

Reference Manual for OT-Analysis kit

David M. Gomez*

2006

Contents

1	Introduction	3
2	Use examples	4
2.1	Example 1: a first analysis	4
2.2	Example 2: reading from rawdata	6
2.3	Doing graphics of some data	7
3	Parameter structures	9
3.1	Analysis Parameters Structure	9
3.1.1	Structure identification	9
3.1.2	Experiment information	9
3.1.3	Experiment design	9
3.1.4	Filtering information	10
3.1.5	Rejection parameters	10
3.1.6	Miscellaneous	10
3.2	Execution Parameters Structure	11
3.2.1	Structure identification	11
3.2.2	Input parameters	11
3.2.3	Output parameters	11
3.2.4	Miscellaneous	11

*Email dgomez@dim.uchile.cl. Department of Mathematical Engineering, University of Chile.

4	Data structures	14
4.1	Rawdata structure	14
4.2	Processed data structure	15
5	MatLab help	17
5.1	borderdetrend	17
5.2	calcHb	17
5.3	computecorr	17
5.4	cutblocks	18
5.5	filtersignal	18
5.6	OTanalysis	19
5.7	OTbigmat2struct	19
5.8	OTprocess	20
5.9	OTread	20
5.10	OTreadraw	21
5.11	OTreject	21
5.12	OTsetup	22
5.13	OTstats	22
5.14	OTstruct2bigmat	23
5.15	readData_FileVer1_06	23
5.16	readmarks	23

1 Introduction

This document presents the OT-Analysis kit. They are a set of useful MatLab scripts for analyzing OT data.

These scripts have been developed by several persons during the years (Peña, Maki, Kovacic, Shukla, among others), and in 2006 reexamined and reformulated by the author of this document, during a predoctoral work at prof. Jacques Mehler's Language, Cognition and Development Lab at SISSA.

The emphasis in the work was to clean the algorithms used to perform the process, and also the MatLab code itself. The total size of the scripts was reduced from about 2.000 lines of code (without comments) to less than 1.135 lines of code (without the documentation comments used for MatLab help command). Also this cleaning produced a much faster process, reducing its required time to about 1/6 of the required by the original scripts. The so-called OT-Analysis kit is now more readable and easier to modify, as each subprocess is performed by specific MatLab functions. These subprocesses, like hemoglobin concentrations computation and signal filtering, raise naturally when considering the whole OT data processing.

Also were taken out some parts of the code that performed unnecessary transformations on the data. The resulting scripts perform an equivalent process to the original ones, but this does not mean that the numerical differences between both outputs are always small. In order to discover the reasons of these numerical differences, before reformulating the scripts to this final version, the author programmed a set of replication scripts, intended to numerically reproduce the results of the original ones. The accuracy of this numerical replication was of the order of $1E-11$ considering the maximum difference between both outputs. These replication scripts were the basis to build the final ones.

As a future line of work, we have the possibility to explore new ways of mathematically analyzing the OT data, specifically concerning the baseline trend correction and the activation measure of an hemodynamic response. As nowadays OT data analysis is far from being a standard process, it is highly recommended to test innovative approaches as well as adapting known techniques from fMRI analysis.

2 Use examples

2.1 Example 1: a first analysis

Suppose that we want to analyze a set of textfiles containing information from 3 subjects. The experiment name is `n3x`, the files are called `n3x*.csv` and located in the subdirectory `n3x` of the current directory. We will run on them a default analysis, and save the rawdata for future uses in the file `rawn3x.mat`, in the same directory as the input files.

First of all, we need to create the parameter structs:

```
>> [ AP EP ] = OTsetup();
Setting Experiment and Analysis parameters...
Experiment name [abc] : n3x
Experiment description [Default experiment] : My experiment
Number of channels [24] :
Machine light saturation value [5] :
Number of conditions [3] :
Condition codes [FBS] :
Condition names (separated by commas) [forward,backward,silence] :
Stimuli period (in secs) [15] :
After-stimuli period (in secs) [9] :
Normalization period (in secs) [5] :
Mark of stimulus-begin [1] :
Mark of stimulus-end [2] :
Lower cutoff frequency (for signal detrend) [0.02] :
Upper cutoff frequency (for noise remotion) [1] :
Proximity tolerance to saturation for rejection [0.05] :
Duration of an artifact (in secs) [0.2] :
Threshold variation for detecting artifacts [0.1] :
Perform baseline fixing between different channels (y/n) [n] :
Setting Execution parameters...
Silent mode (y/n) [n] :
Keep intermediate data (for debugging) (y/n) [n] :
Input mode (t=textfile/r=rawdata) [t] :
Input path [./] : ./n3x/
Input file(s) [abc*.csv] : n3x*.csv
Save rawdata (y/n) [n] : y
```

```
Output path [./] : ./n3x/  
Output file [] : rawn3x.mat
```

Note that almost all the experiment and analysis parameters were set to default (by pressing ENTER), and that the changes we needed only corresponded to execution parameters. We can now run the analysis by calling `OTanalysis`.

```
>> [ bb rej ] = OTanalysis(AP, EP);  
Looking for pattern n3x*.csv...  
9 data files found.  
Reading files.  
File: n3x01B_MES_Probe1.csv...  
File: n3x01F_MES_Probe1.csv...  
File: n3x01S_MES_Probe1.csv...  
File: n3x02B_MES_Probe1.csv...  
File: n3x02F_MES_Probe1.csv...  
File: n3x02S_MES_Probe1.csv...  
File: n3x03B_MES_Probe1.csv...  
File: n3x03F_MES_Probe1.csv...  
File: n3x03S_MES_Probe1.csv...  
Saving rawdata...  
Reading marks...  
Calculating the Hb signals...  
Filtering the Hb signals...  
Extracting blocks positions...  
Separating blocks info...  
Detrending the Hb signals...
```

Considering that our experimental measurements last about 6 minutes for each condition, `OTanalysis` can take about 3 minutes to perform this job, that is approximately 1 minute per baby. This mean time can vary depending on the free memory of the system.

If we want to keep the processed data, we can save it directly to disk:

```
>> save ./n3x/processeddata.mat bb rej
```

Finally, if we want to do statistics on the processed data we call `OTstats`:

```
>> [ avgs wavgs cor ] = OTstats(bb, rej, './n3x/n3xstats.csv');
```

If we only need to create the CSV file for doing the statistics, we can also use simply

```
>> OTstats(bb, rej, './n3x/n3xstats.csv');
```

2.2 Example 2: reading from rawdata

Suppose that we now want to change the lower cutoff frequency for the filtering to 0.05 [Hz]. We can use the rawdata that we used in the previous example, to avoid reading all the information from the text files again. This time we will set the silent mode ON, and let us note that now we do not need the save the rawdata.

We can modify the settings by calling OTsetup:

```
[ AP EP ] = OTsetup(AP, EP);
Setting Experiment and Analysis parameters...
Experiment name [n3x] :
Experiment description [My experiment] :
Number of channels [24] :
Machine light saturation value [5] :
Number of conditions [3] :
Condition codes [FBS] :
Condition names (separated by commas) [forward,backward,silence] :
Stimuli period (in secs) [15] :
After-stimuli period (in secs) [9] :
Normalization period (in secs) [5] :
Mark of stimulus-begin [1] :
Mark of stimulus-end [2] :
Lower cutoff frequency (for signal detrend) [0.02] : 0.05
Upper cutoff frequency (for noise remotion) [1] :
Proximity tolerance to saturation for rejection [0.05] :
Duration of an artifact (in secs) [0.2] :
Threshold variation for detecting artifacts [0.1] :
Perform baseline fixing between different channels (y/n) [n] :
Setting Execution parameters...
Silent mode (y/n) [n] : y
```

```
Keep intermediate data (for debugging) (y/n) [n] :
Input mode (t=textfile/r=rawdata) [t] : r
Input path [./n3x/] :
Input file(s) [n3x*.csv] : rawn3x.mat
Save rawdata (y/n) [y] : n
```

Note that we can output the modified parameter structures to other variables, if we want keep the original ones.

We run the analysis again by using

```
>> [ bb2 rej2 ] = OTanalysis(AP, EP);
```

As we put silent mode ON, `OTanalysis` will not produce any text output during its process. Now the elapsed time was only 50 seconds, so we saved a good amount of time by keeping the rawdata.

Finally, to prepare the statistics file is just like the first example:

```
>> OTstats(bb2, rej2, './n3x/n3xstats_differentfilter.csv');
```

2.3 Doing graphics of some data

Suppose that we have a processed data structure `bb` (containing only one baby) that was made with the *keepintermediate* flag ON. We can do some graphics from the data it holds.

1. Global light signal of condition F, all channels together

```
>> plot(bb.F.data);
```

2. Global light signal of condition F, only channels 1 and 2

```
>> plot(bb.F.data(:,1:2));
```

3. Global OXY and DXY signal of condition B, only channel 7

```
>> plot(bb.B.hb(:,7,:));
```

4. Global OXY signal of condition B, only channel 7

```
>> plot(bb.B.hb(:,7,1));
```

5. Global DXY signal of condition B, all channels together

```
>> plot(bb.B.hb(:, :, 2));
```

6. First block OXY detrended signal of condition S, all channels together

```
>> plot(bb.S.blocks(1).data(:, :, 1));
```

3 Parameter structures

The OT-Analysis kit uses two parameter structures. These are passed by several of the kit scripts. The first one handles parameters related to the experiment and analysis of the data, whereas the second relates to each execution parameters. Both of these structures can be created by hand or using the kit script OTsetup.

3.1 Analysis Parameters Structure

This structure specifies parameters related to each experiment and its analysis. In table 1 we can find all the parameters contained in it, and their default values.

3.1.1 Structure identification

Parameters *whatami* and *version* are used only for identifying the structure.

3.1.2 Experiment information

These parameters specify each experiment. *experimentname* is a three-letter code that identifies the experiment, whereas *description* is a longer identifier.

nchannels accounts for the number of channels that the used OT machine has.

nconds is the number of conditions that the experiment has. Each condition is identified using a single upper-case letter. *conds* is a string (of length *nconds*) containing these letters. *condnames* is a cell-array containing the complete names of the experimental conditions.

3.1.3 Experiment design

Each experiment is designed to present some stimuli during a fixed amount of time, and then switching it off for observing the evolution of its associated hemodynamic response. These parameters specify the duration of each stimulus in *stimperiod* and a post-stimulus relaxation time in *afterstimperiod*.

For this information to be correctly analyzed, it is necessary to consider a normalization period before the stimulus and after the relaxation time. Its duration is set in *normperiod*.

Therefore, we define a ‘block’ to be all the samples measured during the initial normalization, stimulus, relaxation or final normalization periods related to a specific stimulus.

All the periods specified in this section are measured in seconds [s].

3.1.4 Filtering information

The analysis of the OT data relies on filtering processes both for noise removal and baseline detrend. The parameters of this section set the cutoff frequencies for the filtering process.

lowfreqcutoff sets the value of the low frequencies filter, to remove baseline trends. These are supposed to be all the slow changes in the dynamics of hemoglobin. Let us note that this baseline detrend method relies on the supposition that baseline trends behave in a periodic (cyclic) way.

highfreqcutoff sets the value of the high frequencies filter, to remove noise of the signal.

Both parameters are measured in Hertz [Hz].

3.1.5 Rejection parameters

These parameters set the criteria for separating good and bad blocks. A good block is a block that does not present artifact variations and that none of its channels get too close to saturation levels.

artifactperiod sets the duration (in seconds [s]) of the time windows the scripts will consider to look for artifact variations in the signal. *artifactthreshold* is the maximum variation that can happen in these windows for the block to be still considered good.

lightsaturationvalue is the value of saturation of the raw light signals. This is a machine-dependent parameter. *lightsaturationtolerance* is the maximum distance that the signal can approach the saturation value without being considered saturated.

3.1.6 Miscellaneous

Each stimulus of a measurement is accompanied by two marks, stating its begin and end points. These marks are numerical values that can be set with *beginmark* and *endmark*.

Finally, *dobaselinefix* is a flag (takes 0-1 values). When it is ON (equals 1), then the data processing performs as first step a baseline fixing. This means

that all the channels of the signal are divided by its mean value during 5 seconds before the begin of the first stimulus. This step was used in the original OT scripts, and it has been included here just for debugging. Its use is not recommended anymore.

3.2 Execution Parameters Structure

This structure specifies parameters related to each execution of the OT-Analysis. In table 2 we can find all the parameters contained in it, and their default values.

3.2.1 Structure identification

Parameters *whatami* and *version* are used only for identifying the structure.

3.2.2 Input parameters

These parameters specify the input of the OT analysis. *inputmode* can take two values: ‘textfile’ or ‘rawdata’. Use the former when you want the data to be read from files produced by the OT machine, and the latter when the data has already been read as saved as a rawdata structure by other OT analysis.

inputpath is the directory path where the input files specified in *inputfiles* are located.

3.2.3 Output parameters

These parameters are used to output the rawdata structure produced after reading the input datafiles. *saveraw* is the name of the file to be saved (must be .MAT), and *outputpath* is the directory where this output will be located.

Setting *saveraw* to an empty string (‘’) makes the rawdata structure not to be saved. In this case, *outputpath* is irrelevant.

3.2.4 Miscellaneous

silent and *keepintermediate* are flags. When the former is turned ON, then the OT-analysis scripts do not produce any text output to the screen during their execution (excepting errors). When the latter is ON, then the processed

Structure identification	
Parameter name	Default value
whatami	'analysisparametersstruct'
version	'1.0'
Experiment information	
Parameter name	Default value
experimentname	'abc'
description	'Default experiment'
nchannels	24
nconds	3
conds	'FBS'
condnames	{'forward', 'backward', 'silence' }
Experiment design	
Parameter name	Default value
stimperiod	15.0
afterstimperiod	9.0
normperiod	5.0
Filtering information	
Parameter name	Default value
lowfreqcutoff	0.02
highfreqcutoff	1.0
Rejection parameters	
Parameter name	Default value
artifactperiod	0.2
artifactthreshold	0.1
lightsaturationvalue	5.0
lightsaturationtolerance	0.05
Miscellaneous	
Parameter name	Default value
beginmark	1
endmark	2
dobaselinefix	0

Table 1: Experiment and Analysis parameters default values.

data structures produced by the scripts keep all the intermediate computations. This is used for debugging, and makes much bigger structures.

Structure identification	
Parameter name	Default value
whatami	'executionparametersstruct'
version	'1.0'
Input parameters	
Parameter name	Default value
inputmode	'textfile'
inputpath	'./'
inputfiles	'abc*.csv'
Output parameters	
Parameter name	Default value
saveraw	'
outputpath	'./'
Miscellaneous	
Parameter name	Default value
silent	0
keepintermediate	0

Table 2: Execution Parameters default values.

4 Data structures

This section describes the two data structures used by the OT-Analysis kit: raw and processed data.

4.1 Rawdata structure

Each rawdata structure contains the measured data and information from one subject. They are composed of the following fields:

whatami Used for structure identification, its value is ‘rawdatastructure’.

fileversion Contains the fileversion code of the original text datafile (the one obtained from the OT machine). By now, its value is always ‘1.06’.

experimentname The three-letter identification code of the experiment.

id Subject’s identification code.

personal Structure specifying personal information of the subject.

analyze Structure specifying the machine analysis information.

conds Conditions measured for the subject (must match the experiment’s conditions, can be just a subset of them). This field is an alphabetic upper-case string, where each letter is the codename of an experimental condition.

Conditions Data For each condition specified in conds, the rawdata structure has a field with the same code letter. These fields are structures also, each one of them containing three fields:

measure is the measurement information.

data is a matrix containing the data collected by the OT measurement.

mark is a vector containing the marks sent by the machine during the measurement.

The contents of the structures *personal*, *analyze* and *measure* can depend on the fileversion code. By now, the OT-Analysis kit relies only in the following values:

measure.wavelength is the vector of wavelengths associated to each light channel of the OT machine. Its measurement unit is supposed to be nanometers [*nm*].

measure.samplingperiod is the sampling rate of the measurement (time between two measurements). Its measurement unit is supposed to be seconds [*s*].

4.2 Processed data structure

If you look a processed data structure, it seems equal to a rawdata structure. In fact, the only change is that *whatami* is now ‘processeddatastructure’.

The process is reflected into the condition data fields of the structure. As in rawdata these fields only contained *measure*, *data* and *mark*, now they have the processed data:

measure is still there, is the measurement information.

stims is a (stimulus \times 2) matrix, whose first and second columns contains the starting and ending points (measure samples) of each stimulus.

blkspos is matrix of the same size as *stims*, whose first and second columns contains the starting and ending points of each block.

blocks is an array containing the processed data separated by block. Each element of the array is composed of the following fields:

rawdata is the same data contained in a rawdata structure, but trimmed to the samples corresponding to the current block.

data is the finally processed data of this block. It is a 3-D array of size (block samples \times channels \times (*OXY* – *DXY*)).

If you turn on the flag *keepintermediate* for the processing, then the processed data structure corresponding to each condition will be the following:

measure is the same as before.

data is the same big matrix contained in the rawdata structure.

mark is the marks vector contained in the rawdata structure.

stims is the same as before.

rawhb is the hemoglobin concentrations 3-D array, for the complete measurement, directly as computed by `calcHb`.

hb is the hemoglobin concentrations 3-D array, for the complete measurement, after being processed by `filtersignal`.

blkspos is the same as before.

blocks is the same as before, but now contains also more information:

rawdata is the same as before.

nodtdata is the extracted hemoglobin signal from the complete measurement, but before applying `borderdetrend`.

data is the same as before.

5 MatLab help

5.1 borderdetrend

`borderdetrend` removes a linear trend from its input such that the values of the function at the borders get close to zero.

`dtdata = borderdetrend(data)` removes a linear trend from `data` such that `dtdata(1)` and `dtdata(end)` go to zero. `dtdata` has the same size as `data`. If `data` is a matrix or a multidimensional array, the `borderdetrend` is performed by its first dimension, over all the possible combinations contained in `data`. `dtdata` has also the same size as `data`.

`dtdata = borderdetrend(data, width)` removes a linear trend from `data` such that the mean values of the first and last `width` samples of `data` go to zero.

5.2 calcHb

`calcHb` computes hemoglobin concentration signals from light absorption signals obtained by OT measurements.

`hb = calcHb(data, wv1)` computes hemoglobin concentration signals from the matrix `data`, which contains as columns the light signals measured. `wv1` is a vector containing for each column of `data` its associated wavelength. If `data` has $2 \cdot N$ columns, then `wv1` must contain $2 \cdot N$ values, and `hb` is a

$$(\text{nsamples} \times N \times (OXY - DXY))$$

3-D array. `nsamples` is the number of rows of `data`. All values in `data` are thresholded to a minimum of 0.001, to avoid zero or negative values of the light signals due to artifacts.

The conversion is done using a modified Beer-Lambertz Law that *does not* take into account the DPF factors associated to each light signal.

`calcHb` requires the data file `e_coef.mat`.

5.3 computecorrs

`computecorrs` compute the correlation coefficients between a set of hemodynamic responses and an 'ideal' hemodynamic response.

`cc = computecorrs(data)` compute the correlation coefficients between the signals contained in `data` and the ideal signal. It is assumed that `data` is

a matrix of size

(signals \times samples)

or

(samples \times signals)

where the number of samples is much greater than the number of signals.
`computecorrs` requires the data file `stims.mat`.

5.4 cutblocks

`cutblocks` computes the beginning and ending positions of all the blocks associated to the stimuli presented in an OT experiment.

`blks = cutblocks(stims, times)` does this job by putting in the first column of `blks` all beginning positions of the blocks and in the second column all ending positions. `stims` keeps the beginning and ending positions of all the stimuli, distributed in the same way.

The `times` structure must contain the following fields, all of them measured in seconds:

- `normperiod`: Normalization windows considered at the beginning and at the end of each block.
- `stimperiod`: Duration of each stimulus.
- `afterstimperiod`: Time considered after the offset of each stimulus for the hemodynamic response to finish.
- `samplingperiod`: Sampling period of the measurement.

Each block has then size

```
ceil( (2*normp. + stimp. + afterstimp.)/samplingp. )
```

5.5 filtersignal

`filtersignal` applies a bandpass frequency filter to a signal.

`fs = filtersignal(s, samp)` takes out from `s` all the wave components of frequencies lower than 0.02 Hz and higher than 1.0 Hz, using a FFT filter. `samp` is the sampling rate associated to the measurement of `s`. `fs` has the same size as `s`. If `s` is a matrix or a multidimensional array, then `fs` is

computed by filtering `s` by its first dimension. `fs` has also the same size as `s`.

`filtersignal(s, samp, low, high)` applies the filtering considering a low frequency cutoff of `low` Hz and a high frequency cutoff of `high` Hz. `low` and `high` must both be positive reals. The exception is that they may be -1, that means that the corresponding frequency filter is disabled.

`filtersignal` is built upon a non completely tested filtering method, so it may change its implementation in future versions.

5.6 OTanalysis

`OTanalysis` performs a complete analysis on OT data.

`bb = OTanalysis(AnPar, ExPar)` takes the parameter structs `AnPar` and `ExPar` built by `OTsetup` and outputs the processed data in the structure array `bb`. Each element of the array corresponds to each particular baby contained in the input file specified in `ExPar`.

`[bb rej] = OTanalysis(AnPar, ExPar)` also creates a rejection structure array `rej`, which contains the rejection codes associated to each tuple (baby, condition, channel, block). For more information on blocks, see `cutblocks`.

See also: `OTsetup`, `OTreject`, `cutblocks`.

5.7 OTbigmat2struct

`OTbigmat2struct` converts processed OT data from the 6-D matrix format to the processed structure array format.

`bb = OTbigmat2struct(mat, exp, id)` builds a processed data structure from the 6-D matrix `mat`. Only the information associated to linear local detrend is considered. The required parameters `exp` and `id` are the experiment name and baby id, respectively. The experimental conditions are labeled by default A, B, C, ...

`OTbigmat2struct(mat, exp, id, conds)` allows to specify the condition names of a particular experiment. `conds` must be an alphabetic string, so each letter will be considered as a different condition name. All the condition codes must be different, and will be converted to upper case.

See also: `OTstruct2bigmat`.

5.8 OTprocess

`OTprocess` performs the process of the OT data analysis itself.

`bb = OTprocess(raw, AnPar, ExPar)` outputs processed data in the structure array `bb`, using the parameters contained in `AnPar` and `ExPar` (these are created by `OTsetup`). `raw` is the rawdata structure array output by `OTread`.

`OTprocess` is part of the whole processing performed by `OTanalysis`.

The main stages composing `OTprocess`, for each (baby-condition) measurement, are:

- Processing the stimuli begin-and-end marks (see `readmarks`).
- If specified in `AnPar`, fix the baseline level between the different channels.
- Computing the hemoglobin concentration signals (see `calcHb`).
- Filtering the hemoglobin signals, for noise and trend remotion (see `filtersignal`).
- Determining the begin-and-end positions for each block (see `cutblocks`), and leaving this information in separate arrays.
- Linear detrending of the hemoglobin signals local to each block (see `borderdetrend`).

The processed data structure array contains all the OT information, and the data itself is separated in substructures with the same names as the experimental conditions.

If `ExPar` specifies it, then all the intermediate computed data is kept into the structure, for debugging purposes. This strongly increases the memory size required for storing it.

See also: `OTsetup`, `OTread`, `OTanalysis`, `readmarks`, `calcHb`, `filtersignal`, `cutblocks`, `borderdetrend`.

5.9 OTread

`OTread` converts to a processed data structure OT data stored in a text datafile. The allowed datafile format is rawdata exported by the OT machine, version 1.06.

`raw = OTread(AnPar, ExPar)` loads into `raw` the data stored in the files specified in the parameter structures `AnPar` and `ExPar` (see `OTsetup`).

`OTread` is part of the whole processing performed by `OTanalysis`.

See also: `OTsetup`, `OTanalysis`, `OTreadraw`, `readData_FileVer1_06`.

5.10 OTreadraw

`OTreadraw` loads to memory OT data stored in `rawdata` structures from a `.MAT` file.

`raw = OTreadraw(AnPar, ExPar)` loads into `raw` the data stored in the files specified in the parameter structures `AnPar` and `ExPar` (see `OTsetup`).

`OTreadraw` is part of the whole processing performed by `OTanalysis`.

See also: `OTsetup`, `OTanalysis`, `OTread`.

5.11 OTreject

`OTreject` computes the rejection codes associated to an experimental condition.

`rej = OTreject(cnd, AnPar, ExPar)` computes the rejection codes associated to condition `cnd` (see `OTprocess`). `AnPar` and `ExPar` are parameter structs built by `OTsetup`. `rej` is a matrix of size

(blocks \times channels)

where each component takes one of the following values:

- 0) OK
- 1) Saturated light signal
- 2) Artifact present
- 3) Both 1 and 2

`OTreject` is part of the whole processing performed by `OTanalysis`. It is possible that future versions of `OTanalysis` consider more rejection codes.

See also: `OTsetup`, `OTanalysis`, `OTprocess`.

5.12 OTsetup

`OTsetup` builds the parameter structures necessary to run most of the OT scripts.

`[AnPar ExPar] = OTsetup()` builds parameter structures `AnPar` and `ExPar` by asking the user to modify a couple of default parameter structures. For each information asked to the user, `OTsetup` provides between brackets its default value. The user then can enter a new value by typing it, or accept the default value by pressing ENTER.

`OTsetup(formerAnPar, formerExPar)` takes as default values the ones specified in `formerAnPar` and `formerExPar`.

5.13 OTstats

`OTstats` computes averages of `OTanalysis` output data. Each one of the outputs is a structure array `sta`, such that for each baby index i and condition C , `STA(i).C` is a matrix containing some averaged information useful to do the statistics of the OT experiment.

`av = OTstats(bb, rej)` computes the average response of all the good trials of all baby data contained in `bb`. Good trials are defined as those not rejected in the rejection structure `rej`. `bb` and `rej` must have the same size. Each matrix contained in `av` has size

$$(\text{nsamples} \times \text{nchannels} \times (OXY - DXY - TOT))$$

`[av wav] = OTstats(bb, rej)` also computes windowed averages. Each matrix in `wav` is obtained by reducing the corresponding `av` matrix to 7 windows of approximately the same length. `wav` is much better than `av` for doing the statistics. The size of `wav` matrices is

$$(7 \times \text{nchannels} \times (OXY - DXY - TOT))$$

`[av wav crr] = OTstats(bb, rej)` also computes the correlation coefficients between each windowed average contained in `wav` and an ‘ideal’ hemodynamic response. Each matrix contained in `crr` has size

$$(\text{nchannels} \times (OXY - DXY - TOT))$$

`OTstats(bb, rej, filename)` also outputs in the file `filename` a matrix containing all the relevant data for the statistical analysis, so it can be read by other software.

This function is not completely parameterized yet, so it assumes the following values: `nchannels = 24`, `nsamples = 340`.

5.14 OTstruct2bigmat

`OTstruct2bigmat` converts processed OT data from the processed structure format to 6-D matrix format.

`mat = OTstruct2bigmat(bb)` builds a 6-D matrix from the processed data structure `bb`. Only the coordinates associated to linear local detrend of `mat` are filled, that means only the positions whose second dimension is 2. If `bb` is a processed structure array, then only the first element of the array is considered.

See also: `OTbigmat2struct`.

5.15 readData_FileVer1_06

`readData_FileVer1_06` reads all the information and OT data contained in a rawdata textfile exported by the OT machine. The datafile version is 1.06.

`[pat ana mea data] = readData_FileVer1_06(txtfile)` reads information from `txtfile` and puts it into four output structures:

- `pat` contains the patient information.
- `ana` contains the analysis information.
- `mea` contains the measurement information.
- `data` contains the whole OT data and experiment marks.

`readData_FileVer1_06(txtfile, nlightchannels)` reads OT data consisting on `nlightchannels` channels. The default value for `nlightchannels` is 48.

5.16 readmarks

`readmarks` reads a matrix of marks and outputs the begin and end instants of each stimulus.

`sts = readmarks(marks)` outputs a matrix whose first column contains the beginning position of each stimulus, and whose second column contains

the ending position. `marks` is a vector containing all the marks read from an OT measurement file.

`readmarks(marks, begin, end)` allows the user to specify different begin and end marks for the stimuli. By default, `begin` is set to 1 and `end` is set to 2.