Package 'ComplexUpset'

July 21, 2025

Type Package

Title Create Complex UpSet Plots Using 'ggplot2' Components

Version 1.3.3

Description UpSet plots are an improvement over Venn Diagram for set overlap visualizations. Striving to bring the best of the 'UpSetR' and 'ggplot2', this package offers a way to create complex overlap visualisations, using simple and familiar tools, i.e. geoms of 'ggplot2'. For introduction to UpSet concept, see Lex et al. (2014) <doi:10.1109/TVCG.2014.2346248>.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/krassowski/complex-upset, https://krassowski.github.io/complex-upset/

BugReports https://github.com/krassowski/complex-upset/issues

Suggests testthat (>= 2.1.0), knitr, rmarkdown, covr, tibble, ggplot2movies, vdiffr, jsonlite, data.table

Imports ggplot2, patchwork, scales, colorspace

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Michał Krassowski [aut, cre] (ORCID: <https://orcid.org/0000-0002-9638-7785>)

Maintainer Michał Krassowski <krassowski.michal+r@gmail.com>

Repository CRAN

Date/Publication 2021-12-11 15:20:03 UTC

Contents

`aes_percentage` *Generate mapping for labeling percentages*

Description

Generate mapping for labeling percentages

Usage

```
aes_percentage(relative_to, digits = 0, sep = "")
```

Arguments

`relative_to`	defines proportion that should be calculated, relative to `'intersection'`, `'group'`, or `'all'` observed values
`digits`	number of digits to show (default=0)
`sep`	separator separator between the digit and percent sign (no separator by default)

`arrange_venn` *Arrange points for Venn diagram*

Description

Arrange points for Venn diagram

Usage

```
arrange_venn(
  data,
  sets = NULL,
  radius = 1.5,
  max_iterations = 10,
  verbose = FALSE,
  outwards_adjust = 1.3,
  extract_sets = FALSE,
  extract_regions = FALSE,
  repeat_in_intersections = FALSE,
  starting_grid_size = "auto"
)
```

Arguments

`data`	a dataframe including binary columns representing membership in sets
`sets`	vector with names of columns representing membership in sets
`radius`	the radius of the circle
`max_iterations`	the maximal number of iterations
`verbose`	should debugging notes be printed?
`outwards_adjust`	the multiplier defining the distance from the centre
`extract_sets`	should only sets be extracted?
`extract_regions`	should all unique regions be extracted?
`repeat_in_intersections`	repeat intersection k times where k is the number of sets it belongs to?
`starting_grid_size`	the starting size of the grid for placement of elements

compare_between_intersections
Compare covariates between intersections

Description

Compare covariates between intersections

Usage

```
compare_between_intersections(
  data,
  intersect,
  test = kruskal.test,
  tests = list(),
  ignore = list(),
  ignore_mode_columns = TRUE,
  mode = "exclusive_intersection",
  ...
)
```

Arguments

data	a dataframe including binary columns representing membership in classes
intersect	which columns should be used to compose the intersection
test	the default test function; it is expected to accept formula and data parameters, and a list with p.value, statistic, and method
tests	a named list with tests for specific variables, overwriting the default test
ignore	a list with names of variables to exclude from testing
ignore_mode_columns	whether the membership columns and size columns for all modes should be ignored
mode	region selection mode; note that modes other than exclusive_intersection repeat observations in different test group, introducing dependencies. See get_size_mode() for accepted values.
...	passed to upset_data()

create_upset_abc_example *Create an example dataset with three sets: A, B and C*

Description

Create an example dataset with three sets: A, B and C

Usage

```
create_upset_abc_example()
```

geom_venn_circle *Circle for Venn diagram*

Description

Circle for Venn diagram

Usage

```
geom_venn_circle(
  data,
  mapping = aes_(),
  sets = NULL,
  radius = 1.5,
  resolution = 100,
  size = 0.8,
  color = "black",
  ...
)
```

Arguments

data	a dataframe including binary columns representing membership in sets
mapping	the aesthetics mapping
sets	vector with names of columns representing membership in sets
radius	the radius of the circle
resolution	the resolution of the circle rasterizer
size	width of the outline
color	the color of the outline
...	Arguments passed on to ggplot2::geom_polygon stat The statistical transformation to use on the data for this layer, as a string.

`position` Position adjustment, either as a string, or the result of a call to a position adjustment function.

`rule` Either `"evenodd"` or `"winding"`. If polygons with holes are being drawn (using the `subgroup` aesthetic) this argument defines how the hole coordinates are interpreted. See the examples in `grid::pathGrob()` for an explanation.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

`geom_venn_label_region`

Label for a region of Venn diagram

Description

Label for a region of Venn diagram

Usage

```
geom_venn_label_region(
  data,
  mapping = aes_(),
  sets = NULL,
  outwards_adjust = 1.3,
  fill = alpha("white", 0.85),
  size = 5,
  label.size = 0,
  ...
)
```

Arguments

`data`	a dataframe including binary columns representing membership in sets
`mapping`	the aesthetics mapping
`sets`	vector with names of columns representing membership in sets
`outwards_adjust`	the multiplier defining the distance from the centre

fill	the fill of the label
size	the text size
label.size	the size of the label outline
...	Arguments passed on to ggplot2::geom_label

stat The statistical transformation to use on the data for this layer, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointy specified with nudge_x or nudge_y.

parse If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

nudge_x Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

nudge_y Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

label.padding Amount of padding around label. Defaults to 0.25 lines.

label.r Radius of rounded corners. Defaults to 0.15 lines.

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

`geom_venn_label_set` *Label for a set of Venn diagram*

Description

Label for a set of Venn diagram

Usage

```
geom_venn_label_set(
  data,
  mapping = aes_(),
  sets = NULL,
  outwards_adjust = 2.5,
  fill = alpha("white", 0.85),
```

```
  size = 5,
  label.size = 0,
  ...
)
```

Arguments

data	a dataframe including binary columns representing membership in sets
mapping	the aesthetics mapping
sets	vector with names of columns representing membership in sets
outwards_adjust	the multiplier defining the distance from the centre
fill	the fill of the label
size	the text size
label.size	the size of the label outline
...	Arguments passed on to ggplot2::geom_label

stat The statistical transformation to use on the data for this layer, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointy specified with nudge_x or nudge_y.

parse If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

nudge_x Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

nudge_y Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with position.

label.padding Amount of padding around label. Defaults to 0.25 lines.

label.r Radius of rounded corners. Defaults to 0.15 lines.

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

`geom_venn_region` *Region of Venn diagram*

Description

Region of Venn diagram

Usage

```
geom_venn_region(data, mapping = aes_(), sets = NULL, resolution = 250, ...)
```

Arguments

`data`	a dataframe including binary columns representing membership in sets
`mapping`	the aesthetics mapping
`sets`	vector with names of columns representing membership in sets
`resolution`	the resolution of the circle rasterizer
`...`	Arguments passed on to `ggplot2::geom_polygon`

`stat` The statistical transformation to use on the data for this layer, as a string.

`position` Position adjustment, either as a string, or the result of a call to a position adjustment function.

`rule` Either `"evenodd"` or `"winding"`. If polygons with holes are being drawn (using the `subgroup` aesthetic) this argument defines how the hole coordinates are interpreted. See the examples in `grid::pathGrob()` for an explanation.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

get_size_mode *Retrieve symbol for given mode that can be used in aesthetics mapping with double bang (!!)*

Description

Retrieve symbol for given mode that can be used in aesthetics mapping with double bang (!!)

Usage

```
get_size_mode(mode, suffix = "_size")
```

Arguments

mode	the mode to use. Accepted values: exclusive_intersection (alias distinct), inclusive_intersection (alias intersect), inclusive_union (alias union), exclusive_union.
suffix	the column suffix in use as passed to upset_data()

intersection_matrix *Prepare layers for sets sizes plot*

Description

Prepare layers for sets sizes plot

Usage

```
intersection_matrix(
  geom = geom_point(size = 3),
  segment = geom_segment(),
  outline_color = list(active = "black", inactive = "grey70")
)
```

Arguments

geom	a geom_point call, allowing to specify parameters (e.g. geom=geom_point(shape='square'))
segment	a geom_segment call, allowing to specify parameters (e.g. segment=geom_segment(linetype='dotted
outline_color	a named list with two colors for outlines of active and inactive dots

`intersection_ratio` *Barplot annotation of relative intersections sizes*

Description

A large intersection size can be driven by a large number of members in a group; to account for that, one can divide the intersection size by the size of a union of the same groups. This cannot be calculated for the null intersection (observations which do not belong to either of the groups).

Usage

```
intersection_ratio(
  mapping = aes(),
  counts = TRUE,
  bar_number_threshold = 0.75,
  text_colors = c(on_background = "black", on_bar = "white"),
  text = list(),
  text_mapping = aes(),
  mode = "distinct",
  denominator_mode = "union",
  ...
)
```

Arguments

`mapping`	additional aesthetics for `geom_bar()`
`counts`	whether to display count number labels above the bars
`bar_number_threshold`	if less than one, labels for bars height greater than this threshold will be placed on (not above) the bars
`text_colors`	a name vector of characters specifying the color when `on_background` and `on_bar` (see `bar_number_threshold`)
`text`	additional parameters passed to `geom_text()`
`text_mapping`	additional aesthetics for `geom_text()`
`mode`	region selection mode, defines which intersection regions will be accounted for when computing the size. See `get_size_mode()` for accepted values.
`denominator_mode`	region selection mode for computing the denominator in ratio. See `get_size_mode()` for accepted values.
`...`	Arguments passed on to `intersection_size` `position` position passed to `geom_bar()`

`intersection_size` *Barplot annotation of intersections sizes*

Description

Barplot annotation of intersections sizes

Usage

```
intersection_size(
  mapping = aes(),
  counts = TRUE,
  bar_number_threshold = 0.85,
  text_colors = c(on_background = "black", on_bar = "white"),
  text = list(),
  text_mapping = aes(),
  mode = "distinct",
  position = position_stack(),
  ...
)
```

Arguments

`mapping`	additional aesthetics for `geom_bar()`
`counts`	whether to display count number labels above the bars
`bar_number_threshold`	if less than one, labels for bars height greater than this threshold will be placed on (not above) the bars
`text_colors`	a name vector of characters specifying the color when `on_background` and `on_bar` (see `bar_number_threshold`)
`text`	additional parameters passed to `geom_text()`
`text_mapping`	additional aesthetics for `geom_text()`
`mode`	region selection mode, defines which intersection regions will be accounted for when computing the size. See `get_size_mode()` for accepted values.
`position`	position passed to `geom_bar()`
`...`	Arguments passed on to `ggplot2::geom_bar`

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`width` Bar width. By default, set to 90% of the resolution of the data.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`orientation` The orientation of the layer. The default (`NA`) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

`stat` Override the default connection between `geom_bar()` and `stat_count()`.

`reverse_log_trans` *Logarithmic scale for use with* `upset_set_size()`

Description

Inspired by Brian Diggs' answer which is CC-BY-SA 4.0.

Usage

```
reverse_log_trans(base = 10)
```

Arguments

`base`	logarithm base (default 10)

`scale_color_venn_mix` *Color scale for Venn diagram*

Description

Color scale for Venn diagram

Usage

```
scale_color_venn_mix(
  data,
  sets = NULL,
  colors = c("red", "blue", "green"),
  na.value = "grey40",
  highlight = NULL,
  active_color = "orange",
  inactive_color = "NA",
  scale = scale_color_manual,
  ...
)
```

Arguments

data	a dataframe including binary columns representing membership in sets
sets	vector with names of columns representing membership in sets
colors	named list of colors for sets (one set=one color)
na.value	value for elements not belonging to any of the sets
highlight	which regions of the diagram to highlight
active_color	color for highlight
inactive_color	color for lack of highlight
scale	the base scale (default=scale_color_manual())
...	Arguments passed on to ggplot2::scale_color_manual

values a set of aesthetic values to map data values to. The values will be matched in order (usually alphabetical) with the limits of the scale, or with breaks if provided. If this is a named vector, then the values will be matched based on the names instead. Data values that don't match will be given na.value.

aesthetics Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the colour and fill aesthetics at the same time, via aesthetics = c("colour", "fill").

breaks One of:

- NULL for no breaks
- waiver() for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output

scale_fill_venn_mix *Fill scale for Venn diagram*

Description

Fill scale for Venn diagram

Usage

```
scale_fill_venn_mix(..., na.value = "NA")
```

Arguments

`...`	Arguments passed on to `scale_color_venn_mix` `data` a dataframe including binary columns representing membership in sets `sets` vector with names of columns representing membership in sets `colors` named list of colors for sets (one set=one color) `highlight` which regions of the diagram to highlight `active_color` color for highlight `inactive_color` color for lack of highlight `scale` the base scale (default=`scale_color_manual()`)
`na.value`	value for elements not belonging to any of the known sets

upset *Compose an UpSet plot*

Description

Compose an UpSet plot

Usage

```
upset(
  data,
  intersect,
  base_annotations = "auto",
  name = "group",
  annotations = list(),
  themes = upset_themes,
  stripes = upset_stripes(),
  labeller = identity,
  height_ratio = 0.5,
  width_ratio = 0.3,
  wrap = FALSE,
```

```
  set_sizes = upset_set_size(),
  mode = "distinct",
  queries = list(),
  guides = NULL,
  encode_sets = TRUE,
  matrix = intersection_matrix(),
  ...
)
```

Arguments

data	a dataframe including binary columns representing membership in classes
intersect	which columns should be used to compose the intersection
base_annotations	a named list with default annotations (i.e. the intersection size barplot)
name	the label shown below the intersection matrix
annotations	a named list of annotations, each being a list with: list(aes=mapping, geom=geom or list of geoms); • (optional) highlight_geom=list of geoms geoms which can be highlighted with queries, • (optional) top_geom=list of geoms which should show up on top of highlighted queries.
themes	a named list of themes for components and annotations, see upset_default_themes()/upset_modify_t
stripes	specification of the stripes appearance created with upset_stripes()
labeller	function modifying the names of the sets (rows in the matrix)
height_ratio	ratio of the intersection matrix to intersection size height
width_ratio	ratio of the overall set size width to intersection matrix width
wrap	whether the plot should be wrapped into a group (makes adding a tile/combining with other plots easier)
set_sizes	the overall set sizes plot, e.g. from upset_set_size() (FALSE to hide)
mode	region selection mode for computing the number of elements in intersection fragment. See get_size_mode() for accepted values.
queries	a list of queries generated with upset_query()
guides	action for legends aggregation and placement ('keep', 'collect', 'over' the set sizes)
encode_sets	whether set names (column in input data) should be encoded as numbers (set to TRUE to overcome R limitations of max 10 kB for variable names for datasets with huge numbers of sets); default TRUE for upset() and FALSE for upset_data().
matrix	the intersection matrix plot
...	Arguments passed on to upset_data min_size minimal number of observations in an intersection for it to be included max_size maximal number of observations in an intersection for it to be included

`min_degree` minimal degree of an intersection for it to be included

`max_degree` maximal degree of an intersection for it to be included

`n_intersections` the exact number of the intersections to be displayed; n largest intersections that meet the size and degree criteria will be shown

`keep_empty_groups` whether empty sets should be kept (including sets which are only empty after filtering by size)

`warn_when_dropping_groups` whether a warning should be issued when empty sets are being removed

`warn_when_converting` whether a warning should be issued when input is not boolean

`sort_sets` whether to sort the rows in the intersection matrix (descending sort by default); one of: `'ascending'`, `'descending'`, `FALSE`

`sort_intersections` whether to sort the columns in the intersection matrix (descending sort by default); one of: `'ascending'`, `'descending'`, `FALSE`

`sort_intersections_by` the mode of sorting, the size of the intersection (cardinality) by default; one of: `'cardinality'`, `'degree'`, `'ratio'`, or any combination of these (e.g. `c('degree', 'cardinality')`)

`sort_ratio_numerator` the mode for numerator when sorting by ratio

`sort_ratio_denominator` the mode for denominator when sorting by ratio

`group_by` the mode of grouping intersections; one of: `'degree'`, `'sets'`

`size_columns_suffix` suffix for the columns to store the sizes (adjust if conflicts with your data)

`intersections` whether only the intersections present in data (`observed`, default), or all intersections (`all`) should be computed; using all intersections for a high number of sets is not computationally feasible - use `min_degree` and `max_degree` to narrow down the selection; this is only useful for modes different from the default exclusive intersection. You can also provide a list with a custom selection of intersections (order is respected when you set `sort_intersections=FALSE`)

`max_combinations_datapoints_n` a fail-safe limit preventing accidental use of `intersections='all'` with a high number of sets and observations

---

`upset_annotate` *Annotation panel shorthand*

Description

Simplifies creation of annotation panels, automatically building aesthetics mappings, at a cost of lower flexibility than when providing a custom mapping; `aes(x=intersection)` is prespecified.

Usage

```
upset_annotate(y, geom)
```

Arguments

y	A string with the name of the y aesthetic
geom	A geom to be used as an annotation

---

upset_data *Prepare data for UpSet plots*

Description

Prepare data for UpSet plots

Usage

```
upset_data(
  data,
  intersect,
  min_size = 0,
  max_size = Inf,
  min_degree = 0,
  max_degree = Inf,
  n_intersections = NULL,
  keep_empty_groups = FALSE,
  warn_when_dropping_groups = FALSE,
  warn_when_converting = "auto",
  sort_sets = "descending",
  sort_intersections = "descending",
  sort_intersections_by = "cardinality",
  sort_ratio_numerator = "exclusive_intersection",
  sort_ratio_denominator = "inclusive_union",
  group_by = "degree",
  mode = "exclusive_intersection",
  size_columns_suffix = "_size",
  encode_sets = FALSE,
  max_combinations_datapoints_n = 10^10,
  intersections = "observed"
)
```

Arguments

data	a dataframe including binary columns representing membership in classes
intersect	which columns should be used to compose the intersection
min_size	minimal number of observations in an intersection for it to be included
max_size	maximal number of observations in an intersection for it to be included
min_degree	minimal degree of an intersection for it to be included
max_degree	maximal degree of an intersection for it to be included

n_intersections
: the exact number of the intersections to be displayed; n largest intersections that meet the size and degree criteria will be shown

keep_empty_groups
: whether empty sets should be kept (including sets which are only empty after filtering by size)

warn_when_dropping_groups
: whether a warning should be issued when empty sets are being removed

warn_when_converting
: whether a warning should be issued when input is not boolean

sort_sets
: whether to sort the rows in the intersection matrix (descending sort by default); one of: 'ascending', 'descending', FALSE

sort_intersections
: whether to sort the columns in the intersection matrix (descending sort by default); one of: 'ascending', 'descending', FALSE

sort_intersections_by
: the mode of sorting, the size of the intersection (cardinality) by default; one of: 'cardinality', 'degree', 'ratio', or any combination of these (e.g. c('degree', 'cardinality'))

sort_ratio_numerator
: the mode for numerator when sorting by ratio

sort_ratio_denominator
: the mode for denominator when sorting by ratio

group_by
: the mode of grouping intersections; one of: 'degree', 'sets'

mode
: region selection mode for sorting and trimming by size. See get_size_mode() for accepted values.

size_columns_suffix
: suffix for the columns to store the sizes (adjust if conflicts with your data)

encode_sets
: whether set names (column in input data) should be encoded as numbers (set to TRUE to overcome R limitations of max 10 kB for variable names for datasets with huge numbers of sets); default TRUE for upset() and FALSE for upset_data()

max_combinations_datapoints_n
: a fail-safe limit preventing accidental use of intersections='all' with a high number of sets and observations

intersections
: whether only the intersections present in data (observed, default), or all intersections (all) should be computed; using all intersections for a high number of sets is not computationally feasible - use min_degree and max_degree to narrow down the selection; this is only useful for modes different from the default exclusive intersection. You can also provide a list with a custom selection of intersections (order is respected when you set sort_intersections=FALSE)

upset_default_themes *Default themes modified by specified arguments*

Description

Return the default UpSet themes with all themes modified with provided arguments

Usage

```
upset_default_themes(...)
```

Arguments

...	arguments passed to theme()

upset_mode *Layer defining the intersection mode for the data to be displayed*

Description

By default the annotations are given data corresponding to the same mode as the mode of the passed in the upset() call.

Usage

```
upset_mode(mode)
```

Arguments

mode	region selection mode, defines which mode data will be made available for the annotation. See get_size_mode() for accepted values.

upset_modify_themes *Default themes modified by specified component-specific arguments*

Description

Return the default UpSet themes with specific themes modified with provided themes

Usage

```
upset_modify_themes(to_update)
```

Arguments

to_update	a named list of themes to be used to modify themes of specific components; see names(upset_themes) for components names.

upset_query *Highlight chosen sets or intersections*

Description

Highlight sets or intersections matching specified query.

Usage

```
upset_query(
  set = NULL,
  intersect = NULL,
  group = NULL,
  only_components = NULL,
  ...
)
```

Arguments

set	name of the set to highlight
intersect	a vector of names for the intersection to highlight; pass NA to select the empty intersection
group	name of the set to highlight when using group_by='sets'
only_components	which components to modify; by default all eligible components will be modified; the available components are 'overall_sizes', 'intersections_matrix', 'Intersection size', and any annotations specified
...	• passed to geoms in modified components

Examples

```
upset_query(intersect=c('Drama', 'Comedy'), color='red', fill='red')
upset_query(set='Drama', fill='blue')
```

upset_set_size *Prepare layers for sets sizes plot*

Description

Prepare layers for sets sizes plot

Usage

```
upset_set_size(
  mapping = aes(),
  geom = geom_bar(width = 0.6),
  position = "left",
  filter_intersections = FALSE
)
```

Arguments

mapping	additional aesthetics
geom	a geom to use
position	on which side of the plot should the set sizes be displayed ('left' or 'right')
filter_intersections	whether the intersections filters (e.g. n_intersections or min_size) should influence displayed set sizes

---

upset_stripes *Define appearence of the stripes*

Description

Define appearence of the stripes

Usage

```
upset_stripes(
  mapping = aes(),
  geom = geom_segment(size = 7),
  colors = c("white", "grey95"),
  data = NULL
)
```

Arguments

mapping	additional aesthetics
geom	a geom to use, should accept x, y, xend, yend and color aesthetics
colors	a vector of colors to repeat as many times as needed for the fill of stripes, or a named vector specifying colors for values of the variable mapped to the color aesthetics in the mapping argument
data	the dataset describing the sets with a column named set and any other columns as needed for mapping

upset_test *Test for differences between intersections*

Description

This is a wrapper around `compare_between_intersections()`, adding sorting by FDR, warnings, etc.

Usage

```
upset_test(data, intersect, ...)
```

Arguments

`data`	a dataframe including binary columns representing membership in classes
`intersect`	which columns should be used to compose the intersection
`...`	Arguments passed on to `compare_between_intersections` `test` the default test function; it is expected to accept `formula` and `data` parameters, and a list with `p.value`, `statistic`, and `method` `tests` a named list with tests for specific variables, overwriting the default test `ignore` a list with names of variables to exclude from testing `ignore_mode_columns` whether the membership columns and size columns for all modes should be ignored `mode` region selection mode; note that modes other than `exclusive_intersection` repeat observations in different test group, introducing dependencies. See `get_size_mode()` for accepted values.

upset_text_percentage *Generate percentage label of the intersection/union sizes ratio*

Description

For use together with `intersection_size` or `intersection_ratio`

Usage

```
upset_text_percentage(digits = 0, sep = "", mode = "distinct")
```

Arguments

`digits`	How many digits to show when rounding the percentage?
`sep`	set to space (`' '`) if you prefer a whitespace between the number and the `\%` sign.
`mode`	region selection mode for computing the numerator in ratio. See `get_size_mode()` for accepted values.

Examples

```
ggplot2::aes(label=!!upset_text_percentage())
```

`upset_themes` *List of default themes for upset components*

Description

List of default themes for upset components

Usage

```
upset_themes
```

Format

An object of class `list` of length 4.

Index