Package ‘wrGraph’

July 9, 2020

Version 1.0.2

Title Graphics in the Context of Analyzing High-Throughput Data

Author Wolfgang Raffelsberger [aut, cre]

Maintainer Wolfgang Raffelsberger <w.raffelsberger@unistra.fr>

Description Additional options for making graphics in the context of analyzing high-throughput data are available here.
This includes automatic segmenting of the current device (eg window) to accommodate multiple new plots,
automatic checking for optimal location of legends in plots, small histograms to insert as legends,
histograms re-transforming axis labels to linear when plotting log2-transformed data,
a violin-plot <doi:10.1080/00031305.1998.10480559> function for a wide variety of input-formats,
principal components analysis (PCA) <doi:10.1080/14786440109462720> with bag-plots <doi:10.1080/00031305.1999.10474494> to highlight and compare the center areas for groups of samples,
generic MA-plots (differential- versus average-value plots) <doi:10.1093/nar/30.4.e15>,
staggered count plots and generation of mouse-over interactive html pages.

Depends R (>= 3.1.0)

Imports graphics, grDevices, RColorBrewer, stats, wrMisc

Suggests dplyr, factoextra, FactoMineR, knitr, limma, rmarkdown, sm

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-09 09:30:03 UTC

R topics documented:

`addBagPlot` *Add bagplot to existing plot*

Description

This function adds a bagplot on an existing (scatter-)plot allowing to highlight the central area of the data. Briefly, a bagplot is a bivariate boxplot, see Bagplot, following the basic idea of a boxplot in two dimensions. Of course, multimodal distributions - if not separated first - may likely lead to mis-interpretation, similarly as it is known for interpreting boxplots. If a group of data consists only of 2 data-points, they will be conected using a straight line. It is recommended using transparent colors to highlight the core part of a group (if only 2 points are available, they will be conected using a straight line), in addition, one could use the option to re-plot all (non-outlyer) points (arguments `reCol`, `rePch` and `reCex` must be used).

Usage

```
addBagPlot(
  x,
  lev1 = 0.5,
  outCoef = 2,
  bagCol = NULL,
  bagCont = bagCol,
  bagLwd = 1.5,
  nCore = 4,
  outlCol = 2,
  outlPch = NULL,
  outlCex = 0.6,
  reCol = NULL,
  rePch = NULL,
  reCex = NULL,
  ctrPch = NULL,
  ctrCol = NULL,
  ctrCex = NULL,
  returnOutL = FALSE,
  callFrom = NULL,
  silent = TRUE
)
```

Arguments

`x`	(matrix, list or data.frame) main numeric input of data/points to plot
`lev1`	(numeric) min content of data for central area (default 0.5 for 50 percent)
`outCoef`	(numeric) parameter for defining outliers (equivalent to `range` in `boxplot`)
`bagCol`	(character or integer) color for filling center part of bagplot (default light transparent grey); Note: It is highly suggested to use transparency, otherwise points underneith will be covered
`bagCont`	(character) color for inner and outer contours of bagplot
`bagLwd`	(numeric) line width for outer contour, set to `NULL` for not diaplaying outer contour (see also `par`)
`nCore`	(integer) decide when center should be determined by median or mean: if number of points reach `nCore` the median will be used
`outlCol`	(character or integer) color for highlighting outlyers (for text and replottig outlyers points), set to `NULL` for not highlighting outlyers at all
`outlPch`	(integer) symbol replotting highlighted outlyers (for text and replottig outlyers points), set to `NULL` for not replotting outlyer-points (see also `par`)
`outlCex`	(numeric) cex type expansion factor for labels of highlighted outlyers, set to `NULL` for not printing (row)names of outlyers (see also `par`)
`reCol`	(character or integer) color for replotting (non-outlyer) points, default set to `NULL` for not replotting
`rePch`	(integer) symbol for replotting (non-outlyer) points, default set to `NULL` for not re-plotting (see also `par`)
`reCex`	(numeric) cex type expansion factor for lfor replotting (non-outlyer) points, default set to `NULL` for not replotting
`ctrPch`	(integer) symbol for shwing group center (see also `par`)
`ctrCol`	(character or integer) color for group center symbol
`ctrCex`	(numeric) cex type expansion factor for size of group center (see also `par`)
`returnOutL`	(logical) decide if rownames of (potential) outlyer values should be returned when running the function
`callFrom`	(character) allow easier tracking of messages produced
`silent`	(logical) suppress messages

Value

plot, optional return of matrix with outlyers

See Also

plotPCAw, princomp

Examples

```
set.seed(2020); dat1 <- matrix(round(rnorm(2000),3),ncol=2); rownames(dat1) <- 1:nrow(dat1)
dat1 <- dat1+ 5*matrix(rep(c(0,1,1,0,0,0,1,1),nrow(dat1)/4),byrow=TRUE,ncol=2)
col1 <- rgb(red=c(153,90,203,255 ),green=c(143,195,211,125),blue=c(204,186,78,115),
  alpha=90,maxColorValue=255)
## suppose we know the repartition into 4 subgroups which we would like to highlight them
grp1 <- rep(1:4,nrow(dat1)/4)
plot(dat1,col=grey(0.8),xlab="x",ylab="y",las=1,pch=grp1)
for(i in 1:4) addBagPlot(dat1[which(grp1==i),],bagCol=col1[i])
## slightly improved
library(wrMisc)
col2 <- convColorToTransp(col1,255)
plot(dat1,col=grey(0.8),xlab="x",ylab="y",las=1,pch=grp1)
for(i in 1:4) addBagPlot(dat1[which(grp1==i),],bagCol=col1[i],outlPch=i,
  outlCol=col2[i],bagLwd=3)
```

---

checkForLegLoc *Find best place on plot for placing legend*

Description

This function tries to find the best location for placing a legend of a bivariate plot, ie scatter-plot. All 4 corners of the data to plot are inspected for the least occupation by data plotted while displaying the content of `sampleGrp`. Alternatively, by setting the argument `showLegend` the user-defined legend will be returned

Usage

```
checkForLegLoc(
  matr,
  sampleGrp = NULL,
  showLegend = TRUE,
  suplSpace = 4,
  testCorner = 1:4,
  silent = TRUE,
  callFrom = NULL
)
```

Arguments

`matr`	(matrix, list or data.frame) main data of plot
`sampleGrp`	(character or factor) with this option the text to be displayed in the legend may be taken into consideration for its length

`showLegend`	(logical or character) decide if `matr` should be checked for best location; if `showLegend` contains any of the standard legend-location designations (eg 'topleft') it will be used in the output
`suplSpace`	(numeric) allows to consider extra room taken in legend by symbol and surrounding space, interpreted as n additional characters
`testCorner`	(integer) which corners should be considered (1=left-top, 2=right-top, right-bottom, left-bottom)
`silent`	(logical) suppress messages
`callFrom`	(character) allow easier tracking of message(s) produced

Value

list with $showL indicating if legend is desired and $loc for the proposition of the best location, $nConflicts gives the counts of conflicts

See Also

`legend`

Examples

```
dat1 <- matrix(c(1:5,1,1:5,5), ncol=2)
grp <- c("abc","efghijk")
(legLoc <- checkForLegLoc(dat1, grp))
plot(dat1, cex=3)
legend(legLoc$loc, legend=grp, text.col=2:3, pch=1, cex=0.8)
```

`convertPlotCoordPix` *Convert points of plot to coordinates in pixels*

Description

This function allows conversion the plotting positions ('x' and 'y' coordiantes) of points in a given plot into coordiantes in pixels (of the entire plotting region). It was designed to be used as coordinates in an html file for mouse-over interactivity (display of names of points and links). Of course, the size of the plotting region is crucial and may not be changed afterwards (if the plot is not written to file using png etc). In turn the function `mouseOverHtmlFile` will use the pixel-coordiantes, allowing to annotate given points of a plot for mouse-over interactive html.

Usage

```
convertPlotCoordPix(
  x,
  y,
  useMar = c(6.2, 4, 4, 2),
  plotDim = c(1400, 800),
  plotRes = 100,
  fromTop = TRUE,
  callFrom = NULL,
  silent = FALSE
)
```

Arguments

`x`	(numeric) initial plotting coordinates on x-axis, names of vector - if available- will be used as IDs
`y`	(numeric) initial plotting coordinates on y-axis
`useMar`	(numeric,length=4) margins defined with plot, see also `par`
`plotDim`	(integer, length=2) dimension of the plotting device in pixels, see also `par`
`plotRes`	(integer) resoltion of plotting device, see also `par`
`fromTop`	(logical) toggle if poordinates should start from top
`callFrom`	(character) allow easier tracking of message(s) produced
`silent`	(logical) suppress messages

Value

matrix with x- and y-coordinates in pixels

See Also

`mouseOverHtmlFile`

Examples

```
df1 <- data.frame(id=letters[1:10],x=1:10,y=rep(5,10),mou=paste("point",letters[1:10]),
  link=file.path(tempdir(),paste(LETTERS[1:10],".html",sep="")),stringsAsFactors=FALSE)
## Typically one wants to get pixel-coordinates for plots written to file.
## Here we'll use R's tempdir, later you may want to choose other locations
pngFile <- file.path(tempdir(),"test01.png")
png(pngFile,width=800, height=600,res=72)
## here we'll just plot a set of horiontal points at default parameters ...
plot(df1[,2:3],las=1,main="test01")
dev.off()
## Note: Special characters should be converted for proper display in html during mouse-over
library(wrMisc)
df1$mou <- htmlSpecCharConv(df1$mou)
## Let's add the x- and y-coordiates of the points in pixels to the data.frame
df1 <- cbind(df1,convertPlotCoordPix(x=df1[,2],y=df1[,3],plotD=c(800,600),plotRes=72))
head(df1)
## using mouseOverHtmlFile() one could now make an html document with interactive
## display of names and clockable links to the coordinates we determined here ...
```

`cumFrqPlot` *Cumulative (or sorted) frequency plot (takes columns of 'dat' as separate series)*

Description

Display data as sorted or cumulative frequency plot. This type of plot represents an alternative to plotting data as histograms. Histograms are very universal and which are very intuitive. However,fine-tuning the bandwith (ie width of the bars) may be very delicate, fine resultion details may often remain hidden. One of the advanges of directly displaying all data-points is that subtile differences may be revealed easier, compared to calssical histograms. Furthermore, the plot prensented her offeres more options to display multiple series of data simultaneaously. Thus, this type of plot may be useful to compare eg results of data normalization. Of course, with very large data-sets (eg > 3000 values) this gain of 'details' will be less important (compared to histograms) and will penalize speed. In such cases the argument `thisResol` will get useful as it allows to reduce the resultion and introduce binning. Alternatively for very large data-sets one may looking into density-plots or vioplots (eg `vioplotW`). The argument `CVlimit` allows optionally excluding extreme values. If numeric (& > 2 columns), its value will be used `exclExtrValues` to identify series with column-median > 'CVlimit'. Of course, exclusion of extreme values should be done with great care, important features of the data may get lost.

Usage

```
cumFrqPlot(
  dat,
  cumSum = FALSE,
  exclCol = NULL,
  colNames = NULL,
  displColNa = TRUE,
  tit = NULL,
  xLim = NULL,
  yLim = NULL,
  xLab = NULL,
  yLab = NULL,
  col = NULL,
  CVlimit = NULL,
  thisResol = NULL,
  supTxtAdj = 0,
  supTxtYOffs = 0,
  useLog = "",
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`dat`	(matrix or data.frame) data to plot/inspect

`cumSum`	(logical) for either plotting cumulates Sums (then `thisResol` for number of breaks) or (if `=FALSE`) simply sorted values -> max resolution
`exclCol`	(integer) columlns to exclude
`colNames`	(character) for alternative column/series names in display, as long as `displColNa=TRUE`
`displColNa`	(logical) display column-names
`tit`	(character) custom title
`xLim`	(numeric) custom limit for x-axis (see also `par`)
`yLim`	(numeric) custom limit for y-axis (see also `par`)
`xLab`	(character) custom x-axis label
`yLab`	(character) custom y-axis label
`col`	(integer or character) custom colors
`CVlimit`	(numeric) for the tag 'outlier column' (uses `exclExtrValues`) identify & mark column with median row-CV > `CVlimit`
`thisResol`	(integer) resolution `res` for binning large data-sets
`supTxtAdj`	(numeric) parameter `adj` for supplemetal text
`supTxtYOffs`	(numeric) supplemental offset for text on y axis
`useLog`	(character) default="", otherwise for setting axis in log-scale "x", "y" or "xy"
`silent`	(logical) suppress messages
`callFrom`	(character) allows easier tracking of message(s) produced

Value

plot only

See Also

`layout`, `exclExtrValues` for decision of potential outliers; `hist`, `vioplotW`

Examples

```
set.seed(2017); dat0 <- matrix(rnorm(500), ncol=5, dimnames=list(NULL,1:5))
cumFrqPlot(dat0, tit="Sorted values")
cumFrqPlot(dat0, cumSum=TRUE, tit="Sum of sorted values")
```

histW *Histogram (version by WR)*

Description

This function proposes a few special tweaks to the general hist function : In a number of settings data are treated and plotted as log-data. This function allows feeding directly data which are already log2 and displaying the x-axis (re-translated) in linear scale (see argument isLog). The default settings allow making (very) small histograms ('low resolution'), which may be used as a rough overview of bandwidth and distribution of values in dat. Similar to hist, by changing the parameters nBars and/or breaks very 'high resolution' histograms can be produced. By default it displays n per set of data (on the top of the figure). Note that the argument for (costom) title main is now called tit.

Usage

```
histW(
  dat,
  fileName = "histW",
  output = "screen",
  nBars = 8,
  breaks = NULL,
  tit = NULL,
  subTi = NULL,
  xLab = NULL,
  yLab = NULL,
  imgxSize = 900,
  useCol = NULL,
  useBord = NULL,
  isLog = TRUE,
  cexSubTi = NULL,
  cropHist = TRUE,
  parDefault = TRUE,
  callFrom = NULL,
  silent = FALSE
)
```

Arguments

dat	(matrix, list or data.frame) data to plot
fileName	(character) name of file for saving graphics
output	(character, length=1) options for output on 'screen' or saving image in various formats (set to 'jpg','png' or 'tif')
nBars	(integer) number of bars in histogram (default for 'low resolution' plot to give rough overview)

`breaks`	(integer) for (partial) compatibility with hist() : use only for number of breaks (or 'FD'), gets priority over 'nBars'
`tit`	(character) custom title
`subTi`	(character) may be `FALSE` for NOT displaying, or any text, otherwise range
`xLab`	(character) custom x-axes label
`yLab`	(character) custom y-axes label
`imgxSize`	(integer) width of image when saving to file, see also `par`
`useCol`	(character or integer) custom colors, see also `par`
`useBord`	(character) custom histogram elements border color, see also `par`
`isLog`	(logical) for lin scale signal intensity values where repesentation needs log, assume log2 if `TRUE`
`cexSubTi`	(numerical) subtitle size (expansion factor cex), see also `par`
`cropHist`	(logical) -not implemented yet- designed for cutting off bars with very low ('insignificant') values
`parDefault`	(logical) to automatic adjusting par(marg=,cex.axis=0.8), see also `par`
`callFrom`	(character) allow easier tracking of message(s) produced
`silent`	(logical) suppress messages

See Also

`hist`

Examples

```
set.seed(2016); dat1 <- round(c(rnorm(200,6,0.5),rlnorm(300,2,0.5),rnorm(100,17)),2)
dat1 <- dat1[which(dat1 <50 & dat1 > 0.2)]
histW(dat1,br="FD",isLog=FALSE)
histW(log2(dat1),br="FD",isLog=TRUE)

## quick overview of distributions
layout(partitionPlot(4))
for(i in 1:4) histW(iris[,i],isLog=FALSE,tit=colnames(iris)[i])
```

`legendHist` *Add histogram to existing plot*

Description

Add histogram at pleace of legend using colors from 'colorRamp'.

Usage

```
legendHist(
  x,
  colRamp = NULL,
  location = "bottomright",
  legTit = NULL,
  cex = 0.7,
  srt = 67,
  offS = NULL,
  border = TRUE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`x`	(numeric) main input/component of plot
`colRamp`	(character or integer) set of colors, default is rainbow-like
`location`	(character) for location of histogram inside existing plot (may be 'br','bl','tl','tr','bottomright', 'bottomleft','topleft','topright')
`legTit`	(character, length=1) optional title for histogram-insert
`cex`	(numeric) expansion factor (see also `par`)
`srt`	(numeric) angle for histogram text labels (90 will give vertical label) (see also `par`)
`offS`	(`NULL` or numeric, length=5) fine-tuning of where histogram-insert will be placed and how elements therein are ditributed (default c(xOff=0.2,yOff=0.25,leftOffS=0.05, upperBarEnd=1.05,txtOff=0.02), 1st and 2nd determine proportio of insert relative to entire plotting region, 3rd defines space left on bottom for text, 4th if bars hit ceiling of insert or proportion to leave, 5th for shifting text towards top when turned other than 90 degrees)
`border`	(logical) decide of draw gray rectangle or not around legend
`silent`	(logical) suppress messages
`callFrom`	(character) allow easier tracking of message(s) produced

Value

figure

Examples

```
dat <- rnorm(90); plot(dat)
legendHist(dat, col=1:5)
```

`MAplotW` *MA-plot (differential intensity versus average intensity)*

Description

This type of plot is very common in high-throughput biology, see MA-plot. Basically one would like to compare numerous independent measures (ie gene transcript or protein abundance values) of 2 samples/data-sets, it is usual to compare a change ('Minus'=M) versus absolute mean value ('Average'=A). In high-throughput biology data are typically already transformed to log2 and thus, the 'M'-value represents a relative change. Besides, output from statistical testing by `moderTest2grp` can be directly read to produce MA plots for diagnostic reasons. Please note, that plotting a very number of points in transparency (eg >10000) may take several seconds.

Usage

```
MAplotW(
  Mvalue,
  Avalue = NULL,
  filtFin = NULL,
  ProjNa = NULL,
  FCthrs = NULL,
  subTxt = NULL,
  grayIncrem = TRUE,
  compNa = NULL,
  batchFig = FALSE,
  cexMa = 1.8,
  cexLa = 0.7,
  limM = NULL,
  limA = NULL,
  cexPt = NULL,
  cexSub = NULL,
  useMar = c(6.2, 4, 4, 2),
  callFrom = "",
  silent = FALSE,
  debug = FALSE
)
```

Arguments

`Mvalue`	(numeric or matrix) data to plot; M-values are typically calculated as difference of log2-abundance values and 'Avalue' the mean of log2-abundance values; M-values and A-values may be given as 2 columsn of a matrix, in this case the argument `Avalue` should remain NULL
`Avalue`	(numeric, list or data.frame) if `NULL` it is assumed that 2nd column of 'Mvalue' contains the A-values to be used

`filtFin`	(matrix or logical) The data may get filtered before plotting: If `FALSE` no filtering will get applied; if matrix of `TRUE/FALSE` it will be used as optional custom filter, otherwise (if `Mvalue` if an `MArrayLM`-object eg from limma) a default filtering based on the `filtFin` element will be applied
`ProjNa`	(character) custom title
`FCthrs`	(numeric) Fold-Change threshold (display as line) give as Fold-change and NOT log2(FC)
`subTxt`	(character) custom sub-title
`grayIncrem`	(logical) if `TRUE`, display overlay of points as increased shades of gray
`compNa`	(character) names of groups compared
`batchFig`	(logical) if `TRUE` figure title and axes legends will be kept shorter for display on fewer splace
`cexMa`	(numeric) expansion factor for font-size of title (see also `par`)
`cexLa`	(numeric) expansion factor `cex` for labels (see also `par`)
`limM`	(numeric, length=2) range of axis M-values
`limA`	(numeric, length=2) range of axis A-values
`cexPt`	(numeric) expansion factor `cex` for points (see also `par`)
`cexSub`	(numeric) expansion factor `cex` for subtitle (see also `par`)
`useMar`	(numeric,length=4) custom margings (see also `par`)
`callFrom`	(character) allow easier tracking of message(s) produced
`silent`	(logical) suppress messages
`debug`	(logical) additional messages for debugging

Value

MA-plot only

See Also

(for PCA) `plotPCAw`

Examples

```
library(wrMisc)
set.seed(2005); mat <- matrix(round(runif(600),1),ncol=6)
rownames(mat) <- c(rep(letters[1:25],each=3),letters[2:26])
MAplotW(mat[,2]-mat[,1], rowMeans(mat))
## assume 2 groups with 3 samples each
matMeans <- rowGrpMeans(mat, gr=gl(2,3,labels=LETTERS[3:4]))
MAplotW(matMeans[,2]-matMeans[,1], matMeans)
## assume 2 groups with 3 samples each and run moderated t-test (from package 'limma')
tRes <- moderTest2grp(mat,gl(2,3))
MAplotW(tRes$Mval, tRes$Amean)
MAplotW(M=tRes$Mval, A=tRes$means, FCth=1.3)
MAplotW(tRes)
MAplotW(tRes, limM=c(-2,2), FCth=1.3)
```

mouseOverHtmlFile *Create mouse-over interactive html-pages (with links)*

Description

This function allows generating html pages with interactive mouse-over to display information for the points of the plot and www-links when clicking based on embedded png file. Basically, an html page will be generated which contains a call to display to an image file specified in `pngFileNa` and in the body below pixel-coordinated will be given for disply of information at mouse-over and embedded links.

Usage

```
mouseOverHtmlFile(
  myCoor,
  pngFileNa,
  HtmFileNa = NULL,
  mouseOverTxt = NULL,
  displSi = c(800, 600),
  colNa = NULL,
  tit = "",
  myHtmTit = "",
  myComment = NULL,
  textAtStart = NULL,
  textAtEnd = NULL,
  pxDiam = 5,
  addLinks = NULL,
  linkExt = NULL,
  htmlExt = "htm",
  callFrom = NULL,
  silent = FALSE
)
```

Arguments

`myCoor`	(matrix or data.frame) with initial x&y coordinates of points for plot; with IDs (1st column !!) & coordinates (2nd & 3rd col), data for mouse-over & link (4th & 5th); NOTE : if 'colNa' NOT given, colnames of 'myCoor' will be inspected & filtered (columns of non-conform names may get lost) !!! Associated with (already existing) figure file 'pngFileNa' and make html page where points may be indicated by mouse-over
`pngFileNa`	(character, length=1) filename for complementary png figure (must already exist)
`HtmFileNa`	(character, length=1) filename for html file produced
`mouseOverTxt`	(character, length=1) text for interactive mouse-over in html, if `NULL`, will use col specified by 1st 'colNa' or (if NULL) rownames of 'myCoor'

`displSi`	(integer, length=2) size of image ('pngFileNa') at display in html (width,height), see also `par`
`colNa`	(character) if not `NULL` min length of 3 to custom specify the column-names to be used : 1st for mouse-over and 2nd+3rd for coordinates associated (and optional 4th for links)
`tit`	(character) title to be displayed on top of figure
`myHtmTit`	(character) title of Html page; 'htmlExt' .. checking and correcting filename-extension (only main Html page)
`myComment`	(character) modify comment embedded in html-document
`textAtStart`	(character) text in html before figure
`textAtEnd`	(character) text in html after figure
`pxDiam`	(integer, length=1) diameter for mouse-over tip to appear (single val or vector), simpler version/solution than with 'Tooltip' package
`addLinks`	(character) for clickable links, either 1) vector of links or 2) single character-chain to be used for pasting to rownames (eg http://www.uniprot.org/uniprot/) or 3) `TRUE` to check presence of 4th name specified in 'colNa' to be useed as columname from 'myCoor' dominates over eventual presence of 4th name in 'colNa'
`linkExt`	(character) if specified : links will get specified ending, define as `NULL` or "" for taking 'addLinks' asIs
`htmlExt`	(character, length=1) extension used when making html files
`callFrom`	(character) allow easier tracking of message(s) produced
`silent`	(logical) suppress messages

Details

Basically theer are two options for defining the path to the image embedded : 1) Absolute path : I turn you can moove the html to different locations, as long as it still can see the png-file the image can be displayed. However, this may not be any more the case when the html file is sent to another person. If the png-file is accessible as url, it should be easily visible. 2) Relative path : The simplest case would be to give only the file-name with no path at all, thus the png-file is supposed to be in the same directory as the html-file. This option is very 'transportable'. Basically the same applies to the clickable links which may be provided. In high-throughput biology one typically points here to data-bases accessible over the internet where urls to specific pages. With UniProt such links can easily be constructed when using protein identifiers as rownames.

Value

plot

See Also

`convertPlotCoordPix`; use `htmlSpecCharConv` to convert special characters for proper html display

Examples

```
## Note, this example writes files to R's tempdir,
## Otherwise, if you simply work in the current directory without spcifying paths you'll
##  get an html with relatove paths, which simply needs the png file in the same path
df1 <- data.frame(id=letters[1:10],x=1:10,y=rep(5,10),mou=paste("point",letters[1:10]),
  link=file.path(tempdir(),paste(LETTERS[1:10],".html",sep="")),stringsAsFactors=FALSE)
## here we'll use R's tempdir, later you may want to choose other locations
pngFile <- file.path(tempdir(),"test01.png")
png(pngFile,width=800, height=600,res=72)
## here we'll just plot a set of horiontal points ...
plot(df1[,2:3],las=1,main="test01")
dev.off()
## Note : Special characters should be converted for display in html pages during mouse-over
library(wrMisc)
df1$mou <- htmlSpecCharConv(df1$mou)
## Let's add the x- and y-coordiates of the points in pixels to the data.frame
df1 <- cbind(df1,convertPlotCoordPix(x=df1[,2],y=df1[,3],plotD=c(800,600),plotRes=72))
head(df1)
## Now make the html-page allowing to display mouse-over to the png made before
htmFile <- file.path(tempdir(),"test01.html")
mouseOverHtmlFile(df1,pngFile,HtmFileNa=htmFile,pxDiam=15,
  textAtStart="Points in the figure are interactive to mouse-over ...",
  textAtEnd="and/or may contain links")
## We still need to make some toy links
for(i in 1:nrow(df1)) cat(paste("point no ",i," : ",df1[i,1]," x=",df1[i,2]," y=",
  df1[i,3],sep=""), file=df1$link[i])
## Now we are ready to open the html file using any browser
## Not run:
browseURL(htmFile)

## End(Not run)
```

`partitionPlot` *Make matrix for layout to partition plotting area*

Description

This function proposes a matrix for use with `layout` to arrange given number of plots to be placed on a page/plotting area. In certain instances the proposed layout may accomodate slightly more plots, eg `nFig=5` can not be arranged in 2 or 3 columns without an empty last spot. Portrait (vertival) or lanscape (horizontal) layout proportions can be chosen. The user can also impose a given number of columns.

Usage

```
partitionPlot(
  nFig,
  returnMatr = TRUE,
  horiz = TRUE,
  figNcol = NULL,
  byrow = TRUE,
  callFrom = NULL
)
```

Arguments

`nFig`	(integer) number of figures to be arrages on single plotting surface (ie window or plotting device)
`returnMatr`	(logical) will return matrix ready for use by `layout`; returns vector with nRow and nCol if `=FALSE`
`horiz`	(logical) will priviledge horizontal layout if `TRUE`
`figNcol`	(integer) optional number of columns
`byrow`	(logical) toggle if output is in order of rows or columns (equivament to `matrix`
`callFrom`	(character) allows easier tracking of messages produced

Value

matrix for use with `layout` or (if `returnMatr=FALSE` numeric vector with number of segements in x- an y-axis)

See Also

`layout`

Examples

```
partitionPlot(5); partitionPlot(14,horiz=TRUE)
```

`plotBy2Groups` *Separate and plot data by 2 groups*

Description

Plot series of data as membership of 2 different grouping vectors (eg by grp=patient and grp2=age-group).

Usage

```
plotBy2Groups(
  dat,
  grp,
  grp2 = NULL,
  col = NULL,
  pch = NULL,
  tit = NULL,
  cex = 2,
  lwd = 0.5,
  lty = 2,
  yLab = NULL,
  cexLab = NULL,
  sepLines = FALSE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`dat`	(numeric) main data (may contain `NA`)
`grp`	(character or factor) grouping of columns of 'dat', eg replicate association
`grp2`	(character or factor) aadditional/secondary grouping of columns of 'dat'
`col`	(character or integer) use custom colors, see also `par`
`pch`	(integer) symbol to mark group-center (see also `par`)
`tit`	(character) custom title
`cex`	(numeric) expansion factor for text (see also `par`)
`lwd`	(integer) line-width (see also `par`)
`lty`	(integer) line-type (see also `par`)
`yLab`	(character) custom y-axis label
`cexLab`	(numeric) expansion factor for labels: 1st value for main groups (`grp`, eg genotypes), 2nd for detailed text (`grp2`, eg animal IDs) (see also `par`)
`sepLines`	(logical) optional drawing of horizontal lines aiming to separate groups (in analogy to support vectors)
`silent`	(logical) suppress messages
`callFrom`	(character) allow easier tracking of message(s) produced

Value

list with `$annot`, `$abund` for initial/raw abundance values and `$quant` with final normalized quantitations, or returns data.frame with annot and quant if `separateAnnot=FALSE`

See Also

`read.table`, `normalizeThis`)

Examples

```
set.seed(2020); rand1 <- round(runif(12),2) +rep(1:3,each=4)
plotBy2Groups(rand1,gl(2,6,labels=LETTERS[5:6]),gl(4,3,labels=letters[1:4]))
```

plotLinReg *Plot linear regression and confidence interval of regression*

Description

This function provides help to display a series of bivariate points given in 'dat' (multiple data formats possible), to model a linear regression and plot the results. Furthermore, a confidence interval to the regression may be added to the plot, regression parameters get be displayed.

Usage

```
plotLinReg(
  dat,
  indepVarLst = NULL,
  dependVar = NULL,
  cusTxt = NULL,
  regrLty = 1,
  regrLwd = 1,
  regrCol = 1,
  confInt = 0.95,
  confCol = NULL,
  xLab = NULL,
  yLab = NULL,
  xLim = NULL,
  yLim = NULL,
  tit = NULL,
  nSignif = 3,
  col = 1,
  pch = 1,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`dat`	(numeric, data.frame or list) main data to plot/inspect. If numeric 'dat' will be used as dependent variable (y-data) together with numeric 'indepVarLst' (independent variable); if list, then list-elments `indepVarLst` and `dependVar` will be used; if matrix, the the 1st and 2nd colum will be used
`indepVarLst`	(character) if 'dat' is list, this designes the list element with the explanatory or independent variable (ie the variable used for explaining, typically x-data)
`dependVar`	(character) if 'dat' is list, this designes the list element with dependent variable (ie the variable to be explained, typically y-data) to test
`cusTxt`	(character) optional custom text to display in subtitle (instead of p-value to H0: slope.regression=0)
`regrLty`	(integer) line type for regression

`regrLwd`	(integer) line width for regression
`regrCol`	(integer) color of regression-line
`confInt`	(numeric, between 0 and 1) the probabiity alpha for the regression interval, if `NULL` no confidence intervall will be plotted/calculated
`confCol`	(character) (background) color for confidence-interval
`xLab`	(character) optional custom x-label
`yLab`	(character) optional custom y-label
`xLim`	(numeric) custom limit for x-axis (see also `par`)
`yLim`	(numeric) custom limit for y-axis (see also `par`)
`tit`	(character) optional title
`nSignif`	(integer) number of significant digits for regression parameters in subtitle of plot
`col`	(integer or character) custom color for points (choose `NULL` for not plotting the actual data)
`pch`	(integer or character) type of symbol for points (see also `par`)
`silent`	(logical) suppress messages
`callFrom`	(character) allows easier tracking of message(s) produced

Value

plot and invisible list containing $data, $linRegr, $confInterval (if calculated)

See Also

`exclExtrValues` for decision of potential outliers; `hist`, `vioplotW`

Examples

```
set.seed(2020); dat1 <- rep(1:6,each=2) +runif(12,0,1)
plotLinReg(dat1, gl(6,2))
# extract elements out of list :
li2 <- list(aa=gl(5,2), bb=dat1[1:10])
plotLinReg(li2, indepVarLst="aa", dependVar="bb")
```

`plotPCAw` *PCA plot with bag-plot to highlight groups*

Description

Function to plot principal components analysis (PCA), with options to show center and potential outliers for each of the groups (columns of data). One of the specificities of this implementation is the integration of bag-plots to better visualize different groups of points (if they can be organized so beforehand as distinct groups) : The main body of data is shown as 'bag-plots' (a bivariate boxplot, see Bagplot) with different transparent colors to highlight the core part of different groups (if they contain more than 2 values per group). Furthermore, group centers are shown as average or median (see 'nGrpForMedian') with stars & index-number (if <25 groups). Layout is automatically set to 2 or 4 subplots (if plotting more than 2 principal components makes sense). Note : This function uses for calulating PCA `prcomp` with default `center=TRUE` and `scale.=FALSE`, (different to princomp() which standardizes by default). Note: `NA`-values cannot (by definition) be processed by PCA - all lines with any non-finite values/content (eg `NA`) will be omitted ! Note : Package RColorBrewer may be used if avaialble. Finally, note that several other packages dedicated to PCA exist, for example FactoMineR offers a very wide spectrum of possibiities, in particular for combined numeric and categorical data.

Usage

```
plotPCAw(
  dat,
  sampleGrp,
  tit = NULL,
  useSymb = c(21:25, 9:12, 3:4),
  center = TRUE,
  scale. = TRUE,
  colBase = NULL,
  useSymb2 = NULL,
  displBagPl = TRUE,
  getOutL = FALSE,
  cexTxt = 1,
  showLegend = TRUE,
  nGrpForMedian = 6,
  pointLabelPar = NULL,
  rowTyName = "genes",
  rotatePC = NULL,
  suplFig = TRUE,
  callFrom = NULL,
  silent = FALSE
)
```

Arguments

`dat`	(matrix, list or data.frame) data to plot. Note: `NA`-values cannot be processed - all lines with non-finite data (eg `NA`) will be omitted !
`sampleGrp`	(character or factor) should be factor describing groups of replicates, NAs are not supported
`tit`	(character) custom title
`useSymb`	(integer) symbols to use (see also `par`)

`center`	(logical or numeric) decide if variables should be shifted to be zero centered, argument passed to `prcomp`
`scale.`	(logical or numeric) decide if scaling to obtain unit variance, argument passed to `prcomp` Alternatively, a vector of length equal the number of columns of x can be supplied. The value is passed to scale.
`colBase`	(character or integer) use custom colors
`useSymb2`	(integer) symbol to mark group-center (no mark of group-center if default NULL) (equivalent to `pch`, see also `par`)
`displBagPl`	(logical) if `TRUE`, show bagPlot (group-center) if >3 points per group otherwise the average-confidence-interval
`getOutL`	(logical) return outlyer samples/values
`cexTxt`	(integer) expansion factor for text (see also `par`)
`showLegend`	(logical) toggle to display legend
`nGrpForMedian`	(integer) decide if group center should be displayed via its average or median value: If group has less than 'nGrpForMedian' values, the average will be used, otherwise the median; if `NULL` no group centers will be displayed
`pointLabelPar`	(character) define formatting for optional labels next to points in main figure (ie PC1 vs PC2); may be `TRUE` or list containing elments 'textLabel','textCol','textCex', 'textOffSet','textAdj' for fine-tuning
`rowTyName`	(character) for subtitle : specify nature of rows (genes, proteins, probesets,...)
`rotatePC`	(integer) optional rotation (by -1) for figure of the principal components specified by index
`suplFig`	(logical) to include plots vs 3rd principal component (PC) and Screeplot
`callFrom`	(character) allow easier tracking of message(s) produced
`silent`	(logical) suppress messages

Value

plot and optional matrix of outlyer-data

See Also

(used in this function for the PCA underneith:) `prcomp`, `princomp`, the package FactoMineR

Examples

```
set.seed(2019); dat1 <- matrix(round(c(rnorm(1000), runif(1000,-0.9,0.9)),2),
  ncol=20, byrow=TRUE) + matrix(rep(rep(1:5,6:2), each=100), ncol=20)
biplot(prcomp(dat1))      # traditional plot
(grp = factor(rep(LETTERS[5:1],6:2)))
plotPCAw(dat1,grp)
```

`staggerdCountsPlot` *Staggered Chart for Ploting Counts to Multiple Leveles of the Threshold used*

Description

The basic idea of this plot is to show how counts data change while shifting a threshold-criterium. At each given threshold the counts are plotted like a staggered bar-chart (or staggered histogram) but without vertical lines to illustrated the almost continuous change from preceedig or following threshold-value. Initially this plot was designed for showing the absolute count-data used when constructing roc-curves (eg using the function `summarizeForROC` of package wrProteo). The main input should furnish the panel of threshold as one column and the coresponding counts data as min 2 columns. The threshold coumns gets specified using the argument `threColumn`, the counts-data may either be specified using argument `countsCol` or be searched using `grep` using column-names containing the text given in argument `varCountNa` with may be combined with a fixed preceeding part given as argument `fixedCountPat`.

Usage

```
staggerdCountsPlot(
  roc,
  threColumn = 1,
  countsCol = NULL,
  fixedCountPat = "n.pos.",
  varCountNa = NULL,
  sortAscending = TRUE,
  vertLine = NULL,
  col = NULL,
  tit = NULL,
  logScale = FALSE,
  las.alph = 2,
  displMaxSpec = TRUE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`roc`	(numeric matrix or data.frame) main input: one column with thresholds and multiple columns of assoicated count data
`threColumn`	(integer or character) to specify the column with threshold-data, in typica proteomics benchmark studies this would be 'alph' (for the statistical test threshold)
`countsCol`	(character of integer, min length=2) choice of column(s) with count-data in 'roc' to be used for display, if not `NULL` will override alternative search of columns using 'varCountNa' and 'fixedCountPat'

`fixedCountPat`	(character) optional pattern to help identifying counts-data: if not `NULL` it will be used as fixed part in column names to get pasted to `varCountNa`. In proteomics benchmark studies this would typically be 'n.pos.'
`varCountNa`	(character) alternative way to select the columns from 'roc': searched using `grep` using column-names containing the text given in argument `varCountNa` with may be combined with a fixed preceeding part given as argument `fixedCountPat` In proteomics benchmark studies this would typically be the species-abbreciations (eg 'H','S','E')
`sortAscending`	(logical) decide if data should be sorted ascending or descending
`vertLine`	(numeric) for optional vertical line, typically used to highlight alpha 0.05
`col`	(character) custom colors, see also `par`
`tit`	(character) cutom title
`logScale`	(logical) display threshld values (x-axis) on log-scale
`las.alph`	(numeric) orientation of label of alpha-cutoff, see also `par`
`displMaxSpec`	(logical) display on right side of figure max count value of contributing group species
`silent`	(logical) suppress messages
`callFrom`	(character) allow easier tracking of message(s) produced

Details

Investigate count data prepared for plotting ROC curves : cumulative counts plot by species (along different statistical test thresholds). Note : Package wrProteo may be used to prepare input (matrix of ROC data).

Value

plot only

See Also

`ecdf`, for preparing input to ROC: function `summarizeForROC` in package wrProteo

Examples

```
set.seed(2019); test1 <- cbind(a=sample.int(n=7,size=50,repl=TRUE),
  b=sample.int(n=11,size=50,repl=TRUE),c=sample.int(n=18,size=50,repl=TRUE))
test1 <- cbind(alph=seq(0,1,length.out=50),a=cumsum(test1[,1]),b=cumsum(test1[,2]),
  c=cumsum(test1[,3]))
staggerdCountsPlot(test1,countsCol=c("a","b","c"))
## example below requires the package wrProteo
```

vioplotW *Violin-plots version W*

Description

This function allows generating Violin plots) using a variety of input formats and offers additional options for colors. Main input may be multiple vectors, a matrix or list of multiple data-elements (entries may be of variable length), individual colors for different sets of data or color-gradients can be specified, and the display of n per set of data was integtated (based on an inspiration from Nabble). It is also possible to plot pairwise half-violins for easier pairwise-comparisons (using `halfViolin="pairwise"`). Many arguments are kept similar to vioplot (this package is not required for this function. Note : Arguments have to be given with full names, lazy evaluation of arguments will not work properly with this function (since '...' is used to capture additional datasets). Note : vioplot offers better options for plotting formulas

Usage

```
vioplotW(
  x,
  ...,
  finiteOnly = TRUE,
  halfViolin = FALSE,
  hh = NULL,
  ylim = NULL,
  nameSer = NULL,
  horizontal = FALSE,
  col = "rainbow",
  border = "black",
  lty = 1,
  tit = NULL,
  las = 1,
  lwd = 1,
  rectCol = "black",
  at = 0,
  add = FALSE,
  wex = 1,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`x`	(matrix, list or data.frame) data to plot, or first series of data
`...`	(numeric) additional sets of data to plot
`finiteOnly`	(logical) eliminate non-finite elements to avoid potential errors (eg when encountering `NA`)

halfViolin	(logical or character) decide with TRUE or FALSE if full or only half of violins should be plotted, if "pairwise" always 2 data-sets will be plotted back-to-back
hh	(numeric, length <4) smoothing parameter (standard deviation to kernel function, if omited anormal optimal smoothing parameter is used); equivalent to argument h in vioplot; see also sm.density
ylim	(NULL or numeric, length=2) custom limit on y-axis, see also par
nameSer	(character) custom label for data-sets or columns (length must match number of data-sets)
horizontal	(logical) orientation of plot
col	(character or integer) custom colors or gradients like 'rainbow', 'grayscale', 'heat.colors', 'topo.colors', 'Spectral' or 'Paired', or you may use colors made by the package colorRamps
border	(character) custom color for figure border
lty	(integer) line-type for linear regression line (see also par)
tit	(character) custom title to figure
las	(integer) orientation of axis labels (see also par)
lwd	(integer) width of line(s) (see also par)
rectCol	(character) color of rectangle
at	(numeric) custom locoation of data-series names, ie the points at which tick-marks are to be drawn, will be passed to axis, it's length ust match the number of data-sets
add	(logical) add to existing plot if TRUE
wex	(integer) relative expansion factor of the violin
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced

Value

figure only

See Also

the package vioplot, sm is used for the density estimation

Examples

```
set.seed(2013)
dat6 <- matrix(round(rnorm(300)+3,1), ncol=6,
 dimnames=list(paste("li",1:50,sep=""), letters[19:24]))
vioplotW(dat6)
## variable number of elements (each n is displayed)
dat6b <- apply(dat6,2,function(x) x[which(x <5)])
dat6b[[4]] <- dat6b[[4]][dat6b[[4]] <4]
vioplotW(dat6b,col="Spectral")
vioplotW(dat6b,col="Spectral",halfViolin="pairwise",horizontal=TRUE)
vioplotW(dat6b,col="Spectral",halfViolin="pairwise",horizontal=FALSE)
```

Index