

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.

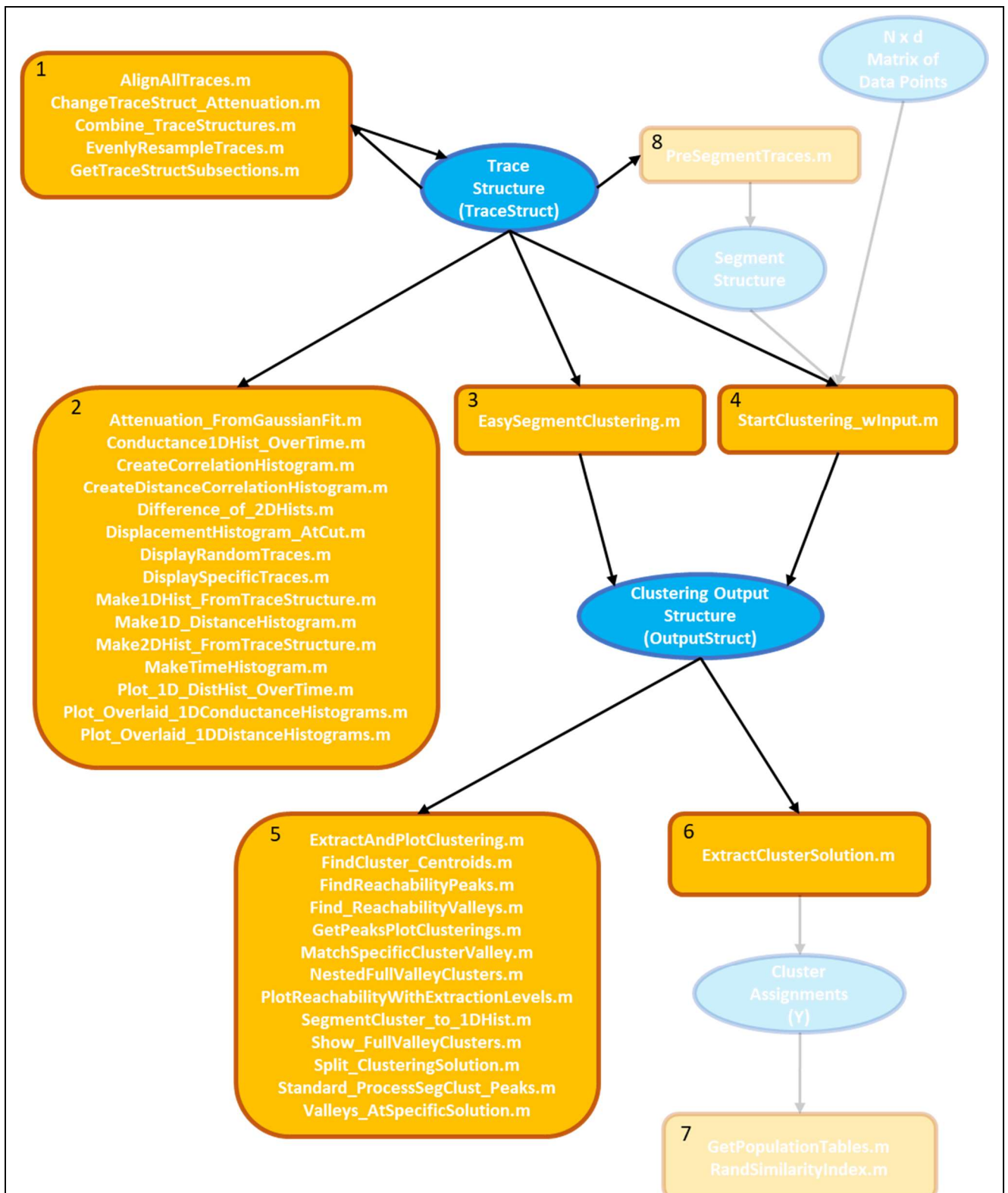
SMAUG functions are designed to balance flexibility with ease of use. To this end, most SMAUG functions can accept numerous input parameters to modify exactly how they behave; however, usually only the first one or two of these input parameters are required. Later parameters will be set to default values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

Function Organization

Scheme 1, below, shows 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.

A brief summary of Scheme 1 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).

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



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.

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 spaced 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 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