Graph Theoretic General Linear Model (GTG)

Software website: www.nitrc.org/projects/metalab_gtg/
Author: Jeffrey M. Spielberg (jpielb2@gmail.com, http://sites.bu.edu/metalab/)
Current version: Beta 0.45 (03.30.16)

Overview:

The main function of this Matlab toolbox is to run a GLM on graph theoretic network properties computed from brain networks. The GLM accepts continuous & categorical between-participant predictors and a categorical within-participant (i.e., repeated measures) predictor. Significance is determined via non-parametric permutation tests. The toolbox allows for computation/testing of both fully connected and thresholded (binarized or not binarized, based on a range of thresholds) networks. Correction for multiple comparisons via the minP (permutation) method is also available (Westfall & Young, 1993). Currently, the toolbox works only with undirected matrices.

The toolbox also provides a preprocessing path for resting state and task fMRI data. Included are several options for partialing nuisance signals (particularly relevant for resting analyses), including total or local (Jo et al., 2013) white matter signal, the first 5 principal components of white matter/ventricular signal (instead of the means; Muschelli et al., 2014), calculation of Saad et al. (2013)'s GCOR, and the use of Chen et al. (2012)'s GNI method to determine whether global signal partialing is needed. In addition, Power et al. (2014)'s motion scrubbing method and Patel et al. (2014)'s WaveletDespike procedure are available. Several deconvolution methods are available. These are particularly important for block-design task fMRI (see below), but they may also provide more accurate connectivity estimates for resting data (e.g., Bush et al. 2015).

Several connectivity methods are available, including Pearson and robust correlation, standard and Ledoit-Wolf regularized partial correlation, mutual information, Patel's Tau, Spearman's Rho, and several copulas.

For block-design task fMRI, the toolbox will compute connectivity matrices for each user-specified condition after partitioning the timeseries by condition. In order to compensate for HDR-related delay, timeseries must be deconvolved, allowing division at the actual onset/offset times. In-house testing (with SPM's deconvolution) indicates that this method produces similar results as simply assuming a 2-second delay and dividing the timeseries without deconvolution, indicating that the deconvolution process does not introduce any major distortion.

Beta-series correlation (Rissman et al., 2004) is used to analyze event-related task fMRI via the method outlined in Mumford et al. (2012). Typically, this method uses a

standard (assumed) HRF, and this option is available in the toolbox (SPM's default HRF). Because the standard HRF may not be accurate, an option is included to estimate the HRF individually for each ROI, for each participant, based on the data. Specifically, the toolbox will use FIR decomposition across all events (independent of condition) to create a mean HRF, which is then used for analysis. Given that this HRF is calculated across all events, extra care should be taken when interpreting findings. Specifically, if one event is more common and/or produces a more coherent HRF, the estimation procedure will be biased toward that event and the resultant HRF will better fit that event. *Caution should be used when using beta-series correlation in this toolbox, particularly with HRF estimation. We are currently testing this implementation, but have not done so rigorously. In addition, the extent to which beta-series correlation is appropriate for graph theory analyses remains unclear.*

Related Publications:

Conference abstract on toolbox:
Spielberg, J.M. (2014). Graph theoretic general linear model (GTG): a MATLAB toolbox. *Brain Connectivity, 4,* A1-A158. doi:10.1089/brain.2014.1501.abstracts

Resting state pathway & graph theory analysis:
Spielberg, J.M., McGlinchey, R.E., Milberg, W.P., & Salat, D.H. (2015). Brain network disturbance related to posttraumatic stress & traumatic brain injury in veterans. *Biological Psychiatry, 78*, 210-216. doi:10.1016/j.biopsych.2015.02.013

Block-design task pathway & graph theory analysis:
Spielberg, J.M., Banich, M.T., Miller, G.A., & Heller, W. (2015). Flexible brain network reconfiguration supporting inhibitory control. *Proceedings of the National Academy of Sciences, 112*, 10020-10025. doi:10.1073/pnas.1500048112

Installation:

1. After unzipping the zip file in an appropriate directory, set the Matlab path to include the folder and all subfolders.
2. To use FSL functions in the preprocessing path, you must change the FSL directory path (line 16) in call_fsl_edit.m (located in the dependencies sub-folder).
3. To use wavelet despiking, you must also download the toolbox (www.brainwavelet.org). Be sure to download the appropriate version for your OS and follow their setup instructions (run the setup.m file).
4. To use least trimmed squares regression, you must also download the LIBRA toolbox (http://wis.kuleuven.be/stat/robust/LIBRA). Be sure to download both the main library and, if needed, the appropriate OS supplement.
5. To calculate/test Small World Propensity, you must also download the small_world_propensity.m script (http://commdetect.weebly.com). This script requires the BioInformatics Matlab toolbox.

Dependencies/Requirements:

All dependencies are contained within the 'dependencies' subfolder, with the exception

of BrainWavelet, FSL (both needed for Stage 1 [preprocessing]), LIBRA (needed for Stage 4), and small_world_propensity.m (needed for Stages 3/4). Because FSL can only be used on Linux/Mac (or emulator), Stage 1 processing must be done on one of those platforms (whereas, the other stages may be done on any platform).

Included dependencies (BIG thanks go to their respective creators):

-Brain Connectivity Toolbox (Rubinov & Sporns, 2010; https://sites.google.com/site/bctnet/Home)

-Modified scripts from the Tools for NIFTI and ANALYZE Image toolbox (www.mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image)

-Robust Correlation Toolbox (http://sourceforge.net/projects/robustcorrtool)

-Kernel-Based Estimation of MI (for continuous variables) (http://www.mathworks.com/matlabcentral/fileexchange/30998-kernel-estimate-for--conditional--mutual-information)

-Bush and Cisler's (2013) non-linear deconvolution script (available by request from the authors, KABush@uams.edu)

-Ledoit-Wolf regularized partial correlation script (https://github.com/brian-lau/highdim/blob/master/+utils/cov1para.m)

-Edited version of Wu et al.'s (2013) spontaneous pseudo-events deconvolution script (http://users.ugent.be/~dmarinaz/HRF_deconvolution.html)

-Selected SPM8 scripts (http://www.fil.ion.ucl.ac.uk/spm/)

-Progress Bar (http://www.mathworks.com/matlabcentral/fileexchange/6922-progressbar)

-Edited version of Rest2GNI script (http://www.mathworks.com/matlabcentral/fileexchange/36864-determine-the-necessity-for-global-signal-regression/content/Rest2GNI.m)

-1 script from REST toolbox (http://resting-fmri.sourceforge.net)

-Matlab implementation of Nichols DVARS.sh script (http://www2.warwick.ac.uk/fac/sci/statistics/staff/academic-research/nichols/scripts/fsl/DVARS.sh)

-Edited version of the FSL script call_fsl.m (standard with FSL, http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/)

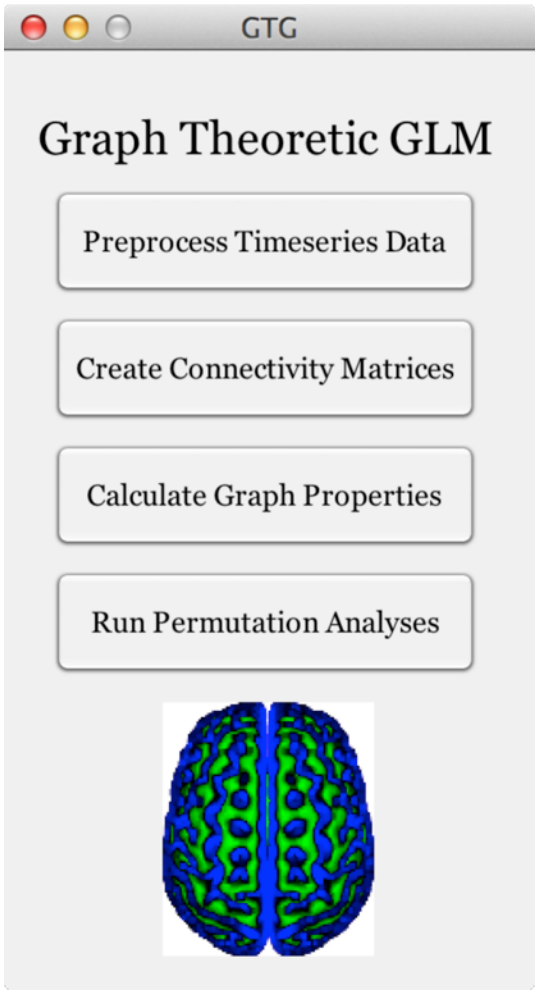
-fdr_bh.m script included for convenience (but not implemented in the toolbox) (http://www.mathworks.com/matlabcentral/fileexchange/27418-benjamini-hochbergyekutieli-procedure-for-controlling-false-discovery-rate/content/fdr_bh.m)

The toolbox will detect the Parallel Computing Toolbox (PCT) and, if present, ask whether to use it. Note that the PCT can sometimes be slow to start, and it may paradoxically be quicker not to use it if your running time is short. Thus, users may want to test running time with and without the PCT.

A minimum of 2GB of RAM is necessary. Given that these analyses are highly parallel, using PCT on a multi-core machine is highly desirable.

Usage

The main GUI can be accessed by typing GTG. The toolbox has four stages, described in more detail below.



GUI vs. Command-Line Mode:

The toolbox is primarily meant to be used in GUI mode. However, using the SAVE button in each GUI allows for the creation of configuration files that can be used at the command line (e.g., for batch processing) or be reopened in the GUI at a later time (using the LOAD button). Note that the output of previously run analyses can also be used to rerun these analyses. The GUI and command line versions are denoted by _GUI or _CMDL appended to the script name. Here is an example usage of the scripts at the command line:

GTG_runpermanalyses_CMDL('test_config.mat')

where test_config.mat is the previously saved configuration file.

Stage 1 – Preprocessing of fMRI Data:

The GUI accepts raw 4D fMRI timeseries data, along with several other inputs, performs preprocessing, and outputs a processed timeseries for each ROI, for each participant. All image inputs should be in .nii or .nii.gz format and LAS orientation.

Inputs:

Cell array of participant identifiers:

-A variable in the matlab workspace containing participant IDs
-Enter the variable name in this field
-IDs must match filenames (e.g., for the filename EPI_001.nii.gz, the ID must be '001' not '1')

Cell array of ROI labels:

-This should be a variable in the matlab workspace containing labels (e.g., 'R OFC') and numeric IDs for ROIs
-Enter the variable name in this field
-The array should contain two columns: the first column containing ROI names, the second the corresponding numeric identifiers (i.e., the numbers that identify each ROI in the 3D ROI image)

Select output filename:

-The basename used for the output file ('_preproc' will be appended)

# of timepoints:

-Currently, this must be consistent across participants

TR:

-Enter the repetition time (TR) in seconds for the fMRI data

Minimum # of voxels in masked ROI:

-To avoid ROIs that are too small to extract a reliable signal, the user can specify the minimum # of voxels that must be present in an ROI (after masking by the brain mask)
-If an ROI has fewer voxels, NaNs are entered in place of a timeseries
-Note that, prior to version 0.39, this value was hardcoded as 5

Minimum % of ROI retained after masking:
-To avoid a situation where large portions of an ROI are removed by masking by the brain mask, the user can specify the minimum % of the ROI that must be retained after masking
-This is done to avoid cases where the extracted signal does not accurately represent the ROI, for example when a large portion of an ROI contains susceptibility artifact, leading to signal dropout and exclusion from the brain mask
-If less than this % is retained, NaNs are entered in place of a timeseries for that ROI, for that participant
-Note that, prior to version 0.39, this value was hardcoded as 50%

Select the 4D functional file for the first participant:
-Press the 'Select' button and navigate to the fMRI timeseries file for the 1st participant contained in the array of participant IDs entered above
-Only the filename of the first participant is needed, because the toolbox locates the rest of the files by replacing the participant ID
-Therefore, filenames must be consistent

Select the 3D brain mask file for the first participant:
-Press the 'Select' button and navigate to the binary brain mask (for the functional data) for the 1st participant contained in the array of IDs entered above
-If the 4D fMRI timeseries is not yet motion corrected, the binary brain mask should correspond to the middle volume in the timeseries
-The purpose of this mask is 2-fold:
    -Constrain which voxels are examined
    -Compute the global signal

Select the 3D ROI mask for the first participant:
-Press the 'Select' button and navigate to the ROI file (containing all desired nodes) for the 1st participant contained in the array of IDs entered above
-Each ROI must have a unique numeric identifier (identifiers need not be consecutive) and must match the #'s entered in the array of ROI labels entered above

Select the 3D white matter mask for the first participant:
-Press the 'Select' button and navigate to 3D binary white matter mask for the 1st participant contained in the array of IDs entered above
-Optional – only needed if the user wishes to partial white matter signal

Select the 3D ventricle mask for the first participant:
-Press the 'Select' button and navigate to 3D binary ventricle mask for the 1st participant contained in the array of IDs entered above
-Optional – only needed if the user wishes to partial ventricular signal

Highpass cutoff:

-Enter the cutoff (in Hz) for the highpass temporal filter

-To only do lowpass filtering, enter a value ≤ 0 in this box

Lowpass cutoff:

-Enter the cutoff (in Hz) for the lowpass temporal filter

-To only do highpass filtering, enter a value ≤ 0 in this box

NOTE: if the ROI, white matter, and ventricular masks have not yet been transformed into functional space, the user must indicate the input space in the pulldown menu. The program will later ask for the correct transform file (to warp to functional space). For standard space to functional, the transform must be a warp file suitable for use with FSL's FNIRT. For structural space to functional, the transform must be a matrix file suitable for use with FSL's FLIRT.

Options (in order of implementation):

Use files if available?:

-In the case that some/all preprocessing was done for some participants, the user enable the toolbox to use these (nifti) files instead of rerunning preprocessing (reducing processing time)

-For the toolbox to detect the files, the filenames must match the names typically given by the toolbox

Slice timing correction:

-Indicate the slice collection order in the pulldown menu

Motion Correction:

-Via FSL's MCFLIRT

Polynomial detrending:

-Indicate the desired polynomial order in the pulldown menu

Wavelet despike:

-Via Patel et al. (2014)'s Brain Wavelet Toolbox

-Performed with the default parameters

-This procedure is likely incompatible with Power et al. (2014)'s motion scrubbing, so only one of these procedures can be used at a time

Temporal filter:

-Three filter options:

-*'Ideal' filter*:

-Based on the filter used in REST v. 1.8

-Achieves complete precision (of cutoff frequency) in the frequency domain at the expense of inducing oscillation in the time domain

-*Butterworth filter*:

-Induces less oscillation in the time domain than the 'Ideal' filter at the expense of a shallower frequency cutoff

-*FSL's nonlinear highpass & Gaussian linear lowpass filter*:

-The non-linearity of the highpass filter may be useful, because non-Gaussianity in lower frequencies may be important in determining causal direction (Mumford & Ramsey, 2014)

Erode WM/ventricle masks:

-Eroded via a sphere

-This is done to reduce the amount of gray matter signal captured in these masks

-Enter the desired sphere radius in fMRI voxel units (e.g., entering 3 in the box will lead to a sphere of radius 3 voxels)

Partialing of nuisance signals:

-Motion correction parameters:

-Original 6 parameters (or, whatever is contained in the .par files)

-t - 1 parameters (a la Friston's autoregressive method)

-Squared parameters

-(2nd order) 1st derivative of parameters

-Global signal:

-(2nd order) 1st derivative of signal

-Given the controversy surrounding partialing of global signal, two further options are available:

-Test necessity of partialling:

-If selected, the toolbox will test the necessity of partialing the global signal on a participant specific basis using Chen et al. (2012)'s GNI method (i.e., global signal is not partialed from data with a GNI > 3).

-A note of caution should be considered when using this method. Specifically, if GNI correlates with variables of interest, it is possible that partialing global signal for only certain participants may lead to false associations (or mask true effects). Therefore, if this method is used, it is highly recommended that the user ascertain whether global partialing (i.e., GNI value) is related to variables of interest.

-Calculate GCOR:

-Calculate a GCOR (Saad et al., 2013) value per participant

-Can be used as a covariate in higher-level analyses to reduce bias (see Saad et al. for further detail)

-White matter signal:

-Default is to extract the signal from the entire white matter mask. However, Jo et al. (2013) suggest that extracting signal from local white matter (i.e., white matter within a 45mm sphere around the current voxel of interest) may help to reduce distance-dependent artifact induced by partialing of the global signal. (NOTE: PCA will not be used to extract local white matter signal)

-(2nd order) 1st derivative of signal

-Ventricular signal:

-(2nd order) 1st derivative of signal

-Use WM & Ventricular Principal Components

-Partials the first five principal components rather than mean signal

-Computed separately for white matter and ventricular signal

-Based on PCA of timeseries of voxels within the white matter/ventricle mask

Motion censoring:

-Scrubbing of motion-related signal via Power et al. (2014)'s method

-This procedure is likely incompatible with Patel et al. (2014)'s despiking, so only one can currently be used at a time

FD cutoff:

-Frame displacement cutoff (in mm) for motion censoring

-FD is the estimated movement from the previous to the current volume

-Typical cutoff values range from .5mm (liberal) to .2mm (conservative)

-Pick cutoff value based on your sample (e.g., if you pick .2mm and this results in throwing out too many participants, you may want to rethink your cutoff or the use of scrubbing in general)

DVARS cutoff:

-Cutoff (in standard deviation units) for motion censoring

-DVARS is roughly the change in mean signal from the previous volume to the current volume

-The selection of threshold has the same considerations as FD

FD/DVARS criteria logic:

-Select OR to be more liberal and censor timepoints that exceed the cutoff for FD OR DVARS

-Select AND to be more conservative and censor timepoints that meet BOTH FD AND DVARS cutoffs

Remove autocorrelation:

-Remove autocorrelation from timeseries, which may be useful for estimating the directionality of effects

-Enter the number of timepoints back for which you want to remove autocorrelation (i.e., the order of the autocorrelation)

-Removing autocorrelation for >1 timepoint will also remove all lesser timepoints (e.g., if you enter 3 timepoints, ar1, ar2, and ar3 will be removed)

Deconvolve timeseries?:

-HRF deconvolution can be performed using several methods, including SPM's method, detection of spontaneous pseudo-events (Wu et al., 2013), or non-linear regression (Bush & Cisler, 2013)

-If selected, deconvolved timeseries for each ROI will be output in addition to the standard non-deconvolved timeseries

-IMPORTANT: Be sure to visually check the timeseries for one or two ROIs for each participant after deconvolution to ensure that they look reasonable

When to Deconvolve?:

-Deconvolution can be performed on a per-voxel basis (before ROI extraction) or after the timeseries have been extracted for each ROI. Deconvolution before ROI

extraction is perhaps more accurate, but far more time consuming. Deconvolution after ROI extraction is the default and recommended option.

ROI timeseries extraction measure:

-Mean across ROI voxels for each timepoint

-Median of values of ROI voxels for each timepoint

-Largest principal component of set of timeseries from voxels in the ROI

- When using PCA (or SVD) extraction, the sign/direction of the extracted component will not always accurately reflect the sign/direction of the underlying signal. In house simulations revealed that PCA extraction accurately reflected the sign/direction of the most prominent underlying signal 90-95% of the time (SVD was accurate only ~50% of the time). However, the mean ROI signal always had accurate sign/direction (even when it didn't accurately capture underlying signal variance). Therefore, for PCA extraction, the toolbox computes the correlation between the largest principal component and the mean signal and, if the correlation is negative, reverses the sign/direction of the PC (i.e., multiply by -1). Note that this is also the procedure used by AFNI for PCA/SVD ROI extraction.

Extract from ROIs:

-To save time, the user can choose to extract the ROI timeseries after slice timing correction/motion correction/detrending and perform all subsequent steps on the mean timeseries

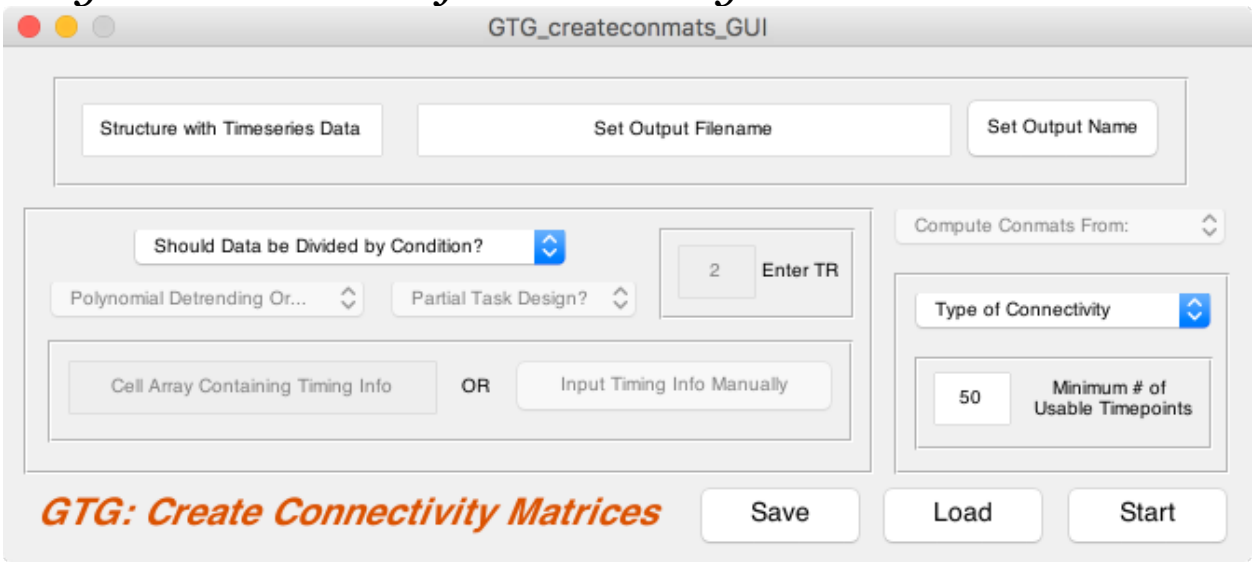
-Incompatible options are disabled: FSL bandpass filter, wavelet despiking, motion scrubbing, calculating GCOR, local white matter partialing

Outputs:

-.mat file containing a structure with preprocessed timeseries for each ROI, along with numerous other variables used in processing

-Logfile

Stage 2 – Creation of Connectivity Matrices:



This GUI accepts the Stage 1 output and creates connectivity matrices (one per participant/repeated condition). Connectivity can be computed on the standard and/or the deconvolved timeseries.

For block-design task fMRI, the toolbox will compute connectivity matrices for each user-specified condition after partitioning the timeseries by condition. In order to compensate for HDR-related delay, timeseries must first be deconvolved, allowing division at the actual onset/offset times. This deconvolution can be performed using several methods in Stage 1. If this has not already been done in Stage 1, the toolbox will do so using SPM's method. In-house testing indicates that this method produces similar results as simply assuming a 2-second delay and dividing the timeseries without deconvolution, indicating that the deconvolution process is not introducing any major distortion.

Beta-series correlation (Rissman et al., 2004) is used to analyze event-related task fMRI via the method outlined in Mumford et al. (2012). Typically, this method uses a standard (assumed) HRF, and this option is available in the toolbox (SPM's default HRF). Because the standard HRF may not be accurate, an option is included to estimate the HRF individually for each ROI, for each participant, based on the data. Specifically, the toolbox will use FIR decomposition across all events (independent of condition) to create a mean HRF, which is then used for analysis. Given that this HRF is calculated across all events, extra care should be taken when interpreting findings. Specifically, if one event is more common or produces a more coherent HRF, the estimation procedure will be biased toward that event and the resultant HRF will better fit that event. *Caution should be used when using beta-series correlation in this toolbox, particularly with HRF estimation. We are currently testing this implementation, but have not performed rigorous testing. In addition, the degree to which beta-series correlation is appropriate for graph theory analyses remains unclear.*

Inputs:

Structure with timeseries data:

-This should be a variable in the Matlab workspace obtained from the output of Stage 1 (load the .mat file into the workspace)
-Enter the name of the variable in this field

Enter TR:
-Optional; only needed for task fMRI

Cell array containing timing info:
-Optional; only needed for task fMRI
-Should be a cell array of size c x 1, where c = the # of task conditions (not blocks/events)
-Each cell in array corresponds to one condition
-Each cell should contain a 2 x b matrix, where b = the # of blocks/events *for that condition*
-1st row of this matrix should contain the onset time of each block/event *in seconds (not TRs*)
-2nd row of this matrix should contain the duration of the corresponding block/event *in seconds (not TRs*)
-This information can also be specified directly by pushing the 'Input Timing Info Manually' button

Minimum # of usable timepoints:
-This is the minimum number of timepoints that you will accept in a timeseries
-Only relevant if motion censoring was used in the previous stage, as that can remove timepoints (variably across participants)
-Enter 2 in the box to use all timeseries

Options:

Should data be divided by condition?:
-Optional; only needed for task fMRI
-For block designs, choose block design
-For event-related designs, choose one of the two options
-To estimate an HRF based on the data for each ROI, choose that option BUT SEE WARNINGS ABOVE

Polynomial Detrending Order:
-Optional; only needed for block design task fMRI
-Detrends within each block
-Use with caution, as this has been observed to occasionally introduce some odd connectivity patterns

Partial Task Design?:
-Optional; only needed for block design task fMRI
-This option can only be performed if deconvolution has not been performed in Stage 1, because it must occur before deconvolution (if you did do deconvolution in Stage 1, but want to use this option, just delete the deconv_ts field from the out structure before entering into the Stage 2 GUI)
-Selection of this option will create predictors for each condition and partial these from each ROI timeseries (before deconvolution)

-This *should* have little to no effect on connectivity estimates, as partialling the task design removes the mean effect of condition, which is essentially also removed by dividing the timeseries by block. So why include this option? Because reviewers may ask for it…..

Compute Conmats From:

-If deconvolution was performed in Stage 1, connectivity matrices can be computed from the original data, the deconvolved data, or both, which is specified here

-If both are specified, two files will be saved

-Separation of conditions in block designs requires deconvolution, and so the toolbox will detect whether deconvolution was performed in Stage 1. Thus, regardless of what is specified here, the toolbox will use the deconvolved timeseries if present and perform deconvolution using SPM's method if not

-Separation of conditions in event-related designs requires timeseries that have NOT been deconvolved. Thus, regardless of what is specified here, the toolbox will use the original timeseries only

Type of Connectivity:

-Ten measures of association can be used:

-Pearson correlation

-Partial correlation

-Thought to reflect *direct* effects to a greater degree

-Much less useful in datasets with many nodes, given the large percentage of variance removed from each timeseries

-Regularized partial correlation

-Ledoit-Wolf regularized partial correlation

-Allows for situations where the number of ROIs is greater than the number of timepoints (rank deficiency)

-Mutual information

-Via the script from the Functional Connectivity Toolbox

-Robust (bendcorr) Correlation

-Via the script available in Corr_toolbox_v2

-Kendall's Tau

-Correlation based on rank-order

-Spearman's Rho

-Correlation based on rank-order

-Copulas

-Alternative to correlation based on the dependence structure between variables. Copula fitting is based on rank dependence, so these methods are similar to Kendall's Tau and Spearman's Rho (and several useful copulas can be calculated directly from Kendall's Tau). Copulas are useful when the multivariate distribution is non-elliptical, because they can be more sensitive to important variance in the distribution tails. Copulas are popular in finance, but may also be useful for neuroscience; however, the copulas currently implemented are not necessarily best for neuroscience, and future work is likely to introduce copulas that better characterize imaging

distributions. ***Thus, the currently implemented copulas should be used with care.***

-The Gaussian version assumes that the multivariate distribution is normal. So, this should give the same results as Pearson correlation most of the time (and probably won't be very useful).

-The t version assumes that the multivariate distribution is a t distribution (i.e., thicker tails).

-The Frank copula is in the Archimedean (one parameter) family and assumes that the multivariate distribution is symmetrical with very fat tails.

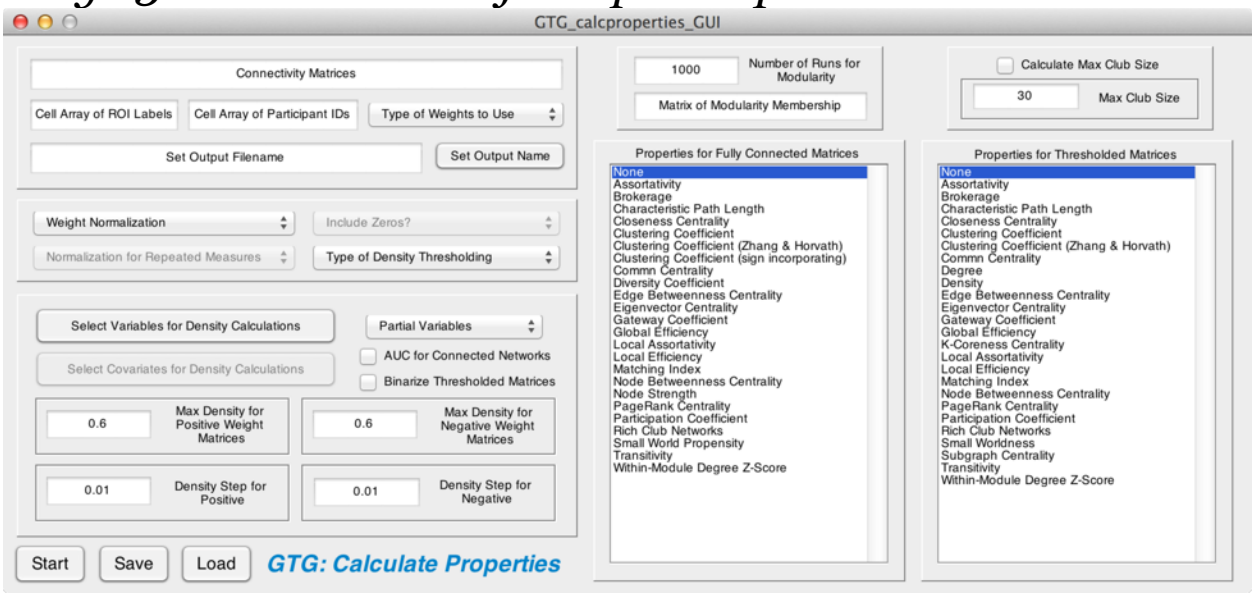
- The Clayton copula can be generated based on Kendall's Tau with this formula: $\alpha = \frac{2\tau}{(1-\tau)}$. The Gumbel copula can be generated based on Kendall's Tau with this formula: $\alpha = \frac{1}{(1-\tau)}$. The Clayton copula assumes that the multivariate distribution has greater dependence in the negative tail than in the positive, whereas the Gumbel copula assumes the opposite.

Outputs:

-Output structure containing connectivity matrices for each participant.

-Logfile

Stage 3 – Calculation of Graph Properties:



This stage takes connectivity matrices as input (can, but does not have to be, the Stage 2 output) and calculates graph theoretic properties for each participant/repeated level (to be used as dependent variables in Stage 4). The majority of properties are calculated via scripts from the Brain Connectivity Toolbox (Rubinov & Sporns, 2010).

This stage requires that entries in the connectivity matrices represent connectivity *strength*, but the measure of connectivity does not matter. In other words, entries could be Pearson correlations (e.g., output from Stage 2) or white matter tract strength (e.g., obtained from diffusion tractography). Therefore, entries do not have to conform to a specific scale.

This stage computes properties for fully-connected and/or (binarized or non-binarized) thresholded networks. For thresholded networks, the toolbox computes properties across a set of density thresholds. The user specifies a desired maximum density (a value of 0.5-0.6 is common) and the desired density step, and the toolbox computes the minimum density. When matrices are 'sparser', it is possible that the specified maximum density cannot be reached. In this case, the toolbox will use the maximum possible density. Therefore, be sure to check the actual maximum density reached (i.e., out.max_dens_pos, out.max_dens_neg). This may occur, for example, for negative weights in resting data.

The minimum density is chosen such that, at the very least, the presence of disconnected networks is not highly correlated with variables of interest (e.g., IVs in Stage 4). Therefore, this computation takes into account variables specified by the user and creates groups of participants by stratifying (e.g., high, medium, low) these variables. Mean networks are created for each (stratification) group, and the minimum density at which that network remains connected is identified. This is done for each group (across each variable, across all selected variables) and for the overall mean network. Then, the maximum of these minima is chosen as the overall minimum density.

After each property is computed for each threshold, a standardized area under the curve (AUC) is computed for each property, creating one value per property, per participant (or one value for each node/edge, for node/edge specific measures).

For both fully connected and thresholded networks, properties will be calculated for positive and negative weights separately (only positive, if absolute value is used). However, it may be the case that an appropriate minimum density cannot be found for negative weights in thresholded matrices, in which case these properties will not be computed.

IMPORTANT: Property values for negative weights are calculated by reversing the sign of the matrices and then treating them the same as the original matrices. Therefore, you should flip the signs for betas and t-vals from statistics calculated with these values when interpreting these findings.

Inputs:

Connectivity matrices:

-This should be a variable in the matlab workspace
-Enter the variable name in this field
-1 matrix per repeated level, per participant
-For p ROIs, n participants, and r repeated levels, this matrix should be p x p x n x r
-NOTE: these matrices can have been created in any program and do not have to correspond to fMRI (e.g., can be diffusion fiber strength)
-If repeated-measures tests are desired in Stage 4, the repeated measure should be indexed in the fourth dimension. For example, for 3 repeated levels, 10 participants, and 30 ROIs, the dimensions of the input matrix should be 30 x 30 x 10 x 3. In this stage (3), each level of the repeated factor will be treated independently. Note, Stage 4 uses polynomial contrasts, so arrange the levels of the repeated measure accordingly.
-If you used this toolbox to create these matrices, you can load the .mat output from Stage 2 and enter 'out.conmats' (without quotation marks) in this field

Cell array of ROI labels:

-This should be a variable in the matlab workspace
-Enter the variable name in this field
-If you used this toolbox to create the connectivity matrices, you can load the .mat output from Stage 2 into the workspace and enter 'out.ROI_labels' (without quotation marks) in this field

Cell array of participant IDs (optional):

-This should be a variable in the matlab workspace
-Enter the variable name in this field
-If you used this toolbox to create the connectivity matrices, you can load the .mat output from Stage 2 and enter 'out.subs' (without quotation marks) in this field

Select variables for density calculations:

-Optional: only needed if calculating properties for thresholded matrices
-These should be variables in the matlab workspace

Select covariates for density calculations:

-Optional: only needed if calculating properties for thresholded matrices

-These should be variables in the matlab workspace

-These variables will be partialled from each of the variables selected above to be used in density calculations

Max density for positive/negative weight matrices:

-Optional; only needed if calculating properties for thresholded matrices

Density step for positive/negative:

-Optional; only needed if calculating properties for thresholded matrices

-Value for step between different densities

-Smaller value leads to potentially better precision but longer processing time

Number of Runs for Modularity:

-Optional; only needed for properties requiring a modular structure and when modularity should be calculated based on the data

-Modularity computation uses the Louvain algorithm and then the Fine-tuning algorithm, from the Brain Connectivity Toolbox

-Because modularity calculation is not deterministic (i.e., it depends on the initial start values), modularization is repeated and the organization that maximizes modularity is chosen

-This value specifies the number of repeated runs (this process is fairly quick, so a large value [1,000] is recommended)

Matrix of Modularity Membership:

Optional; only needed for properties requiring a modular structure and when modularity should *NOT* be calculated based on the data

-By default modularity is computed based on the overall mean network (across participants/repeated levels), but a matrix containing a predetermined modularization can be entered here instead

-This should be a variable in the matlab workspace

-Enter the name of the variable in this field

-This should be a vector of size p x 1 (p = # of nodes), and each entry corresponds to the module for that node (node order must be identical to that in the ROI Labels variable)

-ALL NODES MUST BE ASSIGNED A MODULE

Max Club Size:

-Optional: only needed for calculating Rich Club networks

-Enter the number corresponding to the maximum club size to calculate

Options:

Type of Weights to Use:

-By default, properties will be calculated for positive weights only

-The user can specify to calculate properties for both positive and negative weights or the absolute value of weights (if only the strength of the relationship is of interest).

Weight Normalization:

-Many properties depend on the average weight in the network, which may confound analyses if not accounted for. Thus, you can normalize weights across the network, individually for each participant by either dividing by the mean, median, or max (likely less robust).

-This is done separately for positive and negative weights.

-Generally, special care should be taken with normalization for properties that are more directly related to weight value.

-Normalization can reduce individual differences in properties, in some cases completely removing difference. Specifically, when normalizing by the mean, total Node Strength will identical (or close) when including 0's. When not including 0's, all variance in total Node Strength will be based on the # of 0's (making it more similar to Density).

-Normalization can also induce false individual differences. For example, when mean normalizing including 0's, there will be individual differences Node Strength for a particular node. However, part of this variance will be due to the strength of other nodes in the network. For example, if two individuals have identical Node Strength for all nodes but amygdala, the network for the participant with higher amygdala strength will be normalized (divided) by a larger value, making the Node Strength of all the other nodes smaller. Thus, normalization will reduce true variance in amygdala Node Strength and induce false variance for all the other nodes.

-Thus, to test differences in Node Strength, for example, but want to use normalization for most properties, you should compute Node Strength separately with no normalization, then copy these values into the 'out' output structure.

-No matter what option is chosen (including no normalization), the data is divided by the absolute value max of the entire dataset, in order to ensure that values will be between 0-1 while retaining the relative distribution of data.

-For negative weights, the absolute value of the mean/median/max is used to normalize, in order to retain the appropriate sign.

Use Zeros?:

-When normalizing weights by the mean or median, you can select to either include or not include zeros in calculating the mean/median. Including zeros will make the overall matrices more equal, whereas not doing so will make the non-zero weights more equal. It is unknown at the current time which option will reduce bias most effectively, and likely this answer will depend on the specific graph property. Therefore, it might be prudent to calculate properties both ways and determine if findings hold up across the two.

-When normalizing weights by the median, the inclusion of zeros may cause result in NaNs for the normalized matrices (i.e., when zero is the median of a network, dividing by zero results in NaN). This may be more likely to occur when examining negative weights, as weight distributions are often positively skewed (e.g., in resting fMRI), making it more likely for the median to be zero.

Normalization for Repeated Measures:

-When normalizing repeated measures data, you have the option to either (i) normalize across repeated levels (preserving within-participant differences) or

(ii) normalize each level separately (equating each level). It is unknown at this time which option is best.

Type of Density Thresholding:

-When thresholding matrices, the weight threshold can either be individualized (meaning that density of each participant's matrix WILL be equal) or equal across participants (meaning that the density of each participant's matrix WILL NOT be equal). If you would like to test differences in Density, but want to use normalization for most properties, you should compute Density separately with no normalization, then copy these values into the 'out' output structure.

Partial Variables:

-Optional; only needed if calculating properties for thresholded matrices

-The stratified groups can be created using either the original variables or variables that have had the variance associated with the other specified variables (and covariates) partialled out

-The reason to use partialled variables is because this is the variance that will actually be tested in the GLMs in Stage 4

AUC for Connected Networks:

-Optional; only needed if calculating properties for thresholded matrices

-Because the procedure described above may still leave some disconnected matrices, the user has the option of also computing the AUC on ONLY connected matrices

Binarize Thresholded Matrices:

-Optional; only needed if calculating properties for thresholded matrices

-The user can select this option to use fully binarized matrices (by default, weight values above the threshold are retained for calculation of most properties)

-Degree, Density, K-Coreness Centrality, Subgraph Centrality, and Small-Worldness all require binarized matrices. Thus, binarized matrices will be calculated and used for these properties whether or not this option is selected.

Calculate Max Club Size:

-Max club size for rich club networks can vary across matrices. Thus, this value can either be computed based on the data or pre-specified.

Properties for Fully Connected Matrices:

-Specify which properties to compute

-Click on each property of interest, holding down cntrl (or whatever works on your system) to specify multiple properties

-All properties that you would like to test should end up highlighted

-See the Appendix A for details regarding these properties

Properties for Thresholded Matrices:

-Specify which properties to compute

-Click on each property of interest, holding down cntrl (or whatever works on your system) to specify multiple properties

-All properties that you would like to test should end up highlighted

-See the Appendix A for details regarding these properties

Output:

Output structure containing graph theoretic properties for each participant

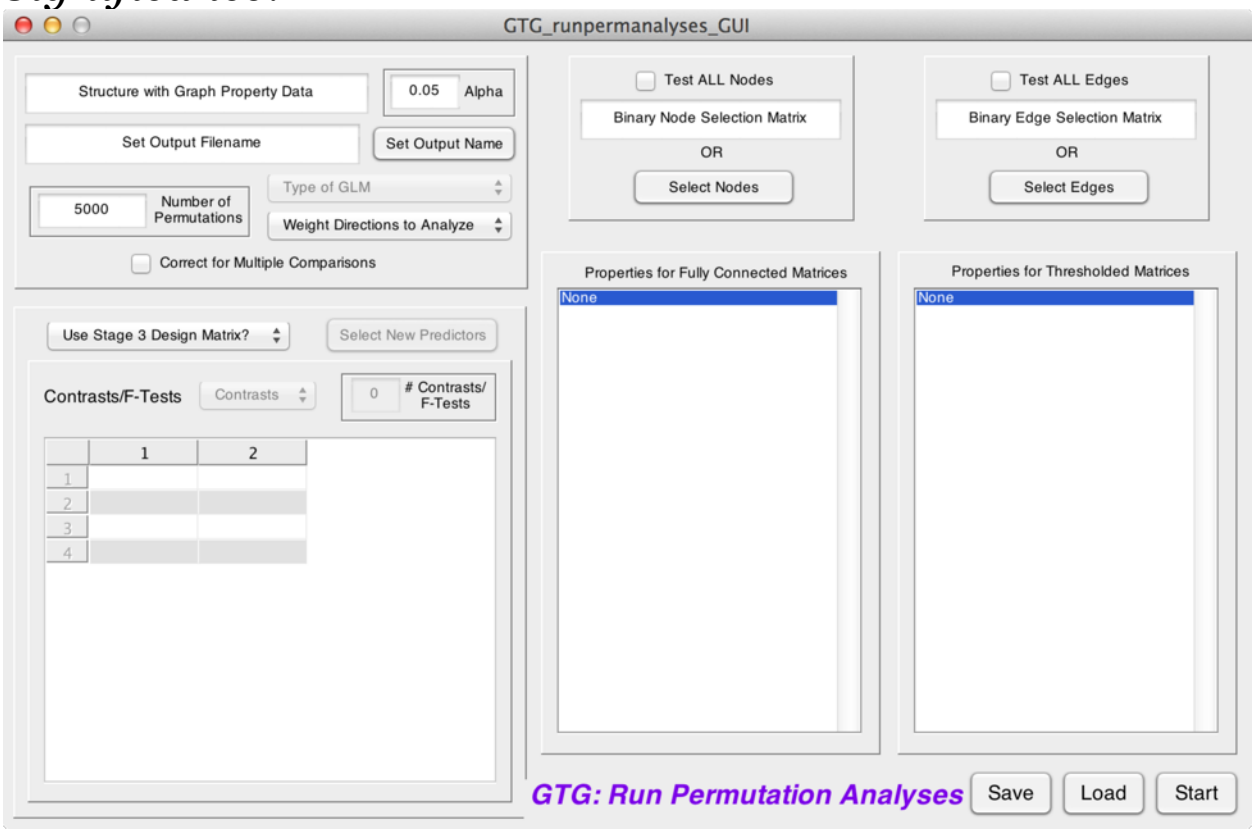
-Properties for fully-connected networks are contained in out.fullmat_graph_meas, and for thresholded networks in out.AUC_thrmat_graph_meas.

-For thresholded matrices, for each property field there is a corresponding field with '_numvalsAUC' appended that indicates the number of values used in computing that particular AUC, allowing the user to determine whether this varies with variables of interest (which might introduce bias)

-If the user specified that AUC should also be computed for only connected networks, the fields corresponding to these values will have '_nodiscon' appended

-The output structure contains other useful values including the modularity structure (mod_grps) and other values used in processing (serving as a logfile)

Stage 4 – Running GLM with Permutation-Based Significance:



This stage calculates GLMs with the graph properties computed in Stage 3 as DVs. The user selects IVs and contrasts/F-tests of these IVs ('contrasts' is used loosely here to include something like [0 0 1], which would test the significance of the third IV; i.e., contrasts do not have to sum to 0). Continuous & categorical between-participant predictors and a categorical within-participant predictor are accepted.

Testing a single predictor or a contrast between predictors:
First, define the design matrix and leave the 'Contrasts' drop-down menu on 'Contrasts'. Next, specify the desired # of contrasts in the box (then hit return or select somewhere else in the GUI). This will create a matrix in the bottom left with a column for each predictor and a row for each contrast. Enter the desired weights for each contrast in the appropriate rows.

Testing a between-participant factor:
If the factor has only two levels, it should be treated as a single predictor (see above). For more than two levels, first define the design matrix. The factor must be specified in the design matrix by q-1 dummy coded variables (q=#of levels), which can be created in MATLAB using dummyvar (Note: this will create q predictors, so only use the first q-1). Next, select 'F-tests' from the 'Contrasts' drop-down menu. Next, specify the desired # of F-tests in the box, and enter a 1 in the column associated with each of the dummy coded variables.

Testing a within-participant factor:
Currently, only a single repeated factor can be specified, with a maximum of eight levels. Additionally, only OLS can be used with repeated-measures. These limitations will be addressed in future releases. Note, the repeated factor must have been specified in Stage 3 (indexed by the 4th dimension, see above). In Stage 4, the toolbox will recognize (based on the Stage 3 output) that there is a repeated measure (and the number of levels). Therefore, nothing different must be done in Stage 4 to test a repeated measure (i.e., specify the desired contrasts/F-tests as described above). Polynomial contrasts are used, and the output will contain (in order):

1) F-test for the between effect (i.e., averaging across levels)
2) F-test for each polynomial contrast (in ascending order, e.g., linear, quadratic, cubic)
3) Omnibus test across all within levels. The test statistic for the omnibus test will either be Wilks' Lambda, if a contrast is specified for between-participant predictors, or F, if an F-test is specified for between-participant predictors.

Significance is determined via non-parametric permutation tests using the method of Freedman & Lane (1983) (e.g., the same method used in FSL's Randomise). Correction for multiple comparisons across all tested properties (except Rich Club) is available and is computed via permutation testing (which accounts for the correlation between DVs).

Because several of the measures are node or edge specific, computation time is greatly increased if 'Test ALL Nodes' is specified. Testing all nodes/edges is also problematic in terms of multiple comparisons (i.e., you don't have to look at all the tests, but it sure is tempting…). Therefore, we strongly recommend examining only specific nodes/edges at this point. Nodes/edges can be selected based on a priori hypotheses. However, we also highly recommend using Zalesky, Fornito, & Bullmore (2012)'s NBS toolbox to identify specific nodes/edges that vary with IVs of interest.

IMPORTANT: Property values for negative weights are calculated by reversing the sign of the matrices and then treating them the same as the original matrices. Therefore, you should flip the signs for betas and t-vals for tests of negative weights when interpreting these findings.

Inputs:

Structure with graph property data

-This should be a variable in the matlab workspace
-Enter the variable name in this field
-This variable is obtained from the output of Stage 3 (load the .mat file into the workspace)

Alpha:

-Enter the alpha to use for significance

Number of Permutations:

-Enter the number of permutations used to build the distribution

Select new predictors:

-Only needed if all IVs of interest were not specified in Stage 3
-These should be variables in the matlab workspace

Binary Node Selection Matrix:
-Only required if you wish to test a subset of nodes for node-specific properties
-Should be a binary p x 1 matrix with a 1 entered for nodes you wish to test, a 0 otherwise
-Instead of using this matrix to select nodes, you can do so by hand by pushing the 'Select Nodes' button

Binary Edge Selection Matrix:
-Only required if you wish to test a subset of edges for edge-specific properties
-Should be a binary p x p matrix with a 1 entered for edges you wish to test, a 0 otherwise
-Instead of using this matrix to select edges, you can do so by hand by pushing the 'Select Edges' button

Options:

Type of GLM:
-OLS = Ordinary Least Squares
-Robust = Robust regression, which down-weights outliers in the dependent variable
-LTS = Least Trimmed Squares, which effectively down-weights multivariate outliers

Weight Directions to Analyze:
-By default, analyses will only be run for properties computed using positive weights (this includes the absolute value of weights)
-The user can specify to also analyze properties computed using negative weights

Correct for Multiple Comparisons:
-If selected, a second set of p-values will be computed for all effects that meet the specified α at an individual level. This second set is corrected for all comparisons specified (all properties selected, all contrasts/F-tests selected), with the exception of Rich Club. The procedure for correcting for multiple comparisons is the minP (permutation) method (Westfall & Young, 1993).
-The minP method can be relatively conservative, because it controls only for Type I error (i.e., FWE). A different option is to use FDR, which controls for both Types I and II error. Although not implemented in the toolbox, a third-party script has been included to do FDR correction (fdr_bh.m).

Use Previous Design Matrix:
-Select 'Yes' if you already entered the variables you wish to model in Stage 3
-Enter 'No' if you wish to use a different design matrix (then push the 'Select New Predictors' button to do so)

Contrasts/F-Tests:
-Select either 'Contrasts' or 'F-Tests'

# of Contrasts/F-Tests:

-Enter the number of contrasts/F-tests you wish to use

Contrast/F-Test Matrix:

-For each row (representing one contrast/F-test), enter a contrast weight in each column (entry can remain 0)

Test ALL Nodes:

-Only applies to testing of node-specific properties

-Check this box to test all nodes, otherwise enter a selection matrix or push the 'Select Nodes' button

Test ALL Edges:

-Only applies to testing of edge-specific properties

-Check this box to test all edges, otherwise enter a selection matrix or push the 'Select Edges' button

Properties for Fully Connected Matrices:

-Specify which properties to test

-Click on each property of interest, holding down cntrl (or whatever works on your system) to specify multiple properties

-All properties that you would like to test should end up highlighted

-See the Appendix A for details regarding these properties

Properties for Thresholded Matrices:

-Specify which properties to test

-Click on each property of interest, holding down cntrl (or whatever works on your system) to specify multiple properties

-All properties that you would like to test should end up highlighted

-See the Appendix A for details regarding these properties

Outputs:

Structure containing test statistics (t/F/Wilks' Lambda values, p-values) for each contrast/F-test, for each property selected

Significant effects (at the chosen alpha) are summarized in an output file (<outname>_sig_analyses.txt) and out_data.sig_find

Appendix A

NOTE: The information in this appendix has been culled from a number of sources, most notably:

Rubinov, M., Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. NeuroImage, 52, 1059-1069.

Rubinov M., Sporns O. (2011) Weight-conserving characterization of complex functional brain networks. NeuroImage, 56, 2068-2079.

Therefore, most of the credit should go to these authors (but, all mistakes are mine).

Types of Measures:
Functional Segregation:
The ability for specialized processing to occur within densely interconnected groups of brain regions.
Measures: Clustering Coefficient, Local Efficiency, Transitivity.

Functional Integration:
The ability to rapidly combine specialized information from distributed brain regions.
Measures: Characteristic Path Length, Global Efficiency.

Centrality (Influence):
The importance of a node (edge) for acting as hubs and facilitating integration.
Measures: Brokerage, Closeness Centrality, Commn Centrality, Degree, (Density for Influence), Diversity Coefficient, Edge Betweenness Centrality, Eigenvector Centrality, Gateway Coefficient, K-Coreness Centrality, Node Betweenness Centrality, Node Strength, Pagerank Centrality, Participation Coefficient, Subgraph Centrality, Within-Module Degree Z-Score.

Resilience:
Network vulnerability to insult.
Measures: Assortativity, Local Assortativity.

Other:
Matching Index, Rich Club Networks, Small Worldness, Small World Propensity

Measures:
Assortativity[‡]:
The correlation between degrees of all nodes on two opposite ends of a link. A measure of *resilience*. Positive values reflect greater resilience.
One value is produced for the entire network.

Brokerage:
A node-strength-weighted version of betweeness centrality, which reflects the extent to

which a node controls shortcut 'bridges.' A measure of *centrality*. Positive values reflect greater centrality.
One value is produced per node. (Not part of the BCT)

Characteristic Path Length (CPL):
The average shortest path between all pairs of nodes. More influenced by shorter paths. A measure of *functional integration*. Higher values reflect less integration.
One value is produced for the entire network.

Closeness Centrality:
The mean shortest path between the node and all other nodes. A measure of *centrality*. Higher values reflect greater closeness.
One output per node. (Not part of the BCT)

Clustering Coefficient:
The fraction of triangles around a node (the mean across the network is also computed). A measure of *functional segregation*. Higher values reflect more clustered connectivity. Onneala et al.'s formula.
Two outputs, one has one value per node, one describes entire network.

Clustering Coefficient (Zhang & Horvath):
The fraction of triangles around a node (the mean across the network is also computed). A measure of *functional segregation*. Higher values reflect more clustered connectivity. Zhang and Horvath's formula, which normalizes (i.e., the denominator) based on Node Strength rather than Degree.
Two outputs, one has one value per node, one describes entire network.

Clustering Coefficient (sign incorporating):
The extent to which the node provides a unique influence (the mean across the network is also computed). In other words, if all weights in the triangle are positive or two are negative and one positive, the node is not providing much unique influence (because $1*1=1$, $-1*-1=1$). However, if all three weights are negative or 2 positive and 1 negative, the node is having a unique influence (because $-1*-1\neq-1$, $1*1\neq-1$). Because this formula takes the sign of the weights into account, properties are NOT computed separately for positive and negative weights. A measure of *functional segregation*. Negative values indicate greater unique influence.
Two outputs, one has one value per node, one describes entire network.

Commn Centrality:
A weighted combination of in-strength and out-strength (in/out referring to module membership), which reflects the extent to which a node is an important bridge between modules. A measure of *centrality*.
One value output per node. (Not part of the BCT)

Degree:
The number of neighbors (edges) of a node. A measure of *centrality*. Reflects the general importance of a node. Higher values reflect greater importance.

Computed only for thresholded networks.
One value is produced for each node.

Density:
Fraction of present connections to possible connections. Similar to the mean degree of all nodes in the network. A measure of *influence* (*centrality* on a network scale). Reflects the total "wiring cost" of network. Higher values reflect more interconnected networks.
Computed only for thresholded networks.
One value is produced for the entire network.

Diversity Coefficient:
Measures the diversity of intermodular connections (the variance of the weights of edges connected to a node). A measure of *centrality*. Higher values reflect greater diversity.
Computed only for fully connected networks.
One value is produced per node.

Edge Betweenness Centrality[†]:
The fraction of all the shortest paths in a network that pass through a given edge. A measure of *centrality*. Higher values suggest that an edge is more important for controlling information flow.
One value is produced for each edge.

Eigenvector Centrality:
The corresponding element of the eigenvector with the largest eigenvalue. A measure of *centrality*. Higher values suggest that a node is connected to other nodes with high eigenvector centrality. Is more reflective of the global (vs. local) prominence of a node.
One value is produced for each node.

Gateway Coefficient:
A variant of the Participation Coefficient. Similar to Participation Coefficient, Gateway Coefficient measures the diversity of intermodular connections of individual nodes, but weights this value by how critical these connections are to both intra and intermodular connectivity (e.g., if a node is the only connection between it's module and another module, it will have a higher gateway coefficient). A measure of *centrality*. Higher values reflect more importance of this node for connectivity.
One value is produced for each node.

Global Efficiency:
The average inverse shortest path between all pairs of nodes. More influenced by longer paths. A measure of *functional integration*. Higher values reflect greater integration.
One value is produced for the entire network.

K-Coreness Centrality:
A k-core if the largest subgraph comprising nodes of at least k degree, and the k-coreness of a node is k if the node belongs to the k-core but not the (k+1)-core. A measure of *centrality*. Higher values reflect greater connectivity to more connected nodes/membership in a relatively more influential network.

Computed only for thresholded networks.
One value is produced for each node

Local Assortativity:
The extent to which a node is strongly connected to nodes of similar vs. higher/lower Node/Strength/Degree. Higher values reflect more similar connectivity.
One output per node.

Local Efficiency:
The average inverse shortest path for neighbors of a node (the mean across the network is also computed). Reflects the efficiency of communication among neighbors when a node is removed. A measure of *functional segregation*. Higher values reflect greater communication.
Two outputs, one has one value per node, one describes entire network.

Matching Index[†]:
A measure of the similarity between two nodes' connectivity profiles. Higher values reflect greater similarity.
One value is produced for each pair of nodes.

Node Betweenness Centrality:
The fraction of all the shortest paths in a network that pass through a node. A measure of *centrality*. Higher values suggest that a node is more important for controlling information flow.
One value is produced for each node.

Node Strength:
The sum of the weights connected to a node (the sum across the network is also computed). A measure of *centrality*. Higher and lower values reflect greater centrality of a node for positive and negative edges, respectively.
Computed only for fully connected networks.
Two outputs, one has one value per node, one describes entire network.

Pagerank Centrality:
Similar to eigenvector centrality. A measure of *centrality*. Higher values suggest greater centrality. Is more reflective of the global (vs. local) prominence of a node.
One value is produced for each node.

Participation Coefficient:
The extent to which a node is connected with nodes in different modules (i.e., measures the diversity of intermodular connectivity). A measure of *centrality*. Higher values reflect more between module connectivity. Nodes with a high within module degree z-score but a low participation coefficient are known as provincial hubs and play an important part in facilitating modular segregation. Nodes with both a high within module degree z-score and a high participation coefficient are known as connector hubs and facilitate intermodular communication.
One value is produced for each node.

Rich Club Networks[‡]:
Reflects the degree to which network hubs tend to be more densely connected among themselves than nodes of a lower degree.
One value is produced for each degree size (up to the maximum degree).

Small Worldness:
The ratio of clustering coefficient to path length (each normalized by random networks). Higher values reflect more small worldness, and networks high in small worldness have both high integration and segregation.
Computed only for thresholded networks.
One value is produced for the entire network. (Not part of the BCT)

Small World Propensity:
Version of small worldness for weighted networks.
Higher values reflect more small worldness, and networks high in small worldness have both high integration and segregation.
Computed only for fully connected networks.
One value is produced for the entire network. (Script is from the Network Community Toolbox by Bassett et al.)

Subgraph Centrality:
A weighted sum of the closed walks of different lengths in the network starting and ending at the node. Reflects the extent to which a node participates in subgraphs. A measure of *centrality*. Higher values reflect greater centrality.
Computed only for thresholded networks.
One value is produced for each node.

Transitivity:
The (normalized) mean clustering coefficient. A measure of *functional segregation*. Higher values reflect more clustered connectivity.
One value is produced for the entire network.

Within-Module Degree Z-Score:
The extent to which a node is connected to other nodes in its module. A measure of *centrality*. Higher values reflect more within module connectivity. Nodes with a high within module degree z-score but a low participation coefficient are known as provincial hubs and play an important part in facilitating modular segregation. Nodes with both a high within module degree z-score and a low participation coefficient are known as connector hubs and facilitate intermodular communication.
One value is produced for each node.

[‡]Calculating this measure depends on the presence of (at least some) zeros; thus, it cannot be calculated when there are non-zero values for all entries of the connectivity matrix; therefore, calculation of this property is automatically turned off when using the absolute value of weights for fully connected matrices

[†]Because a value is produced for each edge, the number of values produced can be quite large and computing/testing can be time consuming.