

# Package ‘aveytoolkit’

November 4, 2014

**Title** Compilation of functions for data analysis

**Version** 0.1

**Description** Provides functionality for simple functions like `Pause()`, and `resetPar()` as well as complex functions like `sigHeatmap`, `PlotTimeCourse`, `ComputeTimeLag`, `ggSmartBoxplot`, etc.

**Depends** R ( $\geq 3.0.2$ )

**License** Stefan Avey wrote most of these functions

**LazyData** true

## R topics documented:

AverageReplicates . . . . .	2
aveytoolkit . . . . .	2
browseIndex . . . . .	3
cbind.fill . . . . .	3
collapseDataset . . . . .	4
fishersMethod . . . . .	5
FoldChange . . . . .	6
geomMean . . . . .	7
getLoginDetails . . . . .	8
ggSmartBoxplot . . . . .	8
MakeDF . . . . .	10
makeTransparent . . . . .	11
Pause . . . . .	11
PlotTimeCourse . . . . .	12
PrepareExpression . . . . .	14
ProcessNames . . . . .	15
RepeatBefore . . . . .	16
resetPar . . . . .	17
runLimma . . . . .	18
sigHeatmap . . . . .	19
VennDiagram . . . . .	20
<b>Index</b>	<b>22</b>

AverageReplicates	<i>AverageReplicates</i>
-------------------	--------------------------

---

**Description**

This function averages replicates in a matrix or data.frame

**Usage**

```
AverageReplicates(eSubSet, numRep)
```

**Arguments**

eSubSet	a matrix or data.frame of values with samples as columns
numRep	the number of replicates

**Value**

a data.frame of averaged values with column names coming from the first of each of the replicates with .avg appended

**Note**

Assumes that the replicates are all next to each other

**Author(s)**

Stefan Avey

**Examples**

```
mat <- matrix(rnorm(1000), ncol=10) ## 10 columns of random uniform numbers
avgMat <- AverageReplicates(mat, numRep=2) ## average adjacent pairs of columns
```

---

aveytoolkit	<i>aveytoolkit.</i>
-------------	---------------------

---

**Description**

aveytoolkit.

---

browseIndex

*browseIndex*


---

**Description**

Simple function to open HTML page of index in the default browser

**Usage**

```
browseIndex(package, lib.loc = NULL)
```

**Arguments**

package	a string with the name of package to use
lib.loc	a string of the directory name of the R library, or <code>&lt;e2&gt;&lt;80&gt;&lt;98&gt;NULL&lt;e2&gt;&lt;80&gt;&lt;99&gt;</code> . The default value of <code>&lt;e2&gt;&lt;80&gt;&lt;98&gt;NULL&lt;e2&gt;&lt;80&gt;&lt;99&gt;</code> corresponds to all libraries currently known.

**Details**

Only works for a single package. Could improve to list an HTML page with multiple packages that you could then choose from. Not currently implemented. Borrows the concepts from `utils::browseVignettes`

**Author(s)**

Stefan Avey

**Examples**

```
browseIndex("utils")
```

---

cbind.fill

*resetPar*


---

**Description**

Simple function to combine multiple objects by column while filling in NAs into extra rows created from differing lengths

**Usage**

```
cbind.fill(...)
```

**Arguments**

...	the objects, that will be converted to a list, to bind column-wise
-----	--

**Value**

the cbind'ed objects passed in

**Author(s)**

Dimitris Rizopoulos and Tyler Rinker

**References**

<http://stackoverflow.com/questions/7962267/cbind-a-df-with-an-empty-df-cbind-fill>

**See Also**

[cbind](#)

**Examples**

```
x<-matrix(1:10,5,2)
y<-matrix(1:16, 4,4)
z<-matrix(1:12, 2,6)

cbind.fill(x,y)
cbind.fill(x,y,z)
cbind.fill(mtcars, mtcars[1:10,])
```

---

collapseDataset

*collapseDataset*


---

**Description**

Collapses a dataset from probes to gene symbols.

**Usage**

```
collapseDataset(exprsVals, platform = NULL, mapVector = NULL, oper = max,
  prefer = c("none", "up", "down"), singleProbeset = FALSE,
  returnProbes = FALSE, deProbes = NULL)
```

**Arguments**

exprsVals	a matrix or data.frame of expression values with rownames denoting the probes.
platform	the microarray platform the data comes from for extracting the gene symbols
mapVector	a named character vector with names specifying the current identifiers (probes matching the rownames of exprsVals) and the values of the vector specifying the gene symbols (or other identifier to collapse to).
oper	the operation used to choose which probe when multiple probes map to the same gene. Default is max which will calculate the maximum of the average.
prefer	one of "none", "up", or "down", can be abbreviated.
singleProbeset	If TRUE, the operation applies to the average of each sample. Otherwise, if FALSE, the operation applies to the probesets over all samples and only one probeset will be selected. Default is FALSE.
returnProbes	if TRUE, a list of the collapsed expression matrix and the probes are both returned (see return).
deProbes	a list with named vectors "up" and "down" giving the names of up and downregulated probes

**Details**

If `singleProbeset` is set to `TRUE`, untested and not recommended, the values for each sample will be taken from the maximum across any probe that maps to that gene. This means that a gene's expression values may be a composition of values from different probes rather than a single probe. If `prefer` is "up", when multiple `deProbes` match the same gene, the upregulated will be chosen. Similarly for "down". Default is "none" and the probe with the oper will be chosen.

**Value**

If `returnProbes` is `TRUE`, a list containing the collapsed dataset in `$exprsVals` and the probes chosen in `$probeSets`. Otherwise, if `returnProbes` is `FALSE`, only the expression matrix is returned.

**Author(s)**

Christopher Bolen, Modified by Stefan Avey

**Examples**

```
## ??
```

---

fishersMethod

*fishersMethod*


---

**Description**

This function combines multiple p-values according to Fisher's Method

**Usage**

```
fishersMethod(x)
```

**Value**

a single combined p-value

**Author(s)**

Mike Love

**References**

<http://mikelove.wordpress.com/2012/03/12/combining-p-values-fishers-method-sum-of-p-values-bino>

**Examples**

```
x <- c(runif(1000, 0, 1), runif(100, .1, .2))
fishersMethod(x)
```

---

FoldChange	<i>FoldChange</i>
------------	-------------------

---

**Description**

Calculate the fold change between pairs of conditions in a matrix or data frame

**Usage**

```
FoldChange(x, condNum, condDen, conditions, grouping, log2Transform = FALSE)
```

**Arguments**

x	matrix or data.frame from which to calculate fold changes with samples in columns.
condNum	a vector of condition(s) to be used as the numerator in the fold change calculation
condDen	a vector of condition(s) to be used as the denominator in the fold change calculation
conditions	a vector with length equal to the number of columns of x containing the condition labels between which to find the fold changes.
grouping	a vector with length equal to the number of columns of x containing a grouping of the samples (e.g. subjects, cell lines, strains).
log2Transform	when 'TRUE', log2 transformation will be applied to x before taking the FC. If 'FALSE' (default) no transformation is applied and x is ASSUMED to be already log transformed.

**Details**

FoldChange takes the fold change of log2 Transformed data by subtracting columns of the x dataframe or matrix depending on the conditions passed in.

**Value**

a data.frame of the fold changes with one column for each fold change

**Author(s)**

Stefan Avey

---

`geomMean`*resetPar*

---

**Description**

Calculate the geometric mean

**Usage**

```
geomMean(x, na.rm = TRUE)
```

**Arguments**

`x` a vector of numeric values

`na.rm` (optional) whether to remove NA values before calculation. Default is TRUE

**Details**

This function handles 0 values by ignoring them

**Value**

the geometric mean of `x`

**Author(s)**

Paul McMurdie

**References**

<http://stackoverflow.com/questions/2602583/geometric-mean-is-there-a-built-in>

**See Also**

[exp](#) [sum](#) [log](#)

**Examples**

```
x <- 1:10
x2 <- x^2
geomMean(x)
mean(x)
geomMean(x2)
mean(x2)
```

---

getLoginDetails	<i>getLoginDetails</i>
-----------------	------------------------

---

**Description**

Uses tcltk to display a prompt for a loginID and password

**Usage**

```
getLoginDetails()
```

**Details**

This function displays a window for a user to enter a loginID and password without showing the password.

**Value**

an invisible named vector of loginID and password

**Author(s)**

Markus Gesmann, Barry Rowlingson

**References**

<http://www.r-bloggers.com/simple-user-interface-in-r-to-get-login-details/> <http://r.789695.n4.nabble.com/tkentry-that-exits-after-RETURN-tt854721.html#none>

**See Also**

[tcltk](#)

**Examples**

```
credentials <- getLoginDetails()
## Do what needs to be done with loginID and password
rm(credentials) # Delete credentials
```

---

ggSmartBoxplot	<i>ggSmartBoxplot</i>
----------------	-----------------------

---

**Description**

Boxplot wrapper for ggplot

**Usage**

```
ggSmartBoxplot(x, mat, splitRowBy = NA, splitColBy = NA, colorBy = NULL,
  rows, cols = NA, whichCols = NA, sep = ".", outlier.shape = 17,
  outlier.color = NULL, fileName = NA, ...)
```



**Arguments**

<code>x</code>	the variable to group by for boxplots
<code>mat</code>	data.frame or matrix of values to plot with samples in columns
<code>splitRowBy</code>	a factor used to split the data by row in <code>facet_grid</code>
<code>splitColBy</code>	a factor used to split the data by col in <code>facet_grid</code>
<code>colorBy</code>	a factor used for coloring. No coloring will be done if NULL (default)
<code>rows</code>	row names or row indices of the items to be plotted
<code>cols</code>	substring to search for with "grep" in column names to be plotted
<code>whichCols</code>	the column indices or full column names
<code>sep</code>	a separator used in searching for cols in the column names
<code>outlier.shape</code>	shape of outliers (default is 17, filled triangle)
<code>outlier.color</code>	color of outliers (default is NULL)
<code>fileName</code>	
<code>...</code>	other arguments that are passed to <code>qplot</code>
<code>filename</code>	the name of a file to write a PDF to or NA to plot in standard graphics device.

**Value**

invisibly returns a named list of the data frame(s) used for plotting the boxplot(s). The names come from converting the rows argument to a character vector.

**Author(s)**

Stefan Avey

**See Also**

[ggplot2](#), [qplot](#)

**Examples**

```
data(OrchardSprays)
## Example of functionality
ggSmartBoxplot(x=OrchardSprays$treatment,
               mat=t(OrchardSprays[,1]),
               rows=1, whichCols=1:ncol(t(OrchardSprays)),
               colorBy=factor(OrchardSprays$rowpos+OrchardSprays$colpos > 9),
               xlab="Treatment")

## NOT RUN:
cellType <- "PBMC"
geneSub <- grep("HLA-A29.1", rownames(expr))
age <- "Young"
ages <- c("Young", "Old")
responses <- c("NR", "R")
subset <- targetFClst[[cellType]]$Age %in% ages &
  targetFClst[[cellType]]$Response %in% responses
ggSmartBoxplot(x=targetFClst[[cellType]][subset, "Time"],
               mat=exprFClst[[cellType]], ylim=c(-1,1),
               rows=geneSub, whichCols=which(subset),
```

```
colorBy=targetFClist[[cellType]][subset,"Response"],
splitRowBy=targetFClist[[cellType]][subset,"Age"],
xlab="Days (Post Vaccination)",
fileName=NA)
```

---

MakeDF

*MakeDF*


---

## Description

Creates a data frame from a list. Useful for when the list elements have unequal lengths and `as.data.frame` fails.

## Usage

```
MakeDF(list, names)
```

## Arguments

<code>list</code>	the list to convert
<code>names</code>	the names of the list

## Value

a data frame of the converted list.

## Author(s)

Josh O'Brien

## References

<http://stackoverflow.com/questions/15753091/convert-mixed-length-named-list-to-data-frame>

## See Also

[gsub](#)

## Examples

```
## Test timing with a 50k-item list
ll <- createList(50000)
nms <- c("a", "b", "c")

system.time(makeDF(ll, nms))
# user system elapsed
# 0.47    0.00    0.47
```

---

makeTransparent	<i>makeTransparent</i>
-----------------	------------------------

---

**Description**

Simple function to make some colors transparent

**Usage**

```
makeTransparent(..., alpha = 0.5)
```

**Arguments**

...	vector or list of colors
alpha	transparency factor in range [0,1]

**Value**

a vector of new colors made transparent

**Author(s)**

Ricardo Oliveros-Ramos

**References**

<http://stackoverflow.com/questions/8047668/transparent-equivalent-of-given-color>

**See Also**

[rgb](#), [col2rgb](#)

**Examples**

```
makeTransparent("red", "blue")
##[1] "#FF00007F" "#0000FF7F"
makeTransparent("red", "blue", alpha=0.8)
## [1] "#FF0000CC" "#0000FFCC"
```

---

Pause	<i>Pause</i>
-------	--------------

---

**Description**

This function prompts for return key and waits until the return is pushed to continue execution. It is used often to view plots coded in a loop one at a time allowing the user to control when the next plot should be displayed

**Usage**

```
Pause(str = "continue", quiet = FALSE)
```

**Arguments**

<code>str</code>	optional string to display. Defaults to "continue".
<code>quiet</code>	if TRUE, no prompt is displayed. Default is FALSE

**Details**

The Pause function uses `readline` to wait until a newline character (produced by the Enter key) is given. Instead of pressing Enter, a newline character can be used to automate this waiting time.

**Value**

NULL is returned by invisible

**Author(s)**

Stefan Avey

**See Also**

[readline](#), [invisible](#)

**Examples**

```
for(p in 1:10) {
  plot(-10:10, (-10:10)^p, type=b)
  Pause(paste0("see plot of x^",p+1))
}
```

---

PlotTimeCourse

*PlotTimeCourse*


---

**Description**

Plot helper function for PlotPCATimeCourse

**Usage**

```
PlotTimeCourse(x, y, colors, groups, sampleNames, pch = 19, plotTitle = "",
  legend.loc = "topleft", plotType = c("times", "points"), alpha = 0.15,
  cex.pt = 1, cex.time = 2, time.adj = c(-0.3, -0.3), arrLen = 0.1,
  lwd = 3, numRep = 3, plotFont = NULL, ctrl = TRUE, hourMarks = TRUE,
  legend.cex = 2, ...)
```

**Arguments**

<code>x</code>	x-values for plotting
<code>y</code>	y-values for plotting
<code>colors</code>	named vector specifying colors for each sample
<code>groups</code>	the virus strain names for the conditions of interest
<code>sampleNames</code>	names of the samples

pch	the plotting character. Default is 19 (a closed circle).
plotTitle	a string used for the plotting title
legend.loc	location of the legend. Default is topleft.
plotType	one of "times" or "points". See Details
alpha	transparency factor passed to the alpha function (scales library)
cex.pt	size of points. Default is 1
cex.time	size of time labels. Default is 2
time.adj	the ammount to adjust the time labels. Default is c(-.3, -.3) which moves them to the lower left
arrLen	length of the arrows plotted at the average of each time point. Default is 0.1
lwd	line width. Default is 3
numRep	the number of replicates. Default is 3
plotFont	which font to use for plotting text
ctrl	should the control time points be included?
hourMarks	should the 4 and 8 hour time points be marked on the plot?
legend.cex	size expansion for the legened. Default is 2.
...	other arguments passed to heatmap.2

### Details

If plotType is "times", ??? Also used to plot 2 genes expression against each other over time. If legend.loc is "none", no legend is plotted. ctrl flag indicates whether or not first numRep values in x and y are from a control measurement

### Value

Nothing is returned

### Note

Colors are assumed to have as the names attribute some part of the sampleName which can uniquely identify it.

### Author(s)

Stefan Avey

---

PrepareExpression	<i>PrepareExpression</i>
-------------------	--------------------------

---

## Description

Takes in an expression matrix or data frame and prepares it for further analysis

## Usage

```
PrepareExpression(eset, target, returnProbes = TRUE, labelColumn = "Label",
  select = colnames(target), collapse = ".")
```

## Arguments

eset	expression information and (potentially) other columns
target	target file where the column names of eset can be matched to 'Label'
returnProbes	whether probe mapping should be returned along with expression values in a list. This will only be returned correctly if there is a column of eset matching SYMBOL in any case.
labelColumn	the column name in the target file to use for matching the column names of eset. Default is "Label"
select	the column names of target to select and merge as the new column names of eset
collapse	

## Details

Wrote this to automate the few lines I always perform to "prepare" an expression set for further processing. I always want to remove the symbols column, rename the column names based on the target file, and (usually) change the rownames to be gene symbols. This function takes in the matrix format that I use to store processed expression files (in a package or file).

## Value

if returnProbes is FALSE: a list of the prepared expression data frame (exprDat) and the (potentially modified) target data frame (target) . if returnProbes is TRUE (default): a list of three elements including the two above and probeMap (a vector mapping from gene symbols to probe names).

## Author(s)

Stefan Avey

## Examples

```
## Creating fake expression matrix
dat <- matrix(rnorm(1000, mean=8, sd=1), nrow=100, ncol=10)
colnames(dat) <- sample(letters[1:10], size=10)
fakeGenes <- as.vector(outer(LETTERS[1:26], LETTERS[1:26], paste0))
x <- data.frame(symbol=fakeGenes[1:nrow(dat)], dat, row.names=paste0("Probe_", 1:nrow(dat)))
head(x) # look at first 6 rows of toy data set
target <- data.frame(Label=letters[1:26], Class=rep(1:3, length.out=26))
head(target)
```

```

preList <- PrepareExpression(x, target, select="Class")
head(preList$exprDat)
head(preList$target)
head(preList$probeMap)

## NOT RUN:
library(HIPC)
data(y3ExprP BMC, y3Target)
preList <- PrepareExpression(y3ExprP BMC, y3Target,
                             select=c("Response", "SubjectID", "Age", "Time"))

```

---

ProcessNames	<i>ProcessNames</i>
--------------	---------------------

---

## Description

Cleans up strings to make them pretty names by removing punctuation, whitespace, and specified substrings

## Usage

```
ProcessNames(strs, stringsToRm = NULL, rmPunct = TRUE, sep = "_")
```

## Arguments

<code>strs</code>	vector or strings to process
<code>stringsToRm</code>	a vector or list of strings to search for and remove from <code>strs</code>
<code>rmPunct</code>	should punctuation be removed? Default is TRUE.
<code>sep</code>	character to replace whitespace

## Details

`stringsToRm` are replaced by " " in the order they are given using `gsub`. After this, punctuation is removed if `rmPunct` is TRUE. Then, leading and/or trailing whitespace will be removed and the `sep` will be used to separate words. This function is useful when reading in other people's data and you want to change the row or column names to legal R names or just shorten the names.

## Value

a vector of modified strings from `strs`

## Author(s)

Stefan Avey

## See Also

[gsub](#)

**Examples**

```
badNames <- c("Whos Birthday?", "[Date]", "gift Received")
## Remove the string "Whos", remove punctuation, and separate words by _
goodNames <- ProcessNames(badNames, stringsToRm="Whos", rmPunct=TRUE, sep=_)
goodNames
## Remove the string "Whos", dont remove punctuation, and put no separation between words
goodNames <- ProcessNames(badNames, stringsToRm="Whos", rmPunct=FALSE, sep=)
goodNames
```

---

RepeatBefore

*RepeatBefore*


---

**Description**

Replaces NAs with the latest non-NA value

**Usage**

```
RepeatBefore(x)
```

**Arguments**

`x` a vector of values

**Details**

NA values will be replaced by the most recent value with a lower index. If there is no non-NA value before the NA appears, it will remain NA.

**Value**

a vector of values

**Author(s)**

Ruben

**References**

<http://stackoverflow.com/questions/7735647/replacing-nas-with-latest-non-na-value>

**See Also**

[rep](#)

**Examples**

```
x = c(NA, NA, a, NA, NA, NA, NA, NA, NA, NA, NA, b, c, d, NA, NA, NA, NA, NA, e)
newX <- RepeatBefore(x)
show(newX)
```



---

resetPar	<i>resetPar</i>
----------	-----------------

---

## Description

Simple function to reset plotting parameters for when things get wonky

## Usage

```
resetPar()
```

## Details

This function resets the graphical parameters from the par function. It flashes a new device on the screen but works to reset parameters. Meant to be used when things get hairy and not coded in scripts

## Value

an invisible named list of parameters returned by calling par

## Author(s)

Gavin Simpson

## References

<http://stackoverflow.com/questions/5789982/reset-par-to-the-default-values-at-startup>

## See Also

[par](#)

## Examples

```
par(oma=c(4,10,2,1))
plot(1,1)
## paramter settings werent saved so do a reset
resetPar()
plot(1,1)
```

runLimma

*runLimma***Description**

This function performs a basic LIMMA analysis on the given expression set

**Usage**

```
runLimma(eset, labels, contrasts, covariates = NULL,
  filterReplicateGenes = TRUE, min.fold.change = 1, min.intensity = 4,
  p.cutoff = 0.05, fitOnly = FALSE)
```

**Arguments**

eset	the expression matrix
labels	the labels for each column of the eset
contrasts	Vector of contrasts to make
block	Vector of factors specifying a blocking variable (i.e. for paired samples or for ). NOT IMPLEMENTED!
covariates	data frame of covariates (of same length as labels) to include in the model. Use this if there are paired samples, etc.
filterReplicateGenes	Only include one probeset for each gene (determined by symbol)
min.fold.change	Minimum log2 fold change to be differentially expressed. Default is 1.
min.intensity	Minimum log2 intensity (at any time) to be differentially expressed. Default is 4.
p.cutoff	FDR corrected cutoff for significant differential expression. Default is 0.05.
fitOnly	If true, will return fit2, rather than the matrix of significant genes. Default is FALSE.

**Details**

Generally, an expression matrix is made up of rows of genes (or any other features) and columns of samples. The matrix has data for multiple classes (which are denoted with the 'labels' parameter) and the classes are compared using the vector of contrasts.

**Value**

depends on fitOnly

**Author(s)**

Christopher Bolen

**See Also**

[limma](#)

## Examples

```
## Example:
## If you have a m X 10 matrix eset, with 5 samples of class A and 5 of class B,
## you could compare class A to class B using the following code:
##
## results = runLimma(eset, c(A,A,A,A,A,B,B,B,B,B), "B-A")
##
## This will return to you a matrix with columns for each comparison and rows for each gene.
## The value in each cells will either be -1, 0, or 1, depending on whether the gene is
## significantly higher in B, not significant, or significantly higher in A, respectively.
## If you want information on p-values and fold changes, set "fitOnly=T", and you can access
## the fit object to get the information.
##
## For other comparisons, you can look at the LIMMA user guide.
```

sigHeatmap

*sigHeatmap*

## Description

Draw heatmap with significance indicated on boxes

## Usage

```
sigHeatmap(hm, pvals, pvalDisplayName = "P-value", cutoff = 0.05,
  showOnly = c("both", "positive", "negative", "all"), main = "",
  mainNewlines = 0, sigChar = "*", Rowv = T, hclustMethod = "ward.D",
  ...)
```

## Arguments

hm	a matrix of values used for drawing the heatmap
pvals	a list or data frame of (possibly FDR corrected but this is not handled by the function) positive p-values
pvalDisplayName	is printed on the heatmap as a legend. Default is "P-value" but might want to change to "Q-value", "FDR", etc.
cutoff	is threshold for significance of pvals. Default is 0.05
showOnly	one of "both", "positive", "negative", or "all" can be abbreviated.
main	a string giving the plot main title. Default is "" (i.e. no title is plotted).
mainNewlines	a non-negative integer specifying the number of newline characters to plot before the main title. Used to make the title appear lower on the page. Default is 0
sigChar	the character used for plotting on top of significant boxes
Rowv	should the rows be reordered, passed into heatmap.2
hclustMethod	passed to the function stats::hclust. The agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). Default is "ward.D".
...	other arguments passed to heatmap.2

**Details**

Only rows with at least one significant column are plotted. If `showOnly` is "both", plots both positive and negative significant changes. If `showOnly` is "positive" or "negative", plots only rows of `hm` with significant positive or negative values respectively. If `showOnly` is "all", all rows of `hm` are shown.

**Value**

a vector indicating which of the rows of `hm` were determined to be significant and subsequently plotted

**Author(s)**

Stefan Avey

**Examples**

```
data(mtcars)
x <- as.matrix(mtcars)
alpha <- 10^-7 # significance threshold
## Caculate whether difference from mean is significant
## This is not done correctly but just to have some sort of significance
diffMean <- mtcars-matrix(colMeans(mtcars),
                          ncol=ncol(mtcars), nrow=nrow(mtcars), byrow=TRUE)
stdErr <- matrix(sapply(mtcars, sd)/sqrt(nrow(mtcars)),
                 ncol=ncol(mtcars), nrow=nrow(mtcars), byrow=TRUE)
tstats <- diffMean/stdErr
pvals <- pt(as.matrix(tstats), nrow(mtcars)-2, lower=FALSE)
op <- par(oma=c(4,0,0,20))
sel <- sigHeatmap(x, pvals=pvals, cutoff=alpha, showOnly="b",
                 main="mtcars Example Heatmap", sigChar="*", notecol=black,
                 notecex=2, Colv=T, Rowv=T, dendrogram="row", trace="none")
par(op)
## Which cars werent selected
rownames(mtcars)[setdiff(1:nrow(mtcars), sel)]
```

---

VennDiagram

*sigHeatmap*


---

**Description**

Draw a venn diagram of 2 or 3 sets

**Usage**

```
VennDiagram(setList, mar = c(0, 0, 1, 0), ...)
```

**Arguments**

`setList` a (named) list of the sets to be plotted. The names will be used on the plot. If the list is unnamed, the default names in [vennDiagram](#)

**Details**

Wrapper around the [limma vennDiagram](#) function to make it simpler.

**Value**

a data frame of binary values indicating membership in each set with rownames giving the set entries.

**Author(s)**

Stefan Avey

**References**

Code modified from <http://research.stowers-institute.org/mcm/venn.R>

**See Also**

[vennDiagram](#)

# Index

## \*Topic **aveytoolkit**

AverageReplicates, [2](#)  
browseIndex, [3](#)  
cbind.fill, [3](#)  
collapseDataset, [4](#)  
fishersMethod, [5](#)  
FoldChange, [6](#)  
geomMean, [7](#)  
getLoginDetails, [8](#)  
ggSmartBoxplot, [8](#)  
MakeDF, [10](#)  
makeTransparent, [11](#)  
Pause, [11](#)  
PlotTimeCourse, [12](#)  
PrepareExpression, [14](#)  
ProcessNames, [15](#)  
RepeatBefore, [16](#)  
resetPar, [17](#)  
runLimma, [18](#)  
sigHeatmap, [19](#)  
VennDiagram, [20](#)

as.data.frame, [10](#)  
AverageReplicates, [2](#)  
aveytoolkit, [2](#)  
aveytoolkit-package (aveytoolkit), [2](#)

browseIndex, [3](#)

cbind, [4](#)  
cbind.fill, [3](#)  
col2rgb, [11](#)  
collapseDataset, [4](#)

exp, [7](#)

fishersMethod, [5](#)  
FoldChange, [6](#)

geomMean, [7](#)  
getLoginDetails, [8](#)  
ggplot2, [9](#)  
ggSmartBoxplot, [8](#)  
gsub, [10](#), [15](#)

invisible, [12](#)

limma, [18](#), [21](#)  
log, [7](#)

MakeDF, [10](#)  
makeTransparent, [11](#)

par, [17](#)  
Pause, [11](#)  
PlotTimeCourse, [12](#)  
PrepareExpression, [14](#)  
ProcessNames, [15](#)

qplot, [9](#)

readline, [12](#)  
rep, [16](#)  
RepeatBefore, [16](#)  
resetPar, [17](#)  
rgb, [11](#)  
runLimma, [18](#)

sigHeatmap, [19](#)  
sum, [7](#)

tcltk, [8](#)

VennDiagram, [20](#)  
vennDiagram, [20](#), [21](#)