SMAUG Toolbox Version 1.1.0
Copyright 2021 LabMonti™
Written by Nathan Bamberger

SMAUG Toolbox: Detailed Function Guide

General Comments

This guide contains an organized list and a *brief* description of each of the main functions available in the SMAUG toolbox. For more information on a particular function, as well as a full list and description of its inputs and outputs, see the information at the top of the function's .m file. Only functions intended to be called directly by users are included in this guide; for information on sub-functions see the comments in the files themselves.

SMAUG functions are designed to balance flexibility with ease of use. To this end, most SMAUG functions can accept numerous input parameters to modify exactly how they behave; however, usually only the first one or two of these input parameters are required. Later parameters will be set to default values if left unspecified by the user.

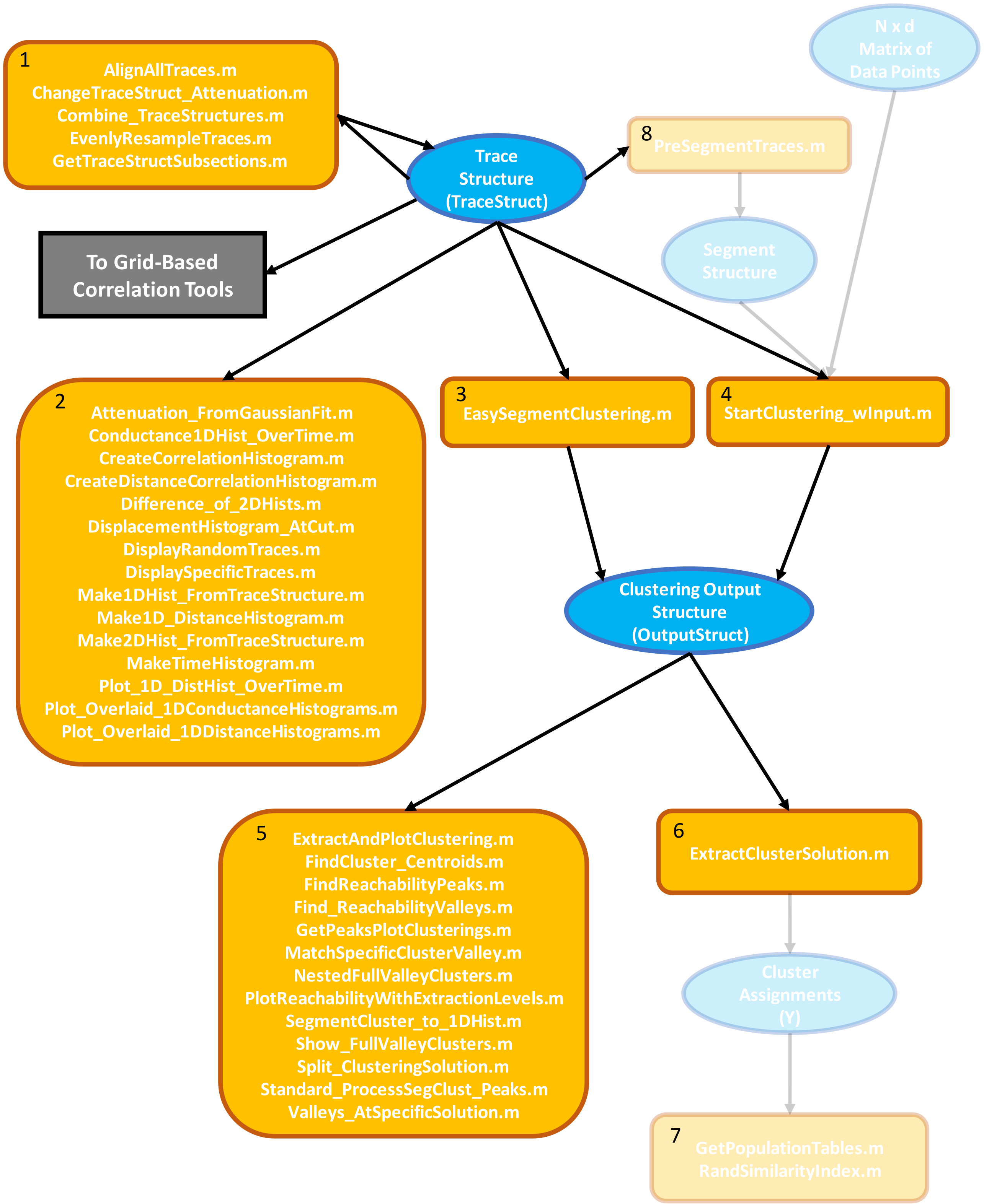
Function Organization

Schemes 1 and 2, below, show a simplified view of how the main SMAUG functions are organized from a data-flow perspective. **Blue ovals** represent specific data structures, **orange boxes** list specific SMAUG functions, and the arrows indicate the primary data type that each function takes as input and produces as output. The sections of Scheme 1 that are likely of minimal importance to most users are **grayed out** to focus attention on the main data-flow pathways.

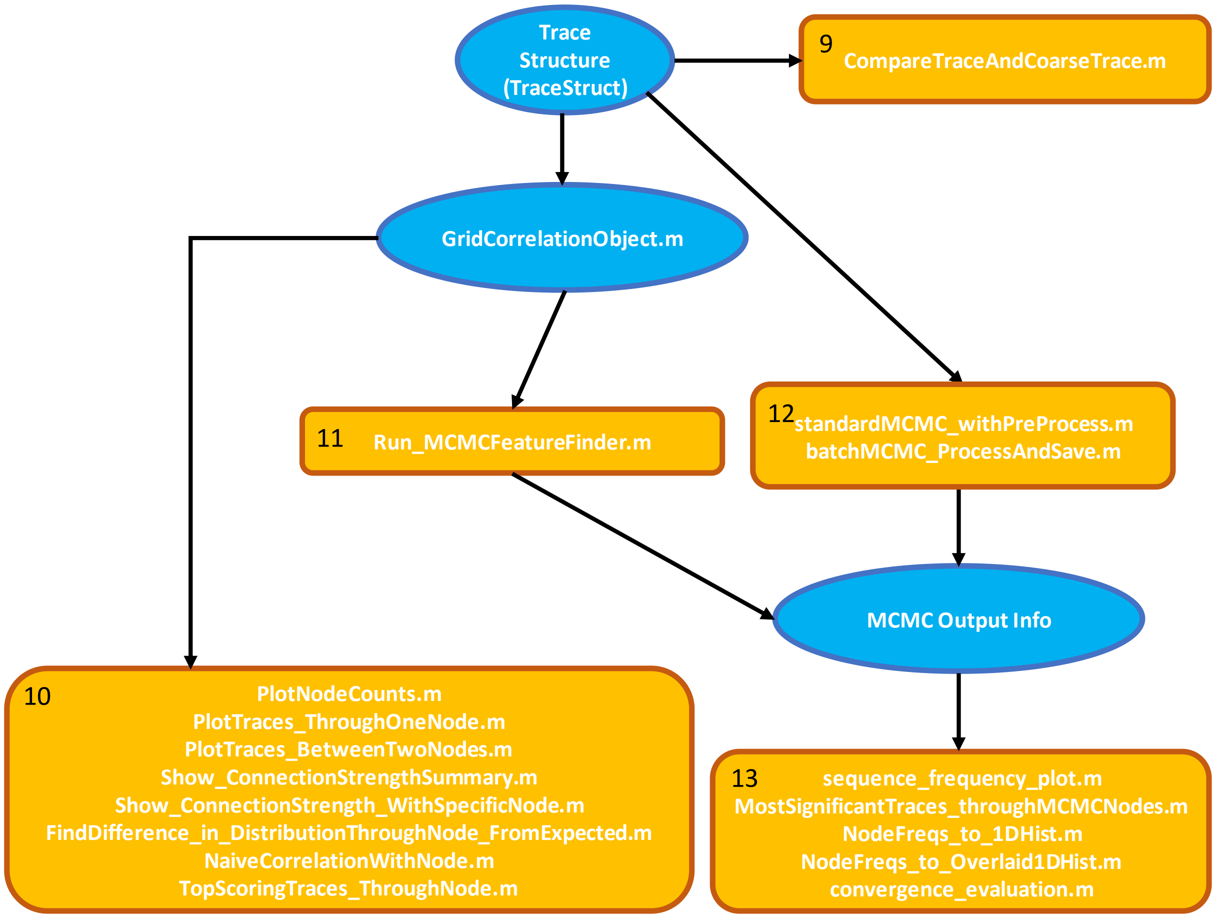
Scheme 1 contains the general analysis capabilities of SMAUG as well as the Segment Clustering algorithm described in Bamberger et al. 2020 (https://doi.org/10.1021/acs.jpcc.0c03612) and its associated analysis functions. A brief summary is as follows: The main data type used by the SMAUG toolbox is a Trace Structure (see How_To_Format_Input_Data.mlx for additional details). Trace Structures can be modified, combined, or split up using the functions in box #1. A variety of graphical analysis methods can be applied to Trace Structures using the functions in box #2 (see Quick_Introduction_To_Other_Analysis_Tools.mlx for additional details). The data in a Trace Structure can be clustered in several different ways by using the functions in boxes #3 and #4 (see Quick_Introduction_To_Clustering.mlx for additional details). In the case of Segment Clustering, a Trace Structure can first be pre-segmented (box #8) and then the resulting Segment Structure can be clustered by using the appropriate clustering mode inside StartClustering_wInput.m (box #4). StartClustering_wInput.m can also accept a matrix of *N* datapoints in *d* dimensions as input for users who wish to make use of the clustering algorithm without assuming any specific data format. Both clustering functions produce a clustering output structure that can then be analyzed using a variety of tools (boxes #5 and #6; see Quick_Introduction_To_Clustering.mlx for additional details). A couple of minor functions (box #7) use as input the cluster assignments produced by ExtractClusterSolution.m (box #6).

Scheme 2 contains the suite of grid-based analysis tools described in Bamberger et al. 2021 (https://arxiv.org/abs/2105.13521). A brief summary is as follows: from the traces contained in a Trace Structure, a GridCorrelationObject can be calculated, which contains coarsened versions of each trace as well as information on the nodes making up those coarse traces. CompareTraceAndCoarseTrace.m (box #9) can be used to visualize the coarsening process for a specific set of traces. From the GridCorrelationObject, a suite of functions can be used to analyze the coarsened traces and correlations between different nodes (box #10; see Quick_Introduction_To_GeneralGridBasedAnalysisTools.mlx for details). In addition, one particularly useful grid-based correlation tools is our Markov Chain Monte Carlo (MCMC) Feature-Finder, which can be run in multiple ways (box #11 and #12), and which produces its own results than can then be analyzed with several functions (box #13; see Quick_Introduction_To_MCMCFeatureFinder.mlx for details).

In addition, the SMAUG toolbox includes several functions for building and working with a "dataset library", which are not listed in Schemes 1 and 2 (see How_To_Format_Input_Data.mlx for additional details).


1
AlignAllTraces.m
ChangeTraceStruct_Attenuation.m
Combine_TraceStructures.m
EvenlyResampleTraces.m
GetTraceStructSubsections.m
Trace
Structure
(TraceStruct)
N x d
Matrix of
Data Points
8
PreSegmentTraces.m
To Grid-Based
Correlation Tools
Segment
Structure
2
Attenuation_FromGaussianFit.m
Conductance1DHist_OverTime.m
CreateCorrelationHistogram.m
CreateDistanceCorrelationHistogram.m
Difference_of_2DHists.m
DisplacementHistogram_AtCut.m
DisplayRandomTraces.m
DisplaySpecificTraces.m
Make1DHist_FromTraceStructure.m
Make1D_DistanceHistogram.m
Make2DHist_FromTraceStructure.m
MakeTimeHistogram.m
Plot_1D_DistHist_OverTime.m
Plot_Overlaid_1DConductanceHistograms.m
Plot_Overlaid_1DDistanceHistograms.m
3
EasySegmentClustering.m
4
StartClustering_wInput.m
Clustering Output
Structure
(OutputStruct)
5
ExtractAndPlotClustering.m
FindCluster_Centroids.m
FindReachabilityPeaks.m
Find_ReachabilityValleys.m
GetPeaksPlotClusterings.m
MatchSpecificClusterValley.m
NestedFullValleyClusters.m
PlotReachabilityWithExtractionLevels.m
SegmentCluster_to_1DHist.m
Show_FullValleyClusters.m
Split_ClusteringSolution.m
Standard_ProcessSegClust_Peaks.m
Valleys_AtSpecificSolution.m
6
ExtractClusterSolution.m
Cluster
Assignments
(Y)
7
GetPopulationTables.m
RandSimilarityIndex.m

Scheme 1. Overview of data-flow between different SMAUG functions. Blue ovals represent specific data structures, orange boxes list specific SMAUG functions, and the arrows indicate the primary data type that each function takes as input and produces as output. Sections that are likely of minimal importance to most users are grayed out to focus attention on the main data-flow pathways. The Grid-Based Correlation tools are shown in Scheme 2 below.


Trace Structure (TraceStruct)
9 CompareTraceAndCoarseTrace.m
GridCorrelationObject.m
11 Run_MCMCFeatureFinder.m
12 standardMCMC_withPreProcess.m
batchMCMC_ProcessAndSave.m
MCMC Output Info
10 PlotNodeCounts.m
PlotTraces_ThroughOneNode.m
PlotTraces_BetweenTwoNodes.m
Show_ConnectionStrengthSummary.m
Show_ConnectionStrength_WithSpecificNode.m
FindDifference_in_DistributionThroughNode_FromExpected.m
NaiveCorrelationWithNode.m
TopScoringTraces_ThroughNode.m
13 sequence_frequency_plot.m
MostSignificantTraces_throughMCMCNodes.m
NodeFreqs_to_1DHist.m
NodeFreqs_to_Overlaid1DHist.m
convergence_evaluation.m

Scheme 2. Continuation of Scheme 1 to include the main Grid-Based Correlation Tools described in Bamberger et al. 2021.

Functions that Modify or Create New Trace Structures

- AlignAllTraces.m
  - Re-aligns all traces to zero inter-electrode distance using the point at which each trace last crosses below a given conductance value
- ChangeTraceStruct_Attenuation.m
  - Modifies the attenuation ratio of a Trace Structure (only relevant for MCBJ datasets)
- Combine_TraceStructures.m
  - Creates a new Trace Structure by combining two or more existing Trace Structures
- EvenlyResampleTraces.m
  - Re-samples all traces in a dataset at evenly spaced values along the distance axis; useful for making the data density in two different datasets comparable
- GetTraceStructSubsections.m
  - Creates a new Trace Structure from one or more sub-sections of an existing Trace Structure

Functions for Analyzing Trace Structures

- Attenuation_FromGaussianFit.m
  - Uses the tunneling slope of each trace to calculate an attenuation ratio, then fits the distribution of these attenuation ratios with a Gaussian function to extract a single “best” attenuation ratio for a dataset
- Conductance1DHist_OverTime.m
  - Overlays 1D conductance histograms for successive chunks of traces from a dataset to determine if any systematic changes are occurring over time
- CreateCorrelationHistogram.m
  - Creates a 2D conductance correlation histogram as developed by Martinek et al. in doi.org/10.1103/PhysRevLett.105.266805
- CreateDistanceCorrelationHistogram.m
  - Creates a 2D distance correlation histogram as described in Bamberger et al. 2020
- Difference_of_2DHists.m
  - Subtracts a distance/conductance histogram for one dataset from a second to produce a 2D histogram of the difference between the two datasets
- DisplacementHistogram_AtCut.m
  - Creates a 1D histogram of the distribution of distances at which each trace last passes below a given conductance value
- DisplayRandomTraces.m
  - Plots randomly chosen traces from a dataset
- DisplaySpecificTraces.m
  - Plots traces identified by their ID#s within a dataset
- Make1DHist_FromTraceStructure.m
  - Creates a 1D conductance histogram for a dataset
- Make1D_DistanceHistogram.m
  - Creates a 1D distance histogram for a dataset by first re-sampling each trace at evenly space values along the y-axis
- Make2DHist_FromTraceStructure.m
  - Creates a 2D distance/conductance histogram for a dataset
- MakeTimeHistogram.m
  - Creates a time histogram for a dataset as introduced by Solomon et al. in doi.org/10.1063/1.4975180.
- Plot_1D_DistHist_OverTime.m
  - Overlays 1D distance histograms for successive chunks of traces from a dataset to determine if any systematic changes are occurring over time
- Plot_Overlaid_1DConductanceHistograms.m
  - Overlays 1D conductance histograms for two or more trace structures
- Plot_Overlaid_1DDistanceHistograms.m
  - Overlays 1D distance histograms for two or more trace structures

Clustering Functions

- EasySegmentClustering.m

  - Implements Segment Clustering as described in Bamberger et al. 2020. Can be used to either perform a single clustering run, or to cluster the same dataset with 12 different minPts parameter values.
- StartClustering_wInput.m
  - Can be used to perform custom clustering runs, either using Segment Clustering or a different clustering mode, with user-adjustable parameter settings
- PreSegmentTraces.m
  - Pre-segments all of the traces from a Trace Structure to create a Segment Structure that can be clustered using StartClustering_wInput. Useful for saving time if the same segments are going to be clustered many times using different clustering parameters.

Functions for Analyzing Clustering Output

- ExtractAndPlotClustering.m
  - Extracts a clustering solution from a clustering output at a given extraction level, and make a plot to display the clustering solution
- ExtractClusterSolution.m
  - Extracts a clustering solution from a clustering output at a given extraction level (note: takes just the RD and CD fields of a clustering output structure as input instead of the whole thing)
- FindCluster_Centroids.m
  - Finds the "median centroid" of each cluster in the parameter space where clustering tool place
- FindReachabilityPeaks.m
  - Locates all peaks in the reachability plot that separate valleys of at least a certain size (note: takes just the RD field of a clustering output structure as input instead of the whole thing)
- Find_ReachabilityValleys.m
  - Locates all valleys in the reachability plot of at least a certain size, each chosen to be as large as possible without merging with a neighboring valley (note: takes just the RD field of a clustering output structure as input instead of the whole thing)
- GetPeaksPlotClusterings.m
  - Finds all peaks in the reachability plot that separate valleys of at least a certain size, then extracts and plots the clustering solution at each of those extraction levels
- GetPopulationTables.m
  - Produces a table of cluster sizes given a list of cluster assignments as input
- MatchSpecificClusterValley.m
  - Given a list of clustering output structures and a specific cluster identified in one of them, finds the single full-valley cluster in each of the other outputs that best matches the specified cluster (based on finding clusters with the most similar median centroids)
- NestedFullValleyClusters.m
  - Displays a reachability plot with each full-valley cluster colored in hierarchically
- PlotReachabilityWithExtractionLevels.m

    - Displays a reachability plot overlaid with the extraction levels corresponding to peaks that separate valleys of at least a certain size
- RandSimilarityIndex.m
    - Given two sets of cluster assignment vectors, calculates the Rand similarity index between those two clustering solutions using the equation developed by William M. Rand in "Objective Criteria for the Evaluation of Clustering Methods" (1971).
- SegmentCluster_to_1DHist.m
    - For a Segment Clustering solution with a specific cluster identified, creates and fits a 1D histogram of the conductance values for all data points belonging to segments in the specified cluster
- Show_FullValleyClusters.m
    - Identifies all "full-valley clusters" of at least a certain size (see Bamberger et al. 2020 for details) and creates a plot to display each one
- Split_ClusteringSolution.m
    - For a given clustering solution, splits the original trace structure for the dataset into separate trace structures for each cluster (note: requires the original trace structure as well as the clustering output structure as input)
- Standard_ProcessSegClust_Peaks.m
    - Given a list of segment clustering outputs for the same dataset and a full-valley cluster specified in one of them, locates the most-similar full-valley cluster in each of the other outputs, then fits the conductance distribution for datapoints belonging to each of those segment clusters
- Valleys_AtSpecificSolution.m
    - Displays the reachability plot with the valleys corresponding to the extracted clusters at a specific extraction level color in

Functions for Analyzing Coarsened Traces and Nodes

- GridCorrelationObject.m
    - This is not a function, but a class definition. It is used to create a grid correlation object for a new dataset, which involves turning each trace into a coarse trace. Different properties of nodes and node-pairs (e.g. connection strength) can then be calculated by calling the appropriate methods.
- Create_StandardGCO.m
    - Creates a grid correlation object for a new datasets stored in a trace structure, then also calls the appropriate methods on it to add all of the "standard" information needed to run most grid-based correlation tools on it.
- PlotNodeCounts.m
    - Makes a plot showing the number of coarse traces that pass through each node.
- PlotTraces_ThroughOneNode.m
    - Plots the coarse traces which pass through a given specified node.
- PlotTraces_BetweenTwoNodes.m
    - Creates a plot which overlays all of the coarse traces that pass through both of two specified nodes.
- Show_ConnectionStrengthSummary.m

  - Makes a plot which shows, for each node, the smoothed root-mean-square (RMS) of the connection strengths of all other nodes with that node. This can sometimes help identify "interesting" nodes, because they should have more very positive and very negative connection strengths with other nodes, leading to higher RMS values.
- Show_ConnectionStrength_WithSpecificNode.m
  - Makes a plot showing the connection strength of each node versus a single selected node. Great for seeing the locations that a given node is correlated or anti-correlated with when you have a node that you already know you are interested in.
- FindDifference_in_DistributionThroughNode_FromExpected.m
  - When a particular node is selected, this function creates a plot showing, for all other nodes, the difference in node-visits between the actual traces that passed through the selected node and random-walk traces passing through the selected node. This information is extremely similar to that found in a connection strength distribution plot made by Show_ConnectionStrength_WithSpecificNode.m, but is more sensitive to small, non-statistically significant differences. On the other hand, these differences can be easier to interpret than connection strengths.
- NaiveCorrelationWithNode.m
  - Makes a plot showing the correlation of all nodes versus a single specified node, simply based on whether traces passed through both nodes or not and not taking into account at all where the two nodes are in relation to each other. We refer to this as the "naïve" correlation because it tends to just identify nearby nodes as correlated, unlike connection strength which knows to expect a trace passing through one node to also pass through nearby nodes. This is only included as a point of comparison; we don't consider it a useful tool.
- TopScoringTraces_ThroughNode.m
  - When a particular node is selected, this function first scores every trace passing through that node according to the average connection strength of all the other nodes the trace passes through, relative to the chosen node. The top scoring traces are then picked out and plotted. This allows any discoveries made with the other tools to be connected back to the original breaking traces by helping to select the traces that actually correspond to a particular rare event, feature, or behavior.

MCMC Feature-Finder and Related Functions

- Run_MCMCFeatureFinder.m
  - Runs the MCMC feature finder on the grid correlation object for a given dataset in order to identify where the most significant node-sequences matching user-specified criteria can be found. Makes several output plots by default.
- standardMCMC_withPreProcess.m
  - Similar to Run_MCMCFeatureFinder.m, but designed for use on a computing cluster where you want to save output plots rather than displaying them. Takes a trace structure as input, so the pre-processing into a grid correlation object is included. Designed to be run using the FeatureFinder_SingleDataset_Script.m script.

- batchMCMC_ProcessAndSave.m
  - Similar to standardMCMC_withPreProcess.m, but for when you want to run the MCMC feature-finder on several datasets using the same set of parameters. Designed to be run using the FeatureFinderLoop_Script.m script.
- sequence_frequency_plot.m
  - Using the node frequencies produced by the MCMC feature-finder, makes a plot showing the probability that each node was included in the node-sequence distribution generated by the MCMC.
- NodeFreqs_to_1DHist.m
  - Takes the node frequencies produced by the MCMC feature-finder and projects them onto either the conductance or distance axes to create a 1D histogram of the node distribution produced by the MCMC.
- NodeFreqs_to_Overlaid1DHist.m
  - Takes the node frequencies produced by the MCMC feature-finder for each of its parallel tempering temperatures and turns them into overlaid 1D histograms by projecting onto either the distance or conductance axis.
- convergence_evaluation.m
  - Performs calculations to evaluate whether or not the MCMC feature-finder has converged by using a version of the Gelman-Rubin diagnostic for each node. Can also make a plot showing histograms of diagnostic values for the different nodes.
- MostSignificantTraces_throughMCMCNodes.m
  - A somewhat more robust version of TopScoringTraces_ThroughNode.m; instead of just scoring the traces passing through a single specified node, this function looks at the node probabilities generated by the MCMC and selects the set of most-probable nodes. It then scores all traces passing through any of those nodes, versus those nodes, and makes plots of the original traces with the highest scores. These traces correspond to the rare feature or behavior identified by the MCMC feature-finder.

Functions for Building and Using a Dataset Library

- build_library.m
  - Reads in the dataset library spreadsheet to create a MATLAB structure array that can be used by other SMAUG functions
- create_name_for_library_entry.m
  - Creates an identifying name for a dataset, given its dataset ID#, from the information about it in the dataset library
- get_matching_library_entry_IDs.m
  - Finds the ID#s of all datasets in the library that match given criteria chosen by the user
- load_library_entry.m
  - Reads in a saved dataset listed in the library, as long as it was saved in SMAUG-Toolbox/DataSets