MULTIPLE CONNECTIVITY ANALYSIS (MULAN)
A MATLAB/PYTHON TOOLBOX

MULAN V2.0: ALGORITHM & EVALUATION

Reference paper: Wang, H. E., Friston, K. J., Bénar, C. G., Woodman, M. M., Chauvel, P., Jirsa, V., and Bernard, C. (2018). MULAN: Evaluation and ensemble statistical inference for functional connectivity. NeuroImage. https://doi.org/10.1016/j.neuroimage.2017.10.036

Contact : Huifang E Wang: elizabethhw@gmail.com

Aix Marseille Univ, Inserm, INS, Institut de Neurosciences des Systemes, Marseille, 13005, France

Contents

1. Introduction

MULAN (MULtiple method ANalysis) is a method that uses fuzzy inference, genetic and algorithms to integrate multiple methods to infer consensus connectivity. And also it includes a platform to systematically evaluate any given connectivity analysis method.

MULAN is an open platform. Any new method can easily be added and tested.

The MULAN toolbox is written in two program languages: MATLAB and Python. MATLAB codes include 42 basic methods and MULAN algorithm. The simulated datasets and the definition of ground truth structures are also in MATLAB. The Python codes include the evaluation and output the graphs.

This user guide has three parts. First, the three demos give users the first flavour how to use MULAN to obtain the connectivity results and demonstrate the results. Second, the cluster computation codes for running on a cluster to perform simultaneous computations. Third, the main structures of the MULAN codes.

The demos and cluster bash files are in the main folder, refered to Fig. 1 as one example. The toolbox is divided into six blocks in six folders: "GenerateData" to simulate datasets, "Calculation" for basic methods, "Inference" for MULAN algorithm, "AdaptiveParameter" for the optimization of parameters, "Evaluation" for statistical evaluation, "Demonstration" for plotting figures. Two additional folders are also included. "ClusterComputation" contains main functions to generate simulated datasets and calculate MULAN results. "FindStructures" generates designed ground-truth structures. Fig. 7 shows the Code structure from file MULAN algorithm codes.mm, which includes hyperlinks to the codes if you are using FreeMind.

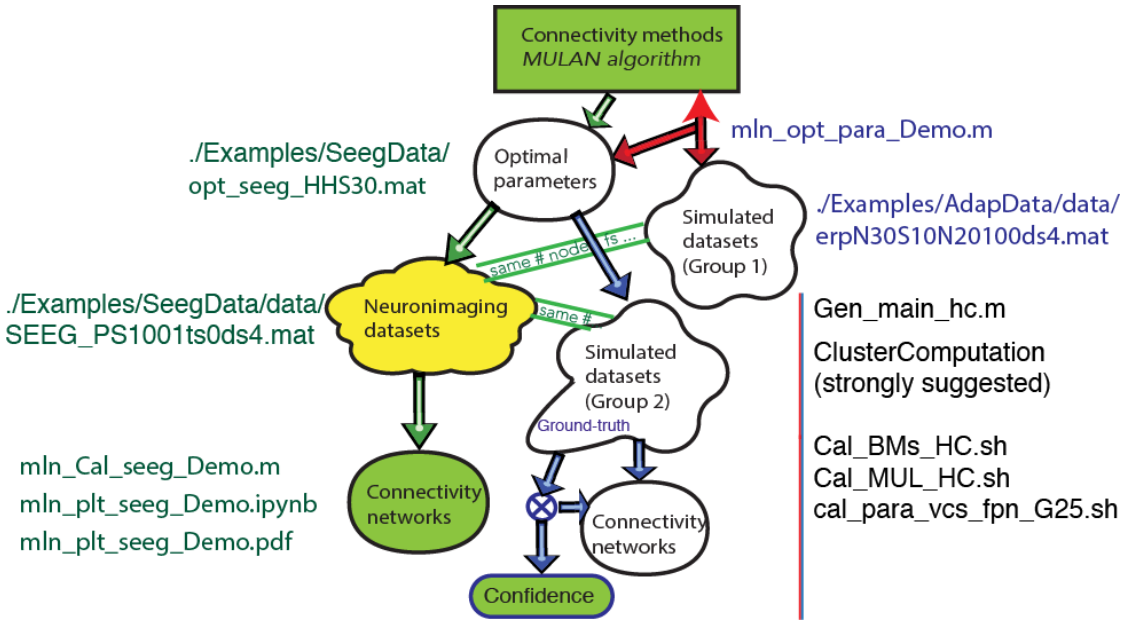


FIGURE 1. Flowing chart for MLAN application and the examples given in the code folder MULANIICodes. Demo 1 is for analyzing an experimental data in green. Given a SEEG dataset and an optimal parameter file in folder $./Examples/SeegData/$, please run $mln\_Cal\_seeg\_Demo$ to obtain the basic methods and MULAN results. For demonstrating the results, please run ipython code: $mln\_plt\_Seeg\_Demo.ipynb$ or just open the corresponding pdf file to review all the codes and results. Demo 2 is for obtaining the optimal parameters, with an example data given in folder $./Examples/AdapData/$. Please run $mln\_opt\_para\_Demo.m$ for the optimal parameters for the given dataset. Notice that the procedure is strongly suggested by multiple datasets, we use 50 datasets. Then the codes are given for doing such procedure for multiple datasets from generating the data to the finial MULAN results in Part 2.

2. Demos

There are three demos which can help the users to use MULAN algorithm to obtain the connectivity results and have a fast access to use the MULAN codes. Demo 1 is to analyze a given experimental dataset and demonstrate the final results; Demo 2 is to obtain the optimal parameter files for a given dataset; Demo 3 is to analyze a given dataset with 84 nodes human connectomes as ground-truth structures by using subnetworks and demonstrating the results for each subnetwork and a whole network.

2.1. Demo 1: Application on a SEEG dataset.

Example 2.1. *Given a SEEG dataset as* $./Examples/SeegData/data/SEEG\_PS1001ts0ds4.mat$ *and an optimal parameter file* $opt\_seeg\_HHS30.mat$.

run $mln\_Cal\_Seeg\_Demo.m$

Results:

- *The results for basic methods can be found in Examples/SeegData/ToutResults/ Tout_1500_SEEG_PS1001tsods4.mat*
- *The results for MULAN: Examples/SeegData/ToutResults/ Adp_1500_SEEG_PS1001tsods4FRN_G5_tak.mat*

Remarks:

- *The folder (./Examples/SeegData/Results/) saves the results from basic methods for each method family. This folder is designed for avoiding repeatedly calculation.*

run $mln\_plt\_Seeg\_Demo.ipynb$

Or ***open*** $mln\_plt\_Seeg\_Demo.pdf$

Remarks:

- *The Ipython Notebook is an interactive computational environment, which allow us to combine code execution, rich text, mathematics and plots. If you can not open the Ipython Notebook on your computer, you also could review all the codes and results by open the .pdf file with the same file name.*

R Fig. 1 shows Demo 1 in green. The codes can be found in main folder, but the data and results are in folder $./Examples/SeegData/$.

R Fig. 2 shows how the ipython codes and part of results, which you can find both in the ipython notebook and pdf file.

```
In [18]: fig= plt.figure(figsize = (20,10))

fig, axes=mc.plot_connectivity_circle_dir_2para(fmlnMat, labelname, n_lines=None, facecolor='white',textcolor='black',
                        node_colors=Color_ch,colormap='mulan',node_angles=node_angles, flagcon = 'F',
                        title='',fig=fig,padding = 4.,fontsize_colorbar=12,fontsize_names=16) #node_angles=node_angles
plt.savefig('ts0'+datasets+'.eps',pdi=600)
```

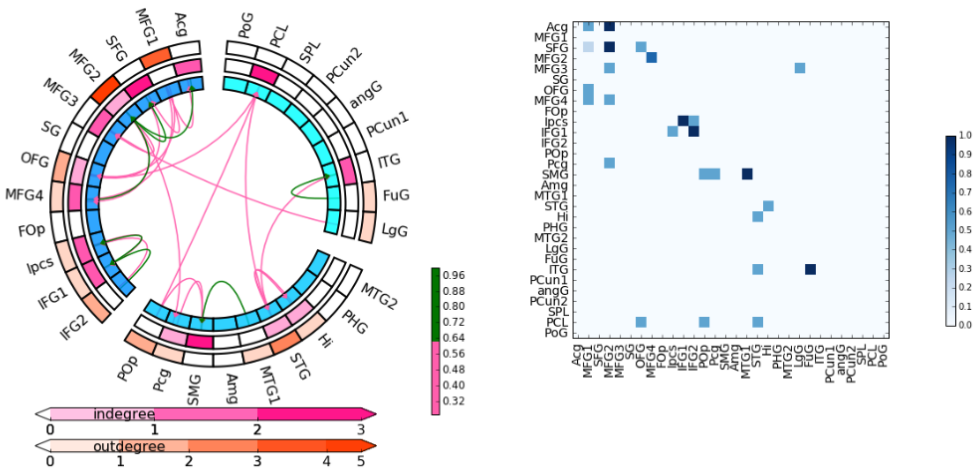


FIGURE 2. A part of ipython notebook codes and results.

2.2. Demo 2: Computation of optimal parameters.

Example 2.2. *Given a dataset as* $./Examples/AdapData/data/erpN30S10N20100ds4.mat$.

run $mln\_opt\_para\_Demo.m$

Results:

- *The results for basic methods:* $Examples/AdapData/ToutResults/Tout_1500_erpN30S10N20100ds4.mat$
- *The optimal parameters and the corresponding MULAN results for 2 different groups of basic methods:* *Fit_FPNG2_1500_erpN30S10N20100ds4.mat ; Fit_FPNG5_1500_erpN30S10N20100ds4.mat*
- *Within Fit*.mat, variable* state.bestscorepfit *for the number of false links (false positive and negative), which is 0 in this example. Variable* state.bestCh *is for the optimal parameters.*

Remarks:

- *Generation of the optimal parameters should be based on the multiple datasets (in our paper, we use 50 datasets). We systemically generate the simulated datasets with known ground-truth structures and use cluster based bash file to compute the optimal parameters from 50 datasets. The sample codes for bash file can be found in part 2 "cluster computation".*

R Fig. 1 shows Demo 2 in read. The codes can be found in main folder, but the data and results are in folder $./Examples/AdapData/$.

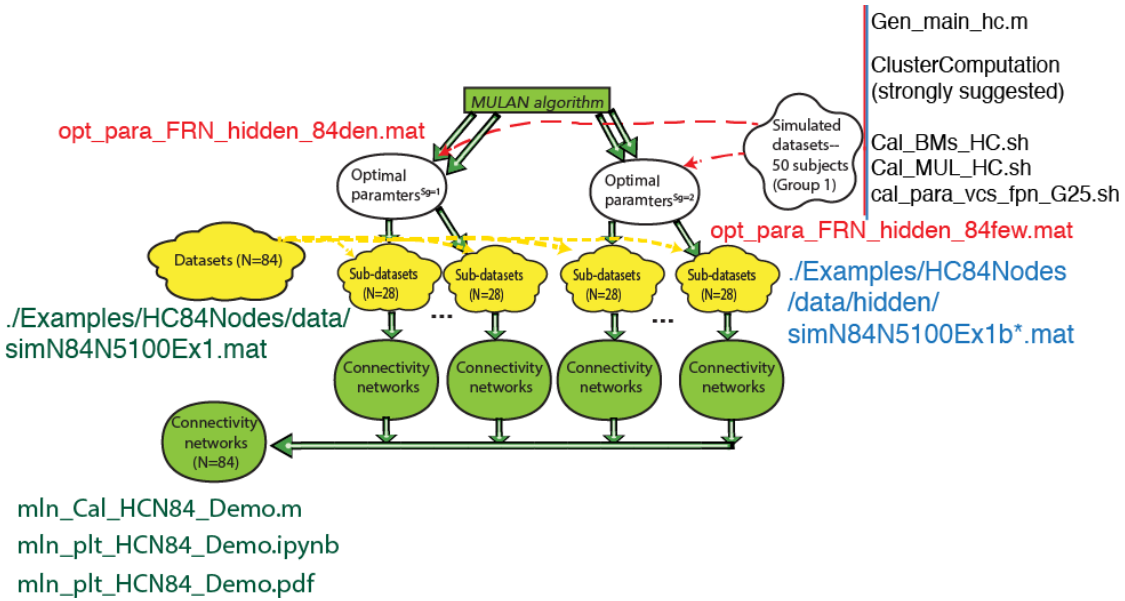


FIGURE 3. Flowing chart for MLAN application on human connectome based datasets using sub-datasets for Demo 3. Given simulated SEEG dataset and an optimal parameter file in folder $./Examples/HC84Nodes/$, please run $mln\_Cal\_HCN84\_Demo$ to obtain 15 sub-datasets with 28 nodes, then the basic methods and MULAN results by using two different optimal parameters for 2 different situations. For demonstrating the results, please run ipython code: $mln\_plt\_HCN84\_Demo.ipynb$ or just open the corresponding pdf file to review all the codes and results. Again the optimal parameters are obtained by multiple datasets with other unrelated 50 datasets.

2.3. **Demo 3: Application on a human connectome based dataset.**

Example 2.3. *Given a human connectome based dataset as* $./Examples/HC84Nodes/data/simN84N5100Ex1.mat$*, and two optimal parameter files:* $opt\_para\_FRN\_hidden\_84den.mat$ *for dense sub-groups when all nodes are within one hemisphere and* $opt\_para\_FRN\_hidden\_84few.mat$ *for dense sub-groups when the nodes are across two hemispheres.*

run $mln\_Cal\_HCN84\_Demo.m$

Results:

- *Generate 15 sub-datasets having 28 nodes: ./Examples/HC84Nodes/hidden/data/simN84N5100Ex1b*.mat; where b* is for 15 subnetworks*
- *The results for basic methods of 15 files: ./Examples/HC84Nodes/hidden/ToutResults/ Tout_1500_simN84N5100Ex1b*.mat*
- *The results for MULAN of 15 files: ./Examples/HC84Nodes/hidden/ToutResults/ Adp_1500_simN84N5100Ex1b*FRN_G5_tak.mat*

run $mln\_plt\_HCN84\_Demo.ipynb$

Or ***open*** $mln\_plt\_HCN84\_Demo.pdf$

Results: saved the connectivity results for the network with 84 nodes in folder $./Examples/HC84Nodes/hidden/Con/$ *:*

- *mlnRsimN84N5100Ex1.mat for MULAN results.*
- *PDCsimN84N5100Ex1.mat for basic method PDC.*
- *BCorrDsimN84N5100Ex1.mat for basic method BCorrD.*

R Fig. 3 shows Demo 2. The codes can be found in main folder, but the data and results are in folder $./Examples/HC84Nodes/$.

R Fig. 4 and Fig. 5 shows how the ipython codes and part of results which you can find both in the ipython notebook and pdf file, which includes all results from each subnetwork but not included in paper for saving space.

```
    mln_plt_Mat_simlated(gtMat,nc,labelname,fig,subplot=121,title='Ground truth network',mcolorm

    mln_plt_Mat_simlated(fmlnMat/2,nc,labelname,fig,subplot=122,title='Calculated MULAN network'
    fig.text(0.35,0.8,'Results for Subnetwork No.'+str(inbgv+1), fontsize = 22, color ='DarkBlue
#fig.savefig(basicdir+'ground_Net_N_L+SG1''.eps',dpi=300)
```

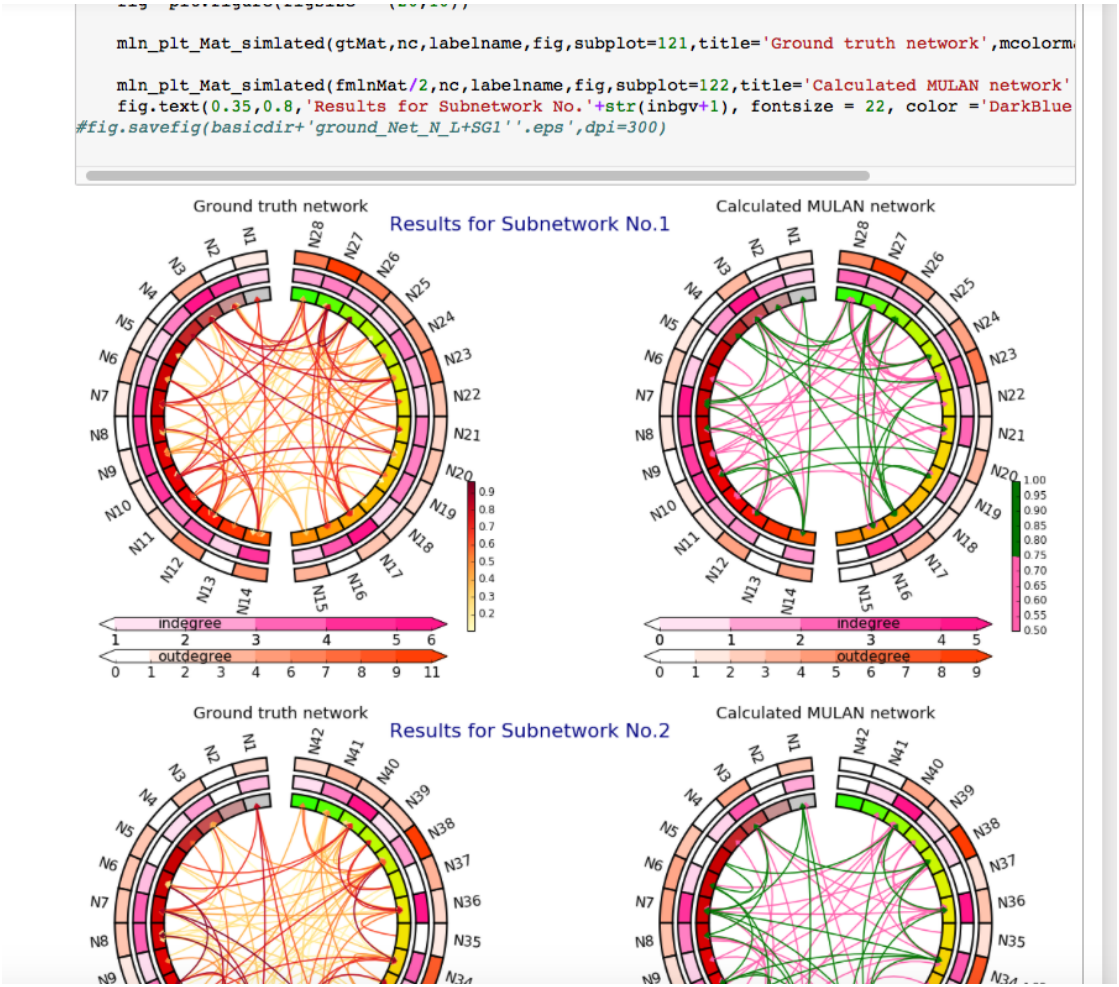
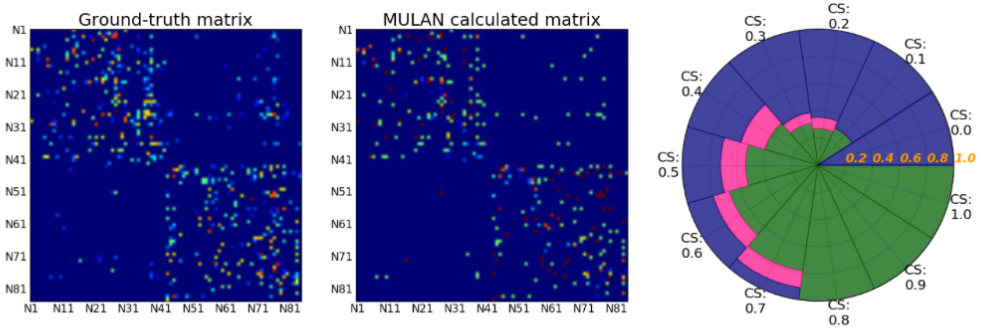


FIGURE 4. A part of ipython notebook codes and results for Demo 3.

```
ax2.set_yticklabels(labelxN84,size=16)
ax2.set_title('MULAN calculated matrix',size=25)
#cbar=plt.colorbar()
#cbar.ax.tick_params(labelsize=20)

ax3=subplot(133,polar=True)
tptnVectorR, tptnVector, pnVector=mi.mln_tptn(fNchanMat,N84gtMat,0.2,1)
md.mln_polar_bar_hf('try',tptnVectorR,ax=ax3)
savefig(savedatadir+datasets+'matric_pie.eps',dpi=300)
```



In [14]:
```
ithethod = [0.1,1]
imlnMat=fNchanMat.copy()
imlnMat[imlnMat<ithethod[0]]=0.
imlnMat[imlnMat>=ithethod[1]]=2.
imlnMat[(imlnMat>=ithethod[0])*(imlnMat<ithethod[1])]=1.
fMat=imlnMat-np.diag(imlnMat.diagonal())

fig= plt.figure(figsize = (20,10))
mln_plt_Mat_simlated(N84gtMat,nchan,labelnameN84,fig,subplot=121,title='Ground truth network',mcolormap='YlOrRd')
#mln_plt_Mat_simlated(fmlnMat[0,:,:],nc,fig,subplot=132,title='Calculated MULAN network',cg=1)

mln_plt_Mat_simlated(fMat/2,nchan,labelnameN84,fig,subplot=122,title='Calculated MULAN network', mcolormap='mulan',cg=0)
savefig(savedatadir+datasets+'mlnNetWork.eps',dpi=300)
```

FIGURE 5. A part of ipython notebook codes and results for Demo 3.

3. Cluster Computation

We apply MULAN for thousands of datasets, which is a benefit from cluster computation. Here we give 4 files to demonstrate 4 main steps to deal with multiple datasets. All there files can be found in the main folder.

3.1. Step 1: Generation of multiple datasets.

Example 3.1. *Generation of multiple datasets.*

run $mln\_main\_few.m$

Results:

- *Generate multiple sub-datasets : you can set the lengths of dataset, type of datasets, and folder etc.*

3.2. Step 2: Calculation of results from the basic methods.

Example 3.2. *Given multiple datasets, to calculate the results from the given basic methods family.*

run $Cal\_BMs\_HC.sh$

Results:

- *Generate all results from basic methods for all datasets in the folder* $./mlnData/ToutResults/Tout$

R Fig. 6 shows a bash file. The compiled code: $./run\_mln\_Cal\_MULAN\_fs\_mlnvcs\_wins\_BM.sh$ can be generated by mcc -m $mln\_Cal\_MULAN\_fs\_mlnvcs\_wins\_BM.m$ in Matlab. If you want to change the calculation parameters, just edit the m.file and compile it for cluster computation.

3.3. Step 3: Calculation of optimal parameters.

Example 3.3. *Given multiple datasets, to calculate the optimal parameters.*

run $cal\_para\_vcs\_fpn\_G25.sh$

Results:

- *Generate all results of the optimal parameters for all datasets in the folder* $./mlnData/ToutResults/Fit_*$.

R This step must run after the step 2 .

3.4. Step 4: Calculation of MULAN results.

Example 3.4. *Given multiple datasets and optimal parameters, to calculate MULAN results.*

run $Cal\_MUL\_HC.sh$

Results:

- *Generate MULAN results for all datasets in the folder* $./mlnData/ToutResults/Adp_*$.

R This step must run after the step 2 and 3.

```
#!/bin/bash
resource="walltime=00:30:00"
property="host>='n02'"



flagTF=T
flagData=M

for basicdir in mlnData/
do

wins=1500

dirname=$basicdir'Win'$wins/
if [ ! -e $dirname ]; then
mkdir $dirname;
cp -r $basicdir'data' $dirname;
fi

a=`ls $dirname'data'`


for prename in $a
do

program="./run_mln_Cal_MULAN_fs_mlnvcs_wins_BM.sh /soft/mcr/v85 $dirname $prename
    $wins"
oarsub -p "$property" -l "$resource" "$program"

done
done

```

FIGURE 6. One example of bash files for step 2.

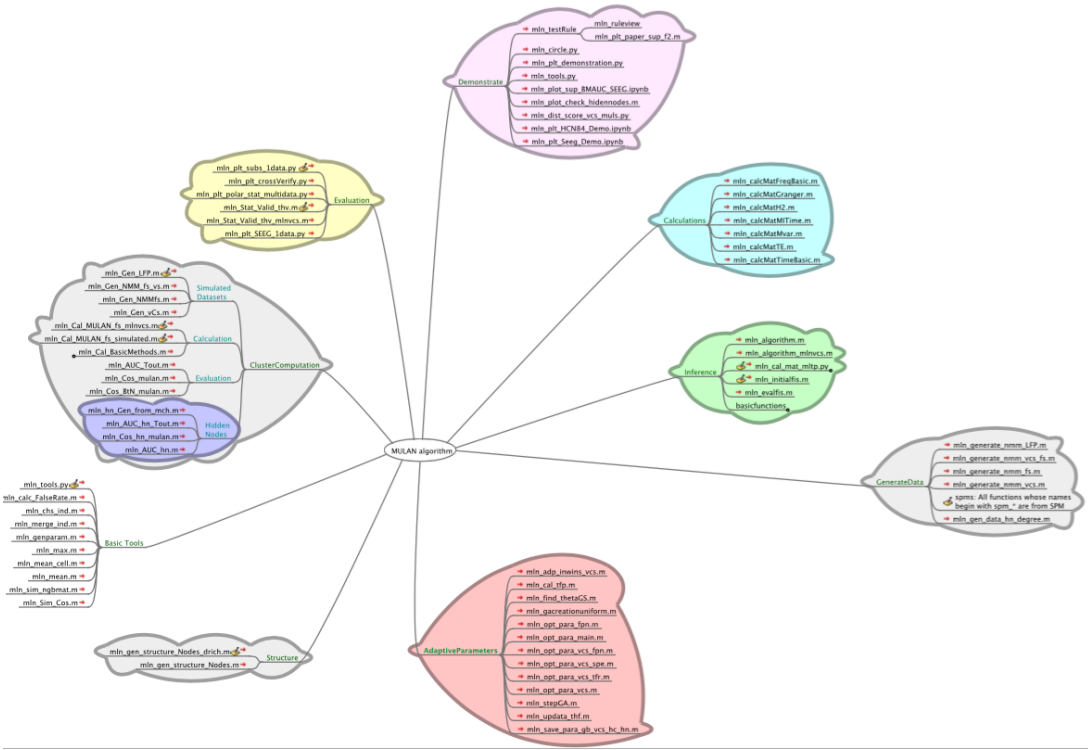


FIGURE 7. The structure of the MULAN codes from the file MULAN algorithm codes.mm which in the root folder of MULAN. This file was edited by Free Mind Version:1.0.1 from http://freemind.sourceforge.net/ and includes hyperlinks to the target codes.

4. Main Stuctures

4.1. **Simulated Datasets.** The main functions for generating simulated datasets are in folder "GenerateData", and the corresponding functions for cluster computation are in folder "ClusterComputation".

Example 4.1. *To generate Neural Mass Models (NMM) datasets with different connection strengths (CS), specifying minimum CS, maximum CS and a given sampling frequency:*

ClusterComputation/mln_Gen_NMM_fs_vs.m;
GenerateData/mln_generate_nmm_vcs_fs.m

e.g. *mln_Gen_NMM_fs_vs Examples nmm 20 1 810 0.1 1 10 250*

Results: Examples/data/nmmN20L10CS10100S1N810.mat

Example 4.2. *To generate LFP datasets from Neural Mass Models with the identical CS specifying CS:*

ClusterComputation/mln_Gen_LFP.m; GenerateData/mln_generate_nmm_LFP.m

e.g. *mln_Gen_LFP Examples nmm 20 1 600 1 10*

Results: Examples/data/nmmN20L10CS100S1N600.mat

R Fig. 8 shows the $.m$ files in folder GenerateData.

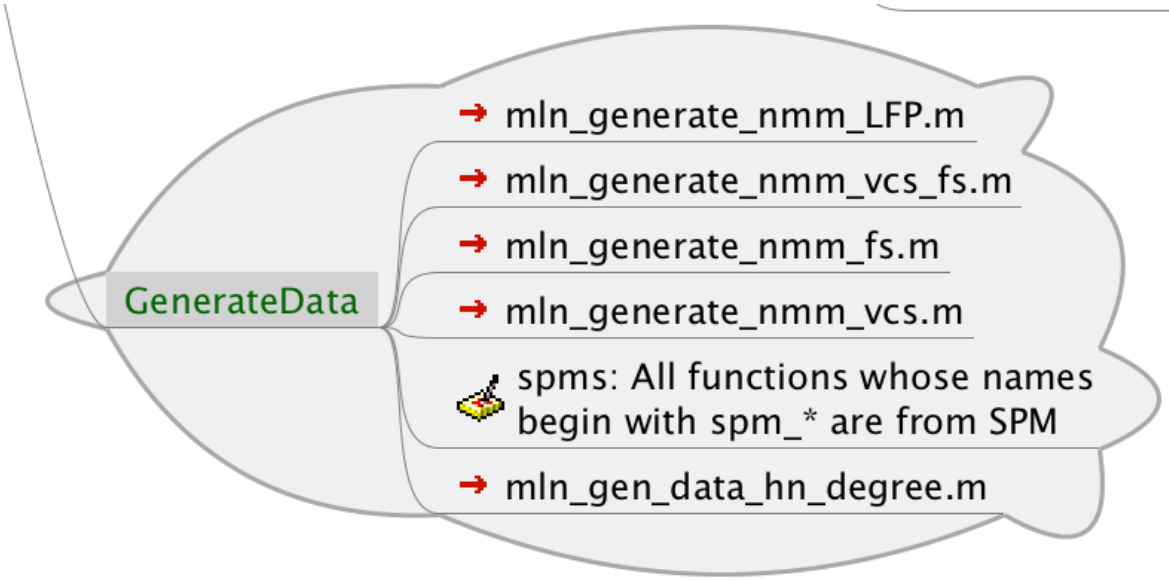


FIGURE 8. Files in folder GenerateData. Note that the files, named in a format such as spm_*, are the codes from SPM (http://www.fil.ion.ucl.ac.uk/spm/).

TABLE 1. File explanation in GenerateData.

Files.	Explanation
$mln\_\cdots\_LFP$	To generate LFP signals for graphs with a specific CS
$mln\_\cdots\_vcs\_fs$	To generate NMM signals for graphs with varying CS and a given sample frequency
$mln\_\cdots\_vcs$	To generate NMM signals for graphs with varying CS
$mln\_\cdots\_fs$	To generate NMM signals with a given sample frequency
$mln\_\cdots\_hn_degree$	To generate NMM signals with hidden nodes, see Section 4.7.

4.2. **Calculation.** The main functions for computing examples are in folder "ClusterComputation". These functions can calculate the connection matrices using basic methods and then using MULAN algorithm. For the simulated datasets, AUC and COS are calculated.

Example 4.3. *Connection matrices generated by MULAN algorithm and basic methods.*

ClusterComputation/mln_Cal_MULAN_fs_mlnvcs;

Parameters:

VGroupMethlog: *a subset of methods families, such as 'TimeBasic', 'Granger', 'FreqAH', 'TE'. The connectivity families and their methods can be found in Table 2.*

calParams: *The Parameters to be set when using basic methods.*

BM: *The structures specifies the basic methods BM1-BM2, BM3 and BM4.*

MULAN parameters: *The parameters th1 and th2 in the codes are for θ_1 and θ_2 in the paper. The other parameters that can be set are: kforMF, the k-value for membership functions; nboot, the number of bootstrapping windows; and nboot2, the number of times MULAN scores are calculated.*

e.g. *mln_Cal_MULAN_fs_mlnvcs Examples nmmN20L10CS10100S1N810 0.5 0.7 T M*

'T' for Basic methods and 'M' for simulated datasets.

Results: Examples/ToutResults/Tout_nmmN20L10CS10100S1N810.mat (results)

Examples/ToutResults/AUC_nmmN20L10CS10100S1N810.mat (evaluation of the results)

e.g. *mln_Cal_MULAN_fs_mlnvcs Examples nmmN20L10CS10100S1N810 0.5 0.7 F M*

'F' for MULAN algorithm and 'M' for simulated datasets.

Results: Examples/ToutResults/Sta_nmmN20L10CS10100S1N810.mat (MULAN results)

Examples/ToutResults/COS_Sta_nmmN20L10CS10100S1N810.mat (evaluation of MULAN results)

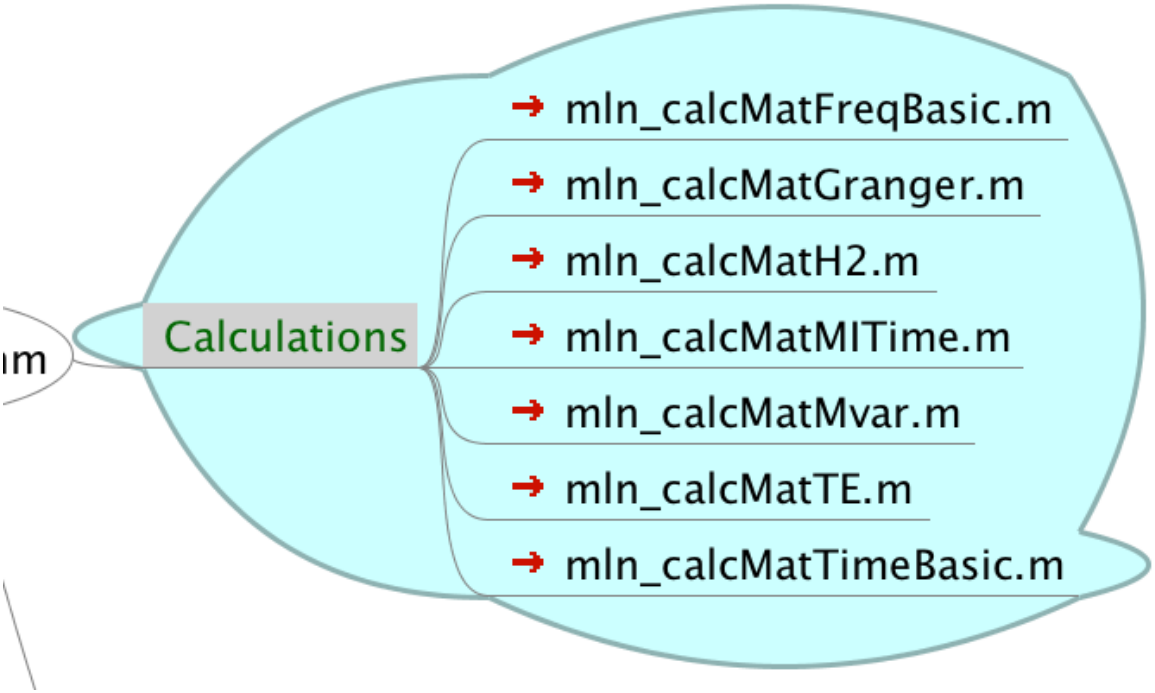


FIGURE 9. Code files in folder Calculations to calculate the results obtained with 40 basic methods from 7 families.

TABLE 2. Methods list: Abbreviation and notations of the connectivity analysis methods. Please refer to the submitted paper for the mathematical definitions.

Abbr.	Notations	Abbr.	Notations
	Correlation		
BCorrU	Bivariate correlation undirected	PCorrU	Partial correlation undirected
BCorrD	Bivariate correlation directed	PCorrD	Partial correlation directed
	h^2		
Bh^{2}U	Bivariate h^2 undirected	Ph^{2}U	Partial h^2 undirected
Bh^{2}D	Bivariate h^2 directed	Ph^{2}D	Partial h^2 directed
	Mutual Information		
BMITU	Bivariate mutual information undirected	PMITU	Partial mutual information undirected
BMITD1	Bivariate mutual information directed 1	PMITD1	Partial mutual information directed 1
BMITD2	Bivariate mutual information directed 2	PMITD2	Partial mutual information directed 2
	Coherence		
BCohF	Bivariate Fourier transforms	PCohF	Partial Fourier transforms
BCohW	Bivariate wavelet transforms	PCohW	Partial wavelet transforms
	Granger family		
GC	Granger causality	PGC	Partial Granger causality
CondGC	Conditional Granger causality		
	Transfer Entropy		
BTEU	Bivariate transfer entropy undirected	PTEU	Partial transfer entropy undirected
BTED	Bivariate transfer entropy directed	PTED	Partial transfer entropy directed
	$\bar{A}\mathcal{H}$		
Af	$\bar{A}$ in the frequency domain	hmvar	$\mathcal{H}$ in the frequency domain
PDC	Partial directed coherence	DTF	Directed transfer function
PDCF	Partial directed coherence factor	DC	Directed coherence
GPDC	Generalized partial directed coherence	ffDTF	Full frequency directed transfer function
GGC	Gewekes Granger Causality	dDTF	Direct directed transfer function
PCOH1	Partial ordinary coherence 1	COH1	Ordinary coherence 1
PCOH2	Partial ordinary coherence 2	COH2	Ordinary coherence 2
MAVR	$\bar{A}$ in the time domain	Smvar	Power Spectrum
AS	$\bar{A}$ square		

4.3. **Inference.** In the previous section, we explained the codes to calculate examples with basic methods. In this section, we focus on the codes for the inference part of MULAN algorithm.

Fig.10 shows the structure of inference.

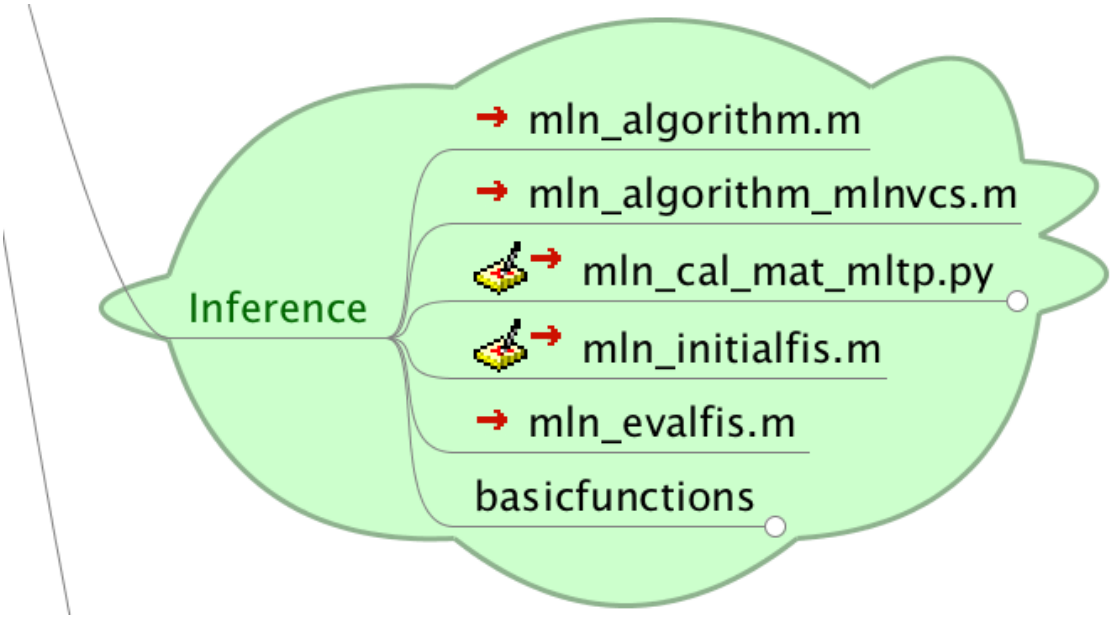


FIGURE 10. First level files in folder Inference which contains the code to perform the MULAN inference computation

TABLE 3. Details of files in the "Inference" folder.

Files.	Explanation
mln_algorithm.m	Matlab code to get MULAN connection matrix for connection strengths larger tha Inputs include the averaged connection matrices obtained from basic methods The θ_1,θ_2, k for the membership functions and basic methods
mln_algorithm_mlnvcs.m	Matlab code to obtain MULAN connection matrix for all connection strengths
mln_cal_mat_mltp.py	Python code to obtain MULAN graph when CSs are not identical, see Fig. 11
mln_initialfis.m	Matlab code to initiate the fuzzy inference system
mln_evalfis.m	Matlab code to output the results of the fuzzy inference system

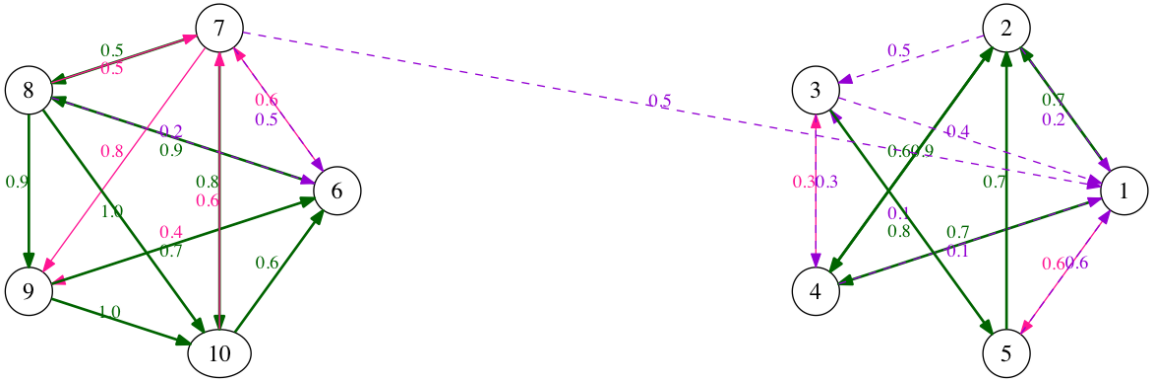


FIGURE 11. An example result from *mln_cal_mat_mltp.py*.

In order to adjust off-line and understand the MULAN inference system, we have created an interface of the MULAN inference system analyzing a single link. The function can be found at *"Demonstration/mln_testRule.m"* (Fig. 12) and the interface is shown in (Fig. 19)

FIGURE 12. The structures pointing to the main functions in order to adjust MULAN rules.

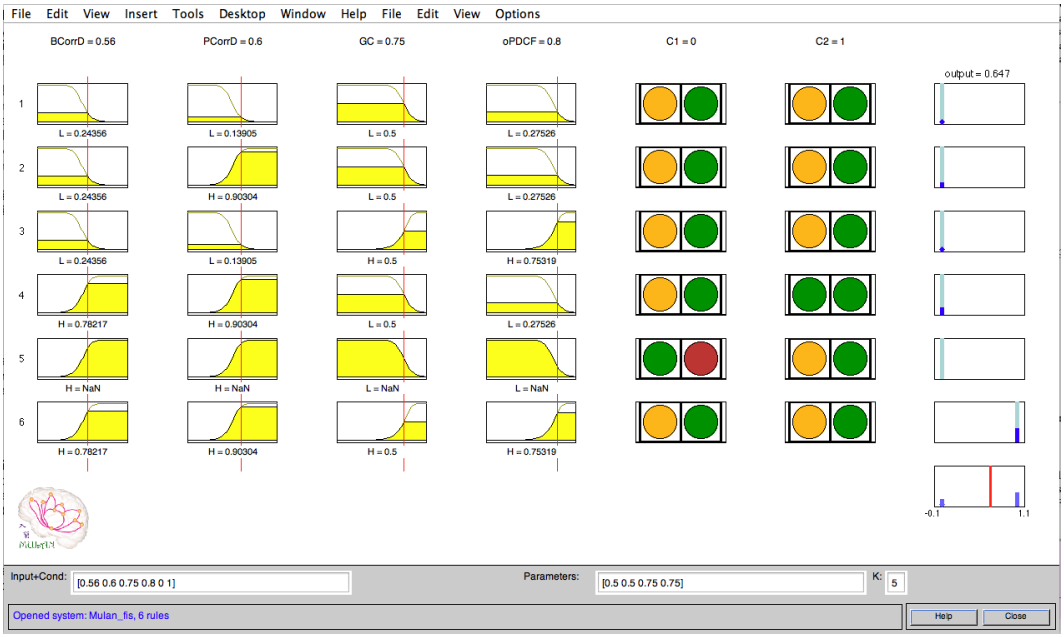


FIGURE 13. A graph user interface from $mln\_testRule.m$.

4.4. **Adaptive Parameters.** This section we will demonstrate all codes for calculating the optimal parameters based on genetic algorithms.

Fig.14 shows the structure of adaptive parameters.

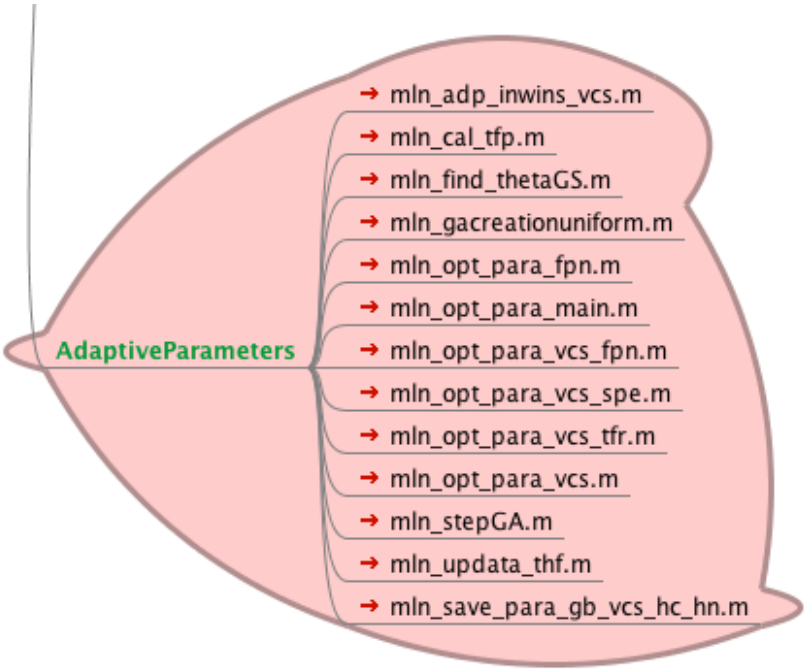


FIGURE 14. First level files in the folder AdptiveParameters which contains the code to perform the MULAN optimization computation

4.5. **Evaluation.** The functions in the folder *"Evaluation"* are used to output MULAN results when considering the statistical results, variable connection strengths, hidden nodes and cross-validated connectivity graphs. The main functions are shown in Fig. 15 and explained in Table 4.

The parameter values, used as examples, can be found inside the files.

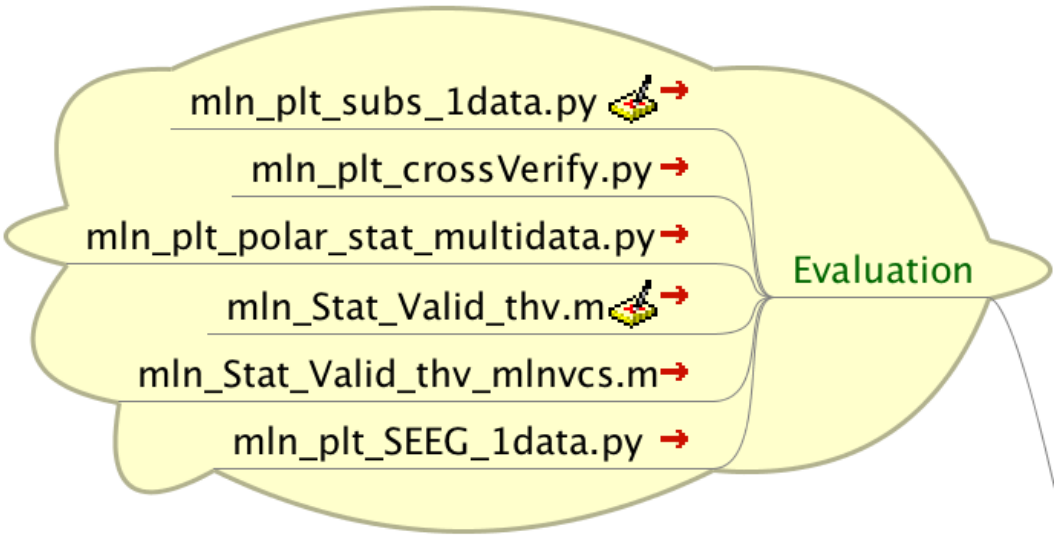


FIGURE 15. Files in folder Evaluation

TABLE 4. Explanation of files in "Evaluation" folder.

Files.	Explanation
$mln\_plt\_subs\_1data.py$	outputs all graphs obtained with a given dataset and their evaluation results
$mln\_plt\_crossVerify.py$	Python code to output a cross-validated connectivity graph Fig. 16 and an evaluation result when using a simulated dataset with known ground-truth structure.
$mln\_plt\_polar\_stat\_multidata.py$	outputs an evaluation result for multiple datasets.
$mln\_plt\_SEEG\_1data.py$	outputs MULAN results for real datasets, for example, SEEG datasets used in the paper.

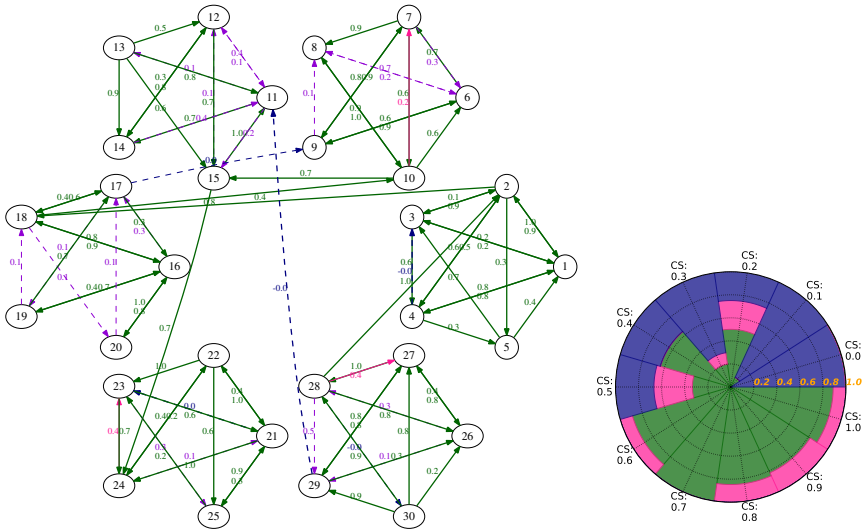


FIGURE 16. A cross-validated connectivity graph and its evaluation result from $mln\_plt\_crossVerify.py$.

4.6. **Demonstration.** The demonstration folder includes the functions generating the figures to visualize the results. The main functions are shown in the Fig. 17 and explanation is in Table ??.

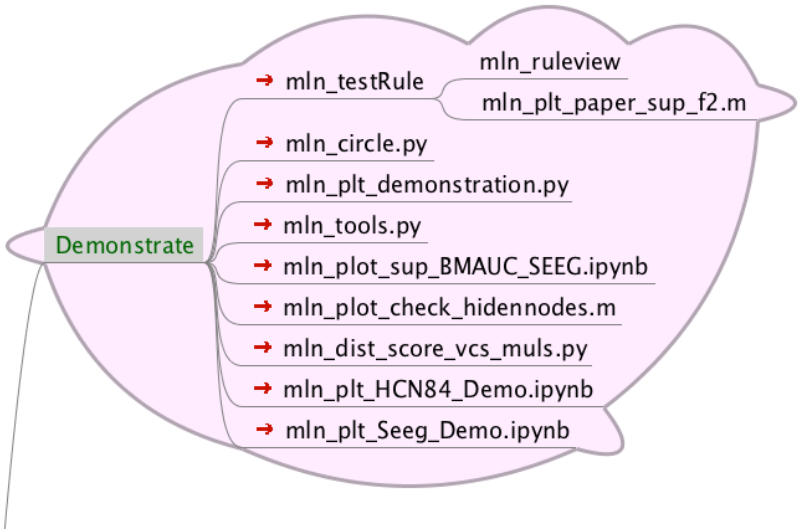
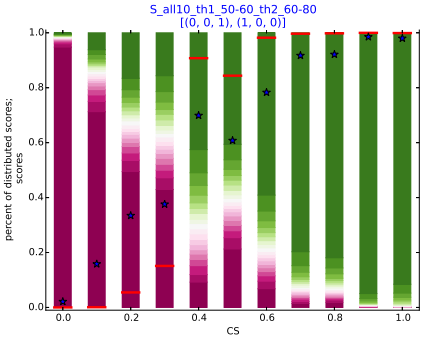
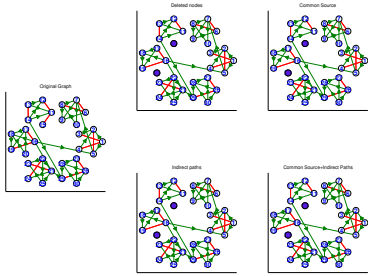


FIGURE 17. Files in folder Demonstration.

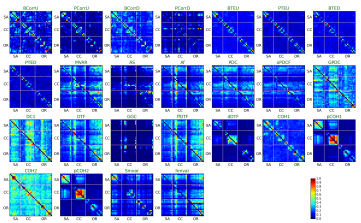


(A) *mln_dist_score_vcs_muls.py.*



(B) *mln_plot_check_hidennodes.m.*

(C)
mln_dist_score_vcs_muls.py.



(D)
mln_plot_sup_BMAUC_SEEG.ipynb.

FIGURE 18. Examples obtained with the functions in folder Demonstration.

4.7. **Cluster computation.** All previously described functions can be used for cluster computation or using a single processor.

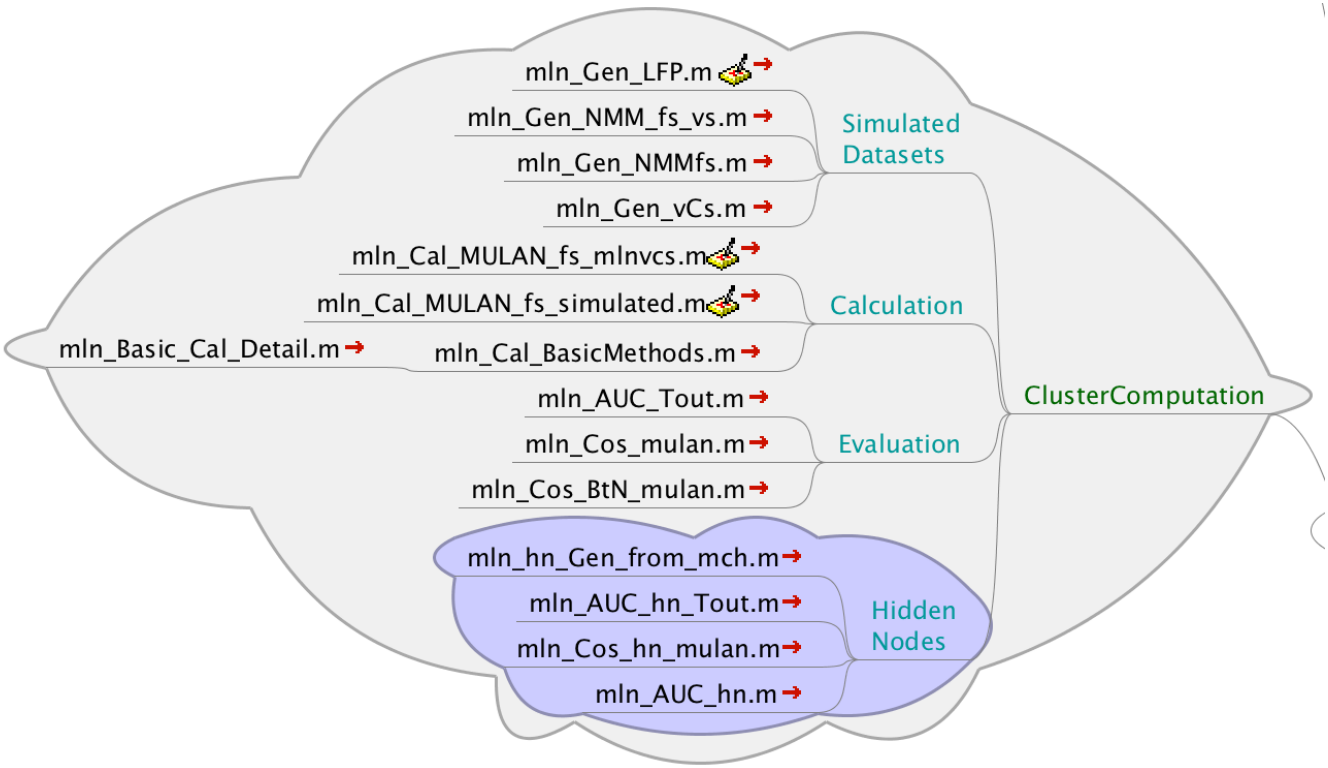


FIGURE 19. Files in folder ClusterComputation.

Appendix A. MULAN data structures

This section describes how to set the data structures of the datasets.

Please refer to the structures of the example dataset as an example:

'./Examples/AdapData/data/erpN30S10N2010ods4.mat'. Data structures:

- Data (Obligation): $nchannel \times timeseries$ double. $nchannel$ is the number of signal channels.
- Params (Obligation): structure
  - fs (Obligation): sample frequency
  - str (Option): $1 \times nchannel$ list of the names of the channels. If this field does not exist or is empty, the names will be numbered.
- Connectivity(Option): $nchannel \times nchannel$ double $M_{i,j}$. The value $M_{i,j}$ is the connection strength from channel j to channel i.

Note that the structure "Params" can include many fields but "fs" is only mandatory field for MULAN Toolbox. The variable 'Connectivity' is only necessary for simulated datasets if the evaluation codes are necessary.

Appendix B. MULAN 1: Interface

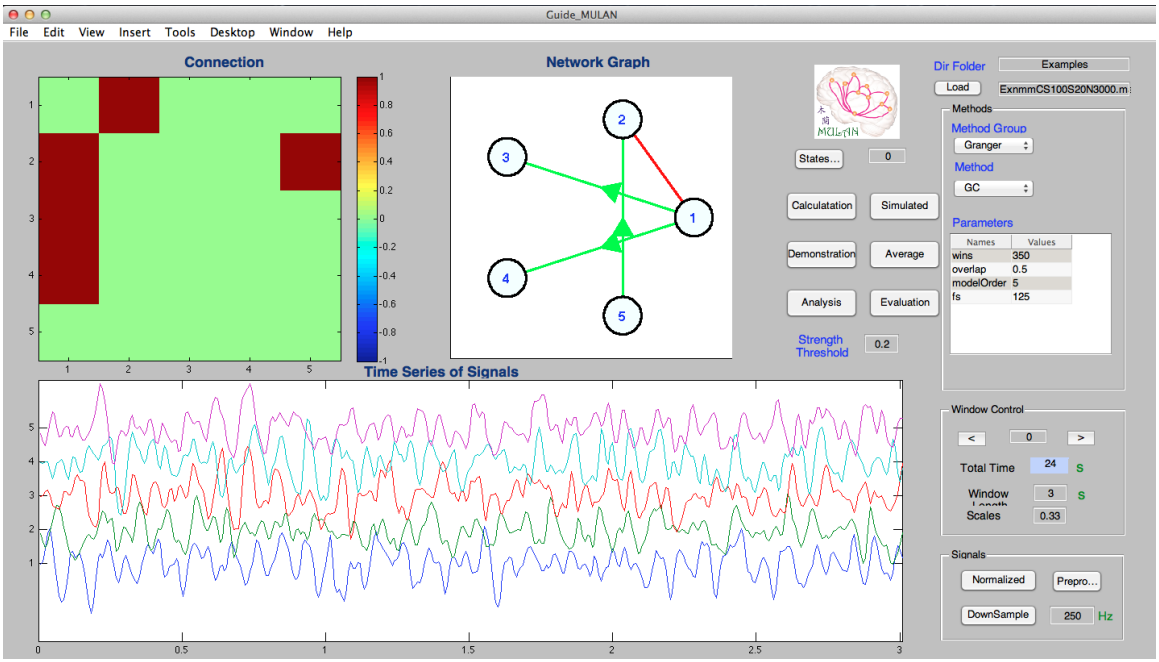


Figure 20. The interactive graphic user interface (GUI) of MULAN1. In this GUI, the top panels are the connection matrix and network graph, which are the results from given methods and parameters set by the Methods panel. The time series of signals of the given dataset (bottom) can be set by the window control panel.

MULAN 1 are available freely from in github: https//github.com/HuifangWang/MULAN.

Start MULAN by two simple options.

Option 1: : Just type "Guide_MULAN" at the Matlab command line and hit enter.

Option 2: : Open file Guide_MULAN.m in MULAN folder, then click the button RUN.

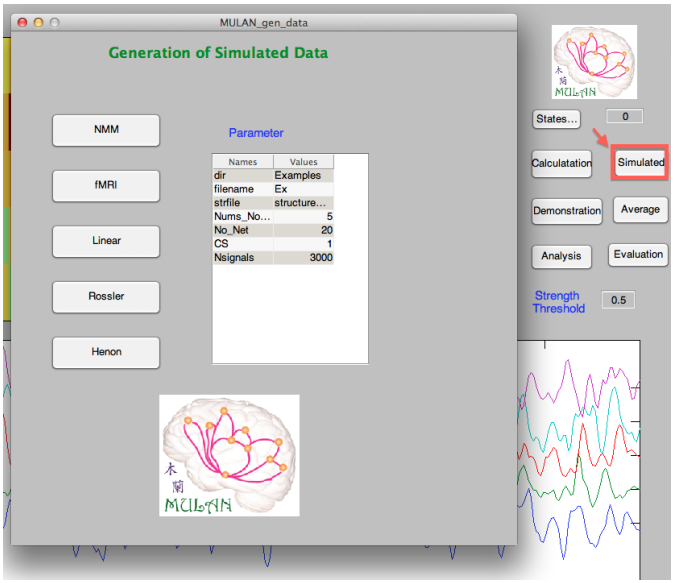


FIGURE 21. Generation of Simulated Data.

B.1. **Simulated data.** The parameters which we can choose are:

dir : the folder which are used to store the results.

filename : the base name you choose; the software will add more details on the filenames of each dataset. For example, 'nmm' for model type, CS for connection strength.

strfile : the structure file which stores the information about underlying structures.

Nums_Node : The numbers of nodes for the datasets

CS : The connection strengths

Nsignals : The lengths of signals which you would like to generate.

Then choose the types of models listed on the left, for example, click NMM button to generate NMM datasets. There is the message dialog to inform you if it has finish successfully or not.

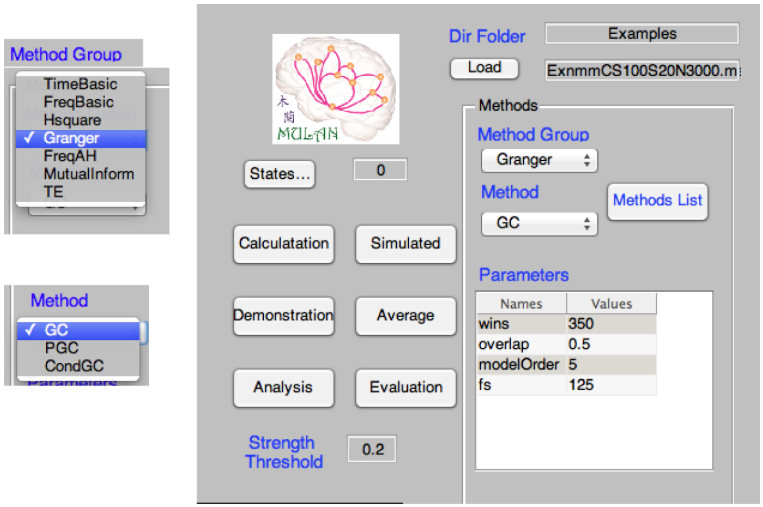


FIGURE 22. setting Panels

B.2. **Setting Panels.**

First we need to specify the following terms, ref to Fig. 22.

Dir Folder : We put the folder name that has the data file we are about to analyze.

File name : We put the file name in which data stores. Put the name of data file and then click Load button.

Method Group : We need choose the method family which we want to use.

Method : We can specify a method from a specified family. We use Granger family as an exmaple in Fig. 22. A methods list on the right provides the abrreviation and corresponding notations about the methods.

Parameters : Once we've chosen the method family, the parameters which need to be specified will be listed in a table.

B.3. **Calculation of connection matrices.** (Fig. 23).

(1) After specifying the data file, method and calculation parameters, click the button Calculation.

(2) Once the calculation finishes, the massage dialog will pop up to inform you that the calculation completed.

(3) Then you will find your results in the same folder of the data with a subfolder Results, which will allow you to access your results afterward so that you can save calculation time when the method parameters are the same.

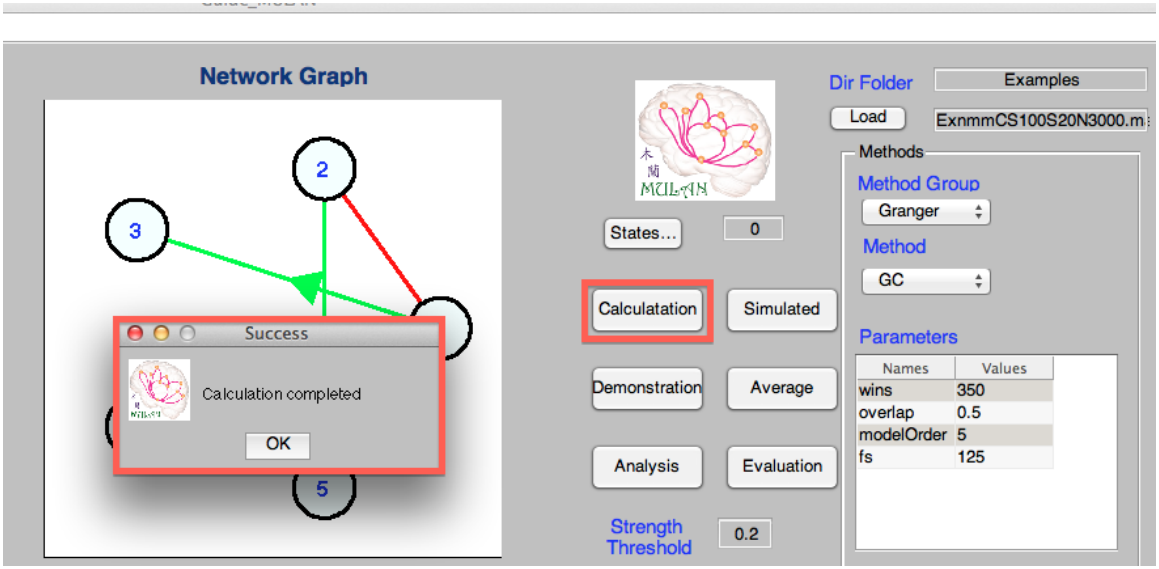


FIGURE 23. Calculation of the computed connection matrices by choosing Methods and Parameters

B.4. **Demonstration of the results.**

After calculation of the computed connection matrices, the results can be visualized (Fig.24):

Step 1 Choose the threshold of connection strength in $[0, 1]$.

Step 2 Click the button Demonstration will show the connection matrix and graph corresponding to the data segment during the current windows.

Step 3 Choose current data segment by directly inputing the current start time point, or by clicking backward/forward buttons.

Step 4 The three windows, Connection, Network Graph and time-series of signals, will change accordingly.

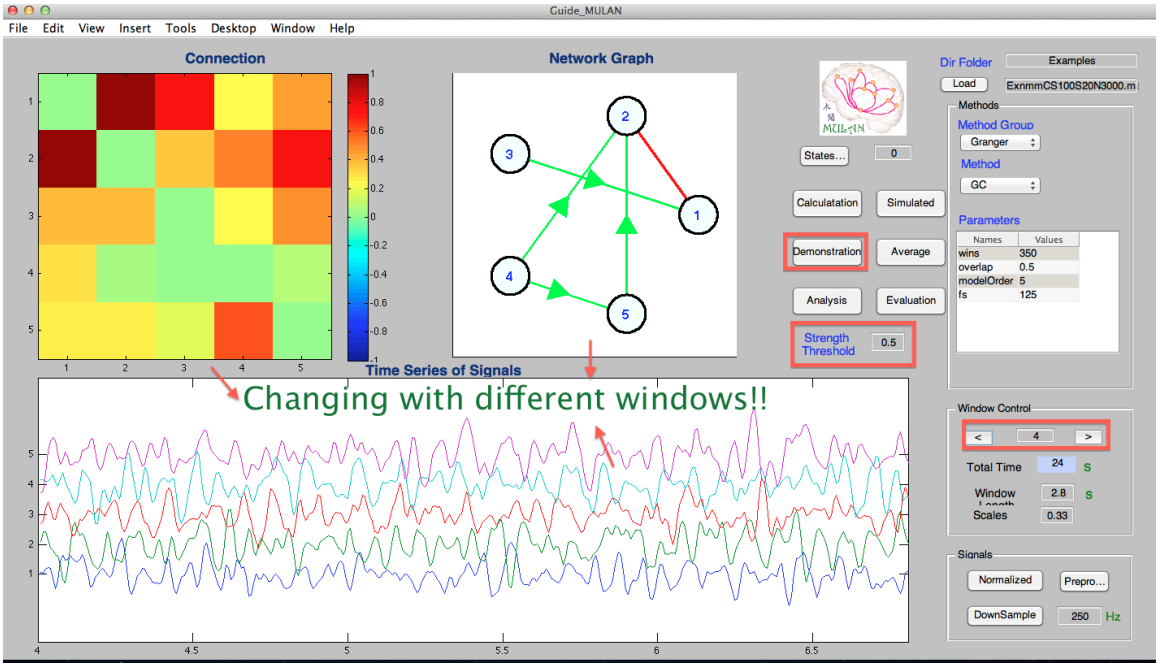


FIGURE 24. Demonstration of the results.

B.5. **Demonstration of average results.**

As we suggested in the paper, the average value is more stable than the results from a single window. In order to view average results over the whole dataset, please click the button Average (Fig.25).

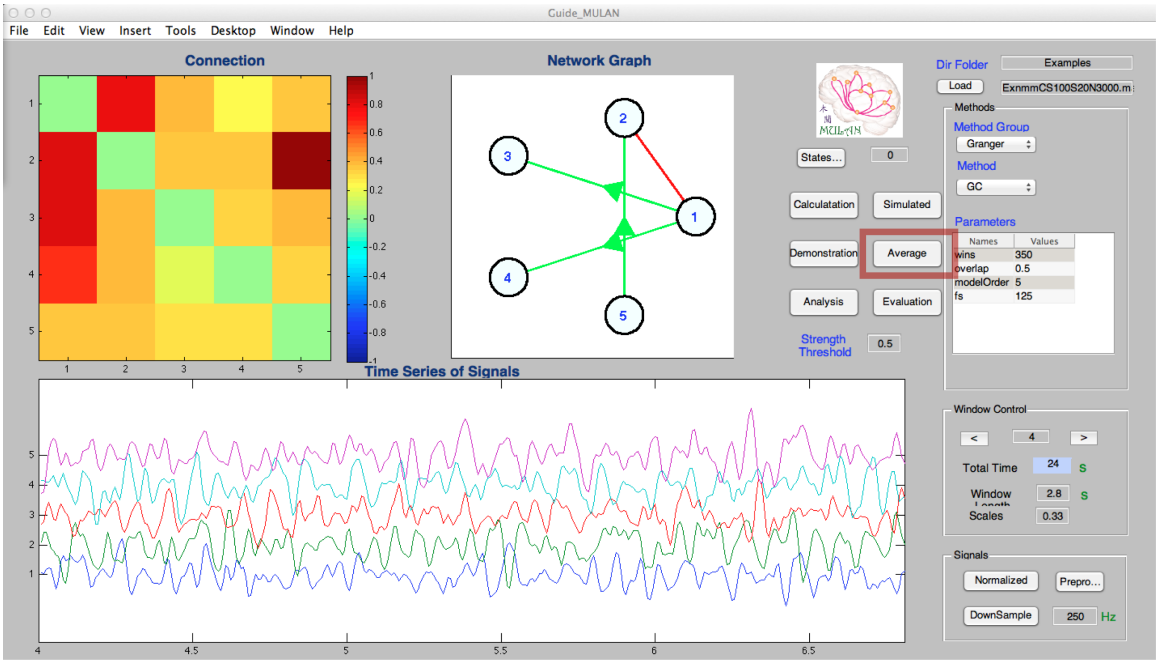


FIGURE 25. Demonstration of the results.

B.6. **Detailed analyses.**

MULAN can provide detailed results by clicking button Analysis. A pop-up figure describes the connectivity strengths between the pairs of channels as function of windows shown in blue and average value over all windows shown in purple, as shown in Fig. 26.

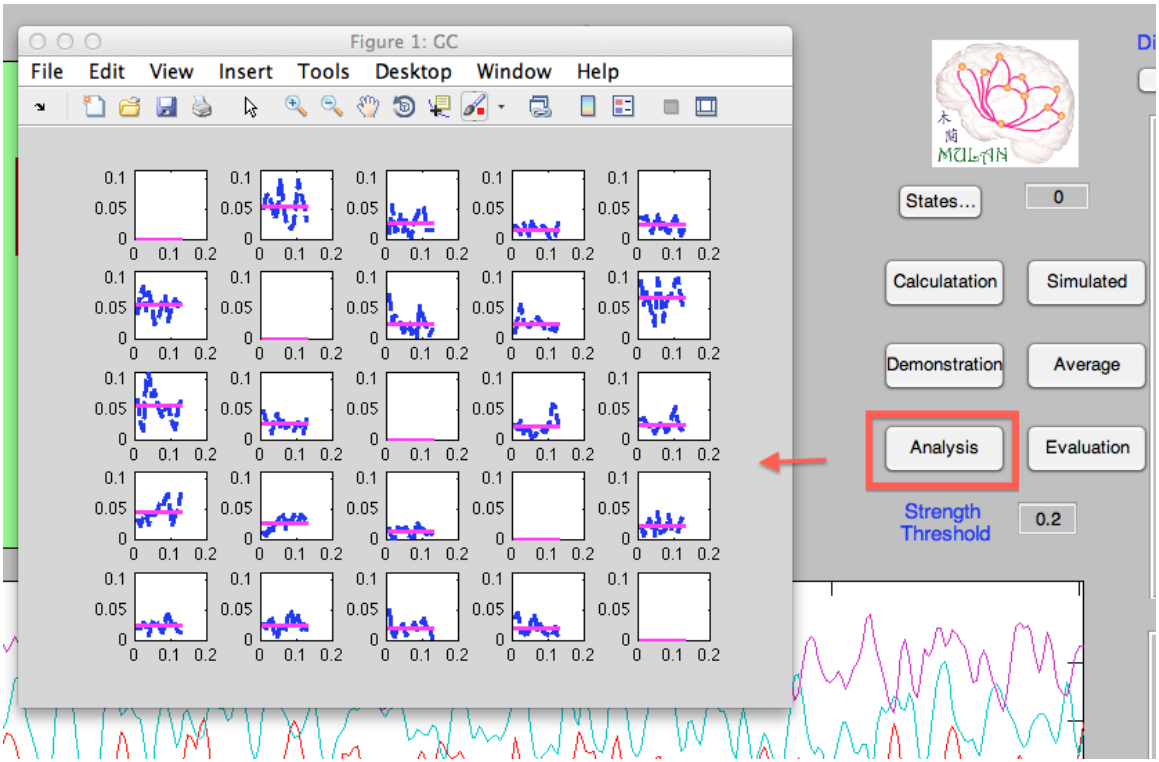


FIGURE 26. Detail results on all windows.

B.7. **Evaluation of methods.**

Click the button Evaluation to bring up a figure displaying evaluation results of methods. Note that the Evaluation button only works in the cases of known ground truth structures. Please refer to Fig. 27.

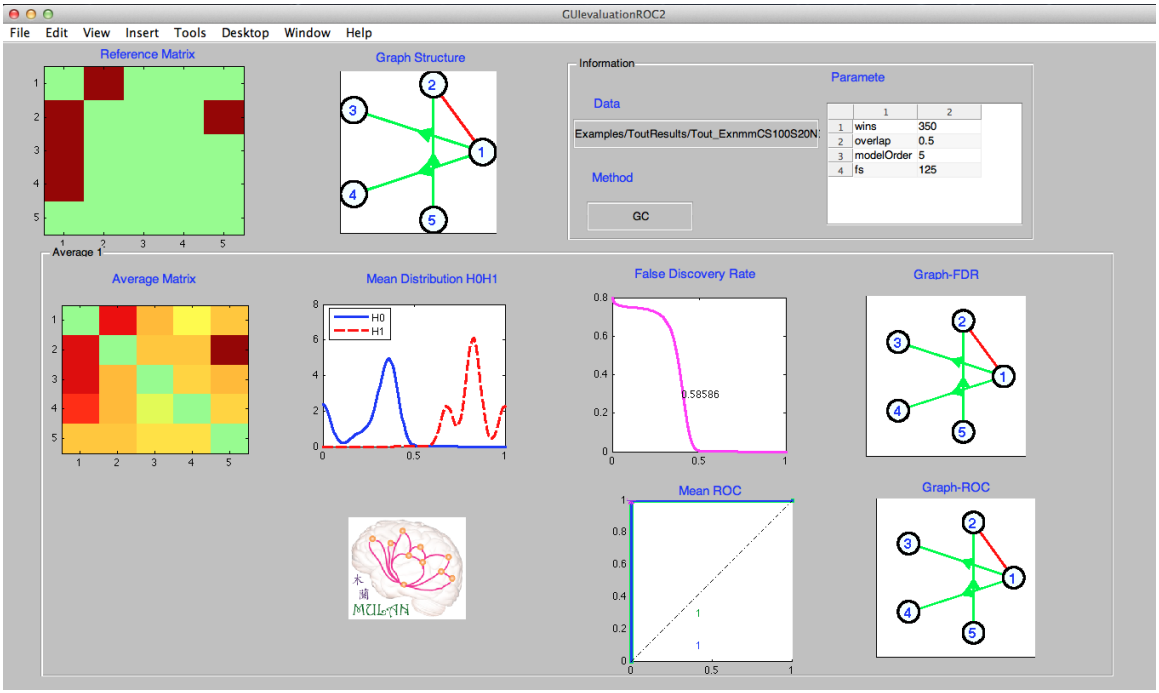


FIGURE 27. Evaluation of results from Granger causality.

Fig. 27 contains the results about *average matrix*, H_0/H_1 *distributions*, *the false discovery rate (FDR)* and *the receiver operating characteristic (ROC)*. Two graphs, to the right of FDR and ROC, are shown by choosing thresholds from two ways: 1) From FDR where FDR=0, 2) From ROC where the optimal threshold corresponds to the point closest to the upper left corner.

Acknowledgements

We are very thankful to the following open source toolboxes from which we were able to derive functions and methods:

- Statistical Parametric Mapping (SPM), from the wellcom Trust Centre for Neuroimaging, available at `http://www.fil.ion.ucl.ac.uk/spm/`
- Granger Causal Connectivity Anlysis toolbox, from Anil K.Seth, available at `http://www.anilseth.com/`.
- BIOSIG-toolbox, from Alois Schloegl, available at `http://biosig.sf.net/`.
- BSMART: A Matlab/C Toolbox for Analyzing Brain Circuits, from Jie Cui, Lei Xu, Steven L. Bressler, Mingzhou Ding, Hualou Liang, available at `http://www.brain-smart.org/`.
- Neurophysiological Biomarker Toolbox (NBT), from Simon-Shlomo Poil, `http://www.nbtwiki.net/`.