Visualisation of Brain Statistics with R-package `ggseg`

Athanasia M. Mowinckel[1] **and Didac Vidal Piñeiro**[1]

[1]**Center for Lifespan Changes in Brain and Cognition, Univeristy of Oslo, PO. box 1094 Blindern, 0317 Oslo, Norway**

Corresponding author:
Athanasia M. Mowinckel[1]

Email address: a.m.mowinckel@psykologi.uio.no

ABSTRACT

The abstract of the article. It can also be on *multiple* lines.

1 INTRODUCTION

Neuroscientific analysis usually requires the use of multiple tools to analyse, visualise, and summarise data. This often makes the process of preparing results for publication laborious, as results must be exported and imported in various formats. In neuroimaging, images of probands' brains are collected and merged together to provide three dimensional representations. Much of the neuroimaging analyses are not necessarily done on the individual voxel level (3-dimensional pixels), but rather on pre-defined brain segmentations, called brain atlases. These brain atlases are plentiful and represent different ways of segmenting the brain into functionally or structurally similar regions (REF). The use of these is wide-spread, as these atlases provide larger meaningful divisions of the brain (REF). While neuroimaging analyses on the voxel-level are usually computed by special software for such analyses, analyses of brain atlas data is usually done in standard statistical software, like R (REF), python (REF), or Matlab (REF).

A key part of understanding and disseminating analysis results, is the visualization of these in a meaningful way. With regards to results from brain atlas analyses, it is most meaningfully represented if projected onto a representation of the brain, rather than other common types of charts (like bar-charts) accompanied by a diagram of the brain label positions. Each atlas has its own labelling depending on what is meaningful for the type of segmentation it is based on. As such, for the reader to fully understand a bar chart with atlas labels, they need to be very familiar with the location of each label to have a clear comprehension of the results. A projection directly onto a brain shape, eases the readability of the results for the reader, and provides clear point of reference even if the used atlas is unfamiliar.

There are several tools that aid R-users in plotting neuroimaging data directly through R using the grammar of graphics as implemented in ggplot2 (REF), such as ggBrain(REF) and ggneuro(REF)(see neuroconductor for compiled neuroimaging packages for R). These are based on plotting imaging files, not results from analyses of brain atlases. Here we introduce the ggseg-package for visualizing results from brain atlas analyses. The ggseg-package was developed to create templates that others might use to project their brain atlas results on to. While its plotting functions are what the users are drawn to, it is the pre-compiled number of data sets for different brain atlases that provides the real functionality needed for this visualization. The package also includes functions to plot atlases in tri-surface mesh plots using plotly (REF).

1.1 Brain atlas selection

There is a multitude of brain atlases to chose from, all with different purposes and segmented based on different criteria. We have initially focused on commonly used and well-established atlases with manageable numbers of segmentations. The ggseg-package thus features data for two well-established parcellations, as implemented in the neuroimaging software Freesurfer (REF); dkt and aseg. The Desikan-Killany cortical atlas (dkt) (REF) and the Automatic Segmentation of Subcortical Structures (aseg) (REF)

Table 1. Table of currently available atlases in either ggseg or ggsegExtra R-package. Most atlases have both polygon and mesh atlases, but the mesh atlases are somewhat easier to create and are thus more plentiful.

Package	Title	Item	Mesh	Polygon
ggseg	Desikan-Killiany Cortical Atlas	dkt	No	Yes
	Desikan-Killiany Cortical Atlas - tri-surf mesh Mesh data for the Desikan-Killany Cortical atlas, with 40 regions in on the cortical surface of the brain.	dkt	Yes	No
	Freesurfer automatic subcortical segmentation of a brain volume	aseg	Yes	Yes
ggsegExtra	Desterieux cortical parcellations	desterieux	Yes	No
	Harvard-Oxford Cortical atlas	hoCort	No	Yes
	Parcellation from a midsagittal slice	midsagittal	No	Yes
	Parcellation from JHU	jhu	No	Yes
	Parcellation from of JHU	jhu	Yes	No
	Parcellation from of white matter	icbm	Yes	No
	Parcellation from of white matter	tracula	Yes	Yes
	Parcellation from the Human Connectome Project	glasser	Yes	Yes
	Schaefer 17 Resting-state Cortical Parcellations	schaefer17	Yes	No
	Schaefer 7 Resting-state Cortical Parcellations	schaefer7	Yes	No
	Yeo 17 Resting-state Cortical Parcellations	yeo17	Yes	Yes
	Yeo 7 Resting-state Cortical Parcellations	yeo7	Yes	Yes

together cover all grey matter of the brain, and are commonly used in analyses elucidating the associations between thickness, volume, or area of these segments and mental health and cognitive function.

While functions and base data can be found in the main `ggseg`-package, a companion package called `ggsegExtra` has numerous other atlases, and is continuously expanded. We encourage the community to help build and expand the atlases available, and to create a repository of established and often used atlases. A summary of all available atlases may be found in Table 1, where in addition to the two default atlases, ggsegExtra has another 12 atlases. The ggseg wiki currently has two different ways you may create and supply atlases to the package as 2D polygons. Work is being done to include instructions on 3D-mesh plot contributions also. The package includes functions like `as_ggseg_atlas()` and `as_ggseg3d_atlas()` to aid users in testing and making custom atlases.

2 TUTORIAL

The aim of this tutorial is to familiarize the reader with the main package functions and the general use of the package.
It will cover the two main functions: `ggseg()` which plots 2D polygons, and `ggseg3d()` which plots tri-surface mesh plots.

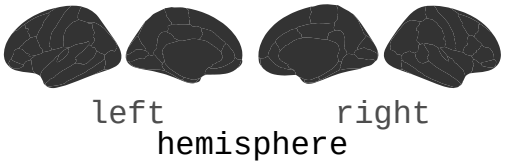


Figure 1. By default `ggseg` will plot the dkt atlas in shaded polygons.

2.1 Plotting polygon data

The function to plot two-dimensional polygon data is the `ggseg`-function. This function will automatically plot the dkt-atlas by default (see Figure 1). As this function is a wrapper for `geom_polygon` from `ggplot2` it can be built upon like all ggplot objects. The plot is kept as simple as possible, with as little extra information to ggplot as possible. This is done for it to be easy to extend the plot with the many ggplot options available.

```
library(ggseg)
library(tidyverse)
ggseg()
```

Because of the way `ggseg` is implemented, you should be able to safely use the function together with other ggplot functions, like themes, scales, and such (see Figure 2). There are many extra options to ggseg, to alter the plot as you wish.

```
p1 <- ggseg(position = "stacked") +
  theme_dark() +
  labs(title="dkt", subtitle = "dark theme")
p2 <- ggseg(position = "stacked") +
  theme_classic() +
  labs(title="dkt", subtitle = "classic theme")

# Combine plots
pp <- cowplot::plot_grid(p1, p2, nrow = 2)

p3 <- ggseg(atlas=aseg) +
  labs(title="aseg", subtitle = "default theme")

# Create a subplot grid
cowplot::plot_grid(pp, p3, nrow = 1)
```

You can stack the hemispheres, view only the medial or lateral side, choose either or both of the hemispheres, or a combination of any of these (see Figure 3). The possibilities of altering the plot depends on the atlas used. There are generally more options available for cortical atlases than subcortical ones, as the atlases have more meaningful ways of viewing the polygons. Subcortical atlases also tend to have structures that span the hemispheres or other typically used referece points of the brain, and as such it is more difficult to label all segments as either, e.g., left or right.

```
lat <- ggseg(view = "medial") +
  labs(title = "Medial view")
left <- ggseg(hemisphere = "left") +
  labs(title = "Left hemi.")
pp <- cowplot::plot_grid(lat, left, nrow=2)

combo <- ggseg(view = "medial",
       hemisphere = "left") +
  labs(title = "Combination")
```

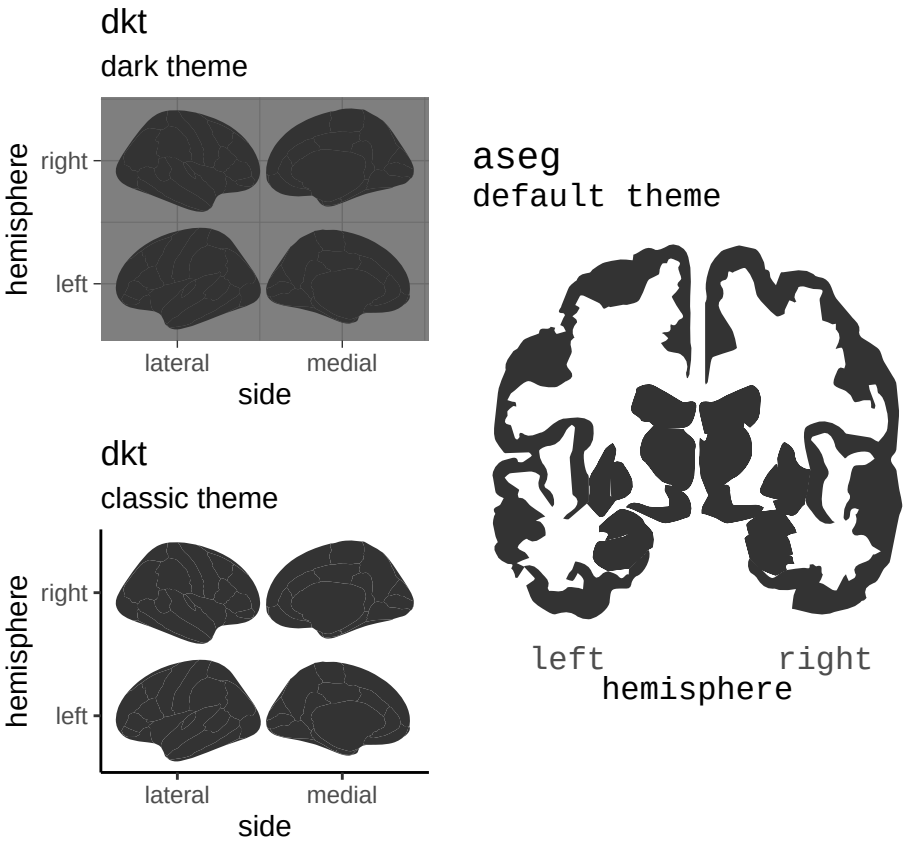


Figure 2. ggseg plots can be used with most standard scales, themes and such that work with ggplot. The special ggseg options for hemispheres, view etc. depend on the atlas used, and some options are only available for certain atlases. There is no 'lateral' or 'medial' views of subcortical atlases.

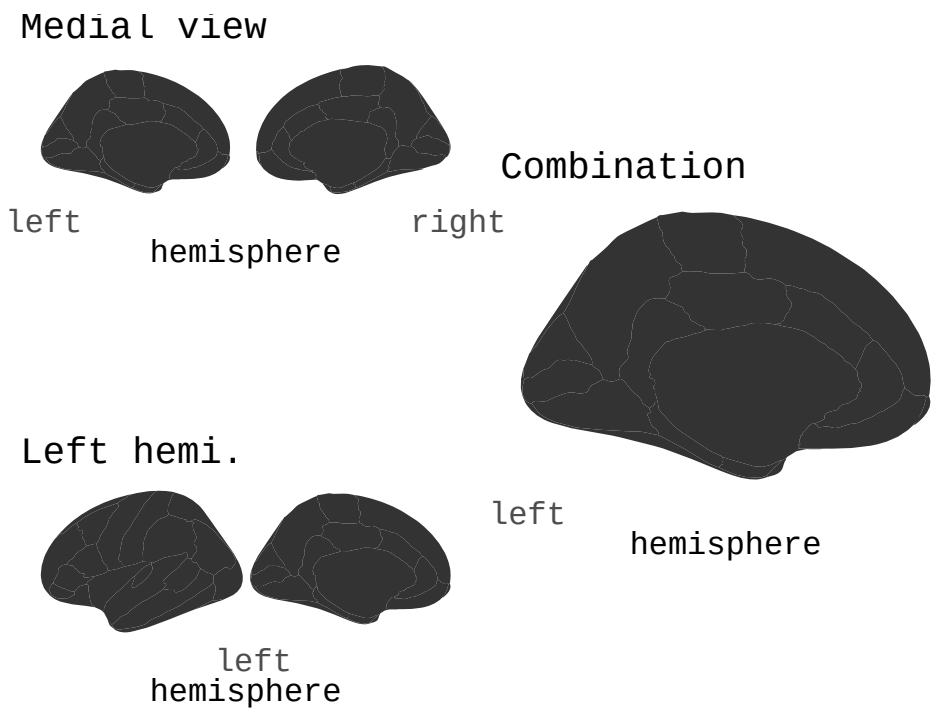


Figure 3. There are several ggseg-special options that may be supplied to control how the plots looks and is organised. Top left: plot using medial view. Bottom left: plot using left hemisphere. Right: plot combining both the options.

```
cowplot::plot_grid(pp, combo, nrow=1)
```

2.1.1 Using own data with fill and colour

`ggseg` accepts any argument you can supply to `geom_polygon` and therefore is easy to work with for those familiar with ggplot functionality. Standard arguments like `fill` to flood the segments with a colour, or `colour` to colour the edges around the segments are typical arguments to provide to the function either as a single value or within the ggplot mapping option `aes`.

```
ggseg(mapping=aes(fill = area), colour="black") +
  scale_fill_brain("dkt") +
  theme(legend.justification=c(1,0),
        legend.position="bottom",
        legend.text = element_text(size = 5)) +
  guides(fill = guide_legend(ncol = 3))
```

Typically you would want each area to have a different colour, corresponding to some descriptive or inference statistic that has been computed. The bulk of the package originates not in its functions, but in the datasets that accompany the package. There is one dataset per atlas, which contains key information regarding the atlas, and coordinates for the segment polygons of the atlas. Before setting up the data from the statistics to project onto the segments, looking at the atlas datasets might help understand what the data needs to look like, or avoid looking like.

```
dkt
```

```
## # A tibble: 80 x 6
##    atlas area              hemi  side    label             ggseg
##    <chr> <chr>             <chr> <chr>   <chr>             <list>
```

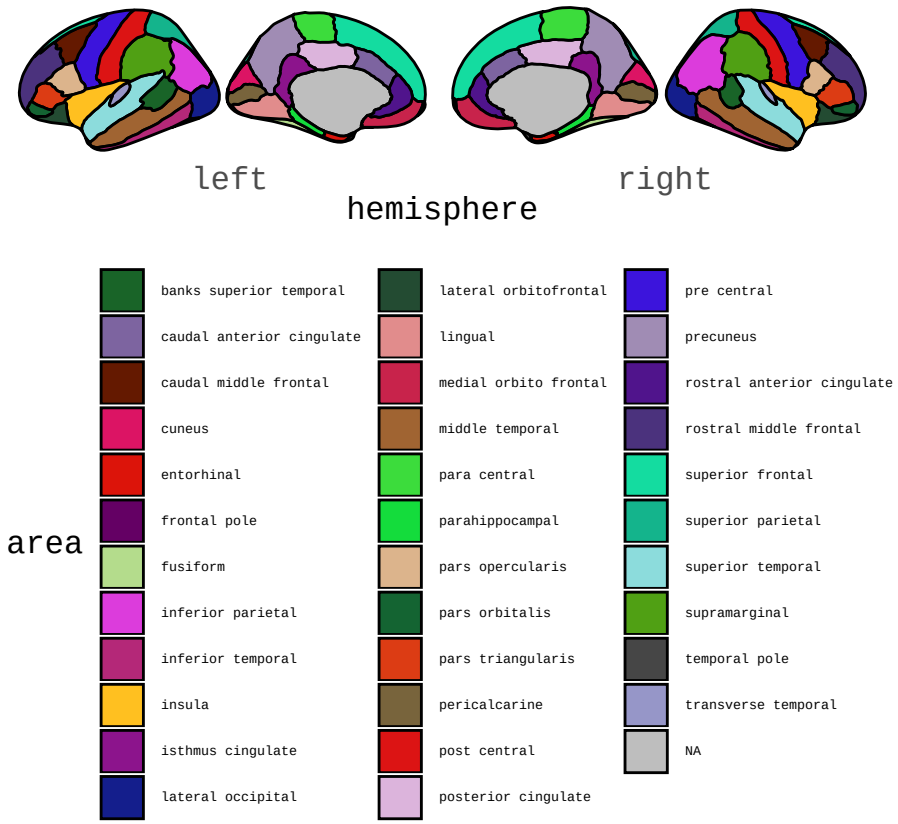


Figure 4. Supplying 'area' to the fill option in ggseg, will use the column 'area' from the accompanying dataset to create a discrete colour palette over the segments in the atlas.

```
##  1 dkt   superior temporal  left  lateral lh_superiortempo~ <tibble [1,494~
##  2 dkt   pre central        left  lateral lh_precentral     <tibble [1,314~
##  3 dkt   post central       left  lateral lh_postcentral    <tibble [1,164~
##  4 dkt   rostral middle fr~ left  lateral lh_rostralmiddle~ <tibble [1,194~
##  5 dkt   insula             left  lateral lh_insula         <tibble [870 x~
##  6 dkt   superior parietal  left  lateral lh_superiorparie~ <tibble [480 x~
##  7 dkt   inferior temporal  left  lateral lh_inferiortempo~ <tibble [906 x~
##  8 dkt   lateral occipital  left  lateral lh_lateraloccipi~ <tibble [738 x~
##  9 dkt   lateral orbitofro~ left  lateral lh_lateralorbito~ <tibble [528 x~
## 10 dkt   superior frontal   left  lateral lh_superiorfront~ <tibble [420 x~
## # ... with 70 more rows
```

The column 'label' was added to the atlas for easy matching with the output from Freesurfer statistics tables. The data is in long-format, meaning each segment has its own row, meaning the data you wish to project onto the segments also need to be in long-format. As data directly from Freesurfer comes in wide-format, this requires some extra wranling.

```
freesurfer_stats <- data.frame(
  id = c(10:12),
  lh_superiortemporal = c(3.32, 4.1, 3.5),
  lh_precentral = c(2.3, 2.5, 2.1),
  lh_rostralmiddlefrontal = c(3.3, 3.2, 3.1)
  )
freesurfer_stats
```

```
##   id lh_superiortemporal lh_precentral lh_rostralmiddlefrontal
## 1 10                3.32           2.3                     3.3
## 2 11                4.10           2.5                     3.2
## 3 12                3.50           2.1                     3.1
```

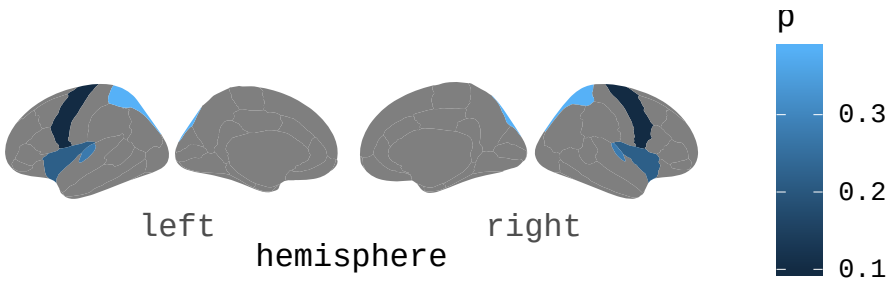
```
freesurfer_long <- freesurfer_stats %>%
  gather(label, thickness, -id)
freesurfer_long
```

```
##   id                    label thickness
## 1 10     lh_superiortemporal      3.32
## 2 11     lh_superiortemporal      4.10
## 3 12     lh_superiortemporal      3.50
## 4 10           lh_precentral      2.30
## 5 11           lh_precentral      2.50
## 6 12           lh_precentral      2.10
## 7 10 lh_rostralmiddlefrontal      3.30
## 8 11 lh_rostralmiddlefrontal      3.20
## 9 12 lh_rostralmiddlefrontal      3.10
```

This data can now be used directly with the `ggseg` function, as the `label` column corresponds in name and content with the `label` column in the atlas data of dkt. The data **must** include a column that has the same name at atleast *some* data matching the values in the corresponding column in the atlas data. In the next example we create some data with 4 rows, and an 'area' and 'p' column. The `ggseg` function will recognise the matching column 'area', and merge the supplied data into the atlas using `dplyr` joins. We use the `p` column as the column flooding the segment with colour.

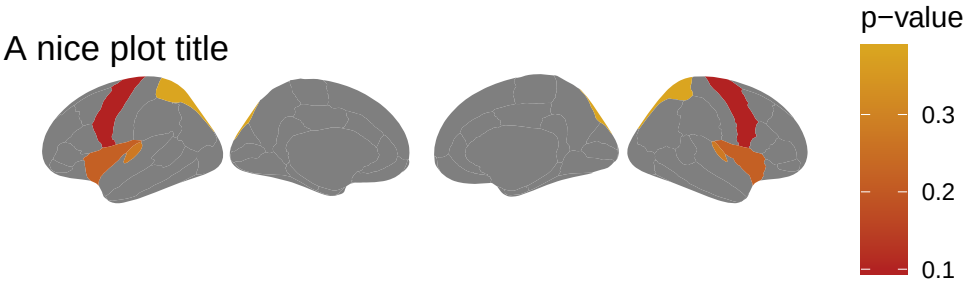
```
someData = data.frame(
  area = c("transverse temporal", "insula",
           "pre central","superior parietal"),
  p = sample(seq(0,.5,.001), 4),
```

```
  stringsAsFactors = FALSE)

p <- ggseg(.data=someData, mapping=aes(fill=p))
p
```



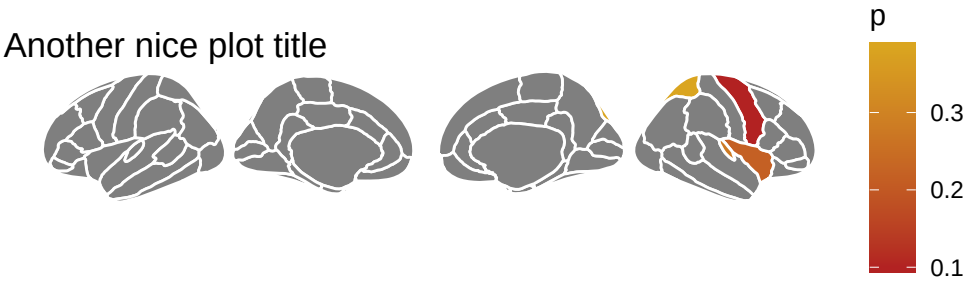
We can also change how this looks, by various ggplot functions, like scales, labs, themes and such.

```
p +
  theme_void() +
  scale_fill_gradient(low="firebrick",high="goldenrod") +
  labs(title="A nice plot title", fill="p-value")
```



If the results are only in one hemisphere, but you still want to plot both of them, make sure your data.fame includes the column `hemi` with either "right" or "left" for this to happen.

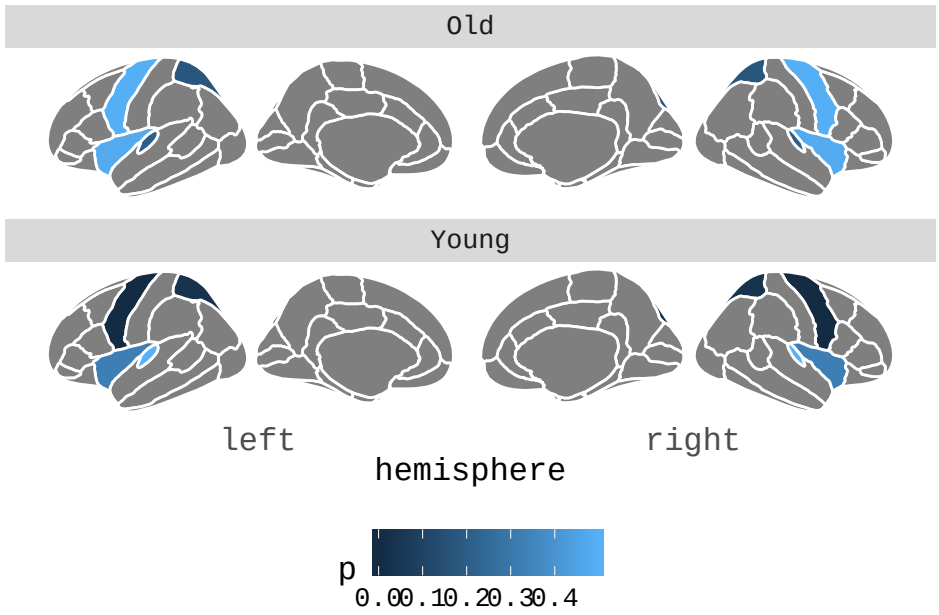
```
someData$hemi = "right"

ggseg(.data=someData, colour="white",mapping=aes(fill=p)) +
  theme_void() +
  scale_fill_gradient(low="firebrick",high="goldenrod") +
  labs(title="Another nice plot title")
```



2.1.2 Creating subplots

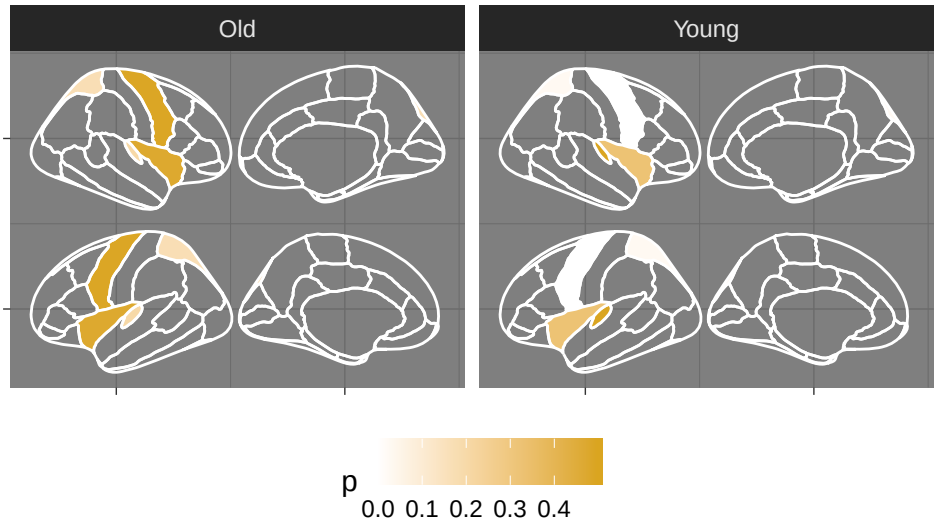
If your data has different statistics for different groups, and you want to plot them all, we can use ggplot's `facet_wrap` or `facet_grid` for that. The data needs to be in long format for this to work, as needed by ggplot, meaning the data from the groups should appear in seperate rows in the data, not in separate columns. Secondly, the data needs to be grouped using `dplyr`'s `group_by` function *before* providing the data to the `ggseg` function. The ggseg-function will detect if the data is grouped, and perform the necessary steps for the `facet`'s to work when they are applied later.

```
someData = data.frame(
  area = rep(c("transverse temporal", "insula",
               "pre central","superior parietal"),2),
  p = sample(seq(0,.5,.001), 8),
  AgeG = c(rep("Young",4), rep("Old",4)),
  stringsAsFactors = FALSE) %>%
  group_by(AgeG)

ggseg(.data=someData, colour="white", mapping=aes(fill=p)) +
  facet_wrap(~AgeG, ncol=1) +
  theme(legend.position = "bottom")
```



Depending on what type of column is used for fill (continuous or discrete), you can also use the differenct `scales` to alter the colours.

```
ggseg(.data = someData, atlas=dkt,
      colour="white", position="stacked",
      mapping=aes(fill=p)) +
  facet_wrap(~AgeG, ncol=2) +
  theme_dark() +
  theme(legend.position = "bottom",
        axis.text = element_blank(),
        axis.title = element_blank()
        ) +
  scale_fill_gradient2(high="goldenrod",
                       low="firebrick",
                       na.value="transparent")
```



All the consepts described above also work with the `aseg` atlas for subcrotical structures, except arguments to see only one hemisphere or lateral or medial surfaces, as these arguments do not make sense for this atlas. Again, inspecting the atlas data directly will be of aid when preparing data to use with the atlas

```
aseg
```

```
## # A tibble: 34 x 6
##    atlas area            hemi  side  label                ggseg
##    <chr> <chr>           <chr> <chr> <fct>                <list>
##  1 aseg  <NA>            right axial <NA>                 <tibble [808 x 5~
##  2 aseg  <NA>            left  axial <NA>                 <tibble [1,014 x~
##  3 aseg  thalamus proper left  axial Left-Thalamus-Proper <tibble [56 x 5]>
##  4 aseg  thalamus proper left  axial Right-Thalamus-Prop~ <tibble [56 x 5]>
##  5 aseg  thalamus proper right axial Left-Thalamus-Proper <tibble [61 x 5]>
##  6 aseg  thalamus proper right axial Right-Thalamus-Prop~ <tibble [61 x 5]>
##  7 aseg  lateral ventri~ right axial Left-Lateral-Ventri~ <tibble [66 x 5]>
##  8 aseg  lateral ventri~ right axial Right-Lateral-Ventr~ <tibble [66 x 5]>
##  9 aseg  hippocampus     left  axial Left-Hippocampus     <tibble [47 x 5]>
## 10 aseg  hippocampus     left  axial Right-Hippocampus    <tibble [47 x 5]>
## # ... with 24 more rows
```

```
ggseg(atlas="aseg", mapping=aes(fill=area))
```

2.2 Plotting 3D mesh data

Creating plots of the brain as 2-dimensional can at times make it difficult to properly understand where an atlas segment is located in the true 3-dimensional brain. This is particularly true when looking within the brain as subcortical structures or white matter tracts. To provide the option of viewing and printing the plots in 3-dimensional space, another function was created using plotly. The data is somewhat more complex than the 2D ggplot polygon version `ggseg`, as it includes options for brain inflation, glass brains and much more, to ease viewing and understanding. Another positive effect of basing this function on plotly, is the interactivity of the plot, which also makes it nice for public dissemination. A lot of credit goes to A.M.Winkler and his Brainder work, which supplied us with the first examples of going from Freesurfer `.srf`-files to `.ply`-files, and whose scripts massively aided us in making this work.

The function `ggseg3d()`, is based in the plotly, it is recommended to get a little familiarized with with plotly.

Out-of-the-box, `ggseg3d()` works without supplying any extra information, and plots the `dkt_3d` atlas in 'LCBC' surface. The 'LCBC' surface is the white matter surface, inflated in 10 steps, to increase visibility within folds of the brain, while still maintining curvature to indicate where folds have been. All `[...]_3d` atlases have a built in `colour` column for default colour plotting of the segments.

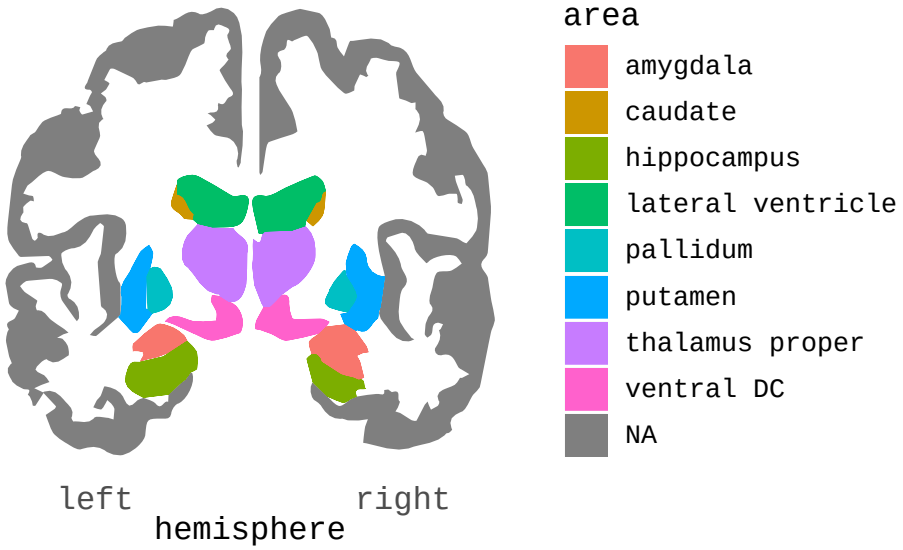
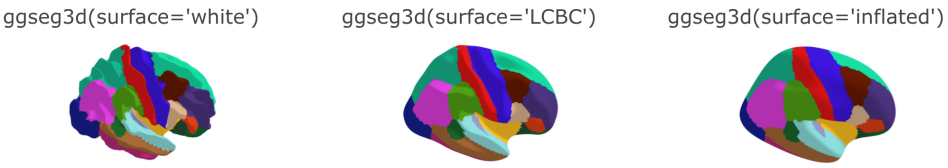


Figure 5. The first 10 rows of the `aseg` dataset, that has some specific differences from the dkt, like missing the option to not view both hemispheres, as the plot would be hard to understand that way.



The cortical atlas data is stored in nested tibbles, which have datasets for three different surfaces and the two hemispheres. The 'data'-column within includes all necessary information to `ggseg3d()` to create a mesh-plot.

```
dkt_3d
```

```
## # A tibble: 6 x 4
##   atlas  surf     hemi  ggseg_3d
##   <chr>  <chr>    <chr> <list>
## 1 dkt_3d inflated left  <tibble [36 x 8]>
## 2 dkt_3d inflated right <tibble [36 x 8]>
## 3 dkt_3d LCBC     left  <tibble [36 x 8]>
## 4 dkt_3d LCBC     right <tibble [36 x 8]>
## 5 dkt_3d white    left  <tibble [36 x 8]>
## 6 dkt_3d white    right <tibble [36 x 8]>
```

To grab all the data for a surface and hemisphere, you should reduce the data to one line, and then `unnest()`

```
dkt_3d %>%
  filter(surf == "inflated" & hemi == "right") %>%
  unnest(ggseg_3d) %>%
  select(-lobe, -acronym)
```

```
## # A tibble: 36 x 9
## atlas surf hemi area colour mesh label roi annot
## <chr> <chr> <chr> <chr> <chr> <list> <chr> <chr> <chr>
## 1 dkt_3d infla~ right <NA> <NA> <list~ rh_medial~ 0001 medialwa~
```

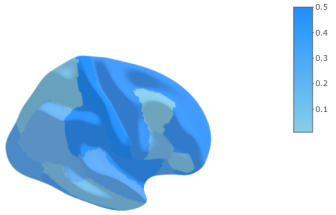
```
## 2 dkt_3d infla~ right banks supe~ #1964~ <list~ rh_bankss~ 0002
bankssts
## 3 dkt_3d infla~ right caudal ant~ #7D64~ <list~ rh_caudal~ 0003
caudalan~
## 4 dkt_3d infla~ right caudal mid~ #6419~ <list~ rh_caudal~ 0004
caudalmi~
## 5 dkt_3d infla~ right corpus cal~ <NA> <list~ rh_corpus~ 0005
corpusca~
## 6 dkt_3d infla~ right cuneus #DC14~ <list~ rh_cuneus 0006 cuneus
## 7 dkt_3d infla~ right entorhinal #DC14~ <list~ rh_entorh~ 0007
entorhin~
## 8 dkt_3d infla~ right fusiform #B4DC~ <list~ rh_fusifo~ 0008
fusiform
## 9 dkt_3d infla~ right inferior p~ #DC3C~ <list~ rh_inferi~ 0009
inferior~
## 10 dkt_3d infla~ right inferior t~ #B428~ <list~ rh_inferi~ 0010
inferior~
## # ... with 26 more rows
```

2.2.1 External data supply

Particularly notice the `mesh` column, which is a list column of lists. In there is all the 6 vectors needed to create the mesh of the tri-surface plot. You'll also need to notice the `label`, `annot` and `area` columns, which are likely the columns you will be matching on when proviging with your own data for colours. You need to be meticulous when fixing your data, be sure it matches. The function *should* give you a warning if it's struggling to match something.

The column you want to use for colour, needs to be supplied to the `colour` option, and you'll likely want to supply it to the `text` option, as this will add another line to the plotly hover information.

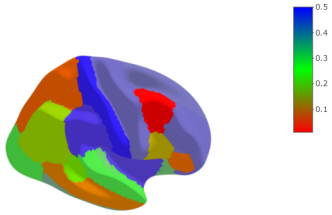
```
someData = dkt_3d %>%
  filter(surf == "inflated" &
         hemi == "right") %>%
  unnest(ggseg_3d) %>%
  select(area) %>%
  na.omit() %>%
  mutate(p = sample(seq(0,.5, length.out = 100 ), nrow(.)) %>%
           round(2))

  ggseg3d(.data = someData, atlas = dkt_3d, colour = "p", text = "p")
```



2.2.2 Colours

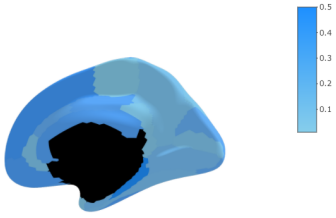
You can provide custom colour palettes either in hex or R-names. Colours will be evenly spaced when creating the colour-scale, and you may provide as many colours as you wish.

```
  ggseg3d(.data = someData, atlas = dkt_3d,
          colour = "p", text = "p",
          palette = c("#ff0000", "#00ff00", "#0000ff"))
```



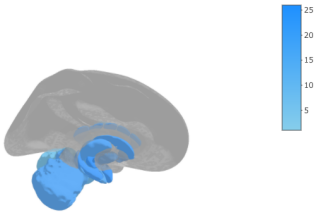
If you want to alter the colour of `NA` regions, supply `na.colour`, either as HEX colour or colour name. This option only takes a single colour.

```
  ggseg3d(.data = someData, atlas = dkt_3d,
          colour = "p", text = "p",
          na.colour = "black", camera="medial")
```



If you are plotting the sub-cortical structures, you might want to reduce the opacity of the `NA` structures, so that you can see the more medial structures. You may also want to add the `glassbrain` option, any value above 0 and up to 1 increases the opacity of a glass brain overlay. The glassbrain option provides a frame of reference for the subcortical structures, and as such it is recommended to add this when viewing these structures. There are three glassbrain options, controlling the opacity, which hemispheres to add, and the colour.

```
somData_aseg = aseg_3d %>%
  unnest() %>%
  select(label) %>%
  filter(!grepl("Ventricle|Putamen", label)) %>%
  mutate(p = seq(1, nrow(.)))

  ggseg3d(.data = somData_aseg, atlas = aseg_3d,
          colour = "p", text = "p",
          na.alpha= .5, glassbrain = .5,
          glassbrain_hemisphere = "left")
```



There are many plotly options that may help you adapt the plot to look the way you wish. In addition to Carson Sievert's book on plotly in R, we recommend resources for modifying axes in 3D plots, the basic introduction to tri-surface plots, and this tutorial on tri-surface plots with plotly in R.

3 SUMMARY

The aim of the `ggseg`-package is to aid researchers easily disseminate analysis results in an easier and more stream-lined fashion. This tutorial should provide instructions on how to use the main package functions, and cover the most commonly used options. The two main functions in the package, `ggseg()` and `ggseg3d()`, create polygon and mesh-plots of brain atlases, using the accompanying package data. In addition to the two default atlases available in `ggseg`, the `ggsegExtra` package includes 12 more atlases. We encourage users of this package to contribute more atlases they find uses for, and help build an extensive repository of brain atlases.