# Package 'braidReports'

May 8, 2026

**Title** Visualize Combined Action Response Surfaces and Report BRAID Analyses

**Version** 1.0.5

**Description** Provides functions to visualize combined action data in 'ggplot2'. Also provides functions for producing full BRAID analysis reports with custom layouts and aesthetics, using the BRAID method originally described in Twarog et al. (2016) <doi:10.1038/srep25523>.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Depends** braidrm (>= 1.0.0), ggplot2, R (>= 3.0)

**Imports** basicdrm, cowplot, grid, gtable, scales, utils

**Suggests** knitr, rmarkdown, MASS

**VignetteBuilder** knitr

**LazyData** true

**NeedsCompilation** no

**Author** Anang Shelat [aut], Nathaniel R. Twarog [aut, cre]

**Maintainer** Nathaniel R. Twarog <nathaniel.twarog@stjude.org>

**Repository** CRAN

**Date/Publication** 2025-09-03 18:30:08 UTC

## Contents

---

`basicBraidAnalysis` *Basic BRAID Analysis Conversion*

---

## Description

Basic BRAID Analysis Conversion

## Usage

```
basicBraidAnalysis(bfit)
```

## Arguments

| | |
|---|---|
| `bfit` | A BRAID fit object of class `braidrm` |

## Details

While we strongly recommend using the `runBraidAnalysis()` function for a more complete treatment of a combination; there may be circumstances in which is necessary or preferable to use an existing BRAID fit object (of class `braidrm`). Thsi function takes such a fit and wraps it in a minimal `braidAnalysis` object which can then be passed to `makeBraidReport()`

## Value

A BRAID analysis object of class `braidanalysis` containin only the results from the given BRAID fit

## Examples

```
surface <- antagonisticExample
fit <- braidrm(measure ~ concA + concB, surface, model="kappa2")

analysis <- basicBraidAnalysis(fit)

names(analysis)
```

basicBraidPlot *Wrapper for Basic Braid Plot*

## Description

Wrapper for Basic Braid Plot

## Usage

```
basicBraidPlot(
  data,
  mapping,
  palette = "RdYlBu",
  direction = -1,
  logscale = TRUE
)
```

## Arguments

| | |
|---|---|
| data | Dataset to use for plot. If not already a data.frame, will be converted to one by fortify() |
| mapping | List of aesthetic mappings to use for plot. geom_braid requires at least the following aesthetics: x, y, and fill. |
| palette | A Brewer color palette passed to the scale_fill_distiller() function |
| direction | The direction of the Brewer color palette, also passed to scale_fill_distiller() |
| logscale | Should x and y variables be plotted on a log scale (defaults to TRUE). If FALSE, scale_x_continuous() and and scale_y_continuous() will still be applied to ensure that a plot produced by this function always has explicitly given scales. |

## Details

Though the geom_braid() function allows for a great deal of flexibility and keeps with the layer-based ethos of ggplot, we find that very often, it is useful to just get a simple first pass view of a response surface, and repeatedly typing out all the necessary layers to make the surface visible can be extremely tiresome. This function wraps a simple set of ggplot functions with the geom_braid() function to produce a standard heatmap plot that works for many of the surfaces we encounter.

Note however that this is just meant to be a convenience function. Any more complex or sophisticated plots should be built using ggplot layers, including geom_braid().

## Value

An object of class ggplot containing a geom_braid() rendering of the respective x, y, and fill aesthetics.

## Examples

```
surface <- synergisticExample
plot <- basicBraidPlot(surface,aes(x=concA,y=concB,fill=measure))
print(plot)
```

---

geom_braid *BRAID Heatmaps*

---

## Description

Summarize and plot measurements of two inputs as a discrete raster or "stained-glass" plot

## Usage

```
geom_braid(
  mapping = NULL,
  data = NULL,
  stat = "braid",
  position = "identity",
  ...,
  space = 1.5,
  trim = TRUE,
  shared = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_braid(
  mapping = NULL,
  data = NULL,
  geom = "tile",
  position = "identity",
  ...,
  space = 1.5,
  trim = TRUE,
  shared = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_braid_glass(
  mapping = NULL,
  data = NULL,
  stat = "braid_glass",
  position = "identity",
```

```
  ...,
  space = 1.5,
  trim = TRUE,
  shared = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_braid_glass(
  mapping = NULL,
  data = NULL,
  geom = "polygon",
  position = "identity",
  ...,
  space = 1.5,
  trim = TRUE,
  shared = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

## Arguments

| | |
|---|---|
| mapping | Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. |
| data | The data to be displayed in this layer. There are three options:<br>If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().<br>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.<br>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)). |
| stat | The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following:<br>• A Stat ggproto subclass, for example StatCount.<br>• A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".<br>• For more information and other ways to specify the stat, see the layer stat documentation. |

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

... — Additional parameters to be passed to ggplot2::geom_tile()

space — Parameter specifying the separation between marginal tiles and the main grid. Describes the distance from the center of the marginal tile to the center of the nearest main grid tile, divided by the width or height of the tile. If a single value is provided, it is used for both left-right and top-bottom margin tiles. If two values are provided, the first is used for left-right margin tiles and the second is used for top-bottom margin tiles.

trim — Should values that are finite in one dimension be dropped if their finite coordinates lie outside the bounds of the main grid?

shared — Should marginal offsets and trimming be calculated separately for each facet if plots are faceted. If FALSE, the default, each facet will have its own bounds and marginal offsets; if TRUE, offsets will be calculated for the full data and shared across all facets.

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 — The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following:

- A Geom ggproto subclass, for example GeomPoint.
- A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".
- For more information and other ways to specify the geom, see the layer geom documentation.

## Details

While the existing ggplot2 package includes several functions that are extremely effective and versatile for visualizing two-dimensional responses, including ggplot2::geom_raster(), ggplot2::geom_tile(),

and ggplot2::geom_contour(), a number of considerations particular to combination data make these functions, as is, somewhat difficult to use. First, these functions are not designed for data in which pairs of x- and y-coordinates are duplicated; yet this is very common in experimental data. While such duplications can be handled prior to calling a visualization function, handling them automatically reduces the barrier to plotting.

A second, and much more challenging consideration, is that for many drug combination studies, drug concentrations are measured as a series of equal ratio dilutions; visualizing such doses is most intuitive on a logarithmic scale. But when inputs are scaled logarithmically, zeros become infinite and are automatically removed by nearly all ggplot2 functions. This makes it very difficult to plot measurements of drugs in isolation and in combination in the same plot. geom_braid addresses this by automatically offsetting any measurements whose transformed coordinates are infinite to margins within the plotted space, so that all values can be plotted together.

While geom_braid is suitable for most response surfaces, some surfaces feature measurements that are not arranged in a evenly spaced checkerboard. For such surfaces, geom_braid_glass produces a set of Voronoi polygons centered on the available transformed coordinates, creating what we call a "stained glass" plot. Marginal points are still represented by rectangles, but with width and height such that boundaries are equidistant between adjacent points.

stat_braid and stat_braid_glass are simply the corresponding stat_ functions for these two functions.

## Examples

```
concentrations <- c(0,2^(-3:3))
surface <- data.frame(
  concA = rep(rep(concentrations,each=length(concentrations)),each=3),
  concB = rep(rep(concentrations,times=length(concentrations)),each=3),
  replicate = rep(c(1,2,3),times=(length(concentrations)^2))
)
surface$actual <- evalBraidModel(
  surface$concA,
 surface$concB,
 c(1, 1, 3, 3, 2, 0, 100, 100, 100)
)
surface$measure <- surface$actual + rnorm(nrow(surface),sd=7)

ggplot(surface,aes(x=concA,y=concB))+
    geom_braid(aes(fill=measure))+
    scale_x_log10()+
    scale_y_log10()+
    scale_fill_distiller(palette="RdYlBu")+
    coord_equal()+
    labs(x="Drug A",y="Drug B",fill="Effect")

glassSurface <- surface
glassSurface$concA[glassSurface$replicate==2] <-
    glassSurface$concA[glassSurface$replicate==2]*1.25
glassSurface$concB[glassSurface$replicate==3] <-
    glassSurface$concB[glassSurface$replicate==3]*1.25
glassSurface$actual <- evalBraidModel(
    glassSurface$concA,
```

```
    glassSurface$concB,
    c(1, 1, 3, 3, -0.5, 0, 60, 100, 100)
)
glassSurface$measure <- glassSurface$actual+rnorm(nrow(glassSurface),sd=7)

ggplot(glassSurface,aes(x=concA,y=concB))+
    geom_braid_glass(aes(fill=measure))+
    scale_x_log10("Drug A")+
    scale_y_log10("Drug B")+
    scale_fill_distiller("Effect",palette="RdYlBu")+
    coord_equal()

glassSurface$tilewidth <- log10(2)*0.9
glassSurface$tilewidth[glassSurface$concA==0] <- log10(2)/2

glassSurface$tileheight <- log10(2)*0.9
glassSurface$tileheight[glassSurface$concB==0] <- log10(2)/2

ggplot(glassSurface,aes(x=concA,y=concB))+
    geom_braid_glass(aes(fill=measure,width=tilewidth,height=tileheight),space=2)+
    scale_x_log10("Drug A")+
    scale_y_log10("Drug B")+
    scale_fill_distiller("Effect",palette="RdYlBu")+
    coord_equal()
```

---

geom_braid_contour *Smoothed BRAID Surface Contours*

---

## Description

Generate contours of a smoothed two-dimensional response surface

## Usage

```
geom_braid_contour(
  mapping = NULL,
  data = NULL,
  stat = "braid_contour",
  position = "identity",
  ...,
  bins = NULL,
  binwidth = NULL,
  breaks = NULL,
  npoints = 50,
  tight = FALSE,
  trim = TRUE,
  shared = FALSE,
  na.rm = FALSE,
  show.legend = NA,
```

```
  inherit.aes = TRUE
)

stat_braid_contour(
  mapping = NULL,
  data = NULL,
  geom = "contour",
  position = "identity",
  ...,
  bins = NULL,
  binwidth = NULL,
  breaks = NULL,
  npoints = 50,
  tight = FALSE,
  trim = TRUE,
  shared = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

## Arguments

| | |
|---|---|
| mapping | Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. |
| data | The data to be displayed in this layer. There are three options:<br>If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().<br>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.<br>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)). |
| stat | The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following:<br>• A Stat ggproto subclass, for example StatCount.<br>• A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".<br>• For more information and other ways to specify the stat, see the layer stat documentation. |
| position | A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: |

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`bins` Number of contour bins. Overridden by `breaks`.

`binwidth` The width of the contour bins. Overridden by `bins`.

`breaks` One of:

- Numeric vector to set the contour breaks
- A function that takes the range of the data and binwidth as input and returns breaks as output. A function can be created from a formula (e.g. ~ fullseq(.x, .y)).

Overrides `binwidth` and `bins`. By default, this is a vector of length ten with `pretty()` breaks.

`npoints` The number of interpolated values in the x- and y- directions that are used to generate the smoothed raster function

`tight` If true, the generated raster will span the precise range of transformed and plotted data; this will produce a range of tiles that are strictly smaller than those produced by geom_braid (as those tiles extend above and below the plotted tile centers). If `FALSE` (the default), the interpolated values will be selected to span the same (slightly larger) range of values that would be covered by running geom_braid with the same parameters.

| | |
|---|---|
| trim | Should values that are finite in one dimension be dropped if their finite coordinates lie outside the bounds of the main grid? |
| shared | Should marginal offsets and trimming be calculated separately for each facet if plots are faceted. If FALSE, the default, each facet will have its own bounds and marginal offsets; if TRUE, offsets will be calculated for the full data and shared across all facets. |
| 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 | The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following:<br>• A Geom ggproto subclass, for example GeomPoint.<br>• A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".<br>• For more information and other ways to specify the geom, see the layer geom documentation. |

## Details

When evaluating a plotted response surface it is often more effective to plot the precise contours at which a set of levels is reached by the combination. Because ggplot2::stat_contour requires that data lie in an evenly space raster grid (and does not support duplicated values), it is difficult to apply to more discrete or irregularly sampled data. This function uses the same smoothing and interpolation utilities as geom_braid_smooth to preprocess and smooth data, which is then passed to the contour estimation code of ggplot2::stat_contour, producing contours which are smoothed and sufficiently regularly spaced.

## Examples

```
surface <- antagonisticExample

ggplot(surface,aes(x=concA,y=concB))+
   geom_braid_smooth(aes(fill=measure))+
   geom_braid_contour(aes(z=measure),colour="black",breaks=((1:9)/10))+
    scale_x_log10("Drug A")+
    scale_y_log10("Drug B")+
    scale_fill_distiller("Effect",palette="RdYlBu")+
    coord_equal()
```

geom_braid_smooth *Smoothed BRAID Surfaces*

## Description

Summarize and plot measurements of two inputs as a smoothed response surface

## Usage

```
geom_braid_smooth(
  mapping = NULL,
  data = NULL,
  stat = "braid_smooth",
  position = "identity",
  ...,
  space = 1.5,
  trim = TRUE,
  shared = FALSE,
  npoints = 50,
  tight = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_braid_smooth(
  mapping = NULL,
  data = NULL,
  geom = "tile",
  position = "identity",
  ...,
  space = 1.5,
  trim = TRUE,
  shared = FALSE,
  npoints = 50,
  tight = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

## Arguments

| | |
|---|---|
| mapping | Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping. |

`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)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used the override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Additional parameters to be passed to `ggplot2::geom_tile()`

`space` Parameter specifying the separation between marginal tiles and the main grid. Describes the distance from the center of the marginal tile to the center of the nearest main grid tile, divided by the width or height of the tile. If a single value is provided, it is used for both left-right and top-bottom margin tiles. If two values are provided, the first is used for left-right margin tiles and the second is used for top-bottom margin tiles.

`trim` Should values that are finite in one dimension be dropped if their finite coordinates lie outside the bounds of the main grid?

`shared` Should marginal offsets and trimming be calculated separately for each facet if plots are faceted. If `FALSE`, the default, each facet will have its own bounds and marginal offsets; if `TRUE`, offsets will be calculated for the full data and shared across all facets.

`npoints` The number of interpolated values in the x- and y- directions that are used to generate the smoothed raster function

| | |
|---|---|
| tight | If true, the generated raster will span the precise range of transformed and plotted data; this will produce a range of tiles that are strictly smaller than those produced by geom_braid (as those tiles extend above and below the plotted tile centers). If FALSE (the default), the interpolated values will be selected to span the same (slightly larger) range of values that would be covered by running geom_braid with the same parameters. |
| 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 | The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following:<br>• A Geom ggproto subclass, for example GeomPoint.<br>• A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".<br>• For more information and other ways to specify the geom, see the layer geom documentation. |

## Details

Like geom_braid, this function involves several pre-processing steps to allow quick visualization of drug combination data. These steps include summarizing duplicated measurements and offsetting non-finite plotted coordinates. In addition to these steps, geom_braid_smooth generates a regular, densely sampled grid of coordinates and smoothly interpolates the given data to produces a smoothed raster heatmap. Smoothing in the x- and y- directions is governed by the width and height aesthetic respectively; if these aesthetics are not included, they are estimated from the minimum spacing of the data.

## Examples

```
surface <- synergisticExample

ggplot(surface,aes(x=concA,y=concB))+
   geom_braid_smooth(aes(fill=measure))+
    scale_x_log10("Drug A")+
    scale_y_log10("Drug B")+
    scale_fill_distiller("Effect",palette="RdYlBu")+
    coord_equal()
```

## kappa_trans *BRAID kappa Transforms*

### Description

Functions to linearize the BRAID interaction parameter kappa, which ordinarily ranges from -2 to infinity. kappa_trans produces a scales transform object which can be used in ggplot2 continuous scale object. scale_x_kappa and scale_y_kappa are wrappers for scale_x_continuous and scale_y_continuous which set the trans or transform parameter to kappa_trans().

### Usage

```
kappa_trans()

scale_x_kappa(...)

scale_y_kappa(...)
```

### Arguments

| | |
|---|---|
| ... | Additional parameter to be passed to scale_*_continuous |

### Value

For kappa_trans a scales transform object. For scale_*_kappa, a continuous position scale layer for a ggplot object.

### Examples

```
transform <- kappa_trans()

transform$transform(c(-1.96, 100))
transform$inverse(c(-1, 1))

data <- merckValues_stable
ggplot(data,aes(x=kappa))+
    geom_density()+
    scale_x_kappa("BRAID kappa")
```

## makeBraidReport *Render a BRAID Report*

### Description

Produces a one page report depicting the results of a full BRAID analysis for a single combination.

## Usage

```
makeBraidReport(analysis, compounds, levels, limits, control = list())
```

## Arguments

| | |
|---|---|
| analysis | An object of class braidAnalysis produced by the runBraidAnalysis() or basicBraidAnalysis() functions |
| compounds | A length-2 character vector containing the names of the two compounds tested in the combination |
| levels | Two levels at which the IAE should be evaluated |
| limits | Two values representing the maximal achievable concentrations for the compounds tested, used to estimate the IAE |
| control | A named list of additional control parameters adjusting the appearance of the resulting report |

## Details

This function attempts, however foolhardily, to encompass many of the details, plots, and values that the user might wish to report for a complete BRAID analysis of a given drug combination. All reports are built for a single 8.5-by-11 inch page, either in landscape or portrait orientation, but reports can be customized to contain more or less information. Here is a full list of what *can* appear in the BRAID report:

- A raw and smoothed plot of the actual measured response data; the raw plot is built using geom_braid() or, for irregularly laid out data, geom_braid_glass(), while the smoothed data is built using geom_braid_smooth(). (Included in all layouts)
- A plot of residual errors and a smoothed surface plot of the predicted additive surface based on the dose response behavior of the individual compounds alone. In cases of pronounced interaction, this will differ significantly from the best-fit BRAID plots. (Included in the dense layout)
- A plot of residual errors and a smoothed surface plot of the best-fitting BRAID surface. (Included in the all layouts)
- A table of the best-fitting BRAID response surface parameters (Included in all layouts)
- A table of estimated IAE values at the specified effect levels (Included in all layouts)
- Two tables of the dose required of one drug to produce a desired effect level (the first value in levels) in the presence of several doses of the other drug; used to gauge the degree of potentiation. (Included in the standard and dense layouts)
- Two plots depicting the predicted dose response of one drug in the presence of various levels of the other, also used to gauge potentiation. (Included in the standard and dense layouts)

So the resulting report page can contain from six (simple layout) to twelve (dense layout) elements depicting different aspects of the BRAID analysis.

The precise appearance of the report page is controlled by various elements of the control parameter. Though the default value of the parameter is an empty list, several fields will be filled in if they are unspecified. The full set of possible control options is:

- abbs: A pair of character strings specifying the abbreviations of the tested compounds. By default, the abbreviations consist of the firs three characters of each compound's name, but for some compound names this is not an appropriate abbreviation Abbreviations are used in many axis labels and tables
- units: If included, a single string or pair of strings specifying the dose units for the two drugs tested, included in axis labels and tables. If left unspecified, units will not be included
- xscale: Either a character string specifying a named transformation object (e.g "log2", "log10", "sqrt") or a ggplot2 continuous x-position scale generated by ggplot2::scale_x_continuous() or related functions. This scale will be applied to the x-dimension of all surface plots and the x-dimension of the first potentiation plot. If a name is specified for the scale, this will be the x-axis label; otherwise other defaults will be used. Default value for this control parameter is "log10".
- yscale: Either a character string specifying a named transformation object (e.g "log2", "log10", "sqrt") or a ggplot2 continuous y-position scale generated by ggplot2::scale_y_continuous() or related functions. This scale will be applied to the y-dimension of all surface plots and the x-dimension of the second potentiation plot. If a name is specified for the scale, this will be the y-axis label; otherwise other defaults will be used. Default value for this control parameter is "log10".
- fillscale: If included, continuous fill scale object generated by one of several ggplot2 continuous fill scale functions. This fill scale will control the fill appearance for all *effect* surface plots; fill colors in residual error plots will use a different color palette. In addition, any names, labels, breaks, transformations, etc. included in this scale will also be applied to the y-axis of both potentiation plots, as those also represent the modeled effect. If unspecified, will be set to a brewer continuous color scale with palette "RdYlBu".
- colorscale: If included, a discrete color scale object generated by one of several ggplot2 discrete color scale functions. This color scale controls the colors chosen for the curves in the two potentiation plots. If left unspecified, will default to the output of ggplot2::scale_color_discrete()
- xname: A string specifying the desired x-axis labels in surface plots. Will be overridden if control parameter xscale is a scale object with a non-empty name attribute. If unspecified, defaults the abbreviation of the first compound followed by the units if included.
- yname: A string specifying the desired y-axis labels in surface plots. Will be overridden if control parameter yscale is a scale object with a non-empty name attribute. If unspecified, defaults the abbreviation of the second compound followed by the units if included.
- effectname: The name of the modeled effect variable. Could be "Growth" or "Survival" or "Activity". The default value is simply "Effect"
- title: A string containing the overall title of the report page. If left unspecified, will simply be the first compound "vs." the second
- contourcolor: Contours of the smoothed surfaces at the levels specified by the parameter levels are included in all smoothed plots. By default, they are black, but specifying this control parameter will change that color
- irregular: If TRUE, the data are not assumed to lie on a regular grid in the plotted, and will be visualized with geom_braid_glass() rather than geom_braid()
- swidth: A numeric value specifying how widely the smooth surfaces should be smoothed, passed as the width aesthetic to geom_braid_smooth()

- sheight: A numeric value specifying how far in the y-direction the smooth surfaces should be smoothed, passed as the height aesthetic to geom_braid_smooth()
- npoints: The density of points used in the smoothed plots. See geom_braid_smooth() for details
- leveltext: A pair of strings indicating how the two IAE levels should be displayed in tables. In some cases, the precise number at which the IAE is calculated does not reflect the level that the user wishes to express. So a user might want to refer to a relative survival value of 0.1 as IAE90 (for 90% killing); or a log2-fold growth inhibition as IAE50 (for 50% inhibition); passing "50" or "90" as the leveltext in such cases will produce the desired appearance. If left unspecified, labels will simply be the string representations of the paramter levels
- metrics: A named numeric or named character vector specifying additional metrics for the combination; they will be added to the table containing the calculated IAE values. Numeric values will be rounded to three significant figures; string values will be included as is. Names will be parsed by default, so the string "CI[90]" will be displayed with the "90" as a subscript
- surftheme: A ggplot2 theme object specifying any additional theme adjustments to add to all response surface plots
- curvetheme: A ggplot2 theme object specifying any additional theme adjustments to add to both potentiation curve plots
- layout: The specific layout to be used; determines which report elements are included. Can be "simple", "standard" (the default), or "dense"
- orientation: The expected orientantion of the rendered page; can be "portrait" (the default) or "landscape"

## Value

A graphical object containing all plots and tables, arranged according to the desired format. The resulting object is optimized for a single page, either portrait or landscape as specified in control

## Examples

```
surface <- synergisticExample
analysis <- runBraidAnalysis(measure~concA+concB, surface,
                             defaults=c(0,1), getCIs=FALSE)

report <- makeBraidReport(analysis,c("A Drug","B Drug"),
                          levels=c(0.5, 0.9),limits=c(5,5))
print(report)

control <- list(abbs=c("A","B"),units=c("\u00B5M"),leveltext=c("50","90"),
                xscale=scale_x_log10(breaks=c(0.1,0.5,2,10),
                labels=as.character),
                fillscale=scale_fill_viridis_c(option="A"),
                colorscale=scale_color_brewer(palette="Set1"),
                title="Example Analysis")
nextReport <- makeBraidReport(analysis,c("A Drug","B Drug"),
                              levels=c(0.5, 0.9),limits=c(5,5),
                              control=control)
print(nextReport)
```

## merckValues_stable *Best-Fit Bayesian Stabilized Merck OPPS BRAID Values*

### Description

A table of BRAID kappa and IAE values resulting from running the version 1.0.0 BRAID fitting code on the Merck oncopolypharmacology screen (OPPS), with moderate Bayesian stabilization of kappa.

### Usage

```
merckValues_stable
```

### Format

A data frame with 5 columns:

**cell_line** The cancer cell line which the combination was tested

**drugA** The first drug in the combination tested

**drugB** The second drug in the combination tested

**kappa** The best-fit value of the BRAID interaction parameter kappa with moderate Bayesian stabilizatoin

**IAE** The index of achievable efficacy (or IAE, a BRAID measure of combined potency) estimated for the best-fit BRAID surface

## merckValues_unstable *Best-Fit Unstabilized Merck OPPS BRAID Values*

### Description

A table of BRAID kappa and IAE values resulting from running the version 1.0.0 BRAID fitting code on the Merck oncopolypharmacology screen (OPPS), with *no* Bayesian stabilization of kappa.

### Usage

```
merckValues_unstable
```

### Format

A data frame with 5 columns:

**cell_line** The cancer cell line which the combination was tested

**drugA** The first drug in the combination tested

**drugB** The second drug in the combination tested

**kappa** The best-fit value of the BRAID interaction parameter kappa with no Bayesian stabilizatoin

**IAE** The index of achievable efficacy (or IAE, a BRAID measure of combined potency) estimated for the best-fit BRAID surface

`runBraidAnalysis` *BRAID Surface Analysis*

## Description

Performs a convenient pre-built set of BRAID and dose-response analysis tasks

## Usage

```
runBraidAnalysis(
  formula,
  data,
  defaults,
  weights = NULL,
  start = NULL,
  direction = 0,
  lower = NULL,
  upper = NULL,
  useBIC = TRUE,
  ...
)

## S3 method for class 'formula'
runBraidAnalysis(
  formula,
  data,
  defaults,
  weights = NULL,
  start = NULL,
  direction = 0,
  lower = NULL,
  upper = NULL,
  useBIC = TRUE,
  ...
)

## Default S3 method:
runBraidAnalysis(
  formula,
  data,
  defaults,
  weights = NULL,
  start = NULL,
  direction = 0,
  lower = NULL,
  upper = NULL,
  useBIC = TRUE,
```

```
  ...
)
```

## Arguments

| | |
|---|---|
| formula | Either an object of class formula such as would be provided to a modeling function like stats::lm(), or a width-2 numeric array vector of concentration pairs (including 0 or Inf). A formula should specify a single output as a function of two inputs, eg. activity ~ conc1 + conc2. |
| data | If forumula is a symbolic formula, a data frame containing the specified values. If formula is a numeric array of concentrations, a numeric vector of response values, the same length as the number of rows of formula. |
| defaults | Default minimal and maximal effect values used to fix effect parameters during model selection. |
| weights | A vector of weights (between 0 and 1) the same length as the data which determines the weight with which each measurement will impact the the sum of squared errors. Weights will be multiplied by errors *before* squaring. If NULL (the default) all weights will be set to 1. Can be a numeric vector, or the name of a column in data if formula is a symbolic formula |
| start | A BRAID parameter vector specifying the first guess where the non-linear optimization should begin. May be a length 7, 8, or 9 vector, though a full length vector is always preferable. If NULL (the default), it will be estimated from the data. |
| direction | Determines the possible directionality of the BRAID model. If 0 (the default) no additional constraints are placed on the parameters. If greater than 0, the fitting will require that the maximal effects are all *greater* than or equal to the minimal effect. If less than 0, the fitting will require that all maximal effect is *less* than or equal to the minimal effect. |
| lower | A numeric vector of lower bounds on the fitted parameter values. May be the same length as the number of fitted parameters, or a full, length-9 vector. Missing or unspecified lower bounds may be included as NA or Inf; if unspecified, lower bounds on the first five parameters (IDMA, IDMB, na, nb, and kappa) will be automatically estimated from the data. Bounds on the minimal and maximal effect parameters however (E0, EfA, EfB, and Ef) will be assumed to be infinite unless specified. A value of NULL, the default, will be treated as all lower parameter bounds being unspecified. |
| upper | A numeric vector of upper bounds on the fitted parameter values. Used in the same way as lower. |
| useBIC | If TRUE (the default), the best (read: most parsimonious) model will be selected from all tested models using the Bayesian information criterion (Schwarz 1978). If FALSE the function will follow the convention of earlier versions of the braidrm package and use the Akaike information criterion (Akaike 1974). |
| ... | Additional parameters to be passed to braidrm::findBestBraid() |

## Value

An object of class braidAnalysis, containing the following values:

- concs: a width-two array containing the two tested doses for each measurement
- act: a numeric vector with as many values as concs has rows, containing the measured values for each measurement
- weights: a numeric vector of weights, the same length as act, specifying the weight given to each measurement in fitting. All weights are 1 by default
- braidFit: a fit object of class braidrm containing the best-fit BRAID surface according to the given constraints
- hillFit1: If the given data contains measurements of the first drug in isolation, those measurements are fit using basicdrm::findBestHillModel; the results of this analysis are stored as an object of class hillrm as hillFit1. If no such measurements are found, this will be NULL
- hillFit2: the corresponding fit for measurements of the second drug alone, if they are included; NULL otherwise

## Examples

```
surface <- synergisticExample

analysis <- runBraidAnalysis(measure~concA+concB, surface, defaults=c(0,1))

names(analysis)
```

# Index