Package 'tabplot'

January 17, 2017

Maintainer Martijn Tennekes <mtennekes@gmail.com>

License GPL-3

Title Tableplot, a Visualization of Large Datasets

Type Package

LazyLoad yes

Author Martijn Tennekes and Edwin de Jonge

Description A tableplot is a visualisation of a (large) dataset with a dozen of variables, both numeric and categorical. Each column represents a variable and each row bin is an aggregate of a certain number of records. Numeric variables are visualized as bar charts, and categorical variables as stacked bar charts. Missing values are taken into account. Also supports large 'ffdf' datasets from the 'ff' package.

Version 1.3-1

URL https://github.com/mtennekes/tabplot

Date 2017-01-16

Depends bit, ff, ffbase (>= 0.12.2)

Imports grid

VignetteBuilder knitr

Suggests shiny (>= 0.6), knitr, classInt, ggplot2

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-01-17 08:36:43

R topics documented:

`tabplot-package` *Tableplot, a visualization of large datasets*

Description

A tableplot is a visualisation of a (large) dataset. Each column represents a variable and each row bin is an aggregate of a certain number of records. For numeric variables, a bar chart of the mean values is depicted. For categorical variables, a stacked bar chart is depicted of the proportions of categories. Missing values are taken into account. Also supports large `ffdf` datasets from the `ff` package.

Details

The main function of the package is `tableplot`, which is used to create a tableplot. Other useful functions are:

- `itableplot` to start a graphical user interface (made with the `shiny` package);
- `tablePrepare` to prepare a large dataset. Tableplotting is much faster when the returned object is passed on to `tableplot` rather than the dataset itself;
- `tablePalettes` to show all quantitative and qualitative palettes that are included;
- `tableSave` to save a tableplot;
- `tableChange` to make layout changes to a tableplot.

For a quick intro, see `vignette("tabplot-vignette")`.

Author(s)

Martijn Tennekes <mtennekes@gmail.com> and Edwin de Jonge

Examples

```
# load diamonds dataset from ggplot2
require(ggplot2)
data(diamonds)

# create tableplot
tableplot(diamonds)
```

`-.tabplot` *Compare two tableplots (experimental)*

Description

Two tableplots can be compared by substracting two tabplot-objects. The result is a tabplot_compare-object object in which absolute and relative differences of mean values are stored, as well as a comparison of frequency tables for categorical variables. This object can be plotted with `plot`.

Usage

```
## S3 method for class 'tabplot'
 tp1 - tp2
```

Arguments

`tp1`	the first tabplot-object
`tp2`	the second tabplot-object

Value

a tabplot_compare-object that contains information about the comparison `tp1-tp2`

Examples

```
# load diamonds dataset from ggplot2
require(ggplot2)
data(diamonds)

# calculate normalized prices to be used as sample probabilities
price.norm <- with(diamonds, price / max(diamonds$price))

# draw samples
exp.diamonds <- diamonds[sample(1:nrow(diamonds), size=10000, prob=price.norm, replace=TRUE),]
chp.diamonds <- diamonds[sample(1:nrow(diamonds), size=10000, prob=1-price.norm, replace=TRUE),]

tp1 <- tableplot(exp.diamonds)
tp2 <- tableplot(chp.diamonds)

plot(tp2 - tp1)
```

bin_data *Bin data*

Description

Working horse for tableplot, does the actual binning

Usage

```
bin_data(p, sortCol = 1L, cols = seq_along(p$data), from = 0, to = 1,
  nbins = 100L, decreasing = FALSE, sample = FALSE, sampleBinSize = 100)
```

Arguments

p	prepared dataset (see tablePrepare)
sortCol	column on which the table will be sorted
cols	columns of the data that will be used.
from	lower boundary in quantiles
to	upper boundary in quantiles
nbins	number of bins
decreasing	sort decreasingly
sample	sample or use whole dataset?
sampleBinSize	sample size per bin

bin_hcc_data *Bin high cardinality data*

Description

Bin high cardinality data

Usage

```
bin_hcc_data(bd, max_levels)
```

Arguments

bd	binned dataset (result of bin_data)
max_levels	maximum number of levels. Each column in bd that has more than max_levels categories is rebinned to max_levels categories.

datetime2fac *Transform a date-time vector to a factor*

Description

Transform a date-time vector from class POSIXt or Date to a factor.

Usage

```
datetime2fac(p, rng = range(p, na.rm = TRUE))
```

Arguments

p	date-time vector
rng	range of the factor.

Details

The range rng is cut according to different pretty rounded time periods. The cut with the number of levels that is closest to 6 is chosen. Vector p is cut accordingly. Values of p outside rng are translated to NA.

Value

A factor vector.

Note

This function is still in development stage, and can be improved and optimized. ff vectors are not implemented yet

See Also

num2fac

Examples

```
d <- as.Date("2012-12-21") + sample.int(500, 1000, replace=TRUE)
d2 <- datetime2fac(d)
levels(d2)

t <- as.POSIXlt(Sys.time(), "GMT") + sample.int(1e5, 1000, replace=TRUE)
t2 <- datetime2fac(t)
levels(t2)
```

itableplot *Graphical User Interface to create tableplots*

Description

This graphical user interface is developed with the shiny package. All datasets that are loaded in the global workspace (data.frame, ffdf, or prepared) are passed on to the GUI.

Usage

```
itableplot()
```

Details

This function replaces the old tabplotGTK package, since it only requires an up-to-date browser (and not software like GTK). Furthermore, maintanance is a lot easier.

Examples

```
## Not run:
require(ggplot2)
data(diamonds)

# load other datasets
data(iris)
data(cars)

itableplot()

## End(Not run)
```

loadPrepare *Loads a prepared object*

Description

Loads a prepared object that has been saved with savePrepare.

Usage

```
loadPrepare(dir)
```

Arguments

dir	directory of the prepared object

Value

the prepared object

num2fac *Transform a numerical vector to a factor*

Description

Transform a numerical vector from class POSIXt or Date to a factor.

Usage

```
num2fac(num, method = "pretty", num_scale = "auto", n = 0, brks = NA)
```

Arguments

num	numeric vector
method	• "pretty" intervals are determined by the base function pretty • "kmeans" the method intervals are determined by the method kmeans where n clusters (i.e. intervals) are found • "fixed" determines the intervals by the argument brks • "discrete" the unique values in num are mapped one to one to the levels of the new factor vector)
num_scale	• "auto" used scale is determined automatically • "lin" num is directly fed to the method pretty or kmeans • "log" a logarithmic transformation of num is fed to the method pretty or kmeans
n	the (desired) number of levels. n=0 means automatic
brks	breaks that determine the levels (only required when method="fixed")

Value

A factor vector

Note

This function is still in development stage, and can be improved and optimized. ff vectors are not implemented yet

See Also

datetime2fac

Examples

```
require(ggplot2)
data(diamonds)

diamonds$price2 <- num2fac(diamonds$price)

tableplot(diamonds)
```

plot.tabplot *Plot a tabplot-object*

Description

Plot a tabplot-object. The arguments of this function, which specify the layout, can also be passed on to tableplot directly.

Usage

```
## S3 method for class 'tabplot'
plot(x, fontsize = 10, legend.lines = 8,
  max_print_levels = 15, text_NA = "missing", title = NULL,
  showTitle = NULL, fontsize.title = 14, showNumAxes = TRUE,
  rotateNames = NA, relative = FALSE, vp = NULL, ...)

## S3 method for class 'tabplot_compare'
plot(x, ...)
```

Arguments

x	tabplot-object or tabplot_compare-object
fontsize	the (maximum) fontsize
legend.lines	the number of lines preserved for the legend
max_print_levels	maximum number of printed category labels in the legend
text_NA	text printed for the missing values category in the legend
title	title of the plot (shown if showTitle==TRUE)
showTitle	show the title. By default FALSE, unless a title is given.
fontsize.title	the fontsize of the title
showNumAxes	plots an x-axis for each numerical variable, along with grid lines (TRUE by default).
rotateNames	logical or numeric value that determines the rotation angle of the column names. If TRUE, they are rotated 90 degrees. By default, column names are rotated when the number of columns is greater than 15.
relative	boolean that determines whether relative scales are used for relative tableplots. If TRUE, then mean.diff.rel<-(mean2-mean1)/mean1*100 are used. If FALSE, then the absolute diference is taken: mean <- mean2-mean.
vp	viewport to draw plot in (for instance useful to stack multiple tableplots)
...	other arguments are not used

Examples

```
# load diamonds dataset from ggplot2
require(ggplot2)
data(diamonds)

tab <- tableplot(diamonds)
plot(tab, title="Shine on you Crazy Diamond!!!",
 fontsize=12,
 legend.lines=7,
 fontsize.title=16)
```

`print.tabplot` *Print a tabplot-object*

Description

Print a tabplot-object

Usage

```
## S3 method for class 'tabplot'
print(x, ...)
```

Arguments

`x`	tabplot object
`...`	arguments passed to other methods

`savePrepare` *Saves a prepared object*

Description

Saves a prepared object that has been created by `tablePrepare`. If `tablePrepare` is called with `dir` specified, then it is already saved using this function.

Usage

```
savePrepare(tp, dir, overwrite = FALSE)
```

Arguments

`tp`	the prepared object (created by `tablePrepare`)
`dir`	directory of the prepared object
`overwrite`	logical. If `dir` already contains files of a prepared object, should they be overwritten?

summary.tabplot *Summarize a tabplot-object*

Description

Summarize a tabplot-object

Usage

```
## S3 method for class 'tabplot'
summary(object, digits = max(3, getOption("digits") - 3),
  ...)
```

Arguments

object	tabplot object
digits	integer, used for number formatting with format
...	arguments passed to other methods

tableChange *Change a tabplot-object*

Description

Make layout changes in a tabplot-object, such as the order of columns, and color palettes.

Usage

```
tableChange(tab, select = NULL, select_string = tab$select,
  decreasing = NULL, pals = list(), colorNA = NULL, numPals = NULL)
```

Arguments

tab	tabplot-object
select	index vector of the desired columns (column names are not supported)
select_string	vector of names of the desired columns
decreasing	determines whether the dataset is sorted decreasingly (TRUE) of increasingly (FALSE).
pals	list of color palettes. Each list item is on of the following: • a palette name in tablePalettes, optionally with the starting color between brackets. • a palette vector

If the list items are unnamed, they are applied to all selected categorical variables (recycled if necessary). The list items can be assigned to specific categorical variables, by naming them accordingly.

`colorNA` color for missing values

`numPals` name(s) of the palette(s) that is(are) used for numeric variables ("Blues", "Greys", or "Greens"). Recycled if necessary.

Value

tabplot-object

Examples

```
# load diamonds dataset from ggplot2
require(ggplot2)
data(diamonds)

# assign tableplot as tabplot object
tab <- tableplot(diamonds)

# modify the tabplot object: reverse order of columns and customize palette
tab <- tableChange(tab, select_string=rev(names(diamonds)),
   pals=list(clarity=gray(seq(0,1,length.out=8))))

# plot modified tabplot object
plot(tab)
```

---

tablePalettes *Show / get all palettes of the tabplot package*

Description

All color palettes are shown and/or returned that can be used for tableplots.

Usage

```
tablePalettes(plot = TRUE)
```

Arguments

`plot` Boolean that determines whether the palettes are plot.

Details

Diverging palettes (for numeric variables): "RdYlBu", "RdYlGn", "PRGn", and "BrBG" These palettes are taken from ColorBrewer (Brewer et al., 2003).

Qualitative palattes (for categorical variables): "Set1", "Set2", "Set3", "Set4", "Set5", "Set6", "Set7", "Set8", "Paired", "HCL1", "HCL2", and "HCL3". The default palette, "Set1", is a colorblind-friendly palette (see Okabe and Ito, 2002). Palettes "Set2" to "Set6" and "Paired" are based on ColorBrewer palettes (Brewer et al., 2003). Palette "Set7", is a colorblind-friedly palette from the dichromat package (see Thomas Lumley , 2012). Palette "Set8" is a palette created by Wijffelaars (2008). The "HCL" Palettes are based on the Hue-Chroma-Luminance color space model (see Zeileis et al., 2009). The color red has been removed from the orignal palettes, since it is occupied by missing values.

Value

list with palettes (silent output)

References

Brewer, Cynthia A., Geoffrey W. Hatchard and Mark A. Harrower, 2003, ColorBrewer in Print: A Catalog of Color Schemes for Maps, Cartography and Geographic Information Science 30(1): 5-32.

Okabe, M. and Ito, K. Color Universal Design (CUD) - How to make figures and presentations that are friendly to Colorblind people, 2002

Wijffelaars, M. Synthesis of Color Palettes. Master's thesis. Supervisors Wijk, J. van, and Vliegen, R. 2008

Thomas Lumley (2013). dichromat: Color Schemes for Dichromats. R package version 2.0-0.

Zeileis, A., Hornik, K., and Murrell, P. Escaping RGBland: Selecting colors for statistical graphics. In Proceedings of Computational Statistics & Data Analysis. 2009, 3259-3270.

---

tableplot *Create a tableplot*

Description

A tableplot is a visualisation of (large) multivariate datasets. Each column represents a variable and each row bin is an aggregate of a certain number of records. For numeric variables, a bar chart of the mean values is depicted. For categorical variables, a stacked bar chart is depicted of the proportions of categories. Missing values are taken into account. Also supports large ffdf datasets from the ff package. For a quick intro, see vignette("tabplot-vignette").

Usage

```
tableplot(dat, select, subset = NULL, sortCol = 1, decreasing = TRUE,
  nBins = 100, from = 0, to = 100, nCols = ncol(dat), sample = FALSE,
  sampleBinSize = 1000, scales = "auto", numMode = "mb-sdb-ml",
  max_levels = 50, pals = list("Set1", "Set2", "Set3", "Set4"),
```

```
change_palette_type_at = 20, rev_legend = FALSE, colorNA = "#FF1414",
colorNA_num = "gray75", numPals = "OrBu", limitsX = NULL,
bias_brokenX = 0.8, IQR_bias = 5, select_string = NULL,
subset_string = NULL, colNames = NULL, filter = NULL, plot = TRUE,
...)
```

Arguments

dat	a data.frame, an ffdf object, or an object created by tablePrepare (see details below). Required.
select	expression indicating the columns of dat that are visualized in the tablelplot Also column indices are supported. By default, all columns are visualized. Use select_string for character strings instead of expressions.
subset	logical expression indicing which rows to select in dat (as in subset). It is also possible to provide the name of a categorical variable: then, a tableplot for each category is generated. Use subset_string for character strings instead of an expressions.
sortCol	column name on which the dataset is sorted. It can be an index, expression name, or a character string. PS: in case of ambiguity, the character string is used like in this example: Sepal.Width <- "Petal.Width"; tableplot(iris, sortCol=Sepal.Width).
decreasing	boolean that determines whether the dataset is sorted decreasingly (TRUE) of increasingly (FALSE).
nBins	number of row bins
from	percentage from which the sorted data is shown
to	percentage to which the sorted data is shown
nCols	the maximum number of columns per tableplot. If this number is smaller than the number of columns selected in datNames, multiple tableplots are generated, where each of them contains the sorted column(s).
sample	boolean that determines whether to sample or use the whole data. Only useful when tablePrepare is used.
sampleBinSize	the number of sampled objects per bin, if sample is TRUE.
scales	determines the horizontal axes of the numeric variables in select. Options: "lin", "log", and "auto" for automatic detection. Either scale is a named vector, where the names correspond to numerical variable names, or scale is unnamed, where the values are applied to all numeric variables (recycled if necessary).
numMode	character value that determines how numeric values are plotted. The value consists of the following building blocks, which are concatenated with the "-" symbol. The default value is "mb-sdb-sdl". Prior to version 1.2, "MB-ML" was the default value. sdb sd bars between mean-sd to mean+sd are shown sdl sd lines at mean-sd and mean+sd are shown mb mean bars are shown MB mean bars are shown, where the color of the bar indicate completeness where positive mean values are blue and negative orange

`ml` mean lines are shown

`ML` mean lines are shown, where positive mean values are blue and negative orange

`mean2` mean values are shown

`max_levels` maximum number of levels for categorical variables. Categorical variables with more levels will be rebinned into `max_levels` levels. Either a positive number or -1, which means that categorical variables are never rebinned.

`pals` list of color palettes. Each list item is on of the following:

- a palette name of `tablePalettes`, optionally with the starting color between brackets.
- a color vector

If the list items are unnamed, they are applied to all selected categorical variables (recycled if necessary). The list items can be assigned to specific categorical variables, by naming them accordingly.

`change_palette_type_at`
number at which the type of categorical palettes is changed. For categorical variables with less than `change_palette_type_at` levels, the palette is recycled if necessary. For categorical variables with `change_palette_type_at` levels or more, a new palette of interpolated colors is derived (like a rainbow palette).

`rev_legend` logical value or vector that determines which legends are reversed. If a vector is provided, the names of the items should the names of (a selection of) the categorical variables.

`colorNA` color for missing values for categorical variables.

`colorNA_num` color for missing values for numeric variables. It is used when all values in a bin are missing. If a part of the values are missing, a brighter color is used (see argument `numPals`).

`numPals` vector of palette names that are used for numeric variables. These names are chosen from the diverging palette names in `tablePalettes`. Either `numPals` is a named vector, where the names correspond to the numerical variable names, or an unnamed vector (recycled if necessary). A "-" prefix in the name reverses the palette. When sd bars are shown (see the argument `numMode` of `plot`), only the righthand-side of the palette is used, where brightness is used to differentiate between mean bar and sd bar. When sd bars are not shown (the default in versions before 1.2), the righthand-side of the palette is used for positive mean values, and the lefthand-side for negative mean values. The brightness of the color is determined by the fraction of missing values.

`limitsX` a list of vectors of length two, where each vector contains a lower and an upper limit value. Either the names of `limitsX` correspond to numerical variable names, or `limitsX` is an unnamed list (recycled if necessary).

`bias_brokenX` parameter between 0 en 1 that determines when the x-axis of a numeric variable is broken. If minimum value is at least `bias_brokenX` times the maximum value, then X axis is broken. To turn off broken x-axes, set `bias_brokenX=1`.

`IQR_bias` parameter that determines when a logarithmic scale is used when `scales` is set to "auto". The argument `IQR_bias` is multiplied by the interquartile range as a test.

select_string	character equivalent of the select argument (particularly useful for programming purposes)
subset_string	character equivalent of the subset argument (particularly useful for programming purposes)
colNames	deprecated; used in older versions of tabplot (prior to 0.12): use select_string instead
filter	deprecated; used in older versions of tabplot (prior to 0.12): use subset_string instead
plot	boolean, to plot or not to plot a tableplot
...	layout arguments, such as fontsize and title, are passed on to plot

Details

For large dataset, we recommend to use tablePrepare which does all the necessary preprocessing that are needed to make any tableplot of the particular dataset. The resulting object of this function is passed on to tableplot (argument dat). Now tableplotting is very fast, and even faster with sampling enabled (sample=TRUE).

Value

tabplot-object (silent output). If multiple tableplots are generated (which can be done by either setting subset to a categorical column name, or by restricting the number of columns with nCols), then a list of tabplot-objects is silently returned.

Note

In early development versions of tabplot (prior to version 1.0) it was possible to sort datasets on multiple columns. To increase to tableplot creation speed, this feature is dropped. For multiple sorting purposes, we recommend to use the subset parameter instead.

See Also

itableplot

Examples

```
# load diamonds dataset from ggplot2
require(ggplot2)
data(diamonds)

# default tableplot
tableplot(diamonds)

# prior to verison 1.2, the mean values of numeric variables are displayed
# without standard deviation (see ?plot.tabplot):
tableplot(diamonds, numMode = "MB-ML")

# most expensive diamonds
tableplot(diamonds,
```

```
  select=c(carat, cut, color, clarity, price),
  sortCol=price,
  from=0,
  to=5)

# for large datasets, we recommend to preprocess the data with tablePrepare:
p <- tablePrepare(diamonds)

# specific subsetting
tableplot(p, subset=price < 5000 & cut=='Ideal')

# change palettes
tableplot(p,
  pals=list(cut="Set4", color="Paired", clarity=grey(seq(0, 1,length.out=7))),
  numPals=c(carat="PRGn", price="BrBG"))

# create a tableplot cut category, and fix scale limits of carat, table, and price
tabs <- tableplot(p, subset=cut,
limitsX=list(carat=c(0,4), table=c(55, 65), price=c(0, 20000)), plot=FALSE)
plot(tabs[[3]], title="Very good cut diamonds")
```

`tablePrepare` *Prepares a dataset for tableplotting*

Description

Tableplots from a large dataset can be generated very fast when the preprocessing stage is done only once. This function preprocesses the dataset, and returns an object that can be passed to `tableplot`. From this stage, tableplots are generated very fast, no matter on which column the data is sorted or how many row bins are chosen.

Usage

```
tablePrepare(x, name = NULL, dir = NULL, ...)
```

Arguments

`x`	`data.frame` or `ffdf`, will be transformed into an `ffdf` object.
`name`	name of the dataset
`dir`	directory to store the prepared object. If unspecified, the prepared object will not be saved, and the underlying data will be stored temporarily in `options("fftempdir")`.
`...`	arguments passed to other methods (at the moment only `overwrite` from `savePrepare`)

Details

The function `bin_data` needs a prepared `data.frame` Prepare transforms the supplied data into an `ffdf` object and calculates the order of each of its columns. Knowing the order of the columns speeds up the binning process considerably, For large `ffdf` objects this may be a time consuming step so it can be wise to call prepare before making a tableplot.

Value

a prepared object, including the data and order of each of the columns

Examples

```
# load diamonds dataset from ggplot2
require(ggplot2)
data(diamonds)

p <- tablePrepare(diamonds)

tableplot(p, nBins=200, sortCol=depth)
tableplot(p, nBins=50, sortCol=price)
```

---

tableSave *Save a tableplot*

Description

Save a tableplot in pdf, eps, svg, wmf, png, jpg, bmp, or tiff format.

Usage

```
tableSave(tab, filename = paste(tab$dataset, ".pdf", sep = ""),
  device = default_device(filename), path = NULL, scale = 1,
  width = par("din")[1], height = par("din")[2], dpi = 300,
  onePage = TRUE, ...)
```

Arguments

tab	a tabplot-object, or a list of tabplot-objects, which are either stacked horizontally or put on multiple pages (for pdf only)
filename	filename with extention (pdf, eps, svg, wmf, png, jpg, bmp, or tiff)
device	device, automatically extracted from filename extension
path	path to save to
scale	scaling factor
width	width (in inches)
height	height (in inches)
dpi	dpi to use for raster graphics
onePage	if true, multiple tab objects are stacked horizontally, else they are printed on multiple pages
...	other arguments passed to plot or the used graphics device

Examples

```
## Not run:
require(ggplot2)
data(diamonds)

# default tableplot
tab <- tableplot(diamonds)

# save tableplot
tableSave(tab, filename="diamonds.png", title="Shine on you Crazy Diamond!!!")

## End(Not run)
```

---

tabplot-object *Object that contains the information to plot a tableplot*

Description

An object of class `tabplot` contains the information to plot a tableplot without the steps that may be time-consuming, such as sorting and aggregating. The function `tableplot` silently returns a tabplot-object (use `plot=FALSE` to supress that the tableplot is plot). The function `tableChange` can be used to change a tabplot-object. The generic functions `plot` and `summary` are used to plot and summarize a tabplot-object.

---

tabplot_compare-object

Object that contains the information to plot the difference of two tableplots (experimental)

Description

A `tabplot_compare` is created by substracting two tableplots (see -.tabplot). For numeric variables, both absolute and relative difference of the mean values are computed. For categorical variables, the freqency tables are compared.

Index