Package ‘freesurfer’

December 8, 2020

Type Package
Title Wrapper Functions for 'Freesurfer'
Version 1.6.8
Description Wrapper functions that interface with 'Freesurfer'
<https://surfer.nmr.mgh.harvard.edu/>, a powerful and
commonly-used 'neuroimaging'
software, using system commands. The goal is to be able to interface with
'Freesurfer' completely in R, where you pass R objects of class 'nifti',
implemented by package 'oro.nifti', and the function executes an 'Freesurfer'
command and returns an R object of class 'nifti' or necessary output.
LazyData true
LazyLoad true
Imports methods, neurobase, tools, R.utils, reshape2, utils
Depends R (>= 3.2.0)
License GPL-3
Suggests magrittr, knitr, oro.nifti (>= 0.7), rgl, rmarkdown, pander,
    fslr (>= 2.9.2)
BugReports https://github.com/muschelli2/freesurfer/issues
SystemRequirements FSL, Freesurfer
    (https://surfer.nmr.mgh.harvard.edu/)
RoxygenNote 7.1.1
VignetteBuilder knitr
Encoding UTF-8
NeedsCompilation no
Author John Muschelli [aut, cre] (<https://orcid.org/0000-0001-6469-1750>),
    Athanasia Mo Mowinckel [ctb] (<https://orcid.org/0000-0002-5756-0223>)
Maintainer John Muschelli <muschelli2@gmail.com>
Repository CRAN
Date/Publication 2020-12-08 20:30:07 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>aparcstats2table</td>
<td>3</td>
</tr>
<tr>
<td>aparcstats2table.help</td>
<td>4</td>
</tr>
<tr>
<td>aparcs_to_bg</td>
<td>5</td>
</tr>
<tr>
<td>asegsstats2table</td>
<td>5</td>
</tr>
<tr>
<td>asegsstats2table.help</td>
<td>6</td>
</tr>
<tr>
<td>checkmnc-methods</td>
<td>7</td>
</tr>
<tr>
<td>check_fs_result</td>
<td>7</td>
</tr>
<tr>
<td>construct_subj_dir</td>
<td>8</td>
</tr>
<tr>
<td>convert_surface</td>
<td>9</td>
</tr>
<tr>
<td>freesurferdir</td>
<td>10</td>
</tr>
<tr>
<td>freesurfer_read3</td>
<td>10</td>
</tr>
<tr>
<td>freesurfer_read3_con</td>
<td>11</td>
</tr>
<tr>
<td>freesurfer_read_curv</td>
<td>12</td>
</tr>
<tr>
<td>freesurfer_read_surf</td>
<td>12</td>
</tr>
<tr>
<td>fs_cmd</td>
<td>13</td>
</tr>
<tr>
<td>fs_help</td>
<td>14</td>
</tr>
<tr>
<td>fs_imgext</td>
<td>15</td>
</tr>
<tr>
<td>fs_lut</td>
<td>15</td>
</tr>
<tr>
<td>fs_subj_dir</td>
<td>16</td>
</tr>
<tr>
<td>fs_version</td>
<td>16</td>
</tr>
<tr>
<td>get_fs</td>
<td>17</td>
</tr>
<tr>
<td>get_fs_output</td>
<td>17</td>
</tr>
<tr>
<td>have_fs</td>
<td>18</td>
</tr>
<tr>
<td>mnc2nii</td>
<td>18</td>
</tr>
<tr>
<td>mnc2nii.help</td>
<td>19</td>
</tr>
<tr>
<td>mris_convert</td>
<td>19</td>
</tr>
<tr>
<td>mris_convert.help</td>
<td>20</td>
</tr>
<tr>
<td>mris_convert_annot</td>
<td>21</td>
</tr>
<tr>
<td>mris_convert_curv</td>
<td>21</td>
</tr>
<tr>
<td>mris_convert_normals</td>
<td>22</td>
</tr>
<tr>
<td>mris_convert_vertex</td>
<td>23</td>
</tr>
<tr>
<td>mris_euler_number</td>
<td>24</td>
</tr>
<tr>
<td>mris_euler_number.help</td>
<td>24</td>
</tr>
<tr>
<td>mri_convert</td>
<td>25</td>
</tr>
<tr>
<td>mri_convert.help</td>
<td>25</td>
</tr>
<tr>
<td>mri_deface</td>
<td>26</td>
</tr>
<tr>
<td>mri_info</td>
<td>27</td>
</tr>
<tr>
<td>mri_info.help</td>
<td>27</td>
</tr>
<tr>
<td>mri_mask</td>
<td>28</td>
</tr>
<tr>
<td>mri_mask.help</td>
<td>28</td>
</tr>
<tr>
<td>mri_normalize</td>
<td>29</td>
</tr>
<tr>
<td>mri_normalize.help</td>
<td>29</td>
</tr>
<tr>
<td>mri_segment</td>
<td>30</td>
</tr>
<tr>
<td>mri_segment.help</td>
<td>31</td>
</tr>
<tr>
<td>mri_surf2surf</td>
<td>31</td>
</tr>
<tr>
<td>mri_surf2surf.help</td>
<td>32</td>
</tr>
</tbody>
</table>
Description

This function calls `aparcstats2table` to convert parcellation statistics to a table.

Usage

```r
aparcstats2table(
  subjects,
  outfile = NULL,
  hemi = c("lh", "rh"),
  measure = c("area", "volume", "thickness", "thicknessstd", "meancurv", "gauscurv",
              "foldind", "curvind"),
  sep = c("tab", "space", "comma", "semicolon"),
  parc = c("aparc", "aparc.a2009s"),
  skip = FALSE,
  subj_dir = NULL,
  opts = "",
  verbose = TRUE
)
```
Arguments

subjects (character) vector of subjects
outfile (character) output filename
hemi (character) hemisphere to run statistics
measure (character) measure to be calculated
sep (character) separator for the output file. This will be an attribute of outfile
parc (character) parcellation to compute on
skip (logical) if subject does not have parcellation, should the command skip that subject (TRUE) or error (FALSE)
subj_dir (character path) if a different subjects directory is to be used other than SUBJECTS_DIR from shell, it can be specified here. Use with care as if the command fail, it may not reset the SUBJECTS_DIR back correctly after the error
opts (character) additional options to aparcstats2table
verbose (logical) print diagnostic messages

Value

Character filename of output file, with the attribute of the separator

Examples

if (have_fs()) {
  fs_subj_dir()
  outfile = aparcstats2table(subjects = "bert",
                           hemi = "lh",
                           meas = "thickness")
}

aparcstats2table.help  Parcellation Stats to Table Help

Description

This calls Freesurfer's aparcstats2table help

Usage

aparcstats2table.help()

Value

Result of fs_help
aparcs_to_bg  
Convert Freesurfer aparcs Table to brainGraph

Description

Converts Freesurfer aparcs table to brainGraph naming convention, relying on `aparcstats2table`

Usage

```r
aparcs_to_bg(subjects, measure, ...)
```

Arguments

- `subjects`: subjects to analyze, passed to `aparcstats2table`
- `measure`: measure to be analyzed, passed to `aparcstats2table`
- `...`: additional arguments passed to `aparcstats2table`

Value

Long data.frame

Examples

```r
if (have_fs()) {
  fs_subj_dir()
  df = aparcs_to_bg(subjects = "bert", measure = "thickness")
  print(head(df))
}
```

asegstats2table  
Parcellation Stats to Table

Description

This function calls `asegstats2table` to convert parcellation statistics to a table

Usage

```r
asegstats2table(
  subjects = NULL,
  inputs = NULL,
  outfile = NULL,
  measure = c("volume", "mean", "std"),
  sep = c("tab", "space", "comma", "semicolon"),
  skip = FALSE,
  subj_dir = NULL,
)```
opts = "",
verbose = TRUE
)

Arguments

subjects (character) vector of subjects
inputs (character paths) vector of input filenames, e.g. aseg.stats.
outfile (character) output filename
measure (character) measure to be calculated
sep (character) separator for the output file. This will be an attribute of outfile
skip (logical) if subject does not have parcellation, should the command skip that subject (TRUE) or error (FALSE)
subj_dir (character path) if a different subjects directory is to be used other than SUBJECTS_DIR from shell, it can be specified here. Use with care as if the command fail, it may not reset the SUBJECTS_DIR back correctly after the error
opts (character) additional options to asegstats2table
verbose (logical) print diagnostic messages

Value

Character filename of output file, with the attribute of the separator

Examples

if (have_fs()) {
  outfile = asegstats2table(subjects = "bert",
               meas = "mean")
}

asegstats2table.help Parcellation Stats to Table Help

Description

This calls Freesurfer's asegstats2table help

Usage

asegstats2table.help()

Value

Result of fs_help
checkmnc-methods  Force object to filename with .mnc extension

Description

Ensures the output to be a character filename (or vector) from an input image or \texttt{nifti} to have .mnc extension and be converted to MNC when necessary.

Usage

\begin{verbatim}
checkmnc(file, ...)

## S4 method for signature 'nifti'
checkmnc(file, ...)

## S4 method for signature 'character'
checkmnc(file, ...)

## S4 method for signature 'list'
checkmnc(file, ...)

ensure_mnc(file, ...)
\end{verbatim}

Arguments

- \texttt{file} character or \texttt{nifti} object
- \texttt{...} options passed to \texttt{checkimg}

Value

Character filename of mnc image

Author(s)

John Muschelli <muschellig2@gmail.com>

\hline

check_fs_result  Check Freesurfer Result

Description

Checks the Freesurfer system command result and will stop or warning based on whether output files exist.
Usage

\[\text{check_fs_result}(\text{res}, \text{fe}_{\text{before}}, \text{fe}_{\text{after}})\]

Arguments

\begin{align*}
\text{res} & \quad \text{(numeric) Result from system command} \\
\text{fe}_{\text{before}} & \quad \text{(logical) did the output file exist before the command ran} \\
\text{fe}_{\text{after}} & \quad \text{(logical) did the output file exist after the command ran}
\end{align*}

construct_subj_dir \quad Construct Subject Directory

Description

This function copies files specified by the types of data, determined by the folder Freesurfer put them in, into a temporary directory for easier separation of data and different structuring of data.

Usage

\[
\text{construct_subj_dir}(
\quad \text{label} = \text{NULL}, \\
\quad \text{mri} = \text{NULL}, \\
\quad \text{stats} = \text{NULL}, \\
\quad \text{surf} = \text{NULL}, \\
\quad \text{touch} = \text{NULL}, \\
\quad \text{subj} = \text{NULL}, \\
\quad \text{subj}_\text{root}_\text{dir} = \text{tempdir()})
\]

Arguments

\begin{align*}
\text{label} & \quad \text{Files to copy to } \text{subj}_\text{root}_\text{dir}/\text{subj}/\text{label} \text{ folder} \\
\text{mri} & \quad \text{Files to copy to } \text{subj}_\text{root}_\text{dir}/\text{subj}/\text{mri} \text{ folder} \\
\text{stats} & \quad \text{Files to copy to } \text{subj}_\text{root}_\text{dir}/\text{subj}/\text{stats} \text{ folder} \\
\text{surf} & \quad \text{Files to copy to } \text{subj}_\text{root}_\text{dir}/\text{subj}/\text{surf} \text{ folder} \\
\text{touch} & \quad \text{Files to copy to } \text{subj}_\text{root}_\text{dir}/\text{subj}/\text{touch} \text{ folder} \\
\text{subj} & \quad \text{Name of subject to make folder for to use for Freesurfer functions. If NULL, a temporary id will be generated} \\
\text{subj}_\text{root}_\text{dir} & \quad \text{Directory to put folder with contents of subj}
\end{align*}

Value

List with the subject name, the SUBJECTS_DIR to use (the directory that contains the subject name), and the types of objects copied
convert_surface

Convert Freesurfer Surface

Description

Reads in a surface file from Freesurfer and separates into vertices and faces

Usage

convert_surface(infile, ...)

Arguments

infile
Input surface file

... additional arguments to pass to mris_convert

Value

List of 3 elements: a header indicating the number of vertices and faces, the vertices, and the faces

Note

This was adapted from the gist: https://gist.github.com/mm--/4a4fc7badacfad874102

Examples

if (have_fs()) {
  infile = file.path(fs_subj_dir(),
                  "bert", "surf", "rh.pial")
  res = convert_surface(infile = infile)
}
freesurferdir          Get Freesurfer’s Directory

Description

Finds the FREESURFER_HOME from system environment or getOption("freesurfer.path") for location of Freesurfer functions and returns it

Usage

freesurferdir()
freesurfer_dir()
fs_dir()

Value

Character path

Examples

if (have_fs()) {
  freesurferdir()
freesurfer_dir()
  fs_dir()
}

freesurfer_read3       Freesurfer Read 3 records

Description

Reads first 3 records of file and returns the rotated value, for checking for other functions.

Usage

freesurfer_read3(file)

Arguments

file    thickness file or anything in surf/ directory from Freesurfer subject

Value

Numeric
**Examples**

```c
if (have_fs()) {
    bert_dir = file.path(fs_subj_dir(), "bert", "surf")
    file = file.path(bert_dir, "lh.thickness")
    out = freesurfer_read3(file)
}
```

---

**freesurfer_read3_con  Freesurfer Read 3 records**

**Description**

Reads first 3 records from a connection and returns the rotated value, for checking for other functions.

**Usage**

```c
freesurfer_read3_con(fid)
```

**Arguments**

- **fid**
  - connection to a thickness file or anything in surf/ directory from Freesurfer subject

**Value**

Numeric

**Examples**

```c
if (have_fs()) {
    bert_dir = file.path(fs_subj_dir(), "bert", "surf")
    file = file.path(bert_dir, "lh.thickness")
    fid = file(file, open = "rb")
    out = freesurfer_read3_con(file)
}
```
**freesurfer_read_curv**  
*Read Freesurfer Curv file*

**Description**

Reads a Freesurfer curvature file according to the FREESURFER_HOME/matlab/read_curv.m file.

**Usage**

```matlab
freesurfer_read_curv(file)
```

**Arguments**

- **file**  
  file name of a curvature file

**Value**

Numeric vector

**Examples**

```matlab
if (have_fs()) {
    bert_dir = file.path(fs_subj_dir(), "bert", "surf")
    file = file.path(bert_dir, "lh.thickness")
    fid = file(file, open = "rb")
    out = freesurfer_read_curv(file)
}
```

---

**freesurfer_read_surf**  
*Read Freesurfer Surface file*

**Description**

Reads a Freesurfer Surface file from the surf/ directory from recon-all

**Usage**

```matlab
freesurfer_read_surf(file)
```

**Arguments**

- **file**  
  surface file (e.g. lh.inflated)

**Value**

List of length 2: vertices and faces are the elements
Examples

```r
if (have_fs()) {
    fname = file.path(fs_subj_dir(), "bert", "surf", "lh.inflated")
    out = freesurfer_read_surf(fname)
}
```

---

**fs_cmd**

*FS Command Wrapper*

---

**Description**

This function calls Freesurfer command passed to `func`

**Usage**

```r
fs_cmd(
    func,  # Freesurfer function
    file,  # image to be manipulated
    outfile = NULL,  # resultant image name (optional)
    retimg = TRUE,  # return image of class nifti
    reorient = FALSE,  # If retimg, should file be reoriented when read in? Passed to readnii.
    intern = FALSE,  # (logical) to be passed to system
    opts = "",  # operations to be passed to func
    verbose = TRUE,  # (logical) print out command before running
    samefile = FALSE,  # (logical) is the output the same file?
    opts_after_outfile = FALSE,  # (logical) should opts come after the outfile in the Freesurfer command?
    ...)
```

**Arguments**

- `func` (character) Freesurfer function
- `file` (character) image to be manipulated
- `outfile` (character) resultant image name (optional)
- `retimg` (logical) return image of class nifti
- `reorient` (logical) If retimg, should file be reoriented when read in? Passed to `readnii`
- `intern` (logical) to be passed to `system`
- `opts` (character) operations to be passed to `func`
- `verbose` (logical) print out command before running
- `samefile` (logical) is the output the same file?
- `opts_after_outfile` (logical) should opts come after the outfile in the Freesurfer command?
fs_help

Description

This function takes in the function and returns the help from Freesurfer for that function.

Usage

\[
\text{fs\_help(func\_name, help\_arg = \"--help\", extra\_args = \"\", ...)}
\]

Arguments

- **func_name**: Freesurfer function name
- **help.arg**: Argument to print help, usually \"--help\"
- **extra.args**: Extra arguments to be passed other than \"--help\"
- **...**: Additional arguments to get_fs

Value

Prints help output and returns output as character vector

Examples

\[
\text{if (have_fs()) \{} \\
\text{fs\_help(func\_name = \"mri\_watershed\")} \\
\text{\}}
\]
**fs_imgext**

*Determine extension of image based on FSLOUTPUTTYPE*

**Description**

Runs `get_fs_output()` to extract FSLOUTPUTTYPE and then gets corresponding extension (such as `.nii.gz`)

**Usage**

`fs_imgext()`

**Value**

Extension for output type

**Examples**

`fs_imgext()`

---

**fs_lut**  
*Freesurfer look up table (LUT)*

**Description**

A `data.frame` with the index, label, and RGBA (red, blue, green, alpha) specification for the segmentations

**Usage**

`fs_lut`

**Format**

An object of class `data.frame` with 1266 rows and 6 columns.
**fs_subj_dir**

*Determine Freesurfer Subjects Directory*

**Description**

Finds the SUBJECTS_DIR from system environment or `getOption("fs.subj_dir")` for subjects dir.

**Usage**

```r
fs_subj_dir()
```

**Value**

SUBJECTS_DIR, such as `\${FREESURFER_HOME}/subjects`

**Examples**

```r
if (have_fs()) {
  fs_subj_dir()
}
```

---

**fs_version**

*Find Freesurfer Version*

**Description**

Finds the Freesurfer version from `FREESURFER_HOME/build-stamp.txt`

**Usage**

```r
fs_version()
```

**Value**

If the version file does not exist, it will throw a warning, but it will return an empty string. Otherwise it will be a string of the version.

**Note**

This will use `fs_dir()` to get the directory of FREESURFER

**Examples**

```r
if (have_fs()) {
  fs_version()
}
```
**get_fs**

Create command declaring FREESURFER_HOME

---

**Description**

Finds the Freesurfer from system environment or getOption("freesurfer.path") for location of Freesurfer functions.

**Usage**

```r
get_fs(bin_app = c("bin", "mni/bin", ""))
```

**Arguments**

- **bin_app** Should bin be added to the freesurfer path? All executables are assumed to be in FREESURFER_HOME/bin/. If not, and bin_app = "", they will be assumed to be in FREESURFER_HOME/.

**Value**

NULL if Freesurfer in path, or bash code for setting up Freesurfer DIR

**Note**

This will use Sys.getenv("FREESURFER_HOME") before getOption("freesurfer.path"). If the directory is not found for Freesurfer in Sys.getenv("FreesurferDIR") and getOption("freesurfer.path"), it will try the default directory /usr/local/freesurfer.

**Examples**

```r
if (have_fs()) {
  get_fs()
}
```

---

**get_fs_output**

Determine Freesurfer output type

---

**Description**

Finds the FSF_OUTPUT_FORMAT from system environment or getOption("fs.outputtype") for output type (nii.gz, nii, ANALYZE, etc).

**Usage**

```r
get_fs_output()
```
**Value**

FSF_OUTPUT_FORMAT, such as nii.gz If none found, uses nii.gz as default

**Examples**

get_fs_output()

---

**have_fs**

*Logical check if Freesurfer is accessible*

**Description**

Uses get_fs to check if FreesurferDIR is accessible or the option freesurfer.path is set and returns logical

**Usage**

have_fs(..., check_license = FALSE)

**Arguments**

- `...` options to pass to `get_fs`
- `check_license` Should a license file be checked to exist?

**Value**

Logical TRUE is Freesurfer is accessible, FALSE if not

**Examples**

have_fs()

---

**mnc2nii**

*Convert MNC to NIfTI*

**Description**

This function calls mnc2nii to convert MNC files to NIfTI

**Usage**

mnc2nii(file, outfile = NULL)
### mnc2nii.help

**MNC to NIfTI Help**

#### Description

This calls Freesurfer's `mnc2nii` help

#### Usage

```r
mnc2nii.help()
```

#### Value

Result of `fs_help`

---

### mris_convert

**Use Freesurfer's MRIs Converter**

#### Description

This function call `mris_convert`, a general conversion program for converting between cortical surface file formats

#### Usage

```r
mris_convert(infile, outfile = NULL, ext = "*.asc", opts = "", verbose = TRUE)
```
Arguments

infile (character) file path for input file
outfile (character) output file path
ext (character) output file extension, default is set to .asc
opts (character) additional options to add to front of command
verbose (logical) print diagnostic messages

Value

Name of output file

Examples

if (have_fs()) {
    bert_surf_dir = file.path(fs_subj_dir(), "bert", "surf")
    asc_file = mris_convert(
        infile = file.path(bert_surf_dir, "lh.white")
    )
}

Description

This calls Freesurfer's mris_convert help

Usage

mris_convert.help()
mris_convert_annot  Convert Annotation file

Description
This function call `mris_convert`, using the `--annot` option

Usage
```
mris_convert_annot(annot, opts = "", ...)
```

Arguments
- `annot` (character) annotation or gifti label data
- `opts` (character) additional options to `mris_convert`
- `...` additional arguments to `mris_convert`

Value
Result of `mris_convert`

Examples
```
if (have_fs()) {
  bert_dir = file.path(fs_subj_dir(), "bert")
  gii_file = mris_convert_annot(
    infile = file.path(bert_dir, "surf", "lh.white"),
    annot = file.path(bert_dir, "label", "lh.aparc.annot"),
    ext = ".gii"
  )
  gii = mris_convert_annot(
    infile = file.path(bert_dir, "surf", "lh.white"),
    annot = gii_file,
    ext = ".gii"
  )
}
```

mris_convert_curv  Convert Curvature file

Description
This function call `mris_convert`, using the `-c` option

Usage
```
mris_convert_curv(curv, opts = "", ...)
```
**mris_convert_normals**

**Convert Surface to Surface normals**

**Description**

This function call `mris_convert`, using the `-n` option.

**Usage**

```
mris_convert_normals(opts = "", ...)  
```

**Arguments**

- `opts` (character) additional options to `mris_convert`
- `...` additional arguments to `mris_convert`

**Value**

Result of `mris_convert`
Examples

```r
if (have_fs()) {
  bert_dir = file.path(fs_subj_dir(), "bert")
  asc_file = mris_convert_normals(
    infile = file.path(bert_dir, "surf", "lh.white")
  )
  readLines(asc_file, n = 6)
}
```

---

**mris_convert_vertex**  
*Convert Surface to vertex file*

### Description

This function calls `mris_convert`, using the `-v` option.

### Usage

```r
mris_convert_vertex(opts = "", ...) 
```

### Arguments

- `opts` (character) additional options to `mris_convert`
- `...` additional arguments to `mris_convert`

### Value

Result of `mris_convert`

### Examples

```r
if (have_fs()) {
  bert_surf_dir = file.path(fs_subj_dir(), "bert", "surf")
  asc_file = mris_convert_vertex(
    infile = file.path(bert_surf_dir, "lh.white")
  )
  readLines(asc_file, n = 6)
}
```
mris_euler_number

MRI Euler Number

Description

This function calls mris_euler_number to calculate the Euler Number

Usage

mris_euler_number(file, outfile = NULL, opts = "")

Arguments

file (character) input filename
outfile (character) output filename
opts (character) additional options to mris_euler_number

Value

Result of system command

Examples

## Not run:
if (have_fs() && requireNamespace("oro.nifti", quietly = TRUE)) {
  img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
  res = mris_euler_number(img, outfile = tempfile(fileext = ".mgz"))
}
## End(Not run)

mris_euler_number.help

MRI Euler Number Help

Description

This calls Freesurfer's mris_euler_number help

Usage

mris_euler_number.help()

Value

Result of fs_help
mri_convert

Use Freesurfer's MRI Converter

Description
This function calls mri_convert to convert an image

Usage
mri_convert(file, outfile, opts = "")

Arguments
- file (character) input filename
- outfile (character) output filename
- opts (character) additional options to mri_convert

Value
Result of system command

Examples
if (have_fs() && requireNamespace("oro.nifti", quietly = TRUE)) {
  img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
  res = mri_convert(img, outfile = tempfile(fileext = ".mgz"))
}

mri_convert.help

MRI Normalize Help

Description
This calls Freesurfer's mri_convert help

Usage
mri_convert.help()

Value
Result of fs_help
Description

This calls Freesurfer’s mri_deface

Usage

mri_deface(file, brain_template = NULL, face_template = NULL, ...)

Arguments

- **file**: File to pass to mri_deface
- **brain_template**: gca brain template file to pass to mri_deface
- **face_template**: gca face template file to pass to mri_deface
- **...**: Additional arguments to pass to fs_cmd

Value

Result of fs_cmd, which type depends on arguments to...

Note

If brain_template or face_template is NULL, they will be downloaded.

Examples

```r
if (have_fs()){
  base_url = "https://surfer.nmr.mgh.harvard.edu/pub/dist/mri_deface"
  url = file.path(base_url, "sample_T1_input.mgz")
  x = tempfile(fileext = ".mgz")
  out = try({
    utils::download.file(url, destfile = x)
  })
  if (!inherits(out, "try-error")) {
    noface = mri_deface(x)
  } else {
    url = paste0(
      "https://raw.githubusercontent.com/muschellij2/kirby21.t1/master/",
      "inst/visit_1/113-01-T1.nii.gz")
    x = tempfile(fileext = ".nii.gz")
    out = try({
      utils::download.file(url, destfile = x)
    })
    noface = mri_deface(x)
  }
}```
mri_info

Description

This calls Freesurfer's mri_info

Usage

mri_info(file, ...)

Arguments

file
File to pass to mri_info

... Additional arguments to pass to fs_cmd

Value

Result of fs_cmd, which type depends on arguments to ...

Examples

if (have_fs() && requireNamespace("oro.nifti", quietly = TRUE)){
  img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
  mri_info(img)
}

mri_info.help

Description

This calls Freesurfer's mri_info help

Usage

mri_info.help()

Value

Result of fs_help
**mri_mask**  
*Use Freesurfer's MRI Mask*

**Description**

This function calls `mri_mask` to mask an image.

**Usage**

```r
mri_mask(file, mask, outfile = NULL, retimg = TRUE, opts = "", ...)
```

**Arguments**

- `file` (character) input filename
- `mask` (character) mask filename
- `outfile` (character) output filename
- `retimg` (logical) return image of class nifti
- `opts` (character) additional options to `mri_mask`
- `...` additional arguments passed to `fs_cmd`.

**Value**

Character or nifti depending on `retimg`.

**Examples**

```r
if (have_fs() && requireNamespace("oro.nifti", quietly = TRUE)) {
  img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
  mask = img > 1
  res = mri_mask(img, mask)
}
```

---

**mri_mask.help**  
*MRI Normalize Help*

**Description**

This calls Freesurfer's `mri_mask` help.

**Usage**

```r
mri_mask.help()
```

**Value**

Result of `fs_help`
mri_normalize

Use Freesurfer's MRI Normalize Algorithm

Description
This function calls mri_normalize to normalize the values of the image, with white matter voxels around 110.

Usage
mri_normalize(file, outfile = NULL, retimg = TRUE, opts = "", ...)

Arguments
file (character) input filename
outfile (character) output filename
retimg (logical) return image of class nifti
opts (character) additional options to mri_normalize
...
additional arguments passed to fs_cmd.

Value
Character or nifti depending on retimg

Examples
## Not run:
if (have_fs()){
  mri_normalize("/path/to/T1.nii.gz")
}
## End(Not run)

mri_normalize.help

MRI Normalize Help

Description
This calls Freesurfer's mri_normalize help

Usage
mri_normalize.help()

Value
Result of fs_help
**mri_segment**

*Use Freesurfer's MRI Segmentation Algorithm*

---

**Description**

This function calls `mri_segment` to segment tissues from an image.

**Usage**

```r
mri_segment(file, outfile = NULL, retimg = TRUE, opts = "", ...)  
```

**Arguments**

- `file` (character) input filename
- `outfile` (character) output filename
- `retimg` (logical) return image of class nifti
- `opts` (character) additional options to `mri_segment`
- `...` additional arguments passed to `fs_cmd`.

**Value**

Character or nifti depending on `retimg`.

**Note**

NOT COMPLETE

**Examples**

```r
## Not run:  
if (have_fs()){  
  mri_segment("/path/to/T1.nii.gz")
}
## End(Not run)
```
**mri_segment.help**  

---

**MRI Segment Help**

---

**Description**

This calls Freesurfer's `mri_segment help`

**Usage**

```r
mri_segment.help()
```

**Value**

Result of `fs_help`

---

**mri_surf2surf**  

---

**Use Freesurfer's `mri_surf2surf` function to resamples one cortical surface onto another**

---

**Description**

This function calls Freesurfer `mri_surf2surf` to resample one cortical surface onto another

**Usage**

```r
mri_surf2surf(
    subject = NULL,
    target_subject = NULL,
    trg_type = c("curv", "w", "mgh", "nii"),
    src_type = c("curv", "w"),
    outfile = NULL,
    hemi = c("lh", "rh"),
    sval = c("thickness"),
    subj_dir = NULL,
    opts = "",
    verbose = TRUE
)
```

**Arguments**

- **subject** (character) vector of subject name
- **target_subject** (character) vector of target subject name
- **trg_type** (character) target file type, can be curv, paint (w), mgh, or nii
- **src_type** (character) source file type, can be curv or paint (w)
mri_surf2surf.help

Freesurfer's mri_surf2surf Help

Description

This calls Freesurfer's mri_surf2surf help

Usage

mri_surf2surf.help()

Value

Result of fs_help
mri_watershed

Use Freesurfer's MRI Watershed Algorithm

Description

This function calls `mri_watershed` to extract a brain from an image, usually for skull stripping.

Usage

```r
mri_watershed(file, outfile = NULL, retimg = TRUE, opts = "", ...)  
```

Arguments

- `file` (character): input filename
- `outfile` (character): output filename
- `retimg` (logical): return image of class `nifti`
- `opts` (character): additional options to `mri_watershed`
- `...`: additional arguments passed to `fs_cmd`.

Value

Character or `nifti` depending on `retimg`

Examples

```r
## Not run:
if (have_fs()){
  mri_watershed("/path/to/T1.nii.gz")
}
## End(Not run)
```

mri_watershed.help

MRI Watershed Help

Description

This calls Freesurfer's `mri_watershed help`

Usage

```r
mri_watershed.help()
```

Value

Result of `fs_help`
nii2mnc \textit{Convert NIfTI to MNC}

\textbf{Description}

This function calls nii2mnc to convert NIfTI to MNC files

\textbf{Usage}

\begin{verbatim}
nii2mnc(file, outfile = NULL)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{file} (character) input filename
  \item \texttt{outfile} (character) output filename
\end{itemize}

\textbf{Value}

Character filename of output

\textbf{Examples}

\begin{verbatim}
if (have_fs() && requireNamespace("oro.nifti", quietly = TRUE)) {
  img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
  mnc = nii2mnc(img)
  img_file = mnc2nii(mnc)
}
\end{verbatim}

\textbf{nii2mnc.help \textit{Convert NIfTI to MNC Help}}

\textbf{Description}

This calls Freesurfer's mnc2nii help

\textbf{Usage}

\begin{verbatim}
nii2mnc.help()
\end{verbatim}

\textbf{Value}

Result of \texttt{fs_help}
**nu_correct**  
*Use Freesurfer's Non-Uniformity Correction*

**Description**

This function calls `nu_correct` to correct for non-uniformity.

**Usage**

\[
\text{nu_correct}(\text{file}, \text{mask} = \text{NULL}, \text{opts} = "", \text{verbose} = \text{TRUE}, \ldots)
\]

**Arguments**

- `file` (character) input filename
- `mask` (character or nifti) Mask to use for correction.
- `opts` (character) additional options to `mri_segment`
- `verbose` print diagnostic messages
- `...` additional arguments passed to `fs_cmd`.

**Value**

Object of class nifti depending on `retimg`.

**Examples**

```r
## Not run:
if (have_fs()){
  nu_correct("/path/to/T1.nii.gz")
}
## End(Not run)
```

---

**nu_correct.help**  
*Non-Uniformity Correction Help*

**Description**

This calls Freesurfer's `nu_correct` help.

**Usage**

\[
\text{nu_correct.help}()
\]

**Value**

Result of `fs_help`
**readmgz**

*Read MGH or MGZ File*

**Description**

This function calls `mri_convert` to convert MGH/MGZ files to NIfTI, then reads it in using `readnii`.

**Usage**

```
readmgz(file)
readmgh(file)
```

**Arguments**

- **file**
  
  (character) input filename

**Value**

Object of class `nifti`

---

**readmnc**

*Read MNC File*

**Description**

This function calls `mnc2nii` to convert MNC files to NIfTI, then reads it in using `readnii`.

**Usage**

```
readmnc(file)
```

**Arguments**

- **file**
  
  (character) input filename

**Value**

Object of class `nifti`
**read_annotation**  
*Read Freesurfer annotation file*

**Description**

Reads Freesurfer binary annotation files that contain information on vertex labels and colours for use in analyses and brain area lookups.

**Usage**

```r
read_annotation(path, verbose = TRUE)
```

**Arguments**

- `path`: path to annotation file, usually with extension `annot`
- `verbose`: logical.

**Details**

This function is heavily based on Freesurfer’s `read_annotation.m` Original Author: Bruce Fischl  
CVS Revision Info: $Author: greve $ $Date: 2014/02/25 19:54:10 $ $Revision: 1.10 $

**Value**

list of 3 with vertices, labels, and colortable

**Examples**

```r
if (have_fs()) {
  bert_dir = file.path(fs_subj_dir(), "bert")
  annot_file = file.path(bert_dir, "label", "lh.aparc.annot")
  res = read_annotation(annot_file)
}
```

---

**read_aseg_stats**  
*Read Anatomical Segmentation Statistics*

**Description**

Reads an `aseg.stats` file from an individual subject

**Usage**

```r
read_aseg_stats(file)
```
Arguments

file  
aseg.stats file from Freesurfer

Value

List of 2 data.frames, one with the global measures and one with the structure-specific measures.

Examples

```r
if (have_fs()) {
  file = file.path(fs_subj_dir(), "bert", "stats", "aseg.stats")
  out = read_aseg_stats(file)
}
```

---

read_fs_label  
Read Label File

Description

Reads an label file from an individual subject

Usage

```r
read_fs_label(file)
```

Arguments

file  
label file from Freesurfer

Value

data.frame with 5 columns:

- **vertex_num**: Vertex Number
- **r_coord**: Coordinate in RL direction
- **a_coord**: Coordinate in AP direction
- **s_coord**: Coordinate in SI direction
- **value**: Value of label (depends on file)

Examples

```r
if (have_fs()) {
  file = file.path(fs_subj_dir(), "bert", "label", "lh.BA1.label")
  if (!file.exists(file)) {
    file = file.path(fs_subj_dir(), "bert", "label", "lh.BA1_exvivo.label")
  }
  out = read_fs_label(file)
}
```
Description

This function reads output from a Freesurfer table command, e.g. `aparcstats2table`, `asegstats2table`.

Usage

```r
read_fs_table(file, sep = NULL, stringsAsFactors = FALSE, header = TRUE, ...)
```

Arguments

- `file` (character path) filename of text file
- `sep` separator to override attribute of file, to pass to `read.table`.
- `stringsAsFactors` (logical) passed to `read.table`.
- `header` Is there a header in the data
- `...` additional arguments to `read.table`

Value

data.frame from the file

Examples

```r
if (have_fs()) {
  outfile = aparcstats2table(subjects = "bert",
                            hemi = "lh",
                            meas = "thickness")
  df = read_fs_table(outfile)
  seg_outfile = asegstats2table(subjects = "bert", meas = "mean")
  df_seg = read_fs_table(seg_outfile)
}
```

## Not run:
### using the pipe
```r
if (requireNamespace("magrittr", quietly = TRUE)) {
  df_seg = asegstats2table(subjects = "bert", meas = "mean") %>%
               read_fs_table
}
```

## End(Not run)
recon

Reconstruction from Freesurfer

Description
Reconstruction from Freesurfer with most of the options implemented.

Usage
recon(
  infile,
  outdir = NULL,
  subjid,
  motioncor = TRUE,
  nuintensitycor = TRUE,
  talairach = TRUE,
  normalization = TRUE,
  skullstrip = TRUE,
  gcareg = TRUE,
  canorm = TRUE,
  careg = TRUE,
  rmneck = TRUE,
  skull_lta = TRUE,
  calabel = TRUE,
  normalization2 = TRUE,
  segmentation = TRUE,
  fill = TRUE,
  tessellate = TRUE,
  smooth1 = TRUE,
  inflate1 = TRUE,
  qsphere = TRUE,
  fix = TRUE,
  finalsurfs = TRUE,
  smooth2 = TRUE,
  inflate2 = TRUE,
  cortribbon = TRUE,
  sphere = TRUE,
  surfreg = TRUE,
  contrasurfreg = TRUE,
  avgcurv = TRUE,
  cortparc = TRUE,
  parcstats = TRUE,
  cortparc2 = TRUE,
  parcstats2 = TRUE,
  aparc2aseg = TRUE,
  verbose = TRUE,
  opts = ""
)
Arguments

infile  Input filename (dcm or nii)
outdir  Output directory
subjid  subject id
motioncor  When there are multiple source volumes, this step will correct for small motions between them and then average them together. The input are the volumes found in file(s) mri/orig/XXX.mgz. The output will be the volume mri/orig.mgz. If no runs are found, then it looks for a volume in mri/orig (or mri/orig.mgz). If that volume is there, then it is used in subsequent processes as if it was the motion corrected volume. If no volume is found, then the process exits with errors.
nuintensitycor  Non-parametric Non-uniform intensity Normalization (N3), corrects for intensity non-uniformity in MR data, making relatively few assumptions about the data. This runs the MINC tool 'nu_correct'. By default, four iterations of nu_correct are run. The flag '-nuiterations' specification of some other number of iterations.
talairach  computes the affine transform from the orig volume to the MNI305 atlas using the MINC program mritotal. Creates the files mri/transform/talairach.auto.xfm and talairach.xfm.
normalization  Performs intensity normalization of the orig volume and places the result in mri/T1.mgz
skullstrip  Removes the skull from mri/T1.mgz and stores the result in mri/brainmask.auto.mgz and mri/brainmask.mgz. Runs the mri_watershed program.
gcareg  Computes transform to align the mri/nu.mgz volume to the default GCA atlas found in FREESURFER_HOME/average. Creates the file mri/transforms/talairach.lta.
canorm  Further normalization, based on GCA model. Creates mri/norm.mgz.
careg  Computes a nonlinear transform to align with GCA atlas. Creates the file mri/transforms/talairach.m3z.
rmneck  The neck region is removed from the NU-corrected volume mri/nu.mgz. Makes use of transform computed from prior CA Register stage. Creates the file mri/nu_noneck.mgz.
skull_lta  Computes transform to align volume mri/nu_noneck.mgz with GCA volume possessing the skull. Creates the file mri/transforms/talairach_with_skull.lta.
calabel  Labels subcortical structures, based in GCA model. Creates the files mri/aseg.auto.mgz and mri/aseg.mgz.
normalization2  Performs a second (major) intensity correction using only the brain volume as the input (so that it has to be done after the skull strip). Intensity normalization works better when the skull has been removed. Creates a new brain.mgz volume. If -noaseg flag is used, then aseg.mgz is not used by mri_normalize.
segmentation  Attempts to separate white matter from everything else. The input is mri/brain.mgz, and the output is mri/wm.mgz. Uses intensity, neighborhood, and smoothness constraints. This is the volume that is edited when manually fixing defects. Calls mri_segment, mri_edit_wm_with_aseg, and mri_pretess. To keep previous edits, run with -keepwmmedits. If -noaseg is used, them mri_edit_wm_aseg is skipped.
fill
This creates the subcortical mass from which the orig surface is created. The mid brain is cut from the cerebrum, and the hemispheres are cut from each other. The left hemisphere is binarized to 255. The right hemisphere is binarized to 127. The input is mri/wm.mgz and the output is mri/filled.mgz. Calls mri_fill. If the cut fails, then seed points can be supplied (see -cc-crs, -pons-crs, -lh-crs, -rh-crs). The actual points used for the cutting planes in the corpus callosum and pons can be found in scripts/ponscc.cut.log. This is the last stage of volumetric processing. If -noaseg is used, then aseg.mgz is not used by mri_fill.

tessellate
This is the step where the orig surface (ie, surf/?h.orig.nofix) is created. The surface is created by covering the filled hemisphere with triangles. Runs mri_tessellate. The places where the points of the triangles meet are called vertices. Creates the file surf/?h.orig.nofix Note: the topology fixer will create the surface ?h.orig.

smooth1
Calls mris_smooth. Smooth1 is the step just after tessellation

inflate1
Inflation of the surf/?h.smoothwm(.nofix) surface to create surf/?h.inflated.

qsphere
Automatic topology fixing. It is a quasi-homeomorphic spherical transformation of the inflated surface designed to localize topological defects for the subsequent automatic topology fixer.

fix
Finds topological defects (ie, holes in a filled hemisphere) using surf/?h.qsphere.nofix, and changes the orig surface (surf/?h.orig.nofix) to remove the defects. Changes the number of vertices. All the defects will be removed, but the user should check the orig surface in the volume to make sure that it looks appropriate. Calls mris_fix_topology.

finalsurfs
Creates the ?h.white and ?h.pial surfaces as well as the thickness file (?h.thickness) and curvature file (?h.curv). The white surface is created by "nudging" the orig surface so that it closely follows the white-gray intensity gradient as found in the T1 volume. The pial surface is created by expanding the white surface so that it closely follows the gray-CSF intensity gradient as found in the T1 volume. Calls mris_make_surfaces.

smooth2
the step just after topology fixing.

inflate2
inflate2 is the step just after topology fixing

cortribbon
Creates binary volume masks of the cortical ribbon, ie, each voxel is either a 1 or 0 depending upon whether it falls in the ribbon or not. Saved as ?h.ribbon.mgz. Uses mgz regardless of whether the -mgz option is used.

sphere
Inflates the orig surface into a sphere while minimizing metric distortion. This step is necessary in order to register the surface to the spherical atlas. (also known as the spherical morph). Calls mris_sphere. Creates surf/?h.sphere.

surfreg
Registers the orig surface to the spherical atlas through surf/?h.sphere. The surfaces are first coarsely registered by aligning the large scale folding patterns found in ?h.sulc and then fine tuned using the small-scale patterns as in ?h.curv. Calls mris_register. Creates surf/?h.sphere.reg.

contrasurfreg
Same as ipsilateral but registers to the contralateral atlas. Creates lh.rh.sphere.reg and rh.lh.sphere.reg.

avgcurv
Resamples the average curvature from the atlas to that of the subject. Allows the user to display activity on the surface of an individual with the folding pattern (ie, anatomy) of a group. Calls mrisp_paint. Creates surf/?h.avg_curv.
reconner

**cortparc**
Assigns a neuroanatomical label to each location on the cortical surface. Incorporates both geometric information derived from the cortical model (sulcus and curvature), and neuroanatomical convention. Calls mris_ca_label. -cortparc creates label/?h.aparc.annot, and -cortparc2 creates /label/?h.aparc.a2005s.annot.

**parcstats**
Runs mris_anatomical_stats to create a summary table of cortical parcellation statistics for each structure, including 1. structure name 2. number of vertices 3. total surface area (mm2) 4. total gray matter volume (mm3) 5. average cortical thickness (mm) 6. standard error of cortical thickness (mm) 7. integrated rectified mean curvature 8. integrated rectified Gaussian curvature 9. folding index 10. intrinsic curvature index. For -parcstats, the file is saved in stats/?h.aparc.stats. For -parcstats2, the file is saved in stats/?h.aparc.a2005s.stats.

**cortparc2**
see cortparc argument

**parcstats2**
see cortparc2 argument

**aparc2aseg**
Maps the cortical labels from the automatic cortical parcellation (aparc) to the automatic segmentation volume (aseg). The result can be used as the aseg would.

**verbose**
print diagnostic messages

**opts**
Additional options

**Value**
Result of `system`

---

**reconner**
*Reconstruction Helper for recon from Freesurfer*

**Description**
Wrapper for the recon-all function in Freesurfer

**Usage**
```r
reconner(
  infile = NULL,
  outdir = NULL,
  subjid = NULL,
  verbose = TRUE,
  opts = "-all",
  force = FALSE
)
```
Arguments

infile  Input filename (dcm or nii)
outdir  Output directory
subj    subject id
verbose print diagnostic messages
opts    Additional options
force   Force running of the reconstruction

Value

Result of system

Note

If you set infile = NULL, then you can omit the -i flag in recon-all

---

recon_all  Reconstruction from Freesurfer for All Steps

Description

Reconstruction from Freesurfer for All Steps

Usage

recon_all(
  infile = NULL,
  outdir = NULL,
  subj = NULL,
  verbose = TRUE,
  opts = "-all",
  ...
)

Arguments

infile  Input filename (dcm or nii)
outdir  Output directory
subj    subject id
verbose print diagnostic messages
opts    Additional options
...    arguments passed to reconner
Value

Result of `system`

Note

If you would like to restart a `recon-all` run, change `opts` so that `opts = "-make all"`

---

**recon_con1**

*Reconstruction from Motion Correction to Skull Strip*

**Description**

Reconstruction from Freesurfer for Step 1-5 (Motion Correction to Skull Strip), which calls `autorecon1` in `recon-all`

**Usage**

```r
recon_con1(infile, outdir = NULL, subjid, verbose = TRUE)
autorecon1(infile, outdir = NULL, subjid, verbose = TRUE)
recon_con2(infile, outdir = NULL, subjid, verbose = TRUE)
autorecon2(infile, outdir = NULL, subjid, verbose = TRUE)
recon_con3(infile, outdir = NULL, subjid, verbose = TRUE)
autorecon3(infile, outdir = NULL, subjid, verbose = TRUE)
```

**Arguments**

- `infile`: Input filename (dcm or nii)
- `outdir`: Output directory
- `subjid`: subject id
- `verbose`: print diagnostic messages

**Value**

Result of `system`

**Note**

See https://surfer.nmr.mgh.harvard.edu/fswiki/recon-all for the steps of each `autorecon1-3`. If you set `infile = NULL`, then you can omit the `-i` flag in `recon-all`. 
run_check_fs_cmd  
*Run and Check a Freesurfer Command*

**Description**

Checks whether an output filename exists before a command has run, prints and runs the command, and then checks the output from the result.

**Usage**

```r
run_check_fs_cmd(cmd, outfile, verbose = TRUE)
```

**Arguments**

- `cmd`: Command to be run
- `outfile`: Output file to be produced
- `verbose`: print diagnostic messages

**Value**

Invisible NULL

**See Also**

- `check_fs_result`

---

set_fs_subj_dir  
*Set Freesurfer Subjects Directory*

**Description**

Sets the SUBJECTS_DIR variable in the system environment or `options("fs.subj_dir" = x)`

**Usage**

```r
set_fs_subj_dir(x = file.path(fs_dir(), "subjects"))
```

**Arguments**

- `x`: path to SUBJECTS_DIR defaults to `file.path(fs_dir(), "subjects")`
**surface_to_obj**  
Convert Freesurfer Surface to Wavefront OBJ

**Description**  
Reads in a surface file from Freesurfer and converts it to a Wavefront OBJ file

**Usage**  
surface_to_obj(infile, outfile = NULL, ...)

**Arguments**
- **infile**: Input surface file
- **outfile**: output Wavefront OBJ file. If NULL, a temporary file will be created
- **...**: additional arguments to pass to convert_surface

**Value**  
Character filename of output file

**Examples**
```r
if (have_fs()) {
infile = file.path(fs_subj_dir(),
   "bert", "surf", "rh.pial")
res = surface_to_obj(infile = infile)
}
```

**surface_to_triangles**  
Convert Freesurfer Surface to Triangles

**Description**  
Reads in a surface file from Freesurfer and converts it into triangles

**Usage**  
surface_to_triangles(infile, ...)

**Arguments**
- **infile**: Input surface file
- **...**: additional arguments to pass to convert_surface
Value

Matrix of triangles with the number of rows equal to the number of faces (not the triplets - total faces)

Examples

```r
if (have_fs()) {
  infile = file.path(fs_subj_dir(),
    "bert", "surf", "rh.pial")
  right_triangles = surface_to_triangles(infile = infile)
  infile = file.path(fs_subj_dir(),
    "bert", "surf", "lh.pial")
  left_triangles = surface_to_triangles(infile = infile)
  if (requireNamespace("rgl", quietly = TRUE)) {
    rgl::rgl.open()
    rgl::rgl.triangles(right_triangles,
      color = rainbow(nrow(right_triangles)))
    rgl::rgl.triangles(left_triangles,
      color = rainbow(nrow(left_triangles)))
  }
  infile = file.path(fs_subj_dir(),
    "bert", "surf", "rh.inflated")
  right_triangles = surface_to_triangles(infile = infile)
  infile = file.path(fs_subj_dir(),
    "bert", "surf", "lh.inflated")
  left_triangles = surface_to_triangles(infile = infile)
  if (requireNamespace("rgl", quietly = TRUE)) {
    rgl::rgl.open()
    rgl::rgl.triangles(left_triangles,
      color = rainbow(nrow(left_triangles)))
    rgl::rgl.triangles(right_triangles,
      color = rainbow(nrow(right_triangles)))
  }
}
```

---

**surf_convert Convert Surface Data to ASCII**

**Description**

This function calls `mri_convert` to convert a measure from surfaces to an ASCII file and reads it in.

**Usage**

```
surf_convert(file, outfile = NULL)
```
Tracker

Arguments

- file: (character) input filename of curvature measure
- outfile: (character) output filename (if wanted to be saved)

Value

data.frame

Examples

```r
if (have_fs()) {
  fname = file.path(fs_subj_dir(), "bert", "surf", "lh.thickness")
  out = surf_convert(fname)
}
```

Description

Wrapper for the trac-all function in Freesurfer

Usage

tracker(infile,outdir = NULL, subjid, verbose = TRUE, opts = "")

Arguments

- infile: Input filename (dcm or nii)
- outdir: Output directory
- subjid: subject id, if NULL, the basename of the infile will be used
- verbose: print diagnostic messages
- opts: Additional options

Value

Result of system
trac_all

Tract Reconstruction Helper for trac-all from Freesurfer for All Steps

Description

Wrapper for the trac-all function in Freesurfer for All Steps

Usage

trac_all(infile, outdir = NULL, subjid, verbose = TRUE, opts = "")

Arguments

- **infile**: Input filename (dcm or nii)
- **outdir**: Output directory
- **subjid**: subject id
- **verbose**: print diagnostic messages
- **opts**: Additional options

Value

Result of `system`

trac_prep

Tract Reconstruction for Each Step

Description

Reconstruction from Freesurfer for Preprocessing, Bedpost, and Path reconstruction

Usage

trac_prep(infile, outdir = NULL, subjid, verbose = TRUE)

trac_bedpost(infile, outdir = NULL, subjid, verbose = TRUE)

trac_path(infile, outdir = NULL, subjid, verbose = TRUE)

Arguments

- **infile**: Input filename (dcm or nii)
- **outdir**: Output directory
- **subjid**: subject id
- **verbose**: print diagnostic messages
Value
  Result of system
Index

* datasets
  fs_lut, 15

aparc_to_bg, 5
aparcstats2table, 3, 5
aparcstats2table.help, 4
asegstats2table, 5
asegstats2table.help, 6
autorecon1 (recon_con1), 45
autorecon2 (recon_con1), 45
autorecon3 (recon_con1), 45

check_fs_result, 7, 46
checking, 7
checkmnc (checkmnc-methods), 7
checkmnc, character-method
  (checkmnc-methods), 7
checkmnc, list-method
  (checkmnc-methods), 7
checkmnc, nifti-method
  (checkmnc-methods), 7
checkmnc-methods, 7
construct_subj_dir, 8
convert_surface, 9, 47

datasets
  get_fs, 14, 17, 18
  get_fs_output, 17
  have_fs, 18
  mnc2nii, 18
  mnc2nii.help, 19
  mri_convert, 25
  mri_convert.help, 25
  mri_deface, 26
  mri_info, 27
  mri_info.help, 27
  mri_mask, 28
  mri_mask.help, 28
  mri_normalize, 29
  mri_normalize.help, 29
  mri_segment, 30
  mri_segment.help, 31
  mri_surf2surf, 31
  mri_surf2surf.help, 32
  mri_watershed, 33
  mri_watershed.help, 33
  mris_convert, 9, 19, 21–23
  mris_convert.help, 20
  mris_convert_annot, 21
  mris_convert_curv, 21
  mris_convert_normals, 22
  mris_convert_vertex, 23
  mris_euler_number, 24
  mris_euler_number.help, 24

  nii2mnc, 34
  nii2mnc.help, 34
  nu_correct, 35
  nu_correct.help, 35

  read.table, 39
  read_annotation, 37
  read_aseg_stats, 37
  read_fs_stats, 38
INDEX

read_fs_table, 39
readmgh (readmgz), 36
readmgz, 36
readmnc, 36
readnii, 13, 36
recon, 40
recon_all, 44
recon_con1, 45
recon_con1, recon_con2, recon_con3
    (recon_con1), 45
recon_con2 (recon_con1), 45
recon_con3 (recon_con1), 45
reconner, 43, 44
run_check_fs_cmd, 46

set_fs_subj_dir, 46
surf_convert, 48
surface_to_obj, 47
surface_to_triangles, 47
system, 13, 14, 43–45, 49–51

trac_all, 50
trac_bedpost (trac_prep), 50
trac_path (trac_prep), 50
trac_prep, 50
tracker, 49