Package ‘fMRItools’

May 30, 2024

Type Package

Title Routines for Common fMRI Processing Tasks

Version 0.4.7

Maintainer Amanda Mejia <mandy.mejia@gmail.com>

Description Supports fMRI (functional magnetic resonance imaging) analysis tasks including reading in 'CIFTI', 'GIFTI' and 'NIFTI' data, temporal filtering, nuisance regression, and aCompCor (anatomical Components Correction) (Muschelli et al. (2014) <doi:10.1016/j.neuroimage.2014.03.028>).

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

Imports stats, matrixStats

Suggests ciftiTools, corpcor, expm, gifti, knitr, rmarkdown, robustbase, pesel, RNifti, oro.nifti, gsignal, testthat (>= 3.0.0), covr, fda, quantreg, graphics, grDevices

RoxygenNote 7.3.1

URL https://github.com/mandymejia/fMRItools

BugReports https://github.com/mandymejia/fMRItools/issues

NeedsCompilation no

Author Amanda Mejia [aut, cre],
Damon Pham [aut] (<https://orcid.org/0000-0001-7563-4727>),
Mark Fiecas [ctb]

Repository CRAN

Date/Publication 2024-05-30 05:00:02 UTC

R topics documented:

`all_binary` *All binary?*

Description

Check if a data vector or matrix is all zeroes and ones. Option to also accept logical values.

Usage

```
all_binary(x, logical_ok = TRUE)
```

Arguments

`x`	The data vector or matrix
`logical_ok`	Is a logical vector or matrix also acceptable? Default: `TRUE`.

Value

Logical. Is `x` binary data?

`all_integers` *All integers?*

Description

Check if a data vector or matrix is all integers.

Usage

```
all_integers(x)
```

Arguments

`x`	The data vector or matrix

Value

Logical. Is `x` all integers?

as.matrix_ifti *Convert CIFTI, NIFTI, or GIFTI input to $T \times V$ matrix*

Description

Convert CIFTI, NIFTI, or GIFTI input to a $T \times V$ matrix by reading it in with the corresponding package and then separating the data from the metadata. Also works with the intermediate R objects created from reading these files: `"xifti"` objects from `ciftiTools`, `"gifti"` objects from `gifti`, `"nifti"` or `"niftiExtension"` objects from `oro.nifti`, and `"niftiImage"` objects from `RNifti`.

For CIFTI files, only intents supported by `ciftiTools` are supported: `dscalar`, `dtseries`, and `dlabel`. For NIFTI file or NIFTI-intermediate R objects, the data will be vectorized/masked.

Usage

```
as.matrix_ifti(
  x,
  meta = FALSE,
  sortSub = FALSE,
  TbyV = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	The object to coerce to a matrix
`meta`	Return metadata too? Default: `FALSE`.
`sortSub`	For CIFTI format input only. Sort subcortex by labels? Default: `FALSE` (sort by array index).
`TbyV`	Return the data matrix in $T \times V$ form? Default: `TRUE`. If `FALSE`, return in $V \times T$ form instead. Using this argument may be faster than transposing after the function call.
`verbose`	Print updates? Default: `FALSE`.
`...`	If `x` is a file path, additional arguments to the function used to read in `x` can be specified here. For example, if `x` is a path to a CIFTI file, `...` might specify which `idx` and `brainstructures` to read in.

Value

If `!meta`, `x` as a matrix. If `meta`, a list of length two: the first entry is `x` as a matrix, and the second entry is the metadata of `x`.

bandstop_filter *Bandstop filter*

Description

Filter out frequencies within a given range using a Chebyshev Type II stopband. Essentially a convenience wrapper for the cheby2 function.

Usage

```
bandstop_filter(X, TR, f1, f2, Rs = 20)
```

Arguments

X	A numeric matrix, with each column being a timeseries to apply the stopband filter. For fMRI data, `X` should be `T` timepoints by `V` brain locations.
TR	The time step between adjacent rows of `x`, in seconds
f1, f2	The frequency limits for the filter, in Hz. `f1` < `f2`
Rs	The amount of attenuation of the stopband ripple, in dB

Value

The filtered data

Examples

```
if (requireNamespace("gsignal", quietly=TRUE)) {
 n_voxels = 1e4
 n_timepoints = 100
 X = cbind(arima.sim(n=100, list(ar=.6)), arima.sim(n=100, list(ar=.6)))
 Y = bandstop_filter(X, .72, .31, .43)
}
```

carpetplot *Carpetplot*

Description

Plot a matrix with `graphics::image`. For fMRI data, this is the "carpetplot" or grayplot coined by (Power, 2017). The `graphics` and `grDevices` packages are required.

Usage

```
carpetplot(
  x,
  qcut = 0.1,
  fname = NULL,
  center = TRUE,
  scale = FALSE,
  colors = "gray255",
  sortSub = TRUE,
  ...
)
```

Arguments

`x`	The $T \times V$ numeric data matrix, or a `"xifti"` object. In the plot, the T index will increase from left to right, and the V will increase from top to bottom.
`qcut`	Sets blackpoint at the `qcut` quantile, and the whitepoint at the `1-qcut` quantile. Default: `.1`. This is equivalent to setting the color range between the 10% and 90% quantiles. The quantiles are computed across the entire data matrix after any centering or scaling. Must be between 0 and .49. If `0` or `NULL` (default), do not clamp the data values.
`fname`	A `.pdf` (highly recommended) or `.png` file path to write the carpetplot to. If `NULL` (default), return the plot directly instead of writing a file.
`center, scale`	Center and scale the data? If `x` is fMRI data which has not otherwise been centered or scaled, it is recommended to center but not scale it (default).
`colors`	`"gray255"` (default) will use a grayscale color ramp from black to white. Otherwise, this should be a character vector of color names to use. Colors will be assigned from the lowest to the highest data value, after any clamping of the data values by `qcut`.
`sortSub`	If `x` is a `"xifti"` object with subcortical data, should the voxels be sorted by structure alphabetically? Default: `TRUE`.
`...`	Additional arguments to `pdf` or `png`, such as width and height.

Value

The image or `NULL, invisibly` if a file was written.

References

- Power, J. D. A simple but useful way to assess fMRI scan qualities. NeuroImage 154, 150-158 (2017).

carpetplot_stack *Stacked carpetplot*

Description

Stacks carpetplots on top of one another by rbinding the matrices.

Usage

```
carpetplot_stack(
  x_list,
  center = TRUE,
  scale = FALSE,
  qcut = 0.1,
  match_scale = TRUE,
  nsep = 0,
  ...
)
```

Arguments

`x_list`	List of data matrices
`center, scale`	Center and scale the data? If `x` is fMRI data which has not otherwise been centered or scaled, it is recommended to center but not scale it (default).
`qcut`	Sets blackpoint at the `qcut` quantile, and the whitepoint at the `1-qcut` quantile. Default: `.1`. This is equivalent to setting the color range between the 10% and 90% quantiles. The quantiles are computed across the entire data matrix after any centering or scaling. Must be between 0 and .49. If `0` or `NULL` (default), do not clamp the data values.
`match_scale`	Match the scales of the carpetplots? Default: `TRUE`.
`nsep`	Equivalent number of data locations for size of gap between carpetplots. Default: zero (no gap).
`...`	Additional arguments to `carpetplot`

Value

`NULL`, invisibly

colCenter *Center matrix columns*

Description

Efficiently center columns of a matrix. (Faster than `base::scale`.)

Usage

```
colCenter(X)
```

Arguments

`X`	The data matrix. Its columns will be centered.

Value

The centered data

color_palette *Color palette*

Description

Color palettes for fMRI data analysis tasks

Usage

```
color_palette(pal = "Beach")
```

Arguments

`pal`	`"Beach"` (default; blue to white to red), `"Sand"` (white to red), or `"Water"` (white to blue).

Value

A data.frame with two columns: `"col"` has the hex code of color, and `"val"` has the placement of colors between zero and one.

CompCor *Anatomical CompCor*

Description

The aCompCor algorithm for denoising fMRI data using noise ROIs data

Usage

```
CompCor(
  X,
  ROI_data = "infer",
  ROI_noise = NULL,
  noise_nPC = 5,
  noise_erosion = NULL,
  center = TRUE,
  scale = TRUE,
  nuisance = NULL
)
```

Arguments

`X` Wide numeric data matrix ($T observations$ by $V variables$, $T << V$). For example, if `X` represents an fMRI run, T should be the number of timepoints and V should be the number of brainordinate vertices/voxels.

Or, a 4D array or NIFTI or file path to a NIFTI (I by J by K by T observations), in which case `ROI_data` must be provided. (The vectorized data will be $T timepoints$ by $V_{in-mask} voxels$)

Or, a `ciftiTools` `"xifti"` object or a file path to a CIFTI (The vectorized data will be $T timepoints$ by $V_{left+right+sub} grayordinates$).

`ROI_data` Indicates the data ROI. Allowed arguments depend on `X`:

If `X` is a matrix, this must be a length V logical vector, where the data ROI is indicated by `TRUE` values. If `"infer"` (default), all columns of `X` will be included in the data ROI (`rep(TRUE, V)`).

If `X` is an array or NIFTI, this must be either a vector of values to expect for out-of-mask voxels in `X`, or a (file path to a) 3D NIFTI. In the latter case, each of the volume dimensions should match the first three dimensions of `X`. Voxels in the data ROI should be indicated by `TRUE` and all other voxels by `FALSE`. If `"infer"` (default), will be set to `c(0, NA, NaN)` (include all voxels which are not constant `0`, `NA`, or `NaN`).

If `X` is a `"xifti"` this must be the `brainstructures` argument to `ciftiTools::read_cifti`. If `"infer"` (default), `brainstructures` will be set to `"all"` (use both left and right cortex vertices, and subcortical voxels).

If `NULL`, the data ROI will be empty. This is useful for obtaining just the noise ROI, if the data and noise are located in separate files.

`ROI_noise` Indicates the noise ROIs for aCompCor. Should be a list where each entry corresponds to a distinct noise ROI. The names of the list should be the ROI names, e.g. `"white_matter"` and `"csf"`. The expected formats of the list entries depends on `X`:

For all types of `X`, `ROI_noise` entries can be a matrix of noise ROI data. The matrix should have T rows, with each column being a data location's timeseries.

If `X` is a matrix, entries can also indicate a noise ROI within `X`. These entries must be a length V logical vector with `TRUE` values indicating locations in `X` within that noise ROI. Since the ROIs must not overlap, the masks must be mutually exclusive with each other, and with `ROI_data`.

If `X` is an array or NIFTI, entries can also indicate a noise ROI within `X`. These entries must be a logical array or (file path to) a 3D NIFTI with the same spatial dimensions as `X`, and with `TRUE` values indicating voxels inside the noise ROI. Since the ROIs must not overlap, the masks must be mutually exclusive with each other, and with `ROI_data`.

(If `X` is a `"xifti"`, entries must be data matrices, since no grayordinate locations in `X` are appropriate noise ROIs).

`noise_nPC` The number of principal components to compute for each noise ROI. Alternatively, values between 0 and 1, in which case they will represent the minimum proportion of variance explained by the PCs used for each noise ROI. The smallest number of PCs will be used to achieve this proportion of variance explained.

Should be a list or numeric vector with the same length as `ROI_noise`. It will be matched to each ROI based on the name of each entry, or if the names are missing, the order of entries. If it is an unnamed vector, its elements will be recycled. Default: `5` (compute the top 5 PCs for each noise ROI).

`noise_erosion` The number of voxel layers to erode the noise ROIs by. Should be a list or numeric vector with the same length as `ROI_noise`. It will be matched to each ROI based on the name of each entry, or if the names are missing, the order of entries. If it is an unnamed vector, its elements will be recycled. Default: `NULL`, which will use a value of 0 (do not erode the noise ROIs). Note that noise erosion can only be performed if the noise ROIs are volumetric.

`center, scale` Center the columns of the noise ROI data by their medians, and scale by their MADs? Default: `TRUE` for both. Note that this argument affects the noise ROI data and not the data that is being cleaned with aCompCor. Centering and scaling of the data being cleaned can be done after this function call.

`nuisance` Nuisance signals to regress from each data column in addition to the noise ROI PCs. Should be a T by N numeric matrix where N represents the number of nuisance signals. To not perform any nuisance regression set this argument to `NULL`, `0`, or `FALSE`. Default: `NULL`.

Details

First, the principal components (PCs) of each noise region of interest (ROI) are calculated. For each ROI, voxels are centered and scaled (can be disabled with the arguments `center` and `scale`), and then the PCs are calculated via the singular value decomposition.

Next, aCompCor is performed to remove the shared variation between the noise ROI PCs and each location in the data. This is accomplished by a nuisance regression using a design matrix with the noise ROI PCs, any additional regressors specified by `nuisance`, and an intercept term. (To detrend the data and perform aCompCor in the same regression, `nuisance` can be set to DCT bases obtained with the function `dct_bases`.)

Value

A list with entries `"data"`, `"noise"`, and potentially `"ROI_data"`.

The entry `"data"` will be a `V x T` matrix where each row corresponds to a data location (if it was originally an array, the locations will be voxels in spatial order). Each row will be a time series with each noise PC regressed from it. This entry will be `NULL` if there was no data.

The entry `"noise"` is a list of noise PC scores, their corresponding variance, and their ROI mask, for each noise ROI.

If the data ROI is not all `TRUE`, the entry `"ROI_data"` will have the ROI mask for the data.

References

- Behzadi, Y., Restom, K., Liau, J. & Liu, T. T. A component based noise correction method (CompCor) for BOLD and perfusion based fMRI. NeuroImage 37, 90-101 (2007).
- Muschelli, J. et al. Reduction of motion-related artifacts in resting state fMRI using aCompCor. NeuroImage 96, 22-35 (2014).

See Also

CompCor_HCP

`CompCor_HCP` *Anatomical CompCor for HCP NIFTI and CIFTI data*

Description

Wrapper to `CompCor` for HCP-format data. Can be used to clean the surface-based CIFTI data with aCompCor using the noise PCs and ROIs calculated from the NIFTI fMRI data and NIFTI mask. Can also be used to just obtain the noise PCs and ROIs without performing aCompCor, if the CIFTI data is not provided.

Usage

```
CompCor_HCP(
  nii,
  nii_labels,
  ROI_noise = c("wm_cort", "csf"),
  noise_nPC = 5,
  noise_erosion = NULL,
  idx = NULL,
  cii = NULL,
  brainstructures = c("left", "right"),
```

```
  center = TRUE,
  scale = TRUE,
  DCT = 0,
  nuisance_too = NULL,
  verbose = FALSE
)
```

Arguments

`nii` — I by J by K by T NIFTI object or array (or file path to the NIFTI) which contains whole-brain data, including the noise ROIs. In the HCP, the corresponding file is e.g. "../Results/rfMRI_REST1_LR/rfMRI_REST1_LR.nii.gz"

`nii_labels` — I by J by K NIFTI object or array (or file path to the NIFTI) which contains the corresponding labels to each voxel in `nii`. Values should be according to this table: https://surfer.nmr.mgh.harvard.edu/fswiki/FsTutorial/AnatomicalROI/FreeSurferColorLUT . In the HCP, the corresponding file is "ROIs/Atlas_wmparc.2.nii.gz".

`ROI_noise` — A list of numeric vectors. Each entry should represent labels in `nii_labels` belonging to a single noise ROI, named by that entry's name. Or, this can be a character vector of at least one of the following: `"wm_cort"` (cortical white matter), `"wm_cblm"` (cerebellar white matter), `"csf"` (cerebrospinal fluid). In the latter case, these labels will be used:

`"wm_cort"` `c(3000:4035, 5001, 5002)`

`"wm_cblm"` `c(7, 46)`

`"csf"` `c(4, 5, 14, 15, 24, 31, 43, 44, 63, 250, 251, 252, 253, 254, 255))`

These default ROIs are based on this forum post: https://www.mail-archive.com/hcp-users@humanconnectome.org/msg00931.html
Default: `c("wm_cort", "csf")`

`noise_nPC` — The number of principal components to compute for each noise ROI. Alternatively, values between 0 and 1, in which case they will represent the minimum proportion of variance explained by the PCs used for each noise ROI. The smallest number of PCs will be used to achieve this proportion of variance explained.

Should be a list or numeric vector with the same length as `ROI_noise`. It will be matched to each ROI based on the name of each entry, or if the names are missing, the order of entries. If it is an unnamed vector, its elements will be recycled. Default: `5` (compute the top 5 PCs for each noise ROI).

`noise_erosion` — The number of voxel layers to erode the noise ROIs by. Should be a list or numeric vector with the same length as `ROI_noise`. It will be matched to each ROI based on the name of each entry, or if the names are missing, the order of entries. If it is an unnamed vector, its elements will be recycled. Default: `NULL`, which will use a value of 0 (do not erode the noise ROIs).

`idx` — A numeric vector indicating the timepoints to use, or `NULL` (default) to use all idx. (Indexing begins with 1, so the first timepoint has index 1 and the last has the same index as the length of the scan.)

`cii` — `"xifti"` (or file path to the CIFTI) from which the noise ROI components will be regressed. In the HCP, the corresponding file is e.g. "../Results/rfMRI_REST1_LR/rfMRI_REST1_LR
If not provided, only the noise components will be returned (no data will be cleaned).

brainstructures
: Choose among "left", "right", and "subcortical". Default: `c("left", "right")` (cortical data only)

center, scale
: Center the columns of the data by median, and scale the columns of the data by MAD? Default: `TRUE` for both. Affects both `X` and the noise data. `center` also applies to `nuisance_too` so if it is `FALSE`, `nuisance_too` must already be centered.

DCT
: Add DCT bases to the nuisance regression? Use an integer to indicate the number of cosine bases. Use `0` (default) to forgo detrending.

  The data must be centered, either before input or with `center`.

nuisance_too
: A matrix of nuisance signals to add to the nuisance regression. Should have T rows. `NULL` to not add additional nuisance regressors (default).

verbose
: Should occasional updates be printed? Default: `FALSE`.

Value

The noise components, and if `cii` is provided, the cleaned surface-based data as a `"xifti"` object.

References

- Behzadi, Y., Restom, K., Liau, J. & Liu, T. T. A component based noise correction method (CompCor) for BOLD and perfusion based fMRI. NeuroImage 37, 90-101 (2007).
- Muschelli, J. et al. Reduction of motion-related artifacts in resting state fMRI using aCompCor. NeuroImage 96, 22-35 (2014).

See Also

CompCor

`coordlist_to_vol` *Convert coordinate list to 3D array*

Description

Converts a sparse coordinate list to its non-sparse volumetric representation.

Usage

```
coordlist_to_vol(coords, fill = FALSE)
```

Arguments

`coords`	The sparse coordinate list. Should be a `"data.frame"` or matrix with voxels along the rows and three or four columns. The first three columns should be integers indicating the spatial coordinates of the voxel. If the fourth column is present, it will be the value used for that voxel. If it is absent, the value will be `TRUE` or `1` if `fill` is not one of those values, and `FALSE` or `0` if `fill` is. The data type will be the same as that of `fill`. The fourth column must be logical or numeric.
`fill`	Logical or numeric fill value for the volume. Default: `FALSE`.

Value

The volumetric data

`crop_vol` *Crop a 3D array*

Description

Remove empty (zero-valued) edge slices from a 3D array.

Usage

```
crop_vol(x)
```

Arguments

`x`	The numeric 3D array to crop.

Value

A list of length two: `"data"`, the cropped array, and `"padding"`, the number of slices removed from each edge of each dimension.

`dct_bases` *Generate cosine bases for the DCT*

Description

Generate cosine bases for the DCT

Usage

```
dct_bases(T_, n)
```

Arguments

`T_`	Length of timeseries
`n`	Number of cosine bases

Value

Matrix with cosine bases along columns

`dct_convert` *DCT and frequency conversion*

Description

Convert between number of DCT bases and Hz of highpass filter

Usage

```
dct_convert(T_, TR, n = NULL, f = NULL)

dct2Hz(T_, TR, n)

Hz2dct(T_, TR, f)
```

Arguments

`T_`	Length of timeseries (number of timepoints)
`TR`	TR of the fMRI scan, in seconds (the time between timepoints)
`n`	Number of cosine bases
`f`	Hz of highpass filter

Details

Provide either `n` or `f` to calculate the other.

If only the total length of the scan is known, you can set that to `TR` and use `T_=1`.

$f = n/(2 * T_* TR)$

Value

If `n` was provided, the highpass filter cutoff (Hz) is returned. Otherwise, if `f` was provided, the number of cosine bases is returned. The result should be rounded before passing to `dct_bases`

despike_3D *3dDespike from AFNI*

Description

Identify and interpolate outliers. See the AFNI documentation for 3dDespike for additional information.

Usage

```
despike_3D(Yt, c1 = 2.5, c2 = 4)
```

Arguments

Yt	The data vector.
c1	spike threshold. Default: `2.5`.
c2	upper range of the acceptable deviation from the fit. Default: `4`.

Examples

```
if (requireNamespace("fda", quietly=TRUE) && requireNamespace("quantreg", quietly=TRUE)) {
 y <- rnorm(99) + cos(seq(99)/15)*3
 y[20] <- 20
 despike_3D(y)
}
```

detrend *Detrending with DCT or FFT*

Description

Detrending with DCT or FFT

Usage

```
detrend(X, TR, f = 0.008, method = c("DCT", "FFT"))
```

Arguments

X	A numeric matrix, with each column being a timeseries to detrend. For fMRI data, `X` should be `T` timepoints by `V` brain locations.
TR	The time step between adjacent rows of `X`, in seconds
f	The frequency of the highpass filter, in Hertz. Default: `.008`
method	`"DCT"` (default) or `"FFT"`.

Value

Detrended X

Examples

```
detrend(matrix(rnorm(700), nrow=100), TR=.72)
```

---

dilate_mask_vol   *Dilate 3D mask*

Description

Dilate a volumetric mask by a certain number of voxel layers. For each layer, any out-of-mask voxel adjacent to at least one in-mask voxel is added to the mask.

Usage

```
dilate_mask_vol(vol, n_dilate = 1, out_of_mask_val = NA, new_val = 1)
```

Arguments

vol	The 3D array to dilate. The mask to dilate is defined by all values not in out_of_mask_val.
n_dilate	The number of layers to dilate the mask by. Default: 1.
out_of_mask_val	A voxel is not included in the mask if and only if its value is in this vector. Default: NA. If vol is simply a logical array with TRUE values for in-mask voxels, use out_of_mask_val=FALSE.
new_val	Value for voxels newly added to the mask. Default: 1. If vol is simply a logical array with TRUE values for in-mask voxels, use new_val=1.

Details

Diagonal voxels are not considered adjacent, i.e. the voxel at (0,0,0) is not adjacent to the voxels at (1,1,0) or (1,1,1), although it is adjacent to (1,0,0).

Value

The dilated vol. It is the same as vol, but dilated voxels are replaced with new_val.

dim_reduce *PCA-based Dimension Reduction and Prewhitening*

Description

Performs dimension reduction and prewhitening based on probabilistic PCA using SVD. If dimensionality is not specified, it is estimated using the method described in Minka (2008).

Usage

```
dim_reduce(X, Q = NULL, Q_max = 100)
```

Arguments

X	A numeric matrix, with each column being a centered timeseries. For fMRI data, X should be T timepoints by V brain locations.
Q	Number of latent dimensions to estimate. If NULL (default), estimated using PESEL (Sobczyka et al. 2020).
Q_max	Maximal number of principal components for automatic dimensionality selection with PESEL. Default: 100.

Value

A list containing the dimension-reduced data (data_reduced, a $V \times Q$ matrix), prewhitening/dimension reduction matrix (H, a QxT matrix) and its (pseudo-)inverse (Hinv, a TxQ matrix), the noise variance (sigma_sq), the correlation matrix of the dimension-reduced data (C_diag, a QxQ matrix), and the dimensionality (Q).

Examples

```
nT <- 30
nV <- 400
nQ <- 7
X <- matrix(rnorm(nV*nQ), nrow=nV) %*% diag(seq(nQ, 1)) %*% matrix(rnorm(nQ*nT), nrow=nQ)
dim_reduce(X, Q=nQ)
```

dual_reg *Dual Regression*

Description

Dual Regression

Usage

```
dual_reg(
  BOLD,
  GICA,
  scale = c("local", "global", "none"),
  scale_sm_xifti = NULL,
  scale_sm_FWHM = 2,
  TR = NULL,
  hpf = 0.01,
  GSR = FALSE
)
```

Arguments

`BOLD` Subject-level fMRI data matrix ($V \times T$). Rows will be centered.

`GICA` Group-level independent components ($V \times Q$)

`scale` `"local"` (default), `"global"`, or `"none"`. Local scaling will divide each data location's time series by its estimated standard deviation. Global scaling will divide the entire data matrix by the mean image standard deviation (`mean(sqrt(rowVars(BOLD)))`).

`scale_sm_xifti, scale_sm_FWHM`
Only applies if `scale=="local"` and `BOLD` represents CIFTI-format data. To smooth the standard deviation estimates used for local scaling, provide a `"xifti"` object with data locations in alignment with `BOLD`, as well as the smoothing FWHM (default: `2`). If no `"xifti"` object is provided (default), do not smooth.

`TR` The temporal resolution of the data, i.e. the time between volumes, in seconds. `TR` is required for detrending with `hpf`.

`hpf` The frequency at which to apply a highpass filter to the data during pre-processing, in Hertz. Default: `0.01` Hertz. Set to `0` to disable the highpass filter.

The highpass filter serves to detrend the data, since low-frequency variance is associated with noise. Highpass filtering is accomplished by nuisance regression of discrete cosine transform (DCT) bases.

Note the `TR` argument is required for highpass filtering. If `TR` is not provided, `hpf` will be ignored.

`GSR` Center BOLD across columns (each image)? This is equivalent to performing global signal regression. Default: `FALSE`.

Value

A list containing the subject-level independent components **S** ($V \times Q$), and subject-level mixing matrix **A** (TxQ).

Examples

```
nT <- 30
nV <- 400
nQ <- 7
mU <- matrix(rnorm(nV*nQ), nrow=nV)
```

```
mS <- mU %*% diag(seq(nQ, 1)) %*% matrix(rnorm(nQ*nT), nrow=nQ)
BOLD <- mS + rnorm(nV*nT, sd=.05)
GICA <- mU
dual_reg(BOLD=BOLD, GICA=mU, scale="local")
```

`dual_reg_parc` *Multiple regression for parcel data*

Description

Multiple regression for parcel data

Usage

```
dual_reg_parc(
  BOLD,
  parc,
  parc_vals,
  scale = c("local", "global", "none"),
  scale_sm_xifti = NULL,
  scale_sm_FWHM = 2,
  TR = NULL,
  hpf = 0.01,
  GSR = FALSE
)
```

Arguments

`BOLD`	Subject-level fMRI data matrix ($V \times T$). Rows will be centered.
`parc`	The parcellation as an integer vector.
`parc_vals`	The parcel values (keys) in desired order, e.g. `sort(unique(parc))`.
`scale`	`"local"` (default), `"global"`, or `"none"`. Local scaling will divide each data location's time series by its estimated standard deviation. Global scaling will divide the entire data matrix by the mean image standard deviation (`mean(sqrt(rowVars(BOLD)))`).
`scale_sm_xifti, scale_sm_FWHM`	Only applies if `scale=="local"` and `BOLD` represents CIFTI-format data. To smooth the standard deviation estimates used for local scaling, provide a `"xifti"` object with data locations in alignment with `BOLD`, as well as the smoothing FWHM (default: `2`). If no `"xifti"` object is provided (default), do not smooth.
`TR`	The temporal resolution of the data, i.e. the time between volumes, in seconds. `TR` is required for detrending with `hpf`.
`hpf`	The frequency at which to apply a highpass filter to the data during pre-processing, in Hertz. Default: `0.01` Hertz. Set to `0` to disable the highpass filter.

The highpass filter serves to detrend the data, since low-frequency variance is associated with noise. Highpass filtering is accomplished by nuisance regression of discrete cosine transform (DCT) bases.

Note the `TR` argument is required for highpass filtering. If `TR` is not provided, `hpf` will be ignored.

`GSR` Center BOLD across columns (each image)? This is equivalent to performing global signal regression. Default: `FALSE`.

Value

A list containing the subject-level independent components **S** ($Q \times V$), and subject-level mixing matrix **A** (TxQ).

`erode_mask_vol` *Erode 3D mask*

Description

Erode a volumetric mask by a certain number of voxel layers. For each layer, any in-mask voxel adjacent to at least one out-of-mask voxel is removed from the mask.

Usage

```
erode_mask_vol(vol, n_erosion = 1, out_of_mask_val = NA)
```

Arguments

`vol` The 3D array to erode. The mask to erode is defined by all values not in `out_of_mask_val`.

`n_erosion` The number of layers to erode the mask by. Default: `1`.

`out_of_mask_val` A voxel is not included in the mask if and only if its value is in this vector. The first value of this vector will be used to replace eroded voxels. Default: `NA`. If `vol` is simply a logical array with `TRUE` values for in-mask voxels, use `out_of_mask_val=FALSE`.

Details

Diagonal voxels are not considered adjacent, i.e. the voxel at (0,0,0) is not adjacent to the voxels at (1,1,0) or (1,1,1), although it is adjacent to (1,0,0).

Value

The eroded `vol`. It is the same as `vol`, but eroded voxels are replaced with `out_of_mask_val[1]`.

fMRItools *fMRItools: Routines for Common fMRI Processing Tasks*

Description

See help(package="fMRItools") for a list of functions.

Author(s)

Maintainer: Amanda Mejia <mandy.mejia@gmail.com>

Authors:

- Damon Pham <damondpham@gmail.com> (ORCID)

Other contributors:

- Mark Fiecas <mfiecas@umn.edu> [contributor]

See Also

Useful links:

- https://github.com/mandymejia/fMRItools
- Report bugs at https://github.com/mandymejia/fMRItools/issues

fsl_bptf bptf *function from FSL*

Description

Copy of bptf highpass filter from FSL. The results are very similar but not identical.

Usage

```
fsl_bptf(orig_data, HP_sigma = 2000)
```

Arguments

orig_data	$T \times V$ data matrix whose columns will be detrended
HP_sigma	The frequency parameter for the highpass filter

Details

Sources: https://cpb-us-w2.wpmucdn.com/sites.udel.edu/dist/7/4542/files/2016/09/fsl_temporal_filt-15sywxn.m https://github.com/rordenlab/niimath/blob/master/src/core32.c

Value

The data with detrended columns

References

- Jenkinson, M., Beckmann, C. F., Behrens, T. E. J., Woolrich, M. W. & Smith, S. M. FSL. NeuroImage 62, 782-790 (2012).

Examples

```
fsl_bptf(matrix(rnorm(700), nrow=100))
```

---

hat_matrix *Hat matrix*

Description

Get the hat matrix from a design matrix.

Usage

```
hat_matrix(design)
```

Arguments

design	The T by Q design matrix

Details

Uses the QR decomposition.

Value

The T by T hat matrix

Examples

```
hat_matrix(cbind(seq(100), 1))
```

infer_format_ifti *Infer fMRI data format*

Description

Infer fMRI data format

Usage

```
infer_format_ifti(BOLD, verbose = FALSE)
```

Arguments

BOLD	The fMRI data
verbose	Print the format? Default: `FALSE`.

Value

A length-two vector. The first element indicates the format: `"CIFTI"` file path, `"xifti"` object, `"GIFTI"` file path, `"gifti"` object, `"NIFTI"` file path, `"nifti"` object, `"RDS"` file path, or `"data"`. The second element indicates the sub-format if relevant; i.e. the type of CIFTI or GIFTI file/object.

infer_format_ifti_vec *Infer fMRI data format for several inputs*

Description

Vectorized version of `infer_format_ifti`. Expects all inputs to have the same format.

Usage

```
infer_format_ifti_vec(BOLD, verbose = FALSE)
```

Arguments

BOLD	The vector of fMRI data, expected to be of one format
verbose	Print the format? Default: `FALSE`.

Details

Raises an error if the elements of `BOLD` do not share the same format.

Value

A length-two vector. The first element indicates the format: `"CIFTI"` file path, `"xifti"` object, `"GIFTI"` file path, `"gifti"` object, `"NIFTI"` file path, `"nifti"` object, `"RDS"` file path, or `"data"`. The second element indicates the sub-format if relevant; i.e. the type of CIFTI or GIFTI file/object.

is_1 — *Is this object the expected data type, and length one?*

Description

Is this object the expected data type, and length one?

Usage

```
is_1(x, dtype = c("numeric", "logical", "character"))
```

Arguments

`x`	The value to check
`dtype`	The data type. Default: `"numeric"`. Also can be `"logical"` or `"character"`

Value

`TRUE` if `x` is `dtype` and length one.

is_constant — *Is this numeric vector constant?*

Description

Is this numeric vector constant?

Usage

```
is_constant(x, TOL = 1e-08)
```

Arguments

`x`	The numeric vector
`TOL`	minimum range of `x` to be considered non-constant. Default: `1e-8`

Value

Is `x` constant?

is_integer — *Is this an integer?*

Description

Is this an integer?

Usage

```
is_integer(x, nneg = FALSE)
```

Arguments

x	The putative integer
nneg	Require x>=0 (non-negative) too?

Value

Logical indicating whether x is an integer

is_posNum — *Is this object a positive number? (Or non-negative)*

Description

Is this object a positive number? (Or non-negative)

Usage

```
is_posNum(x, zero_ok = FALSE)
```

Arguments

x	The value to check
zero_ok	Is a value of zero ok?

Value

Logical indicating if x is a single positive or non-negative number

mat2UT — *Matrix to Upper Triangular Vector*

Description

Returns the vectorized upper triangle of a square matrix

Usage

```
mat2UT(x)
```

Arguments

`x`	A square matrix

Value

The vectorized upper triangle of x.

match_exactly — *Do these character vectors match exactly?*

Description

Checks if a user-defined character vector matches an expected character vector. That is, they share the same lengths and entries in the same order. For vectors of the same lengths, the result is `all(a == b)`.

Usage

```
match_exactly(
  user,
  expected,
  fail_action = c("message", "warning", "stop", "nothing")
)
```

Arguments

`user`	Character vector of user input.
`expected`	Character vector of expected/allowed values.
`fail_action`	If any value in `user` could not be matched, or repeated matches occurred, what should happen? Possible values are `"message"` (default), `"warning"`, `"stop"`, and `"nothing"`.

Details

Attributes are ignored.

Value

Logical. Do `user` and `expected` match?

`match_input` *Match user inputs to expected values*

Description

Match each user input to an expected/allowed value. Raise a warning if either several user inputs match the same expected value, or at least one could not be matched to any expected value. `ciftiTools` uses this function to match keyword arguments for a function call. Another use is to match brainstructure labels ("left", "right", or "subcortical").

Usage

```
match_input(
  user,
  expected,
  fail_action = c("stop", "warning", "message", "nothing"),
  user_value_label = NULL
)
```

Arguments

`user`	Character vector of user input. These will be matched to `expected` using `match.arg`.
`expected`	Character vector of expected/allowed values.
`fail_action`	If any value in `user` could not be matched, or repeated matches occurred, what should happen? Possible values are `"stop"` (default; raises an error), `"warning"`, and `"nothing"`.
`user_value_label`	How to refer to the user input in a stop or warning message. If `NULL`, no label is used.

Value

The matched user inputs.

mean_squares — *Compute mean squares from variance decomposition*

Description

Compute mean squares from variance decomposition

Usage

```
mean_squares(vd)
```

Arguments

`vd`	The variance decomposition

Value

The mean squares

Mode — *Mode of data vector*

Description

Get mode of a data vector. But use the median instead of the mode if all data values are unique.

Usage

```
Mode(x)
```

Arguments

`x`	The data vector

Value

The mode

norm_BOLD *Normalize BOLD data*

Description

Center the data across space and/or time, detrend, and scale, in that order. For dual regression, row centering is required and column centering is not recommended. Scaling and detrending depend on the user preference.

Usage

```
norm_BOLD(
  BOLD,
  center_rows = TRUE,
  center_cols = FALSE,
  scale = c("local", "global", "none"),
  scale_sm_xifti = NULL,
  scale_sm_FWHM = 2,
  TR = NULL,
  hpf = 0.01
)
```

Arguments

`BOLD` fMRI numeric data matrix ($V \times T$)

`center_rows, center_cols`
: Center BOLD data across rows (each data location's time series) or columns (each time point's image)? Default: `TRUE` for row centering, and `FALSE` for column centering.

`scale` : `"global"` (default), `"local"`, or `"none"`. Global scaling will divide the entire data matrix by the mean image standard deviation (`mean(sqrt(rowVars(BOLD)))`). Local scaling will divide each data location's time series by its estimated standard deviation.

`scale_sm_xifti, scale_sm_FWHM`
: Only applies if `scale=="local"` and `BOLD` represents CIFTI-format data. To smooth the standard deviation estimates used for local scaling, provide a `"xifti"` object with data locations in alignment with `BOLD`, as well as the smoothing FWHM (default: `2`). If no `"xifti"` object is provided (default), do not smooth.

`TR` : The temporal resolution of the data, i.e. the time between volumes, in seconds. `TR` is required for detrending with `hpf`.

`hpf` : The frequency at which to apply a highpass filter to the data during pre-processing, in Hertz. Default: `0.01` Hertz. Set to `0` to disable the highpass filter.

The highpass filter serves to detrend the data, since low-frequency variance is associated with noise. Highpass filtering is accomplished by nuisance regression of discrete cosine transform (DCT) bases.

Note the `TR` argument is required for highpass filtering. If `TR` is not provided, `hpf` will be ignored.

Value

Normalized BOLD data matrix ($V \times T$)

`nuisance_regression` *Nuisance regression*

Description

Performs nuisance regression. Important note: the data and design matrix must both be centered, or an intercept must be included in the design matrix.

Usage

```
nuisance_regression(Y, design)
```

Arguments

`Y`	The $T \times V$ or $V \times T$ data.
`design`	The $T \times Q$ matrix of nuisance regressors.

Value

The data after nuisance regression.

Examples

```
Y <- matrix(rnorm(700), nrow=100)
design <- cbind(seq(100), 1)
nuisance_regression(Y, design)
```

`pad_vol` *Pad 3D Array*

Description

Pad a 3D array by a certain amount in each direction, along each dimension. This operation is like the opposite of cropping.

Usage

```
pad_vol(x, padding, fill = NA)

uncrop_vol(x, padding, fill = NA)
```

Arguments

- `x` — A 3D array, e.g. `unvec_vol(xifti$data$subcort, xifti$meta$subcort$mask)`.
- `padding` — A 3×2 matrix indicating the number of slices to add at the beginning (first column) and end (second column) of each of dimension, e.g. `xifti$meta$subcort$mask_padding`.
- `fill` — Value to pad with. Default: `NA`.

Value

The padded array

Examples

```
x <- array(seq(24), dim=c(2,3,4))
y <- pad_vol(x, array(1, dim=c(3,2)), 0)
stopifnot(all(dim(y) == dim(x)+2))
stopifnot(sum(y) == sum(x))
z <- crop_vol(y)$data
stopifnot(identical(dim(x), dim(z)))
stopifnot(max(abs(z - x))==0)
```

PCA — *PCA for tall matrix*

Description

Efficient PCA for a tall matrix (many more rows than columns). Uses the SVD of the covariance matrix. The dimensionality of the result can be preset with `Q` or estimated with PESEL.

Usage

```
PCA(X, center = TRUE, Q = NULL, Q_max = 100, Vdim = 0)
```

Arguments

- `X` — The tall numeric matrix for which to compute the PCA. For fMRI data, `X` should be `V` brain locations by `T` timepoints.
- `center` — Center the columns of `X`? Default: `TRUE`. Set to `FALSE` if already centered. Centered data is required to compute PCA.
- `Q` — Number of latent dimensions to estimate. If `NULL` (default), estimated using PESEL (Sobczyka et al. 2020).
- `Q_max` — Maximal number of principal components for automatic dimensionality selection with PESEL. Default: `100`.
- `Vdim` — Number of principal directions to obtain. Default: `0`. Can also be `"Q"` to set equal to the value of `Q`. Note that setting this value less than `Q` does not speed up computation time, but does save on memory. Note that the directions will be with respect to `X`, not its covariance matrix.

Value

The SVD decomposition

Examples

```
U <- matrix(rnorm(900), nrow=300, ncol=3)
V <- matrix(rnorm(15), nrow=3, ncol=5)
PCA(U %*% V)
```

---

pct_sig — *Convert data values to percent signal.*

Description

Convert data values to percent signal.

Usage

```
pct_sig(X, center = median, by = c("column", "all"))
```

Arguments

`X`	a T by N numeric matrix. The columns will be normalized to percent signal.
`center`	A function that computes the center of a numeric vector. Default: `median`. Other common options include `mean` and `mode`.
`by`	Should the center be measured individually for each `"column"` (default), or should the center be the same across `"all"` columns?

Value

`X` with its columns normalized to percent signal. (A value of 85 will represent a -15% signal change.)

---

plot_FC — *Plot FC*

Description

Plot a functional connectivity matrix.

Usage

```
plot_FC(
  FC,
  zlim = c(-1, 1),
  diag_val = NULL,
  title = "FC matrix",
  cols = color_palette("Beach"),
  cleg_ticks_by = diff(zlim)/2,
  cleg_digits = NULL,
  labels = NULL,
  lines = NULL,
  lines_col = "black",
  lines_lwd = 1,
  cex = 0.8
)
```

Arguments

- `FC` — The functional connectivity matrix, a square numeric matrix with values between -1 and 1.
- `zlim` — The minimum and maximum range of the color scale. Default: `c(-1, 1)`. If in descending order, the color scale will be reversed.
- `diag_val` — Set to `NA` for white, `1`, or `NULL` (default) to not modify the diagonal values in `FC`.
- `title` — (Optional) Plot title.
- `cols` — Character vector of colors for the color scale. Default: `color_palette("Beach")`.
- `cleg_ticks_by` — Spacing between ticks on the color legend. Default: `range(zlim)/2`.
- `cleg_digits` — How many decimal digits for the color legend. Default: `NULL` to set automatically.
- `labels` — A character vector of length `length(lines)+1`, giving row/column labels for the submatrices divided by `lines`. If `NULL` (default), do not add labels.
- `lines` — Add lines to divide the FC matrix into submatrices? Default: `NULL` (do not draw lines). Use `seq(nN)` to delineate each individual row and column.
- `lines_col, lines_lwd` — Color and line width of the `lines`. Default: black lines of width `1`.
- `cex` — Text size. Default: `0.8`.

---

read_nifti — *Wrapper to functions for reading NIFTIs*

Description

Tries `RNifti::readNifti`, then `oro.nifti::readNIfTI`. If neither package is available an error is raised.

Usage

```
read_nifti(nifti_fname)
```

Arguments

`nifti_fname`	The file name of the NIFTI.

Details

For `oro.nifti::readNIFTI` the argument `reorient=FALSE` will be used.

Value

The NIFTI

`scale_design_mat` *Scale a design matrix*

Description

Scale the columns of a matrix by dividing each column by its highest-magnitude value, and then subtracting its mean.

Usage

```
scale_design_mat(x, doRows = FALSE)
```

Arguments

`x`	A $T \times K$ numeric matrix. In the context of a design matrix for a GLM analysis of task fMRI, T is the number of time points and K is the number of task covariates.
`doRows`	Scale the rows instead? Default: `FALSE`.

Value

The scaled design matrix

Examples

```
scale_design_mat(cbind(seq(7), 1, rnorm(7)))
```

scale_med *Robust scaling*

Description

Centers and scales the columns of a matrix robustly

Usage

```
scale_med(mat, TOL = 1e-08, drop_const = TRUE, doRows = FALSE)
```

Arguments

mat	A numeric matrix. Its columns will be centered and scaled.
TOL	Columns with MAD below this value will be considered constant. Default: 1e-8
drop_const	Drop constant columns? Default: TRUE. If FALSE, set to NA instead.
doRows	Center and scale the rows instead? Default: FALSE.

Details

Centers each column on its median, and scales each column by its median absolute deviation (MAD). If there are constant-valued columns, they are removed if drop_const or set to NA if !drop_const, and a warning is raised. If all columns are constant, an error is raised.

Value

The input matrix with its columns centered and scaled.

scale_timeseries *Scale the BOLD timeseries*

Description

Scale the BOLD timeseries

Usage

```
scale_timeseries(
  BOLD,
  scale = c("auto", "mean", "sd", "none"),
  transpose = TRUE
)
```

Arguments

BOLD	fMRI data as a locations by time ($V \times T$) numeric matrix.

scale	Option for scaling the BOLD response.
	
	`\code{"auto"} (default) will use \code{"mean"} scaling except if demeaned`
	
	data is detected (if any mean is less than one), in which case `"sd"` scaling will be used instead.
	
	`\code{"mean"} scaling will scale the data to percent local signal change.`
	
	`\code{"sd"} scaling will scale the data by local standard deviation.`
	
	`\code{"none"} will only center the data, not scale it.`

transpose	Transpose `BOLD` if there are more columns than rows? (Because we usually expect the number of voxels to exceed the number of time points.) Default: `TRUE`.

Value

Scale to units of percent local signal change and centers

`sign_flip` *Sign match ICA results*

Description

Flips all source signal estimates (S) to positive skew

Usage

```
sign_flip(x)
```

Arguments

x	The ICA results: a list with entries `"S"` and `"M"`

Value

`x` but with positive skew source signals `skew_pos` *Positive skew?*

Description

Does the vector have a positive skew?

Usage

```
skew_pos(x)
```

Arguments

`x`	The numeric vector for which to calculate the skew. Can also be a matrix, in which case the skew of each column will be calculated.

Value

`TRUE` if the skew is positive or zero. `FALSE` if the skew is negative.

`sum_neighbors_vol` *Sum of each voxel's neighbors*

Description

For each voxel in a 3D logical or numeric array, sum the values of the six neighboring voxels.

Usage

```
sum_neighbors_vol(arr, pad = 0)
```

Arguments

`arr`	The 3D array.
`pad`	In order to compute the sum, the array is temporarily padded along each edge with the value of `pad`. `0` (default) will mean that edge voxels reflect the sum of 3-5 neighbors whereas non-edge voxels reflect the sum of 6 neighbors. An alternative is to use a value of `NA` so that edge voxels are `NA`-valued because they did not have a complete set of six neighbors. Perhaps another option is to use `mean(arr)`.

Details

Diagonal voxels are not considered adjacent, i.e. the voxel at (0,0,0) is not adjacent to the voxels at (1,1,0) or (1,1,1), although it is adjacent to (1,0,0).

Value

An array with the same dimensions as `arr`. Each voxel value will be the sum across the immediate neighbors. If `arr` was a logical array, this value will be between 0 and 6.

`unmask_mat` *Unmask matrix data*

Description

Insert empty rows or columns to a matrix. For example, medial wall vertices can be added back to the cortex data matrix.

Usage

```
unmask_mat(x, mask, mask_dim = 1, fill = NA)
```

Arguments

`x`	The data matrix to unmask.
`mask`	The logical mask: the number of `TRUE` values should match the size of the (`mask_dim`)th dimension in `dat`.
`mask_dim`	Rows, `1` (default), or columns, `2`.
`fill`	The fill value for the inserted rows/columns. Default: `NA`.

Value

The unmasked matrix.

`unvec_mat` *Transform vector data to image*

Description

From a $v \times p$ matrix of vectorized data and an $m \times n$ image mask with v in-mask locations, create a list of p $m \times n$ data arrays in which the mask locations are filled in with the vectorized data values.

Consider using `abind::abind` to merge the result into a single array.

Usage

```
unvec_mat(x, mask, fill_value = NA)
```

Arguments

x	$v \times p$ matrix, where v is the number of voxels within a mask and p is the number of vectors to transform into matrix images.
mask	$m \times n$ logical matrix in which v entries are TRUE and the rest are FALSE.
fill_value	Out-of-mask value in the output image. Default: NA.

Value

A list of masked values from x

Examples

```
x <- unvec_mat(
 cbind(seq(3), seq(2,4), seq(3,5)),
 matrix(c(rep(TRUE, 3), FALSE), ncol=2),
 0
)
y <- array(c(1,2,3,0,2,3,4,0,3,4,5,0), dim=c(2,2,3))
stopifnot(identical(x[[1]], y[,,1]))
stopifnot(identical(x[[2]], y[,,2]))
stopifnot(identical(x[[3]], y[,,3]))
```

---

unvec_vol *Convert vectorized data back to volume*

Description

Un-applies a mask to vectorized data to yield its volumetric representation. The mask and data should have compatible dimensions: the number of rows in dat should equal the number of locations within the mask.

Usage

```
unvec_vol(dat, mask, fill = NA)
```

Arguments

dat	Data matrix with locations along the rows and measurements along the columns. If only one set of measurements were made, this may be a vector.
mask	Volumetric binary mask. TRUE indicates voxels inside the mask.
fill	The value for locations outside the mask. Default: NA.

Value

The 3D or 4D unflattened volume array

UT2mat *Upper Triangular Vector to Matrix*

Description

Returns the symmatric square matrix from a vector containing the upper triangular elements

Usage

```
UT2mat(x, diag = 0, LT = 0)
```

Arguments

x	A vector containing the upper triangular elements of a square, symmetric matrix.
diag	A scalar value to use for the diagonal values of the matrix, or "x" if x includes the diagonal values. Default: NA.
LT	A scalar value to use for the lower triangular values of the matrix. Default: 0.

Value

A symmetric matrix with the values of x in the upper and lower triangles and the value diag on the diagonal.

validate_design_mat *Validate design matrix*

Description

Coerces design to a numeric matrix, and optionally checks that the number of rows is as expected. Sets constant-valued columns to 1, and scales all other columns.

Usage

```
validate_design_mat(design, T_ = NULL)
```

Arguments

design	The design matrix
T_	the expected number of rows in design. Default: NULL (no expected value to validate).

Value

The (modified) design matrix

`var_decomp` *Compute variance decomposition*

Description

Calculate the various ANOVA sums of squares for repeated measures data.

Usage

```
var_decomp(x, verbose = FALSE)
```

Arguments

`x`	The data as a 3D array: measurements by subjects by variables. (Alternatively, a matrix that is measurements by subjects, if only one variable exists.)
`verbose`	If `TRUE`, display progress of algorithm. Default: `FALSE`.

Value

The variance decomposition

`vox_locations` *Get coordinates of each voxel in a mask*

Description

Made for obtaining voxel locations in 3D space from the subcortical metadata of CIFTI data: the volumetric mask, the transformation matrix and the spatial units.

Usage

```
vox_locations(mask, trans_mat, trans_units = NULL)
```

Arguments

`mask`	3D logical mask
`trans_mat`	Transformation matrix from array indices to spatial coordinates.
`trans_units`	Units for the spatial coordinates (optional).

Value

A list: `coords` and `trans_units`.

Index