short guide to R at http://personality-project.org/r. In addition, see

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at http://personality-project.org/r/book/

Examples

```
#See the separate man pages
#to test most of the psych package run the following
#test.psych()
```

`ability` *16 ability items scored as correct or incorrect.*

Description

16 multiple choice ability items 1525 subjects taken from the Synthetic Aperture Personality Assessment (SAPA) web based personality assessment project are saved as `iqitems`. Those data are shown as examples of how to score multiple choice tests and analyses of response alternatives. When scored correct or incorrect, the data are useful for demonstrations of tetrachoric based factor analysis `irt.fa` and finding tetrachoric correlations.

Usage

```
data(iqitems)
```

Format

A data frame with 1525 observations on the following 16 variables. The number following the name is the item number from SAPA.

`reason.4` Basic reasoning questions

`reason.16` Basic reasoning question

`reason.17` Basic reasoning question

`reason.19` Basic reasoning question

`letter.7` In the following alphanumeric series, what letter comes next?

`letter.33` In the following alphanumeric series, what letter comes next?

`letter.34` In the following alphanumeric series, what letter comes next

`letter.58` In the following alphanumeric series, what letter comes next?

`matrix.45` A matrix reasoning task

`matrix.46` A matrix reasoning task

`matrix.47` A matrix reasoning task

`matrix.55` A matrix reasoning task

`rotate.3` Spatial Rotation of type 1.2

`rotate.4` Spatial Rotation of type 1.2

`rotate.6` Spatial Rotation of type 1.1

`rotate.8` Spatial Rotation of type 2.3

Details

16 items were sampled from 80 items given as part of the SAPA (`http://sapa-project.org`) project (Revelle, Wilt and Rosenthal, 2009; Condon and Revelle, 2014) to develop online measures of ability. These 16 items reflect four lower order factors (verbal reasoning, letter series, matrix reasoning, and spatial rotations. These lower level factors all share a higher level factor ('g').

This data set may be used to demonstrate item response functions, `tetrachoric` correlations, or `irt.fa` as well as `omega` estimates of of reliability and hierarchical structure.

In addition, the data set is a good example of doing item analysis to examine the empirical response probabilities of each item alternative as a function of the underlying latent trait. When doing this, it appears that two of the matrix reasoning problems do not have monotonically increasing trace lines for the probability correct. At moderately high ability (theta = 1) there is a decrease in the probability correct from theta = 0 and theta = 2.

Source

The example data set is taken from the Synthetic Aperture Personality Assessment personality and ability test at `http://sapa-project.org`. The data were collected with David Condon from 8/08/12 to 8/31/12.

References

Revelle, William, Wilt, Joshua, and Rosenthal, Allen (2010) Personality and Cognition: The Personality-Cognition Link. In Gruszka, Alexandra and Matthews, Gerald and Szymura, Blazej (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

Condon, David and Revelle, William, (2014) The International Cognitive Ability Resource: Development and initial validation of a public-domain measure. Intelligence, 43, 52-64.

Examples

```
data(ability)
#not run
# ability.irt <- irt.fa(ability)
# ability.scores <- score.irt(ability.irt,ability)
```

---

affect — *Two data sets of affect and arousal scores as a function of personality and movie conditions*

Description

A recurring question in the study of affect is the proper dimensionality and the relationship to various personality dimensions. Here is a data set taken from two studies of mood and arousal using movies to induce affective states.

Usage

```
data(affect)
```

Details

These are data from two studies conducted in the Personality, Motivation and Cognition Laboratory at Northwestern University. Both studies used a similar methodology:

Collection of pretest data using 5 scales from the Eysenck Personality Inventory and items taken from the Motivational State Questionnaire (see `msq`. In addition, state and trait anxiety measures were given. In the "maps" study, the Beck Depression Inventory was given also.

Then subjects were randomly assigned to one of four movie conditions: 1: Frontline. A documentary about the liberation of the Bergen-Belsen concentration camp. 2: Halloween. A horror film. 3: National Geographic, a nature film about the Serengeti plain. 4: Parenthood. A comedy. Each film clip was shown for 9 minutes. Following this the MSQ was given again.

Data from the MSQ were scored for Energetic and Tense Arousal (EA and TA) as well as Positive and Negative Affect (PA and NA).

Study flat had 170 participants, study maps had 160.

These studies are described in more detail in various publications from the PMC lab. In particular, Revelle and Anderson, 1997 and Rafaeli and Revelle (2006). An analysis of these data has also appeared in Smillie et al. (2012).

Source

Data collected at the Personality, Motivation, and Cognition Laboratory, Northwestern University.

References

Revelle, William and Anderson, Kristen Joan (1997) Personality, motivation and cognitive performance: Final report to the Army Research Institute on contract MDA 903-93-K-0008

Rafaeli, Eshkol and Revelle, William (2006), A premature consensus: Are happiness and sadness truly opposite affects? Motivation and Emotion, 30, 1, 1-12.

Smillie, Luke D. and Cooper, Andrew and Wilt, Joshua and Revelle, William (2012) Do Extraverts Get More Bang for the Buck? Refining the Affective-Reactivity Hypothesis of Extraversion. Journal of Personality and Social Psychology, 103 (2), 206-326.

Examples

```
data(affect)
describeBy(affect[-1],group="Film")
pairs.panels(affect[14:17],bg=c("red","black","white","blue")[affect$Film],pch=21,
    main="Affect varies by movies ")
errorCircles("EA2","TA2",data=affect,group="Film",labels=c("Sad","Fear","Neutral","Humor")
, main="Enegetic and Tense Arousal by Movie condition")
errorCircles(x="PA2",y="NA2",data=affect,group="Film",labels=c("Sad","Fear","Neutral","
Humor"),  main="Positive and Negative Affect by Movie condition")
```

`alpha` *Find two estimates of reliability: Cronbach's alpha and Guttman's Lambda 6.*

Description

Internal consistency measures of reliability range from ω_h to α to ω_t. This function reports two estimates: Cronbach's coefficient α and Guttman's λ_6. Also reported are item - whole correlations, α if an item is omitted, and item means and standard deviations.

Usage

```
alpha(x, keys=NULL,cumulative=FALSE, title=NULL, max=10,na.rm = TRUE,
   check.keys=FALSE,n.iter=1,delete=TRUE,use="pairwise",warnings=TRUE,n.obs=NULL)
```

Arguments

`x`	A data.frame or matrix of data, or a covariance or correlation matrix
`keys`	If some items are to be reversed keyed, then either specify the direction of all items or just a vector of which items to reverse
`title`	Any text string to identify this run
`cumulative`	should means reflect the sum of items or the mean of the items. The default value is means.
`max`	the number of categories/item to consider if reporting category frequencies. Defaults to 10, passed to `link{response.frequencies}`

`na.rm`	The default is to remove missing values and find pairwise correlations
`check.keys`	if TRUE, then find the first principal component and reverse key items with negative loadings. Give a warning if this happens.
`n.iter`	Number of iterations if bootstrapped confidence intervals are desired
`delete`	Delete items with no variance and issue a warning
`use`	Options to pass to the cor function: "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". The default is "pairwise"
`warnings`	By default print a warning and a message that items were reversed. Suppress the message if warnings = FALSE
`n.obs`	If using correlation matrices as input, by specify the number of observations, we can find confidence intervals

Details

Alpha is one of several estimates of the internal consistency reliability of a test.

Surprisingly, more than a century after Spearman (1904) introduced the concept of reliability to psychologists, there are still multiple approaches for measuring it. Although very popular, Cronbach's α (1951) underestimates the reliability of a test and over estimates the first factor saturation.

α (Cronbach, 1951) is the same as Guttman's $\lambda 3$ (Guttman, 1945) and may be found by

$$\lambda_3 = \frac{n}{n-1}\Bigl(1 - \frac{tr(\vec{V})_x}{V_x}\Bigr) = \frac{n}{n-1}\frac{V_x - tr(\vec{V}_x)}{V_x} = \alpha$$

Perhaps because it is so easy to calculate and is available in most commercial programs, alpha is without doubt the most frequently reported measure of internal consistency reliability. Alpha is the mean of all possible spit half reliabilities (corrected for test length). For a unifactorial test, it is a reasonable estimate of the first factor saturation, although if the test has any microstructure (i.e., if it is "lumpy") coefficients β (Revelle, 1979; see `ICLUST`) and ω_h (see `omega`) are more appropriate estimates of the general factor saturation. ω_t (see `omega`) is a better estimate of the reliability of the total test.

Guttman's Lambda 6 (G6) considers the amount of variance in each item that can be accounted for the linear regression of all of the other items (the squared multiple correlation or smc), or more precisely, the variance of the errors, e_j^2, and is

$$\lambda_6 = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum(1 - r_{smc}^2)}{V_x}.$$

The squared multiple correlation is a lower bound for the item communality and as the number of items increases, becomes a better estimate.

G6 is also sensitive to lumpyness in the test and should not be taken as a measure of unifactorial structure. For lumpy tests, it will be greater than alpha. For tests with equal item loadings, alpha > G6, but if the loadings are unequal or if there is a general factor, G6 > alpha. alpha is a generalization of an earlier estimate of reliability for tests with dichotomous items developed by Kuder and Richardson, known as KR20, and a shortcut approximation, KR21. (See Revelle, in prep).

Alpha and G6 are both positive functions of the number of items in a test as well as the average intercorrelation of the items in the test. When calculated from the item variances and total test variance, as is done here, raw alpha is sensitive to differences in the item variances. Standardized alpha is based upon the correlations rather than the covariances.

A useful index of the quality of the test that is linear with the number of items and the average correlation is the Signal/Noise ratio where

$$s/n = \frac{n\bar{r}}{1 - n\bar{r}}$$

(Cronbach and Gleser, 1964; Revelle and Condon (in press)).

More complete reliability analyses of a single scale can be done using the `omega` function which finds ω_h and ω_t based upon a hierarchical factor analysis.

Alternative functions `score.items` and `cluster.cor` will also score multiple scales and report more useful statistics. "Standardized" alpha is calculated from the inter-item correlations and will differ from raw alpha.

Four alternative item-whole correlations are reported, three are conventional, one unique. raw.r is the correlation of the item with the entire scale, not correcting for item overlap. std.r is the correlation of the item with the entire scale, if each item were standardized. r.drop is the correlation of the item with the scale composed of the remaining items. Although each of these are conventional statistics, they have the disadvantage that a) item overlap inflates the first and b) the scale is different for each item when an item is dropped. Thus, the fourth alternative, r.cor, corrects for the item overlap by subtracting the item variance but then replaces this with the best estimate of common variance, the smc. This is similar to a suggestion by Cureton (1966).

If some items are to be reversed keyed then they can be specified by either item name or by item location. (Look at the 3rd and 4th examples.) Automatic reversal can also be done, and this is based upon the sign of the loadings on the first principal component (Example 5). This requires the check.keys option to be TRUE. Previous versions defaulted to have check.keys=TRUE, but some users complained that this made it too easy to find alpha without realizing that some items had been reversed (even though a warning was issued!). Thus, I have set the default to be check.keys=FALSE with a warning that some items need to be reversed (if this is the case). To suppress these warnings, set warnings=FALSE.

Scores are based upon the simple averages (or totals) of the items scored. Reversed items are subtracted from the maximum + minimum item response for all the items.

When using raw data, standard errors for the raw alpha are calculated using equation 2 and 3 from Duhhachek and Iacobucci (2004). This is problematic because some simulations suggest these values are too small. It is probably better to use bootstrapped value

Bootstrapped resamples are found if n.iter > 1. These are returned as the boot object. They may be plotted or described.

Value

`total`	a list containing
`raw_alpha`	alpha based upon the covariances
`std.alpha`	The standarized alpha based upon the correlations
`G6(smc)`	Guttman's Lambda 6 reliability
`average_r`	The average interitem correlation
`mean`	For data matrices, the mean of the scale formed by summing the items

`sd`	For data matrices, the standard deviation of the total score
`alpha.drop`	A data frame with all of the above for the case of each item being removed one by one.
`item.stats`	A data frame including
`n`	number of complete cases for the item
`raw.r`	The correlation of each item with the total score, not corrected for item overlap.
`std.r`	The correlation of each item with the total score (not corrected for item overlap) if the items were all standardized
`r.cor`	Item whole correlation corrected for item overlap and scale reliability
`r.drop`	Item whole correlation for this item against the scale without this item
`mean`	for data matrices, the mean of each item
`sd`	For data matrices, the standard deviation of each item
`response.freq`	For data matrices, the frequency of each item response (if less than 20)
`boot`	a 6 column by n.iter matrix of boot strapped resampled values
`Unidim`	An index of unidimensionality
`Fit`	The fit of the off diagonal matrix

Note

By default, items that correlate negatively with the overall scale will be reverse coded. This option may be turned off by setting check.keys = FALSE. If items are reversed, then each item is subtracted from the minimum item response + maximum item response where min and max are taken over all items. Thus, if the items intentionally differ in range, the scores will be off by a constant. See `scoreItems` for a solution.

Two experimental measures of Goodness of Fit are returned in the output: Unidim and Fit. They are not printed or displayed, but are available for analysis. The first is an index of how well the modeled average correlations actually reproduce the original correlation matrix. The second is how well the modeled correlations reproduce the off diagonal elements of the matrix. Both are indices of squared residuals compared to the squared original correlations. These two measures are under development and might well be modified or dropped in subsequent versions.

Author(s)

William Revelle

References

Cronbach, L.J. (1951) Coefficient alpha and the internal strucuture of tests. Psychometrika, 16, 297-334.

Cureton, E. (1966). Corrected item-test correlations. Psychometrika, 31(1):93-96.

Cronbach, L.J. and Gleser G.C. (1964)The signal/noise ratio in the comparison of reliability coefficients. Educational and Psychological Measurement, 24 (3) 467-480.

Duhachek, A. and Iacobucci, D. (2004). Alpha's standard error (ase): An accurate and precise confidence interval estimate. Journal of Applied Psychology, 89(5):792-808.

Guttman, L. (1945). A basis for analyzing test-retest reliability. Psychometrika, 10 (4), 255-282.

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. Springer. (Available online at http://personality-project.org/r/book).

Revelle, W. Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 1979, 14, 57-74.

Revelle, W. and Condon, D.C. Reliability. In Irwing, P., Booth, T. and Hughes, D. (Eds). the Wiley-Blackwell Handbook of Psychometric Testing (in press).

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74 (1) 1145-154.

See Also

omega, ICLUST, guttman, scoreItems, cluster.cor

Examples

```
set.seed(42) #keep the same starting values
#four congeneric measures
r4 <- sim.congeneric()
alpha(r4)
#nine hierarchical measures -- should actually use omega
r9 <- sim.hierarchical()
alpha(r9)

# examples of two independent factors that produce reasonable alphas
#this is a case where alpha is a poor indicator of unidimensionality
two.f <- sim.item(8)
#specify which items to reverse key by name
 alpha(two.f,keys=c("V1","V2","V7","V8"))
 #by location
 alpha(two.f,keys=c(1,2,7,8))
 #automatic reversal base upon first component
alpha(two.f)
#an example with discrete item responses  -- show the frequencies
items <- sim.congeneric(N=500,short=FALSE,low=-2,high=2,
        categorical=TRUE) #500 responses to 4 discrete items with 5 categories
a4 <- alpha(items$observed)  #item response analysis of congeneric measures
a4
#summary just gives Alpha
summary(a4)
```

Bechtoldt *Seven data sets showing a bifactor solution.*

Description

Holzinger-Swineford (1937) introduced the bifactor model of a general factor and uncorrelated group factors. The Holzinger data sets are original 14 * 14 matrix from their paper as well as a 9 *9 matrix used as an example by Joreskog. The Thurstone correlation matrix is a 9 * 9 matrix of correlations of ability items. The Reise data set is 16 * 16 correlation matrix of mental health items. The Bechtholdt data sets are both 17 x 17 correlation matrices of ability tests.

Usage

```
data(Thurstone)
data(Thurstone.33)
data(Holzinger)
data(Holzinger.9)
data(Bechtoldt)
data(Bechtoldt.1)
data(Bechtoldt.2)
data(Reise)
```

Details

Holzinger and Swineford (1937) introduced the bifactor model (one general factor and several group factors) for mental abilities. This is a nice demonstration data set of a hierarchical factor structure that can be analyzed using the `omega` function or using sem. The bifactor model is typically used in measures of cognitive ability.

There are several ways to analyze such data. One is to use the `omega` function to do a hierarchical factoring using the schmid-leiman transformation. Another is to a regular factor analysis and use either a `bifactor` or `biquartimin` rotation. These latter two functions implement the Jennrich and Bentler (2011) bifactor and biquartimin transformations.

The 14 variables are ordered to reflect 3 spatial tests, 3 mental speed tests, 4 motor speed tests, and 4 verbal tests. The sample size is 355.

Another data set from Holzinger (Holzinger.9) represents 9 cognitive abilities (Holzinger, 1939) and is used as an example by Karl Joreskog (2003) for factor analysis by the MINRES algorithm and also appears in the LISREL manual as example NPV.KM.

Another classic data set is the 9 variable Thurstone problem which is discussed in detail by R. P. McDonald (1985, 1999) and and is used as example in the sem package as well as in the PROC CALIS manual for SAS. These nine tests were grouped by Thurstone and Thurstone, 1941 (based on other data) into three factors: Verbal Comprehension, Word Fluency, and Reasoning. The original data came from Thurstone and Thurstone (1941) but were reanalyzed by Bechthold (1961) who broke the data set into two. McDonald, in turn, selected these nine variables from the larger set of 17 found in Bechtoldt.2. The sample size is 213.

Another set of 9 cognitive variables attributed to Thurstone (1933) is the data set of 4,175 students reported by Professor Brigham of Princeton to the College Entrance Examination Board. This set does not show a clear bifactor solution but is included as a demonstration of the differences between a maximimum likelihood factor analysis solution versus a principal axis factor solution.

More recent applications of the bifactor model are to the measurement of psychological status. The Reise data set is a correlation matrix based upon >35,000 observations to the Consumer Assessment of Health Care Provideers and Systems survey instrument. Reise, Morizot, and Hays (2007) describe a bifactor solution based upon 1,000 cases.

The five factors from Reise et al. reflect Getting care quickly (1-3), Doctor communicates well (4-7), Courteous and helpful staff (8,9), Getting needed care (10-13), and Health plan customer service (14-16).

The two Bechtoldt data sets are two samples from Thurstone and Thurstone (1941). They include 17 variables, 9 of which were used by McDonald to form the Thurstone data set. The sample sizes are 212 and 213 respectively. The six proposed factors reflect memory, verbal, words, space, number and reasoning with three markers for all expect the rote memory factor. 9 variables from this set appear in the Thurstone data set.

Two more data sets with similar structures are found in the `Harman` data set.

- Bechtoldt.1: 17 x 17 correlation matrix of ability tests, N = 212.
- Bechtoldt.2: 17 x 17 correlation matrix of ability tests, N = 213.
- Holzinger: 14 x 14 correlation matrix of ability tests, N = 355
- Holzinger.9: 9 x 9 correlation matrix of ability tests, N = 145
- Reise: 16 x 16 correlation matrix of health satisfaction items. N = 35,000
- Thurstone: 9 x 9 correlation matrix of ability tests, N = 213
- Thurstone.33: Another 9 x 9 correlation matrix of ability items, N=4175

Source

Holzinger: Holzinger and Swineford (1937)
Reise: Steve Reise (personal communication)
sem help page (for Thurstone)

References

Bechtoldt, Harold, (1961). An empirical study of the factor analysis stability hypothesis. Psychometrika, 26, 405-432.

Holzinger, Karl and Swineford, Frances (1937) The Bi-factor method. Psychometrika, 2, 41-54

Holzinger, K., & Swineford, F. (1939). A study in factor analysis: The stability of a bifactor solution. Supplementary Educational Monograph, no. 48. Chicago: University of Chicago Press.

McDonald, Roderick P. (1999) Test theory: A unified treatment. L. Erlbaum Associates. Mahwah, N.J.

Reise, Steven and Morizot, Julien and Hays, Ron (2007) The role of the bifactor model in resolving dimensionality issues in health outcomes measures. Quality of Life Research. 16, 19-31.

Thurstone, Louis Leon (1933) The theory of multiple factors. Edwards Brothers, Inc. Ann Arbor

Thurstone, Louis Leon and Thurstone, Thelma (Gwinn). (1941) Factorial studies of intelligence. The University of Chicago Press. Chicago, Il.

Examples

```
if(!require(GPArotation)) {message("I am sorry, to run omega requires GPArotation")
        } else {
#holz <- omega(Holzinger,4, title = "14 ability tests from Holzinger-Swineford")
#bf <- omega(Reise,5,title="16 health items from Reise")
#omega(Reise,5,labels=colnames(Reise),title="16 health items from Reise")
thur.om <- omega(Thurstone,title="9 variables from Thurstone") #compare with
thur.bf   <- fa(Thurstone,3,rotate="biquartimin")
factor.congruence(thur.om,thur.bf)
}
```

---

bestScales *A set of functions for factorial and empirical scale construction*

Description

When constructing scales through rational, factorial, or empirical means, it is useful to examine the content of the items that relate most highly to each other (e.g., the factor loadings of `fa.lookup` of a set of items) , or to some specific set of criteria (e.g., `bestScales`). Given a dictionary of item content, these routines will sort by factor loading or criteria correlations and display the item content.

Usage

```
bestScales(x, criteria, cut = 0.1, n.item = 10, overlap = FALSE,
           dictionary = NULL, digits = 2)
bestItems(x,criteria=1,cut=.3, abs=TRUE, dictionary=NULL,cor=TRUE,digits=2)
lookup(x,y,criteria=NULL)
fa.lookup(f,dictionary,digits=2)
item.lookup(f,m, dictionary,cut=.3, digits = 2)
```

Arguments

`x`	A data matrix or data frame depending upon the function.
`y`	A data matrix or data frame or a vector
`criteria`	Which variables (by name or location) should be the empirical target for bestScales and bestItems
`f`	The object returned from either a factor analysis (fa) or a principal components analysis (principal)
`cut`	Return all values in abs(x[,c1]) > cut.
`abs`	if TRUE, sort by absolute value in bestItems
`dictionary`	a data.frame with rownames corresponding to rownames in the f$loadings matrix or colnames of the data matrix or correlation matrix, and entries (may be multiple columns) of item content.

m	A data frame of item means
cor	if x is not a square matrix, should correlations be found?
n.item	How many items make up an empirical scale
overlap	Are the correlations with other criteria fair game for bestScales
digits	round to digits

Details

bestItems and lookup are simple helper functions to summarize correlation matrices or factor loading matrices. bestItems will sort the specified column (criteria) of x on the basis of the (absolute) value of the column. The return as a default is just the rowname of the variable with those absolute values > cut. If there is a dictionary of item content and item names, then include the contents as a two column matrix with rownames corresponding to the item name and then as many fields as desired for item content. (See the example dictionary bfi.dictionary).

lookup is used by bestItems and will find values in c1 of y that match those in x. It returns those rows of y of that match x. Suppose that you have a "dictionary" of the many variables in a study but you want to consider a small subset of them in a data set x. Then, you can find the entries in the dictionary corresponding to x by lookup(rownames(x),y) If the column is not specified, then it will match by rownames(y).

fa.lookup is used when examining the output of a factor analysis and one wants the corresponding variable names and contents. The returned object may then be printed in LaTex by using the df2latex function with the char option set to TRUE.

Similarly, given a correlation matrix, r, of the x variables, if you want to find the items that most correlate with another item or scale, and then show the contents of that item from the dictionary, bestItems(r,c1=column number or name of x, contents = y)

bestScales will find up to n.items that have absolute correlations with a criterion greater than cut. If the overlap option is FALSE (default) the other criteria are not used.

item.lookup combines the output from a factor analysis fa with simple descriptive statistics (a data frame of means) with a dictionary. Items are grouped by factor loadings > cut, and then sorted by item mean. This allows a better understanding of how a scale works, in terms of the meaning of the item endorsements.

Value

bestScales returns the correlation of the empirically constructed scale with each criteria and the items used in the scale. If a dictionary is specified, it also returns a list (value) that shows the item content. Also returns the keys list so that scales can be found using cluster.cor or scoreItems.

bestItems returns a sorted list of factor loadings or correlations with the labels as provided in the dictionary.

lookup is a very simple implementation of the match function.

fa.lookup takes a factor/cluster analysis object (or just a keys like matrix), sorts it using fa.sort and then matches by row.name to the corresponding dictionary entries.

Note

To create a dictionary, create an object with row names as the item numbers, and the columns as the item content. See the `link{bfi.dictionary}` as an example.

Note

Although empirical scale construction is appealing, it has the basic problem of capitalizing on chance. Thus, be careful of over interpreting the results unless working with large samples.

Author(s)

William Revelle

References

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. Springer. (Available online at http://personality-project.org/r/book).

See Also

fa, iclust,principal

Examples

```
bs <- bestScales(bfi,criteria=c("gender","education","age"),dictionary=bfi.dictionary)
bs
f5 <- fa(bfi,5)
m <- colMeans(bfi,na.rm=TRUE)
item.lookup(f5,m,dictionary=bfi.dictionary[2])
fa.lookup(f5,dictionary=bfi.dictionary[2])  #just show the item content, not the source of the items
```

---

bfi *25 Personality items representing 5 factors*

Description

25 personality self report items taken from the International Personality Item Pool (ipip.ori.org) were included as part of the Synthetic Aperture Personality Assessment (SAPA) web based personality assessment project. The data from 2800 subjects are included here as a demonstration set for scale construction, factor analysis, and Item Response Theory analysis. Three additional demographic variables (sex, education, and age) are also included.

Usage

```
data(bfi)
data(bfi.dictionary)
```

Format

A data frame with 2800 observations on the following 28 variables. (The q numbers are the SAPA item numbers).

`A1` Am indifferent to the feelings of others. (q_146)

`A2` Inquire about others' well-being. (q_1162)

`A3` Know how to comfort others. (q_1206)

`A4` Love children. (q_1364)

`A5` Make people feel at ease. (q_1419)

`C1` Am exacting in my work. (q_124)

`C2` Continue until everything is perfect. (q_530)

`C3` Do things according to a plan. (q_619)

`C4` Do things in a half-way manner. (q_626)

`C5` Waste my time. (q_1949)

`E1` Don't talk a lot. (q_712)

`E2` Find it difficult to approach others. (q_901)

`E3` Know how to captivate people. (q_1205)

`E4` Make friends easily. (q_1410)

`E5` Take charge. (q_1768)

`N1` Get angry easily. (q_952)

`N2` Get irritated easily. (q_974)

`N3` Have frequent mood swings. (q_1099

`N4` Often feel blue. (q_1479)

`N5` Panic easily. (q_1505)

`O1` Am full of ideas. (q_128)

`O2` Avoid difficult reading material.(q_316)

`O3` Carry the conversation to a higher level. (q_492)

`O4` Spend time reflecting on things. (q_1738)

`O5` Will not probe deeply into a subject. (q_1964)

`gender` Males = 1, Females =2

`education` 1 = HS, 2 = finished HS, 3 = some college, 4 = college graduate 5 = graduate degree

`age` age in years

Details

The first 25 items are organized by five putative factors: Agreeableness, Conscientiousness, Extraversion, Neuroticism, and Opennness. The scoring key is created using `make.keys`, the scores are found using `score.items`.

These five factors are a useful example of using `irt.fa` to do Item Response Theory based latent factor analysis of the `polychoric` correlation matrix. The endorsement plots for each item, as well as the item information functions reveal that the items differ in their quality.

The item data were collected using a 6 point response scale: 1 Very Inaccurate 2 Moderately Inaccurate 3 Slightly Inaccurate 4 Slightly Accurate 5 Moderately Accurate 6 Very Accurate

as part of the Synthetic Apeture Personality Assessment (SAPA http://sapa-project.org) project. To see an example of the data collection technique, visit http://SAPA-project.org. The items given were sampled from the International Personality Item Pool of Lewis Goldberg using the sampling technique of SAPA. This is a sample data set taken from the much larger SAPA data bank.

Source

The items are from the ipip (Goldberg, 1999). The data are from the SAPA project (Revelle, Wilt and Rosenthal, 2010) , collected Spring, 2010 (http://sapa-project.org).

References

Goldberg, L.R. (1999) A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. In Mervielde, I. and Deary, I. and De Fruyt, F. and Ostendorf, F. (eds) Personality psychology in Europe. 7. Tilburg University Press. Tilburg, The Netherlands.

Revelle, W., Wilt, J., and Rosenthal, A. (2010) Personality and Cognition: The Personality-Cognition Link. In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

See Also

bi.bars to show the data by age and gender, irt.fa for item factor analysis applying the irt model.

Examples

```
data(bfi)
describe(bfi)

 keys.list <-
  list(agree=c("-A1","A2","A3","A4","A5"),conscientious=c("C1","C2","C3","-C4","-C5"),
extraversion=c("-E1","-E2","E3","E4","E5"),neuroticism=c("N1","N2","N3","N4","N5"),
openness = c("O1","-O2","O3","O4","-O5"))
  keys <- make.keys(bfi,keys.list)

 scores <- scoreItems(keys[1:27,],bfi[1:27]) #don't score age
 scores
 #show the use of the fa.lookup with a dictionary
 fa.lookup(keys,bfi.dictionary[,1:4])
```

bi.bars — *Draw pairs of bargraphs based on two groups*

Description

When showing e.g., age or education distributions for two groups, it is convenient to plot them back to back. bi.bars will do so.

Usage

```
bi.bars(x,grp,horiz,color,label=NULL,...)
```

Arguments

x	The data to be drawn
grp	a grouping variable.
horiz	horizontal (default) or vertical bars
color	colors for the two groups – defaults to blue and red
label	If specified, labels for the dependent axis
...	Further parameters to pass to the graphing program

Details

A trivial, if useful, function to draw back to back histograms/barplots. One for each group.

Value

a graphic

Author(s)

William Revelle

Examples

```
data(bfi)
with(bfi,{bi.bars(age,gender,ylab="Age",main="Age by males and females")
      bi.bars(education,gender,xlab="Education",main="Education by gender",horiz=FALSE)})
```

---

biplot.psych *Draw biplots of factor or component scores by factor or component loadings*

Description

Extends the biplot function to the output of `fa`, `fa.poly` or `principal`. Will plot factor scores and factor loadings in the same graph. If the number of factors > 2, then all pairs of factors are plotted. Factor score histograms are plotted on the diagonal. The input is the resulting object from `fa`, `principal`, or }code{linkfa.poly with the scores=TRUE option. Points may be colored according to other criteria.

Usage

```
## S3 method for class 'psych'
biplot(x, labels=NULL,cex=c(.75,1),main="Biplot from fa",
hist.col="cyan",xlim.s=c(-3,3),ylim.s=c(-3,3),xlim.f=c(-1,1),ylim.f=c(-1,1),
maxpoints=100,adjust=1.2,col,pos, arrow.len = 0.1,pch=16,choose=NULL,
cuts=1,cutl=.0,group=NULL,...)
```

Arguments

`x`	The output from `fa`, `fa.poly` or `principal` with the scores=TRUE option
`labels`	if NULL, draw the points with the plot character (pch) specified. To identify the data points, specify labels= 1:n where n is the number of observations, or labels =rownames(data) where data was the data set analyzed by the factor analysis.
`cex`	A vector of plot sizes of the data labels and of the factor labels
`main`	A main title for a two factor biplot
`hist.col`	If plotting more than two factors, the color of the histogram of the factor scores
`xlim.s`	x limits of the scores. Defaults to plus/minus three sigma
`ylim.s`	y limits of the scores.Defaults to plus/minus three sigma
`xlim.f`	x limits of the factor loadings.Defaults to plus/minus 1.0
`ylim.f`	y limits of the factor loadings.Defaults to plus/minus 1.0
`maxpoints`	When plotting 3 (or more) dimensions, at what size should we switch from plotting "o" to plotting "."
`adjust`	an adjustment factor in the histogram
`col`	a vector of colors for the data points and for the factor loading labels
`pos`	If plotting labels, what position should they be in? 1=below, 2=left, 3 top, 4 right. If missing, then the assumption is that labels should be printed instead of data points.
`arrow.len`	the length of the arrow head
`pch`	The plotting character to use. pch=16 gives reasonable size dots. pch="." gives tiny points. If adding colors, use pch between 21 and 25. (see examples).

choose	Plot just the specified factors
cuts	Do not label cases with abs(factor scores) < cuts) (Actually, the distance of the x and y scores from 0)
cutl	Do not label variables with communalities in the two space < cutl
group	A vector of a grouping variable for the scores. Show a different color and symbol for each group.
...	more options for graphics

Details

Uses the generic biplot function to take the output of a factor analysis `fa`, `fa.poly` or principal components analysis `principal` and plot the factor/component scores along with the factor/component loadings.

This is an extension of the generic biplot function to allow more control over plotting points in a two space and also to plot three or more factors (two at time).

This will work for objects produced by `fa`, `fa.poly` or `principal` if they applied to the original data matrix. If however, one has a correlation matrix based upon the output from `tetrachoric` or `polychoric`, and has done either `fa` or `principal` on the correlations, then obviously, we can not do a biplot. However, both of those functions produce a weights matrix, which, in combination with the original data can be used to find the scores by using `factor.scores`. Since biplot.psych is looking for two elements of the x object: x$loadings and x$scores, you can create the appropriate object to plot. See the third example.

Author(s)

William Revelle

See Also

`fa`, `fa.poly`, `principal`, `fa.plot`, `pairs.panels`

Examples

```
#the standard example
data(USArrests)
fa2 <- fa(USArrests,2,scores=TRUE)
biplot(fa2,labels=rownames(USArrests))

# plot the 3 factor solution
data(bfi)
fa3 <- fa(bfi[1:200,1:15],3,scores=TRUE)
biplot(fa3)
#just plot factors 1 and 3 from that solution
biplot(fa3,choose=c(1,3))

#
fa2 <- fa(bfi[16:25],2)  #factor analysis
fa2$scores <- fa2$scores[1:100,]  #just take the first 100
#now plot with different colors and shapes for males and females
```

```
biplot(fa2,pch=c(24,21)[bfi[1:100,"gender"]],group =bfi[1:100,"gender"],
   main="Biplot of Conscientiousness and Neuroticism by gender")


r <- cor(bfi[1:200,1:10], use="pairwise") #find the correlations
f2 <- fa(r,2)
x <- list()
x$scores <- factor.scores(bfi[1:200,1:10],f2)
x$loadings <- f2$loadings
class(x) <- c('psych','fa')
biplot(x,main="biplot from correlation matrix and factor scores")
```

---

block.random *Create a block randomized structure for n independent variables*

Description

Random assignment of n subjects with an equal number in all of N conditions may done by block randomization, where the block size is the number of experimental conditions. The number of Independent Variables and the number of levels in each IV are specified as input. The output is a the block randomized design.

Usage

```
block.random(n, ncond = NULL)
```

Arguments

n	The number of subjects to randomize. Must be a multiple of the number of experimental conditions
ncond	The number of conditions for each IV. Defaults to 2 levels for one IV. If more than one IV, specify as a vector. If names are provided, they are used, otherwise the IVs are labeled as IV1 ... IVn

Value

blocks	A matrix of subject numbers, block number, and randomized levels for each IV

Note

Prepared for a course on Research Methods in Psychology http://personality-project.org/revelle/syllabi/205/205.syllabus.html

Author(s)

William Revelle

Examples

```
br <- block.random(n=24,c(2,3))
pairs.panels(br)
br <- block.random(96,c(time=4,drug=3,sex=2))
pairs.panels(br)
```

---

blot *Bond's Logical Operations Test – BLOT*

Description

35 items for 150 subjects from Bond's Logical Operations Test. A good example of Item Response Theory analysis using the Rasch model. One parameter (Rasch) analysis and two parameter IRT analyses produce somewhat different results.

Usage

```
data(blot)
```

Format

A data frame with 150 observations on 35 variables. The BLOT was developed as a paper and pencil test for children to measure Logical Thinking as discussed by Piaget and Inhelder.

Details

Bond and Fox apply Rasch modeling to a variety of data sets. This one, Bond's Logical Operations Test, is used as an example of Rasch modeling for dichotomous items. In their text (p 56), Bond and Fox report the results using WINSTEPS. Those results are consistent (up to a scaling parameter) with those found by the rasch function in the ltm package. The WINSTEPS seem to produce difficulty estimates with a mean item difficulty of 0, whereas rasch from ltm has a mean difficulty of -1.52. In addition, rasch seems to reverse the signs of the difficulty estimates when reporting the coefficients and is effectively reporting "easiness".

However, when using a two parameter model, one of the items (V12) behaves very differently.

This data set is useful when comparing 1PL, 2PL and 2PN IRT models.

Source

The data are taken (with kind permission from Trevor Bond) from the webpage http://homes.jcu.edu.au/~edtgb/book/data/Bon
and read using read.fwf.

References

T.G. Bond. BLOT:Bond's Logical Operations Test. Townsville, Australia: James Cook University. (Original work published 1976), 1995.

T. Bond and C. Fox. (2007) Applying the Rasch model: Fundamental measurement in the human sciences. Lawrence Erlbaum, Mahwah, NJ, US, 2 edition.

See Also

See also the irt.fa and associated plot functions.

Examples

```
data(blot)
#not run
#library(ltm)
#bblot.rasch <- rasch(blot, constraint = cbind(ncol(blot) + 1, 1))  #a 1PL model
#blot.2pl <- ltm(blot~z1)  #a 2PL model
#do the same thing with functions in psych
#blot.fa <- irt.fa(blot)  # a 2PN model
#plot(blot.fa)
```

---

bock *Bock and Liberman (1970) data set of 1000 observations of the LSAT*

Description

An example data set used by McDonald (1999) as well as other discussions of Item Response Theory makes use of a data table on 10 items (two sets of 5) from the Law School Admissions Test (LSAT). Included in this data set is the original table as well as the reponses for 1000 subjects on the first set (Figure Classification) and second set (Debate).

Usage

```
data(bock)
```

Format

A data frame with 32 observations on the following 8 variables.

- `index` 32 response patterns
- `Q1` Responses to item 1
- `Q2` Responses to item 2
- `Q3` Responses to item 3
- `Q4` Responses to item 4
- `Q5` Responses to item 5
- `Ob6` count of observations for the section 6 test
- `Ob7` count of observations for the section 7 test

Two other data sets are derived from the bock dataset. These are converted using the table2df function.

- **lsat6** reponses to 5 items for 1000 subjects on section 6
- **lsat7** reponses to 5 items for 1000 subjects on section 7

Details

The lsat6 data set is analyzed in the ltm package as well as by McDonald (1999). lsat7 is another 1000 subjects on part 7 of the LSAT. Both sets are described by Bock and Lieberman (1970). Both sets are useful examples of testing out IRT procedures and showing the use of `tetrachoric` correlations and item factor analysis using the `irt.fa` function.

Source

R. Darrell Bock and M. Lieberman (1970). Fitting a response model for dichotomously scored items. Psychometrika, 35(2):179-197.

References

R.P. McDonald. Test theory: A unified treatment. L. Erlbaum Associates, Mahwah, N.J., 1999.

Examples

```
data(bock)
responses <- table2df(bock.table[,2:6],count=bock.table[,7],
        labs= paste("lsat6.",1:5,sep=""))
describe(responses)
## maybe str(bock.table) ; plot(bock.table) ...
```

`burt` *11 emotional variables from Burt (1915)*

Description

Cyril Burt reported an early factor analysis with a circumplex structure of 11 emotional variables in 1915. 8 of these were subsequently used by Harman in his text on factor analysis. Unfortunately, it seems as if Burt made a mistake for the matrix is not positive definite. With one change from .87 to .81 the matrix is positive definite.

Usage

```
data(burt)
```

Format

A correlation matrix based upon 172 "normal school age children aged 9-12".

Sociality Sociality

Sorrow Sorrow

Tenderness Tenderness

Joy Joy

Wonder Wonder

Elation Elation

Disgust Disgust

Anger Anger

Sex Sex

Fear Fear

Subjection Subjection

Details

The Burt data set is interesting for several reasons. It seems to be an early example of the organizaton of emotions into an affective circumplex, a subset of it has been used for factor analysis examples (see `Harman.Burt`, and it is an example of how typos affect data. The original data matrix has one negative eigenvalue. With the replacement of the correlation between Sorrow and Tenderness from .87 to .81, the matrix is positive definite.

Alternatively, using `cor.smooth`, the matrix can be made positive definite as well, although cor.smooth makes more (but smaller) changes.

Source

(retrieved from the web at http://www.biodiversitylibrary.org/item/95822#790) Following a suggestion by Jan DeLeeuw.

References

Burt, C.General and Specific Factors underlying the Primary Emotions. Reports of the British Association for the Advancement of Science, 85th meeting, held in Manchester, September 7-11, 1915. London, John Murray, 1916, p. 694-696 (retrieved from the web at http://www.biodiversitylibrary.org/item/95822#790)

See Also

`Harman.Burt` in the `Harman` dataset and `cor.smooth`

Examples

```
data(burt)
eigen(burt)$values  #one is negative!
burt.new <- burt
burt.new[2,3] <- burt.new[3,2] <- .81
eigen(burt.new)$values  #all are positive
bs <- cor.smooth(burt)
round(burt.new - bs,3)
```

`circ.tests` *Apply four tests of circumplex versus simple structure*

Description

Rotations of factor analysis and principal components analysis solutions typically try to represent correlation matrices as simple structured. An alternative structure, appealing to some, is a circumplex structure where the variables are uniformly spaced on the perimeter of a circle in a two dimensional space. Generating these data is straightforward, and is useful for exploring alternative solutions to affect and personality structure.

Usage

```
circ.tests(loads, loading = TRUE, sorting = TRUE)
```

Arguments

`loads`	A matrix of loadings `loads` here
`loading`	Are these loadings or a correlation matrix `loading`
`sorting`	Should the variables be sorted `sorting`

Details

"A common model for representing psychological data is simple structure (Thurstone, 1947). According to one common interpretation, data are simple structured when items or scales have non-zero factor loadings on one and only one factor (Revelle & Rocklin, 1979). Despite the commonplace application of simple structure, some psychological models are defined by a lack of simple structure. Circumplexes (Guttman, 1954) are one kind of model in which simple structure is lacking.

"A number of elementary requirements can be teased out of the idea of circumplex structure. First, circumplex structure implies minimally that variables are interrelated; random noise does not a circumplex make. Second, circumplex structure implies that the domain in question is optimally represented by two and only two dimensions. Third, circumplex structure implies that variables do not group or clump along the two axes, as in simple structure, but rather that there are always interstitial variables between any orthogonal pair of axes (Saucier, 1992). In the ideal case, this quality will be reflected in equal spacing of variables along the circumference of the circle (Gurtman, 1994; Wiggins, Steiger, & Gaelick, 1981). Fourth, circumplex structure implies that variables have a constant radius from the center of the circle, which implies that all variables have equal communality on the two circumplex dimensions (Fisher, 1997; Gurtman, 1994). Fifth, circumplex structure implies that all rotations are equally good representations of the domain (Conte & Plutchik, 1981; Larsen & Diener, 1992). (Acton and Revelle, 2004)

Acton and Revelle reviewed the effectiveness of 10 tests of circumplex structure and found that four did a particularly good job of discriminating circumplex structure from simple structure, or circumplexes from ellipsoidal structures. Unfortunately, their work was done in Pascal and is not easily available. Here we release R code to do the four most useful tests:

1 The Gap test of equal spacing

2 Fisher's test of equality of axes

3 A test of indifference to Rotation

4 A test of equal Variance of squared factor loadings across arbitrary rotations.

To interpret the values of these various tests, it is useful to compare the particular solution to simulated solutions representing pure cases of circumplex and simple structure. See the example output from `circ.simulation` and compare these plots with the results of the circ.test.

Value

A list of four items is returned. These are the gap, fisher, rotation and variance test results.

`gaps`	gap.test
`fisher`	fisher.test
`RT`	rotation.test
`VT`	variance.test

Note

Of the 10 criterion discussed in Acton and Revelle (2004), these tests operationalize the four most useful.

Author(s)

William Revelle

References

Acton, G. S. and Revelle, W. (2004) Evaluation of Ten Psychometric Criteria for Circumplex Structure. Methods of Psychological Research Online, Vol. 9, No. 1 http://personality-project.org/revelle/publications/acton.revelle.mpr110_10.pdf

See Also

To understand the results of the circ.tests it it best to compare it to simulated values. Thus, see `circ.simulation`, `sim.circ`

Examples

```
circ.data <- circ.sim(24,500)
circ.fa <- fa(circ.data,2)
plot(circ.fa,title="Circumplex Structure")
ct <- circ.tests(circ.fa)
#compare with non-circumplex data
simp.data <- item.sim(24,500)
simp.fa <- fa(simp.data,2)
plot(simp.fa,title="Simple Structure")
st <- circ.tests(simp.fa)
res <- rbind(ct[1:4],st[1:4])
rownames(res) <- c("circumplex","Simple")
```

```
print(res,digits=2)
```

`cities` *Distances between 11 US cities*

Description

Airline distances between 11 US cities may be used as an example for multidimensional scaling or cluster analysis.

Usage

```
data(cities)
```

Format

A data frame with 11 observations on the following 11 variables.

`ATL` Atlana, Georgia

`BOS` Boston, Massachusetts

`ORD` Chicago, Illinois

`DCA` Washington, District of Columbia

`DEN` Denver, Colorado

`LAX` Los Angeles, California

`MIA` Miami, Florida

`JFK` New York, New York

`SEA` Seattle, Washington

`SFO` San Francisco, California

`MSY` New Orleans, Lousianna

Details

An 11 x11 matrix of distances between major US airports. This is a useful demonstration of multiple dimensional scaling.

city.location is a dataframe of longitude and latitude for those cities.

Note that the 2 dimensional MDS solution does not perfectly capture the data from these city distances. Boston, New York and Washington, D.C. are located slightly too far west, and Seattle and LA are slightly too far south.

Source

http://www.timeanddate.com/worldclock/distance.html

Examples

```
data(cities)
city.location[,1] <- -city.location[,1]
#not run
#an overlay map can be added if the package maps is available
#
#
#libary(maps)
#map("usa")
#title("MultiDimensional Scaling of US cities")
#points(city.location)

plot(city.location, xlab="Dimension 1", ylab="Dimension 2",
   main ="Multidimensional scaling of US cities")
city.loc <- cmdscale(cities, k=2) #ask for a 2 dimensional solution  round(city.loc,0)
city.loc <- -city.loc
 city.loc <- rescale(city.loc,apply(city.location,2,mean),apply(city.location,2,sd))
points(city.loc,type="n")
text(city.loc,labels=names(cities))
```

`cluster.fit` *cluster Fit: fit of the cluster model to a correlation matrix*

Description

How well does the cluster model found by `ICLUST` fit the original correlation matrix? A similar algorithm `factor.fit` is found in `VSS`. This function is internal to ICLUST but has more general use as well.

In general, the cluster model is a Very Simple Structure model of complexity one. That is, every item is assumed to represent only one factor/cluster. Cluster fit is an analysis of how well this model reproduces a correlation matrix. Two measures of fit are given: cluster fit and factor fit. Cluster fit assumes that variables that define different clusters are orthogonal. Factor fit takes the loadings generated by a cluster model, finds the cluster loadings on all clusters, and measures the degree of fit of this somewhat more complicated model. Because the cluster loadings are similar to, but not identical to factor loadings, the factor fits found here and by `factor.fit` will be similar.

Usage

```
cluster.fit(original, load, clusters, diagonal = FALSE)
```

Arguments

`original`	The original correlation matrix being fit
`load`	Cluster loadings – that is, the correlation of individual items with the clusters, corrected for item overlap
`clusters`	The cluster structure
`diagonal`	Should we fit the diagonal as well?

Details

The cluster model is similar to the factor model: R is fitted by C'C. Where C <- Cluster definition matrix x the loading matrix. How well does this model approximate the original correlation matrix and how does this compare to a factor model?

The fit statistic is a comparison of the original (squared) correlations to the residual correlations. Fit = 1 - r*2/r2 where r* is the residual correlation of data - model and model = C'C.

Value

`clusterfit` The cluster model is a reduced form of the factor loading matrix. That is, it is the product of the elements of the cluster matrix * the loading matrix.

`factorfit` How well does the complete loading matrix reproduce the correlation matrix?

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

http://personality-project.org/r/r.ICLUST.html

See Also

VSS, ICLUST, factor2cluster, cluster.cor, factor.fit

Examples

```
r.mat<- Harman74.cor$cov
iq.clus <- ICLUST(r.mat,nclusters =2)
fit <- cluster.fit(r.mat,iq.clus$loadings,iq.clus$clusters)
fit
```

---

cluster.loadings *Find item by cluster correlations, corrected for overlap and reliability*

Description

Given a n x n correlation matrix and a n x c matrix of -1,0,1 cluster weights for those n items on c clusters, find the correlation of each item with each cluster. If the item is part of the cluster, correct for item overlap. Part of the ICLUST set of functions, but useful for many item analysis problems.

Usage

```
cluster.loadings(keys, r.mat, correct = TRUE,SMC=TRUE)
```

Arguments

keys	Cluster keys: a matrix of -1,0,1 cluster weights
r.mat	A correlation matrix
correct	Correct for reliability
SMC	Use the squared multiple correlation as a communality estimate, otherwise use the greatest correlation for each variable

Details

Given a set of items to be scored as (perhaps overlapping) clusters and the intercorrelation matrix of the items, find the clusters and then the correlations of each item with each cluster. Correct for item overlap by replacing the item variance with its average within cluster inter-item correlation.

Although part of ICLUST, this may be used in any SAPA (http://sapa-project.org) application where we are interested in item- whole correlations of items and composite scales.

These loadings are particularly interpretable when sorted by absolute magnitude for each cluster (see ICLUST.sort).

Value

loadings	A matrix of item-cluster correlations (loadings)
cor	Correlation matrix of the clusters
corrected	Correlation matrix of the clusters, raw correlations below the diagonal, alpha on diagonal, corrected for reliability above the diagonal
sd	Cluster standard deviations
alpha	alpha reliabilities of the clusters
G6	G6* Modified estimated of Guttman Lambda 6
count	Number of items in the cluster

Note

Although part of ICLUST, this may be used in any SAPA application where we are interested in item- whole correlations of items and composite scales.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

ICLUST: http://personality-project.org/r/r.ICLUST.html

See Also

ICLUST, factor2cluster, cluster.cor

Examples

```
r.mat<- Harman74.cor$cov
clusters <- matrix(c(1,1,1,rep(0,24),1,1,1,1,rep(0,17)),ncol=2)
cluster.loadings(clusters,r.mat)
```

---

cluster.plot *Plot factor/cluster loadings and assign items to clusters by their highest loading.*

Description

Cluster analysis and factor analysis are procedures for grouping items in terms of a smaller number of (latent) factors or (observed) clusters. Graphical presentations of clusters typically show tree structures, although they can be represented in terms of item by cluster correlations.

Cluster.plot plots items by their cluster loadings (taken, e.g., from ICLUST) or factor loadings (taken, eg., from fa). Cluster membership may be assigned apriori or may be determined in terms of the highest (absolute) cluster loading for each item.

If the input is an object of class "kmeans", then the cluster centers are plotted.

Usage

```
cluster.plot(ic.results, cluster = NULL, cut = 0, labels=NULL,
          title = "Cluster plot",pch=18,pos,show.points=TRUE,choose=NULL,...)
fa.plot(ic.results, cluster = NULL, cut = 0, labels=NULL,title,
           jiggle=FALSE,amount=.02,pch=18,pos,show.points=TRUE,choose=NULL,...)
factor.plot(ic.results, cluster = NULL, cut = 0, labels=NULL,title,jiggle=FALSE,
                  amount=.02,pch=18,pos,show.points=TRUE,...)  #deprecated
```

Arguments

ic.results	A factor analysis or cluster analysis output including the loadings, or a matrix of item by cluster correlations. Or the output from a kmeans cluster analysis.
cluster	A vector of cluster membership
cut	Assign items to clusters if the absolute loadings are > cut
labels	If row.names exist they will be added to the plot, or, if they don't, labels can be specified. If labels =NULL, and there are no row names, then variables are labeled by row number.)
title	Any title

jiggle	When plotting with factor loadings that are almost identical, it is sometimes useful to "jiggle" the points by jittering them. The default is to not jiggle.
amount	if jiggle=TRUE, then how much should the points be jittered?
pch	factor and clusters are shown with different pch values, starting at pch+1
pos	Position of the text for labels for two dimensional plots. 1=below, 2 = left, 3 = above, 4= right
show.points	When adding labels to the points, should we show the points as well as the labels. For many points, better to not show them, just the labels.
choose	Specify the factor/clusters to plot
...	Further options to plot

Details

Results of either a factor analysis or cluster analysis are plotted. Each item is assigned to its highest loading factor, and then identified by variable name as well as cluster (by color). The cluster assignments can be specified to override the automatic clustering by loading. Both of these functions may be called directly or by calling the generic plot function. (see example).

Value

Graphical output is presented.

Author(s)

William Revelle

See Also

ICLUST, ICLUST.graph, fa.graph, plot.psych

Examples

```
circ.data <- circ.sim(24,500)
circ.fa <- fa(circ.data,2)
plot(circ.fa,cut=.5)
f5 <- fa(bfi[1:25],5)
plot(f5,labels=colnames(bfi)[1:25],show.points=FALSE)
plot(f5,labels=colnames(bfi)[1:25],show.points=FALSE,choose=c(1,2,4))
```

`cluster2keys` *Convert a cluster vector (from e.g., kmeans) to a keys matrix suitable for scoring item clusters.*

Description

The output of the kmeans clustering function produces a vector of cluster membership. The `score.items` and `cluster.cor` functions require a matrix of keys. cluster2keys does this.

May also be used to take the output of an `ICLUST` analysis and find a keys matrix. (By doing a call to the `factor2cluster` function.

Usage

```
cluster2keys(c)
```

Arguments

`c`	A vector of cluster assignments or an object of class “kmeans" that contains a vector of clusters.

Details

Note that because kmeans will not reverse score items, the clusters defined by kmeans will not necessarily match those of ICLUST with the same number of clusters extracted.

Value

`keys`	A matrix of keys suitable for score.items or cluster.cor

Author(s)

William Revelle

See Also

`cluster.cor`,`score.items`, `factor2cluster`, `make.keys`

Examples

```
test.data <- Harman74.cor$cov
kc <- kmeans(test.data,4)
keys <- cluster2keys(kc)
keys  #these match those found by ICLUST
cluster.cor(keys,test.data)
```

cohen.kappa — *Find Cohen's kappa and weighted kappa coefficients for correlation of two raters*

Description

Cohen's kappa (Cohen, 1960) and weighted kappa (Cohen, 1968) may be used to find the agreement of two raters when using nominal scores.

weighted.kappa is (probability of observed matches - probability of expected matches)/(1 - probability of expected matches). Kappa just considers the matches on the main diagonal. Weighted kappa considers off diagonal elements as well.

Usage

```
cohen.kappa(x, w=NULL,n.obs=NULL,alpha=.05)
wkappa(x, w = NULL)    #deprectated
```

Arguments

`x`	Either a two by n data with categorical values from 1 to p or a p x p table. If a data array, a table will be found.
`w`	A p x p matrix of weights. If not specified, they are set to be 0 (on the diagonal) and (distance from diagonal) off the diagonal)^2.
`n.obs`	Number of observations (if input is a square matrix.
`alpha`	Probability level for confidence intervals

Details

When cateogorical judgments are made with two cateories, a measure of relationship is the phi coefficient. However, some categorical judgments are made using more than two outcomes. For example, two diagnosticians might be asked to categorize patients three ways (e.g., Personality disorder, Neurosis, Psychosis) or to categorize the stages of a disease. Just as base rates affect observed cell frequencies in a two by two table, they need to be considered in the n-way table (Cohen, 1960).

Kappa considers the matches on the main diagonal. A penalty function (weight) may be applied to the off diagonal matches. If the weights increase by the square of the distance from the diagonal, weighted kappa is similar to an Intra Class Correlation (`ICC`).

Derivations of weighted kappa are sometimes expressed in terms of similarities, and sometimes in terms of dissimilarities. In the latter case, the weights on the diagonal are 1 and the weights off the diagonal are less than one. In this, if the weights are 1 - squared distance from the diagonal / k, then the result is similar to the ICC (for any positive k).

cohen.kappa may use either similarity weighting (diagonal = 0) or dissimilarity weighting (diagonal = 1) in order to match various published examples.

The input may be a two column data.frame or matrix with columns representing the two judges and rows the subjects being rated. Alternatively, the input may be a square n x n matrix of counts or proportion of matches. If proportions are used, it is necessary to specify the number of observations (n.obs) in order to correctly find the confidence intervals.

The confidence intervals are based upon the variance estimates discussed by Fleiss, Cohen, and Everitt who corrected the formulae of Cohen (1968) and Blashfield.

Value

`kappa`	Unweighted kappa
`weighted.kappa`	The default weights are quadratric.
`var.kappa`	Variance of kappa
`var.weighted`	Variance of weighted kappa
`n.obs`	number of observations
`weight`	The weights used in the estimation of weighted kappa
`confid`	The alpha/2 confidence intervals for unweighted and weighted kappa
`plevel`	The alpha level used in determining the confidence limits

Note

As is true of many R functions, there are alternatives in other packages. The Kappa function in the vcd package estimates unweighted and weighted kappa and reports the variance of the estimate. The input is a square matrix. The ckappa and wkappa functions in the psy package take raw data matrices.

To avoid confusion with Kappa (from vcd) or the kappa function from base, the function was originally named wkappa. With additional features modified from psy::ckappa to allow input with a different number of categories, the function has been renamed cohen.kappa.

Unfortunately, to make it more confusing, the weights described by Cohen are a function of the reciprocals of those discucssed by Fleiss and Cohen. The cohen.kappa function uses the appropriate formula for Cohen or Fleiss-Cohen weights.

Author(s)

William Revelle

References

Banerjee, M., Capozzoli, M., McSweeney, L and Sinha, D. (1999) Beyond Kappa: A review of interrater agreement measures The Canadian Journal of Statistics / La Revue Canadienne de Statistique, 27, 3-23

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20 37-46

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. Psychological Bulletin, 70, 213-220.

Fleiss, J. L., Cohen, J. and Everitt, B.S. (1969) Large sample standard errors of kappa and weighted kappa. Psychological Bulletin, 72, 332-327.

Zwick, R. (1988) Another look at interrater agreement. Psychological Bulletin, 103, 374 - 378.

Examples

```
#rating data (with thanks to Tim Bates)
rater1 = c(1,2,3,4,5,6,7,8,9) # rater one's ratings
rater2 = c(1,3,1,6,1,5,5,6,7) # rater one's ratings
cohen.kappa(x=cbind(rater1,rater2))

#data matrix taken from Cohen
cohen <- matrix(c(
0.44, 0.07, 0.09,
0.05, 0.20, 0.05,
0.01, 0.03, 0.06),ncol=3,byrow=TRUE)

#cohen.weights  weight differences
cohen.weights <- matrix(c(
0,1,3,
1,0,6,
3,6,0),ncol=3)


cohen.kappa(cohen,cohen.weights,n.obs=200)
#cohen reports .492 and .348

#another set of weights
#what if the weights are non-symmetric
wc <- matrix(c(
0,1,4,
1,0,6,
2,2,0),ncol=3,byrow=TRUE)
cohen.kappa(cohen,wc)
#Cohen reports kw = .353

cohen.kappa(cohen,n.obs=200)  #this uses the squared weights

fleiss.cohen <- 1 - cohen.weights/9
cohen.kappa(cohen,fleiss.cohen,n.obs=200)

#however, Fleiss, Cohen and Everitt weight similarities
fleiss <- matrix(c(
106, 10,4,
22,28, 10,
2, 12,  6),ncol=3,byrow=TRUE)

#Fleiss weights the similarities
weights <- matrix(c(
 1.0000, 0.0000, 0.4444,
 0.0000, 1.0000, 0.6667,
 0.4444, 0.6667, 1.0000),ncol=3)

 cohen.kappa(fleiss,weights,n.obs=200)

 #another example is comparing the scores of two sets of twins
```

```
 #data may be a 2 column matrix
 #compare weighted and unweighted
 #also look at the ICC for this data set.
 twins <- matrix(c(
    1, 2,
    2, 3,
    3, 4,
    5, 6,
    6, 7), ncol=2,byrow=TRUE)
  cohen.kappa(twins)

#data may be explicitly categorical
x <- c("red","yellow","blue","red")
y <- c("red",  "blue", "blue" ,"red")
xy.df <- data.frame(x,y)
ck <- cohen.kappa(xy.df)
ck
ck$agree

#finally, input can be a data.frame of ratings from more than two raters
ratings <- matrix(rep(1:5,4),ncol=4)
ratings[1,2] <- ratings[2,3] <- ratings[3,4] <- NA
ratings[2,1] <- ratings[3,2] <- ratings[4,3] <- 1
cohen.kappa(ratings)
```

`comorbidity` *Convert base rates of two diagnoses and their comorbidity into phi, Yule, and tetrachorics*

Description

In medicine and clinical psychology, diagnoses tend to be categorical (someone is depressed or not, someone has an anxiety disorder or not). Cooccurrence of both of these symptoms is called comorbidity. Diagnostic categories vary in their degree of comorbidity with other diagnostic categories. From the point of view of correlation, comorbidity is just a name applied to one cell in a four fold table. It is thus possible to analyze comorbidity rates by considering the probability of the separate diagnoses and the probability of the joint diagnosis. This gives the two by two table needed for a phi, Yule, or tetrachoric correlation.

Usage

```
comorbidity(d1, d2, com, labels = NULL)
```

Arguments

`d1`	Proportion of diagnostic category 1
`d2`	Proportion of diganostic category 2
`com`	Proportion of comorbidity (diagnostic category 1 and 2)
`labels`	Names of categories 1 and 2

Value

twobytwo	The two by two table implied by the input
phi	Phi coefficient of the two by two table
Yule	Yule coefficient of the two by two table
tetra	Tetrachoric coefficient of the two by two table

Author(s)

William Revelle

See Also

phi, Yule

Examples

```
comorbidity(.2,.15,.1,c("Anxiety","Depression"))
```

---

cor.ci *Bootstrapped confidence intervals for raw and composite correlations*

Description

Although normal theory provides confidence intervals for correlations, this is particularly problematic with Synthetic Aperture Personality Assessment (SAPA) data where the individual items are Massively Missing at Random. Bootstrapped confidence intervals are found for Pearson, Spearman, Kendall, tetrachoric, or polychoric correlations and for scales made from those correlations.

Usage

```
cor.ci(x, keys = NULL, n.iter = 100,  p = 0.05,overlap = FALSE,
 poly = FALSE, method = "pearson", plot=TRUE,...)
 corCi(x, keys = NULL, n.iter = 100,  p = 0.05,overlap = FALSE,
 poly = FALSE, method = "pearson", plot=TRUE,...)
```

Arguments

x	The raw data
keys	If NULL, then the confidence intervals of the raw correlations are found. Otherwise, composite scales are formed from the keys applied to the correlation matrix (in a logic similar to cluster.cor but without the bells and whistles) and the confidence of those composite scales intercorrelations.
n.iter	The number of iterations to bootstrap over. This will be very slow if using tetrachoric/or polychoric correlations.

`p`	The upper and lower confidence region will include 1-p of the distribution.
`overlap`	If true, the correlation between overlapping scales is corrected for item overlap.
`poly`	if FALSE, then find the correlations using the method specified (defaults to Pearson). If TRUE, the polychoric correlations will be found (slowly). Because the polychoric function uses multicores (if available), and cor.ci does as well, the number of cores used is options("mc.cores")^2.
`method`	"pearson","spearman", "kendall"
`plot`	Show the correlation plot with correlations scaled by the probability values. To show the matrix in terms of the confidence intervals, use `cor.plot.upperLowerCi`.
`...`	Other parameters for axis (e.g., cex.axis to change the font size, srt to rotate the numbers in the plot)

Details

The original data are and correlations are found. If keys are specified (the normal case), then composite scales based upon the correlations are found and reported. This is the same procedure as done using `cluster.cor` or `scoreItems`.

Then, n.iter times, the data are recreated by sampling subjects (rows) with replacement and the correlations (and composite scales) are found again (and again and again). Mean and standard deviations of these values are calculated based upon the Fisher Z transform of the correlations. Summary statistics include the original correlations and their confidence intervals. For those who want the complete set of replications, those are available as an object in the resulting output.

Although particularly useful for SAPA (`http://sapa-project.org`) type data, this will work for any normal data set as well.

Although the correlations are shown automatically as a `cor.plot`, it is possible to show the upper and lower confidence intervals by using `cor.plot.upperLowerCi`. This will also return, invisibly, a matrix for printing with the lower and upper bounds of the correlations shown below and above the diagonal.

Value

`rho`	The original (composite) correlation matrix.
`means`	Mean (Fisher transformed) correlation
`sds`	Standard deviation of Fisher transformed correlations
`ci`	Mean +/- alpha/2 of the z scores as well as the alpha/2 and 1-alpha/2 quantiles. These are labeled as lower.emp(ircal), lower.norm(al), upper.norm and upper.emp.
`replicates`	The observed replication values so one can do one's own estimates

Author(s)

William Revelle

References

For SAPA type data, see Revelle, W., Wilt, J., and Rosenthal, A. (2010) Personality and Cognition: The Personality-Cognition Link. In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

See Also

`make.keys`, `cluster.cor`, and `scoreItems` for forming synthetic correlation matrices from composites of item correlations. See `scoreOverlap` for correcting for item overlap in scales. See also `corr.test` for standard significance testing of correlation matrices. See also `lowerCor` for finding and printing correlation matrices, as well as `lowerMat` for displaying them. Also see `cor.plot.upperLowerCi` for displaying the confidence intervals graphically.

Examples

```
cor.ci(bfi[1:200,1:10])  # just the first 10 variables
#The keys have overlapping scales
  keys.list <- list(agree=c("-A1","A2","A3","A4","A5"), conscientious= c("C1",
  "C2","C3","-C4","-C5"),extraversion=c("-E1","-E2","E3","E4","E5"), neuroticism=
  c("N1", "N2", "N3","N4","N5"), openness = c("O1","-O2","O3","O4","-O5"),
 alpha=c("-A1","A2","A3","A4","A5","C1","C2","C3","-C4","-C5","N1","N2","N3","N4","N5"),
beta = c("-E1","-E2","E3","E4","E5","O1","-O2","O3","O4","-O5") )
  keys <- make.keys(bfi,keys.list)

#do not correct for item overlap
rci <-  cor.ci(bfi[1:200,],keys,n.iter=10,main="correlation with overlapping scales")
#also shows the graphic -note the overlap
#correct for overlap
rci <-  cor.ci(bfi[1:200,],keys,overlap=TRUE, n.iter=10,main="Correct for overlap")
#show the confidence intervals
ci <- cor.plot.upperLowerCi(rci)  #to show the upper and lower confidence intervals
ci   #print the confidence intervals in matrix form
```

---

cor.plot *Create an image plot for a correlation or factor matrix*

Description

Correlation matrices may be shown graphically by using the image function to emphasize structure. This is a particularly useful tool for showing the structure of correlation matrices with a clear structure. Partially meant for the pedagogical value of the graphic for teaching or discussing factor analysis and other multivariate techniques.

Usage

```
corPlot(r,numbers=FALSE,colors=TRUE,n=51,main=NULL,zlim=c(-1,1),
  show.legend=TRUE, labels=NULL,n.legend=10,keep.par=TRUE,select=NULL,
   pval=NULL,cuts=c(.001,.01),cex,MAR,upper=TRUE,diag=TRUE,...)
```

```
cor.plot(r,numbers=FALSE,colors=TRUE,n=51,main=NULL,zlim=c(-1,1),
  show.legend=TRUE, labels=NULL,n.legend=10,keep.par=TRUE,select=NULL,
   pval=NULL,cuts=c(.001,.01),cex,MAR,upper=TRUE,diag=TRUE,...)


cor.plot.upperLowerCi(R,numbers=TRUE,cuts=c(.001,.01,.05),select=NULL,
      main="Upper and lower confidence intervals of correlations",...)
```

Arguments

`r`	A correlation matrix or the output of `fa`, `principal` or `omega`.
`R`	The object returned from `cor.ci`
`numbers`	Display the numeric value of the correlations. Defaults to FALSE.
`colors`	Defaults to TRUE and colors use colors from the colorRampPalette from red through white to blue, but colors=FALSE will use a grey scale
`n`	The number of levels of shading to use. Defaults to 51
`main`	A title. Defaults to "correlation plot"
`zlim`	The range of values to color – defaults to -1 to 1
`show.legend`	A legend (key) to the colors is shown on the right hand side
`labels`	if NULL, use column and row names, otherwise use labels
`n.legend`	How many categories should be labelled in the legend?
`keep.par`	restore the graphic parameters when exiting
`pval`	scale the numbers by their pvals, categorizing them based upon the values of cuts
`cuts`	Scale the numbers by the categories defined by pval < cuts
`select`	Select the subset of variables to plot
`cex`	Character size. Should be reduced a bit for large numbers of variables.
`MAR`	Allows for adjustment of the margins if using really long labels or big fonts
`upper`	Should the upper off diagonal matrix be drawn, or left blank?
`diag`	Should we show the diagonal?
`...`	Other parameters for axis (e.g., cex.axis to change the font size, srt to rotate the numbers in the plot)

Details

When summarizing the correlations of large data bases or when teaching about factor analysis or cluster analysis, it is useful to graphically display the structure of correlation matrices. This is a simple graphical display using the image function.

The difference between mat.plot with a regular image plot is that the primary diagonal goes from the top left to the lower right. zlim defines how to treat the range of possible values. -1 to 1 and the color choice is more reasonable. Setting it as c(0,1) will lead to negative correlations treated as zero. This is advantageous when showing general factor structures, because it makes the 0 white.

The default shows a legend for the color coding on the right hand side of the figure.

Inspired, in part, by a paper by S. Dray (2008) on the number of components problem.

Modified following suggestions by David Condon and Josh Wilt to use a more meaningful color choice ranging from dark red (-1) through white (0) to dark blue (1). Further modified to include the numerical value of the correlation. (Inspired by the corrplot package). These values may be scaled according the the probability values found in `cor.ci` or `corr.test`.

Unless specified, the font size is dynamically scaled to have a cex = 10/max(nrow(r),ncol(r). This can produce fairly small fonts for large problems. The font size of the labels may be adjusted using cex.axis which defaults to one.

By default `cor.ci` calls cor.plot.upperLowerCi and scales the correlations based upon "significance" values. The correlations plotted are the upper and lower confidence boundaries. To show the correlations themselves, call cor.plot directly.

If using the output of `corr.test`, the upper off diagonal will be scaled by the corrected probability, the lower off diagonal the scaling is the uncorrected probabilities.

If using the output of `corr.test` or `cor.ci` as input to `cor.plot.upperLowerCi`, the upper off diagonal will be the upper bounds and the lower off diagonal the lower bounds of the confidence intervals.

Author(s)

William Revelle

References

Dray, Stephane (2008) On the number of principal components: A test of dimensionality based on measurements of similarity between matrices. Computational Statistics \& Data Analysis. 52, 4, 2228-2237.

See Also

`fa`, `mat.sort`, `cor.ci`, `corr.test`.

Examples

```
cor.plot(Thurstone,main="9 cognitive variables from Thurstone")
#just blue implies positive manifold
#select just some variables to plot
cor.plot(Thurstone, zlim=c(0,1),main="9 cognitive variables from Thurstone",select=1:4)

#now red means less than .5
cor.plot(mat.sort(Thurstone),TRUE,zlim=c(0,1),
       main="9 cognitive variables from Thurstone (sorted by factor loading) ")
simp <- sim.circ(24)
cor.plot(cor(simp),main="24 variables in a circumplex")

#scale by raw and adjusted probabilities
rs <- corr.test(sat.act[1:200,] ) #find the probabilities of the correlations
cor.plot(r=rs$r,numbers=TRUE,pval=rs$p,main="Correlations scaled by probability values")
 #Show the upper and lower confidence intervals
cor.plot.upperLowerCi(R=rs,numbers=TRUE)
```

cor.smooth	*Smooth a non-positive definite correlation matrix to make it positive definite*

Description

Factor analysis requires positive definite correlation matrices. Unfortunately, with pairwise deletion of missing data or if using tetrachoric or polychoric correlations, not all correlation matrices are positive definite. cor.smooth does a eigenvector (principal components) smoothing. Negative eigen values are replaced with 100 * eig.tol, the matrix is reproduced and forced to a correlation matrix using cov2cor.

Usage

```
cor.smooth(x,eig.tol=10^-12)
cor.smoother(x,cut=.01)
```

Arguments

x	A correlation matrix or a raw data matrix.
eig.tol	the minimum acceptable eigenvalue.
cut	Report all abs(residuals) > cut

Details

The smoothing is done by eigen value decomposition. eigen values < eig.tol are changed to 100 * eig.tol. The positive eigen values are rescaled to sum to the number of items. The matrix is recomputed (eigen.vectors %*% diag(eigen.values) %*% t(eigen.vectors) and forced to a correlation matrix using cov2cor. (See Bock, Gibbons and Muraki, 1988 and Wothke, 1993).

This does not implement the Knol and ten Berge (1989) solution, nor do nearcor and posdefify in sfmsmisc, not does nearPD in Matrix. As Martin Maechler puts it in the posdedify function, "there are more sophisticated algorithms to solve this and related problems."

cor.smoother examines all of nvar minors of rank nvar-1 by systematically dropping one variable at a time and finding the eigen value decomposition. It reports those variables, which, when dropped, produce a positive definite matrix. It also reports the number of negative eigenvalues when each variable is dropped. Finally, it compares the original correlation matrix to the smoothed correlation matrix and reports those items with absolute deviations great than cut. These are all hints as to what might be wrong with a correlation matrix.

Value

The smoothed matrix with a warning reporting that smoothing was necessary (if smoothing was in fact necessary).

Author(s)

William Revelle

References

R. Darrell Bock, Robert Gibbons and Eiji Muraki (1988) Full-Information Item Factor Analysis. Applied Psychological Measurement, 12 (3), 261-280.

Werner Wothke (1993), Nonpositive definite matrices in structural modeling. In Kenneth A. Bollen and J. Scott Long (Editors),Testing structural equation models, Sage Publications, Newbury Park.

D.L. Knol and JMF ten Berge (1989) Least squares approximation of an improper correlation matrix by a proper one. Psychometrika, 54, 53-61.

See Also

tetrachoric, polychoric, fa and irt.fa, and the burt data set.

See also nearcor and posdefify in the sfsmisc package and nearPD in the Matrix package.

Examples

```
bs <- cor.smooth(burt)  #burt data set is not positive definite
plot(burt[lower.tri(burt)],bs[lower.tri(bs)],ylab="smoothed values",xlab="original values")
abline(0,1,lty="dashed")

round(burt - bs,3)
fa(burt,2) #this throws a warning that the matrix yields an improper solution
#Smoothing first throws a warning that the matrix was improper,
#but produces a better solution
fa(cor.smooth(burt),2)

#this next example is a correlation matrix from DeLeuw used as an example
#in Knol and ten Berge.
#the example is also used in the nearcor documentation
 cat("pr is the example matrix used in Knol DL, ten Berge (1989)\n")
 pr <- matrix(c(1,     0.477, 0.644, 0.478, 0.651, 0.826,
0.477, 1,     0.516, 0.233, 0.682, 0.75,
0.644, 0.516, 1,     0.599, 0.581, 0.742,
0.478, 0.233, 0.599, 1,     0.741, 0.8,
0.651, 0.682, 0.581, 0.741, 1,     0.798,
0.826, 0.75,  0.742, 0.8,   0.798, 1),
      nrow = 6, ncol = 6)

sm <- cor.smooth(pr)
resid <- pr - sm
# several goodness of fit tests
# from Knol and ten Berge
tr(resid %*% t(resid)) /2

# from nearPD
sum(resid^2)/2
```

cor.wt *The sample size weighted correlation may be used in correlating aggregated data*

Description

If using aggregated data, the correlation of the means does not reflect the sample size used for each mean. cov.wt in RCore does this and returns a covariance matrix or the correlation matrix. The cor.wt function weights by sample size or by standard errors and by default return correlations.

Usage

```
cor.wt(data,vars=NULL, w=NULL,sds=NULL, cor=TRUE)
```

Arguments

data	A matrix or data frame
vars	Variables to analyze
w	A set of weights (e.g., the sample sizes)
sds	Standard deviations of the samples (used if weighting by standard errors)
cor	Report correlations (the default) or covariances

Details

A weighted correlation is just $r_{ij} = \frac{\sum(wt_k(x_{ik} - x_{jk})}{\sqrt{wt_{ik}\sum(x_{ik}^2)wt_jk\sum(x_{jk}^2)}}$ where x_{ik} is a deviation from the weighted mean.

The weighted correlation is appropriate for correlating aggregated data, where individual data points might reflect the means of a number of observations. In this case, each point is weighted by its sample size (or alternatively, by the standard error). If the weights are all equal, the correlation is just a normal Pearson correlation.

Used when finding correlations of group means found using statsBy.

Value

cor	The weighted correlation
xwt	The data as weighted deviations from the weighted mean
wt	The weights used (calculated from the sample sizes).
mean	The weighted means
xc	Unweighted, centered deviation scores from the weighted mean
xs	Deviation scores weighted by the standard error of each sample mean

Note

A generalization of cov.wt in core R

Author(s)

William Revelle

See Also

See Also as cov.wt, statsBy

Examples

```
means.by.age <- statsBy(sat.act,"age")
wt.cors <- cor.wt(means.by.age)
lowerMat(wt.cors$r)  #show the weighted correlations
unwt <- lowerCor(means.by.age$mean)
mixed <- lowerUpper(unwt,wt.cors$r)  #combine both results
cor.plot(mixed,TRUE,main="weighted versus unweighted correlations")
diff <- lowerUpper(unwt,wt.cors$r,TRUE)
cor.plot(diff,TRUE,main="differences of weighted versus unweighted correlations")
```

---

cor2dist *Convert correlations to distances (necessary to do multidimensional scaling of correlation data)*

Description

A minor helper function to convert correlations (ranging from -1 to 1) to distances (ranging from 0 to 2). $d = \sqrt{(2(1-r))}$.

Usage

```
cor2dist(x)
```

Arguments

x	If square, then assumed to be a correlation matrix, otherwise the correlations are found first.

Value

dist: a square matrix of distances.

Note

For an example of doing multidimensional scaling on data that are normally factored, see Revelle (in prep)

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at http://personality-project.org/r/book/

`corFiml` *Find a Full Information Maximum Likelihood (FIML) correlation or covariance matrix from a data matrix with missing data*

Description

Makes use of functions adapted from the lavaan package to find FIML covariance/correlation matrices. FIML can be much slower than the normal pairwise deletion option of cor, but provides slightly more precise estimates.

Usage

```
corFiml(x, covar = FALSE,show=FALSE)
```

Arguments

`x`	A data.frame or data matrix
`covar`	By default, just return the correlation matrix. If covar is TRUE, return a list containing the covariance matrix and the ML fit function.
`show`	If show=TRUE, then just show the patterns of missingness, but don't do the FIML. Useful for understanding the process of fiml.

Details

In the presence of missing data, Full Information Maximum Likelihood (FIML) is an alternative to simply using the pairwise correlations. The implementation in the lavaan package for structural equation modeling has been adapted for the simpler case of just finding the correlations or covariances.

The pairwise solution for any pair of variables is insensitive to other variables included in the matrix. On the other hand, the ML solution depends upon the entire set of items being correlated. This will lead to slightly different solutions for different subsets of variables.

The basic FIML algorithm is to find the pairwise ML solution for covariances and means for every pattern of missingness and then to weight the solution by the size of every unique pattern of missingness.

Value

`cor`	The correlation matrix found using FIML
`cov`	The covariance matrix found using FIML
`fx`	The ML fit function

Note

The functions used in lavaan are not exported and so have been copied (and simplified) to the psych package.

Author(s)

Wiliam Revelle

See Also

To use the resulting correlations, see `fa`. To see the pairwise pattern of missingness, see `count.pairwise`.

Examples

```
rML <- corFiml(bfi[20:27])
rpw <- cor(bfi[20:27],use="pairwise")
round(rML - rpw,3)
mp <- corFiml(bfi[20:27],show=TRUE)
mp
```

`corr.test` *Find the correlations, sample sizes, and probability values between elements of a matrix or data.frame.*

Description

Although the cor function finds the correlations for a matrix, it does not report probability values. corr.test uses cor to find the correlations for either complete or pairwise data and reports the sample sizes and probability values as well. For symmetric matrices, raw probabilites are reported below the diagonal and correlations adjusted for multiple comparisons above the diagonal. In the case of different x and ys, the default is to adjust the probabilities for multiple tests.

Usage

```
corr.test(x, y = NULL, use = "pairwise",method="pearson",adjust="holm", alpha=.05,ci=TRUE)
corr.p(r,n,adjust="holm",alpha=.05)
```

Arguments

`x`	A matrix or dataframe
`y`	A second matrix or dataframe with the same number of rows as x
`use`	use="pairwise" is the default value and will do pairwise deletion of cases. use="complete" will select just complete cases.
`method`	method="pearson" is the default value. The alternatives to be passed to cor are "spearman" and "kendall"

`adjust` What adjustment for multiple tests should be used? ("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"). See `p.adjust` for details about why to use "holm" rather than "bonferroni").

`alpha` alpha level of confidence intervals

`r` A correlation matrix

`n` Number of observations if using corr.p. May be either a matrix (as returned from corr.test, or a scaler. Set to n- np if finding the significance of partial correlations. (See below).

`ci` By default, confidence intervals are found. However, this leads to a great slow-down of speed. So, for just the rs, ts and ps, set ci=FALSE

Details

corr.test uses the `cor` function to find the correlations, and then applies a t-test to the individual correlations using the formula

$$t = \frac{r * \sqrt(n-2)}{\sqrt(1-r^2)}$$

$$se = \sqrt(\frac{1-r^2}{n-2})$$

The t and Standard Errors are returned as objects in the result, but are not normally displayed. Confidence intervals are found and printed if using the print(short=FALSE) option. These are found by using the fisher z transform of the correlation, and the standard error of the z transforms is

$$se = \sqrt(\frac{1}{n-3})$$

.

The probability values may be adjusted using the Holm (or other) correction. If the matrix is symmetric (no y data), then the original p values are reported below the diagonal and the adjusted above the diagonal. Otherwise, all probabilities are adjusted (unless adjust="none"). This is made explicit in the output.

`corr.p` may be applied to the results of `partial.r` if n is set to n - s (where s is the number of variables partialed out) Fisher, 1924.

Value

`r` The matrix of correlations

`n` Number of cases per correlation

`t` value of t-test for each correlation

`p` two tailed probability of t for each correlation. For symmetric matrices, p values adjusted for multiple tests are reported above the diagonal.

`se` standard error of the correlation

`ci` the alpha/2 lower and upper values

Note

For very large matrices (> 200 x 200), there is a noticeable speed improvement if confidence intervals are not found.

See Also

cor.test for tests of a single correlation, Hmisc::rcorr for an equivalant function, r.test to test the difference between correlations, and cortest.mat to test for equality of two correlation matrices.

Also see cor.ci for bootstrapped confidence intervals of Pearson, Spearman, Kendall, tetrachoric or polychoric correlations. In addition cor.ci will find bootstrapped estimates of composite scales based upon a set of correlations (ala cluster.cor).

In particular, see p.adjust for a discussion of p values associated with multiple tests.

Other useful functions related to finding and displaying correlations include lowerCor for finding the correlations and then displaying the lower off diagonal using the lowerMat function. lowerUpper to compare two correlation matrices.

Examples

```
ct <- corr.test(attitude)  #find the correlations and give the probabilities
ct #show the results
corr.test(attitude[1:3],attitude[4:6]) #reports all values corrected for multiple tests

#corr.test(sat.act[1:3],sat.act[4:6],adjust="none")  #don't adjust the probabilities

#take correlations and show the probabilities as well as the confidence intervals
print(corr.p(cor(attitude[1:4]),30),short=FALSE)

#don't adjust the probabilities
print(corr.test(sat.act[1:3],sat.act[4:6],adjust="none"),short=FALSE)
```

---

correct.cor *Find dis-attenuated correlations given correlations and reliabilities*

Description

Given a raw correlation matrix and a vector of reliabilities, report the disattenuated correlations above the diagonal.

Usage

```
correct.cor(x, y)
```

Arguments

x	A raw correlation matrix
y	Vector of reliabilities

Details

Disattenuated correlations may be thought of as correlations between the latent variables measured by a set of observed variables. That is, what would the correlation be between two (unreliable) variables be if both variables were measured perfectly reliably.

This function is mainly used if importing correlations and reliabilities from somewhere else. If the raw data are available, use `score.items`, or `cluster.loadings` or `cluster.cor`.

Examples of the output of this function are seen in `cluster.loadings` and `cluster.cor`

Value

Raw correlations below the diagonal, reliabilities on the diagonal, disattenuated above the diagonal.

Author(s)

Maintainer: William Revelle `<revelle@northwestern.edu>`

References

http://personality-project.org/revelle/syllabi/405.syllabus.html

See Also

`cluster.loadings` and `cluster.cor`

Examples

```
# attitude from the datasets package
#example 1 is a rather clunky way of doing things

a1 <- attitude[,c(1:3)]
a2 <- attitude[,c(4:7)]
x1 <- rowSums(a1)  #find the sum of the first 3 attitudes
x2 <- rowSums(a2)   #find the sum of the last 4 attitudes
alpha1 <- alpha(a1)
alpha2 <- alpha(a2)
x <- matrix(c(x1,x2),ncol=2)
x.cor <- cor(x)
alpha <- c(alpha1$total$raw_alpha,alpha2$total$raw_alpha)
round(correct.cor(x.cor,alpha),2)
#
#much better - although uses standardized alpha
clusters <- matrix(c(rep(1,3),rep(0,7),rep(1,4)),ncol=2)
cluster.loadings(clusters,cor(attitude))
# or
clusters <- matrix(c(rep(1,3),rep(0,7),rep(1,4)),ncol=2)
cluster.cor(clusters,cor(attitude))
#
#best
scores <- score.items(matrix(c(rep(1,3),rep(0,7),rep(1,4)),ncol=2),attitude)
```

```
scores$corrected
```

`cortest.bartlett` *Bartlett's test that a correlation matrix is an identity matrix*

Description

Bartlett (1951) proposed that -ln(det(R)*(N-1 - (2p+5)/6) was distributed as chi square if R were an identity matrix. A useful test that residuals correlations are all zero.

Usage

```
cortest.bartlett(R, n = NULL,diag=TRUE)
```

Arguments

`R`	A correlation matrix. (If R is not square, correlations are found and a warning is issued.
`n`	Sample size (if not specified, 100 is assumed).
`diag`	Will replace the diagonal of the matrix with 1s to make it a correlation matrix.

Details

More useful for pedagogical purposes than actual applications. The Bartlett test is asymptotically chi square distributed.

Note that if applied to residuals from factor analysis (`fa`) or principal components analysis (`principal`) that the diagonal must be replaced with 1s. This is done automatically if diag=TRUE. (See examples.)

Value

`chisq`	Assymptotically chisquare
`p.value`	Of chi square
`df`	The degrees of freedom

Author(s)

William Revelle

References

Bartlett, M. S., (1951), The Effect of Standardization on a chi square Approximation in Factor Analysis, Biometrika, 38, 337-344.

See Also

cortest.mat, cortest.normal, cortest.jennrich

Examples

```
set.seed(42)
x <- matrix(rnorm(1000),ncol=10)
r <- cor(x)
cortest.bartlett(r)      #random data don't differ from an identity matrix
data(bfi)
cortest.bartlett(bfi[1:200,1:10])    #not an identity matrix
f3 <- fa(Thurstone,3)
f3r <- f3$resid
cortest.bartlett(f3r,n=213,diag=FALSE)  #incorrect

cortest.bartlett(f3r,n=213,diag=TRUE)  #correct (by default)
```

---

cortest.mat

Chi square tests of whether a single matrix is an identity matrix, or a pair of matrices are equal.

Description

Steiger (1980) pointed out that the sum of the squared elements of a correlation matrix, or the Fisher z score equivalents, is distributed as chi square under the null hypothesis that the values are zero (i.e., elements of the identity matrix). This is particularly useful for examining whether correlations in a single matrix differ from zero or for comparing two matrices. Jennrich (1970) also examined tests of differences between matrices.

Usage

```
cortest.normal(R1, R2 = NULL, n1 = NULL, n2 = NULL, fisher = TRUE) #the steiger test
cortest(R1,R2=NULL,n1=NULL,n2 = NULL, fisher = TRUE,cor=TRUE) #same as cortest.normal
cortest.jennrich(R1,R2,n1=NULL, n2=NULL)  #the Jennrich test
cortest.mat(R1,R2=NULL,n1=NULL,n2 = NULL) #an alternative test
```

Arguments

R1	A correlation matrix. (If R1 is not rectangular, and cor=TRUE, the correlations are found).
R2	A correlation matrix. If R2 is not rectangular, and cor=TRUE, the correlations are found. If R2 is NULL, then the test is just whether R1 is an identity matrix.
n1	Sample size of R1
n2	Sample size of R2
fisher	Fisher z transform the correlations?

cor	By default, if the input matrices are not symmetric, they are converted to correlation matrices. That is, they are treated as if they were the raw data. If cor=FALSE, then the input matrices are taken to be correlation matrices.

Details

There are several ways to test if a matrix is the identity matrix. The most well known is the chi square test of Bartlett (1951) and Box (1949). A very straightforward test, discussed by Steiger (1980) is to find the sum of the squared correlations or the sum of the squared Fisher transformed correlations. Under the null hypothesis that all the correlations are equal, this sum is distributed as chi square. This is implemented in `cortest` and `cortest.normal`

Yet another test, is the Jennrich(1970) test of the equality of two matrices. This compares the differences between two matrices to the averages of two matrices using a chi square test. This is implemented in `cortest.jennrich`.

Yet another option `cortest.mat` is to compare the two matrices using an approach analogous to that used in evaluating the adequacy of a factor model. In factor analysis, the maximum likelihood fit statistic is
$f = log(trace((FF' + U2)^{-1}R) - log(|(FF' + U2)^{-1}R|) - n.items.$

This in turn is converted to a chi square

$\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$ (see `fa`.)

That is, the model (M = FF' + U2) is compared to the original correlation matrix (R) by a function of $M^{-1}R$. By analogy, in the case of two matrices, A and B, `cortest.mat` finds the chi squares associated with $A^{-1}B$ and AB^{-1}. The sum of these two χ^2 will also be a χ^2 but with twice the degrees of freedom.

Value

chi2	The chi square statistic
df	Degrees of freedom for the Chi Square
prob	The probability of observing the Chi Square under the null hypothesis.

Note

Both the cortest.jennrich and cortest.normal are probably overly stringent. The ChiSquare values for pairs of random samples from the same population are larger than would be expected. This is a good test for rejecting the null of no differences.

Author(s)

William Revelle

References

Steiger, James H. (1980) Testing pattern hypotheses on correlation matrices: alternative statistics and some empirical results. Multivariate Behavioral Research, 15, 335-352.

Jennrich, Robert I. (1970) An Asymptotic χ^2 Test for the Equality of Two Correlation Matrices. Journal of the American Statistical Association, 65, 904-912.

See Also

cortest.bartlett

Examples

```
x <- matrix(rnorm(1000),ncol=10)
cortest.normal(x)  #just test if this matrix is an identity
x <- sim.congeneric(loads =c(.9,.8,.7,.6,.5),N=1000,short=FALSE)
y <- sim.congeneric(loads =c(.9,.8,.7,.6,.5),N=1000,short=FALSE)
cortest.normal(x$r,y$r,n1=1000,n2=1000) #The Steiger test
cortest.jennrich(x$r,y$r,n1=100,n2=1000) # The Jennrich test
cortest.mat(x$r,y$r,n1=1000,n2=1000)   #twice the degrees of freedom as the Jennrich
```

---

cosinor *Functions for analysis of circadian or diurnal data*

Description

Circadian data are periodic with a phase of 24 hours. These functions find the best fitting phase angle (cosinor), the circular mean, circular correlation with circadian data, and the linear by circular correlation

Usage

```
cosinor(angle,x=NULL,code=NULL,data=NULL,hours=TRUE,period=24,
            plot=FALSE,opti=FALSE,na.rm=TRUE)
cosinor.plot(angle,x=NULL,data = NULL, IDloc=NULL, ID=NULL,hours=TRUE, period=24,
 na.rm=TRUE,ylim=NULL,ylab="observed",xlab="Time (double plotted)",
 main="Cosine fit",add=FALSE,multi=FALSE,typ="l",...)
 cosinor.period(angle,x=NULL,code=NULL,data=NULL,hours=TRUE,period=seq(23,26,1),
            plot=FALSE,opti=FALSE,na.rm=TRUE)
circadian.phase(angle,x=NULL,code=NULL,data=NULL,hours=TRUE,period=24,
            plot=FALSE,opti=FALSE,na.rm=TRUE)
circadian.mean(angle,data=NULL, hours=TRUE,na.rm=TRUE)
circadian.sd(angle,data=NULL,hours=TRUE,na.rm=TRUE)
circadian.stats(angle,data=NULL,hours=TRUE,na.rm=TRUE)
circadian.F(angle,group,data=NULL,hours=TRUE,na.rm=TRUE)
circadian.reliability(angle,x=NULL,code=NULL,data = NULL,min=16,
         oddeven=FALSE, hours=TRUE,period=24,plot=FALSE,opti=FALSE,na.rm=TRUE)
circular.mean(angle,na.rm=TRUE) #angles in radians
circadian.cor(angle,data=NULL,hours=TRUE,na.rm=TRUE)  #angles in radians
circular.cor(angle,na.rm=TRUE) #angles in radians
circadian.linear.cor(angle,x=NULL,data=NULL,hours=TRUE)
```

Arguments

angle	A data frame or matrix of observed values with the time of day as the first value (unless specified in code) angle can be specified either as hours or as radians)
code	A subject identification variable
data	A matrix or data frame of data. If specified, then angle and code are variable names (or locations). See examples.
group	If doing comparisons by groups, specify the group code.
min	The minimum number of observations per subject to use when finding split half reliabilities.
oddeven	Reliabilities are based upon odd and even items (TRUE) or first vs. last half (FALSE). Default is first and last half.
period	Although time of day is assumed to have a 24 hour rhythm, other rhythms may be fit. If calling cosinor.period, a range may be specified.
IDloc	Which column number is the ID field
ID	What specific subject number should be plotted for one variable
plot	if TRUE, then plot the first variable (angle)
opti	opti=TRUE: iterative optimization (slow) or opti=FALSE: linear fitting (fast)
hours	If TRUE, measures are in 24 hours to the day, otherwise, radians
x	A set of external variables to correlate with the phase angles
na.rm	Should missing data be removed?
ylim	Specify the range of the y axis if the defaults don't work
ylab	The label of the yaxis
xlab	Labels for the x axis
main	the title of the graphic
add	If doing multiple (spagetti) plots, set add = TRUE for the second and beyond plots
multi	If doing multiple (spagetti) plots, set multi=TRUE for the first and subsequent plots
typ	Pass the line type to graphics
...	any other graphic parameters to pass

Details

When data represent angles (such as the hours of peak alertness or peak tension during the day), we need to apply circular statistics rather than the more normal linear statistics (see Jammalamadaka (2006) for a very clear set of examples of circular statistics). The generalization of the mean to circular data is to convert each angle into a vector, average the x and y coordinates, and convert the result back to an angle. A statistic that represents the compactness of the observations is R which is the (normalized) vector length found by adding all of the observations together. This will achieve a maximum value (1) when all the phase angles are the same and a minimum (0) if the phase angles are distributed uniformly around the clock.

The generalization of Pearson correlation to circular statistics is straight forward and is implemented in cor.circular in the circular package and in circadian.cor here. Just as the Pearson r is a ratio of covariance to the square root of the product of two variances, so is the circular correlation. The circular covariance of two circular vectors is defined as the average product of the sines of the deviations from the circular mean. The variance is thus the average squared sine of the angular deviations from the circular mean. Circular statistics are used for data that vary over a period (e.g., one day) or over directions (e.g., wind direction or bird flight). Jammalamadaka and Lund (2006) give a very good example of the use of circular statistics in calculating wind speed and direction.

The code from CircStats and circular was adapted to allow for analysis of data from various studies of mood over the day. Those two packages do not seem to handle missing data, nor do they take matrix input, but rather emphasize single vectors.

The cosinor function will either iteratively fit cosines of the angle to the observed data (opti=TRUE) or use the circular by linear regression to estimate the best fitting phase angle. If cos.t <- cos(time) and sin.t = sin(time) (expressed in hours), then beta.c and beta.s may be found by regression and the phase is $sign(beta.c) * acos(beta.c/\sqrt(beta.c^2 + beta.s^2)) * 12/pi$

Simulations (see examples) suggest that with incomplete times, perhaps the optimization procedure yields slightly better fits with the correct phase than does the linear model, but the differences are very small. In the presence of noisey data, these advantages seem to reverse. The recommendation thus seems to be to use the linear model approach (the default). The fit statistic reported for cosinor is the correlation of the data with the model [cos(time - acrophase)].

The circadian.reliability function splits the data for each subject into a first and second half (by default, or into odd and even items) and then finds the best fitting phase for each half. These are then correlated (using circadian.cor) and this correlation is then adjusted for test length using the conventional Spearman-Brown formula. Returned as object in the output are the statistics for the first and second part, as well as an ANOVA to compare the two halves.

circular.mean and circular.cor are just circadian.mean and circadian.cor but with input given in radians rather than hours.

The circadian.linear.cor function will correlate a set of circular variables with a set of linear variables. The first (angle) variables are circular, the second (x) set of variables are linear.

The circadian.F will compare 2 or more groups in terms of their mean position. This is adapted from the equivalent function in the circular pacakge. This is clearly a more powerful test the more each group is compact around its mean (large values of R).

Value

phase	The phase angle that best fits the data (expressed in hours if hours=TRUE).
fit	Value of the correlation of the fit. This is just the correlation of the data with the phase adjusted cosine.
mean.angle	A vector of mean angles
n,mean,sd	The appropriate circular statistic.
correl	A matrix of circular correlations or linear by circular correlations
R	R is the vector length (0-1) of the mean vector when finding circadian statistics using circadian.stats
z,p	z is the number of observations x R^2. p is the probability of a z.

phase.rel	The reliability of the phase measures. This is the circular correlation between the two halves adjusted using the Spearman-Brown correction.
fit.rel	The split half reliability of the fit statistic.
split.F	Do the two halves differ from each other? One would hope not.
group1,group2	The statistics from each half
splits	The individual data from each half.

Note

These functions have been adapted from the circular package to allow for ease of use with circadian data, particularly for data sets with missing data and multiple variables of interest.

Author(s)

William Revelle

References

See circular statistics Jammalamadaka, Sreenivasa and Lund, Ulric (2006),The effect of wind direction on ozone levels: a case study, Environmental and Ecological Statistics, 13, 287-298.

See Also

See the circular and CircStats packages.

Examples

```
time <- seq(1:24) #create a 24 hour time
pure <- matrix(time,24,18)
colnames(pure) <- paste0("H",1:18)
pure <- data.frame(time,cos((pure - col(pure))*pi/12)*3 + 3)
    #18 different phases but scaled to 0-6  match mood data
matplot(pure[-1],type="l",main="Pure circadian arousal rhythms",
    xlab="time of day",ylab="Arousal")
op <- par(mfrow=c(2,2))
 cosinor.plot(1,3,pure)
 cosinor.plot(1,5,pure)
 cosinor.plot(1,8,pure)
 cosinor.plot(1,12,pure)

p <- cosinor(pure) #find the acrophases (should match the input)

#now, test finding the acrophases for  different subjects on 3 variables
#They should be the first 3, second 3, etc. acrophases of pure
pp <- matrix(NA,nrow=6*24,ncol=4)
pure <- as.matrix(pure)
pp[,1] <- rep(pure[,1],6)
pp[1:24,2:4] <- pure[1:24,2:4]
pp[25:48,2:4] <- pure[1:24,5:7] *2   #to test different variances
pp[49:72,2:4] <- pure[1:24,8:10] *3
```

```
pp[73:96,2:4] <- pure[1:24,11:13]
pp[97:120,2:4] <- pure[1:24,14:16]
pp[121:144,2:4] <- pure[1:24,17:19]
pure.df <- data.frame(ID = rep(1:6,each=24),pp)
colnames(pure.df) <- c("ID","Time",paste0("V",1:3))
cosinor("Time",3:5,"ID",pure.df)

op <- par(mfrow=c(2,2))
 cosinor.plot(2,3,pure.df,IDloc=1,ID="1")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="2")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="3")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="4")

 #now, show those in one panel as spagetti plots
op <- par(mfrow=c(1,1))
cosinor.plot(2,3,pure.df,IDloc=1,ID="1",multi=TRUE,ylim=c(0,20),ylab="Modeled")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="2",multi=TRUE,add=TRUE,lty="dotdash")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="3",multi=TRUE,add=TRUE,lty="dashed")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="4",multi=TRUE,add=TRUE,lty="dotted")

set.seed(42)   #what else?
noisy <- pure
noisy[,2:19]<- noisy[,2:19] + rnorm(24*18,0,.2)

n <- cosinor(time,noisy) #add a bit of noise

small.pure <- pure[c(8,11,14,17,20,23),]
small.noisy <- noisy[c(8,11,14,17,20,23),]
small.time <- c(8,11,14,17,20,23)

cosinor.plot(1,3,small.pure,multi=TRUE)
cosinor.plot(1,3,small.noisy,multi=TRUE,add=TRUE,lty="dashed")


# sp <- cosinor(small.pure)
# spo <- cosinor(small.pure,opti=TRUE) #iterative fit
# sn <- cosinor(small.noisy) #linear
# sno <- cosinor(small.noisy,opti=TRUE) #iterative
# sum.df <- data.frame(pure=p,noisy = n, small=sp,small.noise = sn,
#         small.opt=spo,small.noise.opt=sno)
# round(sum.df,2)
# round(circadian.cor(sum.df[,c(1,3,5,7,9,11)]),2)  #compare alternatives
#
# #now, lets form three "subjects" and show how the grouping variable works
# mixed.df <- rbind(small.pure,small.noisy,noisy)
# mixed.df <- data.frame(ID=c(rep(1,6),rep(2,6),rep(3,24)),
#           time=c(rep(c(8,11,14,17,20,23),2),1:24),mixed.df)
# group.df <- cosinor(angle="time",x=2:20,code="ID",data=mixed.df)
# round(group.df,2)  #compare these values to the sp,sn,and n values done separately
```

count.pairwise *Count number of pairwise cases for a data set with missing (NA) data.*

Description

When doing cor(x, use= "pairwise"), it is nice to know the number of cases for each pairwise correlation. This is particularly useful when doing SAPA type analyses.

Usage

```
count.pairwise(x, y = NULL,diagonal=TRUE)
pairwiseDescribe(x,diagonal=FALSE)
```

Arguments

x	An input matrix, typically a data matrix ready to be correlated.
y	An optional second input matrix
diagonal	if TRUE, then report the diagonal, else fill the diagonals with NA

Value

result = matrix of counts of pairwise observations

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

Examples

```
## Not run:
x <- matrix(rnorm(1000),ncol=6)
y <- matrix(rnorm(500),ncol=3)
x[x < 0] <- NA
y[y > 1] <- NA

count.pairwise(x)
count.pairwise(y)
count.pairwise(x,y)
count.pairwise(x,diagonal=FALSE)
pairwiseDescribe(x)

## End(Not run)
```

`cta` *Simulate the C(ues) T(endency) A(ction) model of motivation*

Description

Dynamic motivational models such as the Dynamics of Action (Atkinson and Birch, 1970, Revelle, 1986) may be reparameterized as a simple pair of differential (matrix) equations (Revelle, 1986, 2008). This function simulates the dynamic aspects of the CTA. The CTA model is discussed in detail in Revelle and Condon (2015).

Usage

```
cta (n=3,t=5000, cues = NULL, act=NULL, inhibit=NULL,expect = NULL, consume = NULL,
tendency = NULL,tstrength=NULL, type="both", fast=2,compare=FALSE,learn=TRUE,reward=NULL)
cta.15(n = 3, t = 5000, cues = NULL, act = NULL, inhibit = NULL, consume = NULL,
   ten = NULL,  type = "both", fast = 2)
```

Arguments

`n`	number of actions to simuate
`t`	length of time to simulate
`cues`	a vector of cue strengths
`act`	matrix of associations between cues and action tendencies
`inhibit`	inhibition matrix
`consume`	Consummation matrix
`ten`	Initial values of action tendencies
`type`	show actions, tendencies, both, or state diagrams
`fast`	display every fast time (skips
`expect`	A matrix of expectations
`tendency`	starting values of tendencies
`tstrength`	a vector of starting value of tendencies
`compare`	Allows a two x two graph to compare two plots
`learn`	Allow the system to learn (self reinforce) over time
`reward`	The strength of the reward for doing an action

Details

A very thorough discussion of the CTA model is available from Revelle (2008). An application of the model is discussed in Revelle and Condon (2015).

`cta.15` is the version used to produce the figures and analysis in Revelle and Condon (2015). `cta` is the most recent version and includes a learning function developed in collaboration with Luke Smillie at the University of Melbourne.

The dynamics of action (Atkinson and Birch, 1970) was a model of how instigating forces elicited action tendencies which in turn elicited actions. The basic concept was that action tendencies had inertia. That is, a wish (action tendency) would persist until satisfied and would not change without an instigating force. The consummatory strength of doing an action was thought in turn to reduce the action tendency. Forces could either be instigating or inhibitory (leading to "negaction").

Perhaps the simplest example is the action tendency (T) to eat a pizza. The instigating forces (F) to eat the pizza include the smell and look of the pizza, and once eating it, the flavor and texture. However, if eating the pizza, there is also a consummatory force (C) which was thought to reflect both the strength (gusto) of eating the pizza as well as some constant consummatory value of the activity (c). If not eating the pizza, but in a pizza parlor, the smells and visual cues combine to increase the tendency to eat the pizza. Once eating it, however, the consummatory effect is no longer zero, and the change in action tendency will be a function of both the instigating forces and the consummatory forces. These will achieve a balance when instigating forces are equal to the consummatory forces. The asymptotic strength of eating the pizza reflects this balance and does not require a “set point" or “comparator".

To avoid the problems of instigating and consummatory lags and the need for a decision mechanism, it is possible to reparameterize the original DOA model in terms of action tendencies and actions (Revelle, 1986). Rather than specifying inertia for action tendencies and a choice rule of always expressing the dominant action tendency, it is useful to distinguish between action tendencies (t) and the actions (a) themselves and to have actions as well as tendencies having inertial properties. By separating tendencies from actions, and giving them both inertial properties, we avoid the necessity of a lag parameter, and by making the decision rule one of mutual inhibition, the process is perhaps easier to understand. In an environment which affords cues for action (c), cues enhance action tendencies (t) which in turn strengthen actions (a). This leads to two differential equations, one describing the growth and decay of action tendencies (t), the other of the actions themselves (a).

$$dt = Sc - Ca$$

and

$$da = Et - Ia$$

. (See Revelle and Condon (2015) for an extensive discussion of this model.)

`cta` simulates this model, with the addition of a learning parameter such that activities strengthen the connection between cues and tendencies. The learning part of the cta model is still under development. `cta.15` represents the state of the cta model as described in the Revelle and Condon (2015) article.

Value

graphical output unless type="none"

`cues`	echo back the cue input

inhibition	echo back the inhibitory matrix
time	time spent in each activity
frequency	Frequency of each activity
tendencies	average tendency strengths
actions	average action strength

Author(s)

William Revelle

References

Atkinson, John W. and Birch, David (1970) The dynamics of action. John Wiley, New York, N.Y.

Revelle, William (1986) Motivation and efficiency of cognitive performance in Brown, Donald R. and Veroff, Joe (ed). Frontiers of Motivational Psychology: Essays in honor of J. W. Atkinson. Springer. (Available as a pdf at http://personality-project.org/revelle/publications/dynamicsofmotivation.pdf.)

Revelle, W. (2008) Cues, Tendencies and Actions. The Dynamics of Action revisted. http://personality-project.org/revelle/publications/cta.pdf

Revelle, W. and Condon, D. (2015) A model for personality at three levels. Journal of Research in Personality http://www.sciencedirect.com/science/article/pii/S0092656615000318

Examples

```
#not run
#cta()   #default values, running over time
#cta(type="state") #default values, in a state space  of tendency 1 versus tendency 2
#these next are examples without graphic output
#not run
#two introverts
#c2i <- c(.95,1.05)
#cta(n=2,t=10000,cues=c2i,type="none")
#two extraverts
#c2e <- c(3.95,4.05)
#cta(n=2,t=10000,cues=c2e,type="none")
#three introverts
#c3i <-  c(.95,1,1.05)
#cta(3,t=10000,cues=c3i,type="none")
#three extraverts
#c3i <- c(3.95,4, 4.05)
#cta(3,10000,c3i,type="none")
#mixed
#c3 <- c(1,2.5,4)
#cta(3,10000,c3,type="none")
```

`cubits` *Galton's example of the relationship between height and 'cubit' or forearm length*

Description

Francis Galton introduced the 'co-relation' in 1888 with a paper discussing how to measure the relationship between two variables. His primary example was the relationship between height and forearm length. The data table (cubits) is taken from Galton (1888). Unfortunately, there seem to be some errors in the original data table in that the marginal totals do not match the table.

The data frame, `heights`, is converted from this table.

Usage

```
data(cubits)
```

Format

A data frame with 9 observations on the following 8 variables.

`16.5` Cubit length < 16.5

`16.75` 16.5 <= Cubit length < 17.0

`17.25` 17.0 <= Cubit length < 17.5

`17.75` 17.5 <= Cubit length < 18.0

`18.25` 18.0 <= Cubit length < 18.5

`18.75` 18.5 <= Cubit length < 19.0

`19.25` 19.0 <= Cubit length < 19.5

`19.75` 19.5 <= Cubit length

Details

Sir Francis Galton (1888) published the first demonstration of the correlation coefficient. The regression (or reversion to mediocrity) of the height to the length of the left forearm (a cubit) was found to .8. There seem to be some errors in the table as published in that the row sums do not agree with the actual row sums. These data are used to create a matrix using `table2matrix` for demonstrations of analysis and displays of the data.

Source

Galton (1888)

References

Galton, Francis (1888) Co-relations and their measurement. Proceedings of the Royal Society. London Series,45,135-145,

See Also

table2matrix, table2df, ellipses, heights, peas,galton

Examples

```
data(cubits)
cubits
heights <- table2df(cubits,labs = c("height","cubit"))
ellipses(heights,n=1,main="Galton's co-relation data set")
ellipses(jitter(heights$height,3),jitter(heights$cubit,3),pch=".",
     main="Galton's co-relation data set",xlab="height",
     ylab="Forearm (cubit)") #add in some noise to see the points
pairs.panels(heights,jiggle=TRUE,main="Galton's cubits data set")
```

---

cushny *A data set from Cushny and Peebles (1905) on the effect of three drugs on hours of sleep, used by Student (1908)*

Description

The classic data set used by Gossett (publishing as Student) for the introduction of the t-test. The design was a within subjects study with hours of sleep in a control condition compared to those in 3 drug conditions. Drug1 was 06mg of L Hscyamine, Drug 2L and Drug2R were said to be .6 mg of Left and Right isomers of Hyoscine. As discussed by Zabell (2008) these were not optical isomers. The detal1, delta2L and delta2R are changes from the baseline control.

Usage

```
data(cushny)
```

Format

A data frame with 10 observations on the following 7 variables.

`Control` Hours of sleep in a control condition

`drug1` Hours of sleep in Drug condition 1

`drug2L` Hours of sleep in Drug condition 2

`drug2R` Hours of sleep in Drug condition 3 (an isomer of the drug in condition 2

`delta1` Change from control, drug 1

`delta2L` Change from control, drug 2L

`delta2R` Change from control, drug 2R

Details

The original analysis by Student is used as an example for the t-test function, both as a paired t-test and a two group t-test. The data are also useful for a repeated measures analysis of variance.

Source

Cushny, A.R. and Peebles, A.R. (1905) The action of optical isomers: II hyoscines. The Journal of Physiology 32, 501-510.

Student (1908) The probable error of the mean. Biometrika, 6 (1) , 1-25.

References

See also the data set sleep and the examples for the t.test

S. L. Zabell. On Student's 1908 Article "The Probable Error of a Mean" Journal of the American Statistical Association, Vol. 103, No. 481 (Mar., 2008), pp. 1- 20

Examples

```
data(cushny)
with(cushny, t.test(drug1,drug2L,paired=TRUE)) #within subjects

error.bars(cushny[1:4],within=TRUE,ylab="Hours of sleep",xlab="Drug condition",
      main="95% confidence of within subject effects")
```

---

densityBy *Create a 'violin plot' or density plot of the distribution of a set of variables*

Description

Among the many ways to describe a data set, one is density plot or violin plot of the data. This is similar to a box plot but shows the actual distribution. Median and 25th and 75th percentile lines are added to the display. If a grouping variable is specified, densityBy will draw violin plots for each variable and for each group.

Usage

```
densityBy(x,grp=NULL,grp.name=NULL,ylab="Observed",xlab="",main="Density plot",density=20,
         restrict=TRUE,xlim=NULL,add=FALSE,col=NULL,pch=20, ...)
violinBy(x,grp=NULL,grp.name=NULL,ylab="Observed",xlab="",main="Density plot",density=20,
         restrict=TRUE,xlim=NULL,add=FALSE,col=NULL,pch=20, ...)
```

Arguments

x	A matrix or data.frame
grp	A grouping variable
grp.name	If the grouping variable is specified, the what names should be give to the group? Defaults to 1:ngrp
ylab	The y label
xlab	The x label

main	Figure title
density	How many lines per inch to draw
restrict	Restrict the density to the observed max and min of the data
xlim	if not specified, will be .5 beyond the number of variables
add	Allows overplotting
col	Allows for specification of colours. The default for 2 groups is blue and red, for more group levels, rainbows.
pch	The plot character for the mean is by default a small filled circle. To not show the mean, use pch=NA
...	Other graphic parameters

Details

Describe the data using a violin plot. Change density to modify the shading. density=NULL will fill with col. The grp variable may be used to draw separate violin plots for each of multiple groups.

Value

The density plot of the data.

Note

Nothing yet

Author(s)

William Revelle

See Also

describe, describeBy and statsBy for descriptive statistics and error.bars and error.bars.by for graphic displays

Examples

```
densityBy(bfi[1:5])
#not run
#violinBy(bfi[1:5],grp=bfi$gender,grp.name=c("M","F"))
#densityBy(sat.act[5:6],sat.act$education,col=rainbow(6))
```

describe	*Basic descriptive statistics useful for psychometrics*

Description

There are many summary statistics available in R; this function provides the ones most useful for scale construction and item analysis in classic psychometrics. Range is most useful for the first pass in a data set, to check for coding errors.

Usage

```
describe(x, na.rm = TRUE, interp=FALSE,skew = TRUE, ranges = TRUE,trim=.1,
type=3,check=TRUE,fast=NULL,quant=NULL,IQR=FALSE)
describeData(x,head=4,tail=4)
```

Arguments

`x`	A data frame or matrix
`na.rm`	The default is to delete missing data. na.rm=FALSE will delete the case.
`interp`	Should the median be standard or interpolated
`skew`	Should the skew and kurtosis be calculated?
`ranges`	Should the range be calculated?
`trim`	trim=.1 – trim means by dropping the top and bottom trim fraction
`type`	Which estimate of skew and kurtosis should be used? (See details.)
`check`	Should we check for non-numeric variables? Slower but helpful.
`fast`	if TRUE, will do n, means, sds, ranges for an improvement in speed. If NULL, will switch to fast mode for large (ncol * nrow > 10^7) problems, otherwise defaults to fast = FALSE
`quant`	if not NULL, will find the specified quantiles (e.g. quant=c(.25,.75) will find the 25th and 75th percentiles)
`IQR`	If TRUE, show the interquartile range
`head`	show the first 1:head cases for each variable in describeData
`tail`	Show the last nobs-tail cases for each variable in describeData

Details

In basic data analysis it is vital to get basic descriptive statistics. Procedures such as `summary` and hmisc::describe do so. The describe function in the `psych` package is meant to produce the most frequently requested stats in psychometric and psychology studies, and to produce them in an easy to read data.frame. The results from describe can be used in graphics functions (e.g., `error.crosses`).

The range statistics (min, max, range) are most useful for data checking to detect coding errors, and should be found in early analyses of the data.

Although describe will work on data frames as well as matrices, it is important to realize that for data frames, descriptive statistics will be reported only for those variables where this makes sense (i.e., not for alphanumeric data).

If the check option is TRUE, variables that are categorical or logical are converted to numeric and then described. These variables are marked with an * in the row name. This is somewhat slower. Note that in the case of categories or factors, the numerical ordering is not necessarily the one expected. For instance, if education is coded "high school", "some college" , "finished college", then the default coding will lead to these as values of 2, 3, 1. Thus, statistics for those variables marked with * should be interpreted cautiously (if at all).

In a typical study, one might read the data in from the clipboard (`read.clipboard`), show the splom plot of the correlations (`pairs.panels`), and then describe the data.

na.rm=FALSE is equivalent to describe(na.omit(x))

When finding the skew and the kurtosis, there are three different options available. These match the choices available in skewness and kurtosis found in the e1071 package (see Joanes and Gill (1998) for the advantages of each one).

If we define $m_r = [\sum (X - mx)^r]/n$ then

Type 1 finds skewness and kurtosis by $g_1 = m_3/(m_2)^{3/2}$ and $g_2 = m_4/(m_2)^2 - 3$.

Type 2 is $G1 = g1 * \sqrt{n * (n-1)}/(n-2)$ and $G2 = (n-1) * [(n+1)g2 + 6]/((n-2)(n-3))$.

Type 3 is $b1 = [(n-1)/n]^{3/2} m_3/m_2^{3/2}$ and $b2 = [(n-1)/n]^{3/2} m_4/m_2^2)$.

The additional helper function `describeData` just scans the data array and reports on whether the data are all numerical, logical/factorial, or categorical. This is a useful check to run if trying to get descriptive statistics on very large data sets where to improve the speed, the check option is FALSE.

The fast=TRUE option will lead to a speed up of about 50% for larger problems by not finding all of the statistics (see NOTE)

Value

A data.frame of the relevant statistics:
item name
item number
number of valid cases
mean
standard deviation
trimmed mean (with trim defaulting to .1)
median (standard or interpolated
mad: median absolute deviation (from the median)
minimum
maximum
skew
kurtosis
standard error

Note

For very large data sets that are data.frames, describe can be rather slow. Converting the data to a matrix first is recommended. However, if the data are of different types, (factors or logical), this is not possible. If the data set includes columns of character data, it is also not possible. Thus, a quick pass with `describeData` is recommended.

For the greatest speed, at the cost of losing information, do not ask for ranges or for skew and turn off check. This is done automatically if the fast option is TRUE or for large data sets.

Note that by default, fast=NULL. But if the number of cases x number of variables exceeds (ncol * nrow > 10^7), fast will be set to TRUE. This will provide just n, mean, sd, min, max, range, and standard errors. To get all of the statistics (but at a cost of greater time) set fast=FALSE.

The problem seems to be a memory limitation in that the time taken is an accelerating function of nvars * nobs. Thus, for a largish problem (72,000 cases with 1680 variables) which might take 330 seconds, doing it as two sets of 840 variable cuts the time down to 80 seconds.

Author(s)

`http://personality-project.org/revelle.html`

Maintainer: William Revelle `<revelle@northwestern.edu>`

References

Joanes, D.N. and Gill, C.A (1998). Comparing measures of sample skewness and kurtosis. The Statistician, 47, 183-189.

See Also

`describe.by`, `skew`, `kurtosi` `interp.median`, `read.clipboard`. Then, for graphic output, see `error.crosses`, `pairs.panels`, `error.bars`, `error.bars.by` and `densityBy`, or `violinBy`

Examples

```
data(sat.act)
describe(sat.act)

describe(sat.act,skew=FALSE)
describe(sat.act,IQR=TRUE) #show the interquartile Range
describe(sat.act,quant=c(.1,.25,.5,.75,.90) ) #find the 10th, 25th, 50th,
                  #75th and 90th percentiles

describeData(sat.act) #the fast version
```

describeBy *Basic summary statistics by group*

Description

Report basic summary statistics by a grouping variable. Useful if the grouping variable is some experimental variable and data are to be aggregated for plotting. Partly a wrapper for by and `describe`

Usage

```
describeBy(x, group=NULL,mat=FALSE,type=3,digits=15,...)
describe.by(x, group=NULL,mat=FALSE,type=3,...)  # deprecated
```

Arguments

`x`	a data.frame or matrix. See note for statsBy.
`group`	a grouping variable or a list of grouping variables
`mat`	provide a matrix output rather than a list
`type`	Which type of skew and kurtosis should be found
`digits`	When giving matrix output, how many digits should be reported?
`...`	parameters to be passed to describe

Details

To get descriptive statistics for several different grouping variables, make sure that group is a list. In the case of matrix output with multiple grouping variables, the grouping variable values are added to the output.

The type parameter specifies which version of skew and kurtosis should be found. See `describe` for more details.

An alternative function (`statsBy`) returns a list of means, n, and standard deviations for each group. This is particularly useful if finding weighted correlations of group means using `cor.wt`. More importantly, it does a proper within and between group decomposition of the correlation.

Value

A data.frame of the relevant statistics broken down by group:
item name
item number
number of valid cases
mean
standard deviation
median
mad: median absolute deviation (from the median)
minimum
maximum

skew
standard error

Author(s)

William Revelle

See Also

describe, statsBy, densityBy and violinBy as well as error.bars and error.bars.by for other graphical displays.

Examples

```
data(sat.act)
describeBy(sat.act,sat.act$gender) #just one grouping variable
#describeBy(sat.act,list(sat.act$gender,sat.act$education))  #two grouping variables
des.mat <- describeBy(sat.act$age,sat.act$education,mat=TRUE) #matrix (data.frame) output
des.mat <- describeBy(sat.act$age,list(sat.act$education,sat.act$gender),
             mat=TRUE,digits=2)  #matrix output
```

---

df2latex   *Convert a data frame, correlation matrix, or factor analysis output to a LaTeX table*

Description

A set of handy helper functions to convert data frames or matrices to LaTeX tables. Although Sweave is the preferred means of converting R output to LaTeX, it is sometimes useful to go directly from a data.frame or matrix to a LaTeX table. cor2latex will find the correlations and then create a lower (or upper) triangular matrix for latex output. fa2latex will create the latex commands for showing the loadings and factor intercorrelations. As the default option, tables are prepared in an approximation of APA format.

Usage

```
df2latex(x,digits=2,rowlabels=TRUE,apa=TRUE,short.names=TRUE,font.size ="scriptsize",
      big.mark=NULL,drop.na=TRUE, heading="A table from the psych package in R",
   caption="df2latex",label="default", char=FALSE,
    stars=FALSE,silent=FALSE,file=NULL,append=FALSE,cut=0,big=0)
cor2latex(x,use = "pairwise", method="pearson", adjust="holm",stars=FALSE,
       digits=2,rowlabels=TRUE,lower=TRUE,apa=TRUE,short.names=TRUE,
     font.size ="scriptsize",
      heading="A correlation table from the psych package in R.",
```

```
      caption="cor2latex",label="default",silent=FALSE,file=NULL,append=FALSE)
fa2latex(f,digits=2,rowlabels=TRUE,apa=TRUE,short.names=FALSE,cumvar=FALSE,
       cut=0,big=.3,alpha=.05,font.size ="scriptsize",
       heading="A factor analysis table from the psych package in R",
       caption="fa2latex",label="default",silent=FALSE,file=NULL,append=FALSE)
omega2latex(f,digits=2,rowlabels=TRUE,apa=TRUE,short.names=FALSE,cumvar=FALSE,cut=.2,
        font.size ="scriptsize",
        heading="An omega analysis table from the psych package in R",
      caption="omega2latex",label="default",silent=FALSE,file=NULL,append=FALSE)

irt2latex(f,digits=2,rowlabels=TRUE,apa=TRUE,short.names=FALSE,
       font.size ="scriptsize", heading="An IRT factor analysis table from R",
       caption="fa2latex",label="default",silent=FALSE,file=NULL,append=FALSE)
ICC2latex(icc,digits=2,rowlabels=TRUE,apa=TRUE,ci=TRUE,
   font.size ="scriptsize",big.mark=NULL, drop.na=TRUE,
    heading="A table from the psych package in R",
  caption="ICC2latex",label="default",char=FALSE,silent=FALSE,file=NULL,append=FALSE)
```

Arguments

x	A data frame or matrix to convert to LaTeX. If non-square, then correlations will be found prior to printing in cor2latex
digits	Round the output to digits of accuracy. NULL for formatting character data
rowlabels	If TRUE, use the row names from the matrix or data.frame
short.names	Name the columns with abbreviated rownames to save space
apa	If TRUE formats table in APA style
cumvar	For factor analyses, should we show the cumulative variance accounted for?
font.size	e.g., "scriptsize", "tiny" or anyother acceptable LaTeX font size.
heading	The label appearing at the top of the table
caption	The table caption
lower	in cor2latex, just show the lower triangular matrix
f	The object returned from a factor analysis using fa or irt.fa.
label	The label for the table
big.mark	Comma separate numbers large numbers (big.mark=",")
drop.na	Do not print NA values
method	When finding correlations, which method should be used (pearson)
use	use="pairwise" is the default when finding correlations in cor2latex
adjust	If showing probabilities, which adjustment should be used (holm)
stars	Should probability asterixs be displayed in cor2latex (FALSE)
char	char=TRUE allows printing tables with character information, but does not allow for putting in commas into numbers
cut	In omega2latex, df2latex and fa2latex, do not print abs(values) < cut
big	In fa2latex and df2latex boldface those abs(values) > big

alpha	If fa has returned confidence intervals, then what values of loadings should be boldfaced?
icc	Either the output of an ICC, or the data to be analyzed.
ci	Should confidence intervals of the ICC be displayed
silent	If TRUE, do not print any output, just return silently – useful if using Sweave
file	If specified, write the output to this file
append	If file is specified, then should we append (append=TRUE) or just write to the file

Value

A LaTeX table. Note that if showing "stars" for correlations, then one needs to use the siunitx package in LaTex. The entire LaTeX output is also returned invisibly. If using Sweave to create tables, then the silent option should be set to TRUE and the returned object saved as a file. See the last example.

Author(s)

William Revelle with suggestions from Jason French and David Condon and Davide Morselli

See Also

The many LaTeX conversion routines in Hmisc.

Examples

```
df2latex(Thurstone,rowlabels=FALSE,apa=FALSE,short.names=FALSE,
        caption="Thurstone Correlation matrix")
df2latex(Thurstone,heading="Thurstone Correlation matrix in APA style")

df2latex(describe(sat.act)[2:10],short.names=FALSE)
cor2latex(Thurstone)
cor2latex(sat.act,short.names=FALSE)
fa2latex(fa(Thurstone,3),heading="Factor analysis from R in quasi APA style")

#If using Sweave to create a LateX table as a separate file then set silent=TRUE
#e.g.,
#LaTex preamble
#....
#<<print=FALSE,echo=FALSE>>=
#f3 <- fa(Thurstone,3)
#fa2latex(f3,silent=TRUE,file='testoutput.tex')
#@
#
#\input{testoutput.tex}
```

diagram *Helper functions for drawing path model diagrams*

Description

Path models are used to describe structural equation models or cluster analytic output. These functions provide the primitives for drawing path models. Used as a substitute for some of the functionality of Rgraphviz.

Usage

```
diagram(fit,...)
dia.rect(x, y = NULL, labels = NULL,  cex = 1,  xlim = c(0, 1), ylim = c(0, 1), ...)
dia.ellipse(x, y = NULL, labels = NULL, cex=1,e.size=.05, xlim=c(0,1), ylim=c(0,1),  ...)
dia.triangle(x, y = NULL, labels =NULL,  cex = 1, xlim=c(0,1),ylim=c(0,1),...)
dia.ellipse1(x,y,e.size=.05,xlim=c(0,1),ylim=c(0,1),...)
dia.shape(x, y = NULL, labels = NULL, cex = 1,
        e.size=.05, xlim=c(0,1), ylim=c(0,1), shape=1, ...)
dia.arrow(from,to,labels=NULL,scale=1,cex=1,adj=2,both=FALSE,pos=NULL,l.cex,gap.size,...)
dia.curve(from,to,labels=NULL,scale=1,...)
dia.curved.arrow(from,to,labels=NULL,scale=1,both=FALSE,...)
dia.self(location,labels=NULL,scale=.8,side=2,...)
dia.cone(x=0, y=-2, theta=45, arrow=TRUE,curves=TRUE,add=FALSE,labels=NULL,
      xlim = c(-1, 1), ylim=c(-1,1),... )
```

Arguments

fit	The results from a factor analysis fa, components analysis principal, omega reliability analysis, omega, cluster analysis iclust or confirmatory factor analysis, cfa, or structural equation model,sem, using the lavaan package.
x	x coordinate of a rectangle or ellipse
y	y coordinate of a rectangle or ellipse
e.size	The size of the ellipse (scaled by the number of variables
labels	Text to insert in rectangle, ellipse, or arrow
cex	adjust the text size
l.cex	Adjust the text size in arrows, defaults to cex which in turn defaults to 1
gap.size	Tweak the gap in an arrow to be allow the label to be in a gap
adj	Where to put the label along the arrows (values are then divided by 4)
both	Should the arrows have arrow heads on both ends?
scale	modifies size of rectangle and ellipse as well as the curvature of curves. (For curvature, positive numbers are concave down and to the left
from	arrows and curves go from
to	arrows and curves go to

location	where is the rectangle?
shape	Which shape to draw
xlim	default ranges
ylim	default ranges
side	Which side of boxes should errors appear
theta	Angle in degrees of vectors
arrow	draw arrows for edges in dia.cone
add	if TRUE, plot on previous plot
curves	if TRUE, draw curves between arrows in dia.cone
pos	The position of the text in dia.arrow. Follows the text positions of 1, 2, 3, 4 or NULL
...	Most graphic parameters may be passed here

Details

The diagram function calls `fa.diagram`, `omega.diagram`, `ICLUST.diagram` or `lavaan.diagram` depending upon the class of the fit input. See those functions for particular parameter values.

The remaining functions are the graphic primitives used by `fa.diagram`, `structure.diagram`, `omega.diagram`, `ICLUST.diagram` and `het.diagram`

They create rectangles, ellipses or triangles surrounding text, connect them to straight or curved arrows, and can draw an arrow from and to the same rectangle.

Each shape (ellipse, rectangle or triangle) has a left, right, top and bottom and center coordinate that may be used to connect the arrows.

Curves are double-headed arrows.

The helper functions were developed to get around the infelicities associated with trying to install Rgraphviz and graphviz.

These functions form the core of `fa.diagram`,`het.diagram`.

Better documentation will be added as these functions get improved. Currently the helper functions are just a work around for Rgraphviz.

dia.cone draws a cone with (optionally) arrows as sides and centers to show the problem of factor indeterminacy.

Value

Graphic output

Author(s)

William Revelle

See Also

The diagram functions that use the dia functions: `fa.diagram`, `structure.diagram`, `omega.diagram`, and `ICLUST.diagram`.

Examples

```
#first, show the primitives
xlim=c(-2,10)
ylim=c(0,10)
plot(NA,xlim=xlim,ylim=ylim,main="Demonstration of diagram functions",axes=FALSE,xlab="",ylab="")
ul <- dia.rect(1,9,labels="upper left",xlim=xlim,ylim=ylim)
ml <- dia.rect(1,6,"middle left",xlim=xlim,ylim=ylim)
ll <- dia.rect(1,3,labels="lower left",xlim=xlim,ylim=ylim)
bl <- dia.rect(1,1,"bottom left",xlim=xlim,ylim=ylim)
lr <- dia.ellipse(7,3,"lower right",xlim=xlim,ylim=ylim,e.size=.07)
ur <- dia.ellipse(7,9,"upper right",xlim=xlim,ylim=ylim,e.size=.07)
mr <- dia.ellipse(7,6,"middle right",xlim=xlim,ylim=ylim,e.size=.07)
lm <- dia.triangle(4,1,"Lower Middle",xlim=xlim,ylim=ylim)
br <- dia.rect(9,1,"bottom right",xlim=xlim,ylim=ylim)
dia.curve(from=ul$left,to=bl$left,"double headed",scale=-1)

dia.arrow(from=lr,to=ul,labels="right to left")
dia.arrow(from=ul,to=ur,labels="left to right")
dia.curved.arrow(from=lr,to=ll,labels ="right to left")
dia.curved.arrow(to=ur,from=ul,labels ="left to right")
dia.curve(ll$top,ul$bottom,"right")  #for rectangles, specify where to point

dia.curve(ll$top,ul$bottom,"left",scale=-1)  #for rectangles, specify where to point
dia.curve(mr,ur,"up")  #but for ellipses, you may just point to it.
dia.curve(mr,lr,"down")
dia.curve(mr,ur,"up")
dia.curved.arrow(mr,ur,"up")  #but for ellipses, you may just point to it.
dia.curved.arrow(mr,lr,"down")  #but for ellipses, you may just point to it.

dia.curved.arrow(ur$right,mr$right,"3")
dia.curve(ml,mr,"across")
dia.curve(ur,lr,"top down")
dia.curved.arrow(br$top,lr$bottom,"up")
dia.curved.arrow(bl,br,"left to right")
dia.curved.arrow(br,bl,"right to left",scale=-1)
dia.arrow(bl,ll$bottom)
dia.curved.arrow(ml,ll$right)
dia.curved.arrow(mr,lr$top)

#now, put them together in a factor analysis diagram
v9 <- sim.hierarchical()
f3 <- fa(v9,3,rotate="cluster")
fa.diagram(f3,error=TRUE,side=3)
```

---

draw.tetra	*Draw a correlation ellipse and two normal curves to demonstrate tetrachoric correlation*

Description

A graphic of a correlation ellipse divided into 4 regions based upon x and y cutpoints on two normal distributions. This is also an example of using the layout function. Draw a bivariate density plot to show how tetrachorics work.

Usage

```
draw.tetra(r, t1, t2,shade=TRUE)
draw.cor(r=.5,expand=10,theta=30,phi=30,N=101,nbcol=30,box=TRUE,
main="Bivariate density  rho = ",cuts=NULL,all=TRUE,ellipses=TRUE,ze=.15)
```

Arguments

`r`	the underlying Pearson correlation defines the shape of the ellipse
`t1`	X is cut at tau
`t2`	Y is cut at Tau
`shade`	shade the diagram (default is TRUE)
`expand`	The relative height of the z axis
`theta`	The angle to rotate the x-y plane
`phi`	The angle above the plane to view the graph
`N`	The grid resolution
`nbcol`	The color resolution
`box`	Draw the axes
`main`	The main title
`cuts`	Should the graphic show cuts (e.g., cuts=c(0,0))
`all`	Show all four parts of the tetrachoric
`ellipses`	Draw a correlation ellipse
`ze`	height of the ellipse if requested

Details

A graphic demonstration of the `tetrachoric` correlation. Used for teaching purposes. The default values are for a correlation of .5 with cuts at 1 and 1. Any other values are possible. The code is also a demonstration of how to use the `layout` function for complex graphics using base graphics.

Author(s)

William Revelle

See Also

`tetrachoric` to find tetrachoric correlations, `irt.fa` and `fa.poly` to use them in factor analyses, `scatter.hist` to show correlations and histograms.

Examples

```
#if(require(mvtnorm)) {
#draw.tetra(.5,1,1)
#draw.tetra(.8,2,1)} else {print("draw.tetra requires the mvtnorm package")
#draw.cor(.5,cuts=c(0,0))}

draw.tetra(.5,1,1)
draw.tetra(.8,2,1)
draw.cor(.5,cuts=c(0,0))
```

---

dummy.code *Create dummy coded variables*

Description

Given a variable x with n distinct values, create n new dummy coded variables coded 0/1 for presence (1) or absence (0) of each variable. A typical application would be to create dummy coded college majors from a vector of college majors.

Usage

```
dummy.code(x)
```

Arguments

x	A vector to be transformed into dummy codes

Details

When coding demographic information, it is typical to create one variable with multiple categorical values (e.g., ethnicity, college major, occupation). `dummy.code` will convert these categories into n distinct dummy coded variables.

If using dummy coded variables as predictors, remember to use n-1 variables.

Value

A matrix of dummy coded variables

Author(s)

William Revelle

Examples

```
new <- dummy.code(sat.act$education)
new.sat <- data.frame(new,sat.act)
round(cor(new.sat,use="pairwise"),2)
```

`Dwyer` *8 cognitive variables used by Dwyer for an example.*

Description

Dwyer (1937) introduced a technique for factor extension and used 8 cognitive variables from Thurstone. This is the example data set used in his paper.

Usage

```
data(Dwyer)
```

Format

The format is: num [1:8, 1:8] 1 0.58 -0.28 0.01 0.36 0.38 0.61 0.15 0.58 1 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:8] "V1" "V2" "V3" "V4" ... ..$: chr [1:8] "V1" "V2" "V3" "V4" ...

Source

Data matrix retyped from the original publication.

References

Dwyer, Paul S. (1937), The determination of the factor loadings of a given test from the known factor loadings of other tests. Psychometrika, 3, 173-178

Examples

```
data(Dwyer)
Ro <- Dwyer[1:7,1:7]
Roe <- Dwyer[1:7,8]
fo <- fa(Ro,2,rotate="none")
fa.extension(Roe,fo)
```

`eigen.loadings` *Convert eigen vectors and eigen values to the more normal (for psychologists) component loadings*

Description

The default procedures for principal component returns values not immediately equivalent to the loadings from a factor analysis. eigen.loadings translates them into the more typical metric of eigen vectors multiplied by the squareroot of the eigenvalues. This lets us find pseudo factor loadings if we have used princomp or eigen.

If we use `principal` to do our principal components analysis, then we do not need this routine.

Usage

```
eigen.loadings(x)
```

Arguments

x	the output from eigen or a list of class princomp derived from princomp

Value

A matrix of Principal Component loadings more typical for what is expected in psychometrics. That is, they are scaled by the square root of the eigenvalues.

Note

Useful for SAPA analyses

Author(s)

< revelle@northwestern.edu >
http://personality-project.org/revelle.html

Examples

```
x <- eigen(Harman74.cor$cov)
x$vectors[1:8,1:4]  #as they appear from eigen
y <- princomp(covmat=Harman74.cor$cov)
y$loadings[1:8,1:4] #as they appear from princomp
eigen.loadings(x)[1:8,1:4] # rescaled by the eigen values
```

---

ellipses *Plot data and 1 and 2 sigma correlation ellipses*

Description

For teaching correlation, it is useful to draw ellipses around the mean to reflect the correlation. This variation of the ellipse function from John Fox's car package does so. Input may be either two vectors or a matrix or data.frame. In the latter cases, if the number of variables >2, then the ellipses are done in the pairs.panels function. Ellipses may be added to existing plots. The minkowski function is included as a generalized ellipse.

Usage

```
ellipses(x, y = NULL, add = FALSE, smooth=TRUE, lm=FALSE,data=TRUE, n = 2,
 span=2/3, iter=3, col = "red", xlab =NULL,ylab= NULL, ...)
minkowski(r=2,add=FALSE,main=NULL,xl=1,yl=1)
```

Arguments

x	a vector,matrix, or data.frame
y	Optional second vector
add	Should a new plot be created, or should it be added to?
smooth	smooth = TRUE -> draw a loess fit
lm	lm=TRUE -> draw the linear fit
data	data=TRUE implies draw the data points
n	Should 1 or 2 ellipses be drawn
span	averaging window parameter for the lowess fit
iter	iteration parameter for lowess
col	color of ellipses (default is red
xlab	label for the x axis
ylab	label for the y axis
...	Other parameters for plotting
r	r=1 draws a city block, r=2 is a Euclidean circle, r > 2 tends towards a square
main	title to use when drawing Minkowski circles
xl	stretch the x axis
yl	stretch the y axis

Details

Ellipse dimensions are calculated from the correlation between the x and y variables and are scaled as sqrt(1+r) and sqrt(1-r).

Value

A single plot (for 2 vectors or data frames with fewer than 3 variables. Otherwise a call is made to `pairs.panels`.

Note

Adapted from John Fox's ellipse and data.ellipse functions.

Author(s)

William Revelle

References

Galton, Francis (1888), Co-relations and their measurement. Proceedings of the Royal Society. London Series, 45, 135-145.

See Also

`pairs.panels`

Examples

```
data(galton)
ellipses(galton,lm=TRUE)
ellipses(galton$parent,galton$child,xlab="Mid Parent Height",
                  ylab="Child Height") #input are two vectors
data(sat.act)
ellipses(sat.act)  #shows the pairs.panels ellipses
minkowski(2,main="Minkowski circles")
minkowski(1,TRUE)
minkowski(4,TRUE)
```

---

epi *Eysenck Personality Inventory (EPI) data for 3570 participants*

Description

The EPI is and has been a very frequently administered personality test with 57 measuring two broad dimensions, Extraversion-Introversion and Stability-Neuroticism, with an additional Lie scale. Developed by Eysenck and Eysenck, 1964. Eventually replaced with the EPQ which measures three broad dimensions. This data set represents 3570 observations collected in the early 1990s at the Personality, Motivation and Cognition lab at Northwestern. The data are included here as demonstration of scale construction.

Usage

```
data(epi)
data(epi.dictionary)
```

Format

A data frame with 3570 observations on the following 57 variables.

`V1` a numeric vector

`V2` a numeric vector

`V3` a numeric vector

`V4` a numeric vector

`V5` a numeric vector

`V6` a numeric vector

`V7` a numeric vector

`V8` a numeric vector

`V9` a numeric vector

`V10` a numeric vector

`V11` a numeric vector

V12 a numeric vector
V13 a numeric vector
V14 a numeric vector
V15 a numeric vector
V16 a numeric vector
V17 a numeric vector
V18 a numeric vector
V19 a numeric vector
V20 a numeric vector
V21 a numeric vector
V22 a numeric vector
V23 a numeric vector
V24 a numeric vector
V25 a numeric vector
V26 a numeric vector
V27 a numeric vector
V28 a numeric vector
V29 a numeric vector
V30 a numeric vector
V31 a numeric vector
V32 a numeric vector
V33 a numeric vector
V34 a numeric vector
V35 a numeric vector
V36 a numeric vector
V37 a numeric vector
V38 a numeric vector
V39 a numeric vector
V40 a numeric vector
V41 a numeric vector
V42 a numeric vector
V43 a numeric vector
V44 a numeric vector
V45 a numeric vector
V46 a numeric vector
V47 a numeric vector
V48 a numeric vector

V49 a numeric vector

V50 a numeric vector

V51 a numeric vector

V52 a numeric vector

V53 a numeric vector

V54 a numeric vector

V55 a numeric vector

V56 a numeric vector

V57 a numeric vector

Details

The original data were collected in a group testing framework for screening participants for subsequent studies. The participants were enrolled in an introductory psychology class between Fall, 1991 and Spring, 1995.

The structure of the E scale has been shown by Rocklin and Revelle (1981) to have two subcomponents, Impulsivity and Sociability. These were subsequently used by Revelle, Humphreys, Simon and Gilliland to examine the relationship between personality, caffeine induced arousal, and cognitive performance.

Source

Data from the PMC laboratory at Northwestern.

References

Eysenck, H.J. and Eysenck, S. B.G. (1968). Manual for the Eysenck Personality Inventory.Educational and Industrial Testing Service, San Diego, CA.

Rocklin, T. and Revelle, W. (1981). The measurement of extraversion: A comparison of the Eysenck Personality Inventory and the Eysenck Personality Questionnaire. British Journal of Social Psychology, 20(4):279-284.

Examples

```
data(epi)
epi.keys <- make.keys(epi,list(E = c(1, 3, -5, 8, 10, 13, -15, 17, -20, 22, 25, 27,
                -29, -32, -34, -37, 39, -41, 44, 46, 49, -51, 53, 56),
   N=c(2, 4, 7, 9, 11, 14, 16, 19, 21, 23, 26, 28, 31, 33, 35, 38, 40,
     43, 45, 47, 50, 52, 55, 57),
   L = c(6, -12, -18, 24, -30, 36, -42, -48, -54),
   I =c(1, 3, -5, 8, 10, 13, 22, 39, -41),
   S = c(-11, -15, 17, -20, 25, 27, -29, -32, -37, 44, 46, -51, 53)))
scores <- scoreItems(epi.keys,epi)
  N <- epi[abs(epi.keys[,"N"]) >0]
  E <- epi[abs(epi.keys[,"E"]) >0]
  fa.lookup(epi.keys[,1:3],epi.dictionary) #show the items and keying information
```

epi.bfi *13 personality scales from the Eysenck Personality Inventory and Big 5 inventory*

Description

A small data set of 5 scales from the Eysenck Personality Inventory, 5 from a Big 5 inventory, a Beck Depression Inventory, and State and Trait Anxiety measures. Used for demonstrations of correlations, regressions, graphic displays.

Usage

```
data(epi.bfi)
```

Format

A data frame with 231 observations on the following 13 variables.

`epiE` EPI Extraversion

`epiS` EPI Sociability (a subset of Extraversion items

`epiImp` EPI Impulsivity (a subset of Extraversion items

`epilie` EPI Lie scale

`epiNeur` EPI neuroticism

`bfagree` Big 5 inventory (from the IPIP) measure of Agreeableness

`bfcon` Big 5 Conscientiousness

`bfext` Big 5 Extraversion

`bfneur` Big 5 Neuroticism

`bfopen` Big 5 Openness

`bdi` Beck Depression scale

`traitanx` Trait Anxiety

`stateanx` State Anxiety

Details

Self report personality scales tend to measure the “Giant 2" of Extraversion and Neuroticism or the “Big 5" of Extraversion, Neuroticism, Agreeableness, Conscientiousness, and Openness. Here is a small data set from Northwestern University undergraduates with scores on the Eysenck Personality Inventory (EPI) and a Big 5 inventory taken from the International Personality Item Pool.

Source

Data were collected at the Personality, Motivation, and Cognition Lab (PMCLab) at Northwestern by William Revelle)

References

http://personality-project.org/pmc.html

Examples

```
data(epi.bfi)
pairs.panels(epi.bfi[,1:5])
describe(epi.bfi)
```

---

error.bars *Plot means and confidence intervals*

Description

One of the many functions in R to plot means and confidence intervals. Can be done using barplots if desired. Can also be combined with such functions as boxplot to summarize distributions. Means and standard errors are calculated from the raw data using describe. Alternatively, plots of means +/- one standard deviation may be drawn.

Usage

```
error.bars(x,stats=NULL, ylab = "Dependent Variable",xlab="Independent Variable",
  main=NULL,eyes=TRUE, ylim = NULL, xlim=NULL,alpha=.05,sd=FALSE, labels = NULL,
   pos = NULL,  arrow.len = 0.05,arrow.col="black", add = FALSE,bars=FALSE,within=FALSE,
    col="blue",...)
error.bars.tab(t,way="columns",raw=FALSE,col=c('blue','red'),...)
```

Arguments

x	A data frame or matrix of raw data
t	A table of frequencies
stats	Alternatively, a data.frame of descriptive stats from (e.g., describe)
ylab	y label
xlab	x label
main	title for figure
ylim	if specified, the limits for the plot, otherwise based upon the data
xlim	if specified, the x limits for the plot, otherwise c(.5,nvar + .5)
eyes	should 'cats eyes' plots be drawn
alpha	alpha level of confidence interval – defaults to 95% confidence interval
sd	if TRUE, draw one standard deviation instead of standard errors at the alpha level
labels	X axis label
pos	where to place text: below, left, above, right

`arrow.len`	How long should the top of the error bars be?
`arrow.col`	What color should the error bars be?
`add`	add=FALSE, new plot, add=TRUE, just points and error bars
`bars`	bars=TRUE will draw a bar graph if you really want to do that
`within`	should the error variance of a variable be corrected by 1-SMC?
`col`	color(s) of the catseyes. Defaults to blue.
`way`	Percentages are based upon the row totals (default) column totals, or grand total of the data Table
`raw`	If raw is FALSE, display the graphs in terms of probability, raw TRUE displays the data in terms of raw counts
`...`	other parameters to pass to the plot function, e.g., typ="b" to draw lines, lty="dashed" to draw dashed lines

Details

Drawing the mean +/- a confidence interval is a frequently used function when reporting experimental results. By default, the confidence interval is 1.96 standard errors of the t-distribution.

If within=TRUE, the error bars are corrected for the correlation with the other variables by reducing the variance by a factor of (1-smc). This allows for comparisons between variables.

The error bars are normally calculated from the data using the describe function. If, alternatively, a matrix of statistics is provided with column headings of values, means, and se, then those values will be used for the plot (using the stats option). If n is included in the matrix of statistics, then the distribution is drawn for a t distribution for n-1 df. If n is omitted (NULL) or is NA, then the distribution will be a normal distribution.

If sd is TRUE, then the error bars will represent one standard deviation from the mean rather than be a function of alpha and the standard errors.

See the last two examples for the case of plotting data with statistics from another function.

Alternatively, `error.bars.tab` will take tabulated data and convert to either row, column or overall percentages, and then plot these as percentages with the equivalent standard error (based upon sqrt(pq/N)).

Value

Graphic output showing the means + x

These confidence regions are based upon normal theory and do not take into account any skew in the variables. More accurate confidence intervals could be found by resampling.

The error.bars.tab function will return (invisibly) the cell means and standard errors.

Author(s)

William Revelle

See Also

error.crosses for two way error bars, error.bars.by for error bars for different groups

In addition, as pointed out by Jim Lemon on the R-help news group, error bars or confidence intervals may be drawn using

function	package
bar.err	(agricolae)
plotCI	(gplots)
xYplot	(Hmisc)
dispersion	(plotrix)
plotCI	(plotrix)

For advice why not to draw bar graphs with error bars, see http://biostat.mc.vanderbilt.edu/wiki/Main/DynamitePlots

Examples

```
x <- replicate(20,rnorm(50))
boxplot(x,notch=TRUE,main="Notched boxplot with error bars")
error.bars(x,add=TRUE)
abline(h=0)

#show 50% confidence regions and color each variable separately
error.bars(attitude,alpha=.5,
   main="50 percent confidence limits",col=rainbow(ncol(attitude)) )

error.bars(attitude,bar=TRUE)  #show the use of bar graphs


#combine with a strip chart and boxplot
stripchart(attitude,vertical=TRUE,method="jitter",jitter=.1,pch=19,
           main="Stripchart with 95 percent confidence limits")
boxplot(attitude,add=TRUE)
error.bars(attitude,add=TRUE,arrow.len=.2)

#use statistics from somewhere else
#by specifying n, we are using the t distribution for confidences
#The first example allows the variables to be spaced along the x axis
my.stats <- data.frame(values=c(1,2,8),mean=c(10,12,18),se=c(2,3,5),n=c(5,10,20))
 error.bars(stats=my.stats,type="b",main="data with confidence intervals")
#don't connect the groups
my.stats <- data.frame(values=c(1,2,8),mean=c(10,12,18),se=c(2,3,5),n=c(5,10,20))
      error.bars(stats=my.stats,main="data with confidence intervals")
#by not specifying value, the groups are equally spaced
my.stats <- data.frame(mean=c(10,12,18),se=c(2,3,5),n=c(5,10,20))
rownames(my.stats) <- c("First", "Second","Third")
error.bars(stats=my.stats,xlab="Condition",ylab="Score")
```

```
#Consider the case where we get stats from describe
temp <- describe(attitude)
error.bars(stats=temp)

#show these do not differ from the other way by overlaying the two
error.bars(attitude,add=TRUE,col="red")

#n is omitted
#the error distribution is a normal distribution
my.stats <- data.frame(mean=c(2,4,8),se=c(2,1,2))
rownames(my.stats) <- c("First", "Second","Third")
error.bars(stats=my.stats,xlab="Condition",ylab="Score")
#n is specified
#compare this with small n which shows larger confidence regions
my.stats <- data.frame(mean=c(2,4,8),se=c(2,1,2),n=c(10,10,3))
rownames(my.stats) <- c("First", "Second","Third")
error.bars(stats=my.stats,xlab="Condition",ylab="Score")


#example of arrest rates (as percentage of condition)
arrest <- data.frame(Control=c(14,21),Treated =c(3,23))
rownames(arrest) <- c("Arrested","Not Arrested")
error.bars.tab(arrest,ylab="Probability of Arrest",xlab="Control vs Treatment",
main="Probability of Arrest varies by treatment")


#Show the raw  rates
error.bars.tab(arrest,raw=TRUE,ylab="Number Arrested",xlab="Control vs Treatment",
main="Count of Arrest varies by treatment")
```

`error.bars.by` *Plot means and confidence intervals for multiple groups*

Description

One of the many functions in R to plot means and confidence intervals. Meant mainly for demonstration purposes for showing the probabilty of replication from multiple samples. Can also be combined with such functions as boxplot to summarize distributions. Means and standard errors for each group are calculated using `describe.by`.

Usage

```
error.bars.by(x,group,by.var=FALSE,x.cat=TRUE,ylab =NULL,xlab=NULL,main=NULL,ylim= NULL,
xlim=NULL, eyes=TRUE,alpha=.05,sd=FALSE,labels=NULL, v.labels=NULL, pos=NULL,
arrow.len=.05,add=FALSE,bars=FALSE,within=FALSE,colors=c("black","blue","red"),
 lty,lines=TRUE, legend=0,pch,density=-10,...)
```

Arguments

x	A data frame or matrix
group	A grouping variable
by.var	A different line for each group (default) or each variable
x.cat	Is the grouping variable categorical (TRUE) or continuous (FALSE
ylab	y label
xlab	x label
main	title for figure
ylim	if specified, the y limits for the plot, otherwise based upon the data
xlim	if specified, the x limits for the plot, otherwise based upon the data
eyes	Should 'cats eyes' be drawn'
alpha	alpha level of confidence interval. Default is 1- alpha =95% confidence interval
sd	sd=TRUE will plot Standard Deviations instead of standard errors
labels	X axis label
v.labels	For a bar plot legend, these are the variable labels
pos	where to place text: below, left, above, right
arrow.len	How long should the top of the error bars be?
add	add=FALSE, new plot, add=TRUE, just points and error bars
bars	Draw a barplot with error bars rather than a simple plot of the means
within	Should the s.e. be corrected by the correlation with the other variables?
colors	groups will be plotted in different colors (mod n.groups). See the note for how to make them transparent.
lty	line type may be specified in the case of not plotting by variables
lines	By default, when plotting different groups, connect the groups with a line of type = lty. If lines is FALSE, then do not connect the groups
legend	Where should the legend be drawn: 0 (do not draw it), 1= lower right corner, 2 = bottom, 3 ... 8 continue clockwise, 9 is the center
pch	The first plot symbol to use. Subsequent groups are pch + group
density	How many lines/inch should fill the cats eyes. If missing, non-transparent colors are used. If negative, transparent colors are used.
...	other parameters to pass to the plot function e.g., lty="dashed" to draw dashed lines

Details

Drawing the mean +/- a confidence interval is a frequently used function when reporting experimental results. By default, the confidence interval is 1.96 standard errors (adjusted for the t-distribution).

This function was originally just a wrapper for error.bars but has been written to allow groups to be organized either as the x axis or as separate lines.

If desired, a barplot with error bars can be shown. Many find this type of plot to be uninformative (e.g., http://biostat.mc.vanderbilt.edu/DynamitePlots) and recommend the more standard dot plot.

Note in particular, if choosing to draw barplots, the starting value is 0.0 and setting the ylim parameter can lead to some awkward results if 0 is not included in the ylim range. Did you really mean to draw a bar plot in this case?

For up to three groups, the colors are by default "black", "blue" and "red". For more than 3 groups, they are by default rainbow colors with an alpha factor (transparency) of .5.

To make colors semitransparent, set the density to a negative number. See the last example.

Value

Graphic output showing the means + x% confidence intervals for each group. For ci=1.96, and normal data, this will be the 95% confidence region. For ci=1, the 68% confidence region.

These confidence regions are based upon normal theory and do not take into account any skew in the variables. More accurate confidence intervals could be found by resampling.

See Also

See Also as error.crosses, error.bars

Examples

```
data(sat.act)
#The generic plot of variables by group
error.bars.by(sat.act[1:4],sat.act$gender,legend=7)
#a bar plot
error.bars.by(sat.act[5:6],sat.act$gender,bars=TRUE,labels=c("male","female"),
    main="SAT V and SAT Q by gender",ylim=c(0,800),colors=c("red","blue"),
    legend=5,v.labels=c("SATV","SATQ"))  #draw a barplot
#a bar plot of SAT by age -- not recommended, see the next plot
error.bars.by(sat.act[5:6],sat.act$education,bars=TRUE,xlab="Education",
   main="95 percent confidence limits of Sat V and Sat Q", ylim=c(0,800),
   v.labels=c("SATV","SATQ"),legend=5,colors=c("red","blue") )
#a better graph uses points not bars
  #plot SAT V and SAT Q by education
error.bars.by(sat.act[5:6],sat.act$education,TRUE, xlab="Education",
    legend=5,labels=colnames(sat.act[5:6]),ylim=c(525,700),
     main="self reported SAT scores by education")
#make the cats eyes semi-transparent by specifying a negative density
error.bars.by(sat.act[5:6],sat.act$education,TRUE, xlab="Education",
    legend=5,labels=colnames(sat.act[5:6]),ylim=c(525,700),
     main="self reported SAT scores by education",density=-10)

#now for a more complicated examples using 25 big 5 items scored into 5 scales
#and showing age trends by decade
#this shows how to convert many levels of a grouping variable (age) into more manageable levels.
data(bfi)   #The Big 5 data
#first create the keys
 keys.list <- list(Agree=c(-1,2:5),Conscientious=c(6:8,-9,-10),
```

```
     Extraversion=c(-11,-12,13:15),Neuroticism=c(16:20),Openness = c(21,-22,23,24,-25))
keys <- make.keys(bfi,keys.list)
#then create the scores for those older than 10 and less than 80
bfis <- subset(bfi,((bfi$age > 10) & (bfi$age < 80)))

scores <- scoreItems(keys,bfis,min=1,max=6) #set the right limits for item reversals
#now draw the results by age

error.bars.by(scores$scores,round(bfis$age/10)*10,by.var=TRUE,
     main="BFI age trends",legend=3,labels=colnames(scores$scores),
       xlab="Age",ylab="Mean item score")

error.bars.by(scores$scores,round(bfis$age/10)*10,by.var=TRUE,
     main="BFI age trends",legend=3,labels=colnames(scores$scores),
       xlab="Age",ylab="Mean item score",density=-10)
```

`error.crosses` *Plot x and y error bars*

Description

Given two vectors of data (X and Y), plot the means and show standard errors in both X and Y directions.

Usage

```
error.crosses(x,y,labels=NULL,main=NULL,xlim=NULL,ylim= NULL,
xlab=NULL,ylab=NULL,pos=NULL,offset=1,arrow.len=.2,alpha=.05,sd=FALSE,add=FALSE,...)
```

Arguments

`x`	A vector of data or summary statistics (from Describe)
`y`	A second vector of data or summary statistics (also from Describe)
`labels`	the names of each pair – defaults to rownames of x
`main`	The title for the graph
`xlim`	xlim values if desired– defaults to min and max mean(x) +/- 2 se
`ylim`	ylim values if desired – defaults to min and max mean(y) +/- 2 se
`xlab`	label for x axis – grouping variable 1
`ylab`	label for y axis – grouping variable 2
`pos`	Labels are located where with respect to the mean?
`offset`	Labels are then offset from this location
`arrow.len`	Arrow length
`alpha`	alpha level of error bars
`sd`	if sd is TRUE, then draw means +/- 1 sd)
`add`	if TRUE, overlay the values with a prior plot
`...`	Other parameters for plot

Details

For an example of two way error bars describing the effects of mood manipulations upon positive and negative affect, see http://personality-project.org/revelle/publications/happy-sad-appendix/FIG.A-6.pdf

The second example shows how error crosses can be done for multiple variables where the grouping variable is found dynamically. The errorCircles example shows how to do this in one step.

Author(s)

William Revelle
<revelle@northwestern.edu>

See Also

To draw error bars for single variables error.bars, or by groups error.bars.by, or to find descriptive statistics describe or descriptive statistics by a grouping variable describeBy and statsBy.

A much improved version is now called errorCircles.

Examples

```
#just draw one pair of variables
desc <- describe(attitude)
x <- desc[1,]
y <- desc[2,]
error.crosses(x,y,xlab=rownames(x),ylab=rownames(y))

#now for a bit more complicated plotting
data(bfi)
desc <- describeBy(bfi[1:25],bfi$gender) #select a high and low group
error.crosses(desc$'1',desc$'2',ylab="female scores",xlab="male scores",main="BFI scores by gender")
 abline(a=0,b=1)

#do it from summary statistics  (using standard errors)
g1.stats <- data.frame(n=c(10,20,30),mean=c(10,12,18),se=c(2,3,5))
g2.stats <- data.frame(n=c(15,20,25),mean=c(6,14,15),se =c(1,2,3))
error.crosses(g1.stats,g2.stats)

#Or, if you prefer to draw +/- 1 sd. instead of 95% confidence
g1.stats <- data.frame(n=c(10,20,30),mean=c(10,12,18),sd=c(2,3,5))
g2.stats <- data.frame(n=c(15,20,25),mean=c(6,14,15),sd =c(1,2,3))
error.crosses(g1.stats,g2.stats,sd=TRUE)

#and seem even fancy plotting: This is taken from a study of mood
#four films were given (sad, horror, neutral, happy)
#with a pre and post test
data(affect)
colors <- c("black","red","white","blue")
films <- c("Sad","Horror","Neutral","Happy")
```

```
affect.mat <- describeBy(affect[10:17],affect$Film,mat=TRUE)
 error.crosses(affect.mat[c(1:4,17:20),],affect.mat[c(5:8,21:24),],
    labels=films[affect.mat$group1],xlab="Energetic Arousal",
     ylab="Tense Arousal",col=colors[affect.mat$group1],pch=16,cex=2)
```

`errorCircles` *Two way plots of means, error bars, and sample sizes*

Description

Given a matrix or data frame, data, find statistics based upon a grouping variable and then plot x and y means with error bars for each value of the grouping variable. If the data are paired (e.g. by gender), then plot means and error bars for the two groups on all variables.

Usage

```
errorCircles(x, y, data, ydata = NULL, group=NULL, paired = FALSE, labels = NULL,
main = NULL, xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,add=FALSE, pos = NULL,
 offset = 1, arrow.len = 0.2, alpha = 0.05, sd = FALSE, bars = TRUE, circles = TRUE, ...)
```

Arguments

`x`	The x variable (by name or number) to plot
`y`	The y variable (name or number) to plot
`data`	The matrix or data.frame to use for the x data
`ydata`	If plotting data from two data.frames, then the y variable of the ydata frame will be used.
`group`	If specified, then `statsBy` is called first to find the statistics by group
`paired`	If TRUE, plot all x and y variables for the two values of the grouping variable.
`labels`	Variable names
`main`	Main title for plot
`xlim`	xlim values if desired– defaults to min and max mean(x) +/- 2 se
`ylim`	ylim values if desired – defaults to min and max mean(y) +/- 2 se
`xlab`	label for x axis – grouping variable 1
`ylab`	label for y axis – grouping variable 2
`add`	If TRUE, add to the prior plot
`pos`	Labels are located where with respect to the mean?
`offset`	Labels are then offset from this location
`arrow.len`	Arrow length
`alpha`	alpha level of error bars
`sd`	if sd is TRUE, then draw means +/- 1 sd)
`bars`	Should error.bars be drawn for both x and y
`circles`	Should circles representing the relative sample sizes be drawn?
`...`	Other parameters for plot

Details

When visualizing the effect of an experimental manipulation or the relationship of multiple groups, it is convenient to plot their means as well as their confidence regions in a two dimensional space.

Value

If the group variable is specified, then the statistics from statsBy are (invisibly) returned.

Note

Basically this is a combination (and improvement) of statsBy with error.crosses. Can also serve some of the functionality of error.bars.by (see the last example).

Author(s)

William Revelle

See Also

statsBy, describeBy, error.crosses

Examples

```
#BFI scores for males and females
errorCircles(1:25,1:25,data=bfi,group="gender",paired=TRUE,ylab="female scores",
      xlab="male scores",main="BFI scores by gender")
 abline(a=0,b=1)
#drop the circles since all samples are the same sizes
errorCircles(1:25,1:25,data=bfi,group="gender",paired=TRUE,circles=FALSE,
     ylab="female scores",xlab="male scores",main="BFI scores by gender")
 abline(a=0,b=1)

 data(affect)
colors <- c("black","red","white","blue")
films <- c("Sad","Horror","Neutral","Happy")
affect.stats <- errorCircles("EA2","TA2",data=affect[-c(1,20)],group="Film",labels=films,
      xlab="Energetic Arousal",ylab="Tense Arousal",ylim=c(10,22),xlim=c(8,20),
      pch=16,cex=2,col=colors, main ="EA and TA pre and post affective movies")
#now, use the stats from the prior run
errorCircles("EA1","TA1",data=affect.stats,labels=films,pch=16,cex=2,col=colors,add=TRUE)

#Can also provide error.bars.by functionality
 errorCircles(2,5,group=2,data=sat.act,circles=FALSE,pch=16,col="blue",
      ylim= c(200,800),main="SATV by education",labels="")
 #just do the breakdown and then show the points
# errorCircles(3,5,group=3,data=sat.act,circles=FALSE,pch=16,col="blue",
#         ylim= c(200,800),main="SATV by age",labels="",bars=FALSE)
```

fa — *Exploratory Factor analysis using MinRes (minimum residual) as well as EFA by Principal Axis, Weighted Least Squares or Maximum Likelihood*

Description

Among the many ways to do latent variable exploratory factor analysis (EFA), one of the better is to use Ordinary Least Squares (OLS) to find the minimum residual (minres) solution. This produces solutions very similar to maximum likelihood even for badly behaved matrices. A variation on minres is to do weighted least squares (WLS). Perhaps the most conventional technique is principal axes (PAF). An eigen value decomposition of a correlation matrix is done and then the communalities for each variable are estimated by the first n factors. These communalities are entered onto the diagonal and the procedure is repeated until the sum(diag(r)) does not vary. Yet another estimate procedure is maximum likelihood. For well behaved matrices, maximum likelihood factor analysis (either in the fa or in the factanal function) is probably preferred. Bootstrapped confidence intervals of the loadings and interfactor correlations are found by fa with n.iter > 1.

Usage

```
fa(r,nfactors=1,n.obs = NA,n.iter=1, rotate="oblimin", scores="regression",
residuals=FALSE, SMC=TRUE, covar=FALSE,missing=FALSE,impute="median",
min.err = 0.001,  max.iter = 50,symmetric=TRUE, warnings=TRUE, fm="minres",
 alpha=.1,p=.05,oblique.scores=FALSE,np.obs,use="pairwise",cor="cor",weight=NULL,...)

fac(r,nfactors=1,n.obs = NA, rotate="oblimin", scores="tenBerge", residuals=FALSE,
 SMC=TRUE, covar=FALSE,missing=FALSE,impute="median",min.err = 0.001,
max.iter=50,symmetric=TRUE,warnings=TRUE,fm="minres",alpha=.1,
oblique.scores=FALSE,np.obs,use="pairwise",cor="cor",weight=NULL,...)

fa.poly(x,nfactors=1,n.obs = NA, n.iter=1, rotate="oblimin", SMC=TRUE,  missing=FALSE,
 impute="median", min.err = .001, max.iter=50, symmetric=TRUE, warnings=TRUE,
 fm="minres",alpha=.1, p =.05,scores="regression", oblique.scores=TRUE,
        weight=NULL,global=TRUE,...)  #deprecated

factor.minres(r, nfactors=1, residuals = FALSE, rotate = "varimax",n.obs = NA,
scores = FALSE,SMC=TRUE, missing=FALSE,impute="median",min.err = 0.001, digits = 2,
 max.iter = 50,symmetric=TRUE,warnings=TRUE,fm="minres")    #deprecated

factor.wls(r,nfactors=1,residuals=FALSE,rotate="varimax",n.obs = NA,
scores=FALSE,SMC=TRUE,missing=FALSE,impute="median", min.err = .001,
 digits=2,max.iter=50,symmetric=TRUE,warnings=TRUE,fm="wls")  #deprecated
```

Arguments

`r` — A correlation or covariance matrix or a raw data matrix. If raw data, the correlation matrix will be found using pairwise deletion. If covariances are supplied, they will be converted to correlations unless the covar option is TRUE.

`x`	For fa.poly.ci, only raw data may be used
`nfactors`	Number of factors to extract, default is 1
`n.obs`	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics. Must be specified if using a correlaton matrix and finding confidence intervals.
`np.obs`	The pairwise number of observations. Used if using a correlation matrix and asking for a minchi solution.
`rotate`	"none", "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor" are orthogonal rotations. "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible oblique transformations of the solution. The default is to do a oblimin transformation, although versions prior to 2009 defaulted to varimax.
`n.iter`	Number of bootstrap interations to do in fa or fa.poly
`residuals`	Should the residual matrix be shown
`scores`	the default="regression" finds factor scores using regression. Alternatives for estimating factor scores include simple regression ("Thurstone"), correlaton preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (`factor.scores`). Although scores="tenBerge" is probably preferred for most solutions, it will lead to problems with some improper correlation matrices.
`SMC`	Use squared multiple correlations (SMC=TRUE) or use 1 as initial communality estimate. Try using 1 if imaginary eigen values are reported. If SMC is a vector of length the number of variables, then these values are used as starting values in the case of fm='pa'.
`covar`	if covar is TRUE, factor the covariance matrix, otherwise factor the correlation matrix
`missing`	if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean
`impute`	"median" or "mean" values are used to replace missing values
`min.err`	Iterate until the change in communalities is less than min.err
`digits`	How many digits of output should be returned– deprecated – now specified in the print function
`max.iter`	Maximum number of iterations for convergence
`symmetric`	symmetric=TRUE forces symmetry by just looking at the lower off diagonal values
`warnings`	warnings=TRUE => warn if number of factors is too many
`fm`	factoring method fm="minres" will do a minimum residual (OLS), fm="wls" will do a weighted least squares (WLS) solution, fm="gls" does a generalized weighted least squares (GLS), fm="pa" will do the principal factor solution, fm="ml" will do a maximum likelihood factor analysis. fm="minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair.
`alpha`	alpha level for the confidence intervals for RMSEA

`p`	if doing iterations to find confidence intervals, what probability values should be found for the confidence intervals
`oblique.scores`	When factor scores are found, should they be based on the structure matrix (default) or the pattern matrix (oblique.scores=TRUE).
`weight`	If not NULL, a vector of length n.obs that contains weights for each observation. The NULL case is equivalent to all cases being weighted 1.
`use`	How to treat missing data, use="pairwise" is the default". See cor for other options.
`cor`	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate
`global`	should overall taus be used in polychoric or should they be found for each pair. Necessary to be set to false in the case of different number of alternatives for each item.
`...`	additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax

Details

Factor analysis is an attempt to approximate a correlation or covariance matrix with one of lesser rank. The basic model is that ${}_nR_n \approx_n F_{kk}F_n' + U^2$ where k is much less than n. There are many ways to do factor analysis, and maximum likelihood procedures are probably the most preferred (see `factanal`). The existence of uniquenesses is what distinguishes factor analysis from principal components analysis (e.g., `principal`). If variables are thought to represent a "true" or latent part then factor analysis provides an estimate of the correlations with the latent factor(s) representing the data. If variables are thought to be measured without error, then principal components provides the most parsimonious description of the data.

The fa function will do factor analyses using one of four different algorithms: minimum residual (minres), principal axes, weighted least squares, or maximum likelihood.

Principal axes factor analysis has a long history in exploratory analysis and is a straightforward procedure. Successive eigen value decompositions are done on a correlation matrix with the diagonal replaced with diag (FF') until $\sum(diag(FF'))$ does not change (very much). The current limit of max.iter =50 seems to work for most problems, but the Holzinger-Harmon 24 variable problem needs about 203 iterations to converge for a 5 factor solution.

Not all factor programs that do principal axes do iterative solutions. The example from the SAS manual (Chapter 26) is such a case. To achieve that solution, it is necessary to specify that the max.iterations = 1. Comparing that solution to an iterated one (the default) shows that iterations improve the solution. In addition, fm="minres" or fm="mle" produces even better solutions for this example.

Principal axes may be used in cases when maximum likelihood solutions fail to converge, although fm="minres" will also do that and tends to produce better (smaller residuals) solutions.

The fm="minchi" option is a variation on the "minres" (ols) solution and minimizes the sample size weighted residuals rather than just the residuals. This was developed to handle the problem of data that Massively Missing Completely at Random (MMCAR) which a condition that happens in the SAPA project.

A problem in factor analysis is to find the best estimate of the original communalities. Using the Squared Multiple Correlation (SMC) for each variable will underestimate the communalities, using 1s will over estimate. By default, the SMC estimate is used. In either case, iterative techniques will tend to converge on a stable solution. If, however, a solution fails to be achieved, it is useful to try again using ones (SMC =FALSE). Alternatively, a vector of starting values for the communalities may be specified by the SMC option.

The iterated principal axes algorithm does not attempt to find the best (as defined by a maximum likelihood criterion) solution, but rather one that converges rapidly using successive eigen value decompositions. The maximum likelihood criterion of fit and the associated chi square value are reported, and will be worse than that found using maximum likelihood procedures.

The minimum residual (minres) solution is an unweighted least squares solution that takes a slightly different approach. It uses the `optim` function and adjusts the diagonal elements of the correlation matrix to mimimize the squared residual when the factor model is the eigen value decomposition of the reduced matrix. MINRES and PA will both work when ML will not, for they can be used when the matrix is singular. At least on a number of test cases, the MINRES solution is slightly more similar to the ML solution than is the PA solution. To a great extent, the minres and wls solutions follow ideas in the `factanal` function.

The weighted least squares (wls) solution weights the residual matrix by 1/ diagonal of the inverse of the correlation matrix. This has the effect of weighting items with low communalities more than those with high communalities.

The generalized least squares (gls) solution weights the residual matrix by the inverse of the correlation matrix. This has the effect of weighting those variables with low communalities even more than those with high communalities.

The maximum likelihood solution takes yet another approach and finds those communality values that minimize the chi square goodness of fit test. The fm="ml" option provides a maximum likelihood solution following the procedures used in `factanal` but does not provide all the extra features of that function.

Test cases comparing the output to SPSS suggest that the PA algorithm matches what SPSS calls uls, and that the wls solutions are equivalent in their fits. The wls and gls solutions have slightly larger eigen values, but slightly worse fits of the off diagonal residuals than do the minres or maximum likelihood solutions. Comparing the results to the examples in Harman (76), the PA solution with no iterations matches what Harman calls Principal Axes (as does SAS), while the iterated PA solution matches his minres solution. The minres solution found in psych tends to have slightly smaller off diagonal residuals (as it should) than does the iterated PA solution.

Although for items, it is typical to find factor scores by scoring the salient items (using, e.g., `scoreItems`) factor scores can be estimated by regression as well as several other means. There are multiple approaches that are possible (see Grice, 2001) and the one taken here was developed by tenBerge et al. (see `factor.scores`. The alternative, which will match factanal is to find the scores using regression – Thurstone's least squares regression where the weights are found by $W = R^(-1)S$ where R is the correlation matrix of the variables ans S is the structure matrix. Then, factor scores are just $Fs = XW$.

In the oblique case, the factor loadings are referred to as Pattern coefficients and are related to the Structure coefficients by $S = P\Phi$ and thus $P = S\Phi^{-1}$. When estimating factor scores, `fa` and `factanal` differ in that `fa` finds the factors from the Structure matrix while `factanal` seems to do it from the Pattern matrix. Thus, although in the orthogonal case, fa and factanal agree perfectly in their factor score estimates, they do not agree in the case of oblique factors. Setting oblique.scores = TRUE will produce factor score estimate that match those of `factanal`.

It is sometimes useful to extend the factor solution to variables that were not factored. This may be done using `fa.extension`. Factor extension is typically done in the case where some variables were not appropriate to factor, but factor loadings on the original factors are still desired.

For dichotomous items or polytomous items, it is recommended to analyze the `tetrachoric` or `polychoric` correlations rather than the Pearson correlations. This may be done by specifying cor="poly" or cor="tet" or cor="mixed" if the data have a mixture of dichotomous, polytomous, and continous variables.

Analysis of dichotomous or polytomous data may also be done by using `irt.fa` or `fa.poly` functions. In the first case, the factor analysis results are reported in Item Response Theory (IRT) terms, although the original factor solution is returned in the results. In the later case, a typical factor loadings matrix is returned, but the tetrachoric/polychoric correlation matrix and item statistics are saved for reanalysis by `irt.fa`. (See also the `mixed.cor` function to find correlations from a mixture of continuous, dichotomous, and polytomous items.)

Of the various rotation/transformation options, varimax, Varimax, quartimax, bentlerT, geominT, and bifactor do orthogonal rotations. Promax transforms obliquely with a target matix equal to the varimax solution. oblimin, quartimin, simplimax, bentlerQ, geominQ and biquartimin are oblique transformations. Most of these are just calls to the GPArotation package. The “cluster” option does a targeted rotation to a structure defined by the cluster representation of a varimax solution. With the optional "keys" parameter, the "target" option will rotate to a target supplied as a keys matrix. (See `target.rot`.)

Two additional target rotation options are available through calls to GPArotation. These are the targetQ (oblique) and targetT (orthogonal) target rotations of Michael Browne. See `target.rot` for more documentation.

The "bifactor" rotation implements the Jennrich and Bentler (2011) bifactor rotation by calling the GPForth function in the GPArotation package and using two functions adapted from the MatLab code of Jennrich and Bentler.

There are two varimax rotation functions. One, Varimax, in the GPArotation package does not by default apply Kaiser normalization. The other, varimax, in the stats package, does. It appears that the two rotation functions produce slightly different results even when normalization is set. For consistency with the other rotation functions, Varimax is probably preferred.

The rotation matrix (rot.mat) is returned from all of these options. This is the inverse of the Th (theta?) object returned by the GPArotation package. The correlations of the factors may be found by $\Phi = \theta'\theta$

There are three ways to handle dichotomous or polytomous responses: `fa` with the cor="poly" option, `fa.poly` which will return the tetrachoric or polychoric correlation matrix, as well as the normal factor analysis output, and `irt.fa` which returns a two parameter irt analysis as well as the normal fa output.

When factor analyzing items with dichotomous or polytomous responses, the `irt.fa` function provides an Item Response Theory representation of the factor output. The factor analysis results are available, however, as an object in the irt.fa output.

`fa.poly` is appropriate if the data are categorical (but just setting the cor="poly" option works as well). It will produce normal factor analysis output but also will save the polychoric matrix (rho) and items difficulties (tau) for subsequent irt analyses. `fa.poly` will, by default, find factor scores if the data are available. The correlations are found using either `tetrachoric` or `polychoric` and then this matrix is factored. Weights from the factors are then applied to the original data to estimate factor scores.

The function `fa` will repeat the analysis n.iter times on a bootstrapped sample of the data (if they exist) or of a simulated data set based upon the observed correlation matrix. The mean estimate and standard deviation of the estimate are returned and will print the original factor analysis as well as the alpha level confidence intervals for the estimated coefficients. The bootstrapped solutions are rotated towards the original solution using target.rot. The factor loadings are z-transformed, averaged and then back transformed. This leads to an error in the case of Heywood cases. The probably better alternative is to just find the mean bootstrapped value and find the confidence intervals based upon the observed range of values. The default is to have n.iter =1 and thus not do bootstrapping.

`fa.poly` will find confidence intervals for a factor solution for dichotomous or polytomous items (set n.iter > 1 to do so). But, so will `fa` with the cor="poly" option. Perhaps more useful is to find the Item Response Theory parameters equivalent to the factor loadings reported in fa.poly by using the `irt.fa` function.

Some correlation matrices that arise from using pairwise deletion or from tetrachoric or polychoric matrices will not be proper. That is, they will not be positive semi-definite (all eigen values >= 0). The `cor.smooth` function will adjust correlation matrices (smooth them) by making all negative eigen values slightly greater than 0, rescaling the other eigen values to sum to the number of variables, and then recreating the correlation matrix. See `cor.smooth` for an example of this problem using the `burt` data set.

For those who like SPSS type output, the measure of factoring adequacy known as the Kaiser-Meyer-Olkin `KMO` test may be found from the correlation matrix or data matrix using the `KMO` function. Similarly, the Bartlett's test of Sphericity may be found using the `cortest.bartlett` function.

For those who want to have an object of the variances accounted for, this is returned invisibly by the print function. (e.g., p <- print(fa(ability))$Vaccounted)

The output from the print.psych.fa function displays the factor loadings (from the pattern matrix, the h2 (communalities) the u2 (the uniquenesses), com (the complexity of the factor loadings for that variable (see below). In the case of an orthogonal solution, h2 is merely the row sum of the squared factor loadings. But for an oblique solution, it is the row sum of the orthogonal factor loadings (remember, that rotations or transformations do not change the communality).

Value

`values`	Eigen values of the common factor solution
`e.values`	Eigen values of the original matrix
`communality`	Communality estimates for each item. These are merely the sum of squared factor loadings for that item.
`rotation`	which rotation was requested?
`n.obs`	number of observations specified or found
`loadings`	An item by factor (pattern) loading matrix of class "loadings" Suitable for use in other programs (e.g., GPA rotation or factor2cluster. To show these by sorted order, use `print.psych` with sort=TRUE
`complexity`	Hoffman's index of complexity for each item. This is just $\frac{(\Sigma a_i^2)^2}{\Sigma a_i^4}$ where a_i is the factor loading on the ith factor. From Hofmann (1978), MBR. See also Pettersson and Turkheimer (2010).

Structure
: An item by factor structure matrix of class “loadings". This is just the loadings (pattern) matrix times the factor intercorrelation matrix.

fit
: How well does the factor model reproduce the correlation matrix. This is just $\frac{\Sigma r_{ij}^2 - \Sigma r_{ij}^{*2}}{\Sigma r_{ij}^2}$ (See VSS, ICLUST, and principal for this fit statistic.

fit.off
: how well are the off diagonal elements reproduced?

dof
: Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters. Let n=Number of items, nf = number of factors then
$dof = n * (n-1)/2 - n * nf + nf * (nf-1)/2$

objective
: Value of the function that is minimized by a maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is
$f = log(trace((FF' + U2)^{-1}R) - log(|(FF' + U2)^{-1}R|) - n.items$. When using MLE, this function is minimized. When using OLS (minres), although we are not minimizing this function directly, we can still calculate it in order to compare the solution to a MLE fit.

STATISTIC
: If the number of observations is specified or found, this is a chi square based upon the objective function, f (see above). Using the formula from factanal(which seems to be Bartlett's test) :
$\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$

PVAL
: If n.obs > 0, then what is the probability of observing a chisquare this large or larger?

Phi
: If oblique rotations (using oblimin from the GPArotation package or promax) are requested, what is the interfactor correlation.

communality.iterations
: The history of the communality estimates (For principal axis only.) Probably only useful for teaching what happens in the process of iterative fitting.

residual
: The matrix of residual correlations after the factor model is applied. To display it conveniently, use the residuals command.

chi
: When normal theory fails (e.g., in the case of non-positive definite matrices), it useful to examine the empirically derived χ^2 based upon the sum of the squared residuals * N. This will differ slightly from the MLE estimate which is based upon the fitting function rather than the actual residuals.

rms
: This is the sum of the squared (off diagonal residuals) divided by the degrees of freedom. Comparable to an RMSEA which, because it is based upon χ^2, requires the number of observations to be specified. The rms is an empirical value while the RMSEA is based upon normal theory and the non-central χ^2 distribution. That is to say, if the residuals are particularly non-normal, the rms value and the associated χ^2 and RMSEA can differ substantially.

crms
: rms adjusted for degrees of freedom

RMSEA
: The Root Mean Square Error of Approximation is based upon the non-central χ^2 distribution and the χ^2 estimate found from the MLE fitting function. With normal theory data, this is fine. But when the residuals are not distributed according to a noncentral χ^2, this can give very strange values. (And thus the

confidence intervals can not be calculated.) The RMSEA is a conventional index of goodness (badness) of fit but it is also useful to examine the actual rms values.

`TLI` The Tucker Lewis Index of factoring reliability which is also known as the non-normed fit index.

`BIC` Based upon χ^2 with the assumption of normal theory and using the χ^2 found using the objective function defined above. This is just $\chi^2 - 2df$

`eBIC` When normal theory fails (e.g., in the case of non-positive definite matrices), it useful to examine the empirically derived eBIC based upon the empirical χ^2 - 2 df.

`R2` The multiple R square between the factors and factor score estimates, if they were to be found. (From Grice, 2001). Derived from R2 is is the minimum correlation between any two factor estimates = 2R2-1.

`r.scores` The correlations of the factor score estimates using the specified model, if they were to be found. Comparing these correlations with that of the scores themselves will show, if an alternative estimate of factor scores is used (e.g., the tenBerge method), the problem of factor indeterminacy. For these correlations will not necessarily be the same.

`weights` The beta weights to find the factor score estimates. These are also used by the `predict.psych` function to find predicted factor scores for new cases.

`scores` The factor scores as requested. Note that these scores reflect the choice of the way scores should be estimated (see scores in the input). That is, simple regression ("Thurstone"), correlaton preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (see `factor.scores`). The correlation between factor score estimates (r.scores) is based upon using the regression/Thurstone approach. The actual correlation between scores will reflect the rotation algorithm chosen and may be found by correlating those scores.

`valid` The validity coffiecient of course coded (unit weighted) factor score estimates (From Grice, 2001)

`score.cor` The correlation matrix of course coded (unit weighted) factor score estimates, if they were to be found, based upon the loadings matrix rather than the weights matrix.

`rot.mat` The rotation matrix as returned from GPArotation.

Note

Thanks to Erich Studerus for some very helpful suggestions about various rotation and factor scoring algorithms, and to Gumundur Arnkelsson for suggestions about factor scores for singular matrices.

The fac function is the original fa function which is now called by fa repeatedly to get confidence intervals.

SPSS will sometimes use a Kaiser normalization before rotating. This will lead to different solutions than reported here. To get the Kaiser normalized loadings, use `kaiser`.

The communality for a variable is the amount of variance accounted for by all of the factors. That is to say, for orthogonal factors, it is the sum of the squared factor loadings (rowwise). The communality is insensitive to rotation. However, if an oblique solution is found, then the communality is not the sum of squared pattern coefficients. In both cases (oblique or orthogonal) the communality is the diagonal of the reproduced correlation matrix where ${}_nR_n = {}_n P_{kk}\Phi_{kk}P_n'$ where P is the pattern matrix and Φ is the factor intercorrelation matrix. This is the same, of course to multiplying the pattern by the structure: $R = PS'$ R = PS' where the Structure matrix is $S = \Phi P$. Similarly, the eigen values are the diagonal of the product ${}_k\Phi_{kk}P'_{nn}P_k$.

A frequently asked question is why are the factor names of the rotated solution not in ascending order? That is, for example, if factoring the 25 items of the bfi, the factor names are MR2 MR3 MR5 MR1 MR4, rather than the seemingly more logical "MR1" "MR2" "MR3" "MR4" "MR5". This is for pedagogical reasons, in that factors as extracted are orthogonal and are in order of amount of variance accounted for. But when rotated (orthogonally) or transformed (obliquely) the simple structure solution does not preserve that order. The factor names are, of course, arbitrary, and are kept with the original names to show the effect of rotation/transformation. To give them names associated with their ordinal position, simply paste("F", 1:nf, sep="") where nf is the number of factors. See the last example.

Correction to documentation: as of September, 2014, the oblique.scores option is correctly explained. (It had been backwards.) The default (oblique.scores=FALSE) finds scores based upon the Structure matrix, while oblique.scores=TRUE finds them based upon the pattern matrix. The latter case matches factanal. This error was detected by Mark Seeto.

Author(s)

William Revelle

References

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.

Grice, James W. (2001), Computing and evaluating factor scores. Psychological Methods, 6, 430-450

Harman, Harry and Jones, Wayne (1966) Factor analysis by minimizing residuals (minres), Psychometrika, 31, 3, 351-368.

Hofmann, R. J. (1978) . Complexity and simplicity as objective indices descriptive of factor solutions. Multivariate Behavioral Research, 13, 247-250.

Pettersson E, Turkheimer E. (2010) Item selection, evaluation, and simple structure in personality data. Journal of research in personality, 44(4), 407-420.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at `http://personality-project.org/r/book/`

See Also

`principal` for principal components analysis (PCA). PCA will give very similar solutions to factor analysis when there are many variables. The differences become more salient as the number variables decrease. The PCA and FA models are actually very different and should not be confused. One is a model of the observed variables, the other is a model of latent variables.

`irt.fa` for Item Response Theory analyses using factor analysis, using the two parameter IRT equivalent of loadings and difficulties.

`VSS` will produce the Very Simple Structure (VSS) and MAP criteria for the number of factors, `nfactors` to compare many different factor criteria.

ICLUST will do a hierarchical cluster analysis alternative to factor analysis or principal components analysis.

predict.psych to find predicted scores based upon new data, fa.extension to extend the factor solution to new variables, omega for hierarchical factor analysis with one general factor. codefa.multi for hierarchical factor analysis with an arbitrary number of higher order factors.

fa.sort will sort the factor loadings into echelon form. fa.organize will reorganize the factor pattern matrix into any arbitrary order of factors and items.

KMO and cortest.bartlett for various tests that some people like.

factor2cluster will prepare unit weighted scoring keys of the factors that can be used with scoreItems.

fa.lookup will print the factor analysis loadings matrix along with the item "content" taken from a dictionary of items. This is useful when examining the meaning of the factors.

anova.psych allows for testing the difference between two (presumably nested) factor models .

Examples

```
#using the Harman 24 mental tests, compare a principal factor with a principal components solution
pc <- principal(Harman74.cor$cov,4,rotate="varimax")   #principal components
pa <- fa(Harman74.cor$cov,4,fm="pa" ,rotate="varimax")  #principal axis
uls <- fa(Harman74.cor$cov,4,rotate="varimax")        #unweighted least squares is minres
wls <- fa(Harman74.cor$cov,4,fm="wls")       #weighted least squares

#to show the loadings sorted by absolute value
print(uls,sort=TRUE)

#then compare with a maximum likelihood solution using factanal
mle <- factanal(covmat=Harman74.cor$cov,factors=4)
factor.congruence(list(mle,pa,pc,uls,wls))
#note that the order of factors and the sign of some of factors may differ

#finally, compare the unrotated factor, ml, uls, and  wls solutions
wls <- fa(Harman74.cor$cov,4,rotate="none",fm="wls")
pa <- fa(Harman74.cor$cov,4,rotate="none",fm="pa")
minres <-  factanal(factors=4,covmat=Harman74.cor$cov,rotation="none")
mle <- fa(Harman74.cor$cov,4,rotate="none",fm="mle")
uls <- fa(Harman74.cor$cov,4,rotate="none",fm="uls")
factor.congruence(list(minres,mle,pa,wls,uls))
#in particular, note the similarity of the mle and min res solutions
#note that the order of factors and the sign of some of factors may differ



#an example of where the ML and PA and MR models differ is found in Thurstone.33.
#compare the first two factors with the 3 factor solution
Thurstone.33 <- as.matrix(Thurstone.33)
mle2 <- fa(Thurstone.33,2,rotate="none",fm="mle")
mle3 <- fa(Thurstone.33,3 ,rotate="none",fm="mle")
pa2 <- fa(Thurstone.33,2,rotate="none",fm="pa")
pa3 <- fa(Thurstone.33,3,rotate="none",fm="pa")
mr2 <- fa(Thurstone.33,2,rotate="none")
```

```
mr3 <- fa(Thurstone.33,3,rotate="none")
factor.congruence(list(mle2,mr2,pa2,mle3,pa3,mr3))

#f5 <- fa(bfi[1:25],5)
#f5  #names are not in ascending numerical order (see note)
#colnames(f5$loadings) <- paste("F",1:5,sep="")
#f5
```

---

fa.diagram *Graph factor loading matrices*

Description

Factor analysis or principal components analysis results are typically interpreted in terms of the major loadings on each factor. These structures may be represented as a table of loadings or graphically, where all loadings with an absolute value > some cut point are represented as an edge (path). `fa.diagram` uses the various `diagram` functions to draw the diagram. `fa.graph` generates dot code for external plotting. `fa.rgraph` uses the Rgraphviz package (if available) to draw the graph. `het.diagram` will draw "heterarchy" diagrams of factor/scale solutions at different levels.

Usage

```
fa.diagram(fa.results,Phi=NULL,fe.results=NULL,sort=TRUE,labels=NULL,cut=.3,
     simple=TRUE, errors=FALSE,g=FALSE,digits=1,e.size=.05,rsize=.15,side=2,
    main,cex=NULL,marg=c(.5,.5,1,.5),adj=1, ...)
het.diagram(r,levels,cut=.3,digits=2,both=TRUE,
        main="Heterarchy diagram",l.cex,gap.size,...)
fa.graph(fa.results,out.file=NULL,labels=NULL,cut=.3,simple=TRUE,
   size=c(8,6), node.font=c("Helvetica", 14),
    edge.font=c("Helvetica", 10), rank.direction=c("RL","TB","LR","BT"),
     digits=1,main="Factor Analysis", ...)
fa.rgraph(fa.results,out.file=NULL,labels=NULL,cut=.3,simple=TRUE,
   size=c(8,6), node.font=c("Helvetica", 14),
    edge.font=c("Helvetica", 10), rank.direction=c("RL","TB","LR","BT"),
     digits=1,main="Factor Analysis",graphviz=TRUE, ...)
```

Arguments

fa.results	The output of factor analysis, principal components analysis, or ICLUST analysis. May also be a factor loading matrix from anywhere.
Phi	Normally not specified (it is is found in the FA, pc, or ICLUST, solution), this may be given if the input is a loadings matrix.
fe.results	the results of a factor extension analysis (if any)
out.file	If it exists, a dot representation of the graph will be stored here (fa.graph)
labels	Variable labels

cut	Loadings with abs(loading) > cut will be shown
simple	Only the biggest loading per item is shown
g	Does the factor matrix reflect a g (first) factor. If so, then draw this to the left of the variables, with the remaining factors to the right of the variables. It is useful to turn off the simple parameter in this case.
r	A correlation matrix for the het.diagram function
levels	A list of the elements in each level
both	Should arrows have double heads (in het.diagram)
size	graph size
sort	sort the factor loadings before showing the diagram
errors	include error estimates (as arrows)
e.size	size of ellipses
rsize	size of rectangles
side	on which side should error arrows go?
cex	modify font size
l.cex	modify the font size in arrows, defaults to cex
gap.size	The gap in the arrow for the label. Can be adjusted to compensate for variations in cex or l.cex
marg	sets the margins to be wider than normal, returns them to the normal size upon exit
adj	how many different positions (1-3) should be used for the numeric labels. Useful if they overlap each other.
node.font	what font should be used for nodes in fa.graph
edge.font	what font should be used for edges in fa.graph
rank.direction	parameter passed to Rgraphviz– which way to draw the graph
digits	Number of digits to show as an edgelable
main	Graphic title, defaults to "factor analyis" or "factor analysis and extension"
graphviz	Should we try to use Rgraphviz for output?
...	other parameters

Details

Path diagram representations have become standard in confirmatory factor analysis, but are not yet common in exploratory factor analysis. Representing factor structures graphically helps some people understand the structure.

fa.diagram does not use Rgraphviz and is the preferred function. fa.graph generates dot code to be used by an external graphics program. It does not have all the bells and whistles of fa.diagram, but these may be done in the external editor.

Hierarchical (bifactor) models may be drawn by specifying the g parameter as TRUE. This allows for an graphical displays of various factor transformations with a bifactor structure (e.g., bifactor and biquartimin. See omega for an alternative way to find these structures.

The het.diagram function will show the case of a hetarchical structure at multiple levels. It can also be used to show the patterns of correlations between sets of scales (e.g., EPI, NEO, BFI). The example is for showing the relationship between 3 sets of 4 variables from the Thurstone data set. The parameters l.cex and gap.size are used to adjust the font size of the labels and the gap in the lines.

In fa.rgraph although a nice graph is drawn for the orthogonal factor case, the oblique factor drawing is acceptable, but is better if cleaned up outside of R or done using fa.diagram.

The normal input is taken from the output of either fa or ICLUST. This latter case displays the ICLUST results in terms of the cluster loadings, not in terms of the cluster structure. Actually an interesting option.

It is also possible to just give a factor loading matrix as input. In this case, supplying a Phi matrix of factor correlations is also possible.

It is possible, using fa.graph, to export dot code for an omega solution. fa.graph should be applied to the schmid$sl object with labels specified as the rownames of schmid$sl. The results will need editing to make fully compatible with dot language plotting.

To specify the model for a structural equation confirmatory analysis of the results, use structure.diagram instead.

Value

fa.diagram: A path diagram is drawn without using Rgraphviz. This is probably the more useful function.

fa.rgraph: A graph is drawn using rgraphviz. If an output file is specified, the graph instructions are also saved in the dot language.

fa.graph: the graph instructions are saved in the dot language.

Note

fa.rgraph requires Rgraphviz. Because there are occasional difficulties installing Rgraphviz from Bioconductor in that some libraries are misplaced and need to be relinked, it is probably better to use fa.diagram or fa.graph.

Author(s)

William Revelle

See Also

omega.graph, ICLUST.graph, structure.diagram to convert the factor diagram to sem modeling code.

Examples

```
test.simple <- fa(item.sim(16),2,rotate="oblimin")
#if(require(Rgraphviz)) {fa.graph(test.simple) }
fa.diagram(test.simple)
f3 <- fa(Thurstone,3,rotate="cluster")
```

```
fa.diagram(f3,cut=.4,digits=2)
f3l <- f3$loadings
fa.diagram(f3l,main="input from a matrix")
Phi <- f3$Phi
fa.diagram(f3l,Phi=Phi,main="Input from a matrix")
fa.diagram(ICLUST(Thurstone,2,title="Two cluster solution of Thurstone"),main="Input from ICLUST")
het.diagram(Thurstone,levels=list(1:4,5:8,3:7))
```

---

fa.extension *Apply Dwyer's factor extension to find factor loadings for extended variables*

Description

Dwyer (1937) introduced a method for finding factor loadings for variables not included in the original analysis. This is basically finding the unattenuated correlation of the extension variables with the factor scores. An alternative, which does not correct for factor reliability was proposed by Gorsuch (1997). Both options are an application of exploratory factor analysis with extensions to new variables.

Usage

```
fa.extension(Roe,fo,correct=TRUE)
fa.extend(r,nfactors=1,ov=NULL,ev=NULL,n.obs = NA, np.obs=NULL,
  correct=TRUE,rotate="oblimin",SMC=TRUE,   warnings=TRUE,
      fm="minres",alpha=.1,omega=FALSE,  ...)
```

Arguments

Roe	The correlations of the original variables with the extended variables
fo	The output from the fa or omega functions applied to the original variables.
correct	correct=TRUE produces Dwyer's solution, correct=FALSE produces Gorsuch's solution
r	A correlation or data matrix with all of the variables to be analyzed by fa.extend
ov	The original variables to factor
ev	The extension variables
nfactors	Number of factors to extract, default is 1
n.obs	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics. Must be specified if using a correlaton matrix and finding confidence intervals.
np.obs	Pairwise number of observations. Required if using fm="minchi", suggested in other cases to estimate the empirical goodness of fit.

rotate	"none", "varimax", "quartimax", "bentlerT", "geominT" and "bifactor" are orthogonal rotations. "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible rotations or transformations of the solution. The default is to do a oblimin transformation, although versions prior to 2009 defaulted to varimax.
SMC	Use squared multiple correlations (SMC=TRUE) or use 1 as initial communality estimate. Try using 1 if imaginary eigen values are reported. If SMC is a vector of length the number of variables, then these values are used as starting values in the case of fm='pa'.
warnings	warnings=TRUE => warn if number of factors is too many
fm	factoring method fm="minres" will do a minimum residual (OLS), fm="wls" will do a weighted least squares (WLS) solution, fm="gls" does a generalized weighted least squares (GLS), fm="pa" will do the principal factor solution, fm="ml" will do a maximum likelihood factor analysis. fm="minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair.
alpha	alpha level for the confidence intervals for RMSEA
omega	Do the extension analysis for an omega type analysis
...	additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax

Details

It is sometimes the case that factors are derived from a set of variables (the Fo factor loadings) and we want to see what the loadings of an extended set of variables (Fe) would be. Given the original correlation matrix Ro and the correlation of these original variables with the extension variables of Roe, it is a straight forward calculation to find the loadings Fe of the extended variables on the original factors. This technique was developed by Dwyer (1937) for the case of adding new variables to a factor analysis without doing all the work over again. But, as discussed by Horn (1973) factor extension is also appropriate when one does not want to include the extension variables in the original factor analysis, but does want to see what the loadings would be anyway.

This could be done by estimating the factor scores and then finding the covariances of the extension variables with the factor scores. But if the original data are not available, but just the covariance or correlation matrix is, then the use of `fa.extension` is most appropriate.

The factor analysis results from either `fa` or `omega` functions applied to the original correlation matrix is extended to the extended variables given the correlations (Roe) of the extended variables with the original variables.

`fa.extension` assumes that the original factor solution was found by the `fa` function.

For a very nice discussion of the relationship between factor scores, correlation matrices, and the factor loadings in a factor extension, see Horn (1973).

The `fa.extend` function may be thought of as a "seeded" factor analysis. That is, the variables in the original set are factored, this solution is then extended to the extension set, and the resulting output is presented as if both the original and extended variables were factored together. This may also be done for an omega analysis.

The example of codefa.extend compares the extended solution to a direct solution of all of the variables using `factor.congruence`.

Value

Factor Loadings of the exended variables on the original factors

Author(s)

William Revelle

References

Paul S. Dwyer (1937) The determination of the factor loadings of a given test from the known factor loadings of other tests. Psychometrika, 3, 173-178

Gorsuch, Richard L. (1997) New procedure for extension analysis in exploratory factor analysis, Educational and Psychological Measurement, 57, 725-740

Horn, John L. (1973) On extension analysis and its relation to correlations between variables and factor scores. Multivariate Behavioral Research, 8, (4), 477-489.

See Also

See Also as fa, principal, Dwyer

Examples

```
 #The Dwyer Example
Ro <- Dwyer[1:7,1:7]
Roe <- Dwyer[1:7,8]
fo <- fa(Ro,2,rotate="none")
fe <- fa.extension(Roe,fo)

#an example from simulated data
set.seed(42)
 d <- sim.item(12)    #two orthogonal factors
 R <- cor(d)
 Ro <- R[c(1,2,4,5,7,8,10,11),c(1,2,4,5,7,8,10,11)]
 Roe <- R[c(1,2,4,5,7,8,10,11),c(3,6,9,12)]
 fo <- fa(Ro,2)
 fe <- fa.extension(Roe,fo)
 fa.diagram(fo,fe=fe)

 #create two correlated factors
 fx <- matrix(c(.9,.8,.7,.85,.75,.65,rep(0,12),.9,.8,.7,.85,.75,.65),ncol=2)
 Phi <- matrix(c(1,.6,.6,1),2)
 sim.data <- sim.structure(fx,Phi,n=1000,raw=TRUE)
 R <- cor(sim.data$observed)
 Ro <- R[c(1,2,4,5,7,8,10,11),c(1,2,4,5,7,8,10,11)]
 Roe <- R[c(1,2,4,5,7,8,10,11),c(3,6,9,12)]
 fo <- fa(Ro,2)
 fe <- fa.extension(Roe,fo)
 fa.diagram(fo,fe=fe)

 #now show how fa.extend works with the same data set
 #note that we have to make sure that the variables are in the order to do the factor congruence
```

```
fe2 <- fa.extend(R,2,ov=c(1,2,4,5,7,8,10,11),ev=c(3,6,9,12),n.obs=1000)
fa.diagram(fe2,main="factor analysis with extension variables")
fa2 <- fa(sim.data$observed[,c(1,2,4,5,7,8,10,11,3,6,9,12)],2)
factor.congruence(fe2,fa2)
summary(fe2)

#an example of extending an omega analysis


fload <- matrix(c(c(c(.9,.8,.7,.6),rep(0,20)),c(c(.9,.8,.7,.6),rep(0,20)),c(c(.9,.8,.7,.6),
       rep(0,20)),c(c(c(.9,.8,.7,.6),rep(0,20)),c(.9,.8,.7,.6))),ncol=5)
gload <- matrix(rep(.7,5))
five.factor <- sim.hierarchical(gload,fload,500,TRUE) #create sample data set
ss <- c(1,2,3,5,6,7,9,10,11,13,14,15,17,18,19)
Ro <- cor(five.factor$observed[,ss])
Re <- cor(five.factor$observed[,ss],five.factor$observed[,-ss])
om5 <-omega(Ro,5)   #the omega analysis
fa.extension(Re,om5) #the extension analysis
```

---

fa.multi *Multi level (hierarchical) factor analysis*

Description

Some factor analytic solutions produce correlated factors which may in turn be factored. If the solution has one higher order, the omega function is most appropriate. But, in the case of multi higher order factors, then the faMulti function will do a lower level factoring and then factor the resulting correlation matrix. Multi level factor diagrams are also shown.

Usage

```
fa.multi(r, nfactors = 3, nfact2 = 1, n.obs = NA, n.iter = 1, rotate = "oblimin",
 scores = "regression", residuals = FALSE, SMC = TRUE, covar = FALSE, missing =
 FALSE,impute = "median", min.err = 0.001, max.iter = 50, symmetric = TRUE, warnings
 =TRUE, fm = "minres", alpha = 0.1, p = 0.05, oblique.scores = FALSE, np.obs = NULL,
 use ="pairwise", cor = "cor", ...)

fa.multi.diagram(multi.results,sort=TRUE,labels=NULL,flabels=NULL,cut=.2,gcut=.2,
    simple=TRUE,errors=FALSE,
    digits=1,e.size=.1,rsize=.15,side=3,main=NULL,cex=NULL,color.lines=TRUE
    ,marg=c(.5,.5,1.5,.5),adj=2, ...)
```

Arguments

The arguments match those of the fa function.

A correlation matrix or raw data matrix

nfactors The desired number of factors for the lower level

nfact2 The desired number of factors for the higher level

n.obs Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics. Must be specified if using a correlaton matrix and finding confidence intervals.

np.obs The pairwise number of observations. Used if using a correlation matrix and asking for a minchi solution.

rotate "none", "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor" are orthogonal rotations. "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible oblique transformations of the solution. The default is to do a oblimin transformation, although versions prior to 2009 defaulted to varimax.

n.iter Number of bootstrap interations to do in fa or fa.poly

residuals Should the residual matrix be shown

scores the default="regression" finds factor scores using regression. Alternatives for estimating factor scores include simple regression ("Thurstone"), correlaton preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (see factor.scores). Although scores="tenBerge" is probably preferred for most solutions, it will lead to problems with some improper correlation matrices.

SMC Use squared multiple correlations (SMC=TRUE) or use 1 as initial communality estimate. Try using 1 if imaginary eigen values are reported. If SMC is a vector of length the number of variables, then these values are used as starting values in the case of fm='pa'.

covar if covar is TRUE, factor the covariance matrix, otherwise factor the correlation matrix

missing if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean

impute "median" or "mean" values are used to replace missing values

min.err Iterate until the change in communalities is less than min.err

max.iter Maximum number of iterations for convergence

symmetric symmetric=TRUE forces symmetry by just looking at the lower off diagonal values

warnings warnings=TRUE => warn if number of factors is too many

fm factoring method fm="minres" will do a minimum residual (OLS), fm="wls" will do a weighted least squares (WLS) solution, fm="gls" does a generalized weighted least squares (GLS), fm="pa" will do the principal factor solution, fm="ml" will do a maximum likelihood factor analysis. fm="minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair.

alpha alpha level for the confidence intervals for RMSEA

p if doing iterations to find confidence intervals, what probability values should be found for the confidence intervals

`oblique.scores`	When factor scores are found, should they be based on the structure matrix (default) or the pattern matrix (oblique.scores=TRUE).
`use`	How to treat missing data, use="pairwise" is the default". See cor for other options.
`cor`	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate
`multi.results`	The results from fa.multi
`labels`	variable labels
`flabels`	Labels for the factors (not counting g)
`size`	size of graphics window
`digits`	Precision of labels
`cex`	control font size
`color.lines`	Use black for positive, red for negative
`marg`	The margins for the figure are set to be wider than normal by default
`adj`	Adjust the location of the factor loadings to vary as factor mod 4 + 1
`main`	main figure caption
`...`	additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax. In addition, for fa.multi.diagram, other options to pass into the graphics packages
`e.size`	the size to draw the ellipses for the factors. This is scaled by the number of variables.
`cut`	Minimum path coefficient to draw
`gcut`	Minimum general factor path to draw
`simple`	draw just one path per item
`sort`	sort the solution before making the diagram
`side`	on which side should errors be drawn?
`errors`	show the error estimates
`rsize`	size of the rectangles

Details

See `fa` and `omega` for a discussion of factor analysis and of the case of one higher order factor.

Value

`f1`	The standard output from a factor analysis from `fa` for the raw variables
`f2`	The standard output from a factor analysis from `fa` for the correlation matrix of the level 1 solution.

Note

This is clearly an early implementation (Feb 14 2016) which might be improved.

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at http://personality-project.org/r/book/

See Also

fa, omega

Examples

```
f31 <- fa.multi(Thurstone,3,1) #compare with \code{\link{omega}}
f31
fa.multi.diagram(f31)
```

---

fa.parallel *Scree plots of data or correlation matrix compared to random "parallel" matrices*

Description

One way to determine the number of factors or components in a data matrix or a correlation matrix is to examine the "scree" plot of the successive eigenvalues. Sharp breaks in the plot suggest the appropriate number of components or factors to extract. "Parallel" analyis is an alternative technique that compares the scree of factors of the observed data with that of a random data matrix of the same size as the original. fa.parallel.poly does this for tetrachoric or polychoric analyses.

Usage

```
fa.parallel(x,n.obs=NULL,fm="minres",fa="both",main="Parallel Analysis Scree Plots",
n.iter=20,error.bars=FALSE,se.bars=TRUE,SMC=FALSE,ylabel=NULL,show.legend=TRUE,
sim=TRUE,quant=.95,cor="cor",use="pairwise")
fa.parallel.poly(x ,n.iter=10,SMC=TRUE,  fm = "minres",correct=TRUE,sim=FALSE,
       fa="both",global=TRUE)
## S3 method for class 'poly.parallel'
plot(x,show.legend=TRUE,fa="both",...)
```

Arguments

x	A data.frame or data matrix of scores. If the matrix is square, it is assumed to be a correlation matrix. Otherwise, correlations (with pairwise deletion) will be found
n.obs	n.obs=0 implies a data matrix/data.frame. Otherwise, how many cases were used to find the correlations.
fm	What factor method to use. (minres, ml, uls, wls, gls, pa) See fa for details.
fa	show the eigen values for a principal components (fa="pc") or a principal axis factor analysis (fa="fa") or both principal components and principal factors (fa="both")
main	a title for the analysis
n.iter	Number of simulated analyses to perform
use	How to treat missing data, use="pairwise" is the default". See cor for other options.
cor	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate. This matches the call to fa
correct	For tetrachoric correlations, should a correction for continuity be applied. (See tetrachoric)
sim	For continuous data, the default is to resample as well as to generate random normal data. If sim=FALSE, then just show the resampled results. These two results are very similar. This does not make sense in the case of correlation matrix, in which case resampling is impossible. In the case of polychoric or tetrachoric data, in addition to randomizing the real data, should we compare the solution to random simulated data. This will double the processing time, but will yiedl basically show the same result.
error.bars	Should error.bars be plotted (default = FALSE)
se.bars	Should the error bars be standard errors (the default) or 1 standard deviation (se.bars=FALSE). With many iterations, the standard errors are very small and some prefer to see the broader range.
SMC	SMC=TRUE finds eigen values after estimating communalities by using SMCs. smc = FALSE finds eigen values after estimating communalities with the first factor.
ylabel	Label for the y axis – defaults to “eigen values of factors and components", can be made empty to show many graphs
show.legend	the default is to have a legend. For multiple panel graphs, it is better to not show the legend
quant	if nothing is specified, the empirical eigen values are compared to the mean of the resampled or simulated eigen values. If a value (e.g., quant=.95) is specified, then the eigen values are compared against the matching quantile of the simulated data. Clearly the larger the value of quant, the few factors/components will be identified.

`global` If doing polychoric analyses (fa.parallel.poly) and the number of alternatives differ across items, it is necessary to turn off the global option

`...` additional plotting parameters, for plot.poly.parallel

Details

Cattell's "scree" test is one of most simple tests for the number of factors problem. Horn's (1965) "parallel" analysis is an equally compelling procedure. Other procedures for determining the most optimal number of factors include finding the Very Simple Structure (VSS) criterion (`VSS`) and Velicer's `MAP` procedure (included in `VSS`). Both the VSS and the MAP criteria are included in the `link{nfactors}` function which also reports the mean item complexity and the BIC for each of multiple solutions. fa.parallel plots the eigen values for a principal components and the factor solution (minres by default) and does the same for random matrices of the same size as the original data matrix. For raw data, the random matrices are 1) a matrix of univariate normal data and 2) random samples (randomized across rows) of the original data.

`fa.parallel.poly` will do parallel analysis for polychoric and tetrachoric factors. If the data are dichotomous, `fa.parallel.poly` will find tetrachoric correlations for the real and simulated data, otherwise, if the number of categories is less than 10, it will find polychoric correlations. Note that fa.parallel.poly is slower than fa.parallel because of the complexity of calculating the tetrachoric/polychoric correlations. The functionality of `fa.parallel.poly` is now included in `fa.parallel` with cor=poly option (etc.) option.

`fa.parallel` now will do tetrachorics or polychorics directly if the cor option is set to "tet" or "poly". As with `fa.parallel.poly` this will take longer.

The means of (ntrials) random solutions are shown. Error bars are usually very small and are suppressed by default but can be shown if requested. If the sim option is set to TRUE (default), then parallel analyses are done on resampled data as well as random normal data. In the interests of speed, the parallel analyses are done just on resampled data if sim=FALSE. Both procedures tend to agree.

As of version 1.5.4, I added the ability to specify the quantile of the simulated/resampled data, and to plot standard deviations or standard errors.

Alternative ways to estimate the number of factors problem are discussed in the Very Simple Structure (Revelle and Rocklin, 1979) documentation (`VSS`) and include Wayne Velicer's `MAP` algorithm (Veicer, 1976).

Parallel analysis for factors is actually harder than it seems, for the question is what are the appropriate communalities to use. If communalities are estimated by the Squared Multiple Correlation (SMC) `smc`, then the eigen values of the original data will reflect major as well as minor factors (see `sim.minor` to simulate such data). Random data will not, of course, have any structure and thus the number of factors will tend to be biased upwards by the presence of the minor factors.

By default, fa.parallel estimates the communalities based upon a one factor minres solution. Although this will underestimate the communalities, it does seem to lead to better solutions on simulated or real (e.g., the `bfi` or Harman74) data sets.

For comparability with other algorithms (e.g, the paran function in the paran package), setting smc=TRUE will use smcs as estimates of communalities. This will tend towards identifying more factors than the default option.

Printing the results will show the eigen values of the original data that are greater than simulated values.

A sad observation about parallel analysis is that it is sensitive to sample size. That is, for large data sets, the eigen values of random data are very close to 1. This will lead to different estimates of the number of factors as a function of sample size. Consider factor structure of the bfi data set (the first 25 items are meant to represent a five factor model). For samples of 200 or less, parallel analysis suggests 5 factors, but for 1000 or more, six factors and components are indicated. This is not due to an instability of the eigen values of the real data, but rather the closer approximation to 1 of the random data as n increases.

When simulating dichotomous data in fa.parallel.poly, the simulated data have the same difficulties as the original data. This functionally means that the simulated and the resampled results will be very similar. Note that fa.parallel.poly has functionally been replaced with fa.parallel with the cor="poly" option.

As with many psych functions, fa.parallel has been changed to allow for multicore processing. For running a large number of iterations, it is obviously faster to increase the number of cores to the maximum possible (using the options("mc.cores=n) command where n is determined from detectCores().

Value

A plot of the eigen values for the original data, ntrials of resampling of the original data, and of a equivalent size matrix of random normal deviates. If the data are a correlation matrix, specify the number of observations.

Also returned (invisibly) are:

`fa.values`	The eigen values of the factor model for the real data.
`fa.sim`	The descriptive statistics of the simulated factor models.
`pc.values`	The eigen values of a principal components of the real data.
`pc.sim`	The descriptive statistics of the simulated principal components analysis.
`nfact`	Number of factors with eigen values > eigen values of random data
`ncomp`	Number of components with eigen values > eigen values of random data
`values`	The simulated values for all simulated trials

Note

Although by default the test is applied to the mean eigen values, this can be modified by setting the quant parameter to any particular quantile. The actual simulated data are also returned (invisibly) in the value object. Thus, it is possible to do descriptive statistics on those to choose a preferred comparison. See the last example (not run)

Author(s)

William Revelle

References

Floyd, Frank J. and Widaman, Keith. F (1995) Factor analysis in the development and refinement of clinical assessment instruments. Psychological Assessment, 7(3):286-299, 1995.

Horn, John (1965) A rationale and test for the number of factors in factor analysis. Psychometrika, 30, 179-185.

Humphreys, Lloyd G. and Montanelli, Richard G. (1975), An investigation of the parallel analysis criterion for determining the number of common factors. Multivariate Behavioral Research, 10, 193-205.

Revelle, William and Rocklin, Tom (1979) Very simple structure - alternative procedure for estimating the optimal number of interpretable factors. Multivariate Behavioral Research, 14(4):403-414.

Velicer, Wayne. (1976) Determining the number of components from the matrix of partial correlations. Psychometrika, 41(3):321-327, 1976.

See Also

fa, nfactors, VSS, VSS.plot, VSS.parallel, sim.minor

Examples

```
#test.data <- Harman74.cor$cov   #The 24 variable Holzinger - Harman problem
#fa.parallel(test.data,n.obs=145)
fa.parallel(Thurstone,n.obs=213)   #the 9 variable Thurstone problem

#set.seed(123)
#minor <- sim.minor(24,4,400) #4 large and 12 minor factors
#ffa.parallel(minor$observed) #shows 5 factors and 4 components -- compare with
#fa.parallel(minor$observed,SMC=FALSE) #which shows 6  and 4 components factors
#a demonstration of parallel analysis of a dichotomous variable
#fp <- fa.parallel(ability)    #use the default Pearson correlation
#fpt <- fa.parallel(ability,cor="tet")  #do a tetrachoric correlation
#fpt <- fa.parallel(ability,cor="tet",quant=.95)  #do a tetrachoric correlation and
#use the 95th percentile of the simulated results
#apply(fp$values,2,function(x) quantile(x,.95))  #look at the 95th percentile of values
#apply(fpt$values,2,function(x) quantile(x,.95))  #look at the 95th percentile of values
#describe(fpt$values)  #look at all the statistics of the simulated values
```

---

fa.sort *Sort factor analysis or principal components analysis loadings*

Description

Although the print.psych function will sort factor analysis loadings, sometimes it is useful to do this outside of the print function. fa.sort takes the output from the fa or principal functions and sorts the loadings for each factor. Items are located in terms of their greatest loading. The new order is returned as an element in the fa list.

Usage

```
fa.sort(fa.results,polar=FALSE)
fa.organize(fa.results,o=NULL,i=NULL,cn=NULL)
```

Arguments

fa.results	The output from a factor analysis or principal components analysis using fa or principal.
polar	Sort by polar coordinates of first two factors (FALSE)
o	The order in which to order the factors
i	The order in which to order the items
cn	new factor names

Details

The fa.results$loadings are replaced with sorted loadings.

fa.organize takes a factor analysis or components output and reorganizes the factors in the o order. Items are organized in the i order. This is useful when comparing alternative factor solutions.

Value

A sorted factor analysis, principal components analysis, or omega loadings matrix.

These sorted values are used internally by the various diagram functions.

The values returned are the same as fa, except in sorted order. In addition, the order is returned as an additional element in the fa list.

Author(s)

William Revelle

See Also

See Also as fa,print.psych, fa.diagram,

Examples

```
test.simple <- fa(sim.item(16),2)
fa.sort(test.simple)
fa.organize(test.simple,c(2,1))  #the factors but not the items have been rearranged
```

---

factor.congruence *Coefficient of factor congruence*

Description

Given two sets of factor loadings, report their degree of congruence (vector cosine). Although first reported by Burt (1948), this is frequently known as the Tucker index of factor congruence.

Usage

```
factor.congruence(x, y=NULL,digits=2,use=NULL)
fa.congruence(x, y=NULL,digits=2,use=NULL)
```

Arguments

`x`	A matrix of factor loadings or a list of matrices of factor loadings
`y`	A second matrix of factor loadings (if x is a list, then y may be empty)
`digits`	Round off to digits
`use`	If NULL, then no loading matrices may contain missing values. If use="complete" then variables with any missing loadings are dropped (with a warning)

Details

Find the coefficient of factor congruence between two sets of factor loadings.

Factor congruences are the cosines of pairs of vectors defined by the loadings matrix and based at the origin. Thus, for loadings that differ only by a scaler (e.g. the size of the eigen value), the factor congruences will be 1.

For factor loading vectors of F1 and F2 the measure of factor congruence, phi, is

$$\phi = \frac{\sum F_1 F_2}{\sqrt{\sum(F_1^2)\sum(F_2^2)}}.$$

It is an interesting exercise to compare factor congruences with the correlations of factor loadings. Factor congruences are based upon the raw cross products, while correlations are based upon centered cross products. That is, correlations of factor loadings are cosines of the vectors based at the mean loading for each factor.

$$\phi = \frac{\sum(F_1 - a)(F_2 - b)}{\sqrt{\sum((F_1 - a)^2)\sum((F_2 - b)^2)}}.$$

.

For congruence coefficients, a = b= 0. For correlations a=mean F1, b= mean F2.

Input may either be matrices or factor analysis or principal components analyis output (which includes a loadings object), or a mixture of the two.

To compare more than two solutions, x may be a list of matrices, all of which will be compared.

Normally, all factor loading matrices should be complete (have no missing loadings). In the case where some loadings are missing, if the use option is specified, then variables with missing loadings are dropped.

Value

A matrix of factor congruences.

Author(s)

<revelle@northwestern.edu>
http://personality-project.org/revelle.html

References

Burt, Cyril (1948) The factorial study of temperamental traits. British Journal of Statistical Psychology, 1(3) 178-203.

Lorenzo-Seva, U. and ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. Methodology: European Journal of Research Methods for the Behavioral and Social Sciences, 2(2):57-64.

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.

Revelle, W. (In preparation) An Introduction to Psychometric Theory with applications in R (http://personality-project.org/r/book/)

See Also

principal, fa

Examples

```
#factor congruence of factors and components, both rotated
#fa <- fa(Harman74.cor$cov,4)
#pc <- principal(Harman74.cor$cov,4)
#factor.congruence(fa,pc)
 #    RC1  RC3  RC2  RC4
#MR1 0.98 0.41 0.28 0.32
#MR3 0.35 0.96 0.41 0.31
#MR2 0.23 0.16 0.95 0.28
#MR4 0.28 0.38 0.36 0.98



#factor congruence without rotation
#fa <- fa(Harman74.cor$cov,4,rotate="none")
#pc <- principal(Harman74.cor$cov,4,rotate="none")
#factor.congruence(fa,pc)   #just show the beween method congruences
#     PC1   PC2   PC3   PC4
#MR1 1.00 -0.04 -0.06 -0.01
#MR2 0.15  0.97 -0.01 -0.15
#MR3 0.31  0.05  0.94  0.11
#MR4 0.07  0.21 -0.12  0.96

#factor.congruence(list(fa,pc))  #this shows the within method congruence as well

 #     MR1   MR2  MR3   MR4  PC1   PC2   PC3   PC4
#MR1  1.00  0.11 0.25  0.06 1.00 -0.04 -0.06 -0.01
#MR2  0.11  1.00 0.06  0.07 0.15  0.97 -0.01 -0.15
#MR3  0.25  0.06 1.00  0.01 0.31  0.05  0.94  0.11
#MR4  0.06  0.07 0.01  1.00 0.07  0.21 -0.12  0.96
#PC1  1.00  0.15 0.31  0.07 1.00  0.00  0.00  0.00
#PC2 -0.04  0.97 0.05  0.21 0.00  1.00  0.00  0.00
#PC3 -0.06 -0.01 0.94 -0.12 0.00  0.00  1.00  0.00
#PC4 -0.01 -0.15 0.11  0.96 0.00  0.00  0.00  1.00
```

```
#pa <- fa(Harman74.cor$cov,4,fm="pa")
# factor.congruence(fa,pa)
#         PA1  PA3  PA2  PA4
#Factor1 1.00 0.61 0.46 0.55
#Factor2 0.61 1.00 0.50 0.60
#Factor3 0.46 0.50 1.00 0.57
#Factor4 0.56 0.62 0.58 1.00


#compare with
#round(cor(fa$loading,pc$loading),2)
#      RC1   RC3   RC2   RC4
#MR1  0.99 -0.18 -0.33 -0.34
#MR3 -0.33  0.96 -0.16 -0.43
#MR2 -0.29 -0.46  0.98 -0.21
#MR4 -0.44 -0.30 -0.22  0.98
```

`factor.fit` *How well does the factor model fit a correlation matrix. Part of the VSS package*

Description

The basic factor or principal components model is that a correlation or covariance matrix may be reproduced by the product of a factor loading matrix times its transpose: F'F or P'P. One simple index of fit is the 1 - sum squared residuals/sum squared original correlations. This fit index is used by `VSS`, `ICLUST`, etc.

Usage

```
factor.fit(r, f)
```

Arguments

r	a correlation matrix
f	A factor matrix of loadings.

Details

There are probably as many fit indices as there are psychometricians. This fit is a plausible estimate of the amount of reduction in a correlation matrix given a factor model. Note that it is sensitive to the size of the original correlations. That is, if the residuals are small but the original correlations are small, that is a bad fit.

Let

$$R* = R - FF'$$

$$fit = 1 - \frac{\sum(R*^2)}{\sum(R^2)}$$

.

The sums are taken for the off diagonal elements.

Value

fit

Author(s)

William Revelle

See Also

VSS, ICLUST

Examples

```
## Not run:
#compare the fit of 4 to 3 factors for the Harman 24 variables
fa4 <- factanal(x,4,covmat=Harman74.cor$cov)
round(factor.fit(Harman74.cor$cov,fa4$loading),2)
#[1] 0.9
fa3 <- factanal(x,3,covmat=Harman74.cor$cov)
round(factor.fit(Harman74.cor$cov,fa3$loading),2)
#[1] 0.88


## End(Not run)
```

`factor.model` *Find R = F F' + U2 is the basic factor model*

Description

The basic factor or principal components model is that a correlation or covariance matrix may be reproduced by the product of a factor loading matrix times its transpose. Find this reproduced matrix. Used by factor.fit, VSS, ICLUST, etc.

Usage

```
factor.model(f,Phi=NULL,U2=TRUE)
```

Arguments

f	A matrix of loadings.
Phi	A matrix of factor correlations
U2	Should the diagonal be model by ff' (U2 = TRUE) or replaced with 1's (U2 = FALSE)

Value

A correlation or covariance matrix.

Author(s)

<revelle@northwestern.edu >
http://personality-project.org/revelle.html

References

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.
Revelle, W. In preparation) An Introduction to Psychometric Theory with applications in R (http://personality-project.org/r/book/)

See Also

ICLUST.graph,ICLUST.cluster, cluster.fit , VSS, omega

Examples

```
f2 <- matrix(c(.9,.8,.7,rep(0,6),.6,.7,.8),ncol=2)
mod <- factor.model(f2)
round(mod,2)
```

---

factor.residuals *R* = R- F F'*

Description

The basic factor or principal components model is that a correlation or covariance matrix may be reproduced by the product of a factor loading matrix times its transpose. Find the residuals of the original minus the reproduced matrix. Used by factor.fit, VSS, ICLUST, etc.

Usage

```
factor.residuals(r, f)
```

Arguments

r	A correlation matrix
f	A factor model matrix or a list of class loadings

Details

The basic factor equation is ${}_nR_n \approx_n F_{kk}F'_n + U^2$. Residuals are just R* = R - F'F. The residuals should be (but in practice probably rarely are) examined to understand the adequacy of the factor analysis. When doing Factor analysis or Principal Components analysis, one usually continues to extract factors/components until the residuals do not differ from those expected from a random matrix.

Value

rstar is the residual correlation matrix.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

See Also

fa, principal, VSS, ICLUST

Examples

```
fa2 <- fa(Harman74.cor$cov,2,rotate=TRUE)
 fa2resid <- factor.residuals(Harman74.cor$cov,fa2)
 fa2resid[1:4,1:4] #residuals with two factors extracted
 fa4 <- fa(Harman74.cor$cov,4,rotate=TRUE)
 fa4resid <- factor.residuals(Harman74.cor$cov,fa4)
 fa4resid[1:4,1:4] #residuals with 4 factors extracted
```

`factor.rotate` *"Hand" rotate a factor loading matrix*

Description

Given a factor or components matrix, it is sometimes useful to do arbitrary rotations of particular pairs of variables. This supplements the much more powerful rotation package GPArotation and is meant for specific requirements to do unusual rotations.

Usage

```
factor.rotate(f, angle, col1=1, col2=2,plot=FALSE,...)
```

Arguments

f	original loading matrix or a data frame (can be output from a factor analysis function
angle	angle (in degrees!) to rotate
col1	column in factor matrix defining the first variable
col2	column in factor matrix defining the second variable
plot	plot the original (unrotated) and rotated factors
...	parameters to pass to fa.plot

Details

Partly meant as a demonstration of how rotation works, factor.rotate is useful for those cases that require specific rotations that are not available in more advanced packages such as GPArotation. If the plot option is set to TRUE, then the original axes are shown as dashed lines.

The rotation is in degrees counter clockwise.

Value

the resulting rotated matrix of loadings.

Note

For a complete rotation package, see GPArotation

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu >

References

http://personality-project.org/r/book

Examples

```
#using the Harman 24 mental tests, rotate the 2nd and 3rd factors 45 degrees
f4<- fa(Harman74.cor$cov,4,rotate="TRUE")
f4r45 <- factor.rotate(f4,45,2,3)
f4r90 <- factor.rotate(f4r45,45,2,3)
print(factor.congruence(f4,f4r45),digits=3) #poor congruence with original
print(factor.congruence(f4,f4r90),digits=3) #factor 2 and 3 have been exchanged and 3 flipped

#a graphic example
data(Harman23.cor)
f2 <- fa(Harman23.cor$cov,2,rotate="none")
op <- par(mfrow=c(1,2))
cluster.plot(f2,xlim=c(-1,1),ylim=c(-1,1),title="Unrotated ")
f2r <- factor.rotate(f2,-33,plot=TRUE,xlim=c(-1,1),ylim=c(-1,1),title="rotated -33 degrees")
```

```
op <- par(mfrow=c(1,1))
```

---

factor.scores *Various ways to estimate factor scores for the factor analysis model*

Description

A fundamental problem with factor analysis is that although the model is defined at the structural level, it is indeterminate at the data level. This problem of factor indeterminancy leads to alternative ways of estimating factor scores, none of which is ideal. Following Grice (2001) four different methods are available here.

Usage

```
factor.scores(x, f, Phi = NULL, method = c("Thurstone", "tenBerge", "Anderson",
       "Bartlett", "Harman","components"),rho=NULL)
```

Arguments

x	Either a matrix of data if scores are to be found, or a correlation matrix if just the factor weights are to be found.
f	The output from the fa function, or a factor loading matrix.
Phi	If a pattern matrix is provided, then what were the factor intercorrelations. Does not need to be specified if f is the output from the fa function.
method	Which of four factor score estimation procedures should be used. Defaults to "Thurstone" or regression based weights. See details below for the other four methods.
rho	If x is a set of data and rho is specified, then find scores based upon the fa results and the correlations reported in rho. Used when scoring fa.poly results.

Details

Although the factor analysis model is defined at the structural level, it is undefined at the data level. This is a well known but little discussed problem with factor analysis.

Factor scores represent estimates of common part of the variables and should not be thought of as identical to the factors themselves. If a factor scores is thought of as a chop stick stuck into the center of an ice cream cone and factor scores are represented by straws anywhere along the edge of the cone the problem of factor indeterminacy becomes clear, for depending on the shape of the cone, two straws can be negatively correlated with each other. (The imagery is taken from Niels Waller, adapted from Stanley Mulaik). In a very clear discussion of the problem of factor score indeterminacy, Grice (2001) reviews several alternative ways of estimating factor scores and considers weighting schemes that will produce uncorrelated factor score estimates as well as the effect of using course coded (unit weighted) factor weights.

factor.scores uses four different ways of estimate factor scores. In all cases, the factor score estimates are based upon the data matrix, X, times a weighting matrix, W, which weights the observed variables.

- method="Thurstone" finds the regression based weights: $W = R^{-1}F$ where R is the correlation matrix and F is the factor loading matrix.
- method="tenBerge" finds weights such that the correlation between factors for an oblique solution is preserved. Note that formula 8 in Grice has a typo in the formula for C and should be: $L = F\Phi^(1/2)$ $C = R^(- 1/2)L(L'R^(- 1)L)^(- 1/2)$ $W = R^(- 1/2)C\Phi^(1/2)$
- method="Anderson" finds weights such that the factor scores will be uncorrelated: $W = U^{-2}F(F'U^{-2}RU^{-2}F)^{-1/2}$ where U is the diagonal matrix of uniquenesses. The Anderson method works for orthogonal factors only, while the tenBerge method works for orthogonal or oblique solutions.
- method = "Bartlett" finds weights given $W = U^{-2}F(F'U^{-2}F)^{-1}$
- method="Harman" finds weights based upon socalled "idealized" variables: $W = F(t(F)F)^{-1}$.
- method="components" uses weights that are just component loadings.

Value

- scores (the factor scores if the raw data is given)
- weights (the factor weights)

Author(s)

William Revelle

References

Grice, James W.,2001, Computing and evaluating factor scores, Psychological Methods, 6,4, 430-450. (note the typo in equation 8)

ten Berge, Jos M.F., Wim P. Krijnen, Tom Wansbeek and Alexander Shapiro (1999) Some new results on correlation-preserving factor scores prediction methods. Linear Algebra and its Applications, 289, 311-318.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at http://personality-project.org/r/book/

See Also

fa, factor.stats

Examples

```
f3 <- fa(Thurstone)
f3$weights  #just the scoring weights
f5 <- fa(bfi,5)
round(cor(f5$scores,use="pairwise"),2)
#compare to the f5 solution
```

`factor.stats` *Find various goodness of fit statistics for factor analysis and principal components*

Description

Chi square and other goodness of fit statistics are found based upon the fit of a factor or components model to a correlation matrix. Although these statistics are normally associated with a maximum likelihood solution, they can be found for minimal residual (OLS), principal axis, or principal component solutions as well. Primarily called from within these functions, factor.stats can be used by itself. Measures of factorial adequacy and validity follow the paper by Grice, 2001.

Usage

```
fa.stats(r=NULL,f,phi=NULL,n.obs=NA,np.obs=NULL,alpha=.1,fm=NULL)
factor.stats(r=NULL,f,phi=NULL,n.obs=NA,np.obs=NULL,alpha=.1,fm=NULL)
```

Arguments

`r`	A correlation matrix or a data frame of raw data
`f`	A factor analysis loadings matrix or the output from a factor or principal components analysis. In which case the r matrix need not be specified.
`phi`	A factor intercorrelation matrix if the factor solution was oblique.
`n.obs`	The number of observations for the correlation matrix. If not specified, and a correlation matrix is used, chi square will not be reported. Not needed if the input is a data matrix.
`np.obs`	The pairwise number of subjects for each pair in the correlation matrix. This is used for finding observed chi square.
`alpha`	alpha level of confidence intervals for RMSEA
`fm`	flag if components are being given statistics

Details

Combines the goodness of fit tests used in `fa` and principal into one function. If the matrix is singular, will smooth the correlation matrix before finding the fit functions. Now will find the RMSEA (root mean square error of approximation) and the alpha confidence intervals similar to a SEM function. Also reports the root mean square residual.

Chi square is found two ways. The first (STATISTIC) applies the goodness of fit test from Maximum Likelihood objective function (see below). This assumes multivariate normality. The second is the empirical chi square based upon the observed residual correlation matrix and the observed sample size for each correlation. This is found by summing the squared residual correlations time the sample size.

Value

`fit` How well does the factor model reproduce the correlation matrix. (See VSS, ICLUST, and principal for this fit statistic.

`fit.off` how well are the off diagonal elements reproduced? This is just 1 - the relative magnitude of the squared off diagonal residuals to the squared off diagonal original values.

`dof` Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters. Let n=Number of items, nf = number of factors then
$dof = n * (n - 1)/2 - n * nf + nf * (nf - 1)/2$

`objective` value of the function that is minimized by maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is
$f = log(trace((FF' + U2)^{-1} R) - log(|(FF' + U2)^{-1} R|) - n.items.$

`STATISTIC` If the number of observations is specified or found, this is a chi square based upon the objective function, f. Using the formula from factanal(which seems to be Bartlett's test) :
$\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$
Note that this is different from the chi square reported by the sem package which seems to use $\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$

`PVAL` If n.obs > 0, then what is the probability of observing a chisquare this large or larger?

`Phi` If oblique rotations (using oblimin from the GPArotation package or promax) are requested, what is the interfactor correlation.

`R2` The multiple R square between the factors and factor score estimates, if they were to be found. (From Grice, 2001)

`r.scores` The correlations of the factor score estimates, if they were to be found.

`weights` The beta weights to find the factor score estimates

`valid` The validity coffiecient of course coded (unit weighted) factor score estimates (From Grice, 2001)

`score.cor` The correlation matrix of course coded (unit weighted) factor score estimates, if they were to be found, based upon the loadings matrix.

`RMSEA` The Root Mean Square Error of Approximation and the alpha confidence intervals. Based upon the chi square non-centrality parameter. This is found as $\sqrt{f/dof - 1(/ - 1)}$

`rms` The empirically found square root of the squared residuals. This does not require sample size to be specified nor does it make assumptions about normality.

`crms` While the rms uses the number of correlations to find the average, the crms uses the number of degrees of freedom. Thus, there is a penalty for having too complex a model.

Author(s)

William Revelle

References

Grice, James W.,2001, Computing and evaluating factor scores, Psychological Methods, 6,4, 430-450.

See Also

fa with fm="pa" for principal axis factor analysis, fa with fm="minres" for minimum residual factor analysis (default). factor.pa also does principal axis factor analysis, but is deprecated, as is factor.minres for minimum residual factor analysis. See principal for principal components.

Examples

```
v9 <- sim.hierarchical()
f3 <- fa(v9,3)
factor.stats(v9,f3,n.obs=500)
f3o <- fa(v9,3,fm="pa",rotate="Promax")
factor.stats(v9,f3o,n.obs=500)
```

`factor2cluster` *Extract cluster definitions from factor loadings*

Description

Given a factor or principal components loading matrix, assign each item to a cluster corresponding to the largest (signed) factor loading for that item. Essentially, this is a Very Simple Structure approach to cluster definition that corresponds to what most people actually do: highlight the largest loading for each item and ignore the rest.

Usage

```
factor2cluster(loads, cut = 0)
```

Arguments

loads	either a matrix of loadings, or the result of a factor analysis/principal components analyis with a loading component
cut	Extract items with absolute loadings > cut

Details

A factor/principal components analysis loading matrix is converted to a cluster (-1,0,1) definition matrix where each item is assigned to one and only one cluster. This is a fast way to extract items that will be unit weighted to form cluster composites. Use this function in combination with cluster.cor to find the corrleations of these composite scores.

A typical use in the SAPA project is to form item composites by clustering or factoring (see ICLUST, principal), extract the clusters from these results (factor2cluster), and then form the composite correlation matrix using cluster.cor. The variables in this reduced matrix may then be used in multiple R procedures using mat.regress.

The input may be a matrix of item loadings, or the output from a factor analysis which includes a loadings matrix.

Value

a matrix of -1,0,1 cluster definitions for each item.

Author(s)

http://personality-project.org/revelle.html

Maintainer: William Revelle < revelle@northwestern.edu >

References

http://personality-project.org/r/r.vss.html

See Also

cluster.cor, factor2cluster, fa, principal, ICLUST

Examples

```
## Not run:
f  <- factanal(x,4,covmat=Harman74.cor$cov)
factor2cluster(f)
## End(Not run)
#                        Factor1 Factor2 Factor3 Factor4
#VisualPerception              0       1       0       0
#Cubes                         0       1       0       0
#PaperFormBoard                0       1       0       0
#Flags                         0       1       0       0
#GeneralInformation            1       0       0       0
#PargraphComprehension         1       0       0       0
#SentenceCompletion            1       0       0       0
#WordClassification            1       0       0       0
#WordMeaning                   1       0       0       0
#Addition                      0       0       1       0
#Code                          0       0       1       0
#CountingDots                  0       0       1       0
#StraightCurvedCapitals        0       0       1       0
#WordRecognition               0       0       0       1
#NumberRecognition             0       0       0       1
#FigureRecognition             0       0       0       1
#ObjectNumber                  0       0       0       1
#NumberFigure                  0       0       0       1
```

```
#FigureWord                0      0      0      1
#Deduction                 0      1      0      0
#NumericalPuzzles          0      0      1      0
#ProblemReasoning          0      1      0      0
#SeriesCompletion          0      1      0      0
#ArithmeticProblems        0      0      1      0
```

`fisherz` *Fisher r to z and z to r and confidence intervals*

Description

Convert a correlation to a z score or z to r using the Fisher transformation or find the confidence intervals for a specified correlation. r2d converts a correlation to an effect size (Cohen's d) and d2r converts a d into an r.

Usage

```
fisherz(rho)
fisherz2r(z)
r.con(rho,n,p=.95,twotailed=TRUE)
r2t(rho,n)
r2d(rho)
d2r(d)
```

Arguments

`rho`	a Pearson r
`z`	A Fisher z
`n`	Sample size for confidence intervals
`p`	Confidence interval
`twotailed`	Treat p as twotailed p
`d`	an effect size (Cohen's d)

Value

z value corresponding to r (fisherz) \ r corresponding to z (fisherz2r) \ lower and upper p confidence intervals (r.con) \ t with n-2 df (r2t) r corresponding to effect size d or d corresponding to r.

Author(s)

Maintainer: William Revelle `<revelle@northwestern.edu >`

Examples

```
cors <- seq(-.9,.9,.1)
zs <- fisherz(cors)
rs <- fisherz2r(zs)
round(zs,2)
 n <- 30
 r <- seq(0,.9,.1)
 rc <- matrix(r.con(r,n),ncol=2)
 t <- r*sqrt(n-2)/sqrt(1-r^2)
 p <- (1-pt(t,n-2))/2
 r.rc <- data.frame(r=r,z=fisherz(r),lower=rc[,1],upper=rc[,2],t=t,p=p)
 round(r.rc,2)
```

`galton` *Galton's Mid parent child height data*

Description

Two of the earliest examples of the correlation coefficient were Francis Galton's data sets on the relationship between mid parent and child height and the similarity of parent generation peas with child peas. This is the data set for the Galton height.

Usage

```
data(galton)
```

Format

A data frame with 928 observations on the following 2 variables.

`parent` Mid Parent heights (in inches)

`child` Child Height

Details

Female heights were adjusted by 1.08 to compensate for sex differences. (This was done in the original data set)

Source

This is just the galton data set from UsingR, slightly rearranged.

References

Stigler, S. M. (1999). Statistics on the Table: The History of Statistical Concepts and Methods. Harvard University Press. Galton, F. (1886). Regression towards mediocrity in hereditary stature. Journal of the Anthropological Institute of Great Britain and Ireland, 15:246-263. Galton, F. (1869). Hereditary Genius: An Inquiry into its Laws and Consequences. London: Macmillan.

Wachsmuth, A.W., Wilkinson L., Dallal G.E. (2003). Galton's bend: A previously undiscovered nonlinearity in Galton's family stature regression data. The American Statistician, 57, 190-192.

See Also

The other Galton data sets: `heights`, `peas`,`cubits`

Examples

```
data(galton)
describe(galton)
 #show the scatter plot and the lowess fit
pairs.panels(galton,main="Galton's Parent child heights")
#but this makes the regression lines look the same
pairs.panels(galton,lm=TRUE,main="Galton's Parent child heights")
 #better is to scale them
pairs.panels(galton,lm=TRUE,xlim=c(62,74),ylim=c(62,74),main="Galton's Parent child heights")
```

`geometric.mean` *Find the geometric mean of a vector or columns of a data.frame.*

Description

The geometric mean is the nth root of n products or e to the mean log of x. Useful for describing non-normal, i.e., geometric distributions.

Usage

```
geometric.mean(x,na.rm=TRUE)
```

Arguments

`x`	a vector or data.frame
`na.rm`	remove NA values before processing

Details

Useful for teaching how to write functions, also useful for showing the different ways of estimating central tendency.

Value

geometric mean(s) of x or x.df.

Note

Not particularly useful if there are elements that are <= 0.

Author(s)

William Revelle

See Also

harmonic.mean, mean

Examples

```
x <- seq(1,5)
x2 <- x^2
x2[2] <- NA
X <- data.frame(x,x2)
geometric.mean(x)
geometric.mean(x2)
geometric.mean(X)
geometric.mean(X,na.rm=FALSE)
```

---

glb.algebraic *Find the greatest lower bound to reliability.*

Description

The greatest lower bound solves the “educational testing problem". That is, what is the reliability of a test? (See guttman for a discussion of the problem). Although there are many estimates of a test reliability (Guttman, 1945) most underestimate the true reliability of a test.

For a given covariance matrix of items, C, the function finds the greatest lower bound to reliability of the total score using the csdp function from the Rcsdp package.

Usage

```
glb.algebraic(Cov, LoBounds = NULL, UpBounds = NULL)
```

Arguments

Cov	A p * p covariance matrix. Positive definiteness is not checked.
LoBounds	A vector $l = (l_1, \ldots, l_p)$ of length p with lower bounds to the diagonal elements x_i. The default l=(0, . . . , 0) does not imply any constraint, because positive semidefiniteness of the matrix $\tilde{C} + Diag(x)$ implies $0 \leq x_i$
UpBounds	A vector u =(u1, . . . , up) of length p with upper bounds to the diagonal elements xi. The default is u = v.

Details

If C is a p * p-covariance matrix, v = diag(C) its diagonal (i. e. the vector of variances $v_i = c_{ii}$), $\tilde{C} = C - Diag(v)$ is the covariance matrix with 0s substituted in the diagonal and x = the vector $x_1, \ldots, x_n$ the educational testing problem is (see e. g., Al-Homidan 2008)

$$\sum_{i=1}^{p} x_i \to \min$$

s.t.

$$\tilde{C} + Diag(x) \geq 0$$

(i.e. positive semidefinite) and $x_i \leq v_i, i = 1, \ldots, p$. This is the same as minimizing the trace of the symmetric matrix

$$\tilde{C} + diag(x) = \begin{pmatrix} x_1 & c_{12} & \ldots & c_{1p} \\ c_{12} & x_2 & \ldots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1p} & c_{2p} & \ldots & x_p \end{pmatrix}$$

s. t. $\tilde{C} + Diag(x)$ is positive semidefinite and $x_i \leq v_i$.

The greatest lower bound to reliability is

$$\frac{\sum_{ij} \bar{c_{ij}} + \sum_i x_i}{\sum_{ij} c_{ij}}$$

Additionally, function glb.algebraic allows the user to change the upper bounds $x_i \leq v_i$ to $x_i \leq u_i$ and add lower bounds $l_i \leq x_i$.

The greatest lower bound to reliability is applicable for tests with non-homogeneous items. It gives a sharp lower bound to the reliability of the total test score.

Caution: Though glb.algebraic gives exact lower bounds for exact covariance matrices, the estimates from empirical matrices may be strongly biased upwards for small and medium sample sizes.

glb.algebraic is wrapper for a call to function csdp of package Rcsdp (see its documentation).

If Cov is the covariance matrix of subtests/items with known lower bounds, rel, to their reliabilities (e. g. Cronbachs α), LoBounds can be used to improve the lower bound to reliability by setting LoBounds <- rel*diag(Cov).

Changing UpBounds can be used to relax constraints $x_i \leq v_i$ or to fix x_i-values by setting LoBounds[i] < -z; UpBounds[i] <- z.

Value

`glb`	The algebraic greatest lower bound
`solution`	The vector x of the solution of the semidefinite program. These are the elements on the diagonal of C.
`status`	Status of the solution. See documentation of csdp in package Rcsdp. If status is 2 or greater or equal than 4, no glb and solution is returned. If status is not 0, a warning message is generated.
`Call`	The calling string

Author(s)

Andreas Moltner
Center of Excellence for Assessment in Medicine/Baden-Wurttemberg
University of Heidelberg

William Revelle
Department of Psychology
Northwestern University Evanston, Illiniois
http://personality-project.org/revelle.html

References

Al-Homidan S (2008). Semidefinite programming for the educational testing problem. Central European Journal of Operations Research, 16:239-249.

Bentler PM (1972) A lower-bound method for the dimension-free measurement of internal consistency. Soc Sci Res 1:343-357.

Fletcher R (1981) A nonlinear programming problem in statistics (educational testing). SIAM J Sci Stat Comput 2:257-267.

Shapiro A, ten Berge JMF (2000). The asymptotic bias of minimum trace factor analysis, with applications to the greatest lower bound to reliability. Psychometrika, 65:413-425.

ten Berge, Socan G (2004). The greatest bound to reliability of a test and the hypothesis of unidimensionality. Psychometrika, 69:613-625.

See Also

For an alternative estimate of the greatest lower bound, see `glb.fa`. For multiple estimates of reliablity, see `guttman`

Examples

```
Cv<-matrix(c(215, 64, 33, 22,
              64, 97, 57, 25,
              33, 57,103, 36,
              22, 25, 36, 77),ncol=4)

Cv                    # covariance matrix of a test with 4 subtests
Cr<-cov2cor(Cv)       # Correlation matrix of tests
if(!require(Rcsdp)) {print("Rcsdp must be installed to find the glb.algebraic")} else {
 glb.algebraic(Cv)     # glb of total score
glb.algebraic(Cr)      # glb of sum of standardized scores

 w<-c(1,2,2,1)         # glb of weighted total score
 glb.algebraic(diag(w) %*% Cv %*% diag(w))
alphas <- c(0.8,0,0,0) # Internal consistency of first test is known

glb.algebraic(Cv,LoBounds=alphas*diag(Cv))

                      # Fix all diagonal elements to 1 but the first:
```

```
lb <- glb.algebraic(Cr,LoBounds=c(0,1,1,1),UpBounds=c(1,1,1,1))
lb$solution[1]       # should be the same as the squared mult. corr.
smc(Cr)[1]
}
```

Gleser   *Example data from Gleser, Cronbach and Rajaratnam (1965) to show basic principles of generalizability theory.*

Description

Gleser, Cronbach and Rajaratnam (1965) discuss the estimation of variance components and their ratios as part of their introduction to generalizability theory. This is a adaptation of their "illustrative data for a completely matched G study" (Table 3). 12 patients are rated on 6 symptoms by two judges. Components of variance are derived from the ANOVA.

Usage

```
data(Gleser)
```

Format

A data frame with 12 observations on the following 12 variables. J item by judge:

`J11` a numeric vector

`J12` a numeric vector

`J21` a numeric vector

`J22` a numeric vector

`J31` a numeric vector

`J32` a numeric vector

`J41` a numeric vector

`J42` a numeric vector

`J51` a numeric vector

`J52` a numeric vector

`J61` a numeric vector

`J62` a numeric vector

Details

Generalizability theory is the application of a components of variance approach to the analysis of reliability. Given a G study (generalizability) the components are estimated and then may be used in a D study (Decision). Different ratios are formed as appropriate for the particular D study.

Source

Gleser, G., Cronbach, L., and Rajaratnam, N. (1965). Generalizability of scores influenced by multiple sources of variance. Psychometrika, 30(4):395-418. (Table 3, rearranged to show increasing patient severity and increasing item severity.

References

Gleser, G., Cronbach, L., and Rajaratnam, N. (1965). Generalizability of scores influenced by multiple sources of variance. Psychometrika, 30(4):395-418.

Examples

```
#Find the MS for each component:
#First, stack the data
data(Gleser)
stack.g <- stack(Gleser)
st.gc.df <- data.frame(stack.g,Persons=rep(letters[1:12],12),
Items=rep(letters[1:6],each=24),Judges=rep(letters[1:2],each=12))
#now do the ANOVA
anov <- aov(values ~ (Persons*Judges*Items),data=st.gc.df)
summary(anov)
```

`Gorsuch` *Example data set from Gorsuch (1997) for an example factor extension.*

Description

Gorsuch (1997) suggests an alternative to the classic Dwyer (1937) factor extension technique. This data set is taken from that article. Useful for comparing `link{fa.extension}` with and without the correct=TRUE option.

Usage

```
data(Gorsuch)
```

Details

Gorsuc (1997) suggested an alternative model for factor extension. His method is appropriate for the case of repeated variables. This is handled in `link{fa.extension}` with correct=FALSE

Source

Richard L. Gorsuch (1997) New Procedure for Extension Analysis in Exploratory Factor Analysis. Educational and Psychological Measurement, 57, 725-740.

References

Dwyer, Paul S. (1937), The determination of the factor loadings of a given test from the known factor loadings of other tests. Psychometrika, 3, 173-178

Examples

```
data(Gorsuch)

Ro <- Gorsuch[1:6,1:6]
Roe <- Gorsuch[1:6,7:10]
fo <- fa(Ro,2,rotate="none")
fa.extension(Roe,fo,correct=FALSE)
```

`Harman` *Two data sets from Harman (1967). 9 cognitive variables from Holzinger and 8 emotional variables from Burt*

Description

Two classic data sets reported by Harman (1967) are 9 psychological (cognitive) variables taken from Holzinger and 8 emotional variables taken from Burt. Both of these are used for tests and demonstrations of various factoring algortithms.

Usage

```
data(Harman)
```

Details

- Harman.Holzinger: 9 x 9 correlation matrix of ability tests, N = 696.
- Harman.Burt: a 8 x 8 correlation matrix of “emotional" items. N = 172

Harman.Holzinger. The nine psychological variables from Harman (1967, p 244) are taken from unpublished class notes of K.J. Holzinger with 696 participants. This is a subset of 12 tests with 4 factors. It is yet another nice example of a bifactor solution. Bentler (2007) uses this data set to discuss reliablity analysis. The data show a clear bifactor structure and are a nice example of the various estimates of reliability included in the `omega` function. Should not be confused with the `Holzinger` or `Holzinger.9` data sets in `bifactor`.

Harman.Burt. Eight “emotional" variables are taken from Harman (1967, p 164) who in turn adapted them from Burt (1939). They are said be from 172 normal children aged nine to twelve. As pointed out by Harman, this correlation matrix is singular and has squared multiple correlations > 1. Because of this problem, it is a nice test case for various factoring algorithms. (For instance, omega will issue warning messages for fm="minres" or fm="pa" but will fail for fm="ml".)

The Burt data set probably has a typo in the original correlation matrix. Changing the Sorrow-Tenderness correlation from .87 to .81 makes the correlation positive definite.

As pointed out by Jan DeLeeuw, the Burt data set is a subset of 8 variables from the original 11 reported by Burt in 1915. That matrix has the same problem. See `burt`.

Other example data sets that are useful demonstrations of factor analysis are the seven bifactor examples in `bifactor` and the 24 ability measures in `Harman74.cor`

There are several other Harman examples in the psych package (i.e., Harman.8) as well as in the dataseta and GPArotation packages. The Harman 24 mental tests problem is in the basic datasets package at Harman74.cor.

Source

Harman (1967 p 164 and p 244.)

References

Harman, Harry Horace (1967), Modern factor analysis. Chicago, University of Chicago Press.

P.Bentler. Covariance structure models for maximal reliability of unit-weighted composites. In Handbook of latent variable and related models, pages 1–17. North Holland, 2007.

Burt, C.General and Specific Factors underlying the Primary Emotions. Reports of the British Association for the Advancement of Science, 85th meeting, held in Manchester, September 7-11, 1915. London, John Murray, 1916, p. 694-696 (retrieved from the web at http://www.biodiversitylibrary.org/item/95822#790)

See Also

See also the original `burt` data set

Examples

```
data(Harman)
cor.plot(Harman.Holzinger)
cor.plot(Harman.Burt)
smc(Harman.Burt)  #note how this produces impossible results
```

`Harman.5` *5 socio-economic variables from Harman (1967)*

Description

Harman (1967) uses 5 socio-economic variables for demonstrations of principal components and factor analysis. This example is used in the SAS manual for Proc Factor as well.

Usage

```
data(Harman.5)
```

Format

A data frame with 12 observations on the following 5 variables.

`population` a numeric vector

`schooling` a numeric vector

`employment` a numeric vector

`professional` a numeric vector

`housevalue` a numeric vector

Details

Harman reports that the data "were taken (not entirely arbitrarily) from a study of the Los Angeles Standard Metropolitan Statistical Area. The twelve individuals are used in the examples are census tracts." (p 13).

Source

Harman, Harry Horace (1967), Modern factor analysis. Chicago, University of Chicago Press.

References

SAS users manual, chapter 26: pages 1123-1192

Examples

```
data(Harman.5)
if(require('GPArotation')){
pc2 <- principal(Harman.5,2,scores=TRUE)
pc2$residual
biplot(pc2,main="Biplot of the Harman 5 socio-demographic variables") }
```

`Harman.8` *Correlations of eight physical variables (from Harman, 1966)*

Description

A classic data set from Harman (1976) reporting the correlations of eight physical variables. Used by Harman for demonstrations of factor analysis (both principal axis and minimum residual).

Usage

```
data(Harman.8)
```

Format

The format is: num [1:8, 1:8] 1 0.846 0.805 0.859 0.473 0.398 0.301 0.382 0.846 1 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:8] "Height" "Arm span" "Length of forearm" "Length of lower leg" ... ..$: chr [1:8] "V1" "V2" "V3" "V4" ...

Details

The Eight Physical Variables problem is taken from Harman (1976) and represents the correlations between eight physical variables for 305 girls. The two correlated clusters represent four measures of "lankiness" and then four measures of "stockiness". The original data were selected from 17 variables reported in an unpublished dissertation by Mullen (1939).

Variable 6 ("Bitrochanteric diamter") is the distance between the outer points of the hips.

The row names match the original Harman paper, the column names have been abbreviated.

The `fa` solution for principal axes (fm="pa") matches the reported minres solution, while the fm="minres" does not.

For those interested in teaching examples using various body measurements, see the body data set in the gclus package.

There are several other Harman examples in the psych package as well as in the dataseta and and GPArotation packages. The Harman 24 mental tests problem is in the basic datasets package at Harman74.cor.

Source

H. Harman and W.Jones. (1966) Factor analysis by minimizing residuals (minres). Psychometrika, 31(3):351-368.

References

Harman, Harry Horace (1976) Modern factor analysis, 3d ed., rev, University of Chicago Press. Chicago.

Harman, Harry Horace and Jones, W. (1966) Factor analysis by minimizing residuals (minres). Psychometrika, 31(3):351-368.

See Also

`Harman`, `Harman.political` and `Harman74.cor`

Examples

```
data(Harman.8)
cor.plot(Harman.8)
fa(Harman.8,2,rotate="none")  #the minres solution
fa(Harman.8,2,rotate="none",fm="pa") #the principal axis solution
```

`Harman.political` *Eight political variables used by Harman (1967) as example 8.17*

Description

Another one of the many Harman (1967) data sets. This contains 8 political variables taken over 147 election areas. The principal factor method with SMCs as communalities match those of table 8.18. The data are used by Dziubian and Shirkey as an example of the Kaiser-Meyer-Olkin test of factor adequacy.

Usage

```
data(Harman.political)
```

Format

The format is: num [1:8, 1:8] 1 0.84 0.62 -0.53 0.03 0.57 -0.33 -0.63 0.84 1 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:8] "Lewis" "Roosevelt" "Party Voting" "Median Rental" ... ..$: chr [1:8] "Lewis" "Roosevelt" "Party Voting" "Median Rental" ...

Details

The communalities from the original table are not included. They are .52, 1.00, .78, .82, .36, .80, .63, and .97

Source

Harman, Harry Horace (1976) Modern factor analysis, 3d ed., rev, University of Chicago Press. Chicago. p 166.

References

Dziuban, Charles D. and Shirkey, Edwin C. (1974) When is a correlation matrix appropriate for factor analysis? Some decision rules. Psychological Bulletin, 81 (6) 358 - 361.

Examples

```
data(Harman.political)
KMO(Harman.political)
```

harmonic.mean *Find the harmonic mean of a vector, matrix, or columns of a data.frame*

Description

The harmonic mean is merely the reciprocal of the arithmetic mean of the reciprocals.

Usage

```
harmonic.mean(x,na.rm=TRUE)
```

Arguments

x	a vector, matrix, or data.frame
na.rm	na.rm=TRUE remove NA values before processing

Details

Included as an example for teaching about functions. As well as for a discussion of how to estimate central tendencies.

Value

The harmonic mean(s)

Note

Included as a simple demonstration of how to write a function

Examples

```
x <- seq(1,5)
x2 <- x^2
x2[2] <- NA
X <- data.frame(x,x2)
harmonic.mean(x)
harmonic.mean(x2)
harmonic.mean(X)
harmonic.mean(X,FALSE)
```

headTail *Combine calls to head and tail*

Description

A quick way to show the first and last n lines of a data.frame, matrix, or a text object. Just a pretty call to `head` and `tail`

Usage

```
headTail(x,hlength=4,tlength=4,digits=2,ellipsis=TRUE)
headtail(x,hlength=4,tlength=4,digits=2,ellipsis=TRUE)
topBottom(x,hlength=4,tlength=4,digits=2)
```

Arguments

`x`	A matrix or data frame or free text
`hlength`	The number of lines at the beginning to show
`tlength`	The number of lines at the end to show
`digits`	Round off the data to digits
`ellipsis`	Separate the head and tail with dots (ellipsis)

Value

The first hlength and last tlength lines of a matrix or data frame with an ellipsis in between. If the input is neither a matrix nor data frame, the output will be the first hlength and last tlength lines.

topBottom is just a call to headTail with ellipsis = FALSE and returning a matrix output.

See Also

`head` and `tail`

Examples

```
headTail(iqitems[1:5],4,8)
```

heights *A data.frame of the Galton (1888) height and cubit data set.*

Description

Francis Galton introduced the 'co-relation' in 1888 with a paper discussing how to measure the relationship between two variables. His primary example was the relationship between height and forearm length. The data table (`cubits`) is taken from Galton (1888). Unfortunately, there seem to be some errors in the original data table in that the marginal totals do not match the table.

The data frame, `heights`, is converted from this table using `table2df`.

Usage

```
data(heights)
```

Format

A data frame with 348 observations on the following 2 variables.

`height` Height in inches

`cubit` Forearm length in inches

Details

Sir Francis Galton (1888) published the first demonstration of the correlation coefficient. The regression (or reversion to mediocrity) of the height to the length of the left forearm (a cubit) was found to .8. The original table `cubits` is taken from Galton (1888). There seem to be some errors in the table as published in that the row sums do not agree with the actual row sums. These data are used to create a matrix using `table2matrix` for demonstrations of analysis and displays of the data.

Source

Galton (1888)

References

Galton, Francis (1888) Co-relations and their measurement. Proceedings of the Royal Society. London Series,45,135-145,

See Also

`table2matrix`, `table2df`, `cubits`, `ellipses`, `galton`

Examples

```
data(heights)
ellipses(heights,n=1,main="Galton's co-relation data set")
```

ICC — *Intraclass Correlations (ICC1, ICC2, ICC3 from Shrout and Fleiss)*

Description

The Intraclass correlation is used as a measure of association when studying the reliability of raters. Shrout and Fleiss (1979) outline 6 different estimates, that depend upon the particular experimental design. All are implemented and given confidence limits.

Usage

```
ICC(x,missing=TRUE,alpha=.05)
```

Arguments

`x`	a matrix or dataframe of ratings
`missing`	if TRUE, remove missing data – work on complete cases only
`alpha`	The alpha level for significance for finding the confidence intervals

Details

Shrout and Fleiss (1979) consider six cases of reliability of ratings done by k raters on n targets.

ICC1: Each target is rated by a different judge and the judges are selected at random. (This is a one-way ANOVA fixed effects model and is found by (MSB- MSW)/(MSB+ (nr-1)*MSW))

ICC2: A random sample of k judges rate each target. The measure is one of absolute agreement in the ratings. Found as (MSB- MSE)/(MSB + (nr-1)*MSE + nr*(MSJ-MSE)/nc)

ICC3: A fixed set of k judges rate each target. There is no generalization to a larger population of judges. (MSB - MSE)/(MSB+ (nr-1)*MSE)

Then, for each of these cases, is reliability to be estimated for a single rating or for the average of k ratings? (The 1 rating case is equivalent to the average intercorrelation, the k rating case to the Spearman Brown adjusted reliability.)

ICC1 is sensitive to differences in means between raters and is a measure of absolute agreement.

ICC2 and ICC3 remove mean differences between judges, but are sensitive to interactions of raters by judges. The difference between ICC2 and ICC3 is whether raters are seen as fixed or random effects.

ICC1k, ICC2k, ICC3K reflect the means of k raters.

The intraclass correlation is used if raters are all of the same "class". That is, there is no logical way of distinguishing them. Examples include correlations between pairs of twins, correlations between raters. If the variables are logically distinguishable (e.g., different items on a test), then the more typical coefficient is based upon the inter-class correlation (e.g., a Pearson r) and a statistic such as `alpha` or `omega` might be used.

Value

`results`	A matrix of 6 rows and 8 columns, including the ICCs, F test, p values, and confidence limits
`summary`	The anova summary table
`stats`	The anova statistics
`MSW`	Mean Square Within based upon the anova

Note

The results for the Lower and Upper Bounds for ICC(2,k) do not match those of SPSS 9 or 10, but do match the definitions of Shrout and Fleiss. SPSS seems to have been using the formula in McGraw and Wong, but not the errata on p 390. They seem to have fixed it in more recent releases (15).

Starting with psych 1.4.2, the confidence intervals are based upon (1-alpha)% at both tails of the confidence interval. This is in agreement with Shrout and Fleiss. Prior to 1.4.2 the confidence intervals were (1-alpha/2)%.

Author(s)

William Revelle

References

Shrout, Patrick E. and Fleiss, Joseph L. Intraclass correlations: uses in assessing rater reliability. Psychological Bulletin, 1979, 86, 420-3428.

McGraw, Kenneth O. and Wong, S. P. (1996), Forming inferences about some intraclass correlation coefficients. Psychological Methods, 1, 30-46. + errata on page 390.

Revelle, W. (in prep) An introduction to psychometric theory with applications in R. Springer. (working draft available at http://personality-project.org/r/book/

Examples

```
sf <- matrix(c(9,    2,   5,    8,
6,    1,   3,    2,
8,    4,   6,    8,
7,    1,   2,    6,
10,   5,   6,    9,
6,   2,   4,    7),ncol=4,byrow=TRUE)
colnames(sf) <- paste("J",1:4,sep="")
rownames(sf) <- paste("S",1:6,sep="")
sf  #example from Shrout and Fleiss (1979)
ICC(sf)
```

iclust — *iclust: Item Cluster Analysis – Hierarchical cluster analysis using psychometric principles*

Description

A common data reduction technique is to cluster cases (subjects). Less common, but particularly useful in psychological research, is to cluster items (variables). This may be thought of as an alternative to factor analysis, based upon a much simpler model. The cluster model is that the correlations between variables reflect that each item loads on at most one cluster, and that items that load on those clusters correlate as a function of their respective loadings on that cluster and items that define different clusters correlate as a function of their respective cluster loadings and the intercluster correlations. Essentially, the cluster model is a Very Simple Structure factor model of complexity one (see VSS).

This function applies the iclust algorithm to hierarchically cluster items to form composite scales. Clusters are combined if coefficients alpha and beta will increase in the new cluster.

Alpha, the mean split half correlation, and beta, the worst split half correlation, are estimates of the reliability and general factor saturation of the test. (See also the omega function to estimate McDonald's coeffients ω_h and ω_t)

Usage

```
iclust(r.mat, nclusters=0, alpha=3, beta=1, beta.size=4, alpha.size=3,
correct=TRUE,correct.cluster=TRUE, reverse=TRUE, beta.min=.5, output=1,
digits=2,labels=NULL,cut=0, n.iterations =0, title="ICLUST", plot=TRUE,
weighted=TRUE,cor.gen=TRUE,SMC=TRUE,purify=TRUE,diagonal=FALSE)

ICLUST(r.mat, nclusters=0, alpha=3, beta=1, beta.size=4, alpha.size=3,
correct=TRUE,correct.cluster=TRUE, reverse=TRUE, beta.min=.5, output=1,
digits=2,labels=NULL,cut=0,n.iterations = 0,title="ICLUST",plot=TRUE,
weighted=TRUE,cor.gen=TRUE,SMC=TRUE,purify=TRUE,diagonal=FALSE)


#iclust(r.mat)    #use all defaults
#iclust(r.mat,nclusters =3)    #use all defaults and if possible stop at 3 clusters
#ICLUST(r.mat, output =3)     #long output shows clustering history
#ICLUST(r.mat, n.iterations =3)  #clean up solution by item reassignment
```

Arguments

r.mat	A correlation matrix or data matrix/data.frame. (If r.mat is not square i.e, a correlation matrix, the data are correlated using pairwise deletion.
nclusters	Extract clusters until nclusters remain (default will extract until the other criteria are met or 1 cluster, whichever happens first). See the discussion below for alternative techniques for specifying the number of clusters.

alpha	Apply the increase in alpha criterion (0) never or for (1) the smaller, 2) the average, or 3) the greater of the separate alphas. (default = 3)
beta	Apply the increase in beta criterion (0) never or for (1) the smaller, 2) the average, or 3) the greater of the separate betas. (default =1)
beta.size	Apply the beta criterion after clusters are of beta.size (default = 4)
alpha.size	Apply the alpha criterion after clusters are of size alpha.size (default =3)
correct	Correct correlations for reliability (default = TRUE)
correct.cluster	Correct cluster -sub cluster correlations for reliability of the sub cluster , default is TRUE))
reverse	Reverse negative keyed items (default = TRUE
beta.min	Stop clustering if the beta is not greater than beta.min (default = .5)
output	1) short, 2) medium, 3) long output (default =1)
labels	vector of item content or labels. If NULL, then the colnames are used. If FALSE, then labels are V1 .. Vn
cut	sort cluster loadings > absolute(cut) (default = 0)
n.iterations	iterate the solution n.iterations times to "purify" the clusters (default = 0)
digits	Precision of digits of output (default = 2)
title	Title for this run
plot	Should ICLUST.rgraph be called automatically for plotting (requires Rgraphviz default=TRUE)
weighted	Weight the intercluster correlation by the size of the two clusters (TRUE) or do not weight them (FALSE)
cor.gen	When correlating clusters with subclusters, base the correlations on the general factor (default) or general + group (cor.gen=FALSE)
SMC	When estimating cluster-item correlations, use the smcs as the estimate of an item communality (SMC=TRUE) or use the maximum correlation (SMC=FALSE).
purify	Should clusters be defined as the original groupings (purify = FAlSE) or by the items with the highest loadings on those original clusters? (purify = TRUE)
diagonal	Should the diagonal be included in the fit statistics. The default is not to include it. Prior to 1.2.8, the diagonal was included.

Details

Extensive documentation and justification of the algorithm is available in the original MBR 1979 http://personality-project.org/revelle/publications/iclust.pdf paper. Further discussion of the algorithm and sample output is available on the personality-project.org web page: http://personality-project.org/r/r.ICLUST.html

The results are best visualized using ICLUST.graph, the results of which can be saved as a dot file for the Graphviz program. http://www.graphviz.org/. The iclust.diagram is called automatically to produce cluster diagrams. The resulting diagram is not quite as pretty as what can be achieved in dot code but is quite adequate if you don't want to use an external graphics program. With the installation of Rgraphviz, ICLUST can also provide cluster graphs.

A common problem in the social sciences is to construct scales or composites of items to measure constructs of theoretical interest and practical importance. This process frequently involves administering a battery of items from which those that meet certain criteria are selected. These criteria might be rational, empirical,or factorial. A similar problem is to analyze the adequacy of scales that already have been formed and to decide whether the putative constructs are measured properly. Both of these problems have been discussed in numerous texts, as well as in myriad articles. Proponents of various methods have argued for the importance of face validity, discriminant validity, construct validity, factorial homogeneity, and theoretical importance.

Revelle (1979) proposed that hierachical cluster analysis could be used to estimate a new coefficient (beta) that was an estimate of the general factor saturation of a test. More recently, Zinbarg, Revelle, Yovel and Li (2005) compared McDonald's Omega to Chronbach's alpha and Revelle's beta. They conclude that ω_h hierarchical is the best estimate. An algorithm for estimating `omega` is available as part of this package.

Revelle and Zinbarg (2009) discuss alpha, beta, and omega, as well as other estimates of reliability.

The original ICLUST program was written in FORTRAN to run on CDC and IBM mainframes and was then modified to run in PC-DOS. The R version of iclust is a completely new version written for the psych package. Please email me if you want help with this version of iclust or if you desire more features.

A requested feature (not yet available) is to specify certain items as forming a cluster. That is, to do confirmatory cluster analysis.

The program currently has three primary functions: cluster, loadings, and graphics.

In June, 2009, the option of weighted versus unweighted beta was introduced. Unweighted beta calculates beta based upon the correlation between two clusters, corrected for test length using the Spearman-Brown prophecy formala, while weighted beta finds the average interitem correlation between the items within two clusters and then finds beta from this. That is, for two clusters A and B of size N and M with between average correlation rb, weighted beta is (N+M)^2 rb/(Va +Vb + 2Cab). Raw (unweighted) beta is 2rab/(1+rab) where rab = Cab/sqrt(VaVb). Weighted beta seems a more appropriate estimate and is now the default. Unweighted beta is still available for consistency with prior versions.

Also modified in June, 2009 was the way of correcting for item overlap when calculating the cluster-subcluster correlations for the graphic output. This does not affect the final cluster solution, but does produce slightly different path values. In addition, there are two ways to solve for the cluster - subcluster correlation.

Given the covariance between two clusters, Cab with average rab = Cab/(N*M), and cluster variances Va and Vb with Va = N + N*(N-1)*ra then the correlation of cluster A with the combined cluster AB is either

a) ((N^2)ra + Cab)/sqrt(Vab*Va) (option cor.gen=TRUE) or b) (Va - N + Nra + Cab)/sqrt(Vab*Va) (option cor.gen=FALSE)

The default is to use cor.gen=TRUE.

Although iclust will give what it thinks is the best solution in terms of the number of clusters to extract, the user will sometimes disagree. To get more clusters than the default solution, just set the nclusters parameter to the number desired. However, to get fewer than meet the alpha and beta criteria, it is sometimes necessary to set alpha=0 and beta=0 and then set the nclusters to the desired number.

Clustering 24 tests of mental ability

A sample output using the 24 variable problem by Harman can be represented both graphically and in terms of the cluster order. The default is to produce graphics using the `diagram` functions. An alternative is to use the Rgraphviz package (from BioConductor). Because this package is sometimes hard to install, there is an alternative option (`ICLUST.graph` to write dot language instructions for subsequent processing. This will create a graphic instructions suitable for any viewing program that uses the dot language. `ICLUST.rgraph` produces the dot code for Graphviz. Somewhat lower resolution graphs with fewer options are available in the `ICLUST.rgraph` function which requires Rgraphviz. Dot code can be viewed directly in Graphviz or can be tweaked using commercial software packages (e.g., OmniGraffle)

Note that for the Harman 24 variable problem, with the default parameters, the data form one large cluster. (This is consistent with the Very Simple Structure (`VSS`) output as well, which shows a clear one factor solution for complexity 1 data.)

An alternative solution is to ask for a somewhat more stringent set of criteria and require an increase in the size of beta for all clusters greater than 3 variables. This produces a 4 cluster solution.

It is also possible to use the original parameter settings, but ask for a 4 cluster solution.

At least for the Harman 24 mental ability measures, it is interesting to compare the cluster pattern matrix with the oblique rotation solution from a factor analysis. The factor congruence of a four factor oblique pattern solution with the four cluster solution is > .99 for three of the four clusters and > .97 for the fourth cluster. The cluster pattern matrix (returned as an invisible object in the output)

In September, 2012, the fit statistics (pattern fit and cluster fit) were slightly modified to (by default) not consider the diagonal (diagonal=FALSE). Until then, the diagonal was included in the cluster fit statistics. The pattern fit is analogous to factor analysis and is based upon the model = P x Structure where Structure is Pattern * Phi. Then R* = R - model and fit is the ratio of sum(r*^2)/sum(r^2) for the off diagonal elements.

Value

`title`	Name of this analysis
`results`	A list containing the step by step cluster history, including which pair was grouped, what were the alpha and betas of the two groups and of the combined group. Note that the alpha values are “standardized alphas” based upon the correlation matrix, rather than the raw alphas that will come from `scoreItems` The print.psych and summary.psych functions will print out just the must important results.
`corrected`	The raw and corrected for alpha reliability cluster intercorrelations.
`clusters`	a matrix of -1,0, and 1 values to define cluster membership.
`purified`	A list of the cluster definitions and cluster loadings of the purified solution. These are sorted by importance (the eigenvalues of the clusters). The cluster membership from the original (O) and purified (P) clusters are indicated along with the cluster structure matrix. These item loadings are the same as those found by the `scoreItems` function and are found by correcting the item-cluster correlation for item overlap by summing the item-cluster covariances with all except that item and then adding in the smc for that item. These resulting correlations are then corrected for scale reliability.

To show just the most salient items, use the cutoff option in `print.psych`

`cluster.fit, structure.fit, pattern.fit`
: There are a number of ways to evaluate how well any factor or cluster matrix reproduces the original matrix. Cluster fit considers how well the clusters fit if only correlations with clusters are considered. Structure fit evaluates R = CC' while pattern fit evaluate R = C inverse (phi) C' where C is the cluster loading matrix, and phi is the intercluster correlation matrix.

`pattern`
: The pattern matrix loadings. Pattern is just C inverse (Phi). The pattern matrix is conceptually equivalent to that of a factor analysis, in that the pattern coefficients are b weights of the cluster to the variables, while the normal cluster loadings are correlations of the items with the cluster. The four cluster and four factor pattern matrices for the Harman problem are very similar.

Note

iclust draws graphical displays with or without using Rgraphiviz. Because of difficulties installing Rgraphviz on many systems, the default it not even try using it. With the introduction of the `diagram` functions, iclust now draws using iclust.diagram which is not as pretty as using Rgraphviz, but more stable. However, Rgraphviz can be used by using `ICLUST.rgraph` to produces slightly better graphics. It is also possible to export dot code in the dot language for further massaging of the graphic. This may be done using `ICLUST.graph`. This last option is probably preferred for nice graphics which can be massaged in any dot code program (e.g., graphviz (http://graphviz.org) or a commercial program such as OmniGraffle.

To view the cluster structure more closely, it is possible to save the graphic output as a pdf and then magnify this using a pdf viewer. This is useful when clustering a large number of variables.

In order to sort the clusters by cluster loadings, use `iclust.sort`.

Author(s)

William Revelle

References

Revelle, W. Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 1979, 14, 57-74.

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 2009.

http://personality-project.org/revelle/publications/iclust.pdf
See also more extensive documentation at http://personality-project.org/r/r.ICLUST.html and
Revelle, W. (in prep) An introduction to psychometric theory with applications in R. To be published by Springer. (working draft available at http://personality-project.org/r/book/

See Also

`iclust.sort`, `ICLUST.graph`, `ICLUST.cluster`, `cluster.fit` , `VSS`, `omega`

Examples

```
test.data <- Harman74.cor$cov
ic.out <- iclust(test.data,title="ICLUST of the Harman data")
summary(ic.out)

#use all defaults and stop at 4 clusters
ic.out4 <- iclust(test.data,nclusters =4,title="Force 4 clusters")
summary(ic.out4)
ic.out1 <- iclust(test.data,beta=3,beta.size=3)  #use more stringent criteria
ic.out  #more complete output
plot(ic.out4)    #this shows the spatial representation
#use a dot graphics viewer on the out.file
dot.graph <- ICLUST.graph(ic.out,out.file="test.ICLUST.graph.dot")
 #show the equivalent of a factor solution
fa.diagram(ic.out4$pattern,Phi=ic.out4$Phi,main="Pattern taken from iclust")
```

`ICLUST.cluster` *Function to form hierarchical cluster analysis of items*

Description

The guts of the `ICLUST` algorithm. Called by `ICLUST` See ICLUST for description.

Usage

```
ICLUST.cluster(r.mat, ICLUST.options,smc.items)
```

Arguments

`r.mat`	A correlation matrix
`ICLUST.options`	A list of options (see `ICLUST`)
`smc.items`	passed from the main program to speed up processing

Details

See `ICLUST`

Value

A list of cluster statistics, described more fully in `ICLUST`

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Note

Although the main code for ICLUST is here in ICLUST.cluster, the more extensive documentation is for ICLUST.

Author(s)

William Revelle

References

Revelle, W. 1979, Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 14, 57-74. http://personality-project.org/revelle/publications/iclust.pdf

See also more extensive documentation at http://personality-project.org/r/r.ICLUST.html

See Also

ICLUST.graph,ICLUST, cluster.fit , VSS, omega

---

iclust.diagram *Draw an ICLUST hierarchical cluster structure diagram*

Description

Given a cluster structure determined by ICLUST, create a graphic structural diagram using graphic functions in the psych package To create dot code to describe the ICLUST output with more precision, use ICLUST.graph. If Rgraphviz has been successfully installed, the alternative is to use ICLUST.rgraph.

Usage

```
iclust.diagram(ic, labels = NULL, short = FALSE, digits = 2, cex = NULL, min.size = NULL,
     e.size =1,colors=c("black","blue"),
     main = "ICLUST diagram",cluster.names=NULL,marg=c(.5,.5,1.5,.5))
```

Arguments

ic	Output from ICLUST
labels	labels for variables (if not specified as rownames in the ICLUST output
short	if short=TRUE, variable names are replaced with Vn
digits	Round the path coefficients to digits accuracy
cex	The standard graphic control parameter for font size modifications. This can be used to make the labels bigger or smaller than the default values.
min.size	Don't provide statistics for clusters less than min.size
e.size	size of the ellipses with the cluster statistics.

colors	postive and negative
main	The main graphic title
cluster.names	Normally, clusters are named sequentially C1 ... Cn. If cluster.names are specified, then these values will be used instead.
marg	Sets the margins to be narrower than the default values. Resets them upon return

Details

iclust.diagram provides most of the power of ICLUST.rgraph without the difficulties involved in installing Rgraphviz. It is called automatically from ICLUST.

Following a request by Michael Kubovy, cluster.names may be specified to replace the normal C1 ... Cn names.

If access to a dot language graphics program is available, it is probably better to use the iclust.graph function to get dot output for offline editing.

Value

Graphical output summarizing the hierarchical cluster structure. The graph is drawn using the diagram functions (e.g., dia.curve, dia.arrow, dia.rect, dia.ellipse) created as a work around to Rgraphviz.

Note

Suggestions for improving the graphic output are welcome.

Author(s)

William Revelle

References

Revelle, W. Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 1979, 14, 57-74.

See Also

ICLUST

Examples

```
v9 <- sim.hierarchical()
v9c <- ICLUST(v9)
test.data <- Harman74.cor$cov
ic.out <- ICLUST(test.data)
#now show how to relabel clusters
ic.bfi <- iclust(bfi[1:25],beta=3) #find the clusters
cluster.names <- rownames(ic.bfi$results) #get the old names
#change the names to the desired ones
cluster.names[c(16,19,18,15,20)] <- c("Neuroticism","Extra-Open","Agreeableness",
```

```
        "Conscientiousness","Open")
#now show the new names
iclust.diagram(ic.bfi,cluster.names=cluster.names,min.size=4,e.size=1.75)
```

ICLUST.graph *create control code for ICLUST graphical output*

Description

Given a cluster structure determined by ICLUST, create dot code to describe the ICLUST output. To use the dot code, use either http://www.graphviz.org/ Graphviz or a commercial viewer (e.g., OmniGraffle). This function parallels ICLUST.rgraph which uses Rgraphviz.

Usage

```
ICLUST.graph(ic.results, out.file,min.size=1, short = FALSE,labels=NULL,
size = c(8, 6), node.font = c("Helvetica", 14), edge.font = c("Helvetica", 12),
 rank.direction=c("RL","TB","LR","BT"), digits = 2, title = "ICLUST", ...)
```

Arguments

ic.results	output list from ICLUST
out.file	name of output file (defaults to console)
min.size	draw a smaller node (without all the information) for clusters < min.size – useful for large problems
short	if short==TRUE, don't use variable names
labels	vector of text labels (contents) for the variables
size	size of output
node.font	Font to use for nodes in the graph
edge.font	Font to use for the labels of the arrows (edges)
rank.direction	LR or RL
digits	number of digits to show
title	any title
...	other options to pass

Details

Will create (or overwrite) an output file and print out the dot code to show a cluster structure. This dot file may be imported directly into a dot viewer (e.g., http://www.graphviz.org/). The "dot" language is a powerful graphic description language that is particulary appropriate for viewing cluster output. Commercial graphics programs (e.g., OmniGraffle) can also read (and clean up) dot files.

ICLUST.graph takes the output from ICLUST results and processes it to provide a pretty picture of the results. Original variables shown as rectangles and ordered on the left hand side (if rank direction is RL) of the graph. Clusters are drawn as ellipses and include the alpha, beta, and size of the cluster. Edges show the cluster intercorrelations.

It is possible to trim the output to not show all cluster information. Clusters < min.size are shown as small ovals without alpha, beta, and size information.

Although it would be nice to process the dot code directly in R, the Rgraphviz package is difficult to use on all platforms and thus the dot code is written directly.

Value

Output is a set of dot commands written either to console or to the output file. These commands may then be used as input to any "dot" viewer, e.g., Graphviz.

Author(s)

<revelle@northwestern.edu >
http://personality-project.org/revelle.html

References

ICLUST: http://personality-project.org/r/r.ICLUST.html

See Also

VSS.plot, ICLUST

Examples

```
## Not run:
test.data <- Harman74.cor$cov
ic.out <- ICLUST(test.data)
out.file <- file.choose(new=TRUE)   #create a new file to write the plot commands to
ICLUST.graph(ic.out,out.file)
now go to graphviz (outside of R) and open the out.file you created
print(ic.out,digits=2)

## End(Not run)


#test.data <- Harman74.cor$cov
#my.iclust <- ICLUST(test.data)
#ICLUST.graph(my.iclust)
#
#
#digraph ICLUST {
#  rankdir=RL;
#  size="8,8";
#  node [fontname="Helvetica" fontsize=14 shape=box, width=2];
#  edge [fontname="Helvetica" fontsize=12];
# label = "ICLUST";
```

```
# fontsize=20;
#V1  [label = VisualPerception];
#V2  [label = Cubes];
#V3  [label = PaperFormBoard];
#V4  [label = Flags];
#V5  [label = GeneralInformation];
#V6  [label = PargraphComprehension];
#V7  [label = SentenceCompletion];
#V8  [label = WordClassification];
#V9  [label = WordMeaning];
#V10  [label = Addition];
#V11  [label = Code];
#V12  [label = CountingDots];
#V13  [label = StraightCurvedCapitals];
#V14  [label = WordRecognition];
#V15  [label = NumberRecognition];
#V16  [label = FigureRecognition];
#V17  [label = ObjectNumber];
#V18  [label = NumberFigure];
#V19  [label = FigureWord];
#V20  [label = Deduction];
#V21  [label = NumericalPuzzles];
#V22  [label = ProblemReasoning];
#V23  [label = SeriesCompletion];
#V24  [label = ArithmeticProblems];
#node [shape=ellipse, width ="1"];
#C1-> V9 [ label = 0.78 ];
#C1-> V5 [ label = 0.78 ];
#C2-> V12 [ label = 0.66 ];
#C2-> V10 [ label = 0.66 ];
#C3-> V18 [ label = 0.53 ];
#C3-> V17 [ label = 0.53 ];
#C4-> V23 [ label = 0.59 ];
#C4-> V20 [ label = 0.59 ];
#C5-> V13 [ label = 0.61 ];
#C5-> V11 [ label = 0.61 ];
#C6-> V7 [ label = 0.78 ];
#C6-> V6 [ label = 0.78 ];
#C7-> V4 [ label = 0.55 ];
#C7-> V1 [ label = 0.55 ];
#C8-> V16 [ label = 0.5 ];
#C8-> V14 [ label = 0.49 ];
#C9-> C1 [ label = 0.86 ];
#C9-> C6 [ label = 0.86 ];
#C10-> C4 [ label = 0.71 ];
#C10-> V22 [ label = 0.62 ];
#C11-> V21 [ label = 0.56 ];
#C11-> V24 [ label = 0.58 ];
#C12-> C10 [ label = 0.76 ];
#C12-> C11 [ label = 0.67 ];
#C13-> C8 [ label = 0.61 ];
#C13-> V15 [ label = 0.49 ];
#C14-> C2 [ label = 0.74 ];
```

```
#C14-> C5 [ label = 0.72 ];
#C15-> V3 [ label = 0.48 ];
#C15-> C7 [ label = 0.65 ];
#C16-> V19 [ label = 0.48 ];
#C16-> C3 [ label = 0.64 ];
#C17-> V8 [ label = 0.62 ];
#C17-> C12 [ label = 0.8 ];
#C18-> C17 [ label = 0.82 ];
#C18-> C15 [ label = 0.68 ];
#C19-> C16 [ label = 0.66 ];
#C19-> C13 [ label = 0.65 ];
#C20-> C19 [ label = 0.72 ];
#C20-> C18 [ label = 0.83 ];
#C21-> C20 [ label = 0.87 ];
#C21-> C9 [ label = 0.76 ];
#C22-> 0 [ label = 0 ];
#C22-> 0 [ label = 0 ];
#C23-> 0 [ label = 0 ];
#C23-> 0 [ label = 0 ];
#C1  [label =   "C1\n  alpha= 0.84\n beta=  0.84\nN= 2"] ;
#C2  [label =   "C2\n  alpha= 0.74\n beta=  0.74\nN= 2"] ;
#C3  [label =   "C3\n  alpha= 0.62\n beta=  0.62\nN= 2"] ;
#C4  [label =   "C4\n  alpha= 0.67\n beta=  0.67\nN= 2"] ;
#C5  [label =   "C5\n  alpha= 0.7\n beta=  0.7\nN= 2"] ;
#C6  [label =   "C6\n  alpha= 0.84\n beta=  0.84\nN= 2"] ;
#C7  [label =   "C7\n  alpha= 0.64\n beta=  0.64\nN= 2"] ;
#C8  [label =   "C8\n  alpha= 0.58\n beta=  0.58\nN= 2"] ;
#C9  [label =   "C9\n  alpha= 0.9\n beta=  0.87\nN= 4"] ;
#C10  [label =   "C10\n  alpha= 0.74\n beta=  0.71\nN= 3"] ;
#C11  [label =   "C11\n  alpha= 0.62\n beta=  0.62\nN= 2"] ;
#C12  [label =   "C12\n  alpha= 0.79\n beta=  0.74\nN= 5"] ;
#C13  [label =   "C13\n  alpha= 0.64\n beta=  0.59\nN= 3"] ;
#C14  [label =   "C14\n  alpha= 0.79\n beta=  0.74\nN= 4"] ;
#C15  [label =   "C15\n  alpha= 0.66\n beta=  0.58\nN= 3"] ;
#C16  [label =   "C16\n  alpha= 0.65\n beta=  0.57\nN= 3"] ;
#C17  [label =   "C17\n  alpha= 0.81\n beta=  0.71\nN= 6"] ;
#C18  [label =   "C18\n  alpha= 0.84\n beta=  0.75\nN= 9"] ;
#C19  [label =   "C19\n  alpha= 0.74\n beta=  0.65\nN= 6"] ;
#C20  [label =   "C20\n  alpha= 0.87\n beta=  0.74\nN= 15"] ;
#C21  [label =   "C21\n  alpha= 0.9\n beta=  0.77\nN= 19"] ;
#C22  [label =   "C22\n  alpha= 0\n beta=  0\nN= 0"] ;
#C23  [label =   "C23\n  alpha= 0\n beta=  0\nN= 0"] ;
#{ rank=same;
#V1;V2;V3;V4;V5;V6;V7;V8;V9;V10;V11;V12;V13;V14;V15;V16;V17;V18;V19;V20;V21;V22;V23;V24;}}
#
#copy the above output to Graphviz and draw it
#see \url{http://personality-project.org/r/r.ICLUST.html} for an example.
```

ICLUST.rgraph *Draw an ICLUST graph using the Rgraphviz package*

Description

Given a cluster structure determined by ICLUST, create a rgraphic directly using Rgraphviz. To create dot code to describe the ICLUST output with more precision, use ICLUST.graph. As an option, dot code is also generated and saved in a file. To use the dot code, use either http://www.graphviz.org/ Graphviz or a commercial viewer (e.g., OmniGraffle).

Usage

```
ICLUST.rgraph(ic.results, out.file = NULL, min.size = 1, short = FALSE,
    labels = NULL, size = c(8, 6), node.font = c("Helvetica", 14),
   edge.font = c("Helvetica", 10), rank.direction=c("RL","TB","LR","BT"),
    digits = 2, title = "ICLUST",label.font=2, ...)
```

Arguments

`ic.results`	output list from ICLUST
`out.file`	File name to save optional dot code.
`min.size`	draw a smaller node (without all the information) for clusters < min.size – useful for large problems
`short`	if short==TRUE, don't use variable names
`labels`	vector of text labels (contents) for the variables
`size`	size of output
`node.font`	Font to use for nodes in the graph
`edge.font`	Font to use for the labels of the arrows (edges)
`rank.direction`	LR or TB or RL
`digits`	number of digits to show
`title`	any title
`label.font`	The variable labels can be a different size than the other nodes. This is particularly helpful if the number of variables is large or the labels are long.
`...`	other options to pass

Details

Will create (or overwrite) an output file and print out the dot code to show a cluster structure. This dot file may be imported directly into a dot viewer (e.g., http://www.graphviz.org/). The "dot" language is a powerful graphic description language that is particulary appropriate for viewing cluster output. Commercial graphics programs (e.g., OmniGraffle) can also read (and clean up) dot files.

ICLUST.rgraph takes the output from ICLUST results and processes it to provide a pretty picture of the results. Original variables shown as rectangles and ordered on the left hand side (if rank direction is RL) of the graph. Clusters are drawn as ellipses and include the alpha, beta, and size of the cluster. Edges show the cluster intercorrelations.

It is possible to trim the output to not show all cluster information. Clusters < min.size are shown as small ovals without alpha, beta, and size information.

Value

Output is a set of dot commands written either to console or to the output file. These commands may then be used as input to any "dot" viewer, e.g., Graphviz.

ICLUST.rgraph is a version of `ICLUST.graph` that uses Rgraphviz to draw on the screen as well.

Additional output is drawn to main graphics screen.

Note

Requires Rgraphviz

Author(s)

`<revelle@northwestern.edu >`
`http://personality-project.org/revelle.html`

References

ICLUST: http://personality-project.org/r/r.ICLUST.html

See Also

`VSS.plot`, `ICLUST`

Examples

```
test.data <- Harman74.cor$cov
ic.out <- ICLUST(test.data)   #uses iclust.diagram instead
```

`ICLUST.sort` *Sort items by absolute size of cluster loadings*

Description

Given a cluster analysis or factor analysis loadings matrix, sort the items by the (absolute) size of each column of loadings. Used as part of ICLUST and SAPA analyses. The columns are rearranged by the

Usage

```
ICLUST.sort(ic.load, cut = 0, labels = NULL,keys=FALSE,clustsort=TRUE)
```

Arguments

`ic.load`	The output from a factor or principal components analysis, or from ICLUST, or a matrix of loadings.
`cut`	Do not include items in clusters with absolute loadings less than cut
`labels`	labels for each item.
`keys`	should cluster keys be returned? Useful if clusters scales are to be scored.
`clustsort`	TRUE will will sort the clusters by their eigenvalues

Details

When interpreting cluster or factor analysis outputs, is is useful to group the items in terms of which items have their biggest loading on each factor/cluster and then to sort the items by size of the absolute factor loading.

A stable cluster solution will be one in which the output of these cluster definitions does not vary when clusters are formed from the clusters so defined.

With the keys=TRUE option, the resulting cluster keys may be used to score the original data or the correlation matrix to form clusters from the factors.

Value

`sorted`	A data.frame of item numbers, item contents, and item x factor loadings.
`cluster`	A matrix of -1, 0, 1s defining each item by the factor/cluster with the row wise largest absolute loading.

...

Note

Although part of the ICLUST set of programs, this is also more useful for factor or principal components analysis.

Author(s)

William Revelle

References

`http://personality-project.org/r/r.ICLUST.html`

See Also

`ICLUST.graph`,`ICLUST.cluster`, `cluster.fit` , `VSS`, `factor2cluster`

`income` *US family income from US census 2008*

Description

US census data on family income from 2008

Usage

```
data(income)
```

Format

A data frame with 44 observations on the following 4 variables.

`value` lower boundary of the income group

`count` Number of families within that income group

`mean` Mean of the category

`prop` proportion of families

Details

The distribution of income is a nice example of a log normal distribution. It is also an interesting example of the power of graphics. It is quite clear when graphing the data that income statistics are bunched to the nearest 5K. That is, there is a clear sawtooth pattern in the data.

The all.income set is interpolates intervening values for 100-150K, 150-200K and 200-250K

Source

US Census: Table HINC-06. Income Distribution to $250,000 or More for Households: 2008

http://www.census.gov/hhes/www/cpstables/032009/hhinc/new06_000.htm

Examples

```
data(income)
with(income[1:40,], plot(mean,prop, main="US family income for 2008",xlab="income",
        ylab="Proportion of families",xlim=c(0,100000)))
with (income[1:40,], points(lowess(mean,prop,f=.3),typ="l"))
describe(income)


with(all.income, plot(mean,prop, main="US family income for 2008",xlab="income",
                ylab="Proportion of families",xlim=c(0,250000)))
with (all.income[1:50,], points(lowess(mean,prop,f=.25),typ="l"))
#curve(100000* dlnorm(x, 10.8, .8), x = c(0,250000),ylab="Proportion")
```

`interp.median` *Find the interpolated sample median, quartiles, or specific quantiles for a vector, matrix, or data frame*

Description

For data with a limited number of response categories (e.g., attitude items), it is useful treat each response category as range with width, w and linearly interpolate the median, quartiles, or any quantile value within the median response.

Usage

```
interp.median(x, w = 1,na.rm=TRUE)
interp.quantiles(x, q = .5, w = 1,na.rm=TRUE)
interp.quartiles(x,w=1,na.rm=TRUE)
interp.boxplot(x,w=1,na.rm=TRUE)
interp.values(x,w=1,na.rm=TRUE)
interp.qplot.by(y,x,w=1,na.rm=TRUE,xlab="group",ylab="dependent",
               ylim=NULL,arrow.len=.05,typ="b",add=FALSE,...)
```

Arguments

`x`	input vector
`q`	quantile to estimate ($0 < q < 1$
`w`	category width
`y`	input vector for interp.qplot.by
`na.rm`	should missing values be removed
`xlab`	x label
`ylab`	Y label
`ylim`	limits for the y axis
`arrow.len`	length of arrow in interp.qplot.by
`typ`	plot type in interp.qplot.by
`add`	add the plot or not
`...`	additional parameters to plotting function

Details

If the total number of responses is N, with median, M, and the number of responses at the median value, Nm >1, and Nb= the number of responses less than the median, then with the assumption that the responses are distributed uniformly within the category, the interpolated median is M - .5w + w*(N/2 - Nb)/Nm.

The generalization to 1st, 2nd and 3rd quartiles as well as the general quantiles is straightforward.

A somewhat different generalization allows for graphic presentation of the difference between interpolated and non-interpolated points. This uses the interp.values function.

If the input is a matrix or data frame, quantiles are reported for each variable.

Value

im	interpolated median(quantile)
v	interpolated values for all data points

See Also

median

Examples

```
interp.median(c(1,2,3,3,3))  # compare with median = 3
interp.median(c(1,2,2,5))
interp.quantiles(c(1,2,2,5),.25)
x <- sample(10,100,TRUE)
interp.quartiles(x)
#
x <-  c(1,1,2,2,2,3,3,3,3,4,5,1,1,1,2,2,3,3,3,3,4,5,1,1,1,2,2,3,3,3,3,4,2)
y <-  c(1,2,3,3,3,3,4,4,4,4,4,1,2,3,3,3,3,4,4,4,4,5,1,5,3,3,3,3,4,4,4,4,4)
x <-  x[order(x)]   #sort the data by ascending order to make it clearer
y <- y[order(y)]
xv <- interp.values(x)
yv <- interp.values(y)
barplot(x,space=0,xlab="ordinal position",ylab="value")
lines(1:length(x)-.5,xv)
points(c(length(x)/4,length(x)/2,3*length(x)/4),interp.quartiles(x))
barplot(y,space=0,xlab="ordinal position",ylab="value")
lines(1:length(y)-.5,yv)
points(c(length(y)/4,length(y)/2,3*length(y)/4),interp.quartiles(y))
data(galton)
interp.median(galton)
interp.qplot.by(galton$child,galton$parent,ylab="child height"
,xlab="Mid parent height")
```

---

iqitems *16 multiple choice IQ items*

Description

16 multiple choice ability items taken from the Synthetic Aperture Personality Assessment (SAPA) web based personality assessment project. The data from 1525 subjects are included here as a demonstration set for scoring multiple choice inventories and doing basic item statistics. For more information on the development of an open source measure of cognitive ability, consult the readings available at the personality-project.org.

Usage

```
data(iqitems)
```

Format

A data frame with 1525 observations on the following 16 variables. The number following the name is the item number from SAPA.

`reason.4` Basic reasoning questions

`reason.16` Basic reasoning question

`reason.17` Basic reasoning question

`reason.19` Basic reasoning question

`letter.7` In the following alphanumeric series, what letter comes next?

`letter.33` In the following alphanumeric series, what letter comes next?

`letter.34` In the following alphanumeric series, what letter comes next

`letter.58` In the following alphanumeric series, what letter comes next?

`matrix.45` A matrix reasoning task

`matrix.46` A matrix reasoning task

`matrix.47` A matrix reasoning task

`matrix.55` A matrix reasoning task

`rotate.3` Spatial Rotation of type 1.2

`rotate.4` Spatial Rotation of type 1.2

`rotate.6` Spatial Rotation of type 1.1

`rotate.8` Spatial Rotation of type 2.3

Details

16 items were sampled from 80 items given as part of the SAPA (`http://sapa-project.org`) project (Revelle, Wilt and Rosenthal, 2009; Condon and Revelle, 2014) to develop online measures of ability. These 16 items reflect four lower order factors (verbal reasoning, letter series, matrix reasoning, and spatial rotations. These lower level factors all share a higher level factor ('g').

This data set and the associated data set (`ability` based upon scoring these multiple choice items and converting them to correct/incorrect may be used to demonstrate item response functions, `tetrachoric` correlations, or `irt.fa` as well as `omega` estimates of of reliability and hierarchical structure.

In addition, the data set is a good example of doing item analysis to examine the empirical response probabilities of each item alternative as a function of the underlying latent trait. When doing this, it appears that two of the matrix reasoning problems do not have monotonically increasing trace lines for the probability correct. At moderately high ability (theta = 1) there is a decrease in the probability correct from theta = 0 and theta = 2.

Source

The example data set is taken from the Synthetic Aperture Personality Assessment personality and ability test at `http://sapa-project.org`. The data were collected with David Condon from 8/08/12 to 8/31/12.

References

Revelle, William, Wilt, Joshua, and Rosenthal, Allen (2010) Personality and Cognition: The Personality-Cognition Link. In Gruszka, Alexandra and Matthews, Gerald and Szymura, Blazej (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

Condon, David and Revelle, William, (2014) The International Cognitive Ability Resource: Development and initial validation of a public-domain measure. Intelligence, 43, 52-64.

Examples

```
## Not run:
data(iqitems)
iq.keys <- c(4,4,4, 6,  6,3,4,4,   5,2,2,4,   3,2,6,7)
score.multiple.choice(iq.keys,iqitems)   #this just gives summary statistics
#convert them to true false
iq.scrub <- scrub(iqitems,isvalue=0)  #first get rid of the zero responses
iq.tf <-  score.multiple.choice(iq.keys,iq.scrub,score=FALSE)
              #convert to wrong (0) and correct (1) for analysis
describe(iq.tf)
#see the ability data set for these analyses
#now, for some item analysis
#iq.irt <- irt.fa(iq.tf)  #do a basic irt
#iq.sc <-score.irt(iq.irt,iq.tf)  #find the scores
#op <- par(mfrow=c(4,4))
#irt.responses(iq.sc[,1], iq.tf)
#op <- par(mfrow=c(1,1))

## End(Not run)
```

`irt.1p` *Item Response Theory estimate of theta (ability) using a Rasch (like) model*

Description

Item Response Theory models individual responses to items by estimating individual ability (theta) and item difficulty (diff) parameters. This is an early and crude attempt to capture this modeling procedure. A better procedure is to use `irt.fa`.

Usage

```
irt.person.rasch(diff, items)
irt.0p(items)
irt.1p(delta,items)
irt.2p(delta,beta,items)
```

Arguments

`diff`	A vector of item difficulties –probably taken from irt.item.diff.rasch
`items`	A matrix of 0,1 items nrows = number of subjects, ncols = number of items
`delta`	delta is the same as diff and is the item difficulty parameter
`beta`	beta is the item discrimination parameter found in `irt.discrim`

Details

A very preliminary IRT estimation procedure. Given scores xij for ith individual on jth item Classical Test Theory ignores item difficulty and defines ability as expected score : abilityi = theta(i) = x(i.) A zero parameter model rescales these mean scores from 0 to 1 to a quasi logistic scale ranging from - 4 to 4 This is merely a non-linear transform of the raw data to reflect a logistic mapping.

Basic 1 parameter (Rasch) model considers item difficulties (delta j): p(correct on item j for the ith subject |theta i, deltaj) = 1/(1+exp(deltaj - thetai)) If we have estimates of item difficulty (delta), then we can find theta i by optimization

Two parameter model adds item sensitivity (beta j): p(correct on item j for subject i |thetai, deltaj, betaj) = 1/(1+exp(betaj *(deltaj- theta i))) Estimate delta, beta, and theta to maximize fit of model to data.

The procedure used here is to first find the item difficulties assuming theta = 0 Then find theta given those deltas Then find beta given delta and theta.

This is not an "official" way to do IRT, but is useful for basic item development. See `irt.fa` and `score.irt` for far better options.

Value

a data.frame with estimated ability (theta) and quality of fit. (for irt.person.rasch)
a data.frame with the raw means, theta0, and the number of items completed

Note

Not recommended for serious use. This code is under development. Much better functions are in the ltm and eRm packages. Similar analyses can be done using `irt.fa` and `score.irt`.

Author(s)

William Revelle

See Also

`sim.irt`, `sim.rasch`, `logistic`, `irt.fa`, `tetrachoric`, `irt.item.diff.rasch` irt.fa *Item Response Analysis by Exploratory Factor Analysis of tetrachoric/polychoric correlations*

Description

Although exploratory factor analysis and Item Response Theory seem to be very different models of binary data, they can provide equivalent parameter estimates of item difficulty and item discrimination. Tetrachoric or polychoric correlations of a data set of dichotomous or polytomous items may be factor analysed using a minimum residual or maximum likelihood factor analysis and the result loadings transformed to item discrimination parameters. The tau parameter from the tetrachoric/polychoric correlations combined with the item factor loading may be used to estimate item difficulties.

Usage

```
irt.fa(x,nfactors=1,correct=TRUE,plot=TRUE,n.obs=NULL,rotate="oblimin",fm="minres",
       sort=TRUE,...)
irt.select(x,y)
fa2irt(f,rho,plot=TRUE,n.obs=NULL)
```

Arguments

x	A data matrix of dichotomous or discrete items, or the result of tetrachoric or polychoric
nfactors	Defaults to 1 factor
correct	If true, then correct the tetrachoric correlations for continuity. (See tetrachoric).
plot	If TRUE, automatically call the plot.irt or plot.poly functions.
y	the subset of variables to pick from the rho and tau output of a previous irt.fa analysis to allow for further analysis.
n.obs	The number of subjects used in the initial analysis if doing a second analysis of a correlation matrix. In particular, if using the fm="minchi" option, this should be the matrix returned by count.pairwise.
rotate	The default rotation is oblimin. See fa for the other options.
fm	The default factor extraction is minres. See fa for the other options.
f	The object returned from fa
rho	The object returned from polychoric or tetrachoric. This will include both a correlation matrix and the item difficulty levels.
sort	Should the factor loadings be sorted before preparing the item information tables. Defaults to TRUE as this is more useful for tabular output.
...	Additional parameters to pass to the factor analysis function

Details

irt.fa combines several functions into one to make the process of item response analysis easier. Correlations are found using either tetrachoric or polychoric. Exploratory factor analyeses with all the normal options are then done using fa. The results are then organized to be reported in terms of IRT parameters (difficulties and discriminations) as well as the more conventional factor analysis output. In addition, because the correlation step is somewhat slow, reanalyses may be done using the correlation matrix found in the first step. In this case, if it is desired to use the fm="minchi" factoring method, the number of observations needs to be specified as the matrix resulting from count.pairwise.

The tetrachoric correlation matrix of dichotomous items may be factored using a (e.g.) minimum residual factor analysis function fa and the resulting loadings, λ_i are transformed to discriminations by $\alpha = \frac{\lambda_i}{\sqrt{1-\lambda_i^2}}$.

The difficulty parameter, δ is found from the τ parameter of the tetrachoric or polychoric function.

$\delta_i = \frac{\tau_i}{\sqrt{1-\lambda_i^2}}$

Similar analyses may be done with discrete item responses using polychoric correlations and distinct estimates of item difficulty (location) for each item response.

The results may be shown graphically using `link{plot.irt}` for dichotomous items or `link{plot.poly}` for polytomous items. These called by plotting the irt.fa output, see the examples). For plotting there are three options: type = "ICC" will plot the item characteristic response function. type = "IIC" will plot the item information function, and type= "test" will plot the test information function. Invisible output from the plot function will return tables of item information as a function of several levels of the trait, as well as the standard error of measurement and the reliability at each of those levels.

The normal input is just the raw data. If, however, the correlation matrix has already been found using tetrachoric, polychoric, or a previous analysis using irt.fa then that result can be processed directly. Because irt.fa saves the rho and tau matrices from the analysis, subsequent analyses of the same data set are much faster if the input is the object returned on the first run. A similar feature is available in omega.

The output is best seen in terms of graphic displays. Plot the output from irt.fa to see item and test information functions.

The print function will print the item location and discriminations. The additional factor analysis output is available as an object in the output and may be printed directly by specifying the $fa object.

The irt.select function is a helper function to allow for selecting a subset of a prior analysis for further analysis. First run irt.fa, then select a subset of variables to be analyzed in a subsequent irt.fa analysis. Perhaps a better approach is to just plot and find the information for selected items.

The plot function for an irt.fa object will plot ICC (item characteristic curves), IIC (item information curves), or test information curves. In addition, by using the "keys" option, these three kinds of plots can be done for selected items. This is particularly useful when trying to see the information characteristics of short forms of tests based upon the longer form factor analysis.

The plot function will also return (invisibly) the informaton at multiple levels of the trait, the average information (area under the curve) as well as the location of the peak information for each item. These may be then printed or printed in sorted order using the sort option in print.

Value

irt	A list of Item location (difficulty) and discrimination
fa	A list of statistics for the factor analyis
rho	The tetrachoric/polychoric correlation matrix
tau	The tetrachoric/polychoric cut points

Note

In comparing irt.fa to the ltm function in the ltm package or to the analysis reported in Kamata and Bauer (2008) the discrimination parameters are not identical, because the irt.fa reports them in units of the normal curve while ltm and Kamata and Bauer report them in logistic units. In addition, Kamata and Bauer do their factor analysis using a logistic error model. Their results match the irt.fa results (to the 2nd or 3rd decimal) when examining their analyses using a normal model. (With thanks to Akihito Kamata for sharing that analysis.)

`irt.fa` reports parameters in normal units. To convert them to conventional IRT parameters, multiply by 1.702. In addition, the location parameter is expressed in terms of difficulty (high positive scores imply lower frequency of response.)

The results of `irt.fa` can be used by `score.irt` for irt based scoring. First run `irt.fa` and then score the results using a two parameter model using `score.irt`.

Author(s)

William Revelle

References

Kamata, Akihito and Bauer, Daniel J. (2008) A Note on the Relation Between Factor Analytic and Item Response Theory Models Structural Equation Modeling, 15 (1) 136-153.

McDonald, Roderick P. (1999) Test theory: A unified treatment. L. Erlbaum Associates.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at http://personality-project.org/r/book/

See Also

`fa`, `sim.irt`, `tetrachoric`, `polychoric` as well as `plot.psych` for plotting the IRT item curves.

See also `score.irt` for scoring items based upon these parameter estimates. `irt.responses` will plot the empirical response curves for the alternative response choices for multiple choice items.

Examples

```
## Not run:
set.seed(17)
d9 <- sim.irt(9,1000,-2.5,2.5,mod="normal") #dichotomous items
test <- irt.fa(d9$items)
test
op <- par(mfrow=c(3,1))
plot(test,type="ICC")
```

```
plot(test,type="IIC")
plot(test,type="test")
par(op)
set.seed(17)
items <- sim.congeneric(N=500,short=FALSE,categorical=TRUE) #500 responses to 4 discrete items
d4 <- irt.fa(items$observed)  #item response analysis of congeneric measures
d4    #show just the irt output
d4$fa  #show just the factor analysis output


op <- par(mfrow=c(2,2))
plot(d4,type="ICC")
par(op)


#using the iq data set for an example of real items
#first need to convert the responses to tf
data(iqitems)
iq.keys <- c(4,4,4, 6, 6,3,4,4,  5,2,2,4,  3,2,6,7)

iq.tf <- score.multiple.choice(iq.keys,iqitems,score=FALSE)  #just the responses
iq.irt <- irt.fa(iq.tf)
print(iq.irt,short=FALSE) #show the IRT as well as factor analysis output
p.iq <- plot(iq.irt)  #save the invisible summary table
p.iq  #show the summary table of information by ability level
#select a subset of these variables
small.iq.irt <- irt.select(iq.irt,c(1,5,9,10,11,13))
small.irt <- irt.fa(small.iq.irt)
plot(small.irt)
#find the information for three subset of iq items
keys <- make.keys(16,list(all=1:16,some=c(1,5,9,10,11,13),others=c(1:5)))
plot(iq.irt,keys=keys)

## End(Not run)
#compare output to the ltm package or Kamata and Bauer   -- these are in logistic units
ls <- irt.fa(lsat6)
#library(ltm)
# lsat.ltm <- ltm(lsat6~z1)
#  round(coefficients(lsat.ltm)/1.702,3)  #convert to normal (approximation)
#
#   Dffclt Dscrmn
#Q1 -1.974  0.485
#Q2 -0.805  0.425
#Q3 -0.164  0.523
#Q4 -1.096  0.405
#Q5 -1.835  0.386


#Normal results  ("Standardized and Marginal")(from Akihito Kamata )
#Item       discrim             tau
#  1       0.4169             -1.5520
#  2       0.4333             -0.5999
#  3       0.5373             -0.1512
```

```
#  4       0.4044             -0.7723
#  5       0.3587             -1.1966
#compare to ls

  #Normal results  ("Standardized and conditional") (from Akihito Kamata )
#item           discrim   tau
#  1           0.3848    -1.4325
#  2           0.3976    -0.5505
#  3           0.4733    -0.1332
#  4           0.3749    -0.7159
#  5           0.3377    -1.1264
#compare to ls$fa and ls$tau

#Kamata and Bauer (2008) logistic estimates
#1   0.826    2.773
#2   0.723    0.990
#3   0.891    0.249
#4   0.688    1.285
#5   0.657    2.053
```

irt.item.diff.rasch *Simple function to estimate item difficulties using IRT concepts*

Description

Steps toward a very crude and preliminary IRT program. These two functions estimate item difficulty and discrimination parameters. A better procedure is to use `irt.fa` or the ltm package.

Usage

```
irt.item.diff.rasch(items)
irt.discrim(item.diff,theta,items)
```

Arguments

`items`	a matrix of items
`item.diff`	a vector of item difficulties (found by irt.item.diff)
`theta`	ability estimate from irt.person.theta

Details

Item Response Theory (aka "The new psychometrics") models individual responses to items with a logistic function and an individual (theta) and item difficulty (diff) parameter.

irt.item.diff.rasch finds item difficulties with the assumption of theta=0 for all subjects and that all items are equally discriminating.

irt.discrim takes those difficulties and theta estimates from `irt.person.rasch` to find item discrimination (beta) parameters.

A far better package with these features is the ltm package. The IRT functions in the psych-package are for pedagogical rather than production purposes. They are believed to be accurate, but are not guaranteed. They do seem to be slightly more robust to missing data structures associated with SAPA data sets than the ltm package.

The `irt.fa` function is also an alternative. This will find `tetrachoric` or `polychoric` correlations and then convert to IRT parameters using factor analysis (`fa`).

Value

a vector of item difficulties or item discriminations.

Note

Under development. Not recommended for public consumption. See `irt.fa` and `score.irt` for far better options.

Author(s)

William Revelle

See Also

`irt.fa`, `irt.person.rasch`

`irt.responses` *Plot probability of multiple choice responses as a function of a latent trait*

Description

When analyzing ability tests, it is important to consider how the distractor alternatives vary as a function of the latent trait. The simple graphical solution is to plot response endorsement frequencies against the values of the latent trait found from multiple items. A good item is one in which the probability of the distractors decrease and the keyed answer increases as the latent trait increases.

Usage

```
irt.responses(theta,items, breaks = 11,show.missing=FALSE, show.legend=TRUE,
legend.location="topleft", colors=NULL,...)
```

Arguments

theta	The estimated latent trait (found, for example by using score.irt).
items	A matrix or data frame of the multiple choice item responses.
breaks	The number of levels of the theta to use to form the probability estimates. May be increased if there are enough cases.
show.legend	Show the legend
show.missing	For some SAPA data sets, there are a very large number of missing responses. In general, we do not want to show their frequency.
legend.location	Choose among c("bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center","none"). The default is "topleft".
colors	if NULL, then use the default colors, otherwise, specify the color choices. The basic color palette is c("black", "blue", "red", "darkgreen", "gold2", "gray50", "cornflowerblue", "mediumorchid2").
...	Other parameters for plots and points

Details

This function is a convenient way to analyze the quality of item alternatives in a multiple choice ability test. The typical use is to first score the test (using, e.g., score.multiple.choice according to some scoring key and to then find the score.irt based scores. Response frequencies for each alternative are then plotted against total score. An ideal item is one in which just one alternative (the correct one) has a monotonically increasing response probability.

Because of the similar pattern of results for IRT based or simple sum based item scoring, the function can be run on scores calculated either by score.irt or by score.multiple.choice. In the latter case, the number of breaks should not exceed the number of possible score alternatives.

Value

Graphic output

Author(s)

William Revelle

References

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at http://personality-project.org/r/book/

See Also

score.multiple.choice, score.irt

Examples

```
data(iqitems)
iq.keys <- c(4,4,4, 6,6,3,4,4,  5,2,2,4,  3,2,6,7)
scores <- score.multiple.choice(iq.keys,iqitems,score=TRUE,short=FALSE)
#note that for speed we can just do this on simple item counts rather
# than IRT based scores.
op <- par(mfrow=c(2,2))  #set this to see the output for multiple items
irt.responses(scores$scores,iqitems[1:4],breaks=11)
op <-  par(op)
```

`kaiser` *Apply the Kaiser normalization when rotating factors*

Description

Kaiser (1958) suggested normalizing factor loadings before rotating them, and then denormalizing them after rotation. The GPArotation package does not (by default) normalize, nor does the `fa` function. Then, to make it more confusing, varimax in stats does,Varimax in GPArotation does not. `kaiser` will take the output of a non-normalized solution and report the normalized solution.

Usage

```
kaiser(f, rotate = "oblimin")
```

Arguments

`f`	A factor analysis output from `fa` or a factor loading matrix.
`rotate`	Any of the standard rotations avaialable in the GPArotation package.

Details

Best results if called from an unrotated solution. Repeated calls using a rotated solution will produce incorrect estimates of the correlations between the factors.

Value

See the values returned by GPArotation functions

Note

Prepared in response to a question about why `fa` oblimin results are different from SPSS.

Author(s)

William Revelle

References

Kaiser, H. F. (1958) The varimax criterion for analytic rotation in factor analysis. Psychometrika 23, 187-200.

See Also

fa

Examples

```
f3 <- fa(Thurstone,3)
f3n <- kaiser(fa(Thurstone,3,rotate="none"))
factor.congruence(f3,f3n)
```

KMO *Find the Kaiser, Meyer, Olkin Measure of Sampling Adequacy*

Description

Henry Kaiser (1970) introduced an Measure of Sampling Adequacy (MSA) of factor analytic data matrices. Kaiser and Rice (1974) then modified it. This is just a function of the squared elements of the 'image' matrix compared to the squares of the original correlations. The overall MSA as well as estimates for each item are found. The index is known as the Kaiser-Meyer-Olkin (KMO) index.

Usage

```
KMO(r)
```

Arguments

r	A correlation matrix or a data matrix (correlations will be found)

Details

Let $S^2 = diag(R^{-1})^{-1}$ and $Q = SR^{-1}S$. Then Q is said to the be the anti-image intercorrelation matrix. Let $sumr2 = \sum R^2$ and $sumq2 = \sum Q^2$ for all off diagonal elements of R and Q, then $SMA = sumr2)/(sumr2 + sumq2)$. Although originally MSA was 1 - sumq2/sumr2 (Kaiser, 1970), this was modified in Kaiser and Rice, (1974) to be $SMA = sumr2)/(sumr2 + sumq2)$. This is the formula used by Dziuban and Shirkey (1974) and by SPSS.

Value

- MSAThe overall Measure of Sampling Adequacy
- MSAiThe measure of sampling adequacy for each item itemImageThe Image correlation matrix (Q)

Author(s)

William Revelle

References

H.~F. Kaiser. (1970) A second generation little jiffy. Psychometrika, 35(4):401–415.

H.~F. Kaiser and J.~Rice. (1974) Little jiffy, mark iv. Educational and Psychological Measurement, 34(1):111–117.

Dziuban, Charles D. and Shirkey, Edwin C. (1974) When is a correlation matrix appropriate for factor analysis? Some decision rules. Psychological Bulletin, 81 (6) 358 - 361.

See Also

See Also as fa, Harman.political.

Examples

```
KMO(Thurstone)
KMO(Harman.political)   #compare to the results in Dziuban and Shirkey (1974)
```

---

logistic — *Logistic transform from x to p and logit transform from p to x*

Description

The logistic function (1/(1+exp(-x)) and logit function (log(p/(1-p)) are fundamental to Item Response Theory. Although just one line functions, they are included here for ease of demonstrations and in drawing IRT models. Also included is the logistic.grm for a graded response model.

Usage

```
logistic(x,d=0, a=1,c=0, z=1)
logit(p)
logistic.grm( x,d=0,a=1.5,c=0,z=1,r=2,s=c(-1.5,-.5,.5,1.5))
```

Arguments

x	Any integer or real value
d	Item difficulty or delta parameter
a	The slope of the curve at x=0 is equivalent to the discrimination parameter in 2PL models or alpha parameter. Is either 1 in 1PL or 1.702 in 1PN approximations.
c	Lower asymptote = guessing parameter in 3PL models or gamma
z	The upper asymptote — in 4PL models

p	Probability to be converted to logit value
r	The response category for the graded response model
s	The response thresholds

Details

These three functions are provided as simple helper functions for demonstrations of Item Response Theory. The one parameter logistic (1PL) model is also known as the Rasch model. It assumes items differ only in difficulty. 1PL, 2PL, 3PL and 4PL curves may be drawn by choosing the appropriate d (delta or item difficulty), a (discrimination or slope), c (gamma or guessing) and z (zeta or upper asymptote).

logit is just the inverse of logistic.

logistic.grm will create the responses for a graded response model for the rth category where cutpoints are in s.

Value

p	logistic returns the probability associated with x
x	logit returns the real number associated with p

Author(s)

William Revelle

Examples

```
curve(logistic(x,a=1.702),-3,3,ylab="Probability of x",
 main="Logistic transform of x",xlab="z score units")
 #logistic with a=1.702 is almost the same as pnorm

curve(pnorm(x),add=TRUE,lty="dashed")
curve(logistic(x),add=TRUE)
text(2,.8, expression(alpha ==1))
text(2,1.0,expression(alpha==1.7))
curve(logistic(x),-4,4,ylab="Probability of x",
           main = "Logistic transform of x in logit units",xlab="logits")
curve(logistic(x,d=-1),add=TRUE)
curve(logistic(x,d=1),add=TRUE)
curve(logistic(x,c=.2),add=TRUE,lty="dashed")
text(1.3,.5,"d=1")
text(.3,.5,"d=0")
text(-1.5,.5,"d=-1")
text(-3,.3,"c=.2")
#demo of graded response model
 curve(logistic.grm(x,r=1),-4,4,ylim=c(0,1),main="Five level response scale",
          ylab="Probability of endorsement",xlab="Latent attribute on logit scale")
 curve(logistic.grm(x,r=2),add=TRUE)
 curve(logistic.grm(x,r=3),add=TRUE)
 curve(logistic.grm(x,r=4),add=TRUE)
 curve(logistic.grm(x,r=5),add=TRUE)
```

```
text(-2.,.5,1)
text(-1.,.4,2)
text(0,.4,3)
text(1.,.4,4)
 text(2.,.4,5)
```

---

lowerUpper — *Combine two square matrices to have a lower off diagonal for one, upper off diagonal for the other*

Description

When reporting correlation matrices for two samples (e.g., males and females), it is convenient to show them as one matrix, with entries below the diagonal representing one matrix, and entries above the diagonal the other matrix. It is also useful to compare a correlation matrix with the residuals from a fitted (e.g., factor) model.

Usage

```
lowerUpper(lower, upper=NULL, diff=FALSE)
```

Arguments

`lower`	A square matrix
`upper`	A square matrix of the same size as the first (if omitted, then the matrix is converted to two symmetric matrices).
`diff`	Find the difference between the first and second matrix and put the results in the above the diagonal entries.

Details

If just one matrix is provided (i.e., upper is missing), it is decomposed into two square matrices, one equal to the lower off diagonal entries, the other to the upper off diagonal entries. In the normal case two symmetric matrices are provided and combined into one non-symmetric matrix with the lower off diagonals representing the lower matrix and the upper off diagonals representing the upper matrix.

If diff is true, the upper off diagonal matrix reflects the differences between the two matrices.

Value

Either one matrix or a list of two

Author(s)

William Revelle

See Also

read.clipboard.lower, cor.plot

Examples

```
b1 <- Bechtoldt.1
b2 <- Bechtoldt.2
b12 <- lowerUpper(b1,b2)
cor.plot(b12)
diff12 <- lowerUpper(b1,b2,diff=TRUE)

cor.plot(t(diff12),numbers=TRUE,main="Bechtoldt1 and the differences from Bechtoldt2")
```

---

make.keys *Create a keys matrix for use by score.items or cluster.cor*

Description

When scoring items by forming composite scales either from the raw data using score.items or from the correlation matrix using cluster.cor, it is necessary to create a keys matrix. This is just a short cut for doing so. The keys matrix is a nvar x nscales matrix of -1,0, 1 and defines the membership for each scale. Items can be specified by location or by name.

Usage

```
make.keys(nvars, keys.list, item.labels = NULL, key.labels = NULL)
```

Arguments

nvars	Number of variables items to be scored
keys.list	A list of the scoring keys,one element for each scale
item.labels	Typically, just the colnames of the items data matrix.
key.labels	Labels for the scales can be specified here, or in the key.list

Details

There are two ways to create keys for the scoreItems and scoreOverlap functions. One is to laboriously do it in a spreadsheet and then copy them into R. The other is to just specify them by item number in a list. Make keys allows one to specify items by name or by location or a mixture of both.

To address items by name it is necessary to specify item names, either by using the item.labels value, or by putting the name of the data file or the colnames of the data file to be scored into the first (nvars) position.

If specifying by number, then nvars is the total number of items in the object to be scored, not just the number of items used.

See the examples for the various options.

Note that make.keys was revised in Sept, 2013 to allow for keying by name.

It is also possible to do several make.keys operations and then combine them using `superMatrix`.

Value

`keys`	a nvars x nkeys matrix of -1, 0, or 1s describing how to score each scale. nkeys is the length of the keys.list

See Also

`scoreItems`, `scoreOverlap`, `cluster.cor` `superMatrix`

Examples

```
data(attitude)  #specify the items by location
 key.list <- list(all=c(1,2,3,4,-5,6,7),
                  first=c(1,2,3),
                  last=c(4,5,6,7))
 keys <- make.keys(7,key.list,item.labels = colnames(attitude))
 keys

 #scores <- score.items(keys,attitude)
 #scores

 data(bfi)
 #first create the keys by location (the conventional way)
 keys.list <- list(agree=c(-1,2:5),conscientious=c(6:8,-9,-10),
 extraversion=c(-11,-12,13:15),neuroticism=c(16:20),openness = c(21,-22,23,24,-25))
 keys <- make.keys(25,keys.list,item.labels=colnames(bfi)[1:25])

 #alternatively, create by a mixture of names and locations
 keys.list <- list(agree=c("-A1","A2","A3","A4","A5"),
conscientious=c("C1","C2","C2","-C4","-C5"),extraversion=c("-E1","-E2","E3","E4","E5"),
neuroticism=c(16:20),openness = c(21,-22,23,24,-25))
keys <- make.keys(bfi,keys.list) #specify the data file to be scored (bfi)
#or
keys <- make.keys(colnames(bfi),keys.list) #specify the names of the variables to be used
#or
#specify the number of variables to be scored and their names in all cases
keys <- make.keys(28,keys.list,colnames(bfi))


 scores <- score.items(keys,bfi)
 summary(scores)
```

mardia *Calculate univariate or multivariate (Mardia's test) skew and kurtosis for a vector, matrix, or data.frame*

Description

Find the skew and kurtosis for each variable in a data.frame or matrix. Unlike skew and kurtosis in e1071, this calculates a different skew for each variable or column of a data.frame/matrix. mardia applies Mardia's tests for multivariate skew and kurtosis

Usage

```
skew(x, na.rm = TRUE,type=3)
kurtosi(x, na.rm = TRUE,type=3)
mardia(x,na.rm = TRUE,plot=TRUE)
```

Arguments

`x`	A data.frame or matrix
`na.rm`	how to treat missing data
`type`	See the discussion in describe.
`plot`	Plot the expected normal distribution values versus the Mahalanobis distance of the subjects.

Details

given a matrix or data.frame x, find the skew or kurtosis for each column (for skew and kurtosis) or the multivariate skew and kurtosis in the case of mardia.

As of version 1.2.3,when finding the skew and the kurtosis, there are three different options available. These match the choices available in skewness and kurtosis found in the e1071 package (see Joanes and Gill (1998) for the advantages of each one).

If we define $m_r = [\sum(X - mx)^r]/n$ then

Type 1 finds skewness and kurtosis by $g_1 = m_3/(m_2)^{3/2}$ and $g_2 = m_4/(m_2)^2 - 3$.

Type 2 is $G1 = g1 * \sqrt{n * (n-1)}/(n-2)$ and $G2 = (n-1) * [(n+1)g2 + 6]/((n-2)(n-3))$.

Type 3 is $b1 = [(n-1)/n]^{3/2} m_3/m_2^{3/2}$ and $b2 = [(n-1)/n]^{3/2} m_4/m_2^2)$.

For consistency with e1071 and with the Joanes and Gill, the types are now defined as above.

However, from revision 1.0.93 to 1.2.3, kurtosi by default gives an unbiased estimate of the kurtosis (DeCarlo, 1997). Prior versions used a different equation which produced a biased estimate. (See the kurtosis function in the e1071 package for the distinction between these two formulae. The default, type 1 gave what is called type 2 in e1071. The other is their type 3.) For comparison with previous releases, specifying type = 2 will give the old estimate. These type numbers are now changed.

Value

skew	if input is a matrix or data.frame, skew is a vector of skews
kurtosi	if input is a matrix or data.frame, kurtosi is a vector of kurtosi
bp1	Mardia's bp1 estimate of multivariate skew
bp2	Mardia's bp2 estimate of multivariate kurtosis
skew	Mardia's skew statistic
small.skew	Mardia's small sample skew statistic
p.skew	Probability of skew
p.small	Probability of small.skew
kurtosis	Mardia's multivariate kurtosis statistic
p.kurtosis	Probability of kurtosis statistic
D	Mahalanobis distance of cases from centroid

Note

The mean function supplies means for the columns of a data.frame, but the overall mean for a matrix. Mean will throw a warning for non-numeric data, but colMeans stops with non-numeric data. Thus, the function uses either mean (for data frames) or colMeans (for matrices). This is true for skew and kurtosi as well.

Author(s)

William Revelle

References

Joanes, D.N. and Gill, C.A (1998). Comparing measures of sample skewness and kurtosis. The Statistician, 47, 183-189.

L.DeCarlo. 1997) On the meaning and use of kurtosis, Psychological Methods, 2(3):292-307,

K.V. Mardia (1970). Measures of multivariate skewness and kurtosis with applications. Biometrika, 57(3):pp. 519-30, 1970.

See Also

describe, describe.by, mult.norm in QuantPsyc, Kurt in QuantPsyc

Examples

```
round(skew(attitude),2)   #type 3 (default)
round(kurtosi(attitude),2)  #type 3 (default)
#for the differences between the three types of skew and kurtosis:
round(skew(attitude,type=1),2)  #type 1
round(skew(attitude,type=2),2)  #type 2
mardia(attitude)
x <- matrix(rnorm(1000),ncol=10)
describe(x)
mardia(x)
```

mat.sort — *Sort the elements of a correlation matrix to reflect factor loadings*

Description

To see the structure of a correlation matrix, it is helpful to organize the items so that the similar items are grouped together. One such grouping technique is factor analysis. mat.sort will sort the items by a factor model (if specified), or any other order, or by the loadings on the first factor (if unspecified)

Usage

```
mat.sort(m, f = NULL)
```

Arguments

m	A correlation matrix
f	A factor analysis output (i.e., one with a loadings matrix) or a matrix of weights

Details

The factor analysis output is sorted by size of the largest factor loading for each variable and then the matrix items are organized by those loadings. The default is to sort by the loadings on the first factor. Alternatives allow for ordering based upon any vector or matrix.

Value

A sorted correlation matrix, suitable for showing with `cor.plot`.

Author(s)

William Revelle

See Also

`fa`, `cor.plot`

Examples

```
data(Bechtoldt.1)
sorted <- mat.sort(Bechtoldt.1,fa(Bechtoldt.1,5))
cor.plot(sorted)
```

`matrix.addition` *A function to add two vectors or matrices*

Description

It is sometimes convenient to add two vectors or matrices in an operation analogous to matrix multiplication. For matrices nXm and mYp, the matrix sum of the i,jth element of nSp = sum(over m) of iXm + mYj.

Usage

```
x %+% y
```

Arguments

`x`	a n by m matrix (or vector if m= 1)
`y`	a m by p matrix (or vector if m = 1)

Details

Used in such problems as Thurstonian scaling. Although not technically matrix addition, as pointed out by Krus, there are many applications where the sum or difference of two vectors or matrices is a useful operation. An alternative operation for vectors is outer(x ,y , FUN="+") but this does not work for matrices.

Value

a n by p matix of sums

Author(s)

William Revelle

References

Krus, D. J. (2001) Matrix addition. Journal of Visual Statistics, 1, (February, 2001).

Examples

```
x <- seq(1,4)
z <- x %+% -t(x)
x
z
#compare with outer(x,-x,FUN="+")
x <- matrix(seq(1,6),ncol=2)
y <- matrix(seq(1,10),nrow=2)
z <- x %+% y
```

```
x
y
z
#but compare this with outer(x ,y,FUN="+")
```

---

mediate *Estimate and display direct and indirect effects of mediators and moderator in path models*

Description

Find the direct and indirect effects of a predictor in path models of mediation and moderation. Bootstrap confidence intervals for the indirect effects. Mediation models are just extended regression models making explicit the effect of particular covariates in the model. Moderation is done by multiplication of the predictor variables. This function supplies basic mediation/moderation analyses for some of the classic problem types.

Usage

```
mediate(y, x, m, data, mod = NULL, n.obs = NULL, use = "pairwise", n.iter = 5000,
       alpha = 0.05, std = FALSE,plot=TRUE)
mediate.diagram(medi,digits=2,ylim=c(3,7),xlim=c(-1,10),show.c=TRUE,
     main="Mediation model",...)
moderate.diagram(medi,digits=2,ylim=c(2,8),main="Moderation model",...)
```

Arguments

y	The dependent variable (or a formula suitable for a linear model)
x	One or more predictor variables
m	One (or more) mediating variables
data	A data frame holding the data or a correlation or covariance matrix.
mod	A moderating variable, if desired
n.obs	If the data are from a correlation or covariance matrix, how many observations were used. This will lead to simulated data for the bootstrap.
use	use="pairwise" is the default when finding correlations or covariances
n.iter	Number of bootstrap resamplings to conduct
alpha	Set the width of the confidence interval to be 1 - alpha
std	standardize the covariances to find the standardized betas
plot	Plot the resulting paths
digits	The number of digits to report in the mediate.diagram.
medi	The output from mediate may be imported into mediate.diagram
ylim	The limits for the y axis in the mediate and moderate diagram functions

`xlim`	The limits for the x axis. Make the minimum more negative if the x by x correlations do not fit.
`show.c`	If FALSE, do not draw the c lines, just the partialed (c') lines
`main`	The title for the mediate and moderate functions
`...`	Additional graphical parameters to pass to mediate.diagram

Details

When doing linear modeling, it is frequently convenient to estimate the direct effect of a predictor controlling for the indirect effect of a mediator. See Preacher and Hayes (2004) for a very thorough discussion of mediation. The mediate function will do some basic mediation and moderation models, with bootstrapped confidence intervals for the mediation/moderation effects.

Functionally, this is just regular linear regression and partial correlation with some different output.

In the case of being provided just a correlation matrix, the bootstrapped values are based upon bootstrapping from data matching the original covariance/correlation matrix with the addition of normal errors. This allows us to test the mediation/moderation effect even if not given raw data.

The function has been tested against some of the basic cases and examples in Hayes (2013) and the associated data sets.

For fine tuning the size of the graphic output, xlim and ylim can be specified in the mediate.diagram function. Otherwise, the graphics produced by mediate and moderate use the default xlim and ylim values.

Value

`total`	The total direct effect of x on y (c)
`direct`	The beta effects of x (c') and m (b) on y
`indirect`	The indirect effect of x through m on y (c-ab)
`mean.boot`	mean bootstrapped value of indirect effect
`sd.boot`	Standard deviation of bootstrapped values
`ci.quant`	The upper and lower confidence intervals based upon the quantiles of the bootstrapped distribution.
`boot`	The bootstrapped values themselves.
`a`	The effect of x on m
`b`	The effect of m on y
`b.int`	The interaction of x and mod (if specified)

Note

There are a number of other packages that do mediation analysis (e.g., sem and lavaan) and they are probably preferred. This function is supplied for the more basic cases, with 1..k y variables, 1..n x variables, and 1 ..j mediators. It will not do two step mediation.

Author(s)

William Revelle

References

Hayes, Andrew F. (2013) Introduction to mediation, moderation, and conditional process analysis: A regression-based approach. Guilford Press.

Preacher, Kristopher J and Hayes, Andrew F (2004) SPSS and SAS procedures for estimating indirect effects in simple mediation models. Behavior Research Methods, Instruments, \& Computers 36, (4) 717-731.

Data from Hayes (2013), Preacher and Hayes (2004), and from Kerchoff (1974)

See Also

setCor and setCor.diagram

Examples

```
#data from Preacher and Hayes (2004)
sobel <- structure(list(SATIS = c(-0.59, 1.3, 0.02, 0.01, 0.79, -0.35,
-0.03, 1.75, -0.8, -1.2, -1.27, 0.7, -1.59, 0.68, -0.39, 1.33,
-1.59, 1.34, 0.1, 0.05, 0.66, 0.56, 0.85, 0.88, 0.14, -0.72,
0.84, -1.13, -0.13, 0.2), THERAPY = structure(c(0, 1, 1, 0, 1,
1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
1, 1, 1, 0), value.labels = structure(c(1, 0), .Names = c("cognitive",
"standard"))), ATTRIB = c(-1.17, 0.04, 0.58, -0.23, 0.62, -0.26,
-0.28, 0.52, 0.34, -0.09, -1.09, 1.05, -1.84, -0.95, 0.15, 0.07,
-0.1, 2.35, 0.75, 0.49, 0.67, 1.21, 0.31, 1.97, -0.94, 0.11,
-0.54, -0.23, 0.05, -1.07)), .Names = c("SATIS", "THERAPY", "ATTRIB"
), row.names = c(NA, -30L), class = "data.frame", variable.labels = structure(c("Satisfaction",
"Therapy", "Attributional Positivity"), .Names = c("SATIS", "THERAPY",
"ATTRIB")))
 #n.iter set to 50 (instead of default of 5000) for speed of example
mediate(1,2,3,sobel,n.iter=50)  #The example in Preacher and Hayes

#the pmi covariance matrix from Hayes. 2013.
#data set from Hayes, 2013 has 123 cases instead of the covariance matrix used here

C.pmi <- structure(c(0.251232840197254, 0.119718779155005, 0.157470345195255,
0.124533519925363, 0.03052112488338, 0.0734039717446355, 0.119718779155005,
1.74573503931761, 0.647207783553245, 0.914575836332134, 0.0133613221378115,
-0.0379181660669066, 0.157470345195255, 0.647207783553245, 3.01572704251633,
1.25128282020525, -0.0224576835932294, 0.73973743835799, 0.124533519925363,
0.914575836332134, 1.25128282020525, 2.40342196454751, -0.0106624017059843,
-0.752990470478475, 0.03052112488338, 0.0133613221378115, -0.0224576835932294,
-0.0106624017059843, 0.229241636678662, 0.884479541516727, 0.0734039717446355,
-0.0379181660669066, 0.73973743835799, -0.752990470478475, 0.884479541516727,
33.6509729441557), .Dim = c(6L, 6L), .Dimnames = list(c("cond",
"pmi", "import", "reaction", "gender", "age"), c("cond", "pmi",
"import", "reaction", "gender", "age")))

 #n.iter set to 50 (instead of default of 5000) for speed of example
mediate(y="reaction",x = "cond",m=c("pmi","import"),data=C.pmi,n.obs=123,n.iter=50)
```

```
#Data from sem package taken from Kerckhoff (and in turn, from Lisrel manual)
R.kerch <- structure(list(Intelligence = c(1, -0.1, 0.277, 0.25, 0.572,
0.489, 0.335), Siblings = c(-0.1, 1, -0.152, -0.108, -0.105,
-0.213, -0.153), FatherEd = c(0.277, -0.152, 1, 0.611, 0.294,
0.446, 0.303), FatherOcc = c(0.25, -0.108, 0.611, 1, 0.248, 0.41,
0.331), Grades = c(0.572, -0.105, 0.294, 0.248, 1, 0.597, 0.478
), EducExp = c(0.489, -0.213, 0.446, 0.41, 0.597, 1, 0.651),
    OccupAsp = c(0.335, -0.153, 0.303, 0.331, 0.478, 0.651, 1
    )), .Names = c("Intelligence", "Siblings", "FatherEd", "FatherOcc",
"Grades", "EducExp", "OccupAsp"), class = "data.frame", row.names = c("Intelligence",
"Siblings", "FatherEd", "FatherOcc", "Grades", "EducExp", "OccupAsp"
))

 #n.iter set to 50 (instead of default of 5000) for speed of demo
mod.k <- mediate("OccupAsp","Intelligence",m= c(2:5),data=R.kerch,n.obs=767,n.iter=50)
mediate.diagram(mod.k)
#Compare the following solution to the path coefficients found by the sem package

mod.k2 <- mediate(y="OccupAsp",x=c("Intelligence","Siblings","FatherEd","FatherOcc"),
     m= c(5:6),data=R.kerch,n.obs=767,n.iter=50)
mediate.diagram(mod.k2,show.c=FALSE) #simpler output
```

`mixed.cor` *Find correlations for mixtures of continuous, polytomous, and dichotomous variables*

Description

For data sets with continuous, polytomous and dichotmous variables, the absolute Pearson correlation is downward biased from the underlying latent correlation. mixed.cor finds Pearson correlations for the continous variables, `polychoric`s for the polytomous items, `tetrachoric`s for the dichotomous items, and the `polyserial` or `biserial` correlations for the various mixed variables. Results include the complete correlation matrix, as well as the separate correlation matrices and difficulties for the polychoric and tetrachoric correlations.

Usage

```
mixed.cor(x = NULL, p = NULL, d=NULL,smooth=TRUE, correct=.5,global=TRUE,
        ncat=8,use="pairwise",method="pearson",weight=NULL)
```

Arguments

x	A set of continuous variables (may be missing) or, if p and d are missing, the variables to be analyzed.
p	A set of polytomous items (may be missing)
d	A set of dichotomous items (may be missing)

smooth	If TRUE, then smooth the correlation matix if it is non-positive definite
correct	When finding tetrachoric correlations, what value should be used to correct for continuity?
global	For polychorics, should the global values of the tau parameters be used, or should the pairwise values be used. Set to local if errors are occurring.
ncat	The number of categories beyond which a variable is considered "continuous".
use	The various options to the cor function include "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". The default here is "pairwise"
method	The correlation method to use for the continuous variables. "pearson" (default), "kendall", or "spearman"
weight	If specified, this is a vector of weights (one per participant) to differentially weight participants. The NULL case is equivalent of weights of 1 for all cases.

Details

This function is particularly useful as part of the Synthetic Apeture Personality Assessment (SAPA) (http://sapa-project.org) data sets where continuous variables (age, SAT V, SAT Q, etc) and mixed with polytomous personality items taken from the International Personality Item Pool (IPIP) and the dichotomous experimental IQ items that have been developed as part of SAPA (see, e.g., Revelle, Wilt and Rosenthal, 2010).

This is a very computationally intensive function which can be speeded up considerably by using multiple cores and using the parallel package. The number of cores to use when doing polychoric or tetrachoric. The greatest step in speed is going from 1 core to 2. This is about a 50% savings. Going to 4 cores seems to have about at 66% savings, and 8 a 75% savings. The number of parallel processes defaults to 2 but can be modified by using the options command: options("mc.cores"=4) will set the number of cores to 4.

Item response analyses using irt.fa may be done separately on the polytomous and dichotomous items in order to develop internally consistent scales. These scale may, in turn, be correlated with each other using the complete correlation matrix found by mixed.cor and using the score.items function.

This function is not quite as flexible as the hetcor function in John Fox's polychor package.

Note that the variables may be organized by type of data: first continuous, then polytomous, then dichotomous. This is done by simply specifying x, p, and d. This is advantageous in the case of some continuous variables having a limited number of categories because of subsetting.

Value

rho	The complete matrix
rx	The Pearson correlation matrix for the continuous items
poly	the polychoric correlation (poly$rho) and the item difficulties (poly$tau)
tetra	the tetrachoric correlation (tetra$rho) and the item difficulties (tetra$tau)

Note

mixed.cor was designed for the SAPA project (http://sapa-project.org) with large data sets with a mixture of continuous, dichotomous, and polytomous data. For smaller data sets, it is sometimes the case that the global estimate of the tau parameter will lead to unstable solutions. This may be corrected by setting the global parameter = FALSE.

When finding correlations between dummy coded SAPA data (e.g., of occupations), the real correlations are all slightly less than zero because of the ipsatized nature of the data. This leads to a non-positive definite correlation matrix because the matrix is no longer of full rank. Smoothing will correct this, even though this might not be desired. Turn off smoothing in this case.

Note that the variables no longer need to be organized by type of data: first continuous, then polytomous, then dichotomous. However, this automatic detection will lead to problems if the variables such as age are limited to less than 8 categories but those category values differ from the polytomous items. The fall back is to specify x, p, and d.

Author(s)

William Revelle

References

W.Revelle, J.Wilt, and A.Rosenthal. Personality and cognition: The personality-cognition link. In A.Gruszka, G. Matthews, and B. Szymura, editors, Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, chapter 2, pages 27-49. Springer, 2010.

See Also

polychoric, tetrachoric, score.items, score.irt

Examples

```
data(bfi)
r <- mixed.cor(bfi[,c(1:5,26,28)])
r
#compare to raw Pearson
#note that the biserials and polychorics are not attenuated
rp <- cor(bfi[c(1:5,26,28)],use="pairwise")
lowerMat(rp)
```

---

msq *75 mood items from the Motivational State Questionnaire for 3896 participants*

Description

Emotions may be described either as discrete emotions or in dimensional terms. The Motivational State Questionnaire (MSQ) was developed to study emotions in laboratory and field settings. The data can be well described in terms of a two dimensional solution of energy vs tiredness and tension versus calmness. Additional items include what time of day the data were collected and a few personality questionnaire scores.

Usage

```
data(msq)
```

Format

A data frame with 3896 observations on the following 92 variables.

`active` a numeric vector

`afraid` a numeric vector

`alert` a numeric vector

`angry` a numeric vector

`anxious` a numeric vector

`aroused` a numeric vector

`ashamed` a numeric vector

`astonished` a numeric vector

`at.ease` a numeric vector

`at.rest` a numeric vector

`attentive` a numeric vector

`blue` a numeric vector

`bored` a numeric vector

`calm` a numeric vector

`cheerful` a numeric vector

`clutched.up` a numeric vector

`confident` a numeric vector

`content` a numeric vector

`delighted` a numeric vector

`depressed` a numeric vector

`determined` a numeric vector

`distressed` a numeric vector

`drowsy` a numeric vector

`dull` a numeric vector

`elated` a numeric vector

`energetic` a numeric vector

`enthusiastic` a numeric vector

`excited` a numeric vector

`fearful` a numeric vector

`frustrated` a numeric vector

`full.of.pep` a numeric vector

`gloomy` a numeric vector

`grouchy` a numeric vector

`guilty` a numeric vector

`happy` a numeric vector

`hostile` a numeric vector

`idle` a numeric vector

`inactive` a numeric vector

`inspired` a numeric vector

`intense` a numeric vector

`interested` a numeric vector

`irritable` a numeric vector

`jittery` a numeric vector

`lively` a numeric vector

`lonely` a numeric vector

`nervous` a numeric vector

`placid` a numeric vector

`pleased` a numeric vector

`proud` a numeric vector

`quiescent` a numeric vector

`quiet` a numeric vector

`relaxed` a numeric vector

`sad` a numeric vector

`satisfied` a numeric vector

`scared` a numeric vector

`serene` a numeric vector

`sleepy` a numeric vector

`sluggish` a numeric vector

`sociable` a numeric vector

`sorry` a numeric vector

`still` a numeric vector

`strong` a numeric vector

`surprised` a numeric vector

tense a numeric vector

tired a numeric vector

tranquil a numeric vector

unhappy a numeric vector

upset a numeric vector

vigorous a numeric vector

wakeful a numeric vector

warmhearted a numeric vector

wide.awake a numeric vector

alone a numeric vector

kindly a numeric vector

scornful a numeric vector

EA Thayer's Energetic Arousal Scale

TA Thayer's Tense Arousal Scale

PA Positive Affect scale

NegAff Negative Affect scale

Extraversion Extraversion from the Eysenck Personality Inventory

Neuroticism Neuroticism from the Eysenck Personality Inventory

Lie Lie from the EPI

Sociability The sociability subset of the Extraversion Scale

Impulsivity The impulsivity subset of the Extraversions Scale

MSQ_Time Time of day the data were collected

MSQ_Round Rounded time of day

TOD a numeric vector

TOD24 a numeric vector

ID subject ID

condition What was the experimental condition after the msq was given

scale a factor with levels msq r original or revised msq

exper Which study were the data collected: a factor with levels AGES BING BORN CART CITY COPE EMIT FAST Fern FILM FLAT Gray imps item knob MAPS mite pat-1 pat-2 PATS post RAFT Rim.1 Rim.2 rob-1 rob-2 ROG1 ROG2 SALT sam-1 sam-2 SAVE/PATS sett swam swam-2 TIME VALE-1 VALE-2 VIEW

Details

The Motivational States Questionnaire (MSQ) is composed of 72 items, which represent the full affective range (Revelle & Anderson, 1998). The MSQ consists of 20 items taken from the Activation-Deactivation Adjective Check List (Thayer, 1986), 18 from the Positive and Negative Affect Schedule (PANAS, Watson, Clark, & Tellegen, 1988) along with the items used by Larsen and Diener (1992). The response format was a four-point scale that corresponds to Russell and Carroll's (1999)

"ambiguous–likely-unipolar format" and that asks the respondents to indicate their current standing (“at this moment") with the following rating scale:
0——————-1——————-2——————-3
Not at all A little Moderately Very much

The original version of the MSQ included 72 items. Intermediate analyses (done with 1840 subjects) demonstrated a concentration of items in some sections of the two dimensional space, and a paucity of items in others. To begin correcting this, 3 items from redundantly measured sections (alone, kindly, scornful) were removed, and 5 new ones (anxious, cheerful, idle, inactive, and tranquil) were added. Thus, the correlation matrix is missing the correlations between items anxious, cheerful, idle, inactive, and tranquil with alone, kindly, and scornful.

Procedure. The data were collected over nine years, as part of a series of studies examining the effects of personality and situational factors on motivational state and subsequent cognitive performance. In each of 38 studies, prior to any manipulation of motivational state, participants signed a consent form and filled out the MSQ. (The procedures of the individual studies are irrelevant to this data set and could not affect the responses to the MSQ, since this instrument was completed before any further instructions or tasks). Some MSQ post test (after manipulations) is available in affect.

The EA and TA scales are from Thayer, the PA and NA scales are from Watson et al. (1988). Scales and items:

Energetic Arousal: active, energetic, vigorous, wakeful, wide.awake, full.of.pep, lively, -sleepy, -tired, - drowsy (ADACL)

Tense Arousal: Intense, Jittery, fearful, tense, clutched up, -quiet, -still, - placid, - calm, -at rest (ADACL)

Positive Affect: active, alert, attentive, determined, enthusiastic, excited, inspired, interested, proud, strong (PANAS)

Negative Affect: afraid, ashamed, distressed, guilty, hostile, irritable , jittery, nervous, scared, upset (PANAS)

The PA and NA scales can in turn can be thought of as having subscales: (See the PANAS-X) Fear: afraid, scared, nervous, jittery (not included frightened, shaky) Hostility: angry, hostile, irritable, (not included: scornful, disgusted, loathing guilt: ashamed, guilty, (not included: blameworthy, angry at self, disgusted with self, dissatisfied with self) sadness: alone, blue, lonely, sad, (not included: downhearted) joviality: cheerful, delighted, energetic, enthusiastic, excited, happy, lively, (not included: joyful) self-assurance: proud, strong, confident, (not included: bold, daring, fearless) attentiveness: alert, attentive, determined (not included: concentrating)

The next set of circumplex scales were taken (I think) from Larsen and Diener (1992). High activation: active, aroused, surprised, intense, astonished Activated PA: elated, excited, enthusiastic, lively Unactivated NA : calm, serene, relaxed, at rest, content, at ease PA: happy, warmhearted, pleased, cheerful, delighted Low Activation: quiet, inactive, idle, still, tranquil Unactivated PA: dull, bored, sluggish, tired, drowsy NA: sad, blue, unhappy, gloomy, grouchy Activated NA: jittery, anxious, nervous, fearful, distressed.

Keys for these separate scales are shown in the examples.

In addition to the MSQ, there are 5 scales from the Eysenck Personality Inventory (Extraversion, Impulsivity, Sociability, Neuroticism, Lie). The Imp and Soc are subsets of the the total extraversion scale.

Source

Data collected at the Personality, Motivation, and Cognition Laboratory, Northwestern University.

References

Rafaeli, Eshkol and Revelle, William (2006), A premature consensus: Are happiness and sadness truly opposite affects? Motivation and Emotion, 30, 1, 1-12.

Revelle, W. and Anderson, K.J. (1998) Personality, motivation and cognitive performance: Final report to the Army Research Institute on contract MDA 903-93-K-0008. (http://www.personality-project.org/revelle/publications/ra.ari.98.pdf).

Thayer, R.E. (1989) The biopsychology of mood and arousal. Oxford University Press. New York, NY.

Watson,D., Clark, L.A. and Tellegen, A. (1988) Development and validation of brief measures of positive and negative affect: The PANAS scales. Journal of Personality and Social Psychology, 54(6):1063-1070.

See Also

affect for an example of the use of some of these adjectives in a mood manipulation study.

make.keys, scoreItems and scoreOverlap for instructions on how to score multiple scales with and without item overlap. Also see fa and fa.extension for instructions on how to do factor analyses or factor extension.

Examples

```
data(msq)
if(FALSE){ #not run in the interests of time
#basic descriptive statistics
describe(msq)
}
#score them for 20 short scales -- note that these have item overlap
#The first 2 are from Thayer
#The next 2 are classic positive and negative affect
#The next 9 are circumplex scales
#the last 7 are msq estimates of PANASX scales (missing some items)
keys <- make.keys(msq[1:75],list(
EA = c("active", "energetic", "vigorous", "wakeful", "wide.awake", "full.of.pep",
       "lively", "-sleepy", "-tired", "-drowsy"),
TA =c("intense", "jittery", "fearful", "tense", "clutched.up", "-quiet", "-still",
       "-placid", "-calm", "-at.rest") ,
PA =c("active", "excited", "strong", "inspired", "determined", "attentive",
          "interested", "enthusiastic", "proud", "alert"),
NAf =c("jittery", "nervous", "scared", "afraid", "guilty", "ashamed", "distressed",
         "upset", "hostile", "irritable" ),
HAct = c("active", "aroused", "surprised", "intense", "astonished"),
aPA = c("elated", "excited", "enthusiastic", "lively"),
uNA = c("calm", "serene", "relaxed", "at.rest", "content", "at.ease"),
pa = c("happy", "warmhearted", "pleased", "cheerful", "delighted" ),
LAct = c("quiet", "inactive", "idle", "still", "tranquil"),
```

```
uPA =c( "dull", "bored", "sluggish", "tired", "drowsy"),
naf = c( "sad", "blue", "unhappy", "gloomy", "grouchy"),
aNA = c("jittery", "anxious", "nervous", "fearful", "distressed"),
Fear = c("afraid" , "scared" , "nervous" , "jittery" ) ,
Hostility = c("angry" ,  "hostile", "irritable", "scornful" ),
Guilt = c("guilty" , "ashamed" ),
Sadness = c( "sad"  , "blue" , "lonely",  "alone" ),
Joviality =c("happy","delighted", "cheerful", "excited", "enthusiastic", "lively", "energetic"),
Self.Assurance=c( "proud","strong" , "confident" , "-fearful" ),
Attentiveness = c("alert" , "determined" , "attentive" )
#acquiscence = c("sleepy" ,  "wakeful" ,  "relaxed","tense")
   ))

msq.scores <- scoreItems(keys,msq[1:75])

#show a circumplex structure for the non-overlapping items
fcirc <- fa(msq.scores$scores[,5:12],2)
fa.plot(fcirc,labels=colnames(msq.scores$scores)[5:12])

#now, find the correlations corrected for item overlap
msq.overlap <- scoreOverlap(keys,msq[1:75])
f2 <- fa(msq.overlap$cor,2)
fa.plot(f2,labels=colnames(msq.overlap$cor),title="2 dimensions of affect, corrected for overlap")
if(FALSE) {
#extend this solution to EA/TA  NA/PA space
fe  <- fa.extension(cor(msq.scores$scores[,5:12],msq.scores$scores[,1:4]),fcirc)
fa.diagram(fcirc,fe=fe,main="Extending the circumplex structure to  EA/TA and PA/NA ")

#show the 2 dimensional structure
f2 <- fa(msq[1:72],2)
fa.plot(f2,labels=colnames(msq)[1:72],title="2 dimensions of affect at the item level")

#sort them by polar coordinates
round(polar(f2),2)
}
```

---

mssd *Find von Neuman's Mean Square of Successive Differences*

Description

Von Neuman et al. (1941) discussed the Mean Square of Successive Differences as a measure of variability that takes into account gradual shifts in mean. This is appropriate when studying errors in ballistics or variability and stability in mood when studying affect. For random data, this will be twice the variance, but for data with a sequential order and a positive autocorrelation, this will be much smaller. This is just an application of the diff an ny functions

Usage

```
mssd(x,group=NULL, lag = 1,na.rm=TRUE)
rmssd(x,group=NULL, lag=1, na.rm=TRUE)
```

Arguments

x	a vector, data.frame or matrix
lag	the lag to use when finding diff
group	A column of the x data.frame to be used for grouping
na.rm	Should missing data be removed?

Details

When examining multiple measures within subjects, it is sometimes useful to consider the variability of trial by trial observations in addition to the over all between trial variation. The Mean Square of Successive Differences (mssd) and root mean square of successive differences (rmssd) find the variance or standard deviation of the trial to trial differences.

$\sigma^2 = \Sigma(x_i - x_{i+1})^2/(n-1)$

In the case of multiple subjects (groups) with multiple observations per subject (group), specify the grouping variable will produce output for each group.

Similar functions are available in the matrixStats package. This is just the variance and standard deviation applied to the result from the diff function.

Value

The variance (mssd) or standard deviation (rmssd) of the lagged differences.

Author(s)

William Revelle

References

Von Neumann, J., Kent, R., Bellinson, H., and Hart, B. (1941). The mean square successive difference. The Annals of Mathematical Statistics, pages 153-162.

See Also

See Also rmssd for the standard deviation or describe for more conventional statistics. describeBy and statsBy give group level statistics.

Examples

```
t <- seq(-pi, pi, .1)
trial <- 1: length(t)
gr <- trial %% 8
c <- cos(t)
ts <- sample(t,length(t))
```

```
cs <- cos(ts)
x.df <- data.frame(trial,gr,t,c,ts,cs)
rmssd(x.df)
rmssd(x.df,gr)
describe(x.df)
#pairs.panels(x.df)
```

`multi.hist` *Multiple histograms with density and normal fits on one page*

Description

Given a matrix or data.frame, produce histograms for each variable in a "matrix" form. Include normal fits and density distributions for each plot.

The number of rows and columns may be specified, or calculated. May be used for single variables.

Usage

```
multi.hist(x,nrow=NULL,ncol=NULL,density=TRUE,freq=FALSE,bcol="white",
      dcol=c("black","black"),dlty=c("dashed","dotted"),
      main="Histogram, Density, and Normal Fit",...)
histBy(x,var,group,density=TRUE,alpha=.5,breaks=21,col,xlab,
            main="Histograms by group",...)
```

Arguments

`x`	matrix or data.frame
`var`	The variable in x to plot in histBy
`group`	The name of the variable in x to use as the grouping variable
`nrow`	number of rows in the plot
`ncol`	number of columns in the plot
`density`	density=TRUE, show the normal fits and density distributions
`freq`	freq=FALSE shows probability densities and density distribution, freq=TRUE shows frequencies
`bcol`	Color for the bars
`dcol`	The color(s) for the normal and the density fits. Defaults to black.
`dlty`	The line type (lty) of the normal and density fits. (specify the optional graphic parameter lwd to change the line size)
`main`	title for each panel
`xlab`	Label for the x variable
`breaks`	The number of breaks in histBy (see hist)
`alpha`	The degree of transparency of the overlapping bars in histBy
`col`	A vector of colors in histBy (defaults to the rainbow)
`...`	additional graphic parameters (e.g., col)

Author(s)

William Revelle

See Also

bi.bars for drawing pairwise histograms

Examples

```
multi.hist(sat.act)
multi.hist(sat.act,bcol="red")
multi.hist(sat.act,dcol="blue")  #make both lines blue
multi.hist(sat.act,dcol= c("blue","red"),dlty=c("dotted", "solid"))
multi.hist(sat.act,freq=TRUE)   #show the frequency plot
multi.hist(sat.act,nrow=2)
histBy(sat.act,"SATQ","gender")
```

---

neo *NEO correlation matrix from the NEO_PI_R manual*

Description

The NEO.PI.R is a widely used personality test to assess 5 broad factors (Neuroticism, Extraversion, Openness, Agreeableness and Conscientiousness) with six facet scales for each factor. The correlation matrix of the facets is reported in the NEO.PI.R manual for 1000 subjects.

Usage

```
data(neo)
```

Format

A data frame of a 30 x 30 correlation matrix with the following 30 variables.

N1 Anxiety

N2 AngryHostility

N3 Depression

N4 Self-Consciousness

N5 Impulsiveness

N6 Vulnerability

E1 Warmth

E2 Gregariousness

E3 Assertiveness

E4 Activity

E5 Excitement-Seeking

E6 PositiveEmotions

O1 Fantasy

O2 Aesthetics

O3 Feelings

O4 Ideas

O5 Actions

O6 Values

A1 Trust

A2 Straightforwardness

A3 Altruism

A4 Compliance

A5 Modesty

A6 Tender-Mindedness

C1 Competence

C2 Order

C3 Dutifulness

C4 AchievementStriving

C5 Self-Discipline

C6 Deliberation

Details

The past thirty years of personality research has led to a general consensus on the identification of major dimensions of personality. Variously known as the “Big 5" or the “Five Factor Model", the general solution represents 5 broad domains of personal and interpersonal experience. Neuroticism and Extraversion are thought to reflect sensitivity to negative and positive cues from the environment and the tendency to withdraw or approach. Openness is sometimes labeled as Intellect and reflects an interest in new ideas and experiences. Agreeableness and Conscientiousness reflect tendencies to get along with others and to want to get ahead.

The factor structure of the NEO suggests five correlated factors as well as two higher level factors. The NEO was constructed with 6 “facets" for each of the five broad factors.

Source

Costa, Paul T. and McCrae, Robert R. (1992) (NEO PI-R) professional manual. Psychological Assessment Resources, Inc. Odessa, FL. (with permission of the author and the publisher)

References

Digman, John M. (1990) Personality structure: Emergence of the five-factor model. Annual Review of Psychology. 41, 417-440.

John M. Digman (1997) Higher-order factors of the Big Five. Journal of Personality and Social Psychology, 73, 1246-1256.

McCrae, Robert R. and Costa, Paul T., Jr. (1999) A Five-Factor theory of personality. In Pervin, Lawrence A. and John, Oliver P. (eds) Handbook of personality: Theory and research (2nd ed.) 139-153. Guilford Press, New York. N.Y.

Revelle, William (1995), Personality processes, Annual Review of Psychology, 46, 295-328.

Joshua Wilt and William Revelle (2009) Extraversion and Emotional Reactivity. In Mark Leary and Rick H. Hoyle (eds). Handbook of Individual Differences in Social Behavior. Guilford Press, New York, N.Y.

Examples

```
data(neo)
n5 <- fa(neo,5)
neo.keys <- make.keys(30,list(N=c(1:6),E=c(7:12),O=c(13:18),A=c(19:24),C=c(25:30)))
n5p <- target.rot(n5,neo.keys) #show a targeted rotation for simple structure
n5p
```

---

omega   *Calculate McDonald's omega estimates of general and total factor saturation*

Description

McDonald has proposed coefficient omega as an estimate of the general factor saturation of a test. One way to find omega is to do a factor analysis of the original data set, rotate the factors obliquely, do a Schmid Leiman transformation, and then find omega. This function estimates omega as suggested by McDonald by using hierarchical factor analysis (following Jensen). A related option is to define the model using omega and then perform a confirmatory factor analysis using the sem package. This is done by omegaSem and omegaFromSem.

Usage

```
omega(m,nfactors=3,fm="minres",n.iter=1,p=.05,poly=FALSE,key=NULL,
    flip=TRUE,digits=2, title="Omega",sl=TRUE,labels=NULL,
   plot=TRUE,n.obs=NA,rotate="oblimin",Phi=NULL,option="equal",covar=FALSE, ...)
omegaSem(m,nfactors=3,fm="minres",key=NULL,flip=TRUE,digits=2,title="Omega",
  sl=TRUE,labels=NULL, plot=TRUE,n.obs=NA,rotate="oblimin",
  Phi = NULL, option="equal",...)

omegah(m,nfactors=3,fm="minres",key=NULL,flip=TRUE,
digits=2,title="Omega",sl=TRUE,labels=NULL, plot=TRUE,
   n.obs=NA,rotate="oblimin",Phi = NULL,option="equal",covar=FALSE,...)
```

Arguments

m	A correlation matrix, or a data.frame/matrix of data, or (if Phi is specified, an oblique factor pattern matrix
nfactors	Number of factors believed to be group factors
n.iter	How many replications to do in omega for bootstrapped estimates
fm	factor method (the default is minres) fm="pa" for principal axes, fm="minres" for a minimum residual (OLS) solution, fm="pc" for principal components (see note), or fm="ml" for maximum likelihood.
poly	should the correlation matrix be found using polychoric/tetrachoric or normal Pearson correlations
key	a vector of +/- 1s to specify the direction of scoring of items. The default is to assume all items are positively keyed, but if some items are reversed scored, then key should be specified.
flip	If flip is TRUE, then items are automatically flipped to have positive correlations on the general factor. Items that have been reversed are shown with a - sign.
p	probability of two tailed conference boundaries
digits	if specified, round the output to digits
title	Title for this analysis
sl	If plotting the results, should the Schmid Leiman solution be shown or should the hierarchical solution be shown? (default sl=TRUE)
labels	If plotting, what labels should be applied to the variables? If not specified, will default to the column names.
plot	plot=TRUE (default) calls omega.diagram, plot =FALSE does not. If Rgraphviz is available, then `omega.graph` may be used separately.
n.obs	Number of observations - used for goodness of fit statistic
rotate	What rotation to apply? The default is oblimin, the alternatives include simplimax, Promax, cluster and target. target will rotate to an optional keys matrix (See `target.rot`)
Phi	If specified, then omega is found from the pattern matrix (m) and the factor intercorrelation matrix (Phi).
option	In the two factor case (not recommended), should the loadings be equal, emphasize the first factor, or emphasize the second factor. See in particular the option parameter in `schmid` for treating the case of two group factors.
covar	defaults to FALSE and the correlation matrix is found (standardized variables.) If TRUE, the do the calculations on the unstandardized variables and use covariances.
...	Allows additional parameters to be passed through to the factor routines.

Details

“Many scales are assumed by their developers and users to be primarily a measure of one latent variable. When it is also assumed that the scale conforms to the effect indicator model of measurement (as is almost always the case in psychological assessment), it is important to support such an interpretation with evidence regarding the internal structure of that scale. In particular, it is important to examine two related properties pertaining to the internal structure of such a scale. The first property relates to whether all the indicators forming the scale measure a latent variable in common.

The second internal structural property pertains to the proportion of variance in the scale scores (derived from summing or averaging the indicators) accounted for by this latent variable that is common to all the indicators (Cronbach, 1951; McDonald, 1999; Revelle, 1979). That is, if an effect indicator scale is primarily a measure of one latent variable common to all the indicators forming the scale, then that latent variable should account for the majority of the variance in the scale scores. Put differently, this variance ratio provides important information about the sampling fluctuations when estimating individuals' standing on a latent variable common to all the indicators arising from the sampling of indicators (i.e., when dealing with either Type 2 or Type 12 sampling, to use the terminology of Lord, 1956). That is, this variance proportion can be interpreted as the square of the correlation between the scale score and the latent variable common to all the indicators in the infinite universe of indicators of which the scale indicators are a subset. Put yet another way, this variance ratio is important both as reliability and a validity coefficient. This is a reliability issue as the larger this variance ratio is, the more accurately one can predict an individual's relative standing on the latent variable common to all the scale's indicators based on his or her observed scale score. At the same time, this variance ratio also bears on the construct validity of the scale given that construct validity encompasses the internal structure of a scale." (Zinbarg, Yovel, Revelle, and McDonald, 2006).

McDonald has proposed coefficient omega_hierarchical (ω_h) as an estimate of the general factor saturation of a test. Zinbarg, Revelle, Yovel and Li (2005) `http://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf` compare McDonald's ω_h to Cronbach's α and Revelle's β. They conclude that ω_h is the best estimate. (See also Zinbarg et al., 2006 and Revelle and Zinbarg (2009)).

One way to find ω_h is to do a factor analysis of the original data set, rotate the factors obliquely, factor that correlation matrix, do a Schmid-Leiman (schmid) transformation to find general factor loadings, and then find ω_h. Here we present code to do that.

ω_h differs as a function of how the factors are estimated. Four options are available, three use the `fa` function but with different factoring methods: the default does a minres factor solution, fm="pa" does a principle axes factor analysis fm="mle" does a maximum likelihood solution; fm="pc" does a principal components analysis using (principal).

For ability items, it is typically the case that all items will have positive loadings on the general factor. However, for non-cognitive items it is frequently the case that some items are to be scored positively, and some negatively. Although probably better to specify which directions the items are to be scored by specifying a key vector, if flip =TRUE (the default), items will be reversed so that they have positive loadings on the general factor. The keys are reported so that scores can be found using the `scoreItems` function. Arbitrarily reversing items this way can overestimate the general factor. (See the example with a simulated circumplex).

β, an alternative to ω_h, is defined as the worst split half reliability (Revelle, 1979). It can be estimated by using ICLUST (a hierarchical clustering algorithm originally developed for main frames and written in Fortran and that is now part of the psych package. (For a very complimentary review of why the ICLUST algorithm is useful in scale construction, see Cooksey and Soutar, 2005).

The `omega` function uses exploratory factor analysis to estimate the ω_h coefficient. It is important to remember that "A recommendation that should be heeded, regardless of the method chosen to estimate ω_h, is to always examine the pattern of the estimated general factor loadings prior to estimating ω_h. Such an examination constitutes an informal test of the assumption that there is a latent variable common to all of the scale's indicators that can be conducted even in the context of EFA. If the loadings were salient for only a relatively small subset of the indicators, this would suggest that there is no true general factor underlying the covariance matrix. Just such an informal assumption test would have afforded a great deal of protection against the possibility of misinterpreting the misleading ω_h estimates occasionally produced in the simulations reported here." (Zinbarg et al., 2006, p 137).

A simple demonstration of the problem of an omega estimate reflecting just one of two group factors can be found in the last example.

Diagnostic statistics that reflect the quality of the omega solution include a comparison of the relative size of the g factor eigen value to the other eigen values, the percent of the common variance for each item that is general factor variance (p2), the mean of p2, and the standard deviation of p2. Further diagnostics can be done by describing (describe) the $schmid$sl results.

Although omega_h is uniquely defined only for cases where 3 or more subfactors are extracted, it is sometimes desired to have a two factor solution. By default this is done by forcing the schmid extraction to treat the two subfactors as having equal loadings.

There are three possible options for this condition: setting the general factor loadings between the two lower order factors to be "equal" which will be the sqrt(oblique correlations between the factors) or to "first" or "second" in which case the general factor is equated with either the first or second group factor. A message is issued suggesting that the model is not really well defined. This solution discussed in Zinbarg et al., 2007. To do this in omega, add the option="first" or option="second" to the call.

Although obviously not meaningful for a 1 factor solution, it is of course possible to find the sum of the loadings on the first (and only) factor, square them, and compare them to the overall matrix variance. This is done, with appropriate complaints.

In addition to ω_h, another of McDonald's coefficients is ω_t. This is an estimate of the total reliability of a test.

McDonald's ω_t, which is similar to Guttman's λ_6, `guttman` but uses the estimates of uniqueness (u^2) from factor analysis to find e_j^2. This is based on a decomposition of the variance of a test score, V_x into four parts: that due to a general factor, $\vec{g}$, that due to a set of group factors, $\vec{f}$, (factors common to some but not all of the items), specific factors, $\vec{s}$ unique to each item, and $\vec{e}$, random error. (Because specific variance can not be distinguished from random error unless the test is given at least twice, some combine these both into error).

Letting $\vec{x} = \vec{cg} + \vec{A}f + \vec{D}s + \vec{e}$ then the communality of item$_j$, based upon general as well as group factors, $h_j^2 = c_j^2 + \sum f_{ij}^2$ and the unique variance for the item $u_j^2 = \sigma_j^2(1 - h_j^2)$ may be used to estimate the test reliability. That is, if h_j^2 is the communality of item$_j$, based upon general as well as group factors, then for standardized items, $e_j^2 = 1 - h_j^2$ and

$$\omega_t = \frac{\vec{1}\vec{cc'}\vec{1} + \vec{1}\vec{AA'}\vec{1}'}{V_x} = 1 - \frac{\sum(1 - h_j^2)}{V_x} = 1 - \frac{\sum u^2}{V_x}$$

Because $h_j^2 \geq r_{smc}^2$, $\omega_t \geq \lambda_6$.

It is important to distinguish here between the two ω coefficients of McDonald, 1978 and Equation 6.20a of McDonald, 1999, ω_t and ω_h. While the former is based upon the sum of squared loadings on all the factors, the latter is based upon the sum of the squared loadings on the general factor.

$$\omega_h = \frac{\vec{1}\vec{cc'}\vec{1}}{V_x}$$

Another estimate reported is the omega for an infinite length test with a structure similar to the observed test (omega H asymptotic). This is found by

$$\omega_{limit} = \frac{\vec{1}\vec{cc'}\vec{1}}{\vec{1}\vec{cc'}\vec{1} + \vec{1}\vec{AA'}\vec{1}'}$$

.

Following suggestions by Steve Reise, the Explained Common Variance (ECV) is also reported. This is the ratio of the general factor eigen value to the sum of all of the eigen values. As such, it is a better indicator of unidimensionality than of the amount of test variance accounted for by a general factor.

The input to omega may be a correlation matrix or a raw data matrix, or a factor pattern matrix with the factor intercorrelations (Phi) matrix.

`omega` is an exploratory factor analysis function that uses a Schmid-Leiman transformation. `omegaSem` first calls `omega` and then takes the Schmid-Leiman solution, converts this to a confirmatory sem model and then calls the sem package to conduct a confirmatory model. ω_h is then calculated from the CFA output. Although for well behaved problems, the efa and cfa solutions will be practically identical, the CFA solution will not always agree with the EFA solution. In particular, the estimated R^2 will sometimes exceed 1. (An example of this is the Harman 24 cognitive abilities problem.)

In addition, not all EFA solutions will produce workable CFA solutions. Model misspecifications will lead to very strange CFA estimates.

`omegaFromSem` takes the output from a sem model and uses it to find ω_h. The estimate of factor indeterminacy, found by the multiple R^2 of the variables with the factors, will not match that found by the EFA model. In particular, the estimated R^2 will sometimes exceed 1. (An example of this is the Harman 24 cognitive abilities problem.)

The notion of omega may be applied to the individual factors as well as the overall test. A typical use of omega is to identify subscales of a total inventory. Some of that variability is due to the general factor of the inventory, some to the specific variance of each subscale. Thus, we can find a number of different omega estimates: what percentage of the variance of the items identified with each subfactor is actually due to the general factor. What variance is common but unique to the subfactor, and what is the total reliable variance of each subfactor. These results are reported in omega.group object and in the last few lines of the normal output.

The summary of the omega object is a reduced set of the most useful output.

The various objects returned from omega include:

Value

`omega hierarchical`	The ω_h coefficient
`omega.lim`	The limit of ω_h as the test becomes infinitly large
`omega total`	The $omega_t$ coefficient
`alpha`	Cronbach's α
`schmid`	The Schmid Leiman transformed factor matrix and associated matrices
`schmid$sl`	The g factor loadings as well as the residualized factors
`schmid$orthog`	Varimax rotated solution of the original factors

`schmid$oblique` The oblimin or promax transformed factors

`schmid$phi` the correlation matrix of the oblique factors

`schmid$gloading`
The loadings on the higher order, g, factor of the oblimin factors

`key` A vector of -1 or 1 showing which direction the items were scored.

`model` a matrix suitable to be given to the sem function for structure equation models

`sem` The output from a sem analysis

`omega.group` The summary statistics for the omega total, omega hierarchical (general) and omega within each group.

`scores` Factor score estimates are found for the Schmid-Leiman solution. To get scores for the hierarchical model see the note.

`various fit statistics`
various fit statistics, see output

Note

Requires the GPArotation package.

The default rotation uses oblimin from the GPArotation package. Alternatives include the simplimax function, as well as `Promax`.

If the factor solution leads to an exactly orthogonal solution (probably only for demonstration data sets), then use the rotate="Promax" option to get a solution.

`omegaSem` requires the sem package. `omegaFromSem` uses the output from the sem package.

`omega` may be run on raw data (finding either Pearson or tetrachoric/polychoric corrlations, depending upon the poly option) a correlation matrix, a polychoric correlation matrix (found by e.g., `polychoric`), or the output of a previous omega run. This last case is particularly useful when working with categorical data using the poly=TRUE option. For in this case, most of the time is spent in finding the correlation matrix. The matrix is saved as part of the omega output and may be used as input for subsequent runs. A similar feature is found in `irt.fa` where the output of one analysis can be taken as the input to the subsequent analyses.

However, simulations based upon tetrachoric and polychoric correlations suggest that although the structure is better defined, that the estimates of omega are inflated over the true general factor saturation.

Omega returns factor scores based upon the Schmid-Leiman transformation. To get the hierarchical factor scores, it is necessary to do this outside of omega. See the example (not run).

Consider the case of the raw data in an object data. Then

f3 <- fa(data,3,scores="tenBerge", oblique.rotation=TRUE f1 <- fa(f3$scores) hier.scores <- data.frame(f1$scores,f3$scores)

When doing fm="pc", principal components are done for the original correlation matrix, but minres is used when examining the intercomponent correlations. A warning is issued that the method was changed to minres for the higher order solution. omega is a factor model, and finding loadings using principal components will overestimate the resulting solution. This is particularly problematic for the amount of group saturation, and thus the omega.group statistics are overestimates.

The last three lines of omega report "Total, General and Subset omega for each subset". These are available as the omega.group object in the output.

The last of these (omega group) is effectively what Steve Reise calls omegaS for the subset omega.

The omega general is the amount of variance in the group that is accounted for by the general factor, the omega total is the amount of variance in the group accounted for by general + group.

This is based upon a cluster solution (that is to say, every item is assigned to one group) and this is why for first column the omega general and group do not add up to omega total. Some of the variance is found in the cross loadings between groups.

Reise and others like to report the ratio of the second line to the first line (what portion of the reliable variance is general factor) and the third row to the first (what portion of the reliable variance is within group but not general. This may be found by using the omega.group object that is returned by omega. (See the last example.)

Author(s)

http://personality-project.org/revelle.html
Maintainer: William Revelle < revelle@northwestern.edu >

References

http://personality-project.org/r/r.omega.html

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at http://personality-project.org/r/book/

Revelle, W. (1979). Hierarchical cluster analysis and the internal structure of tests. Multivariate Behavioral Research, 14, 57-74. (http://personality-project.org/revelle/publications/iclust.pdf)

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74, 1, 145-154. (http://personality-project.org/revelle/publications/rz09.pdf

Zinbarg, R.E., Revelle, W., Yovel, I., & Li. W. (2005). Cronbach's Alpha, Revelle's Beta, McDonald's Omega: Their relations with each and two alternative conceptualizations of reliability. Psychometrika. 70, 123-133. http://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf

Zinbarg, R., Yovel, I. & Revelle, W. (2007). Estimating omega for structures containing two group factors: Perils and prospects. Applied Psychological Measurement. 31 (2), 135-157.

Zinbarg, R., Yovel, I., Revelle, W. & McDonald, R. (2006). Estimating generalizability to a universe of indicators that all have one attribute in common: A comparison of estimators for omega. Applied Psychological Measurement, 30, 121-144. DOI: 10.1177/0146621605278814 http://apm.sagepub.com/cgi/reprint/30/2/121

See Also

omega.graph ICLUST, ICLUST.graph, VSS, schmid , make.hierarchical

Examples

```
## Not run:
 test.data <- Harman74.cor$cov
# if(!require(GPArotation)) {message("Omega requires GPA rotation" )} else {
      my.omega <- omega(test.data)
      print(my.omega,digits=2)
#}

#create 9 variables with a hierarchical structure
v9 <- sim.hierarchical()
#with correlations of
round(v9,2)
#find omega
v9.omega <- omega(v9,digits=2)
v9.omega

#create 8 items with a two factor solution, showing the use of the flip option
sim2 <- item.sim(8)
omega(sim2)  #an example of misidentification-- remember to look at the loadings matrices.
omega(sim2,2)  #this shows that in fact there is no general factor
omega(sim2,2,option="first") #but, if we define one of the two group factors
     #as a general factor, we get a falsely high omega
#apply omega to analyze 6 mental ability tests
data(ability.cov)   #has a covariance matrix
omega(ability.cov$cov)

#om <- omega(Thurstone)
#round(om$omega.group,2)
#round(om$omega.group[2]/om$omega.group[1],2)  #fraction of reliable that is general variance
# round(om$omega.group[3]/om$omega.group[1],2)  #fraction of reliable that is group variance

#To find factor score estimates for the hierarchical model it is necessary to
#do two extra steps.

#Consider the case of the raw data in an object data.  (An example from simulation)
# set.seed(42)
# gload <- matrix(c(.9,.8,.7),nrow=3)
# fload <- matrix(c(.8,.7,.6,rep(0,9),.7,.6,.5,rep(0,9),.7,.6,.4),   ncol=3)
# data <- sim.hierarchical(gload=gload,fload=fload, n=100000, raw=TRUE)
#
# f3 <- fa(data$observed,3,scores="tenBerge", oblique.scores=TRUE)
# f1 <- fa(f3$scores)

# om <- omega(data$observed,sl=FALSE) #draw the hierarchical figure
# The scores from om are based upon the schmid-leiman factors and although the g factor
# is identical, the group factors are not.
# This is seen in the following correlation matrix
# hier.scores <- cbind(om$scores,f1$scores,f3$scores)
# lowerCor(hier.scores)


## End(Not run)
```

omega.graph *Graph hierarchical factor structures*

Description

Hierarchical factor structures represent the correlations between variables in terms of a smaller set of correlated factors which themselves can be represented by a higher order factor.

Two alternative solutions to such structures are found by the omega function. The correlated factors solutions represents the effect of the higher level, general factor, through its effect on the correlated factors. The other representation makes use of the Schmid Leiman transformation to find the direct effect of the general factor upon the original variables as well as the effect of orthogonal residual group factors upon the items.

Graphic presentations of these two alternatives are helpful in understanding the structure. omega.graph and omega.diagram draw both such structures. Graphs are drawn directly onto the graphics window or expressed in “dot" commands for conversion to graphics using implementations of Graphviz (if using omega.graph).

Using Graphviz allows the user to clean up the Rgraphviz output. However, if Graphviz and Rgraphviz are not available, use omega.diagram.

See the other structural diagramming functions, fa.diagram and structure.diagram.

In addition

Usage

```
omega.diagram(om.results,sl=TRUE,sort=TRUE,labels=NULL,flabels=NULL,cut=.2,
gcut=.2,simple=TRUE,  errors=FALSE, digits=1,e.size=.1,rsize=.15,side=3,
    main=NULL,cex=NULL,color.lines=TRUE,marg=c(.5,.5,1.5,.5),adj=2, ...)
omega.graph(om.results, out.file = NULL,  sl = TRUE, labels = NULL, size = c(8, 6),
    node.font = c("Helvetica", 14), edge.font = c("Helvetica", 10),
    rank.direction=c("RL","TB","LR","BT"), digits = 1, title = "Omega", ...)
```

Arguments

om.results	The output from the omega function
out.file	Optional output file for off line analysis using Graphviz
sl	Orthogonal clusters using the Schmid-Leiman transform (sl=TRUE) or oblique clusters
labels	variable labels
flabels	Labels for the factors (not counting g)
size	size of graphics window
node.font	What font to use for the items
edge.font	What font to use for the edge labels
rank.direction	Defaults to left to right
digits	Precision of labels

cex	control font size
color.lines	Use black for positive, red for negative
marg	The margins for the figure are set to be wider than normal by default
adj	Adjust the location of the factor loadings to vary as factor mod 4 + 1
title	Figure title
main	main figure caption
...	Other options to pass into the graphics packages
e.size	the size to draw the ellipses for the factors. This is scaled by the number of variables.
cut	Minimum path coefficient to draw
gcut	Minimum general factor path to draw
simple	draw just one path per item
sort	sort the solution before making the diagram
side	on which side should errors be drawn?
errors	show the error estimates
rsize	size of the rectangles

Details

While omega.graph requires the Rgraphviz package, omega.diagram does not. codeomega requires the GPArotation package.

Value

clust.graph	A graph object
sem	A matrix suitable to be run throughe the sem function in the sem package.

Note

omega.graph requires rgraphviz. – omega requires GPArotation

Author(s)

http://personality-project.org/revelle.html
Maintainer: William Revelle < revelle@northwestern.edu >

References

http://personality-project.org/r/r.omega.html

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. http://personality-project.org/r/book

Revelle, W. (1979). Hierarchical cluster analysis and the internal structure of tests. Multivariate Behavioral Research, 14, 57-74. (http://personality-project.org/revelle/publications/iclust.pdf)

Zinbarg, R.E., Revelle, W., Yovel, I., & Li. W. (2005). Cronbach's Alpha, Revelle's Beta, McDonald's Omega: Their relations with each and two alternative conceptualizations of reliability. Psychometrika. 70, 123-133. http://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf

Zinbarg, R., Yovel, I., Revelle, W. & McDonald, R. (2006). Estimating generalizability to a universe of indicators that all have one attribute in common: A comparison of estimators for omega. Applied Psychological Measurement, 30, 121-144. DOI: 10.1177/0146621605278814 http://apm.sagepub.com/cgi/reprint/30/2/121

See Also

omega, make.hierarchical, ICLUST.rgraph

Examples

```
#24 mental tests from Holzinger-Swineford-Harman
if(require(GPArotation) ) {om24 <- omega(Harman74.cor$cov,4) } #run omega

#
#example hierarchical structure from Jensen and Weng
if(require(GPArotation) ) {jen.omega <- omega(make.hierarchical())}
```

`outlier` *Find and graph Mahalanobis squared distances to detect outliers*

Description

The Mahalanobis distance is $D^2 = (x-\mu)'\Sigma^-1(x-\mu)$ where Σ is the covariance of the x matrix. D2 may be used as a way of detecting outliers in distribution. Large D2 values, compared to the expected Chi Square values indicate an unusual response pattern. The mahalanobis function in stats does not handle missing data.

Usage

```
outlier(x, plot = TRUE, bad = 5,na.rm = TRUE, xlab, ylab, ...)
```

Arguments

x	A data matrix or data.frame
plot	Plot the resulting QQ graph
bad	Label the bad worst values
na.rm	Should missing data be deleted
xlab	Label for x axis
ylab	Label for y axis
...	More graphic parameters, e.g., cex=.8

Details

Adapted from the mahalanobis function and help page from stats.

Value

The D2 values for each case

Author(s)

William Revelle

References

Yuan, Ke-Hai and Zhong, Xiaoling, (2008) Outliers, Leverage Observations, and Influential Cases in Factor Analysis: Using Robust Procedures to Minimize Their Effect, Sociological Methodology, 38, 329-368.

See Also

mahalanobis

Examples

```
#first, just find and graph the outliers
d2 <- outlier(sat.act)
#combine with the data frame and plot it with the outliers highlighted in blue
sat.d2 <- data.frame(sat.act,d2)
pairs.panels(sat.d2,bg=c("yellow","blue")[(d2 > 25)+1],pch=21)
```

---

p.rep — *Find the probability of replication for an F, t, or r and estimate effect size*

Description

The probability of replication of an experimental or correlational finding as discussed by Peter Killeen (2005) is the probability of finding an effect in the same direction upon an exact replication. For articles submitted to Psychological Science, p.rep needs to be reported.

F, t, p and r are all estimates of the size of an effect. But F, t, and p also are also a function of the sample size. Effect size, d prime, may be expressed as differences between means compared to within cell standard deviations, or as a correlation coefficient. These functions convert p, F, and t to d prime and the r equivalent.

Usage

```
p.rep(p = 0.05, n=NULL,twotailed = FALSE)
p.rep.f(F,df2,twotailed=FALSE)
p.rep.r(r,n,twotailed=TRUE)
p.rep.t(t,df,df2=NULL,twotailed=TRUE)
```

Arguments

`p`	conventional probability of statistic (e.g., of F, t, or r)
`F`	The F statistic
`df`	Degrees of freedom of the t-test, or of the first group if unequal sizes
`df2`	Degrees of freedom of the denominator of F or the second group in an unequal sizes t test
`r`	Correlation coefficient
`n`	Total sample size if using r
`t`	t-statistic if doing a t-test or testing significance of a regression slope
`twotailed`	Should a one or two tailed test be used?

Details

The conventional Null Hypothesis Significance Test (NHST) is the likelihood of observing the data given the null hypothesis of no effect. But this tells us nothing about the probability of the null hypothesis. Peter Killeen (2005) introduced the probability of replication as a more useful measure. The probability of replication is the probability that an exact replication study will find a result in the *same direction* as the original result.

p.rep is based upon a 1 tailed probability value of the observed statistic.

Other frequently called for statistics are estimates of the effect size, expressed either as Cohen's d, Hedges g, or the equivalent value of the correlation, r.

For p.rep.t, if the cell sizes are unequal, the effect size estimates are adjusted by the ratio of the mean cell size to the harmonic mean cell size (see Rownow et al., 2000).

Value

`p.rep`	Probability of replication
`dprime`	Effect size (Cohen's d) if more than just p is specified
`prob`	Probability of F, t, or r. Note that this can be either the one-tailed or two tailed probability value.
`r.equivalent`	For t-tests, the r equivalent to the t (see Rosenthal and Rubin(2003), Rosnow, Rosenthal, and Rubin, 2000))

.

Note

The p.rep value is the one tailed probability value of obtaining a result in the same direction.

References

Cummings, Geoff (2005) Understanding the average probability of replication: comment on Killeen 2005). Psychological Science, 16, 12, 1002-1004).

Killeen, Peter H. (2005) An alternative to Null-Hypothesis Significance Tests. Psychological Science, 16, 345-353

Rosenthal, R. and Rubin, Donald B.(2003), r-sub(equivalent): A Simple Effect Size Indicator. Psychological Methods, 8, 492-496.

Rosnow, Ralph L., Rosenthal, Robert and Rubin, Donald B. (2000) Contrasts and correlations in effect-size estimation, Psychological Science, 11. 446-453.

Examples

```
p.rep(.05)  #probability of replicating a result if the original study had a  p = .05
p.rep.f(9.0,98)  #probability of replicating  a result with F = 9.0 with 98 df
p.rep.r(.4,50)    #probability of replicating a result if r =.4 with n = 50
p.rep.t(3,98)   #probability of replicating a result if t = 3 with df =98
p.rep.t(2.14,84,14) #effect of equal sample sizes (see Rosnow et al.)
```

`paired.r` *Test the difference between (un)paired correlations*

Description

Test the difference between two (paired or unpaired) correlations. Given 3 variables, x, y, z, is the correlation between xy different than that between xz? If y and z are independent, this is a simple t-test of the z transformed rs. But, if they are dependent, it is a bit more complicated.

Usage

```
paired.r(xy, xz, yz=NULL, n, n2=NULL,twotailed=TRUE)
```

Arguments

`xy`	r(xy)
`xz`	r(xz)
`yz`	r(yz)
`n`	Number of subjects for first group
`n2`	Number of subjects in second group (if not equal to n)
`twotailed`	Calculate two or one tailed probability values

Details

To find the z of the difference between two independent correlations, first convert them to z scores using the Fisher r-z transform and then find the z of the difference between the two correlations. The default assumption is that the group sizes are the same, but the test can be done for different size groups by specifying n2.

If the correlations are not independent (i.e., they are from the same sample) then the correlation with the third variable r(yz) must be specified. Find a t statistic for the difference of thee two dependent correlations.

Value

a list containing the calculated t or z values and the associated two (or one) tailed probability.

`t`	t test of the difference between two dependent correlations
`p`	probability of the t or of the z
`z`	z test of the difference between two independent correlations

Author(s)

William Revelle

See Also

`r.test` for more tests of independent as well as dependent (paired) tests. `p.rep.r` for the probability of replicating a particular correlation. `cor.test` from stats for testing a single correlation and `corr.test` for finding the values and probabilities of multiple correlations. See also `set.cor` to do multiple correlations from matrix input.

Examples

```
paired.r(.5,.3, .4, 100) #dependent correlations
paired.r(.5,.3,NULL,100) #independent correlations same sample size
paired.r(.5,.3,NULL, 100, 64) # independent correlations, different sample sizes
```

`pairs.panels` *SPLOM, histograms and correlations for a data matrix*

Description

Adapted from the help page for pairs, pairs.panels shows a scatter plot of matrices (SPLOM), with bivariate scatter plots below the diagonal, histograms on the diagonal, and the Pearson correlation above the diagonal. Useful for descriptive statistics of small data sets. If lm=TRUE, linear regression fits are shown for both y by x and x by y. Correlation ellipses are also shown. Points may be given different colors depending upon some grouping variable.

Usage

```
## S3 method for class 'panels'
pairs(x, smooth = TRUE, scale = FALSE, density=TRUE,ellipses=TRUE,
    digits = 2,method="pearson", pch = 20, lm=FALSE,cor=TRUE,jiggle=FALSE,factor=2,
    hist.col="cyan",show.points=TRUE,rug=TRUE, breaks = "Sturges",cex.cor=1,wt=NULL, ...)
```

Arguments

x	a data.frame or matrix
smooth	TRUE draws loess smooths
scale	TRUE scales the correlation font by the size of the absolute correlation.
density	TRUE shows the density plots as well as histograms
ellipses	TRUE draws correlation ellipses
lm	Plot the linear fit rather than the LOESS smoothed fits.
digits	the number of digits to show
method	method parameter for the correlation ("pearson","spearman","kendall")
pch	The plot character (defaults to 20 which is a '.').
cor	If plotting regressions, should correlations be reported?
jiggle	Should the points be jittered before plotting?
factor	factor for jittering (1-5)
hist.col	What color should the histogram on the diagonal be?
show.points	If FALSE, do not show the data points, just the data ellipses and smoothed functions
rug	if TRUE (default) draw a rug under the histogram, if FALSE, don't draw the rug
breaks	If specified, allows control for the number of breaks in the histogram (see the hist function)
cex.cor	If this is specified, this will change the size of the text in the correlations. this allows one to also change the size of the points in the plot by specifying the normal cex values. If just specifying cex, it will change the character size, if cex.cor is specified, then cex will function to change the point size.
wt	If specified, then weight the correlations by a weights matrix (see note for some comments)
...	other options for pairs

Details

Shamelessly adapted from the pairs help page. Uses panel.cor, panel.cor.scale, and panel.hist, all taken from the help pages for pairs. Also adapts the ellipse function from John Fox's car package.

pairs.panels is most useful when the number of variables to plot is less than about 6-10. It is particularly useful for an initial overview of the data.

To show different groups with different colors, use a plot character (pch) between 21 and 25 and then set the background color to vary by group. (See the second example).

When plotting more than about 10 variables, it is useful to set the gap parameter to something less than 1 (e.g., 0). Alternatively, consider using cor.plot

In addition, when plotting more than about 100-200 cases, it is useful to set the plotting character to be a point. (pch=".")

Sometimes it useful to draw the correlation ellipses and best fitting lowess without the points. (points.false=TRUE).

Value

A scatter plot matrix (SPLOM) is drawn in the graphic window. The lower off diagonal draws scatter plots, the diagonal histograms, the upper off diagonal reports the Pearson correlation (with pairwise deletion).

If lm=TRUE, then the scatter plots are drawn above and below the diagonal, each with a linear regression fit. Useful to show the difference between regression lines.

Note

If the data are either categorical or character, this is flagged with an astrix for the variable name. If character, they are changed to factors before plotting.

The wt parameter allows for scatter plots of the raw data while showing the weighted correlation matrix (found by using cor.wt). The current implementation uses the first two columns of the weights matrix for all analyses. This is useful, but not perfect. The use of this option would be to plot the means from a statsBy analysis and then display the weighted correlations by specifying the means and ns from the statsBy run. See the final (not run) example.

See Also

pairs which is the base from which pairs.panels is derived, cor.plot to do a heat map of correlations, and scatter.hist to draw a single correlation plot with histograms and best fitted lines.

To find the probability "significance" of the correlations using normal theory, use corr.test. To find confidence intervals using boot strapping procedures, use cor.ci. To graphically show confidence intervals, see cor.plot.upperLowerCi.

Examples

```
pairs.panels(attitude)   #see the graphics window
data(iris)
pairs.panels(iris[1:4],bg=c("red","yellow","blue")[iris$Species],
        pch=21,main="Fisher Iris data by Species") #to show color grouping

pairs.panels(iris[1:4],bg=c("red","yellow","blue")[iris$Species],
  pch=21+as.numeric(iris$Species),main="Fisher Iris data by Species",hist.col="red")
   #to show changing the diagonal

#demonstrate not showing the data points
data(sat.act)
pairs.panels(sat.act,show.points=FALSE)
#better yet is to show the points as a period
```

```
pairs.panels(sat.act,pch=".")
#show many variables with 0 gap between scatterplots
# data(bfi)
# pairs.panels(bfi,show.points=FALSE,gap=0)

#plot raw data points and then the weighted correlations.
#output from statsBy
sb <- statsBy(sat.act,"education")
pairs.panels(sb$mean,wt=sb$n)  #report the weighted correlations
#compare with
pairs.panels(sb$mean) #unweighed correlations
```

`parcels` *Find miniscales (parcels) of size 2 or 3 from a set of items*

Description

Given a set of n items, form n/2 or n/3 mini scales or parcels of the most similar pairs or triplets of items. These may be used as the basis for subsequent scale construction or multivariate (e.g., factor) analysis.

Usage

```
parcels(x, size = 3, max = TRUE, flip=TRUE,congruence = FALSE)
keysort(keys)
```

Arguments

`x`	A matrix/dataframe of items or a correlation/covariance matrix of items
`size`	Form parcels of size 2 or size 3
`flip`	if flip=TRUE, negative correlations lead to at least one item being negatively scored
`max`	Should item correlation/covariance be adjusted for their maximum correlation
`congruence`	Should the correlations be converted to congruence coefficients?
`keys`	Sort a matrix of keys to reflect item order as much as possible

Details

Items used in measuring ability or other aspects of personality are typically not very reliable. One suggestion has been to form items into homogeneous item composites (HICs), Factorially Homogeneous Item Dimensions (FHIDs) or mini scales (parcels). Parcelling may be done rationally, factorially, or empirically based upon the structure of the correlation/covariance matrix. `link{parcels}` facilitates the finding of parcels by forming a keys matrix suitable for using in `score.items`. These keys represent the n/2 most similar pairs or the n/3 most similar triplets.

The algorithm is straightforward: For size = 2, the correlation matrix is searched for the highest correlation. These two items form the first parcel and are dropped from the matrix. The procedure is repeated until there are no more pairs to form.

For size=3, the three items with the greatest sum of variances and covariances with each other is found. This triplet is the first parcel. All three items are removed and the procedure then identifies the next most similar triplet. The procedure repeats until n/3 parcels are identified.

Value

`keys`	A matrix of scoring keys to be used to form mini scales (parcels) These will be in order of importance, that is, the first parcel (P1) will reflect the most similar pair or triplet. The keys may also be sorted by average item order by using the keysort function.

Author(s)

William Revelle

References

Cattell, R. B. (1956). Validation and intensification of the sixteen personality factor questionnaire. Journal of Clinical Psychology , 12 (3), 205 -214.

See Also

`score.items` to score the parcels or `iclust` for an alternative way of forming item clusters.

Examples

```
parcels(Thurstone)
keys <- parcels(bfi)
keys <- keysort(keys)
score.items(keys,bfi)
```

`partial.r` *Find the partial correlations for a set (x) of variables with set (y) removed.*

Description

A straightforward application of matrix algebra to remove the effect of the variables in the y set from the x set. Input may be either a data matrix or a correlation matrix. Variables in x and y are specified by location.

Usage

```
partial.r(m, x, y)
```

Arguments

m	A data or correlation matrix
x	The variable numbers associated with the X set.
y	The variable numbers associated with the Y set

Details

It is sometimes convenient to partial the effect of a number of variables (e.g., sex, age, education) out of the correlations of another set of variables. This could be done laboriously by finding the residuals of various multiple correlations, and then correlating these residuals. The matrix algebra alternative is to do it directly. To find the confidence intervals and "significance" of the correlations, use the `corr.p` function with n = n - s where s is the numer of covariates.

Value

The matrix of partial correlations.

Author(s)

William Revelle

References

Revelle, W. (in prep) An introduction to psychometric theory with applications in R. To be published by Springer. (working draft available at http://personality-project.org/r/book/

See Also

`mat.regress` for a similar application for regression

Examples

```
jen <- make.hierarchical()    #make up a correlation matrix
round(jen[1:5,1:5],2)
par.r <- partial.r(jen,c(1,3,5),c(2,4))
cp <- corr.p(par.r,n=98)  #assumes the jen data based upon n =100.
print(cp,short=FALSE)  #show the confidence intervals as well
```

---

peas *Galton's Peas*

Description

Francis Galton introduced the correlation coefficient with an analysis of the similarities of the parent and child generation of 700 sweet peas.

Usage

```
data(peas)
```

Format

A data frame with 700 observations on the following 2 variables.

`parent` The mean diameter of the mother pea for 700 peas

`child` The mean diameter of the daughter pea for 700 sweet peas

Details

Galton's introduction of the correlation coefficient was perhaps the most important contribution to the study of individual differences. This data set allows a graphical analysis of the data set. There are two different graphic examples. One shows the regression lines for both relationships, the other finds the correlation as well.

Source

Stanton, Jeffrey M. (2001) Galton, Pearson, and the Peas: A brief history of linear regression for statistics intstructors, Journal of Statistics Education, 9. (retrieved from the web from http://www.amstat.org/publications/jse/v
reproduces the table from Galton, 1894, Table 2.

The data were generated from this table.

References

Galton, Francis (1877) Typical laws of heredity. paper presented to the weekly evening meeting of the Royal Institution, London. Volume VIII (66) is the first reference to this data set. The data appear in

Galton, Francis (1894) Natural Inheritance (5th Edition), New York: MacMillan).

See Also

The other Galton data sets: `heights`, `galton`,`cubits`

Examples

```
data(peas)
pairs.panels(peas,lm=TRUE,xlim=c(14,22),ylim=c(14,22),main="Galton's Peas")
describe(peas)
pairs.panels(peas,main="Galton's Peas")
```

`phi` *Find the phi coefficient of correlation between two dichotomous variables*

Description

Given a 1 x 4 vector or a 2 x 2 matrix of frequencies, find the phi coefficient of correlation. Typical use is in the case of predicting a dichotomous criterion from a dichotomous predictor.

Usage

```
phi(t, digits = 2)
```

Arguments

`t`	a 1 x 4 vector or a 2 x 2 matrix
`digits`	round the result to digits

Details

In many prediction situations, a dichotomous predictor (accept/reject) is validated against a dichotomous criterion (success/failure). Although a polychoric correlation estimates the underlying Pearson correlation as if the predictor and criteria were continuous and bivariate normal variables, and the tetrachoric correlation if both x and y are assumed to dichotomized normal distributions, the phi coefficient is the Pearson applied to a matrix of 0's and 1s.

The phi coefficient was first reported by Yule (1912), but should not be confused with the `Yule` Q coefficient.

For a very useful discussion of various measures of association given a 2 x 2 table, and why one should probably prefer the `Yule` Q coefficient, see Warren (2008).

Given a two x two table of counts

a	b	a+b (R1)
c	d	c+d (R2)
a+c(C1)	b+d (C2)	a+b+c+d (N)

convert all counts to fractions of the total and then \ Phi = [a- (a+b)*(a+c)]/sqrt((a+b)(c+d)(a+c)(b+d)) =\ (a - R1 * C1)/sqrt(R1 * R2 * C1 * C2)

This is in contrast to the Yule coefficient, Q, where \ Q = (ad - bc)/(ad+bc) which is the same as \ [a- (a+b)*(a+c)]/(ad+bc)

Since the phi coefficient is just a Pearson correlation applied to dichotomous data, to find a matrix of phis from a data set involves just finding the correlations using cor or `lowerCor` or `corr.test`.

Value

phi coefficient of correlation

Author(s)

William Revelle with modifications by Leo Gurtler

References

Warrens, Matthijs (2008), On Association Coefficients for 2x2 Tables and Properties That Do Not Depend on the Marginal Distributions. Psychometrika, 73, 777-789.

Yule, G.U. (1912). On the methods of measuring the association between two attributes. Journal of the Royal Statistical Society, 75, 579-652.

See Also

phi2tetra ,Yule, Yule.inv Yule2phi, tetrachoric and polychoric

Examples

```
phi(c(30,20,20,30))
phi(c(40,10,10,40))
x <- matrix(c(40,5,20,20),ncol=2)
phi(x)
```

---

phi.demo *A simple demonstration of the Pearson, phi, and polychoric corelation*

Description

A not very interesting demo of what happens if bivariate continuous data are dichotomized. Bascially a demo of r, phi, and polychor.

Usage

```
phi.demo(n=1000,r=.6, cuts=c(-2,-1,0,1,2))
```

Arguments

n	number of cases to simulate
r	correlation between latent and observed
cuts	form dichotomized variables at the value of cuts

Details

A demonstration of the problem of different base rates on the phi correlation, and how these are partially solved by using the polychoric correlation. Not one of my more interesting demonstrations. See http://personality-project.org/r/simulating-personality.html and http://personality-project.org/r/r.datageneration.html for better demonstrations of data generation.

Value

a matrix of correlations and a graphic plot. The items above the diagonal are the tetrachoric correlations, below the diagonal are raw correlations.

Author(s)

William Revelle

References

http://personality-project.org/r/simulating-personality.html and http://personality-project.org/r/r.datageneration.html for better demonstrations of data generation.

See Also

VSS.simulate,item.sim

Examples

```
#demo <- phi.demo() #compare the phi (lower off diagonal and polychoric correlations
# (upper off diagonal)
#show the result from tetrachoric  which corrects for zero entries by default
#round(demo$tetrachoric$rho,2)
#show the result from phi2poly
#tetrachorics above the diagonal, phi below the diagonal
#round(demo$phis,2)
```

---

phi2tetra *Convert a phi coefficient to a tetrachoric correlation*

Description

Given a phi coefficient (a Pearson r calculated on two dichotomous variables), and the marginal frequencies (in percentages), what is the corresponding estimate of the tetrachoric correlation?

Given a two x two table of counts

a	b
c	d

The phi coefficient is (a - (a+b)*(a+c))/sqrt((a+b)(a+c)(b+d)(c+c)).

This function reproduces the cell entries for specified marginals and then calls the tetrachoric function. (Which was originally based upon John Fox's polychor function.) The phi2poly name will become deprecated in the future.

Usage

```
phi2tetra(ph,m,n=NULL,correct=TRUE)
phi2poly(ph,cp,cc,n=NULL,correct=TRUE) #deprecated
```

Arguments

ph	phi
m	a vector of the selection ratio and probability of criterion. In the case where ph is a matrix, m is a vector of the frequencies of the selected cases
correct	When finding tetrachoric correlations, should we correct for continuity for small marginals. See `tetrachoric` for a discussion.
n	If the marginals are given as frequencies, what was the total number of cases?
cp	probability of the predictor – the so called selection ratio
cc	probability of the criterion – the so called success rate.

Details

used to require the mvtnorm package but this has been replaced with mnormt

Value

a tetrachoric correlation

Author(s)

William Revelle

See Also

`tetrachoric`, `Yule2phi.matrix`, `phi2poly.matrix`

Examples

```
 phi2tetra(.3,c(.5,.5))
#phi2poly(.3,.3,.7)
```

`plot.psych` *Plotting functions for the psych package of class "psych"*

Description

Combines several plotting functions into one for objects of class "psych". This can be used to plot the results of `fa`, `irt.fa`, `VSS`, `ICLUST`, `omega`, `factor.pa`, or `principal`.

Usage

```
## S3 method for class 'psych'
plot(x,labels=NULL,...)
## S3 method for class 'irt'
plot(x,xlab,ylab,main,D,type=c("ICC","IIC","test"),cut=.3,labels=NULL,
 keys=NULL, xlim,ylim,y2lab,lncol="black",...)
## S3 method for class 'poly'
plot(x,D,xlab,ylab,xlim,ylim,main,type=c("ICC","IIC","test"),cut=.3,labels,
     keys=NULL,y2lab,lncol="black",...)
## S3 method for class 'residuals'
plot(x,main,type=c("qq","chi","hist","cor"),std, bad=4,
     numbers=TRUE, upper=FALSE,diag=FALSE,...)
```

Arguments

x	The object to plot
labels	Variable labels
xlab	Label for the x axis – defaults to Latent Trait
ylab	Label for the y axis
xlim	The limits for the x axis
ylim	Specify the limits for the y axis
main	Main title for graph
type	"ICC" plots items, "IIC" plots item information, "test" plots test information, defaults to IIC.,"qq" does a quantile plot,"chi" plots chi square distributions,"hist" shows the histogram,"cor" does a corPlot of the residuals.
D	The discrimination parameter
cut	Only plot item responses with discrimiantion greater than cut
keys	Used in plotting irt results from irt.fa.
y2lab	ylab for test reliability, defaults to "reliability"
bad	label the most 1.. bad items in residuals
numbers	if using the cor option in plot residuals, show the numeric values
upper	if using the cor option in plot residuals, show the upper off diagonal values
diag	if using the cor option in plot residuals, show the diagonal values
std	Standardize the resduals?
lncol	The color of the lines in the IRT plots. Defaults to all being black, but it is possible to specify lncol as a vector of colors to be used.
...	other calls to plot

Details

Passes the appropriate values to plot. For plotting the results of `irt.fa`, there are three options: type = "IIC" (default) will plot the item characteristic respone function. type = "IIC" will plot the item information function, and type= "test" will plot the test information function.

Note that plotting an irt result will call either plot.irt or plot.poly depending upon the type of data that were used in the original `irt.fa` call.

These are calls to the generic plot function that are intercepted for objects of type "psych". More precise plotting control is available in the separate plot functions. plot may be used for psych objects returned from `fa`, `irt.fa`, `ICLUST`, `omega`, as well as `principal`

A "jiggle" parameter is available in the fa.plot function (called from plot.psych when the type is a factor or cluster. If jiggle=TRUE, then the points are jittered slightly (controlled by amount) before plotting. This option is useful when plotting items with identical factor loadings (e.g., when comparing hypothetical models).

Objects from `irt.fa` are plotted according to "type" (Item informations, item characteristics, or test information). In addition, plots for selected items may be done if using the keys matrix. Plots of irt information return three invisible objects, a summary of information for each item at levels of the trait, the average area under the curve (the average information) for each item as well as where the item is most informative.

If plotting multiple factor solutions in plot.poly, then main can be a vector of names, one for each factor. The default is to give main + the factor number.

It is also possible to create irt like plots based upon just a scoring key and item difficulties, or from a factor analysis and item difficulties. These are not true IRT type analyses, in that the parameters are not estimated from the data, but are rather indications of item location and discrimination for arbitrary sets of items. To do this, find `irt.stats.like` and then plot the results.

`plot.residuals` allows the user to graphically examine the residuals of models formed by `fa`, `irt.fa`, `omega`, as well as `principal` and display them in a number of ways. "qq" will show quantiles of standardized or unstandardized residuals, "chi" will show quantiles of the squared standardized or unstandardized residuals plotted against the expected chi square values, "hist" will draw the histogram of the raw or standardized residuals, and "cor" will show a corPlot of the residual correlations.

Value

Graphic output for factor analysis, cluster analysis and item response analysis.

Note

More precise plotting control is available in the separate plot functions.

Author(s)

William Revelle

See Also

`VSS.plot` and `fa.plot`, `cluster.plot`, `fa`, `irt.fa`, `VSS`, `ICLUST`, `omega`, or `principal`

Examples

```
test.data <- Harman74.cor$cov
f4 <- fa(test.data,4)
plot(f4)
plot(resid(f4))
plot(resid(f4),main="Residuals from a 4 factor solution",qq=FALSE)
#not run
#data(bfi)
#e.irt <- irt.fa(bfi[11:15])  #just the extraversion items
#plot(e.irt)   #the information curves
#
ic <- iclust(test.data,3)   #shows hierarchical structure
plot(ic)                    #plots loadings
#
```

`polar` *Convert Cartesian factor loadings into polar coordinates*

Description

Factor and cluster analysis output typically presents item by factor correlations (loadings). Tables of factor loadings are frequently sorted by the size of loadings. This style of presentation tends to make it difficult to notice the pattern of loadings on other, secondary, dimensions. By converting to polar coordinates, it is easier to see the pattern of the secondary loadings.

Usage

```
polar(f, sort = TRUE)
```

Arguments

`f`	A matrix of loadings or the output from a factor or cluster analysis program
`sort`	sort=TRUE: sort items by the angle of the items on the first pair of factors.

Details

Although many uses of factor analysis/cluster analysis assume a simple structure where items have one and only one large loading, some domains such as personality or affect items have a more complex structure and some items have high loadings on two factors. (These items are said to have complexity 2, see `VSS`). By expressing the factor loadings in polar coordinates, this structure is more readily perceived.

For each pair of factors, item loadings are converted to an angle with the first factor, and a vector length corresponding to the amount of variance in the item shared with the two factors.

For a two dimensional structure, this will lead to a column of angles and a column of vector lengths. For n factors, this leads to n* (n-1)/2 columns of angles and an equivalent number of vector lengths.

Value

`polar`	A data frame of polar coordinates

Author(s)

William Revelle

References

Rafaeli, E. & Revelle, W. (2006). A premature consensus: Are happiness and sadness truly opposite affects? Motivation and Emotion. \

Hofstee, W. K. B., de Raad, B., & Goldberg, L. R. (1992). Integration of the big five and circumplex approaches to trait structure. Journal of Personality and Social Psychology, 63, 146-163.

See Also

`ICLUST`, `cluster.plot`, `circ.tests`, `fa`

Examples

```
circ.data <- circ.sim(24,500)
circ.fa <- fa(circ.data,2)
circ.polar <- round(polar(circ.fa),2)
circ.polar
#compare to the graphic
cluster.plot(circ.fa)
```

`polychor.matrix` *Phi or Yule coefficient matrix to polychoric coefficient matrix*

Description

A set of deprecated functions that have replaced by `Yule2tetra` and `Yule2phi`.

Some older correlation matrices were reported as matrices of Phi or of Yule correlations. That is, correlations were found from the two by two table of counts:

$$\begin{array}{cc} a & b \\ c & d \end{array}$$

Yule Q is (ad - bc)/(ad+bc).

With marginal frequencies of a+b, c+d, a+c, b+d.

Given a square matrix of such correlations, and the proportions for each variable that are in the a

+ b cells, it is possible to reconvert each correlation into a two by two table and then estimate the corresponding polychoric correlation (using John Fox's polychor function.

Usage

```
Yule2poly.matrix(x, v)  #deprectated
phi2poly.matrix(x, v)     #deprectated
Yule2phi.matrix(x, v)     #deprectated
```

Arguments

x	a matrix of phi or Yule coefficients
v	A vector of marginal frequencies

Details

These functions call Yule2poly, Yule2phi or phi2poly for each cell of the matrix. See those functions for more details. See phi.demo for an example.

Value

A matrix of correlations

Author(s)

William Revelle

Examples

```
#demo <- phi.demo()
#compare the phi (lower off diagonal and polychoric correlations (upper off diagonal)
#show the result from poly.mat
#round(demo$tetrachoric$rho,2)
#show the result from phi2poly
#tetrachorics above the diagonal, phi below the diagonal
#round(demo$phis,2)
```

---

predict.psych — *Prediction function for factor analysis or principal components*

Description

Finds predicted factor/component scores from a factor analysis or components analysis of data set A predicted to data set B. Predicted factor scores use the weights matrix used to find estimated factor scores, predicted components use the loadings matrix. Scores are either standardized with respect to the prediction sample or based upon the original data.

Usage

```
## S3 method for class 'psych'
predict(object, data,old.data,...)
```

Arguments

`object`	the result of a factor analysis or principal components analysis of data set A
`data`	Data set B, of the same number of variables as data set A.
`old.data`	if specified, the data set B will be standardized in terms of values from the old data. This is probably the preferred option.
`...`	More options to pass to predictions

Value

Predicted factor/components scores. The scores are based upon standardized items where the standardization is either that of the original data (old.data) or of the prediction set. This latter case can lead to confusion if just a small number of predicted scores are found.

Note

Thanks to Reinhold Hatzinger for the suggestion and request

Author(s)

William Revelle

See Also

`fa`, `principal`

Examples

```
set.seed(42)
x <- sim.item(12,500)
f2 <- fa(x[1:250,],2,scores="regression")  # a two factor solution
p2 <- principal(x[1:250,],2,scores=TRUE)  # a two component solution
round(cor(f2$scores,p2$scores),2)  #correlate the components and factors from the A set
#find the predicted scores (The B set)
pf2 <- predict(f2,x[251:500,],x[1:250,])
  #use the original data for standardization values
pp2 <- predict(p2,x[251:500,],x[1:250,])
 #standardized based upon the first set
round(cor(pf2,pp2),2)   #find the correlations in the B set
#test how well these predicted scores match the factor scores from the second set
fp2 <- fa(x[251:500,],2,scores=TRUE)
round(cor(fp2$scores,pf2),2)

pf2.n <- predict(f2,x[251:500,])  #Standardized based upon the new data set
round(cor(fp2$scores,pf2.n))
   #predict factors of set two from factors of set 1, factor order is arbitrary
```

```
#note that the signs of the factors in the second set are arbitrary
```

---

principal *Principal components analysis (PCA)*

Description

Does an eigen value decomposition and returns eigen values, loadings, and degree of fit for a specified number of components. Basically it is just doing a principal components analysis (PCA) for n principal components of either a correlation or covariance matrix. Can show the residual correlations as well. The quality of reduction in the squared correlations is reported by comparing residual correlations to original correlations. Unlike princomp, this returns a subset of just the best nfactors. The eigen vectors are rescaled by the sqrt of the eigen values to produce the component loadings more typical in factor analysis.

Usage

```
principal(r, nfactors = 1, residuals = FALSE,rotate="varimax",n.obs=NA, covar=FALSE,
 scores=TRUE,missing=FALSE,impute="median",oblique.scores=TRUE,method="regression",...)
```

Arguments

`r`	a correlation matrix. If a raw data matrix is used, the correlations will be found using pairwise deletions for missing values.
`nfactors`	Number of components to extract
`residuals`	FALSE, do not show residuals, TRUE, report residuals
`rotate`	"none", "varimax", "quatimax", "promax", "oblimin", "simplimax", and "cluster" are possible rotations/transformations of the solution. See `fa` for all rotations avaiable.
`n.obs`	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics.
`covar`	If false, find the correlation matrix from the raw data or convert to a correlation matrix if given a square matrix as input.
`scores`	If TRUE, find component scores
`missing`	if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean
`impute`	"median" or "mean" values are used to replace missing values
`oblique.scores`	If TRUE (default), then the component scores are based upon the structure matrix. If FALSE, upon the pattern matrix.
`method`	Which way of finding component scores should be used. The default is "regression"
`...`	other parameters to pass to functions such as factor.scores or the various rotation functions.

Details

Useful for those cases where the correlation matrix is improper (perhaps because of SAPA techniques).

There are a number of data reduction techniques including principal components analysis (PCA) and factor analysis (EFA). Both PC and FA attempt to approximate a given correlation or covariance matrix of rank n with matrix of lower rank (p). $_nR_n \approx_n F_{kk}F_n' + U^2$ where k is much less than n. For principal components, the item uniqueness is assumed to be zero and all elements of the correlation or covariance matrix are fitted. That is, $_nR_n \approx_n F_{kk}F_n'$ The primary empirical difference between a components versus a factor model is the treatment of the variances for each item. Philosophically, components are weighted composites of observed variables while in the factor model, variables are weighted composites of the factors.

For a n x n correlation matrix, the n principal components completely reproduce the correlation matrix. However, if just the first k principal components are extracted, this is the best k dimensional approximation of the matrix.

It is important to recognize that rotated principal components are not principal components (the axes associated with the eigen value decomposition) but are merely components. To point this out, unrotated principal components are labelled as PCi, while rotated PCs are now labeled as RCi (for rotated components) and obliquely transformed components as TCi (for transformed components). (Thanks to Ulrike Gromping for this suggestion.)

Rotations and transformations are either part of psych (Promax and cluster), of base R (varimax), or of GPArotation (simplimax, quartimax, oblimin, etc.).

Of the various rotation/transformation options, varimax, Varimax, quartimax, bentlerT, geominT, and bifactor do orthogonal rotations. Promax transforms obliquely with a target matix equal to the varimax solution. oblimin, quartimin, simplimax, bentlerQ, geominQ and biquartimin are oblique transformations. Most of these are just calls to the GPArotation package. The “cluster” option does a targeted rotation to a structure defined by the cluster representation of a varimax solution. With the optional "keys" parameter, the "target" option will rotate to a target supplied as a keys matrix. (See `target.rot`.)

The rotation matrix (rot.mat) is returned from all of these options. This is the inverse of the Th (theta?) object returned by the GPArotation package. The correlations of the factors may be found by $\Phi = \theta'\theta$

Some of the statistics reported are more appropriate for (maximum likelihood) factor analysis rather than principal components analysis, and are reported to allow comparisons with these other models.

Although for items, it is typical to find component scores by scoring the salient items (using, e.g., `score.items`) component scores are found by regression where the regression weights are $R^{-1}\lambda$ where λ is the matrix of component loadings. The regression approach is done to be parallel with the factor analysis function `fa`. The regression weights are found from the inverse of the correlation matrix times the component loadings. This has the result that the component scores are standard scores (mean=0, sd = 1) of the standardized input. A comparison to the scores from `princomp` shows this difference. princomp does not, by default, standardize the data matrix, nor are the components themselves standardized. The regression weights are found from the Structure matrix, not the Pattern matrix. If the scores are found with the covar option = TRUE, then the scores are not standardized but are just mean centered.

Jolliffe (2002) discusses why the interpretation of rotated components is complicated. The approach used here is consistent with the factor analytic tradition. The correlations of the items with the component scores closely matches (as it should) the component loadings (as reported in the structure matrix).

The output from the print.psych function displays the component loadings (from the pattern matrix), the h2 (communalities) the u2 (the uniquenesses), com (the complexity of the component loadings for that variable (see below). In the case of an orthogonal solution, h2 is merely the row sum of the squared component loadings. But for an oblique solution, it is the row sum of the (squared) orthogonal component loadings (remember, that rotations or transformations do not change the communality).

Value

`values`	Eigen Values of all components – useful for a scree plot
`rotation`	which rotation was requested?
`n.obs`	number of observations specified or found
`communality`	Communality estimates for each item. These are merely the sum of squared factor loadings for that item.
`complexity`	Hoffman's index of complexity for each item. This is just $\frac{(\Sigma a_i^2)^2}{\Sigma a_i^4}$ where a_i is the factor loading on the ith factor. From Hofmann (1978), MBR. See also Pettersson and Turkheimer (2010).
`loadings`	A standard loading matrix of class "loadings"
`fit`	Fit of the model to the correlation matrix
`fit.off`	how well are the off diagonal elements reproduced?
`residual`	Residual matrix – if requested
`dof`	Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters (number of items * number of factors - nf*(nf-1)/2. That is, dof = niI * (ni-1)/2 - ni * nf + nf*(nf-1)/2.
`objective`	value of the function that is minimized by maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is $f = log(trace((FF'+U2)^{-1}R) - log(\|(FF'+U2)^{-1}R\|) - n.items$. Because components do not minimize the off diagonal, this fit will be not as good as for factor analysis.
`STATISTIC`	If the number of observations is specified or found, this is a chi square based upon the objective function, f. Using the formula from `factanal`: $\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$
`PVAL`	If n.obs > 0, then what is the probability of observing a chisquare this large or larger?
`phi`	If oblique rotations (using oblimin from the GPArotation package) are requested, what is the interfactor correlation.
`scores`	If scores=TRUE, then estimates of the factor scores are reported
`weights`	The beta weights to find the principal components from the data
`R2`	The multiple R square between the factors and factor score estimates, if they were to be found. (From Grice, 2001) For components, these are of course 1.0.

valid — The correlations of the component score estimates with the components, if they were to be found and unit weights were used. (So called course coding).

rot.mat — The rotation matrix used to produce the rotated component loadings.

Note

By default, the accuracy of the varimax rotation function seems to be less than the Varimax function. This can be enhanced by specifying eps=1e-14 in the call to principal if using varimax rotation. Furthermore, note that Varimax by default does not apply the Kaiser normalization, but varimax does. Gottfried Helms compared these two rotations with those produced by SPSS and found identical values if using the appropriate options. (See the last two examples.)

Author(s)

William Revelle

References

Grice, James W. (2001), Computing and evaluating factor scores. Psychological Methods, 6, 430-450

Jolliffe, I. (2002) Principal Component Analysis (2nd ed). Springer.

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at http://personality-project.org/r/book/

See Also

VSS (to test for the number of components or factors to extract), VSS.scree and fa.parallel to show a scree plot and compare it with random resamplings of the data), factor2cluster (for course coding keys), fa (for factor analysis), factor.congruence (to compare solutions), predict.psych to find factor/component scores for a new data set based upon the weights from an original data set.

Examples

```
#Four principal components of the Harman 24 variable problem
#compare to a four factor principal axes solution using factor.congruence
pc <- principal(Harman74.cor$cov,4,rotate="varimax")
mr <- fa(Harman74.cor$cov,4,rotate="varimax")  #minres factor analysis
pa <- fa(Harman74.cor$cov,4,rotate="varimax",fm="pa")  # principal axis factor analysis
round(factor.congruence(list(pc,mr,pa)),2)

pc2 <- principal(Harman.5,2,rotate="varimax")
pc2
round(cor(Harman.5,pc2$scores),2)  #compare these correlations to the loadings
#now do it for unstandardized scores, and transform obliquely
pc2o <- principal(Harman.5,2,rotate="promax",covar=TRUE)
pc2o
round(cov(Harman.5,pc2o$scores),2)
pc2o$Structure    #this matches the covariances with the scores
biplot(pc2,main="Biplot of the Harman.5 socio-economic variables",labels=paste0(1:12))
```

```
#For comparison with SPSS  (contributed by Gottfried Helms)
pc2v <- principal(iris[1:4],2,rotate="varimax",normalize=FALSE,eps=1e-14)
print(pc2v,digits=7)
pc2V <- principal(iris[1:4],2,rotate="Varimax",eps=1e-7)
print(pc2V,digits=7)
```

`print.psych` *Print and summary functions for the psych class*

Description

Give limited output (print) or somewhat more detailed (summary) for most of the functions in psych.

Usage

```
## S3 method for class 'psych'
print(x,digits=2,all=FALSE,cut=NULL,sort=FALSE,short=TRUE,lower=TRUE,...)

## S3 method for class 'psych'
summary(object,digits=2,items=FALSE,...)
## S3 method for class 'psych'
anova(object,object2,...)
```

Arguments

`x`	Output from a psych function (e.g., factor.pa, omega,ICLUST, score.items, cluster.cor
`object`	Output from a psych function
`items`	items=TRUE (default) does not print the item whole correlations
`digits`	Number of digits to use in printing
`all`	if all=TRUE, then the object is declassed and all output from the function is printed
`cut`	Cluster loadings < cut will not be printed. For the factor analysis functions (fa and factor.pa etc.), cut defaults to 0, for ICLUST to .3, for omega to .2.
`sort`	Cluster loadings are in sorted order
`short`	Controls how much to print
`lower`	For square matrices, just print the lower half of the matrix
`object2`	Another object from fa or omega
`...`	More options to pass to summary and print

Details

Most of the psych functions produce too much output. print.psych and summary.psych use generic methods for printing just the highlights. To see what else is available, ask for the structure of the particular object: (str(theobject).

Alternatively, to get complete output, unclass(theobject) and then print it. This may be done by using the all=TRUE option.

As an added feature, if the promax function is applied to a factanal loadings matrix, the normal output just provides the rotation matrix. print.psych will provide the factor correlations. (Following a suggestion by John Fox and Uli Keller to the R-help list). The alternative is to just use the Promax function directly on the factanal object.

Value

Various psych functions produce copious output. This is a way to summarize the most important parts of the output of the score.items, cluster.scores, and ICLUST functions. See those (`score.items`, `cluster.cor`, `cluster.loadings`, or `ICLUST`) for details on what is produced.

Note

See `score.items`, `cluster.cor`, `cluster.loadings`, or `ICLUST`for details on what is printed.

Author(s)

William Revelle

Examples

```
data(bfi)
 keys.list <- list(agree=c(-1,2:5),conscientious=c(6:8,-9,-10),
 extraversion=c(-11,-12,13:15),neuroticism=c(16:20),openness = c(21,-22,23,24,-25))
 keys <- make.keys(25,keys.list,item.labels=colnames(bfi[1:25]))
 scores <- score.items(keys,bfi[1:25])
 scores
 summary(scores)
```

`Promax` *Perform bifactor, promax or targeted rotations and return the inter factor angles.*

Description

The bifactor rotation implements the rotation introduced by Jennrich and Bentler (2011) by calling GPForth in the GPArotation package. promax is an oblique rotation function introduced by Hendrickson and White (1964) and implemented in the promax function in the stats package. Unfortunately, promax does not report the inter factor correlations. Promax does. TargetQ does a target rotation with elements that can be missing (NA), or numeric (e.g., 0, 1). It uses the GPArotation package. target.rot does general target rotations to an arbitrary target matrix. The default target rotation is for an independent cluster solution. equamax facilitates the call to GPArotation to do an equamax rotation. Equamax, although available as a specific option within GPArotation is easier to call by name if using equamax. The varimin rotation suggested by Ertl (2013) is implemented by appropriate calls to GPArotation.

Usage

```
bifactor(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
biquartimin(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
TargetQ(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000,Target=NULL)
Promax(x, m = 4)
target.rot(x,keys=NULL)
varimin(L, Tmat = diag(ncol(L)), normalize = FALSE, eps = 1e-05, maxit = 1000)
vgQ.bimin(L)   #called by bifactor
vgQ.targetQ(L,Target=NULL)  #called by TargetQ
vgQ.varimin(L)  #called by varimin
equamax(L, Tmat=diag(ncol(L)), eps=1e-5, maxit=1000)
```

Arguments

`x`	A loadings matrix
`m`	the power to which to raise the varimax loadings (for Promax)
`keys`	An arbitrary target matrix, can be composed of any weights, but probably -1,0, 1 weights. If missing, the target is the independent cluster structure determined by assigning every item to it's highest loaded factor.
`L`	A loadings matrix
`Target`	A matrix of values (mainly 0s, some 1s, some NAs) to which the matrix is transformed.
`Tmat`	An initial rotation matrix
`normalize`	parameter passed to optimization routine (GPForth in the GPArotation package
`eps`	parameter passed to optimization routine (GPForth in the GPArotation package)
`maxit`	parameter passed to optimization routine (GPForth in the GPArotation package)

Details

The two most useful of these six functions is probably biquartimin which implements the oblique bifactor rotation introduced by Jennrich and Bentler (2011). The second is TargetQ which allows for missing NA values in the target. Next best is the orthogonal case, bifactor. None of these seem to be implemented in GPArotation (yet).

The difference between biquartimin and bifactor is just that the latter is the orthogonal case which is documented in Jennrich and Bentler (2011). It seems as if these two functions are sensitive to the starting values and random restarts (modifying T) might be called for.

bifactor output for the 24 cognitive variable of Holzinger matches that of Jennrich and Bentler as does output for the Chen et al. problem when fm="mle" is used and the Jennrich and Bentler solution is rescaled from covariances to correlations.

Promax is a very direct adaptation of the stats::promax function. The addition is that it will return the interfactor correlations as well as the loadings and rotation matrix.

varimin implements the varimin criterion proposed by Suitbert Ertl (2013). Rather than maximize the varimax criterion, it minimizes it. For a discussion of the benefits of this procedure, consult Ertel (2013).

In addition, these functions will take output from either the factanal, `fa` or earlier (`factor.pa`, `factor.minres` or `principal`) functions and select just the loadings matrix for analysis.

equamax is just a call to GPArotation's cFT function (for the Crawford Ferguson family of rotations.

TargetQ implements Michael Browne's algorithm and allows specification of NA values. The Target input is a list (see examples). It is interesting to note how powerful specifying what a factor isn't works in defining a factor. That is, by specifying the pattern of 0s and letting most other elements be NA, the factor structure is still clearly defined.

The target.rot function is an adaptation of a function of Michael Browne's to do rotations to arbitrary target matrices. Suggested by Pat Shrout.

The default for target.rot is to rotate to an independent cluster structure (every items is assigned to a group with its highest loading.)

target.rot will not handle targets that have linear dependencies (e.g., a pure bifactor model where there is a g loading and a group factor for all variables).

Value

`loadings`	Oblique factor loadings
`rotmat`	The rotation matrix applied to the original loadings to produce the promax soluion or the targeted matrix
`Phi`	The interfactor correlation matrix

Note

A direct adaptation of the stats:promax function following suggestions to the R-help list by Ulrich Keller and John Fox. Further modified to do targeted rotation similar to a function of Michael Browne.

varimin is a direct application of the GPArotation GPForth function modified to do varimin.

Author(s)

William Revelle

References

Ertel, S. (2013). Factor analysis: healing an ailing model. Universitatsverlag Gottingen.

Hendrickson, A. E. and White, P. O, 1964, British Journal of Statistical Psychology, 17, 65-70.

Jennrich, Robert and Bentler, Peter (2011) Exploratory Bi-Factor Analysis. Psychometrika, 1-13

See Also

`promax`, `fa`, or `principal` for examples of data analysis and `Holzinger` or `Bechtoldt` for examples of bifactor data. `factor.rotate` for 'hand rotation'.

Examples

```
jen <- sim.hierarchical()
f3 <- fa(jen,3,rotate="varimax")
f3   #not a very clean solution
Promax(f3)
target.rot(f3)
m3 <- fa(jen,nfactors=3)
Promax(m3)  #example of taking the output from factanal
#compare this rotation with the solution from a targeted rotation aimed for
#an independent cluster solution
target.rot(m3)
#now try a bifactor solution
fb <-fa(jen,3,rotate="bifactor")
fq <- fa(jen,3,rotate="biquartimin")
#Suitbert Ertel has suggested varimin
fm <-  fa(jen,3,rotate="varimin") #the Ertel varimin
fn <- fa(jen,3,rotate="none")  #just the unrotated factors
#compare them
factor.congruence(list(f3,fb,fq,fm,fn))
# compare an oblimin with a target rotation using the Browne algorithm
 #note that we are changing the factor #order (this is for demonstration only)
 Targ <- make.keys(9,list(f1=1:3,f2=7:9,f3=4:6))
 Targ <- scrub(Targ,isvalue=1)  #fix the 0s, allow the NAs to be estimated
 Targ <- list(Targ)  #input must be a list
#show the target
 Targ
 fa(Thurstone,3,rotate="TargetQ",Target=Targ)  #targeted rotation
#compare with oblimin
fa(Thurstone,3)
```

---

psych.misc *Miscellaneous helper functions for the psych package*

Description

This is a set of minor, if not trivial, helper functions. lowerCor finds the correlation of x variables and then prints them using lowerMat which is a trivial, but useful, function to round off and print the lower triangle of a matrix. reflect reflects the output of a factor analysis or principal components analysis so that one or more factors is reflected. (Requested by Alexander Weiss.) progressBar prints out ... as a calling routine (e.g., `tetrachoric`) works through a tedious calculation. shannon finds the Shannon index (H) of diversity or of information. test.all tests all the examples in a package. best.items sorts a factor matrix for absolute values and displays the expanded items names. fa.lookup returns sorted factor analysis output with item labels.

Usage

```
psych.misc()
```

```
lowerCor(x,digits=2,use="pairwise",method="pearson")
lowerMat(R, digits = 2)
tableF(x,y)
reflect(f,flip=NULL)
progressBar(value,max,label=NULL)
shannon(x,correct=FALSE,base=2)
test.all(pl,package="psych",dependencies
      = c("Depends", "Imports", "LinkingTo"),find=FALSE,skip=NULL)
```

Arguments

R	A rectangular matrix or data frame (probably a correlation matrix)
x	A data matrix or data frame or a vector depending upon the function.
y	A data matrix or data frame or a vector
f	The object returned from either a factor analysis (fa) or a principal components analysis (principal)
digits	round to digits
use	Should pairwise deletion be done, or one of the other options to cor
method	"pearson", "kendall", "spearman"
value	the current value of some looping variable
max	The maximum value the loop will achieve
label	what function is looping
flip	The factor or components to be reversed keyed (by factor number)
correct	Correct for the maximum possible information in this item
base	What is the base for the log function (default=2, e implies base = exp(1))
pl	The name of a package (or list of packages) to be activated and then have all the examples tested.
package	Find the dependencies for this package, e.g., psych
dependencies	Which type of dependency to examine?
find	Look up the dependencies, and then test all of their examples
skip	Do not test these dependencies

Details

`lowerCor` prints out the lower off diagonal matrix rounded to digits with column names abbreviated to digits + 3 characters, but also returns the full and unrounded matrix. By default, it uses pairwise deletion of variables. It in turn calls

`lowerMat` which does the pretty printing.

It is important to remember to not call `lowerCor` when all you need is `lowerMat`!

Value

tableF is fast alternative to the table function for creating two way tables of numeric variables. It does not have any of the elegant checks of the table function and thus is much faster. Used in the tetrachoric and polychoric functions to maximize speed.

The lower triangle of a matrix, rounded to digits with titles abbreviated to digits + 3 (lowerMat) or a series of dots (progressBar).

lowerCor prints the lower diagonal correlation matrix but returns (invisibly) the full correlation matrix found with the use and method parameters. The default values are for pairwise deletion of variables, and to print to 2 decimal places.

tableF (for tableFast) is a cut down version of table that does no error checking, nor returns pretty output, but is significantly faster than table. It will just work on two integer vectors. This is used in polychoric an tetrachoric for about a 50% speed improvement for large problems.

shannon finds Shannon's H index of information. Used for estimating the complexity or diversity of the distribution of responses in a vector or matrix.

$$H = -\sum p_i log(p_i)$$

test.all allows one to test all the examples in specified package. This allows us to make sure that those examples work when other packages (e.g., psych) are also loaded. This is used when developing revisions to the psych package to make sure the the other packages work. Some packages will not work and/or crash the system (e.g., DeducerPlugInScaling requires Java and even with Java, crashes when loaded, even if psych is not there!). Alternatively, if testing a long list of dependencies, you can skip the first part by specifying them by name.

See Also

corr.test to find correlations, count the pairwise occurrences, and to give significance tests for each correlation. r.test for a number of tests of correlations, including tests of the difference between correlations. lowerUpper will display the differences between two matrices.

Examples

```
lowerMat(Thurstone)
lb <- lowerCor(bfi[1:10])  #finds and prints the lower correlation matrix,
  # returns the square matrix.
#fiml <- corFiml(bfi[1:10])     #FIML correlations require lavaan package
#lowerMat(fiml)  #to get pretty output
f3 <- fa(Thurstone,3)
f3r <- reflect(f3,2)  #reflect the second factor
#find the complexity of the response patterns of the iqitems.
round(shannon(iqitems),2)
#test.all('BinNor')  #Does the BinNor package work when we are using other packages
bestItems(lb,3,cut=.1)
#to make this a latex table
#df2latex(bestItems(lb,2,cut=.2))
#
data(bfi.dictionary)
f2 <- fa(bfi[1:10],2)
fa.lookup(f2,bfi.dictionary)
```

`r.test` *Tests of significance for correlations*

Description

Tests the significance of a single correlation, the difference between two independent correlations, the difference between two dependent correlations sharing one variable (Williams's Test), or the difference between two dependent correlations with different variables (Steiger Tests).

Usage

```
r.test(n, r12, r34 = NULL, r23 = NULL, r13 = NULL, r14 = NULL, r24 = NULL,
          n2 = NULL,pooled=TRUE, twotailed = TRUE)
```

Arguments

`n`	Sample size of first group
`r12`	Correlation to be tested
`r34`	Test if this correlation is different from r12, if r23 is specified, but r13 is not, then r34 becomes r13
`r23`	if ra = r(12) and rb = r(13) then test for differences of dependent correlations given r23
`r13`	implies ra =r(12) and rb =r(34) test for difference of dependent correlations
`r14`	implies ra =r(12) and rb =r(34)
`r24`	ra =r(12) and rb =r(34)
`n2`	n2 is specified in the case of two independent correlations. n2 defaults to n if if not specified
`pooled`	use pooled estimates of correlations
`twotailed`	should a twotailed or one tailed test be used

Details

Depending upon the input, one of four different tests of correlations is done. 1) For a sample size n, find the t value for a single correlation.

2) For sample sizes of n and n2 (n2 = n if not specified) find the z of the difference between the z transformed correlations divided by the standard error of the difference of two z scores.

3) For sample size n, and correlations r12, r13 and r23 test for the difference of two dependent correlations (r12 vs r13).

4) For sample size n, test for the difference between two dependent correlations involving different variables.

For clarity, correlations may be specified by value. If specified by location and if doing the test of dependent correlations, if three correlations are specified, they are assumed to be in the order r12, r13, r23. Consider the example the example from Steiger: where Masculinity at time 1 (M1) correlates with Verbal Ability .5 (r12), femininity at time 1 (F1) correlates with Verbal ability r13 =.4, and M1 correlates with F1 (r23= .1). Then, given the correlations: r12 = .4, r13 = .5, and r23 = .1, t = -.89 for n =103, i.e., r.test(n=103, r12=.4, r13=.5,r23=.1)

Value

`test`	Label of test done
`z`	z value for tests 2 or 4
`t`	t value for tests 1 and 3
`p`	probability value of z or t

Note

Steiger specifically rejects using the Hotelling T test to test the difference between correlated correlations. Instead, he recommends Williams' test. (See also Dunn and Clark, 1971). These tests follow Steiger's advice.

Author(s)

William Revelle

References

Olkin, I. and Finn, J. D. (1995). Correlations redux. Psychological Bulletin, 118(1):155-164.

Steiger, J.H. (1980), Tests for comparing elements of a correlation matrix, Psychological Bulletin, 87, 245-251.

Williams, E.J. (1959) Regression analysis. Wiley, New York, 1959.

See Also

See also `corr.test` which tests all the elements of a correlation matrix, and `cortest.mat` to compare two matrices of correlations. r.test extends the tests in `paired.r`,`r.con`

Examples

```
n <- 30
r <- seq(0,.9,.1)
rc <- matrix(r.con(r,n),ncol=2)
test <- r.test(n,r)
r.rc <- data.frame(r=r,z=fisherz(r),lower=rc[,1],upper=rc[,2],t=test$t,p=test$p)
round(r.rc,2)

r.test(50,r)
r.test(30,.4,.6)       #test the difference between two independent correlations
r.test(103,.4,.5,.1)   #Steiger case A of dependent correlations
r.test(n=103, r12=.4, r13=.5,r23=.1)
#for complicated tests, it is probably better to specify correlations by name
r.test(n=103,r12=.5,r34=.6,r13=.7,r23=.5,r14=.5,r24=.8)   #steiger Case B
```

`rangeCorrection` *Correct correlations for restriction of range. (Thorndike Case 2)*

Description

In applied settings, it is typical to find a correlation between a predictor and some criterion. Unfortunately, if the predictor is used to choose the subjects, the range of the predictor is seriously reduced. This restricts the observed correlation to be less than would be observed in the full range of the predictor. A correction for this problem is well known as Thorndike Case 2:

Let R the unrestricted correlaton, r the restricted correlation, S the unrestricted standard deviation, s the restricted standard deviation, then

R = (rS/s)/ sqrt(1-r^2 + r^2(S^2/s^2)).

Several other cases of restriction were also considered by Thorndike and are implemented in `rangeCorrection`.

Usage

```
rangeCorrection(r,sdu,sdr,sdxu=NULL,sdxr=NULL,case=2)
```

Arguments

`r`	The observed correlation
`sdu`	The unrestricted standard deviation)
`sdr`	The restricted standard deviation
`sdxu`	Unrestricted standard deviation for case 4
`sdxr`	Restricted standard deviation for case 4
`case`	Which of the four Thurstone/Stauffer cases to use

Details

When participants in a study are selected on one variable, that will reduce the variance of that variable and the resulting correlation. Thorndike (1949) considered four cases of range restriction. Others have continued this discussion but have changed the case numbers.

Can be used to find correlations in a restricted sample as well as the unrestricted sample. Not the same as the correction to reliability for restriction of range.

Value

The corrected correlation.

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at http://personality-project.org/r/book/

Stauffer, Joseph and Mendoza, Jorge. (2001) The proper sequence for correcting correlation coefficients for range restriction and unreliability. Psychometrika, 66, 63-68.

See Also

cRRr in the psychometric package.

Examples

```
rangeCorrection(.33,100.32,48.19) #example from Revelle (in prep) Chapter 4.
```

---

read.clipboard *shortcut for reading from the clipboard*

Description

Input from the clipboard is easy but a bit obscure, particularly for Mac users. This is just an easier way to do so. Data may be copied to the clipboard from Exel spreadsheets, csv files, or fixed width formatted files and then into a data.frame. Data may also be read from lower (or upper) triangular matrices and filled out to square matrices.

Usage

```
read.clipboard(header = TRUE, ...)   #assumes headers and tab or space delimited
read.clipboard.csv(header=TRUE,sep=',',...)  #assumes headers and comma delimited
read.clipboard.tab(header=TRUE,sep='\t',...)  #assumes headers and tab delimited
 #read in a matrix given the lower off diagonal
 read.clipboard.lower(diag=TRUE,names=FALSE,...)
 read.clipboard.upper(diag=TRUE,names=FALSE,...)

#read in data using a fixed format width (see read.fwf for instructions)
read.clipboard.fwf(header=FALSE,widths=rep(1,10),...)
read.https(filename,header=TRUE)
```

Arguments

header	Does the first row have variable labels
sep	What is the designated separater between data fields?
diag	for upper or lower triangular matrices, is the diagonal specified or not
names	for read.clipboard.lower or upper, are colnames in the the first column
widths	how wide are the columns in fixed width input. The default is to read 10 columns of size 1.

filename name or address of remote https file to read

... Other parameters to pass to read

Details

A typical session of R might involve data stored in text files, generated online, etc. Although it is easy to just read from a file (particularly if using file.choose(), copying from the file to the clipboard and then reading from the clipboard is also very convenient (and somewhat more intuitive to the naive user). This is particularly convenient when copying from a text book or article and just moving a section of text into R.)

Based upon a suggestion by Ken Knoblauch to the R-help listserve.

If the input file that was copied into the clipboard was an Excel file with blanks for missing data, then read.clipboard.tab() will correctly replace the blanks with NAs. Similarly for a csv file with blank entries, read.clipboard.csv will replace empty fields with NA.

read.clipboard.lower and read.clipboard.upper are adapted from John Fox's read.moments function in the sem package. They will read a lower (or upper) triangular matrix from the clipboard and return a full, symmetric matrix for use by factanal, factor.pa , ICLUST , etc. If the diagonal is false, it will be replaced by 1.0s. These two function were added to allow easy reading of examples from various texts and manuscripts with just triangular output.

Many articles will report lower triangular matrices with variable labels in the first column. read.clipboard.lower will handle this case. Names must be in the first column if names=TRUE is specified.

Other articles will report upper triangular matrices with variable labels in the first row. read.clipboard.upper will handle this. Note that labels in the first column will not work for read.clipboard.upper. The names, if present, must be in the first row.

read.clipboard.fwf will read fixed format files from the clipboard. It includes a patch to read.fwf which will not read from the clipboard or from remote file. See read.fwf for documentation of how to specify the widths.

Value

the contents of the clipboard.

Author(s)

William Revelle

Examples

```
#my.data <- read.clipboad()
#my.data <- read.clipboard.csv()
#my.data <- read.clipboad(header=FALSE)
#my.matrix <- read.clipboard.lower()
```

rescale *Function to convert scores to "conventional " metrics*

Description

Psychologists frequently report data in terms of transformed scales such as "IQ" (mean=100, sd=15, "SAT/GRE" (mean=500, sd=100), "ACT" (mean=18, sd=6), "T-scores" (mean=50, sd=10), or "Stanines" (mean=5, sd=2). The `rescale` function converts the data to standard scores and then rescales to the specified mean(s) and standard deviation(s).

Usage

```
rescale(x, mean = 100, sd = 15,df=TRUE)
```

Arguments

`x`	A matrix or data frame
`mean`	Desired mean of the rescaled scores- may be a vector
`sd`	Desired standard deviation of the rescaled scores
`df`	if TRUE, returns a data frame, otherwise a matrix

Value

A data.frame (default) or matrix of rescaled scores.

Author(s)

William Revelle

See Also

See Also `scale`

Examples

```
T <- rescale(attitude,50,10) #all put on same scale
describe(T)
T1 <- rescale(attitude,seq(0,300,50),seq(10,70,10)) #different means and sigmas
describe(T1)
```

residuals.psych *Extract residuals from various psych objects*

Description

Residuals in the various psych functions are extracted and then may be "pretty" printed.

Usage

```
## S3 method for class 'psych'
residuals(object,...)
## S3 method for class 'psych'
resid(object,...)
```

Arguments

object	The object returned by a psych function.
...	Other parameters to be passed to residual (ignored but required by the generic function)

Details

Currently implemented for `fa`, `principal`, `omega`, `irt.fa`, and `fa.extension`.

Value

residuals: a matrix of residual estimates

Author(s)

William Revelle

Examples

```
f3 <- fa(Thurstone,3)
residuals(f3)
```

reverse.code *Reverse the coding of selected items prior to scale analysis*

Description

Some IRT functions require all items to be coded in the same direction. Some data sets have items that need to be reverse coded (e.g., 6 -> 1, 1 -> 6). reverse.code will flip items based upon a keys vector of 1s and -1s. Reversed items are subtracted from the item max + item min. These may be specified or may be calculated.

Usage

```
reverse.code(keys, items, mini = NULL, maxi = NULL)
```

Arguments

keys	A vector of 1s and -1s. -1 implies reverse the item
items	A data set of items
mini	if NULL, the empirical minimum for each item. Otherwise, a vector of minima
maxi	f NULL, the empirical maximum for each item. Otherwise, a vector of maxima

Details

Not a very complicated function, but useful in the case that items need to be reversed prior to using IRT functions from the ltm or eRM packages. Most psych functions do not require reversing prior to analysis, but will do so within the function.

Value

The corrected items.

Examples

```
original <- matrix(sample(6,50,replace=TRUE),10,5)
keys <- c(1,1,-1,-1,1)  #reverse the 3rd and 4th items
new <- reverse.code(keys,original,mini=rep(1,5),maxi=rep(6,5))
original[1:3,]
new[1:3,]
```

`sat.act` *3 Measures of ability: SATV, SATQ, ACT*

Description

Self reported scores on the SAT Verbal, SAT Quantitative and ACT were collected as part of the Synthetic Aperture Personality Assessment (SAPA) web based personality assessment project. Age, gender, and education are also reported. The data from 700 subjects are included here as a demonstration set for correlation and analysis.

Usage

```
data(sat.act)
```

Format

A data frame with 700 observations on the following 6 variables.

`gender` males = 1, females = 2

`education` self reported education 1 = high school ... 5 = graduate work

`age` age

`ACT` ACT composite scores may range from 1 - 36. National norms have a mean of 20.

`SATV` SAT Verbal scores may range from 200 - 800.

`SATQ` SAT Quantitative scores may range from 200 - 800

Details

hese items were collected as part of the SAPA project (http://sapa-project.org)to develop online measures of ability (Revelle, Wilt and Rosenthal, 2009). The score means are higher than national norms suggesting both self selection for people taking on line personality and ability tests and a self reporting bias in scores.

See also the iq.items data set.

Source

http://personality-project.org

References

Revelle, William, Wilt, Joshua, and Rosenthal, Allen (2009) Personality and Cognition: The Personality-Cognition Link. In Gruszka, Alexandra and Matthews, Gerald and Szymura, Blazej (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

Examples

```
data(sat.act)
describe(sat.act)
pairs.panels(sat.act)
```

scaling.fits — *Test the adequacy of simple choice, logistic, or Thurstonian scaling.*

Description

Given a matrix of choices and a vector of scale values, how well do the scale values capture the choices? That is, what is size of the squared residuals given the model versus the size of the squared choice values?

Usage

```
scaling.fits(model, data, test = "logit", digits = 2, rowwise = TRUE)
```

Arguments

model	A vector of scale values
data	A matrix or dataframe of choice frequencies
test	"choice", "logistic", "normal"
digits	Precision of answer
rowwise	Are the choices ordered by column over row (TRUE) or row over column False)

Details

How well does a model fit the data is the classic problem of all of statistics. One fit statistic for scaling is the just the size of the residual matrix compared to the original estimates.

Value

GF	Goodness of fit of the model
original	Sum of squares for original data
resid	Sum of squares for residuals given the data and the model
residual	Residual matrix

Note

Mainly for demonstration purposes for a course on psychometrics

Author(s)

William Revelle

References

Revelle, W. (in preparation) Introduction to psychometric theory with applications in R, Springer. http://personality-project.org/r/book

See Also

thurstone, vegetables

scatter.hist *Draw a scatter plot with associated X and Y histograms, densitie and correlation*

Description

Draw a X Y scatter plot with associated X and Y histograms with estimated densities. Partly a demonstration of the use of layout. Also includes lowess smooth or linear model slope, as well as correlation. Adapted from addicted to R example 78

Usage

```
scatter.hist(x,y=NULL,smooth=TRUE,ab=FALSE,correl=TRUE,density=TRUE,ellipse=TRUE,
  digits=2, method,cex.cor=1,title="Scatter plot + histograms",xlab=NULL,ylab=NULL,...)
```

Arguments

x	The X vector, or the first column of a data.frame or matrix.
y	The Y vector, of if X is a data.frame or matrix, the second column of X
smooth	if TRUE, then loess smooth it
ab	if TRUE, then show the best fitting linear fit
correl	TRUE: Show the correlation
density	TRUE: Show the estimated densities
ellipse	TRUE: draw 1 and 2 sigma ellipses and smooth
digits	How many digits to use if showing the correlation
method	Which method to use for correlation ("pearson","spearman","kendall") defaults to "pearson"
cex.cor	Adjustment for the size of the correlation
xlab	Label for the x axis
ylab	Label for the y axis
title	An optional title
...	Other parameters for graphics

Details

Just a straightforward application of layout and barplot, with some tricks taken from pairs.panels. The various options allow for correlation ellipses (1 and 2 sigma from the mean), lowess smooths, linear fits, density curves on the histograms, and the value of the correlation. ellipse = TRUE implies smooth = TRUE)

Note

Adapted from Addicted to R example 78

Author(s)

William Revelle

See Also

pairs.panels for multiple plots, multi.hist for multiple histograms.

Examples

```
data(sat.act)
with(sat.act,scatter.hist(SATV,SATQ))
#or for something a bit more splashy
scatter.hist(sat.act[5:6],pch=(19+sat.act$gender),col=c("blue","red")[sat.act$gender])
```

Schmid *12 variables created by Schmid and Leiman to show the Schmid-Leiman Transformation*

Description

John Schmid and John M. Leiman (1957) discuss how to transform a hierarchical factor structure to a bifactor structure. Schmid contains the example 12 x 12 correlation matrix. schmid.leiman is a 12 x 12 correlation matrix with communalities on the diagonal. This can be used to show the effect of correcting for attenuation. Two additional data sets are taken from Chen et al. (2006).

Usage

```
data(Schmid)
```

Details

Two artificial correlation matrices from Schmid and Leiman (1957). One real and one artificial covariance matrices from Chen et al. (2006).

- Schmid: a 12 x 12 artificial correlation matrix created to show the Schmid-Leiman transformation.
- schmid.leiman: A 12 x 12 matrix with communalities on the diagonal. Treating this as a covariance matrix shows the 6 x 6 factor solution
- Chen: An 18 x 18 covariance matrix of health related quality of life items from Chen et al. (2006). Number of observations = 403. The first item is a measure of the quality of life. The remaining 17 items form four subfactors: The items are (a) Cognition subscale: “Have difficulty reasoning and solving problems?" “React slowly to things that were said or done?"; “Become confused and start several actions at a time?" “Forget where you put things

or appointments?"; “Have difficulty concentrating?" (b) Vitality subscale: “Feel tired?" “Have enough energy to do the things you want?" (R) “Feel worn out?" ; “Feel full of pep?" (R). (c) Mental health subscale: “Feel calm and peaceful?"(R) “Feel downhearted and blue?"; “Feel very happy"(R) ; “Feel very nervous?" ; “Feel so down in the dumps nothing could cheer you up? (d) Disease worry subscale: “Were you afraid because of your health?"; “Were you frustrated about your health?"; “Was your health a worry in your life?" .

- West: A 16 x 16 artificial covariance matrix from Chen et al. (2006).

Source

John Schmid Jr. and John. M. Leiman (1957), The development of hierarchical factor solutions.Psychometrika, 22, 83-90.

F.F. Chen, S.G. West, and K.H. Sousa.(2006) A comparison of bifactor and second-order models of quality of life. Multivariate Behavioral Research, 41(2):189-225, 2006.

References

Y.-F. Yung, D.Thissen, and L.D. McLeod. (1999) On the relationship between the higher-order factor model and the hierarchical factor model. Psychometrika, 64(2):113-128, 1999.

Examples

```
data(Schmid)
cor.plot(Schmid,TRUE)
print(fa(Schmid,6,rotate="oblimin"),cut=0)  #shows an oblique solution
round(cov2cor(schmid.leiman),2)
cor.plot(cov2cor(West),TRUE)
```

`schmid` *Apply the Schmid Leiman transformation to a correlation matrix*

Description

One way to find omega is to do a factor analysis of the original data set, rotate the factors obliquely, do a Schmid Leiman transformation, and then find omega. Here is the code for Schmid Leiman. The S-L transform takes a factor or PC solution, transforms it to an oblique solution, factors the oblique solution to find a higher order (g) factor, and then residualizes g out of the the group factors.

Usage

```
schmid(model, nfactors = 3, fm = "minres",digits=2,rotate="oblimin",
            n.obs=NA,option="equal",Phi=NULL,covar=FALSE,...)
```

Arguments

`model`	A correlation matrix
`nfactors`	Number of factors to extract
`fm`	the default is to do minres. fm="pa" for principal axes, fm="pc" for principal components, fm = "minres" for minimum residual (OLS), pc="ml" for maximum likelihood
`digits`	if digits not equal NULL, rounds to digits
`rotate`	The default, oblimin, produces somewhat more correlated factors than the alternative, simplimax. The third option is the promax criterion
`n.obs`	Number of observations, used to find fit statistics if specified. Will be calculated if input is raw data
`option`	When asking for just two group factors, option can be for "equal", "first" or "second"
`Phi`	If Phi is specified, then the analysis is done on a pattern matrix with the associated factor intercorrelation (Phi) matrix. This allows for reanalysess of published results
`covar`	Defaults to FALSE and finds correlations. If set to TRUE, then do the calculations on the unstandardized variables.
`...`	Allows additional parameters to be passed to the factoring routines

Details

Schmid Leiman orthogonalizations are typical in the ability domain, but are not seen as often in the non-cognitive personality domain. S-L is one way of finding the loadings of items on the general factor for estimating omega.

A typical example would be in the study of anxiety and depression. A general neuroticism factor (g) accounts for much of the variance, but smaller group factors of tense anxiety, panic disorder, depression, etc. also need to be considerd.

An alternative model is to consider hierarchical cluster analysis techniques such as `ICLUST`.

Requires the GPArotation package.

Although 3 factors are the minimum number necessary to define the solution uniquely, it is occasionally useful to allow for a two factor solution. There are three possible options for this condition: setting the general factor loadings between the two lower order factors to be "equal" which will be the sqrt(oblique correlations between the factors) or to "first" or "second" in which case the general factor is equated with either the first or second group factor. A message is issued suggesting that the model is not really well defined.

A diagnostic tool for testing the appropriateness of a hierarchical model is p2 which is the percent of the common variance for each variable that is general factor variance. In general, p2 should not have much variance.

Value

`sl`	loadings on g + nfactors group factors, communalities, uniqueness, percent of g2 of h2

orthog	original orthogonal factor loadings
oblique	oblique factor loadings
phi	correlations among the transformed factors
gload	loadings of the lower order factors on g
...	

Author(s)

William Revelle

References

http://personality-project.org/r/r.omega.html gives an example taken from Jensen and Weng, 1994 of a S-L transformation.

See Also

omega, omega.graph, fa.graph, ICLUST,VSS

Examples

```
jen <- sim.hierarchical()  #create a hierarchical demo
if(!require(GPArotation)) {
   message("I am sorry, you must have GPArotation installed to use schmid.")} else {
   p.jen <- schmid(jen,digits=2)   #use the oblimin rotation
p.jen <- schmid(jen,rotate="promax") #use the promax rotation
}
```

---

score.alpha — *Score scales and find Cronbach's alpha as well as associated statistics*

Description

Given a matrix or data.frame of k keys for m items (-1, 0, 1), and a matrix or data.frame of items scores for m items and n people, find the sum scores or average scores for each person and each scale. In addition, report Cronbach's alpha, the average r, the scale intercorrelations, and the item by scale correlations. (Superseded by score.items).

Usage

```
score.alpha(keys, items, labels = NULL, totals=TRUE,digits = 2) #deprecated
```

Arguments

keys	A matrix or dataframe of -1, 0, or 1 weights for each item on each scale
items	Data frame or matrix of raw item scores
labels	column names for the resulting scales
totals	Find sum scores (default) or average score
digits	Number of digits for answer (default =2)

Details

This function has been replaced with `score.items` (for multiple scales) and `alpha` for single scales.

The process of finding sum or average scores for a set of scales given a larger set of items is a typical problem in psychometric research. Although the structure of scales can be determined from the item intercorrelations, to find scale means, variances, and do further analyses, it is typical to find the sum or the average scale score.

Various estimates of scale reliability include "Cronbach's alpha", and the average interitem correlation. For k = number of items in a scale, and av.r = average correlation between items in the scale, alpha = k * av.r/(1+ (k-1)*av.r). Thus, alpha is an increasing function of test length as well as the test homeogeneity.

Alpha is a poor estimate of the general factor saturation of a test (see Zinbarg et al., 2005) for it can seriously overestimate the size of a general factor, and a better but not perfect estimate of total test reliability because it underestimates total reliability. None the less, it is a useful statistic to report.

Value

`scores`	Sum or average scores for each subject on the k scales
`alpha`	Cronbach's coefficient alpha. A simple (but non-optimal) measure of the internal consistency of a test. See also beta and omega.
`av.r`	The average correlation within a scale, also known as alpha 1 is a useful index of the internal consistency of a domain.
`n.items`	Number of items on each scale
`cor`	The intercorrelation of all the scales
`item.cor`	The correlation of each item with each scale. Because this is not corrected for item overlap, it will overestimate the amount that an item correlates with the other items in a scale.

Author(s)

William Revelle

References

An introduction to psychometric theory with applications in R (in preparation). http://personality-project.org/r/book

See Also

`score.items`, `alpha`, `correct.cor`, `cluster.loadings`, `omega`

Examples

```
y <- attitude     #from the datasets package
keys <- matrix(c(rep(1,7),rep(1,4),rep(0,7),rep(-1,3)),ncol=3)
labels <- c("first","second","third")
x <- score.alpha(keys,y,labels) #deprecated
```

score.irt *Find Item Response Theory (IRT) based scores for dichotomous or polytomous items*

Description

irt.fa finds Item Response Theory (IRT) parameters through factor analysis of the tetrachoric or polychoric correlations of dichtomous or polytomous items. score.irt uses these parameter estimates of discrimination and location to find IRT based scores for the responses. As many factors as found for the correlation matrix will be scored.

Usage

```
score.irt(stats=NULL, items, keys=NULL,cut = 0.3,bounds=c(-5,5),mod="logistic")
 #the higher order call just calls one of the next two
  #for dichotomous items
score.irt.2(stats, items,keys=NULL,cut = 0.3,bounds=c(-5,5),mod="logistic")
  #for polytomous items
score.irt.poly(stats, items, keys=NULL, cut = 0.3,bounds=c(-5,5))
    #to create irt like statistics for plotting
irt.stats.like(items,stats,keys=NULL,cut=.3)

irt.tau(x)
```

Arguments

stats	Output from irt.fa is used for parameter estimates of location and discrimination. Stats may also be the output from a normal factor analysis (fa)
items	The raw data, may be either dichotomous or polytomous.
keys	A keys matrix of which items should be scored for each factor
cut	Only items with discrimination values > cut will be used for scoring.
x	The raw data to be used to find the tau parameter in irt.tau
bounds	The lower and upper estimates for the fitting function
mod	Should a logistic or normal model be used in estimating the scores?

Details

Although there are more elegant ways of finding subject scores given a set of item locations (difficulties) and discriminations, simply finding that value of theta θ that best fits the equation $P(x|\theta) = 1/(1+exp(\beta(\delta-\theta))$ for a score vector X, and location δ and discrimination β provides more information than just total scores. With complete data, total scores and irt estimates are almost perfectly correlated. However, the irt estimates provide much more information in the case of missing data.

The bounds parameter sets the lower and upper limits to the estimate. This is relevant for the case of a subject who gives just the lowest score on every item, or just the top score on every item. In this case, the scores are estimated by finding the probability of missing every item taken, converting this to a quantile score based upon the normal distribution, and then assigning a z value equivalent to 1/2 of that quantile. Similarly, if a person gets all the items they take correct, their score is defined as the quantile of the z equivalent to the probability of getting all of the items correct, and then moving up the distribution half way. If these estimates exceed either the upper or lower bounds, they are adjusted to those boundaries.

There are several more elegant packages in R that provide Full Information Maximum Likeliood IRT based estimates. The estimates from score.irt do not do so. However, the score.irt seems to do a good job of recovering the basic structure.

The keys matrix is a matrix of 1s, 0s, and -1s reflecting whether an item should be scored or not scored for a particular factor. See `score.items` or `make.keys` for details. The default case is to score all items with absolute discriminations > cut.

If one wants to score scales taking advantage of differences in item location but not do a full irt analysis, then find the item difficulties from the raw data using `irt.tau` or combine this information with a scoring keys matrix (see `score.items` and codemake.keys and create quasi-irt statistics using `irt.stats.like`.

There are conventionally two different metrics and models that are used. The logistic metric and model and the normal metric and model. These are chosen using the mod parameter.

Value

`scores`	A data frame of theta estimates, total scores based upon raw sums, and estimates of fit.

Note

Still under development. Suggestions for improvement are most appreciated.

score.irt is just a wrapper to score.irt.poly and score.irt.2

Author(s)

William Revelle

References

Kamata, Akihito and Bauer, Daniel J. (2008) A Note on the Relation Between Factor Analytic and Item Response Theory Models Structural Equation Modeling, 15 (1) 136-153.

McDonald, Roderick P. (1999) Test theory: A unified treatment. L. Erlbaum Associates.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at http://personality-project.org/r/book/

See Also

`irt.fa` for finding the parameters. For more conventional scoring algorithms see `score.items`. `irt.responses` will plot the empirical response patterns for the alternative response choices for multiple choice items. For more conventional IRT estimations, see the ltm package.

Examples

```
if(FALSE) {  #not run in the interest of time, but worth doing
d9 <- sim.irt(9,1000,-2.5,2.5,mod="normal") #dichotomous items
test <- irt.fa(d9$items)
scores <- score.irt(test,d9$items)
scores.df <- data.frame(scores,true=d9$theta) #combine the estimates with the true thetas.
pairs.panels(scores.df,pch=".",
main="Comparing IRT and classical with complete data")
#with all the data, why bother ?

#now delete some of the data
d9$items[1:333,1:3] <- NA
d9$items[334:666,4:6] <- NA
d9$items[667:1000,7:9] <- NA
scores <- score.irt(test,d9$items)
scores.df <- data.frame(scores,true=d9$theta) #combine the estimates with the true thetas.
pairs.panels(scores.df, pch=".",
main="Comparing IRT and classical with random missing data")
 #with missing data, the theta estimates are noticably better.
}

v9 <- sim.irt(9,1000,-2.,2.,mod="normal") #dichotomous items
items <- v9$items
test <- irt.fa(items)
total <- rowSums(items)
ord <- order(total)
items <- items[ord,]


#now delete some of the data - note that they are ordered by score
items[1:333,5:9] <- NA
items[334:666,3:7] <- NA
items[667:1000,1:4] <- NA
scores <- score.irt(test,items)
unitweighted <- score.irt(items=items,keys=rep(1,9)) #each item has a discrimination of 1
#combine the estimates with the true thetas.
scores.df <- data.frame(v9$theta[ord],scores,unitweighted)

colnames(scores.df) <- c("True theta","irt theta","total","fit","rasch","total","fit")
pairs.panels(scores.df,pch=".",main="Comparing IRT and classical with missing data")
 #with missing data, the theta estimates are noticably better estimates
 #of the generating theta than calling them all equal
```

---

score.multiple.choice *Score multiple choice items and provide basic test statistics*

Description

Ability tests are typically multiple choice with one right answer. score.multiple.choice takes a scoring key and a data matrix (or data.frame) and finds total or average number right for each participant. Basic test statistics (alpha, average r, item means, item-whole correlations) are also reported.

Usage

```
score.multiple.choice(key, data, score = TRUE, totals = FALSE, ilabels = NULL,
      missing = TRUE, impute = "median", digits = 2,short=TRUE,skew=FALSE)
```

Arguments

key	A vector of the correct item alternatives
data	a matrix or data frame of items to be scored.
score	score=FALSE, just convert to right (1) or wrong (0). score=TRUE, find the totals or average scores and do item analysis
totals	total=FALSE: find the average number correct total=TRUE: find the total number correct
ilabels	item labels
missing	missing=TRUE: missing values are replaced with means or medians missing=FALSE missing values are not scored
impute	impute="median", replace missing items with the median score impute="mean": replace missing values with the item mean
digits	How many digits of output
short	short=TRUE, just report the item statistics, short=FALSE, report item statistics and subject scores as well
skew	Should the skews and kurtosi of the raw data be reported? Defaults to FALSE because what is the meaning of skew for a multiple choice item?

Details

Basically combines score.items with a conversion from multiple choice to right/wrong.

The item-whole correlation is inflated because of item overlap.

The example data set is taken from the Synthetic Aperture Personality Assessment personality and ability test at http://test.personality-project.org.

Value

scores	Subject scores on one scale
missing	Number of missing items for each subject
item.stats	scoring key, response frequencies, item whole correlations, n subjects scored, mean, sd, skew, kurtosis and se for each item
alpha	Cronbach's coefficient alpha
av.r	Average interitem correlation

Author(s)

William Revelle

See Also

score.items, omega

Examples

```
data(iqitems)
iq.keys <- c(4,4,4, 6,6,3,4,4,  5,2,2,4,  3,2,6,7)
score.multiple.choice(iq.keys,iqitems)
#just convert the items to true or false
iq.tf <- score.multiple.choice(iq.keys,iqitems,score=FALSE)
describe(iq.tf)  #compare to previous results
```

---

scoreItems   *Score item composite scales and find Cronbach's alpha, Guttman lambda 6 and item whole correlations*

Description

Given a matrix or data.frame of k keys for n items (-1, 0, 1), and a matrix or data.frame of items scores for m items and N people, find the sum scores or average scores for each person and each scale. In addition, report Cronbach's alpha, Guttman's Lambda 6, the average r, the scale intercorrelations, and the item by scale correlations (raw and corrected for item overlap). Replace missing values with the item median or mean if desired. Will adjust scores for reverse scored items. See make.keys for a convenient way to make the keys file. If the input is a square matrix, then it is assumed that the input is a covariance or correlation matix and scores are not found, but the item statistics are reported. (Similar functionality to cluster.cor). response.frequencies reports the frequency of item endorsements fore each response category for polytomous or multiple choice items.

Usage

```
scoreItems(keys, items, totals = FALSE, ilabels = NULL,missing=TRUE, impute="median",
 delete=TRUE, min = NULL, max = NULL, digits = 2)
 score.items(keys, items, totals = FALSE, ilabels = NULL,missing=TRUE, impute="median",
 delete=TRUE, min = NULL, max = NULL, digits = 2)
  response.frequencies(items,max=10,uniqueitems=NULL)
```

Arguments

keys	A matrix or dataframe of -1, 0, or 1 weights for each item on each scale. May be created by hand, or by using make.keys
items	Matrix or dataframe of raw item scores
totals	if TRUE find total scores, if FALSE (default), find average scores
ilabels	a vector of item labels.
missing	missing = TRUE is the normal case and data are imputed according to the impute option. missing=FALSE, only complete cases are scored.

impute	impute="median" replaces missing values with the item median, impute = "mean" replaces values with the mean response. impute="none" the subject's scores are based upon the average of the keyed, but non missing scores.
delete	if delete=TRUE, automatically delete items with no variance (and issue a warning)
min	May be specified as minimum item score allowed, else will be calculated from data. min and max should be specified if items differ in their possible minima or maxima. See notes for details.
max	May be specified as maximum item score allowed, else will be calculated from data. Alternatively, in response frequencies, it is maximum number of alternative responses to count.
uniqueitems	If specified, the set of possible unique response categories
digits	Number of digits to report

Details

The process of finding sum or average scores for a set of scales given a larger set of items is a typical problem in applied psychometrics and in psychometric research. Although the structure of scales can be determined from the item intercorrelations, to find scale means, variances, and do further analyses, it is typical to find scores based upon the sum or the average item score. For some strange reason, personality scale scores are typically given as totals, but attitude scores as averages. The default for scoreItems is the average as it would seem to make more sense to report scale scores in the metric of the item.

Various estimates of scale reliability include "Cronbach's alpha", Guttman's Lambda 6, and the average interitem correlation. For k = number of items in a scale, and av.r = average correlation between items in the scale, alpha = k * av.r/(1+ (k-1)*av.r). Thus, alpha is an increasing function of test length as well as the test homeogeneity.

Surprisingly, more than a century after Spearman (1904) introduced the concept of reliability to psychologists, there are still multiple approaches for measuring it. Although very popular, Cronbach's α (1951) underestimates the reliability of a test and over estimates the first factor saturation.

α (Cronbach, 1951) is the same as Guttman's λ_3 (Guttman, 1945) and may be found by

$$\lambda_3 = \frac{n}{n-1}\Bigl(1 - \frac{tr(\vec{V})_x}{V_x}\Bigr) = \frac{n}{n-1}\frac{V_x - tr(\vec{V}_x)}{V_x} = \alpha$$

Perhaps because it is so easy to calculate and is available in most commercial programs, alpha is without doubt the most frequently reported measure of internal consistency reliability. Alpha is the mean of all possible spit half reliabilities (corrected for test length). For a unifactorial test, it is a reasonable estimate of the first factor saturation, although if the test has any microstructure (i.e., if it is "lumpy") coefficients β (Revelle, 1979; see ICLUST) and ω_h (see omega) (McDonald, 1999; Revelle and Zinbarg, 2009) are more appropriate estimates of the general factor saturation. ω_t (see omega) is a better estimate of the reliability of the total test.

Guttman's Lambda 6 (G6) considers the amount of variance in each item that can be accounted for the linear regression of all of the other items (the squared multiple correlation or smc), or more precisely, the variance of the errors, e_j^2, and is

$$\lambda_6 = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum(1 - r_{smc}^2)}{V_x}.$$

The squared multiple correlation is a lower bound for the item communality and as the number of items increases, becomes a better estimate.

G6 is also sensitive to lumpyness in the test and should not be taken as a measure of unifactorial structure. For lumpy tests, it will be greater than alpha. For tests with equal item loadings, alpha > G6, but if the loadings are unequal or if there is a general factor, G6 > alpha. Although it is normal when scoring just a single scale to calculate G6 from just those items within the scale, logically it is appropriate to estimate an item reliability from all items available. This is done here and is labeled as G6* to identify the subtle difference.

Alpha and G6* are both positive functions of the number of items in a test as well as the average intercorrelation of the items in the test. When calculated from the item variances and total test variance, as is done here, raw alpha is sensitive to differences in the item variances. Standardized alpha is based upon the correlations rather than the covariances. alpha is a generalization of an earlier estimate of reliability for tests with dichotomous items developed by Kuder and Richardson, known as KR20, and a shortcut approximation, KR21. (See Revelle, in prep; Revelle and Condon, in press.).

A useful index is the ratio of reliable variance to unreliable variance and is known as the Signal/Noise ratio. This is just

$$s/n = \frac{n\bar{r}}{1 - n\bar{r}}$$

(Cronbach and Gleser, 1964; Revelle and Condon (in press)).

Standard errors for unstandardized alpha are reported using the formula from Duhachek and Iacobucci (2005).

More complete reliability analyses of a single scale can be done using the `omega` function which finds ω_h and ω_t based upon a hierarchical factor analysis. Alternative estimates of the Greatest Lower Bound for the reliability are found in the `guttman` function.

Alpha is a poor estimate of the general factor saturation of a test (see Revelle and Zinbarg, 2009; Zinbarg et al., 2005) for it can seriously overestimate the size of a general factor, and a better but not perfect estimate of total test reliability because it underestimates total reliability. None the less, it is a common statistic to report. In general, the use of alpha should be discouraged and the use of more appropriate estimates (ω_h and ω_t) should be encouraged.

Correlations between scales are attenuated by a lack of reliability. Correcting correlations for reliability (by dividing by the square roots of the reliabilities of each scale) sometimes help show structure.

By default, missing values are replaced with the corresponding median value for that item. Means can be used instead (impute="mean"), or subjects with missing data can just be dropped (missing = FALSE). For data with a great deal of missingness, yet another option is to just find the average of the available responses (impute="none"). This is useful for findings means for scales for the SAPA project (see `https://sapa-project.org`) where most scales are estimated from random sub samples of the items from the scale. In this case, the alpha reliabilities are seriously overinflated because they are based upon the total number of items in each scale. The "alpha observed" values are based upon the average number of items answered in each scale using the standard form for alpha a function of inter-item correlation and number of items.

`scoreItems` can be applied to correlation matrices to find just the reliability statistics. This will be done automatically if the items matrix is square and none of the values in the matrix are less than -1 or greater than 1.

Value

scores	Sum or average scores for each subject on the k scales
alpha	Cronbach's coefficient alpha. A simple (but non-optimal) measure of the internal consistency of a test. See also beta and omega. Set to 1 for scales of length 1.
av.r	The average correlation within a scale, also known as alpha 1, is a useful index of the internal consistency of a domain. Set to 1 for scales with 1 item.
G6	Guttman's Lambda 6 measure of reliability
G6*	A generalization of Guttman's Lambda 6 measure of reliability using all the items to find the smc.
n.items	Number of items on each scale
item.cor	The correlation of each item with each scale. Because this is not corrected for item overlap, it will overestimate the amount that an item correlates with the other items in a scale.
cor	The intercorrelation of all the scales based upon the interitem correlations (see note for why these differ from the correlations of the observed scales themselves).
corrected	The correlations of all scales (below the diagonal), alpha on the diagonal, and the unattenuated correlations (above the diagonal)
item.corrected	The item by scale correlations for each item, corrected for item overlap by replacing the item variance with the smc for that item
response.freq	The response frequency (based upon number of non-missing responses) for each alternative.
missing	How many items were not answered for each scale
num.ob.item	The average number of items with responses on a scale. Used in calculating the alpha.observed– relevant for SAPA type data structures.

Note

It is important to recognize in the case of massively missing data (e.g., data from a Synthetic Aperture Personality Assessment (https://sapa-project.org) study where perhaps only 10-50% of the items per scale are given to any one subject)) that the number of items per scale, and hence standardized alpha, is not the nominal value and hence alpha of the observed scales will be overestimated. For this case (impute="none"), an additional alpha (alpha.ob) is reported.

More importantly in this case of massively missing data, there is a difference between the correlations of the composite scales based upon the correlations of the items and the correlations of the scored scales based upon the observed data. That is, the cor object will have correlations as if all items had been given, while the correlation of the scores object will reflect the actual correlation of the scores. For SAPA data, it is recommended to use the cor object. Confidence of these correlations may be found using the cor.ci function.

Further note that the inter-scale correlations are based upon the correlations of scales formed from the covariance matrix of the items. This will differ from the correlation of scales based upon the correlation of the items. Thus, although scoreItems will produce reliabilities and intercorrelations from either the raw data or from a correlation matrix, these values will differ slightly. In addition, with a great deal of missing data, the scale intercorrelations will differ from the correlations of the scores produced, for the latter will be attenuated.

An alternative to classical test theory scoring is to use score.irt to find score estimates based upon Item Response Theory. This is particularly useful in the case of SAPA data which tend to be massively missing. It is also useful to find scores based upon polytomous items following a factor analysis of the polychoric correlation matrix (see irt.fa).

When reverse scoring items from a set where items differ in their possible minima or maxima, it is important to specify the min and max values. Items are reversed by subtracting them from max + min. Thus, if items range from 1 to 6, items are reversed by subtracting them from 7. But, if the data set includes other variables, (say an id field) that far exceeds the item min or max, then the max id will incorrectly be used to reverse key. min and max can either be single values, or vectors for all items.

Author(s)

William Revelle

References

Cronbach, L.J. and Gleser G.C. (1964)The signal/noise ratio in the comparison of reliability coefficients. Educational and Psychological Measurement, 24 (3) 467-480.

Duhachek, A. and Iacobucci, D. (2004). Alpha's standard error (ase): An accurate and precise confidence interval estimate. Journal of Applied Psychology, 89(5):792-808.

McDonald, R. P. (1999). Test theory: A unified treatment. L. Erlbaum Associates, Mahwah, N.J.

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. http://personality-project.org/r/book

Revelle, W. and Condon, D.C. Reliability. In Irwing, P., Booth, T. and Hughes, D. (Eds). the Wiley-Blackwell Handbook of Psychometric Testing (in press).

Revelle W. and R.E. Zinbarg. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74(1):145-154.

Zinbarg, R. E., Revelle, W., Yovel, I. and Li, W. (2005) Cronbach's alpha, Revelle's beta, and McDonald's omega h, Their relations with each other and two alternative conceptualizations of reliability, Psychometrika, 70, 123-133.

See Also

make.keys for a convenient way to create the keys file, score.multiple.choice for multiple choice items,
alpha, correct.cor, cluster.cor , cluster.loadings, omega, guttman for item/scale analysis.

If scales are formed from overlapping sets of items, their correlations will be inflated. This is corrected for when using the scoreOverlap function which, although it will not produce scores, will report scale intercorrelations corrected for item overlap.

In addition, the irt.fa function provides an alternative way of examining the structure of a test and emphasizes item response theory approaches to the information returned by each item and the total test. Associated with these IRT parameters is the score.irt function for finding IRT based scores as well as irt.responses to show response curves for the alternatives in a multiple choice test.

Examples

```
#see  the example including the bfi data set
data(bfi)
keys.list <- list(agree=c("-A1","A2","A3","A4","A5"),
  conscientious=c("C1","C2","C3","-C4","-C5"),extraversion=c("-E1","-E2","E3","E4","E5"),
  neuroticism=c("N1","N2","N3","N4","N5"), openness = c("O1","-O2","O3","O4","-O5"))
  keys <- make.keys(bfi,keys.list)
 scores <- scoreItems(keys,bfi,min=1,max=6)
 summary(scores)
 #to get the response frequencies, we need to not use the age variable
 scores <- scoreItems(keys[1:27,],bfi[1:27],min=1,max=6)
 scores
 #The scores themselves are available in the scores$scores object.  I.e.,
 describe(scores$scores)


 #compare this output to that for the impute="none" option for SAPA type data
 #first make many of the items missing in a missing pattern way
 missing.bfi <- bfi
 missing.bfi[1:1000,3:8] <- NA
 missing.bfi[1001:2000,c(1:2,9:10)] <- NA
 scores <- scoreItems(keys,missing.bfi,impute="none",min=1,max=6)
 scores
 describe(scores$scores)  #the actual scores themselves
```

---

scoreOverlap *Find correlations of composite variables (corrected for overlap) from a larger matrix.*

Description

Given a n x c cluster definition matrix of -1s, 0s, and 1s (the keys) , and a n x n correlation matrix, or an N x n data matrix, find the correlations of the composite clusters. The keys matrix can be entered by hand, copied from the clipboard (read.clipboard), or taken as output from the factor2cluster or make.keys functions. Similar functionality to scoreItems which also gives item by cluster correlations.

Usage

```
scoreOverlap(keys, r, correct = TRUE, SMC = TRUE, av.r = TRUE, item.smc = NULL,
     impute = TRUE)
cluster.cor(keys, r.mat, correct = TRUE,SMC=TRUE,item.smc=NULL,impute=TRUE)
```

Arguments

keys A matrix of cluster keys

r.mat	A correlation matrix
r	Either a correlation matrix or a raw data matrix
correct	TRUE shows both raw and corrected for attenuation correlations
SMC	Should squared multiple correlations be used as communality estimates for the correlation matrix?
item.smc	the smcs of the items may be passed into the function for speed, or calculated if SMC=TRUE
impute	if TRUE, impute missing scale correlations based upon the average interitem correlation, otherwise return NA.
av.r	Should the average r be used in correcting for overlap? smcs otherwise.

Details

This are two of the functions used in the SAPA (http://sapa-project.org) procedures to form synthetic correlation matrices. Given any correlation matrix of items, it is easy to find the correlation matrix of scales made up of those items. This can also be done from the original data matrix or from the correlation matrix using `scoreItems` which is probably preferred unless the keys are overlapping.

In the case of overlapping keys, (items being scored on multiple scales), `scoreOverlap` will adjust for this overlap by replacing the overlapping covariances (which are variances when overlapping) with the corresponding best estimate of an item's "true" variance using either the average correlation or the smc estimate for that item. This parallels the operation done when finding alpha reliability. This is similar to ideas suggested by Cureton (1966) and Bashaw and Anderson (1966) but uses the smc or the average interitem correlation (default).

A typical use in the SAPA project is to form item composites by clustering or factoring (see `fa`, `ICLUST`, `principal`), extract the clusters from these results (`factor2cluster`), and then form the composite correlation matrix using `cluster.cor`. The variables in this reduced matrix may then be used in multiple correlatin procedures using `mat.regress`.

The original correlation is pre and post multiplied by the (transpose) of the keys matrix.

If some correlations are missing from the original matrix this will lead to missing values (NA) for scale intercorrelations based upon those lower level correlations. If impute=TRUE (the default), a warning is issued and the correlations are imputed based upon the average correlations of the non-missing elements of each scale.

Because the alpha estimate of reliability is based upon the correlations of the items rather than upon the covariances, this estimate of alpha is sometimes called "standardized alpha". If the raw items are available, it is useful to compare standardized alpha with the raw alpha found using `scoreItems`. They will differ substantially only if the items differ a great deal in their variances.

`scoreOverlap` answers an important question when developing scales and related subscales, or when comparing alternative versions of scales. For by removing the effect of item overlap, it gives a better estimate the relationship between the latent variables estimated by the observed sum (mean) scores.

Value

cor	the (raw) correlation matrix of the clusters

sd	standard deviation of the cluster scores
corrected	raw correlations below the diagonal, alphas on diagonal, disattenuated above diagonal
alpha	The (standardized) alpha reliability of each scale.
G6	Guttman's Lambda 6 reliability estimate is based upon the smcs for each item in a scale. G6 uses the smc based upon the entire item domain.
av.r	The average inter item correlation within a scale
size	How many items are in each cluster?

Note

See SAPA Revelle, W., Wilt, J., and Rosenthal, A. (2010) Personality and Cognition: The Personality-Cognition Link. In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

The second example uses the msq data set of 72 measures of motivational state to examine the overlap between four lower level scales and two higher level scales.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

Bashaw, W. and Anderson Jr, H. E. (1967). A correction for replicated error in correlation coefficients. Psychometrika, 32(4):435-441.

Cureton, E. (1966). Corrected item-test correlations. Psychometrika, 31(1):93-96.

See Also

factor2cluster, mat.regress, alpha, and most importantly, scoreItems, which will do all of what cluster.cor does for most users. cluster.cor is an important helper function for iclust

Examples

```
#use the msq data set that shows the structure of energetic and tense arousal
small.msq <- msq[ c("active", "energetic", "vigorous", "wakeful", "wide.awake",
"full.of.pep", "lively", "sleepy", "tired", "drowsy","intense", "jittery", "fearful",
 "tense", "clutched.up", "quiet", "still",    "placid", "calm", "at.rest") ]
small.R <- cor(small.msq,use="pairwise")
keys <- make.keys(small.R,list(
EA = c("active", "energetic", "vigorous", "wakeful", "wide.awake", "full.of.pep",
       "lively", "-sleepy", "-tired", "-drowsy"),
TA =c("intense", "jittery", "fearful", "tense", "clutched.up", "-quiet", "-still",
       "-placid", "-calm", "-at.rest") ,

high.EA = c("active", "energetic", "vigorous", "wakeful", "wide.awake", "full.of.pep",
       "lively"),
low.EA =c("sleepy", "tired", "drowsy"),
lowTA= c("quiet", "still", "placid", "calm", "at.rest"),
```

```
highTA = c("intense", "jittery", "fearful", "tense", "clutched.up")
   ))

adjusted.scales <- scoreOverlap(keys,small.R)
#compare with unadjusted
confounded.scales <- cluster.cor(keys,small.R)
summary(adjusted.scales)
summary(confounded.scales)
```

---

scrub *A utility for basic data cleaning and recoding. Changes values outside of minimum and maximum limits to NA.*

Description

A tedious part of data analysis is addressing the problem of miscoded data that need to be converted to NA or some other value. For a given data.frame or matrix, scrub will set all values of columns from=from to to=to that are less than a set (vector) of min values or more than a vector of max values to NA. Can also be used to do basic recoding of data for all values=isvalue to newvalue.

The length of the where, isvalue, and newvalues must either match, or be 1.

Usage

```
scrub(x, where, min, max,isvalue,newvalue)
```

Arguments

x	a data frame or matrix
where	The variables to examine. (Can be by name or by column number)
min	a vector of minimum values that are acceptable
max	a vector of maximum values that are acceptable
isvalue	a vector of values to be converted to newvalue (one per variable)
newvalue	a vector of values to replace those that match isvalue

Details

Solves a tedious problem that can be done directly but that is sometimes awkward. Will either replace specified values with NA or

Value

The corrected data frame.

Note

Probably could be optimized to avoid one loop

Author(s)

William Revelle

See Also

reverse.code, rescale for other simple utilities.

Examples

```
data(attitude)
x <- scrub(attitude,isvalue=55) #make all occurrences of 55 NA
x1 <- scrub(attitude, where=c(4,5,6), isvalue =c(30,40,50),
     newvalue = c(930,940,950)) #will do this for the 4th, 5th, and 6th variables
x2 <- scrub(attitude, where=c(4,4,4), isvalue =c(30,40,50),
            newvalue = c(930,940,950)) #will just do it for the 4th column
#get rid of a complicated set of cases and replace with missing values
y <- scrub(attitude,where=2:4,min=c(20,30,40),max= c(120,110,100),isvalue= c(32,43,54))
y1 <- scrub(attitude,where="learning",isvalue=55,newvalue=999) #change a column by name
y2 <- scrub(attitude,where="learning",min=45,newvalue=999) #change a column by name

y3 <- scrub(attitude,where="learning",isvalue=c(45,48),
    newvalue=999) #change a column by name look for multiple values in that column
y4 <- scrub(attitude,where="learning",isvalue=c(45,48),
      newvalue= c(999,-999)) #change values in one column to one of two different things
```

SD *Find the Standard deviation for a vector, matrix, or data.frame - do not return error if there are no cases*

Description

Find the standard deviation of a vector, matrix, or data.frame. In the latter two cases, return the sd of each column. Unlike the sd function, return NA if there are no observations rather than throw an error.

Usage

```
SD(x, na.rm = TRUE)   #deprecated
```

Arguments

x	a vector, data.frame, or matrix
na.rm	na.rm is assumed to be TRUE

Details

Finds the standard deviation of a vector, matrix, or data.frame. Returns NA if no cases.

Just an adaptation of the stats:sd function to return the functionality found in R < 2.7.0 or R >= 2.8.0 Because this problem seems to have been fixed, SD will be removed eventually.

Value

The standard deviation

Note

Until R 2.7.0, sd would return a NA rather than an error if no cases were observed. SD brings back that functionality. Although unusual, this condition will arise when analyzing data with high rates of missing values. This function will probably be removed as 2.7.0 becomes outdated.

Author(s)

William Revelle

See Also

These functions use SD rather than sd: describe.by, skew, kurtosi

Examples

```
data(attitude)
apply(attitude,2,sd) #all complete
attitude[,1] <- NA
SD(attitude) #missing a column
describe(attitude)
```

---

setCor *Set Correlation and Multiple Regression from matrix or raw input*

Description

Finds Cohen's Set Correlation between a predictor set of variables (x) and a criterion set (y). Also finds multiple correlations between x variables and each of the y variables. Will work with either raw data or a correlation matrix. A set of covariates (z) can be partialled from the x and y sets. Regression diagrams are automatically included.

Usage

```
setCor(y,x,data,z=NULL,n.obs=NULL,use="pairwise",std=TRUE,square=FALSE,
       main="Regression Models",plot=TRUE)
setCor.diagram(sc,main="Regression model",digits=2,show=TRUE,...)
set.cor(y,x,data,z=NULL,n.obs=NULL,use="pairwise",std=TRUE,square=FALSE,
       main="Regression Models",plot=TRUE)    #an alias to setCor
mat.regress(y, x,data, z=NULL,n.obs=NULL,use="pairwise",square=FALSE)
matReg(x,y,C,n.obs=0)
```

Arguments

`y`	either the column numbers of the y set (e.g., c(2,4,6) or the column names of the y set (e.g., c("Flags","Addition")
`x`	either the column numbers of the x set (e.g., c(1,3,5) or the column names of the x set (e.g. c("Cubes","PaperFormBoard")
`data`	a matrix or data.frame of correlations or, if not square, of raw data
`C`	A variance/covariance matrix, or a correlation matrix
`z`	the column names or numbers of the set of covariates
`n.obs`	If specified, then confidence intervals, etc. are calculated, not needed if raw data are given
`use`	find the correlations "pairwise" (default) or just use "complete" cases (to match the lm function)
`std`	Report standardized betas (based upon the correlations) or raw (based upon covariances)
`main`	The title for setCor.diagram
`square`	if FALSE, then square matrices are treated as correlation matrices not as data matrices. In the rare case that one has as many cases as variables, then set square=TRUE.
`sc`	The output of setCor may be used for drawing diagrams
`digits`	How many digits should be displayed in the setCor.diagram?
`show`	Show the matrix correlation between the x and y sets?
`plot`	By default, setCor makes a plot of the results, set to FALSE to suppress the plot
`...`	Additional graphical parameters for setCor.diagram

Details

Although it is more common to calculate multiple regression and canonical correlations from the raw data, it is, of course, possible to do so from a matrix of correlations or covariances. In this case, the input to the function is a square covariance or correlation matrix, as well as the column numbers (or names) of the x (predictor), y (criterion) variables, and if desired z (covariates). The function will find the correlations if given raw data.

The output is a set of multiple correlations, one for each dependent variable in the y set, as well as the set of canonical correlations.

An additional output is the R2 found using Cohen's set correlation (Cohen, 1982). This is a measure of how much variance and the x and y set share.

Cohen (1982) introduced the set correlation, a multivariate generalization of the multiple correlation to measure the overall relationship between two sets of variables. It is an application of canoncial correlation (Hotelling, 1936) and $1 - \prod(1 - \rho_i^2)$ where ρ_i^2 is the squared canonical correlation. Set correlation is the amount of shared variance (R2) between two sets of variables. With the addition of a third, covariate set, set correlation will find multivariate R2, as well as partial and semi partial R2. (The semi and bipartial options are not yet implemented.) Details on set correlation may be found in Cohen (1982), Cohen (1988) and Cohen, Cohen, Aiken and West (2003).

R2 between two sets is just

$$R^2 = 1 - \frac{|R_{yx}|}{|R_y|\,|R_x|} = 1 - \prod(1 - \rho_i^2)$$

where R is the complete correlation matrix of the x and y variables and Rx and Ry are the two sets involved.

Unfortunately, the R2 is sensitive to one of the canonical correlations being very high. An alternative, T2, is the proportion of additive variance and is the average of the squared canonicals. (Cohen et al., 2003), see also Cramer and Nicewander (1979). This average, because it includes some very small canonical correlations, will tend to be too small. Cohen et al. admonition is appropriate: "In the final analysis, however, analysts must be guided by their substantive and methodological conceptions of the problem at hand in their choice of a measure of association." (p613).

Yet another measure of the association between two sets is just the simple, unweighted correlation between the two sets. That is,

$$R_{uw} = \frac{1R_{xy}1'}{(1R_{yy}1')^{.5}(1R_{xx}1')^{.5}}$$

where Rxy is the matrix of correlations between the two sets. This is just the simple (unweighted) sums of the correlations in each matrix. This technique exemplifies the robust beauty of linear models and is particularly appropriate in the case of one dimension in both x and y, and will be a drastic underestimate in the case of items where the betas differ in sign.

When finding the unweighted correlations, as is done in `alpha`, items are flipped so that they all are positively signed.

A typical use in the SAPA project is to form item composites by clustering or factoring (see `fa`,`ICLUST`, `principal`), extract the clusters from these results (`factor2cluster`), and then form the composite correlation matrix using `cluster.cor`. The variables in this reduced matrix may then be used in multiple R procedures using `set.cor`.

Although the overall matrix can have missing correlations, the correlations in the subset of the matrix used for prediction must exist.

If the number of observations is entered, then the conventional confidence intervals, statistical significance, and shrinkage estimates are reported.

If the input is rectangular, correlations or covariances are found from the data.

The print function reports t and p values for the beta weights, the summary function just reports the beta weights.

`matReg` is primarily a helper function for `mediate` but is a general multiple regression function given a covariance matrix and the specified x, and y variables. Its output includes betas, se, t, p and R2. It does not work on data matrices, nor does it take formula input.

Value

`beta`	the beta weights for each variable in X for each variable in Y
`R`	The multiple R for each equation (the amount of change a unit in the predictor set leads to in the criterion set).
`R2`	The multiple R2 (% variance acounted for) for each equation

se	Standard errors of beta weights (if n.obs is specified)
t	t value of beta weights (if n.obs is specified)
Probability	Probability of beta = 0 (if n.obs is specified)
shrunkenR2	Estimated shrunken R2 (if n.obs is specified)
setR2	The multiple R2 of the set correlation between the x and y sets

itemresidualThe residual correlation matrix of Y with x and z removed

ruw	The unit weighted multiple correlation
Ruw	The unit weighted set correlation

Note

As of April 30, 2011, the order of x and y was swapped in the call to be consistent with the general y ~ x syntax of the lm and aov functions. In addition, the primary name of the function was switched to set.cor from mat.regress to reflect the estimation of the set correlation.

The denominator degrees of freedom for the set correlation does not match that reported by Cohen et al., 2003 in the example on page 621 but does match the formula on page 615, except for the typo in the estimation of F (see Cohen 1982). The difference seems to be that they are adding in a correction factor of df 2 = df2 + df1.

Author(s)

William Revelle

Maintainer: William Revelle <revelle@northwestern.edu>

References

J. Cohen (1982) Set correlation as a general multivariate data-analytic method. Multivariate Behavioral Research, 17(3):301-341.

J. Cohen, P. Cohen, S.G. West, and L.S. Aiken. (2003) Applied multiple regression/correlation analysis for the behavioral sciences. L. Erlbaum Associates, Mahwah, N.J., 3rd ed edition.

H. Hotelling. (1936) Relations between two sets of variates. Biometrika 28(3/4):321-377.

E.Cramer and W. A. Nicewander (1979) Some symmetric, invariant measures of multivariate association. Psychometrika, 44:43-54.

See Also

cluster.cor, factor2cluster,principal,ICLUST, link{cancor} and cca in the yacca package.

Examples

```
#the Kelly data from Hoteling
kelly <- structure(list(speed = c(1, 0.4248, 0.042, 0.0215, 0.0573), power = c(0.4248,
1, 0.1487, 0.2489, 0.2843), words = c(0.042, 0.1487, 1, 0.6693,
0.4662), symbols = c(0.0215, 0.2489, 0.6693, 1, 0.6915), meaningless = c(0.0573,
```

```
0.2843, 0.4662, 0.6915, 1)), .Names = c("speed", "power", "words",
"symbols", "meaningless"), class = "data.frame", row.names = c("speed",
"power", "words", "symbols", "meaningless"))


kelly

setCor(1:2,3:5,kelly)
#Hotelling reports canonical correlations of .3073 and .0583  or squared correlations of
# 0.09443329 and 0.00339889 vs. our values of  0.0946 0.0035,


setCor(y=c(7:9),x=c(1:6),data=Thurstone,n.obs=213)
#now try partialling out some variables
set.cor(y=c(7:9),x=c(1:3),z=c(4:6),data=Thurstone) #compare with the previous
#compare complete print out with summary printing
sc <- setCor(x=c("gender","education"),y=c("SATV","SATQ"),data=sat.act) # regression from raw data
sc
summary(sc)
```

---

sim *Functions to simulate psychological/psychometric data.*

Description

A number of functions in the psych package will generate simulated data with particular structures. These functions include `sim` for a factor simplex, and `sim.simplex` for a data simplex, `sim.circ` for a circumplex structure, `sim.congeneric` for a one factor factor congeneric model, `sim.dichot` to simulate dichotomous items, `sim.hierarchical` to create a hierarchical factor model, `sim.item` a more general item simulation, `sim.minor` to simulate major and minor factors, `sim.omega` to test various examples of omega, `sim.parallel` to compare the efficiency of various ways of deterimining the number of factors, `sim.rasch` to create simulated rasch data, `sim.irt` to create general 1 to 4 parameter IRT data by calling `sim.npl` 1 to 4 parameter logistic IRT or `sim.npn` 1 to 4 paramater normal IRT, `sim.poly` to create polytomous ideas by calling `sim.poly.npn` 1-4 parameter polytomous normal theory items or `sim.poly.npl` 1-4 parameter polytomous logistic items, and `sim.poly.ideal` which creates data following an ideal point or unfolding model by calling `sim.poly.ideal.npn` 1-4 parameter polytomous normal theory ideal point model or `sim.poly.ideal.npl` 1-4 parameter polytomous logistic ideal point model.

`sim.structural` a general simulation of structural models, and `sim.anova` for ANOVA and lm simulations, and `sim.VSS`. Some of these functions are separately documented and are listed here for ease of the help function. See each function for more detailed help.

Usage

```
sim(fx=NULL,Phi=NULL,fy=NULL,alpha=.8,lambda = 0,n=0,mu=NULL,raw=TRUE)
sim.simplex(nvar =12, alpha=.8,lambda=0,beta=1,mu=NULL, n=0)
sim.general(nvar=9,nfact =3, g=.3,r=.3,n=0)
sim.minor(nvar=12,nfact=3,n=0,g=NULL,fbig=NULL,fsmall = c(-.2,.2),bipolar=TRUE)
```

```
sim.omega(nvar=12,nfact=3,n=500,g=NULL,sem=FALSE,fbig=NULL,fsmall =
  c(-.2,.2),bipolar=TRUE,om.fact=3,flip=TRUE,option="equal",ntrials=10)
sim.parallel(ntrials=10,nvar = c(12,24,36,48),nfact = c(1,2,3,4,6),
n = c(200,400))
sim.rasch(nvar = 5,n = 500, low=-3,high=3,d=NULL, a=1,mu=0,sd=1)
sim.irt(nvar = 5, n = 500, low=-3, high=3,a=NULL,c=0,z=1,d=NULL,mu=0,sd=1,mod="logistic")
sim.npl(nvar = 5, n = 500, low=-3,high=3,a=NULL,c=0,z=1,d=NULL,mu=0,sd=1)
sim.npn(nvar = 5, n = 500, low=-3,high=3,a=NULL,c=0,z=1,d=NULL,mu=0,sd=1)
sim.poly(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,
    mu=0,sd=1,cat=5,mod="logistic")
sim.poly.npn(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL, mu=0,sd=1,cat=5)
sim.poly.npl(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL, mu=0,sd=1,cat=5)
sim.poly.ideal(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,
   mu=0,sd=1,cat=5,mod="logistic")
sim.poly.ideal.npn(nvar = 5,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL, mu=0,sd=1,cat=5)
sim.poly.ideal.npl(nvar = 5,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,
      mu=0,sd=1,cat=5,theta=NULL)
sim.poly.mat(R,m,n)
```

Arguments

fx	The measurement model for x. If NULL, a 4 factor model is generated
Phi	The structure matrix of the latent variables
fy	The measurement model for y
mu	The means structure for the fx factors
n	Number of cases to simulate. If n=0 or NULL, the population matrix is returned.
raw	if raw=TRUE, raw data are returned as well.
nvar	Number of variables for a simplex structure
nfact	Number of large factors to simulate in sim.minor,number of group factors in sim.general,sim.omega
g	General factor correlations in sim.general and general factor loadings in sim.omega and sim.minor
sem	Should the sim.omega function do both an EFA omega as well as a CFA omega using the sem package?
r	group factor correlations in sim.general
alpha	the base correlation for an autoregressive simplex
lambda	the trait component of a State Trait Autoregressive Simplex
beta	Test reliability of a STARS simplex
fbig	Factor loadings for the main factors. Default is a simple structure with loadings sampled from (.8,.6) for nvar/nfact variables and 0 for the remaining. If fbig is specified, then each factor has loadings sampled from it.
bipolar	if TRUE, then positive and negative loadings are generated from fbig
om.fact	Number of factors to extract in omega

`flip`	In omega, should item signs be flipped if negative
`option`	In omega, for the case of two factors, how to weight them?
`fsmall`	nvar/2 small factors are generated with loadings sampled from (-.2,0,.2)
`ntrials`	Number of replications per level
`low`	lower difficulty for sim.rasch or sim.irt
`high`	higher difficulty for sim.rasch or sim.irt
`a`	if not specified as a vector, the descrimination parameter a = α will be set to 1.0 for all items
`d`	if not specified as a vector, item difficulties (d = δ) will range from low to high
`c`	the gamma parameter: if not specified as a vector, the guessing asymptote is set to 0
`z`	the zeta parameter: if not specified as a vector, set to 1
`sd`	the standard deviation for the underlying latent variable in the irt simulations
`mod`	which IRT model to use, mod="logistic" simulates a logistic function, otherwise, a normal function
`cat`	Number of categories to simulate in sim.poly. If cat=2, then this is the same as simulating t/f items and sim.poly is functionally equivalent to sim.irt
`theta`	The underlying latent trait value for each simulated subject
`R`	A correlation matrix to be simulated using the sim.poly.mat function
`m`	The matrix of marginals for all the items

Details

Simulation of data structures is a very useful tool in psychometric research and teaching. By knowing “truth" it is possible to see how well various algorithms can capture it. For a much longer discussion of the use of simulation in psychometrics, see the accompany vignettes.

The simulations documented here are a miscellaneous set of functions that will be documented in other help files eventually.

The default values for `sim.structure` is to generate a 4 factor, 12 variable data set with a simplex structure between the factors. This, and the simplex of items (`sim.simplex`) can also be converted in a STARS model with an autoregressive component (alpha) and a stable trait component (lambda).

Two data structures that are particular challenges to exploratory factor analysis are the simplex structure and the presence of minor factors. Simplex structures `sim.simplex` will typically occur in developmental or learning contexts and have a correlation structure of r between adjacent variables and r^n for variables n apart. Although just one latent variable (r) needs to be estimated, the structure will have nvar-1 factors.

An alternative version of the simplex is the State-Trait-Auto Regressive Structure (STARS) which has both a simplex state structure, with autoregressive path alpha and a trait structure with path lambda. This simulated in `sim.simplex` by specifying a non-zero lambda value.

Many simulations of factor structures assume that except for the major factors, all residuals are normally distributed around 0. An alternative, and perhaps more realistic situation, is that the there are a few major (big) factors and many minor (small) factors. The challenge is thus to identify the major factors. sim.minor generates such structures. The structures generated can be thought of as havinga a major factor structure with some small correlated residuals. To make these simulations complete, the possibility of a general factor is considered. For simplicity, sim.minor allows one to specify a set of loadings to be sampled from for g, fmajor and fminor. Alternatively, it is possible to specify the complete factor matrix.

Another structure worth considering is direct modeling of a general factor with several group factors. This is done using sim.general.

Although coefficient ω is a very useful indicator of the general factor saturation of a unifactorial test (one with perhaps several sub factors), it has problems with the case of multiple, independent factors. In this situation, one of the factors is labelled as "general" and the omega estimate is too large. This situation may be explored using the sim.omega function with general left as NULL. If there is a general factor, then results from sim.omega suggests that omega estimated either from EFA or from SEM does a pretty good job of identifying it but that the EFA approach using Schmid-Leiman transformation is somewhat more robust than the SEM approach.

The four irt simulations, sim.rasch, sim.irt, sim.npl and sim.npn, simulate dichotomous items following the Item Response model. sim.irt just calls either sim.npl (for logistic models) or sim.npn (for normal models) depending upon the specification of the model.

The logistic model is

$$P(i,j) = \gamma + \frac{\zeta - \gamma}{1 + e^{\alpha(\delta - \theta)}}$$

where γ is the lower asymptote or guesssing parameter, ζ is the upper asymptote (normally 1), α is item discrimination and δ is item difficulty. For the 1 Paramater Logistic (Rasch) model, gamma=0, zeta=1, alpha=1 and item difficulty is the only free parameter to specify.

For the 2PL and 2PN models, a = α and d = δ are specified.
For the 3PL or 3PN models, items also differ in their guessing parameter c =γ.
For the 4PL and 4PN models, the upper asymptote, z= ζ is also specified.
(Graphics of these may be seen in the demonstrations for the logistic function.)

The normal model (irt.npn calculates the probability using pnorm instead of the logistic function used in irt.npl, but the meaning of the parameters are otherwise the same. With the a = α parameter = 1.702 in the logistic model the two models are practically identical.

In parallel to the dichotomous IRT simulations are the poly versions which simulate polytomous item models. They have the additional parameter of how many categories to simulate. In addition, the sim.poly.ideal functions will simulate an ideal point or unfolding model in which the response probability varies by the distance from each subject's ideal point. Some have claimed that this is a more appropriate model of the responses to personality questionnaires. It will lead to simplex like structures which may be fit by a two factor model. The middle items form one factor, the extreme a bipolar factor.

The previous functions all assume one latent trait. Alternatively, we can simulate dichotomous or polytomous items with a particular structure using the sim.poly.mat function. This takes as input the population correlation matrix, the population marginals, and the sample size. It returns categorical items with the specified structure.

Other simulation functions in psych are:

sim.structure A function to combine a measurement and structural model into one data matrix. Useful for understanding structural equation models. Combined with structure.diagram to see the proposed structure.

sim.congeneric A function to create congeneric items/tests for demonstrating classical test theory. This is just a special case of sim.structure.

sim.hierarchical A function to create data with a hierarchical (bifactor) structure.

sim.item A function to create items that either have a simple structure or a circumplex structure.

sim.circ Create data with a circumplex structure.

sim.dichot Create dichotomous item data with a simple or circumplex structure.

sim.minor Create a factor structure for nvar variables defined by nfact major factors and nvar/2 “minor" factors for n observations.

Although the standard factor model assumes that K major factors (K « nvar) will account for the correlations among the variables

$$R = FF' + U^2$$

where R is of rank P and F is a P x K matrix of factor coefficients and U is a diagonal matrix of uniquenesses. However, in many cases, particularly when working with items, there are many small factors (sometimes referred to as correlated residuals) that need to be considered as well. This leads to a data structure such that

$$R = FF' + MM' + U^2$$

where R is a P x P matrix of correlations, F is a P x K factor loading matrix, M is a P x P/2 matrix of minor factor loadings, and U is a diagonal matrix (P x P) of uniquenesses.

Such a correlation matrix will have a poor χ^2 value in terms of goodness of fit if just the K factors are extracted, even though for all intents and purposes, it is well fit.

sim.minor will generate such data sets with big factors with loadings of .6 to .8 and small factors with loadings of -.2 to .2. These may both be adjusted.

sim.parallel Create a number of simulated data sets using sim.minor to show how parallel analysis works. The general observation is that with the presence of minor factors, parallel analysis is probably best done with component eigen values rather than factor eigen values, even when using the factor model.

sim.anova Simulate a 3 way balanced ANOVA or linear model, with or without repeated measures. Useful for teaching research methods and generating teaching examples.

sim.multilevel To understand some of the basic concepts of multilevel modeling, it is useful to create multilevel structures. The correlations of aggregated data is sometimes called an 'ecological correlation'. That group level and individual level correlations are independent makes such inferences problematic. This simulation allows for demonstrations that correlations within groups do not imply, nor are implied by, correlations between group means.

Author(s)

William Revelle

References

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at http://personality-project.org/r/book/

See Also

See above

Examples

```
simplex <- sim.simplex() #create the default simplex structure
lowerMat(simplex) #the correlation matrix
#create a congeneric matrix
congeneric <- sim.congeneric()
lowerMat(congeneric)
R <- sim.hierarchical()
lowerMat(R)
#now simulate categorical items with the hierarchical factor structure.
#Let the items be dichotomous with varying item difficulties.
marginals = matrix(c(seq(.1,.9,.1),seq(.9,.1,-.1)),byrow=TRUE,nrow=2)
X <- sim.poly.mat(R=R,m=marginals,n=1000)
lowerCor(X) #show the raw correlations
#lowerMat(tetrachoric(X)$rho) # show the tetrachoric correlations (not run)
#generate a structure
fx <- matrix(c(.9,.8,.7,rep(0,6),c(.8,.7,.6)),ncol=2)
fy <- c(.6,.5,.4)
Phi <- matrix(c(1,0,.5,0,1,.4,0,0,0),ncol=3)
R <- sim.structure(fx,Phi,fy)
cor.plot(R$model) #show it graphically

simp <- sim.simplex()
#show the simplex structure using cor.plot
cor.plot(simp,colors=TRUE,main="A simplex structure")
#Show a STARS model
simp <- sim.simplex(alpha=.8,lambda=.4)
#show the simplex structure using cor.plot
cor.plot(simp,colors=TRUE,main="State Trait Auto Regressive Simplex" )
```

`sim.anova` *Simulate a 3 way balanced ANOVA or linear model, with or without repeated measures.*

Description

For teaching basic statistics, it is useful to be able to generate examples suitable for analysis of variance or simple linear models. sim.anova will generate the design matrix of three independent variables (IV1, IV2, IV3) with an arbitrary number of levels and effect sizes for each main effect and interaction. IVs can be either continuous or categorical and can have linear or quadratic effects. Either a single dependent variable or multiple (within subject) dependent variables are generated according to the specified model. The repeated measures are assumed to be tau equivalent with a specified reliability.

Usage

```
sim.anova(es1 = 0, es2 = 0, es3 = 0, es12 = 0, es13 = 0,
    es23 = 0, es123 = 0, es11=0,es22=0, es33=0,n = 2,n1 = 2, n2 = 2, n3 = 2,
    within=NULL,r=.8,factors=TRUE,center = TRUE,std=TRUE)
```

Arguments

`es1`	Effect size of IV1
`es2`	Effect size of IV2
`es3`	Effect size of IV3
`es12`	Effect size of the IV1 x IV2 interaction
`es13`	Effect size of the IV1 x IV3 interaction
`es23`	Effect size of the IV2 x IV3 interaction
`es123`	Effect size of the IV1 x IV2 * IV3 interaction
`es11`	Effect size of the quadratric term of IV1
`es22`	Effect size of the quadratric term of IV2
`es33`	Effect size of the quadratric term of IV3
`n`	Sample size per cell (if all variables are categorical) or (if at least one variable is continuous), the total sample size
`n1`	Number of levels of IV1 (0) if continuous
`n2`	Number of levels of IV2
`n3`	Number of levels of IV3
`within`	if not NULL, then within should be a vector of the means of any repeated measures.
`r`	the correlation between the repeated measures (if they exist). This can be thought of as the reliablility of the measures.
`factors`	report the IVs as factors rather than numeric
`center`	center=TRUE provides orthogonal contrasts, center=FALSE adds the minimum value + 1 to all contrasts
`std`	Standardize the effect sizes by standardizing the IVs

Details

A simple simulation for teaching about ANOVA, regression and reliability. A variety of demonstrations of the relation between anova and lm can be shown.

The default is to produce categorical IVs (factors). For more than two levels of an IV, this will show the difference between the linear model and anova in terms of the comparisons made.

The within vector can be used to add congenerically equivalent dependent variables. These will have intercorrelations (reliabilities) of r and means as specified as values of within.

To demonstrate the effect of centered versus non-centering, make factors = center=FALSE. The default is to center the IVs. By not centering them, the lower order effects will be incorrect given the higher order interaction terms.

Value

y.df is a data.frame of the 3 IV values as well as the DV values.

`IV1 ... IV3`	Independent variables 1 ... 3
`DV`	If there is a single dependent variable
`DV.1 ... DV.n`	If within is specified, then the n within subject dependent variables

Author(s)

William Revelle

See Also

The general set of simulation functions in the psych package `sim`

Examples

```
set.seed(42)
data.df <- sim.anova(es1=1,es2=.5,es13=1)  # one main effect and one interaction
describe(data.df)
pairs.panels(data.df)   #show how the design variables are orthogonal
#
summary(lm(DV~IV1*IV2*IV3,data=data.df))
summary(aov(DV~IV1*IV2*IV3,data=data.df))
set.seed(42)
 #demonstrate the effect of not centering the data on the regression
data.df <- sim.anova(es1=1,es2=.5,es13=1,center=FALSE)  #
describe(data.df)
#
#this one is incorrect, because the IVs are not centered
summary(lm(DV~IV1*IV2*IV3,data=data.df))

summary(aov(DV~IV1*IV2*IV3,data=data.df)) #compare with the lm model
#now examine multiple levels and quadratic terms
set.seed(42)
data.df <- sim.anova(es1=1,es13=1,n2=3,n3=4,es22=1)
summary(lm(DV~IV1*IV2*IV3,data=data.df))
summary(aov(DV~IV1*IV2*IV3,data=data.df))
pairs.panels(data.df)
#
data.df <- sim.anova(es1=1,es2=-.5,within=c(-1,0,1),n=10)
pairs.panels(data.df)
```

sim.congeneric *Simulate a congeneric data set*

Description

Classical Test Theory (CTT) considers four or more tests to be congenerically equivalent if all tests may be expressed in terms of one factor and a residual error. Parallel tests are the special case where (usually two) tests have equal factor loadings. Tau equivalent tests have equal factor loadings but may have unequal errors. Congeneric tests may differ in both factor loading and error variances.

Usage

```
sim.congeneric(loads = c(0.8, 0.7, 0.6, 0.5),N = NULL,  err=NULL, short = TRUE,
              categorical=FALSE, low=-3,high=3,cuts=NULL)
```

Arguments

N	How many subjects to simulate. If NULL, return the population model
loads	A vector of factor loadings for the tests
err	A vector of error variances – if NULL then error = 1 - loading 2
short	short=TRUE: Just give the test correlations, short=FALSE, report observed test scores as well as the implied pattern matrix
categorical	continuous or categorical (discrete) variables.
low	values less than low are forced to low
high	values greater than high are forced to high
cuts	If specified, and categorical = TRUE, will cut the resulting continuous output at the value of cuts

Details

When constructing examples for reliability analysis, it is convenient to simulate congeneric data structures. These are the most simple of item structures, having just one factor. Mainly used for a discussion of reliability theory as well as factor score estimates.

The implied covariance matrix is just pattern %*% t(pattern).

Value

model	The implied population correlation matrix if N=NULL or short=FALSE, otherwise the sample correlation matrix
pattern	The pattern matrix implied by the loadings and error variances
r	The sample correlation matrix for long output
observed	a matrix of test scores for n tests
latent	The latent trait and error scores

Author(s)

William Revelle

References

Revelle, W. (in prep) An introduction to psychometric theory with applications in R. To be published by Springer. (working draft available at http://personality-project.org/r/book/

See Also

item.sim for other simulations, fa for an example of factor scores, irt.fa and polychoric for the treatment of item data with discrete values.

Examples

```
test <- sim.congeneric(c(.9,.8,.7,.6))   #just the population matrix
test <- sim.congeneric(c(.9,.8,.7,.6),N=100)   # a sample correlation matrix
test <- sim.congeneric(short=FALSE, N=100)
round(cor(test$observed),2) # show  a congeneric correlation matrix
f1=fa(test$observed,scores=TRUE)
round(cor(f1$scores,test$latent),2)
     #factor score estimates are correlated with but not equal to the factor scores
set.seed(42)
#500 responses to 4 discrete items
items <- sim.congeneric(N=500,short=FALSE,low=-2,high=2,categorical=TRUE)
d4 <- irt.fa(items$observed)  #item response analysis of congeneric measures
```

---

sim.hierarchical	*Create a population or sample correlation matrix, perhaps with hierarchical structure.*

Description

Create a population orthogonal or hierarchical correlation matrix from a set of factor loadings and factor intercorrelations. Samples of size n may be then be drawn from this population. Return either the sample data, sample correlations, or population correlations. This is used to create sample data sets for instruction and demonstration.

Usage

```
sim.hierarchical(gload=NULL, fload=NULL, n = 0, raw = FALSE,mu = NULL)
make.hierarchical(gload=NULL, fload=NULL, n = 0, raw = FALSE)  #deprecated
```

Arguments

`gload`	Loadings of group factors on a general factor
`fload`	Loadings of items on the group factors
`n`	Number of subjects to generate: N=0 => population values
`raw`	raw=TRUE, report the raw data, raw=FALSE, report the sample correlation matrix.
`mu`	means for the individual variables

Details

Many personality and cognitive tests have a hierarchical factor structure. For demonstration purposes, it is useful to be able to create such matrices, either with population values, or sample values.

Given a matrix of item factor loadings (fload) and of loadings of these factors on a general factor (gload), we create a population correlation matrix by using the general factor law (R = F' theta F where theta = g'g).

To create sample values, we use the `mvrnorm` function from MASS.

The default is to return population correlation matrices. Sample correlation matrices are generated if n >0. Raw data are returned if raw = TRUE.

The default values for gload and fload create a data matrix discussed by Jensen and Weng, 1994.

Although written to create hierarchical structures, if the gload matrix is all 0, then a non-hierarchical structure will be generated.

Value

a matrix of correlations or a data matrix

Author(s)

William Revelle

References

http://personality-project.org/r/r.omega.html
Jensen, A.R., Weng, L.J. (1994) What is a Good g? Intelligence, 18, 231-258.

See Also

`omega`, `schmid`, `ICLUST`, `VSS` for ways of analyzing these data. Also see `sim.structure` to simulate a variety of structural models (e.g., multiple correlated factor models). The simulation uses the `mvrnorm` function from the MASS package.

Examples

```
gload <-  gload<-matrix(c(.9,.8,.7),nrow=3)    # a higher order factor matrix
fload <-matrix(c(                    #a lower order (oblique) factor matrix
           .8,0,0,
           .7,0,.0,
           .6,0,.0,
            0,.7,.0,
            0,.6,.0,
            0,.5,0,
            0,0,.6,
            0,0,.5,
            0,0,.4),   ncol=3,byrow=TRUE)

jensen <- sim.hierarchical(gload,fload)    #the test set used by omega
round(jensen,2)

#simulate a non-hierarchical structure
fload <- matrix(c(c(c(.9,.8,.7,.6),rep(0,20)),c(c(.9,.8,.7,.6),rep(0,20)),
    c(c(.9,.8,.7,.6),rep(0,20)),c(c(c(.9,.8,.7,.6),rep(0,20)),c(.9,.8,.7,.6))),ncol=5)
gload <- matrix(rep(0,5))
five.factor <- sim.hierarchical(gload,fload,500,TRUE) #create sample data set
#do it again with a hierachical structure
gload <- matrix(rep(.7,5)  )
five.factor.g <- sim.hierarchical(gload,fload,500,TRUE) #create sample data set
#compare these two with omega
#not run
#om.5 <- omega(five.factor$observed,5)
#om.5g <- omega(five.factor.g$observed,5)
```

---

sim.item *Generate simulated data structures for circumplex, spherical, or simple structure*

Description

Rotations of factor analysis and principal components analysis solutions typically try to represent correlation matrices as simple structured. An alternative structure, appealing to some, is a circumplex structure where the variables are uniformly spaced on the perimeter of a circle in a two dimensional space. Generating simple structure and circumplex data is straightforward, and is useful for exploring alternative solutions to affect and personality structure. A generalization to 3 dimensional (spherical) data is straightforward.

Usage

```
sim.item(nvar = 72, nsub = 500, circum = FALSE, xloading = 0.6, yloading = 0.6,
 gloading = 0, xbias = 0, ybias = 0, categorical = FALSE, low = -3, high = 3,
 truncate = FALSE, cutpoint = 0)
sim.circ(nvar = 72, nsub = 500, circum = TRUE, xloading = 0.6, yloading = 0.6,
```

```
  gloading = 0, xbias = 0, ybias = 0, categorical = FALSE, low = -3, high = 3,
  truncate = FALSE, cutpoint = 0)
sim.dichot(nvar = 72, nsub = 500, circum = FALSE, xloading = 0.6, yloading = 0.6,
    gloading = 0, xbias = 0, ybias = 0, low = 0, high = 0)
item.dichot(nvar = 72, nsub = 500, circum = FALSE, xloading = 0.6, yloading = 0.6,
  gloading = 0, xbias = 0, ybias = 0, low = 0, high = 0)
sim.spherical(simple=FALSE, nx=7,ny=12 ,nsub = 500,  xloading =.55, yloading = .55,
   zloading=.55, gloading=0, xbias=0,  ybias = 0, zbias=0,categorical=FALSE,
   low=-3,high=3,truncate=FALSE,cutpoint=0)
con2cat(old,cuts=c(0,1,2,3),where)
```

Arguments

nvar	Number of variables to simulate
nsub	Number of subjects to simulate
circum	circum=TRUE is circumplex structure, FALSE is simple structure
simple	simple structure or spherical structure in sim.spherical
xloading	the average loading on the first dimension
yloading	Average loading on the second dimension
zloading	the average loading on the third dimension in sim.spherical
gloading	Average loading on a general factor (default=0)
xbias	To introduce skew, how far off center is the first dimension
ybias	To introduce skew on the second dimension
zbias	To introduce skew on the third dimension – if using sim.spherical
categorical	continuous or categorical variables.
low	values less than low are forced to low (or 0 in item.dichot)
high	values greater than high are forced to high (or 1 in item.dichot)
truncate	Change all values less than cutpoint to cutpoint.
cutpoint	What is the cutpoint
nx	number of variables for the first factor in sim.spherical
ny	number of variables for the second and third factors in sim.spherical
old	a matrix or data frame
cuts	Values of old to be used as cut points when converting continuous values to categorical values
where	Which columns of old should be converted to categorical variables. If missing, then all columns are converted.

Details

This simulation was originally developed to compare the effect of skew on the measurement of affect (see Rafaeli and Revelle, 2005). It has been extended to allow for a general simulation of affect or personality items with either a simple structure or a circumplex structure. Items can be continuous normally distributed, or broken down into n categories (e.g, -2, -1, 0, 1, 2). Items can be distorted by limiting them to these ranges, even though the items have a mean of (e.g., 1).

The addition of item.dichot allows for testing structures with dichotomous items of different difficulty (endorsement) levels. Two factor data with either simple structure or circumplex structure are generated for two sets of items, one giving a score of 1 for all items greater than the low (easy) value, one giving a 1 for all items greater than the high (hard) value. The default values for low and high are 0. That is, all items are assumed to have a 50 percent endorsement rate. To examine the effect of item difficulty, low could be -1, high 1. This will lead to item endorsements of .84 for the easy and .16 for the hard. Within each set of difficulties, the first 1/4 are assigned to the first factor factor, the second to the second factor, the third to the first factor (but with negative loadings) and the fourth to the second factor (but with negative loadings).

It is useful to compare the results of sim.item with sim.hierarchical. sim.item will produce a general factor that runs through all the items as well as two orthogonal factors. This produces a data set that is hard to represent with standard rotation techniques. Extracting 3 factors without rotation and then rotating the 2nd and 3rd factors reproduces the correct solution. But simple oblique rotation of 3 factors, or an `omega` analysis do not capture the underlying structure. See the last example.

Yet another structure that might be appealing is fully complex data in three dimensions. That is, rather than having items representing the circumference of a circle, items can be structured to represent equally spaced three dimensional points on a sphere. `sim.spherical` produces such data.

Value

A data matrix of (nsub) subjects by (nvar) variables.

Author(s)

William Revelle

References

Variations of a routine used in Rafaeli and Revelle, 2006; Rafaeli, E. & Revelle, W. (2006). A premature consensus: Are happiness and sadness truly opposite affects? Motivation and Emotion. http://personality-project.org/revelle/publications/rafaeli.revelle.06.pdf

Acton, G. S. and Revelle, W. (2004) Evaluation of Ten Psychometric Criteria for Circumplex Structure. Methods of Psychological Research Online, Vol. 9, No. 1 (formerly (http://www.dgps.de/fachgruppen/methoden/mpr-online/issue22/mpr110_10.pdf) also at http://personality-project.org/revelle/publications/acton.revelle.mpr110_10.pdf

See Also

See Also the implementation in this to generate numerous simulations. `simulation.circ`, `circ.tests` as well as other simulations (`sim.structural` `sim.hierarchical`)

Examples

```
round(cor(circ.sim(nvar=8,nsub=200)),2)
plot(fa(circ.sim(16,500),2)$loadings,main="Circumplex Structure") #circumplex structure
#
```

```
#
plot(fa(item.sim(16,500),2)$loadings,main="Simple Structure") #simple structure
#
cluster.plot(fa(item.dichot(16,low=0,high=1),2))

 set.seed(42)

 data <- mnormt::rmnorm(1000, c(0, 0), matrix(c(1, .5, .5, 1), 2, 2)) #continuous data
 new <- con2cat(data,c(-1.5,-.5,.5,1.5))  #discreet data
 polychoric(new)
#not run
#x12 <- sim.item(12,gloading=.6)
#f3 <- fa(x12,3,rotate="none")
#f3  #observe the general factor
#oblimin(f3$loadings[,2:3])  #show the 2nd and 3 factors.
#f3 <- fa(x12,3)   #now do it with oblimin rotation
#f3  # not what one naively expect.
```

---

sim.multilevel *Simulate multilevel data with specified within group and between group correlations*

Description

Multilevel data occur when observations are nested within groups. This can produce correlational structures that are sometimes difficult to understand. This simulation allows for demonstrations that correlations within groups do not imply, nor are implied by, correlations between group means. The correlations of aggregated data is sometimes called an 'ecological correlation'. That group level and individual level correlations are independent makes such inferences problematic.

Usage

```
sim.multilevel(nvar = 9, ngroups = 4, ncases = 16, rwg, rbg, eta)
```

Arguments

nvar	Number of variables to simulate
ngroups	The number of groups to simulate
ncases	The number of simulated cases
rwg	The within group correlational structure
rbg	The between group correlational structure
eta	The correlation of the data with the within data

Details

The basic concepts of the independence of within group and between group correlations is discussed very clearly by Pedhazur (1997) as well as by Bliese (2009). This function merely simulates pooled correlations (mixtures of between group and within group correlations) to allow for a better understanding of the problems inherent in multi-level modeling.

Data (wg) are created with a particular within group structure (rwg). Independent data (bg) are also created with a between group structure (rbg). Note that although there are ncases rows to this data matrix, there are only ngroups independent cases. That is, every ngroups case is a repeat. The resulting data frame (xy) is a weighted sum of the wg and bg. This is the inverse procedure for estimating estimating rwg and rbg from an observed rxy which is done by the `statsBy` function.

Value

`wg`	A matrix (ncases * nvar) of simulated within group scores
`bg`	A matrix (ncases * nvar) of simulated between group scores
`xy`	A matrix ncases * (nvar +1) of pooled data

Author(s)

William Revelle

References

P. D. Bliese. Multilevel modeling in R (2.3) a brief introduction to R, the multilevel package and the nlme package, 2009.

Pedhazur, E.J. (1997) Multiple regression in behavioral research: explanation and prediction. Harcourt Brace.

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at http://personality-project.org/r/book/

See Also

`statsBy` for the decomposition of multi level data and `withinBetween` for an example data set.

Examples

```
#get some parameters to simulate
data(withinBetween)
wb.stats <- statsBy(withinBetween,"Group")
rwg <- wb.stats$rwg
rbg <- wb.stats$rbg
eta <- rep(.5,9)

#simulate them.  Try this again to see how it changes
XY <- sim.multilevel(ncases=100,ngroups=10,rwg=rwg,rbg=rbg,eta=eta)
lowerCor(XY$wg)  #based upon 89 df
lowerCor(XY$bg)  #based upon 9 df   --
```

`sim.structure` *Create correlation matrices or data matrices with a particular measurement and structural model*

Description

Structural Equation Models decompose correlation or correlation matrices into a measurement (factor) model and a structural (regression) model. sim.structural creates data sets with known measurement and structural properties. Population or sample correlation matrices with known properties are generated. Optionally raw data are produced.

It is also possible to specify a measurement model for a set of x variables separately from a set of y variables. They are then combined into one model with the correlation structure between the two sets.

Finally, the general case is given a population correlation matrix, generate data that will reproduce (with sampling variability) that correlation matrix. `sim.correlation`.

Usage

```
sim.structure(fx=NULL,Phi=NULL, fy=NULL, f=NULL, n=0, uniq=NULL, raw=TRUE,
  items = FALSE, low=-2,high=2,d=NULL,cat=5, mu=0)
sim.structural(fx=NULL, Phi=NULL, fy=NULL, f=NULL, n=0, uniq=NULL, raw=TRUE,
      items = FALSE, low=-2,high=2,d=NULL,cat=5, mu=0)  #deprecated
sim.correlation(R,n=1000,data=FALSE)
```

Arguments

`fx`	The measurement model for x
`Phi`	The structure matrix of the latent variables
`fy`	The measurement model for y
`f`	The measurement model
`n`	Number of cases to simulate. If n=0, the population matrix is returned.
`uniq`	The uniquenesses if creating a covariance matrix
`raw`	if raw=TRUE, raw data are returned as well for n > 0.
`items`	TRUE if simulating items, FALSE if simulating scales
`low`	Restrict the item difficulties to range from low to high
`high`	Restrict the item difficulties to range from low to high
`d`	A vector of item difficulties, if NULL will range uniformly from low to high
`cat`	Number of categories when creating binary (2) or polytomous items
`mu`	A vector of means, defaults to 0
`R`	The correlation matrix to reproduce
`data`	if TRUE, return the raw data, otherwise return the sample correlation matrix.

Details

Given the measurement model, fx and the structure model Phi, the model is f %*% Phi %*% t(f). Reliability is f %*% t(f). $f\phi f'$ and the reliability for each test is the items communality or just the diag of the model.

If creating a correlation matrix, (uniq=NULL) then the diagonal is set to 1, otherwise the diagonal is diag(model) + uniq and the resulting structure is a covariance matrix.

Given the model, raw data are generated using the mvnorm function.

A special case of a structural model are one factor models such as parallel tests, tau equivalent tests, and congeneric tests. These may be created by letting the structure matrix = 1 and then defining a vector of factor loadings. Alternatively, make.congeneric will do the same.

`sim.correlation` will create data sampled from a specified correlation matrix for a particular sample size. If desired, it will just return the sample correlation matrix. With data=TRUE, it will return the sample data as well.

Value

`model`	The implied population correlation or covariance matrix
`reliability`	The population reliability values
`r`	The sample correlation or covariance matrix
`observed`	If raw=TRUE, a sample data matrix

Author(s)

William Revelle

References

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at `http://personality-project.org/r/book/`

See Also

`make.hierarchical` for another structural model and `make.congeneric` for the one factor case. `structure.list` and `structure.list` for making symbolic structures.

Examples

```
fx <-matrix(c( .9,.8,.6,rep(0,4),.6,.8,-.7),ncol=2)
fy <- matrix(c(.6,.5,.4),ncol=1)
rownames(fx) <- c("V","Q","A","nach","Anx")
rownames(fy)<- c("gpa","Pre","MA")
Phi <-matrix( c(1,0,.7,.0,1,.7,.7,.7,1),ncol=3)
gre.gpa <- sim.structural(fx,Phi,fy)
print(gre.gpa,2)
#correct for attenuation to see structure

round(correct.cor(gre.gpa$model,gre.gpa$reliability),2)
congeneric <- sim.structure(f=c(.9,.8,.7,.6)) # a congeneric model
```

congeneric

---

sim.VSS *create VSS like data*

Description

Simulation is one of most useful techniques in statistics and psychometrics. Here we simulate a correlation matrix with a simple structure composed of a specified number of factors. Each item is assumed to have complexity one. See circ.sim and item.sim for alternative simulations.

Usage

```
sim.VSS(ncases=1000, nvariables=16, nfactors=4, meanloading=.5,dichot=FALSE,cut=0)
```

Arguments

ncases	number of simulated subjects
nvariables	Number of variables
nfactors	Number of factors to generate
meanloading	with a mean loading
dichot	dichot=FALSE give continuous variables, dichot=TRUE gives dichotomous variables
cut	if dichotomous = TRUE, then items with values > cut are assigned 1, otherwise 0.

Value

a ncases x nvariables matrix

Author(s)

William Revelle

See Also

VSS, ICLUST

Examples

```
## Not run:
simulated <- sim.VSS(1000,20,4,.6)
vss <- VSS(simulated,rotate="varimax")
VSS.plot(vss)

## End(Not run)
```

simulation.circ *Simulations of circumplex and simple structure*

Description

Rotations of factor analysis and principal components analysis solutions typically try to represent correlation matrices as simple structured. An alternative structure, appealing to some, is a circumplex structure where the variables are uniformly spaced on the perimeter of a circle in a two dimensional space. Generating these data is straightforward, and is useful for exploring alternative solutions to affect and personality structure.

Usage

```
simulation.circ(samplesize=c(100,200,400,800), numberofvariables=c(16,32,48,72))
circ.sim.plot(x.df)
```

Arguments

samplesize	a vector of sample sizes to simulate
numberofvariables	vector of the number of variables to simulate
x.df	A data frame resulting from simulation.circ

Details

"A common model for representing psychological data is simple structure (Thurstone, 1947). According to one common interpretation, data are simple structured when items or scales have non-zero factor loadings on one and only one factor (Revelle & Rocklin, 1979). Despite the commonplace application of simple structure, some psychological models are defined by a lack of simple structure. Circumplexes (Guttman, 1954) are one kind of model in which simple structure is lacking.

"A number of elementary requirements can be teased out of the idea of circumplex structure. First, circumplex structure implies minimally that variables are interrelated; random noise does not a circumplex make. Second, circumplex structure implies that the domain in question is optimally represented by two and only two dimensions. Third, circumplex structure implies that variables do not group or clump along the two axes, as in simple structure, but rather that there are always interstitial variables between any orthogonal pair of axes (Saucier, 1992). In the ideal case, this quality will be reflected in equal spacing of variables along the circumference of the circle (Gurtman, 1994; Wiggins, Steiger, & Gaelick, 1981). Fourth, circumplex structure implies that variables have a constant radius from the center of the circle, which implies that all variables have equal communality on the two circumplex dimensions (Fisher, 1997; Gurtman, 1994). Fifth, circumplex structure implies that all rotations are equally good representations of the domain (Conte & Plutchik, 1981; Larsen & Diener, 1992)." (Acton and Revelle, 2004)

Acton and Revelle reviewed the effectiveness of 10 tests of circumplex structure and found that four did a particularly good job of discriminating circumplex structure from simple structure, or circumplexes from ellipsoidal structures. Unfortunately, their work was done in Pascal and is not easily available. Here we release R code to do the four most useful tests:

The Gap test of equal spacing

Fisher's test of equality of axes

A test of indifference to Rotation

A test of equal Variance of squared factor loadings across arbitrary rotations.

Included in this set of functions are simple procedure to generate circumplex structured or simple structured data, the four test statistics, and a simple simulation showing the effectiveness of the four procedures.

`circ.sim.plot` compares the four tests for circumplex, ellipsoid and simple structure data as function of the number of variables and the sample size. What one can see from this plot is that although no one test is sufficient to discriminate these alternative structures, the set of four tests does a very good job of doing so. When testing a particular data set for structure, comparing the results of all four tests to the simulated data will give a good indication of the structural properties of the data.

Value

A data.frame with simulation results for circumplex, ellipsoid, and simple structure data sets for each of the four tests.

Note

The simulations default values are for sample sizes of 100, 200, 400, and 800 cases, with 16, 32, 48 and 72 items.

Author(s)

William Revelle

References

Acton, G. S. and Revelle, W. (2004) Evaluation of Ten Psychometric Criteria for Circumplex Structure. Methods of Psychological Research Online, Vol. 9, No. 1 (formerly at http://www.dgps.de/fachgruppen/methoden/mpr-online/issue22/mpr110_10.pdf and now at http://personality-project.org/revelle/publications/acton.revelle.mpr110_10.pdf.

See Also

See also `circ.tests`, `sim.circ`, `sim.structural`, `sim.hierarchical`

Examples

```
#not run
demo <- simulation.circ()
boxplot(demo[3:14])
title("4 tests of Circumplex Structure",sub="Circumplex, Ellipsoid, Simple Structure")
circ.sim.plot(demo[3:14])  #compare these results to real data
```

smc *Find the Squared Multiple Correlation (SMC) of each variable with the remaining variables in a matrix*

Description

The squared multiple correlation of a variable with the remaining variables in a matrix is sometimes used as initial estimates of the communality of a variable.

SMCs are also used when estimating reliability using Guttman's lambda 6 guttman coefficient.

The SMC is just 1 - 1/diag(R.inv) where R.inv is the inverse of R.

Usage

```
smc(R,covar=FALSE)
```

Arguments

R	A correlation matrix or a dataframe. In the latter case, correlations are found.
covar	if covar = TRUE and R is either a covariance matrix or data frame, then return the smc * variance for each item

Value

a vector of squared multiple correlations. Or, if covar=TRUE, a vector of squared multiple correlations * the item variances

If the matrix is not invertible, then a vector of 1s is returned.

In the case of correlation or covariance matrices with some NAs, those variables with NAs are dropped and the SMC for the remaining variables are found. The missing SMCs are then estimated by finding the maximum correlation for that column (with a warning).

Author(s)

William Revelle

See Also

mat.regress, fa

Examples

```
R <- make.hierarchical()
round(smc(R),2)
```

spider *Make "radar" or "spider" plots.*

Description

Radar plots and spider plots are just two of the many ways to show multivariate data. radar plots correlations as vectors ranging in length from 0 (corresponding to r=-1) to 1 (corresponding to an r=1). The vectors are arranged radially around a circle. Spider plots connect the end points of each vector. The plots are most appropriate if the variables are organized in some meaningful manner.

Usage

```
spider(y,x,data,labels=NULL,rescale=FALSE,center=FALSE,connect=TRUE,overlay=FALSE,
    scale=1,ncolors=31,fill=FALSE,main=NULL,...)
radar(x,labels=NULL,center=FALSE,connect=FALSE,scale=1,ncolors=31,fill=FALSE,
     add=FALSE,linetyp="solid", main="Radar Plot",...)
```

Arguments

y	The y variables to plot. Each y is plotted against all the x variables
x	The x variables defining each line. Each y is plotted against all the x variables
data	A correlation matrix from which the x and y variables are selected
labels	Labels (assumed to be colnames of the data matrix) for each x variable
rescale	If TRUE, then rescale the data to have mean 0 and sd = 1. This is used if plotting raw data rather than correlations.
center	if TRUE, then lines originate at the center of the plot, otherwise they start at the mid point.
connect	if TRUE, a spider plot is drawn, if FALSE, just a radar plot
scale	can be used to magnify the plot, to make small values appear larger.
ncolors	if ncolors > 2, then positive correlations are plotted with shades of blue and negative correlations shades of red. This is particularly useful if fill is TRUE. ncolors should be an odd number, so that neutral values are coded as white.
fill	if TRUE, fill the polygons with colors scaled to size of correlation
overlay	If TRUE, plot multiple spiders on one plot, otherwise plot them as separate plots
add	If TRUE, add a new spider diagram to the previous one.
linetyp	see lty in the par options
main	A label or set of labels for the plots
...	Additional parameters can be passed to the underlying graphics call

Details

Displaying multivariate profiles may be done by a series of lines (see, e.g., matplot), by colors (see, e.g., cor.plot, or by radar or spider plots.

To show just one variable as a function of several others, use radar. To make multiple plots, use spider. An additional option when comparing just a few y values is to do overlay plots. Alternatively, set the plotting options to do several on one page.

Value

Either a spider or radar plot

Author(s)

William Revelle

See Also

cor.plot

Examples

```
op <- par(mfrow=c(3,2))
spider(y=1,x=2:9,data=Thurstone,connect=FALSE) #a radar plot
spider(y=1,x=2:9,data=Thurstone) #same plot as a spider plot
 spider(y=1:3,x=4:9,data=Thurstone,overlay=TRUE)
 #make a somewhat oversized plot
spider(y=26:28,x=1:25,data=cor(bfi,use="pairwise"),fill=TRUE,scale=2)
par(op)
```

---

splitHalf *Alternative estimates of test reliabiity*

Description

Eight alternative estimates of test reliability include the six discussed by Guttman (1945), four discussed by ten Berge and Zergers (1978) ($\mu_0 \dots \mu_3$) as well as β (the worst split half, Revelle, 1979), the glb (greatest lowest bound) discussed by Bentler and Woodward (1980), and ω_h and ω_t (McDonald, 1999; Zinbarg et al., 2005). Greatest and lowest split-half values are found by brute force or sampling.

Usage

```
splitHalf(r,raw=FALSE,brute=FALSE,n.sample=10000,covar=FALSE,check.keys=TRUE,
           key=NULL,use="pairwise")
guttman(r,key=NULL)
tenberge(r)
glb(r,key=NULL)
glb.fa(r,key=NULL)
```

Arguments

r	A correlation or covariance matrix or raw data matrix.
raw	return a vector of split half reliabilities
brute	Use brute force to try all combinations of n take n/2.
n.sample	if brute is false, how many samples of split halves should be tried?
covar	Should the covariances or correlations be used for reliability calculations
check.keys	If TRUE, any item with a negative loading on the first factor will be flipped in sign
key	a vector of -1, 0, 1 to select or reverse key items. See if the key vector is less than the number of variables, then item numbers to be reverse can be specified.
use	Should we find the correlations using "pairwise" or "complete" (see ?cor)

Details

Surprisingly, more than a century after Spearman (1904) introduced the concept of reliability to psychologists, there are still multiple approaches for measuring it. Although very popular, Cronbach's α (1951) underestimates the reliability of a test and over estimates the first factor saturation. Using splitHalf for tests with 16 or fewer items, all possible splits may be found fairly easily. For tests with 17 or more items, n.sample splits are randomly found. Thus, for 16 or fewer items, the upper and lower bounds are precise. For 17 or more items, they are close but will probably slightly underestimate the highest and overestimate the lowest reliabilities.

The guttman function includes the six estimates discussed by Guttman (1945), four of ten Berge and Zergers (1978), as well as Revelle's β (1979) using splitHalf. The companion function, omega calculates omega hierarchical (ω_h) and omega total (ω_t).

Guttman's first estimate λ_1 assumes that all the variance of an item is error:

$$\lambda_1 = 1 - \frac{tr(\vec{V}_x)}{V_x} = \frac{V_x - tr(\vec{V}_x)}{V_x}$$

This is a clear underestimate.

The second bound, λ_2, replaces the diagonal with a function of the square root of the sums of squares of the off diagonal elements. Let $C_2 = \vec{1}(\vec{V} - diag(\vec{V})^2\vec{1}'$, then

$$\lambda_2 = \lambda_1 + \frac{\sqrt{\frac{n}{n-1}C_2}}{V_x} = \frac{V_x - tr(\vec{V}_x) + \sqrt{\frac{n}{n-1}C_2}}{V_x}$$

Effectively, this is replacing the diagonal with n * the square root of the average squared off diagonal element.

Guttman's 3rd lower bound, λ_3, also modifies λ_1 and estimates the true variance of each item as the average covariance between items and is, of course, the same as Cronbach's α.

$$\lambda_3 = \lambda_1 + \frac{\frac{V_X - tr(\vec{V}_X)}{n(n-1)}}{V_X} = \frac{n\lambda_1}{n-1} = \frac{n}{n-1}\Big(1 - \frac{tr(\vec{V})_x}{V_x}\Big) = \frac{n}{n-1}\frac{V_x - tr(\vec{V}_x)}{V_x} = \alpha$$

This is just replacing the diagonal elements with the average off diagonal elements. $\lambda_2 \geq \lambda_3$ with $\lambda_2 > \lambda_3$ if the covariances are not identical.

λ_3 and λ_2 are both corrections to λ_1 and this correction may be generalized as an infinite set of successive improvements. (Ten Berge and Zegers, 1978)

$$\mu_r = \frac{1}{V_x}\big(p_o + (p_1 + (p_2 + \dots (p_{r-1} + (p_r)^{1/2})^{1/2} \dots)^{1/2})^{1/2}\big), r = 0, 1, 2, \dots$$

where

$$p_h = \sum_{i \neq j} \sigma_{ij}^{2h}, h = 0, 1, 2, \dots r-1$$

and

$$p_h = \frac{n}{n-1}\sigma_{ij}^{2h}, h = r$$

tenberge and Zegers (1978). Clearly $\mu_0 = \lambda_3 = \alpha$ and $\mu_1 = \lambda_2$. $\mu_r \geq \mu_{r-1} \geq \dots \mu_1 \geq \mu_0$, although the series does not improve much after the first two steps.

Guttman's fourth lower bound, λ_4 was originally proposed as any spit half reliability but has been interpreted as the greatest split half reliability. If $\vec{X}$ is split into two parts, $\vec{X}_a$ and $\vec{X}_b$, with correlation r_{ab} then

$$\lambda_4 = 2\Big(1 - \frac{V_{X_a} + V_{X_b}}{V_X}\Big) = \frac{4r_{ab}}{V_x} = \frac{4r_{ab}}{V_{X_a} + V_{X_b} + 2r_{ab}V_{X_a}V_{X_b}}$$

which is just the normal split half reliability, but in this case, of the most similar splits. For 16 or fewer items, this is found by trying all possible splits. For 17 or more items, this is estimated by taking n.sample random splits.

λ_5, Guttman's fifth lower bound, replaces the diagonal values with twice the square root of the maximum (across items) of the sums of squared interitem covariances

$$\lambda_5 = \lambda_1 + \frac{2\sqrt{\bar{C}_2}}{V_X}.$$

Although superior to λ_1, λ_5 underestimates the correction to the diagonal. A better estimate would be analogous to the correction used in λ_3:

$$\lambda_{5+} = \lambda_1 + \frac{n}{n-1}\frac{2\sqrt{\bar{C}_2}}{V_X}.$$

λ_6,Guttman's final bound considers the amount of variance in each item that can be accounted for the linear regression of all of the other items (the squared multiple correlation or smc), or more precisely, the variance of the errors, e_j^2, and is

$$\lambda_6 = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum(1 - r_{smc}^2)}{V_x}$$

.

The smc is found from all the items. A modification to Guttman λ_6, λ_6* reported by the `score.items` function is to find the smc from the entire pool of items given, not just the items on the selected scale.

Guttman's λ_4 is the greatest split half reliability. Although originally found here by combining the output from three different approaches,this has now been replaced by using `splitHalf` to find the maximum value by brute force (for 16 or fewer items) or by taking a substantial number of random splits.

The algorithms that had been tried before included:

a) Do an ICLUST of the reversed correlation matrix. ICLUST normally forms the most distinct clusters. By reversing the correlations, it will tend to find the most related clusters. Truly a weird approach but tends to work.

b) Alternatively, a kmeans clustering of the correlations (with the diagonal replaced with 0 to make pseudo distances) can produce 2 similar clusters.

c) Clusters identified by assigning items to two clusters based upon their order on the first principal factor. (Highest to cluster 1, next 2 to cluster 2, etc.)

These three procedures will produce keys vectors for assigning items to the two splits. The maximum split half reliability is found by taking the maximum of these three approaches. This is not elegant but is fast.

The brute force and the sampling procedures seem to provide more stable and larger estimates.

Yet another procedure, implemented in `splitHalf` is actually form all possible (for n items <= 16) or sample 10,000 (or more) split halfs corrected for test length. This function returns the best and worst splits as item keys that can be used for scoring purposes, if desired.

There are three greatest lower bound functions. One, glb finds the greatest split half reliability, λ_4. This considers the test as set of items and examines how best to partition the items into splits. The other two, `glb.fa` and `glb.algebraic`, are alternative ways of weighting the diagonal of the matrix.

`glb.fa` estimates the communalities of the variables from a factor model where the number of factors is the number with positive eigen values. Then reliability is found by

$$glb = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum (1 - h^2)}{V_x}$$

This estimate will differ slightly from that found by `glb.algebraic`, written by Andreas Moeltner which uses calls to csdp in the Rcsdp package. His algorithm, which more closely matches the description of the glb by Jackson and Woodhouse, seems to have a positive bias (i.e., will over estimate the reliability of some items; they are said to be = 1) for small sample sizes. More exploration of these two algorithms is underway.

Compared to `glb.algebraic`, `glb.fa` seems to have less (positive) bias for smallish sample sizes (n < 500) but larger for large (> 1000) sample sizes. This interacts with the number of variables so that equal bias sample size differs as a function of the number of variables. The differences are, however small. As samples sizes grow, `glb.algebraic` seems to converge on the population value while glb.fa has a positive bias.

Value

`beta`	The worst split half reliability. This is an estimate of the general factor saturation.
`tenberge$mu1`	tenBerge mu 1 is functionally alpha
`tenberge$mu2`	one of the sequence of estimates mu1 ... mu3
`glb`	glb found from factor analysis
`keys`	scoring keys from each of the alternative methods of forming best splits

Author(s)

William Revelle

References

Cronbach, L.J. (1951) Coefficient alpha and the internal strucuture of tests. Psychometrika, 16, 297-334.

Guttman, L. (1945). A basis for analyzing test-retest reliability. Psychometrika, 10 (4), 255-282.

Revelle, W. (1979). Hierarchical cluster-analysis and the internal structure of tests. Multivariate Behavioral Research, 14 (1), 57-74.

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 2009.

Ten Berge, J. M. F., & Zegers, F. E. (1978). A series of lower bounds to the reliability of a test. Psychometrika, 43 (4), 575-579.

Zinbarg, R. E., Revelle, W., Yovel, I., & Li, W. (2005). Cronbach's α , Revelle's β , and McDonald's ω_h): Their relations with each other and two alternative conceptualizations of reliability. Psychometrika, 70 (1), 123-133.

See Also

alpha, omega, ICLUST, glb.algebraic

Examples

```
data(attitude)
splitHalf(attitude)
splitHalf(attitude,covar=TRUE) #do it on the covariances
glb(attitude)
glb.fa(attitude)
if(require(Rcsdp)) {glb.algebraic(cor(attitude)) }
guttman(attitude)

#to show the histogram of all possible splits for the ability test
#sp <- splitHalf(ability,raw=TRUE)  #this saves the results
#hist(sp$raw,breaks=101,ylab="SplitHalf reliability",main="SplitHalf
#    reliabilities of a test with 16 ability items")
sp <- splitHalf(bfi[1:10],key=c(1,9,10))
```

---

statsBy *Find statistics (including correlations) within and between groups for basic multilevel analyses*

Description

When examining data at two levels (e.g., the individual and by some set of grouping variables), it is useful to find basic descriptive statistics (means, sds, ns per group, within group correlations) as well as between group statistics (over all descriptive statistics, and overall between group correlations). Of particular use is the ability to decompose a matrix of correlations at the individual level into correlations within group and correlations between groups.

Usage

```
statsBy(data, group, cors = FALSE, cor="cor", method="pearson", use="pairwise",
poly=FALSE, na.rm=TRUE)
statsBy.boot(data,group,ntrials=10,cors=FALSE,replace=TRUE,method="pearson")
statsBy.boot.summary(res.list,var="ICC2")
faBy(stats, nfactors = 1, rotate = "oblimin", fm = "minres", free = TRUE, all=FALSE,
   min.n = 12,quant=.1, ...)
```

Arguments

data	A matrix or dataframe with rows for subjects, columns for variables. One of these columns should be the values of a grouping variable.
group	The names or numbers of the variable in data to use as the grouping variables.
cors	Should the results include the correlation matrix within each group? Default is FALSE.
cor	Type of correlation/covariance to find within groups and between groups. The default is Pearson correlation. To find within and between covariances, set cor="cov". Although polychoric, tetrachoric, and mixed correlations can be found within groups, this does not make sense for the between groups or the pooled within groups. In this case, correlations for each group will be as specified, but the between groups and pooled within will be Pearson. See the discussion below.
method	What kind of correlations should be found (default is Pearson product moment)
use	How to treat missing data. use="pairwise" is the default
poly	Find polychoric correlations within groups if requested.
na.rm	Should missing values be deleted (na.rm=TRUE) or should we assume the data clean?
ntrials	The number of trials to run when bootstrapping statistics
replace	Should the bootstrap be done by permuting the data (replace=FALSE) or sampling with replacement (replace=TRUE)
res.list	The results from statsBy.boot may be summarized using boot.stats
var	Name of the variable to be summarized from statsBy.boot
stats	The output of statsBy
nfactors	The number of factors to extract in each subgroup
rotate	The factor rotation/transformation
fm	The factor method (see fa for details)

`free`	Allow the factor solution to be freely estimated for each individual (see note).
`all`	Report individual factor analyses for each group as well as the summary table
`min.n`	The minimum number of within subject cases before we factor analyze it.
`quant`	Show the upper and lower quant quantile of the factor loadings in faBy
`...`	Other parameters to pass to the fa function

Details

Multilevel data are endemic in psychological research. In multilevel data, observations are taken on subjects who are nested within some higher level grouping variable. The data might be experimental (participants are nested within experimental conditions) or observational (students are nested within classrooms, students are nested within college majors.) To analyze this type of data, one uses random effects models or mixed effect models, or more generally, multilevel models. There are at least two very powerful packages (nlme and multilevel) which allow for complex analysis of hierarchical (multilevel) data structures. `statsBy` is a much simpler function to give some of the basic descriptive statistics for two level models. It is meant to supplement true multilevel modeling.

For a group variable (group) for a data.frame or matrix (data), basic descriptive statistics (mean, sd, n) as well as within group correlations (cors=TRUE) are found for each group.

The amount of variance associated with the grouping variable compared to the total variance is the type 1 IntraClass Correlation (ICC1): $ICC1 = (MSb - MSw)/(MSb + MSw*(npr-1))$ where npr is the average number of cases within each group.

The reliability of the group differences may be found by the ICC2 which reflects how different the means are with respect to the within group variability. $ICC2 = (MSb - MSw)/MSb$. Because the mean square between is sensitive to sample size, this estimate will also reflect sample size.

Perhaps the most useful part of `statsBy` is that it decomposes the observed correlations between variables into two parts: the within group and the between group correlation. This follows the decomposition of an observed correlation into the pooled correlation within groups (rwg) and the weighted correlation of the means between groups discussed by Pedazur (1997) and by Bliese in the multilevel package.

$r_{xy} = eta_{x_{wg}} * eta_{y_{wg}} * r_{xy_{wg}} + eta_{x_{bg}} * eta_{y_{bg}} * r_{xy_{bg}}$

where r_{xy} is the normal correlation which may be decomposed into a within group and between group correlations $r_{xy_{wg}}$ and $r_{xy_{bg}}$ and eta is the correlation of the data with the within group values, or the group means.

It is important to realize that the within group and between group correlations are independent of each other. That is to say, inferring from the 'ecological correlation' (between groups) to the lower level (within group) correlation is inappropriate. However, these between group correlations are still very meaningful, if inferences are made at the higher level.

There are actually two ways of finding the within group correlations pooled across groups. We can find the correlations within every group, weight these by the sample size and then report this pooled value (pooled). This is found if the cors option is set to TRUE. It is logically equivalent to doing a sample size weighted meta-analytic correlation. The other way, rwg, considers the covariances, variances, and thus correlations when each subject's scores are given as deviation score from the group mean.

If finding tetrachoric, polychoric, or mixed correlations, these two estimates will differ, for the pooled value is the weighted polychoric correlation, but the rwg is the Pearson correlation.

Confidence values and significance of $r_{xy_{wg}}$, pwg, reflect the pooled number of cases within groups, while $r_{xy_{bg}}$, pbg, the number of groups. These are not corrected for multiple comparisons.

withinBetween is an example data set of the mixture of within and between group correlations. sim.multilevel will generate simulated data with a multilevel structure.

The statsBy.boot function will randomize the grouping variable ntrials times and find the statsBy output. This can take a long time and will produce a great deal of output. This output can then be summarized for relevant variables using the statsBy.boot.summary function specifying the variable of interest. These two functions are useful in order to find if the mere act of grouping leads to large between group correlations.

Consider the case of the relationship between various tests of ability when the data are grouped by level of education (statsBy(sat.act,"education")) or when affect data are analyzed within and between an affect manipulation (statsBy(flat,group="Film")). Note in this latter example, that because subjects were randomly assigned to Film condition for the pretest, that the pretest ICC1s cluster around 0.

faBy uses the output of statsBy to perform a factor analysis on the correlation matrix within each group. If the free parameter is FALSE, then each solution is rotated towards the group solution (as much as possible). The output is a list of each factor solution, as well as a summary matrix of loadings and interfactor correlations for all groups.

Value

means	The means for each group for each variable.
sd	The standard deviations for each group for each variable.
n	The number of cases for each group and for each variable.
ICC1	The intraclass correlation reflects the amount of total variance associated with the grouping variable.
ICC2	The intraclass correlation (2) reflecting how much the groups means differ.
F	The F from a one-way anova of group means.
rwg	The pooled within group correlations.
rbg	The sample size weighted between group correlations.
etawg	The correlation of the data with the within group values.
etabg	The correlation of the data with the group means.
pbg	The probability of the between group correlation
pwg	The probability of the within group correlation
r	In the case that we want the correlations in each group, r is a list of the within group correlations for every group. Set cors=TRUE
within	is just another way of displaying these correlations. within is a matrix which reports the lower off diagonal correlations as one row for each group.
pooled	The sample size weighted correlations. This is just within weighted by the sample sizes. The cors option must be set to TRUE to get this. See the note.

Note

If finding polychoric correlations, the two estimates of the pooled within group correlations will differ, for the pooled value is the weighted polychoric correlation, but the rwg is the Pearson correlation.

The statsBy.boot function will sometimes fail if sampling with replacement because if the group sizes differ drastically, some groups will be empty. In this case, sample without replacement.

The statsBy.boot function can take a long time. (As I am writing this, I am running 1000 replications of a problem with 64,000 cases and 84 groups. It is taking about 3 seconds per replication on a MacBook Pro.)

The faBy function takes the output of statsBy (with the cors=TRUE option) and then factors each individual subject. By default, the solutions are organized so that the factors "match" the group solution in terms of their order. It is also possible to attempt to force the solutions to match by order and also by using the TargetQ rotation function. (free=FALSE)

Author(s)

William Revelle

References

Pedhazur, E.J. (1997) Multiple regression in behavioral research: explanation and prediction. Harcourt Brace.

See Also

describeBy and the functions within the multilevel package.

Examples

```
#Taken from Pedhazur, 1997
pedhazur <- structure(list(Group = c(1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L,
2L), X = c(5L, 2L, 4L, 6L, 3L, 8L, 5L, 7L, 9L, 6L), Y = 1:10), .Names = c("Group",
"X", "Y"), class = "data.frame", row.names = c(NA, -10L))
pedhazur
ped.stats <- statsBy(pedhazur,"Group")
ped.stats


#Now do this for the sat.act data set
sat.stats <- statsBy(sat.act,c("education","gender"),cor=TRUE)   #group by two grouping variables
print(sat.stats,short=FALSE)
lowerMat(sat.stats$pbg)  #get the probability values

#show means by groups
round(sat.stats$mean)

#Do separate factor analyses for each group
#faBy(sat.stats,1)
```

`structure.diagram` *Draw a structural equation model specified by two measurement models and a structural model*

Description

Graphic presentations of structural equation models are a very useful way to conceptualize sem and confirmatory factor models. Given a measurement model on x (xmodel) and on y (ymodel) as well as a path model connecting x and y (phi), draw the graph. If ymodel is not specified, just draw the measurement model (xmodel + phi). If the Rx or Ry matrices are specified, show the correlations between the x variables, or y variables.

Perhaps even more usefully, the function returns a model appropriate for running directly in the *sem package* written by John Fox. For this option to work directly, it is necessary to specfy that errrors=TRUE.

Input can be specified as matrices or the output from `fa`, factanal, or a rotation package such as *GPArotation*.

For symbolic graphs, the input matrices can be character strings or mixtures of character strings and numeric vectors.

As an option, for those without Rgraphviz installed, structure.sem will just create the sem model and skip the graph. (This functionality is now included in structure.diagram.)

structure.diagram will draw the diagram without using Rgraphviz and is probably the preferred option. structure.graph will be removed eventually.

lavaan.diagram will draw either cfa or sem results from the lavaan package (> .4.0)

Usage

```
structure.diagram(fx, Phi=NULL,fy=NULL,labels=NULL,cut=.3,errors=FALSE,simple=TRUE,
   regression=FALSE,lr=TRUE,Rx=NULL,Ry=NULL,digits=1,e.size=.1,
    main="Structural model", ...)
structure.graph(fx,  Phi = NULL,fy = NULL, out.file = NULL, labels = NULL, cut = 0.3,
   errors=TRUE, simple=TRUE,regression=FALSE, size = c(8, 6),
    node.font = c("Helvetica", 14), edge.font = c("Helvetica", 10),
    rank.direction = c("RL", "TB", "LR", "BT"), digits = 1,
     title = "Structural model", ...)
structure.sem(fx,  Phi = NULL, fy = NULL,out.file = NULL, labels = NULL,
     cut = 0.3, errors=TRUE, simple=TRUE,regression=FALSE)
lavaan.diagram(fit,title,...)
```

Arguments

`fx`	a factor model on the x variables.
`Phi`	A matrix of directed relationships. Lower diagonal values are drawn. If the upper diagonal values match the lower diagonal, two headed arrows are drawn. For a single, directed path, just the value may be specified.

`fy`	a factor model on the y variables (can be empty)
`Rx`	The correlation matrix among the x variables
`Ry`	The correlation matrix among the y variables
`out.file`	name a file to send dot language instructions.
`labels`	variable labels if not specified as colnames for the matrices
`cut`	Draw paths for values > cut
`fit`	The output from a lavaan cfa or sem
`errors`	draw an error term for observerd variables
`simple`	Just draw one path per x or y variable
`regression`	Draw a regression diagram (observed variables cause Y)
`lr`	Direction of diagram is from left to right (lr=TRUE, default) or from bottom to top (lr=FALSE)
`e.size`	size of the ellipses in structure.diagram
`main`	main title of diagram
`size`	page size of graphic
`node.font`	font type for graph
`edge.font`	font type for graph
`rank.direction`	Which direction should the graph be oriented
`digits`	Number of digits to draw
`title`	Title of graphic
`...`	other options to pass to Rgraphviz

Details

The recommended function is structure.diagram which does not use Rgraphviz but which does not produce dot code either.

All three function return a matrix of commands suitable for using in the sem package. (Specify errors=TRUE to get code that will run directly in the sem package.)

The structure.graph output can be directed to an output file for post processing using the dot graphic language but requires that Rgraphviz is installed.

The figure is organized to show the appropriate paths between:

The correlations between the X variables (if Rx is specified)
The X variables and their latent factors (if fx is specified)
The latent X and the latent Y (if Phi is specified)
The latent Y and the observed Y (if fy is specified)
The correlations between the Y variables (if Ry is specified)

A confirmatory factor model would specify just fx and Phi, a structural model would include fx, Phi, and fy. The raw correlations could be shown by just including Rx and Ry.

`lavaan.diagram` may be called from the `diagram` function which also will call `fa.diagram`, `omega.diagram` or `iclust.diagram`, depending upon the class of the fit.

Other diagram functions include `fa.diagram`, `omega.diagram`. All of these functions use the various dia functions such as `dia.rect`, `dia.ellipse`, `dia.arrow`, `dia.curve`, `dia.curved.arrow`, and `dia.shape`.

Value

sem
: (invisible) a model matrix (partially) ready for input to John Fox's sem package. It is of class "mod" for prettier output.

dotfile
: If out.file is specified, a dot language file suitable for using in a dot graphics program such as graphviz or Omnigraffle.

A graphic structural diagram in the graphics window

Author(s)

William Revelle

See Also

`fa.graph`, `omega.graph`, `sim.structural` to create artificial data sets with particular structural properties.

Examples

```
fx <- matrix(c(.9,.8,.6,rep(0,4),.6,.8,-.7),ncol=2)
fy <- matrix(c(.6,.5,.4),ncol=1)
Phi <- matrix(c(1,0,0,0,1,0,.7,.7,1),ncol=3,byrow=TRUE)
f1 <- structure.diagram(fx,Phi,fy,main="A structural path diagram")

#symbolic input
X2 <- matrix(c("a",0,0,"b","e1",0,0,"e2"),ncol=4)
colnames(X2) <- c("X1","X2","E1","E2")
phi2 <- diag(1,4,4)
phi2[2,1] <- phi2[1,2] <- "r"
f2 <- structure.diagram(X2,Phi=phi2,errors=FALSE,main="A symbolic model")

#symbolic input with error
X2 <- matrix(c("a",0,0,"b"),ncol=2)
colnames(X2) <- c("X1","X2")
phi2 <- diag(1,2,2)
phi2[2,1] <- phi2[1,2] <- "r"
f3 <- structure.diagram(X2,Phi=phi2,main="an alternative representation")

#and yet another one
X6 <- matrix(c("a","b","c",rep(0,6),"d","e","f"),nrow=6)
colnames(X6) <- c("L1","L2")
rownames(X6) <- c("x1","x2","x3","x4","x5","x6")
Y3 <- matrix(c("u","w","z"),ncol=1)
colnames(Y3) <- "Y"
rownames(Y3) <- c("y1","y2","y3")
phi21 <- matrix(c(1,0,"r1",0,1,"r2",0,0,1),ncol=3)
colnames(phi21) <- rownames(phi21) <-  c("L1","L2","Y")
f4 <- structure.diagram(X6,phi21,Y3)



# and finally, a regression model
```

```
X7 <- matrix(c("a","b","c","d","e","f"),nrow=6)
f5 <- structure.diagram(X7,regression=TRUE)

#and a really messy regession model
x8 <- c("b1","b2","b3")
r8 <- matrix(c(1,"r12","r13","r12",1,"r23","r13","r23",1),ncol=3)
f6<- structure.diagram(x8,Phi=r8,regression=TRUE)
```

---

structure.list *Create factor model matrices from an input list*

Description

When creating a structural diagram or a structural model, it is convenient to not have to specify all of the zero loadings in a structural matrix. structure.list converts list input into a design matrix. phi.list does the same for a correlation matrix. Factors with NULL values are filled with 0s.

Usage

```
structure.list(nvars, f.list,f=NULL, f.labels = NULL, item.labels = NULL)
phi.list(nf,f.list, f.labels = NULL)
```

Arguments

nvars	Number of variables in the design matrix
f.list	A list of items included in each factor (for structure.list, or the factors that correlate with the specified factor for phi.list
f	prefix for parameters – needed in case of creating an X set and a Y set
f.labels	Names for the factors
item.labels	Item labels
nf	Number of factors in the phi matrix

Details

This is almost self explanatory. See the examples.

Value

factor.matrix	a matrix of factor loadings to model

See Also

structure.graph for drawing it, or sim.structure for creating this data structure.

Examples

```
fx <- structure.list(9,list(F1=c(1,2,3),F2=c(4,5,6),F3=c(7,8,9)))
fy <- structure.list(3,list(Y=c(1,2,3)),"Y")
phi <- phi.list(4,list(F1=c(4),F2=c(1,4),F3=c(2),F4=c(1,2,3)))
fx
phi
fy
```

---

superMatrix *Form a super matrix from two sub matrices.*

Description

Given the matrices nXm, and jYk, form the super matrix of dimensions (n+j) and (m+k) with with elements x and y along the super diagonal. Useful when considering structural equations. The measurement models x and y can be combined into a larger measurement model of all of the variables. If either x or y is a list of matrices, then recursively form a super matrix of all of those elements.

Usage

```
superMatrix(x,y)
super.matrix(x, y)  #Deprecated
```

Arguments

x	A n x m matrix or a list of such matrices
y	A j x k matrix or a list of such matrices

Details

Several functions, e.g., `sim.structural`,`structure.graph`, `make.keys` use matrices that can be thought of as formed from a set of submatrices. In particular, when using `make.keys` in order to score a set of items (`scoreItems` or `scoreOverlap`) or to form specified clusters (`cluster.cor`), it is convenient to define different sets of scoring keys for different sets of items and to combine these scoring keys into one super key.

Value

A (n+j) x (m +k) matrix with appropriate row and column names

Author(s)

William Revelle

See Also

sim.structural,structure.graph, make.keys

Examples

```
mx <- matrix(c(.9,.8,.7,rep(0,4),.8,.7,.6),ncol=2)
my <- matrix(c(.6,.5,.4))

colnames(mx) <- paste("X",1:dim(mx)[2],sep="")
rownames(mx) <- paste("Xv",1:dim(mx)[1],sep="")
colnames(my) <- "Y"
rownames(my) <- paste("Yv",1:3,sep="")
mxy <- superMatrix(mx,my)
#show the use of a list to do this as well
key1 <- make.keys(6,list(first=c(1,-2,3),second=4:6,all=1:6))  #make a scoring key
key2 <- make.keys(4,list(EA=c(1,2),TA=c(3,4)))
superMatrix(list(key1,key2))
```

---

table2matrix — *Convert a table with counts to a matrix or data.frame representing those counts.*

Description

Some historical sets are reported as summary tables of counts in a limited number of bins. Transforming these tables to data.frames representing the original values is useful for pedagogical purposes. (E.g., transforming the original Galton table of height x cubits in order to demonstrate regression.) The column and row names must be able to be converted to numeric values.

Usage

```
table2matrix(x, labs = NULL)
table2df(x, count=NULL,labs = NULL)
```

Arguments

x	A two dimensional table of counts with row and column names that can be converted to numeric values.
count	if present, then duplicate each row count times
labs	Labels for the rows and columns. These will be used for the names of the two columns of the resulting matrix

Details

The original Galton (1888) of heights by cubits (arm length) is in tabular form. To show this as a correlation or as a scatter plot, it is useful to convert the table to a matrix or data frame of two columns.

This function may also be used to convert an item response pattern table into a data table. e.g., the Bock data set bock.

Value

A matrix (or data.frame) of sum(x) rows and two columns.

Author(s)

William Revelle

See Also

cubits and bock data sets

Examples

```
data(cubits)
cubit <- table2matrix(cubits,labs=c("height","cubit"))
describe(cubit)
ellipses(cubit,n=1)
data(bock)
responses <- table2df(bock.table[,2:6],count=bock.table[,7],labs= paste("lsat6.",1:5,sep=""))
describe(responses)
```

`test.psych` *Testing of functions in the psych package*

Description

Test to make sure the psych functions run on basic test data sets

Usage

```
test.psych(first=1,last=5,short=TRUE,all=FALSE,fapc=FALSE)
```

Arguments

`first`	first=1: start with dataset first
`last`	last=5: test for datasets until last
`short`	short=TRUE - don't return any analyses
`all`	To get around a failure on certain Solaris 32 bit systems, all=FALSE is the default
`fapc`	if fapc=TRUE, then do a whole series of tests of factor and principal component extraction and rotations.

Details

When modifying the psych package, it is useful to make sure that adding some code does not break something else. The test.psych function tests the major functions on various standard data sets. It also shows off a number of the capabilities of the psych package.

Uses 5 standard data sets:
USArrests Violent Crime Rates by US State (4 variables)
attitude The Chatterjee-Price Attitude Data
Harman23.cor$cov Harman Example 2.3 8 physical measurements
Harman74.cor$cov Harman Example 7.4 24 mental measurements
ability.cov$cov 8 Ability and Intelligence Tests

It also uses the bfi and ability data sets from psych

Value

`out` if short=FALSE, then list of the output from all functions tested

Warning

Warning messages will be thrown by fa.parallel and sometimes by fa for random datasets.

Note

Although test.psych may be used as a quick demo of the various functions in the psych packge, in general, it is better to try the specific functions themselves. The main purpose of test.psych is to make sure functions throw error messages or correct for weird conditions.

The datasets tested are part of the standard R data sets and represent some of the basic problems encountered.

When version 1.1.10 was released, it caused errors when compiling and testing on some Solaris 32 bit systems. The all option was added to avoid this problem (since I can't replicate the problem on Macs or PCs). all=TRUE adds one more test, for a non-positive definite matrix.

Author(s)

William Revelle

Examples

```
#test <- test.psych()
#not run
#test.psych(all=TRUE)
#    f3 <- fa(bfi[1:15],3,n.iter=5)
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="Varimax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="varimax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="bifactor")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="varimin")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="bentlerT")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="geominT")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="equamax")
```

```
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="Promax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="cluster")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="biquartimin")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="equamax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="Promax")
#
#     fpoly <- fa(bfi[1:10],2,n.iter=5,cor="poly")
#     f1 <- fa(ability,n.iter=4)
#     f1p <- fa(ability,n.iter=4,cor="tet")
```

`tetrachoric` *Tetrachoric, polychoric, biserial and polyserial correlations from various types of input*

Description

The tetrachoric correlation is the inferred Pearson Correlation from a two x two table with the assumption of bivariate normality. The polychoric correlation generalizes this to the n x m table. Particularly important when doing Item Response Theory or converting comorbidity statistics using normal theory to correlations. Input may be a 2 x 2 table of cell frequencies, a vector of cell frequencies, or a data.frame or matrix of dichotomous data (for tetrachoric) or of numeric data (for polychoric). The biserial correlation is between a continuous y variable and a dichotmous x variable, which is assumed to have resulted from a dichotomized normal variable. Biserial is a special case of the polyserial correlation, which is the inferred latent correlation between a continuous variable (X) and a ordered categorical variable (e.g., an item response). Input for these later two are data frames or matrices. Requires the mnormt package.

Usage

```
tetrachoric(x,y=NULL,correct=.5,smooth=TRUE,global=TRUE,weight=NULL,na.rm=TRUE,
     delete=TRUE)
polychoric(x,smooth=TRUE,global=TRUE,polycor=FALSE,ML=FALSE, std.err=FALSE,
     weight=NULL,correct=.5,progress=TRUE,na.rm=TRUE,  delete=TRUE)
biserial(x,y)
polyserial(x,y)
polydi(p,d,taup,taud,global=TRUE,ML = FALSE, std.err = FALSE,
     weight=NULL,progress=TRUE,na.rm=TRUE,delete=TRUE,correct=.5)
#deprecated  use polychoric instead
poly.mat(x, short = TRUE, std.err = FALSE, ML = FALSE)
```

Arguments

`x` The input may be in one of four forms:

a) a data frame or matrix of dichotmous data (e.g., the lsat6 from the bock data set) or discrete numerical (i.e., not too many levels, e.g., the big 5 data set, bfi) for polychoric, or continuous for the case of biserial and polyserial.

b) a 2 x 2 table of cell counts or cell frequencies (for tetrachoric) or an n x m table of cell counts (for both tetrachoric and polychoric).

c) a vector with elements corresponding to the four cell frequencies (for tetrachoric)

d) a vector with elements of the two marginal frequencies (row and column) and the comorbidity (for tetrachoric)

`y`	A (matrix or dataframe) of discrete scores. In the case of tetrachoric, these should be dichotomous, for polychoric not too many levels, for biserial they should be discrete (e.g., item responses) with not too many (<10?) categories.
`correct`	Correction value to use to correct for continuity in the case of zero entry cell for tetrachoric, polychoric, polybi, and mixed.cor. See the examples for the effect of correcting versus not correcting for continuity.
`smooth`	if TRUE and if the tetrachoric/polychoric matrix is not positive definite, then apply a simple smoothing algorithm using cor.smooth
`global`	When finding pairwise correlations, should we use the global values of the tau parameter (which is somewhat faster), or the local values (global=FALSE)? The local option is equivalent to the polycor solution, or to doing one correlation at a time. global=TRUE borrows information for one item pair from the other pairs using those item's frequencies. This will make a difference in the presence of lots of missing data. With very small sample sizes with global=FALSE and correct=TRUE, the function will fail (for as yet underdetermined reasons.
`polycor`	A no longer used option, kept to stop other packages from breaking.
`weight`	A vector of length of the number of observations that specifies the weights to apply to each case. The NULL case is equivalent of weights of 1 for all cases.
`short`	short=TRUE, just show the correlations, short=FALSE give the full hetcor output from John Fox's hetcor function if installed and if doing polychoric Deprecated
`std.err`	std.err=FALSE does not report the standard errors (faster) deprecated
`progress`	Show the progress bar (if not doing multicores)
`ML`	ML=FALSE do a quick two step procedure, ML=TRUE, do longer maximum likelihood — very slow! Deprecated
`na.rm`	Should missing data be deleted
`delete`	Cases with no variance are deleted with a warning before proceeding.
`p`	The polytomous input to polydi
`d`	The dichotomous input to polydi
`taup`	The tau values for the polytomous variables – if global=TRUE
`taud`	The tau values for the dichotomous variables – if globabl = TRUE

Details

Tetrachoric correlations infer a latent Pearson correlation from a two x two table of frequencies with the assumption of bivariate normality. The estimation procedure is two stage ML. Cell frequencies for each pair of items are found. In the case of tetrachorics, cells with zero counts are replaced with .5 as a correction for continuity (correct=TRUE).

The data typically will be a raw data matrix of responses to a questionnaire scored either true/false (tetrachoric) or with a limited number of responses (polychoric). In both cases, the marginal frequencies are converted to normal theory thresholds and the resulting table for each item pair is converted to the (inferred) latent Pearson correlation that would produce the observed cell frequencies with the observed marginals. (See `draw.tetra` and `draw.cor` for illustrations.)

This is a very computationally intensive function which can be speeded up considerably by using multiple cores and using the parallel package. The number of cores to use when doing polychoric or tetrachoric may be specified using the options command. The greatest step in speed is going from 1 core to 2. This is about a 50% savings. Going to 4 cores seems to have about at 66% savings, and 8 a 75% savings. The number of parallel processes defaults to 2 but can be modified by using the `options` command: options("mc.cores"=4) will set the number of cores to 4.

The tetrachoric correlation is used in a variety of contexts, one important one being in Item Response Theory (IRT) analyses of test scores, a second in the conversion of comorbity statistics to correlation coefficients. It is in this second context that examples of the sensitivity of the coefficient to the cell frequencies becomes apparent:

Consider the test data set from Kirk (1973) who reports the effectiveness of a ML algorithm for the tetrachoric correlation (see examples).

Examples include the lsat6 and lsat7 data sets in the `bock` data.

The polychoric function forms matrices of polychoric correlations by an local function (polyc) and will also report the tau values for each alternatives. Earlier versions used John Fox's polychor function which has now been replaced by the polyc function.

polychoric replaces poly.mat and is recommended. poly.mat is an alternative wrapper to the polycor function.

biserial and polyserial correlations are the inferred latent correlations equivalent to the observed point-biserial and point-polyserial correlations (which are themselves just Pearson correlations).

The polyserial function is meant to work with matrix or dataframe input and treats missing data by finding the pairwise Pearson r corrected by the overall (all observed cases) probability of response frequency. This is particularly useful for SAPA procedures (`http://sapa-project.org`) with large amounts of missing data and no complete cases.

Ability tests and personality test matrices will typically have a cleaner structure when using tetrachoric or polychoric correlations than when using the normal Pearson correlation. However, if either alpha or omega is used to find the reliability, this will be an overestimate of the squared correlation of a latent variable the observed variable.

A biserial correlation (not to be confused with the point-biserial correlation which is just a Pearson correlation) is the latent correlation between x and y where y is continuous and x is dichotomous but assumed to represent an (unobserved) continuous normal variable. Let p = probability of x level 1, and q = 1 - p. Let zp = the normal ordinate of the z score associated with p. Then, $rbi = rs * \sqrt(pq)/zp$.

The 'ad hoc' polyserial correlation, rps is just $r = r * sqrt(n-1)/n)\sigma y/\sum(zpi)$ where zpi are the ordinates of the normal curve at the normal equivalent of the cut point boundaries between the item responses. (Olsson, 1982)

All of these were inspired by (and adapted from) John Fox's polychor package which should be used for precise ML estimates of the correlations. See, in particular, the hetcor function in the polychor package. The results from polychoric match the polychor answers to at least 5 decimals when using correct=FALSE, and global = FALSE.

Particularly for tetrachoric correlations from sets of data with missing data, the matrix will sometimes not be positive definite. Various smoothing alternatives are possible, the one done here is to do an eigen value decomposition of the correlation matrix, set all negative eigen values to 10 * .Machine$double.eps, normalize the positive eigen values to sum to the number of variables, and then reconstitute the correlation matrix. A warning is issued when this is done.

For very small data sets, the correction for continuity for the polychoric correlations can lead to difficulties, particularly if using the global=FALSE option, or if doing just one correlation at a time. Setting a smaller correction value (i.e., correct =.1) seems to help.

For combinations of continous, categorical, and dichotomous variables, see `mixed.cor`.

If using data with a variable number of response alternatives, it is necessary to use the global=FALSE option in polychoric.

Value

`rho`	The (matrix) of tetrachoric/polychoric/biserial correlations
`tau`	The normal equivalent of the cutpoints
`fixed`	If any correlations were adjusted for continuity, the total number of adjustments will be reported.

Note

For tetrachoric, in the degenerate case of a cell entry with zero observations, a correction for continuity is applied and .5 is added to the cell entry. A warning is issued. If correct=FALSE the correction is not applied. This correction is, by default, on. It can be adjusted by specifying a smaller value. See the examples.

For correct=FALSE, the results agree perfectly with John Fox's polycor function.

Switched to using sadmvn from the mnormt package to speed up by 50%.

Author(s)

William Revelle

References

A. Gunther and M. Hofler. Different results on tetrachorical correlations in mplus and stata-stata announces modified procedure. Int J Methods Psychiatr Res, 15(3):157-66, 2006.

David Kirk (1973) On the numerical approximation of the bivariate normal (tetrachoric) correlation coefficient. Psychometrika, 38, 259-268.

U.Olsson, F.Drasgow, and N.Dorans (1982). The polyserial correlation coefficient. Psychometrika, 47:337-347.

See Also

`mixed.cor` to find the correlations between mixtures of continuous, polytomous, and dichtomous variables. See also the polychor function in the polycor package. `irt.fa` uses the tetrachoric function to do item analysis with the `fa` factor analysis function. `draw.tetra` shows the logic behind a tetrachoric correlation (for teaching purpuses.)

Examples

```
#if(require(mnormt)) {
data(bock)
tetrachoric(lsat6)
polychoric(lsat6)  #values should be the same
tetrachoric(matrix(c(44268,193,14,0),2,2))  #MPLUS reports.24

#Do not apply continuity correction -- compare with previous analysis!
tetrachoric(matrix(c(44268,193,14,0),2,2),correct=0)

#the default is to add correct=.5 to 0 cells
tetrachoric(matrix(c(61661,1610,85,20),2,2)) #Mplus reports .35
tetrachoric(matrix(c(62503,105,768,0),2,2)) #Mplus reports -.10
tetrachoric(matrix(c(24875,265,47,0),2,2)) #Mplus reports  0

polychoric(matrix(c(61661,1610,85,20),2,2)) #Mplus reports .35
polychoric(matrix(c(62503,105,768,0),2,2)) #Mplus reports -.10
polychoric(matrix(c(24875,265,47,0),2,2)) #Mplus reports  0

#Do not apply continuity correction- compare with previous analysis
tetrachoric(matrix(c(24875,265,47,0),2,2), correct=0)
polychoric(matrix(c(24875,265,47,0),2,2), correct=0)  #the same result
#these next examples are impossible!

tetrachoric(c(0.02275000, 0.0227501320, 0.500000000))
tetrachoric(c(0.0227501320, 0.0227501320, 0.500000000))

#give a vector of two marginals and the comorbidity
tetrachoric(c(.2, .15, .1))
tetrachoric(c(.2, .1001, .1))
 #} else {
 #       message("Sorry, you must have mnormt installed")}

# 4 plots comparing biserial to point biserial and latent Pearson correlation
set.seed(42)
x.4 <- sim.congeneric(loads =c(.9,.6,.3,0),N=1000,short=FALSE)
y  <- x.4$latent[,1]
for(i in 1:4) {
x <- x.4$observed[,i]
r <- round(cor(x,y),1)
ylow <- y[x<= 0]
yhigh <- y[x > 0]
yc <- c(ylow,yhigh)
rpb <- round(cor((x>=0),y),2)
rbis <- round(biserial(y,(x>=0)),2)
ellipses(x,y,ylim=c(-3,3),xlim=c(-4,3),pch=21 - (x>0),
       main =paste("r = ",r,"rpb = ",rpb,"rbis =",rbis))

dlow <- density(ylow)
dhigh <- density(yhigh)
points(dlow$y*5-4,dlow$x,typ="l",lty="dashed")
lines(dhigh$y*5-4,dhigh$x,typ="l")
```

```
}
```

---

thurstone *Thurstone Case V scaling*

Description

Thurstone Case V scaling allows for a scaling of objects compared to other objects. As one of the cases considered by Thurstone, Case V makes the assumption of equal variances and uncorrelated distributions.

Usage

```
thurstone(x, ranks = FALSE, digits = 2)
```

Arguments

`x`	A square matrix or data frame of preferences, or a rectangular data frame or matrix rank order choices.
`ranks`	TRUE if rank orders are presented
`digits`	number of digits in the goodness of fit

Details

Louis L. Thurstone was a pioneer in psychometric theory and measurement of attitudes, interests, and abilities. Among his many contributions was a systematic analysis of the process of comparative judgment (thurstone, 1927). He considered the case of asking subjects to successively compare pairs of objects. If the same subject does this repeatedly, or if subjects act as random replicates of each other, their judgments can be thought of as sampled from a normal distribution of underlying (latent) scale scores for each object, Thurstone proposed that the comparison between the value of two objects could be represented as representing the differences of the average value for each object compared to the standard deviation of the differences between objects. The basic model is that each item has a normal distribution of response strength and that choice represents the stronger of the two response strengths. A justification for the normality assumption is that each decision represents the sum of many independent inputs and thus, through the central limit theorem, is normally distributed.

Thurstone considered five different sets of assumptions about the equality and independence of the variances for each item (Thurston, 1927). Torgerson expanded this analysis slightly by considering three classes of data collection (with individuals, between individuals and mixes of within and between) crossed with three sets of assumptions (equal covariance of decision process, equal correlations and small differences in variance, equal variances).

The data may be either a square matrix of dataframe of preferences (as proportions with the probability of the column variable being chosen over the row variable) or a matrix or dataframe of rank orders (1 being prefered to 2, etc.)

Value

GF	Goodness of fit 1 = 1 - sum(squared residuals/squared original) for lower off diagonal. Goodness of fit 2 = 1 - sum(squared residuals/squared original) for full matrix.
residual	square matrix of residuals (of class dist)
data	The original choice data
...	

Author(s)

William Revelle

References

Thurstone, L. L. (1927) A law of comparative judgments. Psychological Review, 34, 273-286.

Revelle, W. An introduction to psychometric theory with applications in R. (in preparation), Springer. http://personality-project.org/r/book

Examples

```
data(vegetables)
thurstone(veg)
```

---

tr   *Find the trace of a square matrix*

Description

Hardly worth coding, if it didn't appear in so many formulae in psychometrics, the trace of a (square) matrix is just the sum of the diagonal elements.

Usage

```
tr(m)
```

Arguments

m	A square matrix

Details

The tr function is used in various matrix operations and is the sum of the diagonal elements of a matrix.

Value

The sum of the diagonal elements of a square matrix.
i.e. tr(m) <- sum(diag(m)).

Examples

```
m <- matrix(1:16,ncol=4)
m
tr(m)
```

Tucker *9 Cognitive variables discussed by Tucker and Lewis (1973)*

Description

Tucker and Lewis (1973) introduced a reliability coefficient for ML factor analysis. Their example data set was previously reported by Tucker (1958) and taken from Thurstone and Thurstone (1941). The correlation matrix is a 9 x 9 for 710 subjects and has two correlated factors of ability: Word Fluency and Verbal.

Usage

```
data(Tucker)
```

Format

A data frame with 9 observations on the following 9 variables.

`t42` Prefixes

`t54` Suffixes

`t45` Chicago Reading Test: Vocabulary

`t46` Chicago Reading Test: Sentences

`t23` First and last letters

`t24` First letters

`t27` Four letter words

`t10` Completion

`t51` Same or Opposite

Details

The correlation matrix from Tucker (1958) was used in Tucker and Lewis (1973) for the Tucker-Lewis Index of factoring reliability.

Source

Tucker, Ledyard (1958) An inter-battery method of factor analysis, Psychometrika, 23, 111-136.

References

L.~Tucker and C.~Lewis. (1973) A reliability coefficient for maximum likelihood factor analysis. Psychometrika, 38(1):1–10.

F.~J. Floyd and K.~F. Widaman. (1995) Factor analysis in the development and refinement of clinical assessment instruments., Psychological Assessment, 7(3):286 – 299.

Examples

```
data(Tucker)
fa(Tucker,2,n.obs=710)
omega(Tucker,2)
```

`vegetables` *Paired comparison of preferences for 9 vegetables*

Description

A classic data set for demonstrating Thurstonian scaling is the preference matrix of 9 vegetables from Guilford (1954). Used by Guiford, Nunnally, and Nunally and Bernstein, this data set allows for examples of basic scaling techniques.

Usage

```
data(vegetables)
```

Format

A data frame with 9 choices on the following 9 vegetables. The values reflect the perecentage of times where the column entry was preferred over the row entry.

`Turn` Turnips

`Cab` Cabbage

`Beet` Beets

`Asp` Asparagus

`Car` Carrots

`Spin` Spinach

`S.Beans` String Beans

`Peas` Peas

`Corn` Corn

Details

Louis L. Thurstone was a pioneer in psychometric theory and measurement of attitudes, interests, and abilities. Among his many contributions was a systematic analysis of the process of comparative judgment (thurstone, 1927). He considered the case of asking subjects to successively compare pairs of objects. If the same subject does this repeatedly, or if subjects act as random replicates of each other, their judgments can be thought of as sampled from a normal distribution of underlying (latent) scale scores for each object, Thurstone proposed that the comparison between the value of two objects could be represented as representing the differences of the average value for each object compared to the standard deviation of the differences between objects. The basic model is that each item has a normal distribution of response strength and that choice represents the stronger of the two response strengths. A justification for the normality assumption is that each decision represents the sum of many independent inputs and thus, through the central limit theorem, is normally distributed.

Thurstone considered five different sets of assumptions about the equality and independence of the variances for each item (Thurston, 1927). Torgerson expanded this analysis slightly by considering three classes of data collection (with individuals, between individuals and mixes of within and between) crossed with three sets of assumptions (equal covariance of decision process, equal correlations and small differences in variance, equal variances).

This vegetable data set is used by Guilford and by Nunnally to demonstrate Thurstonian scaling.

Source

Guilford, J.P. (1954) Psychometric Methods. McGraw-Hill, New York.

References

Nunnally, J. C. (1967). Psychometric theory., McGraw-Hill, New York.

Revelle, W. An introduction to psychometric theory with applications in R. (in preparation), Springer. http://personality-project.org/r/book

See Also

thurstone

Examples

```
data(vegetables)
thurstone(veg)
```

VSS *Apply the Very Simple Structure, MAP, and other criteria to determine the appropriate number of factors.*

Description

There are multiple ways to determine the appropriate number of factors in exploratory factor analysis. Routines for the Very Simple Structure (VSS) criterion allow one to compare solutions of varying complexity and for different number of factors. Graphic output indicates the "optimal" number of factors for different levels of complexity. The Velicer MAP criterion is another good choice. nfactors finds and plots several of these alternative estimates.

Usage

```
vss(x, n = 8, rotate = "varimax", diagonal = FALSE, fm = "minres",
n.obs=NULL,plot=TRUE,title="Very Simple Structure",use="pairwise",cor="cor",...)
VSS(x, n = 8, rotate = "varimax", diagonal = FALSE, fm = "minres",
n.obs=NULL,plot=TRUE,title="Very Simple Structure",use="pairwise",cor="cor",...)
nfactors(x,n=20,rotate="varimax",diagonal=FALSE,fm="minres",n.obs=NULL,
             title="Number of Factors",pch=16,use="pairwise", cor="cor",...)
```

Arguments

x	a correlation matrix or a data matrix
n	Number of factors to extract – should be more than hypothesized!
rotate	what rotation to use c("none", "varimax", "oblimin","promax")
diagonal	Should we fit the diagonal as well
fm	factoring method – fm="pa" Principal Axis Factor Analysis, fm = "minres" minimum residual (OLS) factoring fm="mle" Maximum Likelihood FA, fm="pc" Principal Components"
n.obs	Number of observations if doing a factor analysis of correlation matrix. This value is ignored by VSS but is necessary for the ML factor analysis package.
plot	plot=TRUE Automatically call VSS.plot with the VSS output, otherwise don't plot
title	a title to be passed on to VSS.plot
pch	the plot character for the nfactors plots
use	If doing covariances or Pearson R, should we use "pairwise" or "complete cases"
cor	What kind of correlation to find, defaults to Pearson but see fa for the choices
...	parameters to pass to the factor analysis program The most important of these is if using a correlation matrix is covmat= xx

Details

Determining the most interpretable number of factors from a factor analysis is perhaps one of the greatest challenges in factor analysis. There are many solutions to this problem, none of which is uniformly the best. "Solving the number of factors problem is easy, I do it everyday before breakfast. But knowing the right solution is harder" (Kaiser, 195x).

Techniques most commonly used include

1) Extracting factors until the chi square of the residual matrix is not significant.

2) Extracting factors until the change in chi square from factor n to factor n+1 is not significant.

3) Extracting factors until the eigen values of the real data are less than the corresponding eigen values of a random data set of the same size (parallel analysis) `fa.parallel`.

4) Plotting the magnitude of the successive eigen values and applying the scree test (a sudden drop in eigen values analogous to the change in slope seen when scrambling up the talus slope of a mountain and approaching the rock face.

5) Extracting principal components until the eigen value < 1.

6) Extracting factors as long as they are interpetable.

7) Using the Very Simple Structure Criterion (VSS).

8) Using Wayne Velicer's Minimum Average Partial (MAP) criterion.

Each of the procedures has its advantages and disadvantages. Using either the chi square test or the change in square test is, of course, sensitive to the number of subjects and leads to the nonsensical condition that if one wants to find many factors, one simply runs more subjects. Parallel analysis is partially sensitive to sample size in that for large samples the eigen values of random factors will be very small. The scree test is quite appealling but can lead to differences of interpretation as to when the scree "breaks". The eigen value of 1 rule, although the default for many programs, seems to be a rough way of dividing the number of variables by 3. Extracting interpretable factors means that the number of factors reflects the investigators creativity more than the data. VSS, while very simple to understand, will not work very well if the data are very factorially complex. (Simulations suggests it will work fine if the complexities of some of the items are no more than 2).

Most users of factor analysis tend to interpret factor output by focusing their attention on the largest loadings for every variable and ignoring the smaller ones. Very Simple Structure operationalizes this tendency by comparing the original correlation matrix to that reproduced by a simplified version (S) of the original factor matrix (F). R = SS' + U2. S is composed of just the c greatest (in absolute value) loadings for each variable. C (or complexity) is a parameter of the model and may vary from 1 to the number of factors.

The VSS criterion compares the fit of the simplified model to the original correlations: VSS = 1 -sumsquares(r*)/sumsquares(r) where R* is the residual matrix R* = R - SS' and r* and r are the elements of R* and R respectively.

VSS for a given complexity will tend to peak at the optimal (most interpretable) number of factors (Revelle and Rocklin, 1979).

Although originally written in Fortran for main frame computers, VSS has been adapted to micro computers (e.g., Macintosh OS 6-9) using Pascal. We now release R code for calculating VSS.

Note that if using a correlation matrix (e.g., my.matrix) and doing a factor analysis, the parameters n.obs should be specified for the factor analysis: e.g., the call is VSS(my.matrix,n.obs=500). Otherwise it defaults to 1000.

Wayne Velicer's MAP criterion has been added as an additional test for the optimal number of components to extract. Note that VSS and MAP will not always agree as to the optimal number.

The nfactors function will do a VSS, find MAP, and report a number of other criteria (e.g., BIC, complexity, chi square, ...)

A variety of rotation options are available. These include varimax, promax, and oblimin. Others can be added. Suggestions are welcome.

Value

A data.frame with entries: map: Velicer's MAP values (lower values are better)
dof: degrees of freedom (if using FA)
chisq: chi square (from the factor analysis output (if using FA)
prob: probability of residual matrix > 0 (if using FA)
sqresid: squared residual correlations
RMSEA: the RMSEA for each number of factors
BIC: the BIC for each number of factors
eChiSq: the empirically found chi square
eRMS: Empirically found mean residual
eCRMS: Empirically found mean residual corrected for df
eBIC: The empirically found BIC based upon the eChiSq
fit: factor fit of the complete model
cfit.1: VSS fit of complexity 1
cfit.2: VSS fit of complexity 2
...
cfit.8: VSS fit of complexity 8
cresidiual.1: sum squared residual correlations for complexity 1
...: sum squared residual correlations for complexity 2 ..8

Author(s)

William Revelle

References

http://personality-project.org/r/vss.html, Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at http://personality-project.org/r/book/

Revelle, W. and Rocklin, T. 1979, Very Simple Structure: an Alternative Procedure for Estimating the Optimal Number of Interpretable Factors, Multivariate Behavioral Research, 14, 403-414. http://personality-project.org/revelle/publications/vss.pdf

Velicer, W. (1976) Determining the number of components from the matrix of partial correlations. Psychometrika, 41, 321-327.

See Also

VSS.plot, ICLUST, omega, fa.parallel

Examples

```
#test.data <- Harman74.cor$cov
#my.vss <- VSS(test.data,title="VSS of 24 mental tests")
#print(my.vss[,1:12],digits =2)
#VSS.plot(my.vss, title="VSS of 24 mental tests")

#now, some simulated data with two factors
```

```
#VSS(sim.circ(nvar=24),fm="minres" ,title="VSS of 24 circumplex variables")
VSS(sim.item(nvar=24),fm="minres" ,title="VSS of 24 simple structure variables")
```

VSS.parallel *Compare real and random VSS solutions*

Description

Another useful test for the number of factors is when the eigen values of a random matrix are greater than the eigen values of a a real matrix. Here we show VSS solutions to random data. A better test is probably fa.parallel.

Usage

```
VSS.parallel(ncases, nvariables,scree=FALSE,rotate="none")
```

Arguments

ncases	Number of simulated cases
nvariables	number of simulated variables
scree	Show a scree plot for random data – see omega
rotate	rotate="none" or rotate="varimax"

Value

VSS like output to be plotted by VSS.plot

Author(s)

William Revelle

References

Very Simple Structure (VSS)

See Also

fa.parallel, VSS.plot, ICLUST, omega

Examples

```
#VSS.plot(VSS.parallel(200,24))
```

VSS.plot *Plot VSS fits*

Description

The Very Simple Structure criterion (VSS) for estimating the optimal number of factors is plotted as a function of the increasing complexity and increasing number of factors.

Usage

```
VSS.plot(x, title = "Very Simple Structure", line = FALSE)
```

Arguments

x	output from VSS
title	any title
line	connect different complexities

Details

Item-factor models differ in their "complexity". Complexity 1 means that all except the greatest (absolute) loading for an item are ignored. Basically a cluster model (e.g., ICLUST). Complexity 2 implies all except the greatest two, etc.

Different complexities can suggest different number of optimal number of factors to extract. For personality items, complexity 1 and 2 are probably the most meaningful.

The Very Simple Structure criterion will tend to peak at the number of factors that are most interpretable for a given level of complexity. Note that some problems, the most interpretable number of factors will differ as a function of complexity. For instance, when doing the Harman 24 psychological variable problems, an unrotated solution of complexity one suggests one factor (g), while a complexity two solution suggests that a four factor solution is most appropriate. This latter probably reflects a bi-factor structure.

For examples of VSS.plot output, see http://personality-project.org/r/r.vss.html

Value

A plot window showing the VSS criterion varying as the number of factors and the complexity of the items.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

http://personality-project.org/r/r.vss.html

See Also

VSS, ICLUST, omega

Examples

```
test.data <- Harman74.cor$cov
my.vss <- VSS(test.data)        #suggests that 4 factor complexity two solution is optimal
VSS.plot(my.vss,title="VSS of Holzinger-Harmon problem")            #see the graphics window
```

VSS.scree *Plot the successive eigen values for a scree test*

Description

Cattell's scree test is one of most simple ways of testing the number of components or factors in a correlation matrix. Here we plot the eigen values of a correlation matrix as well as the eigen values of a factor analysis.

Usage

```
scree(rx,factors=TRUE,pc=TRUE,main="Scree plot",hline=NULL,add=FALSE)
VSS.scree(rx, main = "scree plot")
```

Arguments

rx	a correlation matrix or a data matrix. If data, then correlations are found using pairwise deletions.
factors	If true, draw the scree for factors
pc	If true, draw the scree for components
hline	if null, draw a horizontal line at 1, otherwise draw it at hline (make negative to not draw it)
main	Title
add	Should multiple plots be drawn?

Details

Among the many ways to choose the optimal number of factors is the scree test. A better function to show the scree as well as compare it to randomly parallel solutions is found found in fa.parallel

Author(s)

William Revelle

References

http://personality-project.org/r/vss.html

See Also

fa.parallel VSS.plot, ICLUST, omega

Examples

```
scree(attitude)
#VSS.scree(cor(attitude)
```

---

winsor *Find the Winsorized scores, means, sds or variances for a vector, matrix, or data.frame*

Description

Among the robust estimates of central tendency are trimmed means and Winsorized means. This function finds the Winsorized scores. The top and bottom trim values are given values of the trimmed and 1- trimmed quantiles. Then means, sds, and variances are found.

Usage

```
winsor(x, trim = 0.2, na.rm = TRUE)
winsor.mean(x, trim = 0.2, na.rm = TRUE)
winsor.means(x, trim = 0.2, na.rm = TRUE)
winsor.sd(x, trim = 0.2, na.rm = TRUE)
winsor.var(x, trim = 0.2, na.rm = TRUE)
```

Arguments

x	A data vector, matrix or data frame
trim	Percentage of data to move from the top and bottom of the distributions
na.rm	Missing data are removed

Details

Among the many robust estimates of central tendency, some recommend the Winsorized mean. Rather than just dropping the top and bottom trim percent, these extreme values are replaced with values at the trim and 1- trim quantiles.

Value

A scalar or vector of winsorized scores or winsorized means, sds, or variances (depending upon the call).

Author(s)

William Revelle with modifications suggested by Joe Paxton and a further correction added (January, 2009) to preserve the original order for the winsor case.

References

Wilcox, Rand R. (2005) Introduction to robust estimation and hypothesis testing. Elsevier/Academic Press. Amsterdam ; Boston.

See Also

interp.median

Examples

```
data(sat.act)
winsor.means(sat.act) #compare with the means of the winsorized scores
y <- winsor(sat.act)
describe(y)
xy <- data.frame(sat.act,y)
#pairs.panels(xy) #to see the effect of winsorizing
x <- matrix(1:100,ncol=5)
winsor(x)
winsor.means(x)
y <- 1:11
winsor(y,trim=.5)
```

---

withinBetween *An example of the distinction between within group and between group correlations*

Description

A demonstration that a correlation may be decomposed to a within group correlation and a between group correlations and these two correlations are independent. Between group correlations are sometimes called ecological correlations, the decomposition into within and between group correlations is a basic concept in multilevel modeling. This data set shows the composite correlations between 9 variables, representing 16 cases with four groups.

Usage

```
data(withinBetween)
```

Format

A data frame with 16 observations on the following 10 variables.

`Group` An example grouping factor.

`V1` A column of 16 observations

`V2` A column of 16 observations

`V3` A column of 16 observations

`V4` A column of 16 observations

`V5` A column of 16 observations

`V6` A column of 16 observations

`V7` A column of 16 observations

`V8` A column of 16 observations

`V9` A column of 16 observations

Details

Correlations between individuals who belong to different natural groups (based upon e.g., ethnicity, age, gender, college major,or country) reflect an unknown mixture of the pooled correlation within each group as well as the correlation of the means of these groups. These two correlations are independent and do not allow inferences from one level (the group) to the other level (the individual). This data set shows this independence. The within group correlations between 9 variables are set to be 1, 0, and -1 while those between groups are also set to be 1, 0, -1. These two sets of correlations are crossed such that V1, V4, and V7 have within group correlations of 1, as do V2, V5 and V8, and V3, V6 and V9. V1 has a within group correlation of 0 with V2, V5, and V8, and a -1 within group correlation with V3, V6 and V9. V1, V2, and V3 share a between group correlation of 1, as do V4, V5 and V6, and V7, V8 and V9. The first group has a 0 between group correlation with the second and a -1 with the third group.

`statsBy` can decompose the observed correlation in the between and within correlations. `sim.multilevel` can produce similar data.

Source

The data were created for this example

References

P. D. Bliese. Multilevel modeling in R (2.3) a brief introduction to R, the multilevel package and the nlme package, 2009.

Pedhazur, E.J. (1997) Multiple regression in behavioral research: explanation and prediction. Harcourt Brace.

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at http://personality-project.org/r/book/

See Also

`statsBy`, `describeBy`, and `sim.multilevel`

Examples

```
data(withinBetween)
pairs.panels(withinBetween,bg=c("red","blue","white","black")[withinBetween[,1]],
     pch=21,ellipses=FALSE)
stats <- statsBy(withinBetween,'Group')
print(stats,short=FALSE)
```

Yule *From a two by two table, find the Yule coefficients of association, convert to phi, or tetrachoric, recreate table the table to create the Yule coefficient.*

Description

One of the many measures of association is the Yule coefficient. Given a two x two table of counts

a	b	R1
c	d	R2
C1	C2	n

Yule Q is (ad - bc)/(ad+bc).
Conceptually, this is the number of pairs in agreement (ad) - the number in disagreement (bc) over the total number of paired observations. Warren (2008) has shown that Yule's Q is one of the "coefficients that have zero value under statistical independence, maximum value unity, and minimum value minus unity independent of the marginal distributions" (p 787).
ad/bc is the odds ratio and Q = (OR-1)/(OR+1)
Yule's coefficient of colligation is Y = (sqrt(OR) - 1)/(sqrt(OR)+1) Yule.inv finds the cell entries for a particular Q and the marginals (a+b,c+d,a+c, b+d). This is useful for converting old tables of correlations into more conventional `phi` or tetrachoric correlations `tetrachoric`
Yule2phi and Yule2tetra convert the Yule Q with set marginals to the correponding phi or tetrachoric correlation.

Bonett and Price show that the Q and Y coefficients are both part of a general family of coefficients raising the OR to a power (c). If c=1, then this is Yule's Q. If .5, then Yule's Y, if c = .75, then this is Digby's H. They propose that c = .5 - (.5 * min(cell probabilty)^2 is a more general coefficient. YuleBonett implements this for the 2 x 2 case, YuleCor for the data matrix case.

Usage

```
YuleBonett(x,c=1,bonett=FALSE,alpha=.05) #find the generalized Yule cofficients
YuleCor(x,c=1,bonett=FALSE,alpha=.05) #do this for a matrix
Yule(x,Y=FALSE)  #find Yule given a two by two table of frequencies
 #find the frequencies that produce a Yule Q given the Q and marginals
Yule.inv(Q,m,n=NULL)
#find the phi coefficient that matches the Yule Q given the marginals
```

```
Yule2phi(Q,m,n=NULL)
Yule2tetra(Q,m,n=NULL,correct=TRUE)


   #Find the tetrachoric correlation given the Yule Q and the marginals
#(deprecated) Find the tetrachoric correlation given the Yule Q and the marginals
Yule2poly(Q,m,n=NULL,correct=TRUE)
```

Arguments

`x`	A vector of four elements or a two by two matrix, or, in the case of YuleBonett or YuleCor, this can also be a data matrix
`c`	1 returns Yule Q, .5, Yule's Y, .75 Digby's H
`bonett`	If FALSE, then find Q, Y, or H, if TRUE, then find the generalized Bonett coefficient
`alpha`	The two tailed probability for confidence intervals
`Y`	Y=TRUE return Yule's Y coefficient of colligation
`Q`	Either a single Yule coefficient or a matrix of Yule coefficients
`m`	The vector c(R1,C2) or a two x two matrix of marginals or a four element vector of marginals. The preferred form is c(R1,C1)
`n`	The number of subjects (if the marginals are given as frequencies
`correct`	When finding a tetrachoric correlation, should small cell sizes be corrected for continuity. See `{link{tetrachoric}` for a discussion.

Details

Yule developed two measures of association for two by two tables. Both are functions of the odds ratio

Value

`Q`	The Yule Q coefficient
`R`	A two by two matrix of counts
`result`	If given matrix input, then a matrix of phis or tetrachorics
`rho`	From YuleBonett and YuleCor
`ci`	The upper and lower confidence intervals in matrix form (From YuleBonett and YuleCor).

Note

Yule.inv is currently done by using the optimize function, but presumably could be redone by solving a quadratic equation.

Author(s)

William Revelle

References

Yule, G. Uday (1912) On the methods of measuring association between two attributes. Journal of the Royal Statistical Society, LXXV, 579-652

Bonett, D.G. and Price, R.M, (2007) Statistical Inference for Generalized Yule Coefficients in 2 x 2 Contingency Tables. Sociological Methods and Research, 35, 429-446.

Warrens, Matthijs (2008), On Association Coefficients for 2x2 Tables and Properties That Do Not Depend on the Marginal Distributions. Psychometrika, 73, 777-789.

See Also

See Also as phi, tetrachoric, Yule2poly.matrix, Yule2phi.matrix

Examples

```
Nach <- matrix(c(40,10,20,50),ncol=2,byrow=TRUE)
Yule(Nach)
Yule.inv(.81818,c(50,60),n=120)
Yule2phi(.81818,c(50,60),n=120)
Yule2tetra(.81818,c(50,60),n=120)
phi(Nach)  #much less
#or express as percents and do not specify n
Nach <- matrix(c(40,10,20,50),ncol=2,byrow=TRUE)
Nach/120
Yule(Nach)
Yule.inv(.81818,c(.41667,.5))
Yule2phi(.81818,c(.41667,.5))
Yule2tetra(.81818,c(.41667,.5))
phi(Nach)  #much less
YuleCor(ability[,1:4],,TRUE)
YuleBonett(Nach,1)  #Yule Q
YuleBonett(Nach,.5)  #Yule Y
YuleBonett(Nach,.75)  #Digby H
YuleBonett(Nach,,TRUE)  #Yule* is a generalized Yule
```

Index