Package ‘eulerr’

December 10, 2022

Title Area-Proportional Euler and Venn Diagrams with Ellipses

Version 7.0.0

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 ellipses and circles via a wide range of inputs and can be visualized in numerous ways.

Depends R (>= 3.3.0)

Imports GenSA, graphics, grDevices, grid, polyclip, polylablr, Rcpp, stats, utils

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

LinkingTo Rcpp (>= 0.12.12), RcppArmadillo (>= 0.7.600.1.0)

SystemRequirements C++14

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr


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

RoxygenNote 7.2.2

Language en-US

NeedsCompilation yes

Author Johan Larsson [aut, cre] (<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 <johanlarsson@outlook.com>
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

```r
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

```r
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 or an ellipse.

**Usage**

```r
euler(combinations, ...)
```

#### Default S3 method:

```r
euler(
  combinations,
  input = c("disjoint", "union"),
  shape = c("circle", "ellipse"),
  loss = c("square", "abs", "region"),
  loss_aggregator = c("sum", "max"),
  control = list(),
  ...
)
```

#### S3 method for class 'data.frame'

```r
euler(
  combinations,
  weights = NULL,
  by = NULL,
  ...
)
```

---
\begin{verbatim}
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 \textbf{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
- **loss**: type of loss to minimize over. If "square" is used together with the value "sum" for \textit{loss_aggregator}, then the resulting loss function is the sum of squared errors, which is the default.
- **loss_aggregator**: how the final loss is computed. "sum" indicates that the sum of the losses computed by \textit{loss} are summed up. "max" indicates
- **control**: a list of control parameters.
  - \textbf{extraopt}: should the more thorough optimizer (currently \texttt{GenSA::GenSA()}) kick in (provided \texttt{extraopt_threshold} is exceeded)? The default is TRUE for ellipses and three sets and FALSE otherwise.
  - \textbf{extraopt_threshold}: threshold, in terms of diagError, for when the extra optimizer kicks in. This will almost always slow down the process considerably. A value of 0 means that the extra optimizer will kick in if there is \textit{any} error. A value of 1 means that it will never kick in. The default is 0.001.
  - \textbf{extraopt_control}: a list of control parameters to pass to the extra optimizer, such as \texttt{max.call}. See \texttt{GenSA::GenSA()}.
- **weights**: a numeric vector of weights of the same length as the number of rows in \textit{combinations}.
- **by**: a factor or character matrix to be used in \texttt{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
\end{verbatim}
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 \( \omega_i \) 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

\[ \beta = \frac{\sum_{i=1}^{n} A_i \omega_i}{\sum_{i=1}^{n} \omega_i^2}. \]

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.

- **ellipses**
  a matrix of h and k (x and y-coordinates for the centers of the shapes), semiaxes a and b, and rotation angle phi

- **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


See Also

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

Examples

```r
# Fit a diagram with circles
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")])

---

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

- **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, rot, col, alpha, fontsize, cex, fontfamily, lineheight, and font

- **strips**  
  col, alpha, fontsize, cex, fontfamily, lineheight, and font

- **legend**  
  arguments to `grid::legendGrob()` as well as col, alpha, fontsize, cex, fontfamily, lineheight, and font
main arguments to `grid::textGrob()

padding `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.

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)

---

fruits

<table>
<thead>
<tr>
<th></th>
<th>Fruits</th>
</tr>
</thead>
</table>

Description

A synthetic 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

<table>
<thead>
<tr>
<th>organisms</th>
<th>Organisms</th>
</tr>
</thead>
</table>

**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

<table>
<thead>
<tr>
<th>pain</th>
<th>Pain distribution data</th>
</tr>
</thead>
</table>

**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


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


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.
plot.euler

Usage

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

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

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. Vectors are assumed to be colors for the fills. See grid::grid.path().
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. See grid::grid.legend().
labels a logical, vector, or list. Vectors are assumed to be text for the labels. See grid::grid.text().
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 arguments that apply to grid::grid.text(), an argument type may also be used which should be a combination of "counts" and "percent". The first item
will be printed first and the second will be printed thereafter inside brackets. The default is `type = "counts"`.

`strips` a list, ignored unless the 'by' argument was used in `euler()`

`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).

`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:

<table>
<thead>
<tr>
<th></th>
<th>fills</th>
<th>edges</th>
<th>labels</th>
<th>quantities</th>
<th>strips</th>
<th>legend</th>
<th>main</th>
</tr>
</thead>
<tbody>
<tr>
<td>col</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fill</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alpha</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lwd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fontsize</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cex</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fontfamily</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineheight</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

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).
plot.eulergram

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 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)

---

**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.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 'venn'
print(x, round = 3, vsep = strrep("-", 0.75 * getOption("width")), ...)

Arguments

x  an object of class '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)

---

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"),
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').
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 'venn', 'euler' with items

- **ellipses** a matrix of h and k (x and y-coordinates for the centers of the shapes), semiaxes a and b, and rotation angle phi
- **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.venn()`, `print.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, 8
  organisms, 9
  pain, 9
  plants, 10

base::as.data.frame.matrix(), 6, 17
base::by(), 4, 16
base::print.data.frame(), 14, 15

data.frame, 8

error_plot, 2
euler, 3
euler(), 2, 3, 5, 10–15, 17
eulerr_options, 7
eulerr_options(), 6, 7, 12

fruits, 8

GenSA::GenSA(), 4
graphics::par(), 7, 8
grDevices::as.graphicsAnnot(), 12
grid.newpage(), 14
grid::gpar(), 7, 8, 12, 13
grid::Grid(), 10, 14
grid::grid.legend(), 11, 13
grid::grid.path(), 11, 13
grid::grid.polyline(), 11, 13
grid::grid.text(), 11–13
grid::legendGrob(), 7
grid::textGrob(), 8
grid::unit(), 8

list, 10

matrix, 9

organisms, 9

pain, 9

plants, 10
plot.euler, 10
plot.euler(), 2, 3, 6, 8, 12–14
plot.eulergram, 13
plot.eulergram(), 3, 10, 13
plot.venn(plot.euler), 10
plot.venn(), 12, 17
print.euler, 14
print.euler(), 6
print.eulergram(plot.eulergram), 13
print.eulergram(), 13
print.venn, 15
print.venn(), 17

table, 9

venn, 15

venn(), 6, 10, 15