

Package ‘loon.ggplot’

September 28, 2021

Type Package

Title A Grammar of Interactive Graphics

Version 1.3.0

Description Provides a bridge between the 'loon' and 'ggplot2' packages. Extends the grammar of ggplot to add clauses to create interactive 'loon' plots. Existing ggplot(s) can be turned into interactive 'loon' plots and 'loon' plots into static ggplot(s); the function 'loon.ggplot()' is the bridge from one plot structure to 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, ggmulti

Imports stats, utils, grDevices, grid, gridExtra, scales, patchwork, rlang

Suggests GGally, magrittr, tidyr, zenplots, 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 2021-09-28 14:00:05 UTC

R topics documented:

active	2
Cartesianxy2Polarxy	3
get_activeGeomLayers	4
get_scaledData	5
ggplot2loon	6
gg_pipe	8
g_getLocations	9
g_getPlots	10
hover	11
interactivity	12
is.CoordPolar	14
is.l_ggplot	15
layout_coords	15
linking	16
loon.ggplot	17
loon2ggplot.l_compound	18
loonLayer	23
l_getSubtitles	24
l_ggplot	25
print.l_ggplot	26
scaleBox	27
scale_multi	28
selection	29
zoom	30
Index	32

active *Modify the active component*

Description

Set active and/or activeGeomLayers

Usage

```
active(active = NULL, activeGeomLayers = NULL)
```

Arguments

active a logical or a logical vector of length n that determines which observations are active (TRUE and hence appear in the plot) and which are inactive (FALSE and hence do not appear). Default is TRUE.

activeGeomLayers

determine which geom layer is interactive by its ‘geom_...’ position in the grammar of the expression. Currently, only `geom_point()` and `geom_histogram()` can be set as the active geom layer(s) so far. (N.B. more than one `geom_point()` layer can be set as an active layer, but only one `geom_histogram()` can be set as an active geom layer and it can be the only active 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	<i>Transform the x, y positions from a Cartesian coordinate to a polar coordinate</i>
---------------------	---

Description

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

Usage

```
Cartesianxy2Polarxy(layerGeom, coordinates, data, ggplotPanelParams, ...)
```

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 <code>ggplot_build(p)\$layout\$panel_params</code> 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, ggplotPanelParams)
plot(polarXY$x, polarXY$y)
```

get_activeGeomLayers *active geom layers*

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.

Value

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

See Also

[ggplot2loon](#)

Examples

```
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)
```

get_scaledData

scale data

Description

It is mainly used in serial axes

Usage

```

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

```

Arguments

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

ggplot2loon

ggplot *to* loon

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 = TRUE,
  exteriorLabelProportion = 1/5,
  canvasHeight = 700,
  canvasWidth = 850,
  tkLabels = NULL
)

```

Arguments

<code>ggObj</code>	a ggplot or ggmatrix object
<code>...</code>	named arguments to modify loon plot states
<code>activeGeomLayers</code>	to determine which geom layer is active. Only <code>geom_point()</code> and <code>geom_histogram()</code> can be set as active geom layer(s) so far. (Notice, more than one <code>geom_point()</code> layers can be set as active layers, but only one <code>geom_histogram()</code> can be set as an active geom layer)
<code>layerId</code>	numerical; which layer to scale to
<code>scaleToFun</code>	scale to function. See zoom .
<code>ggGuides</code>	logical (default FALSE) to determine whether to draw a ggplot background or not.
<code>parent</code>	parent widget path (Tk toplevel)
<code>pack</code>	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.
<code>exteriorLabelProportion</code>	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 = $1 / \text{exteriorLabelProportion}$ for the plot size in <code>tkgrid()</code> .
<code>canvasHeight</code>	the height of canvas
<code>canvasWidth</code>	the width of canvas
<code>tkLabels</code>	Deprecated: logical (or NULL) to indicate whether the plot(s) are to be wrapped by exterior labels (title, subtitle, xlabel or ylabel) using <code>tk.grid()</code>

Value

a loon single or compound widget

Examples

```
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 #####
if(requireNamespace("GGally")) {
  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 though

Details

When "+" and "%>%" both appear in pipe operations, "%>%" takes the priority of "+", e.g:

```
mtcars %>% ggplot(aes(mpg, wt, colour = cyl)) + geom_point() %>% ggplot2looon(),
```

error would occur. The reason is

```
geom_point() %>% ggplot2looon()
```

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

```
if(require(magrittr) && interactive()) {
  ## Not run:
  # Error
  g <- mtcars %>%
    ggplot(aes(mpg, wt, colour = cyl)) +
    geom_point() %>%
    ggplot2looon()

  ## End(Not run)
  g <- mtcars %>%
    gg_pipe(
      ggplot(aes(mpg, wt, colour = cyl)) + geom_point()
    ) %>%
    ggplot2looon()
}
```

`g_getLocations`

get locations for ggmatrix

Description

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

Usage

```

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](#)

<code>g_getPlots</code>	<i>get ggplots</i>
-------------------------	--------------------

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,
  showNearestColor = FALSE
)

## Default S3 method:
g_getPlots(
  target,
  asAes = TRUE,
  selectedOnTop = TRUE,
  showNearestColor = FALSE
)

```

```
## S3 method for class 'l_pairs'
g_getPlots(
  target,
  asAes = TRUE,
  selectedOnTop = TRUE,
  showNearestColor = FALSE
)
```

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.

showNearestColor logical and default is FALSE; if TRUE, the legend of color and fill (hex code) would be converted to the R built-in color names. For some hex codes, there are no precise matching. Consequently, these colors will be converted to the R built-in color names which are the "nearest" of these hex codes.

Value

a list of ggplots.

See Also

[l_getPlots](#), [g_getLocations](#)

hover

Modify the hover component

Description

Provides a pop up display as the mouse hovers over a plot element in the interactive plot.

Usage

```
hover(itemLabel = NULL, showItemLabels = NULL)
```

Arguments

itemLabel A character vector of length n with a string to be used to pop up when the mouse hovers above that element.

showItemLabels A single logical value: TRUE if pop up labels are to appear on hover, FALSE (the default) if they are not.

Value

a ggproto object

See Also

[active](#), [linking](#), [zoom](#), [selection](#), [interactivity](#)

Examples

```
if(interactive()) {  
  
  l_ggplot(mpg, mapping = aes(x = displ, 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	The string identifying the group of linked plots that the current plot will join. Default is none.
linkingKey	The length <i>n</i> character vector of unique keys. Default will be "0", "1", ..., "n-1" where <i>n</i> is the number of elements (e.g., points) displayed.
linkedStates	The character vector of display states to be linked. These can be "color", "selected", "active", "size" and "glyph" for an 'l_plot' object and "color", "selected", "active" for an 'l_hist' object. (These roughly correspond to aesthetics in a 'ggplot'.)
sync	Either "pull" (the default) or "push" to indicate whether the values of the linked states of the plot are to be pulled from those of the other plots in the linking group, or the values are to be pushed to all other plots in the linking group. This matters only when joining an existing group of plots and the default value is typically the right thing to do.
active	a logical or a logical vector of length <i>n</i> that determines which observations are active (TRUE and hence appear in the plot) and which are inactive (FALSE and hence do not appear). Default is TRUE.
activeGeomLayers	determine which geom layer is interactive by its 'geom_...' position in the grammar of the expression. Currently, only <code>geom_point()</code> and <code>geom_histogram()</code> can be set as the active geom layer(s) so far. (N.B. more than one <code>geom_point()</code> layer can be set as an active layer, but only one <code>geom_histogram()</code> can be set as an active geom layer and it can be the only active layer.)
selected	a logical or a logical vector of length <i>n</i> that determines which observations are selected (TRUE and hence appear highlighted in the plot) and which are not. Default is FALSE and no points are highlight.
selectBy	A string determining how selection will occur in the interactive plot. Default is "sweeping" where a rectangular region is reshaped or "swept" out to select observations.; alternately "brushing" will indicate that a fixed rectangular region is moved about the display to select observations.
selectionLogic	One of "select" (the default), "deselect", and "invert". The first highlights observations as selected, the second downlights them, and the third inverts them (downlighting highlight observations and highlighting downlighted ones).
layerId	numerical; which layer to scale to
scaleToFun	scale to function. See zoom .
itemLabel	A character vector of length <i>n</i> with a string to be used to pop up when the mouse hovers above that element.
showItemLabels	A single logical value: TRUE if pop up labels are to appear on hover, FALSE (the default) if they are not.
...	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	Description	Subfunction
Linking	Linking several plots to discover the pattern of interest	linking
Selection	Highlight the subset of interest	selection
Active	Determine which points appear	active
Hover	Query in interactive graphics	hover
Zoom	Region Modification	zoom

Value

a ggproto object

Examples

```
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

```
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

`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

`layout_coords(target)`

Arguments

target an object `ggplot2loon()` returns

Value

a layout coordinate matrix

linking

*Modify the linking component***Description**

A group-key-state linking model is used to link plots in loon. This allows changes in one plot to propagate to all plots in the same linkingGroup and enables interactive features like brushing. Elements to be matched between plots are identified by linkingKey; within each plot, the key for each element (e.g., case, observation) is unique. The linkedStates identify which display states (e.g., "color") should change in concert with other plots in the linkingGroup.

Usage

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

Arguments

linkingGroup	The string identifying the group of linked plots that the current plot will join. Default is none.
linkingKey	The length n character vector of unique keys. Default will be "0", "1", ..., "n-1" where n is the number of elements (e.g., points) displayed.
linkedStates	The character vector of display states to be linked. These can be "color", "selected", "active", "size" and "glyph" for an 'l_plot' object and "color", "selected", "active" for an 'l_hist' object. (These roughly correspond to aesthetics in a 'ggplot'.)
sync	Either "pull" (the default) or "push" to indicate whether the values of the linked states of the plot are to be pulled from those of the other plots in the linking group, or the values are to be pushed to all other plots in the linking group. This matters only when joining an existing group of plots and the default value is typically the right thing to do.

Value

a ggproto object

See Also

[active](#), [selection](#), [zoom](#), [hover](#), [interactivity](#), [l_getLinkedStates](#), [l_setLinkedStates](#), [l_configure](#)

Examples

```

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

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 'zenplot'
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

```
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.l_compound

Turn a loon widget to a ggplot object

Description

Create a ggplot object from a loon widget

Usage

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

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

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

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

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

## S3 method for class 'l_layer_scatterplot'
loon2ggplot(
```

```
target,
asAes = TRUE,
selectedOnTop = TRUE,
showNearestColor = FALSE,
...
)

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

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

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

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

## Default S3 method:
loon2ggplot(
  target,
  asAes = TRUE,
  selectedOnTop = TRUE,
  showNearestColor = FALSE,
```

```

    ...
  )

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

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

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

```

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.
showNearestColor	logical and default is FALSE; if TRUE, the legend of color and fill (hex code) would be converted to the R built-in color names. For some hex codes, there are no precise matching. Consequently, these colors will be converted to the R built-in color names which are the "nearest" of these hex codes.
...	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

```
## Not run:
if(interactive()) {
  library(zenplots)
  library(patchwork)
  stopifnot(packageVersion("zenplots") > "1.0.4")
  zen <- zenplots::zenplot(iris, plot1d = "density", pkg = "loon")
  ggzen <- loon.ggplot(zen)
  ggzen +
    patchwork::plot_annotation(title = "This is a ggplot")
}

## End(Not run)
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['color'])),
      labels = c("setosa", "versicolor", "virginica")
    )

  # histogram
  lh <- l_hist(mtcars$mpg,
```

```

        color = factor(mtcars$gear))

gh0 <- loon2ggplot(lh)
# facet by `fill`
gh0 + facet_wrap(~fill)

# set `asAes = FALSE`
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

<code>l_getSubtitles</code>	<i>Return the subtitles from an <code>l_facet_ggplot</code> object</i>
-----------------------------	--

Description

Return the subtitles from an `l_facet_ggplot` object

Usage

```
l_getSubtitles(target)
```

Arguments

target	an <code>l_facet_ggplot</code> object. If the ggplot object is faceted (either by <code>facet_wrap</code> or <code>facet_grid</code>), an <code>l_facet_ggplot</code> 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

```

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	<i>Box scaling in 3D rotation</i>
----------	-----------------------------------

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 <code>as.numeric(.)</code> will be applied successfully if <code>is.numeric(.)</code> 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().
...	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

```
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)
}

```

 selection

Modify the selected component

Description

Set which elements (i.e., observations) are "selected". These will be shown as highlighted in the plot using the current "highlight" colour (see [l_userOptions](#)).

Usage

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

Arguments

selected	a logical or a logical vector of length n that determines which observations are selected (TRUE and hence appear highlighted in the plot) and which are not. Default is FALSE and no points are highlight.
selectBy	A string determining how selection will occur in the interactive plot. Default is "sweeping" where a rectangular region is reshaped or "swept" out to select observations.; alternately "brushing" will indicate that a fixed rectangular region is moved about the display to select observations.
selectionLogic	One of "select" (the default), "deselect", and "invert". The first highlights observations as selected, the second downlights them, and the third inverts them (downlighting highlight observations and highlighting downlighted ones).

Details

There are two ways to directly select elements on, for example, a scatterplot using the mouse: either by "sweeping" or by "brushing". "Sweeping" allows us to sweep out a contiguous rectangular area of the plot, while, by "brushing", a fixed rectangular area is brushes across the plot selecting all points within the rectangle.

The selection logic give users more flexibility to users to not only highlight the elements, but also to downlight, and even to invert selections (changing the highlighted to downlighted, and vice versa).

Value

a ggproto object

See Also

[active](#), [linking](#), [zoom](#), [hover](#), [interactivity](#), [l_userOptions](#)

Examples

```
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)
}
```

zoom

Zoom Plot Region

Description

Change the visible plot region by scaling to different elements of the display.

Usage

```
zoom(layerId = NULL, scaleToFun = NULL)
```

Arguments

`layerId` numerical; which layer to scale the plot by.
`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_scaletto_plot`; else if the layer is a general `l_layer` widget, the default `scaleToFun` would be `l_scaletto_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

scale to	Subfunction
plot	<code>l_scaletto_plot</code>
world	<code>l_scaletto_world</code>
active	<code>l_scaletto_active</code>
selected	<code>l_scaletto_selected</code>
layer	<code>l_scaletto_layer</code>

Users can also supply their own function, providing its arguments match those of the functions shown in the above table.

Value

a `ggproto` object

See Also

[active](#), [linking](#), [selection](#), [hover](#), [interactivity](#)

Examples

```
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_scaletto_selected)
}
```

Index

active, [2](#), [12](#), [14](#), [16](#), [30](#), [31](#)

Cartesianxy2Polarxy, [3](#)

g_getLocations, [9](#), [11](#)
g_getPlots, [10](#), [10](#)
get_activeGeomLayers, [4](#), [31](#)
get_scaledData, [5](#)
gg_pipe, [8](#)
ggplot, [25](#)
ggplot2loon, [5](#), [6](#), [18](#), [25](#)

hover, [3](#), [11](#), [14](#), [16](#), [30](#), [31](#)

interactivity, [3](#), [12](#), [12](#), [16](#), [30](#), [31](#)
is.CoordPolar, [14](#)
is.l_ggplot, [15](#)

l_configure, [16](#)
l_getLinkedStates, [16](#)
l_getLocations, [10](#)
l_getPlots, [11](#)
l_getSubtitles, [24](#)
l_ggplot, [18](#), [25](#)
l_info_states, [13](#)
l_scale3D, [27](#)
l_scaleto_active, [31](#)
l_scaleto_layer, [31](#)
l_scaleto_plot, [31](#)
l_scaleto_selected, [31](#)
l_scaleto_world, [31](#)
l_setLinkedStates, [16](#)
l_userOptions, [29](#), [30](#)
layout_coords, [15](#)
linking, [3](#), [12](#), [14](#), [16](#), [30](#), [31](#)
loon.ggplot, [17](#), [25](#)
loon2ggplot, [18](#)
loon2ggplot (loon2ggplot.l_compound), [18](#)
loon2ggplot.l_compound, [18](#)
loonLayer, [23](#)

print.l_ggplot, [25](#), [26](#)

scale_multi, [28](#)
scaleBox, [27](#)
selection, [3](#), [12](#), [14](#), [16](#), [29](#), [31](#)

zoom, [3](#), [7](#), [12–14](#), [16](#), [30](#), [30](#)