Package: eulerr (via r-universe)

November 22, 2026

Title Area-Proportional Euler and Venn Diagrams

Version 8.4.3

Description Generate area-proportional Euler diagrams using numerical optimization. An Euler diagram is a generalization of a Venn diagram, relaxing the criterion that all interactions need to be represented. Diagrams may be fit with circles, ellipses, squares, and rectangles via a wide range of inputs and can be visualized in numerous ways. The fitting engine is described in Larsson and Lee (2026) <doi:10.48550/arXiv.2611.02198>.

License MIT + file LICENSE

URL https://github.com/jolars/eulerr, https://jolars.github.io/eulerr/, https://eunoia.bz

BugReports https://github.com/jolars/eulerr/issues

Depends R (>= 4.2)

Imports graphics, grDevices, grid, parallel, stats, utils

Suggests covr, knitr, lattice, pBrackets, RConics, rmarkdown, spelling, testthat

VignetteBuilder knitr

Config/rextendr/version 0.5.0

Config/roxygen2/version 8.0.0

Encoding UTF-8

Language en-US

LazyData true

SystemRequirements Cargo (Rust's package manager), rustc >= 1.88.0

NeedsCompilation yes

Author Johan Larsson [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-4029-5945>), A. Jonathan R. Godfrey [ctb], Peter Gustafsson [ctb], David H. Eberly [ctb] (geometric algorithms), Emanuel Huber [ctb] (root solver code), Florian Privé [ctb]

Maintainer Johan Larsson <johan@jolars.co>

Config/pak/sysreqs libclang-dev

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

Date/Publication 2026-11-22 00:28:43 UTC

RemoteUrl https://github.com/cran/eulerr

RemoteRef HEAD

RemoteSha c3d424c6606dc738f6ce5b210d6f31ab485852e3

Contents

error_plot . . . 2
euler . . . 3
eulergram-compose . . . 9
eulerr_options . . . 10
fitted.euler . . . 11
fruits . . . 12
organisms . . . 13
pain . . . 13
plants . . . 14
plot.euler . . . 14
plot.eulergram . . . 20
print.euler . . . 21
print.eulerr_venn . . . 22
residuals.euler . . . 23
venn . . . 23

Index **26**

`error_plot` *Error plot for* `euler` *objects*

Description

This is a diagnostic tool for evaluating the fit from a call to `euler()` visually. A color key is provided by default, which represents the chosen error metric so that one can easily detect which areas in the diagram to be skeptical about.

Usage

```
error_plot(
  x,
  type = c("regionError", "residuals"),
  quantities = TRUE,
  pal = NULL,
  ...
)
```

Arguments

x	an object of class euler, typically the result of a call to euler().
type	error metric. 'regionError' is the difference in *percentage points* from the input
quantities	whether to draw the error metric on the plot
pal	color palette for the fills in the legend
...	arguments passed down to plot.euler(). Currently, providing fills, legend, or strips are not allowed and will return a warning.

Details

Notice that this function is purely provided for diagnostic reasons and does not come with the same kind of customization that plot.euler() provides: the color legend can only be customized in regards to its color palette and another key (instead of labels) is completely turned off.

Value

Returns an object of class eulergram, which will be plotted on the device in the same manner as objects from plot.euler(). See plot.eulergram() for details.

See Also

plot.euler(), euler(), plot.eulergram()

Examples

```
error_plot(euler(organisms), quantities = FALSE)
```

---

euler *Area-proportional Euler diagrams*

Description

Fit Euler diagrams (a generalization of Venn diagrams) using numerical optimization to find exact or approximate solutions to a specification of set relationships. The shape of the diagram may be a circle, an ellipse, an axis-aligned rectangle, or an axis-aligned square.

Usage

```
euler(combinations, ...)

## Default S3 method:
euler(
  combinations,
  input = c("disjoint", "union"),
  transform = identity,
```

```
  shape = c("circle", "ellipse", "rectangle", "square", "rotated_rectangle"),
  loss = c("sum_squared", "sum_absolute", "sum_absolute_region_error",
   "sum_squared_region_error", "max_absolute", "max_squared", "root_mean_squared",
    "stress", "diag_error", "log_sum_absolute", "smooth_sum_absolute",
   "smooth_sum_absolute_region_error", "smooth_max_absolute", "smooth_max_squared",
    "smooth_diag_error", "smooth_log_sum_absolute"),
  loss_aggregator = NULL,
  complement = NULL,
  control = list(),
  ...
)

## S3 method for class 'data.frame'
euler(
  combinations,
  weights = NULL,
  by = NULL,
  sep = "_",
  factor_names = TRUE,
  ...
)

## S3 method for class 'matrix'
euler(combinations, ...)

## S3 method for class 'table'
euler(combinations, ...)

## S3 method for class 'list'
euler(combinations, ...)
```

Arguments

combinations	set relationships as a named numeric vector, matrix, or data.frame (see **methods (by class)**)
...	arguments passed down to other methods
input	type of input: disjoint identities (`'disjoint'`) or unions (`'union'`).
transform	a function applied to the areas of the *disjoint* (exclusive) regions before fitting. The default, `base::identity()`, leaves the areas untouched. A monotone transform such as `base::log1p()` can keep small regions legible when set sizes span several orders of magnitude. The transform is applied to the exclusive regions because those are the additive atoms the diagram fits to; as a consequence the area of a whole set or union no longer equals `transform()` of its size, only the individual visible regions carry the transformed scale. The function must return a non-negative, finite value for each region (and for `complement`, when given). Has no effect on `venn()` diagrams, whose geometry is fixed.
shape	geometric shape used in the diagram: one of `"circle"`, `"ellipse"`, `"rectangle"`, `"square"`, or `"rotated_rectangle"`. The rotated rectangle is fit with a derivative-

free optimizer and is also the only shape able to draw a true four-set Venn diagram (see `venn()`).

`loss` type of loss to minimize over. The default, `"sum_squared"`, minimizes the sum of squared errors. The available options mirror the loss functions exposed by the `eunoia` Rust crate that powers the optimizer:

- `"sum_squared"`: normalized sum of squared errors (default).
- `"sum_absolute"`: normalized sum of absolute errors.
- `"sum_absolute_region_error"`: normalized sum of absolute region errors.
- `"sum_squared_region_error"`: normalized sum of squared region errors.
- `"max_absolute"`: normalized maximum absolute error.
- `"max_squared"`: normalized maximum squared error.
- `"root_mean_squared"`: normalized root-mean-squared error.
- `"stress"`: venneuler-style stress.
- `"diag_error"`: eulerAPE-style `diagError`.
- `"log_sum_absolute"`: sum of absolute errors on `log1p`-transformed areas, which stops large regions from dominating the fit.
- `"smooth_sum_absolute"`, `"smooth_sum_absolute_region_error"`, `"smooth_max_absolute"`, `"smooth_max_squared"`, `"smooth_diag_error"`, `"smooth_log_sum_absolute"` — gradient-friendly (Huber) surrogates of the corresponding non-smooth losses, controlled by `control$loss_eps`.

`loss_aggregator`

deprecated; use `loss` directly instead. Pre-1.0 code that combined `loss` (`"square"`/`"abs"`/`"region"`) with `loss_aggregator` (`"sum"`/`"max"`) still works but emits a warning; the combination is mapped to the equivalent new `loss` value.

`complement` an optional single non-negative number giving the area of the *complement*, that is, the universe outside every named set. When supplied, the fitter jointly optimizes a containing rectangle together with the diagram shapes so that the area of the rectangle minus the union of (clipped) shapes matches `complement`. This is the classical "everything not in any set" region; see `plot.euler()` for how it is rendered. Defaults to `NULL` (no container; classical shape-only fit). Not supported for `venn()`.

`control` a list of control parameters.

- `extraopt`: should the global-search fallback optimizer (CMA-ES) kick in when the primary optimizer's `diagError` exceeds `extraopt_threshold`? The default is `TRUE` for three-set ellipse fits and `FALSE` otherwise.
- `extraopt_threshold`: threshold, in terms of `diagError`, for when the CMA-ES fallback kicks in. A value of 0 means it will kick in for an* error; a value of 1 means it will never kick in. Default `0.001`.
- `tolerance`: convergence tolerance passed to the underlying solver. Tighter values give more accurate fits at higher cost. Default `1e-8`.
- `max_sets`: maximum number of sets the underlying engine will accept. Defaults to `NULL`, which uses the engine's built-in default of 32. Region masks are stored in a bitset, so values may be raised up to 63 (the absolute hard cap). Going higher is rarely useful in practice since fully-overlapping diagrams have `2^n - 1` regions.

- `n_threads`: number of threads used to fan out the optimizer's restart loop. A positive integer pins a private thread pool of that size, while `NULL` uses all available cores. This is purely a wall-time knob: the fitted diagram is identical regardless of the thread count. The default uses half of the available logical cores (but a single thread under `R CMD check`, to respect CRAN's two-core policy). It can be overridden globally with the `eulerr.n_threads` option or the `EULERR_NUM_THREADS` environment variable, and otherwise honors R's conventional `mc.cores` option (or `MC_CORES` environment variable).
- `optimizer`: the final-layout optimizer. The default, `"auto"`, lets the engine pick a sensible optimizer for the chosen shape and loss. To force a particular one, use any of `"levenberg_marquardt"`, `"lbfgs"`, `"nelder_mead"`, `"mads"` (mesh-adaptive direct search, derivative-free and well suited to non-smooth losses), `"cma_es"`, `"cma_es_lm"`, `"trf"`, or `"cma_es_trf"`.
- `n_restarts`: number of full-pipeline restarts; the lowest-loss result is kept. Higher values improve the chance of finding the global optimum at proportionally higher cost. `NULL` (the default) lets the engine choose (10, automatically reduced for small smooth-loss fits).
- `loss_eps`: smoothing parameter for the `"smooth_*"` losses; pick roughly 1% of the typical residual magnitude. Smaller values track the non-smooth loss more closely but give noisier gradients. Default `0.01`.

`weights` a numeric vector of weights of the same length as the number of rows in `combinations`.

`by` a factor or character matrix to be used in `base::by()` to split the data.frame or matrix of set combinations

`sep` a character to use to separate the dummy-coded factors if there are factor or character vectors in 'combinations'.

`factor_names` whether to include factor names when constructing dummy codes

Details

If the input is a matrix or data frame and argument by is specified, the function returns a list of euler diagrams.

The function minimizes the residual sums of squares,

$$\sum_{i=1}^{n}(A_i - \omega_i)^2,$$

by default, where ω_i the size of the ith disjoint subset, and A_i the corresponding area in the diagram, that is, the unique contribution to the total area from this overlap. The loss function can, however, be controlled via the `loss` argument.

`euler()` also returns `stress` (from **venneuler**), as well as `diagError`, and `regionError` from **eulerAPE**.

The *stress* statistic is computed as

$$\frac{\sum_{i=1}^{n}(A_i - \beta\omega_i)^2}{\sum_{i=1}^{n} A_i^2},$$

where

$$\beta = \sum_{i=1}^{n} A_i \omega_i / \sum_{i=1}^{n} \omega_i^2.$$

`regionError` is computed as

$$\left| \frac{A_i}{\sum_{i=1}^{n} A_i} - \frac{\omega_i}{\sum_{i=1}^{n} \omega_i} \right|.$$

`diagError` is simply the maximum of regionError.

Value

A list object of class `'euler'` with the following parameters.

`shapes`	a data frame of fitted shape parameters. One row per set with a `type` column (one of `"circle"`, `"ellipse"`, `"rectangle"`, `"square"`, `"rotated_rectangle"`), the center coordinates `h` and `k`, and the shape-specific columns: `a`, `b`, `phi` for ellipses/circles; `width` and `height` for rectangles; `side` (plus mirrored `width`/`height`) for squares; `width`, `height`, and `phi` (rotation, in radians) for rotated rectangles. Columns that don't apply to the chosen shape are `NA`.
`ellipses`	for `shape = "circle"` and `shape = "ellipse"` fits, the legacy 5-column data frame of `h`, `k`, `a`, `b`, `phi`. This slot is deprecated in favour of `shapes` and is not populated for rectangle/square fits.
`original.values`	set relationships in the input
`fitted.values`	set relationships in the solution
`residuals`	residuals
`regionError`	the difference in percentage points between each disjoint subset in the input and the respective area in the output
`diagError`	the largest `regionError`
`stress`	normalized residual sums of squares

Methods (by class)

- `euler(default)`: a named numeric vector, with combinations separated by an ampersand, for instance `A&B = 10`. Missing combinations are treated as being 0.
- `euler(data.frame)`: a `data.frame` of logicals, binary integers, or factors.
- `euler(matrix)`: a matrix that can be converted to a data.frame of logicals (as in the description above) via `base::as.data.frame.matrix()`.
- `euler(table)`: A table with `max(dim(x)) < 3`.
- `euler(list)`: a list of vectors, each vector giving the contents of that set (with no duplicates). Vectors in the list must be named.

References

Wilkinson L. Exact and Approximate Area-Proportional Circular Venn and Euler Diagrams. IEEE Transactions on Visualization and Computer Graphics (Internet). 2012 Feb (cited 2016 Apr 9);18(2):321-31. Available from: doi:10.1109/TVCG.2011.56

Micallef L, Rodgers P. eulerAPE: Drawing Area-Proportional 3-Venn Diagrams Using Ellipses. PLOS ONE (Internet). 2014 Jul (cited 2016 Dec 10);9(7):e101717. Available from: doi:10.1371/journal.pone.0101717

See Also

plot.euler(), print.euler(), eulerr_options(), venn()

Examples

```
# Fit a diagram with circles
combo <- c(A = 2, B = 2, C = 2, "A&B" = 1, "A&C" = 1, "B&C" = 1)
fit1 <- euler(combo)

# Investigate the fit
fit1

# Refit using ellipses instead
fit2 <- euler(combo, shape = "ellipse")

# Investigate the fit again (which is now exact)
fit2

# Plot it
plot(fit2)

# A set with no perfect solution
euler(c(
  "a" = 3491, "b" = 3409, "c" = 3503,
  "a&b" = 120, "a&c" = 114, "b&c" = 132,
  "a&b&c" = 50
))


# Using grouping via the 'by' argument through the data.frame method
euler(fruits, by = list(sex, age))


# Using the matrix method
euler(organisms)

# Using weights
euler(organisms, weights = c(10, 20, 5, 4, 8, 9, 2))

# The table method
euler(pain, factor_names = FALSE)
```

```
# A euler diagram from a list of sample spaces (the list method)
euler(plants[c("erigenia", "solanum", "cynodon")])
```

---

eulergram-compose *Compose Euler Diagrams*

Description

Arrange two eulergram objects side-by-side or stacked, building up multi-panel layouts with operator syntax. Compositions can be nested arbitrarily, e.g. (p1 | p2) / p3.

Usage

```
## S3 method for class 'eulergram'
e1 | e2

## S3 method for class 'eulergram'
e1 / e2
```

Arguments

e1, e2	eulergram objects, typically returned by plot.euler().

Details

| arranges the two plots horizontally; / stacks them vertically. The result is itself an eulergram, so further composition chains naturally.

Consecutive operators of the same direction are flattened into a single row or column of equally sized panels. Thus p1 | p2 | p3 | p4 produces one row of four equal-width panels rather than a lopsided nest of binary splits. Use parentheses (or mix | and /) to force sub-groups: in (p1 | p2) / p3, p3 spans the full bottom row while p1 and p2 split the top row equally.

The gap between adjacent plots is controlled by the composition$spacing entry of eulerr_options(), which must be a grid::unit() and defaults to grid::unit(1, "lines").

Value

An eulergram containing the composed layout.

See Also

plot.euler(), eulerr_options()

Examples

```
p1 <- plot(euler(c(A = 1, B = 8, "A&B" = 1)))
p2 <- plot(euler(c(A = 1, C = 1, "A&C" = 1)))

p1 | p2
p1 / p2

p3 <- plot(euler(c(X = 3, Y = 2, "X&Y" = 1)))
(p1 | p2) / p3
```

`eulerr_options` *Get or set global graphical parameters for eulerr*

Description

This function provides a means to set default parameters for functions in eulerr. Query `eulerr_options()` (without any argument) to see all the available options and read more about the plot-related ones in `grid::gpar()` and `graphics::par()`.

Usage

```
eulerr_options(...)
```

Arguments

`...`	objects to update the global graphical parameters for **eulerr** with.

Details

Currently, the following items will be considered:

pointsize size in pts to be used as basis for fontsizes and some margin sizes in the resulting plot

fills a list of items `fill` and `alpha`

patterns a list of items `type`, `angle`, `col`, `lwd`, and `alpha`

edges a list of items `col`, `alpha`, `lex`, `lwd`, and `lty`

labels a list of items `rot`, `col`, `alpha`, `fontsize`, `cex`, `fontfamily`, `fontface`, `lineheight`, and `font`

quantities a list of items `type`, `template`, `format`, `total`, `rot`, `col`, `alpha`, `fontsize`, `cex`, `fontfamily`, `lineheight`, and `font`

annotations a list of items `rot`, `col`, `alpha`, `fontsize`, `cex`, `fontfamily`, `lineheight`, `font`, and `labels` (a named character vector keyed by subset name). Used to add a third stacked text element per region below the quantity.

glyphs defaults for dot and member-label glyph placement and styling, including `mode`, `arrangement`, `obstacle_policy`, `target_aspect_ratio`, `gap`, `boundary_gap`, `seed`, `radius`, `scale`, `min_scale`, `max_attempts`, `max_items`, and graphical parameters.

strips `col`, `alpha`, `fontsize`, `cex`, `fontfamily`, `lineheight`, and `font`

legend arguments to `grid::legendGrob()` as well as `col`, `alpha`, `fontsize`, `cex`, `symbol_size` (symbol size multiplier, independent of text size; defaults to `cex` if `NULL`), `fontfamily`, `lineheight`, and `font`

main arguments to `grid::textGrob()`

complement a list of styling defaults for the container box and its complement label drawn when `euler()` is called with `complement =`. Items: `fill`, `alpha`, `col`, `lty`, `lwd`, `lex`, `fontsize`, `cex`, `font`, `fontfamily`, `lineheight`. The default `lty = 2` draws the container with a dashed outline.

padding a `grid::unit()` giving the padding between various elements in plots from `plot.euler()`, which you can change if you, for instance, want to increase spacing between labels, quantities, and percentages.

margin a `grid::unit()` giving the fixed margin reserved around the diagram inside the plot canvas. Increase it if shape outlines or exterior labels are clipped at the plot edge, or decrease it to give the diagram more room. A fixed (rather than relative) unit is recommended so that the margin keeps pace with stroke width and label text, which are themselves fixed sizes.

composition a list controlling how `eulergram` objects are arranged when composed via `|` or `/`. Contains a single `spacing` item (a `grid::unit()`) that sets the gap between adjacent plots.

Value

This function gets or sets updates in the global environment that are used in `plot.euler()`.

See Also

`plot.euler()`, `grid::gpar()`, `graphics::par()`

Examples

```
eulerr_options(edges = list(col = "blue"), fontsize = 10)
eulerr_options(n_threads = 2)
```

---

fitted.euler *Fitted values of euler object*

Description

Fitted values of euler object

Usage

```
## S3 method for class 'euler'
fitted(object, dense = FALSE, ...)
```

Arguments

`object`	object of class `'euler'`
`dense`	if `TRUE`, return a vector covering every combination of the non-empty sets (`2^n - 1` entries), filling absent combinations with 0. The default (`FALSE`) returns the sparse vector stored on the object, which only contains entries that were either requested as input or fit to a non-zero area. Use `dense = TRUE` only for diagrams with few sets. The full enumeration is exponential in the number of non-empty sets.
`...`	ignored

Value

A named numeric vector of fitted areas keyed by combination label (set names joined by `&`).

`fruits` *Fruits*

Description

A synthethic data set of preferences for fruits and their overlaps, generated only to be a showcase for the examples for this package.

Usage

```
fruits
```

Format

A data.frame with 100 observations of 5 variables:

banana whether the person likes bananas, a logical

apple whether the person likes apples, a logical

orange whether the person likes oranges, a logical

sex the sex of the person, a factor with levels 'male' and 'female'

age the age of the person, a factor with levels 'child' and 'adult'

organisms *Organisms*

Description

Example data from the **VennMaster** package.

Usage

```
organisms
```

Format

A matrix with 7 observations, consisting of various organisms, and 5 variables: *animal, mammal, plant, sea*, and, *spiny*, indicating whether the organism belongs to the category or not.

Details

Note that this data is difficult to fit using an Euler diagram, even if we use ellipses, which is clear if one chooses to study the various overlaps in the resulting diagrams.

Source

https://github.com/sysbio-bioinf/VennMaster/blob/master/data_examples/deploy/example1.list

pain *Pain distribution data*

Description

Data from a study on pain distribution for patients with persistent neck pain in relation to a whiplash trauma.

Usage

```
pain
```

Format

A flat table (cross-table) with with sex in columns and pain distribution in rows and integer counts making up the cells of the table.

Disclaimer

Note that the maintainer of this package is an author of the source for this data.

Source

Westergren H, Larsson J, Freeman M, Carlsson A, Jöud A, Malmström E-M. Sex-based differences in pain distribution in a cohort of patients with persistent post-traumatic neck pain. Disability and Rehabilitation. 2017 Jan 27

`plants` *Plants*

Description

Data on plants and the states in the US and Canada they occur in.

Usage

```
plants
```

Format

A list with 33,721 plants, each containing a character vector listing the states in the US and Canada in which they occur. The names in the list specify the species or genus of the plant.

Source

USDA, NRCS. 2008. The PLANTS Database, 31 December 2008). National Plant Data Center, Baton Rouge, LA 70874-4490 USA.

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository http://archive.ics.uci.edu/ml/. Irvine, CA: University of California, School of Information and Computer Science.

`plot.euler` *Plot Euler and Venn diagrams*

Description

Plot diagrams fit with `euler()` and `venn()` using `grid::Grid()` graphics. This function sets up all the necessary plot parameters and computes the geometry of the diagram. `plot.eulergram()`, meanwhile, does the actual plotting of the diagram. Please see the **Details** section to learn about the individual settings for each argument.

Usage

```
## S3 method for class 'euler'
plot(
  x,
  fills = TRUE,
  patterns = FALSE,
  edges = TRUE,
  legend = FALSE,
  labels = identical(legend, FALSE),
  quantities = FALSE,
  annotations = NULL,
  glyphs = FALSE,
  strips = NULL,
  bg = FALSE,
  main = NULL,
  complement = TRUE,
  rotate = 0,
  n = 200L,
  adjust_labels = TRUE,
  ...
)

## S3 method for class 'eulerr_venn'
plot(
  x,
  fills = TRUE,
  patterns = FALSE,
  edges = TRUE,
  legend = FALSE,
  labels = identical(legend, FALSE),
  quantities = TRUE,
  glyphs = FALSE,
  strips = NULL,
  bg = FALSE,
  main = NULL,
  complement = TRUE,
  n = 200L,
  adjust_labels = TRUE,
  ...
)

## S3 method for class 'venn'
plot(...)
```

Arguments

x	an object of class 'euler', generated from euler()
fills	a logical, vector, or list of graphical parameters for the fills in the diagram. Vec-

tors are assumed to be colors for the fills. See grid::grid.path(). Named fill vectors can be matched in fills$mode = "disjoint" (default) or fills$mode = "union".

patterns — a logical, vector, or list of graphical parameters for fill patterns in the diagram. Vectors are assumed to be pattern types (currently "stripes" or NA), where NA means no pattern. Supported list items are type, angle, col, lwd, and alpha. Named pattern vectors can be matched in patterns$mode = "disjoint" (default) or patterns$mode = "union".

edges — a logical, vector, or list of graphical parameters for the edges in the diagram. Vectors are assumed to be colors for the edges. See grid::grid.polyline().

legend — a logical scalar or list. If a list, the item side can be used to set the location of the legend and symbol_size can be used to scale the legend symbols independently of the text size. See grid::grid.legend().

labels — a logical, vector, or list. Vectors are assumed to be text for the labels. See grid::grid.text(). In addition to the grid::gpar() fields, the following placement controls are supported (delegated to the eunoia Rust crate): labels$position = "inside" (default) uses region anchors; labels$position = "outside" places each set name just outside its own outline, without a leader. labels$angular_steps controls the outside placement search resolution (default 180). labels$placement ("raycast" (default), "force_directed", "matched", or "elbow") selects the strategy used when a label does not fit inside its region. "raycast", "force_directed", and "matched" produce straight leader lines: "raycast" places the label along the centroid→POI ray, "force_directed" relaxes labels with a polygon-aware force solver, and "matched" places labels on a ring hugging the diagram silhouette, spreading them in proportion to their widths so that neither the label boxes nor the leaders can cross. "elbow" produces d3-pie style orthogonal leaders, stacking exterior labels in left/right columns reached by a three-segment polyline. labels$margin (numeric) overrides the per-region margin between an exterior label and the diagram (default is half the larger of the label's width and height); labels$tether ("poi" (default) or "boundary") chooses where the leader line attaches on the source region; labels$gap controls the visible gap between the leader tip and the label box edge — a bare numeric is interpreted as lines (font-relative), a grid::unit() is honored as given, and the default NULL tracks eulerr_options()$padding so the gap matches the spacing between label and quantity; labels$leader is a list (col, alpha, lwd, lty, lex) styling the leader line drawn from the tether to the exterior label. Strategy-specific knobs live in their own sublists: labels$force_directed = list(iterations = ...) sets the iteration cap for the force-directed solver, and labels$elbow = list(min_gap = ...) sets the minimum vertical centre-to-centre spacing between stacked label boxes in an elbow column.

quantities — a logical, vector, or list. Vectors are assumed to be text for the quantities' labels, which by default are the original values in the input to euler(). In addition to plain vectors, quantities$labels can also be a named vector keyed by subset names (e.g., "A", "B", "A&B"), which is useful for supplying custom text for overlap regions. If quantities$labels is NULL, quantities$format can be used to control number formatting as a list with an item fun (a function such as signif() or round()) and optional extra arguments passed to that function

(for example, `list(fun = prettyNum, big.mark = ",")`). `quantities$total` can be used to set an external denominator for percent/fraction quantities (instead of the plotted total). to arguments that apply to `grid::grid.text()`, an argument `type` may also be used which should be a combination of `"counts"`, `"percent"`, and `"fraction"`. The first item will be printed first and the second will be printed thereafter inside brackets. The default is `type = "counts"`. For finer control over the rendered text, set `quantities$template` to a string with `{counts}`, `{percent}`, and/or `{fraction}` placeholders, for example `"{counts}\n{percent}"` to put the count and percentage on separate lines or `"n={counts} ({percent})"` for arbitrary layout. When `template` is set it overrides `type`; the set of placeholders in the template determines which values are computed.

`annotations` free-form per-region text rendered as a third stacked element below the quantity (or below the label when no quantity is drawn). Accepts a named character vector keyed by subset name (e.g. `c(A = "n = 12", "A&B" = "n = 3")`) as a shorthand for `list(labels = <vector>)`, or a list with `labels` plus `grid::gpar()` fields (`col`, `alpha`, `fontsize`, `cex`, `fontfamily`, `lineheight`, `font`, `rot`). Regions absent from `labels` are not annotated. The composite tag bbox grows to include the annotation, so exterior placement and leader lines adapt automatically. Defaults to slightly smaller text than `labels`/`quantities` (`cex = 0.8`).

`glyphs` a logical or list controlling glyphs packed inside exclusive regions. `TRUE` draws one equal-sized dot per unit of the original exclusive values; these values must be nonnegative integers. A list may set `mode` to `"dots"` (default) or `"members"`, `arrangement` to `"uniform"`, `"rectangular"`, `"low_discrepancy"`, or `"random"`, and supply `gap`, `seed`, `max_attempts`, and `max_items`. Uniform dots use a hexagonal grid; rectangular dots use aligned rows and columns. Low-discrepancy placement gives an even, deterministic scatter. Only random placement uses `seed`. Dot mode also supports a named integer `counts` vector, `radius`, `boundary_gap`, and `grid::gpar()` fields. For dots, `gap` controls spacing between dots as a fraction of the radius. `boundary_gap` sets extra clearance from region boundaries, holes, and label obstacles as a fraction of the radius. Its default, `NULL`, uses `gap`, keeping boundary and dot spacing coupled. Member mode requires a named list such as `labels = list(A = c("Ada", "Grace"), "A&B" = "Katherine")` and supports `scale`, `min_scale`, and text graphical parameters. Uniform and rectangular member placement both pack names in rows. With either arrangement, `target_aspect_ratio` sets a preferred width-to-height ratio for each block of rows, including its spacing. For example, `1` favors a roughly square block. The default, `NULL`, keeps the first complete row packing. This preference preserves the number of placed names and does not affect automatic scale selection. Automatic safety limits are 2,000 dots and 500 member labels unless `max_items` is supplied. `obstacle_policy = "best_effort"` (default) prefers positions clear of measured labels but allows overlap when items do not fit. Set `obstacle_policy = "strict"` to require clearance and warn about unplaced items. Strict placement keeps the usual sizing floors, so items can remain unplaced even with automatic sizing. Glyphs are drawn above region fills and edges but below labels.

`strips` a list, ignored unless the `'by'` argument was used in `euler()`. In addition to graphical parameters, this argument can include `labels = list(top = ..., left = ...)` for custom strip labels. Unnamed labels are interpreted in display

order. Named labels are matched by factor levels and then reordered to display order.

`bg` a logical, character, or list controlling the background grob. Character values are interpreted as the background fill color.

`main` a title for the plot in the form of a character, expression, list or something that can be sensibly converted to a label via `grDevices::as.graphicsAnnot()`. A list of length one can be provided, in which case its only element is used as the label. If a list of longer length is provided, an item named `'label'` must be provided (and will be used for the actual text).

`complement` a logical, character, or list controlling the container box and complement region for diagrams fit with `complement =` in `euler()`. `TRUE` (default) draws the container with a dashed outline (`lty = 2`), no fill, and the complement count inside the complement region. `FALSE` suppresses the container and its label entirely. A character value is treated as a fill color shorthand. A list accepts `fill`, `alpha`, `col`, `lty`, `lwd`, `lex` (outline + label gpar), `fontsize`, `cex`, `font`, `fontfamily`, `lineheight` (label only), and `label` (custom text — defaults to the complement count). Also accepts the same placement controls as `labels` (`placement`, `margin`, `tether`, `gap`, `leader`, `force_directed`, `elbow`) for the complement count label. Has no effect if the diagram was fit without `complement =`. Defaults can be set via `eulerr_options(complement = ...)`.

`rotate` a numeric value giving the angle in degrees by which to rotate the entire diagram layout. Positive values rotate counter-clockwise. Defaults to `0` (no rotation).

`n` number of vertices for the `edges` and `fills`

`adjust_labels` a logical. If `TRUE`, adjustment will be made to avoid overlaps or out-of-limits plotting of labels, quantities, and percentages.

`...` parameters to update `fills` and `edges` with and thereby a shortcut to set these parameters `grid::grid.text()`.

Details

The only difference between `plot.euler()` and `plot.venn()` is that `quantities` is set to `TRUE` by default in the latter and `FALSE` in the former.

Most of the arguments to this function accept either a logical, a vector, or a list where

- logical values set the attribute on or off,
- vectors are shortcuts to commonly used options (see the individual parameters), and
- lists enable fine-grained control, including graphical parameters as described in `grid::gpar()` and control arguments that are specific to each argument.

The various `grid::gpar()` values that are available for each argument are:

	fills	edges	labels	quantities	annotations	strips	legend	main
col		x	x	x	x	x	x	x
fill	x							
alpha	x	x	x	x	x	x	x	x
lty		x						

lwd	x						
lex	x						
fontsize		x	x	x	x	x	x
cex		x	x	x	x	x	x
fontfamily		x	x	x	x	x	x
lineheight		x	x	x	x	x	x
font		x	x	x	x	x	x

Defaults for these values, as well as other parameters of the plots, can be set globally using `eulerr_options()`.

If the diagram has been fit using the `data.frame` or `matrix` methods and using the `by` argument, the plot area will be split into panels for each combination of the one to two factors. The `fills`, `patterns`, `edges`, `labels`, `quantities`, and `annotations` arguments each accept an optional `by_group` entry: a named list of override lists keyed by panel name (the names of the fitted object). For multi-by fits the panel name is the levels joined by `.`, e.g. `"Male.German"`. Panels not listed in `by_group` use the top-level settings unchanged. Only graphical fields (and `rot` for `labels`, `quantities`, and `annotations`) may be overridden per panel; structural fields such as `quantities$type`, `quantities$format`, `annotations$labels`, or named-by-subset `fills$fill` must be set at the top level.

For users who are looking to plot their diagram using another package, all the necessary parameters can be collected if the result of this function is assigned to a variable (rather than printed to screen).

Value

Provides an object of class `'eulergram'` , which is a description of the diagram to be drawn. `plot.eulergram()` does the actual drawing of the diagram.

See Also

`euler()`, `plot.eulergram()`, `grid::gpar()`, `grid::grid.polyline()`, `grid::grid.path()`, `grid::grid.legend()`, `grid::grid.text()`

Examples

```
fit <- euler(c("A" = 10, "B" = 5, "A&B" = 3))

# Customize colors, remove borders, bump alpha, color labels white
plot(fit,
     fills = list(fill = c("red", "steelblue4"), alpha = 0.5),
     labels = list(col = "white", font = 4))

# Add quantities to the plot
plot(fit, quantities = TRUE)

# Add free-form per-region annotations below the counts
plot(
  fit,
  quantities = TRUE,
  annotations = c(A = "mean = 35", "A&B" = "mean = 41")
```

```
)

# Add a custom legend and retain quantities
plot(fit, quantities = TRUE, legend = list(labels = c("foo", "bar")))

# Plot without fills and distinguish sets with border types instead
plot(fit, fills = "transparent", lty = 1:2)

# Save plot parameters to plot using some other method
diagram_description <- plot(fit)

# Plots using 'by' argument
plot(euler(fruits[, 1:4], by = list(sex)), legend = TRUE)

# Per-panel styling with `by_group`
plot(
  venn(fruits[, 1:4], by = list(sex)),
  quantities = list(
    by_group = list(
      male = list(col = "steelblue"),
      female = list(col = "tomato")
    )
  )
)
```

`plot.eulergram` *Print (plot) Euler diagram*

Description

This function is responsible for the actual drawing of `'eulergram'` objects created through `plot.euler()`. `print.eulergram()` is an alias for `plot.eulergram()`, which has been provided so that `plot.euler()` gets called automatically.

Usage

```
## S3 method for class 'eulergram'
plot(x, newpage = TRUE, ...)

## S3 method for class 'eulergram'
print(x, ...)
```

Arguments

`x`	an object of class `'eulergram'`, usually the output of `plot.euler()`
`newpage`	if `TRUE`, opens a new page via `grid.newpage()` to draw on
`...`	ignored

Value

A plot is drawn on the current device using `grid::Grid()` graphics.

`print.euler` *Print a summary of an Euler diagram*

Description

This function is responsible for printing fits from `euler()` and provides a summary of the fit. Prints a data frame of the original set relationships and the fitted values as well as `diagError` and `stress` statistics.

Usage

```
## S3 method for class 'euler'
print(x, round = 3, vsep = strrep("-", 0.75 * getOption("width")), ...)
```

Arguments

`x`	`'euler'` object from `euler()`
`round`	number of decimal places to round to
`vsep`	character string to paste in between `euler` objects when `x` is a nested `euler` object
`...`	arguments passed to `base::print.data.frame()`

Value

Summary statistics of the fitted Euler diagram are printed to screen.

See Also

`euler()`, `base::print.data.frame()`

Examples

```
euler(organisms)
```

`print.eulerr_venn` *Print a summary of a Venn diagram*

Description

This function is responsible for printing objects from from `venn()` and provides a simple description of the number of sets and the specifications for the ellipses of the Venn diagram.

Usage

```
## S3 method for class 'eulerr_venn'
print(x, round = 3, vsep = strrep("-", 0.75 * getOption("width")), ...)

## S3 method for class 'venn'
print(...)
```

Arguments

`x`	an object of class `'eulerr_venn'`
`round`	number of digits to round the ellipse specification to
`vsep`	character string to paste in between `euler` objects when `x` is a nested `euler` object
`...`	arguments passed to `base::print.data.frame()`

Value

Summary statistics of the fitted Venn diagram are printed to screen.

See Also

`venn()`, `base::print.data.frame()`

Examples

```
venn(organisms)
```

residuals.euler *Residuals of euler object*

Description

Residuals of euler object

Usage

```
## S3 method for class 'euler'
residuals(object, dense = FALSE, ...)
```

Arguments

object	object of class 'euler'
dense	same meaning as in fitted.euler()
...	ignored

Value

A named numeric vector of residuals (input minus fit) keyed by combination label.

venn *Venn diagrams*

Description

This function fits Venn diagrams using an interface that is almost identical to euler(). Strictly speaking, Venn diagrams are Euler diagrams where every intersection is visible, regardless of whether or not it is zero. In almost every incarnation of Venn diagrams, however, the areas in the diagram are also *non-proportional* to the input; this is also the case here.

Usage

```
venn(combinations, ...)

## Default S3 method:
venn(
  combinations,
  input = c("disjoint", "union"),
  shape = c("ellipse", "rotated_rectangle"),
  names = letters[length(combinations)],
  ...
)
```

```
## S3 method for class 'table'
venn(combinations, ...)

## S3 method for class 'data.frame'
venn(
  combinations,
  weights = NULL,
  by = NULL,
  sep = "_",
  factor_names = TRUE,
  ...
)

## S3 method for class 'matrix'
venn(combinations, ...)

## S3 method for class 'list'
venn(combinations, ...)
```

Arguments

combinations	set relationships as a named numeric vector, matrix, or data.frame (see **methods (by class)**)
...	arguments passed down to other methods
input	type of input: disjoint identities (`'disjoint'`) or unions (`'union'`).
shape	geometric shape used in the diagram, either `"ellipse"` (the default, supporting up to five sets) or `"rotated_rectangle"` (supporting up to four sets). Rotated rectangles draw a true four-set Venn diagram in which all 15 regions are visible.
names	a character vector for the names of each set of the same length as 'combinations'. Must not be `NULL` if `combinations` is a one-length numeric.
weights	a numeric vector of weights of the same length as the number of rows in `combinations`.
by	a factor or character matrix to be used in `base::by()` to split the data.frame or matrix of set combinations
sep	a character to use to separate the dummy-coded factors if there are factor or character vectors in 'combinations'.
factor_names	whether to include factor names when constructing dummy codes

Value

Returns an object of class `'eulerr_venn', 'venn', 'euler'` with items

shapes	a data frame of the precomputed shape parameters (one row per set). For `shape = "ellipse"` the columns are `type, h, k, a, b, phi`; for `shape = "rotated_rectangle"` they are `type, h, k, width, height, phi`.
ellipses	the legacy 5-column data frame (`h, k, a, b, phi`), kept for back-compat alongside the canonical `shapes` slot. Only present for `shape = "ellipse"`.

`original.values`
: set relationships in the input

`fitted.values`
: set relationships in the solution

Methods (by class)

- `venn(default)`: a named numeric vector, with combinations separated by an ampersand, for instance `A&B = 10`. Missing combinations are treated as being 0.
- `venn(table)`: A table with `max(dim(x)) < 3`.
- `venn(data.frame)`: a `data.frame` of logicals, binary integers, or factors.
- `venn(matrix)`: a matrix that can be converted to a data.frame of logicals (as in the description above) via `base::as.data.frame.matrix()`.
- `venn(list)`: a list of vectors, each vector giving the contents of that set (with no duplicates). Vectors in the list do not need to be named.

See Also

`plot.eulerr_venn()`, `print.eulerr_venn()`, `euler()`

Examples

```
# The trivial version
f1 <- venn(5, names = letters[1:5])
plot(f1)

# Using data (a numeric vector)
f2 <- venn(c(A = 1, "B&C" = 3, "A&D" = 0.3))

# The table method
venn(pain, factor_names = FALSE)

# Using grouping via the 'by' argument through the data.frame method
venn(fruits, by = list(sex, age))


# Using the matrix method
venn(organisms)

# Using weights
venn(organisms, weights = c(10, 20, 5, 4, 8, 9, 2))

# A venn diagram from a list of sample spaces (the list method)
venn(plants[c("erigenia", "solanum", "cynodon")])
```

Index

∗ **datasets**
    fruits, 12
    organisms, 13
    pain, 13
    plants, 14
/.eulergram *(*eulergram-compose*)*, 9

base::as.data.frame.matrix(), *7*, *25*
base::by(), *6*, *24*
base::identity(), *4*
base::log1p(), *4*
base::print.data.frame(), *21, 22*

data.frame, *12*

error_plot, 2
euler, 3
euler(), *2*, *3*, *6*, *11*, *14–19*, *21*, *23*, *25*
eulergram-compose, 9
eulerr_options, 10
eulerr_options(), *8–10*, *19*

fitted.euler, 11
fitted.euler(), *23*
fruits, 12

graphics::par(), *10, 11*
grDevices::as.graphicsAnnot(), *18*
grid.newpage(), *20*
grid::gpar(), *10, 11*, *17–19*
grid::Grid(), *14*, *21*
grid::grid.legend(), *16*, *19*
grid::grid.path(), *16*, *19*
grid::grid.polyline(), *16*, *19*
grid::grid.text(), *16–19*
grid::legendGrob(), *11*
grid::textGrob(), *11*
grid::unit(), *9*, *11*, *16*

list, *14*

matrix, *13*

organisms, 13

pain, 13
plants, 14
plot.euler, 14
plot.euler(), *3*, *5*, *8, 9*, *11*, *18*, *20*
plot.eulergram, 20
plot.eulergram(), *3*, *14*, *19, 20*
plot.eulerr_venn *(*plot.euler*)*, 14
plot.eulerr_venn(), *25*
plot.venn *(*plot.euler*)*, 14
plot.venn(), *18*
print.euler, 21
print.euler(), *8*
print.eulergram *(*plot.eulergram*)*, 20
print.eulergram(), *20*
print.eulerr_venn, 22
print.eulerr_venn(), *25*
print.venn *(*print.eulerr_venn*)*, 22

residuals.euler, 23
round(), *16*

signif(), *16*

table, *13*

venn, 23
venn(), *4, 5*, *8*, *14*, *22*