Package 'threeBrain'

May 30, 2022

Type Package

Title 3D Brain Visualization

Version 0.2.5

Description A fast, interactive cross-platform, and easy to share 'WebGL'-based 3D brain viewer that visualizes 'FreeSurfer' and/or 'AFNI/SUMA' surfaces. The viewer widget can be either standalone or embedded into 'R-shiny' applications. The standalone version only require a web browser with 'WebGL2' support (for example, 'Chrome', 'Firefox', 'Safari'). It can be inserted into any websites; see <https://dipterix.org/project/threebrain/>) as an example. The 'R-shiny' support allows the 3D viewer to be dynamically generated from reactive user inputs. This feature has been fully adopted by 'RAVE' <https://openwetware.org/wiki/RAVE>, an interactive toolbox to analyze 'iEEG' data. Documentation about 'threeBrain' is provided by <https://dipterix.org/threeBrain/> and several vignettes included in this package. To cite the package, please check our 'NeuroImage' paper by Magnotti, Wang, and Beauchamp (2020, <doi:10.1016/j.neuroimage.2020.117341>), or see 'citation(``threeBrain'')' for details.

License GPL-3 | file LICENSE

Encoding UTF-8

RoxygenNote 7.2.0

Language en-US

URL https://dipterix.org/threeBrain/
https://github.com/dipterix/threeBrain

BugReports https://github.com/dipterix/threeBrain/issues

Imports utils, grDevices, graphics, dipsaus, xml2, shiny (>= 1.2.0), digest (>= 0.6.22), freesurferformats (>= 0.1.7), jsonlite (>= 1.5), stringr (>= 1.3.1), htmlwidgets (>= 1.3), R6 (>= 2.3.0), gifti (>= 0.7.5), oro.nifti (>= 0.9.1)

Suggests knitr, rmarkdown, DT

VignetteBuilder knitr

NeedsCompilation no

Author Zhengjia Wang [aut, cre, cph],
John Magnotti [aut],
Xiang Zhang [ctb, res],
Brian Metzger [res],
Elizabeth Nesbitt [res],
Meng Li [ths],
Michael Beauchamp [aut, dtc]

Maintainer Zhengjia Wang <dipterix.wang@gmail.com>

Repository CRAN

Date/Publication 2022-05-30 07:00:05 UTC

R topics documented:

`AbstractGeom` *R6 Class - Abstract Class of Geometries*

Description

R6 Class - Abstract Class of Geometries

Author(s)

Zhengjia Wang

`BlankGeom` *A geometry that renders nothing*

Description

This is mainly used when you want to upload group data only

`brain_proxy` *Shiny Proxy for Viewer*

Description

Shiny Proxy for Viewer

Usage

```
brain_proxy(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

`outputId`	shiny output ID
`session`	shiny session, default is current session (see `domains`)

Value

`R6` class `ViewerProxy`

`brain_setup` *Setup Package, Install Environment*

Description

Setup Package, Install Environment

Usage

```
brain_setup(continued = FALSE, show_example = TRUE, ...)
```

Arguments

`continued`	logical, there are two phases of setting up environment. You probably need to restart R session after the first phase and continue setting up.
`show_example`	whether to show example of 'N27' subject at the end.
`...`	ignored

Author(s)

Zhengjia Wang

`check_freesurfer_path` *Function to check whether 'FreeSurfer' folder has everything we need*

Description

Function to check whether 'FreeSurfer' folder has everything we need

Usage

```
check_freesurfer_path(
  fs_subject_folder,
  autoinstall_template = FALSE,
  return_path = FALSE,
  check_volume = FALSE,
  check_surface = FALSE
)
```

Arguments

fs_subject_folder	character, path to ‘fs‘ project directory or ‘RAVE‘ subject directory
autoinstall_template	logical, whether ‘N27‘ brain should be installed if missing
return_path	logical, whether to return ‘FreeSurfer‘ path
check_volume	logical, whether to check volume data
check_surface	logical, whether to check surface data (not implemented yet)

Value

logical whether the directory is valid or, if return_path is true, return ‘FreeSurfer‘ path

---

create_group *Create a geometry group containing multiple geometries*

Description

Create a geometry group containing multiple geometries

Usage

```
create_group(name, position = c(0, 0, 0), layer = 1)
```

Arguments

name	string, name of the geometry
position	x,y,z location of the group
layer	layer of the group. reserved

Details

A geometry group is a container of multiple geometries. The geometries within the same group share the same shift and rotations (see example 1). In ECoG/iEEG world, you might have ’MRI’, ’CT’, ’FreeSurfer’ that have different orientations. For example, if you want to align MRI to FreeSurfer, Instead of calculating the position of each geometries, you can just put all MRI components into a group, and then set transform of this group, making the group aligned to FreeSurfer.

GeomGroup also can be used to store large data. To generate 3D viewer, ‘threeBrain‘ needs to dynamically serialize data into JSON format, which can be read by browsers. However, a FreeSurfer brain might be ~30 MB. This is a very large size and might take ~5 seconds to serialize. To solve this problem, GeomGroup supports cache in its ‘set_group_data‘ method. This method supports caching static serialized data into a JSON file, and allows the files to be loaded as static data objects. By "static", I mean the data is not supposed to be dynamic, and it should be "read-only". In JavaScript code, I also optimized such that you don’t need to load these large datasets repeatedly. And this allows you to load multiple subjects’ brain in a short time.

Value

a GeomGroup instance

Author(s)

Zhengjia Wang

Examples

```
# Example 1: relative position

# create group
g = create_group('Group A')

# create two spheres at 10,0,0, but s2 is relative to group A
s1 = geom_sphere('Sphere 1', radius = 2, position = c(10,0,0))
s2 = geom_sphere('Sphere 2', radius = 2, position = c(10,0,0), group = g)

# set transform (rotation)
g$set_transform(matrix(c(
  0,1,0,0,
  1,0,0,0,
  0,0,1,0,
  0,0,0,1
), byrow = TRUE, ncol = 4))

# global position for s2 is 0,10,0
threejs_brain(s1, s2)

# Example 2: cache

## Not run:

# download N27 brain
# Make sure you have N27 brain downloaded to `default_template_directory()`
# download_N27()

template_dir <- default_template_directory()

dat = threeBrain::read_fs_asc(
  file.path(template_dir, 'N27/surf/lh.pial.asc')
)
vertex = dat$vertices[,1:3]
face = dat$faces[,1:3]

# 1. dynamically serialize
mesh = geom_freemesh('lh', vertex = vertex, face = face, layer = 1)

# 2. cache
# Create group, all geometries in this group are relatively positioned
tmp_file = tempfile()
mesh = geom_freemesh('Left Hemisphere cached', vertex = vertex,
```

```
                    face = face, cache_file = tmp_file)

## End(Not run)
```

DataCubeGeom *R6 Class - Generate Data Cube Geometry*

Description

R6 Class - Generate Data Cube Geometry

Author(s)

Zhengjia Wang

DataCubeGeom2 *R6 Class - Generate Data Cube Geometry via 3D Volume Texture*

Description

R6 Class - Generate Data Cube Geometry via 3D Volume Texture

Author(s)

Zhengjia Wang

---

default_template_directory
Default Directory to Store Template Brain

Description

Default Directory to Store Template Brain

Usage

```
default_template_directory(check = FALSE)
```

Arguments

check	logical, check if the folder is missing, is so, create one. This option ensures the folder is always created.

Details

When `threeBrain.template_dir` is not set or invalid, the function checks 'RAVE' (R Analysis and Visualization for 'iEEG', https://openwetware.org/wiki/RAVE) folder at home directory. If this folder is missing, then returns results from `R_user_dir('threeBrain', 'data')`. To override the default behavior, use `options(threeBrain.template_dir=...)`.

Value

A directory path where template brain is stored at; see also `download_N27`

Examples

```
default_template_directory()
```

`FreeGeom` *R6 Class - Generate Geometry from Vertices and Face Indices*

Description

R6 Class - Generate Geometry from Vertices and Face Indices

`freesurfer_brain` *Read 'FreeSurfer' surface and volume files*

Description

Read 'FreeSurfer' surface and volume files

Usage

```
freesurfer_brain(
  fs_subject_folder,
  subject_name,
  additional_surfaces = NULL,
  aligned_ct = NULL,
  use_cache = TRUE,
  use_141 = getOption("threeBrain.use141", TRUE)
)

freesurfer_brain2(
  fs_subject_folder,
  subject_name,
  volume_types = "t1",
```

```
  surface_types = "pial",
  curvature = "sulc",
  atlas_types = "aparc+aseg",
  ct_path = NULL,
  use_cache = TRUE,
  use_141 = getOption("threeBrain.use141", TRUE),
  ...
)
```

Arguments

`fs_subject_folder`	character, 'FreeSurfer' subject folder, or 'RAVE' subject folder
`subject_name`	character, subject code to display with only letters and digits
`additional_surfaces`	character array, additional surface types to load, such as 'white', 'smoothwm'
`aligned_ct`	character, path to 'ct_aligned_mri.nii.gz', used for electrode localization
`use_cache`	logical, whether to use cached 'json' files or from raw 'FreeSurfer' files
`use_141`	logical, whether to use standard 141 brain for surface file, default is `getOption('threeBrain.use141', TRUE)`
`volume_types`	volume types, right now only support T1 image
`surface_types`	surface types to load
`curvature`	curvature data. Only support `"sulc"` for current version
`atlas_types`	atlas types to be loaded, choices are `'aparc+aseg'`, `'aparc.a2009s+aseg'`, `'aparc.DKTatlas+aseg'`, `'aseg'`
`ct_path`	an aligned CT file in 'Nifti' format
`...`	ignored

Details

This function is under FreeSurfer license. 1. Volumes: 3D viewer uses 'mri/T1.mgz' from 'FreeSurfer' to show the volume information. 'T1.mgz' results from step 1 to 5 in 'FreeSurfer' command 'recon-all -autorecon1', which aligns the original 'DICOM' image to 'RAS' coordinate system, resamples to volume with `256x256x256` voxels (tri-linear by default, check https://surfer.nmr.mgh.harvard.edu/fswiki/recon-all for more information).

2. Surface: There are two options for surface files. The first choice is using 'std.141' brain generated by 'AFNI/SUMA'. This surface file re-calculates vertices from standard 141 space, which averages the "surface" of 141 subjects. If you want to map surface electrodes across different subjects, you might want to consider this case as it's especially designed for surface mapping. However, you'll need 'AFNI/SUMA' installed to generate the surface file. The details can be found via https://openwetware.org/wiki/Beauchamp:CorticalSurfaceHCP, and the 'AFNI/SUMA' command related is 'SurfToSurf'. Please generate the files to '[FREESURFER SUBJECT DIR]/SUMA/'. The file name follows the convention of 'std.141.[lr]h.[SURFACE TYPE].[POSTFIX]', where 'lh' means left hemisphere and 'rh' means right hemisphere; 'SURFACE TYPE' can be 'pial', 'white', 'smoothwm', and 'POSTFIX' can be 'asc', 'gii'. If multiple files for the same surface type exists,

the search order will be 'asc > gii'. The other option is to use mesh files directly from 'FreeSurfer' output located at '[FREESURFER SUBJECT DIR]/surf'. If you want to use these surface, make sure they are converted to 'asc' or 'gii' format.

3. Electrode registration and transforms This package provides two ways to map electrodes to standard space. For surface electrodes, if standard 141 brain is provided, then the first option is to snap electrodes to the nearest vertices in subject space. The key is the vertex number matches across different subjects, hence the location of corresponding vertices at template brain are the mapped electrode coordinates. If standard 141 brain is missing, or the electrode type is 'stereo EEG', then the second option is volume mapping. The idea is to map electrodes to 'MNI305' brain. The details can be found at https://surfer.nmr.mgh.harvard.edu/fswiki/CoordinateSystems. To perform volume mapping, we need 'FreeSurfer' folder 'mri/transforms'. Currently, only linear 'Talairach' transform matrix is supported (located at 'talairach.xfm').

4. Coordinates The 3D viewer in this package uses the center of volume as the origin (0, 0, 0).

Author(s)

Zhengjia Wang

Examples

```
## Not run:
# Please run `download_N27()` if `N27` is not at `default_template_directory()`

# Import from `FreeSurfer` subject folder
brain = threeBrain::freesurfer_brain(
  fs_subject_folder = file.path(default_template_directory(), 'N27'),
  subject_name = 'N27',
  additional_surfaces = c('white', 'smoothwm')
)

# Visualize. Alternatively, you can use brain$plot(...)
plot( brain )

## End(Not run)
```

`freesurfer_lut` *Query the 'FreeSurfer' labels*

Description

Query the 'FreeSurfer' labels

Usage

```
freesurfer_lut
```

Format

An object of class `list` of length 2.

Details

The 'FreeSurfer' atlases use https://surfer.nmr.mgh.harvard.edu/fswiki/FsTutorial/AnatomicalROI/FreeSurferColorLUT look-up table to query indexes. The 'threeBrain' electrode localization also uses this table to export the `'FSLabel'` from electrode. If volume type is set to `'aparc_aseg'`, then please also use this table to filter.

Examples

```
freesurfer_lut$from_key(0:10)

freesurfer_lut$get_key("ctx-lh-supramarginal")
```

GeomGroup *R6 Class - Generate Group of Geometries*

Description

R6 Class - Generate Group of Geometries

Author(s)

Zhengjia Wang

---

geom_freemesh *Creates any mesh geometry given vertices and face indices*

Description

Creates any mesh geometry given vertices and face indices

Usage

```
geom_freemesh(
  name,
  vertex = NULL,
  face = NULL,
  position = c(0, 0, 0),
  layer = 1,
  cache_file = NULL,
  group = NULL
)
```

Arguments

name	unique string in a scene to tell apart from different objects
vertex	position of each vertices (3 columns)
face	face indices indicating which 3 vertices to be linked (3 columns)
position	x,y,z location of the geometry
layer	visibility of the geometry, used when there are multiple cameras 1 is visible for all cameras
cache_file	cache vertex and face data into group
group	a GeomGroup object, if null, then the group will be generated automatically

Details

When generating a free mesh internally, a group must be specified, therefore if group is NULL here, then a group will be generated. However, it's always recommended to pass a group to the free mesh.

Author(s)

Zhengjia Wang

Examples

```
## Not run:
# Make sure you have N27 brain downloaded to `default_template_directory()`
# threeBrain::download_N27()

n27_dir = file.path(default_template_directory(), "N27")
surf_type = 'pial'

# Locate mesh files
lh = read_fs_asc(file.path(n27_dir, sprintf('surf/lh.%s.asc', surf_type)))
rh = read_fs_asc(file.path(n27_dir, sprintf('surf/rh.%s.asc', surf_type)))

# Create groups
group = create_group(name = sprintf('Surface - %s (N27)', surf_type))

# create mesh
lh_mesh = geom_freemesh(
  name = sprintf('FreeSurfer Left Hemisphere - %s (N27)', surf_type),
  vertex = lh$vertices[,1:3],
  face = lh$faces[,1:3],
  group = group
)
rh_mesh = geom_freemesh(
  name = sprintf('FreeSurfer Right Hemisphere - %s (N27)', surf_type),
  vertex = rh$vertices[,1:3],
  face = rh$faces[,1:3],
  group = group
)
```

```
# Render
threejs_brain(lh_mesh, rh_mesh)



## End(Not run)
```

`geom_sphere` *Create sphere geometry*

Description

Create sphere geometry

Usage

```
geom_sphere(
  name,
  radius,
  position = c(0, 0, 0),
  layer = 1,
  group = NULL,
  value = NULL,
  time_stamp = NULL
)
```

Arguments

`name`	unique string in a scene to tell apart from different objects
`radius`	size of sphere
`position`	x,y,z location of the sphere
`layer`	visibility of the geometry, used when there are multiple cameras 1 is visible for all cameras
`group`	a GeomGroup object
`value, time_stamp`	color of the sphere, used for animation/color rendering

Author(s)

Zhengjia Wang

Examples

```
# Create a sphere with animation
g = lapply(1:10, function(ii){
  v = rep(ii, 10)
  v[1:ii] = 1:ii
 geom_sphere(paste0('s', ii), ii, value = v, position = c(11 * ii, 0,0), time_stamp = (1:10)/10)
})
threejs_brain(.list = g)
```

`get_digest_header` *Function to read digest header*

Description

Function to read digest header

Usage

```
get_digest_header(file, key, if_error = NULL, .list = NULL)
```

Arguments

`file`	file path to a 'JSON' file
`key`	character, key to extract
`if_error`	value to return if key not found or read error occurs
`.list`	alternative list to supply if file is missing

`import-fs-suma` *Import 'FreeSurfer' or 'SUMA' files into the viewer structure*

Description

Import 'T1-MRI', surface files, curvature/'sulcus', atlas, and 'Talairach' transform matrix into 'json' format. These functions are not intended to be called directly, use `import_from_freesurfer` instead.

Usage

```
import_fs(
  subject_name,
  fs_path,
  quiet = FALSE,
  dtype = c("T1", "surface", "curv", "atlas_volume", "atlas_surface", "xform"),
  sub_type = NULL,
  hemisphere = c("l", "r"),
  ...
)

import_suma(
  subject_name,
  fs_path,
  quiet = FALSE,
  dtype = c("T1", "surface", "curv", "atlas_volume", "atlas_surface", "xform"),
  sub_type = NULL,
  hemisphere = c("l", "r"),
  ...
)
```

Arguments

subject_name	character, subject code
fs_path	path to 'FreeSurfer' folder
quiet, ...	passed from or to other methods.
dtype	data type to import, choices are 'T1', 'surface', 'curv', 'atlas_volume', 'atlas_surface', 'xform'
sub_type	detailed files to import. 'atlas_surface' is not supported for now
hemisphere	which hemisphere to import, ignored when dtype is in 'T1', 'atlas_volume', 'atlas_surface', 'xform'.

Value

logical, TRUE if the file is or has been cached, or FALSE if the file is missing.

---

import_from_freesurfer

Import from 'FreeSurfer' and create 'JSON' cache for 3D viewer

---

Description

Import from 'FreeSurfer' and create 'JSON' cache for 3D viewer

Usage

```
import_from_freesurfer(fs_path, subject_name, quiet = FALSE)
```

Arguments

fs_path	'FreeSurfer' subject directory
subject_name	subject code
quiet	whether to suppress message or not

Value

None.

---

localization_module *Launch a 'shiny' application to localize electrodes*

Description

Launch a 'shiny' application to localize electrodes

Usage

```
localization_module(
  subject_code,
  fs_path,
  ct_path = NULL,
  surfaces = "pial",
  use_141 = TRUE,
  shiny_options = list(launch.browser = TRUE),
  save_path = tempfile(pattern = "electrode", fileext = ".csv"),
  ...,
  control_presets = NULL,
  side_display = FALSE,
  controllers = list()
)
```

Arguments

subject_code	subject code
fs_path	the subject's 'FreeSurfer' path
ct_path	the file path of 'CT' scans that have already been aligned to 'T1'; must be in 'NIFTI' format
surfaces	which surfaces to load
use_141	whether to try 'SUMA' standard 141 surface; default is true
shiny_options	shiny application options; see options in shinyApp

save_path	a temporary file where the electrode table should be cached; this file will be used to keep track of changes in case the application is crashed or shutdown
...	other parameters to pass into freesurfer_brain2
control_presets, side_display, controllers	passed to threejs_brain

Value

A list of 'ui' elements, 'server' function, and a stand-alone 'app'

Examples

```
# This example require N27 template brain to be installed
# see `?download_N27` for details

# using N27 to localize
fs_path <- file.path(default_template_directory(), "N27")
if(dir.exists(fs_path)){
  module <- localization_module("N27", fs_path)

  if(interactive()){
    print(module$app)
  }
}
```

---

merge_brain *Create Multi-subject Template*

Description

Create Multi-subject Template

Usage

```
merge_brain(
  ...,
  .list = NULL,
  template_surface_types = NULL,
  template_subject = unname(getOption("threeBrain.template_subject", "N27")),
  template_dir = default_template_directory()
)
```

Arguments

`..., .list`	`Brain2` objects
`template_surface_types`	which template surface types to load, default is auto-guess
`template_subject`	character, subject code to be treated as template, default is 'N27'
`template_dir`	the parent directory where template subject is stored in

Author(s)

Zhengjia Wang

`read_fs_asc` *Read 'FreeSurfer' ascii file*

Description

Read 'FreeSurfer' ascii file

Usage

```
read_fs_asc(file)
```

Arguments

`file`	file location

Value

a list of vertices and face indices

`read_fs_labels` *Read FreeSurfer Annotations*

Description

Read FreeSurfer Annotations

Usage

```
read_fs_labels(path, vertex_number)
```

Arguments

`path`	label path
`vertex_number`	force to reset vertex number if raw file is incorrect

read_fs_m3z *Read 'FreeSurfer' m3z file*

Description

Read 'FreeSurfer' m3z file

Usage

```
read_fs_m3z(filename)
```

Arguments

filename	file location, usually located at 'mri/transforms/talairach.m3z'

Details

An 'm3z' file is a 'gzip' binary file containing a dense vector field that describes a 3D registration between two volumes/images. This implementation follows the 'Matlab' implementation from the 'FreeSurfer'. This function is released under the 'FreeSurfer' license: https://surfer.nmr.mgh.harvard.edu/fswiki/FreeSurferSoftwareLicense.

Value

registration data

read_fs_mgh_mgz *Read 'FreeSurfer' 'mgz/mgh' file*

Description

Read 'FreeSurfer' 'mgz/mgh' file

Usage

```
read_fs_mgh_mgz(filename)
```

Arguments

filename	file location

Value

list contains coordinate transforms and volume data

read_gii2 *Function to load surface data from 'Gifti' files*

Description

The function 'read_gii2' is a dynamic wrapper of Python 'nibabel' loader. If no Python is detected, it will switch to 'gifti::readgii'.

Usage

```
read_gii2(path)
```

Arguments

path	'Gifti' file path

Format

An R function acting as safe wrapper for `nibabel.load`.

read_mgz *Function to load 'FreeSurfer' 'mgz/mgh' file*

Description

The function 'read_mgz' is a dynamic wrapper of Python 'nibabel' loader. If no Python is detected, it will switch to built-in function 'read_fs_mgh_mgz', which has limited features.

Usage

```
read_mgz(path)
```

Arguments

path	'mgz/mgh' file path

Format

An R function acting as safe wrapper for `nibabel.load`.

`renderBrain` *Shiny Renderer for threeBrain Widgets*

Description

Shiny Renderer for threeBrain Widgets

Arguments

`expr`	R expression that calls three_brain function or Brain object
`env`	environment of expression to be evaluated
`quoted`	is expr quoted? Default is false.

Author(s)

Zhengjia Wang

`reorient_volume` *Function to reshape data to 'RAS' order*

Description

Function to reshape data to 'RAS' order

Usage

```
reorient_volume(volume, Torig)
```

Arguments

`volume,`	3-mode tensor (voxels), usually from 'mgz', 'nii', or 'BRIK' files
`Torig`	a 4x4 transform matrix mapping volume ('CRS') to 'RAS'

Value

Reshaped tensor with dimensions corresponding to 'R', 'A', and 'S'

save_brain *Save threeBrain widgets to local file system*

Description

Save threeBrain widgets to local file system

Usage

```
save_brain(
  widget,
  directory,
  filename = "index.html",
  assetpath = "lib/",
  datapath = "lib/threebrain_data-0/",
  title = "3D Viewer",
  as_zip = FALSE
)
```

Arguments

widget	generated from function 'threejs_brain'.
directory	directory to save the widget.
filename	default is 'index.html', filename of the widget index file.
assetpath	where to put css or JavaScript to, must be relative to directory.
datapath	where to store data to, must be relative to directory.
title	widget title.
as_zip	whether to create zip file "compressed.zip".

Author(s)

Zhengjia Wang

SphereGeom *R6 Class - Generate Sphere Geometry*

Description

R6 Class - Generate Sphere Geometry

Author(s)

Zhengjia Wang

`template_subject` *Download and Manage Template Subjects*

Description

Download and Manage Template Subjects

Usage

```
download_template_subject(
  subject_code = "N27",
  url,
  template_dir = default_template_directory()
)

download_N27(make_default = FALSE, ...)

set_default_template(
  subject_code,
  view = TRUE,
  template_dir = default_template_directory()
)

threebrain_finalize_installation(
  upgrade = c("ask", "always", "never"),
  async = TRUE
)

available_templates()
```

Arguments

`subject_code`	character with only letters and numbers (Important); default is 'N27'
`url`	zip file address; must be specified if `subject_code` is not from the followings: `'bert'`, `'cvs_avg35'`, `'cvs_avg35_inMNI152'`, `'fsaverage'`, `'fsaverage_sym'`, or `'N27'`
`template_dir`	parent directory where subject's 'FreeSurfer' folder should be stored
`make_default`	logical, whether to make 'N27' default subject
`...`	more to pass to `download_template_subject`
`view`	whether to view the subject
`upgrade`	whether to check and download 'N27' brain interactively. Choices are 'ask', 'always', and 'never'
`async`	whether to run the job in parallel to others; default is true

Details

To view electrodes implanted in multiple subjects, it's highly recommended to view them in a template space The detail mapping method is discussed in function `freesurfer_brain`.

To map to a template space, one idea is to find someone whose brain is normal. In our case, the choice is subject 'N27', also known as 'Colin 27'. function `download_N27` provides a simple and easy way to download a partial version from the Internet.

If you have any other ideas about template brain, you can use function `set_default_template(subject_code, template_dir)` to redirect to your choice. If your template brain is a 'Zip' file on the Internet, we provide function `download_template_subject` to automatically install it.

Author(s)

Zhengjia Wang

---

threejsBrainOutput *Shiny Output for threeBrain Widgets*

Description

Shiny Output for threeBrain Widgets

Arguments

`outputId`	unique identifier for the widget
`width, height`	width and height of the widget. By default width="100 and height="500px".
`reportSize`	whether to report widget size in shiny `session$clientData`

Author(s)

Zhengjia Wang

---

threejs_brain *Create a Threejs Brain and View it in Browsers*

Description

Create a Threejs Brain and View it in Browsers

Usage

```
threejs_brain(
  ...,
  .list = list(),
  width = NULL,
  height = NULL,
  background = "#FFFFFF",
  cex = 1,
  timestamp = TRUE,
  side_canvas = FALSE,
  side_zoom = 1,
  side_width = 250,
  side_shift = c(0, 0),
  side_display = TRUE,
  control_panel = TRUE,
  control_presets = NULL,
  control_display = TRUE,
  camera_center = c(0, 0, 0),
  camera_pos = c(500, 0, 0),
  start_zoom = 1,
  coords = NULL,
  symmetric = 0,
  default_colormap = "Value",
  palettes = NULL,
  value_ranges = NULL,
  value_alias = NULL,
  show_inactive_electrodes = TRUE,
  surface_colormap = system.file("palettes", "surface", "ContinuousSample.json",
    package = "threeBrain"),
  voxel_colormap = system.file("palettes", "datacube2", "FreeSurferColorLUT.json",
    package = "threeBrain"),
  videos = list(),
  widget_id = "threebrain_data",
  tmp_dirname = NULL,
  debug = FALSE,
  token = NULL,
  controllers = list(),
  browser_external = TRUE,
  global_data = list(),
  global_files = list(),
  custom_javascript = NULL,
  show_modal = "auto"
)
```

Arguments

`..., .list`	geometries inherit from AbstractGeom
`width, height`	positive integers. Width and height of the widget. By default width='100%',

	and height varies.
background	character, background color such as "#FFFFFF" or "white"
cex	positive number, relative text magnification level
timestamp	logical, whether to show time-stamp at the beginning
side_canvas	logical, enable side cameras to view objects from fixed perspective
side_zoom	numerical, if side camera is enabled, zoom-in level, from 1 to 5
side_width	positive integer, side panel size in pixels
side_shift	integer of length two, side panel shift in pixels ('CSS style': top, left)
side_display	logical, show/hide side panels at beginning
control_panel	logical, enable control panels for the widget
control_presets	characters, presets to be shown in control panels
control_display	logical, whether to expand/collapse control UI at the beginning
camera_center	numerical, length of three, XYZ position where camera should focus at
camera_pos	XYZ position of camera itself, default (0, 0, 500)
start_zoom	numerical, positive number indicating camera zoom level
coords	NULL to hide coordinates or numeric vector of three.
symmetric	numerical, default 0, color center will be mapped to this value
default_colormap	character, which color map name to display at startup
palettes	named list, names corresponds to color-map names if you want to change color palettes
value_ranges	named list, similar to palettes, value range for each values
value_alias	named list, legend title for corresponding variable
show_inactive_electrodes	logical, whether to show electrodes with no values
surface_colormap	a color map or its path generated by create_colormap(gtype="surface") to render surfaces vertices; see create_colormap for details.
voxel_colormap	a color map or its path generated by create_colormap(gtype="volume") to render volume such as atlases; see create_colormap for details.
videos	named list, names corresponds to color-map names, and items are generated from video_content
widget_id	character, internally used as unique identifiers for widgets; only use it when you have multiple widgets in one website
tmp_dirname	character path, internally used, where to store temporary files
debug	logical, internally used for debugging
token	unique character, internally used to identify widgets in JS localStorage
controllers	list to override the settings, for example proxy$get_controllers()

`browser_external`
: logical, use system default browser (default) or builtin one.

`global_data, global_files`
: internally use, mainly to store orientation matrices and files.

`custom_javascript`
: customized temporary 'JavaScript' code that runs after ready state; available 'JavaScript' variables are:

  `'groups'` input information about each group

  `'geoms'` input information about each geometry

  `'settings'` input information about canvas settings

  `'scene'` 'threejs' scene object

  `'canvas'` canvas object

  `'gui'` controls data panel

  `'presets'` preset `'gui'` methods

`show_modal`
: logical or `"auto"`, whether to show a modal instead of direct rendering the viewers; designed for users who do not have 'WebGL' support; only used in shiny applications

Author(s)

Zhengjia Wang

Examples

```
library(threeBrain)

# Please use `download_N27` to download N27 Collins template brain
n27_path <- file.path(default_template_directory(), "N27")
if( dir.exists(n27_path) ) {

  brain <- freesurfer_brain2(n27_path, "N27",
                             surface_types = c('pial', 'smoothwm'))
  print(brain)

  brain$plot(
    background = "#000000",
    controllers = list(
      'Voxel Type' = 'aparc_aseg',
      'Surface Type' = 'smoothwm',
      'Surface Color' = 'sync from voxels',
      'Blend Factor' = 1,
      'Right Opacity' = 0.3,
      'Overlay Sagittal' = TRUE
    ),
    show_modal = TRUE
  )

}
```

three_scatter *3D Scatter Plot*

Description

3D Scatter Plot

Usage

```
three_scatter(
  x,
  y,
  z,
  size = 1,
  col = 1,
  label = NULL,
  group = 1,
  timestamp = NULL,
  pal = NULL,
  scale = 1,
  axis = TRUE,
  control_panel = TRUE,
  control_presets = NULL,
  camera_pos,
  ...
)
```

Arguments

x, y, z	numeric vectors with the same length n.
size	size for each point.
col	color vector/matrix, can be either numeric or factor. Its length (vector) or nrow (matrix) must be either n or 1.
label	text label of each observation.
group	categorical group names of each points.
timestamp	numeric vector, length of 0 or ncol(col).
pal	color palette, vector of colors, can be integers.
scale	'auto', NULL, or numeric, rescale the final coordinates. Default 1, no re-scale.
axis	logical, draw axis.
control_panel	logical, show sidebar (control panel).
control_presets	if control_panel is true, which widgets to show.
camera_pos	initial camera position, auto assign if missing.
...	other arguments passing to threejs_brain.

Author(s)

Zhengjia Wang

Examples

```
#' Continuous color example:

data("iris")
three_scatter(x = iris$Sepal.Length, y = iris$Sepal.Width,
              z = iris$Petal.Length, size = 0.1,
              col = iris$Petal.Width, group = iris$Species,
              pal = c('orange', 'blue3', 'darkgreen'),
              start_zoom = 12, axis = FALSE)


# Discrete example:

x = rnorm(26, c(10, 10, -20))
y = rnorm(26, c(10, -10, 10))
z = rnorm(26, c(10, 40, -10))
three_scatter(x, y, z, size = 1, col = sample(letters[1:3], 20, TRUE),
              pal = c('orange', 'blue3', 'darkgreen'))
```

TubeGeom *R6 Class - Generate Tube Geometry*

Description

R6 Class - Generate Tube Geometry

Author(s)

Zhengjia Wang

---

video_content *Add video content to the viewer*

Description

Add video content to the viewer

Usage

```
video_content(
  path,
  duration = Inf,
  time_start = 0,
  asp_ratio = 16/9,
  local = TRUE
)
```

Arguments

path	local file path or 'URL'
duration	duration of the video
time_start	start time relative to the stimuli onset
asp_ratio	aspect ratio; default is 16/9
local	used only when path is a 'URL': whether to download the video before generating the viewer; see 'Details'

Details

The video path can be either local file path or a 'URL' from websites. When path is from the internet, there are two options: download the video before generating the viewer, or directly use the 'URL'.

If download happens before a viewer is generated (local=TRUE), then the video content is local. The viewer will be self-contained. However, the distribution will contain the video, and the archive size might be large.

If raw 'URL' is used (local=FALSE), then viewer is not self-contained as the video link might break anytime. The 'screenshot' and 'record' function might be limited if the 'URL' has different domain than yours. However, the distribution will not contain the video, hence smaller. This works in the scenarios when it is preferred not to share video files or they are licensed, or simply distribution is limited. Besides, this method is slightly faster than the local alternatives.

---

view_ct_t1 *View CT with T1 image*

Description

View aligned CT scan with T1 images

Usage

```
view_ct_t1(
  subject_code,
  fs_path,
  ct_path = file.path(fs_path, "RAVE", "coregistration", "ct2t1.nii.gz")
)
```

Arguments

subject_code	subject code
fs_path	FreeSurfer subject path
ct_path	where CT file is stored, require 'Nifti' format

---

voxel_colormap *Color maps for volume or surface data*

Description

Color maps for volume or surface data

Usage

```
create_colormap(
  gtype = c("surface", "volume"),
  dtype = c("continuous", "discrete"),
  key,
  color,
  value,
  alpha = FALSE,
  con = NULL,
  ...
)

save_colormap(cmap, con)

freeserfer_colormap(con)

load_colormap(con)
```

Arguments

gtype	geometry type, choices are "surface", "volume"
dtype	data type, "continuous" or "discrete"
key	non-negative integer vector corresponding to color values; its length must exceed 1; see 'Details'
color	characters, corresponding to color strings for each key
value	actual value for each key
alpha	whether to respect transparency
con	a file path to write results to or to read from. The file path can be passed as voxel_colormap into threejs_brain.
...	used by continuous color maps, passed to colorRampPalette
cmap	color map object

Details

Internal 'JavaScript' shader implementation uses integer color keys to connect color palettes and corresponding values. The keys must be non-negative.

Zero key is a special color key reserved by system. Please avoid using it for valid values.

Value

A list of color map information

Examples

```
# Creates a symmetric continuous colormap with 3 keys
# The color range is -10 to 10
# The colors are 'blue','white','red' for these keys

pal <- create_colormap(
  gtype = "volume", dtype = "continuous",
  key = c(1,2,3), value = c(-10,0,10),
  color = c('blue','white','red'))

print( pal )

# ---------------- Get colormap key from a value ------------

# returns key index starting from
pal$get_key( -10 )

# nearest value
pal$get_key( 2 )

# set threshold, key is now 0 (no color)
pal$get_key( 2, max_delta = 1 )


# ---------------- Save and load ----------------
f <- tempfile( fileext = '.json' )
save_colormap( pal, f )
cat(readLines(f), sep = '\n')

load_colormap(f)
```

---

voxel_cube *Generate volume data from 'MNI' coordinates*

Description

Generate volume data from 'MNI' coordinates

Usage

```
add_voxel_cube(brain, name, cube, size = c(256, 256, 256), matrix_world = NULL)

create_voxel_cube(
  mni_ras,
  value,
  colormap,
  keys = colormap$get_key(value),
  dimension = c(256, 256, 256)
)
```

Arguments

brain	a 'threeBrain' brain object generated from freesurfer_brain2 or merge_brain. If you have 'rave' package installed, the brain can be generated from rave::rave_brain2
name	the name of voxel cube, only letters, digits and '_' are allowed; other characters will be replaced by '_'
cube	a 3-mode array; see the following example
size	the actual size of the volume, usually dot multiplication of the dimension and voxel size
matrix_world	the transform matrix of the volume
mni_ras	'MNI' 'RAS' coordinates, should be a n-by-3 matrix
value	data values (length n); used if keys is missing
colormap	a color map generated from create_colormap; see voxel_colormap for details
keys	integer color-keys generated from a color map with length of n; alternatively, you could specify value and colormap to generate keys automatically
dimension	volume dimension; default is a 256 x 256 x 256 array cube; must be integers and have length of 3

Value

create_voxel_cube returns a list of cube data and other informations; add_voxel_cube returns the brain object

Examples

```
# requires N27 brain to be installed
# use `download_N27()` to download template Collins brain


# sample MNI coords
tbl <- read.csv(system.file(
  'sample_data/example_cube.csv', package = 'threeBrain'
))
head(tbl)
```

```
# load colormap
cmap <- load_colormap(system.file(
  'palettes/datacube2/Mixed.json', package = 'threeBrain'
))

x <- create_voxel_cube(
  mni_ras = tbl[, c('x', 'y', 'z')],
  keys = tbl$key,
  dimension = c(128, 128, 128)
)


n27_path <- file.path(default_template_directory(), "N27")
if( dir.exists(n27_path) ) {
  brain <- merge_brain()

  # or add_voxel_cube(brain, 'example', x$cube)
  x$add_to_brain(brain, 'example')

  brain$plot(controllers = list(
    "Voxel Type" = 'example',
    'Right Opacity' = 0.3,
    'Left Opacity' = 0.3,
    'Background Color' = '#000000'
  ), voxel_colormap = cmap)
}
```

Index