Brief Guide to STATA Commands

Eric M. Uslaner December, 1999

use	use a data set–e.g.: use anes92
save	save a data set: save anes92–or save anes92,replace (if the data set already exists) save anes92,old (for STATA 5 format) Or Control+S
clear	clear existing data in memory
set mem 80m	allocate 80 megs of memory to STATA (default depends on machine) NOTE THAT STATA has a maximum limit of 2047 variables no matter how much memory you have.
exit	exit STATA
help ____	help for STATA command _____ help contents gives a list of STATA commands where you can get help.
search ____	search the on-line manual for all references to ____ (e.g., search regress gives all references to regress in STATA—it's a lot!)
log	set log file for output, e.g., log using c:\log\mylog will produce a file called mylog.log, which you can edit in any ASCII word processor. Variations: log using c:\log\mylog,append will add to existing file log using c:\log\mylog,replace will replace existing file log close will close log file (can reopen with append)
if	restricts commands to those cases that fulfill condition: e.g., sum var1 if partyid==1 (note two equal signs needed) will produce a summary (see below) of var1 only for partyid==1 (e.g., Democrats)
in	restricts commands to ranges you specify: sum var1 in 1/20 will summarize only the first 20 cases of var1
describe (des)	produces a list of variables with format and labels
ds	produces a list of variables without format or labels
codebook	will produce codebook of variable information (takes time!)
inspect	will provide mini-histograms of data
summarize (sum)	will provide summaries of data (means, sds, numbers of cases, max, min); sum varlist,detail will provide breakdowns by quartiles
lsum	(If installed): lsum varlist will give you summary only for cases that are not missing for any variable.
list varlist	will print out all values of variables in varlist
sort	sorts variables in ascending order
gsort	(If installed): sorts variables in ascending or descending order
order	changes the order of variables to your preferences
aorder	(If installed): orders variables alphabetically

recode	recode variables, as in: recode var1 3=2 2=1 1=0 4=. Note that var1 is recoded to missing (see mvdecode). Also note that you can only recode one variable at a time in a recode command (but see "for" below).
mvencode	mvencode changes all occurrences of missing to # in the specified varlist (see below).
mvdecode	mvdecode changes all occurrences of # to missing in the specified varlist–e.g., mvdecode var1-var99,mv(999) changes all instances of 999 in var1 through var99 to system missing.
generate (gen)	create new variables, the equivalent of compute in SPSS. E.g., gen pcincome=income/pop where pop=population. A neat feature is: gen byte democrat=(partyid==1) creates a dummy variable "democrat" that is equal to 1 if partyid==1 and 0 otherwise (you generally need to recode missing values in the new dummy variable to be missing in the new variable), which is best done with the replace command. Note that the byte command makes the new variable a "byte" variable –occupying less space in the data set. You can also use generate with many functions, such as mean, while also using the command *by*. E.g., gen incstate=income,by(state) which will give you a contextual variable of the income level in each state computed from income figures and the state code (in variable state).
egen	like gen (it stands for extensions to generate), egen handles some more complex functions. While gen x=mean(var1) will produce a constant containing the mean for var1, egen z=rmean(var2 var3 var4 var7) will produce a variable that has the row means for variables 2, 3, 4, and 7. rsum, e.g., gives the row sum of the variables in the list and rmiss gives the number of missing variables of variables in the list.

replace	like generate, but the variable already exists. E.g.: replace democrat=. if partyid==. More generally: gen byte inccat=0 if income < 10000 replace inccat=1 if income >= 10000 & income < 30000 replace inccat=2 if income >=20000 & income < 60000 replace inccat=3 if income >=60000 This will give you a four category (0, 1, 2, 3) variable for inccat (income category). Alternatively, if you have the *egen* function _gcut installed, you may type: egen byte inccat=cut income,at(0, 10000, 30000, 60000), which will accomplish the same thing as the above four commands.
rename	renames variables, as in ren var1 partyid If the old dataset simply names variables var1, etc., you can use rename to change their names. If installed, *renames* lets you rename many variables in one command.
label var "------"	creates variable labels, as in: label var partyid "party identification"
label define	defines value labels for variable, e.g.: label define partyid 1 Democrat 2 Independent 3 Republican Or: label define partyid 1 "dem identify" 2 "independ identify" 3 "repub identify"
label values	Use after label define, e.g., label values partyid partyid assigns the value label partyid to the variable partyid (label comes first, variable name second)

genl	*If installed:* Lets you generate a new variable and assign variable label in one command: genl pcincome=income/pop,label(per capita income)
merge	merges two data sets. There must be at least one common variable and both data sets must be sorted on that variable (and saved). If the common variable is xxx, the syntax is: merge xxx using mydata merge creates a variable called _merge that helps you to determine whether the merge worked correctly. _merge = 3 occurs when the observation is in both data sets (what you normally expect). _merge = 1 occurs when the observation only occurs in the master data set (in memory) and _merge=2 occurs when the observation only occurs in the data set that you are bringing in (mydata). If you expect that all observations are in both data sets, and some values of _merge are either 1 or 2, you have not merged correctly. Once you have completed the merge, you can delete the variable _merge. If you have already installed *mmerge*, it will sort the data for you and will drop the _merge variable.
drop	Use drop to delete variables or observations. drop var1 eliminates var1 drop var1 in 1500/2000 drops observations 1500-2000 for var1 drop in 1500/2000 drops observations 1500-2000 for *all* variables label drop partyid drops the label partyid
keep	the opposite of drop; using keep drops everything that is NOT in the keep command.
dropmiss	if installed: a neat utility that automatically drops all variables where all values are missing (don't laugh–a lot of surveys have questions such as this).

set more	"more" is the command that controls scrolling. set more on : output stops at the bottom of the window until you hit spacebar set more off : output scrolls continously (most useful if output is being directed to a log file and you don't need it to stop each page)
set matsize	set matsize sets the maximum number of variables that can be included in any of Stata's model-estimation commands. Sometimes you will run up against a "matsize too small" command if you are doing a particularly memory intensive task. Then you should type in something like: set matsize 500 STATA *may* refuse to do this, saying that data in memory would be lost. Then you need to clear the data set, reload it, and type in the command for matsize, and type in the command for your estimation again.
recast or compress	Used STAT Transfer on a SAS data set and forgot to invoke the "optimize" option? Then you have a huge data set, since SAS always saves data as double precision (lots of memory used). So what do you do? Use STATA's compress command to change variables to their smallest possible size. Simply type "compress" and hit enter. If you only need to change a few variables, use STATA's recast function to change a particular variable from double to byte: recast byte partyid

lookfor

A good enough reason to change from SPSS or SAS to STATA! Suppose that you think that you have some variables on party identification in your data set, but can't remember what you called them–but you know that they *either* have variable labels with "party" in them or are called party__. You type:

lookfor party
And STATA returns (as in describe)"

partyid	party identification
pid3	3-point party identification
giveprty	contributed money to political party

for

The mother of all STATA commands. It allows repetition of STATA commands.

The STATA manual states:

for listtype list : command containing_X

If listtype is	then list is a
varlist	varlist
newlist	new varlist
numlist	numlist (see help numlist)
anylist	list of words

Basic syntax examples:

```
. for var m*: replace X = X/10
. for new u1-u10: gen X = uniform()
. for num 1/5:  count if freq==X
. for any . -1:  replace x = . if y==X
```

In each example, elements from the list are substituted one at a time for the capital X and the command executed.

Or, in English:

for var var1 var2 var3 var4 partyid: recode X 4=3 3=2 2=1
for num 1 2 3 4: recode varX 4=3 3=2 2=1
for new newvar1 newvar 2 newvar3 \ for var var1 var2 var3: gen X=Y/pop

The first for statement recodes four variables at a time using for. The second for statement recodes four variables at a time when these variables have similar names (var1, var2, var3, var4): You can do the same thing by writing:

for var var1-var4: recode X 4=3 3=2 2=1

The third statement creates four new variables (using the place holder Y) from four old variables (using the place holder X).

by repeats STATA commands. E.g.,

by partyid: regress y var1-var4

performs the regression of var1 through var 4 on y for each category of partyid. Note that you must first sort the data set on partyid–unless you have already installed *bys*, which automatically sorts the data for you. Note: Not all commands allow you to use either by or bys.

Key statistical commands:

regress	OLS regression
reg3	two-stage and three-stage least squares
factor	factor analysis
anova	analysis of variance
tab (tabulate)	cross tabs (for more complex tables, see the "tables" command): Note that tab has a very useful data manipulation feature: tab var1,gen(newvar) This tabulates var1 and creates new variables (newvar1 through newvarx) for the x categories of var1. So if you type: tab partyid,gen(newvar) you will get: newvar1 (label partyid==1) for Democrats newvar2 (label partyid==2) for Independents newvar3 (label partyid==3) for Republicans and you might want to type: for newvar1 newvar2 newvar3 \ for any "democrat" "independ" "republic": ren X = Y and then: for democrat independ republic: replace X = . if partyid==. Or, alternatively: for democrat independ republic: recode X 0 = . if partyid==.
tab1	univariate tabulate (for multiple variables)
probit	probit analysis
logit	logit analysis
oprobit or ologit	ordered probit/logit analysis
mlogit	multinomial logit
z or zz	programs for calculating marginal effects after probit or logit
m or mm	programs for calculating marginal effects after multinomial logit
tobit	tobit analysis
dtobit	tobit with marginal effects (or dtobit2, if installed)
sureg	seemingly unrelated systems of equations
corr	correlation
pwcorr	pair-wise correlations
pcorr	partial correlations
ttest	ttests
sparl	if installed: provides scatterplot of two variables with regression line drawn and R^2 and regression equation above the graph.

I've referred to files that you might install yourself. Where do you get these files? On the web, go to:

http://ideas.uqam.ca/ideas/data/bocbocode.html

which is a site maintained by Boston College Economics Professor Kit Baum. You can also find a link to it and other useful resources at (of coures):

http://www.stata.com

and you can get lots of useful files by subscribing to STATALIST, a daily listserv on STATA issues that you can find by going to STATA's web site (warning: if you join, you should select the digest option, lest you get 50 or so distinct messages each day).

Stata Quick Reference

This quick reference shows examples of the Stata commands illustrated in the Stata class and Stata learning modules. The Stata commands are preceded by a period (but do not type the period when you type the command). For example,

```
. list
```

lists out the observations for the datafile in memory

Fundamentals of Using Stata (Part I)

- Starting and Stopping Stata
- Descriptive Information & Statistics
- Getting Help

Fundamentals of Using Stata (Part II)

- Exploring Data with Graphics
- Using "if" with Stata Commands
- Overview of Statistical Tests in Stata
- General Syntax of Stata Commands

Reading Data in Stata

- Using and Saving Stata data files
- Inputting data into Stata using the Stata Data Editor
- Inputting data into Stata
- Reading Dates into Stata and using date variables

Basic Data Management in Stata

- Labeling Data, Variables, and Values
- Creating and Recoding Variables
- Subsetting Variables and Observations

Advanced Data Management in Stata

- Collapsing data across observations
- Combining Stata data files
- Reshaping data from wide to long
- Reshaping data from long to wide

Making and Running do files

Fundamentals of Using Stata (Part I)

STARTING & STOPPING STATA

Starting Stata on the PC. Click Start, Programs, Stata, Intercooled Stata . At SSC you can Click Stata in Applications window.

Stopping Stata. Type exit in the command window.

DESCRIPTIVE INFORMATION & STATISTICS

```
. describe
```

provides information about the current data file, including the number of variables & observations and a listing of the variables in a data file

```
. codebook
```

produces codebook like information for the current data file.

```
. inspect
```

provides a quick overview of datafile.

```
. list model mpg
```

lists out the variables model and mpg

```
. count
```

counts the number of observations

```
. tabulate mpg
```

makes a table of mpg

```
. tabulate rep78 foreign
```

makes a two way table of rep78 by foreign

```
. summarize mpg price
```

produces summary statistics of mpg and price

```
. sort foreign
. by foreign: summarize(mpg)
```

produces summary statistics for mpg separately for foreign & domestic cars

```
. tabulate foreign, summarize(mpg)
```

produces summary statistics for mpg by foreign (prior sorting not required)

GETTING HELP

```
. help summarize
```

shows stata help page for the summarize command (also try the pulldown menu clicking help then Stata Command)

```
. search memory
```

searches online help for the keyword memory (also try the pulldown menu clicking help then search) Stata Web Site. See the Stata web site at http://www.stata.com OAC Web Site. See the OAC web site at http://www.oac.ucla.edu . From the home page, click Training & Consulting and then click Statistical Computing for information about OAC Classes, Consulting Services & Computing Services, including access to the online Stata Learning Modules . Also see the page with Resources to Help You Learn and Use Stata .

Fundamentals of Using Stata (Part II)

EXPLORING DATA WITH GRAPHICS

```
. graph weight
```

make a histogram of the variable weight

```
. graph weight , box
```

make a boxplot of the variable weight

```
. graph weight price
```

make a scatterplot of price & weight

```
. graph weight price, twoway oneway box
```

show scatterplot of weight and price with boxplots and oneway plots

```
. graph mpg weight price, matrix
```

show scatterplot matrix of mpg weight & price

```
. graph weight, by(foreign)
```

make separate histograms for foreign and domestic cars

the **by(foreign)** could be added to most graphs to get separate graphs

for foreign and domestic cars.

USING "if" WITH STATA COMMANDS

```
. summarize price if (rep78 >=4)
```

Creates summary statistics of price for observations where rep78 is 4 or greater

(and also missing if there is missing data).

```
. summarize price if (rep78 >=4) & (rep78 != .)
```

Creates summary statistics of price for observations where rep78 is 4 or

greater and rep78 is not missing.

```
. summarize price if (rep78 == 1) | (rep78 == 2)
```

Creates summary statistics of price for observations where rep78 is

1 or rep78 is 2.

OVERVIEW OF STATISTICAL TESTS IN STATA

```
. ttest price , by(foreign)
```

performs a t-test on price by foreign (which has 2 values, 0 and 1 representing domestic and foreign cars).

```
. tabulate rep78 foreign, chi2
```

performs a chi-square test of independence to see if rep78 is independent of foreign

```
. correlate price mpg weight rep78
```

displays correlations among price mpg weight and rep78 .

```
. pwcorr price mpg weight rep78, obs
```

displays correlations among price mpg weight and rep78 using pairwise deletion for missing data, and displaying the number of observations for each pair.

```
. regress price mpg weight
```

performs OLS regression analysis predicting price from mpg weight.

```
. oneway price rep78
```

performs analysis of variance with price as the dependent variable and rep78 as the indendent variable.

```
. anova price rep78 mpg weight, continuous(mpg weight)
```

performs analysis of variance with price as the dependent variable, rep78 as the
indendent variable, and .mpg weight as covariates.

GENERAL SYNTAX OF STATA COMMANDS

The summarize command is used to illustrate the general syntax of Stata commands.

```
[by varlist:] summarize [varlist] [if exp] [in range] ,
[options]
```

Examples

```
. summarize mpg
. summarize mpg if (weight < 2000)
. summarize mpg in 1/10
. summarize mpg , detail
. summarize mpg if (foreign == 1) , detail

. sort foreign
. by foreign: summarize mpg
```

Here are more examples illustrating the use of in

```
. summarize in 1
```
summary statistits for observation number 1
```
. summarize in 1/10
```
summary statistics for observation number 1 - 10
```
. summarize in -10/-1
```
summary statistics for 10th from last to last observation

Here are more examples illustrating the use of if

The [if exp] can be simple, like

```
. summarize if (mpg == 20)
```
if mpg is 20
```
. summarize if (mpg < 20)
```
if mpg is less than 20
```
. summarize if (mpg <= 20)
```
if mpg is leass than or equal to 20
```
. summarize if (mpg != 20)
```
if mpg is not 20 (*but see below if mpg has missing data*)
```
. summarize if (mpg > 20)
```
if mpg is greater than 20 (*but see below if mpg has missing data*)
```
. summarize if (mpg >= 20)
```
if mpg is greater than or equal to 20 (*but see below if mpg has missing data*)

If mpg has missing data, the [if exp] for !=, >, and >= should be written as below if you want the missing values to be excluded.

```
. summarize if (mpg != 20) & (mpg != .)
```

if mpg is not 20 *and mpg is not missing*

```
. summarize if (mpg > 20)  & (mpg != .)
```

if mpg is greater than 20 *and mpg is not missing*

```
. summarize if (mpg >= 20) & (mpg != .)
```

if mpg is greater than or equal to 20 *and mpg is not missing*

The [if exp] can be complex, using & and | to join parts together

```
. summarize if (foreign==1) & (mpg < 30)
```

summarize if foreign is 1 AND mpg under 30

```
. summarize if (foreign==1) | (mpg < 30)
```

summarize if foreign is 1 OR mpg under 30

Reading Data into Stata

USING AND SAVING STATA DATA FILES

```
. use auto
```

use the file called auto.dta in the current directory

```
. clear
```

clears out the existing data in memory

```
. use auto, clear
```

clears out the existing data in memory, then uses the file auto.dta

```
. save auto2
```

saves the data currently in memory to the file called auto2.dta

```
. save auto2, replace
```

saves the data currently in memory to the file called auto2.dta and replaces the file if it currently exists.

```
. set memory 2m
```

allocates 2 megabytes of memory for data in memory, permitting you to use a file up to 2 megabytes in size.

INPUTTING DATA USING THE STATA DATA EDITOR

Here are steps you can follow for using the Stata data editor...

```
. clear
```

Clears out any existing data.

```
. edit
```

Starts the Stata data editor. Once in the editor...

- Type in variables for the first observation (use tab to move across variables).

- Double click each column to supply a variable name and label.

- Enter all the rest of your data.

- When you are done, click the X in the top right of the window to close the window.

```
. save mydata
```

Saves the data from the data editor in a file called mydata.dta

INPUTTING DATA INTO STATA

```
. insheet using auto2.raw
```

reads in the comma or tab delimited file called auto2.raw taking the variable names from the first line of data.

```
. insheet make mpg weight price using auto3.raw
```

reads in the comma or tab delimited file called auto3.raw naming the variables mpg weight and price.

```
. infile str15 make mpg weight price using auto4.raw
```

reads in the space separated file named auto4.raw. The variable make should be surrounded by quotes if it has embedded blanks.

```
. infix str make 1-13 mpg 15-16 weight 18-21 price 23-26
using auto5.raw
```

reads in the fixed format file named auto5.raw

READING DATES INTO STATA AND USING DATE VARIABLES

Under Construction

Basic Data Management in Stata

LABELING DATA, VARIABLES, & VALUES

```
. label data "1978 auto data"
```

assign a label to the datafile currently in memory

```
. label variable foreign "origin of car, foreign or
domestic"
```

assign a label to the variable foreign

```
. label define labfor 0 "domestic car" 1 "foreign car"
. label values foreign labfor
```

create the value label labfor and assign it to the variable foreign

CREATING AND RECODING VARIABLES

```
. generate len_ft = length / 12
```

Create a new variable len_ft which is length divided by 12

```
. replace len_ft = length / 12
```

Change values of an existing variable named len_ft

```
. generate weight3 = .
. replace weight3 = 1 if (weight <=2640)
. replace weight3 = 2 if (weight >=2641) & (weight <=3370)
. replace weight3 = 3 if (weight >=3371) & (weight != .)
```

recode weight into weight3, having 3 categories, 1 2 3 using "replace if"

```
. generate weight3a = weight
. recode weight3a min/2650=1 2651/3370=2 3371/max=3
```

Recode weight into weight3a, having 3 categories, 1 2 3 using generate and recode.

```
. generate weightfd = weight
. recode weightfd min/3360=0 3361/max=1 if foreign==0
. recode weightfd min/2180=0 2181/max=1 if foreign==1
```

recode weight into weightfd, having 2 categories, but using different cutoffs for foreign and domestic cars

SUBSETTING VARIABLES AND OBSERVATIONS

```
. keep make mpg price
```

keeps just the variables make mpg and price for the data file in memory

```
. drop displ gratio
```

drops the variables displ and gratio for the data file in memory

```
. drop if (rep78 == .)
```

drops observations where the variable rep78 is missing for the data file in memory

```
. keep if (rep78 <= 3)
```

keeps observations where the variable rep78 is less than or equal to 3 for the data file in memory

```
. use make mpg price using auto
```

uses the file auto and reads in only the variables make mpg and price

```
. use auto if (rep78 <= 3)
```

uses the file auto and reads in only the observations where rep78 is 3 or less

```
. use make mpg price rep78 using auto if (rep78 <= 3)
```

uses the file auto and reads in only the variables make mpg price rep78 and only the observations where the variable rep78 is 3 or less

COLLAPSING DATA ACROSS OBSERVATIONS

```
. collapse age, by(famid)
```

Creates one record per family (famid) with the average of age within each family.

```
. collapse (mean) avgage=age avgwt=wt, by(famid)
```

Creates one record per family (famid) with the average of age (called avgage) and

average wt (called avgwt) within each family.

```
. collapse (mean) avgage=age avgwt=wt (count) numkids=age,
by(famid)
```

Same as above example, but also counts the number of kids within each family calling

that numkids. Assumes age is not missing.

```
. tabulate sex, generate(sexdum)
. collapse (sum) girls=sexdum1 boys=sexdum2, by(famid)
```

Counts the number of boys and girls in each family by using tabulate to create dummy variables

based on sex and then summing the dummy variables within each family.

COMBINING STATA DATA FILES

```
. use dads, clear
. append using moms
```

Appends the dads & moms files together by stacking them one atop the other.

```
. use dads, clear
. sort famid
. save dads, replace

. use faminc, clear
. sort famid
. save faminc, replace

. use dads, clear

. merge famid using faminc
```

Match merges the "dads" file with the "faminc" file on "famid". The four steps are...

1. sort "dads" on famid and save that file
2. sort "kids" on famid and save that file
3. use the "dads" file
4. merge the "dads" file with the "kids" file using "famid" to match them.

RESHAPING DATA FROM WIDE TO LONG

Wide format

Obs	famid	faminc96	faminc97	faminc98
1.	1	40000	40500	41000
2.	2	45000	45400	45800
3.	3	75000	76000	77000

Long Format

Obs	famid	year	faminc
1.	1	96	40000
2.	1	97	40500
3.	1	98	41000
4.	2	96	45000
5.	2	97	45400
6.	2	98	45800
7.	3	96	75000
8.	3	97	76000
9.	3	98	77000

```
. reshape long faminc, i(famid) j(year)
```

Changes the data from wide format with one record per famid to one record for every year (96 97 98) for every famid.

```
The general syntax of "reshape long" can be expressed as...

. reshape long <stem of wide vars>, i(<wide id var>) j(<var
for suffix>)
```

where

<stem of wide vars> is the stem of the wide variables, e.g. faminc

<wide id var> is the variable that uniquely identifies wide observations, e.g. famid

<var for suffix> is the variable that will contain the suffix of the wide variables, e.g. year

RESHAPING DATA FROM LONG TO WIDE

Long format

Obs	famid	birth	age
1.	1	1	9
2.	1	2	6
3.	1	3	3
4.	2	1	8
5.	2	2	6
6.	2	3	2
7.	3	1	6
8.	3	2	4
9.	3	3	2

```
Wide format
Obs     famid     age1     age2      age3
  1.          1           9           6           3
  2.          2           8           6           2
  3.          3           6           4           2

. reshape wide age, j(birth) i(famid)
```

Changes the data from long format with one record per kid to long format with one record per famid.

The general syntax of "reshape wide" can be expressed as...

```
. reshape wide <long var(s)>, i(<wide id var>) j(<var with suffix>)
```

where

<long vars> is the name of the long variable(s) to be made wide, e.g. age

<wide id var> is the variable that uniquely identifies wide observations, e.g. famid

<var with suffix> is the variable from the long file that contains the suffix for the wide variables, e.g. age

Other

MAKING AND RUNNING DO FILES

```
. do test1.do
```

executes the stata commands in test1.do, displays output

```
. run test1.do
```

executes the stata commands in test1.do, but displays no output

```
. doedit test1.do
```

brings up do file editor with test1.do in it

```
. doedit
```

brings up do file editor with an empty file

A typical do file, say it is called test.do, may look like this...

```
log using test.log , replace
```

<do file commands here>

```
log close
```

Stata Class Notes

Create and Modify Variables

1.0 Stata commands in this unit

. generate
. replace
. recode
. egen

2.0 Demonstration and Explanation

2.1 Create and modify variables

```
. use hsb2, clear
. generate total = read + write
. summarize total
. replace total = total + 2*math
. summarize total
. generate sex = gender
. tabulate sex
. recode sex 1=0 2=1
. tabulate sex
```

The generate command allows you to create new variables. The replace command allows you to change an existing variable. The recode command allows you the change specific values of the variables.

2.2 Egen

```
. egen zread = std(read)     * standard scores for read
. list read zread
. summarize read zread
. egen rmean = mean(read),by(ses)     * mean read for each ses
. list ses read rmean
. egen mread = median(read), by(prog)     * median read for each prog
. list prog read mread
. egen rread = rank(read)     * rank for read
. list read rread
```

egen stands for extended generate and is an extremely powerful command that has many options for creating new variables. Only a few of these options are demonstrated above. Here is a list of some of the other options:

Egen Functions

count	number of non-missing vlaues
diff	compares variables, 1 if different, 0 otherwise
fill	fill with a pattern
group	creates a group id from a list of variables
iqr	interquartile range
ma	moving average
max	maximum value
mean	mean
median	median
min	minimum value
pctile	percentile
rank	rank
rmean	mean across variables
sd	standard deviation
std	standard scores
sum	sums

3.0 Try the commands on your own

. generate tot = read + write + math
. summarize tot
. replace tot = read + math + science
. summarize tot
. generate newprog = prog
. recode newprog 1/3=2 2=1
. tabulate nprog
. egen aread = mean(read),by(prog)
. list prog read aread

6.0 Web Notes

The Stata Class Notes are available on the World Wide Web by visiting ...
http://www.oac.ucla.edu/training/stata/notes/

The dataset hsb2.dta can be loaded directly into Stata, over the Internet, using the following commands:
use http://www.oac.ucla.edu/training/stata/notes/hsb2

10 Jun 1999 - pbe
updated 06/24/99

Stata Class Notes

Creating Your Own Datasets

1.0 Stata commands in this unit

By now you know that we will not be showing all of the options for any of the commands.

. clear
. edit
. save
. infile
. insheet

2.0 Demonstration and Explanation

. clear

> **The clear command clears out the dataset that is currently in memory. We need to do this before we can create or read a new dataset.**

2.1 A small dataset

name	midterm	final
Smith	79	84
Jones	87	86
Brown	91	94
Adraktas	80	84

2.2 Creating a dataset using the 'Data Editor'

. edit

> **The edit command opens up a spreadsheet like window in which you can enter and change data. You can also get to the 'Data Editor' from the pull-down 'Window' menu or by clicking on the 'Data Editor' icon on the tool bar.**
>
> **Enter values and press return. Double click on the column head and you can change the name of the variables. When you are done click the 'close box' for the 'Data Editor' window.**

. save grades
. save grades, replace

The save command will save the dataset as grades.dta. Editing the dataset changes data in the computer's memory, it does not change the data that is stored on the computer's disk. The replace option allows you to save a changed file to the disk, replacing the original file.

. list
. summarize

Let's list the contents and run some statistics on the new data set

2.3 Creating a dataset from an ASCII file

. clear
. type ascii.raw
. infile str10 name midterm final using ascii.raw
. list

The infile command is used to read data from an external ascci file. The names of the variables are given followed by the keyword using which in turn is followed by the name of the file. str10 is not a variable name but indicates that name is a string variable up to 10 characters long.

The ASCII file called ascii.raw that looks like this:

```
"Smith"        79        84

"Jones"        87        86

"Brown"        91        94

"Adraktas"     80        84
```

2.4 Creating a dataset from a spreadsheet or database

. clear
. type spread.raw
. insheet using spread.raw
. list

> **The insheet command is used to read data from a file created by a spreadsheet or database program. The values in the file must be either comma or tab delimited. The names are included in the file.**
>
> **The spreadsheet file called spread.raw that looks like this:**

```
name,midterm,final

Smith,79,84

Jones,87,86

Brown,91,94

Adraktas,80,84
```

3.0 Try the commands on your own

. clear
. edit
. save grades
. save grades, replace
. list
. summarize
. clear
. infile str10 name midterm final using ascii.raw
. clear
. insheet using spread.raw

Stata Class Notes

Let's Get Organized

1.0 Stata commands in this unit

. order
. rename
. label data
. label variable
. label define
. label values
. replace
. recode
. note:
. notes
. save, replace

2.0 Demonstration and Explanation

Let's begin by using a new data set, schdat.dta, it looks like this:

id	a1	t1	gender	a2	t2	tgender
1	25	48	1	24	95	1
2	23	46	2	21	94	1
3	17	40	1	21	84	1
4	19	40	1	19	87	0
5	21	41	2	24	83	0
6	24	47	2	22	96	0
7	16	35	2	19	76	0
8	21	43	1	23	82	1
9	19	39	2	20	78	1
10	18	38	1	20	80	1

. cd A:\statacls
. use schdat, clear
. describe

> **The describe tells us the names of the variables but doesn't provide much more information. Here's the scoop on the data: a1 and a2 are scores on two assignments, t1 and t2 are the scores on the midterm and final respectively, and for gender, 1's are males and 2's are females. The variable tgender is the gender of the teacher and is scored 0 for male and 1 for female. None of this is obvious from looking at the data, so let's get organized.**

2.1 ordering & rename

```
. order id gender tgender a1 a2 t1 t2
. rename a1 assign1
. rename a2 assign2
. rename t1 midterm
. rename t2 final
```

The order command changes the order of the varibles. The four rename commands change the names of some of the variables to more meaningful ones. This is a good start but we really need to add some labels to make things clear

2.2 Some labels

```
. label data "Fall 1999 Stat 100 Scores"
. label variable gender "student gender"
. label variable tgender "teacher gender"
. generate total = assign1 + assign2 + midterm + final
. replace total = total/2
. label variable total "total score"
. describe
```

The label data command places a label on the whole dataset. The label variable command makes labels that help explain individual variables. replace replaces the value of total with the value of total/2. Next we need to get gender and tgender scored that same and assign value labels.

2.3 recode & more labels

Let's recode both gender and tgender so that 1 is male and 0 is female.

```
. recode gender 2=0
. recode tgender 0=1 1=0
. label define sex 1 "male" 0 "female"
. label values gender sex
. label values tgender sex
. describe
. tab1 gender tgender
. tab1 gender tgender, nolabel
```

The recode command allows us to code both gender and tgender the same way. The label define command creates a definition for the values 0 and 1 called sex. The label values command connects the values defined for sex with the values in gender and tgender.

2.4 Make a note of this

```
. note: gender is self-report
. note: the final was a take-home exam
. notes
. save schdat2, replace
. use schdat2, clear
```

> The note: (note the colon, ":") command allows you to place notes into the dataset. The command notes displays the notes. The save, replace saves the dataset as schdat.dta replacing the previous version.

3.0 Try the commands on your own

```
. cd A:\statacls
. use schdat, clear
. describe
. order id gender tgender a1 a2 t1 t2
. rename a1 assign1
. rename a2 assign2
. rename t1 midterm
. rename t2 final
. label data "Fall 1999 Stat 100 Scores"
. label variable gender "student gender"
. label variable tgender "teacher gender"
. generate total = assign1 + assign2 + midterm + final . label variable total "total
score"
. recode gender 2=0
. recode tgender 0=1 1=0
. label define sex 1 "male" 2 "female"
. label values gender sex
. label values tgender sex
. describe
. codebook gender tgender
. note: gender is self-report
. note: the final was a take-home exam
. notes
. save schdat2
```

22 Jun 1999 - pbe