

Package: alluvial (via r-universe)

October 10, 2026

Type Package

Title Alluvial Diagrams

Version 0.2-0

Date 2017-04-07

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

URL <http://mbojan.github.io/alluvial/>

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

Imports dplyr, tidyr

Suggests devtools, testthat, reshape2, knitr, rmarkdown,

License MIT + file LICENSE

LazyLoad yes

LazyData yes

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/roxygen2/markdown TRUE

VignetteBuilder knitr

Encoding UTF-8

Depends R (>= 3.5)

Config/pak/sysreqs libicu-dev

Repository <https://mbojan.r-universe.dev>

Date/Publication 2026-05-05 15:09:12 UTC

RemoteUrl <https://github.com/mbojan/alluvial>

RemoteRef HEAD

RemoteSha 4003444cb43c3b2bf8961951912563cd73c81629

Contents

alluvial	2
alluvial_ts	5
Polpan	7
Refugees	9
Index	10

alluvial	<i>Alluvial diagram</i>
----------	-------------------------

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,  
  mar = c(2, 1, 1, 1),  
  cex = par("cex"),  
  xlim_offset = c(0, 0),  
  ylim_offset = c(0, 0),  
  cex.axis = par("cex.axis"),  
  axes = TRUE,  
  ann = TRUE,  
  title = NULL  
)
```

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

<code>border</code>	vector of border colors for the stripes
<code>layer</code>	numeric, order of drawing of the stripes
<code>hide</code>	logical, should particular stripe be plotted
<code>alpha</code>	numeric, vector of transparency of the stripes
<code>gap.width</code>	numeric, relative width of inter-category gaps
<code>xw</code>	numeric, the distance from the set axis to the control points of the xspline
<code>cw</code>	numeric, width of the category axis
<code>blocks</code>	logical, whether to use blocks to tie the flows together at each category, versus contiguous ribbons (also admits character value "bookends")
<code>ordering</code>	list of numeric vectors allowing to reorder the alluvia on each axis separately, see Examples
<code>axis_labels</code>	character, labels of the axes, defaults to variable names in the data
<code>mar</code>	numeric, plot margins as in par
<code>cex, cex.axis</code>	numeric, scaling of fonts of category labels and axis labels respectively. See par .
<code>xlim_offset, ylim_offset</code>	numeric vectors of length 2, passed to <code>xlim</code> and <code>ylim</code> of plot , and allow for adjusting the limits of the plotting region
<code>axes</code>	logical, whether to draw axes, defaults to TRUE
<code>ann</code>	logical, whether to draw annotations: category labels. Defaults to TRUE
<code>title</code>	character, plot title

Value

Invisibly a list with elements:

<code>endpoints</code>	A data frame with data on locations of the stripes with columns: <code>...</code> Vectors/data frames supplied to <code>alluvial</code> through <code>...</code> that define the axes <code>.bottom, .top</code> Y locations of bottom and top coordinates respectively at which the stripes originate from the axis <code>.axis</code> <code>.axis</code> Axis number counting from the left
<code>category_midpoints</code>	List of vectors of Y locations of category block midpoints.
<code>alluvium_midpoints</code>	A data frame with location of midpoints on each alluvium segment with columns: <code>...</code> Vectors/data frames supplied to <code>alluvial</code> through the <code>...</code> <code>.axis_from, .axis_to</code> IDs of axes that a segment originates from and goes to <code>.x, .y</code> X and Y locations of the alluvium midpoints <code>.slope</code> The (approximate) slope of the alluvium at the midpoint

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

<code>dat</code>	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
<code>wave</code>	numeric, curve wavyness defined in terms of x axis data range - i.e. bezier point offset. Experiment to get this right
<code>ygap</code>	numeric, vertical distance between polygons - a multiple of 10% of the mean data value
<code>col</code>	colour, value or vector of length matching the number of unique categories. Individual colours of vector are mapped to categories in alpha-numeric order
<code>alpha</code>	numeric, [0,1] polygon fill transparency
<code>plotdir</code>	character, string ('up', 'down' or 'centred') giving the vertical alignment of polygon stacks
<code>rankup</code>	logical, rank polygons on time axes upward by magnitude (largest to smallest) or not
<code>lab.cex</code>	numeric, category label font size
<code>lab.col</code>	colour, of category label
<code>xmargin</code>	numeric [0,1], proportional space for category labels
<code>axis.col</code>	colour, of axes
<code>title</code>	character, plot title
<code>title.cex</code>	numeric, plot title font size
<code>axis.cex</code>	numeric, font size of x-axis break labels
<code>grid</code>	logical, plot vertical axes
<code>grid.col</code>	colour, of grid axes
<code>grid.lwd</code>	numeric, line width of grid axes
<code>leg.mode</code>	logical, draw y-axis scale legend inside largest data point (TRUE default) or alternatively with custom position/value (FALSE)
<code>leg.x, leg.y</code>	numeric [0,1], x/y positions of legend if leg.mode = FALSE
<code>leg.cex</code>	numeric, legend text size
<code>leg.col</code>	colour, of legend lines and text
<code>leg.lty</code>	numeric, code for legend line type
<code>leg.lwd</code>	numeric, legend line width
<code>leg.max</code>	numeric, legend scale line width
<code>xlab, ylab</code>	character, x-axis / y-axis titles
<code>xlab.pos, ylab.pos</code>	numeric, perpendicular offset for axis titles
<code>lwd</code>	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
<code>...</code>	arguments to pass to <code>polygon()</code>

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)
}

```

 Polpan

POLPAN voting flows

Description

Voting choices across three consecutive Polish parliamentary elections (2007, 2011, 2015) reported by respondents of the Polish Panel Study (POLPAN). Each row represents a unique combination of voting choices and gives the number of respondents with that pattern.

Usage

```
data(Polpan)
```

Format

A data frame with 51 rows and 4 columns:

party2007 Party voted for in the 2007 parliamentary election.

party2011 Party voted for in the 2011 parliamentary election.

party2015 Party voted for in the 2015 parliamentary election.

n Number of respondents with this combination of voting choices.

Each of the partyXXXX columns takes one of the following values: "PO" (Civic Platform / *Platforma Obywatelska*), "PiS" (Law and Justice / *Prawo i Sprawiedliwosc*), "PSL" (Polish People's Party / *Polskie Stronnictwo Ludowe*), "Left" (left-wing parties), "Other", or "not voted".

Source

Polish Panel Study (POLPAN, <https://www.polpan.org>), Marta Kołczyńska (<https://martakolczynska.com/>) and her [blog post](#).

Examples

```
# color palette
p <- c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
       "#FF7F00", "#FFFF33")

# How did the 2007 voters vote in subsequent elections?
alluvial(
  Polpan[,1:3],
  freq = Polpan$n,
  col = p[match(Polpan$party2007, sort(unique(Polpan$party2007)))]
)

# Remove flows with freqs less than 10
alluvial(
  Polpan[,1:3],
  freq = Polpan$n,
  col = p[match(Polpan$party2007, sort(unique(Polpan$party2007)))]],
  hide = Polpan$n < 5
)

# How did the 2015 voters vote in preceding elections?
alluvial(
  Polpan[,1:3],
  freq = Polpan$n,
  col = p[match(Polpan$party2015, sort(unique(Polpan$party2007)))]],
  hide = Polpan$n < 5
)
```

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

* **datasets**

Polpan, [7](#)

alluvial, [2](#)

alluvial_ts, [5](#)

par, [3](#)

plot, [3](#)

Polpan, [7](#)

Refugees, [9](#)