R topics documented:

+ .ggb ................................................................. 3
+ .ggbrain_images ............................................. 3
annotate_coordinates ........................................... 4
annotate_panel .................................................. 5
count_neighbors ............................................... 5
define ............................................................ 6
fill_from_edge ................................................... 7
find_threads ..................................................... 8
flood_fill ....................................................... 9
geom_brain ........................................................ 9
gem_outline ..................................................... 11
gem_region_label ............................................... 13
gem_region_label_repel ....................................... 14
gem_region_text ............................................... 15
gem_region_text_repel ....................................... 15
ggbrain .......................................................... 16
ggbrain_images ................................................ 17
ggbrain_label ................................................... 22
ggbrain_layer ................................................... 24
ggbrain_layer_brain ......................................... 28
ggbrain_layer_outline ....................................... 30
ggbrain_panel .................................................. 32
ggbrain_plot .................................................... 35
ggplot_add.ggbrain_label ................................... 37
ggplot_add.ggbrain_layer ................................... 37
ggplot_add.ggbrain_panel ................................... 38
images ........................................................... 38
integer_breaks .................................................. 39
montage .......................................................... 40
nn_impute ........................................................ 41
plot.ggb .......................................................... 42
plot.ggbrain_panel .......................................... 42
plot.ggbrain_plot ............................................. 43
range_breaks ................................................... 43
render ........................................................... 44
scale_fill_bisided .......................................... 44
slices ........................................................... 45

Index 47
Description

addition operator for ggb object to support ggplot-like syntax

Usage

```r
## S3 method for class 'ggb'
o1 + o2
```

Arguments

- `o1`: the first object inheriting the ggb class
- `o2`: the second object inheriting the ggb class

Details

Note that the addition operator always clones the underlying `o1` object rather than modifying it in place.

Value

a modified version of the `o1` object with `o2` added to it

Description

addition operator for combining ggbrain_images objects

Usage

```r
## S3 method for class 'ggbrain_images'
o1 + o2
```

Arguments

- `o1`: first ggbrain_images object
- `o2`: second ggbrain_images object
annotate_coordinates

Details

Note that the addition does not modify either existing object. Rather, the first object is cloned and
the second is added to it. If you want to add one ggbrain_images object to another in place (i.e.,
modifying the extant object), use the $add() method.

Value

combined ggbrain_images object

---

annotate_coordinates  Adds the coordinate labels to each panel based on the location of the
slice along the slicing axis (e.g., z = 15)

Description

Adds the coordinate labels to each panel based on the location of the slice along the slicing axis
(e.g., z = 15)

Usage

annotate_coordinates(x = "right", y = "bottom", ...)

Arguments

x  the x position of the coordinate label. If numeric, it is assumed to be the pixel
position along the x axis (e.g., 26). In addition, convenience values of "left",
"right", or "q[1-100]" can be used to look up the left-most, right-most, or
quantile-based positions along the x axis.

y  the y position of the coordinate label. If numeric, it is assumed to be the pixel
position along the y axis (e.g., 26). In addition, convenience values of 'top',
"bottom", or "q[1-100]" can be used to look up the top-most, bottom-most, or
quantile-based positions along the y axis.

...  any other arguments to ggplot2::annotate, which will be passed through to each
panel

Value

a ggb object with the action 'add_annotations', used in a gbrain addition chain
**annotate_panel**

*Adds custom annotations to a single panel on the ggbrain plot*

**Description**

Adds custom annotations to a single panel on the ggbrain plot

**Usage**

```r
annotate_panel(x = "middle", y = "middle", slice_index = NULL, ...)
```

**Arguments**

- **x**
  
  the x position of the annotation. If numeric, it is assumed to be the pixel position along the x axis (e.g., 26). In addition, convenience values of 'left', 'right', or 'q[1-100]' can be used to look up the left-most, right-most, or quantile-based positions along the x axis.

- **y**

  the y position of the annotation. If numeric, it is assumed to be the pixel position along the y axis (e.g., 26). In addition, convenience values of 'left', 'right', or 'q[1-100]' can be used to look up the left-most, right-most, or quantile-based positions along the x axis.

- **slice_index**

  the slice number to which this annotation is added. These are numbered in the wrapping order from patchwork::wrap_plots, which will normally go from top-left to bottom-right.

- **...**

  Additional parameters passed to ggplot2::annotate such as label or geom

**Details**

Note that this only handles a single annotation on a single panel!

**Value**

a ggb object with the relevant annotations field and an action of "add_annotations"

---

**count_neighbors**

*This function counts the number of neighboring/touching pixels in a 2D binary image*

**Description**

This function counts the number of neighboring/touching pixels in a 2D binary image

**Arguments**

- **im**

  A boolean matrix representing a binary image

- **diagonal**

  Whether to count diagonal elements as valid neighbors
Details

This is an internal function used by geom_outline to clean up outlines

Value

A matrix of the same size as \( m \) containing the number of neighboring pixels

Author(s)

Michael Hallquist

---

`define` Adds contrast definitions to the ggbrain plot

Description

Adds contrast definitions to the ggbrain plot

Usage

```r
define(contrasts = NULL)
```

Arguments

- `contrasts`: a character vector or list containing contrasts to be computed as part of the ggbrain object definition.

Details

Contrasts must take the form of `<name> := <value expression>` or must use a named vector. Note that defining a contrast does not directly impact the appearance of the plot unless the contrast is named in a geom_* layer.

Also note that contrasts can be specified in the definition of a layer. Thus, the define function has two primary virtues. First, it allows for the conceptual separation of contrast definition versus usage inside a geom_* layer, which is particularly useful if a contrast is used across several layers. Second, it allows downstream layers to further modify the contrast, such as when we compute a

Value

A `ggb` object with the relevant contrasts and an action of `add_contrasts`
fill_from_edge

This function finds holes by flood filling TRUE into a 2D binary image, starting from the edge

Description

This function finds holes by flood filling TRUE into a 2D binary image, starting from the edge
find_threads

Arguments

im A boolean matrix representing a binary image
nedges An integer specifying how many starting points along the edge to use for filling
TRUE. The starts are northwest (1), southwest (2), southeast (3), and northeast
(4). The compute time increases with the number of starts.

Details

This is an internal function used by geom_outline to clean up outlines

Value

A matrix of the same size as im containing the number of neighboring pixels

Author(s)

Michael Hallquist

find_threads This function finds 'threads' hanging off of the edges of blobs in an
image, allowing the user to trim them

Description

This function finds 'threads' hanging off of the edges of blobs in an image, allowing the user to trim them

Arguments

im A numeric matrix representing an image, with non-zero values representing pixels to display
min_neighbors the minimum number of neighbors a pixel must have to be retained
maxit the maximum number of iterations to run the thread trimming algorithm. Default: 15.
diagonal Whether to count diagonal elements as valid neighbors

Details

This algorithm runs count_neighbors iteratively until no pixel exceeds the trimming threshold
min_neighbors or the maximum number of iterations, maxit, is reached.
By running iteratively, long tails are trimmed sequentially by pruning the most disconnected voxels.

Value

A logical matrix matrix of the same size as im containing the number of neighboring pixels
**flood_fill**

This function flood fills a binary image with TRUE for any value of FALSE

**Description**

This function flood fills a binary image with TRUE for any value of FALSE

**Arguments**

- `im`: A boolean matrix reference representing a binary image
- `x`: the starting x position for fill
- `y`: the starting y position for fill
- `r`: the number of rows in `im`
- `c`: the number of columns in `im`

**Details**

This is an internal function used by `geom_outline` to clean up outlines

**Value**

Nothing. The matrix `im` is modified in place (by reference)

**Author(s)**

Michael Hallquist

---

**geom_brain**

*Adds a raster layer to the ggbrain plot, displaying pixels from the specified layer definition*

**Description**

Adds a raster layer to the ggbrain plot, displaying pixels from the specified layer definition
Usage

geom_brain(
    definition = NULL,
    name = NULL,
    fill = NULL,
    fill_scale = NULL,
    mapping = NULL,
    limits = NULL,
    breaks = NULL,
    show_legend = TRUE,
    interpolate = FALSE,
    unify_scales = TRUE,
    alpha = NULL,
    blur_edge = NULL,
    fill_holes = NULL,
    remove_specks = NULL,
    trim_threads = NULL
)

Arguments

definition a character string of the contrast or image definition used to define this layer. Can be a simple image name (e.g., 'underlay') or a contrast string (e.g., 'overlay[overlay > 5]')

name the name of this layer, used for referencing in layer and panel modifications

fill A character string indicating the color used to fill all non-NA pixels in this layer. This is used to set the fill color, in distinction to color mapping: mapping=aes(fill=<variable>).

fill_scale a ggplot scale_fill_* object used for mapping the fill column to the color of pixels in this layer.

mapping the aesthetic mapping of the layer data to the display. Should be an aes() object and supports fill (color of filled pixels). Default is aes(fill=value), which maps the numeric value of the layer data to the fill color of the squares at each spatial position. For labeled data, you might use aes(fill=<label_col_name>).

limits if provided, sets the upper and lower bounds on the scale

breaks if provided, a function to draw the breaks on the fill scale

show_legend if TRUE, show the fill scale in the plot legend

interpolate passes to geom_raster and controls whether the fill is interpolated over continuous space

unify_scales if TRUE, when this layer is reused across panels, unify the scales to match

alpha a number between 0 and 1 that sets the alpha transparency of this layer. Default: 1

blur_edge the standard deviation (sigma) of a Gaussian kernel applied to the edge of this layer to smooth it. This makes the layer less jagged in appearance and is akin to antialiasing.
fill_holes  An optional positive integer specifying the size of holes (in pixels) inside clusters to be filled by nearest neighbor imputation. Default: 0.

remove_specks  An optional positive integer specifying the size of specks (in pixels) to be removed from each slice prior to display. Specks are small clusters that may be distracting and contribute to a 'salt and pepper' appearance.

trim_threads  the minimum number of neighboring pixels (including diagonals) that must be present to keep a pixel.

Details

Note that the fill_scale and limits must be specified at the time of the geom_brain creation in order for them to be mapped properly within ggplot. Because we overlay many raster layers in a ggplot object that all use the fill aesthetic mapping, it becomes hard to map the color scales after the layer is created using the typical + scale_fill_* syntax, and similarly for scale limits.

Value

a ggb object populated with the relevant geom_brain and the action of 'add_layers'

Examples

```r
# T1-weighted template
t1 <- system.file("extdata", "mni_template_2009c_3mm.nii.gz", package = "ggbrain")

# signed reward prediction error map
signed_pe <- system.file("extdata", "pe_ptfce_fwep_0.05.nii.gz", package = "ggbrain")

gg_obj <- ggbrain() +
  images(c(underlay = t1, overlay = signed_pe)) +
  slices(c("x = 25%", "x = 75%")) +
  geom_brain("underlay") +
  geom_brain(definition="overlay[overlay > 1]", fill_scale=ggplot2::scale_fill_viridis_c("pos z"))
```

geom_outline  Adds an outline layer to the ggbrain plot, displaying outlines from the non-missing pixels in the specified layer definition

Description

Adds an outline layer to the ggbrain plot, displaying outlines from the non-missing pixels in the specified layer definition

Usage

```r
geom_outline(
  definition = NULL,
  name = NULL,
  outline = NULL,
)```
geom_outline

outline_scale = NULL,
mapping = ggplot2::aes(outline = NULL, fill = NULL),
size = NULL,
limits = NULL,
breaks = integer_breaks(),
show_legend = TRUE,
interpolate = FALSE,
unify_scales = TRUE,
alpha = 1,
blur_edge = NULL,
fill_holes = NULL,
remove_specks = NULL,
trim_threads = NULL,
dil_ero = 0L
)

Arguments

definition a character string of the contrast or image definition used to define this layer. Can be a simple image name (e.g., 'underlay') or a contrast string (e.g., 'overlay[overlay > 5]')

name the name of this layer, used for referencing in layer and panel modifications

outline A character string indicating the color used to draw outlines in this layer. This is used to set the outline color, in distinction to outline color mapping: mapping=aes(outline=<variable>)

outline_scale a ggplot scale_fill_* object used for mapping the fill column to the color of pixels in this layer.

mapping the aesthetic mapping of the layer data to the display. Should be an aes() object and supports outline (outline color of pixels). Default is aes(outline=NULL), which uses a set outline color.

size the size of outlines to be drawn in pixel units. Default: 1

limits if provided, sets the upper and lower bounds on the scale

breaks if provided, a function to draw the breaks on the fill scale

show_legend if TRUE, show the fill scale in the plot legend

interpolate passes to geom_raster and controls whether the fill is interpolated over continuous space

unify_scales if TRUE, when this layer is reused across panels, unify the scales to match

alpha a number between 0 and 1 that sets the alpha transparency of this layer. Default: 1

blur_edge the standard deviation (sigma) of a Gaussian kernel applied to the edge of this layer to smooth it. This makes the layer less jagged in appearance and is akin to antialiasing.

fill_holes An optional positive integer specifying the size of holes (in pixels) inside clusters to be filled by nearest neighbor imputation. Default: 0.
**geom_region_label**

Variant of geom_label used for plotting region labels on slices

### Description

Variant of geom_label used for plotting region labels on slices

### Usage

```r
geom_region_label(image, label_column = "label", min_px = 1L, ...)
```
14

**geom_region_label_repel**

**Arguments**

- **image**: The name of the image within the underlying ggbrain_slices object that contains the labeled data positions
- **label_column**: The column name name for the labels to use within the slice data
- **min_px**: The minimum number of pixels present on a slice that will result in a text label. Default: 1
  - ... All other parameters passed through to geom_label

**Value**

- a ggb object with the relevant ggbrain_label field and an action of "add_region_labels"

**Description**

Variant of geom_label_repel used for plotting region labels on slices with separation from other labels

**Usage**

- geom_region_label_repel(image, label_column = "label", min_px = 1L, ...)

**Arguments**

- **image**: The name of the image within the underlying ggbrain_slices object that contains the labeled data positions
- **label_column**: The column name name for the labels to use within the slice data
- **min_px**: The minimum number of pixels present on a slice that will result in a text label. Default: 1
  - ... All other parameters passed through to geom_label_repel

**Value**

- a ggb object with the relevant ggbrain_label field and an action of "add_region_labels"
**geom_region_text**

Variant of geom_text used for plotting region labels on slices

**Description**

Variant of geom_text used for plotting region labels on slices

**Usage**

```r
geom_region_text(image, label_column = "label", min_px = 1L, ...)
```

**Arguments**

- **image**
  The name of the image within the underlying ggbrain_slices object that contains the labeled data positions
- **label_column**
  The column name name for the labels to use within the slice data
- **min_px**
  The minimum number of pixels present on a slice that will result in a text label. Default: 1
- ... All other parameters passed through to geom_text

**Value**

A ggb object with the relevant ggbrain_label field and an action of "add_region_labels"

---

**geom_region_text_repel**

Variant of geom_text_repel used for plotting region labels on slices with separation from other labels

**Description**

Variant of geom_text_repel used for plotting region labels on slices with separation from other labels

**Usage**

```r
geom_region_text_repel(image, label_column = "label", min_px = 1L, ...)
```

**Arguments**

- **image**
  The name of the image within the underlying ggbrain_slices object that contains the labeled data positions
- **label_column**
  The column name name for the labels to use within the slice data
- **min_px**
  The minimum number of pixels present on a slice that will result in a text label. Default: 1
- ... All other parameters passed through to geom_text_repel
**ggbrain**

*create ggb container object for a given plot*

**Description**

create ggb container object for a given plot

**Usage**

```r
ggbrain(
  images = NULL,
  slices = NULL,
  title = NULL,
  bg_color = "grey8",
  text_color = "grey92",
  base_size = 14
)
```

**Arguments**

- **images**: a character vector or existing ggbrain_images object defining which images should be included in this plot
- **slices**: a set of slices to be added to the plot
- **title**: the overall title to be added to the plot
- **bg_color**: The background color of the overall plot
- **text_color**: The default text color of the overall plot (passes through to panels)
- **base_size**: The base size of fonts used in the plot (cf. theme_minimal)

**Value**

a ggb object containing basic information for a ggbrain plot such as background color, text color, and font size
ggbrain_images R6 class for compiling images to render in ggplot

Description

R6 class for compiling images to render in ggplot

Details

Note that this class is exported only for power users and rarely needs to be called directly in typical use of the package. Instead, look at images().

Value

a ggbrain_images R6 class containing fields related to a set of NIfTI images imported into R

Active bindings

zero_tol the (positive) numeric value that should be treated as indistinguishable from zero. This value is used to set small values in the images to exactly zero for proper masking. Default 1e-6
slices a character vector of cached slice specifications to be used in $get_slices()
contrasts a character vector of cached contrast specifications to be used in $get_slices()

Methods

Public methods:

• ggbrain_images$new()
• ggbrain_images$add()
• ggbrain_images$add_labels()
• ggbrain_images$add_images()
• ggbrain_images$filter_images()
• ggbrain_images$dim()
• ggbrain_images$get_image_names()
• ggbrain_images$get_images()
• ggbrain_images$get_headers()
• ggbrain_images$remove_images()
• ggbrain_images$winsorize_images()
• ggbrain_images$na_images()
• ggbrain_images$summary()
• ggbrain_images$get_nz_indices()
• ggbrain_images$add_slices()
• ggbrain_images$add_contrasts()
Method **new()**: create ggbrain_images object consisting of one or more NIfTI images

*Usage:*

```r
ggbrain_images$new(images = NULL, volumes = NULL, labels = NULL, filter = NULL)
```

*Arguments:*

- **images**: a character vector of file names containing NIfTI images to read
- **volumes**: the volumes to be read from each element of images. By default, this is 1, in which case the first volume is used, which is appropriate for all 3-D images. For 4-D images, volumes gives you more flexibility over the volume to display.
- **labels**: A named list of data.frames with labels that map to values in the integer-valued/atlas elements of images. If a single data.frame is passed, it will be accepted if only a single image is passed, too. These are then assumed to correspond
- **filter**: A named list of filter expressions to be applied to particular images. The names of the list correspond to the names of the images provided. Each element of the list can either be a character vector denoting a filtering expression (e.g., 'value < 100') or a numeric vector denoting values of the image that should be retained (e.g., c(5, 10, 12)).

Method **add()**: method to add another ggbrain_images object to this one

*Usage:*

```r
ggbrain_images$add(obj)
```

*Arguments:*

- **obj**: the ggbrain_images object to combine with this one

Method **add_labels()**: add a labels data.frame that connects an integer-valued image with a set of labels

*Usage:*

```r
ggbrain_images$add_labels(...)
```

*Arguments:*

- **...**: named arguments containing data.frame objects for each image to be labeled. The argument name should match the image name to be labeled and the value should be a data.frame containing `value` and `label`.

Details: As a result of `$add_labels`, the `$get_slices` method will always remap the numeric values for label images to the corresponding text-based labels in the label data. In addition, a new attribute will be returned called "slice_labels" that contains a row for each region represented in each slice.

Method **add_images()**: add one or more images to this ggbrain_images object

*Usage:*

```r
ggbrain_images$add_images(...)
```
ggbrain_images$add_images(images = NULL, volumes = NULL)

Arguments:
images a character vector of file names containing NIfTI images to read
volumes a number indicating the volume within the images to read. At present, this must be a
single number – perhaps in the future, it could be a vector so that many timepoints in a 4-D
image could be displayed.

Method filter_images(): filters an image based on an expression such as a subsetting opera-
tion

Usage:
ggbrain_images$filter_images(filter = NULL)

Arguments:
filter a character string or numeric vector of the filter to apply

Details: if expr is a numeric vector, only values in this set will be retained. If a character string
expression is used, it should use the variable name ‘value’ to refer to the numeric values to be
filtered, such as ‘value > 10’.

Method dim(): return the 3D dimensions of the images contained in this object

Usage:
ggbrain_images$dim()

Method get_image_names(): return the names of the images contained in this object

Usage:
ggbrain_images$get_image_names()

Method get_images(): return the RNifti objects of one or more images contained in this object

Usage:
ggbrain_images$get_images(img_names = NULL, drop = TRUE)

Arguments:
img_names The names of images to return. Use $get_image_names() if you’re uncertain about
what is available.
drop If TRUE, a single image is returned as an RNifti object, rather than a single-element list
containing that object.

Method get_headers(): return the NIfTI headers for one or more images contained in this object

Usage:
ggbrain_images$get_headers(img_names = NULL, drop = TRUE)

Arguments:
img_names The names of images whose header are returned. Use $get_image_names() if
you’re uncertain about what is available.
drop If TRUE, a single header is returned as an niftiHeader object, rather than a single-element
list containing that object.
Method `remove_images()`: method for removing one or more images from the `ggbrain_images` object

Usage:
```r
ggbrain_images$remove_images(img_names)
```

Arguments:
- `img_names` names of images to remove from object

Method `winsorize_images()`: winsorize the tails of a set of images to pull in extreme values

Usage:
```r
ggbrain_images$winsorize_images(img_names, quantiles = c(0.001, 0.999))
```

Arguments:
- `img_names` The names of images in the `ggbrain_images` object to be winsorized
- `quantiles` The lower and upper quantiles used to define the thresholds for winsorizing.

Method `na_images()`: method to set values less than `threshold` to NA

Usage:
```r
ggbrain_images$na_images(img_names, threshold = NULL)
```

Arguments:
- `img_names` The names of images in the `ggbrain_images` object whose values should be set to NA
- `threshold` The threshold value whose absolute value used to determine which voxels to set to NA. If NULL, use the `pvt_zero_tol` field (default 1e-6).

Method `summary()`: print a summary of the `ggbrain_images` object

Usage:
```r
ggbrain_images$summary()
```

Method `get_nz_indices()`: return the indices of non-zero voxels

Usage:
```r
ggbrain_images$get_nz_indices(img_names = NULL)
```

Arguments:
- `img_names` The names of images in the `ggbrain_images` object whose non-zero indices should be looked up

Details: Note that this function looks for non-zero voxels in any of the images specified by `img_names`.

Method `add_slices()`: adds one or more slices to the cached slices that will be retrieved by `$get_slices()` when no `slices` argument is passed.

Usage:
```r
ggbrain_images$add_slices(slices = NULL)
```

Arguments:
- `slices` a character vector containing one or more slices to be extracted by `$get_slices`. Uses the syntax "<xyz>=<number>". Example: c("x=10", "y=50%")
Method `add_contrasts()`: adds one or more contrasts to the cached contrasts that will be retrieved by `get_slices()` when no contrasts argument is passed.

Usage:
ggbrain_images$add_contrasts(contrasts = NULL)

Arguments:
contrasts a character vector containing one or more contrasts to be extracted by `get_slices`. Uses the syntax "<img_name>[subset_expression] + <img_name>".

Method `reset_slices()`: remove all cached slice settings

Usage:
ggbrain_images$reset_slices()

Method `get_slices()`: get slice data for one or more slices based on their coordinates

Usage:
ggbrain_images$get_slices(
slices = NULL,
img_names = NULL,
contrasts = NULL,
fill_labels = FALSE,
make_square = TRUE,
remove_null_space = TRUE
)

Arguments:
slices a vector of slice positions
img_names a character vector of images contained in the ggbrain_images object to be sliced
contrasts a named character vector of contrasts to be calculated for each slice
fill_labels if TRUE, the numeric value of the image will be used for any value that does not have a corresponding label in the labels data.frame. Default: FALSE
make_square If TRUE, make all images square and of the same size
remove_null_space If TRUE, remove slices where all values are approximately zero

Details: This function always returns a data.frame where each row represents a slice requested by the user. The $slice_data element is a list-column where each element is itself a list of slice data for a given layer/image (e.g., underlay or overlay) . The $slice_matrix is a list-column where each element is a list of 2-D matrices, one per layer/image. @return a ggbrain_slices object containing the requested slices and contrasts

Method `get_slices_inplane()`: get_slices_inplane is mostly an internal function for getting one or more slices from a given plane

Usage:
ggbrain_images$get_slices_inplane(
imgs = NULL,
slice_numbers,
plane,
drop = FALSE
)
Arguments:
- **imgs**: The names of images to slice
- **slice_numbers**: The numbers of slices in the specified plant to grab
- **plane**: The image plane to slice. Must be "coronal", "sagittal", or "axial"
- **drop**: if TRUE, a single slice is returned as a 2D matrix instead of a 3D matrix with a singleton first dimension

Returns: A 3D matrix of slices x dim1 x dim2

Method **get_labels()**: return a list of data.frames containing labels for a given image

Usage:
```r
ggbbrain_images$get_labels()
```

Details: the names of the list correspond directly with the names of the images

Method **lookup_slices()**: internal function to lookup which slices to display along each axis based on their quantile, xyz coordinate, or ijk coordinate

Usage:
```r
ggbbrain_images$lookup_slices(slices, ignore_null_space = TRUE)
```

Arguments:
- **slices**: A character vector of coordinates for slices to display
- **ignore_null_space**: If TRUE, any coordinates specified as quantiles (e.g., x = 50%) use the quantiles of only the non-zero slices (ignoring blank slices)

Method **clone()**: The objects of this class are cloneable with this method.

Usage:
```r
ggbbrain_images$clone(deep = FALSE)
```

Arguments:
- **deep**: Whether to make a deep clone.

---

**ggbrain_label**

R6 class for adding labels to a ggbrain_panel

Description

R6 class for adding labels to a ggbrain_panel

Value

a ggbrain_label R6 class containing fields related to ggbrain plot labels

Public fields

- **addl_args**: a named list of additional argument to be passed to geom_text/geom_label at render
Active bindings

data a data.frame containing labels to be printed on the panel. Must contain dim1, dim2, and label as columns. The dim1 and dim2 columns control where the labels will appear on the panel
image A character string specifying the image to which these labels pertain
label_column A character string indicating which data.frame column should be used for drawing labels
min_px A positive integer indicating the minimum number of pixels present on slice that will generate a label

Methods

Public methods:

• ggbrain_label$new()
• ggbrain_label$add_to_gg()
• ggbrain_label$clone()

Method new(): create a new ggbrain_label object

Usage:
ggbrain_label$new(data = NULL, geom = "text", image = NULL, label_column = NULL, min_px = NULL, ...

Arguments:

data a data.frame containing labels to be printed on the panel. Must contain dim1, dim2, and label as columns. The dim1 and dim2 columns control where the labels will appear on the panel
geom The geom type to be plotted. Must be "text" or "label", corresponding to geom_text and geom_label, respectively.
image A string specifying the image to which these labels pertain
label_column the column in data that should be drawn as labels on the plot
min_px the minimum number of pixels
... All other arguments that will be passed directly to geom_text or geom_label such as hjust, size, and color

Method add_to_gg(): add this text layer to an existing ggplot object

Usage:
ggbrain_label$add_to_gg(base_gg)

Arguments:
base_gg the ggplot object to which we add the layer

Method clone(): The objects of this class are cloneable with this method.
Usage:

ggbrain_label$clone(deep = FALSE)

Arguments:

deep  Whether to make a deep clone.

Description

R6 class for a single layer of a ggbrain panel

Details

Note that this class is exported only for power users and rarely needs to be called directly in typical use of the package. Instead, look at geom_brain() and geom_outline().

Value

a ggbrain_layer R6 class containing fields related to a visual layer on the ggbrain plot

Active bindings

- name  the name of this layer, used for referencing in layer and panel modifications
- all_na  whether all values for this layer are NA in the data field
- definition  a character string specifying the image name or contrast that defines this layer
- source  a character string specifying the layer source within a relevant ggbrain_slices object. This is used to lookup the right layer information when combining slices and layers together Note that multiple layers can potentially have the same source, which is why a 1:1 mapping to name does not work
- data  the data.frame containing relevant data for this layer.
- show_legend  a logical indicating whether to show or hide the fill/color scale
- unify_scales  a logical indicating whether to unify scale limits and levels when this layer is added across many panels
- bisided  read-only access to whether this layer uses a bisided color scale
- categorical_fill  read-only access to whether this layer has a categorical fill scale
- fill_column  read-only access to layer fill column
- fill_scale  a scale_fill_* object containing the ggplot2 fill scale for this layer
- alpha  sets the alpha transparency of this layer.
- blur_edge  controls the standard deviation (sigma) of a Gaussian blur applied to the layer at the edge
- trim_threads  iteratively trim any pixels that have fewer than this number of neighboring pixels
- fill_holes  controls the size of holes to be filled for display (in pixels)
- remove_specks  controls the size of specks to be removed (in pixels)
Methods

Public methods:

• `ggbrain_layer$new()`
• `ggbrain_layer$set_limits()`
• `ggbrain_layer$set_pos_limits()`
• `ggbrain_layer$set_neg_limits()`
• `ggbrain_layer$set_breaks()`
• `ggbrain_layer$set_pos_breaks()`
• `ggbrain_layer$set_neg_breaks()`
• `ggbrain_layer$plot()`
• `ggbrain_layer$add_to_gg()`
• `ggbrain_layer$get_data()`
• `ggbrain_layer$is_empty()`
• `ggbrain_layer$clone()`

Method `new()`: create a new `ggbrain_layer` object

*Usage:*

```r
ggbrain_layer$new(  
  name = NULL,  
  definition = NULL,  
  limits = NULL,  
  breaks = integer_breaks(),  
  show_legend = TRUE,  
  interpolate = NULL,  
  unify_scales = TRUE,  
  alpha = NULL,  
  blur_edge = NULL,  
  fill_holes = NULL,  
  remove_specks = NULL,  
  trim_threads = NULL,  
  data = NULL
)
```

*Arguments:*

- `name`: the name of this layer, used for referencing in layer and panel modifications
- `definition`: an optional character string defining the image or contrast that should be used to lookup data from a `ggbrain_slices` object. This is mostly used internally by the `ggbrain + syntax` to allow layers to be defined without data in advance of the plot.
- `limits`: if provided, sets the upper and lower bounds on the scale
- `breaks`: if provided, a function to draw the breaks on the color scale
- `show_legend`: if TRUE, show the scale on the plot legend
- `interpolate`: passes to `geom_raster` and controls whether the fill is interpolated over continuous space
- `unify_scales`: if TRUE, when this layer is reused across panels, unify the scales to match
- `alpha`: fixed alpha transparency of this layer (use `mapping` for alpha mapping’*)
blur_edge  the standard deviation (sigma) of a Gaussian kernel applied to the edge of this layer to smooth it. This makes the layer less jagged in appearance and is akin to antialiasing.

fill_holes  the size of holes (in pixels) inside clusters to be filled by nearest neighbor imputation prior to display

remove_specks  the size of specks (in pixels) to be removed from each slice prior to display

trim_threads  the minimum number of neighboring pixels (including diagonals) that must be present to keep a pixel

data  the data.frame containing image data for this layer. Must contain "dim1", "dim2", and "value" as columns

Method set_limits(): set the limits for this layer’s scale

Usage:
ggbrain_layer$set_limits(limits)

Arguments:
limits  a 2-element numeric vector setting the lower and upper limits on the layer’s scale

Method set_pos_limits(): set the limits for this layer’s positive scale (only relevant to bisided)

Usage:
ggbrain_layer$set_pos_limits(limits)

Arguments:
limits  a 2-element numeric vector setting the lower and upper limits on the layer’s positive scale

Method set_neg_limits(): set the limits for this layer’s positive scale (only relevant to bisided)

Usage:
ggbrain_layer$set_neg_limits(limits)

Arguments:
limits  a 2-element numeric vector setting the lower and upper limits on the layer’s positive scale

Method set_breaks(): set the breaks element of this layer’s scale

Usage:
ggbrain_layer$set_breaks(breaks)

Arguments:
breaks  a function used to label the breaks

Method set_pos_breaks(): set the breaks element of this layer’s positive scale (only relevant to bisided)

Usage:
ggbrain_layer$set_pos_breaks(breaks)

Arguments:
breaks  a function used to label the positive breaks
**Method** set_neg_breaks(): set the breaks element of this layer's negative scale (only relevant to bisided)

*Usage:*

```r
ggbrain_layer$set_neg_breaks(breaks)
```

*Arguments:*

- `breaks` a function used to label the negative breaks

**Method** plot(): plot this layer alone (mostly for debugging)

*Usage:*

```r
ggbrain_layer$plot()
```

**Method** add_to_gg(): method to add this layer to an existing ggplot object

*Usage:*

```r
ggbrain_layer$add_to_gg(base_gg)
```

*Arguments:*

- `base_gg` the ggplot object to which we add the layer

**Method** get_data(): return the data.frame associated with this layer

*Usage:*

```r
ggbrain_layer$get_data(add_layer_name = FALSE)
```

*Arguments:*

- `add_layer_name` if TRUE, adds a layer_name column to the data.frame for record-keeping. Default: FALSE.

**Method** is_empty(): returns TRUE if all values are NA or if the data has 0 rows

*Usage:*

```r
ggbrain_layer$is_empty()
```

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```r
ggbrain_layer$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.
ggbrain_layer_brain  

R6 class for a single layer of a ggbrain panel using fill geom

Description

R6 class for a single layer of a ggbrain panel using fill geom

Details

Note that this class is exported only for power users and rarely needs to be called directly in typical use of the package. Instead, look at geom_brain().

Value

a ggbrain_layer_brain R6 class with fields related to a brain visual layer (relates to geom_brain)

Super class

ggbrain::ggbrain_layer -> ggbrain_layer_brain

Active bindings

fill controls color of the filled in pixels for non-NA (valid) voxels. Note that this sets the fill color, while the mapping=aes(fill=<value>) would map the fill to a column in the data, consistent with ggplot2 logic.

mapping the ggplot2 aesthetic mapping between the data columns and the display

Methods

Public methods:

• ggbrain_layer_brain$new()
• ggbrain_layer_brain$clone()

Method new(): create a new ggbrain_layer object

Usage:

ggbrain_layer_brain$new(
  name = NULL,
  definition = NULL,
  limits = NULL,
  breaks = integer_breaks(),
  show_legend = TRUE,
  interpolate = NULL,
  unify_scales = TRUE,
  alpha = NULL,
  mapping = ggplot2::aes(fill = value),
)
Arguments:
name the name of this layer, used for referencing in layer and panel modifications
definition an optional character string defining the image or contrast that should be used to lookup data from a ggbrain_slices object. This is mostly used internally by the ggbrain + syntax to allow layers to be defined without data in advance of the plot.
limits if provided, sets the upper and lower bounds on the scale
breaks if provided, a function to draw the breaks on the color scale
show_legend if TRUE, show the scale on the plot legend
interpolate passes to geom_raster and controls whether the fill is interpolated over continuous space
unify_scales if TRUE, when this layer is reused across panels, unify the scales to match
alpha a number between 0 and 1 that sets the alpha transparency of this layer. Default: 1
mapping the aesthetic mapping of the layer data to the display. Should be an aes() object and supports fill (color of filled pixels). Default is aes(fill=value), which maps the numeric value of the layer data to the fill color of the squares at each spatial position. For labeled data, you might use aes(fill=<label_col_name>).
fill A character string indicating the color used to fill all non-NA pixels in this layer. This is used in distinction to mapping=aes(fill=<variable>).
fill_scale a ggplot scale object used for mapping the value column as the fill color for the layer.
blur_edge the standard deviation (sigma) of a Gaussian kernel applied to the edge of this layer to smooth it. This makes the layer less jagged in appearance and is akin to antialiasing.
fill_holes the size of holes (in pixels) inside clusters to be filled by nearest neighbor imputation prior to display
remove_specks the size of specks (in pixels) to be removed from each slice prior to display
trim_threads the minimum number of neighboring pixels (including diagonals) that must be present to keep a pixel
data the data.frame containing image data for this layer. Must contain "dim1", "dim2", and "value" as columns

Details: To set mapping, you must provide a ggplot2 aes() object. A geom_brain() layer requires a fill aesthetic mapping, which controls the fill color of regions.

Method clone(): The objects of this class are cloneable with this method.

Usage:
ggbrain_layer_brain$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.
**Description**

R6 class for a single layer of a ggbrain panel using outline geom

R6 class for a single layer of a ggbrain panel using outline geom

**Details**

Note that this class is exported only for power users and rarely needs to be called directly in typical use of the package. Instead, look at `geom_outline()`.

**Value**

a `ggbrain_layer_outline` R6 class with fields related to a brain visual layer (relates to `geomoutline`)

**Super class**

`ggbrain::ggbrain_layer` -> `ggbrain_layer_outline`

**Active bindings**

- mapping: the ggplot2 aesthetic mapping between the data columns and the display
- outline: controls color of outline draw around non-NA (valid) voxels
- outline_scale: a scale_fill_* object containing the ggplot2 outline color scale for this layer
- size: controls size of outline drawn around non-NA (valid) voxels
- dil_ero: controls the number of pixels to dilate (> 0) or erode (< 0) the outline

**Methods**

**Public methods:**

- `ggbrain_layer_outline$new()`
- `ggbrain_layer_outline$clone()`

**Method new():** create a new ggbrain_layer object

*Usage:*

```r
ggbrain_layer_outline$new(
  name = NULL,
  definition = NULL,
  limits = NULL,
  breaks = integer_breaks(),
  show_legend = TRUE,
  interpolate = NULL,
  unify_scales = TRUE,
)```
alpha = NULL,
mapping = ggplot2::aes(outline = NULL, fill = NULL),
outline = NULL,
outline_scale = NULL,
size = NULL,
blur_edge = NULL,
fill_holes = NULL,
remove_specks = NULL,
trim_threads = NULL,
dil_ero = NULL,
data = NULL
)

Arguments:

name the name of this layer, used for referencing in layer and panel modifications
definition an optional character string defining the image or contrast that should be used to lookup data from a ggbrain_slices object. This is mostly used internally by the ggbrain + syntax to allow layers to be defined without data in advance of the plot.
limits if provided, sets the upper and lower bounds on the scale
breaks if provided, a function to draw the breaks on the color scale
show_legend if TRUE, show the scale on the plot legend
interpolate passes to geom_raster and controls whether the fill is interpolated over continuous space
unify_scales if TRUE, when this layer is reused across panels, unify the scales to match
alpha a number between 0 and 1 that sets the alpha transparency of this layer. Default: 1
mapping the aesthetic mapping of the layer data to the display. Should be an aes() object and supports outline (color of outline around clusters). Default is aes(outline=value), which maps the numeric value of the layer data to the outline color of the squares at around spatial regions. For labeled data, you might use aes(fill=<label_col_name>).
outline A character string indicating the color used to outline all non-NA pixels in this layer.
This is used in distinction to mapping=aes(outline=<variable>).
outline_scale a ggplot scale object used for mapping the value column as the outline color for the layer.
size controls the thickness of outlines
blur_edge the standard deviation (sigma) of a Gaussian kernel applied to the edge of this layer to smooth it. This makes the layer less jagged in appearance and is akin to antialiasing.
fill_holes the size of holes (in pixels) inside clusters to be filled by nearest neighbor imputation prior to display
remove_specks the size of specks (in pixels) to be removed from each slice prior to display
trim_threads the minimum number of neighboring pixels (including diagonals) that must be present to keep a pixel
dil_ero the number of pixels to dilate (> 0) or erode (<0) the outline.
data the data.frame containing image data for this layer. Must contain "dim1", "dim2", and "value" as columns

Details: To set mapping, you must provide a ggplot2 aes() object. A geom_outline() layer requires an outline aesthetic mapping, which controls the color of outlines drawn around regions.
note that the ggbrain_layer_outline class maps onto *_fill fields

**Method clone()**: The objects of this class are cloneable with this method.

*Usage:*

```r
ggbrain_layer_outline$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.

---

**Description**

R6 class for a single panel of a ggbrain image

**Details**

Note that this class is exported only for power users and rarely needs to be called directly in typical use of the package. Instead, look at `slices()`.

**Value**

a `ggbrain_panel` R6 class with fields related to a panel on the ggbrain plot

**Public fields**

- `gg` The ggplot object that contains the panel

**Methods**

**Public methods:**

- `ggbrain_panel$new()`
- `ggbrain_panel$reset_limits()`
- `ggbrain_panel$plot()`
- `ggbrain_panel$add_to_gg()`
- `ggbrain_panel$add_layer()`
- `ggbrain_panel$remove_layers()`
- `ggbrain_panel$get_data()`
- `ggbrain_panel$get_layer_names()`
- `ggbrain_panel$get_layers()`
- `ggbrain_panel$set_layer_order()`
- `ggbrain_panel$clone()`

**Method new()**: create a new `ggbrain_panel` object
**Usage:**

```r
ggbrain_panel$new(
    layers = NULL,
    title = NULL,
    bg_color = NULL,
    text_color = NULL,
    border_color = NULL,
    border_size = NULL,
    xlab = NULL,
    ylab = NULL,
    theme_custom = NULL,
    annotations = NULL,
    region_labels = NULL
)
```

**Arguments:**

- `layers` a list of `ggbrain_layer` objects to form the panel
- `title` a title for the panel added to the `ggplot` object using `ggtitle()`
- `bg_color` the color used for the background of the plot. Default: 'gray10' (nearly black)
- `text_color` the color used for text displayed on the plot. Default: 'white'.
- `border_color` the color used for drawing a border around the plot. Default: 'gray50'
- `border_size` the size of the border line drawn around the panel. Default: NULL. If this value is greater than zero, a border of this size and with color `border_color` will be drawn around the panel
- `xlab` The label to place on x axis. Default is NULL.
- `ylab` The label to place on y axis. Default is NULL.
- `theme_custom` Any custom `theme()` settings to be added to the plot
- `annotations` a data.frame containing all annotations to be added to this plot. Each row is cleaned up and passed to `ggplot2::annotate()`
- `region_labels` a list of `ggbrain_label` objects with data for plotting region labels on this panel

**Method**: `reset_limits()`: Reset the scale limits for the specified layers

```r
Usage:
ggbrain_panel$reset_limits(layer_names)
```

**Arguments:**

- `layer_names` not implemented yet

**Method**: `plot()`: plot the panel

```r
Usage:
ggbrain_panel$plot(use_global_limits = TRUE)
```

**Arguments:**

- `use_global_limits` Not implemented at present

**Method**: `add_to_gg()`: add one or more custom ggplot settings to the panel
Usage:

```r
ggbrain_panel$add_to_gg(list_args)
```

**Arguments:**

- `list_args` A list containing elements to add to the ggplot object

**Details:** Note that passing in an expression such as `theme_bw() + ggtitle("hello")` will not work because it creates an object that cannot be added sequentially to the ggplot. As noted in ggplot2’s documentation (https://ggplot2.tidyverse.org/reference/gg-add.html), to programmatically add elements to a ggplot, pass in a list where each element is added sequentially.

**Method** `add_layer()`: adds a ggplot_layer object to the panel

```r
ggbrain_panel$add_layer(layer_obj)
```

**Arguments:**

- `layer_obj` a ggbrain_layer object to add to the panel

**Method** `remove_layers()`: removes one or more layers by name

```r
ggbrain_panel$remove_layers(layer_names)
```

**Arguments:**

- `layer_names` a character string of the layers to remove from the panel

**Method** `get_data()`: returns the data for all layers in the object

```r
ggbrain_panel$get_data()
```

**Method** `get_layer_names()`: returns the names of the layers in this panel, ordered from bottom to top

```r
ggbrain_panel$get_layer_names()
```

**Method** `get_layers()`: returns a list of ggbrain_layer objects that comprise this panel

```r
ggbrain_panel$get_layers()
```

**Method** `set_layer_order()`: sets the order of layers from bottom to top based on the layer names provided

```r
ggbrain_panel$set_layer_order(ordered_names = NULL)
```

**Arguments:**

- `ordered_names` the names of the layers in the desired order from bottom to top. All layer names must be provided, not just a subset

**Method** `clone()`: The objects of this class are cloneable with this method.

```r
ggbrain_panel$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.
An R6 class for constructing a ggbrain plot from a ggbrain_slices object

Description
An R6 class for constructing a ggbrain plot from a ggbrain_slices object

Details
Note that this class is exported only for power users and rarely needs to be called directly in typical use of the package. Instead, look at ggbrain().

Value
a ggbrain_plot R6 class containing fields related to a ggbrain plot object

Active bindings
slices a ggbrain_slices object containing all slice data for this plot
layers a list of ggbrain_layer objects for this plot. Note that in assignment, the input can be a list of ggbrain_layer objects, or a list of lists where each inner element specifies the settings for that layer. Example: list(list(name='hello', fill_scale=scale_fill_distiller()))
annotations a list of annotations to be added to this plot
region_labels a list of region_labels to be added to this plot
panel_settings a list of panel settings (aesthetics) to be added to this plot
title overall plot title, added to composite plot by patchwork::plot_annotation()
bg_color background color of plot
text_color the color of text use across panels (can be overridden by panel settings)
base_size the base size of text used in ggplot theming

Methods
Public methods:
- ggbrain_plot$new()
- ggbrain_plot$add_layers()
- ggbrain_plot$reset_layers()
- ggbrain_plot$generate_plot()
- ggbrain_plot$plot()
- ggbrain_plot$clone()

Method new(): instantiate a new instance of a ggbrain_plot object
Usage:
ggbrain_plot$new(
  title = NULL,
  bg_color = NULL,
  text_color = NULL,
  base_size = NULL,
  slice_data = NULL
)

Arguments:
title overall plot title
bg_color background color of plot
text_color text color of plot
base_size base size of text used in ggplot theming
slice_data a ggbrain_slices object generated by ggbrain_images$get_slices()

Method add_layers(): adds one or more ggbrain_layer objects to this plot
Usage:
ggbrain_plot$add_layers(layers = NULL)

Arguments:
layers a list of ggbrain_layer objects (can also be a list that just specifies names, definitions, etc.)

Method reset_layers(): removes all existing layers from this ggbrain_plot object
Usage:
ggbrain_plot$reset_layers()

Method generate_plot(): generate the plot
Usage:
ggbrain_plot$generate_plot(layers = NULL, slice_indices = NULL)

Arguments:
layers a list of layers to be displayed on each panel, the order of which yields the bottom-to-top drawing order within ggplot2. Each element of layers should be a list that follows the approximate structure of the ggbrain_layer class, minimally including the layer name, which is used to lookup data of images or contrasts within the slice_data object. If NULL, all layers in the slices object will be plotted. If only a character string is passed, then those layers will be plotted with default scales.
slice_indices An optional subset of slice indices to display from the stored slice data

Details: In addition to name, the elements of a layer can include fill_scale a ggplot2 scale object for coloring the layer. Should be a scale_fill_* object. limits the numeric limits to use for the color scale of this layer breaks the scale breaks to use for the color scale of this layer show_legend if FALSE, the color scale will not appear in the legend

Method plot(): return a plot of all panels as a patchwork object
Usage:
ggbrain_plot$plot(guides = "collect")

Arguments:
guides Passes through to patchwork::plot_layout to control how legends are combined across plots. The default is "collect", which collects legends within a given nesting level (removes duplicates).

Method clone(): The objects of this class are cloneable with this method.

Usage:
ggbrain_plot$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

---

S3 method to support adding ggbrain_label objects to an existing ggplot object

Description

S3 method to support adding ggbrain_label objects to an existing ggplot object

Usage

## S3 method for class 'ggbrain_label'
ggplot_add(object, plot, object_name)

Arguments

object the ggbrain_layer object to be added to an existing ggplot
plot the ggplot object
object_name not used, but required by ggplot_add

---

S3 method to support adding ggbrain_layer objects to an existing ggplot object

Description

S3 method to support adding ggbrain_layer objects to an existing ggplot object
Usage

```r
## S3 method for class 'ggbrain_layer'
ggplot_add(object, plot, object_name)
```

Arguments

- **object**: the `ggbrain_layer` object to be added to an existing `ggplot`
- **plot**: the `ggplot` object
- **object_name**: not used, but required by `ggplot_add`

Description

S3 method to support adding `ggbrain_layer` objects to an existing `ggplot` object

Usage

```r
## S3 method for class 'ggbrain_panel'
.ggplot_add(object, plot, object_name)
```

Arguments

- **object**: the `ggbrain_layer` object to be added to an existing `ggplot`
- **plot**: the `ggplot` object
- **object_name**: not used, but required by `ggplot_add`

Description

Add images to a `ggbrain` object

Usage

```r
images(images = NULL, volumes = NULL, labels = NULL, filter = NULL)
```
**integer_breaks**

**Arguments**

- **images**: a character vector or ggbrain_images object containing NIfTI images to add to this plot.
- **volumes**: a number indicating the volume within the images to display. At present, this must be a single number – perhaps in the future, it could be a vector so that many timepoints in a 4-D image could be displayed.
- **labels**: a data.frame or named list of data.frame objects corresponding to images that should be labeled. You can only provide a data.frame if there is a single image being added. If multiple images are added, the names of the labels list are used to align the labels with a given matching image.
- **filter**: a named list or character string specifying an expression of values to retain in the image, or a numeric vector of values to retain. Calls ggbrain_images$filter_image()

**Value**

a ggb object with the relevant images and an action of 'add_images'

**Examples**

```r
t1 <- system.file("extdata", "mni_template_2009c_3mm.nii.gz", package = "ggbrain")
gg_obj <- ggbrain() +
  images(c(underlay = t1))
```

---

**integer_breaks**

*breaks function to encourage integer-valued breaks, based on input from pretty*

**Description**

breaks function to encourage integer-valued breaks, based on input from pretty

**Usage**

integer_breaks(n = 5, ...)

**Arguments**

- **n**: number of breaks (default = 5)
- **...**: Additional arguments passed to the pretty() function

**Details**


**Value**

a function for generating integer-valued breaks on a continuous scale
montage

Convenience function to add many slices in a montage along one of the 3D planes

Description

Convenience function to add many slices in a montage along one of the 3D planes

Usage

montage(
  plane = NULL,
  n = 12,
  min = 0.1,
  max = 0.9,
  min_coord = NULL,
  max_coord = NULL
)

Arguments

plane a character string specifying the 3D plane: "sagittal", "axial", "coronal", "x", "y", or "z"

n number of slices to add in this plane. Default: 12

min the lowest quantile to be included in the montage (between 0 and 1). Default: 0.1

max the highest quantile to be included in the montage (between 0 and 1). Default: 0.9

min_coord the lowest spatial position (in image coordinate space) to be included in the montage.

max_coord the highest spatial position (in image coordinate space) to be included in the montage.

Details

This can be used with slices to make a quick montage, such as slices(montage("axial", 10)).

Also note that use of standardized coordinates (in quantiles, using min and max) is mutually exclusive with the the image coordinate specifications min_coord and max_coord.

Value

a character string containing the slice positions along the requested axis
Examples

t1 <- system.file("extdata", "mni_template_2009c_3mm.nii.gz", package = "ggbrain")
gg_obj <- ggbrain() +
  images(c(underlay = t1)) +
  slices(montage("sagittal", 15))

nn_impute

Imputes missing values in a 2D matrix based on the nearest non-missing neighbors in a given radius

Description

Imputes missing values in a 2D matrix based on the nearest non-missing neighbors in a given radius

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_mat</td>
<td>a 2D matrix to fill using nearest neighbors</td>
</tr>
<tr>
<td>neighbors</td>
<td>the number of closest non-NA neighboring values to return within in_mat. Default is 4.</td>
</tr>
<tr>
<td>radius</td>
<td>the radius (in pixels) around each missing value to search for non-missing neighbors. Default is 8.</td>
</tr>
<tr>
<td>aggfun</td>
<td>the function used to aggregate the neighbors in imputation. Supports &quot;mean&quot;, &quot;median&quot;, and &quot;mode.&quot;</td>
</tr>
<tr>
<td>ignore_zeros</td>
<td>if TRUE, then zero is not a valid imputation value (since these are not data in NIfTIs)</td>
</tr>
</tbody>
</table>

Details

The "mode" aggfun should only be used when the matrix in_mat can be converted to integers without loss of information (i.e., the data are integerish values already).

Value

A copy of the matrix with NA values imputed by their nearest neighbors
plot.ggb

S3 method to allow for plot() syntax with ggbrain (ggb) objects

Description

S3 method to allow for plot() syntax with ggbrain (ggb) objects

Usage

## S3 method for class 'ggb'
plot(x, ...)

Arguments

x the ggb object to be plotted

... additional argument passed to the plot method

plot.ggbrain_panel

S3 method to allow for plot() syntax with ggbrain_panel objects

Description

S3 method to allow for plot() syntax with ggbrain_panel objects

Usage

## S3 method for class 'ggbrain_panel'
plot(x, ...)

Arguments

x the ggbrain_panel object to be plotted

... additional argument passed to the plot method
plot.ggbrain_plot

S3 method to allow for plot() syntax with ggbrain_panel objects

Description

S3 method to allow for plot() syntax with ggbrain_panel objects

Usage

## S3 method for class 'ggbrain_plot'
plot(x, ...)

Arguments

x the ggbrain_plot object to be plotted
...
additional argument passed to the plot method

range_breaks

breaks function for including min + max with labels, and a few unlabeled ticks in between

Description

breaks function for including min + max with labels, and a few unlabeled ticks in between

Usage

range_breaks(n = 3, digits = 2)

Arguments

n number of breaks added within the min-max range
digits number of decimal places to display

Value

a function for generating breaks on a continuous scale with the min and max labeled
render

Function to convert ggb object to ggplot/patchwork object

Description
Function to convert ggb object to ggplot/patchwork object

Usage
render()

Value
a ggb object with the action 'render', used in a ggbrain addition chain

scale_fill_bisided

scale for plotting separate color gradients for positive and negative values

Description
scale for plotting separate color gradients for positive and negative values

Usage
scale_fill_bisided(
  name = ggplot2::waiver(),
  neg_scale = scale_fill_distiller(palette = "Blues", direction = 1),
  pos_scale = scale_fill_distiller(palette = "Reds"),
  symmetric = TRUE
)

Arguments
name the scale name to be printed in the legend (above positive scale)
neg_scale a scale_fill_* object used for negative values
pos_scale a scale_fill_* object used for positive values
symmetric if TRUE, the limits of the positive scale will equal the inverse limits of the negative scale. Said differently, this makes the positive and negative scales symmetric

Details
Note that this will absolutely not work as a general purpose ggplot2 scale! The positive/negative combination is achieved by adding two layers/geoms behind the scenes with different color scale.
slices

Value

A ggplot2 scale of type ScaleContinuous that includes negative and positive fill scales internally in the \$neg_scale and \$pos_scale elements.

Description

Adds slices to the ggbrain plot, including additional panel aesthetics.

Usage

slices(
  coordinates = NULL,
  title = NULL,
  bg_color = NULL,
  text_color = NULL,
  border_color = NULL,
  border_size = NULL,
  xlab = NULL,
  ylab = NULL,
  theme_custom = NULL
)

Arguments

coordinates a character vector specifying the x, y, or z coordinates of the slices to be added.
title a title for the slice panels added to the ggplot object using ggtitle()
bg_color the color used for the background of the panels. Default: 'gray10' (nearly black)
text_color the color used for text displayed on the panels. Default: 'white'.
border_color the color used for drawing a border around on the panels. Default: 'gray50' (though borders are not drawn by default).
border_size the size of the border line drawn around the panels. Default: NULL. If this value is greater than zero, a border of this size and with color border_color will be drawn around the panels.
xlab The label to place on x axis. Default is NULL.
ylab The label to place on y axis. Default is NULL.
theme_custom Any custom theme() settings to be added to the panels.
Details

note that if you pass in multiple coordinates (as a vector), the title, bg_color, and other attributes will be reused for all slices added by this operation. Thus, if you want to customize specific slices or groups of slices, use multiple addition operations, as in slices(c('x=10', 'y=15'), bg_color='white') + slices(c('x=18', 'y=22'), bg_color='black').

Value

a ggb object with the relevant slices and an action of 'add_slices'

Examples

```r
t1 <- system.file("extdata", "mni_template_2009c_3mm.nii.gz", package = "ggbrain")
gg_obj <- ggbrain() +
  images(c(underlay = t1)) +
  slices(c("x = 25%", "x = 75%"), border_color = "blue")
```

Index

+.ggb, 3
+.ggbrain_images, 3

annotate_coordinates, 4
annotate_panel, 5

count_neighbors, 5

define, 6

fill_from_edge, 7
find_threads, 8
flood_fill, 9

geom_brain, 9
geom_outline, 11
geom_region_label, 13
geom_region_label_repel, 14
geom_region_text, 15
geom_region_text_repel, 15

ggbrain, 16

ggbrain::ggbrain_layer, 28, 30
ggbrain_images, 17
ggbrain_label, 22
ggbrain_layer, 24
ggbrain_layer_brain, 28
ggbrain_layeroutline, 30
ggbrain_panel, 32
ggbrain_plot, 35
ggplot_add.ggbrain_label, 37
ggplot_add.ggbrain_layer, 37
ggplot_add.ggbrain_panel, 38

images, 38
integer_breaks, 39

montage, 40

nn_impute, 41

plot.ggb, 42