**Description**

A simple, vectorized function that computes a Red/Blue color for plotting microarray expression data.

**Usage**

```r
GetColor(value, GreenRed=FALSE, DisplayRange=3)
dChip.colors(n)
greenred.colors(n)
```

**Arguments**

- **value**: The vector of expression values.
- **GreenRed**: If TRUE the Green-Red colors are produced, otherwise Red-Blue are produced.
- **DisplayRange**: A parameter controlling the range of value's that will be plotted.
- **n**: An integer saying how many colors to be in the palette.

**Details**

`GetColor` is a simple mapping into RGB land provided by Cheng Li. `dChip.colors` provides functionality similar to that of `topo.colors` for the red–blue colors used for genome plots. `greenred.colors` does the same for the green-black-red gradient.

**Value**

A vector of RGB colors suitable for plotting in R.

**Author(s)**

R. Gentleman, based on an original by C. Li.
Examples

```r
set.seed(10)
x <- rnorm(10)
GetColor(x)
dChip.colors(10)
```

Description

'Makesense' takes either an `ExpressionSet` object or a matrix of gene expressions and will produce a smoothed positive and negative strands for all chromosomes.

Usage

```r
Makesense(expr, lib, ...)
```

Arguments

- `expr`: Either an `ExpressionSet` or a matrix of gene expressions with genes as rows and columns as samples.
- `lib`: The name of the Bioconductor annotation data package that will be used to provide mappings from probes to chromosomal locations, such as `hgu95av2.db` or `hgu133a.db`. If `expr` is an `ExpressionSet`, the argument defaults to the annotation slot of the `ExpressionSet`.
- `...`: Currently, the only optional argument is `f`, the smoother span to be passed to `lowess`. Its value should be in the interval of (0,1). This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness. The default value for this argument is 1/10.

Details

The `expr` argument can either be of class `ExpressionSet` or `matrix`, where the latter represents the matrix of gene expressions.

If the `expr` argument is an `ExpressionSet`, the `lib` argument will use the annotation slot. Users can override this behaviour and supply their own `lib` argument if they wish. If the `ExpressionSet` has no value associated with the annotation slot (which should not happen, but is possible) then the user must supply the `lib` argument manually or the function will throw an error.

Value

A list of 2 components:

- `ans2`: A list, whose components correspond to samples in the same order as appearing in the columns of `expr`. Each component is also a list, named by chromosomes "1"-"22", "X" and "Y". Each named component is again a list with two elements named "posS" and "negS", corresponding to the positive and negative strands of a chromosome, each of which is an object returned by 'lowess'.
- `lib`: A string giving the name of the annotation data package to use. Optional if `expr` is an `ExpressionSet`. 
alongChrom

Author(s)

Robert Gentleman and Xiaochun Li

See Also

plotChr

Examples

library("hgu133a.db")
data(expressionSet133a)
esetobj <- Makesense(exprs(expressionSet133a), "hgu133a")
esetobj2 <- Makesense(expressionSet133a[1:200, ])

alongChrom A function for plotting expression data from an ExpressionSet for a given chromosome.

Description

Given a particular ExpressionSet object, a chromLocation object, and a chromosome name, will plot selected ExpressionSet data using various methods.

Usage

alongChrom(eSet, chrom, specChrom, xlim, whichGenes,
plotFormat=c("cumulative", "local","image"),
xloc=c("equispaced","physical"),
scale=c("none","zscale","rankscale","rangescale","zrobustscale"),
geneSymbols=FALSE, byStrand=FALSE, colors="red", lty=1, type="S", ...)

Arguments

eSet The ExpressionSet object to be used.
chrom The desired chromosome.
specChrom An object of type chromLocation for the species being represented.
xlim A pair of values - either character or integer, which will denote the range of genes to display (based on base pair: either directly in the case of integers, or using the locations of the named genes if character). If not supplied, the entire chromosome is used.
whichGenes If supplied, will limit the displayed genes to the ones provided in this vector.
xloc Determines whether the X axis points (gene names) will be displayed according to their relative position on the chromosome (physical), or spaced evenly (equispaced). Default is equispaced.
plotFormat Determines the method which to plot the data.
scale Determines what method of scaling will be applied to the data. Default is none.
geneSymbols Notes whether to use Affy IDs or Gene Symbols, default is Affy IDs
byStrand
Determines whether to show the entire plot at once, or a split plot by strands. Default is a singular plot

lty
A vector of line types, which will be cycled.

type
Plot type, from par. Defaults to “S”.

colors
A vector of colors for the plots, which will be cycled.

... Any remaining graphics commands may be passed along as per plot()

Details
The genes on the chromosome of interest are extracted from the chromLocation object passed in, which are then intersected with the genes listed in the ExpressionSet. These remaining genes will then be plotted according to the plotFormat argument. If image is specified, an image plot is created showing the expression levels of the samples by gene, using a colour map to denote the levels. If cumulative is chosen, the cumulative expression level is plotted against the genes for each sample. Likewise, if local is used, the raw data is plotted for each sample against the genes using a boxplot format.

Not all parameters are honored for all plotformats. xloc, lty, and type are only used with the cumulative plotformat.

Author(s)
Jeff Gentry

Examples
```r
data(sample.ExpressionSet)
## A bit of a hack to not have a package dependency on hgu95av2
## but need to fiddle w/ the warn level to not fail the example anyways.
curWarn <- options(warn=0)
on.exit(options(warn=curWarn$warn), add=TRUE)
if (require("hgu95av2.db")) {
  z <- buildChromLocation("hgu95av2")
lty <- c(1, 2, 3, 4, 5)
cols <- c("red", "green", "blue", "orange", "magenta", "black")
cols <- cols[sample.ExpressionSet$type]
if (interactive()) {
  par(ask=TRUE)
}

## Here we're using xlim to denote a physical region to display
xlim <- c(87511280,127717880)
for (xl in c("equispaced", "physical"))
  for (sc in c("none","rangescale"))
    alongChrom(sample.ExpressionSet, "1", z, xlim=xlim, xloc=xl,
               plotFormat="cumulative", scale=sc,lty=lty, colors=cols)
}

## Here we're looking for specific genes
which <- c("31540_at","31583_at","31508_at","31529_at","31439_f_at",
          "31729_at")
## Gene "31529_at" does not exist in the current set of genes,
## here it demonstrates how genes not available are dropped.
for (xl in c("equispaced", "physical"))
```

amplicon.plot

for (sc in c("none","rangescale"))
{
    alongChrom(sample.ExpressionSet, "1", z, which=which, xloc=xl,
    plotFormat="cumulative", scale=sc,lty=lty, col=cols)
}

## Do an image plot
for (bs in c(TRUE,FALSE))
    alongChrom(sample.ExpressionSet, "1", z, xlim=xlim, plotFormat="image",
    scale="zscale", byStrand=bs)

## A boxplot
for (st in c(TRUE,FALSE))
    alongChrom(sample.ExpressionSet, "1", z, plotFormat="local",
    colors=cols, byStrand=st)
else print("Example can not be run without the hgu95av2 data package")

amplicon.plot 

Create an amplicon plot

Description

Given a two-sample test statistic and an ExpressionSet this function plots regions of the genome that are either highly expressed (in red) or have low expression (blue) differentially in the two groups.

Usage

amplicon.plot(ESET, FUN, genome)

Arguments

ESET 
an object of class ExpressionSet

FUN 
A two sample test function suitable for esApply.

geno
me 
A character string of the base name for the annotation.

Details

In some genetic studies we are interested in finding regions of the genome where there are a set of highly expressed genes in some subgroup of the population. This set of highly (or lowly) expressed genes is often of great interest. For example in breast cancer the HER–2 gene is on an amplicon. In some patients approximately 5 genes located near HER–2 are all amplified.

These plot should help in the search for such regions.

Value

No value is returned. This function is executed purely for side effect.

Author(s)

Robert Gentleman
cColor

A function for marking specific probes on a cPlot.

Description

Given a set of probes, will highlight them in the color desired on a plot which has already been created via the function cPlot().

Usage

cColor(probes, color, plotChroms, scale=c("relative","max"), glen=0.4)

Arguments

- **probes**: The probes that are being highlighted.
- **color**: The color to highlight the probes.
- **plotChroms**: An object of type chromLocation which contains all the gene information to be plotted.
- **scale**: Whether to plot the graph scaled absolutely or relative by chromosome. Default is absolute.
- **glen**: The length of the gene line plotted.

Details

It is important to call the function cPlot() first. This function will then search for the specific locations of the probes desired, which are contained within the plotChroms instance of a chromLocation class. It will then pass these on to the plotting routine to highlight the desired locations. NOTE: It is important that plotChroms, scale and glen parameters are the same as used for cPlot().

Author(s)

Jeff Gentry

See Also

cPlot, chromLocation-class
cPlot

A plotting function for chromosomes.

Description

Given a chromLocation object, will plot all the gene locations from that object.

Usage

cPlot(plotChroms, useChroms=chromNames(plotChroms),
      scale=c("relative","max"), fg="white", bg="lightgrey",
      glen=0.4, xlab="", ylab="Chromosome", main = organism(plotChroms))

Arguments

plotChroms  An object of type chromLocation which contains all the gene information to be plotted.
useChroms  A vector of chromosome names to be used in the plot. Default is to use all the chromosomes from the plotChroms object.
scale  Passed on to cScale as it’s scale argument. Determines whether the graph is scaled on a relative or absolute basis.
fg  The colour to be used for the genes. Default is white.
bg  The colour to be used for the background of the plot. Defaults to lightgrey.
glen  A scaling factor applied to the plotted length of each gene. Defaults to 0.4 - it is recommended that this not be set larger then 0.5 as it will cause overlap between chromosomes.
xlab  A label for the x axis.
ylab  A label for the y axis.
main  A main label for the plot.

Details

This function will first use the lengths of the chromosomes, stored in the object to create scaling factors for the X axis. Once the scaling factors are determined, the chromLocation object which is passed in is used to determine all the gene locations/strand information/etc, which is then plotted for the user.
cScale

A function for mapping chromosome length to a number of points.

Description

Given a number of points (generally representing the number of points on a plot’s axis), and a vector of chromosome lengths - will generate a vector of the same length as the one passed in containing scaling factors for each chromosome.

Usage

cScale(points, cLengths, method=c("max", "relative"), chrom)

Arguments

points The number of points to scale the chromosome length to.
cLengths A vector of chromosome lengths.
method Determines whether to use relative or absolute scaling. Default is "max" (absolute).
chrom Which chrom to determine the scale for

Details

The scale factor is calculated in a manner based on the method argument. If method is max, the factor is derived by dividing the points argument by each chromosome’s length (in base pairs). If the method chosen is relative, then the scale is determined by dividing the points argument by the maximum chromosome length, and applying that value to each chromosome.
Example

```r
## A bit of a hack to not have a package dependency on hgu95av2
## but need to fiddle w/ the warn level to not fail the example anyways.
curWarn <- options(warn=0)
on.exit(options(warn=curWarn), add=TRUE)
if (require("hgu95av2.db")) {
  z <- buildChromLocation("hgu95av2")
  for (sc in c("max","relative"))
    scale <- cScale(1000, chromLengths(z),sc,"Y")
} else print("This example needs the hgu95av2 data package")
```

### connection-class

**Virtual S4 classes for method dispatching**

**Description**

Virtual S4 classes for method dispatching

**Details**

These two classes are used to simulate the S3 class 'file' extending class 'connection' for method dispatching. They can’t be instantiated!

**Author(s)**

Florian Hahne

### expressionSet133a

**A small dataset for testing**

**Description**

An artificial Affymetrix hgu133a dataset, with one covariate `cov1`.

**Usage**

data(expressionSet133a)

**Format**

The data are artificial. There are 6 cases labeled 1 to 6 and and 22283 genes as in an Affymetrix U133a chips. There is one covariate (factor) whose values are "type 1" for the first 3 samples and "type 2" for the last 3 samples.
Examples

data(expressionSet133a)

groupedHeatmap  
Heatmap of a matrix with grouped rows and columns

Description

The function uses `grid.rect` and `grid.rect` to draw a heatmap with grouped rows and columns.

Usage

```r
groupedHeatmap(z, frow, fcol,
    fillcolours = c("#2166ac","#4393c3","#92c5de","#d1e5f0","#fefefe","#fddbc7","#f4a582","#d6604d","#b2182b"),
    bordercolour = "#e0e0e0",
    zlim = range(z, na.rm=TRUE))
```

Arguments

- `z`  
  A matrix with row and column names.
- `frow`  
  A factor of length `nrow(z)` indicating the row grouping.
- `fcol`  
  A factor of length `ncol(z)` indicating the column grouping.
- `fillcolours`  
  A character vector of colours from which the colour map is obtained through interpolation.
- `bordercolour`  
  Either a character vector of length 1, specifying the border colour of the heatmap tiles, or NULL or NA, which indicates that the border colour should match the fill colour.
- `zlim`  
  Lower and upper limit of `z` values represented in the colour map.

Details

The function can be called within other drawing operations from the grid package, e.g. within a viewport.

Value

The function is called for its side effect, drawing text and rectangles on the current viewport.

Author(s)

Wolfgang Huber [http://www.ebi.ac.uk/huber](http://www.ebi.ac.uk/huber)

See Also

`grid.text`, `grid.rect`
histStack

Examples

```r
data("mtcars")

groupedHeatmap(
  scale(mtcars),
  frow = factor(sapply(strsplit(rownames(mtcars), " "), ",", 1)),
  fcol = factor(round(seq_len(ncol(mtcars))/3)))
```

histStack

Stacked histogram

Description

Stacked histogram

Usage

```r
histStack(x, breaks, breaksFun=paste, ylab="frequency", ...)
```

Arguments

- `x` A list of numeric vectors.
- `breaks` Histogram breaks, as in `hist`
- `breaksFun` Function, can be used to control the formatting of the bin labels. See example.
- `ylab` Label for the Y-axis on the plot
- `...` Further arguments that get passed to `barplot`

Details

The function calls `hist` for each element of `x` and plots the frequencies as a stacked barplot using `barplot` with `beside=FALSE`.

Value

The function is called for its side effect, producing a barplot on the active graphics device. It returns the result of the call to `barplot`.

Author(s)

Wolfgang Huber [http://www.ebi.ac.uk/huber](http://www.ebi.ac.uk/huber)

Examples

```r
x <- list(rnorm(42), rnorm(42)+2)
br <- seq(-3, 5, length=13)
cols <- c("#1D267B", "#ceffc0")
histStack(x, breaks=br, col=cols)

histStack(x, breaks=br, col=cols,
          breaksFun=function(z) paste(signif(z, 3)))
```
Write an HTML IMG tag together with a MAP image map.

Arguments

- **object**: Matrix with 4 columns, specifying the coordinates of the mouse-sensitive region. Each row specifies the corners of a rectangle within the image, in the following order: (left x, lower y, right x, upper y). Note that the point (x=0, y=0) is at the left upper side of the image.
- **con**: Connection to which the image map is written.
- **tags**: Named list whose elements are named character vectors. Names must correspond to node names in `object`. See details.
- **imgname**: Character. Name of the image file (for example PNG file) that contains the plot.

Details

The most important tags are **TITLE**, **HREF**, and **TARGET**. If the list `tags` contains an element with name **TITLE**, then this must be a named character vector containing the tooltips that are to be displayed when the mouse moves over a node. The names of the nodes are specified in the `names` attribute of the character vector and must match those of `object`.

Similarly, **HREF** may be used to specify hyperlinks that the browser can follow when the mouse clicks on a node, and **TARGET** to specify the target browser window.

Currently, only rectangular regions are implemented; the actual shape of the nodes as specified in `object` is ignored. Also, tags for edges of the graph are currently not supported.

This function is typically used with the following sequence of steps:

1. generate your graphic and save it as a bitmap file, e.g., using the **jpeg**, **png**, or **bitmap** device. At this stage, you also need to figure out the pixel coordinates of the interesting regions within your graphic. Since the mapping between device coordinates and pixel coordinates is not obvious, this may be a little tricky. See the examples below, and for a more complex example, see the source code of the function **plotPlate**.

2. open an HTML page for writing and write HTML header, e.g., using the **openHtmlPage** function.

3. Call the **imageMap** function.

4. Optionally, write further text into the HTML connection.

5. Close HTML file, e.g., using the **closeHtmlPage** function.
The function is called for its side effect, which is writing text into the connection `con`.

**Author(s)**

Wolfgang Huber [http://www.dkfz.de/abt0840/whuber](http://www.dkfz.de/abt0840/whuber)

**See Also**

`plotPlate`, `writeLines`

**Examples**

```r
f1 = paste(tempfile(), ".html", sep="")
f2 = paste(tempfile(), ".html", sep="")
fpng = tempfile()

if(capabilities()["png"]) {

## create the image
  colors = c("#E41A1C","#377EB8","#4DAF4A","#984EA3","#FF7F00","#FFFF33","#A65628","#F781BF","#999999")
  width = 512
  height = 256
  png(fpng, width=width, height=height)
  par(mai=rep(0,4))
  plot(0,xlim=c(0,width-1),ylim=c(0,height-1),xaxs="i",yaxs="i",type="n",bty="n")
  cx=floor(runif(100)*(width-1))
  cy=floor(runif(100)*(height-1))
  coord=cbind(cx, cy, cx+10, cy+10)
  rect(coord[,1], height-coord[,2], coord[,3], height-coord[,4],
       col=sample(colors, 100, replace=TRUE))
  text(width/2, height-3, "Klick me!", adj=c(0.5, 1), font=2)
  dev.off()

## create the frame set
  cat("<html><head><title>Hello world</title></head><\n",
       "<frameset rows="280,*" border="0">\n",
       "<frame name="banner" src="file://", f2, ");\n",
       "<frame name="main" scrolling="auto">"\n",
       "</frameset>\n", sep="",file=f1)

## create the image map
  href =sample(c("www.bioconductor.org", "www.r-project.org"),nrow(coord),replace=TRUE)
  title =sample(as.character(packageDescription("geneplotter")),nrow(coord),replace=TRUE)
  con = file(f2, open="w")
  imageMap(coord, con,
            list(HREF=paste("http://", href, sep=""),
                 TITLE=title, TARGET=rep("main", nrow(coord))), fpng)
  close(con)

  cat("Now have a look at file ", f1, " with your browser.\n", sep="")
}
```
Description

This function makes a chromOrd object.

Usage

make.chromOrd(genome, gnames)

Arguments

genome A character string.
gnames A character vector of the genes to be selected.

Details

This function reads in a lot of annotation data and creates a list with one element for each chromosome. The elements of this list are indices indicating the order of the genes that are on that chromosome (and in the annotation data set being used).

Value

A list of chromOrd type. One element for each chromosome. Suitable for reordering other values according to the chromosomal location.

Author(s)

Robert Gentleman

See Also

amplicon.plot

Examples

data(sample.ExpressionSet)
make.chromOrd("hgu95A", featureNames(sample.ExpressionSet))
**multiecdf**

Multiple empirical cumulative distribution functions (ecdf) and densities

**Description**

Plot multiple empirical cumulative distribution functions (ecdf) and densities with user interface similar to that of `boxplot`.

**Usage**

```r
multiecdf(x, 

## S3 method for class 'formula':
multiecdf(formula, data = NULL, xlab, na.action = NULL, 

## S3 method for class 'matrix':
multiecdf(x, xlab, 

## Default S3 method:
multiecdf(x, xlim,
    col=brewer.pal(9, "Set1"),
    main="multiecdf",
    xlab,
    do.points=FALSE,
    subsample=TRUE, 

multidensity(x, 

## S3 method for class 'formula':
multidensity(formula, data = NULL, xlab, na.action = NULL, 

## S3 method for class 'matrix':
multidensity(x, xlab, 

## Default S3 method:
multidensity(x,
    bw="nrd0",
    xlim,
    ylim,
    col = brewer.pal(9, "Set1"),
    main = "multidensity",
    xlab,
    lty = 1L , 
```

**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.
- `na.action` a function which indicates what should happen when the data contain NAs. The default is to ignore missing values in either the response or the group.
- `x` a list of numeric vectors.
- `bw` the smoothing bandwidth, see the manual page for `density`. If `bw` is a character string specifying a rule to choose the bandwidth, this rule is applied to `x[[1]]` and then the same numerical value of `bw` is used throughout.
xlim  Range of the x axis. If missing, the data range is used.
ylim  Range of the y axis. If missing, the range of the density estimates is used.
col, lty  Line colors and line type.
main  Plot title.
xlab  x-axis label.
do.points  logical; if TRUE, also draw points at the knot locations.
subsample  logical; if TRUE, subsamples of size 1000 are used to compute and plot the ecdf for list items with many observations (>1000)
...  Further arguments that get passed on to the plot functions.

Details

The usefulness of multidensity can vary, depending on the data and because of smoothing artifacts. multiecdf will in many cases be preferable.

Value

For the multidensity functions, a list of density objects.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

See Also

boxplot, ecdf density

Examples

f = 1 + (runif(1000)>0.5)
x = rnorm(length(f), mean=f, sd=f)
multiecdf(x~f)
multidensity(x~f)
panel.smoothScatter

Arguments

- **name**: Character. File name (without the extension `.html`).
- **title**: Character. Value of the `title` tag in the HTML header.
- **con**: Connection.

Details

See example.

Value

For `openHtmlPage`, a **connection**.

Author(s)

Wolfgang Huber [http://www.dkfz.de/abt0840/whuber](http://www.dkfz.de/abt0840/whuber)

Examples

```r
fn <- tempfile()
con <- openHtmlPage(fn, "My page")
writeLines("Hello world", con)
closeHtmlPage(con)
readLines(paste(fn, ".html", sep=""))
```

panel.smoothScatter

*Lattice panel function analogous to smoothScatter*

Description

This function allows the user to place `smoothScatter` plots into lattice graphics.

Usage

```r
panel.smoothScatter(x, y = NULL,
  nbin = 64, cuts = 255,
  bandwidth,
  colramp =
  colorRampPalette(c("white", brewer.pal(9, "Blues"))),
  nrpoints = 100,
  transformation = function(x) x^0.25,
  pch = ".",
  cex = 1, col="black",
  range.x,
  ...,
  subscripts)
```
Arguments

- **x**: Numeric vector containing x-values or n by 2 matrix containing x and y values.
- **y**: Numeric vector containing y-values (optional). The length of x must be the same as that of y.
- **nbin**: Numeric vector of length 1 (for both directions) or 2 (for x and y separately) containing the number of equally spaced grid points for the density estimation.
- **cuts**: number of cuts defining the color gradient
- **bandwidth**: Numeric vector: the smoothing bandwidth. If missing, these functions come up with a more or less useful guess. This parameter then gets passed on to the function `bkde2D`.
- **colramp**: Function accepting an integer n as an argument and returning n colors.
- **nrpoints**: Numeric vector of length 1 giving number of points to be superimposed on the density image. The first nrpoints points from those areas of lowest regional densities will be plotted. Adding points to the plot allows for the identification of outliers. If all points are to be plotted, choose nrpoints = Inf.
- **transformation**: Function that maps the density scale to the color scale.
- **pch, cex**: graphical parameters for the nrpoints “outlying” points shown in the display range.x see `bkde2D` for details.
- **col**: points color parameter
- **...**: Further arguments that are passed on to `panel.levelplot`.
- **subscripts**: ignored, but necessary for handling of ... in certain situations. Likely to be removed in future.

Details

This replicates the display part of the `smoothScatter` function by replacing standard graphics calls by grid-compatible ones.

Value

The function is called for its side effects, namely the production of the appropriate plots on a graphics device.

Author(s)

Deepayan Sarkar ⟨deepayan.sarkar@r-project.org⟩

See Also

`smoothScatter`

Examples

```r
ddf <- as.data.frame(matrix(rnorm(40000), ncol = 4) + 3 * rnorm(10000))
ddf[, c(2,4)] <- (-ddf[, c(2,4)])
xyplot(V1 ~ V2 + V3, ddf, outer = TRUE,
       panel = panel.smoothScatter, aspect = "iso")
splom(ddf, panel = panel.smoothScatter, nbin = 64)
```
plotChr

Plot Smoothed Sense/Anti-sense of Specified Chromosomes

Description

For a given chromosome, plot the smooths of the sense and the anti-sense from 5' to 3' (left to right on x-axis).

Usage

plotChr(chrN, senseObj, cols = rep("black", length(senseObj[[1]])), log = FALSE, xloc = c("equispaced", "physical"), geneSymbols = FALSE, ngenes = 20, lines.at = NULL, lines.col = "red")

Arguments

chrN The desired chromosome, e.g. for humans it would be a character string in the set of c(1:22, "X", "Y").

senseObj The result of Makesense.

cols A vector of colors for the lines in the plot, typically specified according to a certain phenotype of samples.

log Logical, whether log-transformation should be taken on the smoothed expressions.

xloc Determines whether the "Representative Genes" will be displayed according to their relative positions on the chromosome (physical), or spaced evenly (equispaced). Default is equispaced.

geneSymbols Logical, whether to use Affy IDs or Gene Symbols for "Representative Genes", default is Affy IDs.

ngenes Desired number of "Representative Genes". The number of actual displayed genes may differ.

lines.at A vector of Affy IDs. Vertical lines will be drawn at specified genes.

lines.col A vector of colors associated with lines.at.

Author(s)

Robert Gentleman and Xiaochun Li

See Also

Makesense

Examples

eexample(Makesense)

if (interactive())
  op <- par(ask=TRUE)

cols <- ifelse(expressionSet133a$cov1="test 1", "red", "green")
plotChr("21", esetobj, cols)

# plot on log-scale:
plotExpressionGraph

A function to plot a graph colored by expression data

Description

Given a graph and expression data for one entity, will plot the graph with the nodes colored according to the expression levels provided.

Usage

plotExpressionGraph(graph, nodeEGmap, exprs, ENTREZIDenvir, mapFun, log = FALSE, nodeAttrs = list(), ...)

Arguments

graph The graph to plot
nodeEGmap A list with element names being node names and the elements being EntrezLink IDs corresponding to those node names.
exprs A vector of expression data, with names being Affymetrix IDs and values being the expression level.
ENTREZIDenvir An environment mapping Affymetrix IDs to EntrezLink IDs, such as the ones provided in the xxx2ENTREZID environments from the Bioconductor data packages (where xxx is a data package).
mapFun A function to map expression levels to colors.
log Whether or not the expression data.
nodeAttrs A list of node attributes, as per plot.graph.
... Any extra arguments to be passed to plot.graph.
**plotExpressionGraph**

**Details**

This function can be used to plot a graph and have the nodes colored according to expression levels provided by the user. The `graph` parameter is a graph object from the `graph` package.

The `nodeEGmap` parameter is a list that maps the nodes of the graphs to EntrezLink IDs. An example of this is the `IMCAEntrezLink` object in the `integrinMediatedCellAdhesion` data set in the `graph` package.

The `exprs` argument is a vector mapping expression levels to Affymetrix IDs. One way to generate an appropriate vector is to extract a single column from an `ExpressionSet`.

The `ENTREZIDenvir` environment maps Affymetrix IDs to EntrezLink IDs. The simplest way to provide this argument is to load the preferred Bioconductor data package (e.g. `hgu95av2.db`) and pass in that package's `xxx2ENTREZID`, where `xxx` is the name of the package.

The `mapFun` function defaults to the function `defMapFun`, which maps nodes to be either blue, green or red depending for expression ranges of 0-100, 101-500, and 501+. In the case where `log` is `TRUE` these ranges are modified with `log2`. Custom versions of this function can be supplied by the user - it must take two parameters, first the expression vector and a boolean value (`log`) specifying if the data has had a `log2` applied to it. The function must return a vector with the same names as the expression vector, but the values of the vector will be color strings.

The `nodeAttrs` list can be specified if any other node attributes are desired to be set by the user. Please see the `plot.graph` man page for more information on this. The other attribute list (`attrs` and `edgeAttrs`) can be passed in via the `...` parameter.

The `IMCAEntrezLink` data structure was created for the purpose of illustrating this program. On Sept 24 2007, the current version of `hgu95av2.db` was used to map from the nodes of IMCA-Graph (in `graph` package) to Entrez identifiers.

**Author(s)**

Jeff Gentry

**See Also**

`plot.graph`, `integrinMediatedCellAdhesion`

**Examples**

```r
if (require("Rgraphviz") && require("hgu95av2.db") &&
    require("fibroEset")) {
  data(integrinMediatedCellAdhesion)
  data(IMCAEntrezLink)
  data(fibroEset)
  attrs <- getDefaultAttrs()
  attrs$graph$rankdir <- "LR"
  plotExpressionGraph(IMCAGraph, IMCAEntrezLink,
    exprs(fibroEset)[,1],
    hgu95av2ENTREZID, attrs = attrs)
}
```
savepng  

Save the contents of the current graphics device to a file

Description

Save the contents of the current graphics device to file

Usage

```r
savepdf(fn, dir, width=6, asp=1)
saveeps(fn, dir, width=6, asp=1)
savepng(fn, dir, width=480, asp=1)
savetiff(fn, dir, density=360, keeppdf=TRUE, ...)
```

Arguments

- `fn` character: name of the output file (without extension). An extension `.pdf`, `.eps`, `.png`, or `.tiff` will be added automatically.
- `dir` character: directory to which the file should be written.
- `width` numeric: width of the image in pixels (png) or inches (pdf, eps).
- `asp` numeric: aspect ratio; height=width*asp.
- `density` pixels per inch (see Details).
- `keeppdf` Should the intermediate PDF file (see Details) be kept? If `FALSE`, it is deleted before the function returns.
- `...` Further arguments that are passed on to `savepdf` (see Details).

Details

The functions are called for their side effect, writing a graphics file.

- `savepdf`, `savepng`, and `saveeps` use the devices `pdf`, `png`, and `postscript`, respectively.
- There is currently no TIFF device for R, so `savetiff` works differently. It relies on the external tool `convert` from the ImageMagick software package. First, `savetiff` produces a PDF files with `savepdf`, then uses `system` to invoke `convert` with the parameter `density`. `savetiff` does not check for the existence of `convert` or the success of the system call, and returns silently no matter what.

Value

Character: name of the file that was written.

Author(s)

Wolfgang Huber [http://www.dkfz.de/abt0840/whuber](http://www.dkfz.de/abt0840/whuber)

See Also

- `dev.copy.pdf`, `png`, `postscript`
smoothScatter

Examples

```r
x = seq(0, 20*pi, len=1000)
plot(x*sin(x), x*cos(x), type="l")

try({
  ## on some machines, some of the devices may not be available
  c(
    savepdf("spiral", dir=tempdir()),
    savepng("spiral", dir=tempdir()),
    saveeps("spiral", dir=tempdir()),
    savetiff("spiral", dir=tempdir())
  )
})
```

smoothScatter produces a smoothed color density representation of the scatterplot, obtained through a kernel density estimate. densCols produces a vector containing colors which encode the local densities at each point in a scatterplot.

Usage

```r
smoothScatter(x, y=NULL,
  nbin=128,
  bandwidth,
  colramp=colorRampPalette(c("white", brewer.pal(9, "Blues"))),
  nrpoints=100,
  transformation=function(x) x^.25,
  xlab=NULL, ylab=NULL, postPlotHook=box,
  pch=".", cex=1,
  xlim, ylim, col="black",
  xaxs=par("xaxs"), yaxs=par("yaxs"), ...)

densCols(x, y=NULL,
  nbin=128,
  bandwidth,
  colramp=colorRampPalette(brewer.pal(9, "Blues")[-(1:3)]))
```

Arguments

- `x` Numeric vector containing x-values or n by 2 matrix containing x and y values.
- `y` Numeric vector containing y-values (optional). The length of `x` must be the same as that of `y`.
- `nbin` Numeric vector of length 1 (for both directions) or 2 (for x and y separately) containing the number of equally spaced grid points for the density estimation.
- `bandwidth` Numeric vector: the smoothing bandwidth. If missing, these functions come up with a more or less useful guess. This parameter then gets passed on to the function `bkde2D`. 
smoothScatter

colramp Function accepting an integer \( n \) as an argument and returning \( n \) colors.

nrpoints Numeric vector of length 1 giving number of points to be superimposed on the density image. The first \( \text{nrpoints} \) points from those areas of lowest regional densities will be plotted. Adding points to the plot allows for the identification of outliers. If all points are to be plotted, choose \( \text{nrpoints} = \text{Inf} \).

transformation Function that maps the density scale to the color scale.

xlab, ylab Character. Gets passed on to \text{image}.

postPlotHook Either NULL or a function with no arguments that will be called after \text{image}.

pch points parameter setting

cex character expansion parameter setting (see \text{par}) passed to the \text{points} phase of the function

xlim, ylim Numeric vectors of length 2, axis limits.

col points color parameter

xaxs, yaxs, ... Further arguments that are passed on to \text{image}.

Details

These functions are convenience wrappers around \text{bkde2D}.

The treatment of the \( x \) and \( y \) arguments attempts to be consistent with that e.g. of \text{plot.default}.

See examples for how to use this function together with \text{pairs}.

Value

\text{smoothScatter} is called for its side-effect, producing a plot on the current graphics device. 
\text{densCols} returns a vector of length \( \text{nrow}(x) \) that contains colors to be used in a subsequent scatterplot. Each color represents the local density around the corresponding point.

Author(s)

Florian Hahne <f.hahne@dkfz.de>

See Also

\text{bkde2D}

Examples

```r
x1 <- matrix(rnorm(1e4), ncol=2)
x2 <- matrix(rnorm(1e4, mean=3, sd=1.5), ncol=2)
x <- rbind(x1,x2)

oldpar <- par(mfrow=c(2,2))
smoothScatter(x, nrpoints=0)
smoothScatter(x)
smoothScatter(x,nrpoints=Inf,colramp=colorRampPalette(RColorBrewer::brewer.pal(9, "YlOrRd")))

colors <- densCols(x)
plot(x, col=colors, pch=20)

# use with pairs:
```
smoothScatter

par(mfrow=c(1,1))
y <- matrix(rnorm(40000), ncol=4) + 3*rnorm(10000)
y[, c(2,4)] <- (-y[, c(2,4)])
pairs(y, panel=function(...) {par(new=TRUE); smoothScatter(..., nrpoints=0)})

par(oldpar)
Index

*Topic IO
  openHtmlPage, 16

*Topic classes
  connection-class, 9

*Topic datasets
  expressionSet133a, 9

*Topic dplot
  Makesense, 2

*Topic error
  savepng, 22

*Topic graphs
  plotExpressionGraph, 20

*Topic hplot
  amplicon.plot, 5
  histStack, 11
  multiecdf, 15
  panel.smoothScatter, 17
  plotChr, 19
  plotExpressionGraph, 20
  smoothScatter, 23

*Topic manip
  GetColor, 1
  imageMap-methods, 12

*Topic programming
  savepng, 22

*Topic utilities
  alongChrom, 3
  cColor, 6
  cPlot, 7
  cScale, 8
  make.chromOrd, 14
  plotExpressionGraph, 20

alongChrom, 3
amplicon.plot, 5, 14

barplot, 11
bkde2D, 18, 23, 24
boxplot, 15, 16
buildACMainLabel (alongChrom), 3

ccolor, 6, 8
chromLocation-class, 6, 8
closeHtmlPage, 12
closeHtmlPage (openHtmlPage), 16
collection, 17
collection-class, 9
cPlot, 6, 7, 9
cScale, 8, 8
cullACXPoints (alongChrom), 3
dChip.colors (getColor), 1
defMapFun (plotExpressionGraph), 20
density, 15, 16
dev.copy, 22
dispACXaxis (alongChrom), 3
doACcumPlot (alongChrom), 3
doACimagePlot (alongChrom), 3
doAClocalPlot (alongChrom), 3
doACmatPlot (alongChrom), 3
edf, 16
emptyACPlot (alongChrom), 3
esApply, 5, 6
expressionSet133a, 9

file-class (connection-class), 9
fixACphysPoints (alongChrom), 3
getAC-closestPos (alongChrom), 3
getACDataEnv (alongChrom), 3
getACExprs (alongChrom), 3
getACGeneSym (alongChrom), 3
getACPlotLabs (alongChrom), 3
getACStrandVals (alongChrom), 3
getColor, 1
getPlotExpressionColors
  (plotExpressionGraph), 20
greenred.colors (getColor), 1
grid.rect, 10
grid.text, 10
groupedHeatmap, 10

highlightACDups (alongChrom), 3
hist, 11
histStack, 11
INDEX

image, 24
imageMap, 12
imageMap (imageMap-methods), 12
imageMap, matrix, connection, list, character-method
  (imageMap-methods), 12
imageMap, matrix-method
  (imageMap-methods), 12
imageMap-methods, 12
IMCAEntrezLink
  (plotExpressionGraph), 20
limitACXRange (alongChrom), 3
log2, 21
make.chromOrd, 6, 14
Makesense, 2, 19
Makesense, ExpressionSet, character-method
  (Makesense), 2
Makesense, ExpressionSet, missing-method
  (Makesense), 2
Makesense, matrix, character-method
  (Makesense), 2
multidensity (multiecdf), 15
multiecdf, 15

openHtmlPage, 12, 16
pairs, 24
panel.levelplot, 18
panel.smoothScatter, 17
par, 24
pdf, 22
plot.default, 24
plot.graph, 21
plotChr, 3, 19
plotExpressionGraph, 20
plotPlate, 12, 13
png, 12, 22
points, 18, 24
postscript, 22

saveeps (savepng), 22
savepdf (savepng), 22
savepng, 22
savetiff (savepng), 22
scaleACData (alongChrom), 3
smoothScatter, 18, 23
system, 22

topo.colors, 1

writeLines, 13