Package 'ggmosaic'

February 9, 2017

Title Mosaic Plots in the 'ggplot2' Framework

Version 0.1.2

Description Mosaic plots in the 'ggplot2' framework. Mosaic plot functionality is provided in a single 'ggplot2' layer by calling the geom 'mosaic'.

Depends R (>= 3.2.0), ggplot2 (>= 2.2.0), productplots (>= 0.1.1)

Imports plotly (>= 4.5.5), dplyr, purrr, tidyr, gridExtra, NHANES

License GPL (>= 2)

URL http://github.com/haleyjeppson/ggmosaic

BugReports https://github.com/haleyjeppson/ggmosaic

LazyData TRUE

RoxygenNote 5.0.1.9000

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Haley Jeppson [aut, cre],
Heike Hofmann [aut],
Di Cook [aut],
Hadley Wickham [ctb]

Maintainer Haley Jeppson <hjeppson@iastate.edu>

Repository CRAN

Date/Publication 2017-02-09 00:47:04

R topics documented:

`ddecker` *Template for a double decker plot. A double decker plot is composed of a sequence of spines in the same direction, with the final spine in the opposite direction.*

Description

Template for a double decker plot. A double decker plot is composed of a sequence of spines in the same direction, with the final spine in the opposite direction.

Usage

```
ddecker(direction = "h")
```

Arguments

`direction`	direction of first split

`geom_mosaic` *Mosaic plots.*

Description

A mosaic plot is a convenient graphical summary of the conditional ditributions in a contingency table and is composed of spines in alternating directions.

Usage

```
geom_mosaic(mapping = NULL, data = NULL, stat = "mosaic",
  position = "identity", na.rm = FALSE, divider = mosaic(),
  offset = 0.01, show.legend = NA, inherit.aes = TRUE, ...)

stat_mosaic(mapping = NULL, data = NULL, geom = "mosaic",
  position = "identity", na.rm = TRUE, divider = mosaic(),
  show.legend = NA, inherit.aes = TRUE, offset = 0.01, ...)
```

Arguments

mapping	Set of aesthetic mappings created by aes or aes_. If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
divider	Divider function. The default divider function is mosaic() which will use spines in alternating directions. The four options for partioning: • vspine Vertical spine partition: width constant, height varies. • hspine Horizontal spine partition: height constant, width varies. • vbar Vertical bar partition: height constant, width varies. • hbar Horizontal bar partition: width constant, height varies.
offset	Set the space between the first spine
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.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = 'red' or size = 3. They may also be parameters to the paired geom/stat.
geom	The geometric object to use display the data

Computed variables

xmin location of bottom left corner

xmax location of bottom right corner

ymin location of top left corner

ymax location of top right corner

Examples

```
data(Titanic)
titanic <- as.data.frame(Titanic)
titanic$Survived <- factor(titanic$Survived, levels=c("Yes", "No"))


ggplot(data=titanic) +
  geom_mosaic(aes(weight=Freq, x=product(Class), fill=Survived))
# good practice: use the 'dependent' variable (or most important variable)
# as fill variable
ggplot(data=titanic) +
  geom_mosaic(aes(weight=Freq, x=product(Class, Age), fill=Survived))

# we can change where we define variables
ggplot(data=titanic, aes(weight = Freq, fill=Survived, x=product(Class, Age))) +
  geom_mosaic()

ggplot(data=titanic) +
  geom_mosaic(aes(weight=Freq, x=product(Class), conds=product(Age), fill=Survived))
ggplot(data=titanic) +
  geom_mosaic(aes(weight=Freq, x=product(Survived, Class), fill=Age))

## Not run:
data(happy, package="productplots")

ggplot(data = happy) + geom_mosaic(aes(x=product(happy)), divider="hbar")
ggplot(data = happy) + geom_mosaic(aes(x=product(happy))) +
  coord_flip()
# weighting is important
ggplot(data = happy) +
  geom_mosaic(aes(weight=wtssall, x=product(happy)))
ggplot(data = happy) + geom_mosaic(aes(weight=wtssall, x=product(health), fill=happy)) +
  theme(axis.text.x=element_text(angle=35))
ggplot(data = happy) +
  geom_mosaic(aes(weight=wtssall, x=product(health), fill=happy), na.rm=TRUE)
ggplot(data = happy) +
  geom_mosaic(aes(weight=wtssall, x=product(health, sex, degree), fill=happy),
  na.rm=TRUE)

# here is where a bit more control over the spacing of the bars is helpful:
# set labels manually:
ggplot(data = happy) +
  geom_mosaic(aes(weight=wtssall, x=product(age), fill=happy), na.rm=TRUE, offset=0) +
  scale_x_productlist("Age", labels=c(17+1:72))
# thin out labels manually:
labels <- c(17+1:72)
labels[labels %% 5 != 0] <- ""
ggplot(data = happy) +
  geom_mosaic(aes(weight=wtssall, x=product(age), fill=happy), na.rm=TRUE, offset=0) +
  scale_x_productlist("Age", labels=labels)
ggplot(data = happy) +
```

```
  geom_mosaic(aes(weight=wtssall, x=product(age), fill=happy, conds = sex),
  divider=mosaic("v"), na.rm=TRUE, offset=0.001) +
  scale_x_productlist("Age", labels=labels)
# facetting works!!!!
ggplot(data = happy) +
  geom_mosaic(aes(weight=wtssall, x=product(age), fill=happy), na.rm=TRUE, offset = 0) +
  facet_grid(sex~.) +
  scale_x_productlist("Age", labels=labels)

ggplot(data = happy) +
  geom_mosaic(aes(weight = wtssall, x = product(happy, finrela, health)),
  divider=mosaic("h"))
ggplot(data = happy) +
  geom_mosaic(aes(weight = wtssall, x = product(happy, finrela, health)), offset=.005)

# Spine example
ggplot(data = happy) +
 geom_mosaic(aes(weight = wtssall, x = product(health), fill = health)) +
 facet_grid(happy~.)

## End(Not run)
```

`hspine` *Horizontal spine partition: height constant, width varies.*

Description

Horizontal spine partition: height constant, width varies.

Usage

```
hspine(data, bounds, offset = offset, max = NULL)
```

Arguments

`data`	bounds data frame
`bounds`	bounds of space to partition
`offset`	space between spines
`max`	maximum value

mosaic *Template for a mosaic plot. A mosaic plot is composed of spines in alternating directions.*

Description

Template for a mosaic plot. A mosaic plot is composed of spines in alternating directions.

Usage

```
mosaic(direction = "h")
```

Arguments

direction	direction of first split

product *Wrapper for a list*

Description

Wrapper for a list

Usage

```
product(x, ...)
```

Arguments

x	name of the variable going into the product plot.
...	arbitrarily many additional variables.

Examples

```
data(Titanic)
titanic <- as.data.frame(Titanic)
titanic$Survived <- factor(titanic$Survived, levels=c("Yes", "No"))
ggplot(data=titanic) +
  geom_mosaic(aes(weight=Freq, x=product(Survived, Class), fill=Survived))
```

scale_productlist *Product scales for mosaic plots*

Description

product scales are especially introduced for use with mosaic plots: they are a hybrid of continuous and discrete scales.

Usage

```
scale_x_productlist(name = waiver(), breaks = product_breaks(),
  minor_breaks = NULL, labels = product_labels(), limits = NULL,
  expand = waiver(), oob = scales:::censor, na.value = NA_real_,
  trans = "identity", position = "bottom", sec.axis = waiver())

ScaleContinuousProduct
```

Arguments

name	The name of the scale. Used as axis or legend title. If NULL, the default, the name of the scale is taken from the first mapping used for that aesthetic.
breaks	One of: • NULL for no breaks • waiver() for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output
minor_breaks	One of: • NULL for no minor breaks • waiver() for the default breaks (one minor break between each major break) • A numeric vector of positions • A function that given the limits returns a vector of minor breaks.
labels	One of: • NULL for no labels • waiver() for the default labels computed by the transformation object • A character vector giving labels (must be same length as breaks) • A function that takes the breaks as input and returns labels as output
limits	A numeric vector of length two providing limits of the scale. Use NA to refer to the existing minimum or maximum.
expand	A numeric vector of length two giving multiplicative and additive expansion constants. These constants ensure that the data is placed some distance away from the axes. The defaults are c(0.05, 0) for continuous variables, and c(0, 0.6) for discrete variables.

oob	Function that handles limits outside of the scale limits (out of bounds). The default replaces out of bounds values with NA.
na.value	Missing values will be replaced with this value.
trans	Either the name of a transformation object, or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "exp", "identity", "log", "log10", "log1p", "log2", "logit", "probability", "probit", "reciprocal", "reverse" and "sqrt". A transformation object bundles together a transform, it's inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called name_trans, e.g. boxcox_trans. You can create your own transformation with trans_new.
position	The position of the axis. "left" or "right" for vertical scales, "top" or "bottom" for horizontal scales
sec.axis	specify a secondary axis

Format

An object of class ScaleContinuousProduct (inherits from ScaleContinuousPosition, ScaleContinuous, Scale, ggproto) of length 4.

---

scale_type.product *Helper function that ggplot2 needs for determining scales on x and y*

Description

Helper function that ggplot2 needs for determining scales on x and y

Usage

```
scale_type.product(x)
```

Arguments

x	variable under consideration

Value

character string "product"

scale_type.productlist *Helper function that ggplot2 needs for determining scales on x and y*

Description

Helper function that ggplot2 needs for determining scales on x and y

Usage

```
scale_type.productlist(x)
```

Arguments

x	variable under consideration

Value

character string "productlist"

spine *Spine partition: divide longest dimesion.*

Description

Spine partition: divide longest dimesion.

Usage

```
spine(data, bounds, offset = offset, max = NULL)
```

Arguments

data	bounds data frame
bounds	bounds of space to partition
offset	space between spines
max	maximum value

StatMosaic *Geom proto*

Description

Geom proto

vspine *Vertical spine partition: width constant, height varies.*

Description

Vertical spine partition: width constant, height varies.

Usage

```
vspine(data, bounds, offset = offset, max = NULL)
```

Arguments

data	bounds data frame
bounds	bounds of space to partition
offset	space between spines
max	maximum value

Index