Package: easylabel (via r-universe)

March 8, 2025

Type Package

Title Interactive Scatter Plot and Volcano Plot Labels

Version 0.3.3

Maintainer Myles Lewis <`myles.lewis@qmul.ac.uk`>

BugReports `https://github.com/myles-lewis/easylabel/issues`

URL `https://github.com/myles-lewis/easylabel`

Description Interactive labelling of scatter plots, volcano plots and Manhattan plots using a 'shiny' and 'plotly' interface. Users can hover over points to see where specific points are located and click points on/off to easily label them. Labels can be dragged around the plot to place them optimally. Plots can be exported directly to PDF for publication. For plots with large numbers of points, points can optionally be rasterized as a bitmap, while all other elements (axes, text, labels & lines) are preserved as vector objects. This can dramatically reduce file size for plots with millions of points such as Manhattan plots, and is ideal for publication.

Language en-gb

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports DT, ggplot2, gtools, memoise, plotly (>= 4.10.0), RColorBrewer, rlang, shiny, shinycssloaders, shinybusy

Suggests AnnotationDbi, knitr, magick, org.Hs.eg.db, qvalue, rmarkdown

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 2.10)

Config/pak/sysreqs make libicu-dev libssl-dev zlib1g-dev

Repository https://myles-lewis.r-universe.dev

RemoteUrl https://github.com/myles-lewis/easylabel

RemoteRef HEAD

RemoteSha 073ad64802358ce6971806a98a0038010110b6c0

Contents

`add_labels` *Add labels to a plotly scatter plot*

Description

Adds labels to a plotly 2d or 3d scatter plot. The labels can be dragged.

Usage

```
add_labels(p, labs, plotGlPixelRatio = 8)
```

Arguments

`p`	A plotly scatter plot object
`labs`	Character vector of labels to match
`plotGlPixelRatio`	Integer passed to plotly.js config which controls pixel resolution of webGl rendering. Larger values increase resolution of points as well as file size.

Details

Labels are identified by searching the rownames of the embedded marker data within the plotly object. The plotly mode bar camera icon is set to export to svg by default.

Value

A plotly plot with added labels

Examples

```
library(plotly)
data(mtcars)

p <- plot_ly(mtcars, x = ~mpg, y = ~wt, color = ~cyl,
             type = 'scatter', mode = 'markers')
p %>% add_labels(c("Mazda RX4", "Fiat 128"))
```

`arial` *Change plotly font to Arial*

Description

Change plotly font to Arial

Usage

```
arial(p, size = 14)
```

Arguments

`p`	A plotly object
`size`	Font size

Value

A plotly object

Examples

```
library(plotly)
data(mtcars)

p <- plot_ly(mtcars, x = ~mpg, y = ~wt, color = ~cyl,
             type = 'scatter', mode = 'markers')
p %>% arial
p %>% arial(18)
```

`easylabel` *Interactive scatter plot labels*

Description

Interactive labelling of scatter plots using shiny/plotly interface.

Usage

```
easylabel(
  data,
  x,
  y,
  labs = NULL,
  startLabels = NULL,
  start_xy = NULL,
  cex.text = 0.72,
  col = NULL,
  colScheme = NULL,
  alpha = 1,
  shape = NULL,
  shapeScheme = 21,
  size = 8,
  sizeRange = c(4, 80),
  xlab = x,
  ylab = y,
  xlim = NULL,
  ylim = NULL,
  xticks = NULL,
  yticks = NULL,
  showOutliers = TRUE,
  outlier_shape = 5,
  outline_col = "white",
  outline_lwd = 0.5,
  plotly_filter = NULL,
  width = 800,
  height = 600,
  showgrid = FALSE,
  zeroline = TRUE,
  hline = NULL,
  vline = NULL,
  mgp = c(1.8, 0.5, 0),
  Ltitle = "",
  Rtitle = "",
  LRtitle_side = 1,
  labelDir = "radial",
  labCentre = NULL,
```

```
  lineLength = 75,
  text_col = "black",
  line_col = "black",
  rectangles = FALSE,
  rect_col = "white",
  border_col = "black",
  padding = 3,
  border_radius = 5,
  showLegend = TRUE,
  legendxy = c(1.02, 1),
  filename = NULL,
  panel.last = NULL,
  fullGeneNames = FALSE,
  AnnotationDb = NULL,
  custom_annotation = NULL,
  output_shiny = TRUE,
  ...
)
```

Arguments

`data`	Dataset (data.frame or data.table) to use for plot.
`x`	Specifies column of x coordinates in `data`.
`y`	Specifies column of y coordinates in `data`.
`labs`	Specifies the column in `data` with label names for points. Label names do not have to be unique. If `NULL` defaults to `rownames(data)`.
`startLabels`	Vector of initial labels. With a character vector, labels are identified in the column specified by `labs`. With a numeric vector, points to be labelled are referred to by row number.
`start_xy`	List containing label annotation starting coordinates. Invoked by `loadlabel()`.
`cex.text`	Font size for labels. Default 0.72 to match plotly font size. See `text()`.
`col`	Specifies which column in `data` affects point colour. Must be categorical. If it is not a factor, it will be coerced to a factor.
`colScheme`	A single colour or a vector of colours for points.
`alpha`	Alpha value for transparency of points.
`shape`	Specifies which column in `data` controls point shapes. If not a factor, will be coerced to a factor.
`shapeScheme`	A single symbol for points or a vector of symbols. See `pch` in `points()`.
`size`	Either a single value for size of points (default 8), or specifies which column in `data` affects point size for bubble charts.
`sizeRange`	Range of size of points for bubble charts.
`xlab`	x axis title. Accepts expressions when exporting base graphics. Set `cex.lab` to alter the font size of the axis titles (default 1). Set `cex.axis` to alter the font size of the axis numbering (default 1).

`ylab`	y axis title. Accepts expressions when exporting base graphics.
`xlim`	The x limits (x1, x2) of the plot.
`ylim`	The y limits of the plot.
`xticks`	List of custom x axis ticks and labels specified as a list of two named vectors `at = ...` and `labels = ...`. Another method is to use `xaxp` as a vector of the form `c(x1, x2, n)` giving the coordinates of the extreme tick marks and the number of intervals between tick-marks.
`yticks`	List of custom y axis ticks and labels specified as a list of two named vectors `at = ...` and `labels = ...`. Another method is to use `yaxp` as a vector of the form `c(y1, y2, n)` giving the coordinates of the extreme tick marks and the number of intervals between tick-marks.
`showOutliers`	Logical whether to show outliers on the margins of the plot.
`outlier_shape`	Symbol for outliers.
`outline_col`	Colour of symbol outlines. Set to `NA` for no outlines.
`outline_lwd`	Line width of symbol outlines.
`plotly_filter`	Refers to a column of logical values in `data` used to filter rows to reduce the number of points shown by plotly. We recommend using this for datasets with >100,000 rows. When saving to pdf, the full original dataset is still plotted. This is useful for plots with millions of points such as Manhattan plots where a subset of points to be labelled is already known.
`width`	Width of the plot in pixels. Saving to pdf scales 100 pixels to 1 inch.
`height`	Height of the plot in pixels.
`showgrid`	Either logical whether to show gridlines, or a character value where "x" means showing x axis gridlines and "y" means showing y axis gridlines.
`zeroline`	Logical whether to show lines at x = 0 and y = 0.
`hline`	Adds horizontal lines at values of y.
`vline`	Adds vertical lines at values of x.
`mgp`	The margin line for the axis title, axis labels and axis line. See `par()`.
`Ltitle`	A character or expression (see plotmath) value specifying text for left side title. Size of font can be changed using `cex.lab`.
`Rtitle`	A character or expression value specifying text for right side title. Size of font can be changed using `cex.lab`.
`LRtitle_side`	On which side of the plot for `Ltitle` and `Rtitle` (1 = bottom, 3 = top). See `mtext()`.
`labelDir`	Initial label line directions. Options include 'radial' (default) for radial lines around the centre of the plot, 'origin' for radial lines around the origin, 'horiz' for horizontal and 'vert' for vertical, 'xellipse' and 'yellipse' for near-horizontal and near-vertical lines arranged in an elliptical way around the centre, 'rect' for rectilinear lines (a mix of horizontal and vertical), 'x' for diagonal lines, 'oct' for lines in 8 directions around the centre.
`labCentre`	Coordinates in x/y units of the central point towards which radial labels converge. Defaults to the centre of the plot.

`lineLength`	Initial length of label lines in pixels.
`text_col`	Colour of label text. If set to `"match"` label text will match the colour of each point.
`line_col`	Colour of label lines. If set to `"match"` label line will match the colour of each point.
`rectangles`	Logical whether to show rectangles around labels (not supported by plotly).
`rect_col`	Colour for filling rectangles (not supported by plotly). If set to `"match"` rectangle fill colour will match the colour of each point.
`border_col`	Colour of rectangle borders (not supported by plotly). Use `border_col = NA` to omit borders. If set to `"match"` rectangle border colour will match the colour of each point.
`padding`	Amount of padding in pixels around label text.
`border_radius`	Amount of roundedness in pixels to apply to label rectangles (not supported by plotly).
`showLegend`	Logical whether to show or hide the legend.
`legendxy`	Vector of coordinates for the position of the legend. Coordinates are in plotly paper reference with `c(0, 0)` being the bottom left corner and `c(1, 1)` being the top right corner of the plot window. Plotly has unusual behaviour in that the x coordinate always aligns the left side of the legend. However, the y coordinate aligns the top, middle or bottom of the legend dependent on whether it is in the top, middle or bottom 1/3 of the plot window. So `c(1, 0)` positions the legend in the bottom right corner outside the right margin of the plot, while `c(1, 0.5)` centre aligns the legend around the centre of y axis.
`filename`	Filename for saving plots to pdf in a browser. Rstudio opens its own pdf file.
`panel.last`	An expression to be evaluated after plotting has taken place but before the axes, title and box are added. This can be useful for adding extra titles, legends or trend lines. Currently only works when saving plots using base graphics and does not work with plotly. See plot.default
`fullGeneNames`	Logical whether to expand gene symbols using Bioconductor AnnotationDbi package. With multiple matches, returns first value only. See `AnnotationDbi::mapIds()`.
`AnnotationDb`	Annotation database to use when expanding gene symbols. Defaults to human gene database `AnnotationDb = org.Hs.eg.db`.
`custom_annotation`	List of annotations to be added via `plotly::layout()`.
`output_shiny`	Logical whether to output a shiny app. If `FALSE` a plotly figure will be returned.
`...`	Further graphical parameters passed to `plot()` when saving via base graphics. The most useful for most users are likely to be `cex.lab` which alters axis title font size (default 1, see `par()`), `cex.axis` which alters axis numbering font size (default 1), and `panel.last` which allows additional plotting functions to be called after the main plot has been plotted but before the labels and label lines are drawn, which will allow the addition of trend lines, extra titles or legends for example (see `plot.default()`).

Details

Instructions:

- Hover over and click on/off genes which you want to label.
- When you have selected all your chosen genes, then drag gene names to move label positions.
- Click the save button to export a PDF or other file types (SVG, jpeg, tiff, png) produced in base graphics. Files are saved to the browser downloads folder by default.
- The Table tab shows a table view of the dataset to help with annotation.

The 'raster points' checkbox converts only the points to a raster which is embedded in pdf or svg exports, while the rest of the plot (axis/label lines/text etc) remain vectorised. This only applies when saving pdf or svg. For other file types the whole plot is rasterized as normal by base graphics.

For efficiency, the raster of the points is cached using the `memoise` package. So if users change the labels or change from pdf to svg, the export is much faster. If the resolution is changed the points will need to be re-rasterized.

In the shiny interface the "Save state" button saves an rds file containing the label positions and state of the plot as an 'easylab' S3 class object to the current working directory. This can be reloaded in future sessions using `loadlabel()`.

Value

By default no return value. If `output_shiny = FALSE` or the shiny button 'Export plotly & exit' is pressed, a plotly figure is returned.

See Also

`easyVolcano()`, `easyMAplot()`, `easyManhattan()`, `loadlabel()`

Examples

```
# Simple example using mtcars dataset
data(mtcars)
# Launch easylabel Shiny app: only run this example in interactive R sessions
if (interactive()) {
easylabel(mtcars, x = 'mpg', y = 'wt', col = 'cyl')
}
```

`easyManhattan` *Interactive Manhattan plot labels*

Description

Interactive labelling of Manhattan plots using 'shiny' and 'plotly' interface.

Usage

```
easyManhattan(
  data,
  chrom = "chrom",
  pos = "pos",
  p = "p",
  labs = "rsid",
  startLabels = NULL,
  pcutoff = 5e-08,
  chromGap = NULL,
  chromCols = c("royalblue", "skyblue"),
  sigCol = "red",
  alpha = 0.7,
  labelDir = "horiz",
  xlab = "Chromosome position",
  ylab = expression("-log"[10] ~ "P"),
  xlim = NULL,
  ylim = NULL,
  outline_col = NA,
  shapeScheme = 16,
  size = 6,
  width = ifelse(transpose, 600, 1000),
  height = ifelse(transpose, 800, 600),
  lineLength = 60,
  npoints = max(c(nrow(data)/5, 1e+06)),
  nplotly = 1e+05,
  npeaks = NULL,
  span = 2e+07,
  transpose = FALSE,
  filename = NULL,
  ...
)
```

Arguments

`data`	The dataset (data.frame or data.table) for the plot.
`chrom`	The column of chomosome values in `data`.
`pos`	The column of SNP positions in `data`.
`p`	The column of p values in `data`.
`labs`	The column of labels in `data`.
`startLabels`	Vector of initial labels. With a character vector, labels are identified in the column specified by `labs`. With a numeric vector, points to be labelled are referred to by row number.
`pcutoff`	Cut-off for p value significance. Defaults to 5E-08.
`chromGap`	Size of gap between chromosomes along the x axis in base pairs. If `NULL` this is automatically calculated dependent on the size of the genome. Default is around 3E07 for a human genome, and smaller for smaller genomes.

`chromCols`	A vector of colours for points by chromosome. Colours are recycled dependent on the length of the vector.
`sigCol`	Colour for statistically significant points. Ignored if set to `NA`.
`alpha`	Transparency for points.
`labelDir`	Option for label lines. See `easylabel()`.
`xlab`	x axis title. Accepts expressions.
`ylab`	y axis title. Accepts expressions.
`xlim`	The x limits (x1, x2) of the plot.
`ylim`	The y limits of the plot.
`outline_col`	Colour of symbol outlines. Passed to `easylabel()`.
`shapeScheme`	A single symbol for points or a vector of symbols. Passed to `easylabel()`.
`size`	Specifies point size. Passed to `easylabel()`.
`width`	Width of the plot in pixels. Saving to pdf scales 100 pixels to 1 inch.
`height`	Height of the plot in pixels.
`lineLength`	Initial length of label lines in pixels.
`npoints`	Maximum number of points to plot when saving the final plot to pdf. By default plots with >1 million points are thinned to speed up plotting. Setting a value of `NA` will plot all points.
`nplotly`	Maximum number of points to display via plotly. We recommend the default setting of 100,000 points (or fewer).
`npeaks`	Number of peaks to label initially.
`span`	a peak is defined as the most significant SNP within a window of width span centred at that SNP.
`transpose`	Logical whether to transpose the plot.
`filename`	Filename for saving to pdf.
`...`	Other arguments passed to `easylabel()`.

Value

By default no return value. If `output_shiny = FALSE` or the shiny button 'Export plotly & exit' is pressed, a plotly figure is returned. See `easylabel()`.

See Also

`easylabel()` `easyVolcano()`

`easyMAplot` *Interactive MA plot labels*

Description

Interactive labelling of MA plots using shiny/plotly interface.

Usage

```
easyMAplot(
  data,
  x = NULL,
  y = NULL,
  padj = NULL,
  fdrcutoff = 0.05,
  colScheme = c("darkgrey", "blue", "red"),
  hline = 0,
  labelDir = "yellipse",
  xlab = expression("log"[2] ~ " mean expression"),
  ylab = expression("log"[2] ~ " fold change"),
  filename = NULL,
  showCounts = TRUE,
  useQ = FALSE,
  ...
)
```

Arguments

`data`	The dataset for the plot. Automatically attempts to recognises DESeq2 and limma objects.
`x`	Name of the column containing mean expression. For DESeq2 and limma objects this is automatically set.
`y`	Name of the column containing log fold change. For DESeq2 and limma objects this is automatically set.
`padj`	Name of the column containing adjusted p values (optional). For DESeq2 and limma objects this is automatically set. If `y` is specified and `padj` is left blank or equal to `y`, nominal unadjusted p values are used for cut-off for significance.
`fdrcutoff`	Cut-off for FDR significance. Defaults to FDR < 0.05. Can be vector with multiple cut-offs. To use nominal P values instead of adjusted p values, set `y` but leave `padj` blank.
`colScheme`	Colour colScheme. Length must match either length(fdrcutoff) + 1 to allow for non-significant genes, or match length(fdrcutoff) * 2 + 1 to accommodates asymmetric colour colSchemes for positive & negative fold change. (see examples).
`hline`	Vector of horizontal lines (default is y = 0).

`labelDir`	Option for label lines. See `easylabel()`.
`xlab`	x axis title. Accepts expressions.
`ylab`	y axis title. Accepts expressions.
`filename`	Filename for saving to pdf.
`showCounts`	Logical whether to show legend with number of differentially expressed genes.
`useQ`	Logical whether to convert nominal P values to q values. Requires the qvalue Bioconductor package.
`...`	Other arguments passed to `easylabel()`.

Value

By default no return value. If `output_shiny = FALSE` or the shiny button 'Export plotly & exit' is pressed, a plotly figure is returned. See `easylabel()`.

See Also

`easylabel()` `easyVolcano()`

`easyVolcano` *Interactive volcano plot labels*

Description

Interactive labelling of volcano plots using shiny/plotly interface.

Usage

```
easyVolcano(
  data,
  x = NULL,
  y = NULL,
  padj = y,
  fdrcutoff = 0.05,
  fccut = NULL,
  colScheme = c("darkgrey", "blue", "red"),
  xlab = expression("log"[2] ~ " fold change"),
  ylab = expression("-log"[10] ~ " P"),
  filename = NULL,
  showCounts = TRUE,
  useQ = FALSE,
  ...
)
```

Arguments

`data`	The dataset for the plot. Automatically attempts to recognises DESeq2 and limma objects.
`x`	Name of the column containing log fold change. For DESeq2 and limma objects this is automatically set.
`y`	Name of the column containing p values. For DESeq2 and limma objects this is automatically set.
`padj`	Name of the column containing adjusted p values (optional). If `y` is specified and `padj` is left blank or equal to `y`, nominal unadjusted p values are used for cut-off for significance instead of adjusted p values.
`fdrcutoff`	Cut-off for FDR significance. Defaults to FDR < 0.05. If `y` is specified manually and `padj` is left blank then this refers to the cut-off for significant points using nominal unadjusted p values.
`fccut`	Optional vector of log fold change cut-offs.
`colScheme`	Colour scheme. If no fold change cut-off is set, 2 colours need to be specified. With a single fold change cut-off, 3 or 5 colours are required, depending on whether the colours are symmetrical about $x = 0$. Accommodates asymmetric colour schemes with multiple fold change cut-offs (see examples).
`xlab`	x axis title. Accepts expressions.
`ylab`	y axis title. Accepts expressions.
`filename`	Filename for saving to pdf.
`showCounts`	Logical whether to show legend with number of differentially expressed genes.
`useQ`	Logical whether to convert nominal P values to q values. Requires the qvalue Bioconductor package.
`...`	Other arguments passed to `easylabel()`.

Value

By default no return value. If `output_shiny = FALSE` or the shiny button 'Export plotly & exit' is pressed, a plotly figure is returned. See `easylabel()`.

See Also

`easylabel()` `easyMAplot()`

`gg_qqplot` *Log QQ p-value plot (ggplot2)*

Description

Produces a QQ plot via ggplot2. Requires a dataframe generated by `qqplot()`.

Usage

```
gg_qqplot(df, scheme = c("darkgrey", "royalblue"))
```

Arguments

`df`	A dataframe generated by `qqplot()`
`scheme`	Vector of 2 colours for plotting non-significant and significant SNPs

Value

A ggplot2 graphics plot object

`loadlabel` *Load easylabel saved state*

Description

Loads a file or easylab object containing saved state with specified labels and their positions.

Usage

```
loadlabel(object, data = NULL, ...)
```

Arguments

`object`	Either a character value specifying an rds file to be loaded, or an object of class 'easylab'.
`data`	Dataset (data.frame or data.table) to use for plot. If not specified, the function will first use embedded data included in `object`; if no data is embedded it will attempt to load the original data object as it was specified in the original call to `easylabel()`.
`...`	Additional arguments passed to `easylabel()`. Can be used to overwrite original arguments.

Value

By default no return value. Calls `easylabel()` which will open a shiny interactive session based on previous settings stored in an 'easylab' class object or rds file of such an object.

`qqplot` *Log QQ p-value plot*

Description

Fast function for generating a log quantile-quantile (QQ) p-value plot

Usage

```
qqplot(
  pval,
  fdr = NULL,
  fdr_cutoff = 0.05,
  scheme = c("darkgrey", "royalblue"),
  npoints = 5e+05,
  show_plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`pval`	A vector of p-values
`fdr`	An optional vector of FDR values to save time if previously computed. If not supplied, these will be calculated using `p.adjust()` using the Benjamini-Hochberg method.
`fdr_cutoff`	Cutoff for FDR significance
`scheme`	Vector of 2 colours for plotting non-significant and significant SNPs
`npoints`	Limits the number of non-significant points being plotted to speed up plotting. See details. Set to `NULL` to plot all points.
`show_plot`	Logical whether to produce a plot via base graphics or just return dataframe ready for plotting.
`verbose`	Whether to show messages
`...`	Optional plotting arguments passed to `qqplot2()`

Details

Produces a fast QQ plot. Particularly useful for analyses with very large numbers of p-values (such as eQTL analysis) which can be slow to plot. The function looks first for all comparisons which reached FDR at the designated cut-off and ensures all of these points are plotted. Additional points which typically overlap substantially near the origin are thinned by random sampling. In this way the plot can be reduced from millions of points to 500,000 points with a plot which is indistinguishable from one with all points plotted. For comparison, set `npoints` to `NULL` to plot all points as usual.

Calling `qqplot()` will result in a base graphics plot. The plotting dataframe is returned invisibly, so users can save time when refining plots by saving the dataframe produced by `qqplot()` and then invoking `qqplot2()` to simply plot the points. Users who prefer ggplot2 can also pass the dataframe generated by `qqplot()` to `gg_qqplot()`.

Value

Generates a plot using base graphics. Also returns a dataframe invisibly which can be used for downstream plotting via either `qqplot2()` or `gg_qqplot()`.

See Also

`qqplot2()` `gg_qqplot()`

`qqplot2` *Log QQ p-value plot (2nd stage)*

Description

Second stage plotting function which accepts dataframe generated by `qqplot()`. This can be used to avoid repeating computation of the QQ plot values.

Usage

```
qqplot2(df, scheme = c("darkgrey", "royalblue"), ...)
```

Arguments

`df`	A dataframe generated by `qqplot()`
`scheme`	Vector of 2 colours for plotting non-significant and significant SNPs
`...`	Optional plotting arguments passed to `plot()`

Value

No return value. Produces a base graphics plot.

`summary.easylab` *Summarise easylabel saved state object*

Description

Prints summary information on an easylabel saved state object.

Usage

```
## S3 method for class 'easylab'
summary(object, ...)
```

Arguments

`object`	Object of class 'easylab'
`...`	Optional arguments for compatibility

Value

No return value. Prints information on the original easylabel call and saved labels.

`volc1` *Example volcano data for vignette*

Description

Example DESeq2 volcano data for vignette

Usage

```
volc1
```

Format

Data frame with 6 rows and 6 variables

`volc2` *Example volcano data for vignette*

Description

Example limma volcano data for vignette

Usage

```
volc2
```

Format

Data frame with 6 rows and 6 variables

Index