Package 'ciftiTools'

February 24, 2024

Type Package

Title Tools for Reading, Writing, Viewing and Manipulating CIFTI Files

Version 0.14.0

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

Description CIFTI files contain brain imaging data in ``grayordinates,'' which represent the gray matter as cortical surface vertices (left and right) and subcortical voxels (cerebellum, basal ganglia, and other deep gray matter). 'ciftiTools' provides a unified environment for reading, writing, visualizing and manipulating CIFTI-format data. It supports the ``dscalar,'' ``dlabel,'' and ``dtseries'' intents. Grayordinate data is read in as a ``xifti'' object, which is structured for convenient access to the data and metadata, and includes support for surface geometry files to enable spatially-dependent functionality such as static or interactive visualizations and smoothing.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

Imports fields, gifti (> 0.7.5), grDevices, oro.nifti, RNifti, RColorBrewer, rgl, viridisLite, xml2

Suggests covr, ggplot2, ggpubr, grid, gridExtra, htmlwidgets, manipulateWidget, knitr, rmarkdown, png, testthat (>= 3.0.0)

RoxygenNote 7.3.1

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

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

NeedsCompilation no

Author Amanda Mejia [aut, cre], Damon Pham [aut] (<https://orcid.org/0000-0001-7563-4727>), John Muschelli [ctb] (<https://orcid.org/0000-0001-6469-1750>)

Repository CRAN

Date/Publication 2024-02-24 15:00:02 UTC

R topics documented:

`add_surf` *Add surface(s) to a* `"xifti"`

Description

Add left or right cortical surface geometry to a `"xifti"` object.

Usage

```
add_surf(xifti, surfL = NULL, surfR = NULL)
```

Arguments

xifti	A "xifti" object.
surfL	(Optional) Left brain surface model. Can be a file path to a GIFTI surface geometry file (ends in "*.surf.gii"), a "gifti" object representing surface geometry, or a "surf" object.
surfR	(Optional) Right brain surface model. Can be a file path to a GIFTI surface geometry file (ends in "*.surf.gii"), a "gifti" object representing surface geometry, or a "surf" object.

Details

surfL will be added to xifti$surf$cortex_left and surfR will be added to xifti$surf$cortex_right. Any existing surfaces will be overwritten.

If the resolutions of the data and surfaces do not match, the surfaces will be resampled to match the resolution of the data. The barycentric resampling method, which is recommended for anatomical surfaces, will be used.

Value

the "xifti" object with added surface geometry components.

See Also

Other manipulating xifti: apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

Other surface-related: boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), mask_surf(), read_surf(), resample_surf(), rotate_surf(), surf_area(), view_surf(), write_surf_gifti()

---

apply_parc *Apply function over locations in each parcel*

Description

Apply a function across all locations in each parcel, for a pair of data and parcellation "xifti" objects that are in registration with one another. By default, the mean value in each parcel is calculated.

Usage

```
apply_parc(
  xii,
  parc,
  FUN = mean,
  mwall_value = NA,
  return_as = c("matrix", "xifti"),
  ...
)
```

Arguments

`xii`	The `"xifti"` data to apply the function over, within each parcel.
`parc`	The `"xifti"` "dlabel" parcellation. Each parcel is defined by a unique key in the label table. If there are multiple columns, only the first column will be used. Alternatively, `parc` can just be a vector of keys whose length is the number of data locations in `"xii"`.
`FUN`	A function that takes as input an $M \times N$ matrix (M locations in a given parcel, and N measurements/columns in `xii`) and outputs a constant-sized (Q) numeric vector. Default: `mean`. Use `colMeans` to obtain the average timeseries of each parcel, such as in order to compute functional connectivity.
`mwall_value`	If there is a medial wall in `xii`, what should value should medial wall locations be replaced with prior to calculation? Default: `NA`.
`return_as`	`"matrix"` (default) where each row corresponds to a parcel, or a `"xifti"` object where each location's value is the value of its corresponding parcel?
`...`	Additional arguments to `FUN`, e.g. `na.rm=TRUE`. Ignored if `FUN=="quick_mean"`.

Value

A $P \times Q$ matrix, where P is the number of parcels and Q is the length of the output of `FUN`. (For `mean`, $Q = 1$).

See Also

Other parcellation-related: `load_parc()`, `load_sub_parc()`, `parc_add_subcortex()`, `parc_borders()`, `parc_vals_to_xifti()`

Other manipulating xifti: `add_surf()`, `apply_xifti()`, `combine_xifti()`, `convert_to_dlabel()`, `merge_xifti()`, `move_to_mwall()`, `newdata_xifti()`, `remap_cifti()`, `remove_xifti()`, `resample_cifti()`, `resample_cifti_from_template()`, `scale_xifti()`, `select_xifti()`, `set_names_xifti()`, `smooth_cifti()`, `transform_xifti()`

apply_xifti *Apply a function along the rows or columns of a "xifti"*

Description

Apply a many-to-N function (e.g. mean) to the rows or columns of a "xifti". If applied row-wise, a "xifti" with N data column(s) is returned. (If the "xifti" had the dlabel intent, and values that are not labels are created, then it is converted to dscalar.) If applied column-wise, a numeric matrix with N rows is returned.

For univariate functions, use transform_xifti instead.

Usage

```
apply_xifti(xifti, margin = c(1, 2), FUN, ...)
```

Arguments

xifti	A "xifti" object.
margin	The dimension along which to apply FUN: 1 for rows (default) and 2 for columns.
FUN	The function. It should take in a numeric vector and return a length-N numeric vector.
...	Additional arguments to FUN

Value

A "xifti" if margin == 1, or a numeric matrix if margin == 2

See Also

Other manipulating xifti: add_surf(), apply_parc(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

as.matrix.xifti *Convert a "xifti" to a matrix*

Description

Converts a "xifti" to a matrix by concatenating the data from each brainstructure along the rows. Surfaces and metadata are discarded.

Usage

```
## S3 method for class 'xifti'
as.matrix(x, ...)
```

Arguments

x	A "xifti" object.
...	Unused

Value

The input as a matrix. Each brainstructure's data is concatenated.

---

as.xifti *Assemble a "xifti" from data*

Description

Assembles cortical data, subcortical data, and/or surface geometry to form a "xifti". The inputs must be data objects (vectors, matrices or arrays, depending on the argument).

Usage

```
as.xifti(
  cortexL = NULL,
  cortexL_mwall = NULL,
  cortexR = NULL,
  cortexR_mwall = NULL,
  mwall_values = c(NA, NaN),
  subcortVol = NULL,
  subcortLabs = NULL,
  subcortMask = NULL,
  surfL = NULL,
  surfR = NULL,
  col_names = NULL,
  HCP_32k_auto_mwall = TRUE,
  validate = TRUE
)

as_xifti(
  cortexL = NULL,
  cortexL_mwall = NULL,
  cortexR = NULL,
  cortexR_mwall = NULL,
  mwall_values = c(NA, NaN),
  subcortVol = NULL,
  subcortLabs = NULL,
  subcortMask = NULL,
  surfL = NULL,
  surfR = NULL
)
```

```
as.cifti(
  cortexL = NULL,
  cortexL_mwall = NULL,
  cortexR = NULL,
  cortexR_mwall = NULL,
  mwall_values = c(NA, NaN),
  subcortVol = NULL,
  subcortLabs = NULL,
  subcortMask = NULL,
  surfL = NULL,
  surfR = NULL
)

as_cifti(
  cortexL = NULL,
  cortexL_mwall = NULL,
  cortexR = NULL,
  cortexR_mwall = NULL,
  mwall_values = c(NA, NaN),
  subcortVol = NULL,
  subcortLabs = NULL,
  subcortMask = NULL,
  surfL = NULL,
  surfR = NULL
)
```

Arguments

`cortexL`, `cortexL_mwall`

Left cortex data and ROI. Each must be a data matrix or vector.

If `cortexL_mwall` is not provided, `cortexL` should have data for all vertices on the left cortical surface ($V_L \times T$ data matrix). There will not be a mask for the medial wall. Not providing the medial wall mask is appropriate for ".dlabels.nii" files where the medial wall may have its own label and therefore should not be treated as missing data.

If `cortexL_mwall` is provided, `cortexL` should either have data for all vertices on the left cortical surface ($V_L \times T$ data matrix, with filler values e.g. `0` or `NaN` for medial wall vertices), or have data only for non-medial wall vertices ($(V_L - mwall_L) \times T$ data matrix). The medial wall mask will be the `0` values in `cortexL_mwall`. The medial wall mask should be provided whenever the medial wall should be treated as missing data.

Since the unmasked cortices must have the same number of vertices, `V_L` should match `V_R`.

`cortexR`, `cortexR_mwall`

Right cortex data and ROI. Each must be a data matrix or vector.

If `cortexR_mwall` is not provided, `cortexR` should have data for all vertices on the right cortical surface ($V_R \times T$ data mre will not be a mask for the medial

wall. Not providing the medial wall mask is appropriate for ".dlabels.nii" files where the medial wall may have its own label and therefore should not be treated as missing data.

If `cortexR_mwall` is provided, `cortexR` should either have data for all vertices on the right cortical surface ($V_R \times T$ data matrix, with filler values e.g. `0` or `NaN` for medial wall vertices), or have data only for non-medial wall vertices ($(V_R - mwall_R) \times T$ data matrix). The medial wall mask will be the `0` values in `cortexR_mwall`. The medial wall mask should be provided whenever the medial wall should be treated as missing data.

Since the unmasked cortices must have the same number of vertices, `V_L` should match `V_R`.

`mwall_values` If `cortex[L/R]_mwall` was not provided, or if it was invalid (i.e. bad length or all `TRUE`), the medial wall mask will be inferred from rows in `cortex[L/R]` that are constantly one of these values. Default: `c(NA, NaN)`. If `NULL`, do not attempt to infer the medial wall from the data values. `NULL` should be used if `NA` or `NaN` are legitimate values that non-medial wall vertices might take on.

`subcortVol, subcortLabs, subcortMask`

`subcortVol` represents the data values of the subcortex. It is either a 3D/4D numeric array ($i \times j \times k \times T$), or a vectorized matrix (V_S voxels by T measurements). If it's vectorized, the voxels should be in spatial order (i index increasing fastest, then j, then k).

`subcortLabs` represents the brainstructure labels of each voxel: see `substructure_table`. It is either a 3D data array ($i \times j \times k$) of integer brainstructure indices, or a V_S length vector in spatial order with brainstructure names as factors or integer indices. The indices should be 3-21 (1 and 2 correspond to left and right cortex, respectively) or 1-19 (cortex labels omitted), with 0 representing out-of-mask voxels.

`subcortMask` is logical 3D data array ($i \times j \times k$) where `TRUE` values indicate subcortical voxels (in-mask). If it is not provided, the mask will be inferred from voxels with labels `0`, `NA`, or `NaN` in `subcortLabs`. If `subcortLabs` are vectorized and `subcortMask` is not provided, the mask cannot be inferred so an error will occur.

`surfL, surfR` (Optional) Surface geometries for the left or right cortex. Can be a surface GIFTI file path or `"surf"` object; see `make_surf` for a full description of valid inputs.

`col_names` Names of each measurement/column in the data.

`HCP_32k_auto_mwall`

If left and/or right cortex data is provided, and the number of vertices matches that of the HCP 32k mesh (29696 on left, and 29716 on right), should the medial wall masks be added to the `"xifti"` if not provided? Default: `TRUE`.

`validate` Validate that the result is a `"xifti"`? Default: `TRUE`. If `FALSE`, the result may not be properly formatted if the inputs were invalid.

Details

Each data or surface component is optional. Metadata components (`cortex[L/R]_mwall`, `subcortLabs`, and `subcortMask`) will be ignored if its corresponding data component is not provided. If no data or surface components are provided, then the `template_xifti` will be returned.

If cortical data are provided without a corresponding medial wall mask, or if the provided mask is invalid or empty, then the medial wall will be inferred from data rows that are constantly a value in `mwall_values`. But if `mwall_values` is `NULL`, no attempt to infer the medial wall will be made and the medial wall metadata entry will be `NULL`.

The total number of grayordinates will be $G = (V_L - mwall_L) + (V_R - mwall_R) + V_S$: $V_L - mwall_L$ left vertices, $V_R - mwall_R$ right vertices and V_S subcortical voxels. T, the total number of measurements (columns of data), must be the same for each brainstructure.

Value

A `"xifti"`

See Also

Other reading: `info_cifti()`, `load_parc()`, `load_surf()`, `read_cifti()`, `read_surf()`, `read_xifti2()`

`boundary_mask_surf` *Boundary region of a mask*

Description

Identify the vertices within `boundary_width` edges of a vertex in the input mask on a triangular mesh. Returns a logical indicating if a vertex is within `boundary_width` edges of the mask.

Usage

```
boundary_mask_surf(faces, mask, boundary_width = 10)
```

Arguments

`faces`	An $F \times 3$ matrix, where each row contains the vertex indices for a given triangular face in the mesh. F is the number of faces in the mesh.
`mask`	A length V logical vector indicating if each vertex is within the input mask.
`boundary_width`	A positive integer representing the width of the boundary to compute. The furthest vertices from the input mask will be this number of edges away from the closest vertex in the input mask. Default: `10`.

Value

A length-V logical vector. Each entry corresponds to the vertex with the same index. The value is true if a vertex is within `boundary_width` edges of a vertex in the mesh, but is not within the mesh itself.

See Also

Other surface-related: `add_surf()`, `edit_mask_surf()`, `even_vert_samp()`, `is.surf()`, `load_surf()`, `mask_surf()`, `read_surf()`, `resample_surf()`, `rotate_surf()`, `surf_area()`, `view_surf()`, `write_surf_gifti()`

ciftiTools *ciftiTools: Tools for Reading and Visualizing CIFTI Brain Files*

Description

Here are groups of commonly-used functions in ciftiTools:

Details

Functions for reading in CIFTI or GIFTI data:

read_xifti: Read in a CIFTI file as a "xifti"

read_xifti2: Read in GIFTI files as a "xifti"

as.xifti: Combine numeric data to form a "xifti"

read_surf: Read in a surface GIFTI file as a "surf"

info_cifti: Read the metadata in a CIFTI file

load_surf: Read in a surface included in ciftiTools

load_parc: Read in a parcellation included in ciftiTools

Functions for writing CIFTI or GIFTI data:

write_cifti: Write a "xifti" to a CIFTI file

write_xifti2: Write a "xifti" to GIFTI and NIFTI files

write_metric_gifti: Write a numeric data matrix to a metric GIFTI file

write_surf_gifti: Write a "surf" to a surface GIFTI file

write_subcort_nifti: Write subcortical data to NIFTI files

separate_cifti: Separate a CIFTI file into GIFTI and NIFTI files

Functions for manipulating "xifti"s:

apply_xifti: Apply a function along the rows or columns of the "xifti" data matrix

combine_xifti: Combine multiple "xifti"s with non-overlapping brain structures

convert_xifti: Convert the intent of a "xifti"

merge_xifti: Concatenate data matrices from multiple "xifti"s

newdata_xifti: Replace the data matrix in a "xifti"

remove_xifti: Remove a brain structure or surface from a "xifti"

select_xifti: Select data matrix columns of a "xifti"

transform_xifti: Apply a univariate transformation to a "xifti" or pair of "xifti"s

add_surf: Add surfaces to a "xifti"

move_from_mwall: Move medial wall vertices back into the "xifti" data matrix

move_to_mwall: Move rows with a certain value into the "xifti" medial wall mask

S3 methods for "xifti"s:

summary **and** print**:** Summarize the contents.

as.matrix**:** Convert data to a locations by measurements numeric matrix.

dim**:** Obtain number of locations and number of measurements.

plot**:** Visualize the cortical surface and/or subcortical data.

+**,** -**,** *****, /**,** ^**,** %%**,** %/%**:** Operation between a "xifti" and a scalar, or between two "xifti"s.

abs**,** ceiling**,** exp**,** floor**,** log**,** round**,** sign**, and** sqrt**:** Univariate transformation of "xifti" data.

Functions for working with surfaces:

read_surf**:** Read in a surface GIFTI file as a "surf"

is.surf**:** Verify a "surf"

write_surf_gifti**:** Write a "surf" to a surface GIFTI file

view_surf**:** Visualize a "surf"

resample_surf**:** Resample a "surf"

rotate_surf**:** Rotate the geometry of a "surf"

Author(s)

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

Authors:

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

Other contributors:

- John Muschelli <muschellij2@gmail.com> (ORCID) [contributor]

See Also

Useful links:

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

ciftiTools.files *ciftiTools files*

Description

CIFTI and surface GIFTI files included in the `ciftiTools` package

Usage

```
ciftiTools.files()
```

Details

The CIFTI files are from NITRC: cifti-2_test_data-1.2.zip at https://www.nitrc.org/frs/?group_id=454

The surfaces are from the HCP and are included according to these data use terms: Data were provided [in part] by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.

Only the inflated surfaces are available as GIFTI files. To access the other surfaces included in the package (very inflated and midthickness), see `load_surf`.

Value

a list of file paths

ciftiTools.getOption *Get a* `ciftiTools` *option*

Description

Gets an R option (with prefix "ciftiTools_") value. See `ciftiTools.listOptions`.

Usage

```
ciftiTools.getOption(opt)
```

Arguments

`opt`	The option.

Value

The value, `val`

ciftiTools.listOptions *List* ciftiTools *options*

Description

List ciftiTools options

Usage

```
ciftiTools.listOptions()
```

Value

data.frame describing the options

ciftiTools.setOption *Set a* ciftiTools *option*

Description

Sets an R option (with prefix "ciftiTools_"). See ciftiTools.listOptions.

Usage

```
ciftiTools.setOption(opt, val)
```

Arguments

opt	The option.
val	The value to set the option as.

Value

The new value, val

combine_xifti *Combine "xifti"s with non-overlapping brain structures*

Description

Combine two to three "xifti"s with non-overlapping brain structures into a single "xifti". The names, intent, and surfaces of the first will be used, if present. To add more surfaces to the result, use add_surf.

Usage

```
combine_xifti(..., xii_list = NULL, meta = c("first", "all"))
```

Arguments

...	The "xifti" objects
xii_list	Alternatively, a list of "xifti" objects. If specified, will ignore ...
meta	"first" (default) to just use the metadata from the first argument, or "all" to include the other metadata in a list.

Value

A "xifti" with data from the inputs

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

convert_to_dlabel *Convert the intent of a CIFTI file or "xifti" object*

Description

Convert the intent of a CIFTI file or "xifti" object

Usage

```
convert_to_dlabel(
  x,
  cifti_target_fname = NULL,
  levels_old = NULL,
  levels = NULL,
  labels = NULL,
  nsig = Inf,
  colors = "Set2",
  add_white = TRUE,
  return_conversion_table = FALSE
)

convert_to_dscalar(x, cifti_target_fname = NULL, names = NULL)

convert_to_dtseries(
  x,
  cifti_target_fname = NULL,
  time_start = 0,
  time_step = 1,
  time_unit = c("second", "hertz", "meter", "radian")
)

convert_xifti(
  x,
  to = c("dscalar", "dtseries", "dlabel"),
  cifti_target_fname = NULL,
  ...
)
```

Arguments

`x` The CIFTI file name or `"xifti"` object to convert.

`cifti_target_fname`
File name for the converted CIFTI. Only used if `x` is a CIFTI file name. If `NULL` (default), will use the same name as `x` but with the extension updated.

`levels_old, levels, labels`
(Optional) `levels_old` is a vector of the original data values. They should all be unique. They may not all occur in the `"xifti"` data, but every datapoint in the `"xifti"` must occur in `levels_old`. If `levels_old` is not provided it will be set to the vector of all unique values in the data, in ascending order.

If `levels` is not provided, the original values will be re-mapped to integers from 0 to $N-1$ (the "Keys" of a "dlabel" CIFTI), with N being the length of `levels_old`. Otherwise, `levels` can be a vector the same length as `levels_old` specifying the corresponding new integers to use (rather than 0 to $N-1$). If `x` is already "dlabel", then by setting `levels_old` to the current label table values and `levels` to the desired new values, the data can be re-leveled (see examples

in function documentation). Note that duplicates in `levels_old` are allowed, to map multiple existing levels to the same new level.

New label names can be set with `labels`. If provided, it must be a character vector with the same length as `levels`. If there are duplicates in `levels`, the first label for a given level will be used. If `labels` is not provided, the new label names will be set to `levels` if it was provided, and `levels_old` if it was not.

Note: `NA` and `NaN` values are handled a bit differently. Data locations that are `NA` or `NaN` will remain unchanged. `NA` and `NaN` should not be included in `levels_old` or `levels`.

`nsig` Take this many significant digits for the data values. If `Inf` (default), do not round.

`colors` (Optional) "ROY_BIG_BL", the name of a ColorBrewer palette (see `RColorBrewer::brewer.pal.info` and colorbrewer2.org), the name of a viridisLite palette, or a character vector of colors. Default: `"Set2"`.

`add_white` Append white to the beginning of the colors? Default: `TRUE`.

`return_conversion_table`
Return the conversion table along with the converted `"xifti"`? Default: `FALSE`. It will give the original `values`, the `values_new` (i.e. the "Keys"), and the new `label` names.

`names` The column names. If `NULL` (default), will be set to "Column 1", "Column 2", ...

`time_start, time_step, time_unit`
(Optional) metadata for the new dtseries

`to` The desired intent: `"dscalar"` (default), `"dtseries"`, or `"dlabel"`

`...` Only used if `x` is a `"xifti"` object. Additional options specific to the target type and intent, e.g. for `convert_to_dlabel`.

Value

If `x` is a CIFTI, the target is a `"dlabel"` and `return_conversion_table`, a length-2 list with the first entry being the ".dlabel" `"xifti"` and the second being the conversion table. Otherwise, the `"xifti"` or the output CIFTI file name is directly returned.

Functions

- `convert_to_dlabel()`: Give the ".dlabel" intent (code 3007/ConnDenseLabel) to an input `"xifti"`. Will use the same label table for each data column. Can also be used to re-assign values in the label table, or to change label names.
- `convert_to_dscalar()`: Give the ".dscalar" intent (code 3006/ConnDenseScalar) to an input CIFTI file or `"xifti"` object. Can also be used to set the names for each column with `names`.
- `convert_to_dtseries()`: Give the ".dtseries" intent (code 3002/ConnDenseSeries) to an input `"xifti"` object. Can also be used to set the time metadata.

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

Examples

```
## Not run:
# Example: change label names
levels_old <- xii$meta$cifti$labels[[1]]$Key
newLabels <- paste0("New Label #", seq(length(levels_old)))
xii <- convert_to_dlabel(xii, levels_old=levels_old, levels=levels_old, labels=newLabels)
# Example: add an empty level
levels_old <- xii$meta$cifti$labels[[1]]$Key
levels_old <- c(levels_old, max(levels_old)+1)
labels <- c(rownames(xii$meta$cifti$labels[[1]]), "Empty")
xii <- convert_to_dlabel(xii, levels_old=levels_old, levels=levels_old, labels=labels)
# Example: set all but the lowest value to the same value & re-label
levels_old <- xii$meta$cifti$labels[[1]]$Key
levels <- ifelse(levels_old==min(levels_old), min(levels_old), min(levels_old)+1)
labels <- ifelse(levels_old==min(levels_old), "Minimum", "Not minimum")
xii <- convert_to_dlabel(xii, levels_old=levels_old, levels=levels, labels=labels)

## End(Not run)
```

---

dim.xifti *Dimensions of a "xifti"*

Description

Returns the number of rows (vertices + voxels) and columns (measurements) in the "xifti" data.

Usage

```
## S3 method for class 'xifti'
dim(x)
```

Arguments

x	A "xifti" object.

Value

The number of rows and columns in the "xifti" data.

edit_mask_surf *Edit mask on surface*

Description

Erode, dilate, or get the borders of a mask along the cortical surface

Usage

```
edit_mask_surf(
  x,
  mwall = NULL,
  surf = NULL,
  hemisphere = c("left", "right"),
  do = c("erode", "dilate", "borders"),
  depth = 1
)

erode_mask_surf(
  x,
  mwall = NULL,
  surf = NULL,
  hemisphere = c("left", "right"),
  depth = 1
)

dilate_mask_surf(
  x,
  mwall = NULL,
  surf = NULL,
  hemisphere = c("left", "right"),
  depth = 1
)
```

Arguments

x, mwall	Vector of the data mask to edit, and the medial wall mask. These can be specified in two ways. First, mwall can be a logical vector with each entry corresponding to a vertex as the cortical surface, and using FALSE values to indicate medial wall vertices. In this first case, x should then be a logical vector with each entry corresponding to a TRUE value in mwall. TRUE values in x should indicate the mask to be edited. Second, mwall can be NULL (default) in which case x should then be a logical vector with each entry corresponding to a vertex on the cortical surface. TRUE values in x should indicate the mask to be edited. In either case, xii$data$cortex_left[,1] and xii$meta$cortex$medial_wall_mask$left should work.

`surf, hemisphere`
: Provide one: the surface in the same resolution as the data, or the name of the hemisphere of the surface to resample and use (default: resample the left surface).

`do`
: `"erode"` (default), `"dilate"`, or `"borders"`. `"erode"` removes faces with at least one vertex not inside the mask. `"dilate"` adds faces with at least one vertex inside the mask. `"borders"` obtains the vertices inside the mask which share a face with at least one vertex not inside the mask.

`depth`
: How many iterations of the edit? Default: `1`. Does not apply to `"borders"`.

Details

The depth of the edit is determined by the number of edges between the vertices. To erode or dilate based on spatial distance (mm), see `-cifti-dilate` and `-cifti-erode`.

Value

`x` after erosion or dilation.

See Also

Other surface-related: `add_surf()`, `boundary_mask_surf()`, `even_vert_samp()`, `is.surf()`, `load_surf()`, `mask_surf()`, `read_surf()`, `resample_surf()`, `rotate_surf()`, `surf_area()`, `view_surf()`, `write_surf_gifti()`

---

even_vert_samp *Evenly sample vertices of mesh*

Description

Get a subset of the mesh vertices that are spatially evenly-sampled, by resampling the mesh and choosing the original vertices closest (Euclidian distance) to the new vertices.

Usage

```
even_vert_samp(surf, n_vert)
```

Arguments

`surf`
: A `"surf"` object

`n_vert`
: The desired number of vertices in the evenly-spaced sample. Note that the actual size of the subset will likely be close to but not exactly `n_vert` because it depends on the size of the resampled surface.

Value

An integer vector giving the indices of the vertices in the subset.

See Also

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), is.surf(), load_surf(), mask_surf(), read_surf(), resample_surf(), rotate_surf(), surf_area(), view_surf(), write_surf_gifti()

---

expand_color_pal *Interpolates between entries in the input palette to make a larger palette with COLOR_RES entries.*

Description

Interpolates between entries in the input palette to make a larger palette with COLOR_RES entries.

Usage

```
expand_color_pal(pal, COLOR_RES = 255)
```

Arguments

pal	The color palette to expand, as a data.frame with two columns: "color" (character: color hex codes) and "value" (numeric, ascending).
COLOR_RES	The number of entries to have in the output palette.

Value

A data.frame with two columns: "color" (character: color hex codes) and "value" (numeric)

See Also

Other coloring: ROY_BIG_BL(), make_color_pal(), use_color_pal()

---

faces_Param *faces*

Description

faces

Arguments

faces	An $F \times 3$ matrix, where each row contains the vertex indices for a given triangular face in the mesh. F is the number of faces in the mesh.

fix_xifti *Fix a "xifti"*

Description

Make adjustments to a putative "xifti" so that it is valid. Each adjustment is reported.

Usage

```
fix_xifti(xifti, verbose = TRUE)
```

Arguments

xifti	A "xifti" object.
verbose	Report each adjustment? Default: TRUE

Details

Right now it only coerces the data to numeric matrices.

Value

The fixed "xifti"

get_wb_cmd_path *Get the Connectome Workbench command path*

Description

Retrieves the path to the Connectome Workbench executable from a file path that may point to the executable itself, or to the Workbench folder which contains it (i.e., "path/to/workbench/bin_linux64/wb_command" or "path/to/workbench".)

Usage

```
get_wb_cmd_path(wb_path)
```

Arguments

wb_path	(Optional) Path to the Connectome Workbench folder or executable.

Value

The path to the Connectome Workbench executable

infer_resolution *Infer resolution from "xifti" and surfaces*

Description

Infer the numbers of vertices on each cortex of a "xifti" object. Also supports the result of info_cifti.

Usage

```
infer_resolution(xifti, surfL = NULL, surfR = NULL)
```

Arguments

xifti	A "xifti" object.
surfL	Left surface
surfR	Right surface

Value

The inferred resolutions for the left and right cortex.

info_cifti *Get CIFTI metadata*

Description

Get CIFTI metadata from the NIFTI header and XML using the Connectome Workbench command -nifti-information. The information is formatted as the meta component in a "xifti" object (see template_xifti), and includes:

1. medial wall masks for the left and right cortex
2. the subcortical labels (ordered spatially)
3. the subcortical mask
4. other NIFTI intent-specific metadata

Usage

```
info_cifti(cifti_fname)

infoCIfTI(cifti_fname)

infocii(cifti_fname)
```

Arguments

`cifti_fname`	File path to a CIFTI file (ending in ".d*.nii").

Details

Additional metadata depends on the type of CIFTI file:

"dtseries" **time_start:** Start time
time_step: The TR
time_unit: Unit of time

"dscalar" **names:** Name of each data column

"dlabels" **names:** (Names of each data column.)
labels: (List of $L \times 5$ data.frames. Row names are the label names. Column names are Key, Red, Green, Blue, and Alpha. List entry names are the names of each data column.)

Value

The metadata component of a `"xifti"` for the input CIFTI file

Connectome Workbench

This function interfaces with the `"-nifti-information"` Workbench command.

Label Levels

`xifti$meta$subcort$labels` is a factor with the following levels:

1. Cortex-L
2. Cortex-R
3. Accumbens-L
4. Accumbens-R
5. Amygdala-L
6. Amygdala-R
7. Brain Stem
8. Caudate-L
9. Caudate-R
10. Cerebellum-L
11. Cerebellum-R
12. Diencephalon-L
13. Diencephalon-R
14. Hippocampus-L
15. Hippocampus-R
16. Pallidum-L

17. Pallidum-R
18. Putamen-L
19. Putamen-R
20. Thalamus-L
21. Thalamus-R

These correspond to the same structures as given by `ft_read_cifti` in the `cifti-matlab` MATLAB toolbox. Note that the first two levels (left and right cortex) are not used.

See Also

Other reading: `as.xifti()`, `load_parc()`, `load_surf()`, `read_cifti()`, `read_surf()`, `read_xifti2()`

`is.cifti` *Validate a "xifti" object*

Description

Check if object is valid for a "xifti". This alias for `is.xifti` is offered as a convenience, and a message will warn the user. We recommend using `is.xifti` instead.

Usage

```
is.cifti(x, messages = TRUE)

is_cifti(x, messages = TRUE)

isCIfTI(x, messages = TRUE)
```

Arguments

x	The putative "xifti".
messages	If x is not a "xifti", print messages explaining the problem? Default is TRUE.

Details

Requirements: it is a list with the same structure as `template_xifti`. The size of each data entry must be compatible with its corresponding mask (medial wall for the cortex and volumetric mask for the subcortex). Metadata should be present if and only if the corresponding data is also present. The surfaces can be present whether or not the cortex data are present.

See the "Label Levels" section for the requirements of `xifti$meta$subcort$labels`.

Value

Logical. Is x a valid "xifti"?

Label Levels

`xifti$meta$subcort$labels` is a factor with the following levels:

1. Cortex-L
2. Cortex-R
3. Accumbens-L
4. Accumbens-R
5. Amygdala-L
6. Amygdala-R
7. Brain Stem
8. Caudate-L
9. Caudate-R
10. Cerebellum-L
11. Cerebellum-R
12. Diencephalon-L
13. Diencephalon-R
14. Hippocampus-L
15. Hippocampus-R
16. Pallidum-L
17. Pallidum-R
18. Putamen-L
19. Putamen-R
20. Thalamus-L
21. Thalamus-R

These correspond to the same structures as given by `ft_read_cifti` in the `cifti-matlab` MATLAB toolbox. Note that the first two levels (left and right cortex) are not used.

See Also

Other common: `read_cifti()`, `resample_cifti()`, `smooth_cifti()`, `view_xifti()`, `write_cifti()`

is.surf *Validate a "surf" object (vertices + faces)*

Description

Check if object is valid for `xifti$surf$cortex_left` or `xifti$surf$cortex_right`, where `xifti` is a `"xifti"` object.

Usage

```
is.surf(x)
```

Arguments

x	The putative `"surf"`.

Details

This is a helper function for `is.xifti`.

Requirements: the `"surf"` must be a list of three components: "vertices", "faces", and "hemisphere". The first two should each be a numeric matrix with three columns. The values in "vertices" represent spatial coordinates whereas the values in "faces" represent vertex indices defining the face. Thus, values in "faces" should be integers between 1 and the number of vertices. The last list entry, "hemisphere", should be "left", "right", or NULL indicating the brain hemisphere which the surface represents.

Value

Logical. Is `x` a valid `"surf"`?

See Also

Other surface-related: `add_surf()`, `boundary_mask_surf()`, `edit_mask_surf()`, `even_vert_samp()`, `load_surf()`, `mask_surf()`, `read_surf()`, `resample_surf()`, `rotate_surf()`, `surf_area()`, `view_surf()`, `write_surf_gifti()`

is.xifti *Validate a "xifti" object.*

Description

Check if object is valid for a `"xifti"` object.

Usage

```
is.xifti(x, messages = TRUE)

is_xifti(x, messages = TRUE)
```

Arguments

x	The putative "xifti" object.
messages	If x is not a "xifti" object, print messages explaining the problem? Default is TRUE.

Details

Requirements: it is a list with the same structure as template_xifti. The size of each data entry must be compatible with its corresponding mask (medial wall for the cortex and volumetric mask for the subcortex). Metadata should be present if and only if the corresponding data is also present. The surfaces can be present whether or not the cortex data are present.

See the "Label Levels" section for the requirements of xifti$meta$subcort$labels.

Value

Logical. Is x a valid "xifti" object?

Label Levels

xifti$meta$subcort$labels is a factor with the following levels:

1. Cortex-L
2. Cortex-R
3. Accumbens-L
4. Accumbens-R
5. Amygdala-L
6. Amygdala-R
7. Brain Stem
8. Caudate-L
9. Caudate-R
10. Cerebellum-L
11. Cerebellum-R
12. Diencephalon-L
13. Diencephalon-R
14. Hippocampus-L
15. Hippocampus-R
16. Pallidum-L

17. Pallidum-R
18. Putamen-L
19. Putamen-R
20. Thalamus-L
21. Thalamus-R

These correspond to the same structures as given by `ft_read_cifti` in the `cifti-matlab` MATLAB toolbox. Note that the first two levels (left and right cortex) are not used.

`load_parc` *Load a parcellation included in* `ciftiTools`

Description

Load a parcellation included in `ciftiTools`.

Usage

```
load_parc(
  name = c("Schaefer_100", "Schaefer_400", "Schaefer_1000", "Yeo_7", "Yeo_17")
)
```

Arguments

`name`	The name of the parcellation to load: `"Schaefer_100"`: (2018) 100 parcels based on the "local-global" approach. `"Schaefer_400"`: (2018) 400 parcels based on the "local-global" approach. `"Schaefer_1000"`: (2018) 1000 parcels based on the "local-global" approach. `"Yeo_7"`: (2011) 7 networks based on fcMRI clustering. Networks are further divided into 51 components. `"Yeo_17"`: (2011) 17 networks based on fcMRI clustering. Networks are further divided into 114 components. `NULL` (default) will load the first choice, where applicable. This argument will affect the indices, colors, and names of each parcel, but not the parcel boundaries.

Details

When using these parcellations, please cite the corresponding paper(s):

1. Yeo, B. T. T. et al. The organization of the human cerebral cortex estimated by intrinsic functional connectivity. J Neurophysiol 106, 1125-1165 (2011).
2. Schaefer, A. et al. Local-Global Parcellation of the Human Cerebral Cortex from Intrinsic Functional Connectivity MRI. Cereb Cortex 28, 3095-3114 (2018).
3. Kong, R. et al. Individual-Specific Areal-Level Parcellations Improve Functional Connectivity Prediction of Behavior. Cerebral Cortex (2021+) doi:10.1093/cercor/bhab101.

Note that the Schaefer parcels have been matched to networks from Kong (2021+).

Value

The parcellation as a dlabel "xifti" with one column. Each key represents one unique parcel.

See Also

Other reading: as.xifti(), info_cifti(), load_surf(), read_cifti(), read_surf(), read_xifti2()

Other parcellation-related: apply_parc(), load_sub_parc(), parc_add_subcortex(), parc_borders(), parc_vals_to_xifti()

---

load_surf *Load a* "surf" *included in* ciftiTools

Description

Load a "surf" object from one of the three 32k anatomical surfaces included in ciftiTools.

Usage

```
load_surf(
  hemisphere = c("left", "right"),
  name = c("inflated", "very inflated", "midthickness"),
  resamp_res = NULL
)
```

Arguments

hemisphere	"left" (default) or "right"
name	The name of the surface geometry to load: "inflated" (default), "very inflated", and "midthickness".
resamp_res	The resolution to resample the surfaces to. If NULL (default) or 32492, do not resample. Note that the barycentric resampling method, which is recommended for anatomical surfaces, will be used.

Details

The surfaces are from the HCP and are included according to these data use terms: Data were provided [in part] by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.

Value

The "surf" object

See Also

Other reading: as.xifti(), info_cifti(), load_parc(), read_cifti(), read_surf(), read_xifti2()

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), mask_surf(), read_surf(), resample_surf(), rotate_surf(), surf_area(), view_surf(), write_surf_gifti()

---

make_color_pal *Make a color palette.*

Description

Control the mapping of values to colors with colors, color_mode, and zlim.

Usage

```
make_color_pal(
  colors = NULL,
  color_mode = c("sequential", "qualitative", "diverging"),
  zlim = NULL
)
```

Arguments

colors	(Optional) "ROY_BIG_BL", the name of a ColorBrewer palette (see RColorBrewer::brewer.pal.info and colorbrewer2.org), the name of a viridisLite palette, or a character vector of colors. NULL (default) will use "ROY_BIG_BL" if color_mode is "sequential" or "diverging", and "Set2" if color_mode is "qualitative". See the description for more details.
color_mode	(Optional) "sequential", "qualitative", or "diverging". Default: "sequential". See the description for more details.
zlim	(Optional) Controls the mapping of values to each color in colors. See the description for more details.

Details

There are three kinds of arguments for colors: "ROY_BIG_BL", the name of a ColorBrewer palette (see RColorBrewer::brewer.pal.info and colorbrewer2.org), the name of a viridisLite palette, or a character vector of color names.

If colors=="ROY_BIG_BL", the "ROY_BIG_BL" palette will be used. It is the same palette as the default for the Connectome Workbench application (https://github.com/Washington-University/workbench/blob/master/src/F The midpoint will be colored black. From the midpoint toward the upper bound, colors will proceed from black to red to yellow. From the midpoint toward the lower bound, colors will proceed from black to blue to purple to green to aqua. Here is how each color mode behaves if colors=="ROY_BIG_BL":

`color_mode=="sequential"` Only half of the palette will be used. If `zlim` is length 2, the higher value will be the maximum and the lower value will be the minimum. Set `zlim[1] > zlim[2]` to reverse the color scale. (Note that the second half, black –> red –> yellow, is used by default. To use the negative half specify `colors=="ROY_BIG_BL_neg"` instead. It will also be used automatically by `xifti_read_surface` when the data range is negative.) `zlim` can also be length 10, in which case each value corresponds to the position of an individual color in the half palette.

`color_mode=="qualitative"` "ROY_BIG_BL" is not recommended for qualitative data, so a warning will be issued. Palette colors will be selected from the landmark "ROY_BIG_BL" colors, with interpolated colors added if the number of colors in the palette (18) is less than this range. `zlim` should be a single number: the number of unique colors to get.

`color_mode=="diverging"` If `zlim` is length 2 or 3, the lowest number will be the lower bound and the highest number will be the upper bound. If `zlim` is length 3, the middle number will be the midpoint (black). The lower and upper bounds will be aqua and yellow, respectively, except if `zlim` is in descending order, in which case the color scale will be reversed (lowest is yellow; highest is aqua). `zlim` can also be length 19, in which case each value corresponds to the position of an individual color in the palette.

If `colors` is the name of an RColorBrewer palette (see `RColorBrewer::brewer.pal.info`) or viridisLite palette, the colors in that palette will be used, and the following behavior applies. If `colors` is a character vector of color names (hex codes or standard R color names), the following behavior applies directly:

`color_mode=="sequential"` If `zlim` is length 2, the higher value will be the maximum and the lower value will be the minimum. Set `zlim[1] > zlim[2]` to reverse the color scale. `zlim` can also be the same length as the palette, in which case each value corresponds to the position of an individual color in the palette.

`color_mode=="qualitative"` `zlim` should be a single number: the number of unique colors to get. Color interpolation will be used if the number of colors in the palette is less than this range. If `length(zlim)==length(colors)`, each color will be mapped to each corresponding value.

`color_mode=="diverging"` If `zlim` is length 2 or 3, the lowest number will be the lower bound and the highest number will be the upper bound. If `zlim` is length 3, the middle number will be the midpoint. Set `zlim` in descending order to reverse the color scale. `zlim` can also be the same length as the palette, in which case each value corresponds to the position of an individual color in the palette.

Value

A data.frame with two columns: `"color"` (character: color hex codes) and `"value"` (numeric)

See Also

Other coloring: `ROY_BIG_BL()`, `expand_color_pal()`, `use_color_pal()`

mask_Param_vertices *mask: vertices*

Description

mask: vertices

Arguments

mask	A length V logical vector indicating if each vertex is within the input mask.

mask_surf *Mask surface*

Description

Mask a surface mesh.

Usage

```
mask_surf(surf, mask)
```

Arguments

surf	A "surf" object
mask	A length V logical vector indicating if each vertex is within the input mask.

Details

Apply a binary mask to a "surf" object (list of vertices and corresponding faces). Vertices not in the mask are removed, and faces (triangles) with any vertices not in the mask are removed. Finally, vertex numbering for the new faces matrix is corrected.

Value

The masked "surf" object.

See Also

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), read_surf(), resample_surf(), rotate_surf(), surf_area(), view_surf(), write_surf_gifti()

merge_xifti *Concatenate "xifti"s*

Description

Concatenate "xifti" objects along the columns. They must have the same brainstructures and resolutions. The first "xifti"'s metadata will be retained, including its intent.

Usage

```
merge_xifti(..., xifti_list = NULL)
```

Arguments

..., xifti_list	Provide as arguments the "xifti"s to concatenate, OR the single argument xifti_list which should be a list of "xifti"s. (If xifti_list is provided all other inputs will be ignored.)

Value

The concatenated "xifti"

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

move_from_mwall *Move data locations from medial wall*

Description

Move all medial wall locations into the cortical data matrices by assigning them a specific value (e.g. NA).

Usage

```
move_from_mwall(xifti, value = NA, name = "Medial_Wall", RGBA = c(1, 1, 1, 0))
```

Arguments

xifti	A "xifti" object.
value	The value to assign the medial wall locations. Default: NA.
name, RGBA	Only used if the "xifti" has the dlabel intent and value is not an already-existing Key. This is the name to assign to the new key for the medial wall locations, as well as a length-four numeric vector indicating the red, green, blue, and alpha values for the color to assign to the new key. These will be reflected in the updated label table. Note that RGBA values must all be in [0, 1]. Currently, only one name and set of RGBA values are supported, meaning that the medial wall locations will have the same Key, name, and color across all data columns in the "xifti". An error will occur if the Key already exists for some columns but not others. Defaults: "Medial_Wall" for "name" and white with 0 alpha for RGBA.

Value

The "xifti" with re-organized data and medial wall masks

See Also

move_to_mwall

unmask_cortex

---

move_to_mwall *Move data locations to the medial wall*

Description

Move cortical data locations with a specific value(s) to the medial wall mask. For example, dlabel CIFTIs often have medial wall vertices set to a specific key value, rather than a medial wall mask. This function can move those data locations from the data matrix to the medial wall mask in the metadata.

Usage

```
move_to_mwall(xifti, values = c(NA, NaN), drop = FALSE)
```

Arguments

xifti	A "xifti" object.
values	Medial wall values. Default: NA and NaN. Data locations in the left and right cortex that are one of these values (across all columns) will be moved to the medial wall mask in the metadata.
drop	Only used if the "xifti" has the dlabel intent. Drop the key(s) in values from the label tables, for columns in which they no longer exist? Default: FALSE.

Value

The "xifti" with re-organized data and medial wall masks

See Also

move_from_mwall

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_temp scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

---

newdata_xifti *Replace the data in a "xifti"*

Description

Replace the data in a "xifti" with new data from a data matrix.

Usage

```
newdata_xifti(xifti, newdata, newnames = NULL)
```

Arguments

xifti	A "xifti" object.
newdata	The $V \times T$ matrix of data values to replace those in xifti with. The left cortex vertices should be at the top, right cortex vertices in the middle, and subcortex vertices at the bottom (when present). If newdata is instead a $V \times Q$ matrix where Q is not T, then any column names or label tables will be removed. (A "dlabel" will be converted to a "dscalar".) Can also be a length-one vector to set all values equally.
newnames	Replaces the names in the xifti. If NULL (default), keep the original names, except if the number of columns in newdata doesn't match that of xifti, in which case no names will be used.

Details

If the "xifti" has V grayordinates and T measurements\, newdata should be a $V \times Q$ matrix. If Q is not equal to T, then any column names or label tables will be removed. (A "dlabel" will be converted to a "dscalar".)

Value

The new "xifti"

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_temp scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

---

parc_add_subcortex *Add subcortex to cortical parcellation*

Description

Add the subcortex, with each brain structure as a separate parcel, to a "dlabel" cortical parcellation.

Usage

```
parc_add_subcortex(parc, parc_sub = "MNI")
```

Arguments

parc	A single-column "dlabel" "xifti" object without subcortical data.
parc_sub	A single-column "xifti" object with only subcortical data. Or, "MNI" (default) to read in and use the MNI subcortex included in ciftiTools. (The Connectome Workbench is required.)

Value

The new parcellation with added subcortical data and labels.

See Also

Other parcellation-related: apply_parc(), load_parc(), load_sub_parc(), parc_borders(), parc_vals_to_xifti()

---

parc_borders *Parcellation borders*

Description

Identify vertices which lie on the border of different parcels.

Usage

```
parc_borders(parc, surf = NULL, hemisphere = c("left", "right"))
```

Arguments

parc	Integer vector the same length as the number of vertices. Each entry indicates the parcel that vertex belongs to.
surf	The surface which the vertices belong to, or just the "faces" component ($F \times 3$ matrix where each row indicates the vertices which comprise a face). If not provided, the (resampled) default hemisphere surface included with ciftiTools will be used.
hemisphere	Only used to choose which default surface to use if is.null(surf). Should be "left" (default) or "right".

Value

Logical vector the same length as parc indicating if the vertex lies on a border.

See Also

Other parcellation-related: apply_parc(), load_parc(), load_sub_parc(), parc_add_subcortex(), parc_vals_to_xifti()

---

parc_vals_to_xifti *Convert parcellation values to "xifti"*

Description

From a parcellation and a corresponding value matrix, make a "xifti" object that has the value vector of each parcel across its locations.

Usage

```
parc_vals_to_xifti(parc, vals)
```

Arguments

parc	A single-column "dlabel" "xifti" object.
vals	A numeric matrix. Rows should correspond to rows in the color table of parc. Columns will become columns in the output "xifti" object.

Value

A "xifti" object

See Also

Other parcellation-related: apply_parc(), load_parc(), load_sub_parc(), parc_add_subcortex(), parc_borders()

plot.surf *S3 method: plot surface*

Description

Visualize a single surface

Usage

```
## S3 method for class 'surf'
plot(x, ...)
```

Arguments

x	A "surf" object
...	Additional arguments to view_xifti_surface. But, the hemisphere argument behaves differently: it can be either left or right to indicate which hemisphere x represents. It is only used if the "hemisphere" metadata entry in x is NULL. If both the argument and the metadata entry are NULL, the surface will be treated as the left hemisphere.

plot.xifti *S3 method: use* view_xifti *to plot a* "xifti" *object*

Description

S3 method: use view_xifti to plot a "xifti" object

Usage

```
## S3 method for class 'xifti'
plot(x, ...)
```

Arguments

x	A "xifti" object.
...	Additional arguments to view_xifti, except what, which will be set to NULL.

read_cifti *Read a CIFTI file*

Description

Read in a CIFTI file as a "xifti" object.

Usage

```
read_cifti(
  cifti_fname = NULL,
  surfL_fname = NULL,
  surfR_fname = NULL,
  brainstructures = c("left", "right"),
  idx = NULL,
  resamp_res = NULL,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  flat = FALSE,
  mwall_values = c(NA, NaN),
  verbose = FALSE,
  ...
)

readCIfTI(
  cifti_fname = NULL,
  surfL_fname = NULL,
  surfR_fname = NULL,
  brainstructures = c("left", "right"),
  idx = NULL,
  resamp_res = NULL,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  flat = FALSE,
  mwall_values = c(NA, NaN),
  verbose = FALSE,
  ...
)

readcii(
  cifti_fname = NULL,
  surfL_fname = NULL,
  surfR_fname = NULL,
  brainstructures = c("left", "right"),
  idx = NULL,
```

```
  resamp_res = NULL,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  flat = FALSE,
  mwall_values = c(NA, NaN),
  verbose = FALSE,
  ...
)

read_xifti(
  cifti_fname = NULL,
  surfL_fname = NULL,
  surfR_fname = NULL,
  brainstructures = c("left", "right"),
  idx = NULL,
  resamp_res = NULL,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  flat = FALSE,
  mwall_values = c(NA, NaN),
  verbose = FALSE,
  ...
)
```

Arguments

`cifti_fname`	File path to a CIFTI file (ending in ".d*.nii").
`surfL_fname`	(Optional) File path to a GIFTI surface geometry file representing the left cortex.
`surfR_fname`	(Optional) File path to a GIFTI surface geometry file representing the right cortex.
`brainstructures`	Character vector indicating which brain structure(s) to obtain: `"left"` (left cortex), `"right"` (right cortex) and/or `"subcortical"` (subcortex and cerebellum). Can also be `"all"` (obtain all three brain structures). Default: `c("left","right")` (cortex only). If a brain structure is indicated but does not exist in the CIFTI file, a warning will occur and that brain structure will be skipped.
`idx`	Numeric vector indicating the data indices (columns) to read. If `NULL` (default), read in all the data. Must be a subset of the indices present in the file, or an error will occur. For high-resolution CIFTI files, reading in only a subset of the data saves memory, but will be slower than reading in the entire file due to the required intermediate steps.
`resamp_res`	Resolution to resample the cortical data and surface to. Default: `NULL` (do not resample). If not `NULL`, the data will have to be read in with `-cifti-separate`, which is slower than `-cifti-convert -to-gifti-ext`.

resamp_method "barycentric" (default) or "adaptive" resampling for the metric or label data. These options correspond to the Workbench command options "BARYCENTRIC" and "ADAP_BARY_AREA", respectively.

While adaptive resampling is recommended for metric or label data, it requires that area[L/R]_original_fname be provided.

Note that surfaces will resampled using barycentric resampling regardless of resamp_method, because barycentric resampling rather than adaptive resampling is recommended for surface data.

areaL_original_fname, areaR_original_fname

File paths to the surfaces to use for vertex area correction during adaptive resampling. (Only used if resampling with the adaptive method.) area[L/R]_original_fname should match the current resolution of the data.

For resampling: the Workbench command for adaptive resampling requires the target surfaces for area correction too. But to make the workflow easier, ciftiTools will resample area[L/R]_original_fname with the barycentric method and use that for the target area.

For remapping: area[L/R]_target_fname must be directly provided.

flat Should the result be flattened into a single matrix?

If FALSE (default), the result will be a "xifti" object.

If TRUE, the result will be a $T \times G$ matrix (T measurements, G grayordinates not including the medial wall if it's excluded from the ROI). All below arguments will be ignored because the brain structures cannot be identified. Surfaces will not be appended. Resampling is also not possible. flat==TRUE is the fastest way to read in just the CIFTI data.

If TRUE, the grayordinates will be ordered by left cortex, right cortex, and then subcortex. Subcortical voxels will be ordered by alphabetical label. However, where each brainstructure (and subcortical structure) begins and ends cannot be determined. The medial wall locations and subcortical brain mask are also not included. The data matrix will be identical to that created by -cifti-convert -to-gifti-ext.

mwall_values If the medial wall locations are not indicated in the CIFTI, use these values to infer the medial wall mask. Default: c(NA, NaN). If NULL, do not attempt to infer the medial wall.

verbose Should occasional updates be printed? Default: FALSE.

... Additional arguments to read_cifti_convert or read_cifti_separate.

Details

First, metadata is obtained with info_cifti. Then, if no resampling is requested, the -cifti-convert -to-gifti-ext Workbench Command is used to "flatten" the data and save it as a metric or label GIFTI file, which is read in and separated by brainstructure according to the metadata (read_cifti_convert). Otherwise, if sampling is requested, then the CIFTI is separated into its GIFTI and NIFTI components, resampled, and then re-assembled (read_cifti_separate). The former is much faster for large CIFTI files, so the latter is only used when necessary for resampling.

If cifti_fname is not provided, then only the surfaces are read in.

Value

If `!flat`, a `"xifti"` object. Otherwise, a $T \times G$ matrix (T measurements, G grayordinates).

Connectome Workbench

This function interfaces with the `"-cifti-convert"` Workbench command if resampling is not needed, and the `"-cifti-separate"` Workbench command if resampling is needed.

Label Levels

`xifti$meta$subcort$labels` is a factor with the following levels:

1. Cortex-L
2. Cortex-R
3. Accumbens-L
4. Accumbens-R
5. Amygdala-L
6. Amygdala-R
7. Brain Stem
8. Caudate-L
9. Caudate-R
10. Cerebellum-L
11. Cerebellum-R
12. Diencephalon-L
13. Diencephalon-R
14. Hippocampus-L
15. Hippocampus-R
16. Pallidum-L
17. Pallidum-R
18. Putamen-L
19. Putamen-R
20. Thalamus-L
21. Thalamus-R

These correspond to the same structures as given by `ft_read_cifti` in the `cifti-matlab` MATLAB toolbox. Note that the first two levels (left and right cortex) are not used.

See Also

Other common: `is.cifti()`, `resample_cifti()`, `smooth_cifti()`, `view_xifti()`, `write_cifti()`

Other reading: `as.xifti()`, `info_cifti()`, `load_parc()`, `load_surf()`, `read_surf()`, `read_xifti2()`

read_surf *Get a "surf" object*

Description

Coerce a file path to a surface GIFTI, a "gifti" object, a list with entries "pointset" and "triangle", or a "surf" to a "surf".

Usage

```
read_surf(surf, expected_hemisphere = NULL, resamp_res = NULL)

make_surf(surf, expected_hemisphere = NULL, resamp_res = NULL)
```

Arguments

surf	Either a file path to a surface GIFTI; a "gifti" read by readgii; a list with entries "pointset" and "triangle"; or, a "surf" object.
expected_hemisphere	The expected hemisphere ("left" or "right") of surf. If the hemisphere indicated in the GIFTI metadata is the opposite, an error is raised. If NULL (default), use the GIFTI hemisphere.
resamp_res	The resolution to resample the surfaces to. If NULL (default), do not resample.

Value

The "surf": a list with components "vertices" (3D spatial locations), "faces" (defined by three vertices), and "hemisphere" ("left", "right", or NULL if unknown).

See Also

Other reading: as.xifti(), info_cifti(), load_parc(), load_surf(), read_cifti(), read_xifti2()

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), mask_surf(), resample_surf(), rotate_surf(), surf_area(), view_surf(), write_surf_gifti()

read_xifti2 *Read in GIFTI files as a "xifti" object*

Description

Read in GIFTI metric or label files as a "xifti" object. May also include surface geometry GIFTI files and perform resampling.

Usage

```
read_xifti2(
  cortexL = NULL,
  cortexL_mwall = NULL,
  cortexR = NULL,
  cortexR_mwall = NULL,
  mwall_values = c(NA, NaN),
  surfL = NULL,
  surfR = NULL,
  resamp_res = NULL,
  col_names = NULL,
  HCP_32k_auto_mwall = TRUE,
  read_dir = NULL,
  validate = TRUE
)
```

Arguments

`cortexL`, `cortexL_mwall`

Left cortex data and ROI. Each must be a path to a metric or label GIFTI file.

If `cortexL_mwall` is not provided, `cortexL` should have data for all vertices on the left cortical surface ($V_L \times T$ data matrix). There will not be a mask for the medial wall. Not providing the medial wall mask is appropriate for ".dlabels.nii" files where the medial wall may have its own label and therefore should not be treated as missing data.

If `cortexL_mwall` is provided, `cortexL` should either have data for all vertices on the left cortical surface ($V_L \times T$ data matrix, with filler values e.g. `0` or `NaN` for medial wall vertices), or have data only for non-medial wall vertices ($(V_L - mwall_L) \times T$ data matrix). The medial wall mask will be the `0` values in `cortexL_mwall`. The medial wall mask should be provided whenever the medial wall should be treated as missing data.

Since the unmasked cortices must have the same number of vertices, `V_L` should match `V_R`, or `resamp_res` must be set.

`cortexR`, `cortexR_mwall`

Right cortex data and ROI. Each must be a path to a metric or label GIFTI file.

If `cortexR_mwall` is not provided, `cortexR` should have data for all vertices on the right cortical surface ($V_R \times T$ data mre will not be a mask for the medial wall. Not providing the medial wall mask is appropriate for ".dlabels.nii" files where the medial wall may have its own label and therefore should not be treated as missing data.

If `cortexR_mwall` is provided, `cortexR` should either have data for all vertices on the right cortical surface ($V_R \times T$ data matrix, with filler values e.g. `0` or `NaN` for medial wall vertices), or have data only for non-medial wall vertices ($(V_R - mwall_R) \times T$ data matrix). The medial wall mask will be the `0` values in `cortexR_mwall`. The medial wall mask should be provided whenever the medial wall should be treated as missing data.

Since the unmasked cortices must have the same number of vertices, `V_L` should match `V_R`, or `resamp_res` must be set.

mwall_values — If `cortex[L/R]_mwall` was not provided, or if it was invalid (i.e. bad length or all `TRUE`), the medial wall mask will be inferred from rows in `cortex[L/R]` that are constantly one of these values. Default: `c(NA, NaN)`. If `NULL`, do not attempt to infer the medial wall from the data values. `NULL` should be used if `NA` or `NaN` are legitimate values that non-medial wall vertices might take on.

surfL, surfR — (Optional) File path(s) to surface GIFTI(s) for the left or right cortex.

resamp_res — Resolution to resample the cortical data and surface to. Default: `NULL` (do not resample). If provided, the original resolutions of the cortex data and surfaces may differ.

col_names — Names of each measurement/column in the data. Overrides names indicated in the data components.

HCP_32k_auto_mwall — If left and/or right cortex data is provided, and the number of vertices matches that of the HCP 32k mesh (29696 on left, and 29716 on right), should the medial wall masks be added to the `"xifti"` if not provided? Default: `TRUE`.

read_dir — (Optional) Append a directory to all file names in the arguments. If `NULL` (default), do not modify file names.

validate — Validate that the result is a `"xifti"` object? Default: `TRUE`. If `FALSE`, the result may not be properly formatted if the inputs were invalid.

Value

The `"xifti"` object containing all the data in the input giftis.

See Also

Other reading: `as.xifti()`, `info_cifti()`, `load_parc()`, `load_surf()`, `read_cifti()`, `read_surf()`

---

remap_cifti *Remap CIFTI data*

Description

Remap CIFTI data between two different spaces, such as between FreeSurfer fsaverage group data and fs_LR group data.

Usage

```
remap_cifti(
  x,
  cifti_target_fname = NULL,
  remap_method = c("adaptive", "barycentric"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  areaL_target_fname = NULL,
  areaR_target_fname = NULL,
```

```
  sphereL_original_fname = NULL,
  sphereR_original_fname = NULL,
  sphereL_target_fname = NULL,
  sphereR_target_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)

remapCIfTI(
  x,
  cifti_target_fname = NULL,
  remap_method = c("adaptive", "barycentric"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  areaL_target_fname = NULL,
  areaR_target_fname = NULL,
  sphereL_original_fname = NULL,
  sphereR_original_fname = NULL,
  sphereL_target_fname = NULL,
  sphereR_target_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)

remapcii(
  x,
  cifti_target_fname = NULL,
  remap_method = c("adaptive", "barycentric"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  areaL_target_fname = NULL,
  areaR_target_fname = NULL,
  sphereL_original_fname = NULL,
  sphereR_original_fname = NULL,
  sphereL_target_fname = NULL,
  sphereR_target_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)

remap_xifti(
  x,
  cifti_target_fname = NULL,
  remap_method = c("adaptive", "barycentric"),
  areaL_original_fname = NULL,
```

```
  areaR_original_fname = NULL,
  areaL_target_fname = NULL,
  areaR_target_fname = NULL,
  sphereL_original_fname = NULL,
  sphereR_original_fname = NULL,
  sphereL_target_fname = NULL,
  sphereR_target_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)
```

Arguments

x	The CIFTI file name or "xifti" object to resample.
cifti_target_fname	File name for the resampled CIFTI. Will be placed in write_dir. If NULL, will be written to "resampled.d*.nii". write_dir will be appended to the beginning of the path.
remap_method	"adaptive" (default) or "adaptive" resampling. These options correspond to the Workbench command options "BARYCENTRIC" and "ADAP_BARY_AREA", respectively. For remapping fs_LR group data to fsaverage, barycentric should be used. For remapping FreeSurfer fsaverage group data to fs_LR, adaptive should be used.
areaL_original_fname, areaL_target_fname	File paths to the left cortex surfaces to use for vertex area correction during adaptive resampling. Required if remap_method is "adaptive".
areaR_original_fname, areaR_target_fname, sphereR_original_fname, sphereR_target_fname	See the corresponding arguments for the left cortex.
sphereL_original_fname, sphereL_target_fname	File paths to the sphere surfaces in the original and target spaces, for the left cortex.
write_dir	Where to write the resampled CIFTI (and surfaces if present.) If NULL (default), will use the current working directory if x was a CIFTI file, and a temporary directory if x was a "xifti" object.
mwall_values	If the medial wall locations are not indicated in the CIFTI, and if ROIcortexL/R_original_fname is not provided, then use these values to infer the medial wall mask. Default: c(NA, NaN). If NULL, do not attempt to infer the medial wall. Correctly indicating the medial wall locations is important for remapping, because the medial wall mask is taken into account during remapping calculations.
verbose	Should occasional updates be printed? Default: TRUE.

Details

Can accept a "xifti" object as well as a path to a CIFTI-file. If the input "xifti" object has surface geometry, it will be removed.

This function is in active development: its arguments and behavior may change greatly in future versions of the package.

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

---

remap_gifti *Remap GIFTI metric or label data*

Description

Remap GIFTI metric or label data between two different spaces, such as between FreeSurfer fsaverage group data and fs_LR group data. This function is a wrapper to resample_gifti.

Usage

```
remap_gifti(
  original_fname,
  target_fname,
  hemisphere = c("left", "right"),
  remap_method = c("adaptive", "barycentric"),
  area_original_fname,
  area_target_fname,
  ROIcortex_original_fname,
  ROIcortex_target_fname,
  sphere_original_fname,
  sphere_target_fname
)
```

Arguments

original_fname	The GIFTI file to remap.
target_fname	Where to save the remapped file.
hemisphere	"left" (default) or "right". An error will occur if the hemisphere indicated in the GIFTI metadata does not match.
remap_method	"adaptive" (default) or "adaptive" resampling. These options correspond to the Workbench command options "BARYCENTRIC" and "ADAP_BARY_AREA", respectively. For remapping between fs_LR group data and FreeSurfer fsaverage group data, adaptive resampling should be used.
area_original_fname, area_target_fname	File paths to the surfaces to use for vertex area correction during adaptive resampling. Required if remap_method is "adaptive".

`ROIcortex_original_fname, ROIcortex_target_fname`
: `ROIcortex_original_fname` is the name of the ROI file corresponding to `original_fname`. Leave as `NULL` (default) if not applicable. If provided, then also provide `ROIcortex_target_fname` to say where to write the remapped ROI file.

`sphere_original_fname, sphere_target_fname`
: File paths to the sphere surfaces in the original and target spaces.

Value

The remapped GIFTI file name, invisibly

Connectome Workbench

This function interfaces with the `"-metric-resample"`, `"-label-resample"`, and/or `"-surface-resample"` Workbench commands, depending on the input.

See Also

Other gifting: `resample_gifti()`, `smooth_gifti()`

---

remove_xifti *Remove a component from a "xifti"*

Description

Remove a brain structure, surface, or subcortical structure from a `"xifti"`.

Usage

```
remove_xifti(xifti, remove = NULL, remove_sub = NULL)
```

Arguments

`xifti`	A `"xifti"` object.
`remove`	A character vector containing one or more of the following: `"cortex_left"`, `"cortex_right"`, `"subcortical"`, `"surf_left"`, and `"surf_right"`. Each of these components will be removed from `xifti`.
`remove_sub`	A vector containing subcortical structures to be removed from `xifti`. Can be specified with names, or with numeric factor values: see `substructure_table`.

Value

The new `"xifti"` with the requested component(s) removed

See Also

Other manipulating xifti: `add_surf()`, `apply_parc()`, `apply_xifti()`, `combine_xifti()`, `convert_to_dlabel()`, `merge_xifti()`, `move_to_mwall()`, `newdata_xifti()`, `remap_cifti()`, `resample_cifti()`, `resample_cifti_from_tem`
`scale_xifti()`, `select_xifti()`, `set_names_xifti()`, `smooth_cifti()`, `transform_xifti()`

resample_cifti *Resample CIFTI data*

Description

Performs spatial resampling of CIFTI data on the cortical surface by separating it into GIFTI and NIFTI files, resampling the GIFTIs, and then putting them together. (The subcortex is not resampled.)

Usage

```
resample_cifti(
  x = NULL,
  cifti_target_fname = NULL,
  surfL_original_fname = NULL,
  surfR_original_fname = NULL,
  surfL_target_fname = NULL,
  surfR_target_fname = NULL,
  resamp_res,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)

resampleCIfTI(
  x = NULL,
  cifti_target_fname = NULL,
  surfL_original_fname = NULL,
  surfR_original_fname = NULL,
  surfL_target_fname = NULL,
  surfR_target_fname = NULL,
  resamp_res,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)

resamplecii(
  x = NULL,
  cifti_target_fname = NULL,
  surfL_original_fname = NULL,
```

```
  surfR_original_fname = NULL,
  surfL_target_fname = NULL,
  surfR_target_fname = NULL,
  resamp_res,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)

resample_xifti(
  x = NULL,
  cifti_target_fname = NULL,
  surfL_original_fname = NULL,
  surfR_original_fname = NULL,
  surfL_target_fname = NULL,
  surfR_target_fname = NULL,
  resamp_res,
  resamp_method = c("barycentric", "adaptive"),
  areaL_original_fname = NULL,
  areaR_original_fname = NULL,
  write_dir = NULL,
  mwall_values = c(NA, NaN),
  verbose = TRUE
)
```

Arguments

x	The CIFTI file name or `"xifti"` object to resample. If `NULL`, the result will be a `"xifti"` with resampled surfaces given by `surfL_original_fname` and `surfR_original_fname`.
cifti_target_fname	File name for the resampled CIFTI. Will be placed in `write_dir`. If `NULL`, will be written to "resampled.d*.nii". `write_dir` will be appended to the beginning of the path.
surfL_original_fname, surfR_original_fname	(Optional) Path to a GIFTI surface geometry file representing the left/right cortex. One or both can be provided. These will be resampled too, and are convenient for visualizing the resampled data. If `x` is a `"xifti"` object with surfaces, these arguments will override the surfaces in the `"xifti"`.
surfL_target_fname, surfR_target_fname	(Optional) File names for the resampled GIFTI surface geometry files. Will be placed in `write_dir`. If `NULL` (default), will use default names created by `resample_cifti_default_fname`.

`resamp_res` Target resolution for resampling (number of cortical surface vertices per hemisphere).

`resamp_method` `"barycentric"` (default) or `"adaptive"` resampling for the metric or label data. These options correspond to the Workbench command options `"BARYCENTRIC"` and `"ADAP_BARY_AREA"`, respectively.

While adaptive resampling is recommended for metric or label data, it requires that `area[L/R]_original_fname` be provided.

Note that surfaces will resampled using barycentric resampling regardless of `resamp_method`, because barycentric resampling rather than adaptive resampling is recommended for surface data.

`areaL_original_fname, areaR_original_fname`
File paths to the surfaces to use for vertex area correction during adaptive resampling. (Only used if resampling with the adaptive method.) `area[L/R]_original_fname` should match the current resolution of the data.

For resampling: the Workbench command for adaptive resampling requires the target surfaces for area correction too. But to make the workflow easier, `ciftiTools` will resample `area[L/R]_original_fname` with the barycentric method and use that for the target area.

For remapping: `area[L/R]_target_fname` must be directly provided.

`write_dir` Where to write the resampled CIFTI (and surfaces if present.) If `NULL` (default), will use the current working directory if `x` was a CIFTI file, and a temporary directory if `x` was a `"xifti"` object.

`mwall_values` If the medial wall locations are not indicated in the CIFTI, use these values to infer the medial wall mask. Default: `c(NA, NaN)`. If `NULL`, do not attempt to infer the medial wall.

Correctly indicating the medial wall locations is important for resampling, because the medial wall mask is taken into account during resampling calculations.

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

Details

Can accept a `"xifti"` object as well as a path to a CIFTI-file.

If surface data is included, it will be resampled with the barycentric method even if `resamp_method=="adaptive"` because the barycentric method is recommended for surface geometry data.

Value

A named character vector of written files: `"cifti"` and potentially `"surfL"` (if `surfL_original_fname` was provided) and/or `"surfR"` (if `surfR_original_fname` was provided).

Connectome Workbench

This function interfaces with the `"-metric-resample"`, `"-label-resample"`, and/or `"-surface-resample"` Workbench commands, depending on the input.

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti_from_templ scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

Other common: is.cifti(), read_cifti(), smooth_cifti(), view_xifti(), write_cifti()

`resample_cifti_from_template`

Resample a CIFTI from a template

Description

Resample a CIFTI from a template, ensuring the new CIFTI's resolution matches that of the template.

Usage

```
resample_cifti_from_template(original_fname, template_fname, target_fname)
```

Arguments

`original_fname`	A CIFTI file to resample.
`template_fname`	A CIFTI file to use as the template.
`target_fname`	The file name to save the resampled CIFTI.

Value

The `target_fname`, invisibly

Connectome Workbench

This function interfaces with the `"-cifti-resample"` Workbench command.

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

resample_gifti *Resample a GIFTI file (with its ROI)*

Description

Perform spatial resampling of GIFTI data on the cortical surface (metric and label), or of GIFTI surface geometry data itself.

Usage

```
resample_gifti(
  original_fname,
  target_fname,
  hemisphere = c("left", "right"),
  file_type = NULL,
  original_res = NULL,
  resamp_res = NULL,
  resamp_method = c("barycentric", "adaptive"),
  area_original_fname = NULL,
  area_target_fname = NULL,
  ROIcortex_original_fname = NULL,
  ROIcortex_target_fname = NULL,
  sphere_original_fname = NULL,
  sphere_target_fname = NULL,
  read_dir = NULL,
  write_dir = NULL
)

resampleGIfTI(
  original_fname,
  target_fname,
  hemisphere,
  file_type = NULL,
  original_res = NULL,
  resamp_res,
  ROIcortex_original_fname = NULL,
  ROIcortex_target_fname = NULL,
  read_dir = NULL,
  write_dir = NULL
)

resamplegii(
  original_fname,
  target_fname,
  hemisphere,
  file_type = NULL,
  original_res = NULL,
```

```
  resamp_res,
  ROIcortex_original_fname = NULL,
  ROIcortex_target_fname = NULL,
  read_dir = NULL,
  write_dir = NULL
)
```

Arguments

original_fname	The GIFTI file to resample.
target_fname	Where to save the resampled file.
hemisphere	"left" (default) or "right". An error will occur if the hemisphere indicated in the GIFTI metadata does not match.
file_type	"metric", "label", "surf", or NULL (default) to infer from original_fname.
original_res	The resolution of the original file. If NULL (default), infer from the file. Alternatively, provide sphere_original_fname which will override original_res. In general, original_res should be used when the original file is in registration with the spheres created by the Workbench command -surface-create-sphere, and sphere_original_fname should be used when it is not compatible.
resamp_res	Target resolution for resampling (number of cortical surface vertices per hemisphere). Alternatively, provide sphere_target_fname which will override resamp_res. In general, resamp_res should be used when the target file will be in registration with the spheres created by the Workbench command -surface-create-sphere, and sphere_target_fname should be used when it is not compatible.
resamp_method	"barycentric" (default) or "adaptive" resampling. These options correspond to the Workbench command options "BARYCENTRIC" and "ADAP_BARY_AREA", respectively. While adaptive resampling is recommended for metric or label data, it requires that area_original_fname be provided.
area_original_fname, area_target_fname	File paths to the surfaces to use for vertex area correction during adaptive resampling. (Ignored if resampling with the barycentric method.) area_original_fname should match the current resolution of the data, and area_target_fname should match resamp_res. If area_target_fname is not provided, area_original_fname will be resampled with the barycentric method, and the result will be used as area_target_fname.
ROIcortex_original_fname	The name of the ROI file corresponding to original_fname. Leave as NULL (default) if this doesn't exist or shouldn't be resampled.
ROIcortex_target_fname	The name of the resampled ROI file. Only applicable if ROIcortex_original_fname is provided.
sphere_original_fname, sphere_target_fname	File paths to the sphere surfaces in the original and target resolutions. If possible, the simpler arguments original_res and resamp_res can be used instead. But those depend on the surface being compatible with that created by

	`-surface-create-sphere`, which isn't always true. Therefore `sphere_original_fname` and `sphere_target_fname` can be used if needed.
`read_dir`	Directory to append to the path of every file name in `original_fname` and `ROIcortex_original_fname`. If `NULL` (default), do not append any directory to the path.
`write_dir`	Directory to append to the path of every file name in `target_fname` and `ROIcortex_target_fname`. If `NULL` (default), do not append any directory to the path.

Value

The resampled GIFTI file name, invisibly

Connectome Workbench

This function interfaces with the `"-metric-resample"`, `"-label-resample"`, and/or `"-surface-resample"` Workbench commands, depending on the input.

See Also

Other gifting: `remap_gifti()`, `smooth_gifti()`

`resample_surf` *Resample a `"surf"` object*

Description

Resample a `"surf"` object by writing it to a GIFTI, using the Connectome Workbench to resample it, and then reading the new file. The barycentric resampling method, which is recommended for anatomical surfaces, will be used.

Usage

```
resample_surf(surf, resamp_res, hemisphere = c("left", "right"))
```

Arguments

`surf`	A `"surf"` object
`resamp_res`	The desired resolution
`hemisphere`	`"left"` or `"right"`. Only used if not indicated by `surf$hemisphere`. An error will be raised if it does not match the hemisphere indicated in the intermediate written GIFTI.

Value

The new `"surf"`

Connectome Workbench

This function interfaces with the "-surface-resample" Workbench command.

See Also

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), mask_surf(), read_surf(), rotate_surf(), surf_area(), view_surf(), write_surf_gifti()

---

rotate_surf *Rotate a "surf" object*

Description

Rotate a "surf". Can be used to adjust the mesh orientation prior to view_xifti_surface.

Usage

```
rotate_surf(surf, r1 = 0, r2 = 0, r3 = 0, units = c("radians", "degrees"))
```

Arguments

surf	The "surf" object: see is.surf.
r1, r2, r3	Angle to rotate along the first, second, and third column's axis, in units (e.g. changing r1 will change the vertex positions in the second and third dimensions/columns, since the mesh is being rotated with respect to the first column's axis). Default: 0. With view_xifti_surface and other mesh rendering functions that use rgl, these rotations seem to correspond to yaw, pitch, and roll, respectively.
units	"radians" (default) or "degrees".

Value

The rotated "surf"

See Also

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), mask_surf(), read_surf(), resample_surf(), surf_area(), view_surf(), write_surf_gifti()

ROY_BIG_BL *"ROY_BIG_BL" color palette*

Description

"ROY_BIG_BL", the default palette from the Connectome Workbench.

Usage

```
ROY_BIG_BL(min = 0, max = 1, mid = NULL, half = NULL, pos_half = FALSE)
```

Arguments

min	The minimum value for the color mapping. As in the original palette, the last color (aqua) is actually placed at the bottom .5\ the minimum and maximum. Default: 0
max	The maximum value for the color mapping. If this value is lower than the minimum, the color mapping will be reversed. If this is equal to the minimum, a palette with only the color black will be returned. Default: 1.
mid	(Optional) The midpoint value for the color mapping. If NULL (default), the true midpoint is used.
half	"positive" or "negative" to use the positive half (black –> red –> yellow) or negative half (black –> blue –> purple –> green –> aqua) only. NULL (default) or FALSE to use entire palette.
pos_half	Deprecated. Use half.

Details

Yields the landmark color hex codes and values for the "ROY_BIG_BL" palette. This is the same color palette as the default Connectome Workbench palette. Source: https://github.com/Washington-University/workbench/blob/master/src/Files/PaletteFile.cxx

Value

A data.frame with two columns: "color" (character: color hex codes) and "value" (numeric)

See Also

Other coloring: expand_color_pal(), make_color_pal(), use_color_pal()

run_wb_cmd *Wrapper for Connectome Workbench Commands*

Description

Runs a Connectome Workbench command that has already been formatted.

Usage

```
run_wb_cmd(cmd, intern = TRUE, ignore.stdout = NULL, ignore.stderr = NULL)
```

Arguments

cmd	The full command, beginning after the workbench path.
intern	Return printed output? If FALSE, return logical indicating success instead. Default: TRUE.
ignore.stdout, ignore.stderr	The "ignore.stdout" and "ignore.stderr" arguments to system. Should be logical or NULL. If NULL (default), messages will be controlled by ciftiTools.getOption("suppress_msgs") and errors will not be ignored.

Value

If intern==TRUE, the printed output of the command. If intern==FALSE, a logical indicating if the command finished successfully.

S3_Math *"xifti" S3 Math methods*

Description

Math methods for "xifti" objects.

Usage

```
## S3 method for class 'xifti'
Math(x, ...)
```

Arguments

x	The "xifti"
...	Additional arguments to the function

Details

Uses transform_xifti.

S3_Ops "xifti" *S3 Ops methods*

Description

Ops methods for "xifti" objects.

Usage

```
## S3 method for class 'xifti'
Ops(e1, e2 = NULL)
```

Arguments

e1, e2	The arguments to the operation. "xifti" objects will be converted to matrices temporarily

Details

Uses transform_xifti.

S3_Summary "xifti" *S3 Summary methods*

Description

Summary methods for "xifti" objects.

Usage

```
## S3 method for class 'xifti'
Summary(..., na.rm = FALSE)
```

Arguments

...	The "xifti" and additional numeric arguments will be converted to matrices
na.rm	Remove NA values? Default: FALSE.

scale_xifti *Scale CIFTI*

Description

Scale CIFTI data. Similar to `scale`.

Usage

```
scale_xifti(xifti, center = TRUE, scale = TRUE)
```

Arguments

`xifti`	A `"xifti"` object.
`center, scale`	Arguments to `scale`.

Value

The input `"xifti"` with scaled columns.

See Also

Other manipulating xifti: `add_surf()`, `apply_parc()`, `apply_xifti()`, `combine_xifti()`, `convert_to_dlabel()`, `merge_xifti()`, `move_to_mwall()`, `newdata_xifti()`, `remap_cifti()`, `remove_xifti()`, `resample_cifti()`, `resample_cifti_from_template()`, `select_xifti()`, `set_names_xifti()`, `smooth_cifti()`, `transform_xifti()`

select_xifti *Select columns of a "xifti"*

Description

Select column indices to keep in a `"xifti"`. Can also be used to reorder the columns.

Usage

```
select_xifti(xifti, idx, add_meta = "select")
```

Arguments

`xifti`	A `"xifti"` object.
`idx`	The column indices to keep, in order.
`add_meta`	Add `idx` to `xifti$meta$cifti$misc[[add_meta]]` for reference. Default: `"select"`. If `NULL` or an empty string, do not add a metadata entry.

Value

The "xifti" with only the selected columns.

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), set_names_xifti(), smooth_cifti(), transform_xifti()

---

separate_cifti *Separate a CIFTI file*

Description

Separate a CIFTI file into GIFTI files for the cortical data and NIFTI files for the subcortical data and labels. ROIs can also be written to indicate the medial wall mask (cortex) and volume mask (subcortex). This uses the Connectome Workbench command -cifti-separate.

Usage

```
separate_cifti(
  cifti_fname,
  brainstructures = NULL,
  cortexL_fname = NULL,
  cortexR_fname = NULL,
  subcortVol_fname = NULL,
  subcortLabs_fname = NULL,
  ROI_brainstructures = "all",
  ROIcortexL_fname = NULL,
  ROIcortexR_fname = NULL,
  ROIsubcortVol_fname = NULL,
  write_dir = NULL
)

separateCIfTI(
  cifti_fname,
  brainstructures = c("left", "right"),
  cortexL_fname = NULL,
  cortexR_fname = NULL,
  subcortVol_fname = NULL,
  subcortLabs_fname = NULL,
  ROI_brainstructures = "all",
  ROIcortexL_fname = NULL,
  ROIcortexR_fname = NULL,
  ROIsubcortVol_fname = NULL,
  write_dir = NULL
)
```

```
separatecii(
  cifti_fname,
  brainstructures = c("left", "right"),
  cortexL_fname = NULL,
  cortexR_fname = NULL,
  subcortVol_fname = NULL,
  subcortLabs_fname = NULL,
  ROI_brainstructures = "all",
  ROIcortexL_fname = NULL,
  ROIcortexR_fname = NULL,
  ROIsubcortVol_fname = NULL,
  write_dir = NULL
)
```

Arguments

`cifti_fname` File path to a CIFTI file (ending in ".d*.nii").

`brainstructures`
(Optional) character vector indicating a subset of brain structure(s) to write: `"left"` cortex, `"right"` cortex, and/or `"subcortical"` structures. Can also be `"all"` to write out all existing brain structures. Default: `c("left","right")`.

`cortexL_fname, cortexR_fname`
(Optional) GIFTI file names (*.[func/label].gii) to save the [left/right] cortex data to. dtseries and dscalar files should use "func", whereas dlabel files should use "label".

If `NULL` and `write_dir` is provided, defaults to `"*[L/R].\[func/label\].gii"`, where * is the file name component of `cifti_fname`.

`subcortVol_fname, subcortLabs_fname`
(Optional) NIFTI file names to save the subcortical [volume/labels] to. Provide both or neither.

If `NULL` and `write_dir` is provided, defaults to `"*[/.labels].nii"`, where * is the file name component of `cifti_fname`.

`ROI_brainstructures`
Which ROIs should be obtained? `"all"` (default) to obtain ROIs for each of the `brainstructures`. `NULL` to not obtain any ROIs. This should be a subset of `brainstructures`.

`ROIcortexL_fname, ROIcortexR_fname`
(Optional) GIFTI file names (*.[func/label].gii) to save the [left/right] cortex ROI to. dtseries and dscalar files should use "func", whereas dlabel files should use "label".

If `NULL` and `write_dir` is provided, defaults to `"*ROI_[L/R].\[func/label\].gii"`, where * is the file name component of `cifti_fname`.

The cortical ROIs typically represent the medial wall mask, with values of 1 for in-ROI (non-medial wall) vertices and 0 for out-of-ROI (medial wall) vertices. Will be written in `write_dir`.

ROIsubcortVol_fname
: (Optional) NIFTI file names to save the subcortical ROI to.

  If NULL and write_dir is provided, defaults to "*ROI.nii", where * is the file name component of cifti_fname.

  The subcortical ROI typically represents the volumetric mask for the entire subcortical structure, with values of 1 for in-ROI (in subcortex) voxels and 0 for out-of-ROI (not in subcortex) voxels. Will be written in write_dir.

write_dir
: (Optional) A path to an existing directory. If provided, every component in the "xifti" will be written to this directory, using automatically-generated names if their *_fname argument was not provided. Otherwise if write_dir is NULL, only the components for which their *_fname was provided will be written.

Details

Time unit, start, and step (dtseries files) will not be written to the GIFTI/NIFTIs. Column names (dscalar files) will not be written to the GIFTIs, as well as label names and colors (dlabel files). (Haven't checked the NIFTIs yet.)

ROI/medial wall behavior: If there are 32k vertices in the left cortex with 3k representing the medial wall, then both cortexL_fname and ROIcortexL_fname will have 32k entries, 3k of which having a value of 0 indicating the medial wall. The non-medial wall entries will have the data values in cortexL_fname and a value of 1 in ROIcortexL_fname. Thus, exporting ROIcortexL_fname is vital if the data values include 0, because 0-valued non-medial wall vertices and medial wall vertices cannot be distinguished from one another within cortexL_fname alone.

Value

A named character vector with the file paths to the written NIFTI and GIFTI files

Connectome Workbench

This function interfaces with the "-cifti-separate" Workbench command.

See Also

Other writing: write_cifti(), write_metric_gifti(), write_subcort_nifti(), write_surf_gifti(), write_xifti2()

---

set_names_xifti *Set "xifti" column names*

Description

Change the column names of a "dscalar" or "dlabel" "xifti" object.

Usage

```
set_names_xifti(xifti, names)
```

Arguments

xifti	A "dscalar" or "dlabel" "xifti" object.
names	The new column names, as a character vector with length equal to the same number of columns in xifti.

Value

xifti with the new column names.

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), smooth_cifti(), transform_xifti()

---

smooth_cifti *Smooth CIFTI data*

Description

Spatially smooth the metric data of a CIFTI file or "xifti" object.

Usage

```
smooth_cifti(
  x,
  cifti_target_fname = NULL,
  surf_FWHM = 5,
  vol_FWHM = 3,
  surfL_fname = NULL,
  surfR_fname = NULL,
  cerebellum_fname = NULL,
  subcortical_zeroes_as_NA = FALSE,
  cortical_zeroes_as_NA = FALSE,
  subcortical_merged = FALSE
)

smoothCIfTI(
  x,
  cifti_target_fname = NULL,
  surf_FWHM = 5,
  vol_FWHM = 5,
  surfL_fname = NULL,
  surfR_fname = NULL,
  cerebellum_fname = NULL,
  subcortical_zeroes_as_NA = FALSE,
  cortical_zeroes_as_NA = FALSE,
```

```
  subcortical_merged = FALSE
)

smoothcii(
  x,
  cifti_target_fname = NULL,
  surf_FWHM = 5,
  vol_FWHM = 5,
  surfL_fname = NULL,
  surfR_fname = NULL,
  cerebellum_fname = NULL,
  subcortical_zeroes_as_NA = FALSE,
  cortical_zeroes_as_NA = FALSE,
  subcortical_merged = FALSE
)

smooth_xifti(
  x,
  cifti_target_fname = NULL,
  surf_FWHM = 5,
  vol_FWHM = 5,
  surfL_fname = NULL,
  surfR_fname = NULL,
  cerebellum_fname = NULL,
  subcortical_zeroes_as_NA = FALSE,
  cortical_zeroes_as_NA = FALSE,
  subcortical_merged = FALSE
)
```

Arguments

`x`	The CIFTI file name or `"xifti"` object to smooth.
`cifti_target_fname`	File name for the smoothed CIFTI. If `NULL`, will be written to "smoothed.d*.nii" in the current working directory if `x` was a CIFTI file, and in a temporary directory if `x` was a `"xifti"` object.
`surf_FWHM, vol_FWHM`	The full width at half maximum (FWHM) parameter for the gaussian surface or volume smoothing kernel, in mm. Default: `5` for cortex (surface) and `3` for subcortex (volume).
`surfL_fname, surfR_fname`	(Required if the corresponding cortex is present) Surface GIFTI files for the left and right cortical surfaces. If not provided, the surfaces in `x` are used, but if those are also not present, the default surfaces will be used.
`cerebellum_fname`	(Optional) Surface GIFTI file for the cerebellar surface

`subcortical_zeroes_as_NA`, `cortical_zeroes_as_NA`
: Should zero-values in the subcortical volume or cortex be treated as NA? Default: `FALSE`.

`subcortical_merged`
: Smooth across subcortical structure boundaries? Default: `FALSE`.

Details

If the CIFTI is a ".dlabel" file (intent 3007), then it will be converted to a ".dscalar" file because the values will no longer be integer indices. Unless the label values were ordinal, this is probably not desired so a warning will be printed.

Can accept a `"xifti"` object as well as a path to a CIFTI-file.

Surfaces are required for each hemisphere in the CIFTI. If they are not provided, the default inflated surfaces will be used.

Conversion for sigma: $\sigma \times 2 \times \sqrt{(2 \times \log(2))} = FWHM$

Value

The `cifti_target_fname`, invisibly, if `x` was a CIFTI file name. A `"xifti"` object if `x` was a `"xifti"` object.

Connectome Workbench

This function interfaces with the `"-cifti-smoothing"` Workbench command.

See Also

Other manipulating xifti: `add_surf()`, `apply_parc()`, `apply_xifti()`, `combine_xifti()`, `convert_to_dlabel()`, `merge_xifti()`, `move_to_mwall()`, `newdata_xifti()`, `remap_cifti()`, `remove_xifti()`, `resample_cifti()`, `resample_cifti_from_template()`, `scale_xifti()`, `select_xifti()`, `set_names_xifti()`, `transform_xifti()`

Other common: `is.cifti()`, `read_cifti()`, `resample_cifti()`, `view_xifti()`, `write_cifti()`

`smooth_gifti` *Smooth a metric GIFTI file*

Description

Smooths metric GIFTI data along the cortical surface. The results are written to a new GIFTI file.

Usage

```
smooth_gifti(
  original_fname,
  target_fname,
  surf_fname = NULL,
  surf_FWHM = 5,
  hemisphere = c("left", "right"),
```

```
  ROI_fname = NULL,
  zeroes_as_NA = FALSE
)

smoothGIfTI(
  original_fname,
  target_fname,
  surf_fname,
  surf_FWHM = 5,
  zeroes_as_NA = FALSE
)

smoothgii(
  original_fname,
  target_fname,
  surf_fname,
  surf_FWHM = 5,
  zeroes_as_NA = FALSE
)
```

Arguments

original_fname	The GIFTI file to smooth.
target_fname	Where to save the smoothed file.
surf_fname	Surface GIFTI files cortical surface along which to smooth. If not provided, the default inflated surfaces will be used.
surf_FWHM	The full width at half maximum (FWHM) parameter for the gaussian surface smoothing kernel, in mm. Default: 5
hemisphere	The cortex hemisphere: "left" or "right". Only used if surf_fname is NULL.
ROI_fname	The ROI to limit smoothing to, as a metric file. This is used to exclude the medial wall from smoothing. If not provided (default) all the data is smoothed across the surface.
zeroes_as_NA	Should zero-values be treated as NA? Default: FALSE.

Value

The smoothed GIFTI file name, invisibly

Connectome Workbench

This function interfaces with the "-metric-smoothing" Workbench command.

See Also

Other gifting: remap_gifti(), resample_gifti()

substructure_table *Substructure table*

Description

Table of labels for cortex hemispheres (left and right) and subcortical substructures. The names used by the CIFTI format and the names used by `ciftiTools` are given.

Usage

```
substructure_table()
```

Details

The names used by `ciftiTools` are based on those in `FT_READ_CIFTI` from the FieldTrip MATLAB toolbox.

Value

A data.frame with each substructure along the rows. The first column gives the CIFTI format name and the second column gives the `ciftiTools` name.

summary.surf *Summarize a "surf" object*

Description

Summary method for class "surf"

Usage

```
## S3 method for class 'surf'
summary(object, ...)

## S3 method for class 'summary.surf'
print(x, ...)

## S3 method for class 'surf'
print(x, ...)
```

Arguments

`object`	Object of class "surf". See `is.surf` and `make_surf`.
`...`	further arguments passed to or from other methods.
`x`	Object of class "surf".

summary.xifti *Summarize a "xifti" object*

Description

Summary method for class "xifti"

Usage

```
## S3 method for class 'xifti'
summary(object, ...)

## S3 method for class 'summary.xifti'
print(x, ...)

## S3 method for class 'xifti'
print(x, ...)
```

Arguments

object	Object of class "xifti".
...	further arguments passed to or from other methods.
x	A "xifti" object.

supported_intents *The NIFTI intents supported by* ciftiTools

Description

Table of CIFTI file types (NIFTI intents) supported by ciftiTools.

Usage

```
supported_intents()
```

Details

See https://www.nitrc.org/forum/attachment.php?attachid=334&group_id=454&forum_id=1955 for information about the different NIFTI intents.

Value

A data.frame with each supported file type along the rows, and column names "extension", "intent_code", "value", and "intent_name"

surf_area *Surface area calculation*

Description

Calculate surface area of a "surf" object by vertex or face. Surface area calculation by vertex matches the Workbench command "-surface-vertex-areas".

Usage

```
surf_area(surf, by = c("vertex", "face"))
```

Arguments

surf	The "surf" object.
by	"vertex" or "face". For "vertex", the result is the area associated with each vertex: the sum the area of each triangular face it is a part of, divided by three. For "face", the result is the surface area of each face.

Value

Vector of surface areas by vertex or face, in the same order as how the vertices or faces are listed in surf. The units are the square of the units of surf$vertices.

See Also

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), mask_surf(), read_surf(), resample_surf(), rotate_surf(), view_surf(), write_surf_gifti()

transform_xifti *Apply a univariate transformation to a "xifti" or pair of "xifti"s.*

Description

Apply a univariate transformation to each value in a "xifti" or pair of "xifti"s. If a pair, they must share the same dimensions (brainstructures) and number of measurements.

Usage

```
transform_xifti(xifti, FUN, xifti2 = NULL, idx = NULL, ...)
```

Arguments

xifti	A "xifti" object.
FUN	The function. If xifti2 is not provided, it should be a univariate function like log or sqrt. If xifti2 is provided, it should take in two arguments, like `+` or pmax.
xifti2	The second xifti, if applicable. Otherwise, NULL (default)
idx	The column indices for which to apply the transformation. If NULL (default), apply to all columns. If two "xifti" objects, were provided, the values in the first (xifti) will be retained for columns that are not transformed.
...	Additional arguments to FUN

Details

If the "xifti" had the dlabel intent, and the transformation creates any value that is not a label value (e.g. a non-integer), then it is converted to a dscalar.

Technically, the function does not have to be univariate: it only has to return the same number of values as the input. The function will be applied to the matrix for each brain structure separately. For example, the function function(q){(q - mean(q)) / sd(q)} will scale each brainstructure, while scale will scale each column of each brainstructure.

Value

A "xifti" storing the result of applying FUN to the input(s). The data dimensions will be the same. The metadata of xifti will be retained, and the metadata of xifti2 will be discarded (if provided).

See Also

Other manipulating xifti: add_surf(), apply_parc(), apply_xifti(), combine_xifti(), convert_to_dlabel(), merge_xifti(), move_to_mwall(), newdata_xifti(), remap_cifti(), remove_xifti(), resample_cifti(), resample_cifti_from_template(), scale_xifti(), select_xifti(), set_names_xifti(), smooth_cifti()

---

unmask_subcortex *Undo the volumetric mask to the subcortex*

Description

Un-applies the mask to the subcortical data in a "xifti" to yield its volumetric representation.

Usage

```
unmask_subcortex(xifti, fill = NA)
```

Arguments

xifti	A "xifti" object.
fill	The value for locations outside the mask. Default: NA.

Value

The 3D or 4D unflattened volume array

---

use_color_pal *Use a color palette*

Description

Applies a palette to a data vector to yield a vector of colors.

Usage

```
use_color_pal(data_values, pal, color_NA = "white", indices = FALSE)
```

Arguments

data_values	The values to map to colors
pal	The palette to use to map values to colors
color_NA	The color to use for NA values. Default: "white".
indices	Return the numeric indices of colors in pal$value rather than the colors themselves. A value of 0 will be used for missing data. Default: FALSE.

Value

A character vector of color names (or integers if indices).

See Also

Other coloring: ROY_BIG_BL(), expand_color_pal(), make_color_pal()

---

vertices_Param *vertices*

Description

vertices

Arguments

vertices	A $V \times 3$ matrix, where each row contains the Euclidean coordinates at which a given vertex in the mesh is located. V is the number of vertices in the mesh

view_comp *View composite of images*

Description

Create a single image which displays multiple image files. Tailored to support composite layouts of plots from view_xifti.

Usage

```
view_comp(
  img,
  ncol = NULL,
  nrow = NULL,
  legend = NULL,
  title = NULL,
  legend_height = 0.3,
  title_height = 0.1,
  title_fsize = 5,
  newpage = is.null(fname),
  fname = NULL,
  ...
)
```

Arguments

img	Character vector of paths to images to include. They will be arranged by row.
ncol, nrow	Control the layout of the composite image. NULL (default) will use approximately same numbers of rows and columns.
legend	File path to a legend image to add, or NULL (default) to not add a legend.
title	A length-one character vector to use as the title, or NULL (default) to not add a title.
legend_height, title_height	Heights of the legend and title, if applicable. Specified relative to all the plots, so .1 would mean the height is a tenth of the height of all the plots. Default: .1 for the title and .3 for the legend.
title_fsize	Multiplier for font size. Default: 5
newpage	Call grid::grid.newpage before rendering? Default: is.null(fname).
fname	If NULL (default), print the result. Otherwise, save to a PNG file at this location. Will override newpage to FALSE.
...	Additional arguments to gridExtra::arrangeGrob. The arguments grobs and layout_matrix should be avoided because they are determined based on img. adjusting widths may be useful, e.g. to make the subcortex subplot be less wide than the cortex subplot.

Details

Requires the following packages: `png`, `grid`, `gridExtra`

How it works: the non-legend images (plots) are composited in a call to `grid::arrangeGrob`. If a title or legend exists, it's added to the top and bottom, respectively, of the plots after with another call to `grid::arangeGrob`.

Value

The composite plot

See Also

Other visualizing: `view_surf()`, `view_xifti()`, `view_xifti_surface()`, `view_xifti_volume()`

`view_surf` *View "surf" object(s)*

Description

Visualize one or two `"surf"` objects(s), or the `"surf"` component(s) in a `"xifti"` using an interactive Open GL window made with `rgl`. The `rgl` package is required.

Usage

```
view_surf(
  ...,
  view = c("both", "lateral", "medial"),
  widget = NULL,
  title = NULL,
  fname = FALSE,
  cex.title = NULL,
  text_color = "black",
  bg = NULL,
  alpha = 1,
  edge_color = NULL,
  vertex_color = NULL,
  vertex_size = 0,
  material = NULL,
  width = NULL,
  height = NULL,
  zoom = NULL
)
```

Arguments

...	One of: A "surf" object; two "surf" objects; or, a "xifti" object. If a "surf" object has an empty "hemisphere" metadata entry, it will be set to the opposite side of the other's if known; otherwise, it will be set to the left side. If both are unknown, the first will be taken as the left and the second as the right.
view	Which view to display: "lateral", "medial", or "both". If NULL (default), both views will be shown. Each view will be plotted in a separate panel row.
widget	Display the plot in an htmlwidget? Should be logical or NULL (default), in which case a widget will be used only if needed (length(idx)>1 & isFALSE(fname), fname is a file path to an .html file, or if rgl.useNULL()).
title	Optional title(s) for the plot(s). It will be printed at the top in a separate subplot with 1/4 the height of the brain cortex subplots. Default: NULL will not use any title if length(idx)==1. Otherwise, it will use the time index (".dtseries") or name (.dscalar or .dlabel) of each data column. To use a custom title(s), use a length 1 character vector (same title for each plot) or length length(idx) character vector (different title for each plot). Set to NULL or an empty character to omit the title. If the title is non-empty but does not appear, try lowering cex.title.
fname	Save the plot(s) (and color legend if applicable)? If isFALSE(fname) (default), no files will be written. If fname is a length-1 character vector ending in ".html", an html with an interactive widget will be written. If neither of the cases above apply, a png image will be written for each idx. If isTRUE(fname) the files will be named by the data column names (underscores will replace spaces). Or, set fname to a length 1 character vector to name files by this suffix followed by the fname_suffix. Or, set fname to a character vector with the same length as idx to name the files exactly.
cex.title	Font size multiplier for the title. NULL (default) will use 2 for titles less than 20 characters long, and smaller sizes for increasingly longer titles.
text_color	Color for text in title and colorbar legend. Default: "black".
bg	Background color. NULL will use "white". Does not affect the color legend or color bar if printed separately: those will always have white backgrounds.
alpha	Transparency value for mesh coloring, between 0 and 1. Default: 1.0 (no transparency).
edge_color	Outline each edge in this color. Default: NULL (do not outline the edges).
vertex_color	Draw each vertex in this color. Default: "black". Vertices are only drawn if vertex_size > 0
vertex_size	Draw each vertex with this size. Default: 0 (do not draw the vertices).
material	A list of materials from material3d to use. For example, list(lit=FALSE, smooth=FALSE) will use exact colors from the color scale, rather than adding geometric shading and interpolating vertex colors. If NULL, use defaults.
width, height	The dimensions of the RGL window, in pixels. If both are NULL (default), these dimensions depend on type of output (Open GL window or widget) and subplots

(hemisphere, view, title, and slider_title) and are chosen to be the largest plot within a 1500 x 700 area (Open GL window) or 600 x 700 area (widget) that maintains a brain hemisphere subplot dimensions ratio of 10 x 7. Specifying only one will set the other to maintain this aspect ratio. Both can be specified to set the dimensions exactly, but note that the dimensions cannot be larger than the screen resolution. (These arguments do not affect the size of the legend, which cannot be controlled.)

The plot will be taller than height to accommodate a title or color bar.

If multiple idx are being composited with together, these arguments refer to a single idx within the composited plot, and not the composited plot itself.

zoom — Adjust the sizes of the brain meshes. Default: NULL (will be set to 0.6 or 160\ widget.)

Details

This function works as a wrapper to view_xifti_surface, but some arguments are not applicable (e.g. color scheme and legend). Also, instead of using the hemisphere argument, name the surface arguments surfL or surfR (see description for parameter ...). Finally, the default value for param is "surf", not "xifti".

Navigating and Embedding the Interactive Plots

To navigate the interactive Open GL window and html widget, left click and drag the cursor to rotate the meshes. Use the scroll wheel or right click and drag to zoom. Press the scroll wheel and drag to change the field-of-view. For Open GL windows, execute snapshot to save the current window as a .png file, close3d to close the window, and view3d to programmatically control the perspective.

To embed an interactive plot in an R Markdown document, first execute rgl::setupKnitr() to prepare the document for embedding the widget. Then execute the plot command as you normally would to create a widget (i.e. without specifying fname, and by requesting more than one idx or by setting widget to TRUE). When the R Markdown document is knitted, the interactive widget should be displayed below the chunk in which the plot command was executed. See the vignette for an example.

Embedding the Static Plots

To embed a static plot in an R Markdown document, first execute rgl::setupKnitr() to prepare the document for embedding the snapshot of the Open GL window. Then execute the plot command as you normally would to create an Open GL window (i.e. without specifying fname, and by requesting only one idx). In the options for the chunk in which the plot command is executed, set rgl=TRUE, format="png". You can also control the image dimensions here e.g. fig.height=3.8, fig.width=5. When the R Markdown document is knitted, the static plots should be displayed below the chunk in which the plot command was executed. See the vignette for an example.

See Also

Other visualizing: view_comp(), view_xifti(), view_xifti_surface(), view_xifti_volume()

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), mask_surf(), read_surf(), resample_surf(), rotate_surf(), surf_area(), write_surf_gifti()

view_xifti *View a "xifti" object*

Description

Displays the data in a "xifti" object using view_xifti_surface and/or view_xifti_volume. Compared to calling these two functions separately on the same data, this function may be more convenient since the automatic choice of color mode and limits is determined across the entire data and shared between the two plots. Also, if writing files the subcortical plots will not overwrite the cortical plots.

Usage

```
view_xifti(xifti, what = NULL, ...)

view_cifti(xifti, ...)

viewCIfTI(xifti, ...)

viewcii(xifti, ...)
```

Arguments

xifti	A "xifti" object.
what	"surface", "volume", or "both". NULL will infer based on the contents of the "xifti": if there is data, plot the surface cortex data if present, and the volumetric subcortical data otherwise. If there is no data, plot the surface geometry if present, and do nothing otherwise.
...	Additional arguments to pass to either view function.

Value

The return value of view_xifti_surface or view_xifti_volume.

See Also

Other common: is.cifti(), read_cifti(), resample_cifti(), smooth_cifti(), write_cifti()

Other visualizing: view_comp(), view_surf(), view_xifti_surface(), view_xifti_volume()

`view_xifti_surface` *View cortical surface data in a* `"xifti"`

Description

Visualize `"xifti"` cortical data using an interactive Open GL window or htmlwidget made with `rgl`. The `rmarkdown` package is required for the htmlwidget functionality.

Usage

```
view_xifti_surface(
  xifti = NULL,
  surfL = NULL,
  surfR = NULL,
  color_mode = "auto",
  zlim = NULL,
  colors = NULL,
  idx = NULL,
  hemisphere = NULL,
  together = NULL,
  together_ncol = NULL,
  together_title = NULL,
  view = c("both", "lateral", "medial"),
  widget = NULL,
  title = NULL,
  slider_title = "Index",
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
  scientific = NA,
  cex.title = NULL,
  text_color = "black",
  bg = NULL,
  NA_color = "white",
  borders = FALSE,
  alpha = 1,
  edge_color = NULL,
  vertex_color = NULL,
  vertex_size = 0,
  material = NULL,
  shadows = 1,
  width = NULL,
  height = NULL,
```

```
  zoom = NULL
)

view_cifti_surface(
  xifti = NULL,
  surfL = NULL,
  surfR = NULL,
  color_mode = "auto",
  zlim = NULL,
  colors = NULL,
  idx = NULL,
  hemisphere = NULL,
  together = NULL,
  together_ncol = NULL,
  together_title = NULL,
  view = c("both", "lateral", "medial"),
  widget = NULL,
  title = NULL,
  slider_title = "Index",
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
  scientific = NA,
  cex.title = NULL,
  text_color = "black",
  bg = NULL,
  NA_color = "white",
  borders = FALSE,
  alpha = 1,
  edge_color = NULL,
  vertex_color = NULL,
  vertex_size = 0,
  material = NULL,
  shadows = 1,
  width = NULL,
  height = NULL,
  zoom = NULL
)

viewCIfTI_surface(
  xifti = NULL,
  surfL = NULL,
  surfR = NULL,
  color_mode = "auto",
```

```
  zlim = NULL,
  colors = NULL,
  idx = NULL,
  hemisphere = NULL,
  together = NULL,
  together_ncol = NULL,
  together_title = NULL,
  view = c("both", "lateral", "medial"),
  widget = NULL,
  title = NULL,
  slider_title = "Index",
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
  scientific = NA,
  cex.title = NULL,
  text_color = "black",
  bg = NULL,
  NA_color = "white",
  borders = FALSE,
  alpha = 1,
  edge_color = NULL,
  vertex_color = NULL,
  vertex_size = 0,
  material = NULL,
  shadows = 1,
  width = NULL,
  height = NULL,
  zoom = NULL
)

viewcii_surface(
  xifti = NULL,
  surfL = NULL,
  surfR = NULL,
  color_mode = "auto",
  zlim = NULL,
  colors = NULL,
  idx = NULL,
  hemisphere = NULL,
  together = NULL,
  together_ncol = NULL,
  together_title = NULL,
  view = c("both", "lateral", "medial"),
```

```
  widget = NULL,
  title = NULL,
  slider_title = "Index",
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
  scientific = NA,
  cex.title = NULL,
  text_color = "black",
  bg = NULL,
  NA_color = "white",
  borders = FALSE,
  alpha = 1,
  edge_color = NULL,
  vertex_color = NULL,
  vertex_size = 0,
  material = NULL,
  shadows = 1,
  width = NULL,
  height = NULL,
  zoom = NULL
)
```

Arguments

xifti	A "xifti" object.
surfL, surfR	(Optional) The brain surface model to use. Each can be a "surf" object, any valid argument to read_surf , or one of "very inflated", "inflated", or "midthickness". If provided, it will override xifti$surf$cortex_left or xifti$surf$cortex_right if it exists. Leave as NULL (default) to use xifti$surf$cortex_left or xifti$surf$cortex_right if it exists, or the default inflated surfaces if it does not exist.
color_mode	(Optional) "sequential", "qualitative", "diverging", or "auto" (default). Auto mode will use the qualitative color mode if the "xifti" object represents a .dlabel CIFTI (intent 3007). Otherwise, it will use the diverging mode if the data contains both positive and negative values, and the sequential mode if the data contains >90\ make_color_pal for more details.
zlim	(Optional) Controls the mapping of values to each color in colors. If the length is longer than one, using -Inf will set the value to the data minimum, and Inf will set the value to the data maximum. See make_color_pal description for more details.
colors	(Optional) "ROY_BIG_BL", vector of colors to use, the name of a ColorBrewer palette (see RColorBrewer::brewer.pal.info and colorbrewer2.org), the name of a viridisLite palette, or a data.frame with columns "color" and "value" (will

override zlim). If NULL (default), will use the positive half of "ROY_BIG_BL" (sequential), "Set2" (qualitative), or the full "ROY_BIG_BL" (diverging). An exception to these defaults is if the "xifti" object represents a .dlabel CIFTI (intent 3007), in which case the colors in the label table will be used. See make_color_pal for more details.

idx — The time/column index of the data to display. NULL (default) will display the first column.

If its length is greater than one, and isFALSE(fname), a widget must be used since a single OpenGL window cannot show multiple indexes. A slider will be added to the widget to control what time/column is being displayed.

hemisphere — Which brain cortex to display: "both" (default), "left", or "right". Each will be plotted in a separate panel column.

If a brain cortex is requested but no surface is available, a default inflated surface will be used.

This argument can also be NULL (default). In this case, the default inflated surface included with ciftiTools will be used for each cortex with data (i.e. if xifti$data$cortex_left and/or xifti$data$cortex_right exist).

Surfaces without data will be colored white.

together — Only applies if saving image files (!isFALSE(fname)). Use this argument to create and save a composite image which combines multiple plots. NULL (default) will not combine any plots. Otherwise, this argument should be a character vector with one or more of the following entries:

"leg" to combine the color legend with each "xifti" data plot. Overrides/ignores legend_embed.

"idx" to place all the plots for the different "idx" in a grid. If the data is not qualitative, a shared color bar will be added to the bottom of the composite. If the data is qualitative, a shared color legend will be added to the bottom only if "leg" is in together. For greater control see view_comp or grid::arrangeGrob.

together_ncol — If "idx" %in% together, this determines the number of columns to use in the array of subplots for different indices. By default, the number of columns and rows will be determined such that they are about equal.

together_title — If a composite image is made based on together, use this argument to add a grand title to the composite image. Should be a length-one character vector or NULL (default) to not add a grand title.

view — Which view to display: "lateral", "medial", or "both". If NULL (default), both views will be shown. Each view will be plotted in a separate panel row.

widget — Display the plot in an htmlwidget? Should be logical or NULL (default), in which case a widget will be used only if needed (length(idx)>1 & isFALSE(fname), fname is a file path to an .html file, or if rgl.useNULL()).

title — Optional title(s) for the plot(s). It will be printed at the top in a separate subplot with 1/4 the height of the brain cortex subplots.

Default: NULL will not use any title if length(idx)==1. Otherwise, it will use the time index (".dtseries") or name (.dscalar or .dlabel) of each data column.

To use a custom title(s), use a length 1 character vector (same title for each plot) or length `length(idx)` character vector (different title for each plot). Set to `NULL` or an empty character to omit the title.

If the title is non-empty but does not appear, try lowering `cex.title`.

`slider_title` Text at bottom of plot that will be added if a slider is used, to provide a title for it. Default: `"Index"`. If `NULL` or an empty character, no title will be added.

`fname` Save the plot(s) (and color legend if applicable)?

If `isFALSE(fname)` (default), no files will be written.

If `fname` is a length-1 character vector ending in ".html", an html with an interactive widget will be written.

If neither of the cases above apply, a png image will be written for each `idx`. If `isTRUE(fname)` the files will be named by the data column names (underscores will replace spaces). Or, set `fname` to a length 1 character vector to name files by this suffix followed by the `fname_suffix`. Or, set `fname` to a character vector with the same length as `idx` to name the files exactly.

`fname_suffix` Either the data column names (`"names"`) or the index value (`"idx"`).

`legend_fname` Save the color legend? Since the legend is the same for each `idx` only one legend is written even if `length(idx)>1`. This argument can be `NULL` to not save the legend, an exact file path, or a length-one character vector with "[fname]" in it, which will name the legend based on `fname\[1\]`. For example, if `fname\[1\]` is `"plots/my_cifti.png"` and `legend_fname` is `"\[fname\]_legend"` (default), then the legend plot will be saved to `"plots/my_cifti_legend.png"`.

`legend_ncol` Number of columns in color legend. If `NULL` (default), use 10 entries per row. Only applies if the color legend is used (qualitative data).

`legend_alllevels`

Show all label levels in the color legend? If `FALSE` (default), just show the levels present in the data being viewed. Only applies if the color legend is used (qualitative data).

`legend_embed` Should the colorbar be embedded in the plot? It will be positioned in the bottom-left corner, in a separate subplot with 1/4 the height of the brain cortex subplots. Default: `TRUE`. If `FALSE`, print/save it separately instead.

Only applies if the color bar is used (sequential or diverging data) or if `"leg" %in% together`. Otherwise the color legend (qualitative data) cannot be embedded at the moment.

`digits` The number of digits for the colorbar legend ticks. If `NULL` (default), let `format` decide.

`scientific` Use scientific notation? If `NA` (default), let `format` decide.

`cex.title` Font size multiplier for the title. `NULL` (default) will use 2 for titles less than 20 characters long, and smaller sizes for increasingly longer titles.

`text_color` Color for text in title and colorbar legend. Default: `"black"`.

`bg` Background color. `NULL` will use `"white"`. Does not affect the color legend or color bar if printed separately: those will always have white backgrounds.

`NA_color` The color for the medial wall and `NA` values. Default: `"white"`. Also used to color the entire surface for `view_surf`.

borders
: Only applicable if color_mode is "qualitative". Border vertices will be identified (those that share a face with at least one vertex of a different value) and colored over. If this argument is TRUE borders will be colored in black; provide the name of a different color to use that instead. If FALSE or NULL (default), do not draw borders.

alpha
: Transparency value for mesh coloring, between 0 and 1. Default: 1.0 (no transparency).

edge_color
: Outline each edge in this color. Default: NULL (do not outline the edges).

vertex_color
: Draw each vertex in this color. Default: "black". Vertices are only drawn if vertex_size > 0

vertex_size
: Draw each vertex with this size. Default: 0 (do not draw the vertices).

material
: A list of materials from material3d to use. For example, list(lit=FALSE, smooth=FALSE) will use exact colors from the color scale, rather than adding geometric shading and interpolating vertex colors. If NULL, use defaults.

shadows
: Number from 0 (maximum added lighting) to 1 (no added lighting) to control the darkness and extent of shadowing on the 3D surface. Default: 1. Shadows help render the shape of the surface, but can be distracting if interpretation of the data depends on small differences in brightness along the color scale.

width, height
: The dimensions of the RGL window, in pixels. If both are NULL (default), these dimensions depend on type of output (Open GL window or widget) and subplots (hemisphere, view, title, and slider_title) and are chosen to be the largest plot within a 1500 x 700 area (Open GL window) or 600 x 700 area (widget) that maintains a brain hemisphere subplot dimensions ratio of 10 x 7. Specifying only one will set the other to maintain this aspect ratio. Both can be specified to set the dimensions exactly, but note that the dimensions cannot be larger than the screen resolution. (These arguments do not affect the size of the legend, which cannot be controlled.)

  The plot will be taller than height to accommodate a title or color bar.

  If multiple idx are being composited with together, these arguments refer to a single idx within the composited plot, and not the composited plot itself.

zoom
: Adjust the sizes of the brain meshes. Default: NULL (will be set to 0.6 or 160\ widget.)

Value

If a png or html file(s) were written, the names of the files for each index (and color legend if applicable) will be returned. Otherwise, the widget itself is returned if a widget was used, and the rgl object IDs are returned if an Open GL window was used. The rgl object IDs are useful for further programmatic manipulation of the Open GL window.

Navigating and Embedding the Interactive Plots

To navigate the interactive Open GL window and html widget, left click and drag the cursor to rotate the meshes. Use the scroll wheel or right click and drag to zoom. Press the scroll wheel and drag to change the field-of-view. For Open GL windows, execute snapshot to save the current window as a .png file, close3d to close the window, and view3d to programmatically control the perspective.

To embed an interactive plot in an R Markdown document, first execute `rgl::setupKnitr()` to prepare the document for embedding the widget. Then execute the plot command as you normally would to create a widget (i.e. without specifying `fname`, and by requesting more than one `idx` or by setting `widget` to `TRUE`). When the R Markdown document is knitted, the interactive widget should be displayed below the chunk in which the plot command was executed. See the vignette for an example.

Embedding the Static Plots

To embed a static plot in an R Markdown document, first execute `rgl::setupKnitr()` to prepare the document for embedding the snapshot of the Open GL window. Then execute the plot command as you normally would to create an Open GL window (i.e. without specifying `fname`, and by requesting only one `idx`). In the options for the chunk in which the plot command is executed, set `rgl=TRUE, format="png"`. You can also control the image dimensions here e.g. `fig.height=3.8, fig.width=5`. When the R Markdown document is knitted, the static plots should be displayed below the chunk in which the plot command was executed. See the vignette for an example.

See Also

Other visualizing: `view_comp()`, `view_surf()`, `view_xifti()`, `view_xifti_volume()`

`view_xifti_volume` *View subcortical data in a* `"xifti"`

Description

Visualize the subcortical data in a `"xifti"` using a series of 2D slices (based on `overlay`) or an interactive widget (based on `papayar::papaya`). Note: `papayar` has been removed from CRAN so the widget is not available. If `papayar` returns to CRAN the widget will be made available again.

Usage

```
view_xifti_volume(
  xifti,
  structural_img = "MNI",
  color_mode = "auto",
  zlim = NULL,
  colors = NULL,
  structural_img_colors = gray(0:255/280),
  title = NULL,
  idx = NULL,
  plane = c("axial", "sagittal", "coronal"),
  convention = c("neurological", "radiological"),
  n_slices = 9,
  slices = NULL,
  together = NULL,
  together_ncol = NULL,
```

```
  together_title = NULL,
  widget = FALSE,
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  fname_sub = FALSE,
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
  scientific = NA,
  cex.title = NULL,
  ypos.title = 0,
  xpos.title = 0,
  orientation_labels = TRUE,
  crop = TRUE,
  text_color = "white",
  bg = NULL,
  width = NULL,
  height = NULL,
  ...
)

view_cifti_volume(
  xifti,
  structural_img = "MNI",
  color_mode = "auto",
  zlim = NULL,
  colors = NULL,
  structural_img_colors = gray(0:255/280),
  title = NULL,
  idx = NULL,
  plane = c("axial", "sagittal", "coronal"),
  convention = c("neurological", "radiological"),
  n_slices = 9,
  slices = NULL,
  together = NULL,
  together_ncol = NULL,
  together_title = NULL,
  widget = FALSE,
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  fname_sub = FALSE,
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
```

```
  scientific = NA,
  cex.title = NULL,
  ypos.title = 0,
  xpos.title = 0,
  orientation_labels = TRUE,
  crop = TRUE,
  text_color = "white",
  bg = NULL,
  width = NULL,
  height = NULL,
  ...
)

viewCIfTI_volume(
  xifti,
  structural_img = "MNI",
  color_mode = "auto",
  zlim = NULL,
  colors = NULL,
  structural_img_colors = gray(0:255/280),
  title = NULL,
  idx = NULL,
  plane = c("axial", "sagittal", "coronal"),
  convention = c("neurological", "radiological"),
  n_slices = 9,
  slices = NULL,
  together = NULL,
  together_ncol = NULL,
  together_title = NULL,
  widget = FALSE,
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  fname_sub = FALSE,
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
  scientific = NA,
  cex.title = NULL,
  ypos.title = 0,
  xpos.title = 0,
  orientation_labels = TRUE,
  crop = TRUE,
  text_color = "white",
  bg = NULL,
  width = NULL,
  height = NULL,
```

```
  ...
)

viewcii_volume(
  xifti,
  structural_img = "MNI",
  color_mode = "auto",
  zlim = NULL,
  colors = NULL,
  structural_img_colors = gray(0:255/280),
  title = NULL,
  idx = NULL,
  plane = c("axial", "sagittal", "coronal"),
  convention = c("neurological", "radiological"),
  n_slices = 9,
  slices = NULL,
  together = NULL,
  together_ncol = NULL,
  together_title = NULL,
  widget = FALSE,
  fname = FALSE,
  fname_suffix = c("names", "idx"),
  fname_sub = FALSE,
  legend_fname = "[fname]_legend",
  legend_ncol = NULL,
  legend_alllevels = FALSE,
  legend_embed = NULL,
  digits = NULL,
  scientific = NA,
  cex.title = NULL,
  ypos.title = 0,
  xpos.title = 0,
  orientation_labels = TRUE,
  crop = TRUE,
  text_color = "white",
  bg = NULL,
  width = NULL,
  height = NULL,
  ...
)
```

Arguments

xifti	A "xifti" object.
structural_img	The structural MRI image on which to overlay the subcortical plot. Can be a file name, "MNI" (default) to use the MNI T1-weighted template included in ciftiTools, or NULL to use a blank image.
color_mode	(Optional) "sequential", "qualitative", "diverging", or "auto" (default).

Auto mode will use the qualitative color mode if the "xifti" object represents a .dlabel CIFTI (intent 3007). Otherwise, it will use the diverging mode if the data contains both positive and negative values, and the sequential mode if the data contains >90\ make_color_pal for more details.

zlim (Optional) Controls the mapping of values to each color in colors. If the length is longer than one, using -Inf will set the value to the data minimum, and Inf will set the value to the data maximum. See make_color_pal description for more details.

colors (Optional) "ROY_BIG_BL", vector of colors to use, the name of a ColorBrewer palette (see RColorBrewer::brewer.pal.info and colorbrewer2.org), the name of a viridisLite palette, or a data.frame with columns "color" and "value" (will override zlim). If NULL (default), will use the positive half of "ROY_BIG_BL" (sequential), "Set2" (qualitative), or the full "ROY_BIG_BL" (diverging). An exception to these defaults is if the "xifti" object represents a .dlabel CIFTI (intent 3007), in which case the colors in the label table will be used. See make_color_pal for more details.

structural_img_colors
Colors to use for the background image. These will be assigned in order from lowest to highest value with equal spacing between the colors. (color_mode, zlim and colors have no bearing on the background image colors.) This argument is used as the col.x argument to oro.nifti::overlay directly. Default: gray(0:255/280). To use the oro.nifti::overlay default instead set this argument to gray(0:64/64).

title Optional title(s) for the plot(s). It will be printed at the top.

Default: NULL will not use any title if length(idx)==1. Otherwise, it will use the time index (".dtseries") or name (.dscalar or .dlabel) of each data column.

To use a custom title(s), use a length 1 character vector (same title for each plot) or length length(idx) character vector (different title for each plot). Set to NULL or an empty character to omit the title.

If the title is non-empty but does not appear, try lowering cex.title.

idx The time/column index of the data to display. NULL (default) will display the first column.

If widget, only one index at a time may be displayed.

If !widget and the length of idx is greater than one, a new plot will be created for each idx. These can be toggled between using the arrows at the top of the display window if working interactively in RStudio; or, these will be written to separate files if !isFALSE(fname).

plane The plane to display for the slices: "axial" (default), "sagittal" or "coronal". Ignored if widget.

convention "neurological" (default) or "radiological". Neurological convention will display the left side of the brain on the left side of axial and coronal images, and in the first few slices of a series of sagittal images. Radiological convention will display the right side of the brain on the left side of axial and coronal images, and in the first few slices of a series of sagittal images.

n_slices The number of slices to display. Default: 9. The slices will be selected in a way that visualizes as much of the subcortex as possible. Ignored if widget.

slices — Which slices to display. If provided, this argument will override n_slices. Should be a numeric vector with integer values between one and the number of slices in plane. Ignored if widget.

together — Only applies if saving image files (!isFALSE(fname)). Use this argument to create and save a composite image which combines multiple plots. NULL (default) will not combine any plots. Otherwise, this argument should be a character vector with one or more of the following entries:

"leg" to combine the color legend with each "xifti" data plot. Overrides/ignores legend_embed.

"idx" to place all the plots for the different "idx" in a grid. If the data is not qualitative, a shared color bar will be added to the bottom of the composite. If the data is qualitative, a shared color legend will be added to the bottom only if "leg" is in together. For greater control see view_comp or grid::arrangeGrob.

together_ncol — If "idx" %in% together, this determines the number of columns to use in the array of subplots for different indices. By default, the number of columns and rows will be determined such that they are about equal.

together_title — If a composite image is made based on together, use this argument to add a grand title to the composite image. Should be a length-one character vector or NULL (default) to not add a grand title.

widget — Create an interactive widget using papayar? Otherwise display static 2D slices. Default: FALSE.

Note that the widget can only display one idx at a time.

Note: papayar has been removed from CRAN so the widget is not available. If papayar returns to CRAN the widget will be made available again.

fname, fname_suffix

Save the plot(s) (and color legend if applicable)?

If isFALSE(fname) (default), no files will be written.

If widget, these arguments are ignored.

If neither of the cases above apply, a png image will be written for each idx. If isTRUE(fname) the files will be named by the data column names (underscores will replace spaces). Or, set fname to a length 1 character vector to name files by this suffix followed by the fname_suffix: either the data column names ("names") or the index value ("idx"). Or, set fname to a character vector with the same length as idx to name the files exactly.

fname_sub — Add "_sub" to the end of the names of the files being saved? Default: FALSE. This is useful if cortical plots of the same data are being saved too.

legend_fname — Save the color legend? Since the legend is the same for each idx only one legend is written even if length(idx)>1. This argument can be NULL to not save the legend, an exact file path, or a length-one character vector with "[fname]" in it, which will name the legend based on fname\[1\]. For example, if fname\[1\] is "plots/my_cifti.png" and legend_fname is "\[fname\]_legend" (default), then the legend plot will be saved to "plots/my_cifti_legend.png".

legend_ncol — Number of columns in color legend. If NULL (default), use 10 entries per row. Only applies if the color legend is used (qualitative data).

`legend_alllevels`
: Show all label levels in the color legend? If `FALSE` (default), just show the levels present in the data being viewed. Only applies if the color legend is used (qualitative data).

`legend_embed`
: Should the colorbar be embedded in the plot? It will be positioned at the bottom. Default: `TRUE`. If `FALSE`, print/save it separately instead.

  Only applies if the color bar is used (sequential or diverging data). The color legend (qualitative data) cannot be embedded at the moment.

`digits`
: The number of digits for the colorbar legend ticks. If `NULL` (default), let `format` decide.

`scientific`
: Use scientific notation? If `NA` (default), let `format` decide.

`cex.title`
: Font size multiplier for the title. `NULL` (default) will use `1.2` for titles less than 20 characters long, and smaller sizes for increasingly longer titles. If saving a PNG and PDF file, the default will also scale with `width` relative to the default value of `width`.

`ypos.title, xpos.title`
: The positioning of the title can be finicky, especially when using an R Markdown document interactively in which case it appears too high in the plot. Use these arguments to nudge the title up or down (`ypos.title`) or left or right (`xpos.title`).

`orientation_labels`
: Show orientation labels at the top left and top right of the plot? These will indicate the directions along the left-right axis for each slice image. Default: `TRUE`. Ignored if `widget`. The vertical positioning is controlled by `ypos.title`, and the font size is controlled by `cex.title`.

`crop`
: Crop the slice subplots to the subcortical structures, instead of showing the full anatomical image? Default: `TRUE`. Ignored if `widget`.

`text_color`
: Color for text in title and colorbar legend. Default: `"white"`. If `"white"`, will use black instead for the color

`bg`
: Background color. `NULL` will use `"black"`. Does not affect the color legend or color bar if printed separately: those will always have white backgrounds.

`width, height`
: The dimensions of the plot, in pixels. Only affects saved images (if `!isFALSE(fname)`). If `NULL`, file dimensions will be 400 x 600 pixels for PNGs and 4 x 6 in. for PDFs.

  Currently, there is no way to control the dimensions of the plot if working interactively in RStudio or creating a knitted R Markdown document. The default appears to be a wide aspect ratio.

`...`
: Additional arguments to pass to `papayar::papaya` or `oro.nifti::overlay`. Note that for `oro.nifti::overlay` the following additional arguments should not be provided since they are pre-determined inside this function or by the arguments listed above: `x`, `y`, `plane`, `col.y`, `col.x`, `zlim.y`, `oma`, `plot.type`, `bg`.

Details

Note that `color_mode`, `zlim`, and `colors` only affect the color scale of the data values whereas `structural_img_colors` only affects the color scale of the background image.

Currently, the color-related arguments only affect the 2D slice view. The color limits and palette must be edited using the widget controls once it's rendered.

Arguments concerning anatomical orientation assume that the subcortical data is stored in the following way: first dimension is normal to the sagittal plane, going left to right; second dimension is normal to the coronal plane, going from the front of the head (anterior) to the back (posterior); third dimension is normal to the axial plane, going from the top of the head (superior) to the neck (inferior).

For non-interactive plots, if `n_slices > 1` and `convention="neurological"`, axial slices are ordered from the neck (inferior) to the top of the head (superior), sagittal slices are ordered left to right, and coronal slices are ordered back (posterior) to front (anterior). If `convention="radiological"`, sagittal slices are instead ordered right to left.

Value

If a png or pdf file(s) were written, the names of the files for each index (and color legend if applicable) will be returned. Otherwise, `NULL` is invisibly returned.

See Also

Other visualizing: `view_comp()`, `view_surf()`, `view_xifti()`, `view_xifti_surface()`

`write_cifti` *Write a CIFTI file from a* `"xifti"` *object*

Description

Write out a `"xifti"` object as a CIFTI file and (optionally) GIFTI surface files.

Usage

```
write_cifti(
  xifti,
  cifti_fname,
  surfL_fname = NULL,
  surfR_fname = NULL,
  verbose = TRUE
)

writeCIfTI(
  xifti,
  cifti_fname,
  surfL_fname = NULL,
  surfR_fname = NULL,
  verbose = TRUE
)

writecii(
```

```
  xifti,
  cifti_fname,
  surfL_fname = NULL,
  surfR_fname = NULL,
  verbose = TRUE
)

write_xifti(
  xifti,
  cifti_fname,
  surfL_fname = NULL,
  surfR_fname = NULL,
  verbose = TRUE
)
```

Arguments

`xifti`	A `"xifti"` object.
`cifti_fname`	File path to a CIFTI file (ending in ".d*.nii").
`surfL_fname, surfR_fname`	If the [left/right] surface is present, it will be a written to a GIFTI file at this file path. If `NULL` (default), do not write out the surface.
`verbose`	Should occasional updates be printed? Default: `TRUE`.

Details

See `write_xifti2` to write a `"xifti"` object out as separate GIFTI and/or NIFTI files instead.

Value

Named character vector of the written files

Connectome Workbench

This function interfaces with the `"-cifti-create-dense-timeseries"`, `"-cifti-create-dense-scalar"`, or `"-cifti-create-label"` Workbench Command, depending on the input.

See Also

Other common: `is.cifti()`, `read_cifti()`, `resample_cifti()`, `smooth_cifti()`, `view_xifti()`

Other writing: `separate_cifti()`, `write_metric_gifti()`, `write_subcort_nifti()`, `write_surf_gifti()`, `write_xifti2()`

`write_metric_gifti` *Write a data matrix to a GIFTI metric file*

Description

Write the data for the left or right cortex to a metric GIFTI file.

Usage

```
write_metric_gifti(
  x,
  gifti_fname,
  hemisphere = c("left", "right"),
  intent = NULL,
  data_type = NULL,
  encoding = NULL,
  endian = c("LittleEndian", "BigEndian"),
  col_names = NULL,
  label_table = NULL
)
```

Arguments

`x`	A $V \times T$ data matrix (V vertices, T measurements). This can also be an object from `gifti::readgii`, or a length T list of length V vectors.
`gifti_fname`	Where to write the GIFTI file.
`hemisphere`	`"left"` (default) or `"right"`. Ignored if `data` is already a `"gifti"` object.
`intent`	"NIFTI_INTENT_*". `NULL` (default) will use metadata if `data` is a `"gifti"` object, or "NONE" if it cannot be inferred. If not `NULL` and `data` is a `"gifti"` object, it will overwrite the existing intent. See https://nifti.nimh.nih.gov/nifti-1/documentation/nifti1fields/nifti1fields_pages/group__NIFTI1__INTENT__CODES.html/document_vi .
`data_type`	the type of `data`: "NIFTI_TYPE_*" where * is "INT32" or "FLOAT32". If `NULL` (default), the data type will be inferred. If not `NULL` and `data` is a `"gifti"` object, it will overwrite the existing data type.
`encoding`	One of "ASCII", "Base64Binary", or "GZipBase64Binary". If `NULL` (default), will use the metadata if `data` is a GIFTI object, or "ASCII" if the `data_type` is "NIFTI_TYPE_INT32" and "GZipBase64Binary" if the `data_type` is "NIFTI_TYPE_FLOAT32". If not `NULL` and `data` is a `"gifti"` object, it will overwrite the existing data type.
`endian`	"LittleEndian" (default) or "BigEndian". If `data` is a `"gifti"` object, it will overwrite the existing endian.
`col_names`	The names of each data column in `gii` (or entries in `gii$data`).

label_table — A data.frame with labels along rows. The row names should be the label names. The column names should be among: "Key", "Red", "Green", "Blue", and "Alpha". The "Key" column is required whereas the others are optional (but very often included). Values in the "Key" column should be non-negative integers, typically beginning with 0. The other columns should be floating-point numbers between 0 and 1.

Although CIFTI files support a different label table for each data column, GIFTI files only support a single label table. So this label table should be applicable to each data column.

Value

Whether the GIFTI was successfully written

See Also

Other writing: `separate_cifti()`, `write_cifti()`, `write_subcort_nifti()`, `write_surf_gifti()`, `write_xifti2()`

---

write_subcort_nifti *Write subcortical data to NIFTI files*

Description

Write subcortical data to NIFTI files representing the data values, subcortical structure labels, and volumetric mask. The input formats of `subcortVol`, `subcortLabs`, and `subcortMask` correspond to the data structures of `xifti$data$subcort`, `xifti$meta$subcort$labels` , and `xifti$meta$subcort$mask` respectively. `subcortVol` and `subcortLabs` should be vectorized, so if they are volumes consider using `RNifti::writeNIfTI`.

Usage

```
write_subcort_nifti(
  subcortVol,
  subcortLabs,
  subcortMask,
  trans_mat = NULL,
  trans_units = NULL,
  col_names = NULL,
  label_table = NULL,
  subcortVol_fname,
  subcortLabs_fname,
  ROIsubcortVol_fname = NULL,
  fill = 0
)
```

Arguments

subcortVol	A vectorized data matrix: V voxels by T measurements
subcortLabs	Numeric (0 and 3-21) or factor vector corresponding to subcortical structure labels. See substructure_table.
subcortMask	Logical volumetric mask. Values of 0 represent out-of-mask voxels (not subcortical), and values of 1 represent in-mask voxels (subcortical),
trans_mat	The TransformationMatrixIJKtoXYZ, or equivalently the desired sform matrix (srow_x, srow_y and srow_z) to write. If NULL, do not write it (all zeroes).
trans_units	The units of trans_mat. Currently not used.
col_names	(Optional) Column names.
label_table	(Optional) data.frame of labels and their colors.
subcortVol_fname, subcortLabs_fname, ROIsubcortVol_fname	File path to a NIFTI to save the corresponding data. ROIsubcortVol_fname is optional but the rest is required.
fill	Values to use for out-of-mask voxels. Default: 0.

Details

All file path arguments are required except ROIsubcortVol_fname. If not provided, the volumetric mask will not be written. (It's redundant with the 0 values in subcortLabs_fname because valid labels have positive indexes.)

Note that for label data (i.e. if label_table is provided) only one label table can be saved.

Value

Named character vector with the "subcortVol", "subcortLabs", and "ROIsubcortVol" file names (if written)

Connectome Workbench

This function interfaces with the "-volume-label-import" Workbench Command.

See Also

Other writing: separate_cifti(), write_cifti(), write_metric_gifti(), write_surf_gifti(), write_xifti2()

`write_surf_gifti` *Write a "surf" to a GIFTI surface file*

Description

Write the data for the left or right surface to a surface GIFTI file.

Usage

```
write_surf_gifti(
  x,
  gifti_fname,
  hemisphere = c("left", "right"),
  encoding = NULL,
  endian = c("LittleEndian", "BigEndian")
)

write_surf(
  x,
  gifti_fname,
  hemisphere = c("left", "right"),
  encoding = NULL,
  endian = c("LittleEndian", "BigEndian")
)
```

Arguments

`x`	A `"surf"` object, an object from `gifti::readgii`, or a list with elements "pointset" and "triangle".
`gifti_fname`	Where to write the GIFTI file.
`hemisphere`	"left" (default) or "right". Ignored if `data` is already a "gifti" object, or if it is a `"surf"` object with the hemisphere metadata already specified.
`encoding`	A length-2 vector with elements chosen among "ASCII", "Base64Binary", and "GZipBase64Binary". If `NULL` (default), will use the metadata if `data` is a "gifti" object, or "GZipBase64Binary" for the "pointset" and "ASCII" for the "triangles" if `data` is not already a GIFTI.
`endian`	"LittleEndian" (default) or "BigEndian".

Value

Whether the GIFTI was successfully written

See Also

Other writing: separate_cifti(), write_cifti(), write_metric_gifti(), write_subcort_nifti(), write_xifti2()

Other surface-related: add_surf(), boundary_mask_surf(), edit_mask_surf(), even_vert_samp(), is.surf(), load_surf(), mask_surf(), read_surf(), resample_surf(), rotate_surf(), surf_area(), view_surf()

---

write_xifti2 *Write a "xifti" object to GIFTI and NIFTI files*

Description

Write metric or label GIFTIs for the cortical surface data and NIFTIs for the subcortical labels and mask in a "xifti" object. Each present brainstructure will be written; if a brainstructure is absent the corresponding file is not written.

Usage

```
write_xifti2(
  xifti,
  brainstructures = NULL,
  cortexL_fname = NULL,
  cortexR_fname = NULL,
  subcortVol_fname = NULL,
  subcortLabs_fname = NULL,
  ROI_brainstructures = "all",
  ROIcortexL_fname = NULL,
  ROIcortexR_fname = NULL,
  ROIsubcortVol_fname = NULL,
  write_dir = NULL,
  verbose = FALSE
)
```

Arguments

xifti	A "xifti" object.
brainstructures	(Optional) character vector indicating a subset of brain structure(s) to write: "left" cortex, "right" cortex, and/or "subcortical" structures. Can also be "all" to write out all existing brain structures. Default: c("left","right").
cortexL_fname, cortexR_fname	(Optional) GIFTI file names (*.[func/label].gii) to save the [left/right] cortex data to. dtseries and dscalar files should use "func", whereas dlabel files should use "label". If NULL and write_dir is provided, defaults to "*[L/R].\[func/label\].gii", where * is the file name component of cifti_fname.

`subcortVol_fname, subcortLabs_fname`
: (Optional) NIFTI file names to save the subcortical [volume/labels] to. Provide both or neither.

  If `NULL` and `write_dir` is provided, defaults to `"*[/.labels].nii"`, where * is the file name component of `cifti_fname`.

`ROI_brainstructures`
: Which ROIs should be obtained? `"all"` (default) to obtain ROIs for each of the `brainstructures`. `NULL` to not obtain any ROIs. This should be a subset of `brainstructures`.

`ROIcortexL_fname, ROIcortexR_fname`
: (Optional) GIFTI file names (*.[func/label].gii) to save the [left/right] cortex ROI to. dtseries and dscalar files should use "func", whereas dlabel files should use "label".

  If `NULL` and `write_dir` is provided, defaults to `"*ROI_[L/R].\[func/label\].gii"`, where * is the file name component of `cifti_fname`.

  The cortical ROIs typically represent the medial wall mask, with values of 1 for in-ROI (non-medial wall) vertices and 0 for out-of-ROI (medial wall) vertices. Will be written in `write_dir`.

`ROIsubcortVol_fname`
: (Optional) NIFTI file names to save the subcortical ROI to.

  If `NULL` and `write_dir` is provided, defaults to `"*ROI.nii"`, where * is the file name component of `cifti_fname`.

  The subcortical ROI typically represents the volumetric mask for the entire subcortical structure, with values of 1 for in-ROI (in subcortex) voxels and 0 for out-of-ROI (not in subcortex) voxels. Will be written in `write_dir`.

`write_dir`
: (Optional) A path to an existing directory. If provided, every component in the `"xifti"` will be written to this directory, using automatically-generated names if their `*_fname` argument was not provided. Otherwise if `write_dir` is `NULL`, only the components for which their `*_fname` was provided will be written.

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

Value

List of written files

See Also

Other writing: `separate_cifti()`, `write_cifti()`, `write_metric_gifti()`, `write_subcort_nifti()`, `write_surf_gifti()`

Index