**TimeSurfer Graphical Interface**

**User Manual**

**Alpha Version**

**05-May-2010**

[Preface](#Preface)5

[What is TimeSurfer](#WhatIsTimeSurfer)5

[Is TimeSurfer the Right Tool?](#IsTStheRightTool)5

[**Fundamental Concepts**](#FundamentalConcepts)**5**

[Data analysis protocols](#AnalysisProtocols)5

[Matlab toolboxes and data structures](#ToolboxesAndStructures)5

[Scripting and graphical user interfaces](#ScriptingAndGUIs)6

[**Scripting**](#Scripting)**6**

[**GUI Overview**](#GUIoverview)**6**

[Study Mode](#StudyModeOverview)6

[Interactive Mode](#InteractiveModeOverview)7

[**Basic Usage**](#BasicUsage)**7**

[Study Mode](#StudyModeUsage)7

[Study Architecture](#StudyArchitecture)8

[What is the STUDY file?](#WhatIsTheSTUDYfile)8

[Subjects, Batches, and Functions](#SubjectsBatchesFunctionsRelationship)8

[Standard Procedure (starting from scratch)](#StandardStudyScratch)9

[Standard Procedure (starting from template)](#StandardStudyTemplate)9

[Data Processing Logs](#Logs)10

[Study Results](#StudyResults)10

[Modifying Parameters - Existing Studies & Templates](#ModifyingParameters)10

[Working with the Batch Manager & Function list](#BatchManagerAndFunctionList)11

[Managing Large Studies](#LargeStudies)11

[Interactive Mode](#InteractiveMode)15

[Standard Procedure](#StandardInteractive)15

[Other interactive tools](#OtherItneractiveTools)15

[Useful Notes](#InteractiveNotes)15

[Tips, Tricks, and Shortcuts](#GeneralTipsTricksShortcuts)15

[**Advanced** **Topics**](#AdvancedTopics)**16**

[Important Concepts](#ImportantConceptsAdvanced)16

[Parameters: generic, function-specific & global](#ParameterTypes)16

[Experimental conditions & event codes](#Events)17

[Generic Parameters](#GenericParameters)17

[rootoutdir, outpath, prefix](#rootoutdir)17

[function\_id, input\_id](#FunctionIDInputID)17

[datafile, datapath, datastring, dataskip, rootindir, inpath](#Datafiles)18

[save\_flag](#save_flag)18

[cluster\_flag, cluster, clusterscript](#cluster_flag)18

[logfile, logfid](#LogfileLogfid)18

[conditionkey, rejectfile, badchanfile, baselinefile](#SpecialFiles)19

[Advanced Rules for Parameter Specification](#RulesForParamSpec)19

[Target parameter selection, rule specification & application](#RulesSpecificationApplication)19

[Constants, Subject IDs, & parameter substitution](#RulesConstSubjIDParams)19

[Modifying & removing rules](#RulesModifyingRem)20

[Tips](#RulesTips)20

[**Quick Start Guide**](#QuickStartGuide)**25**

[**TimeSurfer Resources**](#Resources)**20**

[Templates](#Templates)21

[Tutorials / Examples](#Examples)21

[Wiki - neuroimaging pbwiki](#Wiki)21

[Google Group - neuroimaging data analysis](#GoogleGroup)21

[External links](#ExternalLinks)21

[**Appendix A - Multimodal Imaging Laboratory**](#MMILGuide)**25**

[Accessing the latest TimeSurfer release](#AccessingTS)25

[Getting started](#MMILGettingStarted)25

[Using MMIL clusters](#MMILclusters)25

[Using MMIL storage](#MMILstorage)25

[**Appendix B - Function Reference**](#FunctionReference)**25**

[Functions listed in the TS Graphical Interface](#FunctionReference)25

[global](#GlobalReference)25

[ts\_process\_ieeg\_data](#ts_process_ieeg_data)8

[ts\_process\_fif\_data](#ts_process_fif_data)8

[ts\_statistics\_wrapper](#ts_statistics_wrapper)8

[ts\_freqanalysis\_fieldtrip](#ts_freqanalysis_fieldtrip)8

[ts\_freqband\_average](#ts_freqband_average)8

[ts\_timefreq\_rejection](#ts_timefreq_rejection)8

[ts\_timefreq\_baseline](#ts_timefreq_baseline)8

[ts\_PLV](#ts_PLV)8

[ts\_dSPM](#ts_dSPM)8

[ts\_tf\_dSPM](#ts_tf_dSPM)8

[ts\_preproc](#ts_preproc)8

[ts\_reject](#ts_reject)8

[ts\_data\_selection](#ts_data_selection)8

[ts\_trials2avg](#ts_trials2avg)8

[ts\_combine\_conditions](#ts_combine_conditions)8

[ts\_autoICA](#ts_autoICA)8

[ts\_manualICA](#ts_manualICA)8

[ts\_ezplot](#ts_ezplot)8

[ts\_syncplot](#ts_syncplot)8

[Table of standard parameter names](#StandardParamsReference)25

[**Appendix C - TimeSurfer Development**](#Development)**25**

[Subversion Repository](#devSVN)25

[TimeSurfer structures](#devTSstruct)25

[TimeSurfer functions](#devTSfunc)25

[Master files](#devTSmaster)25

[Future developments](#devFuture)25

Note: The Preface, Fundamental Concepts, and GUI Overview present the main ideas and challenges that motivated the design of the graphical interface. Several tutorials have been prepared to illustrate how template setup files can be used to quickly set up and execute complicated analysis protocols. The Basic Usage section should be read before working through tutorials or attempting to modify templates. The TimeSurfer Resources section and Appendix A discuss how to access the software, report bugs, and other logistical issues. Advanced Topics and the Function Reference provide information that is especially important for working with large studies or creating setup files from scratch for new analysis protocols.

**Preface**

**What is TimeSurfer?**

TimeSurfer is a Matlab toolbox developed originally at UCSD's Multimodal Imaging Laboratory to analyze neuroimaging data. MEG, EEG, and intracranial EEG (MEEG) data are represented using a generic structure within TimeSurfer, enabling the same functions to analyze data acquired by different imaging modalities. This approach is similar to FieldTrip's, and it is easy to convert between TimeSurfer and FieldTrip data structures. In addition to offering new tools, the TimeSurfer package wraps around functions of several third party packages like FieldTrip and EEGLAB, making it easier to implement complex analysis protocols that rely on multiple software packages.

A recent addition to the TimeSurfer (TS) toolbox is a graphical program (TS GUI) that encapsulates TS functions and orchestrates complex analysis protocols. The primary purpose of this manual is to introduce the new graphical interface and provide sufficient documentation for researchers to begin using it. Particular TS functions are documented in the Function Reference and demonstrated in the TS tutorials and examples. The fastest way to get started using the TS GUI is to read the Quick Start Guide in Appendix B.

**Is TimeSurfer the Right Tool?**

FieldTrip vs EEGLAB vs Chronux vs TimeSurfer

**Fundamental Concepts**

Matlab toolboxes like TimeSurfer, FieldTrip, and EEGLAB can be used to analyze MEEG time series to study the brain and cognition.

**Data analysis protocols**

Analysis protocols take experimental results and transform them into an interpretable form that can help answer experimental questions. They consist of sequences of computational operations and procedures that may involve both human interaction and automated processes. They may yield quantitative results, images, sounds, or other perceptible manifestations that are more informative than the original time series.

(excellent discussion on analysis protocols <http://fieldtrip.fcdonders.nl/tutorial/introduction>)

**Matlab toolboxes and data structures**

"Matlab is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran" (<http://www.mathworks.com/products/matlab/>). A Matlab toolbox is a set of functions that work together for some common purpose. Several Matlab toolboxes have been developed to facilitate the analysis of MEEG time series. Examples include FieldTrip, EEGLAB, and Chronux. The time series and information about it are usually stored in toolbox-specific, standardized structures that the toolbox functions are designed to process. Functions in toolboxes for processing MEEG data typically input a data structure, perform some operation, and return a structure containing the results.

**Scripting and graphical user interfaces**

Functions in a toolbox can be used by executing commands that include them or by interacting with a graphical interface that encapsulates them. The former method is employed at the command-line or in scripts and gives maximum control to users (ex. FieldTrip). The latter method uses user interface controls such as buttons and list boxes and usually gives the user very little control beyond what was hard-coded in the graphical interface (ex. EEGLAB). TimeSurfer has relied on a script-based approach for several years. The new graphical interface introduces a novel middle ground between approaches that are exclusively script-based and others that are based on graphical interfaces.

**Scripting**

See the [FieldTrip](#ExternalLinks) documentation for an introduction to the script-based approach. The same approach can be adapted to TimeSurfer by substituting functions from the TimeSurfer function reference at the end of this manual.

Analysis protocols in matlab: <http://fieldtrip.fcdonders.nl/tutorial/analysis_protocols>

Creating a clear and efficient script: <http://fieldtrip.fcdonders.nl/tutorial/scripting>

**GUI Overview**

The TimeSurfer (TS) graphical user interface (GUI) was developed (1) to facilitate batch processing and (2) to enhance user-driven data analysis & exploration [by combining the efficiency and ease of using graphical interfaces with the low-level control of a script-based or command-line approach].

The GUI has two operating modes, one for each objective. *Study mode* is for batch processing and *Interactive mode* is for user-driven data processing. Most Matlab toolboxes for neurophysiological data analysis focus on user-driven data processing. Study mode is possibly the most important feature of the TS GUI that is not found in most other toolboxes. Interactive mode in the TS GUI fills an important gap between approaches taken by other toolboxes (this will be discussed below).

**Study Mode**

The idea: in the simplest case, an analysis protocol is a sequence of functions in which the data input to one function is the output from the previous function. The data can be conceptualized as flowing down a stream being repeatedly transformed along the way. The effect of a particular function on the input data is determined by the values assigned to the function-specific parameters. Consequently, an analysis protocol can be defined by listing the parameters with values in an ordered list of functions. This is the idea behind Study mode.

In Study mode, the user selects functions from a drop-down menu ([Figure 1](#Figure1)). The parameters of selected functions are then listed with default values in the order that the functions are selected. The user can then change parameter values as desired and execute the sequence of functions.

In more complicated protocols, the stream may split into different paths. In the case of diverging processing streams, it makes more sense to think of an analysis protocol containing two independent sequences of functions. In Study mode, distinct (not necessarily independent) sequences of functions are defined as different batches. Consequently, analysis protocols may have multiple batches.

It is often desirable to apply the same analysis protocol to multiple data sets, possibly representing time series recorded from different subjects in an experiment. In this case, the same or similar analysis protocol will usually be applied to multiple subjects with only a small subset of parameters having values that vary across subjects. Parameters that vary across subjects specify information like output directories and raw data files. The primary purpose of Study mode is to simplify the specification of analysis protocols for studies that include multiple subjects with multiple (possibly different) batches and parameters that vary across subjects. A second aim of Study mode is to centralize (1) the specification of analysis protocols, (2) browsing subject-specific results that may be distributed over a disorganized directory tree, and (3) recording and retrieving notes about the results of the analysis.

Study mode organizes parameters by subject, batch, and function, makes it easy to define and modify analysis protocols for large studies, and connects analysis protocols to the results and observations.

**Interactive Mode**

Interactive mode can be used for data exploration, visual artifact rejection, on-demand visualization, and user-driven analysis of a small number of data sets. It is an interface for working with data sets interactively. The same data can be processed seamlessly using both the GUI and the command-line. The analysis specification and results can be saved to disk. Whereas FieldTrip is most frequently used for script-based user-driven data processing and EEGLAB is used for GUI-based user-driven data processing, Interactive mode in the TS GUI represents a hybrid of script-based and GUI-based approaches.

Note: "user-driven processing" is any analysis where the user must execute each operation. It contrasts with automated (batch) processing.

**Basic Usage**

**Study Mode**

Study mode is for automated batch processing. It can be used to manage analyses for a task with multiple subjects and/or multiple batches. It is an interface for specifying parameters for all analyses for all subjects for a single task. That is, Study mode is used to create and modify a task-level analysis specification and to execute analysis protocols as specified. The complete study specification can be saved to a MAT-file and re-loaded at a later point in time. Furthermore, Study mode includes a browser for viewing results by subject and recording observations.

Study Architecture:

A study specification consists of one or more subjects where each subject can have multiple independent batches, each batch contains functions that contribute to analysis protocols or that stand alone, and each function has parameters that are assigned values. Associated with each function are the required and optional parameters of the function as well as higher-level parameters that tell the GUI how you want the function to be executed (ex. whether to execute it locally or submit it to a cluster).

What is the STUDY file?

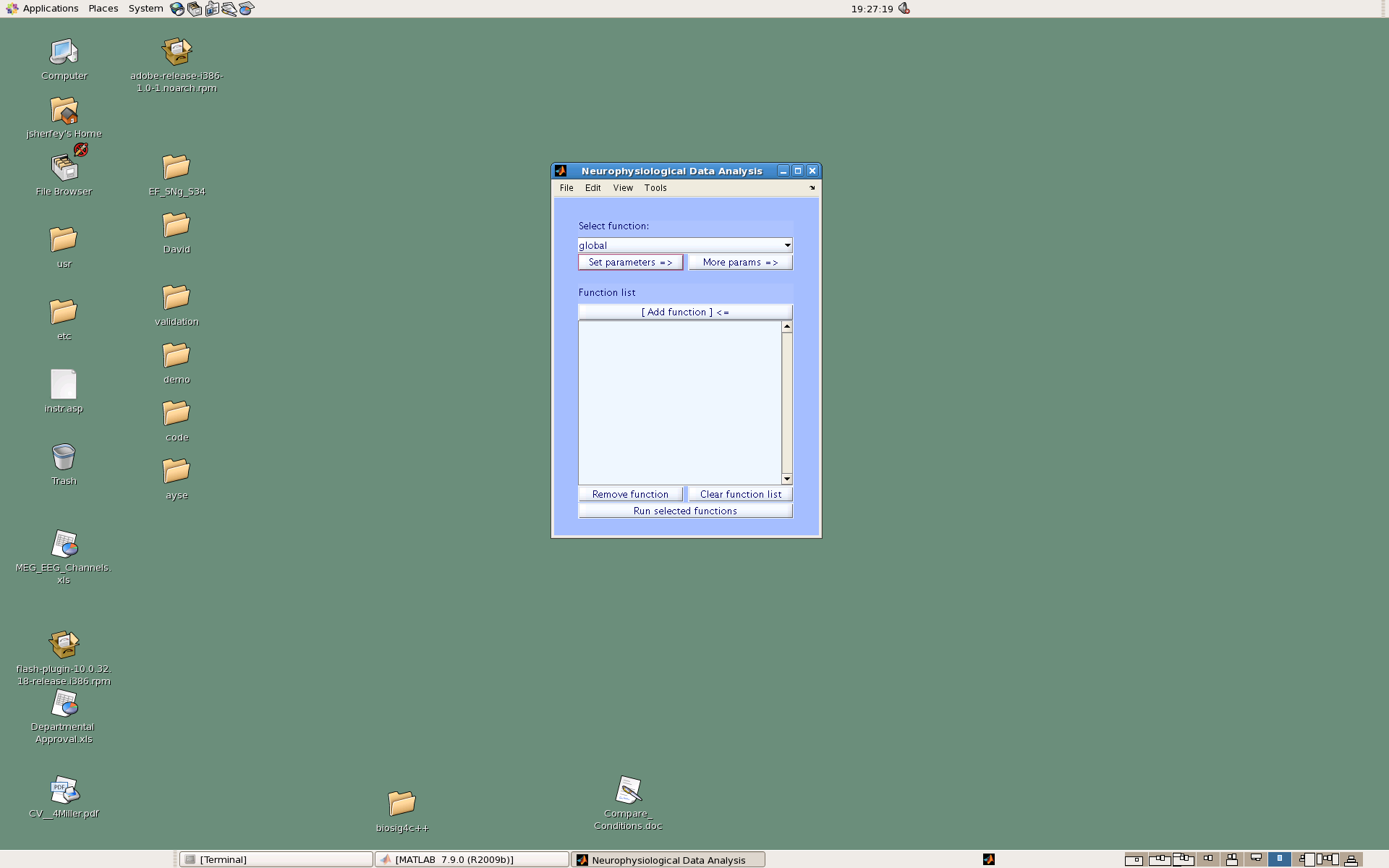


Figure 1. The main figure in study mode.

Function List

Function Drop-Down Menu

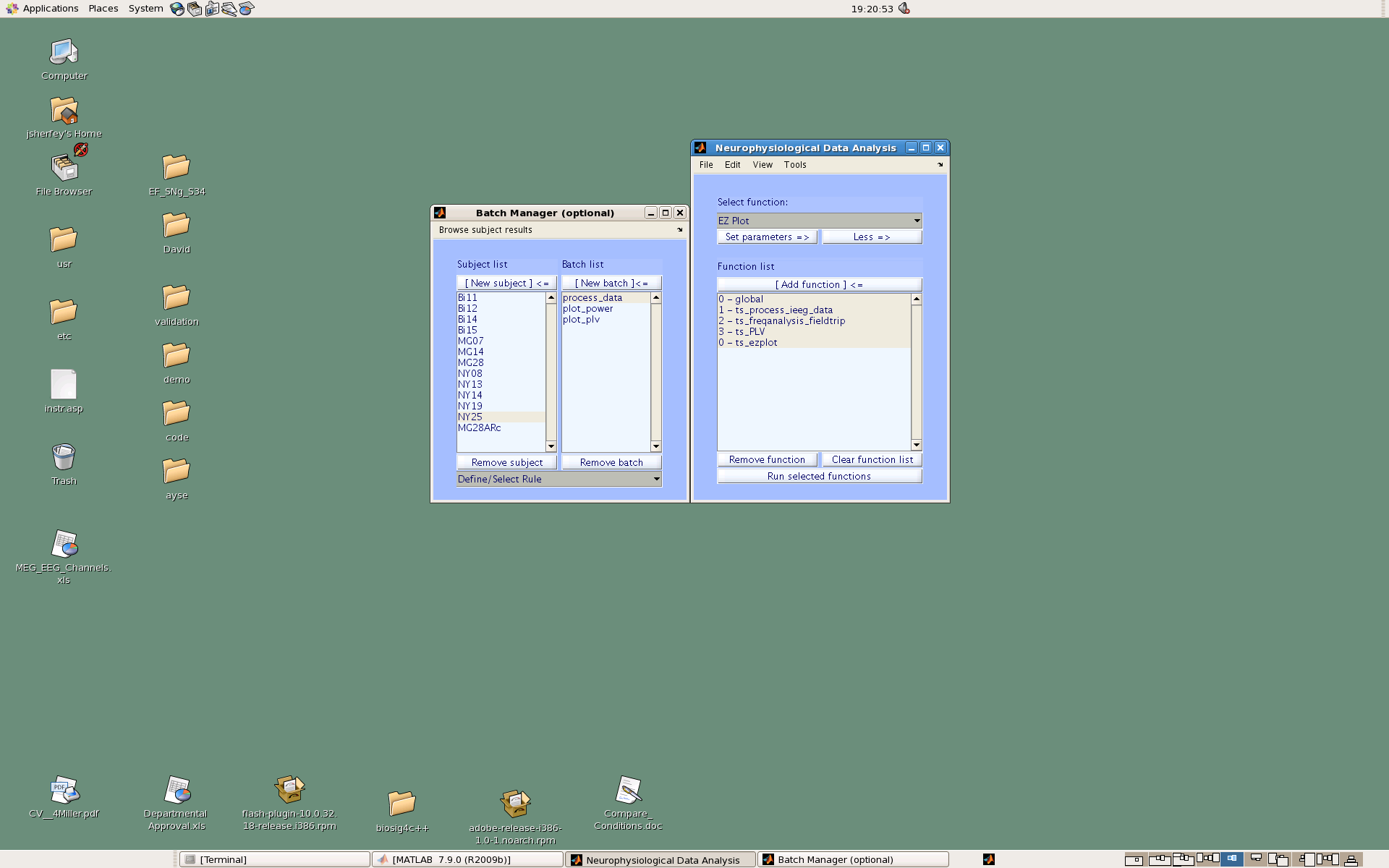
Run Button

It is a MAT-file that contains parameter specifications for all subjects, batches, and functions as well as information on the state and appearance of the GUI controls. Loading the STUDY.mat file will restore the study specification and GUI display present at the time of saving ([Figure 2](#Figure2)).

How are changes to subjects, batches, and functions related?

When the user selects a function from the drop-down menu on the main figure, the function's parameters are added to the [right parameter figure](#RightParameterFigure). Functions added to the function list are defined by the parameter values in the right parameter figure. New batches are always defined by the contents of the function list, and new subjects are always defined by the contents of the [batch list](#Figure2). Thus, parameters in the [right parameter figure](#RightParameterFigure) define functions added to the function list, which defines batches added to the batch list, which defines subjects added to the [subjects list](#Figure2). If a subject with a batch is not defined, then a template subject and batch will be created to store parameter values for functions added to the function list ([Figure 3](#Figure3)). Any modification of the function list or a parameter in it represents a persistent change to the function list's parent batch; any function in the function list can be modified ([see Modifying Parameters From Existing Studies & Templates](#ModifyingParameters)).

Note: default subject and batch names can be changed by right-clicking in the appropriate list. Adding a new subject while the batch list is empty results in a new item in the subject list without being associated with any batches. However, batches can be added later. The batch and function lists work the same way.



**Browse**

**Results**

**Rules**

Figure 2. Reloading previous study or template restores the state at the time of saving.

**Subject**

**List**

**Batch List**

**Function List**

Standard Procedure (starting from scratch):

1. Select functions from the main figure's drop-down menu and specify their parameter values in the [right parameter figure](#RightParameterFigure) ("New Function Parameters"). Tip: hover over parameter names to get parameter info.
2. Click "Add function" to add functions in the right figure to the function list.

Note: this will open the Batch Manager and define a template subject and batch #1. Both the subject and batch can be renamed by right-clicking in the associated list.

1. Execute the analysis protocol by clicking "Run selected functions."
2. Save the study for later use (File => Save study / Load study).

Standard Procedure (starting from template):

1. Load the template (File => Load study).
2. Set common analysis parameters for one subject to task-specific values ([see Modifying Parameters](#ModifyingParameters)).
3. Make one copy of the modified template subject for each subject in the study.
4. Set values for parameters that vary across subjects, manually or using rules ([see Advanced Rules](#RulesForParamSpec)).
5. Other possible alterations: reorder batches or functions, combine batches, copy & modify functions, etc.
6. Select a set of subjects, batches, and functions to execute and click "Run selected functions." Only the selected functions will be executed.

Data Processing Logs:

* Standard output ("session.log")
* List of files created ("outfiles.log")
* Parameter list ("parms.log")
* Parameter mat file ("session.mat")
* STUDY definition file ("TIMESURFER\_STUDY\_date.mat")

Note: the STUDY definition file is saved to Logs in the current directory. All other logs are saved to the root output directory (rootoutdir, current directory by default).

Study Results:

* Browsing results

Open the Results Browser by clicking on Browse subject results in the Batch Manager's menu ([Figure 2](#Figure2)) or by selecting View => Browse results. The browser consists of two three boxes next to each other with buttons for moving to the previous or next subject ([Figure 4](#Figure4)). The leftmost box lists all directories where results were successfully saved for the current subject. The center box lists all the files in the selected directory. The rightmost box is used for recording notes (comments) about the contents of the file selected in the center box.

* Viewing results

File content is displayed in a content-specific manner when a file is selected in the center box. Images (ex. jpeg) and text (ex. log files) pop up in a new window, MAT-files (ex. TS structures) are loaded and temporarily activate Interactive mode, and audio files (ex. wav) are played.

* Recording Observations

Any text typed in the right (comment) box will be automatically recorded along with the full filename of the file selected in the center box and a timestamp. The first item in the left box is the word "Notes." Clicking on Notes will cause all files with recorded notes to be listed in the center box. Clicking on one of those files in the center box will display the file content as before and the note in the comment box. Notes can be modified or deleted at any time.

* Viewing and exporting notes

Notes can be recorded for any file for any subject. To display (in the command window) all notes for all subjects, click on "All Notes" below the comment box. To save all notes to a text file, click on "Export Notes."

Modifying Parameters From Existing Studies & Templates (three methods):

* Parameter Overview.

Click "Parameter overview" in the [left parameter figure](#LeftParameterFigure) or View => Parameter overview. Modify values in the text box and click "Save & Close." This works for only one subject at a time.

* [Left parameter figure](#LeftParameterFigure) ("Study Parameters").
  + Right-click on parameter value, modify value, & click "Ok."
  + Use auxiliary controls if present (ex. for file or directory selection).

Note: can be used to set the same parameter for multiple subjects with similar batches simultaneously.

* Rules ([see Advanced Rules for Parameter Specification](#RulesForParamSpec)).

Working with the Batch Manager & Function list:

* Add, copy, rename, remove, reorder (subjects, batches, & functions)
  + Add: An empty subject can be created if the batch list is empty. An empty batch can be created if the function list is empty ("Clear function list" button). Functions must be selected from the function drop-down menu in the main figure before they can be added to the function list.
  + Copy: subjects and batches can be copied by selecting the item to copy and clicking "Add." Functions can be copied in the same way from the function list if the [right parameter figure](#RightParameterFigure) does not have any functions to display.
  + Rename: subjects and batches can be renamed by right-clicking on the appropriate label in the subject or batch list and then typing the new name. Functions cannot be renamed.
  + Remove: the "Remove" button removes items from lists. Multiple functions can be removed simultaneously from the function list.
  + Reorder: the "u" and "d" keys can be used to move an selected item **u**p or **d**own in the function, batch, and subject lists.
* Modifying several subjects simultaneously
  + When multiple subjects are selected, only the batches that are present in all selected subjects will be displayed in the batch list.
  + Batches can be added or removed for multiple subjects simultaneously using the same buttons as for one subject. However, there are a couple differences.
  + Importantly, if a batch is copied for multiple subjects, subject-specific values are preserved by copying the batch separately for each subject. On the other hand, when a new batch is added to multiple subjects simultaneously, the same batch will be added to all of them.

Managing Large Studies (subjects, batches, rules, results):

Event-related task analyses often involve multiple subjects and complicated analysis protocols. In such cases, you will want to define multiple subjects and may create multiple batches.

* More reasons for defining multiple batches include
* Separating conceptually distinct procedures.
* Separating functions that require different global values.
* Separating sequences that are interrupted by delays (ex. manual steps, cluster queue).
* Subject templates - a subject template should have all the batches for the analysis protocol with task-specific parameters values (the values that will not change across subjects). New subjects should be created by copying the template: select the template and click "Add subject." Complete the analysis specification for the subject by setting subject-specific parameters (manually or using rules).
* Rules are very useful when working with a large number of subjects. Rules specify how a parameter value varies as a function of the subject. It needs to be applied only once to set the value for every instance of a parameter for all subjects ([see Advanced Rules for Parameter Specification](#RulesForParamSpec) for details).

Example: If a plotting function is used 10 times in the analysis protocol, requires a parameter that stores a subject-specific file name, and there are 20 subjects, then you would have to set that parameter 200 times. If you use the TS GUI naming convention and create a rule for the parameter, then setting all 200 values is as easy as selecting all subjects and choosing the appropriate rule from a drop-down menu.

**Left Parameter Figure** **Right Parameter Figure**

**"Study Parameters" "New Function Parameters"**

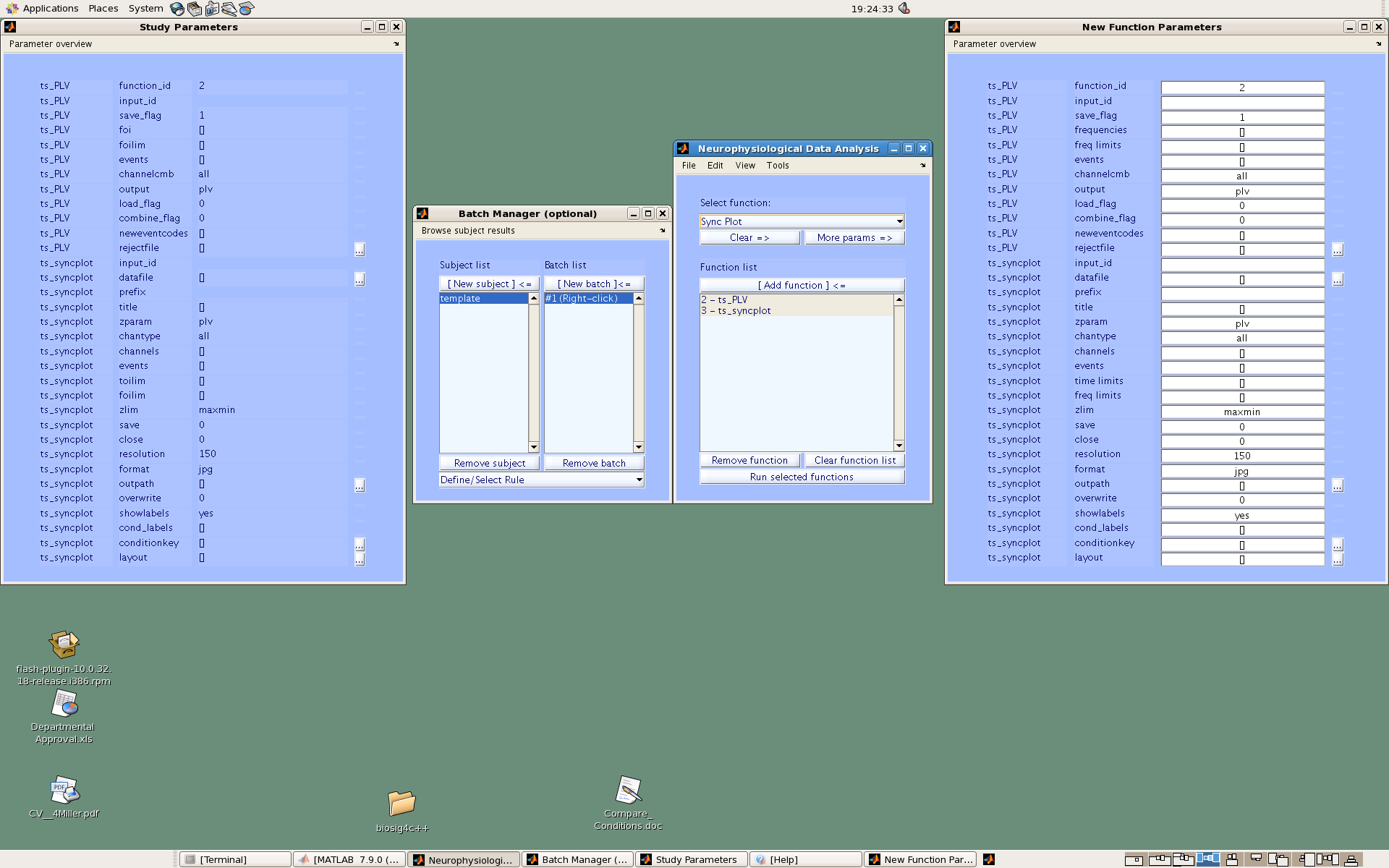
****

Figure 3. Adding functions to study using the default subject and batch. Note that parameter values are more difficult to change once they are added to a batch because all changes are persistent and there is no undo option. Also note that subject and batch labels can be changed by right-clicking.

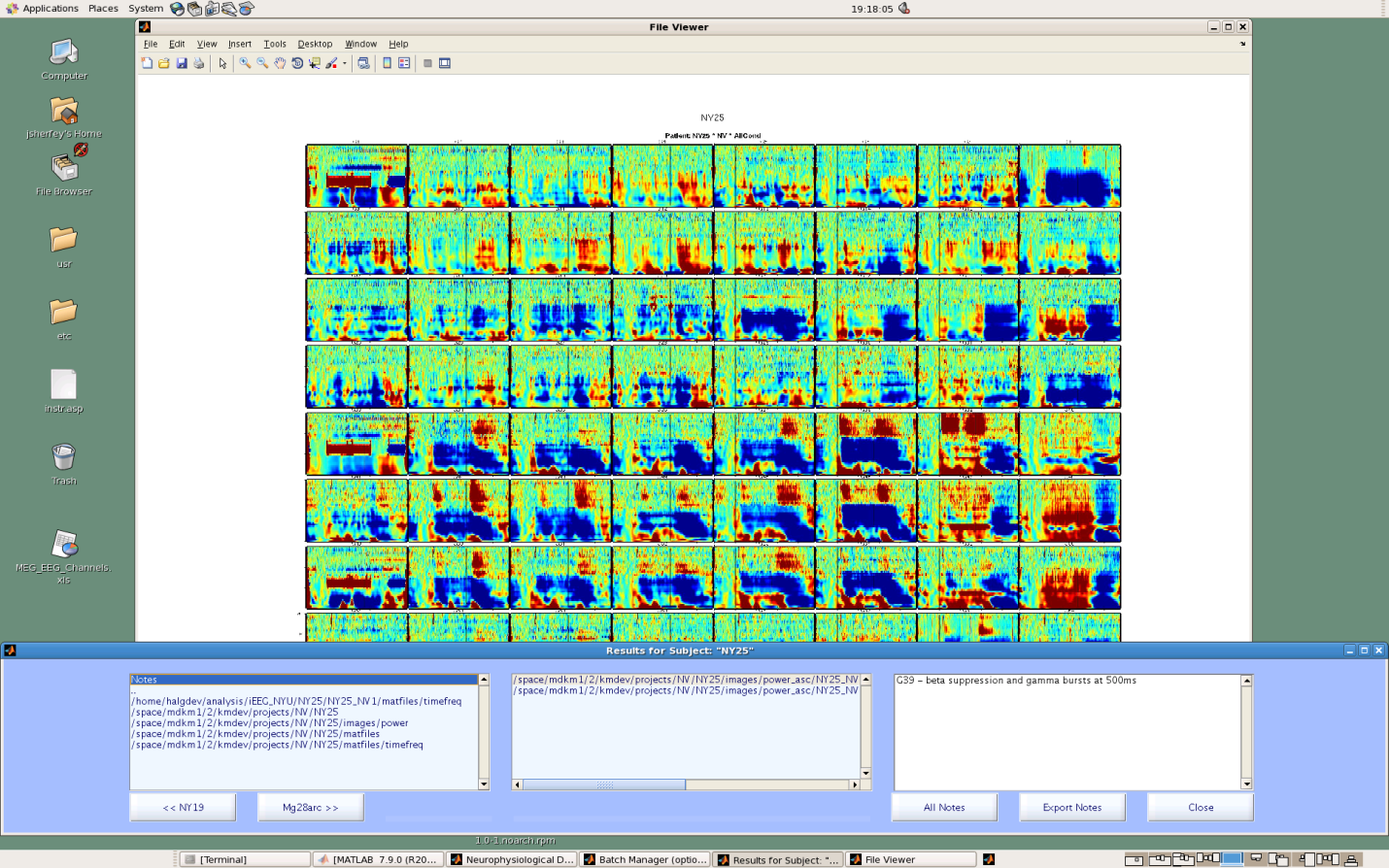


Figure 4. Browsing output directories, viewing file content, recording observations in comment box, and retrieving a list of noted files.

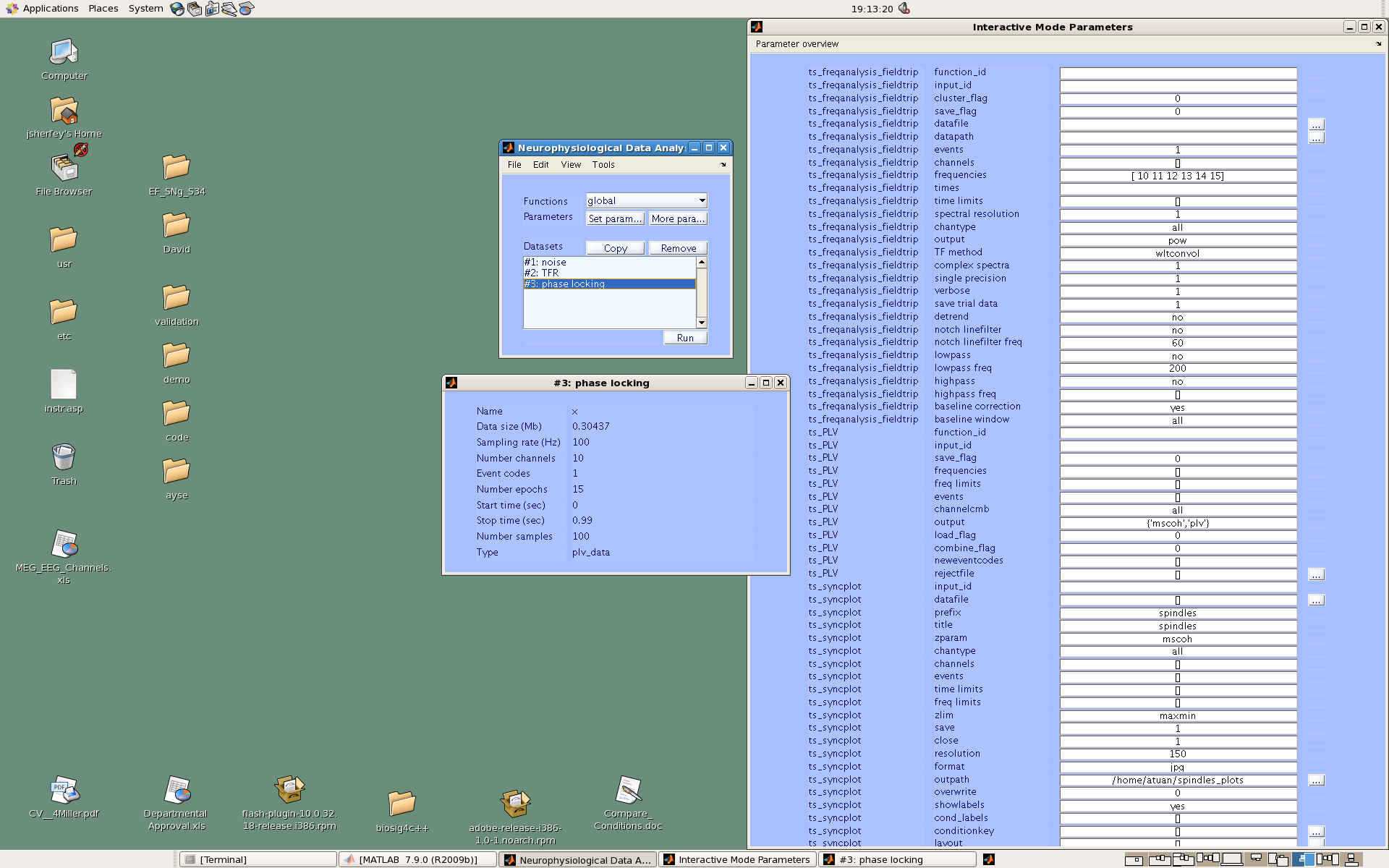


Figure 5. Interactive mode. Calculating and plotting phase-locking values in interactive mode. Upon clicking run, the data will be passed from one function to the next in the order that functions are listed in the right figure.

**Interactive Mode**

Standard Procedure:

1. Getting data into TS GUI - Two options: Load & Grab.
   1. File => Load data.

Supported file types:

* + 1. EEG formats: \*.eeg, \*.avg, \*.cnt, \*.vhdr; \*.set (coming soon)
    2. MEG formats: \*.fif
    3. Matrices: dimensions must be [chan x time x trials] or [chan x time x freq x trials].
    4. MAT-files: containing TimeSurfer structures or matrices.
  1. File => Grab data. Use data already in base workspace.

1. Select function in drop-down menu and specify parameter values (repeat for multiple functions).
2. Select data set in Datasets list ([Figure 5](#Figure5)).
3. Click "Run" below the Datasets list.
4. Save results (File => Save data) and the analysis parameters (File => Save study).

Other interactive tools:

* Artifact rejection (Tools => Artifact rejection).
  + Visual rejection using FieldTrip tools (Tools => Artifact rejection => Visual rejection).
  + Save reject info in text file and MAT file (Tools => Artifact rejection => Save reject\_data).
  + Display reject info in command window (Tools => Artifact rejection => View reject\_data).
* EZ Plotting (Tools => Plot => ezplot). Plot ERPs, epochs, spectral power, and other results.

Useful Notes:

* Multiple datasets can be read and renamed by right-clicking labels in the Datasets list.
* Remove a function from "Interactive Mode Parameters" by right-clicking on the function name.
* Get parameter documentation by right-clicking parameter names in "Interactive Mode Parameters."
* A dataset will be overwritten if the resulting data type matches the original data type.
* A dataset can be copied by clicking "Copy" or removed by clicking "Remove."
* Start TS GUI in Interactive mode by typing: timesurfer('interactive').

**Tips, Tricks, and Shortcuts**

* A STUDY file is saved automatically every time an automated process completes. It is saved to Logs/TIMESURFER\_STUDY\_*date*.mat. Loading that file and clicking "Run selected functions" will repeat the process. Loading that file, modifying parameters, and clicking "Run selected functions" can be done to run a variant on a previous analysis.
* Standardize your directory structure & naming convention. This is especially useful when defining rules.
* Figures can be redrawn if you close them by typing: timesurfer('redraw'). This is very important if you accidentally close the main figure before saving your results.
* Add parameters (Edit => Add parameter): adds a parameter with a user-specified default value to every instance of a user-selected function.
* Remove parameters (Edit => Remove parameter): removes a parameter from every instance of a user-selected function.
* Add scripts (Edit => Add script): adds a script to the function list.
* You can start the GUI in interactive mode by typing: timesurfer('interactive').
* "u" & "d" in function, batch, and subject lists to reorder items.
* "d" in rule definition window to remove rule.
* Scroll bar (not very good).
* Show More/Less parameters. Less (default) displays those parameters that are considered basic. More displays all available parameters.
* Right-click function name in [right parameter figure](#RightParameterFigure) to remove it from list.
* Right-click parameter names in right parameter figure for parameter documentation.
* Right-click subjects, batches, and datasets in corresponding lists to rename them.
* If a function is shown in the [left parameter figure](#LeftParameterFigure) and then selected in the [function drop-down menu](#FunctionDropDownMenu), then the parameters in the right parameter figure will match those in the left.
* A function selected in the function list will be copied when "Add function" is clicked if the right parameter figure is not shown.
* Combining batches: select all batches to combine and click "Add batch." To combine a subset of multiple batches, do the same thing and then remove the unwanted functions from the function list in the newly created batch.

**Advanced Topics**

**Important Concepts**

* Parameters: generic, function-specific, and global
  + **Generic** **parameters** tell TS GUI what to do and can be set for any function
  + **Functions-specific parameters** are the optional and required parameters for particular functions (see the [Function Reference](#FunctionReference) or type help *function\_name* in Matlab for a list of available parameters)
  + Each batch can contain at most one set of global parameters. Any generic or function-specific parameter can be defined as a global parameter. All **global parameters** are passed to all functions in the batch. This is useful for parameters specifying things like output directories that will be common to multiple functions. For instance, if you want to save the results of five functions to a directory during one run and then to a different directory in a later run, the parameter specifying the output directory can be made global and modified once; otherwise, the parameter would have to be changed for all five functions separately. Defining global parameters can reduce chances for human error and time spent adjusting parameters. Important note: if the same parameter is defined as global and is also set within a function's parameter list, then the function-specific parameter value will override the global parameter for that function (the other functions will still use the global value). This can be useful when most (but not all) functions have the same value for the same parameter. To define a parameter as global, select "global" from the [function drop-down menu](#FunctionDropDownMenu). If the parameter is missing from the [right parameter figure](#RightParameterFigure), use **Edit => Add Parameter** to add the desired parameter.
* Connection between analysis protocols, functions, and batches ([see Study mode](#SubjectsBatchesFunctionsRelationship)).
* Events and conditions in epoched data (REWORD)

Experimental conditions are identified using event codes in TS GUI. A file called *conditionkey* (see below) can be used to map event codes to condition labels for generating plot legends. Most TS processing functions have an **events** parameter. In many such functions, the list of events can be specified two ways, depending on how you want the data to be processed. In many cases, for instance, events 1-4 could be specified by setting *events* to either [1 2 3 4] or {[1] [2] [3] [4]}. The former syntax would cause the all four conditions to be loaded and processed together. The latter syntax would cause the first condition to be loaded, processed, cleared from memory, and so on. The second syntax is useful for memory management but only works when conditions can be processed independently. When the second syntax is used and jobs are submitted to the cluster, then the independently-processed conditions will be submitted to the cluster as separate jobs and processed in parallel. Refer to the Function Reference for more information on function-specific usage of the *events* parameter or right-click on the parameter name in the right parameter figure in TS GUI.

**Generic Parameters**

The most frequently used generic parameters are *rootoutdir*, *prefix*, *datafile*, *conditionkey*, and *rejectfile*. Other important parameters include *function\_id*, *input\_id*, and *cluster\_flag*.

* **rootoutdir, outpath, prefix**

All output files will be saved in the *rootoutdir* directory (or a subdirectory). *rootoutdir* is required by many functions and is set to the current directory if not set by the user. It is recommended that the user set *rootoutdir* to different directories for different subjects. *outpath* is an optional parameter used to name subdirectories for saving output files. The user does not usually need to set *outpath*. When *outpath* is set, files will be saved to *rootoutdir/outpath*. *prefix* specifies a prefix to use for all output file names.

* **function\_id, input\_id**

A given step in an analysis stream may process data stored on disk or in memory. In either case, the data to be processed is often the output of an earlier step. *function\_id* is an optional, generic parameter that can be used to identify a particular step. *input\_id* is an optional, generic parameter that can be used to link a function's input to the output of an earlier step. The two are linked by setting the *input\_id* of one step to the *function\_id* of an earlier step. Both parameters must be numeric. Different instances of the same function in an analysis protocol can have different *function\_id*'s. It is also acceptable for different functions in a protocol to have the same *function\_id*; in that case, their results would be combined and passed together to any downstream function with a matching *input\_id*. TS GUI assigns default values to *function\_id* if it is not set by the user. Functions in the function list are always listed with their *function\_id*. Having the function\_id in the function list makes it easy to identify what value to use for a new function's input\_id. These parameters are most useful when data are saved to multiple files by upstream functions.

* **datafile, datapath, datastring, dataskip, rootindir, inpath**

Files containing data can be specified in several ways. In many cases, this can be done most simply by setting function\_id and input\_id. However, in other cases it may be better to specify the files yourself. TS GUI has several generic parameters that it uses to try to locate the data if it is not present in memory and input\_id is empty. Set *datafile* if you know the full filename(s). Set *datapath* to specify the path where the data are located. *datapath* and *datafile* can be used together if *datafile* is a list of filenames that exclude paths. *rootindir* and *inpath* are related in a manner similar to *rootoutdir* and *outpath*. They provide an alternative means of specifying the input data path. If a path is specified using any means without giving file names, then all files in the specified directory will be read. *datastring* and *dataskip* can be used to eliminate incorrect files from a list of files. *datastring* can be used to specify a string that must be present in all input file names. *dataskip* can be used to specify a string that cannot be present in any input file name. Any file names containing the string given by *dataskip* and any files not containing the string given by *datastring* will be excluded from the list. *datastring* and *dataskip* can be used together and either parameter can be a string or a cell array of strings.

Tip: Use *datafile* to specify raw data files in the first function of the analysis protocol (ex. *ts\_process\_ieeg\_data*), then use *function\_id* & *input\_id* for most subsequent processing; also, specify *datastring* + *dataskip* if *input\_id* produces a list of files that is not selective enough.

* **save\_flag**

*save\_flag* can be used to specify whether or not results should be saved to disk. If *save\_flag* is set to 0, the output will be passed directly to the next function that recognizes the output (the "recognition" process is a program design issue covered in the Developer documentation). If *save\_flag* is set to 1, the output will be saved to a filename that is automatically constructed based on *rootoutdir*, *outpath*, *prefix*, and function-specific parameter values. The filename can be specified by the user using the *filename* parameter.

* **cluster\_flag, cluster, clusterscript**

TS GUI can process functions in several ways. By default, all functions will run on the local machine. However, it can also generate stand-alone Matlab scripts that execute the function. Furthermore, TS GUI can submit those scripts to a queue on a Rocks cluster following the Multimodal Imaging Laboratory's standard procedure. A script can be generated and submitted to a cluster for any function by setting *cluster\_flag* to 1. When *cluster\_flag* is set to 1, TS GUI submits the job to the cluster specified by *cluster* using the job submission linux shell script specified by *clusterscript*. By default, *cluster\_flag* is set to 0, *cluster* is set to "mmilcluster," and *clusterscript* is set to "qmatjobs."

* **logfile, logfid**

Standard output is recorded to a log file and/or displayed in Matlab's command window. Standard output will be appended to any file specified using *logfile*. If a file is already open in matlab with a file identifier, fid (see fopen), then standard output can be appended to the open file by setting *logfid* to fid. Standard output is only displayed in the command window by default.

* **conditionkey, rejectfile, badchanfile, baselinefile**

Several files can be used by multiple TS functions and are recognized by TS GUI. *conditionkey* specifies a CSV file listing event codes and condition labels used to generate plot legends. *rejectfile* specifies a MAT file containing information on which trials and channels should be excluded from the analysis. *badchanfile* is a TXT file listing channel labels to exclude from the analysis. *baselinefile* is a MAT file containing data to use for calculate baseline means, standard deviations, etc when subtracting the baseline, calculating z-scores, or performing other operations that depend an explicitly-defined baseline interval.

* **itype, otype**

Recap: parameters can be function-specific or generic and optionally global in scope. The flow of data through the analysis protocol can be controlled by creating multiple batches and setting the following generic parameters: function\_id, input\_id, datafile, datapath, datastring, dataskip, rootindir, inpath, and save\_flag. Where files are saved and how they are named can be controlled by setting rootoutdir, outpath, prefix, and filename. Standard output can be logged by setting logfile or logfid, and cluster computing is controlled using cluster\_flag, cluster, and clusterscript. Any parameter can be global, but redundant values assigned at the function-level override global values.

**Advanced Rules for Parameter Specification**

Rules use substitution to specify how parameter values vary across subjects. Defining a rule involves (1) parameter selection, (2) rule specification, and (3) rule application. Rules are often defined for parameters indicating things like file names and directories but can be defined for any parameter. Substitutions occur when a rule is applied to set parameter values and can involve the substitution of constants and parameter values.

* **Parameter selection, rule specification & application**

Parameter selection and rule specification occur in the Rule Definition window. Rules are applied using the Batch Manager. The Rule Definition window is opened by selecting the first item in the "Define / Select Rule" drop-down menu of the Batch Manager.

* + Parameter selection. The parameter targeted by a rule must be selected first. There are two drop-down menus in the Rule Definition window to help the user indicate the parameter that the rule is for, one lists functions and the other lists parameters. The parameter menu lists the parameters available for the presently selected function. A rule will be defined for whatever function and parameter are selected in the function and parameter menus when the user clicks "Add."
  + Rule specification. The rule is a character string typed into the rule specification text box. It will be added to the study when the user clicks "Add" in the Rule Definition window. When a rule is added, it will be added to the list box below the rule specification text box and the rule selection menu at the bottom of the Batch Manager.
  + Rule application. When a rule is selected from the rule selection menu of the Batch Manager, it is immediately applied to all occurrences of the target parameter in all selected batches for all selected subjects. Note: a rule will be applied only to the subjects and batches that are highlighted in the Batch Manager when it is selected in the Batch Manager's rule menu.
* **Constants, Subject IDs, and Parameters**
  + Subject IDs are the strings listed in the Subject list of the Batch Manager. The string "SUBJECTID" can be used to substitute the Subject ID. Clicking on the Subject button in the Rule Definition window simply inserts "SUBJECTID" in the rule specification text box. There is no difference between using the button and manually typing "SUBJECTID." Substitution of Subject IDs is frequently the most useful kind of substitution.
  + Constants can be used to simplify rule definition for parameters with character values. A constant must have a character value and is referenced using a user-specified character label. The constant value will be substituted for each occurrence of the label when a rule is applied.
    - To create a constant: (1) Select the "Const" item in the "Const" drop-down menu, and (2) Input a label and value in the dialog box that pops up. Note: the new label will be added to the *Const* menu.
    - To use a constant: (1) Selecting the constant's label from the *Const* menu will cause its value to be immediately inserted into the rule specification text box, or (2) Including the constant's label in the rule specification will cause its value to be substituted when the rule is applied.
    - Example: let "datafile" be a list of raw data files, file1 & file2, that are both contained in the same directory "somedirectory." Define a constant labeled "PATH" with the value "/home/john/somedirectory." Add the following rule: {'PATH/file1','PATH/file2'}. Apply it by selecting the appropriate subject(s) and batch(es) and then selecting the new rule from the rule drop-down menu in the Batch Manager.
  + Parameters can be referenced in rules and indicate that the value of the parameter for each subject should be substituted when the rule is applied. This can be used to set any parameter value that is dependent on other parameters.
    - Parameters are referenced within a rule using the following syntax: {{functionX|parameterY}}. That string will be replaced by the value of parameterY of functionX independently for each selected subject when the rule is applied.
    - Parameters can be added to the Parameter drop-down menu in the Rule Definition window by selecting the first item in the Parameter drop-down menu. The user will be prompted for a function, parameter, and optional label. If the user assigns a label to the parameter, then the label can be used directly in the rule to reference the parameter.
    - Either the {{function|parameter}} syntax or a user-assigned label can be used in a rule to indicate a particular parameter value should be substituted.
* **Modifying and Removing**
  + - When a rule is selected from the list box in the Rule Definition window, the rule will appear in the rule text box and the corresponding function and parameter will be selected in parameter selection menus. This is very useful for creating a modified version of an existing parameter. Simply, select a rule, modify it, and add the new rule.
    - A rule can be removed by selecting it in the list box and typing the character "d" for delete.
* **Tips**
  + Use rules to establish directory and file naming conventions.
  + Use rules to specify parameter values that are dependent on other parameters.

**Quick Start Guide**

Prerequisite: the TimeSurfer package must be present in your Matlab path.

(See [Accessing the latest TimeSurfer release](#AccessingTS))

1. Determine the functions you need to perform your analysis (see the [Function Reference](#FunctionReference)).

Tip: look for [Tutorials](#Examples), [Examples](#Examples), and [Templates](#Templates) that demonstrate similar analyses.

1. Open Matlab and run the TimeSurfer graphical interface (type "timesurfer" in the Command Window).
2. Select functions, set parameters, select data, and run analysis.

If you have a [STUDY file](#WhatIsTheSTUDYfile) (template or example):

* 1. Load the file (File => Load Study).
  2. [Modify parameters](#ModifyingParameters).
  3. Load data (File => Load data) or specify the [datafile](#Datafiles) parameter.
  4. Execute the analysis by clicking "[Run selected functions](#Figure2)" in the main figure.

If you are starting from scratch:

1. Select functions using the [Function drop-down menu](#FunctionDropDownMenu).
2. Specify parameter values in the [right parameter figure](#RightParameterFigure).
3. Load data (File => Load data) or specify the [datafile](#Datafiles) parameter.
4. Execute the analysis. If you loaded data: click "Run selected functions." Otherwise, click "Add Function" and then "Run selected functions."
5. View results: load (File => Load data) and plot results (Tools => Plot => ezplot).

To quickly look at data or results: load it (File => Load data) and use the EZ plotting tools (Tools => Plot).

Recommendations:

* Create all subject directories in the same root container.
* Define "root" and "task" as constants in the Rule Definition window and use rules in template.
* Set subject IDs to match the names of subject directories in the Batch Manager's Subject list.

**TimeSurfer Resources**

Task-independent templates with predefined rules and batches with a "template" subject have been prepared to help users set up common analysis protocols for event-related designs. All templates and examples are accompanied by text files describing their use. Step-by-step tutorials have been prepared to illustrate how to use templates in different situations. A wiki exists to share information on neuroimaging studies and techniques for our collaborators. A Google Group exists for users of the TimeSurfer package to report bugs, request changes, and ask questions.

The simplest way to learn which functions are needed for a particular analysis is to work through the tutorials. The tutorials were created to demonstrate the most useful features of the program and to introduce the user to the most important TimeSurfer functions for standard analysis protocols. Additional examples demonstrate the set up for other common analysis protocols.

Tutorials, templates, and examples are packaged with the TimeSurfer software.

**Tutorials.**

Most tutorials include both a \*.mat file and a \*.docx file. The \*.mat files contain the templates with which the tutorials work. The \*.docx files contain step-by-step illustrations of the procedure.

* Tutorial1\_InteractiveERP.docx (.mat)
* Tutorial2\_InteractivePLV.docx (.mat)
* Tutorial3\_TemplatePLV.docx (.mat)
* Tutorial4\_VisualRejection.docx
* Tutorial5\_TemplateGamma\_complete\_ieeg.docx (.mat)

**Templates for standard analyses** (in addition to those listed with the tutorials).

* Gamma\_analysis\_iEEG\_template.mat
* Gamma\_analysis\_MEG\_template.mat

**PB Wiki** (MGH, NYU, UCSD)

<http://neuroimaging.pbworks.com/TS-Home>

Email Dr. Thomas Thesen (thomas.thesen@med.nyu.edu) to request access.

**Google Group** (report bugs, ask questions, make requests, etc)

<http://groups.google.com/group/neuroimaging-data-analysis?hl=en>

Open membership.

**SourceForge Project** (open source content - inactive since May 2009)

<http://mmil-ts.sourceforge.net/>

**External Links**

# Collaborating institutes

* + [Multimodal Imaging Laboratory](http://mmil.ucsd.edu/), University of California, San Diego, USA

(http://mmil.ucsd.edu/)

* + [Cognitive Neurophysiology Research Group](http://mmil.ucsd.edu/thomas/group/index.html), New York University School of Medicine, USA

(http://mmil.ucsd.edu/thomas/group/index.html)

* + [Center for Neural Science](http://www.cns.nyu.edu/), New York University, USA

(http://www.cns.nyu.edu/)

* + [Department of Neurology](http://www.massgeneral.org/neurology/), Massachusetts General Hospital, USA

(http://www.massgeneral.org/neurology/)

# Associated wikis

* + [UCSD-NYU-MGH Neuroimaging Wiki](http://neuroimaging.pbworks.com/../../../) (http://neuroimaging.pbworks.com/)
  + [UCSD-NYU-MGH Intracranial EEG Wiki](http://ieeg.pbwiki.com/) (http://ieeg.pbwiki.com/)

# EEG/MEG software

* + [FieldTrip](http://neuroimaging.ruhosting.nl/fieldtrip): (http://neuroimaging.ruhosting.nl/fieldtrip)
  + [EEGLAB](http://www.sccn.ucsd.edu/eeglab) (http://www.sccn.ucsd.edu/eeglab)

**Appendix A - Details for the Multimodal Imaging Laboratory**

**Accessing the latest TimeSurfer release**

1. On the MMIL network: /home/halgsvn/matlab
2. Obtain a copy of the software by typing this at the linux shell:

svn co svn+ssh://[username@ip84.ucsd.edu/var/svn/repos/mmil](http://username@ip84.ucsd.edu/var/svn/repos/mmil)

Note: this will create a directory called "mmil" that contains the software package.

1. Request a copy (visit the Neuroimaging Data Analysis Google Group page for contact information).

The software package must be added to your Matlab path before it can be used. There are two ways to do that:

1. In Matlab: File => Set path. Then, add the TimeSurfer directory with subdirectories and click save.
2. Add to /home/user/matlab/startup.m: addpath(genpath(TSPATH));

where TSPATH = "/home/halgsvn/matlab" on the MMIL network or "/yourdirectory/mmil" if you used svn to obtain a copy of the software package.

Templates, examples, and tutorials are located in the **doc** directory.

Templates for MMIL standard analysis protocols are located in **doc/mmil**.

**Getting started**

1. Access TS package and add to Matlab path.
2. Determine what is needed for your analysis protocol & find the appropriate TS functions.
3. Carry out the analysis at the command line, using Matlab scripts, or the TS GUI.

Learn from the examples and tutorials. Use templates when possible. Consult the wiki and Google Group discussion board for additional information. Use the Google Group to communicate with other users and developers.

**Using MMIL clusters**

* Clusters: mmilcluster (Eric Halgren); mmilcluster2 & mmilcluster3 (Anders Dale)
* Batch processing: save m-files to directory X, /home/user/batchdirs/X, along with a \*.txt file called "scriptlist.txt" listing the names of the m-files without the ".m" extension.
* Job submission: log onto one of the clusters and type: qmatjobs X where X is the name of the directory in /home/user/batchdirs that contains the Matlab scripts to be executed. qmatjobs is a linux shell script that pre-allocates 2GB of memory for each job. Other versions of qmatjobs allocate different amounts of memory for each job. Specifically, qmatjobs2, qmatjobs4, qmatjobs8, and qmatjobs\_unlimited allocate 4GB, 8GB, 16GB, and no limit, respectively.
* Recommendations - use the version of qmatjobs that allocates the smallest amount of memory that can handle a single job. If each job has less memory allocated to it, then more jobs can run simultaneously on the same cluster node.

**Using MMIL storage**

* Policy - data belonging Eric Halgren and collaborators are stored on the MMIL network on RAIDs linked to /home/halgdev. The data are organized into several directories: incoming (for recently acquired or uploaded data), data (for longer-term storage of raw data), analysis (for results of single-subject analyses), projects (for group analyses and project-specific results). Ander's Data ...
* Recommendations - consult the MMIL system administrator before storing or accessing data on the MMIL network for the first time. Storage conventions and policy are likely to change.

**Appendix B - Function Reference**

Functions in the [function drop-down menu](#FunctionDropDownMenu):

1. [ts\_process\_ieeg\_data](#ts_process_ieeg_data) description
2. [ts\_process\_fif\_data](#ts_process_fif_data) description
3. [ts\_statistics\_wrapper](#ts_statistics_wrapper) etc.
4. [ts\_freqanalysis\_fieldtrip](#ts_freqanalysis_fieldtrip)
5. [ts\_freqband\_average](#ts_freqband_average)
6. [ts\_timefreq\_rejection](#ts_timefreq_rejection)
7. [ts\_timefreq\_baseline](#ts_timefreq_baseline)
8. [ts\_PLV](#ts_PLV)
9. [ts\_dSPM](#ts_dSPM)
10. [ts\_tf\_dSPM](#ts_tf_dSPM)
11. [ts\_preproc](#ts_preproc)
12. [ts\_reject](#ts_reject)
13. [ts\_data\_selection](#ts_data_selection)
14. [ts\_trials2avg](#ts_trials2avg)
15. [ts\_combine\_conditions](#ts_combine_conditions)
16. [ts\_autoICA](#ts_autoICA)
17. [ts\_manualICA](#ts_manualICA)
18. [ts\_ezplot](#ts_ezplot)
19. [ts\_syncplot](#ts_syncplot)

Note: add purposes to this list

Undocumented functions: ...

List of the most common parameters ([see details at end of Function Reference](#StandardParamsReference)):

bpfilter, bpfreq, ...

Tip: when common parameters are constant within batches, make them global (Edit => Add parameter) and remove them from individual functions (clear the field or remove the parameter altogether: Edit => Remove parameter).

List of [Generic Parameters](#GenericParameters) (can be specified for any function in TS GUI):

function\_id, input\_id, cluster\_flag, save\_flag, verbose, ...

Tip: ...

**[global](#ParameterTypes)** - global parameters will be passed to all functions.

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | N/A |
|  | | |
| **Inputs:** | | |
|  | *rootindir* | directory containing all input files. See Generic Parameters section above. |
|  | *rootoutdir* | All output files will be saved in the *rootoutdir* directory (or subdirectory). See Generic Parameters section above. |
|  | *prefix* | [proc] specifies a prefix to use for all output file names. See Generic Parameters section above. |
|  | *inpath* | [matfiles] an optional parameter used to name subdirectories for retrieving input files. See Generic Parameters section above. |
|  | *outpath* | [matfiles] an optional parameter used to name subdirectories for saving output files. See Generic Parameters section above |
|  | *overwrite* | Whether to overwrite files |
|  | *cluster* | [mmilcluster]Any cluster jobs will be submitted to this cluster (string) |
|  | *clusterscript* | [qmatjobs2]Shell script used to submit jobs to the cluster |
|  | *clusterheader* |  |
|  | *logfile* | [] filename for log file (stdout will be appended) |
|  | *logfid* | [1] file FID for log file |
|  | *badchanfile* | txt file listing channel labels to skip |
| **Note:** | "global" parameters can be defined using TS GUI. | |
| **Author:** |  | N/A |
| **See also:** |  |  |

**ts\_process\_ieeg\_data** - import, epoch, and preprocess intracranial EEG data; combine conditions, and save results to MAT files.

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | [ filenames, alldata ] = ts\_process\_ieeg\_data( datafile, 'key1', val1, 'key2', val2, ...); |
|  |  |  |
| **Inputs:** | | |
|  | *datafile* | **[\*.eeg, \*.cnt, \*.mat]** File(s) containing continuous or epoched data. List multiple files in cell array. Ex. {'file1.eeg','file2.eeg'}. |
| **Optional inputs:** | | |
|  | *filename* | **[acceptable]** Description... Example... Notes/Comments |
|  | *channamefile* | **[acceptable]** Description... Example... Notes/Comments |
|  | *badchanfile* | txt file listing channel labels to skip |
|  | *rejectfile* | MAT file containing reject\_data |
|  | *channels* | vector of channel indices - overrides chantype |
|  | *chantype* | options: {all mag grad1 grad2 eeg other grad meg} |
|  | *events* | event codes of conditions to process |
|  | *saveepochs\_flag* | whether to save epochs |
|  | *saveepochs\_raw\_flag* | save epochs after loading \*.eeg file |
|  | *saveepochs\_post\_flag* | save epochs after the preprocessing following artifact rejection |
|  | *saveaverages\_flag* | save averages after everything. |
|  | *oldeventcodes* | Array of event codes to change. For multiple files - specify old event codes in one numeric array per file and place all arrays in a cell. Ex. Files 1 & 2 both have event codes 1-3: {[1 2 3] [1 2 3]} |
|  | *neweventcodes* | Array of event codes to replace the old event codes. See help for oldeventcodes for handling multiple files. |
|  | *stim\_delay* | time vector offset |
|  | *timelimits* | epoch time limits [begin end] in seconds |
|  | *visualreject* | [0,1] whether to do visual rejection using FieldTrip rejectvisual |
|  | *reject\_method* | [summary, channel, trial]  **summary** show a single number for each channel and trial (default)  **channel** show the data per channel, all trials at once  **trial** show the data per trial, all channels at once |
|  | *reject\_metric* | [var, min, max, maxabs, range, kurtosis]  **var** variance within each channel (default)  **min** minimum value in each channel  **max** maximum value in each channel  **maxabs** maximum absolute value in each channel  **range** range from min to max in each channel  **kurtosis** kurtosis, i.e. measure of peakedness of the amplitude |
|  | *noise\_start* |  |
|  | *noise\_end* |  |
|  | *recode\_rules* |  |
|  | *epoch\_combinations* | combinations for epoch\_data (see ts\_combine\_conditions) |
|  | *epoch\_neweventcodes* | neweventcodes for epoch\_data (see ts\_combine\_conditions) |
|  | *average\_combinations* | combinations for avg\_data (see ts\_combine\_conditions) |
|  | *average\_neweventcodes* | neweventcodes for avg\_data (see ts\_combine\_conditions) |
|  | *combo\_calc* | [weighted] Type: help ts\_combine\_conditions (options: ''weighted'' ''avg'' or ''sum'') |
|  | *coordinatefile* |  |
|  | *coordfield* |  |
|  | ts\_reject() options |  |
|  | ts\_preproc() options |  |
| **Outputs:** |  |  |
|  | *filenames* | cell array of files saved by ts\_process\_ieeg\_data |
|  | *alldata* | structure containing epoch\_data or avg\_data |
| **Note:** |  |  |
| **Author:** |  | Jason Sherfey |
| **See also:** |  | ts\_process\_fif\_data, ts\_iEEG\_eeg2epoch, ts\_loadcnt, ts\_loadedf |

**ts\_process\_fif\_data** - process neuromag MEG/EEG data

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | ts\_process\_fif\_data(datafile,'key1',val1,'key2',val2,...); |
|  |  | e.g. ts\_process\_fif\_data('/usr/data/test.fif','valid\_event\_codes',[1:6]); |
| **Inputs:** | | |
|  | *datafile* | full or relative path name of Neuromag fif raw data file |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *rejectfile* |  |
|  | *layoutfile* |  |
|  | *baselinefile* |  |
|  | *prefix* | [proc] |
|  | *browseraw* | [0] |
|  | *saveepochs\_flag* | [0] |
|  | *rootoutdir* | [pwd] |
|  | *write\_fif\_flag* | [0,1] |
|  | *fifname\_evcode\_flag* | [0,1] |
|  | *events\_fnames* |  |
|  | *trigchan* | [STI101] |
|  | *event\_recode\_rules* |  |
|  | *valid\_event\_codes* |  |
|  | *code\_excl* |  |
|  | *time\_excl\_pre* |  |
|  | *time\_excl\_post* |  |
|  | *trig\_minduration* |  |
|  | *max\_num\_trials* |  |
|  | *stim\_delay* |  |
|  | *prestim\_dur* |  |
|  | *poststim\_dur* |  |
|  | *reject\_mag* |  |
|  | *reject\_grad* |  |
|  | *reject\_eeg* |  |
|  | *reject\_eog* |  |
|  | *bandpass\_flag* |  |
|  | *bandpass\_low\_cf* |  |
|  | *bandpass\_low\_tb* |  |
|  | *bandpass\_high\_cf* |  |
|  | *bandpass\_high\_tb* |  |
|  | *dsfact* |  |
|  | *detrend\_flag* |  |
|  | *baseline\_flag* |  |
|  | *baseline\_start* |  |
|  | *baseline\_end* |  |
|  | *noise\_start* |  |
|  | *noise\_end* |  |
|  | *ncov\_ex\_evnts* |  |
|  | *badchans* |  |
|  | *badchanfile* |  |
|  | *readtrans\_flag* |  |
|  | *post\_subnull\_flag* |  |
|  | *null\_event* |  |
|  | *post\_bandpass\_flag* |  |
|  | *post\_dsfact* |  |
|  | *post\_detrend\_flag* |  |
|  | *post\_baseline\_flag* |  |
|  | *post\_stim\_delay* |  |
|  | *post\_badchans* |  |
|  | *post\_badchanfile* |  |
|  | *post\_rm\_badchans\_flag* |  |
|  | *ICA\_auto\_flag* |  |
|  | *ICA\_manual\_flag* |  |
|  | *ICA\_ref\_chan* |  |
|  | *ICA\_chantype* |  |
|  | *ICA\_maxsteps* |  |
|  | *ICA\_ntrials* |  |
|  | *ICA\_ncomponents* |  |
|  | *ICA\_rescale\_flag* |  |
|  | *ICA\_sorttrials* |  |
|  | *forceflag* |  |
|  | *loop\_param* |  |
|  | *saveperevent* |  |
| **Outputs:** |  |  |
|  |  |  |
| **Note:** |  | This program is designed so that the output of each processing step is stored as a mat file (in the matfiles dir that will be automatically created |
| **Author:** |  | Don Hagler, 04/26/06 |
| **See also:** |  | ts\_avg\_fif\_data, ts\_process\_ieeg\_data |

**ts\_statistics\_wrapper** - perform statistical analysis using FieldTrip functions, timelockstatistics and freqstatistics

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | stat\_data = ts\_statistics( data, 'key1', val1, 'key2', val2, ...); |
|  |  |  |
| **Inputs:** | | |
|  | *data* | **[epoch\_data or avg\_data]** structure must contain the conditions to compare |
|  | *events* | **[n1 n2]** to compare event codes n1 and n2. {[n1 n2] [n1 n3] ...} |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *cluster\_flag* | Whether to submit job to cluster |
|  | *datafile* |  |
|  | *datapath* |  |
|  | *datastring* |  |
|  | *dataskip* |  |
|  | *inpath* | [matfiles] subdirectory within rootindir to search for required data files by default |
|  | *savepath* |  |
|  | *outpath* | [matfiles/stats] subdirectory within rootoutdir to save results by default |
|  | *outfile* |  |
|  | *rejectfile* |  |
|  | *baselinefile* |  |
|  | *prefix* |  |
|  | *numrandomization* | **[500]** the number of monte carlo iterations (default = 500) |
|  | *conditions* |  |
|  | *channel* | vector of channel indices - overrides chantype |
|  | *channels* | vector of channel indices - overrides chantype |
|  | *chantype* | options: {'all' 'mag' 'grad1' 'grad2' 'eeg' 'other' 'grad' 'meg'} |
|  | *latency* | **[begin end]** or ''all'' - times to be analyzed |
|  | *method* | **[montecarlo=default**] statistical method for comparing conditions (options: {''montecarlo'' ''analytic'' ''stats'' ''glm'') |
|  | *statistic* | sample level statistic for thresholding prior to clustering (options: {''indepsamplesT'' ''indepsamplesF'' ''indepsamplesregrT'' ''indepsamplesZcoh'' ''depsamplesT'' ''depsamplesF'' ''depsamplesregrT'' ''actvsblT'' ''ttest'' ''ttest2'' ''paired-ttest'' ''anova1'' ''kruskalwallis'') |
|  | *correctm* | options: {''no'' ''max'' ''cluster'' ''bonferoni'' ''fdr''} |
|  | *alpha* | alpha level of statistical test |
|  | *tail* | tail of statistical test |
|  | *ivar* |  |
|  | *uvar* |  |
|  | *wvar* |  |
|  | *feedback* | options: {''gui'' ''text'' ''textbar'' ''no''} |
|  | *clusterstatistic* | cluster level statistic (corrects for multiple comparisons)(options: {''maxsum'' ''maxsize'' ''wcm'') |
|  | *clusterthreshold* | options: {''parametric'' ''nonparametric''} |
|  | *clusteralpha* | alpha level of sample-level test |
|  | *clustercrtival* |  |
|  | *clustertail* | **[0]** tail of sample-level test |
|  | *avgoverchan* | **[no,yes]** whether to average over channels |
|  | *avgovertime* | [**no,yes]** whether to average over time |
|  | *minnbchan* | minimum number of channels to define a spatial cluster |
|  | *neighbours* | neighbourhoods |
|  | *design* | design matrix (see FieldTrip documentation for timelockstatistics) |
|  | *lpfilter* | lowpass filter |
|  | *lpfreq* | lowpass frequency in Hz |
|  | *lpfiltord* | lowpass filter order |
|  | *lpfilttype* | digital filter type - "but" (default) or "fir" |
|  | *lpfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *hpfilter* | highpass filter |
|  | *hpfreq* | highpass frequency in Hz |
|  | *hpfiltord* | highpass filter order |
|  | *hpfilttype* | digital filter type - "but" (default) or "fir" |
|  | *hpfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *bpfilter* | bandpass filter |
|  | *bpfreq* | bandpass frequency range - specified as [low high] in Hz |
|  | *bpfiltord* | bandpass filter order |
|  | *bpfilttype* | digital filter type - "but" (default) or "fir" |
|  | *bpfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *bsfilter* | bandstop filter |
|  | *bsfreq* | bandstop frequency range - specified as [low high] in Hz |
|  | *bsfiltord* | bandstop filter order |
|  | *bsfilttype* | digital filter type - "but" (default) or "fir" |
|  | *bsfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *lnfilter* | line noise removal using notch filter |
|  | *lnfreq* | line noise frequency in Hz - default 50Hz |
|  | *lnfiltord* | jump preserving median filter |
|  | *dftfilter* | line noise removal using discrete fourier transform |
|  | *dftfreq* | line noise frequencies for DFT filter - default [60 120 180] Hz |
|  | *medianfilter* | jump preserving median filter |
|  | *medianfiltord* | length of median filter |
|  | *blc* | yes/no - baseline correction |
|  | *blcwindow* | [begin end] in seconds |
|  | *detrend* | yes/no - linear detrending |
|  | *polyremoval* | yes/no - polynomial detrending |
|  | *loop\_param* |  |
| **Outputs:** |  |  |
|  | *stat\_data* | % stat\_data - a statistics data structure containing the  % following fields:  % num\_sensors - number of channels  % sensor\_info - information on the channels  % sfreq - sampling frequency  % stats - a structure array with following s  % condition - the condition number in relation to the original data (-1 for between conditions)  % event\_code - the event code (-1 for between conditions)  % prob - the result of the test p-value - a matrix [channels x time points]  % mask - mask (0 for bad channels)  % time - vector of time points  % parms - method - the trip method used  % statistic - the trip statistic used  % alpha - alpha value used  % num\_trials - number of trials  % norm\_bl - result of Kolmogorov-Smirnov test for baseline data (0 for between conditions)  % posclusters - a structure array with statistical results for positive clusters (sorted by p-value)  % - prob - p-values  % - clusterstat - cluster-level statistics  % negclusters - a structure array with statistical results for negative clusters (sorted by p-value)  % - prob - p-values  % - clusterstat - cluster-level statistics  % posclusterslabelmat - matrix [channels x time points] indicating pos clusters to which (channel,time)-pairs belong  % negclusterslabelmat - matrix [channels x time points] indicating neg clusters to which (channel,time)-pairs belong |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Rajan Patel, Jason Sherfey |
| **See also:** |  | ts\_statistics, timelockstatistics, freqstatistics |

**ts\_freqanalysis\_fieldtrip** - perform time-frequency analysis using the FieldTrip function freqanalysis

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | timefreq\_data = ts\_freqanalysis\_fieldtrip( epoch\_data, 'key1', val1, 'key2', val2, ...); |
|  |  |  |
| **Inputs:** | | |
|  | *epoch\_data* | a valid TimeSurfer epoch data set |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *cluster\_flag* | Whether to submit job to cluster |
|  | *save\_flag* |  |
|  | *datafile* |  |
|  | *datapath* |  |
|  | *datastring* |  |
|  | *dataskip* |  |
|  | *inpath* | subdirectory within rootindir to search for required data files by default |
|  | *savepath* |  |
|  | *outpath* | subdirectory within rootoutdir to save results by default |
|  | *outfile* |  |
|  | *rejectfile* |  |
|  | *baselinefile* |  |
|  | *prefix* |  |
|  | *loop\_param* |  |
|  | *events* | Array of event codes to analyze. Codes can be listed two ways. Ex) [1 2 3] or {[1] [2] [3]}. The difference is that all three conditions are loaded into memory at the same time using the first notation whereas they are loaded and cleared from memory separately using the latter notation. Furthermore, when submitting jobs to the cluster - the former will submit all three as one job whereas the latter will submit three jobs (one per condition) that can be processed in parallel. |
|  | *channels* |  |
|  | *foi* | Frequencies of interest (Hz) |
|  | *toi* |  |
|  | *toilim* |  |
|  | *sf* | Spectral resolution in standard deviations of the gaussian. Either give one value per frequency or a single value to have a constant frequency resolution. Since time and frequency resolution scale inversely, constant frequency resolution implies constant resolution in time. As sf decreases the frequency resolution increases and time resolution decreases. The opposite is true of increasing sf. |
|  | *st* |  |
|  | *chantype* |  |
|  | *output* |  |
|  | *method* | method for time-frequency analysis (default = wltconvol - complex morlet wavelet analysis) |
|  | *savecomplex\_flag* | whether to save complex spectra when saving trials (always saved for averages) |
|  | *savesingles\_flag* | whether to save data with single precision instead of double |
|  | *verbose* |  |
|  | *visualreject\_flag* |  |
|  | *trials\_flag* | whether to save trials in addition to averages |
|  | *detrend* |  |
|  | *lnfilter* |  |
|  | *lnfreq* |  |
|  | *lpfilter* |  |
|  | *lpfreq* |  |
|  | *hpfilter* |  |
|  | *hpfreq* |  |
|  | *blc* |  |
|  | *blcwindow* |  |
| **Outputs:** |  |  |
|  | *timefreq\_data* |  |
|  |  |  |
| **Note:** |  | Look at freqanalysis documentation for information about method and related parameters |
| **Author:** |  | Rajan Patel, Jason Sherfey |
| **See also:** |  | freqanalysis, ts\_freqband\_average, ts\_sync (formerly ts\_PLV) |

**ts\_freqband\_average** - calculate average time-frequency data over a frequency band

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | epoch\_data = ts\_freqband\_average( timefreq\_data, 'key1', val1, 'key2', val2, ...); |
|  |  |  |
| **Inputs:** | | |
|  | *timefreq\_data* | valid TimeSurfer structure produced by ts\_freqanalysis\_fieldtrip or a header with a list of files containing the time-frequency data. |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* |  |
|  | *cluster\_flag* | Whether to submit job to cluster |
|  | *datafile* |  |
|  | *inpath* | subdirectory within rootindir to search for required data files by default |
|  | *outpath* | subdirectory within rootoutdir to save results by default |
|  | *prefix* |  |
|  | *loop\_param* |  |
|  | *freqband* |  |
|  | *events* |  |
|  | *conditions* |  |
|  | *chantype* | {''mag'' ''grad'' ''grad1'' ''grad2'' ''eeg'' ''other'' ''meg'' ''all''} |
|  | *channel* |  |
|  | *findex* | index into the frequency vector indicating which frequencies to average |
|  | *blc* |  |
|  | *baselinetype* | {''zscore'' ''relative'' ''relchange'' ''absolute''} |
|  | *blcwindow* |  |
|  | *baseline\_data* |  |
|  | *baselinefile* |  |
|  | *verbose* |  |
|  | *freqcorr* | whether to correct for frequency drop-off in power |
|  | *toilim* | time limits [begin end] in seconds |
|  | *rejectfile* | MAT file containing reject\_data |
|  | *reject\_data* |  |
| **Outputs:** |  |  |
|  | *epoch\_data* | structure containing the power waveform |
|  |  |  |
| **Note:** |  | all time-domain functions that process epoch\_data can also process the epoch data produced by this function. The power waveform will be called a "TF wave." |
| **Author:** |  | Jason Sherfey |
| **See also:** |  | ts\_freqanalysis\_fieldtrip |

**ts\_timefreq\_rejection** - perform visual artifact rejection on TF waves using FieldTrip tools

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | [cleandata, badchans, badtrials] = ts\_timefreq\_rejection(data, 'key1',val1,...); |
|  |  |  |
| **Inputs:** | | |
|  | *datafile* | file containing epoch data |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *events* | event codes of conditions to process |
|  | *method* | [summary] |
|  | *metric* | [var] |
|  | *latency* | [begin end]: x-limits that are plotted |
|  | *badchanfile* |  |
|  | *badchans* |  |
|  | *rejectfile* |  |
|  | *toilim* | [begin end]: data that is processed |
|  | *fpath* | path for saving reject\_data |
|  | *fname* | mat-file name for reject\_data |
| **Outputs inputs:** | | |
|  | *cleandata* | same as input minus rejected trials and channels |
|  | *badchans* | vector of channel indices for rejected channels |
|  | *badtrials* | vector of channel indices for rejected trials |
| **Note:** |  |  |
| **Author:** |  |  |
| **See also:** |  | Jason Sherfey |

**ts\_timefreq\_baseline** - calculate a common baseline for normalization/scaling of power data

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | baseline\_data = ts\_timefreq\_baseline( datafile, 'key1',val1,...); |
|  |  |  |
| **Inputs:** | | |
|  | *datafile* |  |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *events* | event codes of conditions to process |
|  | *blcwindow* | [-inf 0] baseline window |
|  | *toilim* | time limits [begin end] in seconds |
|  | *foi* | Frequencies of interest (Hz) |
|  | *rejectfile* | MAT file containing reject\_data |
| **Outputs:** |  |  |
|  | *Output1* | Description... Notes/Comments |
|  | *Output2* | Description... Notes/Comments |
| **Note:** |  |  |
| **Author:** |  | Name, Institution, date |
| **See also:** |  | List of related functions |

**ts\_sync** (formerly ts\_PLV in TS) - calculate metrics from time-frequency representations or spectral estimation

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | sync\_data = ts\_sync(timefreq\_data, 'key1',val1,...); |
|  |  |  |
| **Inputs:** | | |
|  | *timefreq\_data* | for nonstationary metrics |
|  | *epoch\_data* | for calculations based on spectral estimation assuming stationarity |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *cluster\_flag* | Whether to submit job to cluster |
|  | *inpath* | subdirectory within rootindir to search for required data files by default |
|  | *outpath* | subdirectory within rootoutdir to save results by default |
|  | *loop\_param* |  |
|  | *save\_flag* |  |
|  | *foi* | vector of frequencies of interest |
|  | *foilim* | frequency limits [low high] in Hz |
|  | *band* | whether to average the metric over frequencies before returning the result |
|  | *events* |  |
|  | *channelcmb* | pair-wise channel combinations for which to calculate synchrony metrics: {''chan1'' ''chan2''; ''chan1'' ''chan3''} |
|  | *verbose* |  |
|  | *output* | {''plv'' ''lag'' ''coh'' ''mscoh'' ''coherency''} |
|  | *load\_flag* | Whether TF trial data is saved in one file per channel per condition (1) or not (0); if 0 then it is assumed the data have been passed directly |
|  | *combine\_flag* | Whether to concatenate conditions or process them independently |
|  | *neweventcodes* | Like neweventcodes in ts\_combine\_conditions |
|  | *rejectfile* |  |
| **Outputs:** |  |  |
|  | *sync\_data* | Contains data matrices with field names based on the "output" parameter |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Jason Sherfey |
| **See also:** |  | ts\_freqanalysis\_fieldtrip, freqdescriptives |

**ts\_dSPM** - dynamic Statistical Parametric Mapping source analysis for MEG/EEG data

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | ts\_dSPM(avg\_data,subjname, 'key1', value1,...); |
|  |  | Format2 |
| **Inputs:** | | |
|  | *avg\_data* | average data structure (see ts\_process\_fif\_data and ts\_avg\_fif\_data) |
|  | *subjname* | freesurfer subject name (must be found in $SUBJECTS\_DIR) |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *loop\_param* | {'conditions' 'ncov\_conditions'} |
|  | *subjdir* | getenv(''SUBJECTS\_DIR'') |
|  | *rootoutdir* | pwd |
|  | *prefix* | dSPM |
|  | *conditions* | [] |
|  | *lh\_dip\_file* | bem/lh\_white.dip |
|  | *rh\_dip\_file* | bem/rh\_white.dip |
|  | *lh\_dip\_info* | [] |
|  | *rh\_dip\_info* | [] |
|  | *lh\_dec\_file* | bem/lh\_white\_7.dec |
|  | *rh\_dec\_file* | bem/rh\_white\_7.dec |
|  | *lh\_dec\_dips* | [] |
|  | *rh\_dec\_dips* | [] |
|  | *ncov\_type* | [0 2] |
|  | *identity\_ncov\_flag* | [false true] |
|  | *calc\_avg\_ncov\_flag* | [false true] |
|  | *ncov\_conditions* |  |
|  | *calc\_scalefacts\_flag* | [false true] |
|  | *noise\_start* | [-inf inf] |
|  | *baseline\_start* | [-inf inf] |
|  | *baseline\_end* | 0 |
|  | *baseline\_flag* | 1 |
|  | *ssp\_projmap* | [] |
|  | *SNR* | [eps inf] |
|  | *noisenorm\_flag* | [false true] |
|  | *depthweight\_flag* | [false true] |
|  | *depthweight\_p* | 0.5 |
|  | *bem\_flag* | [false true] |
|  | *radii* | [] |
|  | *conductivities* | [0.3 0.012 0.3] |
|  | *conduct\_scalefact* | [0 inf] |
|  | *nlayers* | [1 3] |
|  | *badchans* | [] |
|  | *badchanfile* | [] |
|  | *usegrad\_flag* | [false true] |
|  | *usemag\_flag* | [false true] |
|  | *useEEG\_flag* | [false true] |
|  | *grad\_scalefact* | [-inf inf] |
|  | *mag\_scalefact* | [-inf inf] |
|  | *EEG\_scalefact* | [-inf inf] |
|  | *overwrite\_output\_flag* | [false true] |
|  | *write\_stc\_flag* | [false true] |
|  | *stc\_scalefact* | [eps inf] |
|  | *write\_mgh\_flag* | [false true] |
|  | *sparsesmooth* | [0 1000] |
|  | *postsmooth* | [0 1000] |
|  | *mbmask\_flag* | [false true] |
|  | *resamp2ico\_flag* | [false true] |
|  | *icolevel* | [1 7] |
|  | *icosmooth* | [0 1000] |
|  | *write\_fif\_flag* | [false true] |
|  | *template\_fif* |  |
|  | *bem\_surf\_files* |  |
|  | *cen\_sph* |  |
|  | *trans* |  |
|  | *transfile* |  |
|  | *alignment\_fif* |  |
|  | *lh\_sourcecov\_wfile* |  |
|  | *rh\_sourcecov\_wfile* |  |
|  | *sourcecov\_thresh* | [-inf inf] |
|  | *sourcecov\_thresh\_abs\_flag* | [false true] |
|  | *sourcecov\_maxvar* | [0 1] |
|  | *sourcecov\_minvar* | [0 1] |
|  | *overwrite\_forward\_flag* | [false true] |
|  | *overwrite\_inverse\_flag* | [false true] |
|  | *refEEG\_coords* |  |
|  | *amd\_inverse\_flag* | [false true] |
|  | *orient\_constr\_flag* | [false true] |
|  | *orient\_tang* | [0 1] |
|  | *smooth\_constr\_flag* | [false true] |
|  | *smooth\_constr\_nsmooth* | [1 100] |
|  | *smooth\_constr\_subfact* | [.01 .1] |
|  | *signed\_sources\_flag* | [false true] |
| **Outputs:** |  |  |
|  | *Output1* | Description... Notes/Comments |
|  | *Output2* | Description... Notes/Comments |
| **Note:** |  |  |
| **Author:** |  | Name, Institution, date |
| **See also:** |  | List of related functions |

**ts\_tf\_dSPM** - time-frequency (spectral) dynamic Statistical Parametric Mapping source analysis for MEG/EEG data

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | Format1 |
|  |  | Format2 |
| **Inputs:** | | |
|  | *Input1* | **[acceptable]** Description... Example... Notes/Comments |
|  | *Input2* | **[acceptable]** Description... Example... Notes/Comments |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *loop\_param* | {'conditions' 'ncov\_conditions'} |
|  | *subjdir* | getenv(''SUBJECTS\_DIR'') |
|  | *rootoutdir* | pwd |
|  | *prefix* | dSPM |
|  | *conditions* | [] |
|  | *lh\_dip\_file* | bem/lh\_white.dip |
|  | *rh\_dip\_file* | bem/rh\_white.dip |
|  | *lh\_dip\_info* | [] |
|  | *rh\_dip\_info* | [] |
|  | *lh\_dec\_file* | bem/lh\_white\_7.dec |
|  | *rh\_dec\_file* | bem/rh\_white\_7.dec |
|  | *lh\_dec\_dips* | [] |
|  | *rh\_dec\_dips* | [] |
|  | *identity\_ncov\_flag* | 0 |
|  | *calc\_avg\_ncov\_flag* | 0 |
|  | *ncov\_conditions* | [] |
|  | *calc\_scalefacts\_flag* | 0 |
|  | *noise\_start* | [] |
|  | *baseline\_start* | #NAME? |
|  | *baseline\_end* | 0 |
|  | *baseline\_flag* | 1 |
|  | *ssp\_projmap* | [] |
|  | *SNR* | 10 |
|  | *noisenorm\_flag* | 1 |
|  | *depthweight\_flag* | 0 |
|  | *depthweight\_p* | 0.5 |
|  | *bem\_flag* | 1 |
|  | *radii* | [] |
|  | *conductivities* | [0.3 0.012 0.3] |
|  | *conduct\_scalefact* | 1 |
|  | *nlayers* | 3 |
|  | *badchans* | [] |
|  | *badchanfile* | [] |
|  | *usegrad\_flag* | 1 |
|  | *usemag\_flag* | 1 |
|  | *useEEG\_flag* | 1 |
|  | *grad\_scalefact* | 10^13 |
|  | *mag\_scalefact* | 10^15 |
|  | *EEG\_scalefact* | 10^6 |
|  | *overwrite\_output\_flag* | 1 |
|  | *write\_stc\_flag* | 1 |
|  | *stc\_scalefact* | 1 |
|  | *write\_mgh\_flag* | 0 |
|  | *write\_mat\_flag* | 1 |
|  | *sparsesmooth* | 10 |
|  | *postsmooth* | 10 |
|  | *mbmask\_flag* | 0 |
|  | *resamp2ico\_flag* | 0 |
|  | *icolevel* | 7 |
|  | *icosmooth* | 3 |
|  | *write\_fif\_flag* | 0 |
|  | *template\_fif* | [] |
|  | *bem\_surf\_files* | {'bem/inner\_skull4.tri' 'bem/outer\_skull4.tri' 'bem/outer\_scalp4.tri'} |
|  | *cen\_sph* | [] |
|  | *trans* | [] |
|  | *transfile* | [] |
|  | *alignment\_fif* | [] |
|  | *lh\_sourcecov\_wfile* | [] |
|  | *rh\_sourcecov\_wfile* | [] |
|  | *sourcecov\_thresh* | 0 |
|  | *sourcecov\_thresh\_abs\_flag* | 1 |
|  | *sourcecov\_maxvar* | 0.9 |
|  | *sourcecov\_minvar* | 0.09 |
|  | *overwrite\_forward\_flag* | 0 |
|  | *overwrite\_inverse\_flag* | 0 |
|  | *refEEG\_coords* | [] |
|  | *amd\_inverse\_flag* | 1 |
|  | *orient\_constr\_flag* | 0 |
|  | *orient\_tang* | 0 |
|  | *smooth\_constr\_flag* | 0 |
|  | *smooth\_constr\_nsmooth* | 10 |
|  | *smooth\_constr\_subfact* | 0.02 |
|  | *signed\_sources\_flag* | 1 |
|  | *tf\_flag* | 0 |
|  | *inv\_ncov* | [] |
|  | *inv\_type* | data |
|  | *tfvals* | [] |
|  | *plv\_flag* | 0 |
|  | *tf\_power\_flag* | 0 |
|  | *first\_trial* | [] |
|  | *err\_flag* | 0 |
|  | *h2write* | {'lh' 'rh'} |
|  | *cortex\_only* | 0 |
|  | *parc\_files* | [] |
|  | *verbose* | 0 |
| **Outputs:** | | |
|  | *Output1* | Description... Notes/Comments |
|  | *Output2* | Description... Notes/Comments |
| **Note:** |  | |
| **Author:** |  | Name, Institution, date |
| **See also:** |  | List of related functions |
|  |  |  |

**ts\_preproc** - preprocess time series using TimeSurfer implementation or FieldTrip's preprocessing function

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | data = ts\_preproc(data, 'key1',val1,...); |
|  |  |  |
| **Inputs:** | | |
|  | *Input1* | **[acceptable]** Description... Example... Notes/Comments |
|  | *Input2* | **[acceptable]** Description... Example... Notes/Comments |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *events* |  |
|  | *fieldtrip\_flag* | options: {0 or 1} |
|  | *bandpass\_flag* |  |
|  | *bandpass\_low\_cf* |  |
|  | *bandpass\_low\_tb* |  |
|  | *bandpass\_high\_cf* |  |
|  | *bandpass\_high\_tb* |  |
|  | *dsfact* | the factor used for downsampling the data |
|  | *detrend\_flag* |  |
|  | *baseline\_flag* |  |
|  | *baseline\_start* |  |
|  | *baseline\_end* |  |
|  | *keeptrials* |  |
|  | *combinations* |  |
|  | *comboeventcodes* |  |
|  | *calc* |  |
|  | *cfg* |  |
|  | *feedback* |  |
|  | *lpfilter* | lowpass filter |
|  | *lpfreq* | lowpass frequency in Hz |
|  | *lpfiltord* | lowpass filter order |
|  | *lpfilttype* | digital filter type - "but" (default) or "fir" |
|  | *lpfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *hpfilter* | highpass filter |
|  | *hpfreq* | highpass frequency in Hz |
|  | *hpfiltord* | highpass filter order |
|  | *hpfilttype* | digital filter type - "but" (default) or "fir" |
|  | *hpfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *bpfilter* | bandpass filter |
|  | *bpfreq* | bandpass frequency range - specified as [low high] in Hz |
|  | *bpfiltord* | bandpass filter order |
|  | *bpfilttype* | digital filter type - "but" (default) or "fir" |
|  | *bpfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *bsfilter* | bandstop filter |
|  | *bsfreq* | bandstop frequency range - specified as [low high] in Hz |
|  | *bsfiltord* | bandstop filter order |
|  | *bsfilttype* | digital filter type - "but" (default) or "fir" |
|  | *bsfiltdir* | filter direction - "twopass" (default) - "onepass" or "onepass-reverse" |
|  | *lnfilter* | line noise removal using notch filter |
|  | *lnfreq* | line noise frequency in Hz - default 50Hz |
|  | *lnfiltord* | jump preserving median filter |
|  | *dftfilter* | line noise removal using discrete fourier transform |
|  | *dftfreq* | line noise frequencies for DFT filter - default [60 120 180] Hz |
|  | *medianfilter* | jump preserving median filter |
|  | *medianfiltord* | length of median filter |
|  | *blc* | yes/no - baseline correction |
|  | *blcwindow* | [begin end] in seconds |
|  | *detrend* | yes/no - linear detrending |
|  | *polyremoval* | yes/no - polynomial detrending |
|  | *latency* | integer - order of polynomial when detrending |
|  | *polyorder* |  |
|  | *hilbert* |  |
|  | *rectify* |  |
|  | *precision* |  |
|  | *reref* |  |
|  | *refchannel* |  |
|  | *implicitref* |  |
|  | *montage* | options: {0,1} |
|  | *saveepochs\_flag* | options: {0,1} |
|  | *saveaverages\_flag* | options: {0,1} |
|  | *returnepochs\_flag* | options: {0,1} |
|  | *returnaverages\_flag* | integer - order of polynomial when detrending |
|  | *saveavgs* | options: {''yes'' ''no''} |
|  | *saveepochs* | options: {''yes'' ''no''} |
|  | *returnavgs* | options: {''yes'' ''no''} |
|  | *returnepochs* | options: {''yes'' ''no''} |
|  | *rootoutdir* |  |
|  | *prefix* |  |
|  | *filename* |  |
|  | *overwrite* | options: {0 or 1} |
|  | *verbose* | options: {0 or 1} |
| **Outputs:** |  |  |
|  | *epoch\_data* |  |
|  | *avg\_data* |  |
| **Note:** |  | Can also save the resulting data structures to disk |
| **Author:** |  | Jason Sherfey |
| **See also:** |  | ts\_avg\_fif\_data, ts\_data\_selection, ts\_reject |

**ts\_reject** - perform various kinds of artifact rejection

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | [data, reject\_data] = ts\_preproc(data, 'key1',val1,...); |
|  |  |  |
| **Inputs:** | | |
|  | *data* | **[avg\_data, epoch\_data, or timefreq\_data]** |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *reject\_mag* | [10000] automatic rejection threshold for magnetometer channels (fT) if 0 then rejection based on magnetometers is disabled \*\*\*\*\*\*\*\*\*\*\*\*\*THIS SAYS THE DEFAULT IS 6000 |
|  | *reject\_grad* | [6000] automatic rejection threshold for gradiometer channels (fT/cm) if 0 then rejection based on gradiometers is disabled\*\*\*\*\*\*\*\*\*\*\*\*\*\*THIS SAYS THE DEFAULT IS 3000 |
|  | *reject\_eeg* | automatic rejection threshold for eeg channels (uV) if 0 then rejection based on eeg is disabled |
|  | *reject\_eog* | [200] automatic rejection threshold for eog channel (uV) if 0 then rejection based on eog is disabled |
|  | *reject\_ieeg* |  |
|  | *ICA\_auto\_flag* | options: {0 or 1} |
|  | *ICA\_manual\_flag* | options: {0 or 1} |
|  | *ICA\_ref\_chan* | EOG061 |
|  | *ICA\_chantype* | all |
|  | *ICA\_maxsteps* | 20 |
|  | *ICA\_ntrial* | 5 |
|  | *ICA\_ncomponents* | 80 |
|  | *ICA\_rescale\_flag* | options: {0 or 1} |
|  | *ICA\_sorttrials* | options: {0 or 1} |
|  | *visualreject* | options: {0 or 1} |
|  | *reject\_method* | summary |
|  | *reject\_metric* | var |
| **Outputs:** |  |  |
|  | *data* | Description... Notes/Comments |
|  | *reject\_data* |  |
| **Note:** |  |  |
| **Author:** |  | Name, Institution, date |
| **See also:** |  | List of related functions |

**ts\_data\_selection** - select a subset of the data

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | data = ts\_data\_selection( data, 'key1', val1, ... ); |
|  |  |  |
| **Inputs:** | | |
|  | *data* | **[any TimeSurfer structure]** |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *condition* | condition number (not event code) used to index data inside structure |
|  | *events* | choose ranges: (conds;chans;trials;toi;foi), remove rejected: (reject\_data;tf\_reject\_data; sensor\_info.badchan); data: epoch\_data; avg\_data; timefreq\_data; stat\_data) |
|  | *channels* | vector of channel indices - overrides chantype |
|  | *chantype* | options: {'all' 'mag' 'grad1' 'grad2' 'eeg' 'other' 'grad' 'meg'} |
|  | *chanlabel* | cell array of strings listing channel labels |
|  | *badchans* | indices of bad channels |
|  | *trials* | vector of trials to return |
|  | *rejects* |  |
|  | *toilim* | limits on the times of interest, [begin end] in seconds |
|  | *toi* | times of interest (sec) |
|  | *foilim* | limits on the frequencies of interest, [low high] in Hz |
|  | *foi* | Frequencies of interest (Hz) |
|  | *badchanfile* | txt file listing channel labels to skip |
|  | *rejectfile* | MAT file containing reject\_data |
| **Outputs:** |  |  |
|  | *data* | same format as input data containing only the data selections |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Jason Sherfey |
| **See also:** |  |  |

**ts\_trials2avg** - average structures with 3D and 4D data matrices with respect to trials (epochs)

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | outdata = ts\_data\_selection( indata, 'key1', val1, ... ); |
|  |  | e.g. avg\_data = ts\_trials2avg(epoch\_data); |
| **Inputs:** | | |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *datafile* |  |
|  | *inpath* | [matfiles] subdirectory within rootindir to search for required data files by default |
|  | *outpath* | [matfiles] subdirectory within rootoutdir to save results by default |
| **Outputs:** |  |  |
|  |  |  |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Jason Sherfey |
| **See also:** |  |  |

**ts\_combine\_conditions** - create meta-conditions from epoch data, timefreq\_data, or avg\_data

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | outdata = ts\_combine\_conditions( indata, 'key1', val1, ... ); |
|  |  | e.g. ... |
| **Inputs:** | | |
|  | *indata* | **[avg\_data, epoch\_data, or timefreq\_data]** |
|  | *combinations* |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *datafile* |  |
|  | *inpath* | subdirectory within rootindir to search for required data files by default |
|  | *outpath* | subdirectory within rootoutdir to save results by default |
|  | *events* |  |
|  | *combinations* | Type: help ts\_combine\_conditions |
|  | *neweventcodes* | Type: help ts\_combine\_conditions |
|  | *calc* | Type: help ts\_combine\_conditions (options: ''weighted'' ''avg'' or ''sum'') |
|  | *logfile* |  |
|  | *logfid* |  |
|  | *reference* | options: ''events'' or ''conditions'' |
|  | *verbose* | options: {0 or 1} |
| **Outputs:** |  |  |
|  | *outdata* | indata plus the new meta-conditions |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Jason Sherfey |
| **See also:** |  |  |

**ts\_autoICA** - automatically remove artifactual independent components

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | epoch\_data = ts\_combine\_conditions(epoch\_data, 'key1', val1, ... ); |
|  |  |  |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *datafile* |  |
|  | *inpath* | subdirectory within rootindir to search for required data files by default |
|  | *outpath* | subdirectory within rootoutdir to save results by default |
|  | *ICA\_ref\_chan* | EOG61 |
|  | *event\_codes* | [] |
|  | *conditions* | all |
|  | *chantype* | {''mag'' ''grad'' ''grad1'' ''grad2'' ''eeg'' ''other'' ''meg'' ''all''} |
|  | *channels* | [] |
|  | *notch* | options: {0 or 1} |
|  | *showcomponents* | options: {0 or 1} |
|  | *rescale* | options: {0 or 1} |
|  | *verbose* | options: {0 or 1} |
|  | *logfile* | [] |
|  | *logfid* | [] |
| **Outputs:** |  |  |
|  | *epoch\_data* | epoch\_data - epoch data without the selected components |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Andrei Irimia |
| **See also:** |  | ts\_manualICA |

**ts\_manualICA** - manually identify and remove artifactual independent components

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | epoch\_data = ts\_manualICA(epoch\_data, 'key1', val1, ... ); |
|  |  |  |
| **Inputs:** | | |
|  | *epoch\_data* |  |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *save\_flag* | whether to save the result or pass it to the next function |
|  | *datafile* |  |
|  | *inpath* | subdirectory within rootindir to search for required data files by default |
|  | *outpath* | subdirectory within rootoutdir to save results by default |
|  | *ntrial* | [] |
|  | *plottype* | options: {''alltrials'' ''activations''} |
|  | *ncomponents* | 80 |
|  | *compperfig* | 40 |
|  | *maxsteps* | 20 |
|  | *eps* | 1.00E-70 |
|  | *event\_codes* | [] |
|  | *conditions* | [] |
|  | *chantype* | {''mag'' ''grad'' ''grad1'' ''grad2'' ''eeg'' ''other'' ''meg'' ''all''} |
|  | *outfile* | [] |
|  | *verbose* | options: {0 or 1} |
|  | *logfile* | [] |
|  | *logfid* | [] |
|  | *rescale* | options: {0 or 1} |
|  | *rootoutdir* | [] |
|  | *prefix* | [] |
| **Outputs:** |  |  |
|  | *epoch\_data* |  |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Andrei Irimia, 03/18/08 |
| **See also:** |  | ts\_autoICA |

**ts\_ezplot** - a single function with many ways of plotting any epoched sensor space data

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | ts\_ezplot(data, 'key1', val1, ... ); |
|  |  | ts\_ezplot( 'key1', val1, ... ); |
| **Inputs:** | | |
|  | *data* | **[any TimeSurfer data structure]** |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *datafile* |  |
|  | *chantype* |  |
|  | *channels* |  |
|  | *events* |  |
|  | *toilim* | limits on the times of interest, [begin end] in seconds |
|  | *foilim* | limits on the frequencies of interest, [low high] in Hz |
|  | *zlim* | y-axis limits for 2D plots or color limits in 3D images. (maxmin; absmax; or [low high]) |
|  | *multiplot* |  |
|  | *topoplot* | whether or not to generate topoplots (if set to 1 then multiplots will not be generated) |
|  | *toprows* | this is the number of rows in a sequence of topoplots when topoplot is set to 1 |
|  | *topcols* | this is the number of columns in a sequence of topoplots when topoplot is set to 1 |
|  | *trials\_flag* | whether to overlay trials (1) or show averages (0). (default = 0) (waveforms only) |
|  | *blc* | 2D plots: whether to baseline correct. 3D plots: whether to translate and/or normalize the data. (zscore; relative; relchange; absolute). absolute = subtract mean. relative = divide by mean. relchange = subtract mean then divide by mean. zscore = subtract mean then divide by standard deviation. metrics are calculated with respect to an interval controlled by ''blcwindow'' |
|  | *baselinetype* | 2D plots: whether to baseline correct. 3D plots: whether to translate and/or normalize the data. (zscore; relative; relchange; absolute). absolute = subtract mean. relative = divide by mean. relchange = subtract mean then divide by mean. zscore = subtract mean then divide by standard deviation. metrics are calculated with respect to an interval controlled by ''blcwindow'' |
|  | *blcwindow* | 2D plots: whether to baseline correct. 3D plots: whether to translate and/or normalize the data. (zscore; relative; relchange; absolute). absolute = subtract mean. relative = divide by mean. relchange = subtract mean then divide by mean. zscore = subtract mean then divide by standard deviation. metrics are calculated with respect to an interval controlled by ''blcwindow'' |
|  | *fontsize* | 8 |
|  | *axisfontsize* | 4 |
|  | *linewidth* | 0.5 |
|  | *linestyle* | - |
|  | *graphcolor* | brgkywrgbkywrgbkywrgbkywbrgkywrgbkywrgbkywrgbkyw |
|  | *zerolines* | 0 |
|  | *axes* |  |
|  | *axis* |  |
|  | *autoscale* | whether or not to scale each channel independently (0=no; 1=yes) |
|  | *newfig* | 1 |
|  | *background* | w |
|  | *save* | 0 |
|  | *close* | 0 |
|  | *resolution* | 150 |
|  | *format* | jpg |
|  | *outpath* | [] |
|  | *overwrite* | 0 |
|  | *alpha* | 0.05 |
|  | *overlay* | 1 |
|  | *footnote* | filename |
|  | *xlim* | maxmin |
|  | *ylim* | maxmin |
|  | *clip* | yes |
|  | *showlabels* | yes |
|  | *showbadchans* | no |
|  | *box* | no |
|  | *stim* | yes |
|  | *colorbar* | no |
|  | *title* |  |
|  | *freqcorr* | Whether to multiply power by (freq)^alpha where alpha is controlled by "freqcorr\_alpha" and freq = the frequency at which power is given. This is useful for bringing out high frequency power that drops off at rates that depend on the underlying systems. Default alpha = 2 (this adjusts for Brownian noise-like power spectra: 1/f amplitude drop-off, power density which decreases 6 dB per octave with increasing frequency (density proportional to 1/f^2)). |
|  | *freqcorr\_alpha* | The exponent used when freqcorr = 1 and power is multiplied by (freq)^alpha: alpha is controlled by "freqcorr\_alpha" and freq = the frequency at which power is given. This is useful for bringing out high frequency power that drops off at rates that depend on the underlying systems. Default alpha = 2 (this adjusts for Brownian noise-like power spectra: 1/f amplitude drop-off, power density which decreases 6 dB per octave with increasing frequency (density proportional to 1/f^2)). |
|  | *statcolor* | b |
|  | *statmarker* | \* |
|  | *transparency* | 0.1 |
|  | *colormap* | jet |
|  | *cond\_labels* | Cell array of strings listing condition labels |
|  | *conditionkey* | CSV file containing condition labels |
|  | *layout* |  |
|  | *statfile* | file name of MAT file containing stats returned by ts\_statistics or ts\_statistics\_wrapper or ts\_timefreq\_statistics |
|  | *badchanfile* | txt file listing channel labels to skip |
|  | *rejectfile* | [] |
|  | *baselinefile* | [] |
|  | *loop\_param* | {'events' 'statfile'} |
|  | *padfactor* | 0.1 |
|  | *comment* | {''auto'' ''xlim'' ''detailed'' ''none''} {default = ''detailed''} |
|  | *vline* | [] |
|  | *hline* | [] |
|  | *guimode* | 0 |
|  | *ticklength* | 0.1 |
|  | *refchan* | [] |
| **Outputs:** |  |  |
|  | *images* | any format that can be saved by the print() command |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Jason Sherfey |
| **See also:** |  | ts\_syncplot, ts\_ezmultiplot, multiplotER, multiplotTFR, topoplot, topoplotER, topoplotTFR |

**ts\_syncplot** - plot and save high-dimensional data matrices bivariate synchronization

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | ts\_syncplot(data, 'key1', val1, ... ); |
|  |  |  |
| **Inputs:** | | |
|  | *data* | [sync\_data, plv\_data] |
|  |  |  |
| **Optional inputs:** | | |
|  | *function\_id* | Numeric identifier used to reference this function''s results as an input to a subsequent function |
|  | *input\_id* | The numeric identifier of an earlier function whose results are this function''s input |
|  | *datafile* | [] |
|  | *datastring* | {'timefreq' 'metric'} |
|  | *prefix* |  |
|  | *title* | sync |
|  | *zparam* | plv |
|  | *chantype* | all |
|  | *channels* | [] |
|  | *events* | [] |
|  | *toilim* | limits on the times of interest, [begin end] in seconds |
|  | *foilim* | limits on the frequencies of interest, [low high] in Hz |
|  | *zlim* | y-axis limits for 2D plots or color limits in 3D images. (maxmin; absmax; or [low high]) |
|  | *fontsize* | 8 |
|  | *axisfontsize* | 4 |
|  | *linewidth* | 0.5 |
|  | *linestyle* | - |
|  | *strnig* | 0 |
|  | *axes* | 1 |
|  | *axis* | tight |
|  | *autoscale* | whether or not to scale each channel independently (0=no; 1=yes) |
|  | *newfig* | 1 |
|  | *background* | w |
|  | *save* | 0 |
|  | *close* | 0 |
|  | *resolution* | 150 |
|  | *format* | jpg |
|  | *outpath* | [] |
|  | *overwrite* | 0 |
|  | *footnote* | filename |
|  | *showlabels* | yes |
|  | *box* | no |
|  | *stim* | yes |
|  | *colorbar* | no |
|  | *title* | [] |
|  | *colormap* | jet |
|  | *cond\_labels* | Cell array of strings listing condition labels |
|  | *conditionkey* | CSV file containing condition labels |
|  | *layout* | [] |
|  | *loop\_param* | {'events' 'statfile'} |
|  | *padfactor* | 0.1 |
|  | *comment* | {''auto'' ''xlim'' ''detailed'' ''none''} {default = ''detailed''} |
| **Outputs:** |  |  |
|  |  |  |
|  |  |  |
| **Note:** |  |  |
| **Author:** |  | Jason Sherfey |
| **See also:** |  | ts\_ezplot, ts\_sync, ts\_PLV, ts\_ezmuliplot |

**Common parameter names**

|  |  |
| --- | --- |
| rootindir | Directory where everything will be saved (default = current working directory) |
| rootoutdir | Directory where everything will be saved (default = current working directory) |
| prefix | Prefix to all output filenames (default = ''proc'') |
| inpath | subdirectory within rootindir to search for required data files by default |
| outpath | subdirectory within rootoutdir to save results by default |
| overwrite | Whether to overwrite files |
| cluster | Any cluster jobs will be submitted to this cluster (string) |
| function\_id | Numeric identifier used to reference this function''s results as an input to a subsequent function |
| input\_id | The numeric identifier of an earlier function whose results are this function''s input |
| cluster\_flag | Whether to submit job to cluster |
| oldeventcodes | Array of event codes to change. For multiple files - specify old event codes in one numeric array per file and place all arrays in a cell. Ex. Files 1 & 2 both have event codes 1-3: {[1 2 3] [1 2 3]} |
| neweventcodes | Array of event codes to replace the old event codes. See help for oldeventcodes for handling multiple files. |
| events | [n1 n2] will compare event codes n1 and n2. {[n1 n2] [n1 n3] ...} |
| blc | yes/no - baseline correction |
| blcwindow | [begin end] in seconds |
| latency | [begin end] or ''all'' - times to be analyzed |
| foi | Frequencies of interest (Hz) |
| lpfreq | lowpass frequency in Hz |
| hpfreq | highpass frequency in Hz |
| bpfreq | bandpass frequency range, specified as [low high] in Hz |
| lnfreq | line noise frequency in Hz (default 60Hz) |
| dftfreq | line noise frequencies for DFT filter (default [60 120 180] Hz) |
| lpfiltord | lowpass filter order |
| hpfiltord | highpass filter order |
| bpfiltord | bandpass filter order |
| bsfiltord | bandstop filter order |
| lnfiltord | line noise notch filter order |
| lpfilttype | digital filter type: ''but'' (default) or ''fir'' |
| hpfilttype | digital filter type: ''but'' (default) or ''fir'' |
| bpfilttype | digital filter type: ''but'' (default) or ''fir'' |
| bsfilttype | digital filter type: ''but'' (default) or ''fir'' |
| channels | vector of channel indices - overrides chantype |
| chantype | options: {'all' 'mag' 'grad1' 'grad2' 'eeg' 'other' 'grad' 'meg'} |
| badchanfile | txt file listing channel labels to skip |
| statfile | file name of MAT file containing stats returned by ts\_statistics or ts\_statistics\_wrapper or ts\_timefreq\_statistics |
| toilim | limits on the times of interest, [begin end] in seconds |
| foilim | limits on the frequencies of interest, [low high] in Hz |
| zlim | y-axis limits for 2D plots or color limits in 3D images. (maxmin; absmax; or [low high]) |
| conditionkey | CSV file containing condition labels |
| baselinetype | {''zscore'' ''relative'' ''relchange'' ''absolute''} |

|  |  |  |  |
| --- | --- | --- | --- |
| **TimeSurfer** | **TS GUI label** | **FieldTrip** | **Description** |
| baseline\_flag | blc | baseline correction | Subtract mean baseline |
| baseline\_start, baseline\_end | blcwindow | baseline window | Interval for mean calculation |
|  |  |  |  |

**Appendix C - TimeSurfer Development**

**Subversion Repository** (directory structure, checkout, update, commit)

**TimeSurfer structures** (types & conversions)

**TimeSurfer functions** (types & conventions)

**Master files (TS GUI only)**

* guidefaults.csv
* filenames.csv
* funspecs.csv
* funtypes.csv

**Future Developments**

* Comprehensive FieldTrip wrapping
* Incorporate structural stream (MRI, DTI, etc)
* Quality Control

FUNCTION - purpose

|  |  |  |
| --- | --- | --- |
| **Usage:** |  | Format1 |
|  |  | Format2 |
| **Inputs:** | | |
|  | *Input1* | **[acceptable]** Description... Example... Notes/Comments |
|  | *Input2* | **[acceptable]** Description... Example... Notes/Comments |
| **Optional inputs:** | | |
|  | *OptInput1* | **[acceptable]** Description... Example... Notes/Comments |
|  | *OptInput1* | **[acceptable]** Description... Example... Notes/Comments |
| **Outputs** | | |
|  | *Output1* | Description... Notes/Comments |
|  | *Output2* | Description... Notes/Comments |
| **Note:** |  | |
| **Author:** |  | Name, Institution, date |
| **See also:** |  | List of related functions |