

Package: brainloc (via r-universe)

October 20, 2026

Type Package

Title Find Region Names and Coordinates for a Point on the Human Brain

Version 0.1.0

Maintainer Tim Schäfer <ts+code@rcmd.org>

Description Given a point on the human brain, identify the closest brain regions according to a brain atlas.

License MIT + file LICENSE

Encoding UTF-8

URL <https://github.com/dfsp-spirit/brainloc>

BugReports <https://github.com/dfsp-spirit/brainloc/issues>

Imports freesurferformats (>= 0.1.16), Rvcg (>= 0.20.2), collections, pkgfilecache (>= 0.1.1)

Suggests knitr, rmarkdown, testthat (>= 2.1.0), fsbrain (>= 0.5.0), rgl, regfusionr (>= 0.1.1)

Additional_repositories <https://dfsp-spirit.r-universe.dev>

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libxml2-dev libssl-dev

Repository <https://dfsp-spirit.r-universe.dev>

Date/Publication 2026-08-03 18:23:15 UTC

RemoteUrl <https://github.com/dfsp-spirit/brainloc>

RemoteRef v0.1.0

RemoteSha 9770ef0763a64c1bc7f163225938d5409e6a7199

Contents

brainparc	2
brainparc_fs	3
brainparc_neighbors	4
cluster_location_details	4
cluster_peaks	5
cluster_region_overlap	6
clusterinfo	6
clusterinfo_from_thresholded_overlay	7
coord_closest_regions	8
coord_closest_vertex	10
coord_MNI152_to_talairach	10
coord_MNI305_info	11
coord_MNI305_to_MNI152	12
fs_home	13
get_clusters	13
get_subjects_dir	14
get_surface	15
get_surface_coords	15
get_talairach_label	16
has_fs	17
hemilist	18
is.brainparc	18
is.clusterinfo	19
name_regions	19
print.brainparc	20
segmentation_centers	21
vertex_closest_regions	22
vertex_regions	23
Index	24

brainparc	Create a brain parcellation from custom data.
-----------	---

Description

Create a brain parcellation from custom data.

Usage

brainparc(surfaces, annots)

Arguments

surfaces	a named list with a single entry, which must contain a hemilist of fs.surface instances. The name can be anything describing the surface. Typically it is something like 'white', 'pial', or 'inflated' for FreeSurfer surfaces.
annots	a named list with at least one entry, all of which must contain a hemilist of minannot (or fs.annot) instances. The name can be anything describing the atlas from which the parcellation originates. Typically it is something like 'aparc', 'aparc.a2009s', or 'Desikan' for FreeSurfer parcellations.

Value

a brainparc instance

Note

A minannot for a surface is just a vector with length equal to the number of surface vertices, that holds one area label per vertex. An area label is an arbitrary character string, but typically a brain region name.

brainparc_fs	<i>Create a brain parcellation from a subject in a FreeSurfer recon-all output directory</i>
--------------	--

Description

Create a brain parcellation from a subject in a FreeSurfer recon-all output directory

Usage

```
brainparc_fs(subjects_dir, subject_id, surface = "white", atlas = c("aparc"))
```

Arguments

subjects_dir	string. The FreeSurfer SUBJECTS_DIR, i.e., a directory containing the data for all your subjects, each in a subdir named after the subject identifier.
subject_id	string. The subject identifier
surface	string. The surface name. E.g., "white", or "pial". Used to construct the name of the surface file to be loaded.
atlas	vector of character strings, the brain atlases to use. For this FreeSurfer version, something like "aparc" for the Desikan-Killiani atlas, "aparc.a2009s" for Destrieux, etc.

Value

a brainparc instance

brainparc_neighbors	<i>Compute adjacency of surface parcellation (annot) regions.</i>
---------------------	---

Description

Given a brain surface and a parcellation into regions, find out which regions are adjacent to each other. A region *i* is adjacent to another region *j* if an edge connects any vertex of *i* with any vertex of *j*. Works with brainparcellations.

Usage

```
brainparc_neighbors(bp)
```

Arguments

bp	a brainparc instance
----	----------------------

Value

hemilist of named integer matrix of regions, expressing whether they are direct neighbors (value 1L) or not (value 0L).

cluster_location_details	<i>Get details on cluster locations.</i>
--------------------------	--

Description

Get details on cluster locations.

Usage

```
cluster_location_details(
  clusterinfo,
  silent = getOption("brainloc.silent", default = FALSE),
  ...
)
```

Arguments

clusterinfo	a clusterinfo instance, see the clusterinfo function on how to get one. Must contain a valid brainparc in field 'brainparc'.
silent	logical, whether to suppress console messages.
...	passed on to cluster_extrema .

Value

a `data.frame` with cluster location details, including MNI152 and Talairach coordinates, Talairach Daemon labels (5 hierarchy levels + full label), and the atlas region name from the first atlas in the `brainparc`. The column names should be self-explanatory.

cluster_peaks	<i>Compute cluster peaks.</i>
---------------	-------------------------------

Description

A peak is a position in a cluster (a vertex) which is assigned the maximum statsmap value over its neighborhood. The neighborhood definition used by this function is the 1-ring neighborhood on the mesh.

Usage

```
cluster_peaks(
  clusterinfo,
  type = "extreme",
  silent = getOption("brainloc.silent", default = FALSE),
  ...
)
```

Arguments

<code>clusterinfo</code>	a <code>clusterinfo</code> instance, see the <code>clusterinfo</code> function to get one.
<code>type</code>	character string, one of 'extreme', 'min' or 'max'. Which cluster value to report. The default of 'extreme' reports the absolutely larger one of the min and the max value and is for the typical use case of t-value maps.
<code>silent</code>	whether to suppress printing messages to stdout.
<code>...</code>	passed on to <code>clusteroverlay_to_annot</code> .

Value

a `data.frame` with cluster peak information. The column names should be self-explanatory.

cluster_region_overlap

Compute overlapping atlas regions for clusters.

Description

For each cluster in the clusterinfo, compute 1) which atlas regions it overlaps with, and to what percentage the cluster is contained in each region. And, 2), how much of each of the overlapping regions are covered by the cluster. The atlases or surface parcellations to compare the clusters to are taken from the clusterinfo\$brainparc field.

Usage

```
cluster_region_overlap(
  clusterinfo,
  silent = getOption("brainloc.silent", default = FALSE)
)
```

Arguments

clusterinfo	a clusterinfo instance, see the clusterinfo function to get one.
silent	whether to suppress printing messages to stdout.

Value

a data.frame with cluster overlap information.

clusterinfo

Create clusterinfo data structure from files or pre-loaded data.

Description

Create clusterinfo data structure from files or pre-loaded data.

Usage

```
clusterinfo(
  lh_overlay,
  rh_overlay,
  lh_statmap,
  rh_statmap,
  template_subject = "fsaverage",
  subjects_dir = file.path(getOption("brainloc.fs_home", default =
    Sys.getenv("FREESURFER_HOME")), "subjects")
)
```

Arguments

lh_overlay	integer vector, the cluster overlay data: one integer per vertex of the left hemisphere. This assigns each vertex to a cluster, and all vertices of one cluster have the same number. Background is typically 1L. If a character string, the parameter will be interpreted as a file path and loaded with <code>freesurferformats::read.fs.morph</code> .
rh_overlay	integer vector, just like <code>lh_overlay</code> , but for the right hemisphere.
lh_statmap	double vector, the stats map. Typically a t-value map. Must have one value per vertex. If a character string, the parameter will be interpreted as file path and loaded with <code>freesurferformats::read.fs.morph</code> .
rh_statmap	double vector, just like <code>lh_statmap</code> , but for the right hemisphere.
template_subject	character string, the template subject name. Typically 'fsaverage' or 'fsaverage6'. Must be a subject in MNI305 space (i.e., the standard FreeSurfer template space). **Important:** The overlay and statmap data passed to this function must be in the same space as the <code>template_subject</code> (usually MNI305 / fsaverage space). Passing data from a different space (e.g., native subject space) will produce silently incorrect coordinate transforms and atlas region assignments.
subjects_dir	character string, file system path to a directory containing the recon-all data for the <code>template_subject</code> . Used to load surfaces and annotations to identify cluster coordinates and atlas regions.

Value

named list with entries 'overlay', 'statmap', and 'metadata': a clusterinfo data structure. Each of the 'overlay' and 'statmap' keys holds a [hemilist](#) of numerical vectors.

See Also

[clusterinfo_from_thresholded_overlay](#) If you do have a thresholded t-map instead of one t-map and one cluster overlay map.

clusterinfo_from_thresholded_overlay

Create clusterinfo instance from thresholded per-vertex data overlays.

Description

Create clusterinfo instance from thresholded per-vertex data overlays.

Usage

```
clusterinfo_from_thresholded_overlay(
  lh_threshmap,
  rh_threshmap,
  value_thresholded = 0,
```

```

template_subject = "fsaverage",
subjects_dir = file.path(getOption("brainloc.fs_home", default =
  Sys.getenv("FREESURFER_HOME")), "subjects")
)

```

Arguments

lh_threshmap double vector, the stats map. Typically a thresholded t-value map. Must have one value per vertex. The value assigned to vertices that have been removed by the thresholding can be set with parameter 'value_thresholded'. If a character string, the parameter will be interpreted as file path and loaded with `freesurferformats::read.fs.morph`.

rh_threshmap like `lh_threshmap`, but for the right hemisphere.

value_thresholded scalar double, the data value of thresholded vertices in the `lh_threshmap` and `rh_threshmap`.

template_subject character string, the template subject name. Typically 'fsaverage' or 'fsaverage6'. Must be a subject in MNI305 space (i.e., the standard FreeSurfer template space). ****Important:**** The overlay and statmap data passed to this function must be in the same space as the `template_subject` (usually MNI305 / fsaverage space). Passing data from a different space (e.g., native subject space) will produce silently incorrect coordinate transforms and atlas region assignments.

subjects_dir character string, file system path to a directory containing the recon-all data for the `template_subject`. Used to load surfaces and annotations to identify cluster coordinates and atlas regions.

Value

a `clusterinfo` instance.

Note

This can only work if there are no clusters which touch each other. I think this can never happen, but we should double-check.

See Also

[clusterinfo](#) If you do have a separate t-map and a cluster overlay map instead of only a thresholded t-map.

`coord_closest_regions` *Find closest regions to coordinate using Euclidean or geodesic distance.*

Description

Finds the closest atlas regions according to the brain surface parcellation 'brainparc' for the given query vertex coordinates. The coordinates must be in the surface space of the surface included in the 'brainparc'.

Usage

```
coord_closest_regions(
  brainparc,
  coordinate,
  linkage = "single",
  distance = "euclidean",
  silent = getOption("brainloc.silent", default = FALSE),
  num_regions_to_report = 5L
)
```

Arguments

brainparc	a brain parcellation, see functions like brainparc_fs to get one.
coordinate	nx3 numerical matrix or vector of length 3, the query point coordinates.
linkage	character string, one of "single" or "centroid". Defines how the distance from a vertex to a region of vertices is computed. "single": Euclidean distance from query vertex to the closest vertex of the atlas region. "centroid": Euclidean distance from query vertex to the mean of the vertex coordinates of the atlas region.
distance	character string, one of "euclidean" or "geodesic". The latter is only supported with linkage = 'single'.
silent	logical, whether to suppress console messages.
num_regions_to_report	scalar integer, the number of closest regions to report (will be reported in increasing order, by distance). Pass NULL to get all regions.

Note

This is a wrapper around [vertex_closest_regions](#). It first computes the surface vertex closest to the given coordinate (using [coord_closest_vertex](#)), then runs [vertex_closest_regions](#) with that vertices coordinates.

See Also

[vertex_closest_regions](#) is faster if you have a vertex index for the surface in 'brainparc' instead of a coordinate.

Examples

```
## Not run:
bp = brainparc_fs(fsbrain::fsaverage.path(), "fsaverage", atlas="aparc");
coord_closest_regions(bp, bp$surfaces$white$1h$vertices[1:3,]);
```

```
## End(Not run)
```

coord_closest_vertex *Find closest surface vertex to a point, using Euclidean distance.*

Description

Find closest surface vertex to a point, using Euclidean distance.

Usage

```
coord_closest_vertex(coordinate, surfaces)
```

Arguments

coordinate nx3 numerical matrix or vector of length 3, the query point coordinates.
surfaces [hemilist](#) of fs.surface instances

Value

a data.frame with columns named 'query_x', 'query_y', 'query_z', 'lh_closest_vertex', 'lh_distance', 'rh_closest_vertex', 'rh_distance', 'both_closest_vertex', 'both_distance', 'both_hemi'.

Examples

```
## Not run:
bp = brainparc_fs(fsbrain::fsaverage.path(), "fsaverage", atlas="aparc");
query_coords = matrix(seq.int(9), ncol = 3, byrow = TRUE);
coord_closest_vertex(query_coords, bp$surfaces$white);

## End(Not run)
```

coord_MNI152_to_talairach

*Transform MNI152 coords to approximate Talairach space using
Matthew Brett's approach.*

Description

Transform MNI152 coordinates to Talairach space using the piecewise linear transform described by Matthew Brett at <http://brainmap.org/training/BrettTransform.html>. This is an **approximation** of Talairach space, not the original Talairach atlas space (which is defined by the AC-PC line and a bounding box from a single post-mortem brain). The transform applies different scaling factors above and below the AC line to account for differences between the MNI152 template brain and the Talairach atlas brain.

Usage

```
coord_MNI152_to_talairach(mni152_coords)
```

Arguments

mni152_coords nx3 numerical matrix of RAS coordinates in MNI152 space.

Value

nx3 numerical matrix of approximate Talairach coordinates.

Note

When using the output of this function with [get_talairach_label](#) to query the Talairach Daemon atlas (talairach.org), be aware that the Daemon is defined in true Talairach space while this transform produces approximate Talairach coordinates. In practice, the Talairach Daemon has fairly coarse labels (lobe-level), so small coordinate discrepancies usually do not affect the result. This approach is widely used in the field (SPM, FSL, etc.).

This implementation is published under the GPL license. See <https://github.com/scn/dipfit/blob/master/mni2tal.m> and <https://github.com/scn/dipfit/blob/master/mni2tal.m> for a Matlab implementation of the method. All credits go to Matthew Brett.

Examples

```
mni_coords = matrix(c(10, 12, 14), nrow = 1, ncol = 3, byrow = TRUE);
coord_MNI152_to_talairach(mni_coords);
```

coord_MNI305_info	<i>Get info on MNI305 surface coordinate in various coordinate systems.</i>
-------------------	---

Description

Get info on MNI305 surface coordinate in various coordinate systems.

Usage

```
coord_MNI305_info(coords_mni305, ...)
```

Arguments

coords_mni305 nx3 matrix of coordinates, typically from fsaverage surface vertices.
 ... Passed on to [coord_MNI305_to_MNI152](#).

Value

named list with coordinate information (matrices)

coord_MNI305_to_MNI152

Transform MNI305 coords (FreeSurfer fsaverage surface) to MNI152 coordinates.

Description

Transform MNI305 coords (FreeSurfer fsaverage surface) to MNI152 coordinates.

Usage

```
coord_MNI305_to_MNI152(
    vertex_coords,
    method = getOption("brainloc.method_MNI305_to_from_MNI152", default = "linear"),
    surface = "orig",
    fs_home = getOption("brainloc.fs_home", default = Sys.getenv("FREESURFER_HOME"))
)
```

Arguments

vertex_coords	nx3 numerical matrix of MNI305 surface RAS coordinates, typically from fsaverage surface vertices.
method	character string, the method to use to map from MNI305 to MNI152 along the way. One of "best_available", "regfusionr", and "linear".
surface	optional character string or hemilist of surfaces, the surface to use to find a surface vertex close to the given query coordinates. Only used if 'method' results in regfusionr being used. Passed on to regfusionr::mni305_coords_to_mni152_coords.
fs_home	optional character string, the path to the FREESURFER_HOME directory from which to load the surfaces from the 'surface' parameter. Only used if 'method' results in regfusionr being used. Passed on to regfusionr::mni305_coords_to_mni152_coords.

Value

nx3 numerical matrix of MNI152 coords.

Note

To verify MNI305→MNI152 results, you can click a vertex in FreeView to get its MNI305 coordinates, run them through this function, and compare the MNI152 output with the values shown by the BioImageSuite MNI→Talairach tool at <https://bioimagesuiteweb.github.io/bisweb-manual/tools/mni2tal.html>. Note that this external tool uses a different Talairach transform (Lacadie et al.) than the one used by [coord_MNI152_to_talairach](#) (Brett), so MNI152 values should match but Talairach values may differ slightly.

Examples

```
## Not run:
# Get MNI152 coordinates for first 3 fsaverage lh vertices:
surf = freesurferformats::read.fs.surface("/opt/freesurfer/subjects/fsaverage/surf/lh.white");
coord_MNI305_to_MNI152(surf$vertices[1:3, ]);

## End(Not run)
```

fs_home	<i>Return FreeSurfer path.</i>
---------	--------------------------------

Description

Return FreeSurfer path.

Usage

```
fs_home()
```

Value

the FreeSurfer path, typically what the environment variable ‘FREESURFER_HOME’ points to.

Note

This function will stop (i.e., raise an error) if the directory cannot be found. It calls [find.freesurferhome](#) internally, see there for more details.

See Also

Other FreeSurfer helper functions: [find.freesurferhome\(\)](#), [get_subjects_dir\(\)](#), [has_fs\(\)](#)

get_clusters	<i>Get the clusters from a clusterinfo instance.</i>
--------------	--

Description

Get the clusters from a clusterinfo instance.

Usage

```
get_clusters(clusterinfo, hemi = "both")
```

Arguments

clusterinfo	a clusterinfo instance.
hemi	character string, one of 'lh', 'rh' or 'both'. The hemisphere(s) for which to return the clusters.

Value

named list, the keys are the cluster names, and the values are integer vectors defining the member vertices.

get_subjects_dir	<i>Find subjects_dir or stop.</i>
------------------	-----------------------------------

Description

Try a number of ways to find the subjects dir, using environment variables and knowledge on common installation paths of FreeSurfer.

Usage

```
get_subjects_dir(
  mustWork = TRUE,
  allow_download = FALSE,
  accept_freesurfer_license = FALSE
)
```

Arguments

mustWork	whether to stop if no subjects_dir can be found.
allow_download	logical, whether to allow downloading the data in case it is not found locally.
accept_freesurfer_license	logical, whether you want to also download fsaverage and fsaverage3, and accept the FreeSurfer license for fsaverage and fsaverage3, available at https://surfer.nmr.mgh.harvard.edu/fs . Defaults to FALSE. If FALSE, nothing will be downloaded.

Value

character string, the subjects directory. If mustWork is FALSE, it will return NULL if no directory was found. If mustWork is TRUE and nothing is found, it stops.

See Also

Other FreeSurfer helper functions: [find.freesurferhome\(\)](#), [fs_home\(\)](#), [has_fs\(\)](#)

get_surface	<i>Extract the surface hemilist from a brainparc.</i>
-------------	---

Description

Extract the surface hemilist from a brainparc.

Usage

```
get_surface(brainparc)
```

Arguments

brainparc a brainparc instance, see [brainparc_fs](#) to get one.

Value

[hemilist](#) of fs.surface instances.

See Also

Other brainparc accessors: [get_surface_coords\(\)](#)

get_surface_coords	<i>Extract the vertex coordinates from the brainparc surface.</i>
--------------------	---

Description

Extract the vertex coordinates from the brainparc surface.

Usage

```
get_surface_coords(brainparc, vertices, hemis)
```

Arguments

brainparc	a brainparc instance, see brainparc_fs to get one.
vertices	integer vector, the query vertex indices.
hemis	vector of character string, the hemispheres for the query vertices. Length must match, length of 1 will be recycled to all vertices.

Value

numerical nx3 matrix, the coordinates for the x query vertices.

See Also

Other brainparc accessors: [get_surface\(\)](#)

get_talairach_label	<i>Retrieve Talairach labels for Talairach coordinates from talairach.org files.</i>
---------------------	--

Description

Retrieve Talairach labels for Talairach coordinates from talairach.org files.

Usage

```
get_talairach_label(
  tal_coords,
  talairach_vol_file = NULL,
  lookup_table_file = NULL,
  check_oob = TRUE
)
```

Arguments

tal_coords	nx3 numeric matrix, the n query Talairach coordinates for which you want to retrieve the labels. (Three columns, each row represents the x,y,z coordinates of a point in Talairach space.)
talairach_vol_file	optional character string, the path to the NIFTI Talairach label volume file from talairach.org. It is named talairach.nii on the website. You can also gzip it and use a much smaller .nii.gz version of the file. Leave at NULL to auto-download from talairach.org if needed.
lookup_table_file	optional character string, the path to the Talairach label index text file from talairach.org. It is named labels.txt on the website. Leave at NULL to auto-download from talairach.org if needed.
check_oob	logical, whether to check for out-of-bounds voxels, which do not fall into the NIFTI volume during the computation (because the talairach coordinate is a bit off). Checking is slower, but if such coordinates exist in your query and you do not check, the function will stop with an error. Leave this alone if in doubt.

Value

data.frame describing labels for the coordinates. The following columns are included: cx,cy,cz: the query coordinates, as given in parameter 'tal_coords'. v1,v2,v3: The voxel indices in the talairach.nii file that map to the query coordinates. label_lv1, ..., label_lv5: the label strings for the location, in a hierarchy with 5 levels (parsed from label_full for you as a convenience). label_full: The raw, full label string for the location. A star * means that no label name is available for the given coordinate at this level.

Note

If you leave the file path arguments `talairach_vol_file` and `lookup_table_file` at `NULL`, the files will be downloaded from `talairach.org` automatically if that is possible (e.g., you have a working internet connection and the server is up). If it fails, the function will stop in the next step.

When using this function or the Talairach.org data in your research, please cite the following two publications: Lancaster JL, Woldorff MG, Parsons LM, Liotti M, Freitas CS, Rainey L, Kochunov PV, Nickerson D, Mikiten SA, Fox PT, "Automated Talairach Atlas labels for functional brain mapping". Human Brain Mapping 10:120-131, 2000. and Lancaster JL, Rainey LH, Summerlin JL, Freitas CS, Fox PT, Evans AC, Toga AW, Mazziotta JC. Automated labeling of the human brain: A preliminary report on the development and evaluation of a forward-transform method. Hum Brain Mapp 5, 238-242, 1997.

Currently it is not possible to search 'in the vicinity' to compensate for a slightly inaccurate query coordinate that may be outside of the brain by just a tiny bit. Please open an issue if you feel this is required.

Examples

```
## Not run:
query_tal_coords = matrix(seq.int(15), nrow=5, ncol=3);
get_talairach_label(query_tal_coords);

## End(Not run)
```

has_fs

Check whether a FreeSurfer installation can be found on the system.

Description

Check whether a FreeSurfer installation can be found on the system.

Usage

```
has_fs()
```

Value

logical, whether a FreeSurfer installation was found.

See Also

Other FreeSurfer helper functions: [find.freesurferhome\(\)](#), [fs_home\(\)](#), [get_subjects_dir\(\)](#)

hemilist	Create hemilist.
----------	------------------

Description

A hemilist is a named list which has at least one of the following fields: lh and rh. The fields can store arbitrary values.

Usage

```
hemilist(lh_data = NULL, rh_data = NULL)
```

Arguments

lh_data	any R object, the lh value.
rh_data	any R object, the rh value.

Value

a hemilist

Note

This function is trivial and mainly exists to document what a hemilist is (so this explanation can be linked to from the documentation of other functions which use the concept).

See Also

Other hemilist functions: [hemilist.wrap\(\)](#), [is.hemilist\(\)](#)

is.brainparc	Check whether object is a brainparc instance.
--------------	---

Description

Check whether object is a brainparc instance.

Usage

```
is.brainparc(x)
```

Arguments

x	any 'R' object
---	----------------

Value

logical, TRUE if parameter `x` is a `brainparc` instance (that is, has "brainparc" in its classes) and FALSE otherwise.

<code>is.clusterinfo</code>	<i>Check whether <code>x</code> is a <code>clusterinfo</code> instance.</i>
-----------------------------	---

Description

Check whether `x` is a `clusterinfo` instance.

Usage

```
is.clusterinfo(x)
```

Arguments

`x` any R object

Value

logical, whether `x` is a `clusterinfo` instance.

<code>name_regions</code>	<i>Get names of regions in labeled volume from color lookup table (colorLUT).</i>
---------------------------	---

Description

Given a brain volume with integer labels as voxel values and a suitable `colorLUT`, extract the region names as a named list.

Usage

```
name_regions(
  volatlas,
  colorlut,
  ignore_not_in_lut = FALSE,
  warn_not_in_lut = FALSE
)
```

Arguments

volatlas	3d integer array, the brain volume segmentation. Can be an <code>fs.volume</code> instance. Typically something like 'subject/mri/aseg.mgz'.
colorlut	a color lookup table data.frame, as returned by <code>read.fs.colortable</code> . Can be NULL if you do not have any, in that case the region names will just be 'region_x', where 'x' is the integer region code in the volatlas.
ignore_not_in_lut	logical, whether to ignore regions that are not found in the colorLUT. If TRUE, the regions will not appear in the named output list. If FALSE, they will appear as 'region_x', where 'x' is the integer region code in the volatlas.
warn_not_in_lut	logical, whether to print a warning if some regions of the volume do not occur in the colorLUT.

Value

named list, the keys are character strings: the region names from the colorLUT, and the values are integers: the region codes from the volume.

print.brainparc	<i>Print description of a brainparc instance (S3).</i>
-----------------	--

Description

Print description of a brainparc instance (S3).

Usage

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

Arguments

x	brainparc instance, with class 'brainparc'.
...	further arguments passed to or from other methods.

Value

Called for side effects.

segmentation_centers *Find center of mass of regions in a brain volume segmentation.*

Description

Find center of mass of regions in a brain volume segmentation.

Usage

```
segmentation_centers(volatlas, named_regions = NULL, vox2ras = diag(4))
```

Arguments

volatlas	3d integer array, the brain volume segmentation. Can be an <code>fs.volume</code> instance. Typically something like 'subject/mri/aseg.mgz'.
named_regions	the regions to consider, a named list where the keys are character strings: the region names from the colorLUT, and the values are integers: the region codes from the volume. See name_regions to get one. Can be NULL, in which case all regions in the volatlas will be used, and the region names will appear as 'region_x', where 'x' is the integer region code in the volatlas.
vox2ras	4x4 numerical matrix, the voxel to RAS coordinate transformation matrix for the volume. If NULL and volatlas is an <code>fs.volume</code> instance, the function will try to extract it from the volume header. The default is the identity matrix, which will return voxel indices instead of RAS coordinates.

Value

data.frame with region names and the center of mass x, y and z coordinates for the regions.

Examples

```
## Not run:
if(has_fs()) {
  fsh = fs_home();
  lutfile = file.path(fsh, "FreeSurferColorLUT.txt");
  segfile = file.path(fsh, 'subjects', 'fsaverage', 'mri', 'aseg.mgz');
  named_regions = name_regions(segfile, lutfile);
  named_regions$Unknown = NULL; # Ignore an atlas region.
  sc = segmentation_centers(segfile, named_regions);
  regions_distance_matrix = dist(sc);
}

## End(Not run)
```

vertex_closest_regions

Find closest regions to vertex using Euclidean or geodesic distance.

Description

Finds the closest atlas regions according to the brain surface parcellation 'brainparc' for the given query vertices.

Usage

```
vertex_closest_regions(
  brainparc,
  vertices,
  hemis,
  linkage = "single",
  distance = "euclidean",
  silent = getOption("brainloc.silent", default = FALSE),
  num_regions_to_report = 5L
)
```

Arguments

brainparc	a brain parcellation, see functions like brainparc_fs to get one.
vertices	integer vector, the query vertex indices (from the surface in the brainparc).
hemis	character string vector, the hemispheres for each of the vertices. Allowed entries are "lh" and "rh", for the left and right brain hemisphere, respectively. Must have same length as the 'vertices' vector (or exactly length 1, in which case we assume that this is the hemi for ALL query vertices).
linkage	character string, one of "single" or "centroid". Defines how the distance from a vertex to a region of vertices is computed. "single": Euclidean distance from query vertex to the closest vertex of the atlas region. "centroid": Euclidean distance from query vertex to the mean of the vertex coordinates of the atlas region.
distance	character string, one of "euclidean" or "geodesic". The latter is only supported with linkage = 'single'.
silent	logical, whether to suppress console messages.
num_regions_to_report	scalar integer, the number of closest regions to report (will be reported in increasing order, by distance). Pass NULL to get all regions.

Value

a data.frame, the column names should be obvious.

See Also

[coord_closest_regions](#) if you have a coordinate (on or near the surface) instead of a vertex.

Examples

```
## Not run:
bp = brainparc_fs(get_subjects_dir(), "fsaverage", atlas="aparc");
vertex_closest_regions(bp, vertices=c(10, 20), hemis=c("lh", "rh"));

## End(Not run)
```

vertex_regions	<i>Find regions for vertices in all brainparc atlases.</i>
----------------	--

Description

Find regions for vertices in all brainparc atlases.

Usage

```
vertex_regions(brainparc, vertices, hemis)
```

Arguments

brainparc	a brain parcellation, see functions like brainparc_fs to get one.
vertices	integer vector, the query vertex indices (from the surface in the brainparc).
hemis	character string vector, the hemispheres for each of the vertices. Allowed entries are "lh" and "rh", for the left and right brain hemisphere, respectively. Must have same length as the 'vertices' vector (or exactly length 1, in which case we assume that this is the hemi for ALL query vertices).

Value

a data.frame, the column names should be obvious.

See Also

[vertex_closest_regions](#) to compute distance to all regions.

Examples

```
## Not run:
bp = brainparc_fs(get_subjects_dir(), "fsaverage", atlas="aparc");
vertex_region(bp, vertices=c(10, 20), hemis=c("lh", "rh"));

## End(Not run)
```

Index

* FreeSurfer helper functions

fs_home, [13](#)
get_subjects_dir, [14](#)
has_fs, [17](#)

* brainparc accessors

get_surface, [15](#)
get_surface_coords, [15](#)

* hemilist functions

hemilist, [18](#)

brainparc, [2](#)

brainparc_fs, [3](#), [9](#), [15](#), [22](#), [23](#)

brainparc_neighbors, [4](#)

cluster_extrema, [4](#)

cluster_location_details, [4](#)

cluster_peaks, [5](#)

cluster_region_overlap, [6](#)

clusterinfo, [6](#), [8](#)

clusterinfo_from_thresholded_overlay,
[7](#), [7](#)

clusteroverlay_to_annot, [5](#)

coord_closest_regions, [8](#), [23](#)

coord_closest_vertex, [9](#), [10](#)

coord_MNI152_to_talairach, [10](#), [12](#)

coord_MNI305_info, [11](#)

coord_MNI305_to_MNI152, [11](#), [12](#)

find.freesurferhome, [13](#)

find.freesurferhome(), [13](#), [14](#), [17](#)

fs_home, [13](#)

fs_home(), [14](#), [17](#)

get_clusters, [13](#)

get_subjects_dir, [14](#)

get_subjects_dir(), [13](#), [17](#)

get_surface, [15](#)

get_surface(), [15](#)

get_surface_coords, [15](#)

get_surface_coords(), [15](#)

get_talairach_label, [11](#), [16](#)

has_fs, [17](#)

has_fs(), [13](#), [14](#)

hemilist, [7](#), [10](#), [12](#), [15](#), [18](#)

hemilist.wrap(), [18](#)

is.brainparc, [18](#)

is.clusterinfo, [19](#)

is.hemilist(), [18](#)

name_regions, [19](#), [21](#)

print.brainparc, [20](#)

read.fs.colortable, [20](#)

segmentation_centers, [21](#)

vertex_closest_regions, [9](#), [22](#), [23](#)

vertex_regions, [23](#)