Package ‘loon.ggplot’

June 10, 2021

Type Package

Title Making 'ggplot2' Plots Interactive with 'loon' and Vice Versa

Version 1.2.1

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 (>= 1.3.2), ggplot2

Imports stats, ggmulti, utils, grDevices, grid, GGally, gridExtra, scales, rlang

Suggests magrittr, tidyr, dplyr, gtable, png, tools, tibble, testthat, knitr, rmarkdown, covr, maps, hexbin, nycflights13, ggplot2movies

RoxygenNote 7.1.1

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 2021-06-10 12:50:06 UTC
R topics documented:

active ................................................................. 2
Cartesianxy2Polarxy ............................................. 3
gt_activeGeomLayers ........................................... 4
gt_scaledData ..................................................... 5
gplot2loon .......................................................... 6
gg_pipe .............................................................. 8
g_getLocations .................................................... 9
g_getPlots ........................................................... 10
hover ................................................................. 11
interactivity ......................................................... 12
is.CoordPolar ....................................................... 14
is_l_ggplot .......................................................... 14
layout_coords ...................................................... 14
linking ............................................................... 15
loon.ggplot .......................................................... 16
loon2ggplot ......................................................... 17
loonLayer ........................................................... 20
l_getSubtitles ...................................................... 21
l_ggplot ............................................................ 21
print_l_ggplot ...................................................... 23
scaleBox ............................................................. 24
scale_multi .......................................................... 24
selection ............................................................ 26
zoom ................................................................. 27

Index 29

---

active

Modify the active component

Description

Set active or activeGeomLayers

Usage

active(active = NULL, activeGeomLayers = NULL)

Arguments

active  a logical determining whether points appear or not (default is TRUE for all points). If a logical vector is given of length equal to the number of points, then it identifies which points appear (TRUE) and which do not (FALSE).
activeGeomLayers
determine which geom layer is interactive. 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)

Value
a ggproto object

See Also
linking, selection, zoom, hover, interactivity

Examples
if(interactive()) {

  # set active layer
  l_ggplot(mtcars, aes(mpg, wt, shape = factor(cyl))) +
  geom_point(colour = "black", size = 4.5) +
  geom_point(colour = "pink", size = 4) +
  geom_point(aes(shape = factor(cyl))) +
  # only show manual transmission cars
  # in the second interactive layer
  active(active = mtcars$am == 1,
         activeGeomLayers = 2)

  # Then, click the 'reactivate' button on loon inspector
  # to display all interactive points
}

Cartesianxy2Polarxy  Transform the x, y positions from a Cartesian coordinate to a polar coordinate

Description
Used in the ‘loonLayer’ construction to access the x, y positions embedded in the polar coordinate system.

Usage
Cartesianxy2Polarxy(layerGeom, coordinates, data, ggplotPanelParams, ...)
get_activeGeomLayers

Arguments

layerGeom  A ggplot layer object
coordinates A ggplot object coordinate system
data  the data used for the transformation
ggplotPanelParams  some non-data panel parameters, i.e. the range of theta, the range of radius, theta major, theta minor, etc. It is obtained from the ggplot_build(p)$layout$panel_params where "p" is a ggplot object
...
for further use

Examples

p <- ggplot(mtcars, aes(wt, mpg)) +
    geom_point() +
    coord_polar()

layerGeom <- p$layers[[1L]]$geom
coordinates <- p$coordinates
build <- ggplot_build(p)
data <- build$data[[1L]]
ggplotPanelParams <- build$layout$panel_params[[1L]]
polarXY <- Cartesianxy2Polarxy(layerGeom, coordinates, data, gggplotPanelParams)
plot(polarXY$x, polarXY$y)

Description

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

Usage

get_activeGeomLayers(ggObj)

Arguments

  ggObj  a ggplot object

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.
**get_scaledData**

*scale data*

**Description**

It is mainly used in serial axes

**Value**

a numerical vector of indices (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 interactive
agL <- get_activeGeomLayers(p2)
#transparency is not allowed in tcltk
ggplot2loon(p2, ggGuides = TRUE, activeGeomLayers = agL)
```
Usage

g_get_scaledData(
    data,
    sequence = NULL,
    scaling = c("variable", "data", "observation", "none"),
    displayOrder = NULL,
    keep = FALSE,
    as.data.frame = FALSE
)

Arguments

data A data frame
sequence vector with variable names that defines the axes sequence. If NULL, it will be set as the column names automatically.
scaling one of ‘variable’, ‘data’, ‘observation’ or ‘none’ to specify how the data is scaled.
displayOrder the order of the display
keep If TRUE, return the variables not shown in sequence as well; else only return the variables defined in sequence.
as.data.frame Return a matrix or a data.frame

Description

Create an interactive ‘loon’ widget from a ggplot object

Usage

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

*ggObj*  
*a ggplot or ggmatrix object*

...  
*named arguments to modify loon plot states*

*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)

*layerId*  
*n numerical; which layer to scale to*

*scaleToFun*  
*scale to function. See zoom.*

*ggGuides*  
*logical (default FALSE) to determine whether to draw a ggplot background or not.*

*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.*

*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*

*tkLabels*  
*Deprecated: logical (or NULL) to indicate whether the plot(s) are to be wrapped by exterior labels (title, subtitle, xlabel or ylabel) using *tk.grid()**

Value

*a loon single or compound widget*

Examples

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

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

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)

---

**gg_pipe**

**Pipe ggplot object**

**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 through
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()
}
```

---

\textit{g_getLocations}

\textit{get locations for \texttt{ggmatrix}}

Description

For the target compound \texttt{loon} plot, determines location in \texttt{ggmatrix}

Usage

\texttt{g_getLocations(target)}

## Default S3 method:
\texttt{g_getLocations(target)}

## S3 method for class 'l_pairs'
\texttt{g_getLocations(target)}
Arguments

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

Value

da 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, asAes = TRUE, selectedOnTop = TRUE)

## Default S3 method:

## S3 method for class 'l_pairs'

g_getPlots(target, asAes = TRUE, selectedOnTop = TRUE)

Arguments

target the (compound) loon plot

asAes logical; set aesthetics attributes, i.e. ‘color’, ‘fill’ as variables (default TRUE) or general visual properties (FALSE). See details

selectedOnTop logical and default is TRUE; whether to display the selected points on top. See details.

Value

a list of ggplots.

See Also

l_getPlots, g_getLocations
hover

Modify the hover component

Description

Query in interactive graphics

Usage

hover(itemLabel = NULL, showItemLabels = NULL)

Arguments

itemLabel The customized querying information.
showItemLabels A logical value. Show item labels or not. Default is FALSE

Value

a ggproto object

See Also

active, linking, zoom, selection, interactivity

Examples

if(interactive()) {

l_ggplot(mpg, mapping = aes(x = disp, y = cty)) +
  geom_point(size = 4) +
  # push the states of scatter plot to the histogram
  hover(itemLabel =
    with(mpg,
      paste0("model: ", manufacturer, ", ", model, "\n",
        "year: ", year, "\n",
        "drive way: ", drv, "\n",
        "fuel type: ", fl)
    ),
    showItemLabels = TRUE
  )
  # hover the mouse on top of any point to query
}

interactivity Modify the interactivity component

Description
Set interactive components (e.g. linking, selection, etc)

Usage
interactivity(
  linkingGroup = NULL,
  linkingKey = NULL,
  linkedStates = NULL,
  sync = NULL,
  active = NULL,
  activeGeomLayers = NULL,
  selected = NULL,
  selectBy = NULL,
  selectionLogic = NULL,
  layerId = NULL,
  scaleToFun = NULL,
  itemLabel = NULL,
  showItemLabels = NULL,
  ...
)

Arguments
linkingGroup A character. Plots only in the same linkingGroup can be linked
linkingKey LinkingKey is the key of linking. Each object in one plot has a unique linking key. Elements in different plots are linked if they share the same linking keys.
linkedStates The states to be linked. It can be "color", "selected", "active", "size" and "glyph" for a 'l_plot' object and "color", "selected", "active" for a 'l_hist' object.
sync The way to synchronize several linked plots. It can be either "pull" (default) or "push". If the sync is "pull", the linked states (aesthetics attributes, e.g. "color", "selected", ...) of the new plot will be pulled from the linked plots; if the sync is "push", the linked states of the new plot will be pushed to the linked plots.
active a logical determining whether points appear or not (default is TRUE for all points). If a logical vector is given of length equal to the number of points, then it identifies which points appear (TRUE) and which do not (FALSE).
activeGeomLayers determine which geom layer is interactive. 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)
**interactivity**

selected: A logical vector. If it is set as TRUE, the elements are highlighted as the graphics are constructed. Default is FALSE

selectBy: Select by "sweeping" (default) or "brushing".

selectionLogic: Selection logic. One of "select" (default), "deselect" and "invert". See details.

layerId: numerical; which layer to scale to

scaleToFun: scale to function. See `zoom`.

itemLabel: The customized querying information.

showItemLabels: A logical value. Show item labels or not. Default is FALSE

... named arguments to modify loon plot states. See `l_info_states`

**Details**

In interactive graphics, there are several fundamental infrastructures, such as querying, linking and selection. Component `interactivity` is used to set these features.

**Interactivity**

<table>
<thead>
<tr>
<th>Interactivity</th>
<th>Description</th>
<th>Subfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linking</td>
<td>Linking several plots to discover the pattern of interest</td>
<td>linking</td>
</tr>
<tr>
<td>Selection</td>
<td>Highlight the subset of interest</td>
<td>selection</td>
</tr>
<tr>
<td>Active</td>
<td>Determine which points appear</td>
<td>active</td>
</tr>
<tr>
<td>Hover</td>
<td>Query in interactive graphics</td>
<td>hover</td>
</tr>
<tr>
<td>Zoom</td>
<td>Region Modification</td>
<td>zoom</td>
</tr>
</tbody>
</table>

**Value**

a `ggproto` object

**Examples**

```r
if(interactive()) {
  # Modify the 'linkingGroup' and 'origin' of a hist object
  l_ggplot(mtcars, mapping = aes(x = wt)) +
    geom_histogram() +
    interactivity(linkingGroup = "mt", origin = 2)

  # linking with the histogram
  l_ggplot(mtcars, mapping = aes(x = wt, y = hp)) +
    geom_point(size = 4) +
    interactivity(linkingGroup = "mt") +
    facet_wrap(~cyl)

  p <- ggplot(economics_long, aes(value)) +
    facet_wrap(~variable, scales = 'free_x') +
    geom_histogram()

  # 'p' is a ggplot object
  p
  # turn static 'ggplot' to interactive 'loon'
  p + interactivity()
}
```
**is.CoordPolar**

*Is polar coordinate system?*

**Description**

Determine whether the `ggplot` object has polar coordinate system

**Usage**

```r
is.CoordPolar(coord)
```

**Arguments**

- `coord` A `ggplot` object coordinate system

---

**is.l_ggplot**

*Reports whether x is a l_ggplot object*

**Description**

Reports whether `x` is a `l_ggplot` object

**Usage**

```r
is.l_ggplot(x)
```

**Arguments**

- `x` An object to test

---

**layout_coords**

*layout matrix*

**Description**

return the layout matrix of a list of loon plots

**Usage**

```r
layout_coords(target)
```

**Arguments**

- `target` an object `ggplot2loon()` returns
**Value**

a layout coordinate matrix

---

**Description**

In interactive graphics, linking is often used to discover the patterns of interest in several plots.

**Usage**

```r
linking(
  linkingGroup = NULL,
  linkingKey = NULL,
  linkedStates = NULL,
  sync = NULL
)
```

**Arguments**

- **linkingGroup**: A character. Plots only in the same linkingGroup can be linked.
- **linkingKey**: LinkingKey is the key of linking. Each object in one plot has a unique linking key. Elements in different plots are linked if they share the same linking keys.
- **linkedStates**: The states to be linked. It can be "color", "selected", "active", "size" and "glyph" for a 'l_plot' object and "color", "selected", "active" for a 'l_hist' object.
- **sync**: The way to synchronize several linked plots. It can be either "pull" (default) or "push". If the sync is "pull", the linked states (aesthetics attributes, e.g. "color", "selected", ...) of the new plot will be pulled from the linked plots; if the sync is "push", the linked states of the new plot will be pushed to the linked plots.

**Value**

a ggproto object

**See Also**

`active`, `selection`, `zoom`, `hover`, `interactivity`

**Examples**

```r
if(interactive() && requireNamespace("dplyr")) {
  h <- l_hist(mtcars$hp,
              linkingKey = rownames(mtcars),
              linkingGroup = "mtcars")

  mtcars %>%
```
mutate(carName = rownames(mtcars)) %>%
  l.ggplot(mapping = aes(x = wt, y = hp, color = factor(cyl))) +
  geom_point(size = 4) +
  # push the states of scatter plot to the histogram
  linking(linkingGroup = "mtcars",
          linkingKey = ~carName,
          sync = "push")
}

loon.ggplot

---

Description

A bridge between loon widgets and gg objects. It can take either a loon widget, a gg object (ggplot, GGally::ggmatrix) or a l.ggplot object, 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, ...)

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

Arguments

x A loon widget, a ggplot object or a l.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. If it is a l.ggplot object, loon.ggplot helps to return a loon widget.

See Also

Richer examples are in loon2ggplot, ggplot2loon, l.ggplot
Examples

```r
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

  ############ l_ggplot ############
  p <- l_ggplot(mpg, aes(displ, fill = factor(cyl))) +
    geom_histogram()
  class(p)
  # Function 'print.l_ggplot' is called automatically
  p
  # Function 'loon.ggplot' helps to return a loon widget
  q <- loon.ggplot(p)
  q
}
```

loon2ggplot

Turn a loon widget to a ggplot object

Description

Create a ggplot object from a loon widget

Usage

loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

# Default S3 method:
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

# S3 method for class 'l_plot'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)
loon2ggplot

## S3 method for class 'l_hist'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_plot3D'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_compound'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_layer_graph'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_layer_histogram'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_layer_scatterplot'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_pairs'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_serialaxes'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

## S3 method for class 'l_ts'
loon2ggplot(target, asAes = TRUE, selectedOnTop = TRUE, ...)

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.

**asAes**
logical; set aesthetics attributes, i.e. `color`, `fill` as variables (default `TRUE`) or general visual properties (`FALSE`). See details

**selectedOnTop**
logical and default is `TRUE`; whether to display the selected points on top. See details.

... arguments used inside `loon2ggplot()`, not used by this method

Details

In ggplot, generally speaking, there are two ways to set the aesthetics attributes, either take it as variables `asAes = TRUE` (set in function `aes()`) or visual properties `asAes = FALSE`. The main benefits to consider it as variables are that 1. legend could be shown; 2. convenient for further analysis.

In loon, if the selected state is changed, the order of the points will be changed so that the highlighted points will be displayed on top. To turn it static, if `selectedOnTop = TRUE`, the points will be partitioned into two groups, one group presents the un-selected (un-highlighted) points and the other group presents selected (highlighted) points. the un-selected group will be drawn first, then
selected group will be displayed on top of it; if `selectedOnTop = FALSE`, no partition is applied so that the displayed order will be identical to the original data set order. This is very helpful when further analysis will be operated in ggplot graphical system, e.g. `+ facet_wrap(...)`. See examples.

**Value**

a ggplot object

**Examples**

```r
if(interactive()) {
  lp <- l_plot(iris,
               color = iris$Species,
               glyph = "circle")
  gp <- loon2ggplot(lp)
  gp # a ggplot object

  # add smooth layer, grouped by color
  gp +
  geom_smooth(aes(color = color)) +
  # give meaningful legend label names
  scale_color_manual(
    # make sure the order is correct
    values = unique(hex12tohex6(lp[, "Var"])),
    labels = c("setosa", "versicolor", "virginica")
  )

  # histogram
  lh <- l_hist(mtcars$mpg,
               color = factor(mtcars$gear))
  gh0 <- loon2ggplot(lh)
  gh0 + facet_wrap(~fill)
  gh1 <- loon2ggplot(lh, asAes = FALSE)
  # Expect the legend, they both are identical
  gh1
  ## Not run: # ERROR # The bins are constructed by `ggplot2::geom_rect()` # Very limited operations can be made
  gh1 + facet_wrap(~fill)

  ## End(Not run)

  # Argument `selectedOnTop`
  p <- l_plot(iris, color = iris$Species)
  p["selected"[iris$Petal.Length > 5] <- TRUE
  g <- loon.ggplot(p)
}
# It looks correct.
g
# facet by "Species"
## Not run:
g + facet_wrap(iris$Species)
## End(Not run)

Something is wrong here. There is a pink point (at least one)
in species “versicolor”! The reason is because after points are
highlighted, the displayed order has been changed. One way to
fix it is to set the `selectedOnTop` as FALSE.

loon.ggplot(p, selectedOnTop = FALSE) +
  facet_wrap(iris$Species)
}

---

loonLayer

Transform geom layers to loon layers

Description

Function `loonLayer` is used to create loon non-interactive layers. For some ggplot2 extension
packages, one can edit this function to realize the transformation.

Usage

loonLayer(
  widget,
  layerGeom,
  data,
  ggplotPanelParams,
  ggObj,
  parent,
  label,
  ...
)

Arguments

  widget        a loon widget
  layerGeom    a ggplot Geom layer object
  data          a data frame (i.e. x, y, etc) of this particular layer
  ggplotPanelParams
                ggplot panel parameters
  ggObj         the ggplot object
  parent        a valid Tk parent widget path.
  label         label used in the layers inspector
  ...           not for users
l_getSubtitles

Return the subtitles from an l_facet_ggplot object

Description

Return the subtitles from an l_facet_ggplot object

Usage

l_getSubtitles(target)

Arguments

target an l_facet_ggplot object. If the ggplot object is faceted (either by facet_wrap or facet_grid), an l_facet_ggplot object will be returned once it is turned to a loon plot.

Value

A list of labels, i.e. subtitles, labels, title, etc

Examples

if(interactive()) {
  p <- ggplot(mpg, aes(displ, hwy)) +
        geom_point() +
        facet_wrap(vars(class))
  lp <- loon.ggplot(p)
  l_getSubtitles(lp)
}

l_ggplot

Automatically create a loon widget

Description

Create a loon widget with ggplot syntax

Usage

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.
- **environment**: DEPRECATED. Used prior to tidy evaluation.

Details

Function `l_ggplot()` wraps function `ggplot()` with assigning an additional class "l_ggplot" to the output. The returned object is called an `l_ggplot` object. To draw a ggplot object, S3 method `print.ggplot` will be rendered so that a static graphic is displayed. While, for an `l_ggplot()` object, S3 method `print.l_ggplot` will be rendered which will return an interactive loon widget.

Value

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

See Also

- `ggplot`, `ggplot2loon`, `print.l_ggplot`
- `loon.ggplot`

Examples

```r
if(interactive()) {
  p <- l_ggplot(mpg, aes(displ, cty)) +
    geom_point(
      size = 4,
      mapping = aes(color = factor(cyl))
    )
  # p is an `l_ggplot` object, `print.l_ggplot(p)` will be called automatically.
  # Then, at printing time, an `l_ggplot` object will be transformed to a `loon` widget
  p

  ## Not run:
  # Assign a widget 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)

  # An alternative way to return a real loon widget from `p` (a `l_ggplot` object)
  # is to call the function `loon.ggplot()`.
```
q <- loon.ggplot(p)
q

# pipe more components
p +
  facet_grid(rows = vars(drv)) +
  linking(linkingGroup = "mpg") +
  ggtitle("displ versus cty")
# a linked bar plot
l_hist(mpg$class, linkingGroup = "mpg")

# a 3D object
# press the button key 'R' to rotate the plot
l_ggplot(mtcars,
          mapping = aes(x = wt, y = hp, z = drat)) +
          geom_point(size = 4) +
          scale_multi()
)

print.l_ggplot

Explicitly draw plot

Description

Explicitly draw plot

Usage

## S3 method for class 'l_ggplot'
print(x, message = TRUE, ...)

Arguments

x plot to display
message logical; if TRUE, the way to create handle will be printed out.
... other arguments used to modify function ggplot2loon

Value

Invisibly returns a loon widget
scaleBox

Box scaling in 3D rotation

Description

the variable is scaled to have equal ranges and, when center = TRUE, to be centred by the average of the min and max.

Usage

scaleBox(center = TRUE)

Arguments

center

either a logical value or numeric-alike vector of length equal to the number of columns of x, where 'numeric-alike' means that as.numeric(.) will be applied successfully if is.numeric(.) is not true.

Value

A trans object

See Also

l_scale3D

scale_multi

Position scales for continuous data (x, y & z)

Description

Scaling the coordinates for 3D visualization

Usage

scale_multi(trans = scaleBox(center = TRUE), ...)

Arguments

trans

For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time". A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called <name>_trans (e.g., scales::boxcox_trans()). You can create your own transformation with scales::trans_new().
scale_multi

... Other arguments passed on to scale_(x|y)_continuous(). To set the position scales, three scales (x, y, z) has to be set simultaneously.

Details

In 3D rotation, different scales of variables x, y and z may cause an issue that the points appear to be off the window even with a minor tweak. Additionally, if one variable is in a large scale, the shape of the 3D plot may be dominated. Setting scale_multi can ensure the scales in the same measurement, as we rotate the plot, most points will stay inside the current view.

Value

a list of the ggproto objects

Examples

```r
if(interactive()) {

dsamp <- dplyr::sample_n(diamonds, 100)

## Not run:
# press `R`, then rotate with a minor tweak,
# Issues:
# 1: the points are off the window
# 2: Always in a line shape
l_ggplot(dsamp, aes(x = carat, y = price,

  z = depth, colour = color)) +

  geom_point()
## End(Not run)

# set scales
l_ggplot(dsamp, aes(x = carat, y = price,

  z = depth, colour = color)) +

  geom_point() +
  scale_multi()

# customized `trans`
logp1_base10_trans <- scales::trans_new(
  name = “logp”,
  trans = function(x) log(x + 1, base = 10),
  inverse = function(x) 10**x - 1,
  breaks = scales::log_breaks())

l_ggplot(dsamp, aes(x = carat, y = price,

  z = depth, colour = color)) +

  geom_point() +
  scale_multi(trans = logp1_base10_trans)
}
```
In interactive graphics, selection is one of the most fundamental tools and used to highlight the subset of interest.

```
selection(selected = NULL, selectBy = NULL, selectionLogic = NULL)
```

**Arguments**

- `selected`: A logical vector. If it is set as TRUE, the elements are highlighted as the graphics are constructed. Default is FALSE.
- `selectBy`: Select by "sweeping" (default) or "brushing".
- `selectionLogic`: Selection logic. One of "select" (default), "deselect" and "invert". See details.

**Details**

There are two ways to directly select elements on the scatterplot using the mouse: either by "sweep" or by "brushing". "Sweeping" allows us to sweep out a contiguous area of the plot, while, in "brushing", the area is always fixed during the selection.

The selection logic gives users more flexibility that users cannot only highlight the elements, but also can delight or invert (the highlighted to delighted, vice versa) the elements.

**Value**

a ggproto object

**See Also**

active, linking, zoom, hover, interactivity

**Examples**

```r
if(interactive()) {
  # highlight the four gear cars
  fourGear <- rep(FALSE, nrow(mtcars))
  fourGear[mtcars$gear == 4] <- TRUE
  l_ggplot(mtcars, mapping = aes(x = wt, y = hp, color = factor(cyl))) +
    geom_point(size = 4) +
  # push the states of scatter plot to the histogram
  selection(selected = fourGear)
}
```
Description

Modify the zoomX, zoomY, panX, panY, etc to change the plot region

Usage

zoom(layerId = NULL, scaleToFun = NULL)

Arguments

layerId  numerical; which layer to scale to
scaleToFun  scale to function. See details.

Details

Argument layerId is used for additional plot region settings. If the layerId is set as NULL (default), the region of the interactive graphics loon will be determined by the ggplot object (i.e. coord_cartesian, xlim, etc); else one can use scaleToFun to modify the region of the layer.

The scaleToFun is a function to scale the region. If it is NULL (default), based on different layers, different scale functions will be applied. For example, if the layer is the main graphic model, i.e. l_plot l_hist, then the default scaleToFun is l_scaleto_plot; else if the layer is a general l_layer widget, the default scaleToFun would be l_scaleto_layer (see get_activeGeomLayers). If it is not NULL, users can select one that precisely tailor their own problems. The table shows the available scaleToFun functions

<table>
<thead>
<tr>
<th>scale to</th>
<th>Subfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot</td>
<td>l_scaleto_plot</td>
</tr>
<tr>
<td>world</td>
<td>l_scaleto_world</td>
</tr>
<tr>
<td>active</td>
<td>l_scaleto_active</td>
</tr>
<tr>
<td>selected</td>
<td>l_scaleto_selected</td>
</tr>
<tr>
<td>layer</td>
<td>l_scaleto_layer</td>
</tr>
</tbody>
</table>

Expect all these, users can customize their own function. Note that, the arguments should match the ones of functions shown in the table.

Value

a ggproto object

See Also

active, linking, selection, hover, interactivity
Examples

```r
if(interactive()) {
  p <- l_ggplot(mtcars,
    mapping = aes(x = hp, y = mpg)) +
    geom_point(mapping = aes(color = factor(gear))) +
    geom_smooth(data = mtcars[mtcars$gear == 4, ],
       method = "lm")
  # a scatter plot with a fitted line on 4 gear cars
  p
  # scale to the second layer (smooth line)
  p + zoom(layerId = 2)
  # highlight the 3 gear cars
  # scale to the selected points
  p +
    selection(mtcars$gear == 3) +
    zoom(layerId = 1,
       scaleToFun = loon::l_scaleto_selected)
}
```
Index

active, 2, 11, 13, 15, 26, 27

Cartesianxy2Polarxy, 3

g_getLocations, 9, 10
g_getPlots, 10, 10
g_get_activeGeomLayers, 4, 27
g_get_scaledData, 5
gg_pipe, 8
ggplot, 22
ggplot2loon, 5, 6, 16, 22

hover, 3, 11, 13, 15, 26, 27

interactivity, 3, 11, 12, 15, 26, 27

is.CoordPolar, 14

is.l_ggplot, 14

l_getLocations, 10
l_getPlots, 10
l_getSubtitles, 21
l_ggplot, 16, 21
l_info_states, 13
l_scale3D, 24
l_scaleto_active, 27
l_scaleto_layer, 27
l_scaleto_plot, 27
l_scaleto_selected, 27
l_scaleto_world, 27
layout_coords, 14
linking, 3, 11, 13, 15, 26, 27
loon.ggplot, 16, 22
loon2ggplot, 16, 17
loonLayer, 20

print.l_ggplot, 22, 23

scale_multi, 24
scaleBox, 24
selection, 3, 11, 13, 15, 26, 27

zoom, 3, 7, 11, 13, 15, 26, 27