ERP Reliability Analysis (ERA) Toolbox

created by

Peter E. Clayson

User Manual for Version 0.4.8

Last Modified 22 June 2018

https://github.com/peclayson/ERA_Toolbox/wiki

Contents

Introduction

Overview

The ERP Reliability Analysis (ERA) toolbox is an open-source Matlab package that uses generalizability (G) theory to evaluate the reliability of ERP data (Clayson & Miller, 2017a). The purpose of the toolbox is to characterize the dependability (G-theory analog of reliability) of ERP scores to facilitate their calculation on a study-by-study basis and increase the reporting of these estimates.

The ERA Toolbox provides information about the minimum number of trials needed for dependable ERP scores and describes the overall dependability of ERP measurements. All information provided by the ERA Toolbox is stratified by group and condition to allow the user to directly compare dependability (e.g., a particular group may require more trials to achieve an acceptable level of dependability than another group).

This document is a user manual for using the ERA Toolbox. For links to the most up-to-date version, bug reports, feature requests, and relevant publications, see the GitHub wiki.

Why Another Toolbox?

Reliability is a property of scores (the data in hand), not a property of measures. This means that P3, error-related negativity (ERN), late positive potential (LPP), (insert your favorite ERP component here) is not reliable in some "universal" sense (Clayson & Miller, 2017b). Since reliability is context dependent, demonstrating the reliability of LPP scores in undergraduates at UCLA does not mean LPP scores recorded from children in New York can be assumed to be reliable. Measurement reliability needs to be demonstrated on a population-by-population, study-by-study, component-by-component basis.

The purpose of the ERA Toolbox is to facilitate the calculation of dependability estimates to characterize observed ERP scores (Clayson & Miller, 2017a). ERP psychometric studies have been useful in suggesting cutoffs and characterizing the overall reliability of ERP components in those studies. When designing a study, that information can help guide decisions about, for example, the number of trials to present to a participant for a given population. However, just because the observed data meet the previously recommended trial cutoff does not mean that the ERP scores are necessarily reliable. ERP score reliability cannot be inferred from trial counts, despite that trial counts and reliability are certainly related.

My hope is that the ERA Toolbox will make it easier to demonstrate the reliability of ERP scores on a study-by-study basis.

Mismeasurement of ERPs leads to misunderstood phenomena and mistaken conclusions. Poor ERP score reliability from mismeasurement compromises validity. Improving ERP measurement, by ensuring score reliability, can improve our trust of the inferences drawn from observed scores and the likelihood of our findings replicating.

For a detailed discussion of reliability as it relates to ERP scores, see Clayson and Miller (2017b). Clayson and Miller (2017b) also highlights some of the contextual factors that can influence ERP score reliability.

For discussions of generalizability theory as it relates to ERP scores, see Clayson and Miller (2017a) and Baldwin, Larson, and Clayson (2015). Clayson and Miller (2017a) provides a walkthrough of the statistical formulas used in the ERA Toolbox and demonstrates how the information provided by the toolbox can be used in decision making.

For published papers that have called for evaluation of ERPs on a study-by-study basis, see Clayson and Miller (2017a, 2017b), Hajcak, Meyer, and Kotov (2017), and Infantolino, Luking, Sauder, Curtin, and Hajcak (2018).

Getting Started

The ERA Toolbox can be downloaded here. The toolbox has been tested using Matlab version 2017a on Mac OS X High Sierra. Once you have downloaded the toolbox, add the directories to your Matlab path (use 'Add with subfolders…').

The toolbox has three dependents: MatlabStan, CmdStan, and MatlabProcessManager. The wiki containing instructions for downloading these can be found here. The current version of the toolbox has been tested using MatlabStan (v 2.15.1.0), CmdStan (v 2.17.0), and MatlabProcessManager (v 0.5.1).

Installing these dependents can be a headache and is not very straightforward as it the instructions are spread across a few places. The toolbox is able to install the dependents for Mac OS 10.10 and above and for Windows 7. For Mac users, input will be required to install XCode command line tools. For Windows users, Rtools will need to be installed already (see Appendix A – Installation).

In order to help streamline the installation process, should you need to do so manually, I've written instructions in Appendix A – Installation.

Getting Help

If you run into any problems, check the wiki FAQ section and GitHub issues page for help. If there is not an answer to your question on the issues page, feel free to post a new issue. When posting an issue, please describe how you encountered the error and then copy and paste the Matlab error into the issue. Please also indicate the Matlab version that you are using and the version of the ERA Toolbox (which can be found at the startup of the toolbox).

I will do my best to respond as quickly as I am able. I hope that by using the GitHub issues page, that people will be able to find answers more quickly (if it's a previously encountered problem) than emailing me directly. But, if you would prefer to email me, my email is peter.clayson@gmail.com.

The GitHub issues page is also a great place to recommend other features for the toolbox. I would love to hear any suggestions that you have!

Acknowledgements

Many people have helped me, whether it be directly or indirectly, with making this toolbox.

First, I have to thank Scott A. Baldwin for developing the dependability formulas for analyzing ERP scores (Baldwin et al., 2015). In addition to developing these formulas, he has been integral in the development of my own understanding of generalizability theory. He has also provided his own recommendations of what I should include in the toolbox and answered many questions along the way.

I also need to thank Gregory A. Miller for encouraging me to develop the toolbox. We wrote a paper on the importance of assessing reliability on a study-by-study

basis and provided guidelines for how reliability should be reported in ERP studies (Clayson & Miller, 2017b). As part of this paper, Dr. Miller recommended that we include code for assessing dependability estimates. That ended up ultimately leading to another paper dedicated to generalizability theory and the ERA Toolbox (Clayson & Miller, 2017a).

Lastly, I must thank Michael J. Larson for encouraging me to develop the toolbox, but especially for encouraging me to develop a GUI for the toolbox. I remember being at a conference and discussing the toolbox with Dr. Larson. He assumed it had so many features that I had not even thought about implementing at that point. I'm still working on implementing some of them! Dr. Larson also helped beta test the software for me. Actually, he is the one that introduced me to the importance of measurement.

Now that I have thanked Drs. Baldwin, Miller, and Larson, I need to say that although I have accepted a lot of advice and feedback from them, I wrote the code for the toolbox. Thus, any bugs or errors are my responsibility, and I take full credit for them. If you encounter any problems or have any questions about the mechanics, please feel free to contact me (peter.clayson@gmail.com).

Citations

When using the ERA toolbox, please cite the ERA Toolbox paper (Clayson & Miller, 2017a), Baldwin et al. (2015) for the formulas using generalizability theory and CmdStan (Stan Development Team, 2017) and Stan for the actual crunching (Carpenter et al., 2017). I think that it is important to cite both CmdStan and Stan, as 1) the toolbox would not be as powerful without them and 2) citing them helps the Stan Development Team get more funding, which means more features for us to use!

Example sentence to include in paper:

> "Dependability estimates were calculated following the formulas provided by Baldwin et al. (2015) using the ERP Reliability Analysis (ERA) Toolbox v 0.4.5 (Clayson & Miller, 2017a). The ERA Toolbox used CmdStan v 2.17.0 (Stan Development Team, 2017) to implement the statistical models in Stan (Carpenter et al., 2017)."

Another example of how to cite the toolbox that also includes how to report dependability estimates can be found in Clayson and Miller (2017a).

License

The ERA Toolbox is covered by the GNU General Public License.

Analyzing Data

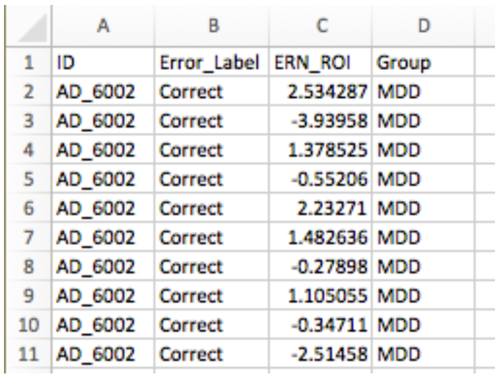
Preparing Data

The ERA Toolbox works on single-subject, single-trial ERP measurements. It does not matter what kind of measurement is used (e.g., latency v amplitude). Before using the toolbox, create a spreadsheet with all of the single-subject, single-trial measurements.

Use the long format for all inputted data. Each line should correspond to one trial, and there can be multiple measurements per line (i.e., you could have multiple ERP measurements for a given trial). When participants do not have good data for a given trial, that trial can simply be excluded, and it is very likely that there will be a different number of trials for each participant. It is also important that there are no missing data in the spreadsheet, otherwise the data will not be able to run in Stan.

Note: When including multiple event types, each participant should have at least one measurement per event type. If a participant does not have at least one measurement for each event type, the participant needs to be removed from the data file before loading the file into the toolkit. Similarly, if using multiple groups, each participant still needs to have at least one measurement for each event type.

At the very least, each line should have an ID variable and a measurement variable. When processing the data in the toolbox, you will be asked to specify which variable is which. Additionally, you can have columns that indicate group membership and event type, although they are not required columns.



	A	B	C	D
1	ID	Error_Label	ERN_ROI	Group
2	AD_6002	Correct	2.534287	MDD
3	AD_6002	Correct	-3.93958	MDD
4	AD_6002	Correct	1.378525	MDD
5	AD_6002	Correct	-0.55206	MDD
6	AD_6002	Correct	2.23271	MDD
7	AD_6002	Correct	1.482636	MDD
8	AD_6002	Correct	-0.27898	MDD
9	AD_6002	Correct	1.105055	MDD
10	AD_6002	Correct	-0.34711	MDD
11	AD_6002	Correct	-2.51458	MDD

This is an example of how your data could be set up. There are columns for the subject ID (ID), group membership (Group), event type (Error_Label), and ERP amplitude measurement (ERN_ROI).

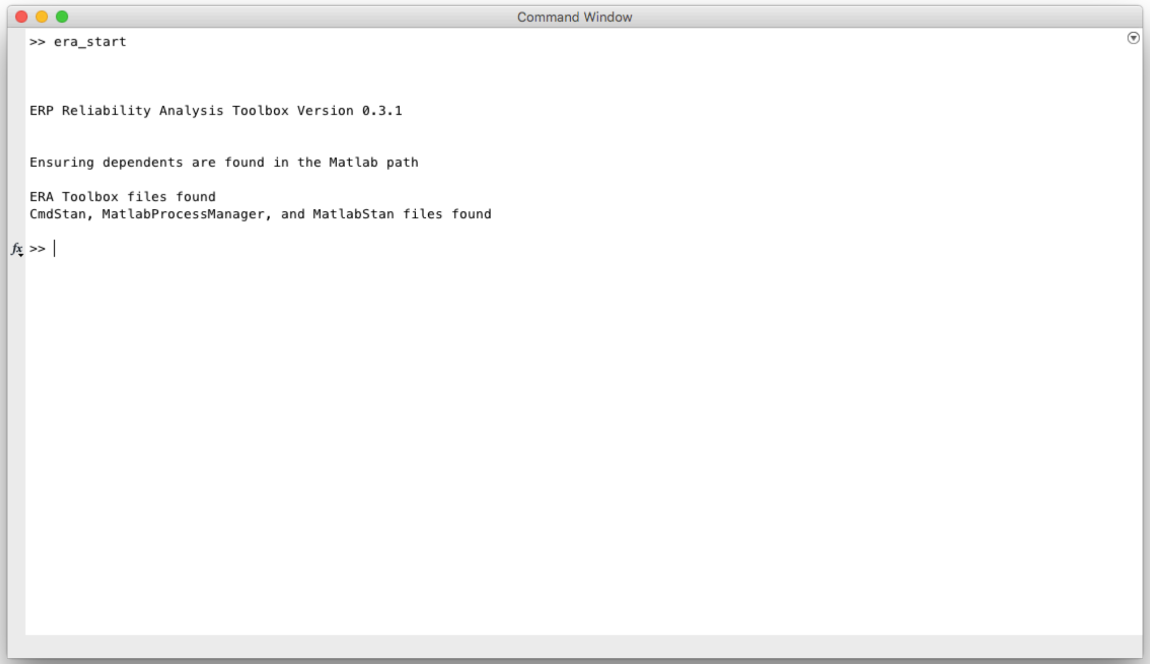
The following file formats are supported by the toolbox.

- .xlsx – Excel File
- .xls – Excel File 97-2003
- .csv – comma-separated value file
- .dat – tab-delimited text
- .ods – OpenDocument spreadsheet

The dataset used throughout this manual was presented in Baldwin et al. (2015).

Start the Toolbox

In order to start the ERA Toolbox, simply type era_start in the Matlab command window and click Enter.



```
>> era_start



ERP Reliability Analysis Toolbox Version 0.3.1


Ensuring dependents are found in the Matlab path

ERA Toolbox files found
CmdStan, MatlabProcessManager, and MatlabStan files found

>> |
```

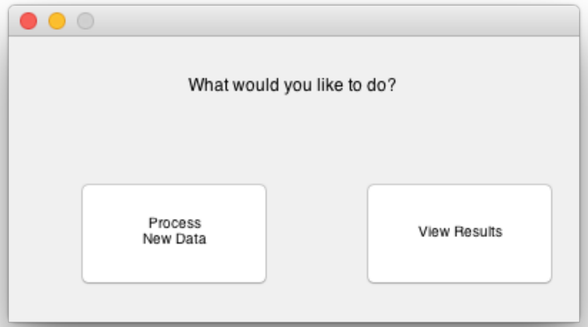
After clicking Enter, you will see some information printed in the Command Window.

The first thing printed is the name of the toolbox and version number. (This is where you can get the version number for reporting bugs/issues.)

The toolbox will then check to ensure that the dependents are located in your Matlab path. If they are not found, they will be added to your Matlab path.

If the toolbox could not locate CmdStan, MatlabProcessManager, or MatlabStan, then you need to follow the instructions here for downloading and installing them. Alternatively, the toolbox will attempt to download and install them for you. This requires little user interaction for Mac users, but it requires a bit more work from Windows users. Linux is not supported at this time for the automatic installation of the dependents.

If everything goes smoothly you will be presented with a gui that requires you to indicated whether you would like to Process New Data or View Results.

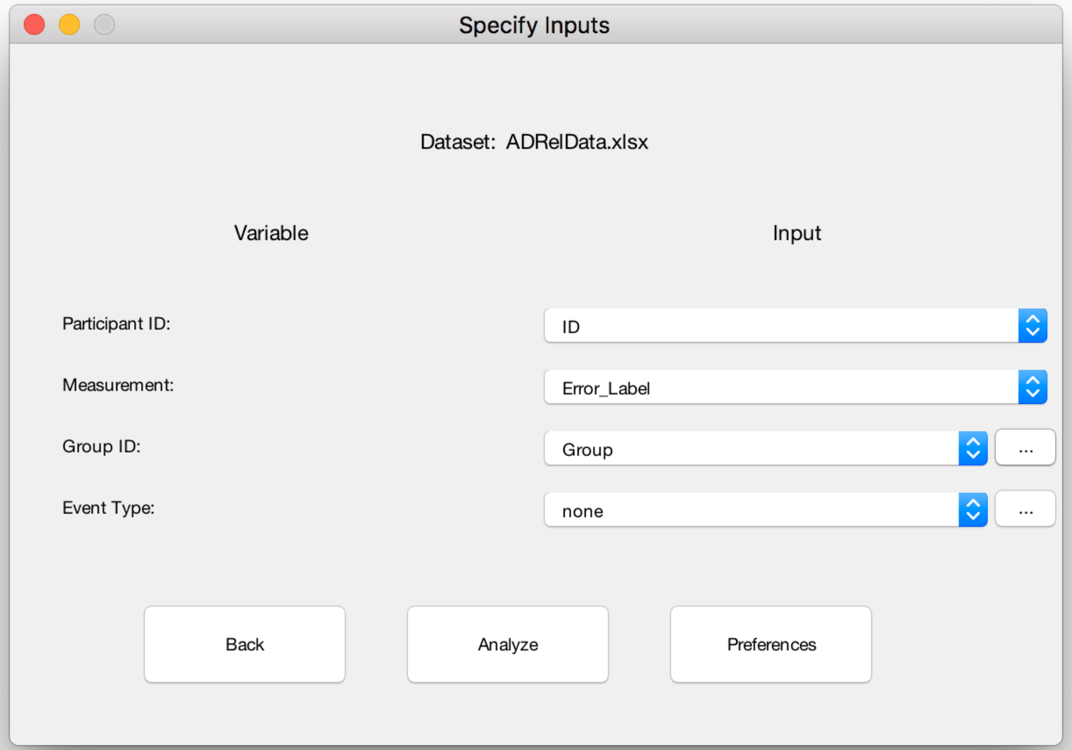


Process New Data: If data have not yet been analyzed by the toolbox, select this option to begin analyzing the data in CmdStan.

View Results: If the data have already been processed in the toolbox, select this option to summarize and view the data.

Process New Data

Selecting the Process New Data button will bring up a window to choose the file that is to be processed. After selecting a file, a gui will pop up allowing you to specify which columns correspond to which variables.



Dataset: ADRelData.xlsx

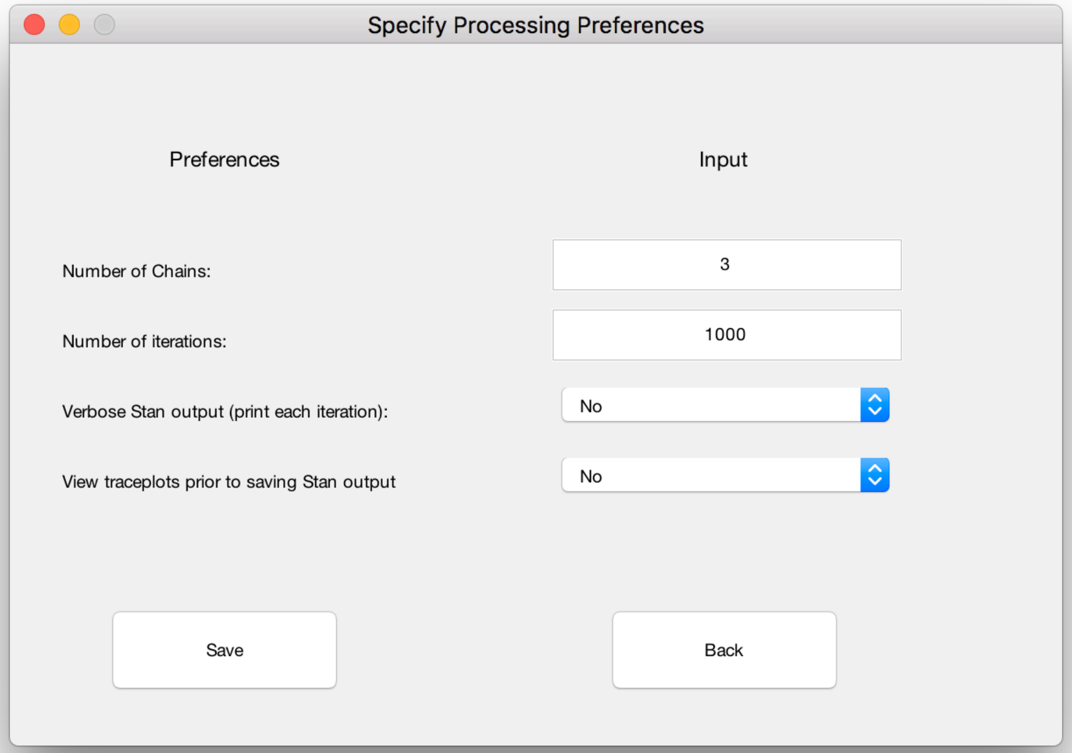
Variable	Input
Participant ID:	ID
Measurement:	Error_Label
Group ID:	Group
Event Type:	none

The gui will display the filename of the loaded dataset. This screen will allow you to select which columns from the spreadsheet belong to which variable: Participant ID, Measurement, Group ID, and Event Type. The columns in your data will be populated in the pulldown lists on the right.

At the very least, the Participant ID and Measurement rows must be populated with column names (notice ‘none’ is not an option in the pull-down menus for Participant ID and Measurement). If there are not columns indicating group membership or event type in the data, then select ‘none’ from the pulldown list.

If you click on the buttons next at the end of the ‘Group ID’ and ‘Event Type’ rows, you will be provided with the option to select a subset of groups or events, respectively. It is important to specify which columns contain the group or event information prior to clicking these buttons. Otherwise, you will get an error if ‘none’ is selected, or you will get a long list of values if a measurement column is selected.

Once you have selected which columns contain the data to be analyzed, select the Preferences button.


Specify Processing Preferences

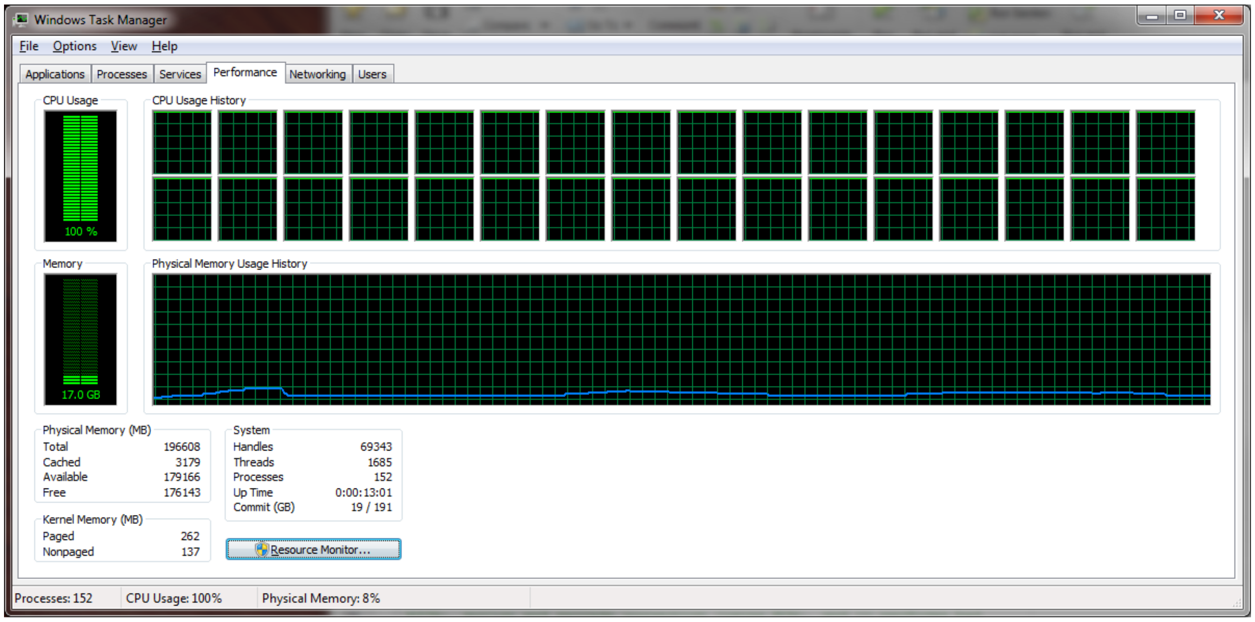
Preferences	Input
Number of Chains:	3
Number of iterations:	1000
Verbose Stan output (print each iteration):	No
View traceplots prior to saving Stan output	No

There are three preferences to choose from for processing.

- Number of Chains: Specifies the number of chains to run for the Markov Chain Monte Carlo estimation procedure in CmdStan. You need at least 3 chains in order to properly assess convergence (the toolbox will not allow you to choose fewer than 3). You can do more if you would like. CmdStan will run the chains in parallel.

  Holding the number of iterations constant, increasing the number of chains will increase the number of draws that go into your estimates. Thus, increasing the number of chains can increase the precision of your estimates.

CmdStan will run each chain in parallel on your computer. If you have a computer with 8 cores (and the RAM to support the computation), you can use more than 3 chains (up to 8 in parallel). CmdStan will not let you run more chains than you have cores in parallel. If you specify more chains than you have cores, CmdStan will run as many as it can in parallel, then execute the other chains when cores free up. This is a quick way to push your computer to its limit. It's kind of fun ☺. As you can see from this beautiful screenshot, not exactly RAM heavy (~17GB for 32 models), but I use all of my processors… The RAM used depends heavily on the number of participants and trials in the dataset.



```
Physical Memory (MB)              System
Total          196608             Handles          69343
Cached           3179             Threads           1685
Available      179166             Processes          152
Free           176143             Up Time     0:00:13:01
                                  Commit (GB)   19 / 191
Kernel Memory (MB)
Paged             262
Nonpaged          137

Processes: 152    CPU Usage: 100%    Physical Memory: 8%
```

- Number of Iterations: Specifies the number of iterations for each chain. There is not a good way to estimate a priori the number of iterations that you need to get estimates that converge. If your model is having trouble converging, then you need to increase the number of iterations. The toolkit's default is 1,000. That is a pretty arbitrary number. I frequently use 10,000, which gives me good convergence for the ERP data I've run through the toolbox (but is usually overkill). Use as many iterations as you need to get your model to converge.

- Verbose Stan Output: If no, nothing will be printed to the command window while CmdStan is crunching. If yes, the progress of CmdStan crunching will be printed in the command window. The model being run will be printed to the command window and the progress of the crunching will be shown.

- View trace plots prior to saving Stan output: If yes, trace plots of the Stan parameters (mean, between-person variance, and within-person variance) will be plotted and the user will be given the option to rerun the Stan model. If no, the trace plots will not be displayed.

To save the preferences click Save. If you do not want to save the changes, just click Back.

Once you have set the preferences you want, click Analyze to run the model.

Analyze

After clicking Analyze, the model will be executed. CmdStan will translate the Stan program to C++ and compile the resulting C++ to an executable. The model may take a while to set up.

CmdStan will take a while to crunch depending on the size of the dataset and the number of iterations (and your computing power).

A few things to note.

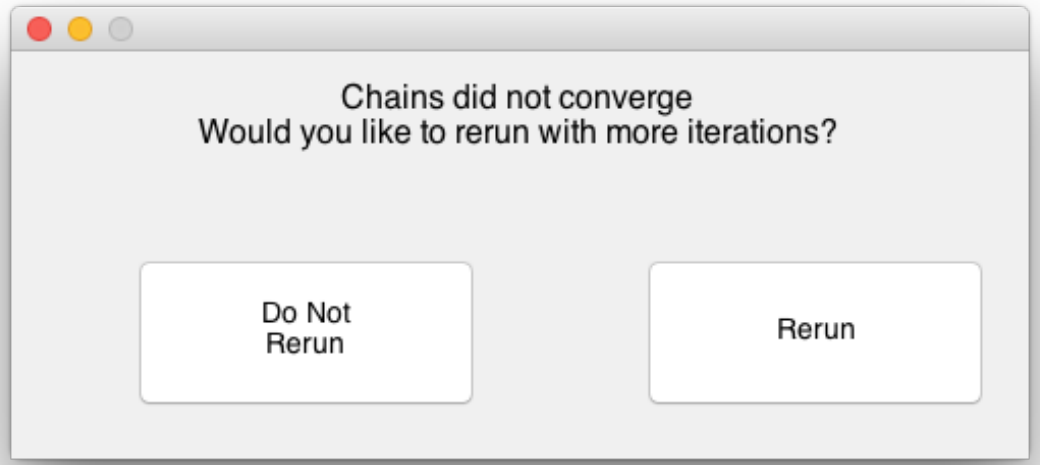
The models run in C++, not Matlab. Thus, terminating Matlab's processing (using ctrl+c) does not terminate the C++ models. If for some reason you need to terminate the processing, you need to use the task manager (Windows) or activity monitor (Mac) to terminate the CmdStan processes (there will be as many separate processes running as chains). You'll also need to use ctrl+c to terminate Matlab's processing. The Matlab command window will not take inputs while CmdStan is running the chains by design.

Convergence of the chains will automatically be checked after the model has finished processing. Convergence is assessed in two ways (see Gelman et al., 2013, pp. 281-286).

1. The potential scale reduction for the scalar estimands ($\hat{R}$) is checked to see whether it is lower than a threshold set at 1.1.

2. The effective sample size for each scalar estimand is checked to ensure it is greater than 10 times the number of chains.

If the convergence criteria are not satisfied, then a gui will pop up asking whether the model should be rerun with more iterations. If convergence criteria are satisfied, the user will be taken to the screen to setup which tables and plots to show for processed data.


Chains did not converge
Would you like to rerun with more iterations?

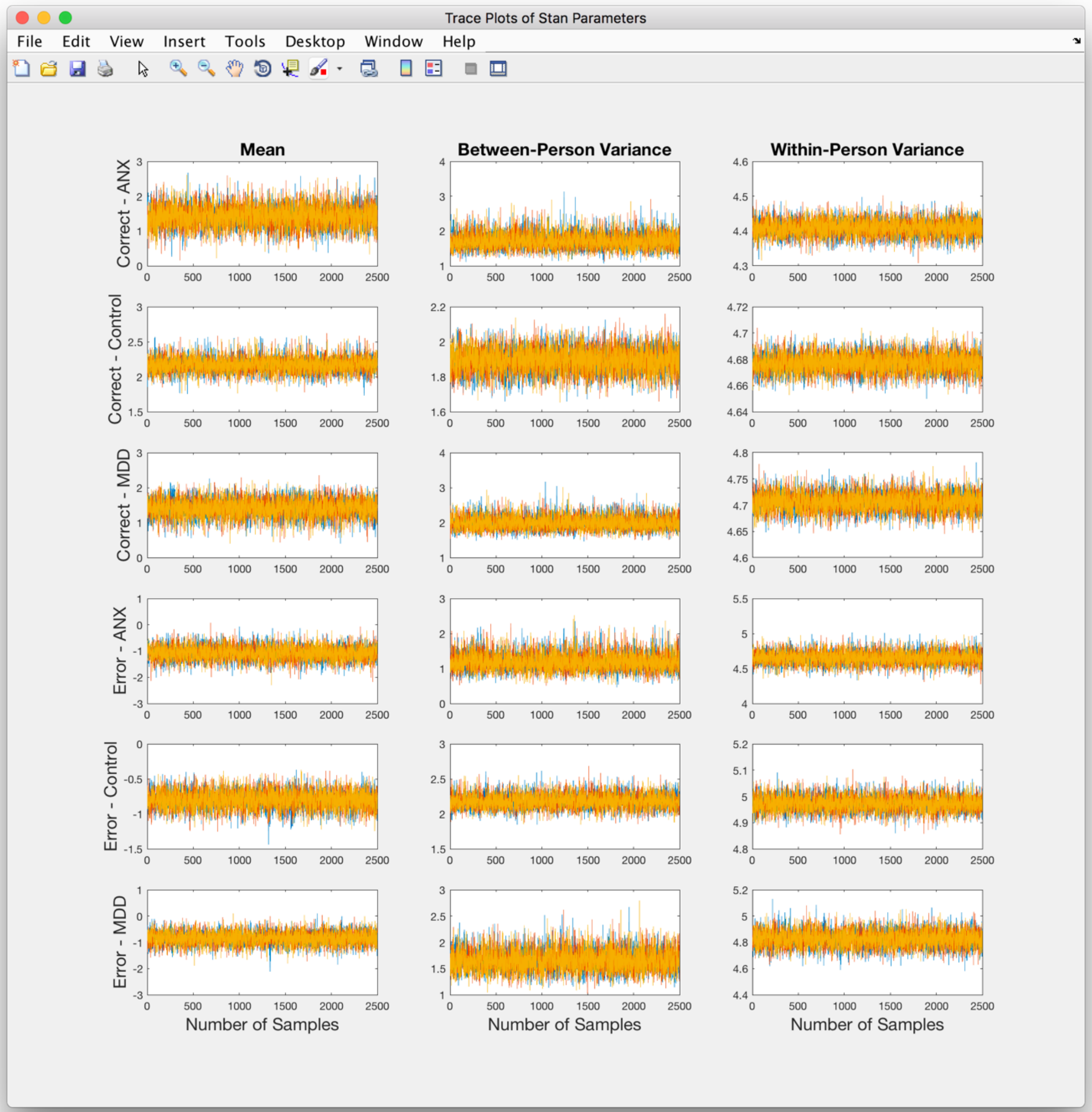
If Rerun is selected, then the toolbox (somewhat arbitrarily) will double the number of iterations are rerun the model. If the number of iterations after being doubled is not greater than 1,000, the number of iterations will be set to 1,000 for the next round of processing.

If Do Not Rerun is selected, the user will be taken back to the gui for specifying the inputs of the model.

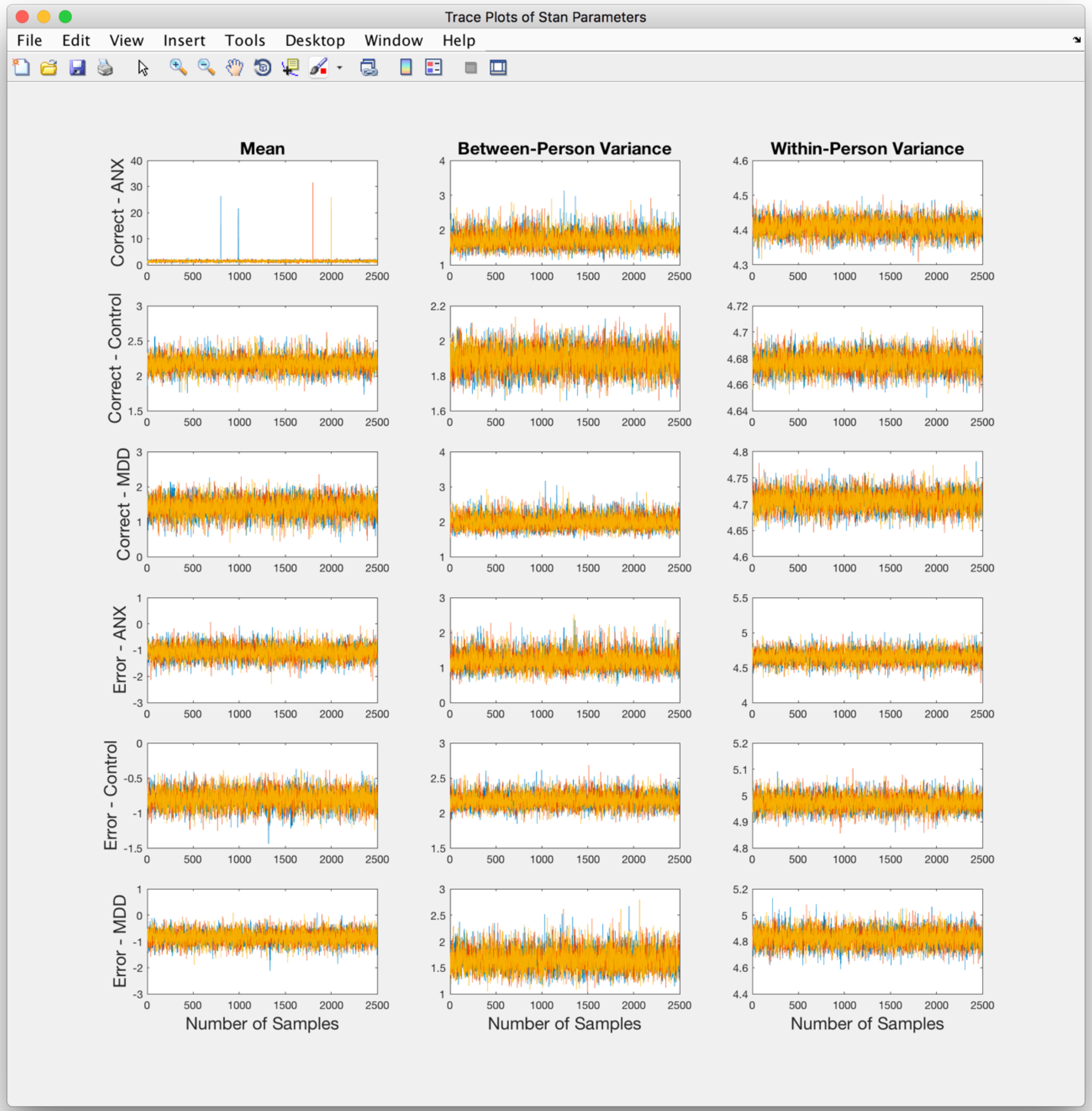
It is difficult to know how many iterations are necessary to satisfy the convergence criteria. If you have the time and computing power, it may be easier to select Do Not Rerun and greatly increase the number of iterations (increase to 10,000?). It can be troublesome to keep rerunning the model over and over again, only doubling the number of iterations each rerun.

Trace Plots

Viewing the trace plots provides a way to visually inspect convergence of the Markov chains. A trace plot shows the parameter values at each sample (i.e., iteration) for each chain. If the model has converged, the plots for each chain should cluster together or overlap. That is, these plots should look like a "fat, hairy caterpillar" (Lunn, Jackson, Best, Thomas, & Spiegelhalter, 2012).



These trace plots show good convergence of the chains. They look like "fat, hairy, caterpillars". None of the chains trails off on its own away from the other chains, and there are no aberrant, large spikes. The above model was run with 5,000 iterations. As a default, Stan uses the first half of the specified iterations to warm up. We end up with the second half of each chain to use, which is why each chain only had 2,500 samples.



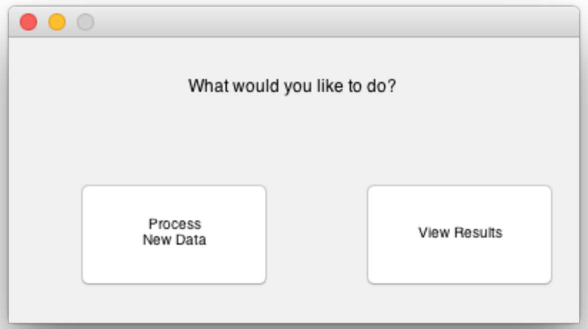
This is a brief example of how a trace plot could show non-convergence. In the top left plot there are large spikes that deviate from the rest of the data. This means that for that parameter the chains did not converge.

Viewing Data

There are two ways to get to the gui for specifying the inputs to view the data.

If the data have just been processed, the gui for specifying the inputs will automatically pop up once the convergence criteria have been met.

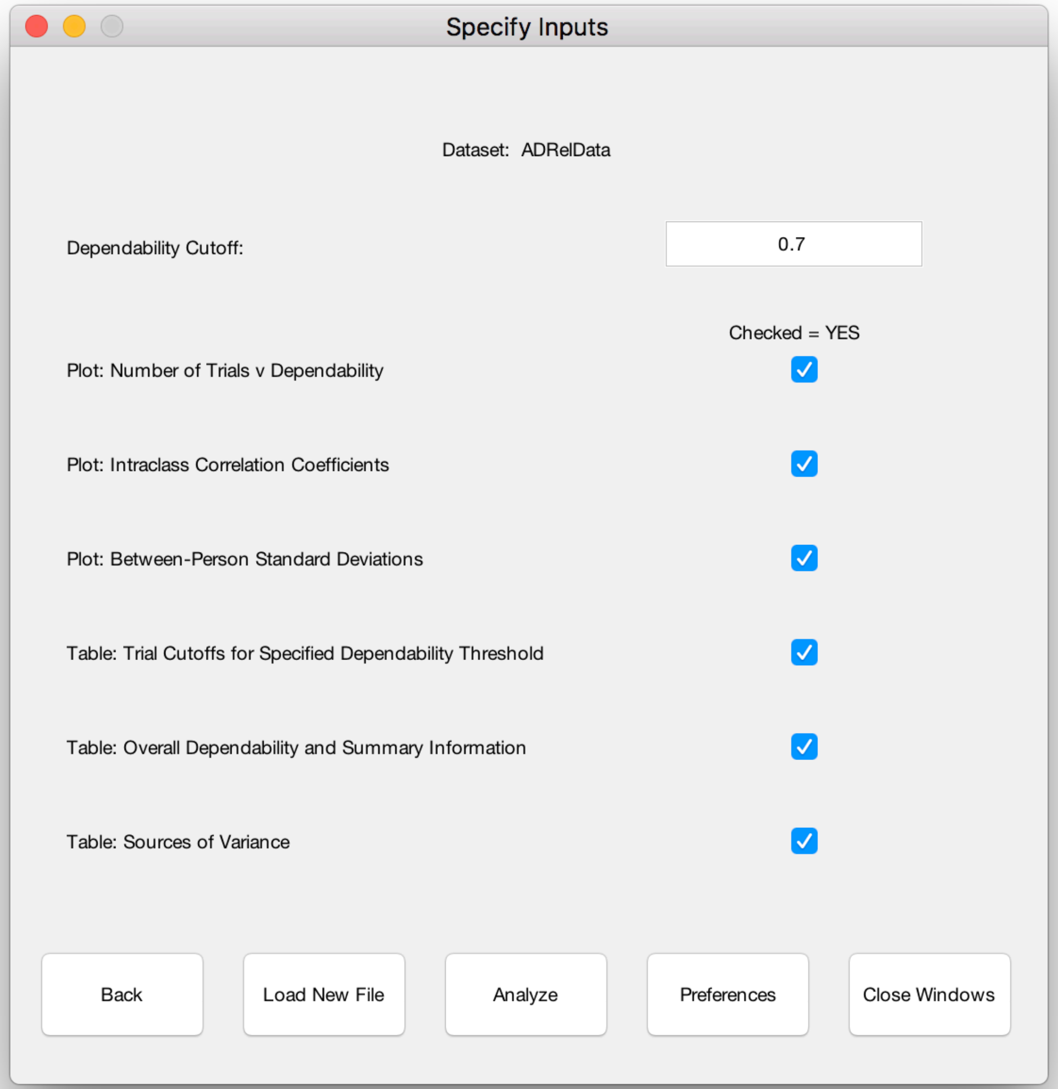
Alternatively, you can execute era_start in the Matlab command window and select View Results.



If View Results is selected, a window will pop up for selecting a data file.

Although a discussion of each output is provided below, Clayson and Miller (2017a) also provide a detailed discussion of each output as well as the algorithms that go into each output. There is also a discussion of how to make decisions about the data based on each output and the tradeoffs for various specifications. It is an excellent supplement to this manual. I encourage looking at the paper if you are unfamiliar with generalizability theory.

Specify Inputs



Dataset: ADRelData

Dependability Cutoff: 0.7

	Checked = YES
Plot: Number of Trials v Dependability	✓
Plot: Intraclass Correlation Coefficients	✓
Plot: Between-Person Standard Deviations	✓
Table: Trial Cutoffs for Specified Dependability Threshold	✓
Table: Overall Dependability and Summary Information	✓
Table: Sources of Variance	✓

There are many options for viewing the processed data. If the box is checked, that indicates that you would like to view that table or plot. If applicable, data will be stratified by group, event, or group and event. More information about each plot is provided in its own section below.

Tip: For more information about any of the inputs, hover your mouse over the text and a brief description will be displayed.

- Dependability Cutoff: Specify the threshold for acceptable dependability. The default threshold is .80. For this analysis a dependability threshold of .70 is used. This is the cutoff that the toolbox will use to search for the number of trials that are needed for dependable estimates. For guidelines and recommendations, as well as the deleterious effects that reliability can have on statistics, see Clayson and Miller (2017b).

- Plot: Number of Trials v Dependability: This produces a plot that shows the relationship between the number of trials included in an average and the dependability estimate.

- Plot: Intraclass Correlation Coefficients: This will display a plot that visualizes the intraclass correlation coefficients (ICCs).

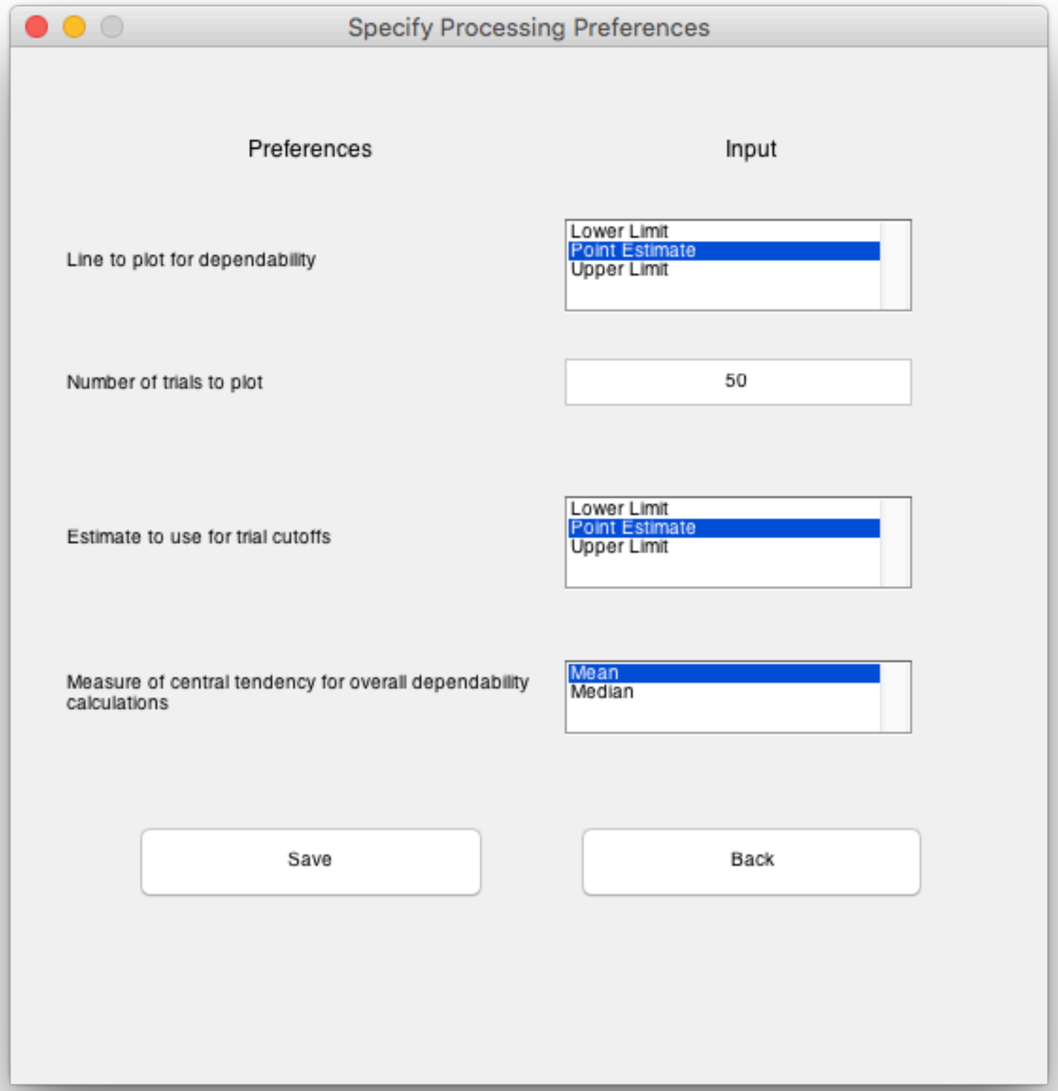
- Plot: Between-Person Standard Deviations: Provides a plot showing the between-person standard deviations.

- Table: Trial Cutoffs for Specified Dependability Threshold: This will show a table with the number of trials needed to achieve the specified dependability threshold and the associated dependability point estimate with a 95% credible interval (aka, fiducial trust limits; these are similar to confidence intervals).

- Tale: Overall Dependability and Summary Information: This table contains most of the summary information for the data. It includes the number of participants who met the threshold for each given event. It also indicates the number of participants who did not meet the threshold. It includes the overall dependability of the data. Summary information (mean, minimum, and maximum) for the number of trials for participants with “good” data (those participants with data that met the dependability threshold) is also shown.

- Table: Sources of Variance: This will produce a table containing the point estimates and 95% credible intervals for the between-person standard deviation, within-person standard deviation, and ICCs.

Selecting Preferences will bring up various options for viewing the data.



Specify Processing Preferences

Preferences	Input
Line to plot for dependability	Lower Limit / **Point Estimate** / Upper Limit
Number of trials to plot	50
Estimate to use for trial cutoffs	Lower Limit / **Point Estimate** / Upper Limit
Measure of central tendency for overall dependability calculations	**Mean** / Median

- Lines to plot for dependability: Specifies the line to be plotted for the Number of Trials v Dependability plot. Options include the dependability point estimate and the lower and upper limits of the 95% credible interval for the dependability estimate.

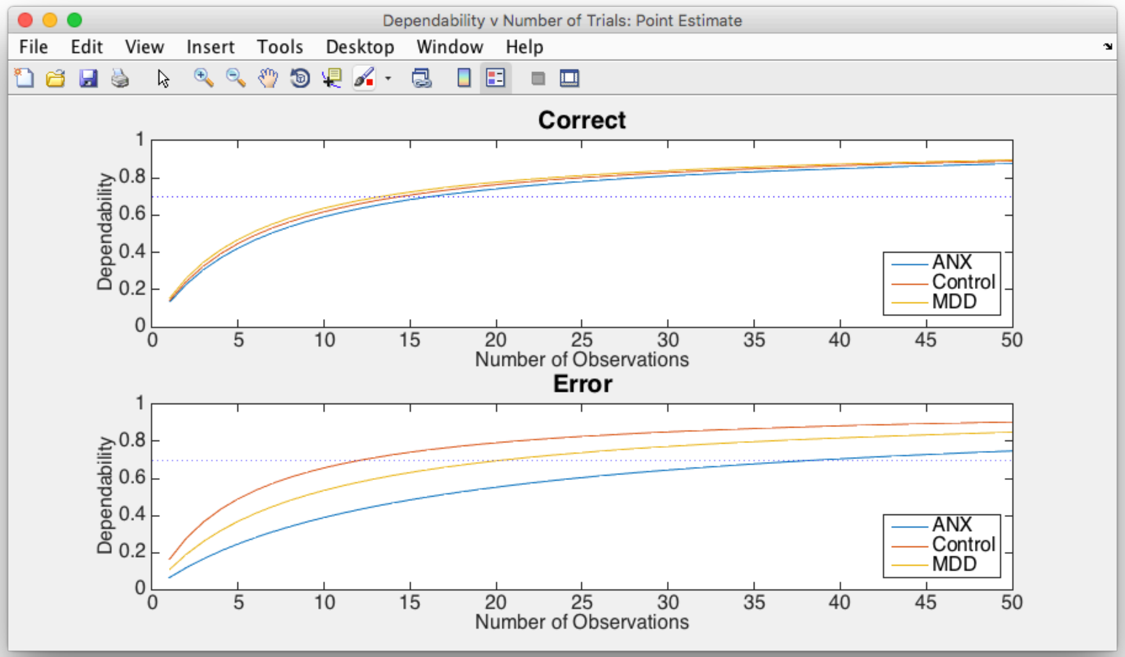
- Number of trials to plot: Indicate the number of trials (x-axis) that should be plotted for the Number of Trials v Dependability plot.

- Estimate to use for trial cutoffs: Specifies which estimates to use for finding the number of trials to achieve acceptable dependability. Options include the dependability point estimate and the lower and upper limits of the 95% credible interval for the dependability estimate.

- Measure of central tendency to use for overall dependability calculations: Specify whether to use the mean or median number of trials in the calculation of the overall dependability estimates. The mean is generally preferred if the data are somewhat normally distributed. However, when the data are skewed, the median can be used.

Viewing Outputs

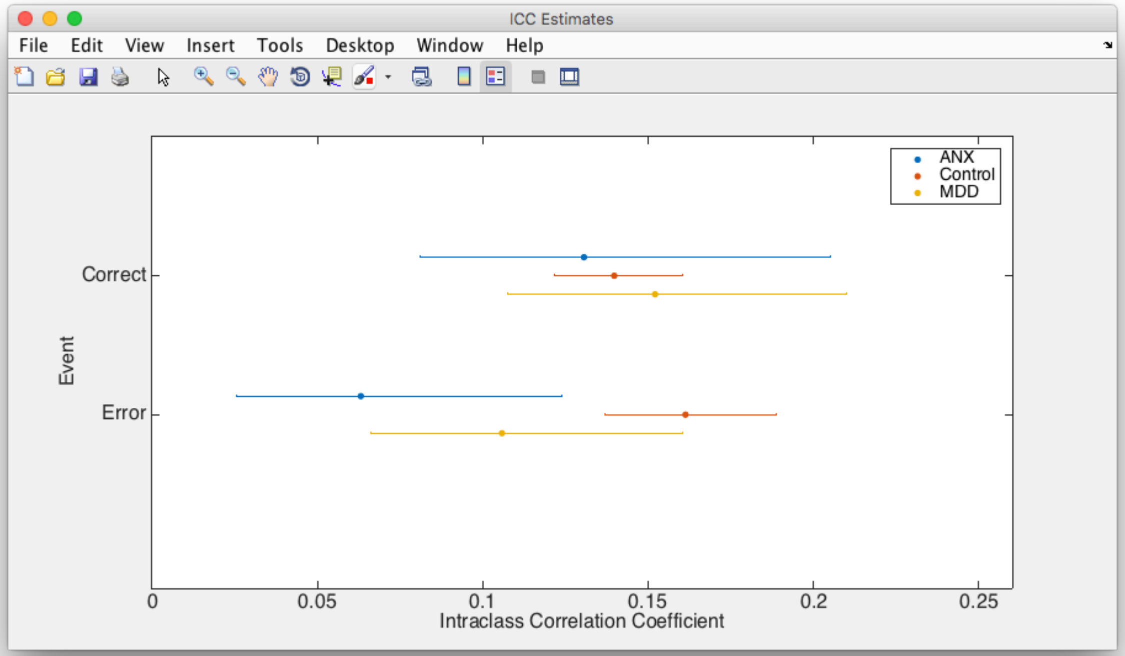
Dependability Plot



This plot shows the impact of increasing the number of trials on dependability estimates. The y-axis is the dependability estimate and the x-axis is the number of observations (trials) included in the estimate (upper limit of x-axis is defined in preferences). Each event is plotted separately, and separate lines show information for separate groups (legend provided). The dotted horizontal line is the specified dependability cutoff. This particular plot shows the point estimate, which is indicated in the menu bar of the window. The plot can be saved by pressing File, then Save.

In the example given, groups show a similar relationship between numbers of trials and dependability for correct trials. The relationship is very different for error trials. Controls obtain an acceptable level of dependability more quickly than the other groups.

ICC Plot

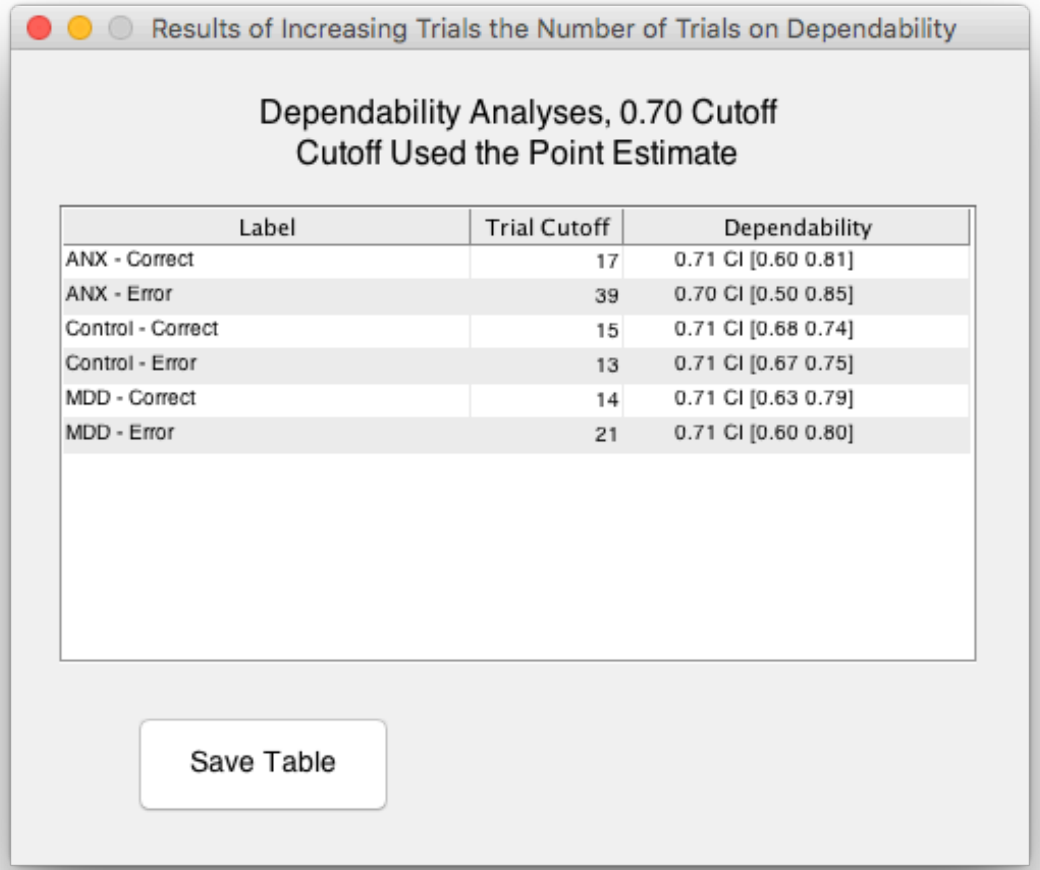


This plot shows the point estimate of the intraclass correlation coefficients (ICCs) with their associated 95% credible intervals. The y-axis is separate events, and the x-axis is ICC. Separate lines are provided for each group. The plot can be saved by pressing File, then Save.

The ICC is calculated using the formula: between-person variance/(between-person variance + error variance). This can be interpreted as the proportion of the total variance that is between persons.

In the example given, groups show fairly similar ICCs for correct trials. For error trials, ICCs are different between the groups. The ANX group is much lower than the Control group and suggests that the ANX group is going to need more error trials than the Control group to achieve dependable measurements.

Cutoff Inclusion Table


Results of Increasing Trials the Number of Trials on Dependability

Dependability Analyses, 0.70 Cutoff
Cutoff Used the Point Estimate

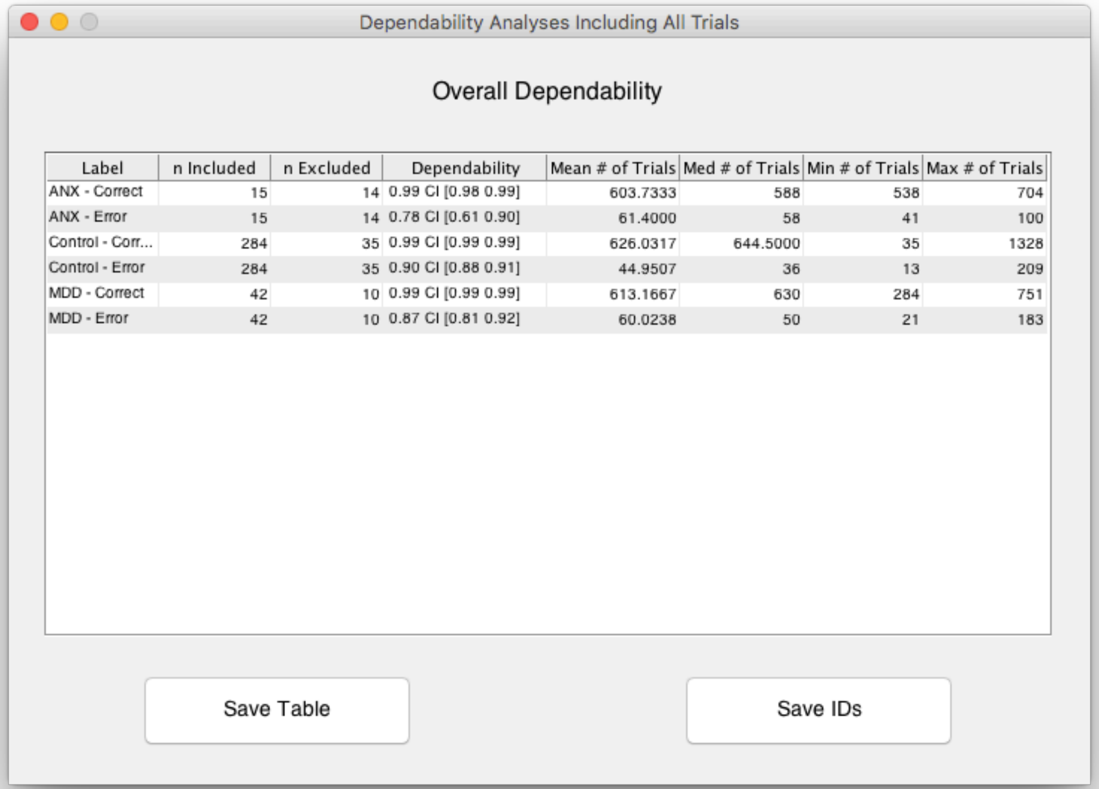
Label	Trial Cutoff	Dependability
ANX - Correct	17	0.71 CI [0.60 0.81]
ANX - Error	39	0.70 CI [0.50 0.85]
Control - Correct	15	0.71 CI [0.68 0.74]
Control - Error	13	0.71 CI [0.67 0.75]
MDD - Correct	14	0.71 CI [0.63 0.79]
MDD - Error	21	0.71 CI [0.60 0.80]

This table shows the numbers of trials needed to achieve an acceptable level of dependability, as specified by the user (indicated in the title of the table). Every possible group and event combination is shown with its specified cutoff and dependability point estimate and 95% credible interval. There is a button to save this table to a file.

A value of -1 indicates that an appropriate cutoff could not be realistically found given the specified dependability threshold. The toolbox will search for a trial cutoff up to 1,000 trials passed the maximum number of trials in the dataset.

Based on this table the user can get a sense of how many trials are needed for the data in hand. These cutoffs could then be used to guide decisions for excluding participants with unreliable measurements (see Clayson & Miller, 2017b).

Overall Dependability Table



Overall Dependability

Label	n Included	n Excluded	Dependability	Mean # of Trials	Med # of Trials	Min # of Trials	Max # of Trials
ANX - Correct	15	14	0.99 CI [0.98 0.99]	603.7333	588	538	704
ANX - Error	15	14	0.78 CI [0.61 0.90]	61.4000	58	41	100
Control - Corr...	284	35	0.99 CI [0.99 0.99]	626.0317	644.5000	35	1328
Control - Error	284	35	0.90 CI [0.88 0.91]	44.9507	36	13	209
MDD - Correct	42	10	0.99 CI [0.99 0.99]	613.1667	630	284	751
MDD - Error	42	10	0.87 CI [0.81 0.92]	60.0238	50	21	183

This table summarizes the data. This gui has two buttons. The Save Table button will save all of this information to a file. The Save IDs button will save a file that includes a column for the participant IDs with good data (those in n Included column) and a column for the participant IDs with bad data (those in n Excluded column). The information in this file could then be used to remove those participants with unreliable measurements from statistical analyses.

- n Included: number of participants with enough trials to survive the dependability threshold. Participants will only be included if they have enough trials for all possible events.

  This has a side effect that should be noted. Consider this example: if you are working with a dataset with 500 people and only person has enough trials for a given condition, then the n Included column will indicate that only 1 person has enough data for each condition. If this person were removed from analysis, then that condition would be deemed bad for all participants. As a result, the n Included column could include 499 participants for the remaining conditions. A value of -1 indicates that there were not any participants with enough data for a given condition.

- n Excluded: number of participants that did not have enough trials to survive the dependability threshold for at least one event.

- Dependability: dependability point estimate with 95% credible interval. This is the estimate for the dependability of the overall data. Either the mean or median (specified in the viewing preferences) can be used in this dependability equation. If the data are extremely skewed, you may consider using the median over the mean.

- ICC: Intraclass correlation coefficient point estimates with 95% credible intervals.

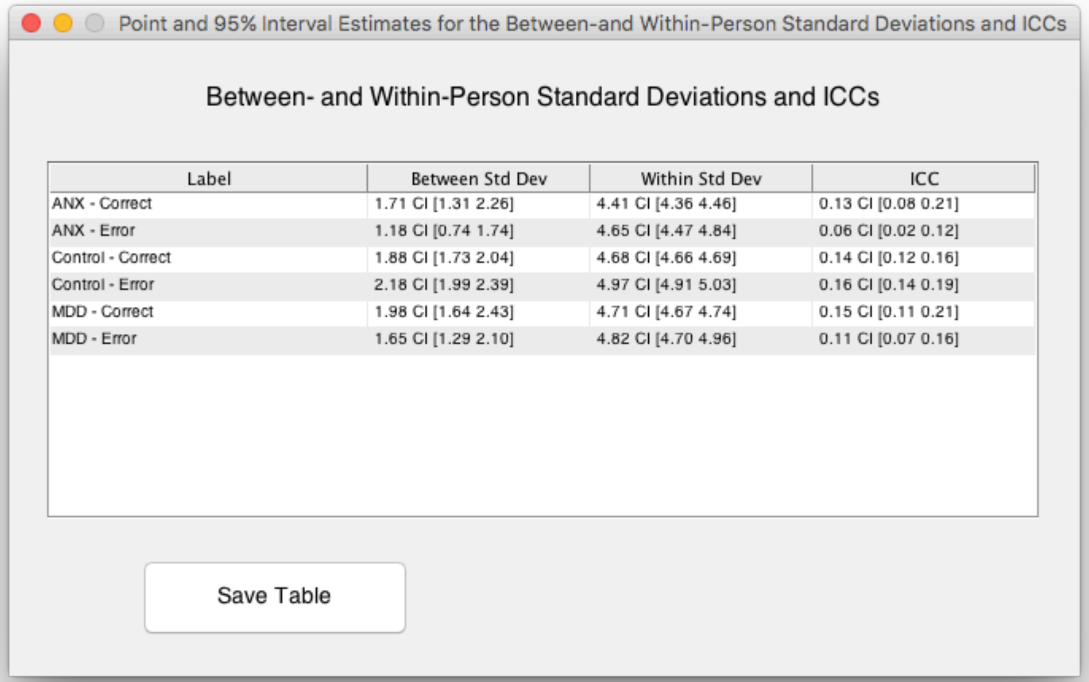
- Mean # of Trials: Mean number of trials for those participants with enough trials to survive the dependability threshold for each event (those participants in the n Included column).

- Med # of Trials: Median number of trials for those participants with enough trials to survive the dependability threshold for each event (those participants in the n Included column).

- Min # of Trials: Minimum number of trials for those participants with enough trials to survive the dependability threshold for each event (those participants in the n Included column).

- Max # of Trials: Maximum number of trials for those participants with enough trials to survive the dependability threshold for each event (those participants in the n Included column).

Relative Sizes of Variance Table


Point and 95% Interval Estimates for the Between-and Within-Person Standard Deviations and ICCs
Save Table

Between- and Within-Person Standard Deviations and ICCs

Label	Between Std Dev	Within Std Dev	ICC
ANX - Correct	1.71 CI [1.31 2.26]	4.41 CI [4.36 4.46]	0.13 CI [0.08 0.21]
ANX - Error	1.18 CI [0.74 1.74]	4.65 CI [4.47 4.84]	0.06 CI [0.02 0.12]
Control - Correct	1.88 CI [1.73 2.04]	4.68 CI [4.66 4.69]	0.14 CI [0.12 0.16]
Control - Error	2.18 CI [1.99 2.39]	4.97 CI [4.91 5.03]	0.16 CI [0.14 0.19]
MDD - Correct	1.98 CI [1.64 2.43]	4.71 CI [4.67 4.74]	0.15 CI [0.11 0.21]
MDD - Error	1.65 CI [1.29 2.10]	4.82 CI [4.70 4.96]	0.11 CI [0.07 0.16]

This table shows the between- and within-person standard deviations. This plot is another way to look at the relative contributions of variance (compared to ICCs). The table can be saved to a file by pressing the Save Table button.

- Between Std Dev: Point estimate for between-person standard deviations and the 95% credible interval.

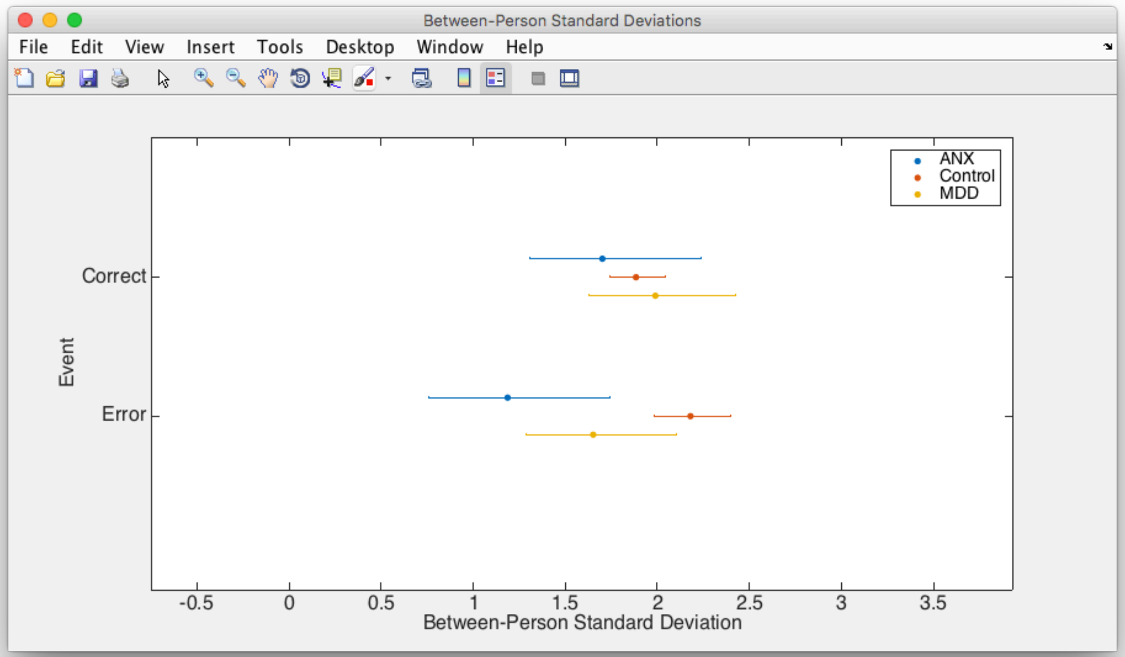
- Within Std Dev: Point estimate for within-person standard deviations and the 95% credible interval.

- ICC: Point estimate for the ICCs and their respective 95% credible intervals

If within-person standard deviations are large compared to between-person standard deviations, many trials are likely to be needed for stable measurements.

Comparing the standard deviations between groups and events also gives a sense of the underlying data contributing to these measurements. For example, the point estimate for Control-Error for between-person standard deviations is higher than the point estimates for the same condition (Error) for ANX and MDD groups. Given this, it makes sense as to why the Control group needs fewer trials than the ANX and MDD groups to obtain stable measurements.

Between-Person Standard Deviation Plot



This plot shows the point estimate of the between-person standard deviations with their associated 95% credible intervals. The y-axis is separate events, and the x-axis is between-person standard deviation. Separate lines are provided for separate groups. The plot can be saved by pressing File, then Save.

See section above for how to use between-person standard deviations.

Default Specifications

In the subroutines directory there is a file, era_defaults.m. This file contains the default specifications that will be used by the toolbox. In order to edit the specifications, open the file and change the desired value. This file will help make life a little easier so you do not have change the preferences to your precise specifications each time you run a dataset.

For example, in the processing preferences section of era_defaults.m, the default number of chains and number of iterations can be changed.

In the file are explanations of each possible input.

It should be noted that for now era_defaults.m will have to be re-edited each time the ERA Toolbox is updated.

References

Baldwin, S. A., Larson, M. J., & Clayson, P. E. (2015). The dependability of electrophysiological measurements of performance monitoring in a clinical sample: A generalizability and decision analysis of the ERN and Pe. *Psychophysiology, 52*, 790-800. doi:10.1111/psyp.12401

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., . . . Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software, 76.* doi:10.18637/jss.v076.i01

Clayson, P. E., & Miller, G. A. (2017a). ERP Reliability Analysis (ERA) Toolbox: An open-source toolbox for analyzing the reliability of event-related potentials. *International Journal of Psychophysiology, 111*, 68-79. doi:10.1016/j.ijpsycho.2016.10.012

Clayson, P. E., & Miller, G. A. (2017b). Psychometric considerations in the measurement of event-related brain potentials: Guidelines for measurement and reporting. *International Journal of Psychophysiology, 111*, 57-67. doi:10.1016/j.ijpsycho.2016.09.005

Gelman, A., Carlin, J. B., Stern, H. E., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis* (3rd ed.). Boca Raton, FL: CRC Pres.

Hajcak, G., Meyer, A., & Kotov, R. (2017). Psychometrics and the neuroscience of individual differences: Internal consistency limits between-subjects effects. *Journal of Abnormal Psychology, 126*, 823-834. doi:10.1037/abn0000274

Infantolino, Z. P., Luking, K. R., Sauder, C. L., Curtin, J. J., & Hajcak, G. (2018). Robust is not necessarily reliable: From within-subjects fMRI contrasts to between-subjects comparisons. *NeuroImage, 173*, 146-152. doi:10.1016/j.neuroimage.2018.02.024

Lunn, D., Jackson, C., Best, N., Thomas, A., & Spiegelhalter, D. (2012). *The BUGS book: A practical introduction to Bayesian analysis* CRC Press.

Stan Development Team. (2017). CmdStan: The command-line interface to Stan, version 2.14.0. Retrieved from http://mc-stan.org

Appendix A – Installation

These are instructions for installing 1) CmdStan, 2) MatlabProcessManager, and 3) MatlabStan. These instructions are provided here. I have copied quite a bit from the CmdStan manual and have embellished here and there. My purpose is to try to make this process straightforward for users of the ERA Toolbox. The entire process, depending on your computing power and internet speed, can take 10-30 minutes. I cover installation on Mac and Windows. Linux users are directed to the CmdStan manual for specific instructions.

(if you run into additional problems using these instructions, let me know and I will do my best to make them clearer)

I recommend trying to use the automatic installation provided by the toolbox. It will save a lot of time, if it goes smoothly. If you are a Mac user, you will be prompted to install Xcode during the automatic installation. If you are a Windows user, you will need to *first* install the Rtools package (see below).

Installing CmdStan

If you'd like, I recommend first creating a directory in your Documents folder. Name the directory: ERADependents. This directory will be separate from the ERA Toolbox itself. The ERA Toolbox dependents will be placed in this folder.

CmdStan requires the make utility and a C++ utility. Mac and Windows specific instructions are provided below.

Mac Installation Instructions

CmdStan recommends installing the Xcode Development Environment as the easiest way to install a C++ development environment on a Mac.

In order to download the latest, stable version of Xcode, go to the App Store on your Mac, search for Xcode, and download it (to access the App Store click on the apple in the top left corner and select 'App Store…').

After installing Xcode, open the application. Then in the top menu, click Xcode, then Preferences. Click the components button on the top row, then click on the install button to the right of the Command Line Tools entry.

After the command line tools have finished downloading, quit Xcode.

Now, double check that everything has installed correctly. Open up the Terminal application (located in the Utilities folder inside the Application folder).
In the Terminal application type,

```
make --version
```

The following output should be displayed (version highlighted).

```
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS
FOR A
PARTICULAR PURPOSE.

This program built for i386-apple-darwin11.3.0
```

Verify that the make version (highlighted) is at version 3.81 or later.

Then, in the Terminal application type,

```
g++ --version
```

The following output should be displayed.

```
Configured with: --
prefix=/Applications/Xcode.app/Contents/Developer/usr --with-
gxx-include-dir=/usr/include/c++/4.2.1
Apple LLVM version 7.3.0 (clang-703.0.31)
Target: x86_64-apple-darwin15.6.0
Thread model: posix
InstalledDir:
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefau
lt.xctoolchain/usr/bin
```

Verify that g++ is at version 4.2.1 (highlighted) or later.

The necessary make and C++ utility are now installed. Next, download and unpack the CmdStan source.

You can download the latest release from GitHub or the Stan website. I recommend downloading that tarball (.tar.gz) rather than the zip. It will be much faster to download and unpack.

Move the tarball or zip file to the ERADependents directory that was made earlier. Unpack the file.

If the tarball has not automatically unpacked the .tar.gz into cmdstan-2.17.0.tar, double-click the .tar.gz to unpack. Double click on the .tar file to unarchive the cmdstan-2.17.0 directory.

Now CmdStan needs to be built. To do this, the Terminal application needs to be opened again (/Applications/Utilities/Terminal).

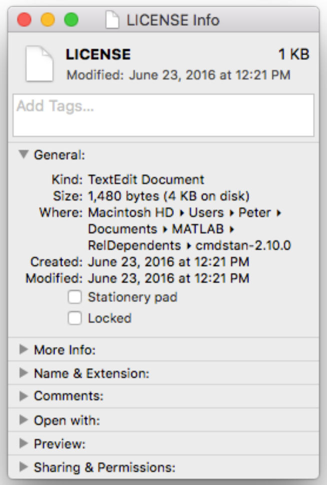
After opening the Terminal application, navigate to the CmdStan directory using the, cd, command.

For example:

```
cd /Users/Peter/Documents/MATLAB/ERADependents/cmdstan-2.17.0
```

If you are unfamiliar with the Terminal command line interface and Mac directory paths, then follow these steps to get your path:

Navigate to the CmdStan directory in the Finder. Locate the license file within the CmdStan directory. Right-click the file and select Get Info. You should see something like the following image:


LICENSE Info

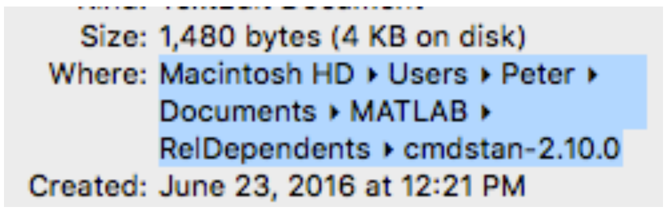
LICENSE 1 KB
Modified: June 23, 2016 at 12:21 PM

Add Tags...

General:
Kind: TextEdit Document
Size: 1,480 bytes (4 KB on disk)
Where: Macintosh HD ▸ Users ▸ Peter ▸ Documents ▸ MATLAB ▸ RelDependents ▸ cmdstan-2.10.0
Created: June 23, 2016 at 12:21 PM
Modified: June 23, 2016 at 12:21 PM
Stationery pad
Locked

More Info:
Name & Extension:
Comments:
Open with:
Preview:
Sharing & Permissions:

Highlight the section following Where: (see below)


Size: 1,480 bytes (4 KB on disk)
Where: Macintosh HD ▸ Users ▸ Peter ▸ Documents ▸ MATLAB ▸ RelDependents ▸ cmdstan-2.10.0
Created: June 23, 2016 at 12:21 PM

With that information highlighted, click Edit in the top menu and then copy. Doing so will copy the information you need to navigate to the cmdstan folder in Terminal.

Now, go to Terminal.

Type: cd

Then, go to Edit in the top menu and select paste. The path to the cmdstan directory should be pasted into Terminal. Click enter, and you should be taken to the cmdstan folder.

To ensure that you are in the cmdstan directory, type:

```
ls
```

The following output should be printed.

```
LICENSE        doc         makefile       stan_2.17.0
README.md      examples        runCmdStanTests.py test-all.sh
bin        make            src
```

Now, use the make command to build CmdStan. Multiple CPU cores can be used when they are available. To specify the number of cores to use, when calling make, used –j# option, where # is the number of CPU cores. If I wanted to build CmdStan using 2 cores, I would type the following into Terminal.

```
make build -j2
```

This process can take a while (10+ minutes depending on number of and speed of cores) and consume quite a bit of memory (2+ GB).

When CmdStan is successfully built, the make program will output

```
--- CmdStan v2.17.0 built ---
```

CmdStan has been successfully installed!

Windows Installation Instructions

The CmdStan manual recommends installing and using the Rtools package for installing a full C++ build environment.

Download the latest, stable (frozen) version of Rtools here. This download process has been tested with **Rtools33.exe**. I recommend using **Rtools33.exe**.

Then, double click on the downloaded file to begin the installation process.

Follow these options:

Language: select language, click Next,

Welcome: click Next,

Information: click Next,

Setup Location: accept default (c:\Rtools), click Next,

Select Components: select default, Package Authoring installation, click Next,

Select Additional Tasks: check Edit Path and Save Version in Registry, click Next,

System Path Report: verify that the paths to c:\Rtools\bin and c:\Rtools\gcc-4.6.3\bin are listed at the beginning of the path, click Next,

Ready to Install: click Next, wait for the installation to complete, then

Finish: click Finish

Now, verify the path. Open a command prompt.
A command prompt can be opened by clicking the start menu, and typing cmd in the search bar, then hitting enter. Alternatively, press the windows key and r at the same time, enter cmd into the text field, and hit enter.

Type

```
PATH
```

In the command prompt. Verify that c:\Rtools\bin is in the PATH environment variable. Also, make sure that c:\Rtools\bin is at the beginning of the path. If not follow these instructions for editing the PATH environment variable.

Then, in the command prompt, verify that g++ and make are installed. To do so, use the following commands:

```
g++ -v
make -v
```

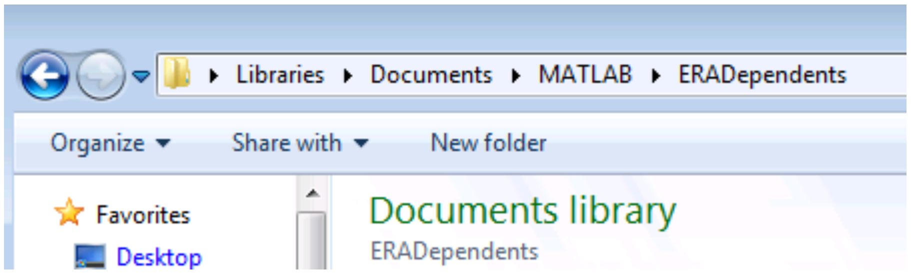
Version information should be outputted for both g++ and make.

Download the latest tarball for CmdStan here and save it in the ERADependents directory.

Open the command prompt again. Change the current directory to ERA Dependents directory.

If you are unfamiliar with the command prompt, here are some instructions for easily obtaining the path to the ERADependents directory and changing the directory in a command prompt.

Navigate to the folder in Windows. Note that in the top of the window a partial path to the location of the ERADependents directory is displayed.



Attempt to highlight the >Libraries>Documents>MATLAB>ERADependents portion (as an example).

Upon attempting to highlight, the full path to the directory will be displayed. Copy that information, by highlighting the full path and pressing CTRL + C at the same time.

Now, back in the command prompt type the following.

```
cd
```

After the cd, type a space, then right click. Select paste and hit enter. The command prompt will show a new line with the new path.

The command prompt should now show that you have navigated to the ERADependents directory where the CmdStan tarball is located.

Now, unpack the tarball using the tar command that was installed as part of the Rtools installation.

In the command prompt, type

```
tar --no-same-owner –xzf cmdstan-2.17.0.tar.gz
```

The CmdStan tarball will be unpacked. This may take a while.

Once the tarball has been unpacked, changed directories in the command prompt to the new CmdStan directory (follow instructions above for changing the path).

From the CmdStan directory in the command prompt, use the make command to build CmdStan. Multiple CPU cores can be used when they are available. To specify the number of cores to use, when calling make, used –j# option, where # is the number of CPU cores. If I wanted to build CmdStan using 2 cores, I would type the following into the command prompt.

```
make build -j2
```

This process can take a while (10+ minutes depending on number of and speed of cores) and consume quite a bit of memory (2+ GB).

When CmdStan is successfully built, the make program will output

```
--- CmdStan v2.17.0 built ---
```

CmdStan has been successfully installed!

Installing MatlabProcessManager

Go to the MatlabProcessManager Gihub page. Download the source code and unpack the zip or tarball in the ERADependents directory.

Installing MatlabStan

You can download MatlabStan here. Download the source code and unpack the zip or tarball in the ERADependents directory.

Then, navigate to the +mstan folder within the MatlabStan directory. Open the file, stan_home.m.

In the stan_home.m file, there will be one line of code that needs to be adjusted.

Change the line 8,

```
d = '/Users/brian/Downloads/cmdstan';
```

to specify the path to your cmdstan directory. You can use the approach I described above in the CmdStan instructions for finding the path to directories (for either mac or windows).

Enter the path to your CmdStan directory. It could look something like this.

```
d = ‘/Users/Peter/ERADependents/cmdstan-2.17.0’;
```

Save the changes.

Then, you need to add the new directories to the Matlab path. You can find instructions for adding directories to the Matlab path here. Use ‘Add with Subfolders…’ and add the entire ERADependents directory. Then click save.

Success!

All three dependents have now been installed.