Package ‘dotsViolin’

October 30, 2023

Title Dot Plots Mimicking Violin Plots

Date 2023-10-19

Version 0.0.1

Maintainer Fernando Roa <froao@ufg.br>

Description Modifies dot plots to have different sizes of dots mimicking violin plots and identifies modes or peaks for them based on frequency and kernel density estimates (Rosenblatt, 1956) <doi:10.1214/aoms/1177728190> (Parzen, 1962) <doi:10.1214/aoms/1177704472>.

Depends R (>= 3.5)

Imports gridExtra, gtools, tidyr, stringr, dplyr, ggplot2, lazyeval, magrittr, rlang, scales, tidyselect

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Fernando Roa [aut, cre], Mariana Pires de Campos Telles [ctb]

Repository CRAN

Date/Publication 2023-10-30 13:20:02 UTC

R topics documented:

dotsViolin *Integrates tables and plots*

Description

A series of functions to get modes/peaks from discrete and continuous variables and integrate them as tables inside plots cite as in: citation("dotsViolin")

dots_and_violin *Makes a composite dot-plot and violin-plot*

Description

This function makes a dot-plot and violin-plot

Usage

```
dots_and_violin(
  dataframe,
  colgroup,
  collabel,
  maxcountcol,
  widthdots,
  maxx,
  labelx,
  desiredorder,
  binwidth,
  adjust,
  binexp,
  fill_group = "fill_group",
  dots = TRUE,
  violin = TRUE
)
```

Arguments

`dataframe`	dataframe
`colgroup`	chr column to group by
`collabel`	label to be used in the plot
`maxcountcol`	numeric variable
`widthdots`	dotsize parameter for geom_dotplot
`maxx`	x axis maximum value
`labelx`	label for x axis

`desiredorder`	order for the colgroup categories
`binwidth`	see, plot_dotviolin
`adjust`	adjust param, see geom_violin
`binexp`	digit to modify size of bins with base 10
`fill_group`	2nd categorical data (use only 2 categories)
`dots`	boolean include dot plot
`violin`	boolean include violin plot

Value

A grid of ggplots that mimics a single plot

Examples

```
fabaceae_mode_counts <- get_modes_counts(fabaceae_clade_n_df, "clade", "parsed_n")
fabaceae_clade_n_df_count <- make_legend_with_stats(fabaceae_mode_counts, "label_count", 1, TRUE)
fabaceae_clade_n_df$label_count <- fabaceae_clade_n_df_count$label_count[match(
  fabaceae_clade_n_df$clade,
  fabaceae_clade_n_df_count$clade
)]
desiredorder1 <- unique(fabaceae_clade_n_df$clade)

dots_and_violin(
  fabaceae_clade_n_df, "clade", "label_count", "parsed_n", 2,
  30, "Chromosome haploid number", desiredorder1, 1, .85, 4,
  "ownwork",
  violin = FALSE
)

dots_and_violin(
  fabaceae_clade_n_df, "clade", "label_count", "parsed_n", 2,
  30, "Chromosome haploid number", desiredorder1, 1, .85, 4,
  dots = FALSE
)
dots_and_violin(
  fabaceae_clade_n_df, "clade", "label_count", "parsed_n", 2,
  30, "Chromosome haploid number", desiredorder1, 1, .85, 4
)

fabaceae_Cx_mode_counts_per_clade_df <- get_peaks_counts_continuous(
  fabaceae_clade_1Cx_df,
  "clade", "Cx", 2, 0.25, 1, 2
)

namecol <- "labelcountcustom"
fabaceae_clade_Cx_peaks_count_df <- make_legend_with_stats(
  fabaceae_Cx_mode_counts_per_clade_df,
  namecol, 1, TRUE
)
fabaceae_clade_1Cx_df$labelcountcustom <-
```

```
    fabaceae_clade_Cx_peaks_count_df$labelcountcustom[match(
      fabaceae_clade_1Cx_df$clade,
      fabaceae_clade_Cx_peaks_count_df$clade
    )]
  desiredorder <- unique(fabaceae_clade_1Cx_df$clade)

  dots_and_violin(
    fabaceae_clade_1Cx_df, "clade", "labelcountcustom", "Cx", 3,
    3, "Genome Size", desiredorder, 0.03, 0.25, 2,
    "ownwork"
  )

  dots_and_violin(
    fabaceae_clade_1Cx_df, "clade", "labelcountcustom", "Cx", 3,
    3, "Genome Size", desiredorder, 0.03, 0.25, 2,
    dots = FALSE
  )
  dots_and_violin(
    fabaceae_clade_1Cx_df, "clade", "labelcountcustom", "Cx", 3,
    3, "Genome Size", desiredorder, 0.03, 0.25, 2,
    "ownwork",
    violin = FALSE
  )
```

fabaceae_clade_1Cx_df *Genome sizes for fabaceae*

Description

fabaceae_clade_1Cx_df: parsed Cx sizes for fabaceae

Usage

```
fabaceae_clade_1Cx_df
```

Format

data.frame with columns:

name OTU, species

clade main fabaceae clade

Cx genome size, Cx

See Also

get_peaks_counts_continuous

`fabaceae_clade_n_df` *chromosomal counts for fabaceae*

Description

fabaceae_clade_n_df: parsed n counts for fabaceae

Usage

```
fabaceae_clade_n_df
```

Format

data.frame with columns:

tip.label OTU, species

clade main fabaceae clade

parsed_n chromosome number, n

See Also

`get_modes_counts`

`get.peaks` *Get peaks of a continuous variable*

Description

This function allows you to get peaks for a continuous variable. Based on the kernel density function

Usage

```
get.peaks(x, bw, signifi, nsmall, ranks = 3)
```

Arguments

`x`	dataframe
`bw`	bandwidth
`signifi`	criteria to bin the data in number of digits
`nsmall`	criteria to approximate (round) data
`ranks`	numeric how many ranks to consider

Value

data.frame

get_modes_counts *get modes, handle ties, ignore less frequent values*

Description

This function comes from an answer for a question in stackoverflow https://stackoverflow.com/questions/42698465/obtaining-3-most-common-elements-of-groups-concatenating-ties-and-ignoring-les

Usage

```
get_modes_counts(data, grouping_col, col2, mode_number = 3)
```

Arguments

`data`	data.frame
`grouping_col`	string split by this column
`col2`	string numerical data column
`mode_number`	numeric number of modes to retrieve

Value

data.frame with modes and counts per group

Examples

```
get_modes_counts(fabaceae_clade_n_df, "clade", "parsed_n")
```

get_peaks_counts_continuous *Peaks of a continuous variable in a dataframe format*

Description

This function allows you to get peaks and summary counts per group for a continuos variable in a dataframe format. Handles ties; least frequent is ignored, except if it is the only one, depends on `get.peaks` function

Usage

```
get_peaks_counts_continuous(
  origtable,
  grouping_col,
  columnname,
  peak_number,
  adjust1,
  signifi,
  nsmall
)
```

Arguments

`origtable`	dataframe
`grouping_col`	column with categories - character
`columnname`	column with numerical data
`peak_number`	number of peaks to get, see get.peaks
`adjust1`	bandwith adjust parameter
`signifi`	see get.peaks function
`nsmall`	see get.peaks function

Value

data.frame

Examples

```
get_peaks_counts_continuous(fabaceae_clade_1Cx_df, "clade", "Cx", 2, 0.25, 1, 2)
```

`make_legend_with_stats` *Make legends with stats*

Description

This function merges all columns in a dataframe to be used as legends

Usage

```
make_legend_with_stats(
  data,
  namecol,
  start_column_idx = 2,
  first_justified_left = FALSE
)
```

Arguments

data	dataframe with columns to be merged into 1
namecol	name to be given to new column
start_column_idx	numeric index of first column to process
first_justified_left	boolean when TRUE justifies first column to the left, defaults to FALSE

Value

data.frame with combined source columns

Examples

```
fabaceae_mode_counts <- get_modes_counts(fabaceae_clade_n_df, "clade", "parsed_n")
fabaceae_clade_n_df_count <- make_legend_with_stats(fabaceae_mode_counts, "label_count", 1, TRUE)
fabaceae_Cx_mode_counts_per_clade_df <- get_peaks_counts_continuous(
  fabaceae_clade_1Cx_df,
  "clade", "Cx", 2, 0.25, 1, 2
)
namecol <- "labelcountcustom"
fabaceae_clade_1Cx_modes_count_df <- make_legend_with_stats(
  fabaceae_Cx_mode_counts_per_clade_df,
  namecol, 1, TRUE
)
```

plot_dotviolin *Makes a dot-plot and violin-plot*

Description

This function makes a dot-plot and violin-plot, internal function

Usage

```
plot_dotviolin(
  dataset,
  par,
  groupcol,
  vary,
  labelx,
  maxx,
  adjust,
  binwidth,
  fill_group = "fill_group",
  font = "mono",
  dots = TRUE,
  violin = TRUE
)
```

Arguments

`dataset`	dataframe with columns to be merged into 1
`par`	dot size
`groupcol`	categories to group
`vary`	numeric variable
`labelx`	x axis label
`maxx`	x axis maximum value
`adjust`	geom_violin adjust parameter
`binwidth`	geom_dotplot binwidth parameter
`fill_group`	2nd category with 2 options as a fill aes argument for geom_dotplot
`font`	font family
`dots`	boolean include dot plot
`violin`	boolean include violin plot

Value

ggplot

Index