Package: TCIU (via r-universe)

October 12, 2026

Title Spacekime Analytics, Time Complexity and Inferential Uncertainty

Version 1.2.8

URL https://github.com/SOCR/TCIU,
https://www.socr.umich.edu/spacekime/,
https://www.socr.umich.edu/TCIU/

BugReports https://github.com/SOCR/TCIU/issues

Description Provide the core functionality to transform longitudinal data to complex-time (kime) data using analytic and numerical techniques, visualize the original time-series and reconstructed kime-surfaces, perform model based (e.g., tensor-linear regression) and model-free classification and clustering methods in the book Dinov, ID and Velev, MV. (2021) ``Data Science: Time Complexity, Inferential Uncertainty, and Spacekime Analytics", De Gruyter STEM Series, ISBN 978-3-11-069780-3. <https://www.degruyter.com/view/title/576646>. The package includes 18 core functions which can be separated into three groups. 1) draw longitudinal data, such as Functional magnetic resonance imaging(fMRI) time-series, and forecast or transform the time-series data. 2) simulate real-valued time-series data, e.g., fMRI time-courses, detect the activated areas, report the corresponding p-values, and visualize the p-values in the 3D brain space. 3) Laplace transform and kimesurface reconstructions of the fMRI data.

Depends R (>= 3.5.0)

Imports stats, ggplot2, dplyr, tidyr, RColorBrewer, fancycut, scales, plotly, gridExtra, ggpubr, ICSNP, rrcov, geometry, DT, forecast, fmri, pracma, zoo, extraDistr, parallel, foreach, spatstat.explore, spatstat.geom, cubature, doParallel, reshape2, MultiwayRegression, interp

Suggests oro.nifti, magrittr, knitr, rmarkdown

License GPL-3

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation yes

SystemRequirements GNU make

Author Yongkai Qiu [aut], Zhe Yin [aut], Jinwen Cao [aut], Yupeng Zhang [aut], Yuyao Liu [aut], Rongqian Zhang [aut], Yueyang Shen [aut, cre], Rouben Rostamian [ctb], Ranjan Maitra [ctb], Daniel Rowe [ctb], Daniel Adrian [ctb] (gLRT method for complex-valued fMRI statistics), Yunjie Guo [aut], Ivo Dinov [aut]

Maintainer Yueyang Shen <petersyy@umich.edu>

Config/pak/sysreqs make

Repository https://cran.r-universe.dev

Date/Publication 2026-01-26 04:35:06 UTC

RemoteUrl https://github.com/cran/TCIU

RemoteRef HEAD

RemoteSha bf50a78f3fd95c68e11aa9cd252c84eec92bd8cc

Contents

fmri_2dvisual . . . 3
fmri_3dvisual . . . 4
fmri_3dvisual_region . . . 6
fmri_image . . . 8
fmri_kimesurface . . . 9
fmri_post_hoc . . . 10
fmri_pval_comparison_2d . . . 11
fmri_pval_comparison_3d . . . 13
fmri_ROI_phase1 . . . 14
fmri_ROI_phase2 . . . 16
fmri_simulate_func . . . 18
fmri_stimulus_detect . . . 19
fmri_time_series . . . 21
fmri_ts_forecast . . . 22
GaussSmoothArray . . . 23
GaussSmoothKernel . . . 23
ILT . . . 24
inv_kimesurface_transform . . . 25
kimesurface_transform . . . 27
LT . . . 28
mask . . . 29
mask_dict . . . 29

mask_label 29
phase1_pval 30
phase2_pval 30
phase3_pval 30
sample_save 31

Index **32**

`fmri_2dvisual` *visualization of the 2D brain (axial, sagittal and coronal) with the activated areas*

Description

a visualization method, using `ggplot2` to draw the brain from axial, sagittal and coronal view with activated area identified by p-values

Usage

```
fmri_2dvisual(
  pval,
  axis_ls,
  hemody_data = NULL,
  mask,
  p_threshold = 0.05,
  legend_show = TRUE,
  method = "scale_p",
  color_pal = "YlOrRd",
  multi_pranges = TRUE,
  mask_width = 1.5
)
```

Arguments

`pval`	a 3D array of p-values used to plot activated area of the brain
`axis_ls`	a list with two elements. The first element is the character of 'x', 'y', 'z'. The second element is an integer showing a specific slice on the fixed axis identified in the first element.
`hemody_data`	a parameter to have the plot with/without hemodynamic contour. The default is NULL to make the plot without hemodynamic contour, otherwise assign a 3D array of the hemodynamic data.
`mask`	a 3D nifti or 3D array of data to show the shell of the brain
`p_threshold`	NULL or a numeric value that can be selected randomly below 0.05 to drop all p-values above the threshold. If 'low5_percent' method is used, make 'p_threshold' as NULL. The default is 0.05.
`legend_show`	a logical parameter to specify whether the final plot has legend

method	a string that represents method for the plot. There are 3 options: 'min_max', 'scale_p' and 'low5_percent'. The default is 'scale_p'. 'min_max' is to draw plot based on the color scale of the minimum and maximum of the p value; 'scale_p' is to draw the plot with fixed color scale for fixed range of p value; 'low5_percent' is to draw the plot for the smallest 5 percent of p value when all the p values are not significant.
color_pal	the name of the color palettes provided by RColorBrewer. The default is "YlOrRd".
multi_pranges	an option under 'scale_p' method to decide whether there are at most 9 colors in the legend for the ranges of p value, or at most 4 colors. The default is TRUE, choosing the larger number of colors for the plot.
mask_width	a numeric value to specify the width of mask contour. The default is 1.5.

Details

The function fmri_2dvisual is used to find activated part of the brain based on given p values from sagittal, axial and coronal view. When providing input of the p-values, the specific plane and index to slice on, the mask data and the hemodynamic data of the brain, a plot will be generated with the heat map for the activated parts, the black contour showing the position of the brain, and the blue contour representing the hemodynamic contour.

Value

a plot drawn by ggplot2

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample 3D data of mask provided by the package
dim(mask)
# sample 3D p value provided by the package
dim(phase2_pval)

# plot the sagittal, coronal and axial view of this p value generated from the brain fMRI data

fmri_2dvisual(phase2_pval, list('x',40), hemody_data=NULL, mask=mask, p_threshold=0.05)
```

---

fmri_3dvisual *visualization of the 3D brain with the activated areas*

Description

a visualization method, using plotly to draw the 3D plot of the brain with the activated areas determined by p-values, which is generated from fMRI data

Usage

```
fmri_3dvisual(
  pval,
  mask,
  p_threshold = 0.05,
  method = "scale_p",
  color_pal = "YlOrRd",
  multi_pranges = TRUE,
  title = NULL
)
```

Arguments

pval	a 3D array of p-values used to plot activated area of the brain
mask	a 3D nifti or 3D array of data to show the shell of the brain
p_threshold	NULL or a numeric value that can be selected randomly below 0.05 to drop all p-values above the threshold. If 'low5_percent' method is used, make 'p_threshold' as NULL. The default is 0.05.
method	a string that represents method for the plot. There are 2 options: 'scale_p' and 'low5_percent'. The default is 'scale_p'. 'scale_p' is to draw the plot with fixed color scale for fixed range of p value. 'low5_percent' is to draw the plot for the smallest 5 percent of p value when all the p values are not significant.
color_pal	the name of the color palettes provided by RColorBrewer. The default is "YlOrRd".
multi_pranges	an option under 'scale_p' method to decide whether there are at most 9 colors in the legend for the ranges of p value, or at most 4 colors. The default is TRUE, choosing the larger number of colors for the plot.
title	the title of the plot. The default is NULL.

Details

The function fmri_3dvisual is used to visualize the 3D plot of the brain with its activated parts based on provided p values. The p values are generated by applying statistical test on fMRI data. When providing input of a 3D p-values data, a 3D interactive plot will be generated with surface for the brain shell and scatter points in different colors and size representing different stimulated levels.

Value

a list of two elements

- plot - the 3d plot of the fMRI data drawn by plotly
- pval_df - data.frame with the p value for each voxel and the specified color for it

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample 3D data of mask provided by the package
dim(mask)
# sample 3D p value provided by the package
dim(phase2_pval)

# make the 3D plot
fmri_3dvisual(phase2_pval, mask, p_threshold = 0.05, method="scale_p")$plot
```

`fmri_3dvisual_region` *visualization of the 3D brain with the activated areas by regions*

Description

an improved visualization method of `fmri_3dvisual`, using `plotly` to draw the 3D plot of the brain with the activated areas region by region

Usage

```
fmri_3dvisual_region(
  pval,
  mask,
  label_index,
  label_name,
  top_num = NULL,
  p_threshold = 0.05,
  method = "scale_p",
  multi_pranges = TRUE,
  color_pal = "YlOrRd",
  rank = NULL,
  title = NULL
)
```

Arguments

`pval`	a 3D or 1D or a list of two 3D array of p-values used to plot activated area of the brain
`mask`	a 3D nifti or 3D array of data to show the regions of the brain
`label_index`	a 1D array listing the label number in the mask
`label_name`	a 1D array corresponding to the name of the label number in the mask
`top_num`	NULL or a numeric value that used for 1D p-values. If specified, the output will show the top num significant regions. The default is NULL.
`p_threshold`	NULL or a numeric value that used for 3D p-values can be selected randomly below 0.05 to drop all p-values above the threshold. If 'low5_percent' method is used, make 'p_threshold' as NULL. The default is 0.05.

method	a string that represents method for the 3D p-values plot. There are 2 options: 'scale_p' and 'low5_percent'. The default is 'scale_p'. 'scale_p' is to draw the plot with fixed color scale for fixed range of p value. 'low5_percent' is to draw the plot for the smallest 5 percent of p value when all the p values are not significant.
multi_pranges	an option under 'scale_p' method to decide whether there are at most 9 colors in the legend for the ranges of 3D p-values, or at most 4 colors. The default is TRUE, choosing the larger number of colors for the plot.
color_pal	the name of the color palettes provided by RColorBrewer. The default is "YlOrRd".
rank	the method that how the trace is ranked. The default is NULL. There are 2 options: 'value' and a vector. 'value' is to draw the 1D p-values by the values from smallest to largest. a vector is to specific the rank of the regions in 3D p-values plot.
title	the title of the plot. The default is NULL.

Details

The function fmri_3dvisual_region is used to visualize the 3D plot of the brain with activated parts region by region. When providing a 1D/3D p-values data, a 3D interactive plot with surface of the brain shell will be generated with either scatter points representing different stimulated levels or large color pieces representing different regions of the brain. When providing a list of two 3D array of p-values, two 3D interactive brains with different scatter points corresponding to the two input 3D p-values will be given.

Value

the 3d plot of the fMRI data drawn by plotly

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample label vector provided in the package
label_index = mask_dict$index
label_name = as.character(mask_dict$name)
label_mask = mask_label

fmri_3dvisual_region(phase1_pval, label_mask, label_index,
                     label_name, title = "phase1 p-values")
fmri_3dvisual_region(phase1_pval, label_mask, label_index,
                     label_name, 5, title = "phase1 top five p-values", rank = "value")

# for 3D visualization, user needs to include empty region in the label
label_index = c(0, label_index)
label_name = c("empty", label_name)
fmri_3dvisual_region(phase2_pval, label_mask, label_index,
                   label_name, title = "phase2 p-values", rank = c(1:length(label_name)))
```

```
fmri_3dvisual_region(list(phase2_pval,phase3_pval), label_mask, label_index,
                     label_name, title = "phase2&3 p-values")
```

`fmri_image` *interactive graph object of the fMRI image*

Description

fMRI image visualization method, based on package `plotly`.

Usage

```
fmri_image(fmridata, option = "manually", voxel_location = NULL, time = NULL)
```

Arguments

`fmridata`	a 4D array contains information for the fMRI spacetime image. The data should only contain the magnitude for the fMRI image.
`option`	The default is 'manually'. If choose 'auto', then this function will lead you to key in the space (x,y,z) parameters and time (time) parameter for this function to generate graphs.
`voxel_location`	a 3D array indicating the spatial location of the brain. If option is auto, set the voxel_location as NULL.
`time`	time location for the voxel

Details

The function `fmri_image` is used to create images for front view, side view, and top view of the fMRI image. When providing the 4D array of the fMRI spacetime image and input the x,y,z position of the voxel, three views of the fMRI image and the time series image of the voxel will be shown.

Value

an interactive graph object of the fMRI image created by `plotly`

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
fmri_generate = fmri_simulate_func(dim_data = c(64, 64, 40), mask = mask)
fmri_image(fmri_generate$fmri_data, option='manually', voxel_location = c(40,22,33), time = 4)
```

`fmri_kimesurface` *interactive graph object of 3D kime-series*

Description

Use `plotly` to display in 3D the kime-series as 2D manifolds (kimesurface) over the cartesian domain.

Usage

```
fmri_kimesurface(fmridata, voxel_location = NULL, is.4d = TRUE)
```

Arguments

`fmridata`	a 4d array which contains the spatial and temporal record of fMRI result or a single real valued vector.
`voxel_location`	a 3d array indicating the spatial location of the brain.
`is.4d`	The default is true. If change to false, need to input a vector instead of array.

Details

The function `fmri_kimesurface` is display in 3D the kime-series as 2D manifolds (kimesurface) over the Cartesian domain. It helps transform the fMRI time-series data at a fixed voxel location into a kimesurface (kime-series). User can choose to provide the 4D array of the fMRI spacetime image and the voxel_location or a single time-series vector, then a 3D visualization will be shown.

Value

an interactive plot in 3D kimesurface

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample fMRI time-series vector of a single voxel
sample_voxel = sample_save[[9]]
## Not run:
fmri_kimesurface(sample_voxel, is.4d = FALSE)[[1]]
fmri_kimesurface(sample_voxel, is.4d = FALSE)[[2]]
fmri_kimesurface(sample_voxel, is.4d = FALSE)[[3]]
fmri_kimesurface(sample_voxel, is.4d = FALSE)[[4]]

## End(Not run)
```

`fmri_post_hoc` *post-hoc process for p values*

Description

This function is used to conduct the post-hoc process (i.e. FDR correction and spatial clustering) for a 3-dimensional p-value array.

Usage

```
fmri_post_hoc(
  p_val_3d,
  fdr_corr = NULL,
  spatial_cluster.thr = NULL,
  spatial_cluster.size = NULL,
  show_comparison = FALSE,
  ...
)
```

Arguments

`p_val_3d`	an array which contains the p-values as the result of fMRI statistical tests.
`fdr_corr`	The default is NULL. Input 'fdr' to conduct FDR correction.
`spatial_cluster.thr`	The default is NULL. Together with spatial_cluster.size are used to filter contiguous clusters of locations in a 3D array that are below some threshold and with some minimum size.
`spatial_cluster.size`	The default is NULL. The size of spatial cluster.
`show_comparison`	The default is FALSE. If TRUE, the output would display the comparison between raw and processed p-values.
`...`	One can specify breaks etc. to modify the comparison histogram in ggplot2.

Details

The function `fmri_post_hoc` would help do the FDR correction and spatial clustering for a 3d p-value array. The FDR correction controls for a low proportion of false positives, while the spatial clustering part help filter out all sparse p-values that are not in specified clusters.

Value

3D p-values after FDR correction or spatial clustering

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample 3D p value provided by the package
dim(phase2_pval)

# do the FDR correction
pval_fdr = fmri_post_hoc(phase2_pval,
                         fdr_corr = 'fdr',
                         spatial_cluster.thr = NULL,
                         spatial_cluster.size = NULL,
                         show_comparison = FALSE)

# do the spatial clustering
pval_posthoc = fmri_post_hoc(pval_fdr,
                             fdr_corr = NULL,
                             spatial_cluster.thr = 0.05,
                             spatial_cluster.size = 5,
                             show_comparison = FALSE)
```

`fmri_pval_comparison_2d` *2D comparison visualization between the p-values*

Description

a plot arrangement method, which uses `gridExtra` to combine multiple 2D plots of the fMRI data together. This can bring convenience for users to compare the result of different statistical tests based on the p values they provide

Usage

```
fmri_pval_comparison_2d(
  pval_ls,
  pval_name_ls,
  axis_i_lses,
  hemody_data = NULL,
  mask,
  p_threshold = 0.05,
  legend_show = TRUE,
  method = "scale_p",
  color_pal = "YlOrRd",
  multi_pranges = TRUE,
  mask_width = 1.5
)
```

Arguments

pval_ls	a list. Each element is a 3D array of p-values data
pval_name_ls	a list with the element as name for the p-values data provided in 'pval_ls'
axis_i_lses	a list with 3 numeric elements or a list of lists. If the elements are numeric, they would specify indices of slice for the three direction. If any direction of the slice need not to be shown, make it as NULL for that element. If elements are lists, each list provides specified cuts for corresponding 3D p-values data.
hemody_data	a parameter to have the plot with/without hemodynamic contour. The default is NULL to make the plot without hemodynamic contour, otherwise assign a 3D array of the hemodynamic data.
mask	a 3D nifti or 3D array of data to show the shell of the brain
p_threshold	NULL or a numeric value that can be selected randomly below 0.05 to drop all p-values above the threshold. If 'low5_percent' method is used, make 'p_threshold' as NULL. The default is 0.05.
legend_show	a logical parameter to specify whether the final plot has legend for all the subplots or the shared legend for all the subplots. The default is TRUE.
method	a string that represents method for the plot. There are 3 options: 'min_max', 'scale_p' and 'low5_percent'. The default is 'scale_p'. 'min_max' is to draw plot based on the color scale of the minimum and maximum of the p-values; 'scale_p' is to draw the plot with fixed color scale for fixed range of p-values; 'low5_percent' is to draw the plot for the smallest 5 percent of p-values when all the p-values are not significant
color_pal	the name of the color palettes provided by RColorBrewer The default is "YlOrRd".
multi_pranges	an option under 'scale_p' method to decide whether there are at most 9 colors in the legend for the ranges of p-values, or at most 4 colors. The default is TRUE, choosing the larger number of colors for the plot.
mask_width	a numeric value to specify the width of mask contour. The default is 1.5.

Details

The function fmri_pval_comparison_2d is used to combine and compare the 2D plots for different 3D arrays of p-values. The plots in each row are generated by one specific 3D p value data. The first column of the integrated plot specifies the name of the 3D p value data (for generation of the plots in that row). The rest of the three columns are the plots from sagittal, coronal and axial view for each 3D p value data.

Value

a combination plot arranged by gridExtra

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample 3D data of mask provided by the package
dim(mask)
# sample 3D p value provided by the package
dim(phase2_pval)
dim(phase3_pval)


fmri_pval_comparison_2d(list(phase2_pval, phase3_pval),
                        list('phase2_pval', 'phase3_pval'),
                        list(list(40, 26, 33), list(40, 26, 33)),
                        hemody_data = NULL,
                        mask = mask, p_threshold = 0.05,
                        legend_show = FALSE, method = 'scale_p',
                        color_pal = "YlOrRd", multi_pranges=TRUE)
```

---

fmri_pval_comparison_3d

comparison between 3d visualization for p-values

Description

a visualization method, use `plotly` to compare the activated parts inside the brain, using two sets of color palettes. The activated parts are localized with different p values.

Usage

```
fmri_pval_comparison_3d(
  pval_3d_ls,
  mask,
  p_threshold,
  method_ls,
  color_pal_ls = list("YlOrRd", "YlGnBu"),
  multi_pranges = TRUE
)
```

Arguments

`pval_3d_ls`	a list of two element, each element is a 3D array of p-values used to plot activated area of the brain
`mask`	a 3D nifti or 3D array of data to show the shell of the brain
`p_threshold`	NULL or a numeric value that can be selected randomly below 0.05 to drop insignificant p-values of no need or drop no p-values. If 'low5_percent' method is used, make 'p_threshold' as NULL. The default is 0.05.

method_ls	a string that represents method for the plot. There are 2 options: 'scale_p' and 'low5_percent'. The default is 'scale_p'. 'scale_p' is to draw the plot with fixed color scale for fixed range of p value; 'low5_percent' is to draw the plot for the smallest 5 percent of p value when all the p values are not significant
color_pal_ls	a list of two element. Each element is the name of the color palettes provided by RColorBrewer. The default is list('YlOrRd', 'YlGnBu').
multi_pranges	an option under 'scale_p' method to decide whether there are at most 9 colors in the legend for the ranges of p value, or at most 4 colors. The default is TRUE, choosing the larger number of colors for the plot.

Details

The function fmri_pval_comparison_3d is used to visualize and compare the 3D plots of the activated parts in one brain shell. The activated parts are plotted based on p-values provided. Note that this comparison can only be made when the masks of the two p values are the same. When providing input of two set of the 3D array of p-values, corresponding p threshold for each p value data, and the method to draw the plot, the plot will be generated with one brain shell and two groups of activated parts in two sets of color palettes. The size and color of the scatter points represent different stimulated levels of the activated parts.

Value

a plot drawn by plotly

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample 3D data of mask provided by the package
dim(mask)
# sample 3D p value provided by the package
dim(phase2_pval)
dim(phase3_pval)

fmri_pval_comparison_3d(list(phase2_pval, phase3_pval), mask,
                        list(0.05, 0.05), list('scale_p', 'scale_p'), multi_pranges=FALSE)
```

---

fmri_ROI_phase1 *p-values on region of interest(ROI) of the brain*

Description

This function takes a 4 dimensional real-valued fMRI data and calculates p-values for the ROIs individually to test whether the ROI is potentially activated. It is the first phase of a ROI 3-phase analysis and usually followed by second phase analysis fmri_ROI_phase2 .

Usage

```
fmri_ROI_phase1(
  fmridata,
  label_mask = NULL,
  label_dict = NULL,
  stimulus_idx = NULL,
  rest_idx = NULL,
  p_threshold = 0.05
)
```

Arguments

fmridata	a 4d array which contains the spatial and temporal record of fmri data
label_mask	a 3D nifti or 3D array of data to indicates the corresponding indices of the ROIs
label_dict	a dataframe which contains the name of ROIs and their corresponding index
stimulus_idx	a vector that specifies when motion happens
rest_idx	a vector that specifies when study participant does not move
p_threshold	NULL or a numeric value that can be selected randomly below 0.05 to drop all p-values above the threshold.

Details

The function fmri_ROI_phase1 is used to calculate p-values of ROIs for a given real-valued fmridata. It first takes in the fmridata and corresponding mask. For a fixed region, the function will first compute Temporal Contrast-to-noise Ratio (tCNR) for each voxel in that region, which is the mean of 80 paired differences in intensity for "on" and "off" states divided by its standard deviation. Second, it will conduct t-test on all tCNRs of a fixed region to see there are significant changes for the ROI during the on and off period. Finally, it will use bonferroni correction to control significant level and select the ROIs with p-values under the significant level to enter next phase analysis.

Value

a list of two elements

- all_ROI - the test result for all ROIs
- sign_ROI - the test result for significant ROIs

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
fmri_generate = fmri_simulate_func(dim_data = c(64, 64, 40), mask = mask)
# p-values for phase 1

result = fmri_ROI_phase1(fmri_generate$fmri_data, mask_label,
                         mask_dict, stimulus_idx = fmri_generate$on_time)
```

`fmri_ROI_phase2` *tensor-on-tensor regression on region of interest(ROI) of the brain*

Description

This function takes a 4d fMRI data and detects locations where stimulus occurs on each region of interest(ROI) of the brain using `MultiwayRegression`. This function could be used as an intermediate step of a three-phase analytics protocol to detect motor areas. The functions to implement this three-phase protocol in a consecutive order is `fmri_ROI_phase2`, `fmri_ROI_phase3` and `fmri_post_hoc` respectively.

Usage

```
fmri_ROI_phase2(
  fmridata,
  label_mask,
  label_dict,
  stimulus_idx,
  stimulus_dur,
  fmri.design_order = 2,
  fmri.stimulus_TR = 3,
  rrr_rank = 3,
  method = "t_test",
  parallel_computing = FALSE,
  ncor = max(detectCores() - 2, 1)
)
```

Arguments

`fmridata`	a 4d array which contains the spatial and temporal record of fmri result.
`label_mask`	a 3d nifti or 3d array of data that shows the labeled brain atlas.
`label_dict`	a dataframe or array or matrix to specify the indices and corresponding names of the ROI. The input of this parameter could take one of the list outputs of the `fmri_ROI_phase2` function as a following step.
`stimulus_idx`	a vector of the start time points of the time period when the fMRI data receives stimulation.
`stimulus_dur`	a vector of the time period when the fMRI data receives stimulation.
`fmri.design_order`	a parameter to specify the order of the polynomial drift terms in `fmri.design` function.
`fmri.stimulus_TR`	a parameter to specify the time between scans in seconds in `fmri.stimulus` function.
`rrr_rank`	a parameter to specify the assumed rank of the coefficient array in `rrr` function.

`method` a string that represents method for calculating p-values from tensor-on-tensor regression coefficients. There are 2 options: 't_test' and 'corrected_t_test'. The default is 't_test'. 't_test' is to calculate the test statistics 't-value' across all voxels in the bounding box of ROI; 'corrected_t_test' is to calculate the test statistics 't-value' by first across each voxel on a temporal basis, and then across all voxels in the bounding box of ROI.

`parallel_computing` a logical parameter to determine whether to use parallel computing to speed up the function or not. The default is FALSE.

`ncor` number of cores for parallel computing. The default is the number of cores of the computer minus 2.

Details

The function `fmri_ROI_phase2` is used to detect locations where stimulus occurs by calculating the p-values of the ROI-based tensor-on-tensor regression. Two methods can be chosen to calculate the p-values from the regression coefficients.

Value

a 3d array storing ROI-based tensor regression p-values for the 4d fMRI data

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample 3D data of labeled brain atlas provided by the package
# this example will use parallel computing and take about ten minutes to finish
dim(mask_label)
# sample dataframe of ROI-based indices and names provided by the package
dim(mask_dict)
# sample 3D data of mask provided by the package
dim(mask)

# calculated p-values
set.seed(1)
fmri_generate = fmri_simulate_func(dim_data = c(64, 64, 40), mask = mask)
fmridata = fmri_generate$fmri_data
stimulus_idx = fmri_generate$ons
stimulus_dur = fmri_generate$dur
# the function will may take a long time, see examples in demo function or vignettes
```

`fmri_simulate_func` *real-valued fMRI data simulation*

Description

a real-valued fMRI data simulation function, used to simply generate a 3D fMRI data associated with brain area with activated parts inside.

Usage

```
fmri_simulate_func(
  dim_data,
  mask = NULL,
  ons = c(1, 21, 41, 61, 81, 101, 121, 141),
  dur = c(10, 10, 10, 10, 10, 10, 10, 10)
)
```

Arguments

`dim_data`	a vector of length 3 to identify the dimension of fMRI data user wants to simulate
`mask`	a 3D array of 1's and 0's or NULL. To specify the area inside the brain shell. One may use the mask data provided by this package, or generate a 3D array of 1's and 0's of the same dimension with the fMRI data to be generated. If NULL, then the function would generate a 3D sphere mask.
`ons`	a vector of the start time points of the time period when the fMRI data receives stimulation
`dur`	a vector of the time period when the fMRI data receives stimulation

Details

The function `fmri_simulate_func` is used to simulate fMRI data with specified dimension and total time points. The fMRI data can be brain-shaped by using the mask data provided in our package, if the dimension fits the same as our data (c(64, 64. 40)). Otherwise, the function will generate a 3D sphere data with multiple activated part inside. The activated parts can be detected based on the p values.

Value

an array with the specified dimension

a list of four elements

- fmri_data - the fMRI data generated by the function as specialized values.
- mask - mask of the fMRI data.
- ons - a vector of the start time points of the time period when the fMRI data receives stimulation.

- dur - a vector of the time period when the fMRI data receives stimulation. Notice that the length of ons is equal to the length of dur, and all the time period when the data does not receive the simulations have the same duration as its former 'on' time period.
- on_time - a vector that specifies when motion happens.

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# sample 3D data of mask provided by the package
dim(mask)

# the input dimension is the dimension we want for our simulated fMRI data
fmri_generate = fmri_simulate_func(dim_data = c(64, 64, 40), mask = mask,
                                   ons = c(1, 21, 41, 61, 81, 101, 121, 141),
                                   dur = c(10, 10, 10, 10, 10, 10, 10, 10))
```

---

fmri_stimulus_detect *fMRI data stimulus detection*

Description

This function takes a real/complex valued fMRI data and detects locations where stimulus occurs

Usage

```
fmri_stimulus_detect(
  fmridata,
  mask = NULL,
  stimulus_idx = NULL,
  rest_idex = NULL,
  method,
  fdr_corr = NULL,
  spatial_cluster.thr = NULL,
  spatial_cluster.size = NULL,
  ons = NULL,
  dur = NULL
)
```

Arguments

fmridata	an array or a vector which contains the spatial and/or temporal record of fMRI result
mask	a 3d array indicating the spatial location of the brain

stimulus_idx	a vector that specifies when motion happens
rest_idex	a vector that specifies when study participant does not move
method	a string that indicates which testing method is to be used. There are 5 options: 'HotellingT2', 'Wilks-Lambda' and 'gLRT'(likelihood ratio test) for complex fMRI data and 't-test', 'wilcoxon-test' for real fMRI data. For 4D real-valued fMRI data, two more options: 'on_off_diff' and 'HRF' method can be applied.
fdr_corr	a logical variable. True if FDR correction is to be applied
spatial_cluster.thr	threshold p-value to be used for spatial clustering
spatial_cluster.size	number of spatially connected voxels to be tested for spatial clustering
ons	a vector with the first time points of the time periods when the fMRI data receives stimulation. The default is NULL. Need to specify when choose the method 'gLRT' or 'HRF'.
dur	a vector of the time length of each stimulated period. The default is NULL. Need to specify when choose the method 'gLRT' or 'HRF'.

Details

The function fmri_stimulus_detect is used to conduct motor area detection. It first takes in a real or complex valued fMRI data, and then users can choose to use various methods to find the spatial regions where motor area is located inside the brain. User can either input the 4d fMRI data and get a 3d array storing p-values or input the fMRI data with smaller dimension (e.g. fix the x,y axis) and get a vector storing p-values. Besides, one can use this function to just calculate raw p-values, and we also provide options so that users can do FDR correction and spatial clustering to get a more accurate result.

Value

If input fMRI data is 4d, return a 3d array storing p-values for the 4d fMRI data. If input fMRI data is less than 4d, return a vector storing p-values for the fMRI data.

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
fmri_generate = fmri_simulate_func(dim_data = c(64, 64, 40), mask = mask)
fmridata = fmri_generate$fmri_data
stimulus_idx = fmri_generate$on_time
ons = fmri_generate$ons
dur = fmri_generate$dur


# p-values using t-test for 4d fMRI data
p_value1 = fmri_stimulus_detect(fmridata = fmridata, mask = mask,
                                stimulus_idx = stimulus_idx,
```

```
                                  method = 't-test')
dim(fmridata)
dim(p_value1)

# p-values using t-test for 2d fMRI data
p_value2 = fmri_stimulus_detect(fmridata = fmridata[40,41,,], mask = mask,
                                  stimulus_idx = stimulus_idx,
                                  method = 't-test')
dim(fmridata[40,41,,])
dim(p_value2)
```

`fmri_time_series` *visualization of the fMRI data (real, imaginary, magnitude, and phase parts) in time series*

Description

a visualization method, use `plotly` to draw the fMRI data in time series

Usage

```
fmri_time_series(fmridata, voxel_location, is.4d = TRUE, ref = NULL)
```

Arguments

`fmridata`	a 4d array which contains the spatial and temporal record of fMRI result or a single complex valued vector
`voxel_location`	a 3d array indicating the spatial location of the brain. If is.4d is false, set the voxel_location as NULL.
`is.4d`	The default is TRUE. If change to false, input a vector instead of a 4d array.
`ref`	The default is NULL. User can input an outside extra reference `plotly` object to include in the final result.

Details

The function `fmri_time_series` is used to create four interactive time series graphs for the real, imaginary, magnitude, and phase parts for the fMRI spacetime data. User can choose to provide the 4d array of the fMRI spacetime image and the voxel_location or a single complex valued vector, then four interactive time series graphs will be shown. Besides, the reference `plotly` object can be added to the final result.

Value

an interactive time series graph object created by `plotly`

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# load sample time-series data of one voxel in the brain provided by the package
sample_voxel = sample_save[[9]]
reference_plot = sample_save[[8]]
fmri_time_series(sample_voxel, voxel_location = NULL, is.4d = FALSE, ref = reference_plot)
```

---

fmri_ts_forecast *forecast the fMRI data based on the time series*

Description

a function to forecast the fMRI data based on the time series

Usage

```
fmri_ts_forecast(fmridata, voxel_location, cut = 10)
```

Arguments

fmridata	a 4D array contains information for the fMRI spacetime image. The data should only contain the magnitude for the fMRI image.
voxel_location	a 3d array indicating the voxel location of the brain
cut	breaking point of the time-series data. The default is 10.

Details

The function `fmri_ts_forecast` is used to forecast with time series. It will fit the best ARIMA model to univariate time series from the input fMRI data.

Value

a figure forecasting the fMRI voxel with time series

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
fmri_generate = fmri_simulate_func(dim_data = c(64, 64, 40), mask = mask)

smoothmod <- GaussSmoothArray(fmri_generate$fmri_data, sigma = diag(3,3))
fmri_ts_forecast(smoothmod,c(41,44,33))
```

GaussSmoothArray *GaussSmoothArray*

Description

An internal function named GaussSmoothArray. Original from AnalyzeFMRI package

Usage

```
GaussSmoothArray(
  x,
  voxdim = c(1, 1, 1),
  ksize = 5,
  sigma = diag(3, 3),
  mask = NULL,
  var.norm = FALSE
)
```

Arguments

x	The array to be smoothed.
voxdim	The dimensions of the volume elements (voxel) that make up the array.
ksize	The dimensions (in number of voxels) of the 3D discrete smoothing kernel used to smooth the array.
sigma	The covariance matrix of the 3D Gaussian smoothing kernel. This matrix doesn't have to be non-singular; zero on the diagonal of sigma indicate no smoothing in that direction.
mask	A 3D 0-1 mask that delimits where the smoothing occurs.
var.norm	Logical flag indicating whether to normalize the variance of the smoothed array.

Value

an array with the size of parameter x

GaussSmoothKernel *GaussSmoothKernel*

Description

An internal function named GaussSmoothKernel. Original from AnalyzeFMRI package

Usage

```
GaussSmoothKernel(voxdim = c(1, 1, 1), ksize = 5, sigma = diag(3, 3))
```

Arguments

voxdim	Dimensions of each voxel.
ksize	Dimensions of the discrete kernel size.
sigma	The covariance matrix of the Gaussian kernel.

Value

a 3 dimensional array with size = (ksize, ksize, ksize)

ILT *numerical method to compute inverse of Laplace Transform*

Description

a function that numerically computes the inverse of Laplace Transform

Usage

```
ILT(
  FUNCT,
  t,
  nterms = 31L,
  m = 1,
  gamma = 0.5,
  fail_val = complex(0),
  msg = TRUE
)
```

Arguments

FUNCT	function object F(z), typically a Laplace Transform of a function f(t)
t	time domain value to evaluate the ILT(F)(t)
nterms	number of terms to use in the numerical inversion (odd number). The default is 31L.
m	width of the contour path in C; too small values may lead to singularities on the negative x-axis; too large valued may lead to numerical instability for large positive x-axis. The default is 1.
gamma	value on the positive x-axis for the vertical line representing the contour. The default is 0.5
fail_val	value to return in event of failure to converge
msg	Boolean to show/hide warnings. The default is TRUE.

Details

This function first uses full optimum contour path to do inverse Laplace Transform. However, if this method fails, the function will automatically change to the method of using Bromwich contour path to do inverse Laplace Transform

Value

a real value computed from inverse Laplace Transform

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# analytic form of Laplace transform of f(t) = t
F = function(z) { 1/(z^2) }
# do inverse Laplace transform on t = 0.2
ILT(F, t = 0.2)
# the result is equal to t = 0.2
```

---

inv_kimesurface_transform

inverse kimesurface transform on a function in different periodic ranges

---

Description

This function applies the inverse kimesurface transform to convert a kimesurface-transformed function back to get the original 1D function in [0, 2*pi] or other similar periodic time range.

Usage

```
inv_kimesurface_transform(
  time_points,
  array_2d,
  num_length = 20,
  m = 1,
  msg = TRUE
)
```

Arguments

time_points	a sequence of points in [0, 2*pi] or other periodic range
array_2d	2D array, got from the kimesurface_transform
num_length	integer, interpolate f(t) to num_length samples in [0 : 2*pi] to extend the plot

m	width of the contour path in C; too small values may lead to singularities on the negative x-axis; too large valued may lead to numerical instability for large positive x-axis. The default is 1.
msg	Boolean to show/hide warnings. The default is TRUE.

Value

a list of two elements

- Smooth_Reconstruction - the smoothed data computed from inverse kimesurface transform, with the same length of time_points
- Raw_Reconstruction - the original unsmoothed data computed from inverse kimesurface transform, with the same length of time_points

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
require(reshape2)
require(ggplot2)

# drop the first row and first column because of divergence on Laplace Transform
x = seq(0, 2, length.out=50)[2:50]; y = seq(0, 2, length.out=50)[2:50];
# do kimesurface transform on sine function
z2_grid = kimesurface_transform(FUNCT = function(t) { sin(t) },
                                real_x = x, img_y = y)

time_points = seq(0+0.001, 2*pi, length.out = 160)
inv_data = inv_kimesurface_transform(time_points, z2_grid)
time_Intensities_ILT_df2 <- as.data.frame(cbind(Re=scale(Re(inv_data$Smooth_Reconstruction)),
                                                Im=scale(Re(inv_data$Raw_Reconstruction)),
                                                  fMRI=scale(Re(sin(time_points))),
                                                  time_points=time_points))
colnames(time_Intensities_ILT_df2) = c("Smooth Reconstruction",
                                       "Raw Reconstruction",
                                       "Original sin()",
                                       "time_points")
df = reshape2::melt(time_Intensities_ILT_df2, id.var = "time_points")
ggplot(df, aes(x = time_points, y = value, colour = variable)) +
       geom_line(linetype=1, lwd=3) +
       ylab("Function Intensity") + xlab("Time") +
       theme(legend.position="top")+
       labs(title= bquote("Comparison between" ~ "f(t)=sin(t)" ~ "
       and Smooth(ILT(LT(fMRI)))(t); Range [" ~ 0 ~":"~ 2*pi~"]"))
```

kimesurface_transform *kimesurface transform on a function with a specified set of complex values*

Description

a function applies the kimesurface transform on a function with a specified set of complex values

Usage

```
kimesurface_transform(
  FUNCT,
  glb_para,
  real_x,
  img_y,
  parallel_computing = FALSE,
  ncor = 6
)
```

Arguments

FUNCT	function object f(t) to conduct kimesurface transform on
glb_para	a vector of global objections that needed to be imported when using parallel computing
real_x	a list of numeric values, which is the real part of a set of complex values
img_y	a list of numeric values, which is the imaginary part of the set of complex values stated above
parallel_computing	logical object to determine whether to use parallel computing to speed up the function or not. The default is FALSE.
ncor	number of cores for parallel computing. The default is 6.

Details

This function applies the kimesurface transform on a 1D function f(t), to have it converted to a 2D function. The input is a set of complex values with the same number of real and imaginary parts. These two parts can specify a 2D plane of the same length and width. The new 2D function is defined on this 2D plane. It mainly does a Laplace Transform and modifies all the function values in a specific way to have them looks better in the plot.

Value

a 2d array that did kimesurface transform for the set of complex value (the real and imaginary parts can construct a 2d plane)

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
# drop the first row and first column because of divergence on Laplace Transform
# do kimesurface transform on sine function
x = seq(0, 2, length.out=50)[2:50]; y = seq(0, 2, length.out=50)[2:50];


kimesurface_transform(FUNCT = function(t) {sin(t)}, real_x = x, img_y = y);
```

LT *numerical method to compute Laplace Transform*

Description

a function that numerically computes the Laplace Transform

Usage

```
LT(FUNCT, z)
```

Arguments

FUNCT	a function object f(t) conducting Laplace Transform
z	a complex domain value used to evaluate the F(z)=LT(f)(z)

Value

a complex value computed from Laplace Transform

Author(s)

SOCR team <http://socr.umich.edu/people/>

Examples

```
f = function(t) { t }; z= 1+1i;
LT(f, z);
# compare with the result from analytic form of Laplace Transform of f(t) = t
# analytic form is below
F = function (z) { 1/z^2 }; F(z)
# the two results are the same
```

mask *mask*

Description

a 64*64*40 3D array representing brain mask

Usage

```
mask
```

Format

a 3D array containing brain mask data

mask_dict *mask_dict*

Description

a data.frame containing the label index corresponding to its label name

Usage

```
mask_dict
```

Format

a data.frame containing the label index corresponding to its label name

mask_label *mask_label*

Description

a 64*64*40 3D array representing brain mask with labels

Usage

```
mask_label
```

Format

a 3D array containing brain mask data

`phase1_pval` *phase1_pval*

Description

a 64*64*40 3D array containing a sample p values for the first phase of three-phase ROI analysis by function fmri_ROI_phase1

Usage

```
phase1_pval
```

Format

a 3D array containing p values

`phase2_pval` *phase2_pval*

Description

a 64*64*40 3D array containing a sample p values for the second phase of three-phase ROI analysis by function fmri_ROI_phase2

Usage

```
phase2_pval
```

Format

a 3D array containing p values

`phase3_pval` *phase3_pval*

Description

a 64*64*40 3D array containing a sample p values for the third phase of three-phase ROI analysis, generated by the post-hoc process for phase2_pval

Usage

```
phase3_pval
```

Format

a 3D array containing p values

sample_save *sample_save*

Description

a list containing some pre-calculated data for generating vignettes

Usage

```
sample_save
```

Format

a list containing some pre-calculated data for generating vignettes

Index

∗ **array**
    mask, 29
    mask_label, 29
    phase1_pval, 30
    phase2_pval, 30
    phase3_pval, 30
∗ **data.frame**
    mask_dict, 29
∗ **list**
    sample_save, 31

fmri_2dvisual, 3
fmri_3dvisual, 4
fmri_3dvisual_region, 6
fmri_image, 8
fmri_kimesurface, 9
fmri_post_hoc, 10
fmri_pval_comparison_2d, 11
fmri_pval_comparison_3d, 13
fmri_ROI_phase1, 14
fmri_ROI_phase2, 16
fmri_simulate_func, 18
fmri_stimulus_detect, 19
fmri_time_series, 21
fmri_ts_forecast, 22

GaussSmoothArray, 23
GaussSmoothKernel, 23

ILT, 24
inv_kimesurface_transform, 25

kimesurface_transform, 27

LT, 28

mask, 29
mask_dict, 29
mask_label, 29

phase1_pval, 30
phase2_pval, 30
phase3_pval, 30

sample_save, 31