Package: ggseg.extra (via r-universe)

May 22, 2026

Title Create Brain Atlases for the 'ggsegverse' Plotting Ecosystem

Version 1.9.9.9001

Description Create brain atlas data sets compatible with the 'ggsegverse' plotting packages in 'R'. Provides pipelines for building cortical, subcortical, and white-matter tract atlases from 'FreeSurfer' annotation files, 'GIFTI' and 'CIFTI' surface formats, 'neuromaps', and volumetric 'NIfTI' images.

License MIT + file LICENSE

URL https://github.com/ggsegverse/ggseg.extra

BugReports https://github.com/ggsegverse/ggseg.extra/issues

Depends R (>= 4.2.0)

Imports cli, dplyr, future, furrr, ggseg.formats, ggseg3d (>= 2.0.0), grDevices, lifecycle, progressr, rlang, sf, stats, tools, utils

Suggests chromote, ciftiTools, freesurfer, freesurferformats, ggplot2, gifti, ggseg, hexSticker, htmlwidgets, knitr, magick, neuromapr, rgl, RNifti, rstudioapi, Rvcg, smoothr, spelling, terra, testthat (>= 3.0.0), tibble, tidyr, quarto, withr

VignetteBuilder knitr, quarto

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements FreeSurfer (subcortical/tract/wholebrain pipelines, and annotation reading for cortical pipelines), ImageMagick (subcortical/tract pipelines only), Connectome Workbench (optional, for CIFTI resampling).

Config/testthat/edition 3

Config/testthat/parallel true

Config/Needs/website ggsegverse/ggseg.docs

Config/pak/sysreqs
libabsl-dev chromium cmake libgdal-dev gdal-bin libgeos-dev make libmagick++-dev gsfonts libicu-dev libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-03-29 18:36:55 UTC

RemoteUrl https://github.com/ggsegverse/ggseg.extra

RemoteRef HEAD

RemoteSha e46b2e3176a1c944909bb202b52797af5d9f779f

Contents

as_verbosity *Coerce a value to a verbosity level*

Description

Converts logical, numeric, or character input to an integer verbosity level: `0L` (silent), `1L` (standard), or `2L` (debug).

Usage

```
as_verbosity(x)
```

Arguments

`x`	Value to coerce. Logical `FALSE` becomes `0L`, `TRUE` becomes `1L`. Numeric values are clamped to 0–2. Invalid input defaults to `1L`.

Value

Integer `0L`, `1L`, or `2L`

Examples

```
as_verbosity(FALSE)
as_verbosity(TRUE)
as_verbosity(2)
```

atlas_simplify *Simplify atlas 2D contours*

Description

Reduce vertex count in the sf geometry of a `ggseg_atlas` object using Douglas-Peucker simplification. Higher `tolerance` values produce simpler shapes with fewer vertices. This avoids re-running the full atlas creation pipeline.

Usage

```
atlas_simplify(atlas, tolerance = 0.5)
```

Arguments

`atlas`	A `ggseg_atlas` object with sf data.
`tolerance`	Simplification tolerance passed to sf::st_simplify(dTolerance). Typical range 0.1–2. Default 0.5.

Value

A modified ggseg_atlas with simplified sf geometry.

Examples

```
## Not run:
atlas <- atlas_simplify(my_atlas, tolerance = 1)
plot(atlas)

## End(Not run)
```

---

atlas_smooth *Smooth atlas 2D contours*

Description

Apply kernel smoothing to the sf geometry of a ggseg_atlas object. Higher smoothness values produce rounder region boundaries. This avoids re-running the full atlas creation pipeline.

Usage

```
atlas_smooth(atlas, smoothness = 5)
```

Arguments

atlas	A ggseg_atlas object with sf data.
smoothness	Smoothing bandwidth passed to smoothr::smooth(method = "ksmooth"). Typical range 3–15. Default 5.

Value

A modified ggseg_atlas with smoothed sf geometry.

Examples

```
## Not run:
atlas <- atlas_smooth(my_atlas, smoothness = 10)
plot(atlas)

## End(Not run)
```

`create_cortical_from_annotation` *Create cortical atlas from FreeSurfer annotation*

Description

[Maturing]

Turn FreeSurfer annotation files into a brain atlas you can plot with ggseg and ggseg3d. Reads the annotation, extracts vertex-to-region assignments, and generates 2D polygon geometry by projecting the inflated mesh triangles to 2D via orthographic projection.

Usage

```
create_cortical_from_annotation(
  input_annot,
  atlas_name = NULL,
  output_dir = NULL,
  hemisphere = c("rh", "lh"),
  views = c("lateral", "medial", "superior", "inferior"),
  tolerance = NULL,
  smooth_refinements = NULL,
  cleanup = NULL,
  verbose = get_verbose(),
  skip_existing = NULL
)
```

Arguments

`input_annot`	Character vector of paths to annotation files. Files should follow FreeSurfer naming convention with `lh.` or `rh.` prefix (e.g., `c("lh.aparc.annot", "rh.aparc.annot")`).
`atlas_name`	Name for the atlas. If NULL, derived from the input filename.
`output_dir`	Directory to store intermediate files (screenshots, masks, contours). Defaults to `tempdir()`.
`hemisphere`	Which hemispheres to include: "lh", "rh", or both.
`views`	Which views to include: "lateral", "medial", "superior", "inferior".
`tolerance`	Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to `sf::st_simplify()`. If not specified, uses `options("ggseg.extra.tolerance")` or the `GGSEG_EXTRA_TOLERANCE` environment variable. Default is 1.
`smooth_refinements`	Number of Chaikin corner-cutting refinements to apply to 2D polygons. Higher values produce smoother region boundaries (typical range: 0–3). 0 disables smoothing. If not specified, uses `options("ggseg.extra.smooth_refinements")` or the `GGSEG_EXTRA_SMOOTH_REFINEMENTS` environment variable. Default is 2.

cleanup	Remove intermediate files after atlas creation. If not specified, uses options("ggseg.extra.cleanup") or the GGSEG_EXTRA_CLEANUP environment variable. Default is TRUE.
verbose	Verbosity level: 0 (silent), 1 (standard progress, default), or 2 (debug, includes FreeSurfer output). Logical values are accepted (TRUE = 1, FALSE = 0). If not specified, uses the value from options("ggseg.extra.verbose") or the GGSEG_EXTRA_VERBOSE environment variable.
skip_existing	Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses options("ggseg.extra.skip_existing") or the GGSEG_EXTRA_SKIP_EXISTING environment variable. Default is TRUE.

Value

A ggseg_atlas object containing region metadata (core), vertex indices for 3D rendering, a colour palette, and sf geometry for 2D plots.

Examples

```
## Not run:
atlas <- create_cortical_from_annotation(
  input_annot = c("lh.aparc.DKTatlas.annot", "rh.aparc.DKTatlas.annot")
)
ggseg(atlas = atlas)

## End(Not run)
```

---

create_cortical_from_cifti

Create cortical atlas from a CIFTI file

Description

[Experimental]

Build a brain atlas from a CIFTI dense label file (.dlabel.nii). The file must be in fsaverage5 space (10,242 vertices per hemisphere).

Usage

```
create_cortical_from_cifti(
  cifti_file,
  atlas_name = NULL,
  output_dir = NULL,
  hemisphere = c("rh", "lh"),
  views = c("lateral", "medial", "superior", "inferior"),
  tolerance = NULL,
  smooth_refinements = NULL,
  cleanup = NULL,
  verbose = get_verbose(),
  skip_existing = NULL
)
```

Arguments

cifti_file	Path to a .dlabel.nii CIFTI file.
atlas_name	Name for the atlas. If NULL, derived from the input filename.
output_dir	Directory to store intermediate files (screenshots, masks, contours). Defaults to tempdir().
hemisphere	Which hemispheres to include: "lh", "rh", or both.
views	Which views to include: "lateral", "medial", "superior", "inferior".
tolerance	Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to sf::st_simplify(). If not specified, uses options("ggseg.extra.tolerance") or the GGSEG_EXTRA_TOLERANCE environment variable. Default is 1.
smooth_refinements	Number of Chaikin corner-cutting refinements to apply to 2D polygons. Higher values produce smoother region boundaries (typical range: 0–3). 0 disables smoothing. If not specified, uses options("ggseg.extra.smooth_refinements") or the GGSEG_EXTRA_SMOOTH_REFINEMENTS environment variable. Default is 2.
cleanup	Remove intermediate files after atlas creation. If not specified, uses options("ggseg.extra.cleanup") or the GGSEG_EXTRA_CLEANUP environment variable. Default is TRUE.
verbose	Verbosity level: 0 (silent), 1 (standard progress, default), or 2 (debug, includes FreeSurfer output). Logical values are accepted (TRUE = 1, FALSE = 0). If not specified, uses the value from options("ggseg.extra.verbose") or the GGSEG_EXTRA_VERBOSE environment variable.
skip_existing	Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses options("ggseg.extra.skip_existing") or the GGSEG_EXTRA_SKIP_EXISTING environment variable. Default is TRUE.

Value

A ggseg_atlas object.

Examples

```
## Not run:
atlas <- create_cortical_from_cifti(
  cifti_file = "parcellation.dlabel.nii"
)

## End(Not run)
```

`create_cortical_from_gifti` *Create cortical atlas from GIFTI annotation files*

Description

[Experimental]

Build a brain atlas from GIFTI label files (`.label.gii`). Assumes fsaverage5 surface space (10,242 vertices per hemisphere).

Usage

```
create_cortical_from_gifti(
  gifti_files,
  atlas_name = NULL,
  output_dir = NULL,
  hemisphere = c("rh", "lh"),
  views = c("lateral", "medial", "superior", "inferior"),
  tolerance = NULL,
  smooth_refinements = NULL,
  cleanup = NULL,
  verbose = get_verbose(),
  skip_existing = NULL
)
```

Arguments

`gifti_files`	Character vector of paths to `.label.gii` files. Hemisphere is detected from filename patterns (`lh.`, `rh.`, `.L.`, `.R.`).
`atlas_name`	Name for the atlas. If NULL, derived from the input filename.
`output_dir`	Directory to store intermediate files (screenshots, masks, contours). Defaults to `tempdir()`.
`hemisphere`	Which hemispheres to include: "lh", "rh", or both.
`views`	Which views to include: "lateral", "medial", "superior", "inferior".
`tolerance`	Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to `sf::st_simplify()`. If not specified, uses `options("ggseg.extra.tolerance")` or the `GGSEG_EXTRA_TOLERANCE` environment variable. Default is 1.
`smooth_refinements`	Number of Chaikin corner-cutting refinements to apply to 2D polygons. Higher values produce smoother region boundaries (typical range: 0–3). 0 disables smoothing. If not specified, uses `options("ggseg.extra.smooth_refinements")` or the `GGSEG_EXTRA_SMOOTH_REFINEMENTS` environment variable. Default is 2.
`cleanup`	Remove intermediate files after atlas creation. If not specified, uses `options("ggseg.extra.cleanup")` or the `GGSEG_EXTRA_CLEANUP` environment variable. Default is TRUE.

verbose	Verbosity level: 0 (silent), 1 (standard progress, default), or 2 (debug, includes FreeSurfer output). Logical values are accepted (TRUE = 1, FALSE = 0). If not specified, uses the value from options("ggseg.extra.verbose") or the GGSEG_EXTRA_VERBOSE environment variable.
skip_existing	Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses options("ggseg.extra.skip_existing") or the GGSEG_EXTRA_SKIP_EXISTING environment variable. Default is TRUE.

Value

A ggseg_atlas object.

Examples

```
## Not run:
atlas <- create_cortical_from_gifti(
  gifti_files = c("lh.aparc.label.gii", "rh.aparc.label.gii")
)

## End(Not run)
```

---

create_cortical_from_labels

Create brain atlas from label files

Description

[Experimental]

Build an atlas from individual FreeSurfer .label files rather than a complete annotation. Each label file defines a single region by listing which surface vertices belong to it.

The function detects hemisphere from filename prefixes (lh. or rh.) and derives region names from the rest of the filename.

Usage

```
create_cortical_from_labels(
  label_files,
  atlas_name = NULL,
  input_lut = NULL,
  output_dir = NULL,
  views = c("lateral", "medial"),
  tolerance = NULL,
  smooth_refinements = NULL,
  cleanup = NULL,
  verbose = get_verbose(),
  skip_existing = NULL
)
```

Arguments

label_files	Paths to .label files. Each file should follow FreeSurfer naming: {hemi}.{regionname}.label (e.g., lh.motor.label).
atlas_name	Name for the atlas. If NULL, derived from the input filename.
input_lut	Path to a color lookup table (LUT) file, or a data.frame with columns region and colour columns (R, G, B or hex).
output_dir	Directory to store intermediate files (screenshots, masks, contours). Defaults to tempdir().
views	Which views to include: "lateral", "medial", "superior", "inferior".
tolerance	Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to sf::st_simplify(). If not specified, uses options("ggseg.extra.tolerance") or the GGSEG_EXTRA_TOLERANCE environment variable. Default is 1.
smooth_refinements	Number of Chaikin corner-cutting refinements to apply to 2D polygons. Higher values produce smoother region boundaries (typical range: 0–3). 0 disables smoothing. If not specified, uses options("ggseg.extra.smooth_refinements") or the GGSEG_EXTRA_SMOOTH_REFINEMENTS environment variable. Default is 2.
cleanup	Remove intermediate files after atlas creation. If not specified, uses options("ggseg.extra.cleanup") or the GGSEG_EXTRA_CLEANUP environment variable. Default is TRUE.
verbose	Verbosity level: 0 (silent), 1 (standard progress, default), or 2 (debug, includes FreeSurfer output). Logical values are accepted (TRUE = 1, FALSE = 0). If not specified, uses the value from options("ggseg.extra.verbose") or the GGSEG_EXTRA_VERBOSE environment variable.
skip_existing	Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses options("ggseg.extra.skip_existing") or the GGSEG_EXTRA_SKIP_EXISTING environment variable. Default is TRUE.

Value

A ggseg_atlas object.

Examples

```
## Not run:
labels <- c("lh.region1.label", "lh.region2.label", "rh.region1.label")
atlas <- create_cortical_from_labels(labels)

## End(Not run)
```

`create_cortical_from_neuromaps` *Create cortical atlas from a neuromaps annotation*

Description

[Experimental]

Build a brain atlas directly from a neuromaps annotation. The annotation is downloaded via `neuromapr::fetch_neuromaps_annotation()`.

Supports both surface (`.func.gii`) and volume (`.nii/.nii.gz`) annotations. Volume annotations in MNI152 space are automatically projected to fsaverage5 via FreeSurfer's `mri_vol2surf`.

Usage

```
create_cortical_from_neuromaps(
  source,
  desc,
  space = "fsaverage",
  density = "10k",
  label_table = NULL,
  n_bins = NULL,
  atlas_name = NULL,
  output_dir = NULL,
  hemisphere = c("rh", "lh"),
  views = c("lateral", "medial", "superior", "inferior"),
  tolerance = NULL,
  smooth_refinements = NULL,
  cleanup = NULL,
  verbose = get_verbose(),
  skip_existing = NULL
)
```

Arguments

`source`	Neuromaps source identifier (e.g., `"schaefer"`).
`desc`	Neuromaps descriptor key (e.g., `"400Parcels7Networks"`).
`space`	Coordinate space. Defaults to `"fsaverage"`.
`density`	Surface vertex density. Defaults to `"10k"`.
`label_table`	Optional data.frame mapping parcel IDs to region names.
`n_bins`	Number of quantile bins for continuous brain maps.
`atlas_name`	Name for the atlas. If NULL, derived from the input filename.
`output_dir`	Directory to store intermediate files (screenshots, masks, contours). Defaults to `tempdir()`.
`hemisphere`	Which hemispheres to include: "lh", "rh", or both.

views — Which views to include: "lateral", "medial", "superior", "inferior".

tolerance — Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to `sf::st_simplify()`. If not specified, uses `options("ggseg.extra.tolerance")` or the `GGSEG_EXTRA_TOLERANCE` environment variable. Default is 1.

smooth_refinements — Number of Chaikin corner-cutting refinements to apply to 2D polygons. Higher values produce smoother region boundaries (typical range: 0–3). 0 disables smoothing. If not specified, uses `options("ggseg.extra.smooth_refinements")` or the `GGSEG_EXTRA_SMOOTH_REFINEMENTS` environment variable. Default is 2.

cleanup — Remove intermediate files after atlas creation. If not specified, uses `options("ggseg.extra.cleanup")` or the `GGSEG_EXTRA_CLEANUP` environment variable. Default is TRUE.

verbose — Verbosity level: `0` (silent), `1` (standard progress, default), or `2` (debug, includes FreeSurfer output). Logical values are accepted (`TRUE` = 1, `FALSE` = 0). If not specified, uses the value from `options("ggseg.extra.verbose")` or the `GGSEG_EXTRA_VERBOSE` environment variable.

skip_existing — Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses `options("ggseg.extra.skip_existing")` or the `GGSEG_EXTRA_SKIP_EXISTING` environment variable. Default is TRUE.

Value

A `ggseg_atlas` object.

Examples

```
## Not run:
atlas <- create_cortical_from_neuromaps(
  source = "abagen",
  desc = "genepc1",
  n_bins = 7
)

## End(Not run)
```

---

create_subcortical_from_volume

Create brain atlas from subcortical segmentation

Description

[Experimental]

Turn a subcortical segmentation volume (like `aseg.mgz`) into a brain atlas with 3D meshes for each structure. The function extracts each labelled region from the volume, creates a surface mesh, and smooths it.

For 2D plotting, the function can also generate slice views by taking snapshots at specified coordinates and extracting contours.

Requires FreeSurfer for mesh generation.

Usage

```
create_subcortical_from_volume(
  input_volume,
  input_lut = NULL,
  atlas_name = NULL,
  views = NULL,
  output_dir = NULL,
  vertex_size_limits = NULL,
  dilate = NULL,
  tolerance = NULL,
  smoothness = NULL,
  verbose = get_verbose(),
  cleanup = NULL,
  skip_existing = NULL,
  decimate = 0.5,
  steps = NULL
)
```

Arguments

input_volume	Path to the segmentation volume. Supports .mgz, .nii, and .nii.gz formats. Typically this is aseg.mgz or a custom segmentation in the same space.
input_lut	Path to a FreeSurfer-style colour lookup table that maps label IDs to region names and colours (e.g., FreeSurferColorLUT.txt or ASegStatsLUT.txt), or a data.frame with columns region and colour columns (R, G, B or hex). If NULL, region names will be generic (e.g., "region_0010") and the atlas will have no palette.
atlas_name	Name for the atlas. If NULL, derived from the input filename.
views	A data.frame specifying projection views with columns name, type ("axial", "coronal", "sagittal"), start (first slice), end (last slice). Default projects entire volume from each direction. Unlike slices, projections show ALL structures in their spatial relationships - like an X-ray view.
output_dir	Directory to store intermediate files (screenshots, masks, contours). Defaults to tempdir().
vertex_size_limits	Numeric vector of length 2 setting minimum and maximum vertex count for polygons. Polygons outside this range are filtered out. Default NULL applies no limits.
dilate	Dilation iterations for 2D polygons. Useful for filling small gaps between structures.
tolerance	Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to sf::st_simplify(). If not specified, uses options("ggseg.extra.tolerance") or the GGSEG_EXTRA_TOLERANCE environment variable. Default is 1.
smoothness	Smoothing factor for 2D contours. Higher values produce smoother region boundaries (typical range: 3–15). Passed to smoothr::smooth(). If not speci-

fied, uses `options("ggseg.extra.smoothness")` or the `GGSEG_EXTRA_SMOOTHNESS` environment variable. Default is 5.

`verbose` Verbosity level: `0` (silent), `1` (standard progress, default), or `2` (debug, includes FreeSurfer output). Logical values are accepted (`TRUE` = 1, `FALSE` = 0). If not specified, uses the value from `options("ggseg.extra.verbose")` or the `GGSEG_EXTRA_VERBOSE` environment variable.

`cleanup` Remove intermediate files after atlas creation. If not specified, uses `options("ggseg.extra.cleanup")` or the `GGSEG_EXTRA_CLEANUP` environment variable. Default is TRUE.

`skip_existing` Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses `options("ggseg.extra.skip_existing")` or the `GGSEG_EXTRA_SKIP_EXISTING` environment variable. Default is TRUE.

`decimate` Mesh decimation factor between 0 and 1. Reduces the number of faces in 3D meshes using quadric edge decimation (via `Rvcg::vcgQEdecim()`). A value of 0.5 reduces faces by 50%. Set to NULL to skip decimation. Requires the Rvcg package. Default is 0.5.

`steps` Which pipeline steps to run. Default NULL runs all steps. Steps are:

- 1: Extract labels from volume and get colour table
- 2: Create meshes for each structure
- 3: Build atlas data (3D only if stopping here)
- 4: Create projection snapshots
- 5: Process images
- 6: Extract contours
- 7: Smooth contours
- 8: Reduce vertices
- 9: Build final atlas with 2D geometry

Use `steps = 1:3` for 3D-only atlas. Use `steps = 7:8` to iterate on smoothing and reduction parameters.

Value

A `ggseg_atlas` object with region metadata (core), 3D meshes, a colour palette, and optionally sf geometry for 2D slice plots.

Examples

```
## Not run:
# Create 3D-only subcortical atlas from aseg
atlas <- create_subcortical_from_volume(
  input_volume = "path/to/aseg.mgz",
  input_lut = "path/to/FreeSurferColorLUT.txt",
  steps = 1:3
)

# View with ggseg3d
ggseg3d(atlas = atlas, hemisphere = "subcort")

# Full atlas with 2D slices
```

```
atlas <- create_subcortical_from_volume(
  input_volume = "path/to/aseg.mgz",
  input_lut = "path/to/ASegStatsLUT.txt"
)

# Post-process to remove/modify regions (functions from ggseg.formats)
atlas <- atlas |>
  atlas_region_remove("White-Matter") |>
  atlas_region_contextual("Cortex")

## End(Not run)
```

`create_tract_from_tractography` *Create brain atlas from white matter tracts*

Description

[Experimental]

Turn tractography streamlines into a brain atlas where each tract is rendered as a 3D tube. The function computes a centerline from the streamlines and generates a tube mesh around it.

You can provide tract data in several formats: TRK files from TrackVis, TCK files from MRtrix, or coordinate matrices directly in R. The function reads the streamlines, extracts a representative centerline (by averaging or selecting the medoid), and builds a tube mesh for 3D rendering.

For tracts with many streamlines, set `tube_radius = "density"` to make the tube thicker where more streamlines pass through.

Usage

```
create_tract_from_tractography(
  input_tracts,
  input_aseg = NULL,
  atlas_name = NULL,
  input_lut = NULL,
  tube_radius = 5,
  tube_segments = 8,
  n_points = 50,
  centerline_method = c("mean", "medoid"),
  views = NULL,
  output_dir = NULL,
  verbose = get_verbose(),
  tolerance = NULL,
  smoothness = NULL,
  cleanup = NULL,
  skip_existing = NULL,
  dilate = NULL,
  vertex_size_limits = NULL,
```

```
  steps = NULL
)
```

Arguments

input_tracts	Paths to tractography files (.trk or .tck), or a named list of coordinate matrices where each matrix has N rows and 3 columns (x, y, z).
input_aseg	Path to a segmentation volume (.mgz, .nii) used to draw cortex outlines in 2D views. Required for steps 2+.
atlas_name	Name for the atlas. If NULL, derived from the input filename.
input_lut	Path to a color lookup table (LUT) file, or a data.frame with columns region and colour columns (R, G, B or hex). Use this to provide tract names and colours. If NULL, names are derived from filenames or list names, and the atlas will have no palette.
tube_radius	Controls the tube thickness. Either a single numeric value for uniform radius, or "density" to scale radius by how many streamlines pass through each point.
tube_segments	Number of segments around the tube circumference. Higher values make smoother tubes but larger meshes. Default 8 is a good balance.
n_points	Number of points to resample the centerline to. All tracts are resampled to this length for consistent tube generation.
centerline_method	How to extract the centerline from multiple streamlines: "mean" averages coordinates point-by-point, "medoid" selects the single most representative streamline.
views	A data.frame specifying projection views. If NULL, uses default_tract_views().
output_dir	Directory to store intermediate files (screenshots, masks, contours). Defaults to tempdir().
verbose	Verbosity level: 0 (silent), 1 (standard progress, default), or 2 (debug, includes FreeSurfer output). Logical values are accepted (TRUE = 1, FALSE = 0). If not specified, uses the value from options("ggseg.extra.verbose") or the GGSEG_EXTRA_VERBOSE environment variable.
tolerance	Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to sf::st_simplify(). If not specified, uses options("ggseg.extra.tolerance") or the GGSEG_EXTRA_TOLERANCE environment variable. Default is 1.
smoothness	Smoothing factor for 2D contours. Higher values produce smoother region boundaries (typical range: 3–15). Passed to smoothr::smooth(). If not specified, uses options("ggseg.extra.smoothness") or the GGSEG_EXTRA_SMOOTHNESS environment variable. Default is 5.
cleanup	Remove intermediate files after atlas creation. If not specified, uses options("ggseg.extra.cleanup") or the GGSEG_EXTRA_CLEANUP environment variable. Default is TRUE.
skip_existing	Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses options("ggseg.extra.skip_existing") or the GGSEG_EXTRA_SKIP_EXISTING environment variable. Default is TRUE.

`dilate` Dilation iterations for 2D polygons. Useful for filling small gaps between structures.

`vertex_size_limits`
Numeric vector of length 2 setting minimum and maximum vertex count for polygons. Polygons outside this range are filtered out. Default NULL applies no limits.

`steps` Which pipeline steps to run. Default NULL runs all steps. Steps are:

- 1: Read tractography and create tube meshes
- 2: Create projection snapshots
- 3: Process images
- 4: Extract contours
- 5: Smooth contours
- 6: Reduce vertices
- 7: Build atlas

Use `steps = 1` for 3D-only atlas. Use `steps = 5:7` to iterate on smoothing and vertex reduction.

Value

A `ggseg_atlas` object with type `"tract"`, containing region metadata, tube meshes for 3D rendering, colours, and optionally sf geometry for 2D projection plots.

Examples

```
## Not run:
# From TRK files (names derived from filenames)
atlas <- create_tract_from_tractography(
  input_tracts = c("cst_left.trk", "cst_right.trk")
)

# With custom names and colours via LUT
atlas <- create_tract_from_tractography(
  input_tracts = c("cst_left.trk", "cst_right.trk"),
  input_lut = "tract_colors.txt"
)

# View with ggseg3d
ggseg3d(atlas = atlas)

## End(Not run)
```

`create_wholebrain_from_volume`
Create atlas from whole-brain volumetric parcellation

Description

[Experimental]

Build a brain atlas from a single volumetric parcellation (NIfTI/MGZ) that contains both cortical and subcortical regions. Cortical regions are projected onto the fsaverage5 surface via FreeSurfer's `mri_vol2surf` and rendered as surface views, while subcortical regions go through the mesh-based subcortical pipeline.

Requires FreeSurfer.

Usage

```
create_wholebrain_from_volume(
  input_volume,
  input_lut = NULL,
  atlas_name = NULL,
  output_dir = NULL,
  projfrac = 0.5,
  projfrac_range = c(0, 1, 0.1),
  subject = "fsaverage5",
  regheader = TRUE,
  min_vertices = 50L,
  cortical_labels = NULL,
  subcortical_labels = NULL,
  cortical_views = c("lateral", "medial", "superior", "inferior"),
  subcortical_views = NULL,
  decimate = 0.5,
  tolerance = NULL,
  smoothness = NULL,
  cleanup = NULL,
  verbose = get_verbose(),
  skip_existing = NULL,
  steps = NULL
)
```

Arguments

`input_volume`	Path to volumetric parcellation in MNI152 space (.mgz, .nii, .nii.gz).
`input_lut`	Path to FreeSurfer-style colour lookup table, or a data.frame with columns `idx`, `label`, `R`, `G`, `B`, `A`. An optional `type` column with values `"cortical"` or `"subcortical"` controls label classification (see **Label classification**). Voxel IDs not listed in the LUT are automatically zeroed out before surface projection (see **Volume pre-processing**). If NULL, generic names and no palette.
`atlas_name`	Name for the atlas. If NULL, derived from the input filename.
`output_dir`	Directory to store intermediate files (screenshots, masks, contours). Defaults to `tempdir()`.
`projfrac`	Cortical depth fraction for projection (0 = white surface, 1 = pial surface). Only used when `projfrac_range` is NULL. Default 0.5.

`projfrac_range` Numeric vector `c(min, max, delta)` for multi-depth projection via `mri_vol2surf --projfrac-max`. Samples at multiple cortical depths and takes the maximum label value at each vertex, giving much better surface coverage than single-depth projection. Default `c(0, 1, 0.1)`. Set to NULL to use single-depth `projfrac` instead.

`subject` Target surface subject. Default "fsaverage5".

`regheader` If TRUE (default), assumes volume RAS coordinates match the subject space and uses `--regheader`. Works well for standard MNI152-space volumes. If FALSE, uses FreeSurfer's `--mni152reg` registration (may produce noisy results due to surf2surf resampling).

`min_vertices` Minimum total vertex count across hemispheres for a label to be classified as cortical by the vertex-count heuristic (see **Label classification**). Ignored when `type` column or explicit label vectors are provided. Default 50.

`cortical_labels`
Character vector of label names to force as cortical. Highest priority; overrides LUT `type` and the vertex-count heuristic.

`subcortical_labels`
Character vector of label names to force as subcortical. Highest priority; overrides LUT `type` and the vertex-count heuristic.

`cortical_views` Views for cortical sub-pipeline. Default `c("lateral", "medial", "superior", "inferior")`.

`subcortical_views`
Views for subcortical sub-pipeline. Default NULL (auto-detected).

`decimate` Mesh decimation ratio for subcortical meshes (0-1). Default 0.5.

`tolerance` Simplification tolerance for 2D polygons. Higher values produce simpler shapes with fewer vertices (typical range: 0.1–2). Passed to `sf::st_simplify()`. If not specified, uses `options("ggseg.extra.tolerance")` or the `GGSEG_EXTRA_TOLERANCE` environment variable. Default is 1.

`smoothness` Smoothing factor for 2D contours. Higher values produce smoother region boundaries (typical range: 3–15). Passed to `smoothr::smooth()`. If not specified, uses `options("ggseg.extra.smoothness")` or the `GGSEG_EXTRA_SMOOTHNESS` environment variable. Default is 5.

`cleanup` Remove intermediate files after atlas creation. If not specified, uses `options("ggseg.extra.cleanup")` or the `GGSEG_EXTRA_CLEANUP` environment variable. Default is TRUE.

`verbose` Verbosity level: `0` (silent), `1` (standard progress, default), or `2` (debug, includes FreeSurfer output). Logical values are accepted (`TRUE` = 1, `FALSE` = 0). If not specified, uses the value from `options("ggseg.extra.verbose")` or the `GGSEG_EXTRA_VERBOSE` environment variable.

`skip_existing` Skip generating output files that already exist, allowing interrupted atlas creation to resume. If not specified, uses `options("ggseg.extra.skip_existing")` or the `GGSEG_EXTRA_SKIP_EXISTING` environment variable. Default is TRUE.

`steps` Which pipeline steps to run. Default NULL runs all steps. Steps are:

- 1: Project volume onto surface
- 2: Split labels into cortical/subcortical

- 3: Run cortical pipeline
- 4: Run subcortical pipeline

Use `steps = 1:2` to run projection and split only.

Value

A named list with elements `cortical` and `subcortical`, each a `ggseg_atlas` object (or NULL if no regions of that type exist).

Label classification

The pipeline must know which labels are cortical (rendered on the surface) and which are subcortical (rendered as 3D meshes / 2D slices). Three mechanisms are available, applied in priority order:

1. **Function arguments** (highest priority): `cortical_labels` and `subcortical_labels` override everything for the specified labels.
2. **LUT `type` column**: If the colour lookup table has a `type` column with values `"cortical"` or `"subcortical"`, that classification is used for any labels not covered by the function arguments. This is the recommended approach for reproducible atlas creation.
3. **Vertex-count heuristic** (fallback): Labels with at least `min_vertices` vertices on the surface projection are classified as cortical; the rest as subcortical.

Volume pre-processing

Before surface projection, the volume is filtered so that only voxel IDs listed in the LUT are kept; all other non-zero voxels are zeroed out. This prevents unlisted structures (e.g. white matter, ventricle masks) from bleeding onto the cortical surface during label dilation.

Cortical voxels are also used to generate the brain-outline reference geometry for the subcortical pipeline. The volume's orientation matrix (`xform`) is used to split cortical voxels by hemisphere: left-hemisphere voxels map to FreeSurfer label 3 (left cortex) and right-hemisphere to label 42 (right cortex).

Human oversight

This is the most complex pipeline in ggsegExtra and the one most likely to need manual correction. Recommended workflow:

1. Run `steps = 1:2` first to project the volume and classify labels.
2. Inspect `result$cortical_labels` and `result$subcortical_labels`. If the automatic split is wrong, either add a `type` column to the LUT or use `cortical_labels` / `subcortical_labels` to override.
3. Run the full pipeline once you are satisfied with the split.
4. Visually inspect the resulting atlas with `ggseg()` / `ggseg3d()`.

The cortical surface projection uses FreeSurfer's cortex label (`{hemi}.cortex.label`) to prevent label dilation into the medial wall. This file ships with fsaverage5 and is always required.

Examples

```
## Not run:
# --- Recommended: LUT with type column ---
lut <- data.frame(
  idx   = c(10, 11, 49, 50, 101:148),
  label = c("Left-Thalamus", "Left-Caudate",
            "Right-Thalamus", "Right-Caudate",
            paste0("cortical_region_", 1:48)),
  type  = c(rep("subcortical", 4), rep("cortical", 48)),
  R = sample(50:220, 52, TRUE),
  G = sample(50:220, 52, TRUE),
  B = sample(50:220, 52, TRUE),
  A = 255L
)

result <- create_wholebrain_from_volume(
  input_volume = "my_atlas.nii.gz",
  input_lut = lut,
  atlas_name = "my_atlas"
)
result$cortical   # surface-based cortical atlas
result$subcortical # mesh-based subcortical atlas

# --- Without type column: automatic classification ---
result <- create_wholebrain_from_volume(
  input_volume = "atlas.nii.gz",
  input_lut = "atlas_LUT.txt",
  atlas_name = "auto_atlas"
)

# --- Inspect classification before full run ---
result <- create_wholebrain_from_volume(
  input_volume = "atlas.nii.gz",
  input_lut = "atlas_LUT.txt",
  steps = 1:2
)
result$cortical_labels
result$subcortical_labels

## End(Not run)
```

---

get_ctab *Read color table and add hex colours*

Description

Reads a FreeSurfer color lookup table and adds hex colour codes for use in plotting.

Usage

```
get_ctab(color_lut)
```

Arguments

color_lut	Path to a color table file, or a data.frame that passes is_ctab().

Value

A data.frame with the original columns plus roi (zero-padded index) and color (hex colour code).

See Also

read_ctab(), is_ctab()

Examples

```
ct <- data.frame(
  idx = 0:1, label = c("Unknown", "Region1"),
  R = c(0L, 205L), G = c(0L, 130L), B = c(0L, 176L), A = c(0L, 0L)
)
get_ctab(ct)
```

---

get_verbose *Get verbose setting*

Description

Returns the verbosity level from option, environment variable, or default. Checks in order: ggseg.extra.verbose option, GGSEG_EXTRA_VERBOSE env var, then defaults to 1L.

Usage

```
get_verbose()
```

Details

Verbosity levels:

- 0 — Silent: no console output
- 1 — Standard (default): pipeline progress and step summaries
- 2 — Debug: includes FreeSurfer command output

Logical values are accepted for backward compatibility (FALSE = 0, TRUE = 1).

Value

Integer 0L, 1L, or 2L

Examples

```
get_verbose()
options(ggseg.extra.verbose = 0)
get_verbose()
options(ggseg.extra.verbose = NULL)
```

---

is_ctab *Check if object is a color table*

Description

Check if object is a color table

Usage

```
is_ctab(x)
```

Arguments

x	Object to check.

Value

TRUE if x is a data.frame with the required color table columns.

Examples

```
ct <- data.frame(
  idx = 0L, label = "Unknown",
  R = 0L, G = 0L, B = 0L, A = 0L
)
is_ctab(ct)
is_ctab(data.frame(x = 1))
```

---

is_verbose *Get verbosity level*

Description

Get verbosity level

Usage

```
is_verbose(verbose = NULL)
```

Arguments

verbose	Optional explicit value. If NULL, reads from option/env via get_verbose(). Accepts logical or integer (0/1/2).

Value

Integer 0L, 1L, or 2L

Examples

```
is_verbose()
is_verbose(FALSE)
is_verbose(2)
```

---

mri_surf2surf_rereg *Re-register an annotation file*

Description

Annotation files are subject specific. Most are registered for fsaverage, but we recommend using fsaverage5 for the mesh plots in ggseg3d, as these contain a decent balance in number of vertices for detailed rendering and speed.

Usage

```
mri_surf2surf_rereg(
  subject,
  annot,
  hemi = c("lh", "rh"),
  target_subject = "fsaverage5",
  output_dir = file.path(freesurfer::fs_subj_dir(), subject, "label"),
  verbose = get_verbose()
)
```

Arguments

subject	subject the original annotation file is registered to
annot	annotation file name (as found in subjects_dir)
hemi	hemisphere (one of "lh" or "rh")
target_subject	subject to re-register the annotation (default fsaverage5)
output_dir	Directory to store intermediate files (screenshots, masks, contours). Defaults to tempdir().
verbose	Verbosity level: 0 (silent), 1 (standard progress, default), or 2 (debug, includes FreeSurfer output). Logical values are accepted (TRUE = 1, FALSE = 0). If not specified, uses the value from options("ggseg.extra.verbose") or the GGSEG_EXTRA_VERBOSE environment variable.

Value

nothing

Examples

```
## Not run:
# For help see:
freesurfer::fs_help("mri_surf2surf")

mri_surf2surf_rereg(
  subject = "bert",
  annot = "aparc.DKTatlas",
  target_subject = "fsaverage5"
)

## End(Not run)
```

---

read_annotation_data *Read annotation data from files*

Description

Reads FreeSurfer annotation files and extracts region information including vertices, colours, and labels for both hemispheres.

Usage

```
read_annotation_data(annot_files)
```

Arguments

annot_files	Character vector of paths to annotation files. Files should follow FreeSurfer naming convention with `lh.` or `rh.` prefix (e.g., `c("lh.aparc.annot", "rh.aparc.annot")`).

Value

A tibble with columns: hemi, region, label, colour, vertices

Examples

```
## Not run:
atlas_data <- read_annotation_data(c(
  "path/to/lh.aparc.annot",
  "path/to/rh.aparc.annot"
))

## End(Not run)
```

read_cifti_annotation *Read CIFTI annotation file*

Description

Reads a CIFTI dense label file (`.dlabel.nii`) and extracts region information for both hemispheres. Returns data in the same format as `read_annotation_data()` for use with the cortical atlas pipeline.

Usage

```
read_cifti_annotation(cifti_file)
```

Arguments

`cifti_file`	Path to a `.dlabel.nii` CIFTI file.

Details

The CIFTI file must be in fsaverage5 space (10,242 vertices per hemisphere). If your file uses a different resolution, resample it first with Connectome Workbench:

```
wb_command -cifti-resample input.dlabel.nii ...
```

Value

A tibble with columns: hemi, region, label, colour, vertices

Examples

```
## Not run:
atlas_data <- read_cifti_annotation("parcellation.dlabel.nii")

## End(Not run)
```

read_ctab *Read FreeSurfer color table*

Description

Read a FreeSurfer color lookup table file (e.g., `FreeSurferColorLUT.txt` or `ASegStatsLUT.txt`). These files map label indices to region names and RGBA colours.

Usage

```
read_ctab(path)
```

Arguments

path	Path to the color table file.

Value

A data.frame with columns: idx, label, R, G, B, A.

See Also

get_ctab() to read and add hex colours, write_ctab() to write

Examples

```
ctab_file <- tempfile()
writeLines(c(
  "  0  Unknown                    0   0   0   0",
  "  1  Left-Cerebral-Cortex      205 130 176   0"
), ctab_file)
read_ctab(ctab_file)
```

---

read_gifti_annotation *Read GIFTI annotation files*

Description

Reads GIFTI annotation (.label.gii) files and extracts region information including vertices, colours, and labels. Returns data in the same format as read_annotation_data() for use with the cortical atlas pipeline.

Usage

```
read_gifti_annotation(gifti_files)
```

Arguments

gifti_files	Character vector of paths to .label.gii files.

Details

Hemisphere is detected from filename patterns: lh., rh., .L., .R.

Value

A tibble with columns: hemi, region, label, colour, vertices

Examples

```
## Not run:
atlas_data <- read_gifti_annotation(c(
  "lh.aparc.label.gii",
  "rh.aparc.label.gii"
))

## End(Not run)
```

`read_neuromaps_annotation` *Read neuromaps annotation files*

Description

Reads neuromaps GIFTI metric files (`.func.gii`) and converts them to the standard annotation format used by the cortical atlas pipeline.

Usage

```
read_neuromaps_annotation(gifti_files, label_table = NULL, n_bins = NULL)
```

Arguments

`gifti_files`	Character vector of paths to `.func.gii` files. Hemisphere is detected from BIDS filename patterns (`hemi-L`, `hemi-R`).
`label_table`	Optional data.frame mapping integer parcel IDs to region names. Must have columns `id` (integer) and `region` (character). Optionally include `colour` (hex string). When `NULL`, regions are named `parcel_1`, `parcel_2`, etc. (parcellation) or `bin_1`, `bin_2`, etc. (continuous).
`n_bins`	Number of quantile bins for continuous data. When `NULL` (default), auto-detected via Sturges' rule (`1 + log2(n)`, clamped to 5–20). Ignored for integer parcellation data.

Details

Automatically detects whether data contains integer parcel IDs (parcellation) or continuous values (brain map). For parcellations, vertex value 0 is treated as medial wall. For continuous data, NaN vertices are medial wall and values are discretized into quantile bins via `n_bins`.

Files must be in fsaverage5 space (10,242 vertices per hemisphere). Use `space = "fsaverage"` with `density = "10k"` when fetching from neuromaps.

Value

A tibble with columns: hemi, region, label, colour, vertices

Examples

```
## Not run:
files <- neuromapr::fetch_neuromaps_annotation(
  "abagen", "genepc1", "fsaverage", density = "10k"
)
atlas_data <- read_neuromaps_annotation(files, n_bins = 7)

## End(Not run)
```

`read_neuromaps_volume` *Read neuromaps volume annotation via surface projection*

Description

Projects an MNI152-space NIfTI volume onto the fsaverage5 surface via FreeSurfer's `mri_vol2surf`, then discretizes the projected per-vertex values using the same binning logic as `read_neuromaps_annotation()`.

Usage

```
read_neuromaps_volume(nifti_file, n_bins = NULL, output_dir = tempdir())
```

Arguments

`nifti_file`	Path to a `.nii` or `.nii.gz` file in MNI152 space.
`n_bins`	Number of quantile bins for continuous data. When `NULL` (default), auto-detected via Sturges' rule. Ignored for integer data.
`output_dir`	Directory for intermediate surface overlay files.

Value

A tibble with columns: hemi, region, label, colour, vertices

Examples

```
## Not run:
atlas_data <- read_neuromaps_volume("map.nii.gz", n_bins = 7)

## End(Not run)
```

read_tractography *Read tractography file*

Description

Load streamlines from a tractography file. Supports TrackVis (`.trk`) and MRtrix (`.tck`) formats. The file format is detected from the extension.

Usage

```
read_tractography(file)
```

Arguments

`file`	Path to a `.trk` or `.tck` file.

Value

A list of matrices, one per streamline. Each matrix has N rows (points along the streamline) and 3 columns (x, y, z coordinates).

See Also

`read_trk()`, `read_tck()` for format-specific readers

Examples

```
## Not run:
streamlines <- read_tractography("bundle.trk")

## End(Not run)
```

setup_atlas_repo *Create a new ggseg atlas package*

Description

Scaffold an R package for distributing a brain atlas. The generated package follows ggseg conventions and includes everything you need: template scripts for atlas creation, documentation stubs, a test suite, and GitHub Actions for automated checking.

Usage

```
setup_atlas_repo(
  path,
  atlas_name = NULL,
  open = rlang::is_interactive(),
  rstudio = TRUE
)
```

Arguments

path	Where to create the package. If the directory exists, it must be empty.
atlas_name	Name of the atlas (lowercase, no spaces). The package name becomes ggseg{AtlasName}. If NULL, derived from the directory name (e.g., path ggsegDkt becomes atlas name dkt).
open	If TRUE, opens the new project in RStudio. Default is TRUE when running interactively.
rstudio	If TRUE, creates an .Rproj file for RStudio users.

Details

The package will be named ggseg{AtlasName} (e.g., ggsegSchaefer for a Schaefer parcellation). After creation, edit the files in data-raw/ to build your atlas, then run devtools::document() and devtools::check().

Value

Invisibly returns the path to the created package.

Examples

```
## Not run:
# Create atlas package in a new directory
setup_atlas_repo("ggsegDkt", "dkt")

# Create in current directory, derive name from path
setup_atlas_repo("ggsegMyatlas")

# Specify full path
setup_atlas_repo("~/projects/ggsegSchaefer", "schaefer")

# Without opening in RStudio
setup_atlas_repo("ggsegHarvard", "harvard", open = FALSE)

## End(Not run)
```

setup_sitrep *Check ggseg.extra setup status*

Description

Performs diagnostic checks to verify that system dependencies and environment variables required by ggseg.extra are properly configured.

Usage

```
setup_sitrep(detail = c("simple", "full"))
```

Arguments

detail	Character. Level of detail to display: • "simple" (default): Quick pass/fail overview • "full": Detailed diagnostics including freesurfer::fs_sitrep()

Value

Invisibly returns a list with check results.

Examples

```
setup_sitrep()
setup_sitrep("full")
```

write_ctab *Write FreeSurfer color table*

Description

Write a color table to file in FreeSurfer format.

Usage

```
write_ctab(x, path)
```

Arguments

x	A data.frame with columns: idx, label, R, G, B, A.
path	Path to write to.

Value

Invisibly returns the lines written.

See Also

read_ctab(), is_ctab()

Examples

```
ct <- data.frame(
  idx = 0:1, label = c("Unknown", "Region1"),
  R = c(0L, 205L), G = c(0L, 130L), B = c(0L, 176L), A = c(0L, 0L)
)
out <- tempfile()
write_ctab(ct, out)
```

Index