Package 'bigPint'

April 12, 2022

Version 1.10.0

Title Big multivariate data plotted interactively

Description Methods for visualizing large multivariate datasets using static and interactive scatter-plot matrices, parallel coordinate plots, volcano plots, and litre plots. Includes examples for visualizing RNA-sequencing datasets and differentially expressed genes.

License GPL-3

Depends R (>= 3.6.0)

Imports DelayedArray (>= 0.12.2), dplyr (>= 0.7.2), GGally (>= 1.3.2), ggplot2 (>= 2.2.1), graphics (>= 3.5.0), grDevices (>= 3.5.0), grid (>= 3.5.0), gridExtra (>= 2.3), hexbin (>= 1.27.1), Hmisc (>= 4.0.3), htmlwidgets (>= 0.9), methods (>= 3.5.2), plotly (>= 4.7.1), plyr (>= 1.8.4), RColorBrewer (>= 1.1.2), reshape (>= 0.8.7), shiny (>= 1.0.5), shinycssloaders (>= 0.2.0), shinydashboard (>= 0.6.1), stats (>= 3.5.0), stringr (>= 1.3.1), SummarizedExperiment (>= 1.16.1), tidyr (>= 0.7.0), utils (>= 3.5.0)

VignetteBuilder knitr

Suggests BiocGenerics (>= 0.29.1), data.table (>= 1.11.8), EDASeq (>= 2.14.0), edgeR (>= 3.22.2), gtools (>= 3.5.0), knitr (>= 1.13), matrixStats (>= 0.53.1), rmarkdown (>= 1.10), roxygen2 (>= 3.0.0), RUnit (>= 0.4.32), tibble (>= 1.4.2),

biocViews Clustering, DataImport, DifferentialExpression, GeneExpression, MultipleComparison, Normalization, Preprocessing, QualityControl, RNASeq, Sequencing, Software, Transcription, Visualization

RoxygenNote 7.1.0

BugReports https://github.com/lindsayrutter/bigPint/issues

URL https://github.com/lindsayrutter/bigPint

NeedsCompilation no

LazyData true

Encoding UTF-8

Author Lindsay Rutter [aut, cre],
        Dianne Cook [aut]

Maintainer Lindsay Rutter <lindsayannerutter@gmail.com>

R topics documented:

bigPint	bigPint *package*

Description

bigPint R API

Details

See the README on [GitHub]

---

convertSEPair   *Convert SummarizedExperiment to contain defined treatment pair*

Description

Reduce a SummarizedExperiment object that initially contains more than two treatment groups to now only contain a user-specified subset pair of treatment groups. Note that this function is only necesary for users using the SummarizedExperiment input object (instead of the combination of data and dataMetrics input objects.)

Usage

```
convertSEPair(dataSE, group1, group2)
```

Arguments

dataSE	SUMMARIZEDEXPERIMENT | Summarized experiment format that can be used in lieu of data and dataMetrics
group1	CHARACTER STRING | Name of one treatment group that will remain in the dataSE object
group2	CHARACTER STRING | Name of second treatment group that will remain in the dataSE object

Value

A new dataSE object that is a subset of the input dataSE in that it now only contains the user-specified pair of treatment groups.

Examples

```
# Example: Read in example SummarizedExperiment object that contains three
# treatment groups (S1, S2, and S3). Reduce it to now only contain two
# treatment groups (S1 and S3).

data(se_soybean_cn_sub)
se_soybean_cn_sub_2 <- convertSEPair(se_soybean_cn_sub, "S1", "S3")
```

---

convertSESubsetGenes   *Convert SummarizedExperiment to only contain defined genes*

Description

Reduce a SummarizedExperiment object so that it only now contains a subset of genes.

Usage

```
convertSESubsetGenes(dataSE, geneList)
```

Arguments

dataSE   SUMMARIZEDEXPERIMENT | Summarized experiment format that can be
              used in lieu of data and dataMetrics

geneList   CHARACTER ARRAY | List of gene IDs to remain in dataSE

Value

A new dataSE object that is a subset of the input dataSE in that it now only contains the user-specified list of genes.

Examples

```
# Example: Read in example SummarizedExperiment object that originally
# contains 5604 genes. Reduce it to now only contain ten genes (specifically
# the ones with the lowest FDR).

suppressMessages(library(dplyr))
suppressMessages(library(SummarizedExperiment))
data(se_soybean_ir_sub)
geneList <- as.data.frame(rowData(se_soybean_ir_sub)) %>%
    arrange(N_P.FDR) %>% filter(row_number() <= 10)
geneList <- geneList[,1]
se_soybean_ir_sub_2 <- convertSESubsetGenes(se_soybean_ir_sub, geneList)
```

---

plotClusters   *Plot static parallel coordinate clusters*

Description

Perform hierarchical clustering analysis and visualize results with parallel coordinate plots. Optionally, save gene IDs within each cluster to .rds files for later use.

Usage

```
plotClusters(
  data,
  dataMetrics = NULL,
  dataSE = NULL,
  geneList = NULL,
  geneLists = NULL,
  threshVar = "FDR",
  threshVal = 0.05,
  clusterAllData = TRUE,
  showPairs = TRUE,
  nC = 4,
  colList = rainbow(nC),
  aggMethod = c("ward.D", "ward.D2", "single", "complete", "average", "mcquitty",
    "median", "centroid"),
  yAxisLabel = "Count",
  xAxisLabel = "Sample",
  lineSize = 0.1,
  lineAlpha = 0.5,
  vxAxis = FALSE,
  outDir = tempdir(),
  saveFile = TRUE,
  verbose = FALSE
)
```

Arguments

`data`	DATA FRAME | Read counts
`dataMetrics`	LIST | Differential expression metrics; default NULL
`dataSE`	SUMMARIZEDEXPERIMENT | Summarized experiment format that can be used in lieu of data and dataMetrics; default NULL
`geneList`	CHARACTER ARRAY | Array of ID values of genes to be drawn from data as parallel coordinate lines. Use this parameter if you have predetermined genes to be drawn. These genes will be clustered. Otherwise, use dataMetrics, threshVar, and threshVal to create clusters to be overlaid as parallel coordinate lines; default NULL. See package website for examples
`geneLists`	LIST | List of ID values of genes already clustered to be drawn from data as parallel coordinate lines. Each list item is an array of genes ID values that are already grouped as a cluster. Unlike the singular geneList object, the plural geneLists object is not be clustered. If you instead wish to cluster genes, use dataMetrics, threshVar, and threshVal or geneList to create clusters to be overlaid as parallel coordinate lines; default NULL. See package website for examples
`threshVar`	CHARACTER STRING | Name of column in dataMetrics object that is used to threshold significance; default "FDR"
`threshVal`	INTEGER | Maximum value to threshold significance from threshVar object; default 0.05

clusterAllData	BOOLEAN [TRUE	FALSE]	Create clusters based on the whole dataset and then assign significant genes to those clusters; default is TRUE. If FALSE, create clusters based on just the significant genes. With either option, the side-by-side boxplot will represent the whole dataset (from data input) and the parallel coordinate lines will represent only the significant genes (those that pass threshVal for threshVar)
showPairs	BOOLEAN [TRUE	FALSE]	When more than three treatment groups are present, for each pairwise comparison, show only the results for that pair of treatment groups; default is TRUE. If FALSE, show results for all treatment groups even though clusters and significance are determined in pairwise fashion. Note this parameter will not make a difference when the data only contains two treatment groups
nC	INTEGER	Number of clusters; default 4
colList	CHARACTER ARRAY	List of colors for each cluster; default is rainbow(nC)
aggMethod	CHARACTER STRING ["ward.D"	"ward.D2"	"single"	"complete"	"average"	"mcquitty"	"median"	"centroid"]	The agglomeration method to be used in the hierarchical clustering; default "ward.D"
yAxisLabel	CHARACTER STRING	Vertical axis label; default "Count"
xAxisLabel	CHARACTER STRING	Horizontal axis label; default "Sample"
lineSize	INTEGER	Size of plotted parallel coordinate lines; default 0.1
lineAlpha	INTEGER	Alpha value of plotted parallel coordinate lines, default 0.5
vxAxis	BOOLEAN [TRUE	FALSE]	Flip x-axis text labels to vertical orientation; default FALSE
outDir	CHARACTER STRING	Output directory to save all images; default tempdir()
saveFile	BOOLEAN [TRUE	FALSE]	Save file to outDir; default TRUE
verbose	BOOLEAN [TRUE	FALSE]	Print each cluster from each cluster size into separate files and print the associated IDs of each cluster from each cluster size into separate .rds files; default is FALSE

Value

List of n elements each containing a grid of parallel coordinate plots, where n is the number of treatment pair combinations in the data object. If the saveFile parameter has a value of TRUE, then each grid of parallel coordinate plots is saved to the location specified in the outDir parameter as a JPG file. If the verbose parameter has a value of TRUE, then a JPG file for each parallel coordinate plot in each grid, RDS file containing the superimposed IDs for each parallel coordinate plot in each grid, and the JPG file of each grid of parallel coordinate plots is saved to the location specified in the outDir parameter.

See Also

[hclust](#) <https://lindsayrutter.github.io/bigPint/articles/clusters.html>

Examples

```
# The first set of five examples use data and dataMetrics
# objects as input. The last set of five examples create the same plots now
# using the SummarizedExperiment (i.e. dataSE) object input.

# Example 1: Perform hierarchical clustering of size four using the
# default agglomeration method "ward.D". Cluster only on the genes that have
# FDR < 1e-7 (n = 113) and overlay these genes.

library(grid)
library(matrixStats)
library(ggplot2)
data(soybean_ir_sub)
soybean_ir_sub[,-1] <- log(soybean_ir_sub[-1]+1)
data(soybean_ir_sub_metrics)
colList = c("#00A600FF", rainbow(5)[c(1,4,5)])
ret <- plotClusters(data=soybean_ir_sub,
    dataMetrics = soybean_ir_sub_metrics, nC=4, colList = colList,
    clusterAllData = FALSE, threshVal = 1e-7, saveFile = FALSE)
grid.draw(ret[["N_P_4"]])

# Example 2: Perform the same analysis, only now create the four groups by
# clustering on all genes in the data (n = 5,604). Then, overlay the genes
# that have FDR < 1e-7 (n = 113) into their corresponding clusters.

ret <- plotClusters(data=soybean_ir_sub,
    dataMetrics = soybean_ir_sub_metrics, nC=4, colList = colList,
    clusterAllData = TRUE, threshVal = 1e-7, saveFile = FALSE)
grid.draw(ret[["N_P_4"]])

# Example 3: Perform the same analysis, only now overlay all genes in the
# data by keeping the dataMetrics object as its default value of NULL.

ret <- plotClusters(data=soybean_ir_sub, nC=4, colList = colList,
    clusterAllData = TRUE, saveFile = FALSE)
grid.draw(ret[["N_P_4"]])

# Example 4: Visualization of gene clusters is usually performed on
# standardized data. Here, hierarchical clustering of size four is performed
# using the agglomeration method "average" on standardized data. Only genes
# with FDR < 0.05 are used for the clustering. Only two of the three
# pairwise combinations of treatment groups (S1 and S2; S1 and S3) have any
# genes with FDR < 0.05. The output plots for these two pairs are examined.

data(soybean_cn_sub)
data(soybean_cn_sub_metrics)
soybean_cn_sub_st <- as.data.frame(t(apply(as.matrix(soybean_cn_sub[,-1]),
    1, scale)))
soybean_cn_sub_st$ID <- as.character(soybean_cn_sub$ID)
soybean_cn_sub_st <- soybean_cn_sub_st[,c(length(soybean_cn_sub_st),
    1:length(soybean_cn_sub_st)-1)]
colnames(soybean_cn_sub_st) <- colnames(soybean_cn_sub)
```

```
nID <- which(is.nan(soybean_cn_sub_st[,2]))
soybean_cn_sub_st[nID,2:length(soybean_cn_sub_st)] <- 0
ret <- plotClusters(data=soybean_cn_sub_st,
    dataMetrics = soybean_cn_sub_metrics, nC=4,
    colList = c("#00A600FF", "#CC00FFFF", "red", "darkorange"),
    lineSize = 0.5, lineAlpha = 1, clusterAllData = FALSE,
    aggMethod = "average", yAxisLabel = "Standardized read count",
    saveFile = FALSE)
names(ret)
grid.draw(ret[["S1_S2_4"]])
grid.draw(ret[["S1_S3_4"]])

# Example 5: Run the same analysis, only now set the verbose parameter to
# value TRUE. This will save images of each individual cluster, .rds files
# that contain the IDs within each cluster, and images of the conglomerate
# clusters to outDir (default tempdir()).

## Not run:
plotClusters(data=soybean_cn_sub_st, dataMetrics = soybean_cn_sub_metrics,
  nC=4, colList = c("#00A600FF", "#CC00FFFF", "red", "darkorange"),
  lineSize = 0.5, lineAlpha = 1, clusterAllData = FALSE,
  aggMethod = "average", yAxisLabel = "Standardized read count",
  verbose = TRUE)

## End(Not run)

# Below are the same five examples, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Example 1: Perform hierarchical clustering of size four using the
# default agglomeration method "ward.D". Cluster only on the genes that have
# FDR < 1e-7 (n = 113) and overlay these genes.

## Not run:
library(grid)
library(matrixStats)
library(ggplot2)
data(se_soybean_ir_sub)
assay(se_soybean_ir_sub) <- log(as.data.frame(assay(se_soybean_ir_sub))+1)
colList = c("#00A600FF", rainbow(5)[c(1,4,5)])
ret <- plotClusters(dataSE=se_soybean_ir_sub, nC=4, colList = colList,
    clusterAllData = FALSE, threshVal = 1e-7, saveFile = FALSE)
grid.draw(ret[["N_P_4"]])

## End(Not run)

## Not run:
# Example 2: Perform the same analysis, only now create the four groups by
# clustering on all genes in the data (n = 5,604). Then, overlay the genes
# that have FDR < 1e-7 (n = 113) into their corresponding clusters.

ret <- plotClusters(dataSE=se_soybean_ir_sub, nC=4, colList = colList,
    clusterAllData = TRUE, threshVal = 1e-7, saveFile = FALSE)
```

```
grid.draw(ret[["N_P_4"]])

## End(Not run)

# Example 3: Perform the same analysis, only now overlay all genes in the
# data by setting the rowData() to NULL.

## Not run:
se_soybean_ir_sub_nm <- se_soybean_ir_sub
rowData(se_soybean_ir_sub_nm) <- NULL
ret <- plotClusters(dataSE=se_soybean_ir_sub_nm, nC=4, colList = colList,
    clusterAllData = TRUE, saveFile = FALSE)
grid.draw(ret[["N_P_4"]])

## End(Not run)

# Example 4: Visualization of gene clusters is usually performed on
# standardized data. Here, hierarchical clustering of size four is performed
# using the agglomeration method "average" on standardized data. Only genes
# with FDR < 0.05 are used for the clustering. Only two of the three
# pairwise combinations of treatment groups (S1 and S2; S1 and S3) have any
# genes with FDR < 0.05. The output plots for these two pairs are examined.

## Not run:
data(se_soybean_cn_sub)
se_soybean_cn_sub_st = se_soybean_cn_sub
assay(se_soybean_cn_sub_st) <-as.data.frame(t(apply(as.matrix(as.data.frame(
    assay(se_soybean_cn_sub)))), 1, scale)))
nID <- which(is.nan(as.data.frame(assay(se_soybean_cn_sub_st))[,1]))
assay(se_soybean_cn_sub_st)[nID,] <- 0
ret <- plotClusters(dataSE=se_soybean_cn_sub_st, nC=4,
    colList = c("#00A600FF", "#CC00FFFF", "red", "darkorange"),
    lineSize = 0.5, lineAlpha = 1, clusterAllData = FALSE,
    aggMethod = "average", yAxisLabel = "Standardized read count",
    saveFile = FALSE)
names(ret)
grid.draw(ret[["S1_S2_4"]])
grid.draw(ret[["S1_S3_4"]])

## End(Not run)

# Example 5: Run the same analysis, only now set the verbose parameter to
# value TRUE. This will save images of each individual cluster, .rds files
# that contain the IDs within each cluster, and images of the conglomerate
# clusters to outDir (default tempdir()).

## Not run:
plotClusters(dataSE=se_soybean_cn_sub_st, nC=4,
    colList = c("#00A600FF", "#CC00FFFF", "red", "darkorange"),
    lineSize = 0.5, lineAlpha = 1, clusterAllData = FALSE,
    aggMethod = "average", yAxisLabel = "Standardized read count",
    verbose = TRUE)
```

```
## End(Not run)
```

---

plotLitre   *Plot static litre plots*

Description

Plot static litre plots.

Usage

```
plotLitre(
  data = data,
  dataMetrics = NULL,
  dataSE = NULL,
  geneList = NULL,
  threshVar = "FDR",
  threshVal = 0.05,
  option = c("hexagon", "allPoints"),
  pointSize = 2,
  pointColor = "orange",
  xbins = 10,
  outDir = tempdir(),
  saveFile = TRUE
)
```

Arguments

data	DATA FRAME	Read counts
dataMetrics	LIST	Differential expression metrics; default NULL
dataSE	SUMMARIZEDEXPERIMENT	Summarized experiment format that can be used in lieu of data and dataMetrics; default NULL
geneList	CHARACTER ARRAY	List of ID values of genes to be drawn from data as litre plots. Use this parameter if you have predetermined genes to be drawn. Otherwise, use dataMetrics, threshVar, and threshVal to create genes to be drawn; default NULL
threshVar	CHARACTER STRING	Name of column in dataMetrics object that is used to threshold significance; default "FDR"
threshVal	INTEGER	Maximum value to threshold significance from threshVar object; default 0.05
option	CHARACTER STRING ["hexagon"	"allPoints"]	The background of plot; default "hexagon"
pointSize	INTEGER	Size of plotted points; default 2

pointColor	CHARACTER STRING	Color of gene superimposed on litre plot; default "orange"
xbins	INTEGER	Number of bins partitioning the range of the plot; default 10
outDir	CHARACTER STRING	Output directory to save all plots; default tempdir()
saveFile	BOOLEAN [TRUE	FALSE]	Save file to outDir; default TRUE

Value

List of n elements of litre plots, where n is the number of genes determined to be superimposed through the dataMetrics or geneList parameter. If the saveFile parameter has a value of TRUE, then each of these litre plots is saved to the location specified in the outDir parameter as a JPG file.

Examples

```
# The first set of three examples use data and dataMetrics
# objects as input. The last set of three examples create the same plots now
# using the SummarizedExperiment (i.e. dataSE) object input.

# Example 1: Create litre plots for each of the 61 genes with FDR < 1e-10.
# Examine the first plot (gene "N_P_Glyma.19G168700.Wm82.a2.v1")

data(soybean_ir_sub)
soybean_ir_sub[,-1] <- log(soybean_ir_sub[,-1]+1)
data(soybean_ir_sub_metrics)
ret <- plotLitre(data = soybean_ir_sub,
    dataMetrics = soybean_ir_sub_metrics, threshVal = 1e-10,
    saveFile = FALSE)
length(ret)
names(ret)[1]
ret[[1]]

# Example 2: Create litre plots for each of the five most significant genes
# (low FDR values). View plot for gene "N_P_Glyma.19G168700.Wm82.a2.v1".

geneList = soybean_ir_sub_metrics[["N_P"]][1:5,]$ID
ret <- plotLitre(data = soybean_ir_sub, geneList = geneList,
    pointColor = "deeppink")
names(ret)
ret[["N_P_Glyma.19G168700.Wm82.a2.v1"]]

# Example 3: Create one litre plot for each of the five most significant
# genes (low FDR values). View the plot for gene
# "N_P_Glyma.19G168700.Wm82.a2.v1". Use points instead of the default
# hexagons as the background.

ret <- plotLitre(data = soybean_ir_sub, geneList = geneList,
    pointColor = "deeppink", option = "allPoints")
names(ret)
ret[["N_P_Glyma.19G168700.Wm82.a2.v1"]]

# Below are the same three examples, only now using the
```

```
# SummarizedExperiment (i.e. dataSE) object as input.

# Example 1: Create litre plots for each of the 61 genes with FDR < 1e-10.
# Examine the first plot (gene "N_P_Glyma.19G168700.Wm82.a2.v1")

## Not run:
data(se_soybean_ir_sub)
assay(se_soybean_ir_sub) <- log(as.data.frame(assay(se_soybean_ir_sub))+1)
ret <- plotLitre(dataSE = se_soybean_ir_sub, threshVal = 1e-10,
    saveFile = FALSE)
length(ret)
names(ret)[1]
ret[[1]]

## End(Not run)

# Example 2: Create litre plots for each of the five most significant genes
# (low FDR values). View plot for gene "N_P_Glyma.19G168700.Wm82.a2.v1".

## Not run:
geneList <- as.data.frame(rowData(se_soybean_ir_sub)) %>%
    arrange(N_P.FDR) %>% filter(row_number() <= 5)
geneList <- geneList[,1]
ret <- plotLitre(dataSE = se_soybean_ir_sub, geneList = geneList,
    pointColor = "deeppink")
names(ret)
ret[["N_P_Glyma.19G168700.Wm82.a2.v1"]]

## End(Not run)

# Example 3: Create one litre plot for each of the five most significant
# genes (low FDR values). View the plot for gene
# "N_P_Glyma.19G168700.Wm82.a2.v1". Use points instead of the default
# hexagons as the background.

## Not run:
ret <- plotLitre(dataSE = se_soybean_ir_sub, geneList = geneList,
    pointColor = "deeppink", option = "allPoints")
names(ret)
ret[["N_P_Glyma.19G168700.Wm82.a2.v1"]]

## End(Not run)
```

---

plotLitreApp   *Plot interactive litre plots*

Description

Plot interactive litre plots.

Usage

```
plotLitreApp(
  data = data,
  dataMetrics = dataMetrics,
  dataSE = NULL,
  geneList = NULL,
  pointColor = "orange",
  option = c("hexagon", "allPoints")
)
```

Arguments

`data`	DATA FRAME | Read counts
`dataMetrics`	LIST | Differential expression metrics
`dataSE`	SUMMARIZEDEXPERIMENT | Summarized experiment format that can be used in lieu of data; default NULL
`geneList`	CHARACTER ARRAY | List of gene IDs to be drawn onto the litre. Use this parameter if you have predetermined subset of genes to be drawn. Otherwise, all genes in the data object can be superimposed on the litre plot; default NULL
`pointColor`	CHARACTER STRING | Color of overlaid points on scatterplot matrix; default "orange"
`option`	CHARACTER STRING ["hexagon" | "allPoints"] | The background of plot; default "hexagon"; "allPoints" may be too slow depending on data

Value

A Shiny application that shows a litre plot background and allows users to superimpose the subset of genes determined to be superimposed through the dataMetrics or geneList parameter. The application allows users to order how to sequentially superimpose the genes by columns in the dataMetrics parameter.

Examples

```
# The first pair of examples use data and dataMetrics
# objects as input. The last pair of examples create the same plots now
# using the SummarizedExperiment (i.e. dataSE) object input.

# Example 1: Create an interactive litre plot for the logged data using
# default background of hexagons.

data(soybean_ir_sub)
data(soybean_ir_sub_metrics)
soybean_ir_sub_log <- soybean_ir_sub
soybean_ir_sub_log[,-1] <- log(soybean_ir_sub[,-1]+1)
app <- plotLitreApp(data = soybean_ir_sub_log,
    dataMetrics = soybean_ir_sub_metrics)
if (interactive()) {
    shiny::runApp(app, port = 1234, launch.browser = TRUE)
```

```
}

# Example 2: Repeat the same process, only now plot background data as
# individual points. Note this may be too slow now that all points are drawn
# in the background.

app <- plotLitreApp(data = soybean_ir_sub_log,
    dataMetrics = soybean_ir_sub_metrics, option = "allPoints",
    pointColor = "red")
if (interactive()) {
    shiny::runApp(app)
}

# Below are the same pair of examples, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Example 1: Create an interactive litre plot for the logged data using
# default background of hexagons.

## Not run:
data(se_soybean_ir_sub)
se_soybean_ir_sub_log <- se_soybean_ir_sub
assay(se_soybean_ir_sub_log) <-
    log(as.data.frame(assay(se_soybean_ir_sub_log))+1)
app <- plotLitreApp(dataSE = se_soybean_ir_sub_log)
if (interactive()) {
    shiny::runApp(app, port = 1234, launch.browser = TRUE)
}

## End(Not run)

# Example 2: Repeat the same process, only now plot background data as
# individual points. Note this may be too slow now that all points are
# drawn in the background.

## Not run:
app <- plotLitreApp(dataSE = se_soybean_ir_sub_log, option = "allPoints",
    pointColor = "red")
if (interactive()) {
    shiny::runApp(app)
}

## End(Not run)
```

plotPCP   *Plot static parallel coordinate plots*

Description

Plot static parallel coordinate plots onto side-by-side boxplot of whole dataset.

Usage

```
plotPCP(
  data,
  dataMetrics = NULL,
  dataSE = NULL,
  geneList = NULL,
  threshVar = "FDR",
  threshVal = 0.05,
  lineSize = 0.1,
  lineColor = "orange",
  vxAxis = FALSE,
  outDir = tempdir(),
  saveFile = TRUE,
  hover = FALSE
)
```

Arguments

data	DATA FRAME	Read counts
dataMetrics	LIST	Differential expression metrics; If both geneList and dataMetrics are NULL, then no genes will be overlaid onto the side-by-side boxplot; default NULL
dataSE	SUMMARIZEDEXPERIMENT	Summarized experiment format that can be used in lieu of data and dataMetrics; default NULL
geneList	CHARACTER ARRAY	List of gene IDs to be drawn onto the scatterplot matrix of all data. If this parameter is defined, these will be the overlaid genes to be drawn. After that, dataMetrics, threshVar, and threshVal will be considered for overlaid genes. If both geneList and dataMetrics are NULL, then no genes will be overlaid onto the side-by-side boxplot; default NULL
threshVar	CHARACTER STRING	Name of column in dataMetrics object that is used to threshold significance; default "FDR"
threshVal	INTEGER	Maximum value to threshold significance from threshVar object; default 0.05
lineSize	INTEGER	Line width of parallel coordinate lines; default 0.1
lineColor	CHARACTER STRING	Color of parallel coordinate lines; default "orange"
vxAxis	BOOLEAN [TRUE	FALSE]	Flip x-axis text labels to vertical orientation; default FALSE
outDir	CHARACTER STRING	Output directory to save all plots; default tempdir()
saveFile	BOOLEAN [TRUE	FALSE]	Save file to outDir; default TRUE
hover	BOOLEAN [TRUE	FALSE]	Allow to hover over points to identify IDs; default FALSE

Value

List of n elements of parallel coordinate plots, where n is the number of treatment pair combinations in the data object. The background of each plot is a side-by-side boxplot of the full data object, and

the parallel coordinate lines on each plot are the subset of genes determined to be superimposed
through the dataMetrics or geneList parameter. If the saveFile parameter has a value of TRUE, then
each parallel coordinate plot is saved to the location specified in the outDir parameter as a JPG file.

Examples

```
# The first set of four examples use data and dataMetrics
# objects as input. The last set of four examples create the same plots now
# using the SummarizedExperiment (i.e. dataSE) object input.

# Example 1: Plot the side-by-side boxplots of the whole dataset without
# overlaying any metrics data by keeping the dataMetrics parameter its
# default value of NULL.

data(soybean_ir_sub)
soybean_ir_sub[,-1] = log(soybean_ir_sub[,-1] + 1)
ret <- plotPCP(data = soybean_ir_sub, saveFile = FALSE)
ret[[1]]

# Example 2: Overlay genes with FDR < 1e-4 as orange parallel coordinate
# lines.

data(soybean_ir_sub_metrics)
ret <- plotPCP(data = soybean_ir_sub, dataMetrics = soybean_ir_sub_metrics,
    threshVal = 1e-4, saveFile = FALSE)
ret[[1]]

# Example 3: Overlay the ten most significant genes (lowest FDR values) as
# blue parallel coordinate lines.

geneList = soybean_ir_sub_metrics[["N_P"]][1:10,]$ID
ret <- plotPCP(data = soybean_ir_sub, geneList = geneList, lineSize = 0.3,
    lineColor = "blue", saveFile = FALSE)
ret[[1]]

# Example 4: Repeat this same procedure, only now set the hover parameter to
# TRUE to allow us to hover over blue parallel coordinate lines and
# determine their individual IDs.

ret <- plotPCP(data = soybean_ir_sub, geneList = geneList, lineSize = 0.3,
    lineColor = "blue", saveFile = FALSE, hover = TRUE)
ret[[1]]

# Below are the same four examples, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Example 1: Plot the side-by-side boxplots of the whole dataset without
# overlaying any metrics. We prevent overlaying metrics by setting the
# rowData() to NULL.

## Not run:
data(se_soybean_ir_sub)
se_soybean_ir_sub[,-1] <- log(se_soybean_ir_sub[,-1]+1)
```

```
se_soybean_ir_sub_nm <- se_soybean_ir_sub
rowData(se_soybean_ir_sub_nm) <- NULL
ret <- plotPCP(dataSE = se_soybean_ir_sub_nm, saveFile = FALSE)
ret[[1]]

## End(Not run)

## Not run:
# Example 2: Overlay genes with FDR < 1e-4 as orange parallel coordinate
# lines.

ret <- plotPCP(dataSE = se_soybean_ir_sub, threshVal = 1e-4,
    saveFile = FALSE)
ret[[1]]

## End(Not run)

# Example 3: Overlay the ten most significant genes (lowest FDR values) as
# blue parallel coordinate lines.

## Not run:
geneList <- as.data.frame(rowData(se_soybean_ir_sub)) %>%
    arrange(N_P.FDR) %>% filter(row_number() <= 10)
geneList <- geneList[,1]
ret <- plotPCP(dataSE = se_soybean_ir_sub, geneList = geneList,
    lineSize = 0.3, lineColor = "blue", saveFile = FALSE)
ret[[1]]

## End(Not run)

# Example 4: Repeat this same procedure, only now set the hover parameter to
# TRUE to allow us to hover over blue parallel coordinate lines and
# determine their individual IDs.

## Not run:
ret <- plotPCP(data = soybean_ir_sub, geneList = geneList, lineSize = 0.3,
    lineColor = "blue", saveFile = FALSE, hover = TRUE)
ret[[1]]

## End(Not run)
```

---

plotPCPApp   *Plot interactive parallel coordinate plots*

Description

Plot interactive parallel coordinate plots.

Usage

```
plotPCPApp(data = data, dataSE = NULL, pointColor = "orange")
```

Arguments

data	DATA FRAME | Read counts for parallel coordinate lines
dataSE	SUMMARIZEDEXPERIMENT | Summarized experiment format that can be used in lieu of data; default NULL
pointColor	CHARACTER STRING | Color of overlaid points on scatterplot matrix; default "orange"

Value

A Shiny application that shows a parallel coordinate plot and allows users to draw rectangular areas across samples and remove genes that are not inside these areas. The user can download a file that contains the gene IDs that remain.

Examples

```
# The first example uses data and dataMetrics objects as
# input. The last example creates the same plot now using the
# SummarizedExperiment (i.e. dataSE) object input.

# Example: Create interactive parallel coordinate plot for genes that have
# FDR < 0.01 and logFC < -4. Standardize genes to have an average of zero
# and a standard deviation of one.

data(soybean_ir_sub)
data(soybean_ir_sub_metrics)

# Create standardized version of data
library(matrixStats)
soybean_ir_sub_st = as.data.frame(t(apply(as.matrix(soybean_ir_sub[,-1]), 1,
    scale)))
soybean_ir_sub_st$ID = as.character(soybean_ir_sub$ID)
soybean_ir_sub_st = soybean_ir_sub_st[,c(length(soybean_ir_sub_st),
    1:length(soybean_ir_sub_st)-1)]
colnames(soybean_ir_sub_st) = colnames(soybean_ir_sub)
nID = which(is.nan(soybean_ir_sub_st[,2]))
soybean_ir_sub_st[nID,2:length(soybean_ir_sub_st)] = 0

library(dplyr, warn.conflicts = FALSE)
plotGenes = filter(soybean_ir_sub_metrics[["N_P"]], FDR < 0.01,
    logFC < -4) %>% select(ID)
pcpDat = filter(soybean_ir_sub_st, ID %in% plotGenes[,1])

app <- plotPCPApp(data = pcpDat, pointColor = "purple")
if (interactive()) {
    shiny::runApp(app, display.mode = "normal")
}
```

```
# Below is the same example, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Example: Create interactive parallel coordinate plot for genes that have
# FDR < 0.01 and logFC < -4. Standardize genes to have an average of zero
# and a standard deviation of one.

## Not run:
data(se_soybean_ir_sub)

# Create standardized version of data
library(matrixStats)
se_soybean_ir_sub_st = se_soybean_ir_sub
assay(se_soybean_ir_sub_st) <-as.data.frame(t(apply(as.matrix(as.data.frame(
    assay(se_soybean_ir_sub)))), 1, scale)))
nID <- which(is.nan(as.data.frame(assay(se_soybean_ir_sub_st))[,1]))
assay(se_soybean_ir_sub_st)[nID,] <- 0

# To subset our SummarizedExperiment data by a list of genes, we can
# invoke the convertSESubsetGenes() method.

library(dplyr, warn.conflicts = FALSE)
geneList <- as.data.frame(rowData(se_soybean_ir_sub_st)) %>%
    filter(N_P.FDR <= 0.01) %>% filter(N_P.logFC < -4)
geneList <- geneList[,1]
pcpDat <- convertSESubsetGenes(se_soybean_ir_sub_st, geneList)

app <- plotPCPApp(dataSE = pcpDat, pointColor = "purple")
if (interactive()) {
    shiny::runApp(app, display.mode = "normal")
}

## End(Not run)
```

---

plotSM   *Plot static scatterplot matrices*

Description

Plot static scatterplot matrix. Optionally, superimpose differentially expressed genes (DEGs) onto scatterplot matrix.

Usage

```
plotSM(
  data = data,
  dataMetrics = NULL,
  dataSE = NULL,
  geneList = NULL,
```

```
    threshVar = "FDR",
    threshVal = 0.05,
    option = c("allPoints", "foldChange", "orthogonal", "hexagon"),
    xbins = 10,
    threshFC = 3,
    threshOrth = 3,
    pointSize = 0.5,
    pointColor = "orange",
    outDir = tempdir(),
    saveFile = TRUE
)
```

Arguments

data	DATA FRAME | Read counts
dataMetrics	LIST | Differential expression metrics; default NULL
dataSE	SUMMARIZEDEXPERIMENT | Summarized experiment format that can be used in lieu of data and dataMetrics; default NULL
geneList	CHARACTER ARRAY | List of gene IDs to be drawn onto the scatterplot matrix of all data. Use this parameter if you have predetermined genes to be drawn. Otherwise, use dataMetrics, threshVar, and threshVal to create genes to be drawn onto the scatterplot matrix; default NULL; used in "hexagon" and "allPoints"
threshVar	CHARACTER STRING | Name of column in dataMetrics object that is used to threshold significance; default "FDR"; used in all options
threshVal	INTEGER | Maximum value to threshold significance from threshVar object; default 0.05; used in all options
option	CHARACTER STRING ["foldChange" | "orthogonal" | "hexagon" | "allPoints"] | The type of plot; default "allPoints"
xbins	INTEGER | Number of bins partitioning the range of the plot; default 10; used in option "hexagon"
threshFC	INTEGER | Threshold of fold change; default 3; used in option "foldChange"
threshOrth	INTEGER | Threshold of orthogonal distance; default 3; used in option "orthogonal"
pointSize	INTEGER | Size of plotted points; default 0.5; used for DEGs in "hexagon" and "allPoints" and used for all points in "foldChange" and "orthogonal"
pointColor	CHARACTER STRING | Color of overlaid points on scatterplot matrix; default "orange"; used for DEGs in "hexagon" and "allPoints" and used for all points in "foldChange" and "orthogonal"
outDir	CHARACTER STRING | Output directory to save all plots; default tempdir(); used in all options
saveFile	BOOLEAN [TRUE | FALSE] | Save file to outDir; default TRUE; used in all options

Details

There are seven options:

- "foldChange": Plots DEGs onto scatterplot matrix of fold changes
- "orthogonal": Plots DEGs onto scatterplot matrix of orthogonal distance
- "hexagon": Plot DEGs onto scatterplot matrix of hexagon binning
- "allPoints": Plot DEGs onto scatterplot matrix of all data points

Value

List of n elements of scatterplot matrices, where n is the number of treatment pair combinations in the data object. The subset of genes that are superimposed are determined through the dataMetrics or geneList parameter. If the saveFile parameter has a value of TRUE, then each of these scatterplot matrices is saved to the location specified in the outDir parameter as a JPG file.

Examples

```
# The first set of six examples use data and dataMetrics
# objects as input. The last set of six examples create the same plots now
# using the SummarizedExperiment (i.e. dataSE) object input.

# Read in data and metrics (need for first set of six examples)
data(soybean_cn_sub)
data(soybean_cn_sub_metrics)
data(soybean_ir_sub)
data(soybean_ir_sub_metrics)

# Create standardized version of data (need for first set of six examples)
library(matrixStats)
library(ggplot2)
soybean_cn_sub_st <- as.data.frame(t(apply(as.matrix(soybean_cn_sub[,-1]),
    1, scale)))
soybean_cn_sub_st$ID <- as.character(soybean_cn_sub$ID)
soybean_cn_sub_st <- soybean_cn_sub_st[,c(length(soybean_cn_sub_st),
    1:length(soybean_cn_sub_st)-1)]
colnames(soybean_cn_sub_st) <- colnames(soybean_cn_sub)
nID <- which(is.nan(soybean_cn_sub_st[,2]))
soybean_cn_sub_st[nID,2:length(soybean_cn_sub_st)] <- 0

# Example 1: Plot scatterplot matrix of points. Saves three plots to outDir
# because saveFile equals TRUE by default.

## Not run:
plotSM(soybean_cn_sub, soybean_cn_sub_metrics)

## End(Not run)

# Example 2: Plot scatterplot matrix of points. Return list of plots so user
# can tailor them (such as add title) and does not save to outDir because
# saveFile equals FALSE.
```

```
ret <- plotSM(soybean_cn_sub, soybean_cn_sub_metrics, pointColor = "pink",
    saveFile = FALSE)
# Determine names of plots in returned list
names(ret)
ret[["S1_S2"]] + ggtitle("S1 versus S2")
ret[["S1_S3"]] + ggtitle("S1 versus S3")
ret[["S2_S3"]] + ggtitle("S2 versus S3")

# Example 3: Plot standardized data as scatterplot matrix of points.

ret <- plotSM(soybean_cn_sub_st, soybean_cn_sub_metrics,
    pointColor = "#00C379", saveFile = FALSE)
ret[[1]] + xlab("Standardized read counts") +
ylab("Standardized read counts")

# Example 4: Plot scatterplot matrix of hexagons.

ret <- plotSM(soybean_cn_sub, soybean_cn_sub_metrics, option = "hexagon",
    xbins = 5, pointSize = 0.1, saveFile = FALSE)
ret[[2]]

# Example 5: Plot scatterplot matrix of orthogonal distance on the logged
# data, first without considering the metrics dataset and then considering
# it.

soybean_ir_sub[,-1] <- log(soybean_ir_sub[,-1] + 1)
ret <- plotSM(soybean_ir_sub, option = "orthogonal", threshOrth = 2.5,
    pointSize = 0.2, saveFile = FALSE)
ret[[1]]
ret <- plotSM(soybean_ir_sub, soybean_ir_sub_metrics, option = "orthogonal",
    threshOrth = 2.5, pointSize = 0.2, saveFile = FALSE)
ret[[1]]

# Example 6: Plot scatterplot matrix of fold change.

ret <- plotSM(soybean_cn_sub, soybean_cn_sub_metrics, option = "foldChange",
    threshFC = 0.5, pointSize = 0.2, saveFile = FALSE)
ret[[1]]

# Below are the same six examples, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Read in data and metrics (need for first set of six examples)
data(se_soybean_cn_sub)
data(se_soybean_ir_sub)

# Create standardized version of data (need for first set of six examples)
library(matrixStats)
library(ggplot2)
library(SummarizedExperiment)
se_soybean_cn_sub_st = se_soybean_cn_sub
assay(se_soybean_cn_sub_st, withDimnames=FALSE) <-as.data.frame(t(
    apply(as.matrix(as.data.frame(assay(se_soybean_cn_sub))), 1, scale)))
```

```
nID <- which(is.nan(as.data.frame(assay(se_soybean_cn_sub_st))[,1]))
assay(se_soybean_cn_sub_st, withDimnames=FALSE)[nID,] <- 0

# Example 1: Plot scatterplot matrix of points. Saves three plots to outDir
# because saveFile equals TRUE by default.

## Not run:
plotSM(dataSE = se_soybean_cn_sub)

## End(Not run)

# Example 2: Plot scatterplot matrix of points. Return list of plots so user
# can tailor them (such as add title) and does not save to outDir because
# saveFile equals FALSE.

## Not run:
ret <- plotSM(dataSE = se_soybean_cn_sub, pointColor = "pink",
    saveFile = FALSE)
# Determine names of plots in returned list
names(ret)
ret[["S1_S2"]] + ggtitle("S1 versus S2")
ret[["S1_S3"]] + ggtitle("S1 versus S3")
ret[["S2_S3"]] + ggtitle("S2 versus S3")

## End(Not run)

# Example 3: Plot standardized data as scatterplot matrix of points.

## Not run:
ret <- plotSM(dataSE = se_soybean_cn_sub_st, pointColor = "#00C379",
    saveFile = FALSE)
ret[[1]] + xlab("Standardized read counts") +
ylab("Standardized read counts")

## End(Not run)

# Example 4: Plot scatterplot matrix of hexagons.

## Not run:
ret <- plotSM(dataSE = se_soybean_cn_sub, option = "hexagon", xbins = 5,
    pointSize = 0.1, saveFile = FALSE)
ret[[2]]

## End(Not run)

# Example 5: Plot scatterplot matrix of orthogonal distance on the logged
# data, first without considering the metrics dataset and then considering
# it.

## Not run:
assay(se_soybean_ir_sub) <- log(as.data.frame(assay(se_soybean_ir_sub))+1)
ret <- plotSM(dataSE = se_soybean_ir_sub, option = "orthogonal",
    threshOrth = 2.5, pointSize = 0.2, saveFile = FALSE)
```

```
ret[[1]]

## End(Not run)

# Example 6: Plot scatterplot matrix of fold change.

## Not run:
ret <- plotSM(dataSE = se_soybean_cn_sub, option = "foldChange",
    threshFC = 0.5, pointSize = 0.2, saveFile = FALSE)
ret[[1]]

## End(Not run)
```

---

plotSMApp   *Plot interactive scatterplot matrices*

Description

Plot interactive scatterplot matrices.

Usage

```
plotSMApp(data = data, dataSE = NULL, xbins = 10)
```

Arguments

data	DATA FRAME	Read counts
dataSE	SUMMARIZEDEXPERIMENT	Summarized experiment format that can be used in lieu of data; default NULL
xbins	INTEGER	Number of bins partitioning the range of the plot; default 10

Value

A Shiny application that shows a scatterplot matrix with hexagon bins and allows users to click on hexagon bins to determine how many genes they each contain. The user can download a file that contains the gene IDs that are located in the clicked hexagon bin.

Examples

```
# The first example uses data and dataMetrics objects as
# input. The last example creates the same plots now using the
# SummarizedExperiment (i.e. dataSE) object input.

# Example: Create interactive scatterplot matrix for first two treatment
# groups of data.

data(soybean_cn_sub)
soybean_cn_sub <- soybean_cn_sub[,1:7]
```

```
app <- plotSMApp(data=soybean_cn_sub)
if (interactive()) {
    shiny::runApp(app)
}

# Below are the same example, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Example: Create interactive scatterplot matrix for first two treatment
# groups of data. When working with the SummarizedExperiment data,
# we can summon the method convertSEPair() to subset our input data
# to only a pair of treatment groups.

## Not run:
data(se_soybean_cn_sub)
se_soybean_cn_sub <- convertSEPair(se_soybean_cn_sub, "S1", "S2")
app <- plotSMApp(dataSE=se_soybean_cn_sub)
if (interactive()) {
    shiny::runApp(app)
}

## End(Not run)
```

---

plotVolcano   *Plot static volcano plot*

Description

Plot static volcano plot.

Usage

```
plotVolcano(
  data = data,
  dataMetrics = dataMetrics,
  dataSE = NULL,
  geneList = NULL,
  threshVar = "FDR",
  threshVal = 0.05,
  option = c("hexagon", "allPoints"),
  logFC = "logFC",
  PValue = "PValue",
  xbins = 10,
  pointSize = 0.5,
  pointColor = "orange",
  outDir = tempdir(),
  saveFile = TRUE,
  hover = FALSE
)
```

Arguments

data	DATA FRAME	Read counts
dataMetrics	LIST	Differential expression metrics. This object must contain one column with magnitude changes (for the logFC parameter) and one column with statistical values (for the PValue parameter), unless geneList is not NULL
dataSE	SUMMARIZEDEXPERIMENT	Summarized experiment format that can be used in lieu of data and dataMetrics; default NULL
geneList	CHARACTER ARRAY	List of gene IDs to be drawn onto the scatterplot matrix of all data. Use this parameter if you have predetermined subset of genes to be superimposed. Otherwise, dataMetrics, threshVar, and threshVal will be used to create genes to be superimposed onto the volcano plot; default NULL
threshVar	CHARACTER STRING	Name of column in dataMetrics object that is used to determine genes to be overlaid; default "FDR"
threshVal	INTEGER	Maximum value to threshold significance from threshVar object; default 0.05
option	CHARACTER STRING ["hexagon"	"allPoints"]	The background of plot; default "hexagon"
logFC	CHARACTER STRING	Name of column in dataMetrics object that contains log fold change values; default "logFC"
PValue	CHARACTER STRING	Name of column in dataMetrics object that contains p-values; default "PValue"
xbins	INTEGER	Number of bins partitioning the range of the plot; default 10
pointSize	INTEGER	Size of plotted points; default 0.5
pointColor	CHARACTER STRING	Color of overlaid points on scatterplot matrix; default "orange"
outDir	CHARACTER STRING	Output directory to save all plots; default tempdir()
saveFile	BOOLEAN [TRUE	FALSE]	Save file to outDir; default TRUE
hover	BOOLEAN [TRUE	FALSE]	Allow to hover over points to identify IDs; default FALSE

Value

List of n elements of volcano plots, where n is the number of treatment pair combinations in the data object. The subset of genes that are superimposed are determined through the dataMetrics or geneList parameter. If the saveFile parameter has a value of TRUE, then each of these volcano plots is saved to the location specified in the outDir parameter as a JPG file.

Examples

```
# The first set of four examples use data and dataMetrics objects as
# input. The last set of four examples create the same plots now
# using the SummarizedExperiment (i.e. dataSE) object input.

# Example 1: Plot volcano plot with default settings for overlaid points
```

```
# (FDR < 0.05).

data(soybean_ir_sub)
data(soybean_ir_sub_metrics)
ret <- plotVolcano(soybean_ir_sub, soybean_ir_sub_metrics, pointSize = 1,
    saveFile = FALSE)
ret[[1]]

# Example 2: Plot volcano plot and overlay points with PValue < 1e-15.

ret <- plotVolcano(soybean_ir_sub, soybean_ir_sub_metrics,
    pointColor = "red", pointSize = 1, threshVar = "PValue",
    threshVal = 1e-15, saveFile = FALSE)
ret[[1]]

# Example 3: Plot volcano plot and overlay points with PValue < 1e-15. This
# time, plot all points (instead of hexagons) for the background.

ret <- plotVolcano(soybean_ir_sub, soybean_ir_sub_metrics,
    pointColor = "red", pointSize = 1, threshVar = "PValue",
    threshVal = 1e-15, option = "allPoints", saveFile = FALSE)
ret[[1]]

# Example 4: Plot volcano plot with points in background and overlay points
# with PValue < 1e-15. This time, use a value of TRUE for the hover
# parameter so that you can hover over overlaid points and determine their
# IDs.

ret <- plotVolcano(soybean_ir_sub, soybean_ir_sub_metrics,
    pointColor = "red", pointSize = 1, threshVar = "PValue",
    threshVal = 1e-15, option = "allPoints", saveFile = FALSE,
    hover = TRUE)
ret[[1]]

# Below are the same four examples, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Example 1: Plot volcano plot with default settings for overlaid points
# (FDR < 0.05).

## Not run:
data(se_soybean_ir_sub)
ret <- plotVolcano(dataSE = se_soybean_ir_sub, pointSize = 1,
    saveFile = FALSE)
ret[[1]]

## End(Not run)

# Example 2: Plot volcano plot and overlay points with PValue < 1e-15.

## Not run:
ret <- plotVolcano(dataSE = se_soybean_ir_sub, pointColor = "red",
    pointSize = 1, threshVar = "PValue", threshVal = 1e-15,
```

```
    saveFile = FALSE)
ret[[1]]

## End(Not run)

# Example 3: Plot volcano plot and overlay points with PValue < 1e-15. This
# time, plot all points (instead of hexagons) for the background.

## Not run:
ret <- plotVolcano(dataSE = se_soybean_ir_sub, pointColor = "red",
    pointSize = 1, threshVar = "PValue", threshVal = 1e-15,
    option = "allPoints", saveFile = FALSE)
ret[[1]]

## End(Not run)

# Example 4: Plot volcano plot with points in background and overlay points
# with PValue < 1e-15. This time, use a value of TRUE for the hover
# parameter so that you can hover over overlaid points and determine their
# IDs.

## Not run:
ret <- plotVolcano(dataSE = se_soybean_ir_sub, pointColor = "red",
    pointSize = 1, threshVar = "PValue", threshVal = 1e-15,
    option = "allPoints", saveFile = FALSE, hover = TRUE)
ret[[1]]

## End(Not run)
```

---

plotVolcanoApp   *Plot interactive volcano plots*

Description

Plot interactive volcano plots.

Usage

```
plotVolcanoApp(
  data = data,
  dataMetrics = dataMetrics,
  dataSE = NULL,
  option = c("hexagon", "allPoints"),
  pointColor = "orange"
)
```

Arguments

data	DATA FRAME | Read counts
dataMetrics	LIST | Differential expression metrics. This object must contain one column named "logFC" and one column named "PValue".
dataSE	SUMMARIZEDEXPERIMENT | Summarized experiment format that can be used in lieu of data; default NULL
option	CHARACTER STRING ["hexagon" | "allPoints"] | The background of plot; default "hexagon"
pointColor	CHARACTER STRING | Color of overlaid points on scatterplot matrix; default "orange"

Value

A Shiny application that shows a volcano plot and allows users to overlay genes depending on two values, usually a statistical value (such as P-value) and a magnitude change value (such as log fold change). The user can download a file that contains the gene IDs that pass these thresholds.

Examples

```
# The first pair of examples use data and dataMetrics objects as input.
# The last pair of examples create the same plots now using the
# SummarizedExperiment (i.e. dataSE) object input.

# Example 1: Create interactive volcano plot of logged data using hexagon
# bins for the background.

data(soybean_cn_sub)
data(soybean_cn_sub_metrics)
app <- plotVolcanoApp(data = soybean_cn_sub,
    dataMetrics = soybean_cn_sub_metrics)
if (interactive()) {
    shiny::runApp(app)
}

# Example 2: Create interactive volcano plot of logged data using points for
# the background.

app <- plotVolcanoApp(data = soybean_cn_sub,
    dataMetrics = soybean_cn_sub_metrics, option = "allPoints",
    pointColor = "magenta")
if (interactive()) {
    shiny::runApp(app)
}

# Below is the same pair of examples, only now using the
# SummarizedExperiment (i.e. dataSE) object as input.

# Example 1: Create interactive volcano plot of logged data using hexagon
# bins for the background.
```

```
## Not run:
data(se_soybean_cn_sub)
app <- plotVolcanoApp(dataSE = se_soybean_cn_sub)
if (interactive()) {
    shiny::runApp(app)
}

## End(Not run)

# Example 2: Create interactive volcano plot of logged data using points for
# the background.

## Not run:
app <- plotVolcanoApp(dataSE = se_soybean_cn_sub, option = "allPoints",
    pointColor = "magenta")
if (interactive()) {
    shiny::runApp(app)
}

## End(Not run)
```

---

se_soybean_cn_sub   *Normalized and subsetted soybean cotyledon SummarizedExperiment*

Description

This dataset contains normalized RNA-sequencing read counts from soybean cotyledon across three
time stages of development. Early stage cotyledons were collected four days after planting and were
green but closed. Middle stage cotyledons were collected while green and open, soon after the plant
generated its first set of unifoliate leaves. Late stage cotyledons were collected immediately after
the initiation of yellowing and shrinking. To save on size, this example dataset was generated by
obtaining a random subset of 1 out of 10 genes from the original resource.

Usage

```
data(se_soybean_cn_sub)
```

Format

(1) A DelayedMatrix with 7,332 rows and 10 variables and 1 row per gene. Accessible by `assay(se_soybean_cn_sub)`.
(2) A List with 1 list element per treatment group combination and 1 row per gene. Accessible by
`rowData(se_soybean_cn_sub)`.

Details

Normalized and subsetted soybean cotyledon SummarizedExperiment object

  • `assay(se_soybean_cn_sub)` Structure:

- S1.1 early stage replicate 1 normalized read counts

- S1.2 early stage replicate 2 normalized read counts

- S1.3 early stage replicate 3 normalized read counts

- S2.1 middle stage replicate 1 normalized read counts

- S2.2 middle stage replicate 2 normalized read counts

- S2.3 middle stage replicate 3 normalized read counts

- S3.1 late stage replicate 1 normalized read counts

- S3.2 late stage replicate 2 normalized read counts

- S3.3 late stage replicate 3 normalized read counts

- rowData(se_soybean_cn_sub) list element Structure:

- ID gene name

- logFC log fold change

- logCPM log counts per million

- LR likelihood ratio

- PValue p-value

- FDR FDR value

References

Brown AV, Hudson KA (2015) Developmental profiling of gene expression in soybean trifoliate leaves and cotyledons. BMC Plant Biol 15:169

See Also

soybean_cn_sub and soybean_cn_sub_metrics

---

se_soybean_ir_sub	*Raw and subsetted soybean leaves iron-metabolism SummarizedExperiment*

Description

This dataset contains raw RNA-sequencing read counts from a soybean dataset that compared leaves that were exposed to iron-rich (iron -postive) soil conditions versus leaves that were exposed to iron-poor (iron-negative) soil conditions. The data was collected 120 minutes after iron conditions were initiated. To save on size, this example dataset was generated by obtaining a random subset of 1 out of 10 genes from the original resource.

Usage

```
data(se_soybean_ir_sub)
```

Format

(1) A DelayedMatrix with 5,604 rows and 7 variables and 1 row per gene. Accessible by `assay(se_soybean_ir_sub)`.
(2) A List with 1 list element per treatment group combination and 1 row per gene. Accessible by
`rowData(se_soybean_ir_sub)`.

A data frame with 5,604 rows and 7 variables

Details

Raw and subsetted soybean leaves data SummarizedExperiment object

- `assay(se_soybean_ir_sub)` Structure:

- N.1 iron-negative condition replicate 1 raw read counts

- N.2 iron-negative condition replicate 2 raw read counts

- N.3 iron-negative condition replicate 3 raw read counts

- P.1 iron-positive condition replicate 1 raw read counts

- P.2 iron-positive condition replicate 2 raw read counts

- P.3 iron-positive condition replicate 3 raw read counts

- `rowData(se_soybean_ir_sub)` list element Structure:

- ID gene name

- logFC log fold change

- logCPM log counts per million

- LR likelihood ratio

- PValue p-value

- FDR FDR value

References

Moran Lauter AN, Graham MA. NCBI SRA bioproject accession: PRJNA318409.

See Also

soybean_ir_sub and soybean_ir_sub_metrics

---

soybean_cn *Normalized soybean cotyledon data*

Description

This dataset contains normalized RNA-sequencing read counts from soybean cotyledon across three time stages of development. Early stage cotyledons were collected four days after planting and were green but closed. Middle stage cotyledons were collected while green and open, soon after the plant generated its first set of unifoliate leaves. Late stage cotyledons were collected immediately after the initiation of yellowing and shrinking.

Format

a `RData` instance, 1 row per gene

A data frame with 73,320 rows and 10 variables

Details

Normalized soybean cotyledon data

- ID gene name
- S1.1 early stage replicate 1 normalized read counts
- S1.2 early stage replicate 2 normalized read counts
- S1.3 early stage replicate 3 normalized read counts
- S2.1 middle stage replicate 1 normalized read counts
- S2.2 middle stage replicate 2 normalized read counts
- S2.3 middle stage replicate 3 normalized read counts
- S3.1 late stage replicate 1 normalized read counts
- S3.2 late stage replicate 2 normalized read counts
- S3.3 late stage replicate 3 normalized read counts

References

Brown AV, Hudson KA (2015) Developmental profiling of gene expression in soybean trifoliate leaves and cotyledons. BMC Plant Biol 15:169

---

soybean_cn_metrics   *Normalized soybean cotyledon metrics*

Description

This data contains metrics for normalized RNA-sequencing read counts from soybean cotyledon across three time stages of development. Early stage cotyledons were collected four days after planting and were green but closed. Middle stage cotyledons were collected while green and open, soon after the plant generated its first set of unifoliate leaves. Late stage cotyledons were collected immediately after the initiation of yellowing and shrinking. The metrics include the log fold change, log counts per million, likelihood ratio, p-values, and FDR values for all genes and all pairwise combinations of treatment groups.

Format

a RData instance, 1 list per treatment group combination and 1 row per gene

A nested list of length 3. Each list contains the metrics for the 73,320 genes for one treatment group combination.

Details

Normalized soybean cotyledon metrics

- ID gene name
- logFC log fold change
- logCPM log counts per million
- LR likelihood ratio
- PValue p-value
- FDR FDR value

See Also

soybean_cn for information about the treatment groups

---

soybean_cn_sub   *Normalized and subsetted soybean cotyledon data*

Description

This dataset contains normalized RNA-sequencing read counts from soybean cotyledon across three time stages of development. Early stage cotyledons were collected four days after planting and were green but closed. Middle stage cotyledons were collected while green and open, soon after the plant generated its first set of unifoliate leaves. Late stage cotyledons were collected immediately after the initiation of yellowing and shrinking. To save on size, this example dataset was generated by obtaining a random subset of 1 out of 10 genes from the original resource.

Usage

```
data(soybean_cn_sub)
```

Format

A data frame with 7,332 rows and 10 variables. 1 row per gene.

Details

Normalized and subsetted soybean cotyledon data

- ID gene name
- S1.1 early stage replicate 1 normalized read counts
- S1.2 early stage replicate 2 normalized read counts
- S1.3 early stage replicate 3 normalized read counts
- S2.1 middle stage replicate 1 normalized read counts
- S2.2 middle stage replicate 2 normalized read counts
- S2.3 middle stage replicate 3 normalized read counts
- S3.1 late stage replicate 1 normalized read counts
- S3.2 late stage replicate 2 normalized read counts
- S3.3 late stage replicate 3 normalized read counts

References

Brown AV, Hudson KA (2015) Developmental profiling of gene expression in soybean trifoliate leaves and cotyledons. BMC Plant Biol 15:169

See Also

soybean_cn from which this dataset is subsetted

```
soybean_cn_sub_metrics
```
Normalized and subsetted soybean cotyledon metrics

Description

This data contains metrics for normalized RNA-sequencing read counts from soybean cotyledon across three time stages of development. Early stage cotyledons were collected four days after planting and were green but closed. Middle stage cotyledons were collected while green and open, soon after the plant generated its first set of unifoliate leaves. Late stage cotyledons were collected immediately after the initiation of yellowing and shrinking. The metrics include the log fold change, log counts per million, likelihood ratio, p-values, and FDR values for all genes and all pairwise combinations of treatment groups. To save on size, this example dataset was generated by obtaining a random subset of 1 out of 10 genes from the original resource.

Usage

```
data(soybean_cn_sub_metrics)
```

Format

a RData instance, List with one list element per treatment group combination and 1 row per gene

Details

Normalized and subsetted soybean cotyledon metrics

- ID gene name
- logFC log fold change
- logCPM log counts per million
- LR likelihood ratio
- PValue p-value
- FDR FDR value

See Also

soybean_cn_sub for information about the treatment groups

---

soybean_ir   *Raw soybean leaves iron-metabolism data*

Description

This dataset contains raw RNA-sequencing read counts from a soybean dataset that compared leaves that were exposed to iron-rich (iron -postive) soil conditions versus leaves that were exposed to iron-poor (iron-negative) soil conditions. The data was collected 120 minutes after iron conditions were initiated.

Format

a RData instance, 1 row per gene

A data frame with 56,044 rows and 7 variables

Details

Raw soybean leaves data

- ID gene name
- N.1 iron-negative condition replicate 1 raw read counts
- N.2 iron-negative condition replicate 2 raw read counts
- N.3 iron-negative condition replicate 3 raw read counts

- P.1 iron-positive condition replicate 1 raw read counts
- P.2 iron-positive condition replicate 2 raw read counts
- P.3 iron-positive condition replicate 3 raw read counts

---

soybean_ir_metrics　　　*Raw soybean leaves iron-metabolism metrics*

Description

This data contains metrics for raw RNA-sequencing read counts from a soybean dataset that compared leaves that were exposed to iron-rich (iron-postive) soil conditions versus leaves that were exposed to iron -poor (iron-negative) soil conditions. The data was collected 120 minutes after iron conditions were initiated. The metrics include the log fold change and the p-values for all genes and all pairwise combinations of treatment groups.

Format

a `RData` instance, 1 list per treatment group combination and 1 row per gene

A nested list of length 1. The list contains the metrics for the 56,044 genes for the one treatment group combination.

Details

Raw soybean leaves metrics

- ID gene name
- logFC log fold change
- PValue p-value

See Also

[soybean_ir](#) for information about the treatment groups

---

soybean_ir_sub　　　*Raw and subsetted soybean leaves iron-metabolism data*

Description

This dataset contains raw RNA-sequencing read counts from a soybean dataset that compared leaves that were exposed to iron-rich (iron -postive) soil conditions versus leaves that were exposed to iron-poor (iron-negative) soil conditions. The data was collected 120 minutes after iron conditions were initiated. To save on size, this example dataset was generated by obtaining a random subset of 1 out of 10 genes from the original resource.

Usage

```
data(soybean_ir_sub)
```

Format

a `RData` instance, 1 row per gene

A data frame with 5,604 rows and 7 variables

Details

Raw and subsetted soybean leaves data

- ID gene name
- N.1 iron-negative condition replicate 1 raw read counts
- N.2 iron-negative condition replicate 2 raw read counts
- N.3 iron-negative condition replicate 3 raw read counts
- P.1 iron-positive condition replicate 1 raw read counts
- P.2 iron-positive condition replicate 2 raw read counts
- P.3 iron-positive condition replicate 3 raw read counts

References

Moran Lauter AN, Graham MA. NCBI SRA bioproject accession: PRJNA318409.

See Also

soybean_ir from which this dataset is subsetted

---

soybean_ir_sub_metrics

Raw and subsetted soybean leaves iron-metabolism metrics

Description

This data contains metrics for raw RNA-sequencing read counts from a soybean dataset that compared leaves that were exposed to iron-rich (iron-postive) soil conditions versus leaves that were exposed to iron -poor (iron-negative) soil conditions. The data was collected 120 minutes after iron conditions were initiated. The metrics include the log fold change and the p-values for all genes and all pairwise combinations of treatment groups. To save on size, this example dataset was generated by obtaining a random subset of 1 out of 10 genes from the original resource.

Usage

```
data(soybean_ir_sub_metrics)
```

Format

a `RData` instance, 1 list per treatment group combination and 1 row per gene

A nested list of length 1. The list contains the metrics for the 5,604 genes for the one treatment group combination.

Details

Raw and subsetted soybean leaves iron-metabolism metrics

- ID gene name
- logFC log fold change
- PValue p-value

References

Moran Lauter AN, Graham MA. NCBI SRA bioproject accession: PRJNA318409.

See Also

[soybean_ir_sub](#) for information about the treatment groups

Index