Package ‘threeBrain’

January 20, 2020

Type Package
Title 3D Brain Visualization
Version 0.1.5

Description In neuroscience, 'AFNI/SUMA' is a great tool to visualize 3D brain. However, it takes efforts to interact and share the viewer to others. In addition, 'AFNI/SUMA' doesn't support Windows platform. In the 'EEG/iEEG' field, it's hard to have multiple cortical electrodes mapped to a template brain for group analysis. Therefore this package is written aimed at providing a fast, stable, interactive and easy to share tool based on 'Three.js', a 'WebGL' engine to render 3D objects in the web browser such that we can display brain surfaces on webpage interactively. This package translates R objects to JavaScript objects via 'JSON' format, and provides 'R-Shiny' interface to manipulate geometries interactively. The visualizations can also serve as standalone widgets that can be easily shared across different platforms. Along with 'rave', another package developed by Beauchamp's lab at Baylor College Medicine, this package provides solutions to easily map surface electrodes from multiple subjects to one template 141 brain.

License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.0.2
Language en-US

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

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

Imports grDevices, graphics, dipsaus, shiny (>= 1.2.0), digest (>= 0.6.22), freesurferformats (>= 0.1.7), crayon (>= 1.3.4), base64enc (>= 0.1-3), htmltools (>= 0.3.6), jsonlite (>= 1.5), stringr (>= 1.3.1), htmlwidgets (>= 1.3), R6 (>= 2.3.0), startup (>= 0.12.0), reticulate (>= 1.13), gifti (>= 0.7.5), oro.nifti (>= 0.9.1)

Suggests knitr, rmarkdown, pryr,

NeedsCompilation no
Author  Zhengjia Wang [aut, cre, cph],
         John Magnotti [aut],
         Brian Metzger [aut],
         Elizabeth Nesbitt [res],
         Michael Beauchamp [aut, dtc, fnd]
Maintainer  Zhengjia Wang <zhengjia.wang@rice.edu>
Repository  CRAN
Date/Publication  2020-01-20 20:50:02 UTC

R topics documented:

AbstractGeom ....................................................... 3
BlankGeom ............................................................ 3
brain_proxy ........................................................... 3
brain_setup ........................................................... 4
check_freesurfer_path .............................................. 4
create_group ......................................................... 5
DataCubeGeom ........................................................ 7
DataCubeGeom2 ....................................................... 7
FreeGeom ............................................................. 7
freesurfer_brain ..................................................... 8
GeomGroup ........................................................... 10
gem_freemesh ....................................................... 10
gem_sphere .......................................................... 11
get_digest_header .................................................... 12
import_from_freesurfer ............................................. 13
merge_brain .......................................................... 13
ravepy ............................................................... 14
read_fs_asc ........................................................... 15
read_fs_labels ....................................................... 15
read_fs_m3z .......................................................... 16
read_fs_mgh_mgz ..................................................... 16
read_gii2 ............................................................. 17
read_mgz ............................................................. 17
renderBrain .......................................................... 18
reorient_volume ....................................................... 18
save_brain ............................................................ 19
SphereGeom .......................................................... 19
template_subject ..................................................... 20
threejsBrainOutput .................................................. 21
threejs_brain ........................................................ 21
three_scatter ........................................................ 23

Index 25
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 getDefaultReactiveDomain)

Value

R6 class ViewerProxy
brain_setup  Setup Package, Install Environment

Description

Setup Package, Install Environment

Usage

brain_setup(
  continued = FALSE,
  show_example = TRUE,
  use_python = FALSE,
  try_conda = 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.
use_python  whether to install python toolbox (recommended, but not by default)
try_conda  try to use ‘conda’ to create ‘RAVEPy’ environment

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
)
create_group

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

# download N27 brain
# Make sure you have N27 brain downloaded to ~/rave_data/others/threeBrain/N27
# download_N27()

dat = threeBrain::read_fs_asc('~/rave_data/others/three_brain/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)
pryr::object_size(mesh) # ~10 MB
threejs_brain(mesh) # ~3 seconds to serialize

# 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)
pryr::object_size(mesh) # ~0.5 MB
threejs_brain(mesh) # serialize at once, load in browser
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

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

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

```r
freesurfer_brain2(
  fs_subject_folder,
  subject_name,
  volume_types = "t1",
  surface_types = "pial",
  curvature = "sulc",
  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
- `...` 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](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](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 details can be found at [https://surfer.nmr.mgh.harvard.edu/fswiki/CoordinateSystems](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**

```r
# Please run 'download_N27()' if 'N27' is not at '~/rave_data/others/three_brain/'

# Import from 'FreeSurfer' subject folder
brain = threeBrain::freesurfer_brain(
  fs_subject_folder = '~/rave_data/others/three_brain/N27/', subject_name = 'N27',
  additional_surfaces = c('white', 'smoothwm')
)

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

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

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.
 examples

# Make sure you have N27 brain downloaded to ~/rave_data/others/threeBrain/N27
# threeBrain::download_N27()

n27_dir = '~/rave_data/others/three_brain/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)
Usage

`geom_sphere`

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

```r
# Create a sphere with animation
v = rep(ii, 10)
v[1:ii] = 1:ii
g = lapply(1:10, function(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)`
import_from_freesurfer

Import from ‘FreeSurfer’ and create ‘JSON’ cache for 3D viewer

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

Description

Import from ‘FreeSurfer’ and create ‘JSON’ cache for 3D viewer

Usage

import_from_freesurfer(fs_path, subject_name)

Arguments

- fs_path: ‘FreeSurfer’ subject directory
- subject_name: subject code

Value

None.

merge_brain

Create Multi-subject Template

Description

Create Multi-subject Template

Usage

merge_brain(
  ..., .list = NULL,
  template_surface_types = NULL,
  template_subject = getOption("threeBrain.template_subject", "N27"),
  template_dir = getOption("threeBrain.template_dir", "~/rave_data/others/three_brain")
)
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

---

ravepy Manage Python Environment from RAVE

Description

Manage Python Environment from RAVE

Usage

ravepy_remove(conda = TRUE)

ravepy_check(quiet = FALSE)

ravepy_register()

ravepy_virtualenv_install(python_path = NULL)

ravepy_conda_install(python_path = NULL)

ravepy_info(quiet = FALSE, return_libs = FALSE)

Arguments

conda logical, is your environment created using conda?

quiet logical, suppress printing information.

python_path character, python path, auto-detected if NULL. Please specify Python3 path

return_libs logical, return python modules?

Author(s)

Zhengjia Wang
**read_fs_asc**

*Read 'FreeSurfer' ascii file*

**Description**

Read 'FreeSurfer' ascii file

**Usage**

```python
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**

```python
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

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

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, Norig)

Arguments

volume,
3-mode tensor (voxels), usually from ‘mgz’, ‘nii’, or ‘BRIK’ files

Norig
a 4×4 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**

```r
download_template_subject(subject_code = "N27",  
url = "https://github.com/dipterix/threeBrain-sample/releases/download/1.0.0/N27.zip",  
template_dir = getOption("threeBrain.template_dir", 
"~/rave_data/others/three_brain")
)
download_N27(make_default = FALSE, ...)

set_default_template(subject_code,  
view = TRUE,  
template_dir = getOption("threeBrain.template_dir", "~/rave_data/others/three_brain")
)
```

**Arguments**

- `subject_code` character with only letters and numbers (Important). Default is ‘N27’
- `url` zip file address
- `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

**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
```r
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,
val_ranges = NULL,
show_inactive_electrodes = TRUE,
widget_id = "threebrain_data",
tmp_dirname = NULL,
debug = FALSE,
token = NULL,
controllers = list(),
browser_external = TRUE,
global_data = list(),
global_files = list()
}

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 timestamp 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
three_scatter

val_ranges named list, similar to palettes, value range for each values

show_inactive_electrodes logical, whether to show electrodes with no values

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.

Author(s)

Zhengjia Wang

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

```r
# 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)
```

```r
# 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'))
```
Index

AbstractGeom, 3
BlankGeom, 3
brain_proxy, 3
brain_setup, 4
check_freesurfer_path, 4
create_group, 5
DataCubeGeom, 7
DataCubeGeom2, 7
download_N27(template_subject), 20
download_template_subject
  (template_subject), 20
FreeGeom, 7
freesurfer_brain, 8
freesurfer_brain2(freesurfer_brain), 8
geom_freemesh, 10
geom_sphere, 11
GeomGroup, 10
get_digest_header, 12
getDefaultReactiveDomain, 3
import_from_freesurfer, 13
merge_brain, 13
N27(template_subject), 20
ravepy, 14
ravepy_check(ravepy), 14
ravepy_conda_install(ravepy), 14
ravepy_info(ravepy), 14
ravepy_register(ravepy), 14
ravepy_remove(ravepy), 14
ravepy_virtualenv_install(ravepy), 14
read_fs_asc, 15
read_fs_labels, 15
read_fs_m3z, 16
read_fs_mgh_mgz, 16
read_gii2, 17
read_mgz, 17
renderBrain, 18
reorient_volume, 18
save_brain, 19
set_default_template
  (template_subject), 20
SphereGeom, 19
template_subject, 20
three_scatter, 23
threejs_brain, 21
threejsBrainOutput, 21