Package ‘loon.ggplot’

June 26, 2020

Type Package

Title Making ‘ggplot2’ Plots Interactive with ‘loon’ and Vice Versa

Version 1.0.1

Description It provides a bridge between the ‘loon’ and ‘ggplot2’ packages. Data analysts who value the grammar pipeline provided by ‘ggplot2’ can turn these static plots into interactive ‘loon’ plots. Conversely, data analysts who explore data interactively with ‘loon’ can turn any ‘loon’ plot into a ‘ggplot2’ plot structure. The function ‘loon.ggplot()’ is applied to one plot structure will return the other.

License GPL-2

BugReports https://github.com/great-northern-diver/loon.ggplot/issues

Depends R (>= 3.4.0), tcltk, methods, loon (> 1.2.3), ggplot2

Imports stats, utils, grDevices, stringr, grid, GGally, gridExtra, magrittr, dplyr, rlang

Suggests png, tools, testthat, knitr, rmarkdown, tidyverse, covr, maps, nycflights13

LazyData true

RoxygenNote 7.1.0

Encoding UTF-8

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Zehao Xu [aut, cre], R. Wayne Oldford [aut]

Maintainer Zehao Xu <z267xu@uwaterloo.ca>

Repository CRAN

Date/Publication 2020-06-26 16:40:03 UTC
The glyph geom is used to create scatterplots with a variety of glyphs such as polygon glyph, serial axes glyph, image glyph, point range glyph and text glyph.

Usage

```r
geom_imageGlyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  images,
  width = 4,
  height = 3,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```
**geom_imageGlyph**

**Arguments**

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

- **data**
  The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x,10)`).

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

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

- **...**
  Other arguments passed on to `ggplot2::layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

- **images**
  A list of images (a raster object, bitmap image). If not provided, `geom_point()` will be called.

- **width**
  Width of image

- **height**
  Height of image

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

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

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

**Value**

- a `geom` layer

**See Also**

- `geom_polygonGlyph`, `geom_pointrangeGlyph`, `geom_serialAxesGlyph`, `geom_textGlyph`

**Examples**

```r
# image glyph

if(requireNamespace("png")) {

```
geom_pointrangeGlyph

```r
img_paths <- list.files(file.path(find.package(package = 'loon'), "images"), full.names = TRUE)
images <- lapply(img_paths, function(path) png::readPNG(path))
p <- ggplot(data = data.frame(x = 1:6, y = 1:6),
            mapping = aes(x = x, y = y)) +
            geom_imageGlyph(images = images, alpha = 0.4, width = 2, height = 1.5)

p
```

**Description**

The glyph geom is used to create scatterplots with a variety of glyphs such as polygon glyph, serialaxes glyph, image glyph, point range glyph and text glyph.

**Usage**

```r
geom_pointrangeGlyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ..., 
  ymin, 
  ymax, 
  showArea = TRUE, 
  linewidth = 1, 
  na.rm = FALSE, 
  show.legend = NA, 
  inherit.aes = TRUE
)
```

**Arguments**

- `mapping`: Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes` = `TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
- `data`: The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - If `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(x, 10)`).

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

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

**...**  
Other arguments passed on to ggplot2::layer These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

**ymin**  
vector with lower y-value of the point range. If not provided, `geom_point()` will be called.

**ymax**  
vector with upper y-value of the point range. If not provided, `geom_point()` will be called.

**showArea**  
If TRUE, the point pch is 21, else it is 1.

**linewidth**  
line width of whisker

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

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

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

**Details**

`geom_pointrangeGlyph()` is very close to `geom_pointrange` but with ‘loon’ API

**Value**

a geom layer

**Aesthetics**

`geom_...Glyph()` understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype
See Also

geom_imageGlyph, geom_pointrangeGlyph, geom_serialAxesGlyph, geom_textGlyph
geom_polygonGlyph, geom_imageGlyph, geom_serialAxesGlyph, geom_textGlyph

Examples

# point range glyph
p <- ggplot(data = data.frame(x = 1:3, y = 1:3),
    mapping = aes(x = x, y = y)) +
    geom_pointrangeGlyph(ymin=(1:3)-(1:3)/5, ymax=(1:3)+(1:3)/5)
p

Description

The glyph geom is used to create scatterplots with a variety glyphs such as polygon glyph, serialaxes glyph, image glyph, point range glyph and text glyph.

Usage

geom_polygonGlyph(
    mapping = NULL,
    data = NULL,
    stat = "identity",
    position = "identity",
    ...,
    polygon_x,
    polygon_y,
    showArea = TRUE,
    linewidth = 1,
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
)

Arguments

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

data The data to be displayed in this layer. There are three options:
    If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
geom_polygonGlyph

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

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

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

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

**...**
Other arguments passed on to `ggplot2::layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

**polygon_x**
Nested list of x-coordinates of polygons, one list element for each scatterplot point. If not provided, `geom_point()` will be called.

**polygon_y**
Nested list of y-coordinates of polygons, one list element for each scatterplot point. If not provided, `geom_point()` will be called.

**showArea**
Boolean to indicate whether area should be shown or not

**linewidth**
Line width of polygon

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

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

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

**Value**

A geom layer

**Aesthetics**

`geom_polygonGlyph()` understands the following aesthetics (required aesthetics are in bold):

- `x`
- `y`
- `alpha`
- `colour`
- `fill`
- `group`
- `shape`
- `size`
- `stroke`
- `linetype`
See Also

`geom_imageGlyph, geom_pointrangeGlyph, geom_serialAxesGlyph, geom_textGlyph`

Examples

```r
# polygon glyph
p <- ggplot(data = data.frame(x = 1:4, y = 1:4), 
            mapping = aes(x = x, y = y)) +
            geom_polygonGlyph(polygon_x = list(x_star, x_cross, x_hexagon, -x_airplane), 
                                polygon_y = list(y_star, y_cross, y_hexagon, y_airplane), 
                                colour = 'black', fill = 'red')

p
```

Description

The glyph geom is used to create scatterplots with a variety glyphs such as polygon glyph, serialaxes glyph, image glyph, point range glyph and text glyph.

Usage

```r
geom_serialAxesGlyph(
  mapping = NULL, 
  data = NULL, 
  stat = "identity", 
  position = "identity", 
  ..., 
  serialAxesData, 
  sequence = NULL, 
  linewidth = 1, 
  scaling = c("variable", "data", "observation", "none"), 
  axesLayout = c("parallel", "radial"), 
  showAxes = FALSE, 
  showArea = FALSE, 
  showEnclosing = FALSE, 
  axesColor = "black", 
  bboxColor = "black", 
  na.rm = FALSE,
  show.legend = NA, 
  inherit.aes = TRUE
)
```
Arguments

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

- **data**: The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x,10)`).

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

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

- **...**: Other arguments passed on to `ggplot2::layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

- **serialAxesData**: a serial axes numerical data set. If not provided, `geom_point()` will be called.

- **sequence**: vector with variable names that defines the axes sequence

- **linewidth**: line width of serial axes plot

- **scaling**: one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled. See Details for more information

- **axesLayout**: either "radial" or "parallel"

- **showAxes**: boolean to indicate whether axes should be shown or not

- **showArea**: boolean to indicate whether area should be shown or not

- **showEnclosing**: boolean to indicate whether enclosing should be shown or not

- **axesColor**: axes color

- **bboxColor**: bounding box color

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

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

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

Value

a geom layer
See Also

geom_polygonGlyph, geom_imageGlyph, geom_pointrangeGlyph, geom_textGlyph

Examples

# serial axes glyph
p <- ggplot(data = iris,
  mapping = aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_serialAxesGlyph(serialAxesData = iris[, -5],
                       axesLayout = "radial")
  p

Description

The glyph geom is used to create scatterplots with a variety glyphs such as polygon glyph, serialaxes glyph, image glyph, point range glyph and text glyph.

Usage

geom_textGlyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  text,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

Arguments

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

data The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
get_activeGeomLayers

stat
position
...
text
na.rm
show.legend
inherit.aes

Value

a geom layer

See Also

geom_polygonGlyph, geom_imageGlyph, geom_pointrangeGlyph, geom_serialAxesGlyph

Examples

# text glyph
p <- ggplot(data = data.frame(x = 1:26, y = 1:26),
  mapping = aes(x = x, y = y)) +
  geom_textGlyph(text = LETTERS, size = (1:26)/5)

get_activeGeomLayers

Description

‘get_activeGeomLayers’ will return the geom layer index which can be active

Usage

get_activeGeomLayers(ggObj)

Arguments

ggObj a ggplot object
get_activeGeomLayers

Details

`ggplot2loon` has an argument called `activeGeomLayers`. It is a vector to determine which geom layers can be active. The default setting is `integer(0)`, however, `ggplot2loon` will automatically search the first `geom_histogram` or `geom_point` layer to make it active. `get_activeGeomLayers` is more like a guidance and give us a hint which one can be set as active.

Value

a numerical vector of indicies (which layer can be interactive)

See Also

`ggplot2loon`

Examples

```r
df <- data.frame(x = 1:3, y = 1:3, colour = c(1,3,5))
xgrid <- with(df, seq(min(x), max(x), length = 50))
interp <- data.frame(
  x = xgrid,
  y = approx(df$x, df$y, xout = xgrid)$y,
  colour = approx(df$x, df$colour, xout = xgrid)$y
)
p1 <- ggplot(data = df, aes(x, y, colour = colour)) +
  geom_line(interp, mapping = aes(x, y, colour = colour), size = 2) +
  geom_point(size = 5)
agL <- get_activeGeomLayers(p1)
ggplot2loon(p1, activeGeomLayers = agL)

p2 <- ggplot(economics) +
  geom_rect(
    aes(xmin = start, xmax = end, fill = party),
    ymin = -Inf, ymax = Inf, alpha = 0.2,
    data = presidential
  ) +
  geom_text(
    aes(x = start, y = 2500, label = name),data = presidential,
    size = 3, vjust = 0, hjust = 0, nudge_x = 50
  ) +
  geom_line(aes(date, unemploy)) +
  scale_fill_manual(values = c("blue", "red"))
# none can be active
agL <- get_activeGeomLayers(p2)
#transparency is not allowed in tcltk
ggplot2loon(p2, ggGuides = TRUE, activeGeomLayers = agL)
```
ggplot2loon

Description

Create an interactive ‘loon’ widget from a ggplot object

Usage

~

```r
ggplot2loon(
  ggObj,
  activeGeomLayers = integer(0),
  ggGuides = FALSE,
  ..., 
  parent = NULL,
  pack = TRUE,
  tkLabels = NULL,
  exteriorLabelProportion = 1/5,
  canvasHeight = 700,
  canvasWidth = 850
)
```

Arguments

- `ggObj` a ggplot or ggmatrix object
- `activeGeomLayers` to determine which geom layer is active. Only `geom_point()` and `geom_histogram()` can be set as active geom layer(s) so far. (Notice, more than one `geom_point()` layers can be set as active layers, but only one `geom_histogram()` can be set as an active geom layer)
- `ggGuides` logical (default FALSE) to determine whether to draw a ggplot background or not.
- `...` named arguments to modify loon plot states
- `parent` parent widget path (Tk toplevel)
- `pack` logical (default TRUE) to pack widgets. If FALSE, widgets will be produced but won’t be packed and so will not appear in the display.
- `tkLabels` logical (or NULL) to indicate whether the plot(s) are to be wrapped with exterior labels (title, subtitle, xlabel or ylabel) using `tk.grid()`. If NULL (default), then exterior labels appear only for multiple facets. If TRUE exterior labels appear regardless; if FALSE no exterior labels appear.
- `exteriorLabelProportion` space assigned to the vertical height/horizontal width of each exterior label expressed as a proportion of a single plot’s height/width. Default is 0.2. This is translated to a row/column span = 1 / exteriorLabelProportion for the plot size in `tkgrid()`.
canvasHeight the height of canvas
canvasWidth the width of canvas

Value

a loon single or compound widget

Examples

```r
if(interactive()) {
  p <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
  g <- ggplot2loon(p)

  # tkLabels
  p <- ggplot(mtcars) + geom_point(aes(x = wt, y = mpg,
                                      colour = factor(gear))) + facet_wrap(~am)
  g1 <- ggplot2loon(p)
  g2 <- ggplot2loon(p, tkLabels = FALSE)
}

df <- data.frame(
  x = rnorm(120, c(0, 2, 4)),
  y = rnorm(120, c(1, 2, 1)),
  z = letters[1:3]
)
df2 <- dplyr::select(df, -z)
scatterplots <- ggplot(df, aes(x, y)) +
  geom_point(data = df2, colour = "grey70") +
  geom_point(aes(colour = z)) +
  facet_wrap(~z)

  # We can select the first geom_point layer to be
  # the active layer as in
  suppressWarnings(
    lp_scatterplots_active1 <- ggplot2loon(scatterplots,
                                            activeGeomLayers = 1,
                                            linkingGroup = "test")
  )
  # Here the grey points are linked (not the coloured ones)

  # We can select the second geom_point layer to be
  # the active layer as in
  lp_scatterplots_active2 <- ggplot2loon(scatterplots, activeGeomLayers = 2)
  # Here the colour points are linked

  # We can also select the both geom_point layers to be
  # the active layer as in
  suppressWarnings(
    lp_scatterplots_active12 <- ggplot2loon(scatterplots, activeGeomLayers = c(1,2))
  )
```
# Here the colour points and grey points are both linked

########### ggmatrix to loon ###########

pm <- GGally::ggpairs(iris, column = 1:4, ggplot2::aes(colour=Species))
lg <- ggplot2loon(pm)

---

**ggSerialAxes**, ggplot serialaxes

**Description**

The ggplot serialaxes graphics displays multivariate data either as a stacked star glyph plot, or as a parallel coordinate plot.

**Usage**

```r
ggSerialAxes(
  ggObj,
  data = NULL,
  axesLabels = NULL,
  showAxes = TRUE,
  showAxesLabels = TRUE,
  scaling = c("variable", "observation", "data", "none"),
  layout = c("parallel", "radial"),
  displayOrder = NULL,
  title = "",
  showLabels = TRUE,
  color = NULL,
  size = NULL,
  showGuides = TRUE,
  showArea = FALSE
)
```

**Arguments**

- **ggObj**: A `ggplot` object
- **data**: A data frame for serialaxes. If `NULL`, data must be set in `ggObj`
- **axesLabels**: A vector with variable names that defines the axes sequence.
- **showAxes**: Logical value to indicate whether axes should be shown or not
- **showAxesLabels**: Logical value to indicate whether axes labels should be shown or not
- **scaling**: one of 'variable', 'data', 'observation' or 'none' to specify how the data is scaled. See Details for more information
- **layout**: either "radial" or "parallel"
**displayOrder**  The display order of the observations.

**title**  title of the display

**showLabels**  Logical value to indicate whether label (mainly **title**) should be shown or not

**color**  Line color

**size**  Line width

**showGuides**  Logical value to indicate whether guides should be shown or not

**showArea**  Logical value to indicate whether to display lines or area

**Value**

a ggplot object

**Examples**

# Blank plot
p <- ggplot(data = mtcars, mapping = aes(colour = factor(cyl)))
# Add serial axes (returns a ggplot object)
g <- ggSerialAxes(p)
g
# An eulerian path of iris variables
# ordSeq <- PairViz::eulerian(4)
ordSeq <- c(1, 2, 3, 1, 4, 2, 3, 4)
ggSerialAxes(
  ggObj = ggplot(data = iris, mapping = aes(colour = Species)),
  axesLabels = colnames(iris)[ordSeq],
  layout = "radial"
)

---

**Description**

Pack a ggplot object forward to ggplot2loon expressions via a pipe-operator "%>%".

**Usage**

`gg_pipe(data, ggObj)`

**Arguments**

- **data**  a data frame to use for ggplot
- **ggObj**  a ggplot object to be passed though
Details

When "+" and ">%" both appear in pipe operations, ">%" takes the priority of "+", e.g:
```
mtcars %>% ggplot(aes(mpg, wt, colour = cyl)) + geom_point() %>% ggplot2loon(),
```
error would occur. The reason is
```
geom_point() %>% ggplot2loon()
```
would run before
```
ggplot(aes(mpg, wt, colour = cyl)) + geom_point().
```
Hence, we need a function `gg_pipe()` to pack the `ggplot` object and force operations happen in order.

Value

a `ggplot` evaluate object

Examples

```r
if(require(magrittr) && interactive()) {
  ## Not run:
  # Error
  g <- mtcars %>%
       ggplot(aes(mpg, wt, colour = cyl)) +
       geom_point() %>%
       ggplot2loon()

  ## End(Not run)
  g <- mtcars %>%
       gg_pipe(
            ggplot(aes(mpg, wt, colour = cyl)) + geom_point()
       ) %>%
       ggplot2loon()
}
```

---

g_getLocations

get locations for `ggmatrix`

Description

For the target compound `loon` plot, determines location in `ggmatrix`

Usage

```r
g_getLocations(target)
```

## Default S3 method:
g_getLocations(target)

## S3 method for class 'l_pairs'
g_getLocations(target)
Arguments

target the (compound) loon plot whose locations are needed to lay out.

Value

a list of an appropriate subset of the named location arguments `c("ncol", "nrow", "layout_matrix", "heights", "widths")`. `layout_matrix` is an `nrow` by `ncol` matrix whose entries identify the location of each plot in `g_getPlots()` by their index.

See Also

`l_getLocations`, `g_getPlots`

Description

For the target compound loon plot, determines all the ggplots based on the compound loon plot.

Usage

g_getPlots(target)

## Default S3 method:
g_getPlots(target)

## S3 method for class 'l_pairs'
g_getPlots(target)

Arguments

target the (compound) loon plot to be laid out.

Value

a list of ggplots.

See Also

`l_getPlots`, `g_getLocations`
**layout_coords**

Description

return the layout matrix of a list of loon plots

Usage

`layout_coords(target)`

Arguments

target an object `ggplot2loon()` returns

**lggplot**

Automatically create a loon widget

Description

It is retired. See `l_ggplot`

Usage

`lggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())`

Arguments

data Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
mapping Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
... Other arguments passed on to methods. Not currently used.
environment DEPRECATED. Used prior to tidy evaluation.
loon.ggplot

Description

A bridge between loon widgets and gg objects. It can take either a loon widget or a gg object (ggplot or ggmatrix), then create a corresponding gg (or loon) graphics.

Usage

loon.ggplot(x, ...)

## S3 method for class 'gg'
loon.ggplot(x, ...)

## S3 method for class 'loon'
loon.ggplot(x, ...)

Arguments

x        A loon widget or a ggplot object.
...      arguments used in either loon2ggplot() or ggplot2loon()

Value

If the input is a ggplot object, the output would be a loon widget; conversely, if the input is a loon widget, then it returns a ggplot object.

See Also

loon2ggplot, ggplot2loon

Examples

if(interactive()) {

    ######### loon --> gg #########
    # loon 3D plot
    l <- with(quakes,
        l_plot3D(long, lat, depth, linkingGroup = "quakes")
    )
    # equivalent to ’loon2ggplot(l)’
    g <- loon.ggplot(l)
    g # a ggplot object

    ######### gg --> loon ############

    # ggplot histogram
g <- ggplot(iris, mapping = aes(Sepal.Length, fill = Species)) + geom_histogram()
# equivalent to `ggplot2loon(g)`
l <- loon.ggplot(g)
l # a loon widget
}

Description
Create a `ggplot` object from a `loon` widget

Usage
loon2ggplot(target, ...)

## Default S3 method:
loon2ggplot(target, ...)

## S3 method for class 'l_plot'
loon2ggplot(target, ...)

## S3 method for class 'l_hist'
loon2ggplot(target, ...)

## S3 method for class 'l_plot3D'
loon2ggplot(target, ...)

## S3 method for class 'l_compound'
loon2ggplot(target, ...)

## S3 method for class 'l_layer_graph'
loon2ggplot(target, ...)

## S3 method for class 'l_layer_histogram'
loon2ggplot(target, ...)

## S3 method for class 'l_layer_scatterplot'
loon2ggplot(target, ...)

## S3 method for class 'l_pairs'
loon2ggplot(target, ...)

## S3 method for class 'l_serialaxes'
loon2ggplot(target, ...)}
## S3 method for class 'l_ts'
loon2ggplot(target, ...)

### Arguments

- **target**: a loon or a vector that specifies the widget, layer, glyph, navigator or context completely. The widget is specified by the widget path name (e.g. '.l0.plot'), the remaining objects by their ids.
- **...**: arguments used inside loon2ggplot(), not used by this method

### Value

a ggplot object

### Examples

```r
if(interactive()) {
  l <- l_plot(iris, color = iris$Species)
  p <- loon2ggplot(l)
  p # a ggplot object
  str(p)
  # add themes
  p + geom_smooth() + theme_linedraw()
}
```

---

### Description

`l_ggplot()` wraps function `ggplot` with assigning a new class "lggplot" to the output `ggplot` object and returns a `lggplot` object. When a `ggplot` object is processed, S3 method `print.ggplot` is rendered, however, if a `lggplot` object is processed, S3 method `print.lggplot` will be rendered which will return a loon widget

### Usage

```r
l_ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

### Arguments

- **data**: Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
- **mapping**: Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
... Other arguments passed on to methods. Not currently used.

Value

It will return a `l ggplot` object with class `c("l ggplot","gg","ggplot")`. Then print a `loon` plot automatically.

See Also

`ggplot`, `ggplot2loon`, `print.l ggplot`

Examples

```r
if(interactive()) {
  p <- l ggplot(mpg, aes(displ, cty)) +
      geom_point() +
      facet_grid(rows = vars(drv))
  # p is a 'l ggplot' object, 'print.l ggplot(p)' is called automatically.
  # Then, the 'l ggplot' object will be transformed to a 'loon' widget
  p
}
```

## Not run:
# get widgets from current path
# suppose the path of 'p' is '.l0.ggplot'
q <- l getFromPath('.l0.ggplot')
# q is a 'loon' widget
q

## End(Not run)

---

<table>
<thead>
<tr>
<th>polygonGlyph</th>
<th>Polygon glyph coordinates</th>
</tr>
</thead>
</table>

Description

Some useful polygon coordinates

Usage

- `x_star`
- `y_star`
- `x_cross`
- `y_cross`
- `x_hexagon`
print.lggplot

y_hexagon
x_airplane
y_airplane

Format
An object of class numeric of length 10.
An object of class numeric of length 10.
An object of class numeric of length 12.
An object of class numeric of length 12.
An object of class numeric of length 6.
An object of class numeric of length 6.
An object of class numeric of length 32.
An object of class numeric of length 32.

See Also
geom_polygonGlyph

Examples
if(requireNamespace("grid")) {
  library(grid)
  grid.newpage()
  grid.polygon(x=(x_star + 1)/2,
    y=(1 - y_star)/2)
  grid.newpage()
  grid.polygon(x=(x_cross + 1)/2,
    y=(y_cross + 1)/2)
  grid.newpage()
  grid.polygon(x=(x_hexagon + 1)/2,
    y=(y_hexagon + 1)/2)
  grid.newpage()
  grid.polygon(x=(-x_airplane + 4)/10,
    y=(-y_airplane + 4)/10)
}

print.lggplot  Explicitly draw plot

Description
Explicitly draw plot
print.lggplot

Usage

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

Arguments

x          plot to display
...        other arguments used to modify function ggplot2loon

Value

Invisibly returns a loon widget
Index

*Topic datasets
  polygonGlyph, 23
  y_airplane (polygonGlyph), 23
  y_cross (polygonGlyph), 23
  y_hexagon (polygonGlyph), 23
  y_star (polygonGlyph), 23

aes(), 3, 4, 6, 9, 10
aes_(), 3, 4, 6, 9, 10
as.character, 11
borders(), 3, 5, 7, 9, 11
fortify(), 3, 4, 7, 9, 10
g_getLocations, 17, 18
g_getPlots, 18, 18
geom_imageGlyph, 2, 6, 8, 10, 11
geom_pointrange, 5
geom_pointrangeGlyph, 3, 4, 6, 8, 10, 11
geom_polygonGlyph, 3, 6, 6, 10, 11, 24
geom_serialAxesGlyph, 3, 6, 8, 8, 11
gem_textGlyph, 3, 6, 8, 10, 10
get_activeGeomLayers, 11
            gg_pipe, 16
            ggplot, 23
            ggplot(), 3, 4, 6, 9, 10
            ggplot2loon, 12, 13, 20, 23
            ggSerialAxes, 15
            l_getLocations, 18
            l_getPlots, 18
            l_ggplot, 19, 22
            layout_coords, 19
            lggplot, 19
            loon.ggplot, 20
            loon2ggplot, 20, 21
            polygonGlyph, 23
            print.lggplot, 23, 24

x_airplane (polygonGlyph), 23
x_cross (polygonGlyph), 23
x_hexagon (polygonGlyph), 23
x_star (polygonGlyph), 23