

# Package ‘freesurfer’

July 22, 2025

**Type** Package

**Title** Wrapper Functions for 'Freesurfer'

**Version** 1.8.1

**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/>)

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** John Muschelli [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-6469-1750/>>),  
Athanasia Mo Mowinckel [ctb] (ORCID:  
<<https://orcid.org/0000-0002-5756-0223/>>)

**Maintainer** John Muschelli <[muschelli2@gmail.com](mailto:muschelli2@gmail.com)>

**Repository** CRAN

**Date/Publication** 2025-05-12 16:00:02 UTC

## Contents

aparcstats2table . . . . .	3
aparcstats2table.help . . . . .	4
aparcstats_to_bg . . . . .	5
asegstats2table . . . . .	5
asegstats2table.help . . . . .	7
checkmnc-methods . . . . .	7
check_fs_result . . . . .	8
construct_subj_dir . . . . .	8
convert_surface . . . . .	9
freesurferdir . . . . .	10
freesurfer_read3 . . . . .	11
freesurfer_read3_con . . . . .	11
freesurfer_read_curv . . . . .	12
freesurfer_read_surf . . . . .	12
fs_cmd . . . . .	13
fs_help . . . . .	14
fs_imgext . . . . .	15
fs_lut . . . . .	15
fs_subj_dir . . . . .	16
fs_version . . . . .	16
get_fs . . . . .	17
get_fs_output . . . . .	17
have_fs . . . . .	18
mnc2nii . . . . .	18
mnc2nii.help . . . . .	19
mrisc_convert . . . . .	19
mrisc_convert.help . . . . .	20
mrisc_convert_annot . . . . .	21
mrisc_convert_curv . . . . .	21
mrisc_convert_normals . . . . .	22
mrisc_convert_vertex . . . . .	23
mrisc_euler_number . . . . .	24
mrisc_euler_number.help . . . . .	24
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 . . . . .	30
mri_surf2surf . . . . .	31
mri_surf2surf.help . . . . .	32

mri_synthstrip . . . . .	32
mri_synthstrip.help . . . . .	33
mri_watershed . . . . .	34
mri_watershed.help . . . . .	34
nii2mnc . . . . .	35
nii2mnc.help . . . . .	35
nu_correct . . . . .	36
nu_correct.help . . . . .	36
readmgz . . . . .	37
readmnc . . . . .	37
read_annotation . . . . .	38
read_aseg_stats . . . . .	38
read_fs_label . . . . .	39
read_fs_table . . . . .	40
recon . . . . .	41
reconner . . . . .	44
recon_all . . . . .	45
recon_con1 . . . . .	46
run_check_fs_cmd . . . . .	47
set_fs_subj_dir . . . . .	47
surface_to_obj . . . . .	48
surface_to_triangles . . . . .	48
surf_convert . . . . .	49
tracker . . . . .	50
trac_all . . . . .	51
trac_prep . . . . .	51

**Index****53**


---

aparcstats2table	<i>Parcellation Stats to Table</i>
------------------	------------------------------------

---

**Description**

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

**Usage**

```
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 out file
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
...	Additional arguments to pass to <a href="#">system</a>

### 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	<i>Convert Freesurfer aparcs Table to brainGraph</i>
--------------	--

---

**Description**

Converts Freesurfer aparcs table to brainGraph naming convention, relying on [aparcstats2table](#)

**Usage**

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

**Arguments**

subjects	subjects to analyze, passed to <a href="#">aparcstats2table</a>
measure	measure to be analyzed, passed to <a href="#">aparcstats2table</a>
...	additional arguments passed to <a href="#">aparcstats2table</a>

**Value**

Long data.frame

**Examples**

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

---

asegstats2table	<i>Parcellation Stats to Table</i>
-----------------	------------------------------------

---

**Description**

This function calls asegstats2table to convert parcellation statistics to a table

**Usage**

```
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 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, ...)  
  
ensure_mnc(file, ...)
```

**Arguments**

file	character or nifti object
...	options passed to <a href="#">checking</a>

**Value**

Character filename of mnc image

**Author(s)**

John Muschelli <muschellij2@gmail.com>

---

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**

```
check_fs_result(res, fe_before, fe_after)
```

**Arguments**

res	(numeric) Result from system command
fe_before	(logical) did the output file exist before the command ran
fe_after	(logical) did the output file exist after the command ran

**Value**

No return value, called for side effects

---

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()
)
```



**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"
touch = NULL

## End(Not run)
```

---

convert_surface	<i>Convert Freesurfer Surface</i>
-----------------	-----------------------------------

---

**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 <a href="#">mris_convert</a>

**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**

```
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**

```
freesurfer_read3_con(fid)
```

**Arguments**

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

**Value**

Numeric

**Examples**

```
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 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**

```
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**

```
freesurfer_read_surf(file)
```

**Arguments**

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

**Value**

List of length 2: vertices and faces are the elements

**Examples**

```
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**

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

**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 <a href="#">readnii</a> .
intern	(logical) to be passed to <a href="#">system</a>
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?
frontopts	(character) options/character to put in before filename
add_ext	(logical) should the extension be added to the outfile
bin_app	(character) appendix to add to <a href="#">get_fs</a>
...	additional arguments passed to <a href="#">system</a> .

**Value**

If retimg then object of class nifti. Otherwise, Result from system command, depends if intern is TRUE or FALSE.

---

fs_help	<i>Wrapper for getting Freesurfer help</i>
---------	--

---

**Description**

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

**Usage**

```
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 <a href="#">get_fs</a>

**Value**

Prints help output and returns output as character vector

**Examples**

```
if (have_fs()) {
  fs_help(func_name = "mri_watershed")
}
```

---

`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	<i>Determine Freesurfer Subjects Directory</i>
-------------	--

---

**Description**

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

**Usage**

```
fs_subj_dir()
```

**Value**

SUBJECTS\_DIR, such as `${FREESURFER_HOME}/subjects`

**Examples**

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

---

fs_version	<i>Find Freesurfer Version</i>
------------	--------------------------------

---

**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**

```
if (have_fs()) {  
  fs_version()  
}
```



---

get_fs	<i>Create command declaring FREESURFER_HOME</i>
--------	---

---

**Description**

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

**Usage**

```
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**

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

---

get_fs_output	<i>Determine Freesurfer output type</i>
---------------	---

---

**Description**

Finds the FSF\_OUTPUT\_FORMAT from system environment or `getOption("fs.outputtype")` for output type (nii.gz, nii, ANALYZE,etc)

**Usage**

```
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	<i>Logical check if Freesurfer is accessible</i>
---------	--

---

**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	<i>Convert MNC to NIFTI</i>
---------	-----------------------------

---

**Description**

This function calls mnc2nii to convert MNC files to NIFTI

**Usage**

```
mnc2nii(file, outfile = NULL, ...)
```

**Arguments**

file (character) input filename  
 outfile (character) output filename  
 ... Additional arguments to pass to [fs\\_cmd](#)

**Value**

Character filename of output

**Examples**

```
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, outfile = tempfile(fileext = ".nii"))
  neurobase::readnii(img_file, verbose = TRUE)
}
```

---

mnc2nii.help

*MNC to NIFTI Help*


---

**Description**

This calls Freesurfer's mnc2nii help

**Usage**

```
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**

```
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
...	Additional arguments to pass to <a href="#">system</a>

**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")
  )
}
```

---

mris\_convert.help

*Help file for Freesurfer MRIs Converter*

---

**Description**

This calls Freesurfer's mris\_convert help

**Usage**

```
mris_convert.help()
```

**Value**

Result of fs\_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 <code>mris_convert</code>
...	additional arguments to <code>mris_convert</code>

### 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 = "", ...)
```

**Arguments**

curv (character) scalar curv overlay file  
 opts (character) additional options to `mris_convert`  
 ... additional arguments to `mris_convert`

**Value**

Result of `mris_convert`

**Note**

The filename of the output may change due to how Freesurfer does curvature conversions and you may need to paste the prefix to get the correct filename, as seen in the example.

**Examples**

```

if (have_fs()) {
  bert_surf_dir = file.path(fs_subj_dir(), "bert", "surf")
  asc_file = mris_convert_curv(
    infile = file.path(bert_surf_dir, "lh.white"),
    curv = file.path(bert_surf_dir, "lh.thickness")
  )
  if (!file.exists(asc_file)) {
    asc_file = file.path(dirname(asc_file), paste0("lh.",
      basename(asc_file)))
  }
  res = read_fs_table(asc_file, header = FALSE)
  colnames(res) = c("index", "coord_1", "coord_2", "coord_3", "value")
  head(res)
}

```

---

`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**

```
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 call [mris\\_convert](#), using the `-v` option

**Usage**

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

**Arguments**

opts                    (character) additional options to [mris\\_convert](#)  
...                     additional arguments to [mris\\_convert](#)

**Value**

Result of [mris\\_convert](#)

**Examples**

```
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

<code>file</code>	(character) input filename
<code>outfile</code>	(character) output filename
<code>opts</code>	(character) additional options to <code>mris_euler_number</code>
<code>...</code>	Additional arguments to pass to <a href="#">fs_cmd</a>

### 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	<i>Use Freesurfer's MRI Converter</i>
-------------	---------------------------------------

---

**Description**

This function calls `mri_convert` to convert an image

**Usage**

```
mri_convert(file, outfile, opts = "", ...)
```

**Arguments**

<code>file</code>	(character) input filename
<code>outfile</code>	(character) output filename
<code>opts</code>	(character) additional options to <code>mri_convert</code>
<code>...</code>	Additional arguments to pass to <code>fs_cmd</code>

**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	<i>MRI Normalize Help</i>
------------------	---------------------------

---

**Description**

This calls Freesurfer's `mri_convert` help

**Usage**

```
mri_convert.help()
```

**Value**

Result of `fs_help`

---

mri_deface	<i>MRI Deface</i>
------------	-------------------

---

### Description

This calls Freesurfer's `mri_deface`

### Usage

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

### Arguments

<code>file</code>	File to pass to <code>mri_deface</code>
<code>brain_template</code>	gca brain template file to pass to <code>mri_deface</code>
<code>face_template</code>	gca face template file to pass to <code>mri_deface</code>
<code>...</code>	Additional arguments to pass to <code>fs_cmd</code>

### 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

```
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_1/113/113-01-T1.nii.gz")
    x = tempfile(fileext = ".nii.gz")
    out = try({
      utils::download.file(url, destfile = x)
    })
    noface = mri_deface(x)
  }
}
```

---

mri_info	<i>MRI information</i>
----------	------------------------

---

**Description**

This calls Freesurfer's `mri_info`

**Usage**

```
mri_info(file, ...)
```

**Arguments**

file	File to pass to <code>mri_info</code>
...	Additional arguments to pass to <code>fs_cmd</code>

**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	<i>MRI information Help</i>
---------------	-----------------------------

---

**Description**

This calls Freesurfer's `mri_info help`

**Usage**

```
mri_info.help()
```

**Value**

Result of `fs_help`

---

mri_mask	<i>Use Freesurfer's MRI Mask</i>
----------	----------------------------------

---

**Description**

This function calls `mri_mask` to mask an image

**Usage**

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

**Arguments**

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

**Value**

Character or nifti depending on `retimg`

**Examples**

```
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	<i>MRI Normalize Help</i>
---------------	---------------------------

---

**Description**

This calls Freesurfer's `mri_mask` help

**Usage**

```
mri_mask.help()
```

**Value**

Result of `fs_help`

---

mri_normalize	<i>Use Freesurfer's MRI Normalize Algorithm</i>
---------------	---

---

**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**

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

**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	<i>MRI Normalize Help</i>
--------------------	---------------------------

---

**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**

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

**Arguments**

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

**Value**

Character or nifti depending on `retimg`

**Note**

NOT COMPLETE

---

mri\_segment.help

*MRI Segment Help*

---

**Description**

This calls Freesurfer's `mri_segment help`

**Usage**

```
mri_segment.help()
```

**Value**

Result of `fs_help`

---

mri_surf2surf	<i>Use Freesurfer's mri_surf2surf function to resamples one cortical surface onto another</i>
---------------	---

---

## Description

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

## 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 = TRUE,
  ...
)
```

## Arguments

<code>subject</code>	(character) vector of subject name
<code>target_subject</code>	(character) vector of target subject name
<code>trg_type</code>	(character) target file type, can be curv, paint (w), mgh, or nii
<code>src_type</code>	(character) source file type, can be curv or paint (w)
<code>outfile</code>	(character) output filename
<code>hemi</code>	(character) hemisphere to run statistics
<code>sval</code>	(character) source file
<code>subj_dir</code>	(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
<code>opts</code>	(character) additional options to <code>mri_surf2surf</code>
<code>verbose</code>	(logical) print diagnostic messages
<code>...</code>	Additional arguments to pass to <a href="#">system</a>

## Value

Name of output file

**Examples**

```

if (have_fs()) {
  out = mri_surf2surf(
    subject = 'bert',
    target_subject = 'fsaverage',
    trg_type = 'curv',
    src_type = 'curv',
    hemi = "rh",
    sval = "thickness")
}

```

---

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\_synthstrip      *Use Freesurfer's 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 mri_mask
...	additional arguments passed to <a href="#">fs_cmd</a> .

### Value

Character or nifti depending on retimg

### 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_synthstrip(img)
}

```

---

mri\_synthstrip.help    *MRI Normalize Help*

---

### Description

This calls Freesurfer's mri\_mask help

### Usage

```
mri_synthstrip.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**

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

**Arguments**

<code>file</code>	(character) input filename
<code>outfile</code>	(character) output filename
<code>retimg</code>	(logical) return image of class nifti
<code>opts</code>	(character) additional options to <code>mri_watershed</code>
<code>...</code>	additional arguments passed to <a href="#">fs_cmd</a> .

**Value**

Character or nifti depending on `retimg`

**Examples**

```
## 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**

```
mri_watershed.help()
```

**Value**

Result of `fs_help`

---

nii2mnc	<i>Convert NIfTI to MNC</i>
---------	-----------------------------

---

**Description**

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

**Usage**

```
nii2mnc(file, outfile = NULL, ...)
```

**Arguments**

<code>file</code>	(character) input filename
<code>outfile</code>	(character) output filename
<code>...</code>	Additional arguments to pass to <a href="#">fs_cmd</a>

**Value**

Character filename of output

**Examples**

```
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)  
}
```

---

nii2mnc.help	<i>Convert NIfTI to MNC Help</i>
--------------	----------------------------------

---

**Description**

This calls Freesurfer's `mnc2nii help`

**Usage**

```
nii2mnc.help()
```

**Value**

Result of `fs_help`

---

nu_correct	<i>Use Freesurfer's Non-Uniformity Correction</i>
------------	---

---

**Description**

This function calls nu\_correct to correct for non-uniformity

**Usage**

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

**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**

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

## End(Not run)
```

---

nu_correct.help	<i>Non-Uniformity Correction Help</i>
-----------------	---------------------------------------

---

**Description**

This calls Freesurfer's nu\_correct help

**Usage**

```
nu_correct.help()
```

**Value**

Result of fs\_help

---

readmgz	<i>Read MGH or MGZ File</i>
---------	-----------------------------

---

**Description**

This function calls `mr_i_convert` to convert MGH/MGZ files to NIfTI, then reads it in using [readnii](#)

**Usage**

```
readmgz(file, ...)
```

```
readmgh(file, ...)
```

**Arguments**

file	(character) input filename
...	Additional arguments to pass to <a href="#">fs_cmd</a>

**Value**

Object of class `nifti`

---

readmnc	<i>Read MNC File</i>
---------	----------------------

---

**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

```
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

```
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

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

**Arguments**

file                   aseg.stats file from Freesurfer  
 lowercase            should the measures and values be lowercase ('TRUE') or kept as is?

**Value**

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

**Examples**

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

---

read_fs_label	<i>Read Label File</i>
---------------	------------------------

---

**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**

```
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)
}
```

---

read_fs_table	<i>Read Freesurfer Table Output</i>
---------------	-------------------------------------

---

## Description

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

## Usage

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

## Arguments

<code>file</code>	(character path) filename of text file
<code>sep</code>	separator to override attribute of file, to pass to <a href="#">read.table</a> .
<code>stringsAsFactors</code>	(logical) passed to <a href="#">read.table</a>
<code>header</code>	Is there a header in the data
<code>...</code>	additional arguments to <a href="#">read.table</a>

## Value

data.frame from the file

## Examples

```
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
if (requireNamespace("magrittr", quietly = TRUE)) {
  df_seg = asegstats2table(subjects = "bert", meas = "mean") %>%
    read_fs_table
}

## End(Not run)
```



---

recon	<i>Reconstruction from Freesurfer</i>
-------	---------------------------------------

---

**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,  
  kontrasurfreg = TRUE,  
  avgcurv = TRUE,  
  cortparc = TRUE,  
  parcstats = TRUE,  
  cortparc2 = TRUE,  
  parcstats2 = TRUE,  
  aparc2aseg = TRUE,  
  verbose = TRUE,  
  opts = ""
```

)

**Arguments**

<code>infile</code>	Input filename (dcm or nii)
<code>outdir</code>	Output directory
<code>subjid</code>	subject id
<code>motioncor</code>	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) <code>mri/orig/XXX.mgz</code> . The output will be the volume <code>mri/orig.mgz</code> . If no runs are found, then it looks for a volume in <code>mri/orig</code> (or <code>mri/orig.mgz</code> ). 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.
<code>nuintensitycor</code>	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 <code>'nu_correct'</code> . By default, four iterations of <code>nu_correct</code> are run. The flag <code>'-nuiterations'</code> specification of some other number of iterations.
<code>talairach</code>	computes the affine transform from the orig volume to the MNI305 atlas using the MINC program <code>mritotal</code> . Creates the files <code>mri/transform/talairach.auto.xfm</code> and <code>talairach.xfm</code> .
<code>normalization</code>	Performs intensity normalization of the orig volume and places the result in <code>mri/T1.mgz</code>
<code>skullstrip</code>	Removes the skull from <code>mri/T1.mgz</code> and stores the result in <code>mri/brainmask.auto.mgz</code> and <code>mri/brainmask.mgz</code> . Runs the <code>mri_watershed</code> program.
<code>gcareg</code>	Computes transform to align the <code>mri/nu.mgz</code> volume to the default GCA atlas found in <code>FREESURFER_HOME/average</code> . Creates the file <code>mri/transforms/talairach.lta</code> .
<code>canorm</code>	Further normalization, based on GCA model. Creates <code>mri/norm.mgz</code> .
<code>careg</code>	Computes a nonlinear transform to align with GCA atlas. Creates the file <code>mri/transform/talairach.m3z</code> .
<code>rmneck</code>	The neck region is removed from the NU-corrected volume <code>mri/nu.mgz</code> . Makes use of transform computed from prior CA Register stage. Creates the file <code>mri/nu_noneck.mgz</code> .
<code>skull_lta</code>	Computes transform to align volume <code>mri/nu_noneck.mgz</code> with GCA volume possessing the skull. Creates the file <code>mri/transforms/talairach_with_skull.lta</code> .
<code>calabel</code>	Labels subcortical structures, based in GCA model. Creates the files <code>mri/aseg.auto.mgz</code> and <code>mri/aseg.mgz</code> .
<code>normalization2</code>	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 <code>brain.mgz</code> volume. If <code>-noaseg</code> flag is used, then <code>aseg.mgz</code> is not used by <code>mri_normalize</code> .
<code>segmentation</code>	Attempts to separate white matter from everything else. The input is <code>mri/brain.mgz</code> , and the output is <code>mri/wm.mgz</code> . Uses intensity, neighborhood, and smoothness constraints. This is the volume that is edited when manually fixing defects. Calls <code>mri_segment</code> , <code>mri_edit_wm_with_aseg</code> , and <code>mri_pretext</code> . To keep previous edits, run with <code>-keeppwmedits</code> . If <code>-noaseg</code> is used, then <code>mri_edit_wm_aseg</code> 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/ponscut.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 mris_paint. Creates surf/?h.avg_curv.

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 <code>mris_ca_label</code> . <code>-cortparc</code> creates <code>label/?h.aparc.annot</code> , and <code>-cortparc2</code> creates <code>/label/?h.aparc.a2005s.annot</code> .
parcstats	Runs <code>mris_anatomical_stats</code> to create a summary table of cortical parcellation statistics for each structure, including 1. structure name 2. number of vertices 3. total surface area (mm <sup>2</sup> ) 4. total gray matter volume (mm <sup>3</sup> ) 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 <code>-parcstats</code> , the file is saved in <code>stats/?h.aparc.stats</code> . For <code>-parcstats2</code> , the file is saved in <code>stats/?h.aparc.a2005s.stats</code> .
cortparc2	see <code>cortparc</code> argument
parcstats2	see <code>cortparc2</code> argument
aparc2aseg	Maps the cortical labels from the automatic cortical parcellation ( <code>aparc</code> ) to the automatic segmentation volume ( <code>aseg</code> ). The result can be used as the <code>aseg</code> 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**

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

**Arguments**

infile	Input filename (dcm or nii)
outdir	Output directory
subjid	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	<i>Reconstruction from Freesurfer for All Steps</i>
-----------	---

---

**Description**

Reconstruction from Freesurfer for All Steps

**Usage**

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

**Arguments**

infile	Input filename (dcm or nii)
outdir	Output directory
subjid	subject id
verbose	print diagnostic messages
opts	Additional options
...	arguments passed to <a href="#">reconner</a>

**Value**

Result of `system`

**Note**

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

---

recon_con1	<i>Reconstruction from Motion Correction to Skull Strip</i>
------------	---

---

**Description**

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

**Usage**

`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**

<code>infile</code>	Input filename (dcm or nii)
<code>outdir</code>	Output directory
<code>subjid</code>	subject id
<code>verbose</code>	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	<i>Run and Check a Freesurfer Command</i>
------------------	---

---

**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**

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

**Arguments**

cmd	Command to be run
outfile	Output file to be produced
verbose	print diagnostic messages
...	Additional arguments to pass to <a href="#">system</a>

**Value**

Invisible NULL

**See Also**

[check\\_fs\\_result](#)

---

set_fs_subj_dir	<i>Set Freesurfer Subjects Directory</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

x	path to SUBJECTS_DIR defaults to <code>file.path(fs_dir(), "subjects")</code>
---	---

**Value**

No return value, called for side effects ('SUBJECTS\_DIR' environment variable set, and 'fs.subj\_dir' option set)

---

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 <a href="#">convert_surface</a>

### Value

Character filename of output file

### Examples

```
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 <a href="#">convert_surface</a>



**Value**

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

**Examples**

```

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::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)
if (requireNamespace("rgl", quietly = TRUE)) {
  rgl::open3d()
  rgl::triangles3d(left_triangles,
                  color = rainbow(nrow(left_triangles)))
  rgl::triangles3d(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, ...)
```

**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**

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

---

 tracker

---

*Tract Reconstruction Helper for trac-all from Freesurfer*


---

**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	<i>Tract Reconstruction Helper for trac-all from Freesurfer for All Steps</i>
----------	---

---

**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	<i>Tract Reconstruction for Each Step</i>
-----------	---

---

**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
- aparcs\_to\_bg, 5
- aparcsstats2table, 3, 5
- aparcsstats2table.help, 4
- asegstats2table, 5
- asegstats2table.help, 7
- autorecon1 (recon\_con1), 46
- autorecon2 (recon\_con1), 46
- autorecon3 (recon\_con1), 46
- check\_fs\_result, 8, 47
- 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, 48
- ensure\_mnc (checkmnc-methods), 7
- freesurfer\_dir (freesurferdir), 10
- freesurfer\_read3, 11
- freesurfer\_read3\_con, 11
- freesurfer\_read\_curv, 12
- freesurfer\_read\_surf, 12
- freesurferdir, 10
- fs\_cmd, 13, 19, 24–30, 33–37, 50
- fs\_dir (freesurferdir), 10
- fs\_help, 14
- fs\_imgext, 15
- fs\_lut, 15
- fs\_subj\_dir, 16
- fs\_version, 16
- get\_fs, 14, 17, 18
- get\_fs\_output, 17
- have\_fs, 18
- mnc2nii, 18, 37
- 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, 30
- mri\_surf2surf, 31
- mri\_surf2surf.help, 32
- mri\_synthstrip, 32
- mri\_synthstrip.help, 33
- mri\_watershed, 34
- mri\_watershed.help, 34
- mrisc\_convert, 9, 19, 21–23
- mrisc\_convert.help, 20
- mrisc\_convert\_annot, 21
- mrisc\_convert\_curv, 21
- mrisc\_convert\_normals, 22
- mrisc\_convert\_vertex, 23
- mrisc\_euler\_number, 24
- mrisc\_euler\_number.help, 24
- nii2mnc, 35
- nii2mnc.help, 35
- nu\_correct, 36
- nu\_correct.help, 36
- read.table, 40
- read\_annotation, 38

read\_aseg\_stats, 38  
read\_fs\_label, 39  
read\_fs\_table, 40  
readmgh (readmgz), 37  
readmgz, 37  
readmnc, 37  
readnii, 14, 37  
recon, 41  
recon\_all, 45  
recon\_con1, 46  
recon\_con1, recon\_con2, recon\_con3  
    (recon\_con1), 46  
recon\_con2 (recon\_con1), 46  
recon\_con3 (recon\_con1), 46  
reconner, 44, 45  
run\_check\_fs\_cmd, 47  
  
set\_fs\_subj\_dir, 47  
surf\_convert, 49  
surface\_to\_obj, 48  
surface\_to\_triangles, 48  
synthstrip (mri\_synthstrip), 32  
system, 4, 14, 20, 31, 44–47, 50–52  
  
trac\_all, 51  
trac\_bedpost (trac\_prep), 51  
trac\_path (trac\_prep), 51  
trac\_prep, 51  
tracker, 50