REU 2016 Documentation: Reverse Engineering Functional Brain Networks from fMRI Data Using Probabilistic Boolean Networks

Erin Boggess
Simpson College

Tiffany Jann
University of California, Berkeley

August 16, 2016

Abstract

1 Introduction

2 Methods/Pipeline

All commands, unless specified, are in reference to the BrainProject_2016 folder. For all MATLAB operations, open MATLAB 2014a or earlier. Add the BrainProject_2016 folder to path and open it.

2.1 Partial Prior Knowledge Network

2.1.1 Gold Standard Networks

Generating Erdos-Renyi Random Networks in (MATLAB)

In the Erdős–Rényi networks, edges are created with the same constant probability. There is no biological justification for using such networks, but we included them to test the general robustness of these network inference methods on a wide range of networks. Random networks are created by the `generateDirectedRandom(nodes, p, iter)` function in the Helper-scripts folder. It takes as argument the number of nodes, the probability of an edge p, and a number identifier `iter` that is appended to the end of the structure file created. The structure file that is created is saved to path `BrainProject_2016/MATLAB/MULAN/GenerateData/structureN[nodes]directedRandom[iter].mat`.
Note that if you wish to generate many random networks at once, you can modify the `start_repeated_scaleFree.m` file in `BrainProject_2016/MATLAB` to call `generateDirectedRandom` instead of `generateScaleFree`, albeit it will do more than just generate gold standard networks (see next section).

```
cd MATLAB/MULAN
generateDirectedRandom(nodes, p, iter) %default p is 0.5
```

Generating Barabasi-Albert Scale Free Networks in (MATLAB)

In the Barabasi-Albert Scale Free Networks, edges are drawn from a power law distribution. These networks have been reportedly observed in biological systems such as protein interaction networks and gene interaction networks [**?**]. Cortical assemblies are also believed to have scale-free network topology [**?**]. Scale free networks are created by the `generateScaleFree(nodes, iter)` function in the Helper-scripts folder. Nodes is the number of nodes and iter is the identifying integer appended to the end of the new structure file's name. The structure file that is created is saved to `MATLAB/MULAN/GenerateData/structureN[nodes]sc`
Note that if you wish to generate many scale free networks at once, you can simply call `start_repeated_scaleFree.m` file in `BrainProject_2016/MATLAB`, albeit it will do more than just generate gold standard networks (see next section).

```
cd MATLAB/MULAN
generateScaleFree(nodes, iter)
```

2.1.2 Network Inference Methods

2.1.3 Evaluating Network Inference Methods

Applying and Evaluating Network Inference Methods in (MATLAB)

The files that actually apply the network inference methods are found in `MATLAB/MULAN/Calculation`, but calling `mln_CalEvaN` will run through generating data from the GSN, inferring networks for all 44 methods, and evaluating the statistical metrics altogether. Note that you can call mln_CalEvaN in the `MATLAB` directory as well as any of its subdirectories. To make documentation simple, we assume all calls to `mln_CalEvaN`, `multisub`, `start_repeated_ScaleFree`, etc. are done in the `MATLAB` directory, but feel free to make the calls deeper in. The documentation for `mln_CalEvaN` is well-written in its comments. However, I suggest keeping `inFolderName` and `inPrenom` as alphanumeric only, no dashes etc. I would also give them the same string, as in the script's example (`nmm`). As of August 2016, the experimental data is 49 nodes and 204 time points, so I would use those values for `inNumNodes` and `inNumTimePts`. However, as of August 2016, when we attempt to apply the sliding windows technique, 204 time points causes major problems. If you are on this step, my intuition says you may want to look into `mln_generateParams.m` in `MATLAB/MULAN/ClusterComputation`.

To use existing data rather than generate new data from a specified GSN, use a variant of the `multisub` scripts found in `MATLAB/Helper-scripts`.

Note that `mln_CalEvaN` deals with one data set at a time. To run several generations, use `start_repeated_scaleFree.m` file in `BrainProject_2016/MATLAB`.

```
mln_CalEvaN inFolderName inPrenom GenerateData/inExistingStruct nmmParams inNumNodes 1 inNumPts 1 fMRI
```

2.1.4 Generating Consensus Networks

Finding the Inference Methods Ranks for AUC and ACC (MATLAB)

In the previous step, each `mln_CalEvaN` call returns one folder named according to the parameter `inFolderName` containing 4 subfolders: `AUC`, `data`, `Results`, and `ToutResults`. We refer to this as a "result folder". The AUC and ACC values for each data set are found in the `Meths.MSAUC` and `Meths.ACC` fields of `inFolderName/AUC/AUC_prenomfmriCS100S1N204.mat`.

`topMethods(path)` will take each result folder in its path, average the AUC and max ACC of all 44 methods, and print out the mean and standard deviation of the mean, positive predictive value, and accuracy metrics, and then the rank of the same three metrics. You need to create a new empty folder and put all relevant result folders into this 'shell'. This means you *can* make a shell folder containing just one result folder, and hence the 'average' AUC and ACC value for all 44 methods would be just the AUCs and ACCs of that result folder.

```
cd MATLAB/shellFolder
generateStatistics(pwd) %or you can call path = pwd, and call generateStatistics(path)
topMethods(pwd)         %or if you called path = pwd, topMethods(path)
```

Using Weighted Sums to Rank Inference Methods (MATLAB AND R)

In the last step, we obtained a print-out in 6 Parts. To calculate the top inference methods using both ACC and AUC together, we use the 4th and 6th outputs: Rank Score (AUC) and Rank Score (ACC). Copy and paste both columns (Method name — Rank Score) of each into their respective inputs in the R script `ConsensusNetworks/consensusNetworkScripts/LinCombAUCACC.Rmd`. Finally, Knit HTML or run the entire document–the script will output a `.html`, `.tex`, and a .csv file. The result table is clear in the HTML and CSV documents, while the LaTeX file is for copying and pasting into LaTeX documents.

Comments and example inputs are left in the file.

Constructing Consensus Networks from K Top Performers for Each Weighted Sum (MATLAB)

Using the results from the above, edit the code in `MATLAB/Helper-scripts/addConsensusNetworks.m` and run the script, once for each weighted sum. To edit the code, fill in the first K methods corresponding to the 9 variables `topK`: `top2, top3, ... , top9, top10` for that weighted sum. After you make the following commands, MULAN will have added one file ending with '`_extended.mat`' to each `AUC` and `Results` folder for every results folder under your `shellFolder`. The last command will rank and print to console all reverse-engineering methods' outputs, including the original 44 and the newly generated consensus networks.

```
cd MATLAB/shellFolder
consensusNetworks(pwd)  %or you can call path = pwd, and then call consensusNetworks(path)
topMethodsExtended(pwd) %or if you called path = pwd, topMethodsExtended(path)
```

2.2 Discretization

2.2.1 Variable Inputs

2.2.2 Applying Discretization Techniques to fMRI Data

Applying Discretization Techniques to fMRI Data (GED PRO TOOLS)

Save the MATLAB data sets from each results folder as CSV files. These can be found in `shellFolder/inFolderName/data`. The GED PRO TOOLS app and user manual can both be found in `Discretization/gedprotools`. As of August 2016, Using GEDPROTOOLS on PC gives you the liberty to create new files and hence assign names as you save your discretized results. On Mac I've had to create dummy CSV files with desired names before hand so I could save the discretized results into those files. Summary: save MATLAB data files to a CSV file, input those CSV files into GEDPROTOOLS, GEDPROTOOLS will output discretized data in CSV files.

Note: there is an additional documentation document located in `Discretization/discretizationScripts/discretizationNamingD` that extensively deals with the namings for the CSV data files. Do heed this if you plan on using my data wrangling R scripts.

2.2.3 Method for Benchmarking Discretization Techniques

Analyzing Discretized Data (R)

You can play around with setting directories in the script, but as it is, to use `AreaBtwnCurves.Rmd`, first *copy* one relevant original data file as well as all its corresponding discretization files (CSV) to `BrainProject_2016/Discretization/discretizationSc` Then change the `goldFile` variable to match the original data file name. If this data file is experimental data, set `isPatientData` to TRUE, otherwise, if the data file is *in silico*, set `isPatientData` to FALSE. Assuming you followed the nomenclature scheme described in `Discretization/discretizationScripts/discretizationNamingDocumentation.rtf`, once you knit HTML, you will see each discretization method ranked by its Absolute Area Between Curves value.

2.3 Generating Boolean Networks

2.3.1 Prologue

2.3.2 Parameter Testing for REACT

2.3.3 Full Simulation of Boolean Networks

Text Files Setup and Naming Before Running REACT (Terminal, Mac)

The executable file in the Sample Input Files folder is compiled for Mac and neither works on PC nor Linux (Ubuntu). We re-compiled the source code from GitHub on PC and saved the PC executable in `REACT/pcREACTor/Reactpc.exe`. We know this executable works with the Sample Input Files, however, it would not run with our files (which run fine on Mac). We assume there that there is either something wrong with our text files but Mac somehow overlooks and carries on, or there is

an incompatibility problem between Mac and Windows text files. Between the four text files we use, we narrowed down the problem to the Reverse-Engineering and Binary Time Series files.

In `REACT/SimBooleanNetworks` and `REACT/SimParamTestInputs` are our trial runs. Refer to these folders for naming purposes. Examples below:

```
DiscreteData_RN1bikmeans.txt    % binary time series
Fileman_RN1bikmeans.txt         % fileman
Params.txt                      % parameters
RevEng_RN1.txt                  % static network

Results_RN1bikmeans.txt         % output from calling
                                % ./React Fileman_RN1bikmeans.txt Results_RN1bikmeans.txt
ScoreResults_RN1bikmeans.txt    % output from running the REACTscores.Rmd script
```

Running REACT (Terminal, Mac)

With all necessary text files as well as the REACT executable in the same folder, go to Terminal `cd` to that folder. As detailed in the tutorial, the command to run one trial of REACT is:

```
./React Fileman_RN1bikmeans.txt Results_RN1bikmeans.txt ScoreResults_RN1bikmeans.txt
```

Our recommendation is to use a text editor such as Sublime or TextEdit with good Find and Replace functions, to help chain commands. This is an example of a mass-chain command:

```
./React Fileman_SF1mean.txt Results_SF1mean.txt &&
./React Fileman_SF1median.txt Results_SF1median.txt &&
./React Fileman_SF1kmeans.txt Results_SF1kmeans.txt &&
./React Fileman_SF1bikmeans.txt Results_SF1bikmeans.txt &&
./React Fileman_SF1equalwidth.txt Results_SF1equalwidth.txt &&
./React Fileman_SF2mean.txt Results_SF2mean.txt &&
./React Fileman_SF2median.txt Results_SF2median.txt &&
./React Fileman_SF2kmeans.txt Results_SF2kmeans.txt &&
./React Fileman_SF2bikmeans.txt Results_SF2bikmeans.txt &&
./React Fileman_SF2equalwidth.txt Results_SF2equalwidth.txt &&
./React Fileman_SF3mean.txt Results_SF3mean.txt &&
./React Fileman_SF3median.txt Results_SF3median.txt &&
./React Fileman_SF3kmeans.txt Results_SF3kmeans.txt &&
./React Fileman_SF3bikmeans.txt Results_SF3bikmeans.txt &&
./React Fileman_SF3equalwidth.txt Results_SF3equalwidth.txt &&
./React Fileman_SF4mean.txt Results_SF4mean.txt &&
./React Fileman_SF4median.txt Results_SF4median.txt &&
./React Fileman_SF4kmeans.txt Results_SF4kmeans.txt &&
./React Fileman_SF4bikmeans.txt Results_SF4bikmeans.txt &&
./React Fileman_SF4equalwidth.txt Results_SF4equalwidth.txt &&
./React Fileman_SF5mean.txt Results_SF5mean.txt &&
./React Fileman_SF5median.txt Results_SF5median.txt &&
./React Fileman_SF5kmeans.txt Results_SF5kmeans.txt &&
./React Fileman_SF5bikmeans.txt Results_SF5bikmeans.txt &&
./React Fileman_SF5equalwidth.txt Results_SF5equalwidth.txt
```

This can be generated by taking

```
./React Fileman_SF1mean.txt Results_SF1mean.txt &&
./React Fileman_SF1median.txt Results_SF1median.txt &&
./React Fileman_SF1kmeans.txt Results_SF1kmeans.txt &&
./React Fileman_SF1bikmeans.txt Results_SF1bikmeans.txt &&
./React Fileman_SF1equalwidth.txt Results_SF1equalwidth.txt
```

---

and using `Find 1 Replace With 2`, `Find 1 Replace With 3`, etc. Then after chaining SF1 through SF5, you can use `Find SF Replace With RN`. Once you have your chained command, copy it into Terminal and press `return`. This will run all the REACT calls in the chain. You may split the command into as many Terminal windows your computer can handle at a time.

Accessing REACT Scores (R, Terminal, Mac)

Running REACT outputs a result file, which includes multiple models, each model containing a polynomial update function for each node, and multiple scores corresponding to the model and each of its update functions. To filter out the score, we wrote a R script located in `REACT/reactScripts/REACTscores.Rmd`. You may want to copy the script to where your result files are located in before running in RStudio/R. To be clear, `REACTscores` will save text files for all Result files identified in the folder in one go. You can just run it once after all your REACT runs are over, but multiple runs will simply overwrite, so you can take a peek before the REACT simulations are over as well.

2.4 Generating Probabilistic Boolean Networks

2.4.1 Creating Our Prototypes

Sliding Across Time Series (R)

With the full discretized data sets in a folder, copy to that folder the R Markdown file found at `REACT/reactScripts/SlideWindowDisc`
In this file, change the `pattern` variable if necessary. Otherwise, knit the script, and it should output to the folder 6 text files per binary time series–the 6 windows.