Package 'alluvial'

July 22, 2025

Type Package

Title Alluvial Diagrams

Version 0.1-2

Date 2016-09-09

Description Creating alluvial diagrams (also known as parallel sets plots) for multivariate and time series-like data.

URL https://github.com/mbojan/alluvial

BugReports https://github.com/mbojan/alluvial/issues

Suggests devtools, testthat, reshape2, knitr, rmarkdown, dplyr

License MIT + file LICENSE

LazyLoad yes

LazyData yes

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Author Michal Bojanowski [aut, cre], Robin Edwards [aut]

Maintainer Michal Bojanowski <michal2992@gmail.com>

Repository CRAN

Date/Publication 2016-09-09 13:08:51

Contents

alluvial *Alluvial diagram*

Description

Drawing alluvial diagrams, also known as parallel set plots.

Usage

```
alluvial(..., freq, col = "gray", border = 0, layer, hide = FALSE,
  alpha = 0.5, gap.width = 0.05, xw = 0.1, cw = 0.1, blocks = TRUE,
  ordering = NULL, axis_labels = NULL, cex = par("cex"),
  cex.axis = par("cex.axis"))
```

Arguments

`...`	vectors or data frames, all for the same number of observations
`freq`	numeric, vector of frequencies of the same length as the number of observations
`col`	vector of colors of the stripes
`border`	vector of border colors for the stripes
`layer`	numeric, order of drawing of the stripes
`hide`	logical, should particular stripe be plotted
`alpha`	numeric, vector of transparency of the stripes
`gap.width`	numeric, relative width of inter-category gaps
`xw`	numeric, the distance from the set axis to the control points of the xspline
`cw`	numeric, width of the category axis
`blocks`	logical, whether to use blocks to tie the flows together at each category, versus contiguous ribbons (also admits character value "bookends")
`ordering`	list of numeric vectors allowing to reorder the alluvia on each axis separately, see Examples
`axis_labels`	character, labels of the axes, defaults to variable names in the data
`cex, cex.axis`	numeric, scaling of fonts of category labels and axis labels respectively. See `par`.

Value

Invisibly a list with elements:

`endpoints`	A list of matrices of y-coordinates of endpoints of the alluvia. x-coordinates are consecutive natural numbers.

Note

Please mind that the API is planned to change to be more compatible with **dplyr** verbs.

Examples

```
# Titanic data
tit <- as.data.frame(Titanic)

# 2d
tit2d <- aggregate( Freq ~ Class + Survived, data=tit, sum)
alluvial( tit2d[,1:2], freq=tit2d$Freq, xw=0.0, alpha=0.8,
         gap.width=0.1, col= "steelblue", border="white",
         layer = tit2d$Survived != "Yes" )

alluvial( tit2d[,1:2], freq=tit2d$Freq,
         hide=tit2d$Freq < 150,
         xw=0.0, alpha=0.8,
         gap.width=0.1, col= "steelblue", border="white",
         layer = tit2d$Survived != "Yes" )

# 3d
tit3d <- aggregate( Freq ~ Class + Sex + Survived, data=tit, sum)

alluvial(tit3d[,1:3], freq=tit3d$Freq, alpha=1, xw=0.2,
         col=ifelse( tit3d$Survived == "No", "red", "gray"),
         layer = tit3d$Sex != "Female",
         border="white")


# 4d
alluvial( tit[,1:4], freq=tit$Freq, border=NA,
         hide = tit$Freq < quantile(tit$Freq, .50),
         col=ifelse( tit$Class == "3rd" & tit$Sex == "Male", "red", "gray") )

# 3d example with custom ordering
# Reorder "Sex" axis according to survival status
ord <- list(NULL, with(tit3d, order(Sex, Survived)), NULL)
alluvial(tit3d[,1:3], freq=tit3d$Freq, alpha=1, xw=0.2,
         col=ifelse( tit3d$Survived == "No", "red", "gray"),
         layer = tit3d$Sex != "Female",
         border="white", ordering=ord)

# Possible blocks options
for (blocks in c(TRUE, FALSE, "bookends")) {

    # Elaborate alluvial diagram from main examples file
    alluvial( tit[, 1:4], freq = tit$Freq, border = NA,
              hide = tit$Freq < quantile(tit$Freq, .50),
              col = ifelse( tit$Class == "3rd" & tit$Sex == "Male",
                            "red", "gray" ),
              blocks = blocks )
}


# Data returned
x <- alluvial( tit2d[,1:2], freq=tit2d$Freq, xw=0.0, alpha=0.8,
```

```
          gap.width=0.1, col= "steelblue", border="white",
          layer = tit2d$Survived != "Yes" )
points( rep(1, 16), x$endpoints[[1]], col="green")
points( rep(2, 16), x$endpoints[[2]], col="blue")
```

---

alluvial_ts *Alluvial diagram for multiple time series data*

Description

This is a variant of alluvial diagram suitable for multiple (cross-sectional) time series. It also works with continuous variables equivalent to time

Usage

```
alluvial_ts(dat, wave = NA, ygap = 1, col = NA, alpha = NA,
  plotdir = "up", rankup = FALSE, lab.cex = 1, lab.col = "black",
  xmargin = 0.1, axis.col = "black", title = NA, title.cex = 1,
  axis.cex = 1, grid = FALSE, grid.col = "grey80", grid.lwd = 1,
  leg.mode = TRUE, leg.x = 0.1, leg.y = 0.9, leg.cex = 1,
  leg.col = "black", leg.lty = NA, leg.lwd = NA, leg.max = NA,
  xlab = NA, ylab = NA, xlab.pos = 2, ylab.pos = 1, lwd = 1, ...)
```

Arguments

dat	data.frame of time-series (or suitable equivalent continuously disaggregated data), with 3 columns (in order: category, time-variable, value) with <= 1 row for each category-time combination
wave	numeric, curve wavyness defined in terms of x axis data range - i.e. bezier point offset. Experiment to get this right
ygap	numeric, vertical distance between polygons - a multiple of 10% of the mean data value
col	colour, value or vector of length matching the number of unique categories. Individual colours of vector are mapped to categories in alpha-numeric order
alpha	numeric, [0,1] polygon fill transparency
plotdir	character, string ('up', 'down' or 'centred') giving the vertical alignment of polygon stacks
rankup	logical, rank polygons on time axes upward by magnitude (largest to smallest) or not
lab.cex	numeric, category label font size
lab.col	colour, of category label
xmargin	numeric [0,1], proportional space for category labels
axis.col	colour, of axes
title	character, plot title

title.cex	numeric, plot title font size
axis.cex	numeric, font size of x-axis break labels
grid	logical, plot vertical axes
grid.col	colour, of grid axes
grid.lwd	numeric, line width of grid axes
leg.mode	logical, draw y-axis scale legend inside largest data point (TRUE default) or alternatively with custom position/value (FALSE)
leg.x, leg.y	numeric [0,1], x/y positions of legend if leg.mode = FALSE
leg.cex	numeric, legend text size
leg.col	colour, of legend lines and text
leg.lty	numeric, code for legend line type
leg.lwd	numeric, legend line width
leg.max	numeric, legend scale line width
xlab, ylab	character, x-axis / y-axis titles
xlab.pos, ylab.pos	numeric, perpendicular offset for axis titles
lwd	numeric, value or vector of length matching the number of unique categories for polygon stroke line width. Individual values of vector are mapped to categories in alpha-numeric order
...	arguments to pass to polygon()

Examples

```
if( require(reshape2) )
{
data(Refugees)
reshape2::dcast(Refugees, country ~ year, value.var = 'refugees')
d <- Refugees

set.seed(39) # for nice colours
cols <- hsv(h = sample(1:10/10), s = sample(3:12)/15, v = sample(3:12)/15)

alluvial_ts(d)
alluvial_ts(d, wave = .2, ygap = 5, lwd = 3)
alluvial_ts(d, wave = .3, ygap = 5, col = cols)
alluvial_ts(d, wave = .3, ygap = 5, col = cols, rankup = TRUE)
alluvial_ts(d, wave = .3, ygap = 5, col = cols, plotdir = 'down')
alluvial_ts(d, wave = .3, ygap = 5, col = cols, plotdir = 'centred', grid=TRUE,
            grid.lwd = 5)
alluvial_ts(d, wave =  0, ygap = 0, col = cols, alpha = .9, border = 'white',
            grid = TRUE, grid.lwd = 5)
alluvial_ts(d, wave = .3, ygap = 5, col = cols, xmargin = 0.4)
alluvial_ts(d, wave = .3, ygap = 5, col = cols, xmargin = 0.3, lab.cex = .7)
alluvial_ts(d, wave = .3, ygap = 5, col = cols, xmargin = 0.3, lab.cex=.7,
            leg.cex=.7, leg.col = 'white')
alluvial_ts(d, wave = .3, ygap = 5, col = cols, leg.mode = FALSE, leg.x = .1,
```

```
            leg.y = .7, leg.max = 3e6)
alluvial_ts(d, wave = .3, ygap = 5, col = cols, plotdir = 'centred', alpha=.9,
            grid = TRUE, grid.lwd = 5, xmargin = 0.2, lab.cex = .7, xlab = '',
            ylab = '', border = NA, axis.cex = .8, leg.cex = .7,
            leg.col='white',
          title = "UNHCR-recognised refugees\nTop 10 countries (2003-13)\n")

# non time-series example - Virginia deaths dataset
d <- reshape2::melt(data.frame(age=row.names(VADeaths), VADeaths), id.vars='age')[,c(2,1,3)]
names(d) = c('pop_group','age_group','deaths')
alluvial_ts(d)
}
```

`Refugees` *Refugees data*

Description

Top 10 countries/territories of origin (excluding "Various") for period 2003-13 of UNHCR statistics on "Persons recognized as refugees under the 1951 UN Convention/1967 Protocol, the 1969 OAU Convention, in accordance with the UNHCR Statute, persons granted a complementary form of protection and those granted temporary protection."

Format

Data frame with the following columns:

country Country or territory of origin

year Year (2003-13)

refugees Persons recognized as refugees under the 1951 UN Convention, etc..

Source

http://data.un.org/Data.aspx?d=UNHCR&f=indID%3aType-Ref

Index