A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Introduction

Welcome to seeVR! This toolbox comprises a series of Matlab functions designed for the purpose of analyzing CVR and related hemodynamic data. Currently there are some dependencies on specific Matlab toolboxes that I hope to remove in future versions in order to make seeVR more applicable and user friendly. For updates please see www.seevr.nl!

The main function files are located in the 'functions' directory. How they are used is outlined in this manual. There are also a series of helper functions and work-in-progress functions (for example related to gas trace import) that will receive more focus and support in the future. I hope to update this manual and the site to keep pace with changes in the toolbox. I also welcome collaborators who wish to help expand this work. And finally, if you see errors or wish for new/additional implementations, please contact me at: a.bhogal@umcutrecht.nl or leave a note at the seevr.nl site.

Table Of Contents

A toolbox for analyzing cerebro-vascular reactivity (CVR) data

General functions used by seeVR

data: 4D (x,y,z,t) or 3D (x,y,t) time-series data

mask: region of interest mask corresponding with input data

grabTimeseries: takes 3D/4D input data and mask and returns an n x p array of voxel time-series and respective coordinates where n=the number of voxels contained within the mask and p=the number of data-points in time-series

usage: [voxel_ts, coordinates] = grab_ts_vec(**data**, **mask**)

meanTimeseries: takes 3D/4D input data and returns a 1 x p vector representing the mean intensity values of the input data.

usage: [img_ts] = meanTimeseries(**data**, **mask**)

normTimeseries: takes 3D/4D input data and returns the %intensity change with respect to some specified baseline period. The baseline period can be passed using the two element vector '**idx**' (i.e. [start_idx end_idx]). If no indices are provided, the user must manually specify the start and end indices using two mouse clicks on a pop-up figure.
usage: [ndata] = normTimeseries(**data**,**mask**,**idx**) without manual selection
usage: [ndata] = normTimeseries(**data**,**mask**) with manual selection

chopTimeseries: takes 3D/4D input data and returns a reduced timeseries using the indices specified by '**idx**'. (i.e. [start_idx end_idx]). If no indices are provided, the user must manually specify the start and end indices using two mouse clicks on a pop-up figure. The time-series header information contained within the global **opts** structure is updated accordingly. Furthermore, the **opts**.xdata parameter is updated to reflect the new length of the time-series (this is useful for plotting). The **opts**.dyn parameter is also updated (defines the number of elements in the temporal dimension).
usage: [ndata] = chopTimeseries(**data**,**mask**,**idx**) without manual selection
usage: [ndata] = chopTimeseries(**data**,**mask**) with manual selection

denoiseData: this function returns de-noised input data using a wavelet-based approach. If the wavelet toolbox is not available, de-noising is applied using the 'smoothdata' function from Matlab. An output figure showing the effect of de-noising on the mean time-series specified by the input mask is saved in the figure directory specified by **opts**.figdir. If opts.figdir is not specified, the figure is saved in the root directory of the dataset being processed.
usage: [denData] = denoiseData(**data**,**mask**,**opts**)

-For wavelet denoising, the wavelet family and denoising level can be specified as: **opts**.wdlevel (default: 'db4') & **opts**.family (default: 3). The **opts**.family options: 'haar', 'dbN', 'fkN', 'coifN', or 'symN' where N is the number of vanishing moments.
-If no wavelet toolbox is present, the used can perform a moving window denoising procedure by specifying the window size and method: **opts**.winsize (default: 0.5*length(**data**)) & **opts**.method (default: 'movmean'). For more information see Matlab documentation for 'smoothdata.m'.

smthData: performs spatial smoothing on 3D or 4D data using a gaussian kernel. The used can specify whether a 2D (i.e. in-plane smoothing or 3D (i.e. volume smoothing) kernel is used. In-plane smoothing may be desired in cases where slices are not acquired consequtively or where data is acquired using muli-band acceleration techniques. To mitigate edge effects for 3D smoothing, data is first dilated prior to smoothing and is then multiplied with the whole brain **mask** prior to smoothing. In-plane smoothing is more accurate while 3D smoothing may be favorable for producing nice image.
usage: [sdata] = smthData(**data**, **mask**, **opts**)
-FWHM: **opts**.FWHM (default 4mm)
-Filtersize: **opts**.filtWidth (default is 5 voxels)
-Spatial smoothing (in-plave/3D): **opts**.spatialdim (default is 2; select 3 for 3D smoothing)

freqSpec: returns the frequency or power spectrum of time-series data along with the average amplitude/power in the mask region and a vector containing the frequency values analyzed based on the repetition time (**opts**.TR)
usage: [fSpec, meanSpec, freq] = smthData(**data**, **mask**, **opts**)

seeVR A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Function

trAlign: temporally aligns input signal with probe of interest - both probe and input signal must have the same sampling rate.

usage: trAlign(**main_probe**, **secondary_probe**, **data_timeseries**, **opts**)
usage: trAlign(**main_probe**, **data_timeseries**, opts)

Input description

main_probe: this is the main_probe vector that is to be aligned with data_timeseries vector. This probe should be interpolated to the same sampling frequency as the data_timeseries. For CVR analysis, this probe typically consists of the end-expired CO_2 values.

secondary_probe (OPTIONAL): this is typically the corresponding interpolated end-expired O_2 vector. If this argument is provided, the trAlign will also output the re-aligned O_2 trace.

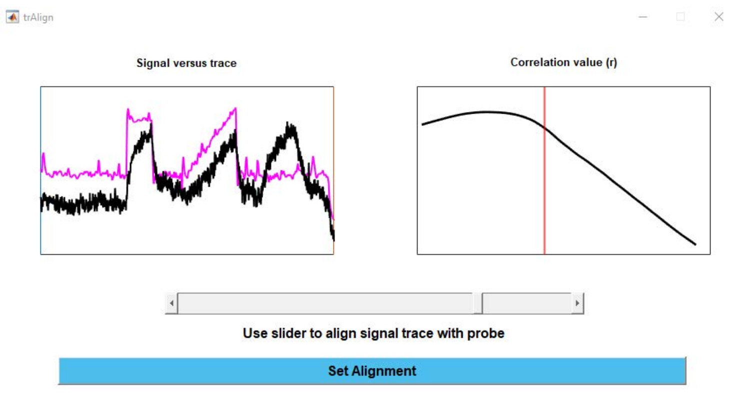
***NOTE**: the main_probe and secondary_probe can be interchanged to for example perform alignment using O_2 instead of CO_2 information

data_timeseries: This is a vector of the data that is to be aligned with the main_probe. This is typically the average signal value recorded in a specific region of interest (i.e. grey matter or cerebellum). The data_timeseries can be generated using the 'meanTimeseries' function (see general functions documentation.

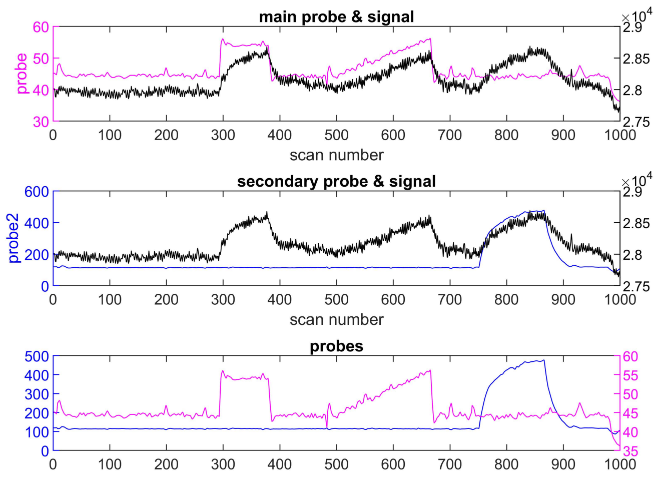
Option description (opts)

opts.figdir: /path to specific director for saving figures only
opts.savedir: /path to general save directory associated with dataset being analyzed
*Only one directory path needs to be specified. If no paths are specified, the alignment figure will be saved in the root directory associated with the dataset being processed.

Process



The trAlign function call will open a GUI. An initial correlation is made; however, noise and other signal fluctuations can sometimes result in sub-optimal alignment. Use the slider to refine the alignment. To set the final alignment press the 'Set Alignment' button.



Upon setting the alignment, the original figure dialogue will close and an additional figure showing the resulting alignment between the main (and optional secondary) probe will be generated. This figure is saved in the directory specified in the opts structure. Finally, the aligned main (and optional secondary) probes (probe1 & probe 2) are output to the Matlab Workspace to be used in further analysis.

seeVR

A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Function usage

remLV: uses 4D (x,y,z,t) data or 3D (x,y,t) data to generate and save temporal SNR maps (tmean/tsd), cov maps (1/tSNR), temporal standard deviation

maps (SD) and 1/SD maps.

usage: [mmask] = remLV(**data**, **mask**, **opts**)

Input description

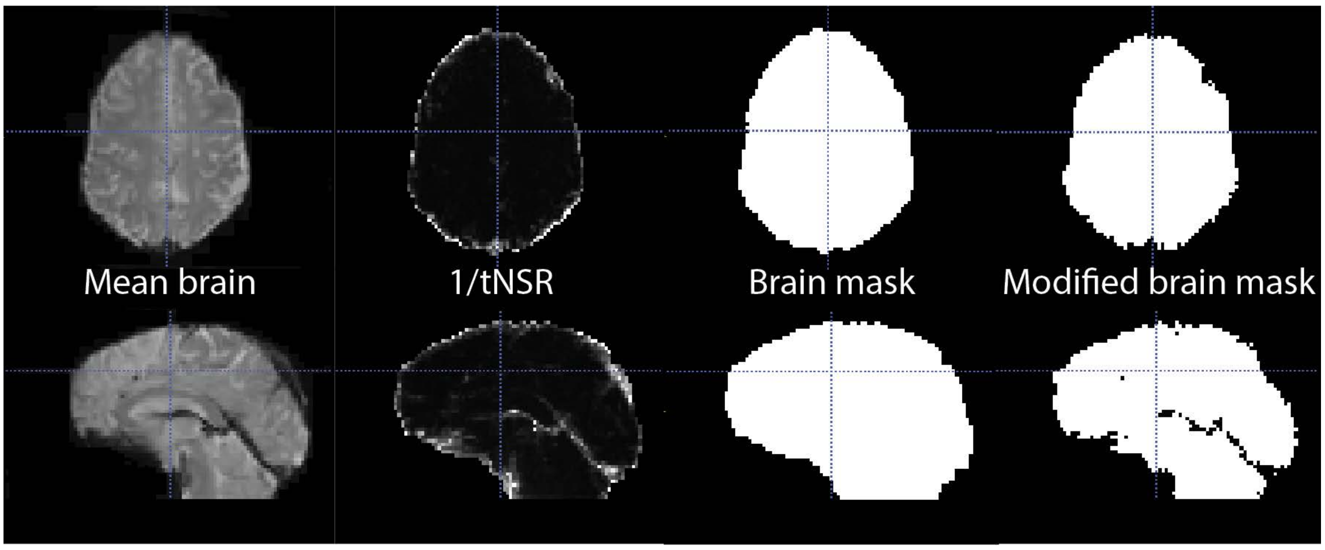
data: 4D (x,y,z,t) timeseries data
mask: mask corresponding to voxels of interest for time-series

Option description (opts)

opts.resultsdir: /path to specific director for saving image files and parameter maps that are created during processing
opts.savedir: /path to general save directory associated with dataset being analyzed
*Only one directory path needs to be specified. If no paths are specified, output files will be saved in the root directory associated with the dataset being processed.

opts.LVthresh: this parameter sets the threshold value that is used to remove high intensity voxels from the tNSR maps (inverse of temporal SNR). When this parameter is not set the default action is to remove voxels within the 98-100th percentile range. Generally, the LVthresh parameter is data-dependent and so care must be taken to ensure mainly large vessel and CSF voxels are removed from the modified whole brain mask. This parameter can be iteratively tuned based on inspection of saved tNSR map.
opts.LVpercentile: this parameter sets the lower limit that determines the percentile range of data to be removed (default: >98th percentile). Setting this parameter will override the opts.LVthresh parameter.

Process



The remLV takes advantage of the fact that signals from large vessels and regions affected by susceptibility differences exhibit high signal variation. This information is used to modify input brain masks such that voxels exceeding a user defined intensity threshold are removed. The function returns a modified brain mask that an be used in further processing and analysis steps.

A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Function usage

convHRF: takes an input vector and performs a convolution with a double gamma function defined as follows:

$$s(t) = \frac{t^{(\alpha_1-1)}\beta_1{}^{-\alpha_1}e^{-\beta_1{}^{-1}t}}{\gamma(\alpha_1)} - \frac{t^{\alpha_2-1}\beta_2{}^{-\alpha_2}e^{-\beta_2{}^{-1}t}}{\delta\gamma(\alpha_2)}$$

The function takes as main input a time-series vector to be convolved with the HRF. The parameters that define s(t) are passed outside of the function call via the **opts** structure. Currently, modifiable parameters are alpha_1 (shifts response onset; effectively shifts the input probe), beta_1 (introduces dispersion), beta_2 (introduces undershoot) and finally sigma (effects the vertical spread of the response). The convHRF function returns an array of HRFs (**HRF_probe**) whose size depends on the number of parameters are passed in the **opts,** as well as the actual **HRF** functions plotted based on the parameters.

This function is inspired by the follow works:
https://pubmed.ncbi.nlm.nih.gov/24252847/ (PMID: 24252847 DOI: 10.1038/jcbfm.2013.200)
https://pubmed.ncbi.nlm.nih.gov/33444087/ (PMID: 33444087 DOI: 10.1177/0271678X20978582)

usage: [**HRF, HRF_probe**] = convHRF(**HRFprobe**, **opts**)

Input description

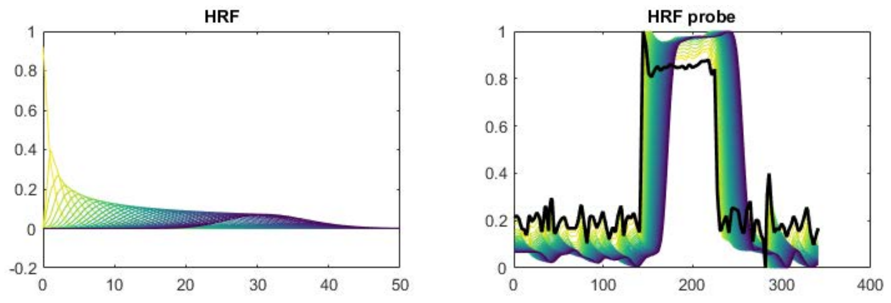
HRFprobe: 1D time-series vector; typically this might be measured end-expired gas values for a particular experiment

Option description (opts)

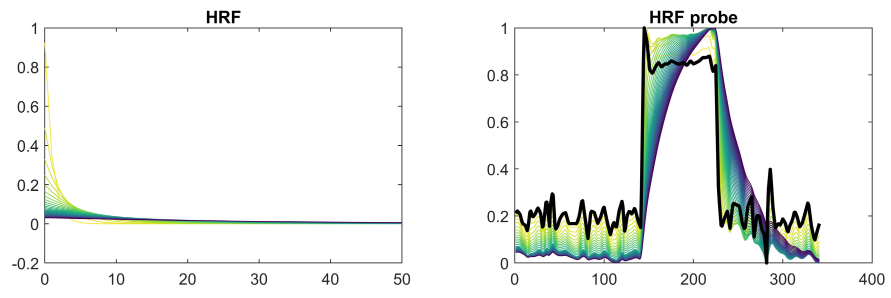
opts.resultsdir: /path to specific director for saving figures. If unspecified, current directory is used.
opts.onset: a 1D vector defining the alpha_1 parameter increments (onset)
opts.disp: a 1D vector defining the beta_2 parameter (dispersion)
opts.under: a 1D vector defining the beta_2 parameter increments (undershoot)
opts.rration: a 1D vector defining the sigma parameter (vertical spread). The default is 1000 to avoid vertical dispersion of the response
opts.plot: setting this parameter to 1 will return plots of the **HRF** and **HRF_probe**

Process

It is quite common to model the hemodynamic response to a stimulus via convolution with an HRF. This function provides the flexibility to generate a wide range of model HRFs. Below are two examples here a series of HRF response functions a are generated by incrementing the **opts**.offset (top) and **opts**.disp parameters (botom). Parameters can also be modified simultaneously.



opts.offset = 1:1:32; **opts**.disp = 1; **opts**.under = 1; **opts**.rratio = 1000



opts.offset = 1; **opts**.disp = 1:1:32; **opts**.under = 1; **opts**.rratio = 1000

A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Function usage

genGS: takes up to two sets of regressors to generate a global signal based on 4D (x,y,z,t) or 3D (x,y,t) input data. This function provides as output: 1) a time-series consisting of the 'residual signal'. This can be seen as a sort of 'pseudo resting-state' signal if both nuisance and data regressors are passed to genGS; 2) a 1D 'residual' timeseries signal that can be considered a global signal regressor; 3) a time-series consisting of the 'cleaned signal'. This series is created by subtracting the 3D/4D residual data from the original 3D/4D input data.

usage: [**cleanData**, **res_ts**, **resData**] = genGS(**data**, **mask**, **nuisance**, **probe**, **opts**)
usage: [**cleanData**, **res_ts**, **resData**] = genGS(**data**, **mask**, [ ], **probe**, **opts**)
usage: [**cleanData**, **res_ts**, **resData**] = genGS(**data**, **mask**, **nuisance**, [ ], **opts**)

Input description

data: 4D (x,y,z,t) time-series data or 3D (x,y,t) time-series data
mask: mask corresponding to voxels of interest for time-series
nuisance: nuisance regressors that might include motion parameters and their derivatives in addition to linear or polynomial terms **probe**: data probes derived from the regressor of interest (i.e. end-tidal CO_2). Derivatives can be generated using the **convHRF** function

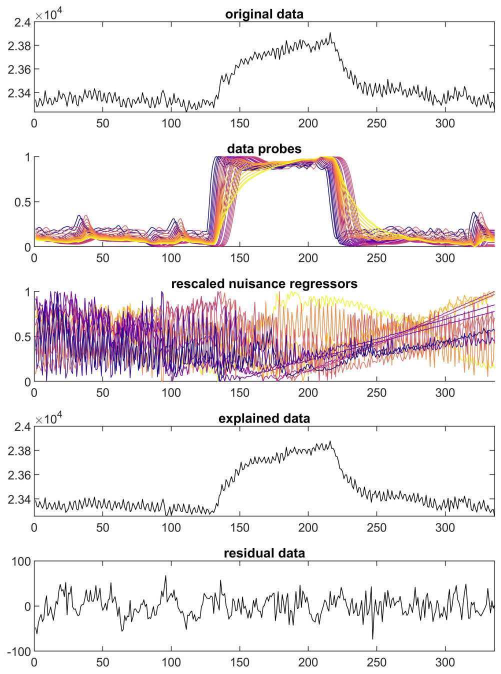
Option description (opts)

opts.figdir: /path to specific director for saving figure files
*If no figdir path is specified, output figure will be saved in current Matlab directory.

Process

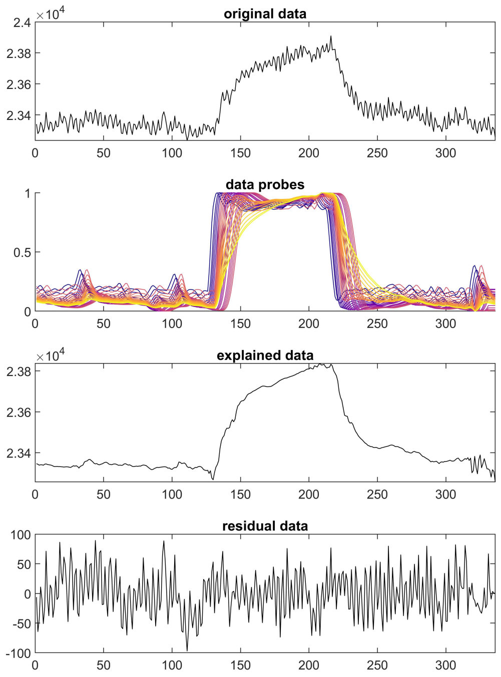
This genGS function uses a priori information from nuisance and data regressors to explain as much of the input signal as possible. Whatever can not be explained by these regressors is attributed to the global signal (i.e. residual motion, physiological noise etc.). It is also possible to run genGS using nuisance or data regressors alone.

Global signal created with nuisance + data regressors



Both the data probes generated using the convHRF function along nuisance motion parameters and their temporal derivatives are used to determine global residual data

Global signal created with data regressors only



Only the data probes generated using the convHRF function are used to determine global residual data

seeVR

A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Function usage

scrubData: takes up a set of nuisance regressor along with a 1D data probe and attempts to 'regress out' nuisance contributions from 4D (x,y,z,t) or 3D (x,y,t) input data. This function provides as output a 4D time-series consisting of the 'cleaned signal'. This series is created by subtracting the combined nuisance contributions from the original 4D input data.

usage: [**cleanData**] = scrubData(**data**, **mask**, **nuisance**, **probe**, **opts**)

Input description

data: 4D (x,y,z,t) time-series data or 3D (x,y,t) time-series data
mask: mask corresponding to voxels of interest for time-series
nuisance: nuisance regressors that might include motion parameters and their derivatives, linear or polynomial terms or a global signal regressor generated using the **genGS** function.
probe: this is the main probe of interest; typically the end-expired CO_2 trace or a 1D model of the applied stimulus paradigm

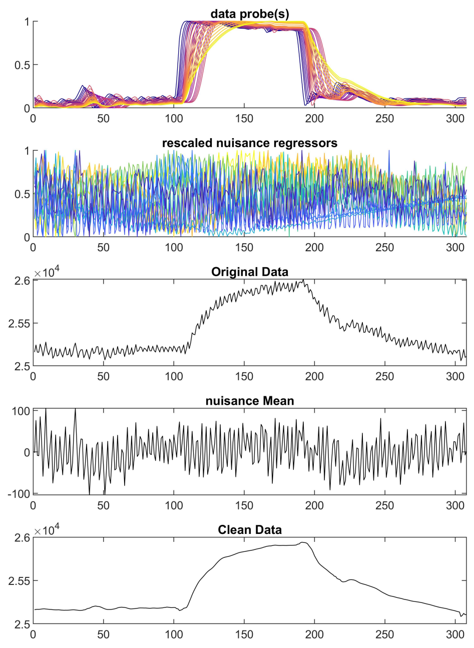
Option description (opts)

opts.figdir: /path to specific director for saving figure files
*If no figdir path is specified, output figure will be saved in current Matlab directory.
opts.save_cleaned: (default: opts.save_cleaned = 0;) if this parameter is set to 1 the cleaned time-series will be saved in the specified savedir
opts.savedir: /path to save cleaned image time-series
*If no savedir path is specified, output figure will be saved in current Matlab directory.

Process

The scrubData function is similar to the **genGS** function except for the fact that it does not consider the data probe(s) when reconstruction the cleaned data. Where the intended use of **genGS** is to use as much a priori data as is available to *explain* the observed signal, the intended use of scrubData is to remove nuisance signals from the data of interest. Therefore, extra caution is required to ensure that nuisance regresors do not strongly correlate with signals of interest (for example head-motion with breath-hold epoch or CO_2 block). Moreover, the data probes can be used as 'place holders' so as not to remove signals of interest in the case that nuisance parameters share subtle correlations with signals of interest (i.e. leading to a stronger weighting towards the data probe response rather than the nuisance response). In cases where there is a strong understanding of the expected response (i.e. a block response to a block stimulus in a healthy volunteer), a two step approach can be taken where **genGS** is used to create a global signal that can be fed into scrubData as a regressor to improve the removal of nuisance signals for more accurate lag analysis using the **lagCVR** function.

Removal of nuisance signals using data probes and nuisance regressors



A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Function usage

fALFF: uses 4D (x,y,z,t) data or 3D (x,y,t) data to generate and save temporal ALFF and functional ALFF (fALFF) maps and associated z-score maps. These maps represent the sum squared amplitude of low frequency fluctuations within specified frequency bands.

usage: [**ALFF_map fALFF_map zALFF_map zfALFF_map**] = fALFF(**data**, **mask**, **refmask**, **opts**)

Input description

data: 4D (x,y,z,t) or 3D (x,y,t) timeseries data
mask: mask corresponding to voxels of interest for time-series
remask: an additional masks in which the reference spectrum for the ALFF map is calculated. Typically this is the global mean, however the **refmask** allows reference from any region to be used; see: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3902859/

Option description (opts)

opts.resultsdir: /path to specific director for saving image files and parameter maps that are created during processing
opts.fpass: a two element vector defining the lower and upper values of the ALFF passband. For neuronal oscillations this is typically set between [0.01 0.08].

Process

The amplitude of low frequency fluctuations has been suggested to reflect the intensity of spontaneous neuronal activity. Further evidence of ALFF/fALFF as a marker of brain pulsation related to cardiac and respiratory effects has also been proposed;
see: https://onlinelibrary.wiley.com/doi/full/10.1002/hbm.25547 = '*Spectral analysis of physiological brain pulsations affecting the BOLD signal*'
For fALFF signals in the seleected frequency band are normalized to the sum squared amplitude across the entire frequency band as defined by the sampling frequency (1/opts.TR). For more information see: doi: 10.1016/j.jneumeth.2008.04.012 - '*An improved approach to detection of amplitude of low-frequency fluctuation (ALFF) for resting-state fMRI: fractional ALFF*'

A toolbox for analyzing cerebro-vascular reactivity (CVR) data

Function usage

lagCVR: generates hemodynamic lag maps and CVR maps along with statistical maps based on 3D (x,y,t) or 4D (x,y,z,t) input data. Lag maps can be calculated using a correlation approach or a GLM based approach using shifted regressors. Maps can be generated based on the original input probe (i.e. end-tidal CO_2 or ROI signal) or based on an optimized BOLD regressor can be generated using the lagCVR function. The optimized regressor (newprobe) can be returned by the function for use in further analysis. This code is inspired by the rapidtide approach: https://rapidtide.readthedocs.io/en/latest/. Python based code for similar analysis is available: https://github.com/bbfrederick/rapidtide.

usage: [newprobe] = remLV(**GMmask**, **mask**, **data,** **probe**, **nuisance**, **opts**)

Input description

GMmask: typically a gray matter mask where signal response is expected to most highly correlate with the input probe. The optimized regressor is calculated based on these ROI voxels. and the **opts**.corr_thresh parameter outlined below.
mask: typically a whole brain mask. All voxels in this mask will be correlated/regressed against the input probe or optimized probe
data:3D (x,y,t) or 4D (x,y,z,t) time-series data
probe: this is the input probe that is typically the end-tidal CO_2 or O_2 breathing trace. The probe can also be a custom time-series (i.e. signal in a particular brain region, or block representation of the applied stimulus etc.). The probe must have the same length is the input time-series.
nuisance: when using the lagged-regressor approach (**opts**.glm_model = 1), this variable can be used to provide nuisance regressors. This can be useful when analyzing the response to O_2 to consider the nuisance signals that can be explained by variations in end-tidal CO_2.

Option description (opts) **Default values in brackets**

opts.prewhite (0)	*Transforms data to have zero mean and unit variance; normally leads to crappy results
opts.interp_factor (4)	*Factor by which to temporally interpolate (up-sample) data; increase to identify lags between TR. **NOTE** increasing interpolation increases processing time; especially for GLM based analysis.
opts.load_probe (0)	*If regressor optimization has been done in a previous run, set to '1' to saved time. Previous regressor will be loaded. **NOTE** this only works if the regressor is the correct length length already exists
opts.save_rts (0)	*Set to '1' to save correlation time-series; this can be nice for visualizing lags but will produce a large file
opts.rescale_probe (1)	*Rescales input probe between 0 and 1; rescaling may be helpful for initial refinement run
opts.trace_corr (1)	*Performs additional analysis using input trace (in addition to optimized regressor). This can be useful in cases where optimization is not favorable or helpful; i.e. correlating a resting state trace or using a BOLD signal as input regressor.
opts.refine_regressor (1)	*Produces optimized regressor (rapidtide method). If '0' a straight correlation will be done with the original input probe
opts.pca_analysis (1)	*PCA analysis to is used to produce optimized regressor. This option currently requires the statistics toolbox. If PCA is not selected, regressor will be produced using time-shifted averaging of voxels.
opts.corr_model (1)	*Performs cross correlation based analysis to produce lag maps. Setting to '0' can allow GLM only analysis using the shifted regressor approach. This is useful if a nuisance regressor is used (or several)
opts.cvr_maps (1)	*Generate CVR maps based on regression of time-series signal with input probe. Lag maps will be used to produce lag-corrected CVR maps. **NOTE** if an O_2 trace is used as input, the 'CVR' map will be an O_2 response maps.
opts.eff_probe (1)	*Generate CVR maps using an effective probe, in which the optimized regressor is transformed to have the same scaled units as the input regressor. This option provides accurate CVR maps that are less sensitive to errors/bias resulting from slowly reacting vasculature. These maps also avoid confounds related to lag-corrected CVR maps. Effective CVR maps tend to be most accurate.
opts.glm_model (0)	*perform GLM based lag regression using the shifted regressor approach. This works well for high temporal resolution data and can accept a nuisance regressor. This is useful when analyzing O_2 data that strong CO_2 signal effects.
opts.uni (0)	*If this is set to '1' negative correlations will be ignored. This can avoid shifting the regressor to the point where anti-correlations can occur; i.e. during block design experiments. **NOTE** this should not be used in cases where negative signals related to vascular steal may occur (i.e. pathological brain)
opts.norm_regr (0)	*Normalizes the regressor between 0-1 for lag analysis
opts.corr_thresh (7)	*This sets the threshold by which to accept correlated voxels during the refinement of the optimized. if this value is set too high, the refinement process can fail and will throw an error statement. Ideally this value should set as high as possible, but is dependent on the SNR of the input data. Any value lower than ~0.3 increases the chances for spurious correlations and a suboptimal regressor.
opts.lowerlagthresh (-2)	*Sets the lower threshold for accepting negative lagged voxels for regressor refinement
opts.upperlagthresh (3)	*Sets the upper threshold for accepting positive lagged voxels for regressor refinement
opts.lowlag (-2)	*Sets the low lag for whole dataset analysis; can be lowered beyond threshold to catch negative lags
opts.highlag (20)	*Sets the upper lag for whole dataset analysis; in healthy individuals max should be between 10-20 **NOTE** lag threshold and boundary parameters are in units of TR(!) and not seconds. Lag values in lag maps are expressed in seconds; i.e. are converted from TR-lag to second-lag when being saved

seeVR

A toolbox for analyzing cerebro-vascular reactivity (CVR) data

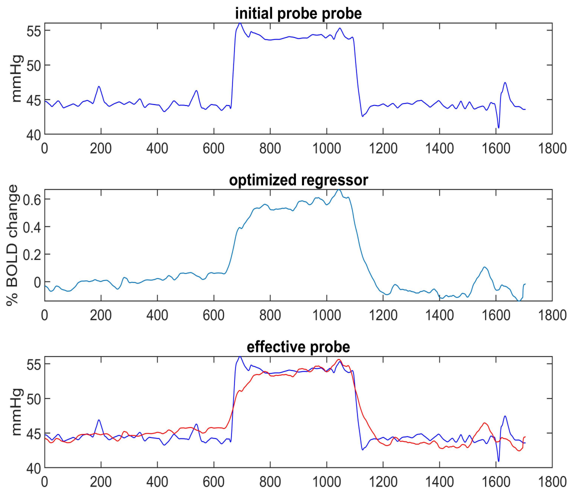
PROCESS

Depending on the options chosen, this function produces a series of hemodynamic and statistical maps outlined as follows:

if opts.corr_model = 1 in directory **/corrLAG**
lag_map.nii.gz - lag map that is shifted with respect to the lowest negative lag, corrected for TR and interpolation; units are seconds
r_map.nii.gz - correlation map (r-values) derived from cross-correlation
uncorr_lag_map.nii.gz - uncorrected lag map; i.e. including possible negative lags and expressed in units of TR*interpolation factor
if opts.trace_corr = 1 in directory **/corrLAG**
lag_map_probe.nii.gz - a corrected lag map (units of seconds) as above but produced using the original (not optimized) input probe
r_map_probe.nii.gz - correlation map (r-values) derived from cross-correlation using original input probe
uncorr_lag_map_probe.nii.gz - uncorrected lag map as above produced using original input probe
If maps are produced using both the optimized regressor and the input probe then an additional lag map (**robustLAG_r.nii.gz**) will be produced. This map used the lag value associated with the highest r-value found between both analysis methods.

If opts.cvr_maps = 1 in directory **/corrLAG/CVR**
bCVR_map.nii.gz - 'base' CVR produced by linearly regressing voxel-signals with original input probe
cCVR_map.nii.gz - 'lag-corrected' CVR map produced by linearly regressing voxel-signals with 'lag' shifted input probe
b(c)R2_map.nii.gz - map of R^2 values associated with linear regression process used for CVR and lag-corrected CVR map creation
b(c)SSE_map.nii.gz - map of SSE values associated with linear regression process used for CVR and lag-corrected CVR map creation
b(c)Tstat_map.nii.gz - map of t-statistic values associated with linear regression process used for CVR and lag-corrected CVR map creation

If opts.eff_probe = 1 additional maps are created based on the effective input probe in **/corrLAG/CVR**
b(c)CVR_eff_map.nii.gz
b(c)R2_eff_map.nii.gz
b(c)SSE_eff_map.nii.gz
b(c)Tstat_eff.nii.gz



If opts.cvr_maps =1 & opts.eff_probe = 1 & opts.trace_corr = 1
then a robust CVR map will be output. The robust CVR uses the Tstat maps to select CVR values associated with the highest t-statistic across the different analysis methods. This may be more accurate in cases where the CVR response in different voxels is better modeled by one of alternate 3 probes (input, effective, optimized). Typically highest t-statistic values are found for the effective probe and so the robust CVR map will most closely resemble the effective CVR map.

if opts.GLM_model = 1 additional maps are created in **/glmLAG**
optiReg_lags.nii.gz - lag map generated using the shifted regressor GLM
optiReg_ES.nii.gz - map of highest beta values (estimates)
optiReg_R2.nii.gz - map of R^2 values associated with the main estimate
optiReg_SSE.nii.gz - map of SSE associated with the main estimate
optiReg_Tstat.nii.gz - map of t-statistics associated with the main estimate
opteReg_estimatriz.nii.gz - a beta (estimate) time-series for all regressors
The GLM requires the the optimized regressor as input. The possibility to a nuisance parameter is included however detailed parameter estimation or statistical output associated this regressor are not currently provided. This will be updated in future releases along with the option to include several nuisance parameters. If opts.trace_corr = 1, a second series of maps are produced based on the input regressor. The prefix term 'inputReg' replaces the prefix term 'optiReg'. This functionality also produces uncorrected lag maps using each of the input and optimized regressor probes.

if opts.cvr_maps = 1 in directory **/glmLAG/CVR**
The base CVR (**bCVR_map.nii.gz**) described above uses linear regression to estimate CVR (or related parameter based on input probe). Based on results of the shifted regressor GLM, a corrected CVR map is produced along with associated statistical maps.
optiReg_cCVR.nii.gz - 'lag-corrected' CVR map produced based on the GLM lag map (**optiReg_lags.nii.gz**)
optiReg_cR2.nii.gz - map of R^2 values associated with CVR regression analysis
optiReg_cSSE.nii.gz - map of SSEvalues associated with CVR regression analysis
optiReg_cTstat.nii.gz - map of t-statistic values associated with CVR regression analysis
Similar to above, if opts.trace_corr = 1, a second set of maps will be produced based on the original input probe. These maps will have the prefix 'inputReg' instead of 'optiReg'.