Package: freesurfer (via r-universe)

May 8, 2026

Type Package

Title Wrapper Functions for 'FreeSurfer'

Version 1.8.1.902

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.

License GPL-3

URL https://muschellij2.github.io/freesurfer/, https://github.com/muschellij2/freesurfer

BugReports https://github.com/muschellij2/freesurfer/issues

Depends R (>= 4.1)

Imports cli, lifecycle, methods, neurobase, R.utils, tools, stats, utils

Suggests covr, knitr, oro.nifti (>= 0.7), rgl, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

LazyLoad true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements FreeSurfer (https://freesurfer.net/)

Repository https://ggsegverse.r-universe.dev

Date/Publication 2026-04-08 19:04:47 UTC

RemoteUrl https://github.com/drmowinckels/freesurfer

RemoteRef refactor

RemoteSha 5b38518eba8ae67f54a7edb2ea9cd4ba731878e5

Contents

`aparcs_to_bg` *Convert Freesurfer aparcs Table to brainGraph*

Description

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

Usage

```
aparcs_to_bg(subjects, measure, clean_col_names = TRUE, ...)
```

Arguments

`subjects`	subjects to analyze, passed to `aparcstats2table`
`measure`	measure to be analyzed, passed to `aparcstats2table`
`clean_col_names`	logical indicating whether to clean column names
`...`	additional arguments passed to `aparcstats2table`

Value

Long `data.frame`

Examples

```
## Not run:
fs_subj_dir()
df = aparcs_to_bg(subjects = "bert", measure = "thickness")
print(head(df))

## End(Not run)
```

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 `nifti` to have `.mnc` extension and be converted to MNC when necessary

Usage

```
checkmnc(file, ...)

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

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

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

checkmnc(file, ...)
```

Arguments

`file`	character or `nifti` object
`...`	options passed to `checkimg`

Value

Character filename of mnc image

Author(s)

John Muschelli <muschellij2@gmail.com>

---

construct_subj_dir *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

```
construct_subj_dir(
  label = NULL,
  mri = NULL,
  stats = NULL,
  surf = NULL,
  touch = NULL,
  subj = NULL,
  subj_root_dir = tempdir(check = TRUE)
)
```

Arguments

label	Files to copy to subj_root_dir/subj/label folder
mri	Files to copy to subj_root_dir/subj/mri folder
stats	Files to copy to subj_root_dir/subj/stats folder
surf	Files to copy to subj_root_dir/subj/surf folder
touch	Files to copy to subj_root_dir/subj/touch folder
subj	Name of subject to make folder for to use for Freesurfer functions. If NULL, a temporary id will be generated
subj_root_dir	Directory to put folder with contents of subj

Value

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

Examples

```
## Not run:
library(freesurfer)
label = "/Applications/freesurfer/subjects/bert/label/aparc.annot.a2009s.ctab"
mri = c(
  "/Applications/freesurfer/subjects/bert/mri/aparc.a2009s+aseg.mgz",
  "/Applications/freesurfer/subjects/bert/mri/aseg.auto.mgz")
stats = c("/Applications/freesurfer/subjects/bert/stats/lh.aparc.stats",
          "/Applications/freesurfer/subjects/bert/stats/aseg.stats")
surf = "/Applications/freesurfer/subjects/bert/surf/lh.thickness"
construct_subj_dir(
  label = label,
  mri = mri,
  stats = stats,
  surf = surf)

## End(Not run)
```

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

```
## Not run:
infile = file.path(fs_subj_dir(),
                   "bert", "surf", "rh.pial")
res = convert_surface(infile = infile)

## End(Not run)
```

freesurfer_read_curv *Read Freesufer Curv file*

Description

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

Usage

```
freesurfer_read_curv(file)
```

Arguments

file	file name of a curvature file

Value

Numeric vector

Examples

```
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

```
freesurfer_read_surf(file, verbose = get_fs_verbosity())
```

Arguments

file	surface file (e.g. lh.inflated)
verbose	(logical) print diagnostic messages

Value

List of length 2: vertices and faces are the elements

Examples

```
fname = file.path(fs_subj_dir(), "bert", "surf", "lh.inflated")
out = freesurfer_read_surf(fname)
```

fs_cmd *Execute FreeSurfer Commands from R*

Description

fs_cmd() is the core wrapper function that executes FreeSurfer command-line tools from within R. It handles file path validation, command construction, execution, and optional return of neuroimaging data as nifti objects.

Usage

```
fs_cmd(
  func,
  file,
  outfile = NULL,
  retimg = TRUE,
  reorient = FALSE,
  verbose = get_fs_verbosity(),
  intern = verbose,
  opts = "",
  opts_after_outfile = FALSE,
  frontopts = "",
  bin_app = "bin",
  timeout_seconds = NULL,
  validate_inputs = TRUE,
  ...
)
```

Arguments

func	Character string specifying the FreeSurfer command to execute (e.g., "mri_convert", "mri_watershed").
file	Character path to the input image file, or a nifti object. If a nifti object is provided, it will be temporarily written to disk.
outfile	Character path to the output image file. If NULL (default), and retimg = TRUE, a temporary file will be created. Set outfile = file to overwrite the input file (with warning).
retimg	Logical; if TRUE (default), returns output as a nifti object. If FALSE, returns the system command result.
reorient	Logical; if TRUE and retimg = TRUE, reorients the image when loading. Passed to neurobase::readnii(). Default is FALSE.
verbose	(logical) print diagnostic messages
intern	Logical; if TRUE, captures command output. Passed to base::system(). Default is FALSE.
opts	Character. Additional options to Freesurfer function.

`opts_after_outfile`
: Logical; if `TRUE`, places `opts` after `outfile` in the command string. Default is `FALSE`.

`frontopts`
: Character string of options to prepend before the input file in the command. Default is `""`.

`bin_app`
: Character string specifying the FreeSurfer bin directory appendix. Options are `"bin"` (default) or `"mni/bin"`.

`timeout_seconds`
: Numeric; command timeout in seconds. If specified, requires the **R.utils** package. Default is `NULL` (no timeout).

`validate_inputs`
: Logical; if `TRUE` (default), validates that input files exist before running the command.

`...`
: Additional arguments passed to `base::system()`.

Details

This function provides a unified interface to FreeSurfer's command-line tools, with several key features:

- Automatic validation of input/output files
- Support for nifti objects as inputs
- Optional timeout for long-running commands
- Flexible command option placement
- Automatic result verification
- Integration with FreeSurfer environment setup

Overwriting Files:

To overwrite the input file, set `outfile = file`. A warning will be displayed for safety. The function checks file existence before and after command execution to verify successful completion.

Command Construction:

The command is constructed in the following order:

1. FreeSurfer environment setup (from `get_fs()`)
2. Command name (`func`)
3. Front options (`frontopts`)
4. Input file path (quoted)
5. Output file path (quoted, if provided)
6. Additional options (`opts`)

Use `opts_after_outfile = TRUE` to place `opts` after the output file.

Value

If `retimg = TRUE`: A nifti object containing the output image, or `NULL` if output creation failed. If `retimg = FALSE`: The result from `base::system()`, typically the exit status or captured output.

See Also

get_fs() for FreeSurfer environment setup, mri_convert() for format conversion, neurobase::readnii() for reading nifti files

Examples

```
## Not run:
# Basic usage: convert MGZ to NIfTI
fs_cmd(
  func = "mri_convert",
  file = "input.mgz",
  outfile = "output.nii.gz",
  opts = "--conform"
)

# Return as nifti object
img <- fs_cmd(
  func = "mri_convert",
  file = "input.mgz",
  retimg = TRUE
)

# Use nifti object as input
library(oro.nifti)
img <- nifti(array(rnorm(10*10*10), dim = c(10, 10, 10)))
result <- fs_cmd(
  func = "mri_convert",
  file = img,
  outfile = "output.nii.gz"
)

## End(Not run)
```

---

fs_dir *Get FreeSurfer Directory Paths*

Description

Functions to retrieve FreeSurfer's installation and subjects directories. Multiple aliases are provided for convenience and backward compatibility.

Usage

```
freesurferdir()

freesurfer_dir()

fs_dir()
```

```
fs_subj_dir()
```

Value

Character; path to the FreeSurfer home or subjects directory.

Functions

- `freesurferdir()`: Get FreeSurfer installation directory
- `freesurfer_dir()`: Alias for freesurferdir
- `fs_dir()`: Short alias for freesurferdir
- `fs_subj_dir()`: Get FreeSurfer subjects directory

See Also

`get_fs_home()` for more detailed information, `have_fs()` to check if FreeSurfer is accessible

Examples

```
# All return the same value
freesurferdir()
freesurfer_dir()
fs_dir()

# Get subjects directory
fs_subj_dir()
```

---

fs_help *Wrapper for getting FreeSurfer help*

Description

This function takes in the function and returns the help from FreeSurfer for that function with simple validation.

Usage

```
fs_help(
  func_name,
  help_arg = "--help",
  extra_args = "",
  timeout_seconds = 30,
  display = TRUE,
  warn = TRUE,
  bin_app = "bin",
  ...
)
```

Arguments

`func_name`	FreeSurfer function name
`help_arg`	Argument to print help, usually "–help"
`extra_args`	Extra arguments to be passed other than `--help`
`timeout_seconds`	Timeout for help command (default 30 seconds)
`display`	Logical; whether to display help output
`warn`	Logical; whether to warn if help is not available
`bin_app`	Character string specifying the FreeSurfer bin directory appendix. Options are `"bin"` (default) or `"mni/bin"` for MNI tools.
`...`	additional arguments to `get_fs`

Value

Prints help output and returns output as character vector

Examples

```
fs_help("mri_watershed")

# For MNI tools
fs_help("nu_correct", help_arg = "-help", bin_app = "mni/bin")
```

`fs_imgext` *Determine extension of image based on Freesurfer output type*

Description

Runs `get_fs_output()` to extract the Freesurfer output type and then gets the corresponding file extension (such as `.nii.gz`).

Usage

```
fs_imgext()
```

Value

Character string representing the file extension for the 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_sitrep *FreeSurfer Situation Report*

Description

Get a situation report on your current FreeSurfer installation and settings within R. This function helps diagnose problems by showing how FreeSurfer paths and options are determined.

Usage

```
fs_sitrep(clear_cache = FALSE, test_commands = TRUE)
```

Arguments

`clear_cache`	(logical) Whether to clear settings cache before reporting
`test_commands`	(logical) Whether to test actual FreeSurfer commands

Examples

```
# Report all FreeSurfer settings
fs_sitrep()
```

fs_version *Find Freesurfer Version*

Description

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

Usage

```
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

```
fs_version()
```

get_fs *Generate FreeSurfer Command Line Environment Setup*

Description

This function generates a bash command string to set up the environment for using FreeSurfer. It ensures the required FreeSurfer installation, license, and environment setup files are validated and included in the command string. The function handles different FreeSurfer binaries like `bin`, `mni/bin`, and others, while ensuring proper initialization of the MNI environment if required.

Usage

```
get_fs(bin_app = c("bin", "mni/bin"), fs_home = get_fs_home(simplify = FALSE))
```

Arguments

`bin_app`	character A vector of options for the binary application directory. Possible options include: • `"bin"`: Default FreeSurfer binary directory. • `"mni/bin"`: Includes MNI initialization. • `""`: Base directory with no specific subdirectories.
`fs_home`	Character string specifying FreeSurfer installation directory. Usually this is determined automatically via `get_fs_home`.

Value

character A bash command string that includes environment setup for FreeSurfer. If the FreeSurfer environment or required configurations cannot be initialized, the function throws an error or issues a warning. On success, the returned string can be used directly in shell operations to load the FreeSurfer environment.

See Also

get_fs_home(), get_fs_license(), get_fs_output()

Examples

```
# Generate a shell command to set up FreeSurfer with the default `bin`
get_fs(bin_app = "bin")

# Generate a shell command to include MNI environment setup
get_fs(bin_app = "mni/bin")
```

---

get_fs_setting *Retrieve FreeSurfer Configuration Settings*

Description

These functions retrieve FreeSurfer configuration settings using a hierarchical lookup system: R options → environment variables → defaults.

Usage

```
get_fs_setting(env_var, opt_var, defaults = NULL, is_path = TRUE)

get_fs_home(simplify = TRUE)

get_fs_license(fs_home = get_fs_home(), simplify = TRUE)

get_fs_subdir(fs_home = get_fs_home(), simplify = TRUE)

get_fs_source(fs_home = get_fs_home(), simplify = TRUE)

get_fs_verbosity(simplify = TRUE)

get_fs_output(simplify = TRUE)

get_mni_bin(fs_home = get_fs_home(), simplify = TRUE)
```

Arguments

env_var	Character; name of the environment variable to check.
opt_var	Character; name of the R option to check.
defaults	Character vector of default paths to try.
is_path	Logical; whether this setting represents a file/directory path.
simplify	Logical; if TRUE, returns only the value. If FALSE, returns a list with detailed information (value, source, exists).
fs_home	Character; FreeSurfer home directory.

Details

Lookup Hierarchy:

Settings are resolved in the following order (first match wins):

1. **R options** (set with `options()`)
2. **Environment variables** (set with `Sys.setenv()`)
3. **Default paths** (platform-specific)

Setting Options:

You can set R options in your `.Rprofile`:

```
options(
  freesurfer.home = "/usr/local/freesurfer",
  freesurfer.subj_dir = "~/freesurfer_subjects",
  freesurfer.verbose = TRUE
)
```

Value

When `simplify = TRUE`: Character string with the setting value, or `NA`. When `simplify = FALSE`: List with `value`, `source`, and `exists` components.

Functions

- `get_fs_setting()`: Core function for retrieving settings
- `get_fs_home()`: Retrieve FreeSurfer installation directory
- `get_fs_license()`: Retrieve FreeSurfer license file path
- `get_fs_subdir()`: Retrieve FreeSurfer subjects directory
- `get_fs_source()`: Retrieve FreeSurfer source script path
- `get_fs_verbosity()`: Retrieve FreeSurfer verbosity setting
- `get_fs_output()`: Retrieve FreeSurfer output format
- `get_mni_bin()`: Retrieve MNI tools directory

See Also

`have_fs()` to check if FreeSurfer is accessible, `fs_sitrep()` for comprehensive diagnostics, `get_fs()` to generate environment setup commands

Examples

```
# Get FreeSurfer home directory
fs_home <- get_fs_home()

# Get detailed information
fs_home_info <- get_fs_home(simplify = FALSE)

# Check license file
license <- get_fs_license()

# Get subjects directory
subj_dir <- get_fs_subdir()

# Check output format
format <- get_fs_output()  # Returns "nii.gz", "nii", "mgz", etc.

# Check verbosity
verbose <- get_fs_verbosity()  # Returns TRUE/FALSE

## Not run:
# Set custom options
options(freesurfer.home = "/custom/path/freesurfer")
get_fs_home()  # Returns: "/custom/path/freesurfer"

## End(Not run)
```

---

have_fs *Logical check if Freesurfer is accessible*

Description

Checks if FreesurferDIR is accessible and optionally if a license file exists.

Usage

```
have_fs(check_license = TRUE)
```

Arguments

check_license	logical Should a license file be checked for existence?

Value

Logical TRUE if Freesurfer is accessible and license (if checked) is found, FALSE otherwise.

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()
```

Arguments

`file`	(character) input filename
`outfile`	(character) output filename
`...`	Additional arguments to pass to `fs_cmd`

Value

Character filename of output

Functions

- `mnc2nii.help()`: Display information about mnc2nii command

Examples

```
img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
mnc = nii2mnc(img)
img_file = mnc2nii(mnc, outfile = temp_file(fileext = ".nii"))
neurobase::readnii(img_file, verbose = get_fs_verbosity())
```

mri_convert *Convert Medical Image Formats with FreeSurfer*

Description

Calls FreeSurfer's `mri_convert` to convert between different medical imaging formats (NIfTI, MGZ, MINC, etc.) with enhanced format validation and error handling.

Usage

```
mri_convert(
  file,
  outfile,
  opts = "",
  format_check = TRUE,
  timeout_seconds = 300,
  ...
)

mri_convert.help(...)
```

Arguments

file	Character; input filename or nifti object.
outfile	Character; output filename.
opts	Character. Additional options to Freesurfer function.
format_check	Logical; whether to validate input/output formats and warn about unexpected formats. Default is TRUE.
timeout_seconds	Numeric; command timeout in seconds. Default is 300.
...	Additional arguments passed to fs_help()

Details

Runtime FreeSurfer CLI help is available via the helper function mri_convert.help(). When called, that helper will attempt to fetch and display the underlying FreeSurfer command-line help if FreeSurfer is installed on the system. The package does not embed the CLI help into the installed Rd at package build time.

This function provides format validation and informative messages about conversions between different image types.

Value

Result of base::system() command, typically exit status.

Functions

- mri_convert.help(): Display FreeSurfer help for mri_convert

FreeSurfer Command Help

When FreeSurfer is installed and available, detailed command-line help for the underlying mri_convert command can be accessed via mri_convert.help(). This provides comprehensive information about FreeSurfer's native command options, file format support, and usage examples.

To see FreeSurfer CLI help: mri_convert.help()

See Also

fs_cmd() for the underlying command wrapper

Examples

```
# Convert nifti object to MGZ format
img <- oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5, 5, 5)))
res <- mri_convert(img, outfile = temp_file(fileext = ".mgz"))

## Not run:
# Convert MGZ to NIfTI
mri_convert("brain.mgz", "brain.nii.gz")

# With additional options
mri_convert("brain.mgz", "brain.nii.gz", opts = "--conform")

## End(Not run)
```

---

mri_deface *MRI Deface*

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 orface_template is NULL, they will be downloaded.

Examples

```
## Not run:
base_url = "https://surfer.nmr.mgh.harvard.edu/pub/dist/mri_deface"
url = file.path(base_url, "sample_T1_input.mgz")
x = temp_file(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_1/113/113-01-T1.nii.gz")
   x = temp_file(fileext = ".nii.gz")
   out = try({
     utils::download.file(url, destfile = x)
   })
   noface = mri_deface(x)
 }

## End(Not run)
```

`mri_info` *MRI information*

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

```
## Not run:
img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
mri_info(img)

## End(Not run)
```

`mri_info.help` *MRI information Help*

Description

This calls Freesurfer's `mri_info` help

Usage

```
mri_info.help(...)
```

Arguments

`...`	Additional arguments passed to `fs_help()`

`mri_mask` *Use Freesurfers MRI Mask*

Description

This function calls `mri_mask` to mask an image

Usage

```
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 Freesurfer function.
`...`	additional arguments passed to `fs_cmd`.

Value

Character or nifti depending on `retimg`

Examples

```
## Not run:
img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
mask = img > 1
res = mri_mask(img, mask)

## End(Not run)
```

---

mri_mask.help *MRI Normalize Help*

Description

This calls Freesurfer's `mri_mask` help

Usage

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

Arguments

`...`	Additional arguments passed to `fs_help()`

---

mri_normalize *Use Freesurfers 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 Freesurfer function.
`...`	additional arguments passed to `fs_cmd`.

Value

Character or nifti depending on `retimg`

Examples

```
## Not run:
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(...)
```

Arguments

`...`	additional arguments to `get_fs`

Value

Result of `fs_help`

`mri_segment` *Use Freesurfers MRI Segmentation Algorithm*

Description

This function calls `mri_segment`

This calls Freesurfer's `mri_segment` help

Usage

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

mri_segment.help(...)
```

Arguments

file	(character) input filename
outfile	(character) output filename
retimg	(logical) return image of class nifti
opts	Character. Additional options to Freesurfer function.
...	Additional arguments passed to fs_help()

Value

Character or nifti depending on retimg

Result of fs_help

Functions

- mri_segment.help(): Display FreeSurfer help for mri_segment

Note

NOT COMPLETE

---

mri_surf2surf *Resample Cortical Surface Data with FreeSurfer*

Description

Calls FreeSurfer's mri_surf2surf to resample one cortical surface onto another, enabling comparison of surface data across subjects or atlases.

Usage

```
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 = get_fs_verbosity(),
  ...
)

mri_surf2surf.help(...)
```

Arguments

subject	Character; source subject name.
target_subject	Character; target subject name (e.g., "fsaverage").
trg_type	Character; target file type. One of "curv", "w" (paint), "mgh", or "nii".
src_type	Character; source file type. One of "curv" or "w" (paint).
outfile	Character; output filename. If NULL, a temporary file is created.
hemi	Character; hemisphere. One of "lh" or "rh".
sval	Character; source value/measure (e.g., "thickness").
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 Freesurfer function.
verbose	(logical) print diagnostic messages
...	Additional arguments passed to fs_help()

Details

This function is commonly used to:

- Project individual subject data onto a template (e.g., fsaverage)
- Resample between different surface resolutions
- Convert between surface file formats

The output filename is automatically prefixed with the hemisphere (e.g., "lh.output.mgz").

Value

Character; path to the output file (prefixed with hemisphere).

Functions

- mri_surf2surf.help(): Display FreeSurfer help for mri_surf2surf

See Also

fs_help() for FreeSurfer command documentation

Examples

```
## Not run:
out <- mri_surf2surf(
  subject = "bert",
  target_subject = "fsaverage",
  trg_type = "curv",
  src_type = "curv",
  hemi = "rh",
  sval = "thickness"
```

```
)

## End(Not run)
```

`mri_synthstrip` *Use Freesurfers MRI SynthStrip*

Description

This function calls `mri_mask` to mask an image

Usage

```
mri_synthstrip(
  file,
  outfile = NULL,
  retimg = TRUE,
  maskfile = NULL,
  opts = "",
  ...
)

synthstrip(
  file,
  outfile = NULL,
  retimg = TRUE,
  maskfile = NULL,
  opts = "",
  ...
)
```

Arguments

`file`	(character) input filename
`outfile`	(character) output filename
`retimg`	(logical) return image of class nifti
`maskfile`	(character) path for mask output
`opts`	Character. Additional options to Freesurfer function.
`...`	additional arguments passed to `fs_cmd`.

Value

Character or nifti depending on `retimg`

Examples

```
## Not run:
mock_nifti = array(rnorm(5*5*5), dim = c(5,5,5))
img = oro.nifti::nifti(mock_nifti)
res = mri_synthstrip(img)

## End(Not run)
```

`mri_synthstrip.help` *MRI Normalize Help*

Description

This calls Freesurfer's `mri_mask` help

Usage

```
mri_synthstrip.help(...)
```

Arguments

`...`	additional arguments to `get_fs`

Value

Result of `fs_help`

`mri_watershed` *Use Freesurfers MRI Watershed Algorithm*

Description

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

This calls Freesurfer's `mri_watershed` help

Usage

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

mri_watershed.help(...)
```

Arguments

file	(character) input filename
outfile	(character) output filename
retimg	(logical) return image of class nifti
opts	Character. Additional options to Freesurfer function.
...	Additional arguments passed to fs_help()

Value

Character or nifti depending on retimg

Result of fs_help

Functions

- mri_watershed.help(): Display FreeSurfer help for mri_watershed

Examples

```
## Not run:
mri_watershed("/path/to/T1.nii.gz")

## End(Not run)
```

---

mris_convert *Convert Cortical Surface File Formats with FreeSurfer*

Description

These functions call FreeSurfer's mris_convert to convert between cortical surface file formats. The base function mris_convert() provides general conversion, while specialized variants handle specific data types.

Usage

```
mris_convert(
  infile,
  outfile = NULL,
  ext = ".asc",
  opts = "",
  verbose = get_fs_verbosity(),
  ...
)

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

mris_convert_curv(curv, opts = "", ...)
```

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

mris_convert_vertex(opts = "", ...)

mris_convert.help(...)
```

Arguments

infile	Character; path to input surface file.
outfile	Character; path to output file. If NULL, a temporary file with the specified extension is created.
ext	Character; output file extension when outfile is NULL. Default is ".asc".
opts	Character. Additional options to Freesurfer function.
verbose	(logical) print diagnostic messages
...	Additional arguments passed to mris_convert.
annot	Character; path to annotation or gifti label file.
curv	Character; path to scalar curvature overlay file.

Details

FreeSurfer's mris_convert is a general conversion program for cortical surface file formats. It can convert between binary and ASCII formats, extract specific data types, and transform between coordinate systems.

Value

Character; path to the output file. For mris_convert(), the output has a "separator" attribute indicating the file's field separator.

Functions

- mris_convert_annot(): Convert with annotation or gifti label data
- mris_convert_curv(): Convert with scalar curvature overlay
- mris_convert_normals(): Extract surface normals
- mris_convert_vertex(): Extract vertex coordinates
- mris_convert.help(): Display FreeSurfer help for mris_convert

Note

For curvature conversions, the output filename may be modified by FreeSurfer. You may need to prepend the hemisphere prefix to get the correct path.

Examples

```
## Not run:
bert_dir <- file.path(fs_subj_dir(), "bert")
bert_surf <- file.path(bert_dir, "surf", "lh.white")

# Basic conversion to ASCII
asc_file <- mris_convert(infile = bert_surf)

# Convert with annotation data
gii_file <- mris_convert_annot(
  infile = bert_surf,
  annot = file.path(bert_dir, "label", "lh.aparc.annot"),
  ext = ".gii"
)

# Convert with curvature overlay
curv_file <- mris_convert_curv(
  infile = bert_surf,
  curv = file.path(bert_dir, "surf", "lh.thickness")
)

# Extract surface normals
normals_file <- mris_convert_normals(infile = bert_surf)

# Extract vertex coordinates
vertex_file <- mris_convert_vertex(infile = bert_surf)

## End(Not run)
```

---

mris_euler_number *MRIs Euler Number*

Description

This function calls `mris_euler_number` to calculate the Euler Number with improved error handling and output management.

Usage

```
mris_euler_number(
  file,
  outfile = NULL,
  opts = "",
  cleanup_temp = TRUE,
  timeout_seconds = 120,
  validate_format = TRUE,
  ...
)
```

Arguments

`file`	(character) input filename - surface file
`outfile`	(character) output filename for results (optional)
`opts`	Character. Additional options to FreeSurfer function.
`cleanup_temp`	(logical) Whether to clean up temporary files
`timeout_seconds`	(numeric) Command timeout in seconds (default 120)
`validate_format`	(logical) Whether to validate input format
`...`	Additional arguments to pass to `fs_cmd`

Value

Character vector containing the Euler number results

Examples

```
## Not run:
img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
res = mris_euler_number(img, outfile = temp_file(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(...)
```

Arguments

`...`	Additional arguments passed to `fs_help()`

nii2mnc *Convert NIfTI to MNC*

Description

This function calls `nii2mnc` to convert NIfTI to MNC files

Usage

```
nii2mnc(file, outfile = tempfile(fileext = ".mnc"), ...)

nii2mnc.help()
```

Arguments

`file`	(character) input filename
`outfile`	(character) output filename
`...`	Additional arguments to pass to `fs_cmd`

Value

Character filename of output

Functions

- `nii2mnc.help()`: Display information about nii2mnc command

Examples

```
img = oro.nifti::nifti(array(rnorm(5*5*5), dim = c(5,5,5)))
mnc = nii2mnc(img)
img_file = mnc2nii(mnc)
```

nu_correct *Use FreeSurfer's Non-Uniformity Correction*

Description

This function calls `nu_correct` to correct for intensity non-uniformity in MRI images using N3 (nonparametric nonuniform normalization).

Usage

```
nu_correct(file, mask = NULL, opts = "", verbose = get_fs_verbosity(), ...)

nu_correct.help(...)
```

Arguments

file	Character; input filename.
mask	Character or nifti; mask to use for correction.
opts	Character. Additional options to Freesurfer function.
verbose	(logical) print diagnostic messages
...	Additional arguments passed to fs_help()

Details

The nu_correct command performs N3 (nonparametric nonuniform intensity normalization) to correct for intensity inhomogeneity in MRI images. This is particularly useful for T1-weighted images where intensity varies across the image due to magnetic field inhomogeneities.

Common options:

- -mask <file>: Use a binary mask
- -distance <value>: N3 spline distance (default 200mm)
- -iterations <value>: Maximum iterations (default 4)
- -stop <value>: Stopping criterion (default 0.01)

Note: This is an MNI tool and uses different help syntax (-help instead of --help).

Value

Object of class nifti.

Functions

- nu_correct.help(): Display information about nu_correct

See Also

mri_normalize() for FreeSurfer's normalization, nu_correct.help() for command information

Examples

```
## Not run:
# Basic usage
nu_correct("/path/to/T1.nii.gz")

# With mask
nu_correct("/path/to/T1.nii.gz", mask = "/path/to/mask.nii.gz")

## End(Not run)
```

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

```
read_annotation(path, verbose = get_fs_verbosity())
```

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

```
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 FreeSurfer Anatomical Segmentation Statistics*

Description

Reads and parses an aseg.stats file produced by FreeSurfer's anatomical segmentation pipeline. Returns both global brain measures and structure- specific statistics.

Usage

```
read_aseg_stats(file, lowercase = TRUE)
```

Arguments

file	Character; path to an aseg.stats file from FreeSurfer.
lowercase	Logical; if TRUE (default), converts measure names and column names to lowercase for easier access. If FALSE, preserves FreeSurfer's original capitalization.

Details

The aseg.stats file contains volumetric and morphometric statistics for subcortical structures and global brain measures computed by FreeSurfer's recon-all pipeline.

Output Structure:
The function returns a list with two data frames:
measures: Global brain measures including:

- Brain segmentation volume
- Estimated total intracranial volume (eTIV)
- Normalization factors

structures: Structure-specific measures including:

- Structure index and name
- Number of voxels
- Volume (mm³)
- Mean intensity and standard deviation
- Minimum, maximum, and range of intensities

File Location:
For a subject processed with recon-all, the file is typically located at: $SUBJECTS_DIR/<subject_id>/stats/aseg.stat

Value

A list with two components:

- measures: A data frame with global brain measures, with columns:
  - measure: Short name of the measure
  - measure_long: Long descriptive name
  - meaning: Description of what the measure represents
  - value: Numeric value of the measure
  - units: Units of measurement
- structures: A data frame with structure-specific statistics, with columns:
  - Index: Structure index
  - SegId: Segmentation ID
  - NVoxels: Number of voxels
  - Volume_mm3: Volume in cubic millimeters
  - StructName: Name of the anatomical structure
  - normMean: Normalized mean intensity
  - normStdDev: Normalized standard deviation
  - normMin: Normalized minimum
  - normMax: Normalized maximum
  - normRange: Normalized range

See Also

- read_fs_table() for reading other FreeSurfer tables
- recon_all() for running the FreeSurfer pipeline
- stats2table() for combining stats from multiple subjects

Examples

```
# Read stats for the "bert" example subject
file <- file.path(fs_subj_dir(), "bert", "stats", "aseg.stats")

if (file.exists(file)) {
  out <- read_aseg_stats(file)

  # Examine global measures
  print(head(out$measures))

  # Get estimated total intracranial volume
  etiv <- out$measures$value[out$measures$measure == "estimatedtotalintracranialvol"]
  cat("eTIV:", etiv, "mm³\n")

  # Examine structure-specific stats
  print(head(out$structures))

  # Get hippocampal volumes
  hippo <- out$structures[grepl("Hippocampus", out$structures$StructName), ]
  print(hippo[, c("StructName", "Volume_mm3")])
}
```

---

read_fs_label *Read Label File*

Description

Reads an label file from an individual subject

Usage

```
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

```
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)
```

---

read_fs_table *Read FreeSurfer Table Output*

Description

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

Usage

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

read_stats_table(
  file,
  sep = NULL,
  header = TRUE,
  validate_format = TRUE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

file	(character path) filename of text file
sep	separator to override attribute of file, to pass to read.table.
header	Is there a header in the data
validate_format	(logical) Whether to warn about unexpected formats
stringsAsFactors	(logical) passed to read.table
...	additional arguments to read.table

Value

data.frame from the file

Examples

```
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)
```

---

read_mgz *Read MGH or MGZ File*

Description

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

Usage

```
read_mgz(file, validate_format = TRUE, cleanup_temp = TRUE, ...)

read_mgh(file, validate_format = TRUE, cleanup_temp = TRUE, ...)
```

Arguments

file	(character) input filename - MGH or MGZ file
validate_format	(logical) Whether to validate file format
cleanup_temp	(logical) Whether to clean up temporary files
...	Additional arguments to pass to fs_cmd

Value

Object of class `nifti`

`read_mnc` *Read MNC File*

Description

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

Usage

```
read_mnc(file)
```

Arguments

`file`	(character) input filename

Value

Object of class `nifti`

`recon` *FreeSurfer Reconstruction Pipeline Functions*

Description

Functions to run FreeSurfer's cortical reconstruction pipeline with varying levels of control and customization.

Usage

```
recon_all(
  subjid = NULL,
  infile = NULL,
  outdir = NULL,
  verbose = get_fs_verbosity(),
  opts = "-all",
  ...
)

recon(
  subjid = NULL,
  infile = NULL,
  outdir = NULL,
```

```
  opts = "",
  options = recon_steps(),
  verbose = get_fs_verbosity()
)

recon_steps()
```

Arguments

subjid	subject id, if NULL, the basename of the infile will be used
infile	Character; path to input file in DICOM or NIfTI format.
outdir	Character, Path to output directory.
verbose	(logical) print diagnostic messages
opts	Character. Additional options to Freesurfer function.
...	Additional arguments passed to `reconner()`.
options	Named list of logical values for `recon()`, specifying which pipeline steps to include. See `recon_steps()` for available options.

Details

Pipeline Overview:

The reconstruction pipeline performs:

1. Motion correction and intensity normalization
2. Skull stripping and subcortical segmentation
3. White matter segmentation
4. Surface generation (white and pial)
5. Cortical parcellation and thickness calculation

Processing Time:

A full reconstruction typically takes 6-24 hours. Use `opts = "-parallel"` for parallel processing if available.

Output Location:

Results stored in `$SUBJECTS_DIR/<subjid>/` with subdirectories: `mri/`, `surf/`, `label/`, `stats/`

Value

Result from `base::system()` call, typically exit status (0 = success).

Functions

- `recon_all()`: Run complete reconstruction pipeline with default settings
- `recon()`: Run reconstruction with step-by-step control
- `recon_steps()`: Get named vector of available reconstruction steps

References

Dale et al. (1999) Neuroimage 9:179-194. Fischl et al. (1999) Neuroimage 9:195-207.

See Also

`read_aseg_stats()` to read segmentation statistics `reconner()` for low-level control of `recon-all`

Examples

```
## Not run:
# Full reconstruction with recon_all
recon_all(
  infile = "T1_scan.nii",
  subjid = "subject01",
  outdir = "~/freesurfer_output"
)

# Step-by-step control with recon
steps <- recon_steps()
steps["normalization"] <- FALSE
recon(
  infile = "T1_scan.nii",
  subjid = "subject02",
  options = steps
)

# Low-level control with reconner
reconner(
  infile = "T1_scan.nii",
  subjid = "subject03",
  opts = "-autorecon1 -parallel"
)

## End(Not run)
```

`recon_manual` *Manual Freesurfer Reconstruction Workflow*

Description

A set of helper functions for running different stages of Freesurfer's `recon-all` pipeline: `autorecon1`, `autorecon2`, and `autorecon3`. These functions are wrappers around the reconner function, simplifying the reconstruction process by specifying pre-defined options for each stage.

Usage

```
recon_con1(infile, subjid, outdir = NULL, verbose = get_fs_verbosity())

autorecon1(infile, subjid, outdir = NULL, verbose = get_fs_verbosity())

recon_con2(infile, subjid, outdir = NULL, verbose = get_fs_verbosity())

autorecon2(infile, subjid, outdir = NULL, verbose = get_fs_verbosity())

recon_con3(infile, subjid, outdir = NULL, verbose = get_fs_verbosity())

autorecon3(infile, subjid, outdir = NULL, verbose = get_fs_verbosity())
```

Arguments

infile	Input filename in DICOM (.dcm) or NIfTI (.nii) format. This is the brain image file to be processed. If NULL, some recon-all functionality may not work.
subjid	subject id, if NULL, the basename of the infile will be used
outdir	Output directory for storing reconstruction results. If NULL, the default Freesurfer subject directory is used.
verbose	(logical) print diagnostic messages

Details

Freesurfer is a widely used software suite for processing and analyzing brain MRI images. The recon-all tool provides a stepwise reconstruction process:

- -autorecon1: Motion correction to skull stripping (steps 1-5)
- -autorecon2: Further processing, including intensity normalization
- -autorecon3: Final surface generation and QC measures

These helpers call specific stages of the recon-all pipeline to provide finer control over processing.

Value

The result of the system call to recon-all, either the command's output or a diagnostic message if an issue occurs.

Functions

- recon_con1(): Performs -autorecon1, which includes steps 1-5 of Freesurfer's reconstruction process.
- autorecon1(): Helper function for recon_con1 with the same functionality.
- recon_con2(): Performs -autorecon2, which includes steps 6+ of Freesurfer's reconstruction process.
- autorecon2(): Helper function for recon_con2 with the same functionality.

- `recon_con3()`: Performs the final stage of Freesurfer's reconstruction process, which includes generating cortical surfaces,
- `autorecon3()`: Helper function for `recon_con3` with the same functionality.

Note

See the official Freesurfer documentation for additional details on each autorecon stage: https://surfer.nmr.mgh.harvard.edu/fswiki/recon-all.

If `infile` is set to `NULL`, the `-i` flag (input file flag) in `recon-all` will be omitted, which may or may not be appropriate depending on your use case.

See Also

reconner, recon, recon_all

Examples

```
## Not run:
# Example: Perform Step 1-5 reconstruction (Motion Correction to Skull Strip)
recon_con1(
  infile = "subject_001.nii",
  outdir = "/output_dir",
  subjid = "subj01",
  verbose = TRUE
)

# Example: Run autorecon1 with equivalent helper function
autorecon1(
  infile = "subject_001.nii",
  outdir = "/output_dir",
  subjid = "subj01",
  verbose = TRUE
)

# Example: Perform further processing for Step 6+ using autorecon2
autorecon2(
  infile = "subject_002.nii",
  outdir = "/output_dir",
  subjid = "subj02",
  verbose = TRUE
)

# Example: Complete reconstruction with autorecon3
autorecon3(
  infile = "subject_003.nii",
  outdir = "/output_dir",
  subjid = "subj03",
  verbose = FALSE
)

## End(Not run)
```

reconner *Reconstruction Helper for FreeSurfer's recon-all*

Description

Wrapper around FreeSurfer's `recon-all` command for brain surface reconstruction from MRI data. Handles input processing, subject directory creation, and command flag management.

Usage

```
reconner(
  subjid = NULL,
  infile = NULL,
  outdir = NULL,
  opts = "-all",
  force = FALSE,
  verbose = get_fs_verbosity()
)
```

Arguments

`subjid`	subject id, if NULL, the basename of the infile will be used
`infile`	Character; path to input file in DICOM or NIfTI format.
`outdir`	Character, Path to output directory.
`opts`	Character. Additional options to Freesurfer function.
`force`	Logical; force execution even if subject directory exists. Default is `FALSE`.
`verbose`	(logical) print diagnostic messages

Details

FreeSurfer's `recon-all` performs cortical reconstruction and volumetric segmentation. This function simplifies usage by:

- Automatically deriving subject ID from input filename if not provided
- Managing subject directory paths
- Providing force option for re-running on existing subjects

Value

Result of `base::system()` call, typically exit status (0 = success).

See Also

`tracker()` for diffusion tractography pipeline

Examples

```
## Not run:
reconner(infile = "input.nii", outdir = "/output_dir", subjid = "subj01")
reconner(infile = "input.nii", outdir = "/output_dir")
reconner(infile = "input.nii", opts = "-autorecon2", force = TRUE)

## End(Not run)
```

`stats2table` *Generalized Stats to Table*

Description

This function serves as a flexible abstraction to execute the FreeSurfer commands `asegstats2table` or `aparcstats2table`, which are primarily used to convert parcellation and segmentation statistics to tabular formats. The function dynamically handles shared logic, constructs command-line arguments, and runs the appropriate FreeSurfer command. Users can specify the type of input (subjects or input file paths) and various configuration options. With the appropriate parameters, it constructs and executes the matching system command.

Usage

```
stats2table(
  type = c("aseg", "aparc"),
  input,
  measure,
  input_type = c("subjects", "inputs"),
  outfile = NULL,
  delim = c("tab", "space", "comma", "semicolon"),
  skip = TRUE,
  subj_dir = NULL,
  opts = "",
  verbose = get_fs_verbosity(),
  ...
)

asegstats2table(
  subjects = NULL,
  inputs = NULL,
  measure = c("volume", "mean", "std"),
  ...
)

aparcstats2table(
  subjects,
  hemi = c("lh", "rh"),
```

```
  measure = c("area", "volume", "thickness", "thicknessstd", "meancurv", "gauscurv",
    "foldind", "curvind"),
  parc = c("aparc", "aparc.a2009s"),
  opts = "",
  ...
)

aparcstats2table.help(...)

asegstats2table.help(...)
```

Arguments

type	(character) Either "aparc" for cortical parcellation or "aseg" for subcortical segmentation.
input	(character) A vector representing either subject IDs (for input_type = "subjects") or file paths (for input_type = "inputs", applicable only to aseg).
measure	(character) The measurement to calculate. For example, "thickness" for cortical measures or "volume" for subcortical.
input_type	(character) Specifies the type of input. Must be one of "subjects" or "inputs".
outfile	(character) Name of the output file. If not specified, a temporary file will be created based on the specified delimiter.
delim	(character) Delimiter for the output file. This can be one of: "tab", "space", "comma", or "semicolon". The output file's delimiter is stored as an attribute for programmatic access.
skip	(logical) If TRUE, skips invalid inputs (e.g., missing files or data) without throwing errors.
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 Freesurfer function.
verbose	(logical) print diagnostic messages
...	Additional arguments passed to fs_help()
subjects	(character) A vector of subject identifiers. This is the primary way to specify inputs when using this function.
inputs	(character paths) A vector of input filenames, e.g. aseg.stats. Alternatively, use subjects to specify subject IDs.
hemi	(character) The hemisphere for which statistics are computed. Options are "lh" (left hemisphere) or "rh" (right hemisphere).
parc	(character) Specifies the parcellation scheme to be used. Options include "aparc" or "aparc.a2009s".

Value

A character string: the path to the output file with its delimiter stored as an attribute.

Functions

- `asegstats2table()`: Converts subcortical segmentation statistics into tabular format by calling the FreeSurfer `asegstats2table` command. `...` is passed to stats2table for additional options.
- `aparcstats2table()`: Converts cortical parcellation statistics into tabular format by calling the FreeSurfer `aparcstats2table` command. `...` is passed to stats2table for additional options.‘
- `aparcstats2table.help()`: Display FreeSurfer help for aparcstats2table
- `asegstats2table.help()`: Display FreeSurfer help for asegstats2table

Examples

```
## Not run:
# Example for asegstats2table
outfile_aseg <- asegstats2table(
    subjects = "bert",
    measure = "mean",
    delim = "tab"
)
print(outfile_aseg)

# Example for aparcstats2table
outfile_aparc <- aparcstats2table(
  subjects = "bert",
  hemi = "lh",
  measure = "thickness",
  delim = "tab",
  opts = "--etiv --scale=1.0"
)
print(outfile_aparc)

## End(Not run)
```

`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, ...)
```

Arguments

file	(character) input filename of curvature measure
outfile	(character) output filename (if wanted to be saved)
...	Additional arguments to pass to fs_cmd

Value

data.frame

Examples

```
## Not run:
   fname = file.path(fs_subj_dir(), "bert", "surf", "lh.thickness")
   out = surf_convert(fname)

## End(Not run)
```

---

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

```
## Not run:
infile = file.path(fs_subj_dir(),
                   "bert", "surf", "rh.pial")
res = surface_to_obj(infile = infile)

## End(Not run)
```

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

```
## Not run:
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)
rgl::open3d()
rgl::triangles3d(right_triangles,
color = rainbow(nrow(right_triangles)))
rgl::triangles3d(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)
rgl::open3d()
rgl::triangles3d(left_triangles,
color = rainbow(nrow(left_triangles)))
rgl::triangles3d(right_triangles,
color = rainbow(nrow(right_triangles)))

## End(Not run)
```

temp_file *Create a Temporary File with a Newly Created Directory*

Description

Create a Temporary File with a Newly Created Directory

Usage

```
temp_file(tmpdir = tempdir(check = TRUE), ...)
```

Arguments

tmpdir	Directory in which to create the temporary file
...	Arguments passed to tempfile()

Value

Full file path to temporary file with created directory

trac *FreeSurfer Diffusion Tractography Pipeline*

Description

These functions provide wrappers for FreeSurfer's `trac-all` command, which performs automated reconstruction of diffusion pathways.

Usage

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

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

```
trac_prep(infile, outdir = NULL, subjid, verbose = get_fs_verbosity())

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

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

tracker.help(...)
```

Arguments

infile	Character; input DWI (diffusion-weighted imaging) filename in DICOM or NIfTI format. Required for initial processing.
outdir	Character, Path to output directory.
subjid	subject id, if NULL, the basename of the infile will be used
verbose	(logical) print diagnostic messages
opts	Character. Additional options to Freesurfer function.
...	Additional arguments passed to `fs_help()`

Details

All functions call the same underlying FreeSurfer `trac-all` command with different option flags.

Value

Result of `base::system()` call, typically exit status (0 = success).

Functions

- `tracker()`: Low-level wrapper for trac-all command with custom options
- `trac_all()`: High-level wrapper running complete tractography pipeline
- `trac_prep()`: Run pre-processing step (step 1: image corrections, registration)
- `trac_bedpost()`: Run bedpost step (step 2: ball-and-stick model fitting)
- `trac_path()`: Run pathway reconstruction step (step 3: probabilistic tractography)
- `tracker.help()`: Display FreeSurfer help for trac-all

See Also

`recon_all()` for structural reconstruction pipeline

Examples

```
## Not run:
# Run full tractography pipeline
trac_all(
  infile = "dwi.nii",
  subjid = "subject01",
```

```
  outdir = "~/tractography_output"
)

# Run step-by-step
trac_prep(infile = "dwi.nii", subjid = "subject02")
trac_bedpost(infile = "dwi.nii", subjid = "subject02")
trac_path(infile = "dwi.nii", subjid = "subject02")

# Run with custom options
tracker(
  infile = "dwi.nii",
  subjid = "subject03",
  opts = "-prep -bedp"
)

## End(Not run)
```

Index