**Bayesian hierarchical signal detection theory**

This documentation is for an application for fitting a Bayesian hierarchical model to binary signal detection data. Please refer to the accompanying paper for information about the model.

Briefly, the model details are:

Suppose we have subjects, stimulus-present trials, noise trials[[1]](#footnote-1).

Each subject generates hits and false alarms .

We assume that the number of hits and false alarms for subject are binomially distributed where the probability of success (a hit for stimulus-present trials and a false alarm for noise trials ) are transformations of the *d’* and *c* for that subject .

That is, and , where is the normal distribution function.

We add a hierarchical distributions to and : and .

Due to the hierarchical structure, we have hyperpriors on each of these hyperparameters. That is:

where the Cauchy distribution is parameterised by the location and scale .

# 1. Starting up the application

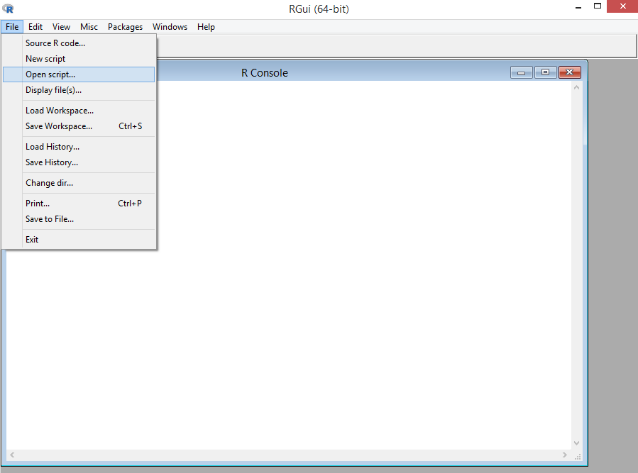
The code required for this application are contained in “app.R”, “app\_make\_data.R”, and “app\_stan\_model.stan”. These should all be in the same directory.

To run the application, we require an installation of R (<https://cran.r-project.org/>). We prefer versions 3.5 or later because between 3.4 and 3.5, some changes were made to how R objects are encoded when they are saved to an .RData file.

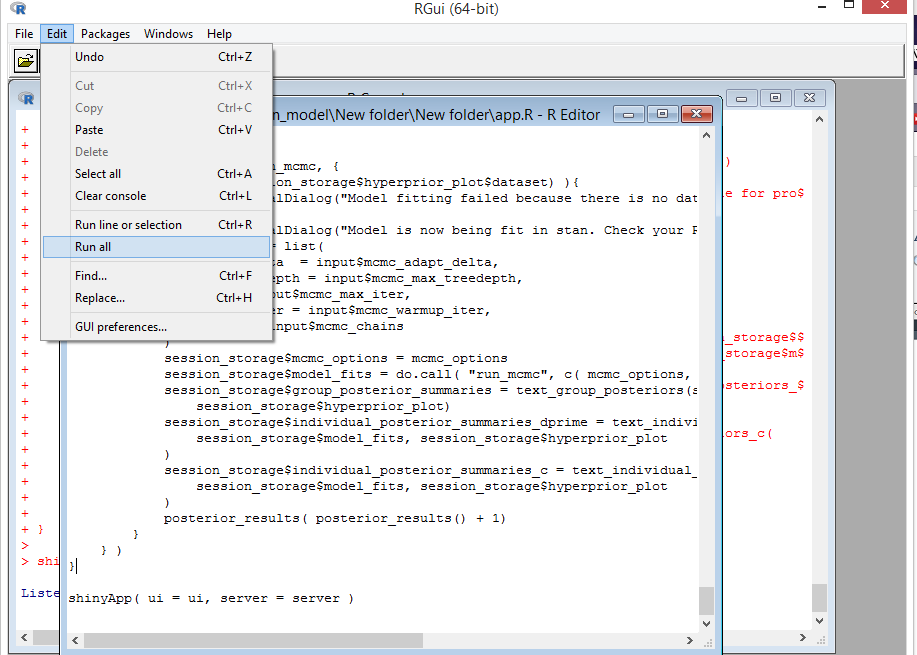
The application requires two R packages to run “shiny” and “rstan”. An attempt has been made to set up the loading of the packages such that they will install them if they are not found, but you may have to manually install them using the “install.packages()” function.

To run the application:

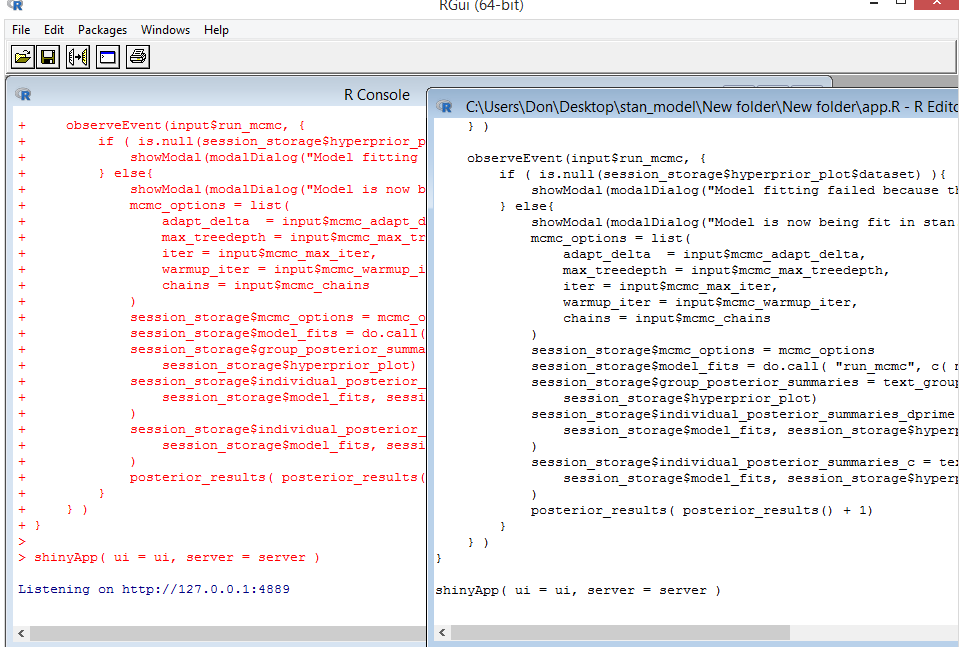
1. Open an R console – the base R gui or another editor (e.g., RStudio).
2. Open the “app.R” file



1. Run the contents of the file. It may take a minute or so for the application to start because we need to compile the .stan model code.

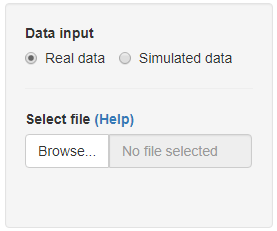


1. If the app successfully starts up, you should see a message “Listening on http://127.0.0.1:xxx”. Then, your default web browser should be automatically opened and the application should appear in one of the tabs. If this does not happen or you wish to use a different web browser, you can manually enter the address above into a web browser of your choice to start an instance of the application.

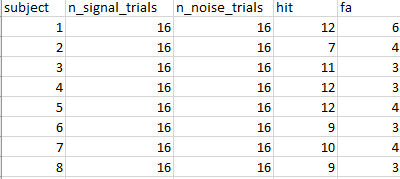


# 2. Getting your data into the application

There are two options for getting data into the application: to upload a .csv file containing your data or to generate some simulated data.

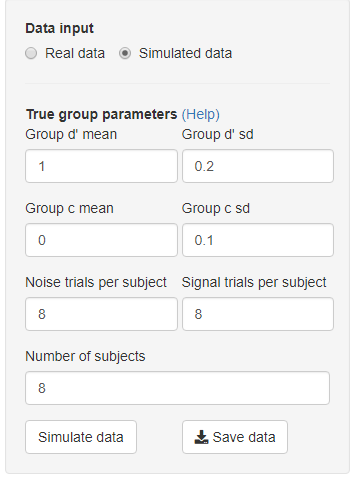


To import data, the structure of the file has to follow some constraints. The file must have five columns containing information about the subject ID, the number of signal trials, the number of noise trials, the number of hits, and the number of false alarms. An example is shown below.

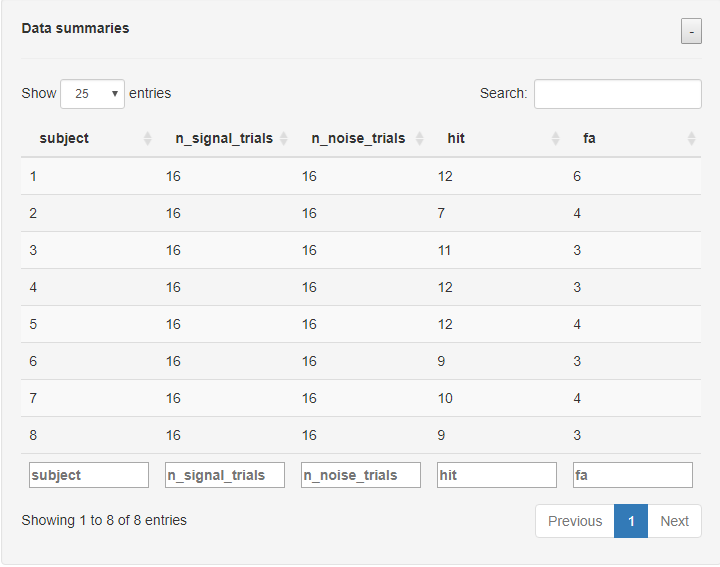


We are fairly flexible about the column names, but refer to the help dialog in the application for more details about what is recognised.

The other option is to simulate some data. The underlying generative model is that d’ and c are normally distributed. For subjects, values of d’ and c are generated and then hits and false alarms are generated according to the specified number of trials.

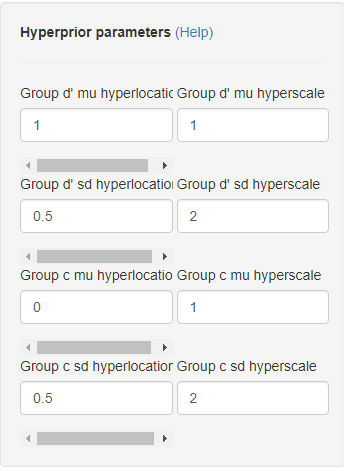


When data is simulated or uploaded, the structure of the data will be shown in the “Data summaries” panel:

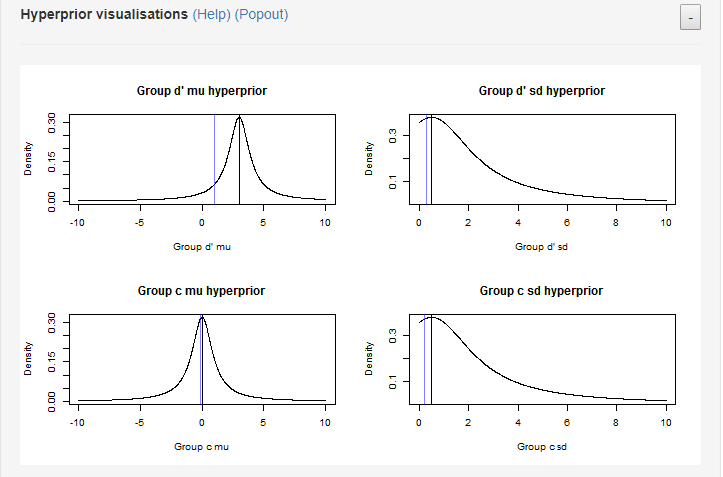


# 3. Constructing your hyperpriors

Hyperprior parameters are selected in this panel.

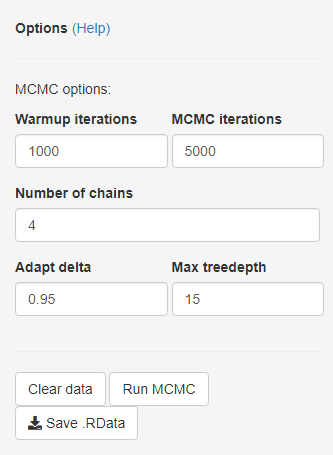


Plots of the densities generated by these parameters are shown in the “hyperprior visualisations” panel. The black vertical lines denote the location. If there is data in the application, empirical point estimates are shown as vertical blue lines. See the help dialog for more details. The blue vertical line can be helpful for choosing hyperprior parameters.



# 4. Fitting the model

The hierarchical model is fit using Stan, which implements Hamiltonian Monte Carlo. There are five parameters that we can change.

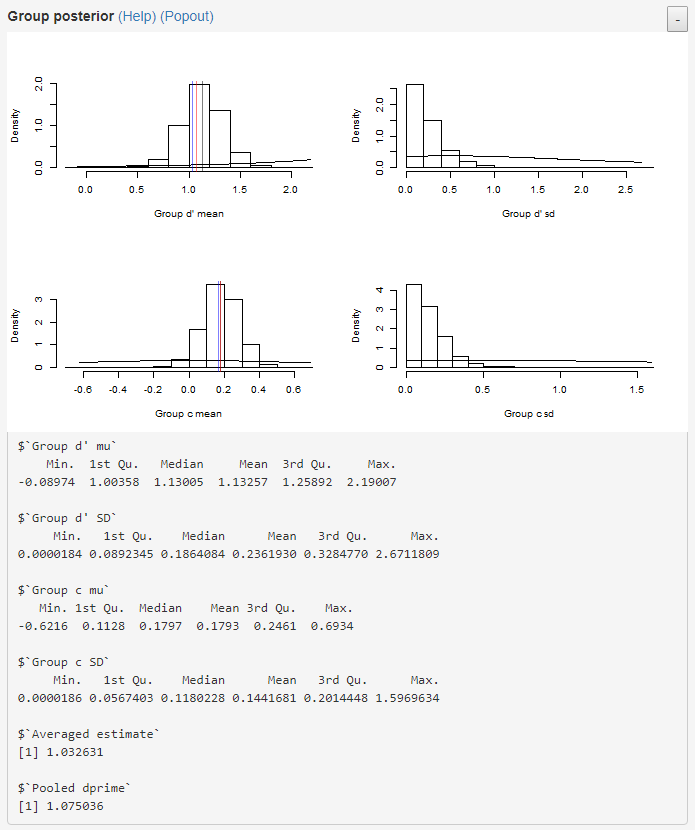


Click “Run MCMC” to fit the model to the data in the application. During model fitting, some progress dialog will appear on your R console.

After running MCMC, you can click “Save .RData” to save the R objects that are generated from model fitting. This will include a dataframe that contains the data, a stan object containing the model fits, and the marginal posterior summaries.

# 5. Assessing model fits

In the “MCMC results” panel, you can find histograms of the marginal posteriors for the group parameter and for individual parameters. There are also some text summaries showing the marginal posterior means, medians, and some quantiles. In addition, comparisons with non-Bayesian estimates are also shown. See the help dialog for more details.



# 6. Final notes

In this application, we have provided rudimentary tools for hierarchical Bayesian analysis of two-alternative detection data.

Ideally, some MCMC diagnostics (e.g., trace plots) should be conducted on the MCMC results in order to ensure the validity of the posterior distributions. There are a lot of diagnostics that you can conduct, and the best ones depend on what your specific problems are. We leave this to the user to conduct.

1. The number of stimulus-present and noise trials can differ between subjects. But for simplicity, we will omit this detail without any loss of generality. [↑](#footnote-ref-1)