Package: ggplotFL (via r-universe)

November 23, 2026

Title Using ggplot2 in FLR

Version 2.7.3

Description Using ggplot2 for FLR. Provides (1) overloaded ggplot methods for various FLR classes, (2) ggplot-based versions of standard plots in the FLCore package, and (3) new geoms for using FLR objects.

X-schema.org-keywords visualization, ggplot2, fisheries, flr, R

License GPL-2

Depends R(>= 4.0), methods, ggplot2(>= 2.4.0), FLCore(>= 2.6.15)

Imports gridExtra, ggrepel, patchwork, rlang, stats, scales, cowplot, data.table

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

URL http://flr-project.org/ggplotFL

BugReports https://github.com/flr/ggplotFL/issues

Additional_repositories http://flr-project.org/R

Collate 'allGenerics.R' 'palette.R' 'geom.R' 'functions.R' 'ggplot.R' 'plot.R' 'special.R'

LazyLoad Yes

Encoding UTF-8

RoxygenNote 7.3.3

Repository https://ices-tools-dev.r-universe.dev

Date/Publication 2026-10-21 16:42:25 UTC

RemoteUrl https://github.com/flr/ggplotFL

RemoteRef HEAD

RemoteSha ca780d8e0dae205dada8b7643b315f418e7fde52

Contents

cohcorrplot 2
eqlabel 3
flpalette 4
geom_flpar 5
geom_flquantiles 7
geom_worm 10
ggplot,FLQuant-method 12
human_numbers 13
integer_breaks 14
label_flqs 15
plot,FLQuant,missing-method 16
plotKobe 21
plotRunstest 22
plotXval 23
pubpng 24
scale_fill_flr 24

Index **26**

`cohcorrplot` *cohcorrplot*

Description

A correlation plot that show and quantifies correlation along cohorts. Typically used on catch or survey abundances-at-age.

Usage

```
cohcorrplot(x, ...)

## S4 method for signature 'FLQuant'
cohcorrplot(x, ...)

## S4 method for signature 'FLCohort'
cohcorrplot(x, diag_size = 16, lower_size = 6)
```

Arguments

`x`	An object with the abundance at age information. FLQuant or FLCohort.
`...`	Any extra arguments
`diag_size`	Font size for labels in diagonal row
`lower_size`	Font size for labels in lower triangle

Details

The method prints a plot assembled as a combination of grid elements, but reurns it as a *gg* object.

Author(s)

The FLR Team

Examples

```
data(ple4)
cohcorrplot(catch.n(ple4))
cohcorrplot(FLCohort(stock.n(ple4)))
```

---

eqlabel *Functions create labels for FLSR models*

Description

These functions create labels from FLSR models and params to be used, for example, on the legend of the ggplot-based plot method for FLSRs

Usage

```
eqlabel(model, param)

modlabel(model, param)
```

Arguments

model	a list of model formulas
param	a list of FLPar objects for the params slot

Examples

```
data(nsher)
srs <- FLSRs(sapply(c('ricker', 'bevholt'), function(x) {
  y <- nsher
  model(y) <- x
  return(fmle(y))
}))
eqlabel(model=lapply(srs, model),
  param=lapply(srs, params))
modlabel(model=lapply(srs, model),
  param=lapply(srs, params))
```

`flpalette` *A high-contrast palette to use in ggplotFL*

Description

Plot methods defined in the ggplotFL package make use by default of a palette with a high contrast, useful to separate time series or categories. The palette consist of seven colours: red, blue, green, violet, orange, yellow and brown. This palette can be inspected at the [colorbrewe2.org site](http://colorbrewer2.org/?type=qualitative&scheme=Set1&n=7#type=qualitative&scheme=Set1&n=7).

Usage

```
flpalette

flpalette_colours(n = length(flpalette))

flpalette_grads(palette = flpalette, reverse = FALSE, ...)

scale_colour_flr(palette = flpalette, discrete = TRUE, reverse = FALSE, ...)
```

Arguments

`n`	Number of colours or individual colours to return, or number of colours to interpolate.
`palette`	Palette subset to create a gradual scale from, defaults to *flpalette*.
`reverse`	Should the palette be reversed, FALSE.
`...`	Elements to subset from palette, by name or position.
`discrete`	Is the palette to be applied to a discrete variable, TRUE.

Format

An object of class `character` of length 8.

Details

The palette is accessible as a named vector, *flpalette*. Two functions are also available to manipualte the palette. One to extract a subset of the palette, *flpalette_colours*, and another to create a gradation of colours between two or more of the palette colours, *flpalette_grads*.

Value

A named vector of colors and HEX codes, or a function to obtain a gradient of colors fo a given length.

Author(s)

The FLR Team

See Also

FLComp

Examples

```
# CHECK flpalette
 flpalette
 scales::show_col(flpalette)
flpalette_colours()
flpalette_colours(5)
flpalette_colours(2:3)
flpalette_grads(flpalette_colours(3))(20)
```

`geom_flpar` *Horizontal lines for FLPar objects*

Description

This ‘geom‘ shows a horizontal line for each ‘param‘ in a ‘FLPar‘ object and labels it using the ‘param‘ dimnames, by calling ‘geom_hline‘ and ‘ggrepel::geom_label_repel‘. Separate labels and lines can be specified for different facets by providing the ‘data‘ argument with an object of class ‘FLPars‘ in which each element is named as the values of the facetting variable.

Usage

```
geom_flpar(mapping = NULL, data, ..., x, na.rm = FALSE)
```

Arguments

`mapping`	Set of aesthetic mappings created by `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)`).
`...`	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`x` Position for params labels on the x axis

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

Aesthetics

'geom_flpar' understands the following aesthetics (required aesthetics are in bold). Some aesthetics apply to only one of the two elements, in parenthesis: - '*x*' - 'y', defaults to 90 - 'yintercept', defaults to params value (line). - 'label', defaults to params names. - 'alpha' - 'colour' - 'linetype' (line) - 'linewidth' (line) - 'fill' (label) - 'angle' (label) - 'family' (label)

Examples

```
data(ple4)
plot(ssb(ple4)) + geom_flpar(data=FLPar(Blim=300000), x=1960)
plot(ssb(ple4)) + geom_flpar(data=FLPar(Blim=300000), x=2015)
plot(ssb(ple4)) + geom_flpar(data=FLPar(Blim=300000, Bpa=230000), x=1960)
# geom works for multiple facets, separate params using name-matching FLPars()
plot(ple4, metrics=list(SSB=ssb, F=fbar)) +
  geom_flpar(data=FLPars(SSB=FLPar(Blim=300000, Bpa=230000),
  F=FLPar(FMSY=0.21)), x=c(1964))
# x and y positions can be altered by param
plot(ple4, metrics=list(SSB=ssb, F=fbar)) +
  geom_flpar(data=FLPars(SSB=FLPar(Blim=300000, Bpa=230000),
  F=FLPar(FMSY=0.21)), x=c(2015, 2015, 1960), y=c(340000, 180000, 0.18))
```

geom_flquantiles *Sampling quantiles*

Description

This 'geom' calculates sampling quantiles and draws a ribbon for the quantile range plus a line for the median (50% quantile).

Usage

```
geom_flquantiles(
  mapping = NULL,
  data = NULL,
  stat = "FLQuantiles",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  na.rm = FALSE,
  probs = c(0.1, 0.5, 0.9),
  alpha = 0.5,
  ...
)

stat_flquantiles(
  mapping = NULL,
  data = NULL,
  geom = "line",
  position = "identity",
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by 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. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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. annotation_borders().

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

probs — Quantiles to compute and draw, defaults to c(0.10, 0.90).

alpha — Transparency for quantile ribbon.

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`geom` The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following:

- A `Geom` ggproto subclass, for example `GeomPoint`.
- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

Details

As this ‘geom‘ outputs two layers, although based on different ‘geoms‘, interactions between common parameters need to be considered. The ‘fill‘ parameter will only affect the quantile range ‘ribbon‘, but ‘colour‘ will be passed to both the ‘ribbon‘ and median ‘line‘ layers. The defaults are no lines on the quantiles and "black" for the median line. The ‘alpha‘ value has been hard coded to 1 for the median line, so only affects the quantile ‘ribbon‘. To change this, call ‘stat_flquantiles‘ directly, as in the examples below.

‘stat_flquantiles‘ will return between one and three ‘y‘ values depending on the number of quantiles requested. If two quantiles are to be calculated, it will return the corresponding ‘ymin‘ and ‘ymax‘, to be used with, for example, ‘geom_ribbon‘. If only one quantile is to be calculated, it will be returned as ‘y‘, to be used typically by ‘geom_line‘. Finally, if three values are passed in the ‘probs‘ argument, all of the above will be returned, in the right order.

Aesthetics

‘geom_flquantiles‘ understands the following aesthetics (required aesthetics are in bold): - ‘*x*‘ - ‘*y*‘ - ‘alpha‘ - ‘colour‘ - ‘fill‘ - ‘group‘ - ‘linetype‘ - ‘linewidth‘ where some of them apply to the ribbons and some of them to the lines.

Computed variables

y quantile, if only one requested or central one when if three

ymin lower quantile, if two or three requested

ymax upper quantile, if two or three requested

Examples

```
data(ple4)
flq <- rnorm(250, catch(ple4), 200000)
ggplot(flq, aes(x=date, y=data)) +
  geom_flquantiles(probs=c(0.25, 0.50, 0.75), fill="red", alpha=0.25)
# Draw two quantiles with two calls to geom_flquantiles
ggplot(flq, aes(x=date, y=data)) +
  geom_flquantiles(probs=c(0.25, 0.50, 0.75), alpha=0.25, fill="red") +
  geom_flquantiles(probs=c(0.10, 0.90), alpha=0.15, fill="red")
# Use it on an FLQuants, colouring by their name
flqs <- FLQuants(A=rnorm(250, catch(ple4), 200000),
  B=rnorm(250, stock(ple4), 200000))
ggplot(flqs, aes(x=date, y=data, colour=qname)) +
  geom_flquantiles(probs=c(0.10, 0.50, 0.90), aes(fill=qname), alpha=c(0.30))
# Or facet them
ggplot(flqs, aes(x=date, y=data)) +
  geom_flquantiles(probs=c(0.10, 0.50, 0.90), fill="red", alpha=c(0.30)) +
  facet_grid(qname~.)
# For greater control, call stat_flquantiles directly with a geom
ggplot(flq, aes(x=year, y=data)) +
 stat_flquantiles(probs=c(0.10, 0.90), geom = "ribbon",
   fill="yellowgreen", alpha=0.30) +
 stat_flquantiles(probs=c(0.01), geom = "line",
   colour = "green4", linetype=3) +
 stat_flquantiles(probs=c(0.99), geom = "line",
   colour = "green4", linetype=3) +
 stat_flquantiles(probs=c(0.25, 0.75), geom = "ribbon",
   fill="green4", alpha=0.30) +
 stat_flquantiles(probs=c(0.50), geom = "line", linewidth=1.5,
   colour = "lightgreen") +
 stat_flquantiles(probs=c(0.50), geom = "line",
   colour = "darkgreen")
ggplot(flq, aes(x=year, y=data)) +
 stat_flquantiles(probs=c(0.50), geom = "line",
   colour = "darkgreen")
plot(flq, probs=0.50)
```

`geom_worm` *A geom for adding worms to probability intervals from geom_flquantiles*

Description

A geom for adding worms to probability intervals from geom_flquantiles

Usage

```
geom_worm(
  data,
```

```
  mapping = aes(colour = iter),
  ...,
  stat = "identity",
  position = "identity",
  na.rm = FALSE
)
```

Arguments

data	Subset of data, select from full object using iter().
mapping	Set of aesthetic mappings created by 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.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept. • The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

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

Aesthetics

'geom_worm' understands the following aesthetics (required aesthetics are in bold): - 'colour' - 'linetype' - 'linewidth'

Examples

```
data(ple4)
x <- rlnorm(200, log(catch(ple4)), 0.3)
plot(x) + geom_worm(data=iter(x, 1:4))

x <- FLQuants(C=rlnorm(200, log(catch(ple4)), 0.3),
  F=rlnorm(200, fbar(ple4), 0.2))
plot(x) + geom_worm(data=iter(x, 1:4))
```

---

ggplot,FLQuant-method *ggplot method for various FLR classes*

Description

The `ggplot()` method has been conveniently overloaded for various FLR classes. A call to `as.data.frame` takes place on `data` before passing all arguments to the original `ggplot` function.

Usage

```
## S4 method for signature 'FLQuant'
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())

## S4 method for signature 'FLQuants'
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())

## S4 method for signature 'FLComp'
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

```
## S4 method for signature 'FLComps'
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())

## S4 method for signature 'FLPar'
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

Arguments

data	An FLQuant object
mapping	An aesthetic mapping, from a call to aes
...	Other arguments to be passod on to ggplot
environment	Where to look for an undefined plot variable.

Details

Please look at the relevant as.data.frame method for each class to understand the naming conventions used in the resulting data.frame

See Also

ggplot, https://github.com/flr/ggplotFL/

Examples

```
dat <- rnorm(1, FLQuant(1, dim=c(5,10)), 0.5)
ggplot(data=dat, aes(data, year)) + geom_point()
data(ple4)
dat <- FLQuants(catch=catch(ple4), ssb=ssb(ple4))
ggplot(data=dat, aes(data, year)) + geom_point() + facet_wrap(~qname)
```

---

human_numbers *A ggplot2 number formatter*

Description

This function formats numbers for output in a 'human' way.

Usage

```
human_numbers(x = NULL, smbl = "", signif = 1)
```

Arguments

x	An object of class numeric
smbl	A character or symbol to be added prior to the number, e.g. an euro sign.
signif	Number of significant figures

Value

A formatted character vector

Author(s)

Iago Mosqueira (EC JRC)

See Also

labeller

`integer_breaks` *Shows integer values in a programmatic and scalable fashion.*

Description

This function provides sensible breaks for integers

Usage

```
integer_breaks(n = 5, ...)
```

Arguments

`n`	Number of breaks
`...`	Arguments to be passed to *pretty*

Value

A function to be called

Author(s)

Iago Mosqueira (WMR)

See Also

labeller pretty

label_flqs *A ggplot2 labeller for FLQuants*

Description

Plots of FLQuants objects use by default facet_grid to separate the different elements. This function generates facet labels that have both the element name and the units of measurement of each, as stored in the 'units' slot of each FLQuant.

Usage

```
label_flqs(x, drop = c("NA", "NC", "m", "f", "z", "prop"))
```

Arguments

x	An object of class FLQuants
drop	Character string to be dropped from the label shen found in the *units* slot. Defaults to c("NA", "NC", "m", "z", "prop")

Details

Certain *units* are dropped from the label, as being uninformative: "NA", "NC", "m", "z", and "prop". This can be selected with the *drop* argument.

Value

A ggplot2 labeller function

Author(s)

Iago Mosqueira (EC JRC)

See Also

labeller

plot,FLQuant,missing-method

ggplot versions of FLR class plot() methods

Description

New basic plot methods for some FLR classes are defined in ggplotFL.

Usage

```
## S4 method for signature 'FLQuant,missing'
plot(x, probs = c(0.05, 0.25, 0.5, 0.75, 0.95), na.rm = FALSE, ...)

## S4 method for signature 'FLQuant,FLQuant'
plot(
  x,
  y,
  ...,
  probs = c(0.05, 0.25, 0.5, 0.75, 0.95),
  na.rm = FALSE,
  iter = NULL
)

## S4 method for signature 'FLQuants,missing'
plot(
  x,
  probs = c(0.05, 0.25, 0.5, 0.75, 0.95),
  na.rm = FALSE,
  worms = iter,
  iter = NULL
)

## S4 method for signature 'FLQuants,FLPar'
plot(x, y, ...)

## S4 method for signature 'FLQuants,FLPars'
plot(x, y, ...)

## S4 method for signature 'FLQuantPoint,missing'
plot(x, mean = TRUE, median = TRUE)

## S4 method for signature 'FLQuantPoint,FLQuant'
plot(x, y, na.rm = FALSE, ...)

## S4 method for signature 'FLQuantPoint,FLQuants'
plot(x, y, na.rm = FALSE, mean = TRUE, median = TRUE, ...)
```

```
## S4 method for signature 'FLPar,missing'
plot(x, names = NULL)

## S4 method for signature 'FLStock,missing'
plot(
  x,
  metrics = list(Rec = rec, SSB = ssb, Catch = catch, F = fbar),
  na.rm = TRUE,
  ...
)

## S4 method for signature 'FLStock,FLStock'
plot(
  x,
  y,
  metrics = list(Rec = rec, SSB = ssb, Catch = catch, F = fbar),
  probs = c(0.1, 0.33, 0.5, 0.66, 0.9),
  na.rm = TRUE,
  iter = NULL,
  ...
)

## S4 method for signature 'FLStock,FLPar'
plot(x, y, metrics = list(Rec = rec, SSB = ssb, Catch = catch, F = fbar), ...)

## S4 method for signature 'FLStocks,missing'
plot(
  x,
 metrics = list(Rec = function(x) unitSums(rec(x)), SB = function(x) unitSums(ssb(x)), C
    = function(x) unitSums(catch(x)), F = function(x) unitMeans(fbar(x))),
  probs = c(0.1, 0.33, 0.5, 0.66, 0.9),
  alpha = c(0.1, 0.4),
  worm = iter,
  iter = NULL,
  ...
)

## S4 method for signature 'FLStocks,missing'
plot(
  x,
 metrics = list(Rec = function(x) unitSums(rec(x)), SB = function(x) unitSums(ssb(x)), C
    = function(x) unitSums(catch(x)), F = function(x) unitMeans(fbar(x))),
  probs = c(0.1, 0.33, 0.5, 0.66, 0.9),
  alpha = c(0.1, 0.4),
  worm = iter,
  iter = NULL,
  ...
)
```

```
## S4 method for signature 'FLStocks,FLPar'
plot(
  x,
  y,
  na.rm = TRUE,
 metrics = function(x, y) FLQuants(SSB = ssb(x)/y[, "ssb", ], F = fbar(x)/y[, "harvest",
    ], Catch = catch(x))
)

## S4 method for signature 'FLStock,FLStocks'
plot(x, y, ...)

## S4 method for signature 'FLSR,missing'
plot(x, y, ...)

## S4 method for signature 'FLSRs,ANY'
plot(x, legend_label = names(x), facets = FALSE, ...)

## S4 method for signature 'FLBiol,missing'
plot(x, metrics = list(Rec = function(x) n(x)[1, ], B = tsb), ...)

## S4 method for signature 'FLBiols,missing'
plot(x, metrics = list(Rec = function(x) n(x)[1, ], B = tsb), ...)

## S4 method for signature 'FLIndexBiomass,missing'
plot(x, y, ...)

## S4 method for signature 'FLIndex,missing'
plot(x)

## S4 method for signature 'FLIndices,missing'
plot(x)
```

Arguments

x	FLR object to plot
probs	Quantiles to calculate along the iter dimension. A vector of length 5, for the lower outer, lower inner, central, upper inner and upper outer quantiles. Defaults to the 66 and 80 percent quantiles, plus median line.
na.rm	Should NAs be deleted in quantile calculations?, defaults to TRUE.
...	Other arguments to be passed to the corresponding ggplot call.
y	FLR object to plot
iter	Individual iterations to show as worm plots over the quantiles.
metrics	function returning an FLQuants for each FLStock
alpha	alpha values for the quantile ribbons, defaults to 0.10 and 0.40.

worm	Individual iterations to show as worm plots over the quantiles.
legend_label	function to create the legend labels

Details

The coertion to *data.frame* that is carried out in the plot methods sets the argument ‘date=TRUE‘. This generates a new column of class ‘POSIXct‘ for the first day of the first month of each season. If the ‘season‘ dimension of the object being plotted is of length greater than one, ‘date‘ will be used as variable on the x axis of the plot. Otherwise, it will be ‘year‘. Keep this in mind when adding extra elements to the plot (see examples below).

A similar mechanism is used for the *y* axis, depending on the length of the ‘iter‘ dimension. For objects with no *iters*, a single line is plotted for each *FLQuant*, and the *y* axis is mapped to the ‘data‘ column of the *data.frame*. For objects with iterations, i.e. with length greater than 1 on the ‘iter‘ dimension, the default plots show the quantiles of the distribution and the *y* axis is mapped to the middle quantile, by default ‘50 examples below on how to refer to these variables when adding elements to the plot.

See Also

ISOdate ggplot

Examples

```
 # Plot a single FLQuant
 data(ple4)
 plot(catch.n(ple4))

 # Plot an FLQuant with iters, shows quantiles
 flq <- rnorm(100, catch(ple4), 60000)
 plot(flq)

 # Specify quantiles, default is c(0.10, 0.33, 0.50, 0.66, 0.90)
 plot(flq, probs=c(0.05, 0.25, 0.50, 0.75, 0.95))

 # Adding extra elements to an FLQuant plot, with seasons
 flq <- FLQuant(runif(200), dim=c(1,15,1,4))
 plot(flq) + geom_point(aes(x=date, y=data, colour=season))

# or without them
 flq <- FLQuant(runif(200), dim=c(1,15))
 plot(flq) + geom_point(aes(x=year, y=data))

# For an object with iter
 flq <- rlnorm(100, flq, 0.4)
 plot(flq) + geom_point(aes(x=year, y=data))

# To plot(FLQuant) as in previous versions of ggplotFL
plot(rnorm(300, catch(ple4), catch(ple4)/2), probs=c(0.10, 0.5, 0.90)) +
  geom_flquantiles(probs=c(0.01), linetype=3, colour="red", alpha=0.1) +
  geom_flquantiles(probs=c(0.99), linetype=3, colour="red", alpha=0.1)
# plot(FLQuant, FLQuant, ...) to place in one facet
```

```
plot(catch(ple4), landings(ple4))
# Add legend by hand
plot(rnorm(200, landings(ple4), 8000), discards(ple4)) +
  scale_colour_discrete(name="Yield (t)", labels=c("Landings", "Discards")) +
  theme(legend.position="bottom")
 # Plot an FLQuants created from ple4 FLStock
 data(ple4)
 plot(FLQuants(SSB=ssb(ple4), rec=rec(ple4)))
 plot(FLQuants(SSB=ssb(ple4), rec=rec(ple4)), probs = NULL)
 # plot for FLQuants, FLPar
 data(ple4)
 rps <- FLPar(F=0.14, Catch=1.29e5, Rec=9.38e5, SSB=1.8e5)
 fqs <- metrics(ple4)
 plot(fqs, rps)
 # Works also if reptsa are given for some panels
 rps <- FLPar(F=0.14, Catch=1.29e5, SSB=1.8e5)
 plot(fqs, rps)
 # plot for FLQuants, FLPars
 data(ple4)
 rps <- FLPars(F=FLPar(Fmsy=0.14, Fpa=0.35), SSB=FLPar(SBmsy=1.8e5, SBlim=1.1e5))
 fqs <- metrics(ple4, list(SSB=ssb, F=fbar))
 plot(fqs, rps) + ylim(c(0, NA))
# plot for FLQuantPoint
fqp <- FLQuantPoint(rlnorm(300, log(catch(ple4)), 0.20))
plot(fqp)
# plot for FLQuantPoint, FLQuant
plot(fqp, rlnorm(3, log(catch(ple4)), 0.20))
# plot for FLQuantPoint, FLQuants
fqp <- FLQuantPoint(rlnorm(300, log(catch(ple4)), 0.20))
fqs <- divide(rlnorm(3, log(catch(ple4)), 0.20))
plot(fqp, fqs)
par <- FLPar(alpha=rnorm(200, 0.6, 0.2), beta=rlnorm(200, 0.8, 0.3))
plot(par)
# plot of an FLStock
 data(ple4)
 plot(ple4)
# plot for FLStock, FLPar
data(ple4)
rps <- FLPar(F=0.14, Catch=1.29e5, Rec=9.38e5, SSB=1.8e5)
plot(ple4, rps)
# plot for FLStocks
data(ple4)
pls <- FLStocks(runA=ple4, runB=qapply(ple4, function(x) x*1.10))
plot(pls)
# geom_flpar can be used draw refpts lines and labels
plot(pls, metrics=list(SSB=ssb, F=fbar)) +
  facet_grid(qname~stock, scales='free') +
  geom_flpar(data=FLPars(SSB=FLPar(Blim=300000, Bpa=230000),
  F=FLPar(FMSY=0.21)), x=c(1960), stock='runA', fill=alpha('white', 0.4))
# plot for FLStocks
data(ple4)
pls <- FLStocks(runA=ple4, runB=qapply(ple4, function(x) x*1.10))
plot(pls)
```

```
# geom_flpar can then be used draw refpts lines and labels
plot(pls, metrics=list(SSB=ssb, F=fbar)) +
  facet_grid(qname~stock, scales='free') +
  geom_flpar(data=FLPars(SSB=FLPar(Blim=300000, Bpa=230000),
  F=FLPar(FMSY=0.21)), x=c(1960), stock='runA', fill=alpha('white', 0.4))
# plot for FLSR
 data(nsher)
 plot(nsher)
 # plot for FLSRs
 data(nsher)
 srs <- FLSRs(sapply(c('segreg', 'bevholt'), function(x) {
   y <- nsher
   model(y) <- x
   return(fmle(y))
 }))
 plot(srs, facets=TRUE)
 plot(srs, legend_label=eqlabel)
 plot(srs, legend_label=modlabel)
 # Plot a FLIndex object
 data(ple4.index)
 plot(ple4.index)
 # Plot a FLIndices object
 data(ple4.indices)
 plot(ple4.indices)
 plot(ple4.indices) +
   geom_smooth(formula=y ~ x, se=FALSE, method="loess", linewidth=0.2)
```

---

plotKobe *Kobe plot of biomass and fishing mortality over MSY*

Description

'Kobe' plot is the common name given to a phase plot that show the time series and current status of stock status using two metrics: one for stock status on the x-axis, generally SB/SBMSY, and one on exploitation level on the y-axis, F/FMSY. The plot is divided in four quadrants, each representing a different combination of stock status and exploitation level. The quadrants are usually coloured green (healthy stock and exploitation), yellow (healthy stock, high exploitation), orange (overfished stock, healthy exploitation) and red (overfished stock, high exploitation).

Usage

```
plotKobe(x, refpts, X = SB/SB[MSY] ~ ssb/SBMSY, Y = F/F[MSY] ~ fbar/FMSY)
```

Arguments

x	An object of class FLStock
refpts	Reference points, class FLPar or list.
X	Formula for x axis metric, defaults to SB/SB[MSY] ~ ssb/SBMSY.
Y	Formula for y axis metric, defaults to F/F[MSY] ~ fbar/FMSY.

Details

The function allows the user to define the metrics to be used on each axis, by using a formula interface. For each axis, a formula specifies the name of the metric and its computation, using available slots and metrics for the FLStock object, as well as reference points provided in the 'refpts' argument. See examples below on the usage of the formula interface.

Value

A ggplot object

Author(s)

Iago Mosqueira (WMR)

See Also

[FLStock] [FLPar]

Examples

```
data(ple4)
# Plot with default MSY refpts
plotKobe(ple4, refpts=FLPar(FMSY=0.28, SBMSY=555000))
# Plot with custom metric on x axis
plotKobe(ple4, refpts=FLPar(FMSY=0.28, SB0=8e5), X=SB/SB[0]~ssb/SB0)
```

---

plotRunstest *Plot the runs test result for one or more time series*

Description

Plot the runs test result for one or more time series

Usage

```
plotRunstest(fit, obs, ...)

## S4 method for signature 'FLQuants,missing'
plotRunstest(fit, combine = TRUE)

## S4 method for signature 'FLQuants,FLQuants'
plotRunstest(fit, obs, combine = TRUE)

## S4 method for signature 'FLQuant,FLQuant'
plotRunstest(fit, obs, combine = TRUE)
```

Arguments

fit	The result of a model fit.
obs	The observations used in the fit.
...	Extra arguments.
combine	Should ages be combined by addition, defaults to TRUE.

Value

An object of class ggplot2::gg

Examples

```
data(nsher)
plotRunstest(fitted(nsher), rec(nsher))
```

---

plotXval *Plot of FLIndices cross-validation by retrospective hindcast*

Description

Plot of FLIndices cross-validation by retrospective hindcast

Usage

```
plotXval(x, y = "missing", order = "inverse")
```

Arguments

x	An *FLIndices* object of the original observations.
y	A list contaning *FLIndices* objects returned by *a4ahcxval*.
order	Order in which retrospective runs are stored, defaults to"inverse".

Value

A ggplot object

Examples

```
# SEE vignette
```

pubpng *Output ggplot object to PNG files with good quality settings.*

Description

Output ggplot object to PNG files with good quality settings.

Usage

```
pubpng(file, plot, width = 1600, height = 1400, res = 200)
```

Arguments

file	Output file path and name.
plot	ggplot object or command.
width	Width of plot, in pixels. Defaults to 1600.
height	Height of plot, in pixels. Defaults to 1400.
res	Resolution, in ppi. Deagults to 200.

Value

TRUE if succesful, while file is saved to disk.

See Also

grDevices::png

scale_fill_flr *High contrast discrete and continuous palettes*

Description

Discrete and continuous versions of the standard ggplotFL palette. See flpalette fo further details.

Usage

```
scale_fill_flr(palette = flpalette, discrete = TRUE, reverse = FALSE, ...)
```

Arguments

palette	Palette subset to create a gradual scale from, defaults to *flpalette*.
discrete	Is the palette to be applied to a discrete variable, TRUE.
reverse	Should the palette be reversed, FALSE.
...	Other arguments to be passed to the scale functions

Value

A function.

Author(s)

The FLR Team

See Also

flpalette

Index

∗ **color**
    flpalette, 4
    scale_fill_flr, 24
∗ **dplot**
    human_numbers, 13
    integer_breaks, 14
    label_flqs, 15
∗ **hplot**
    plotKobe, 21
∗ **methods**
    cohcorrplot, 2

aes, *13*
aes(), *5*, *7*, *11*
annotation_borders(), *8*
as.data.frame, *12*

cohcorrplot, 2
cohcorrplot,FLCohort-method
        *(*cohcorrplot*)*, 2
cohcorrplot,FLQuant-method
        *(*cohcorrplot*)*, 2

eqlabel, 3

facet_grid, *15*
FLComp, *5*
flpalette, 4, *24*, *25*
flpalette_colours *(*flpalette*)*, 4
flpalette_grads *(*flpalette*)*, 4
FLQuant, *15*
FLQuants, *15*
fortify(), *5*, *7*

geom_flpar, 5
geom_flquantiles, 7
geom_worm, 10
ggplot, *12*, *13*, *19*
ggplot(), *5*, *7*
ggplot,FLComp-method
        *(*ggplot,FLQuant-method*)*, 12
ggplot,FLComps-method
        *(*ggplot,FLQuant-method*)*, 12
ggplot,FLPar-method
        *(*ggplot,FLQuant-method*)*, 12
ggplot,FLQuant-method, 12
ggplot,FLQuants-method
        *(*ggplot,FLQuant-method*)*, 12

human_numbers, 13

integer_breaks, 14
ISOdate, *19*

key glyphs, *6*, *9*, *11*

label_flqs, 15
labeller, *14*, *15*
layer geom, *9*
layer position, *8*, *12*
layer stat, *8*, *11*
layer(), *5*, *6*, *8*, *9*, *11*

modlabel *(*eqlabel*)*, 3

plot,FLBiol,missing-method
        *(*plot,FLQuant,missing-method*)*,
        16
plot,FLBiols,missing-method
        *(*plot,FLQuant,missing-method*)*,
        16
plot,FLIndex,missing-method
        *(*plot,FLQuant,missing-method*)*,
        16
plot,FLIndexBiomass,missing-method
        *(*plot,FLQuant,missing-method*)*,
        16
plot,FLIndices,missing-method
        *(*plot,FLQuant,missing-method*)*,
        16

plot,FLPar,missing-method
(plot,FLQuant,missing-method), 16
plot,FLQuant,FLQuant-method
(plot,FLQuant,missing-method), 16
plot,FLQuant,missing-method, 16
plot,FLQuantPoint,FLQuant-method
(plot,FLQuant,missing-method), 16
plot,FLQuantPoint,FLQuants-method
(plot,FLQuant,missing-method), 16
plot,FLQuantPoint,missing-method
(plot,FLQuant,missing-method), 16
plot,FLQuants,FLPar-method
(plot,FLQuant,missing-method), 16
plot,FLQuants,FLPars-method
(plot,FLQuant,missing-method), 16
plot,FLQuants,missing-method
(plot,FLQuant,missing-method), 16
plot,FLSR,missing-method
(plot,FLQuant,missing-method), 16
plot,FLSRs,ANY-method
(plot,FLQuant,missing-method), 16
plot,FLSRs,missing-method
(plot,FLQuant,missing-method), 16
plot,FLStock,FLPar-method
(plot,FLQuant,missing-method), 16
plot,FLStock,FLStock-method
(plot,FLQuant,missing-method), 16
plot,FLStock,FLStocks,missing-method
(plot,FLQuant,missing-method), 16
plot,FLStock,FLStocks-method
(plot,FLQuant,missing-method), 16
plot,FLStock,missing-method
(plot,FLQuant,missing-method), 16
plot,FLStocks,FLPar-method
(plot,FLQuant,missing-method), 16
plot,FLStocks,missing-method
(plot,FLQuant,missing-method), 16
plotKobe, 21
plotRunstest, 22
plotRunstest,FLQuant,FLQuant-method
(plotRunstest), 22
plotRunstest,FLQuants,FLQuants-method
(plotRunstest), 22
plotRunstest,FLQuants,missing-method
(plotRunstest), 22
plotXval, 23
pretty, *14*
pubpng, 24

scale_colour_flr *(*flpalette*)*, 4
scale_fill_flr, 24
stat_flquantiles *(*geom_flquantiles*)*, 7