Package 'ssbrain'

May 9, 2023

Type Package

Title Take Brain Screenshots Using CIFTI And Border Files

Version 0.1.0

Author Nathan L. Anderson

Maintainer Nathan L. Anderson <NLAnderson9@gmail.com>

Description This package allows you to provide CIFTI (dscalar/dlabel/dconn) and/or border files to take screenshots of brain surfaces, similar to what you would be able to produce in Connectome Workbench.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports dplyr,
gifti,
gplots,
grDevices,
magick,
rgl,
tidyr,
viridisLite,
xml2

Depends R (>= 4.1.1)

R topics documented:

`+.ssbrain` *Overloaded Plus*

Description

This is an overloaded function for '+' that allows for combining different `ssbrain` elements.

Usage

```
## S3 method for class 'ssbrain'
obj1 + obj2
```

Arguments

`obj1`	The existing `ssbrain` object
`obj2`	The new `ssbrain` object to add

Value

A new `ssbrain` object with information from both previous objects.

`.onAttach` *Message on Open*

Description

Present the start message when opening

Usage

```
.onAttach(...)
```

Arguments

`...`	Items passed to attach

`add_border` *(Internal) Set a Border*

Description

The internal function that calculates the result for `ss_border`

Usage

```
add_border(obj1, obj2)
```

Arguments

`obj1`	The existing `ssbrain` object
`obj2`	The new `ssborder` object to add

`add_dconn` *(Internal) Set a Seed*

Description

The internal function that calculates the result for `ss_seed`

Usage

```
add_dconn(obj1, obj2)
```

Arguments

`obj1`	The existing `ssbrain` object
`obj2`	The new `ssdconn` object to add

add_dlabel *(Internal) Set a Dlabel*

Description

The internal function that calculates the result for `ss_dlabel`

Usage

```
add_dlabel(obj1, obj2)
```

Arguments

`obj1`	The existing `ssbrain` object
`obj2`	The new `ssdlabel` object to add

add_dscalar *(Internal) Set a Dscalar*

Description

The internal function that calculates the result for `ss_dscalar`

Usage

```
add_dscalar(obj1, obj2)
```

Arguments

`obj1`	The existing `ssbrain` object
`obj2`	The new `ssdscalar` object to add

add_view *(Internal) Set a Viewpoint*

Description

The internal function that calculates the result for `ss_view`

Usage

```
add_view(obj1, obj2)
```

Arguments

`obj1`	The existing `ssbrain` object
`obj2`	The new `ssview` object to add

`captureBrain` *Capture a Visually-displayed Brain Surface*

Description

This function opens an rgl window and displays a brain surface, then saves a screenshot of it.

Usage

```
captureBrain(
  brain,
  hemisphere,
  filename,
  width = 800,
  height = 500,
  crop = TRUE,
  cropmargin = 10
)
```

Arguments

`brain`	An `ssbrain` object created by ss_surf (and potentially modified by other functions)
`hemisphere`	Which brain hemisphere to display (either "left" or "right")
`filename`	A string with the full path to the PNG file you'd like to create
`width`	The width of the window. Default is 800.
`height`	The height of the window. Default is 500.
`crop`	Whether or not to crop the resulting image. Default is TRUE.
`cropmargin`	The margin of whitespace to leave around the brain when cropping. Default is 10.

Examples

```
## Not run:
my_brain = ss_surf(surf="fsaverage6") +
  ss_dscalar(dscalar_filename = "/path/to/my_file.dscalar.nii")

captureBrain(my_brain, hemisphere = "left", filename = "/path/to/my/output_image.png")

## End(Not run)
```

checkBin — *Check data bins*

Description

Check to see which bin contains the value

Usage

```
checkBin(x, y)
```

Arguments

x	The levels (bins) of the output of the `cut` function
y	The value to check

Value

TRUE or FALSE

closeBrainViewers — *Close All Open Brain Windows*

Description

Closes all currently-open brain windows

Usage

```
closeBrainViewers()
```

Examples

```
## Not run:
my_brain = ss_surf(surf="fsaverage6") +
  ss_dscalar(dscalar_filename = "/path/to/my_file.dscalar.nii")

showBrain(my_brain, hemisphere = "left")

closeBrainViewers()

## End(Not run)
```

`colorbarGenerator` *Generate Color Bar*

Description

Given a name or color palette(s), generate the necessary information to use them in brain maps

Usage

```
colorbarGenerator(colorbar_name, neg_colorpalette, pos_colorpalette)
```

Arguments

`colorbar_name`	A string with the name of the color bar
`neg_colorpalette`	(Optional argument) A custom color palette created by colorRampPalette, to be used for negative values
`pos_colorpalette`	(Optional argument) A custom color palette created by colorRampPalette, to be used for positive values

`createOutputList` *Create Color Maps*

Description

Creates the necessary outputs for colormaps used by ss_dscalar and ss_seed

Usage

```
createOutputList(colorbar_name, pos_list, neg_list)
```

Arguments

`colorbar_name`	A string with the name of the color bar
`pos_list`	The list of positive color values to create the palette
`neg_list`	The list of negative color values to create the palette

distill_dconn *'Distill' a Connectivity Dconn File*

Description

This function "distills" a symmetrical .dconn.nii file containing connectivity information. It exports the correlation between each vertex and the specified vertex, resulting in a vector the length of the number of vertices. Functionally, it converts the dconn into a dscalar.

Usage

```
distill_dconn(dconn_filepath, seed_value, num_verts)
```

Arguments

`dconn_filepath`	A string with the full path to the .dconn.nii file
`seed_value`	A value for the seed (the row number, not vertex number)
`num_verts`	The number of vertices (row/columns) in your dconn file

Value

An object of the same type returned by importCifti when used on a dscalar file.

Examples

```
## Not run:
dconn_seed_data = distill_dconn(dconn_filepath = "/path/to/my/dconn/my_data.dconn.nii",
                                seed_value = 2214, num_verts = 81924)

## End(Not run)
```

exportCifti *Export a CIFTI Data File or Border File*

Description

This function exports a CIFTI data file (or border file) from R, using a vector of data

Usage

```
exportCifti(cifti_filename, data, luttpath)
```

Arguments

`cifti_filename`	A string with the full path to the CIFTI data file you'd like to create
`data`	A vector of data values.
`luttpath`	(Optional argument) A path to a lutt color file; used for dlabel and border files. Default is the Yeo 17-parcel color scheme.

Details

This function currently only supports `.dscalar.nii`, `.dlabel.nii`, and `.border` files. For border files, outputs should be named filename.border. These will automatically be changed into left/right hemisphere files called filename_lh.border and filename_rh.border

Examples

```
## Not run:
my_data = runif(81924, 1.0, 100.0) # a vector of 81,924 random values between 1 and 100
exportCifti("/path/to/my/output.dscalar.nii", my_data)

my_data = sample(1:15, 81924) # a vector of 81,924 random integers between 1 and 15
exportCifti("/path/to/my/output.dlabel.nii", my_data)

## End(Not run)
```

`getSeedVector` *Get a Linear Vector of Seeds*

Description

This function calculates the values of every seed that lie in a (relatively) straight line between the two seeds provided. It allows you to "draw" a series of seeds across the cortex, by looping through the resulting list with `ss_seed`.

Usage

```
getSeedVector(brain, seed_start, seed_end, dlabel)
```

Arguments

`brain`	Any `ss_surf` object in the same mesh space that you plan to use for `ss_seed` (i.e. fsaverage6, fsaverage7)
`seed_start`	The value of the starting seed to draw from
`seed_end`	The value of the ending seed to draw toward
`dlabel`	(Optional argument)

Value

A list with three items: vertices (Euclidean coordinates for each vertex), faces (the 3 vertex indices that make up each face), and num_verts (the total number of vertices in the mesh)

Examples

```
## Not run:
my_brain = ss_surf(surf = "fsaverage6")
my_seed_vector = getSeedVector(my_brain, 33424, 13264)

## End(Not run)
```

`get_viewmatrix` *Calculation of View Matrix*

Description

This function calculates the coordinates that `rgl` needs to rotate a brain.

Usage

```
get_viewmatrix(hemisphere, side, wb_X, wb_Y, wb_Z)
```

Arguments

`hemisphere`	"left" or "right"
`side`	"lateral" or "medial"
`wb_X`	The x-axis rotation as shown in Connectome Workbench
`wb_Y`	The y-axis rotation as shown in Connectome Workbench
`wb_Z`	The z-axis rotation as shown in Connectome Workbench

Value

A view matrix that can be passed to rgl::view3d as a userMatrix.

`importBorder` *Import a Border File*

Description

This function imports a border file into R, providing information on vertices (and their ordering) used to form borders.

Usage

```
importBorder(border_name, border_surfname)
```

Arguments

`border_name`	A string with the full path to a border file
`border_surfname`	A string with the full path to the GIFTI surface used to create the border

Value

A gifti file containing border information

Examples

```
## Not run:
border_data = importBorder("/path/to/border/my_borderfile_lh.border",
                           border_surfname = "/path/to/surface/lh.pial_infl2.surf.gii")

## End(Not run)
```

`importCifti` *Import a CIFTI Data File*

Description

This function imports a CIFTI data file into R, providing information for each surface vertex (depending on the filetype).

Usage

```
importCifti(cifti_name, data_only = FALSE)
```

Arguments

`cifti_name`	A string with the full path to a CIFTI data file
`data_only`	If TRUE, only provides a vector of the data. If FALSE, provides a full `gifti` object, including metadata. Default is FALSE.

Details

This function currently only supports `.dscalar.nii`, `.dlabel.nii`, and `.dconn.nii` files.

Value

A gifti object (data_only = FALSE) or a vector of data (data_only = TRUE)

Examples

```
## Not run:
cifti = importCifti("/path/to/cifti/my_file.dlabel.nii")

# If `data_only` is FALSE (the default), the data inside the resulting
# object is located in `$data$normal`. For example:

cifti_object = importCifti("/path/to/cifti/my_cifti.dscalar.nii")
cifti_data = cifti_object$data$normal

# is equivalent to

cifti_data = importCifti("/path/to/cifti/my_cifti.dscalar.nii",
                          data_only = TRUE)

## End(Not run)
```

`importSurface` *Import a GIFTI Surface File*

Description

This function imports a GIFTI surface file into R, providing information for vertex coordinates in Euclidean space and the vertices used to draw triangular faces.

Usage

```
importSurface(surface_name)
```

Arguments

`surface_name`	A string with the full path to a GIFTI surface file

Value

A list with three items: vertices (Euclidean coordinates for each vertex), faces (the 3 vertex indices that make up each face), and num_verts (the total number of vertices in the mesh)

Examples

```
## Not run:
surface_data = importSurface("/path/to/surface/lh.pial_infl2.surf.gii")

## End(Not run)
```

`is.border` *Check if Border*

Description

This function checks if an object is of the class `ssborder`.

Usage

```
is.border(x)
```

Arguments

`x`	The object to check

Value

TRUE or FALSE

Examples

```
## Not run:
new_border = ss_border(border_filename = "/path/to/my/datafile.dlabel.nii", hemisphere = "left")
is.border(new_border)

## End(Not run)
```

`is.dconn` *Check if Dconn*

Description

This function checks if an object is of the class `ssdconn`.

Usage

```
is.dconn(x)
```

Arguments

`x`	The object to check

Value

TRUE or FALSE

Examples

```
## Not run:
new_dconn = ss_seed(dconn_filename = "/path/to/my/datafile.dconn.nii", seed_value = 2213)
is.dconn(new_dconn)

## End(Not run)
```

`is.dlabel` *Check if Dlabel*

Description

This function checks if an object is of the class `ssdlabel`.

Usage

```
is.dlabel(x)
```

Arguments

`x`	The object to check

Value

TRUE or FALSE

Examples

```
## Not run:
new_dlabel = ss_dlabel(dlabel_filename = "/path/to/my/datafile.dlabel.nii")
is.dlabel(new_dlabel)

## End(Not run)
```

`is.dscalar` *Check if Dscalar*

Description

This function checks if an object is of the class `ssdscalar`.

Usage

```
is.dscalar(x)
```

Arguments

`x`	The object to check

Value

TRUE or FALSE

Examples

```
## Not run:
new_dscalar = ss_dscalar(dscalar_filename = "/path/to/my/datafile.dscalar.nii")
is.dscalar(new_dscalar)

## End(Not run)
```

`is.view` *Check if View*

Description

This function checks if an object is of the class `ssview`.

Usage

```
is.view(x)
```

Arguments

`x`	The object to check

Value

TRUE or FALSE

Examples

```
## Not run:
new_view = ss_view(side="lateral", rotation = "inferior_temporal")
is.view(new_view)

## End(Not run)
```

`listColorbars` *List Available Colorbars*

Description

This function lists the colorbars available in the `ssbrain` package for use with the `ss_dscalar` and `ss_seed` functions.

Usage

```
listColorbars()
```

`listRotations` *List Available Rotations*

Description

This function lists the pre-set rotations available in the `ssbrain` package for use with the `ss_view` function.

Usage

```
listRotations()
```

`rgb_fun1` *Create a Color Code*

Description

Create a color HEX code from an RGB triple in a vector

Usage

```
rgb_fun1(x)
```

Arguments

`x`	A vector with three RGB values

set_wbpath	*Set Workbench Directory*

Description

This allows you to set the path to your Workbench executable file.

Usage

```
set_wbpath(value)
```

Arguments

value	A string with the full filepath to your Workbench executable. It should end with "wb_command"

Details

This sets a global option for your R environment. It only needs to be run the first time you start R. It is advised that you put this function at the top of every ssbrain script.

Examples

```
## Not run:
set_wbpath("/path/to/directory/for/workbench/wb_command")

## End(Not run)
```

showBrain	*Visually Display Brain Surface*

Description

This function opens an rgl window and displays a brain surface.

Usage

```
showBrain(brain, hemisphere, width = 800, height = 500)
```

Arguments

brain	An ssbrain object created by ss_surf (and potentially modified by other functions)
hemisphere	Which brain hemisphere to display (either "left" or "right")
width	The width of the window. Default is 800.
height	The height of the window. Default is 500.

Examples

```
## Not run:
my_brain = ss_surf(surf="fsaverage6") +
  ss_dscalar(dscalar_filename = "/path/to/my_file.dscalar.nii")

showBrain(my_brain, hemisphere = "left")

## End(Not run)
```

`ss_border` *Set a Border*

Description

This function sets a border data file on the surface of the brain

Usage

```
ss_border(
  border_filename,
  hemisphere,
  borders = NULL,
  border_width = 5,
  border_colors = NULL,
  offset = TRUE
)
```

Arguments

`border_filename`	A string with the full filepath to a border file
`hemisphere`	The hemisphere of the border file, either "left" or "right
`borders`	A single border, either an integer corresponding to the border number in the file (e.g. 5) or the name in the file (e.g. "LANG"). Can also be a vector of integers/names if multiple borders should be displayed. If omitted, all borders in the file will be plotted.
`border_width`	The width of the borders to be plotted. Defaults to 5.
`border_colors`	A list of colors the same length as `borders`; colors can be R color names (e.g. "red"), hex codes (e.g. "#FF0000"), or RGB triples (e.g. c(255,0,0)). If omitted, the default colors of the file will be used.
`offset`	Whether the border should be slightly raised up (offset) above the brain (TRUE) or not (FALSE). Defaults to TRUE.

Details

Note: Adding multiple `ss_border` items to the same object will cause overlapping brain maps, with the most recently-added on top.

Examples

```
## Not run:
# Here, all borders will be plotted and the file's colorscheme will be used
my_brain = ss_surf(surf="fsaverage6") +
  ss_border(border_filename = "/path/to/my/datafile_lh.border", hemisphere = "left")

# Here, only borders 1 and 5 from the file will be used, and they are colored "red" and "green"
my_brain = ss_surf(surf="fsaverage6")

my_brain_new = my_brain +
  ss_border(border_filename = "/path/to/my/datafile_lh.border",
            hemisphere = "left",
            borders = c(1,5),
            border_colors = list("red", "green"))

# Here, labels 8 through 10 will be plotted, with different
# color specifications used for each
my_brain = ss_surf(surf="fsaverage6") +
  ss_border(border_filename = "/path/to/my/datafile_lh.border",
            hemisphere = "left",
            borders = 8:10,
            border_colors = list("#FF00FF", c(212,118,97), "palegreen"))

# Here, three borders are plotted by name, and all are black
my_brain = ss_surf(surf="fsaverage6") +
  ss_border(border_filename = "/path/to/my/datafile_lh.border",
            hemisphere = "left",
            borders = c("LANG", "VIS", "AUD"),
            border_colors = "black")

## End(Not run)
```

`ss_dlabel` *Set a Dlabel*

Description

This function sets a dlabel data file on the surface of the brain

Usage

```
ss_dlabel(dlabel_filename, labels = NULL, colors = NULL)
```

Arguments

`dlabel_filename`	A string with the full filepath to a dlabel file
`labels`	A single label, either an integer corresponding to the label number in the file (e.g. 5) or the name in the file (e.g. "LANG"). Can also be a vector of integers/names if multiple labels should be displayed. If omitted, all labels in the file will be plotted.

colors	A list of colors the same length as labels; colors can be R color names (e.g. "red"), hex codes (e.g. "#FF0000"), or RGB triples (e.g. c(255,0,0)). If omitted, the default colors of the file will be used.

Details

Note: Adding multiple ss_dlabel items to the same object will cause overlapping brain maps, with the most recently-added on top.

Examples

```
## Not run:
# Here, all labels will be plotted and the file's colorscheme will be used
my_brain = ss_surf(surf="fsaverage6") +
  ss_dlabel(dlabel_filename = "/path/to/my/datafile.dlabel.nii")

# Here, only labels 1 and 5 from the file will be used, and they are colored "red" and "green"
my_brain = ss_surf(surf="fsaverage6")

my_brain_new = my_brain +
  ss_dlabel(dlabel_filename = "/path/to/my/datafile.dlabel.nii",
            labels = c(1,5),
            colors = list("red", "green"))

# Here, labels 8 through 10 will be plotted, with different
# color specifications used for each
my_brain = ss_surf(surf="fsaverage6") +
  ss_dlabel(dlabel_filename = "/path/to/my/datafile.dlabel.nii",
  labels = 8:10,
  colors = list("#FF00FF", c(212,118,97), "palegreen"))

## End(Not run)
```

---

ss_dscalar *Set a Dscalar*

Description

This function sets a dscalar data file on the surface of the brain

Usage

```
ss_dscalar(
  dscalar_filename,
  colorbar = "FSL",
  show = "all",
  pos_thresh = 1,
  neg_thresh = -1,
  pos_colorrange = c(1, 3),
  neg_colorrange = c(-1, -3),
  pos_palette,
  neg_palette
)
```

Arguments

`dscalar_filename`	A string with the full filepath to a dscalar file
`colorbar`	A string containing the name of a colorbar to use (same names as Connectome Workbench)
`show`	Whether to show positive data, negative data, or all data. Options are "all", "pos", or "neg". Defaults to "all".
`pos_thresh`	The threshold below which to hide positive data. Defaults to 1.
`neg_thresh`	The option above which to hide negative data. Defaults to -1.
`pos_colorrange`	The positive range in which to spread the `colorbar`'s colors. Defaults to c(1,3)
`neg_colorrange`	The negative range in which to spread the `colorbar`'s colors. Defaults to c(-1,-3)
`pos_palette`	(Experimental) A color palette created by `colorRampPalette`; overrides `colorbar` for positive colors.
`neg_palette`	(Experimental) A color palette created by `colorRampPalette`; overrides `colorbar` for negative colors.

Details

Note: Adding multiple `ss_dscalar` items to the same object will cause overlapping brain maps, with the most recently-added on top.

Examples

```
## Not run:
my_brain = ss_surf(surf="fsaverage6") +
  ss_dscalar(dscalar_filename = "/path/to/my/datafile.dscalar.nii")

my_brain = ss_surf(surf="fsaverage6")

my_brain_new = my_brain +
  ss_dscalar(dscalar_filename = "/path/to/my/datafile.dscalar.nii",
             colorbar = "ROY-BIG-BL",
             show= "pos",
             pos_thresh = 10,
             pos_colorrange = c(10,50))

## End(Not run)
```

`ss_seed` *Set a Seed*

Description

This function sets a seed (based on a vertex row number) and then creates a map from a corresponding dconn file containing a functional connectivity matrix.

Usage

```
ss_seed(
  dconn_filename,
  seed_value,
  colorbar = "JET256",
  thresh = 0.2,
  colorrange = c(0.2, 0.6),
  show_seed_sphere = TRUE,
  seed_sphere_size = 2,
  seed_sphere_color = "white",
  seed_data,
  palette
)
```

Arguments

`dconn_filename`	A string with the full filepath to a dconn file
`seed_value`	The number of the seed (row number, not vertex number)
`colorbar`	A string containing the name of a colorbar to use (same names as Connectome Workbench)
`thresh`	The option above which to hide data. Defaults to 0.2.
`colorrange`	The range in which to spread the `colorbar`'s colors. Defaults to c(0.2, 0.6)
`show_seed_sphere`	Whether or not to draw a sphere where the seed is (TRUE) or not (FALSE). Defaults to TRUE.
`seed_sphere_size`	If show_seed_sphere is TRUE, the radius of the sphere. Defaults to 2.
`seed_sphere_color`	If show_seed_sphere is TRUE, the color of the sphere. Defaults to "white".
`seed_data`	(Optional argument) Rather than passing a dconn file, you can directly pass in an R matrix containing the functional connectivity data. This can save time if you want to draw multiple seeds from the same dconn matrix. This overrides `dconne_filename`.
`palette`	(Experimental) A color palette created by `colorRampPalette`; overrides `colorbar`.

Details

Note: Adding multiple `ss_seed` items to the same object will cause overlapping brain maps, with the most recently-added on top.

Examples

```
## Not run:
my_brain = ss_surf(surf="fsaverage6") +
  ss_seed(dconn_filename = "/path/to/my/datafile.dconn.nii", seed_value = 31272)

my_brain = ss_surf(surf="fsaverage6")

my_brain_new = my_brain +
  ss_dscalar(dconn_filename = "/path/to/my/datafile.dconn.nii",
             seed_value = 2213,
             colorbar = "TURBO",
             thresh = 0.3,
             colorrange = c(0.3, 0.8),
             seed_sphere_size = 4,
             seed_sphere_color = "red")

# Warning: this may take a while (especially for fsaverage7),
# and will likely require a lot of memory!
dconn_data = importCifti(cifti_name = "/path/to/my/datafile.dconn.nii", data_only = TRUE)
my_brain = ss_surf(surf="fsaverage6") +
  ss_seed(seed_data = dconn_data, seed_value = 31272)

## End(Not run)
```

---

ss_surf *Set a Surface Mesh*

Description

This is the core function of the ssbrain package. It creates an object that will store all of the surface information you want to display. It sets which surface mesh you plan to use.

Usage

```
ss_surf(surf = NULL, surfL = NULL, surfR = NULL)
```

Arguments

surf	A string with the surface mesh you wish to use. Options are "fsaverage6" and "fsaverage7".
surfL	(Optional argument) If you don't want to use fs6/fs7, you can provide the full path to a surface GIFTI file for the left hemisphere.
surfR	(Optional argument) If you don't want to use fs6/fs7, you can provide the full path to a surface GIFTI file for the right hemisphere.

Value

An ssbrain object with the specified surface mesh(es).

Examples

```
## Not run:
my_brain = ss_surf(surf="fsaverage6")

my_brain = ss_surf(surfL = "/path/to/surface/lh.pial_infl2.surf.gii")

## End(Not run)
```

`ss_view` *Set a Viewpoint*

Description

This function sets the view of the brain in the viewer window. It determines if a lateral or medial view is used, and the x/y/z axis rotation of the brain.

Usage

```
ss_view(side = "lateral", rotation = c(0, 90, 90))
```

Arguments

`side`	Which side of the brain to show, either "lateral" or "medial". Defaults to "lateral".
`rotation`	The rotation applied to the `side` chosen. Either a pre-defined rotation (e.g. "orbitofrontal"), or a vector of x/y/z angles (e.g. c(90,120,90)).

Details

Note: `ss_view` is the only ss_* function that **overwrites** any prexisting view, rather than adding on additional elements.

Examples

```
## Not run:
my_brain = ss_surf(surf="fsaverage6") +
  ss_view(side="lateral", rotation = "inferior_temporal")

my_brain = ss_surf(surf="fsaverage6")

my_brain_new = my_brain +
  ss_view(side="medial", rotation = c(30,90,90))

## End(Not run)
```

`start_msg` *Start Message*

Description

The start message to present when opening

Usage

```
start_msg()
```

Index