Package 'beeswarm'

August 29, 2016

Version 0.2.3

Date 2016-04-25

Title The Bee Swarm Plot, an Alternative to Stripchart

Author Aron Eklund

Maintainer Aron Eklund <eklund@cbs.dtu.dk>

Imports stats, graphics, grDevices, utils

Description The bee swarm plot is a one-dimensional scatter plot like ``stripchart'', but with closely-packed, non-overlapping points.

License Artistic-2.0

URL http://www.cbs.dtu.dk/~eklund/beeswarm/

NeedsCompilation no

Repository CRAN

Date/Publication 2016-04-25 17:02:42

R topics documented:

`beeswarm` *Bee swarm plot*

Description

Create a bee swarm plot. A bee swarm plot is a one-dimensional scatter plot similar to `stripchart`, but with various methods to separate coincident points such that each point is visible. Also, `beeswarm` introduces additional features unavailable in `stripchart`, such as the ability to control the color and plotting character of each point.

Usage

```
beeswarm(x, ...)

## S3 method for class 'formula'
beeswarm(formula, data = NULL, subset, na.action = NULL,
         pwpch = NULL, pwcol = NULL, pwbg = NULL, dlab, glab, ...)

## Default S3 method:
beeswarm(x,
    method = c("swarm", "center", "hex", "square"),
    vertical = TRUE, horizontal = !vertical,
    cex = 1, spacing = 1, breaks = NULL,
    labels, at = NULL,
    corral = c("none", "gutter", "wrap", "random", "omit"),
    corralWidth, side = 0L,
    priority = c("ascending", "descending", "density", "random", "none"),
    pch = par("pch"), col = par("col"), bg = NA,
    pwpch = NULL, pwcol = NULL, pwbg = NULL,
    do.plot = TRUE, add = FALSE, axes = TRUE, log = FALSE,
    xlim = NULL, ylim = NULL, dlim = NULL, glim = NULL,
    xlab = NULL, ylab = NULL, dlab = "", glab = "",
    ...)
```

Arguments

`formula`	A formula, such as `y ~ grp`, where `y` is a numeric vector of data values to be split into groups according to the grouping variable `grp` (usually a factor).
`data`	A data.frame (or list) from which the variables in `formula` should be taken.
`subset`	An optional vector specifying a subset of observations to be used.
`na.action`	A function which indicates what should happen when the data contain `NA`s. The default is to quietly ignore missing values in either the response or the group.
`x`	A numeric vector, or a data frame or list of numeric vectors, each of which is plotted as an individual swarm.
`method`	Method for arranging points (see Details).
`vertical, horizontal`	Orientation of the plot. `horizontal` takes precedence if both are specified.
`cex`	Size of points relative to the default given by `par("cex")`. Unlike other plotting functions, this must be a single value.
`spacing`	Relative spacing between points.
`breaks`	Breakpoints (optional). If `NULL`, breakpoints are chosen automatically. If `NA`, bins are not used (similar to `stripchart` with `method = "stack"`).
`labels`	Labels for each group. Recycled if necessary. By default, these are inferred from the data.
`at`	Numeric vector giving the locations where the swarms should be drawn; defaults to `1:n` where *n* is the number of groups.

corral	Method to adjust points that would be placed outside their own group region (see Details).
corralWidth	Width of the "corral" in user coordinates. If missing, a sensible value will be chosen.
side	Direction to perform jittering: 0: both directions; 1: to the right or upwards; -1: to the left or downwards.
priority	Order used to perform point layout when method is "swarm"; ignored otherwise (see Details).
pch, col, bg	Plotting characters and colors, specified by group. Recycled if necessary (see Details).
pwpch, pwcol, pwbg	"Point-wise" plotting characters and colors, specified for each data point (see Details).
do.plot	Draw a plot?
add	Add to an existing plot?
axes	Draw axes and box?
log	Use a logarithmic scale on the data axis?
xlim, ylim	Limits of the plot.
dlim, glim	An alternative way to specify limits (see Details).
xlab, ylab	Axis labels.
dlab, glab	An alternative way to specify axis labels (see Details).
...	Further arguments passed to plot.

Details

Several methods for placing the points are available; each method uses a different algorithm to avoid overlapping points.

The default method, swarm, places points in increasing order. If a point would overlap an existing point, it is shifted sideways (along the group axis) by a minimal amount sufficient to avoid overlap. breaks is ignored.

The other three methods first discretize the values along the data axis, in order to create more efficient packing: square places the points on a square grid, whereas hex uses a hexagonal grid. center uses a square grid to produce a symmetric swarm. By default, the number of breakpoints for discretization is determined by a combination of the available plotting area and the plotting character size. The discretization of the data can be explicitly controlled using breaks. If breaks is set to NA, the data will not be grouped into intervals; this may be a sensible option if the data is already discrete.

In contrast to most other plotting functions, changing the size of the graphics device will often change the position of the points.

The plotting characters and colors can be controlled in two ways. First, the arguments pch, col and bg can specify plotting characters and colors in the same way as stripchart and boxplot: in short, the arguments apply to each group as a whole (and are recycled if necessary).

Alternatively, the "point-wise" characteristics of each individual data point can be controlled using pwpch, pwcol, and pwbg, which override pch, col and bg if these are also specified. These arguments can be specified as a list or vector. If supplied using the formula method, the arguments can be specified as part of the formula interface; i.e. they are affected by data and subset.

The dlab and glab labels may be used instead of xlab and ylab if those are not specified. dlab applies to the continuous data axis (the Y axis unless horizontal is TRUE); glab to the group axis. Likewise, dlim and glim can be used to specify limits of the axes instead of xlim or ylim.

This function is intended to be mostly compatible with calls to stripchart or boxplot. Thus, code that works with these functions should work with beeswarm with minimal modification.

By default, swarms from different groups are not prevented from overlapping. Thus, large data sets, or data sets with uneven distributions, may produce somewhat unpleasing beeswarms. If this is a problem, consider reducing cex. Another approach is to control runaway points (those that would be plotted outside a region allotted to each group) with the corral argument: The default, "none", does not control runaway points. "gutter" collects runaway points along the boundary between groups. "wrap" implements periodic boundaries. "random" places runaway points randomly in the region. "omit" omits runaway points. See Examples below.

When using the "swarm" method, priority controls the order in which the points are placed; this generally has a noticeable effect on the resulting appearance. "ascending" gives the "traditional" beeswarm plot in which the points are placed in an ascending order. "descending" is the opposite. "density" prioritizes points with higher local density. "random" places points in a random order. "none" places points in the order provided.

Value

A data frame with plotting information, invisibly.

See Also

stripchart, boxplot

Examples

```
## One of the examples from 'stripchart'
beeswarm(decrease ~ treatment,
  data = OrchardSprays, log = TRUE,
  pch = 16, col = rainbow(8))

## One of the examples from 'boxplot', with a beeswarm overlay
 boxplot(len ~ dose, data = ToothGrowth,
           main = "Guinea Pigs' Tooth Growth",
           xlab = "Vitamin C dose mg",
           ylab = "Tooth length")
 beeswarm(len ~ dose, data = ToothGrowth, col = 2, add = TRUE)

## Compare the 4 methods
op <- par(mfrow = c(2,2))
for (m in c("swarm", "center", "hex", "square")) {
```

```
  beeswarm(len ~ dose, data = ToothGrowth, method = m, main = m)
}
par(op)

## Demonstrate the use of 'pwcol'
data(breast)
beeswarm(time_survival ~ ER, data = breast,
  pch = 16, pwcol = 1 + as.numeric(event_survival),
  xlab = "", ylab = "Follow-up time (months)",
  labels = c("ER neg", "ER pos"))
legend("topright", legend = c("Yes", "No"),
  title = "Censored", pch = 16, col = 1:2)

## The list interface
distributions <- list(runif = runif(200, min = -3, max = 3),
                      rnorm = rnorm(200),
                      rlnorm = rlnorm(200, sdlog = 0.5))
beeswarm(distributions, col = 2:4)

## Demonstrate 'pwcol' with the list interface
myCol <- lapply(distributions, function(x) cut(x, breaks = quantile(x), labels = FALSE))
beeswarm(distributions, pch = 16, pwcol = myCol)
legend("bottomright", legend = 1:4, pch = 16, col = 1:4, title = "Quartile")

## Demonstrate the 'corral' methods
par(mfrow = c(2,3))
beeswarm(distributions, col = 2:4,
  main = 'corral = "none" (default)')
beeswarm(distributions, col = 2:4, corral = "gutter",
  main = 'corral = "gutter"')
beeswarm(distributions, col = 2:4, corral = "wrap",
  main = 'corral = "wrap"')
beeswarm(distributions, col = 2:4, corral = "random",
  main = 'corral = "random"')
beeswarm(distributions, col = 2:4, corral = "omit",
  main = 'corral = "omit"')

## Demonstrate 'side' and 'priority'
par(mfrow = c(2,3))
beeswarm(distributions, col = 2:4,
  main = 'Default')
beeswarm(distributions, col = 2:4, side = -1,
  main = 'side = -1')
beeswarm(distributions, col = 2:4, side = 1,
  main = 'side = 1')
beeswarm(distributions, col = 2:4, priority = "descending",
  main = 'priority = "descending"')
beeswarm(distributions, col = 2:4, priority = "random",
  main = 'priority = "random"')
beeswarm(distributions, col = 2:4, priority = "density",
  main = 'priority = "density"')
```

`breast` *Lymph-node-negative primary breast tumors*

Description

Tumor molecular measurements and outcome from breast cancer patients.

Usage

```
data(breast)
```

Format

A data frame with 286 observations on the following 5 variables.

`ER` Estrogen receptor status (factor with levels `neg`, `pos`)

`ESR1` Expression of the ESR1 gene (numeric)

`ERBB2` Expression of the ERBB2 gene (numeric)

`time_survival` Time in months (numeric)

`event_survival` Coded event: 0 = censored, 1 = metastasis (numeric)

Details

ER, ESR1, and ERBB2 were measured on a tumor specimen taken at surgery (time = 0).

ESR1 and ERBB2 expression values were determined by microarray probe sets 205225_at and 216836_s_at using RMA-normalized data.

Source

Wang Y, Klijn JG, Zhang Y, Sieuwerts AM, Look MP, Yang F, Talantov D, Timmermans M, Meijer-van Gelder ME, Yu J, Jatkoe T, Berns EM, Atkins D, Foekens JA. Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. Lancet. 2005 Feb 19-25;365(9460):671-9.

Examples

```
data(breast)

with(breast,
  plot(ESR1, ERBB2, col = as.numeric(ER))
)
```

bxplot *Plot quantile lines*

Description

Plot lines indicating the specified quantiles for each group. This function is intended as a simplified interpretation of boxplot, which can be combined with a beeswarm (or stripchart) plot.

Usage

```
bxplot(x, ...)

## S3 method for class 'formula'
bxplot(formula, data = NULL, ..., subset, na.action = NULL)

## Default S3 method:
bxplot(x, probs = c(0.25, 0.5, 0.75),
    vertical = TRUE, horizontal = !vertical, add = FALSE,
    col = par("col"), lty = par("lty"), lwd = NULL,
    at = NULL, width = 0.75, ...)
```

Arguments

formula	A formula, such as `y ~ grp`, where `y` is a numeric vector of data values to be split into groups according to the grouping variable `grp` (usually a factor).
data	A data.frame (or list) from which the variables in `formula` should be taken.
subset	An optional vector specifying a subset of observations to be used.
na.action	A function which indicates what should happen when the data contain `NA`s. The default is to quietly ignore missing values in either the response or the group.
x	A numeric vector, or a data frame or list of numeric vectors, each of which is considered as a group.
probs	A numeric vector of probabilities with values in [0,1]
vertical, horizontal	Orientation of the plot. `horizontal` takes precedence if both are specified.
add	Add to an existing plot?
col, lty	Color and line type for each probability.
lwd	Line width for each probability (see below).
at	Numeric vector giving the locations where the swarms should be drawn; defaults to `1:n` where *n* is the number of groups.
width	Width of the lines.
...	Further arguments passed to boxplot.

Details

This function is intended as a minimalistic interpration of boxplot; however, the quantiles plotted by bxplot are not necessarily the same as the hinges plotted by a boxplot.

Notice that specifying a vector of graphical parameters such as lwd or col will refer to each of probs, *not* to each group in the data (as one might expect by analogy with boxplot).

If lwd is NULL, and if the probs includes 0.5, lwd will be set to 3 times par("lwd") for probs=0.5, and par("lwd") for the others. (Thus something resembling the median line and hinges of a boxplot is produced by default.)

Value

None.

Examples

```
## bxplot on bottom
beeswarm(len ~ dose, data = ToothGrowth)
bxplot(len ~ dose, data = ToothGrowth, add = TRUE)

## bxplot on top
bxplot(decrease ~ treatment, data = OrchardSprays, probs = 0.5, col = 2)
beeswarm(decrease ~ treatment, data = OrchardSprays, add = TRUE)

## Show deciles
data(breast)
bxplot(time_survival ~ event_survival, data = breast,
  probs = seq(0, 1, by = 0.1), col = rainbow(10))
beeswarm(time_survival ~ event_survival, data = breast,
  pch = 21, bg = "gray75", add = TRUE)
```

---

swarmx *Adjust 1-d data to separate coincident points*

Description

Take a series of points lying in a horizontal or vertical line, and jitter them in the other dimension such that no points are overlapping.

Usage

```
swarmx(x, y,
    xsize = xinch(0.08, warn.log = FALSE),
    ysize = yinch(0.08, warn.log = FALSE),
    log = NULL, cex = par("cex"), side = 0L,
    priority = c("ascending", "descending", "density", "random", "none"))
swarmy(x, y,
    xsize = xinch(0.08, warn.log = FALSE),
```

```
    ysize = yinch(0.08, warn.log = FALSE),
    log = NULL, cex = par("cex"), side = 0L,
    priority = c("ascending", "descending", "density", "random", "none"))
```

Arguments

x, y	Coordinate vectors in any format supported by xy.coords.
xsize, ysize	Width and height of the plotting character in user coordinates.
log	Character string indicating which axes are logarithmic, as in plot.default, or NULL to figure it out automatically.
cex	Relative plotting character size.
side	Direction to perform jittering: 0: both directions; 1: to the right or upwards; -1: to the left or downwards.
priority	Method used to perform point layout (see below).

Details

For swarmx, the input coordinates must lie in a vertical line. For swarmy, the input coordinates must lie in a horizontal line.

swarmx adjusts coordinates to the left or right; swarmy adjusts coordinates up or down.

priority controls the order in which the points are placed; this has generally has a noticeable effect on the resulting appearance. "ascending" gives the "traditional" beeswarm plot in which the points are placed in an ascending order. "descending" is the opposite. "density" prioritizes points with higher local density. "random" places points in a random order. "none" places points in the order provided.

Usually it makes sense to call this function after a plotting device has already been set up (e.g. when adding points to an existing plot), so that the default values for xsize, ysize, and log will be appropriate.

Value

A data frame with columns x and y with the new coordinates.

See Also

beeswarm, jitter

Examples

```
## Plot points in one dimension
index <- rep(0, 100)
values <- rnorm(100)
plot(index, values, xlim = c(-0.5, 2.5))
points(swarmx(index + 1, values), col = 2)
points(swarmx(index + 2, values, cex = 1.5), col = 3, cex = 1.5)
```

```
## Try the horizontal direction, with a log scale
plot(values, index, log = "x", ylim = c(-1, 2))
points(swarmy(values, index + 1), col = 2)

## Newer examples using "side" and "priority"
plot(c(-0.5, 3.5), range(values), type = 'n')
points(swarmx(index + 0, values), col = 1)
points(swarmx(index + 0.9, values, side = -1), col = 2)
points(swarmx(index + 1.1, values, side =  1, priority = "descending"), col = 3)
points(swarmx(index + 2  , values, priority = 'density'), col = 4)
points(swarmx(index + 3  , values, priority = 'random'), col = 5)
```

Index