Package 'iSEEu'

October 17, 2020

Type Package

Title iSEE Universe

Version 1.0.1

Date 2020-05-01

Description iSEEu (the iSEE universe) contains diverse functionality to extend the usage of the iSEE package, including additional classes for the panels, or modes allowing easy configuration of iSEE applications.

License MIT + file LICENSE

Encoding UTF-8

Depends iSEE

Imports methods, S4Vectors, shiny, SummarizedExperiment, SingleCellExperiment, ggplot2, DT, stats

Suggests scRNAseq, scater, scran, airway, edgeR, AnnotationDbi, org.Hs.eg.db, GO.db, knitr, rmarkdown, BiocStyle, htmltools, Rtsne, uwot, testthat (>= 2.1.0)

URL https://github.com/iSEE/iSEEu

BugReports https://github.com/iSEE/iSEEu/issues

biocViews ImmunoOncology, Visualization, GUI, DimensionReduction, FeatureExtraction, Clustering, Transcription, GeneExpression, Transcriptomics, SingleCell, CellBasedAssays

RoxygenNote 7.1.0

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/iSEEu

git_branch RELEASE_3_11

git_last_commit 31fc5ea

git_last_commit_date 2020-05-01

Date/Publication 2020-10-16

Author Kevin Rue-Albrecht [aut, cre] (<https://orcid.org/0000-0003-3899-3872>), Charlotte Soneson [aut] (<https://orcid.org/0000-0003-3833-2169>), Federico Marini [aut] (<https://orcid.org/0000-0003-3252-7758>), Aaron Lun [aut] (<https://orcid.org/0000-0002-3564-4813>), Michael Stadler [ctb]

Maintainer Kevin Rue-Albrecht <kevinrue67@gmail.com>

R topics documented:

`DifferentialStatisticsTable-class`
Differential statistics table

Description

A table that dynamically computes differential statistics based on a selected subset of samples. Comparisons are made between the active selection in the transmitting panel and (i) all non-selected points, if no saved selections are available; or (ii) each subset of points in each saved selection.

Slot overview

The following slots control the thresholds used in the visualization:

- `LogFC`, a numeric scalar indicating the log-fold change threshold to test against. Defaults to zero.
- `TestMethod`, string indicating the test to use (based on the `findMarkers` function from **scran**). This can be `"t"` (default), `"wilcox"` or `"binom"`.
- `Assay`, string indicating the assay to use for testing. Defaults to the first named assay in the SummarizedExperiment.

In addition, this class inherits all slots from its parent RowTable, Table and Panel classes.

Constructor

`DifferentialStatisticsTable(...)` creates an instance of a DifferentialStatisticsTable class, where any slot and its value can be passed to `...` as a named argument.

Supported methods

In the following code snippets, `x` is an instance of a DifferentialStatisticsTable class. Refer to the documentation for each method for more details on the remaining arguments.

For setting up data values:

- `.cacheCommonInfo(x)` adds a `"DifferentialStatisticsTable"` entry containing `valid.assay.names`. This will also call the equivalent RowTable method.

- `.refineParameters(x,se)` returns `x` after setting `"Assay"` to the first valid value. This will also call the equivalent RowTable method for further refinements to `x`. If valid assay names are not available, `NULL` is returned instead.

For defining the interface:

- `.defineDataInterface(x,se,select_info)` returns a list of interface elements for manipulating all slots described above.
- `.panelColor(x)` will return the specified default color for this panel class.
- `.fullName(x)` will return `"Differential statistics table"`.
- `.hideInterface(x)` will return `TRUE` for UI elements related to multiple row selections, otherwise calling the method for RowTable.

For monitoring reactive expressions:

- `.createObservers(x,se,input,session,pObjects,rObjects)` sets up observers for all new slots described above, as well as in the parent classes via the RowTable method.

For creating the table:

- `.generateTable(x,envir)` will create a data.frame of newly computed statistics in `envir`. The method will return the commands required to do so.

Examples

```
library(scRNAseq)
library(scater)

sce <- ReprocessedAllenData(assays="tophat_counts")
sce <- logNormCounts(sce, exprs_values="tophat_counts")
sce <- runPCA(sce, ncomponents=4)
sce <- runTSNE(sce)

dst <- DifferentialStatisticsTable(PanelId=1L, PanelWidth=8L,
    ColumnSelectionSource="ReducedDimensionPlot1")

rdp <- ReducedDimensionPlot(PanelId=1L,
    ColorByFeatureSource="DifferentialStatisticsTable1")

if (interactive()) {
    iSEE(sce, initial=list(rdp, dst))
}
```

`DynamicReducedDimensionPlot-class`

Dynamic reduced dimension plot

Description

A dimensionality reduction plot that dynamically recomputes the coordinates for the samples, based on the selected subset of samples (and possibly features) in transmitting panels. All samples in active and saved multiple selections are used here.

Slot overview

The following slots control the thresholds used in the visualization:

- `Type`, a string specifying the type of dimensionality reduction method to use. This can be `"PCA"` (default), `"TSNE"` or `"UMAP"`, which uses the relevant functions from the **scater** package.
- `NGenes`, an integer scalar specifying the number of highly variable genes to use in the dimensionality reduction. Only used if an explicit selection of features is not made in the app. Defaults to 1000.
- `Assay`, string indicating the assay to use for the calculations. Defaults to the first named assay in the SummarizedExperiment.

In addition, this class inherits all slots from its parent ColumnDotPlot, DotPlot and Panel classes.

Constructor

`DynamicReducedDimensionPlot(...)` creates an instance of a DynamicReducedDimensionPlot class, where any slot and its value can be passed to `...` as a named argument.

Supported methods

In the following code snippets, `x` is an instance of a DynamicReducedDimensionPlot class. Refer to the documentation for each method for more details on the remaining arguments.

For setting up data values:

- `.cacheCommonInfo(x)` adds a `"DynamicReducedDimensionPlot"` entry containing `valid.assay.names`. This will also call the equivalent ColumnDotPlot method.
- `.refineParameters(x,se)` returns `x` after setting `"Assay"` to the first valid value. This will also call the equivalent ColumnDotPlot method for further refinements to `x`. If valid assay names are not available, `NULL` is returned instead.

For defining the interface:

- `.defineDataInterface(x,se,select_info)` returns a list of interface elements for manipulating all slots described above.
- `.panelColor(x)` will return the specified default color for this panel class.
- `.fullName(x)` will return `"Dynamic reduced dimension plot"`.

For monitoring reactive expressions:

- `.createObservers(x,se,input,session,pObjects,rObjects)` sets up observers for all new slots described above, as well as in the parent classes via the ColumnDotPlot method.

For creating the plot:

- `.generateDotPlotData(x,envir)` will create a data.frame of newly computed coordinates in `envir`. The method will return the commands required to do so as well as a list of labels.

For handling multiple selections:

- `.multiSelectionInvalidated(x)` will always return `TRUE`, as any change in the upstream selection of points will alter the coordinates and invalidate any brush/lasso on `x`.

Author(s)

Aaron Lun

Examples

```
library(scRNAseq)
library(scater)

sce <- ReprocessedAllenData(assays="tophat_counts")
sce <- logNormCounts(sce, exprs_values="tophat_counts")
sce <- runPCA(sce, ncomponents=4)
sce <- runTSNE(sce)

drdp <- DynamicReducedDimensionPlot(PanelId=1L, Assay="logcounts",
    ColumnSelectionSource="ReducedDimensionPlot1")

if (interactive()) {
    iSEE(sce, initial=list(ReducedDimensionPlot(PanelId=1L), drdp))
}
```

GeneSetTable-class *Gene set table*

Description

A table where each row is a gene set and can be clicked to transmit a multiple feature selection to another panel. This usually requires some set-up with `.setIdentifierType` and related functions, see Examples.

Slot overview

The following slots control the thresholds used in the visualization:

- `Type`, string specifying the type of gene set collection to show. Defaults to `"GO"`.
- `Selected`, a string containing the name of the currently selected gene set. Defaults to `""`, i.e., no selection.
- `Search`, a string containing the regular expression for the global search. Defaults to `""`, i.e., no search.
- `SearchColumns`, a character vector where each entry contains the search string for each column. Defaults to an empty character vector, i.e., no search.

In addition, this class inherits all slots from its parent Panel class.

Constructor

`GeneSetTable(...)` creates an instance of a GeneSetTable class, where any slot and its value can be passed to `...` as a named argument.

Supported methods

In the following code snippets, `x` is an instance of a DifferentialStatisticsTable class. Refer to the documentation for each method for more details on the remaining arguments.

For defining the interface:

- `.defineDataInterface(x,se,select_info)` returns a list of interface elements for manipulating all slots described above.
- `.panelColor(x)` will return the specified default color for this panel class.
- `.fullName(x)` will return `"Gene set table"`.
- `.hideInterface(x)` will return `TRUE` for UI elements related to multiple selections, otherwise calling the method for Panel.

For monitoring reactive expressions:

- `.createObservers(x,se,input,session,pObjects,rObjects)` sets up observers for all new slots described above, as well as in the parent classes via the Panel method.

For creating the table:

- `.generateOutput(x,envir)` will create a data.frame of gene set descriptions in `envir`, based on the `mode="show"` output of `.getGeneSetCommands`. It will also return the commands required to do so.
- `.renderOutput(x,se,...,output,pObjects,rObjects)` will add a `datatable` widget to the output, which is used to render the aforementioned data.frame.

For controlling the multiple selections:

- `.multiSelectionDimension(x)` returns `"row"`.
- `.multiSelectionCommands(x,index)` returns a string specifying the commands to be used to extract the identities of the genes in the currently selected set, based on the `mode="extract"` output of `.getGeneSetCommands`. `index` is ignored.
- `.multiSelectionActive(x)` returns the name of the currently selected gene set, unless no selection is made, in which case `NULL` is returned.
- `.multiSelectionClear(x)` returns `x` but with the `Selected` slot replaced by an empty string.
- `.multiSelectionAvailable(x,contents)` returns `contents$available`, which is set to the number of features in `se`.

Author(s)

Aaron Lun

Examples

```
library(scRNAseq)
sce <- LunSpikeInData(location=FALSE)

library(scater)
sce <- logNormCounts(sce)

library(scran)
rowData(sce) <- cbind(rowData(sce), modelGeneVarWithSpikes(sce, "ERCC"))

# This defaults to 'org.Hs.eg.db' with 'ENTREZID'.
.setOrganism("org.Mm.eg.db")
.setIdentifierType("ENSEMBL")
gst <- GeneSetTable(PanelId=1L)

rdp <- RowDataPlot(RowSelectionSource="GeneSetTable1",
    SelectionEffect="Color",
```

```
    XAxis="Row data", XAxisRowData="mean", YAxis="total")

rdt <- RowDataTable(RowSelectionSource="GeneSetTable1")

if (interactive()) {
    iSEE(sce, initial=list(gst, rdp, rdt))
}
```

`MAPlot-class` *The MAPlot class*

Description

The MAPlot is a RowDataPlot subclass that is dedicated to creating a MA plot. It retrieves the log-fold change and average abundance and creates a row-based plot where each point represents a feature. Users are expected to load relevant statistics into the `rowData` of a SummarizedExperiment.

Slot overview

The following slots control the thresholds used in the visualization:

- `PValueField`, a string specifying the field of `rowData` containing the p-values. See `?.getAcceptablePValueFiel` for more details.
- `PValueThreshold`, a numeric scalar in (0, 1] specifying the threshold to use on the (adjusted) p-value. Defaults to 0.05.
- `LogFCThreshold`, a non-negative numeric scalar specifying the threshold to use on the log-fold change. Defaults to 0.
- `PValueCorrection`, a string specifying the multiple testing correction to apply. Defaults to `"BH"`, but can take any value from `p.adjust.methods`.

In addition, this class inherits all slots from its parent RowDataPlot, RowDotPlot, DotPlot and Panel classes.

Constructor

`MAPlot(...)` creates an instance of a MAPlot class, where any slot and its value can be passed to `...` as a named argument.

Supported methods

In the following code snippets, `x` is an instance of a RowDataPlot class. Refer to the documentation for each method for more details on the remaining arguments.

For setting up data values:

- `.cacheCommonInfo(x)` adds a `"MAPlot"` entry containing `pval.rowData.names`, `ave.rowData.names` and `lfc.rowData.names`. Each of these is a character vector of permissible names for p-values, average abundances and log-fold changes, respectively; see `?.getAcceptablePValueFields` for details. This will also call the equivalent RowDataPlot method.
- `.refineParameters(x,se)` returns `x` after setting `XAxis="Row data"`. This will also call the equivalent RowDataPlot method for further refinements to `x`. If valid p-value and log-fold change fields are not available, `NULL` is returned instead.

For defining the interface:

- `.defineDataInterface(x,se,select_info)` returns a list of interface elements for manipulating all slots described above.
- `.panelColor(x)` will return the specified default color for this panel class.
- `.allowableXAxisChoices(x,se)` returns a character vector specifying the acceptable average abundance-related variables in `rowData(se)` that can be used as choices for the x-axis, see `?.getAcceptableAveAbFields`.
- `.allowableYAxisChoices(x,se)` returns a character vector specifying the acceptable log-fold change-related variables in `rowData(se)` that can be used as choices for the y-axis, see `?.getAcceptableLogFCFields`.
- `.hideInterface(x,field)` will return `TRUE` for `field="XAxis"`, otherwise it will call the RowDataPlot method.
- `.fullName(x)` will return `"MA plot"`.

For monitoring reactive expressions:

- `.createObservers(x,se,input,session,pObjects,rObjects)` sets up observers for all new slots described above, as well as in the parent classes via the RowDataPlot method.

For creating the plot:

- `.generateDotPlotData(x,envir)` will create a data.frame of row metadata variables in `envir`. This should contain average abundances on the x-axis and log-fold changes on the y-axis, in addition to an extra field specifying whether or not the feature was considered to be significantly up or down. The method will return the commands required to do so as well as a list of labels.
- `.prioritizeDotPlotData(x,envir)` will create variables in `envir` marking the priority of points. Significant features receive higher priority (i.e., are plotted over their non-significant counterparts) and are less aggressively downsampled when `Downsample=TRUE`. The method will return the commands required to do this as well as a logical scalar indicating that rescaling of downsampling resolution is performed.
- `.colorByNoneDotPlotField(x)` will return a string specifying the field of the data.frame (generated by `.generateDotPlotData`) containing the significance information. This is to be used for coloring when `ColorBy="None"`.
- `.colorByNoneDotPlotScale(x)` will return a string containing a **ggplot2** command to add a default color scale when `ColorBy="None"`.
- `.generateDotPlot(x,labels,envir)` returns a list containing `plot` and `commands`, using the inital ColumnDataPlot ggplot and adding horizontal lines demarcating the log-fold change threshold.

Author(s)

Aaron Lun

See Also

RowDataPlot, for the base class.

Examples

```
# Making up some results:
se <- SummarizedExperiment(matrix(rnorm(10000), 1000, 10))
rownames(se) <- paste0("GENE_", seq_len(nrow(se)))
rowData(se)$PValue <- runif(nrow(se))
rowData(se)$LogFC <- rnorm(nrow(se))
rowData(se)$AveExpr <- rnorm(nrow(se))

if (interactive()) {
    iSEE(se, initial=list(MAPlot()))
}
```

---

modeEmpty *App pre-configured to launch with no visible panel*

Description

This mode launches an app that does not display any panel.

Usage

```
modeEmpty(...)
```

Arguments

`...`	Arguments passed to `iSEE()`.

Details

This mode presents the advantage to launch an interface in a minimal amount of time, as it does not need to render any panel when the interface is launched. Users can then use the `"Organize panels"` widget to select panels to display in the interface.

Value

A Shiny app object is returned.

Examples

```
example("SingleCellExperiment")
rownames(sce) <- paste0("G", 1:200)
colnames(sce) <- paste0("C", 1:100)

app <- modeEmpty(sce)
if (interactive()) {
  shiny::runApp(app, port=1234)
}
```

modeGating *App pre-configured to link multiple feature assay plots*

Description

This mode launches a Shiny App preconfigured with multiple chain-linked feature expression plots for interactive data exploration of the SingleCellExperiment or `SummarizedExperiment` object.

Usage

```
modeGating(se, features, plotAssay = NA_character_, ..., plotWidth = 4)
```

Arguments

`se`	An object that coercible to SingleCellExperiment-class
`features`	`data.frame` with columns named `x` and `y` that define the features on the axes of the linked plots. Plots are serially linked from the first row to the last.
`plotAssay`	The assay (one of assayNames(se)) to use for the plots (character vector of length either 1 or equal to `nrow(features)`).
`...`	Additional arguments passed to `iSEE()`.
`plotWidth`	The grid width of linked plots (numeric vector of length either 1 or equal to `nrow(features)`

Value

A Shiny app object is returned.

Examples

```
library(scRNAseq)

# Example data ----
sce <- ReprocessedAllenData(assays="tophat_counts")
class(sce)

library(scater)
sce <- logNormCounts(sce, exprs_values="tophat_counts")

# Select top variable genes ----

plot_count <- 6
rv <- rowVars(assay(sce, "tophat_counts"))
top_var <- head(order(rv, decreasing=TRUE), plot_count*2)
top_var_genes <- rownames(sce)[top_var]

plot_features <- data.frame(
    x=head(top_var_genes, plot_count),
    y=tail(top_var_genes, plot_count),
    stringsAsFactors=FALSE
 )

# launch the app itself ----
```

```
app <- modeGating(sce, features = plot_features)
if (interactive()) {
  shiny::runApp(app, port=1234)
}
```

`modeReducedDim` *App pre-configured to compare multiple reduced dimension plots*

Description

This mode launches a Shiny App preconfigured with multiple linked reduced dimension plots for interactive data exploration of the `SingleCellExperiment` object.

Usage

```
modeReducedDim(
  se,
  includeNames = reducedDimNames(se),
  colorBy = NULL,
  ...,
  plotWidth = NULL
)
```

Arguments

`se`	An object that coercible to SingleCellExperiment
`includeNames`	Character vector with the names of reduced dimensions to display as individual panels. The default uses all available in `reducedDimNames(se)`.
`colorBy`	Character scalar controlling coloring of cells. Must match either to one of `colnames(colData(se))` or `rownames(se)`. If coloring by a colData column, a column data plot is opened in addition to the reduced dimension panels. If coloring by a feature, a row statistics table is openend in addition to the reduced dimension panels, from which the latter are receiving the color.
`...`	Additional arguments passed to `iSEE`.
`plotWidth`	The grid width of linked plots (numeric vector of length either 1 or equal to `length(includeNames)`). The total width of the window is 12, so `plotWidth = 4` for example will show three panels per row. If `plotWidth = NULL` (the default), a value will be estimated depending on the number of reduced dimension panels.

Value

A Shiny app object is returned.

Examples

```
library(scRNAseq)

# Example data ----
sce <- ReprocessedAllenData(assays="tophat_counts")
class(sce)

library(scater)
sce <- logNormCounts(sce, exprs_values="tophat_counts")
sce <- runPCA(sce, ncomponents = 30)
sce <- runTSNE(sce)
sce <- runUMAP(sce)
reducedDimNames(sce)

# launch the app ----
# ... coloring by a column data variable
app <- modeReducedDim(sce, colorBy = "Primary.Type")
if (interactive()) {
    shiny::runApp(app, port=1234)
}
# ... coloring by a feature
app <- modeReducedDim(sce, colorBy = "Scnn1a")
if (interactive()) {
    shiny::runApp(app, port=1234)
}
```

ReducedDimensionHexPlot-class

The ReducedDimensionHexPlot class

Description

The ReducedDimensionHexPlot is a ReducedDimensionPlot subclass that is dedicated to creating a reduced dimension plot summarisign data points in hexagonal bins.

Slot overview

The following slots control the parameters used in the visualization:

- `BinResolution`, a numeric positive scalar specifying the number of hexagonal bins in both vertical and horizontal directions. Defaults to 100.

In addition, this class inherits all slots from its parent ReducedDimensionPlot, ColumnDotPlot, DotPlot and Panel classes.

Constructor

`ReducedDimensionHexPlot(...)` creates an instance of a ReducedDimensionHexPlot class, where any slot and its value can be passed to `...` as a named argument.

Supported methods

In the following code snippets, x is an instance of a ReducedDimensionHexPlot class. Refer to the documentation for each method for more details on the remaining arguments.

For defining the interface:

- `.panelColor(x)` will return the specified default color for this panel class.
- `.hideInterface(x,field)` will return `TRUE` for `field="Downsample"`, otherwise it will call the ReducedDimensionPlot method.

For monitoring reactive expressions:

- `.createObservers(x,se,input,session,pObjects,rObjects)` sets up observers for all new slots described above, as well as in the parent classes via the ReducedDimensionPlot method.

For defining the panel name:

- `.fullName(x)` will return `"Hexagonal reduced dimension plot"`.

Author(s)

Kevin Rue-Albrecht

See Also

ReducedDimensionPlot, for the base class.

Examples

```
library(scRNAseq)

# Example data ----
sce <- ReprocessedAllenData(assays="tophat_counts")
class(sce)

library(scater)
sce <- logNormCounts(sce, exprs_values="tophat_counts")

sce <- runPCA(sce, ncomponents=4)
sce <- runTSNE(sce)
rowData(sce)$ave_count <- rowMeans(assay(sce, "tophat_counts"))
rowData(sce)$n_cells <- rowSums(assay(sce, "tophat_counts") > 0)

# launch the app itself ----

if (interactive()) {
    iSEE(sce, initial=list(
        ReducedDimensionHexPlot(BinResolution=50),
        ReducedDimensionPlot()
    ))
}
```

utils-de *Acceptable fields for DE panels*

Description

Set or get the acceptable fields to use for all Panel instances related to differential expression, including VolcanoPlot and MAPlot.

Usage

```
.getAcceptablePValueFields()

.getAcceptableLogFCFields()

.getAcceptableAveAbFields()

.setAcceptablePValueFields(value)

.setAcceptableLogFCFields(value)

.setAcceptableAveAbFields(value)
```

Arguments

value	Character vector of acceptable fields (usually in the rowData) for a given statistic.

Value

.getAcceptablePValueFields will return a character vector of acceptable names for p-value fields.

.getAcceptableLogFCFields will return a character vector of acceptable names for log-FC fields.

.getAcceptableAveAbFields will return a character vector of acceptable names for average abundance fields.

The setter functions will define the set of acceptable fields and return NULL invisibly.

Author(s)

Aaron Lun

Examples

```
old <- .getAcceptablePValueFields()
old

.setAcceptablePValueFields("YAY")
.getAcceptablePValueFields()

# Restoring.
.setAcceptablePValueFields(old)
```

`utils-geneset` *Gene set utilities*

Description

Utility functions to control the behavior of the GeneSetTable.

Usage

```
.getIdentifierType()

.setIdentifierType(value)

.getOrganism()

.setOrganism(value)

.getGeneSetCommands(collection, mode)

.setGeneSetCommands(value)
```

Arguments

`value`	For `.setIdentifierType` and `.setOrganism`, a string containing the type of identifier or organism package to use. For `.setGeneSetCommands`, a named list containing two character vectors, see Details.
`collection`	String specifying the gene set collection.
`mode`	String specifying the mode of operation for the returned commands.

Details

By default, `.getGeneSetCommands` will extract GO and KEGG terms. The organism and identifier type relates to the manner in which this default extraction is performed.

Users can add their own gene set collections by supplying a named list to `.setGeneSetCommands`. Each element of the list should be a named character vector of length two, with names `"show"` and `"extract"` - see the return value for what these are. The names of the list should be unique and will be used in the GeneSetTable interface.

Alternatively, any element of the list may be `NULL`, in which case it is excluded from the interface. This is useful for setting, e.g., `GO=NULL` to ignore the in-built GO terms.

Value

`.getIdentifierType` will return the identifier type to use, defaulting to `"ENTREZID"`.

`.getOrganism` will return the organism package to use, defaulting `"org.Hs.eg.db"`.

`.getGeneSetCommands` will return:

- If `mode="show"`, a string containing R commands that create `tab`, a data.frame of all gene sets for a given `collection`.

- If `mode="extract"`, a format string containing R commands that (after formatting) create `selected`, a character vector of gene identities for the selected gene set. This format string should accept one string argument corresponding to the deparsed name of the gene set.

Each of the setter functions will set the corresponding option and return `NULL`, invisibly.

Author(s)

Aaron Lun

See Also

GeneSetTable, where these functions have their effect.

Examples

```
.setIdentifierType("ENSEMBLID")
.getIdentifierType()

.setOrganism("org.Mm.eg.db")
.getOrganism()

.getGeneSetCommands("GO", "show")
.getGeneSetCommands("GO", "extract")

.setGeneSetCommands(
    list(AaronRandomCollection=
        c(
            show='tab <- some_function_to_list_my_gene_sets()',
            extract='selected <- some_function_to_get_one_gene_set(%s)'
        )
    )
)

.getGeneSetCommands("AaronRandomCollection", "show")
.getGeneSetCommands("AaronRandomCollection", "extract")
```

VolcanoPlot-class *The VolcanoPlot class*

Description

The VolcanoPlot is a RowDataPlot subclass that is dedicated to creating a volcano plot. It retrieves the log-fold change and p-value from and creates a row-based plot where each point represents a feature. Users are expected to load relevant statistics into the `rowData` of a SummarizedExperiment.

Slot overview

The following slots control the thresholds used in the visualization:

- `PValueThreshold`, a numeric scalar in (0, 1] specifying the threshold to use on the (adjusted) p-value. Defaults to 0.05.

- `LogFCThreshold`, a non-negative numeric scalar specifying the threshold to use on the log-fold change. Defaults to 0.
- `PValueCorrection`, a string specifying the multiple testing correction to apply. Defaults to `"BH"`, but can take any value from `p.adjust.methods`.

In addition, this class inherits all slots from its parent RowDataPlot, RowDotPlot, DotPlot and Panel classes.

Constructor

`VolcanoPlot(...)` creates an instance of a VolcanoPlot class, where any slot and its value can be passed to `...` as a named argument.

Supported methods

In the following code snippets, `x` is an instance of a RowDataPlot class. Refer to the documentation for each method for more details on the remaining arguments.

For setting up data values:

- `.cacheCommonInfo(x)` adds a `"MAPlot"` entry containing `pval.rowData.names` and `lfc.rowData.names`. Each of these is a character vector of permissible names for p-values and log-fold changes, respectively; see `?.getAcceptablePValueFields` for details. This will also call the equivalent RowDataPlot method.
- `.refineParameters(x,se)` returns `x` after setting `XAxis="Row data"`. This will also call the equivalent RowDataPlot method for further refinements to `x`. If valid p-value and log-fold change fields are not available, `NULL` is returned instead.

For defining the interface:

- `.defineDataInterface(x,se,select_info)` returns a list of interface elements for manipulating all slots described above.
- `.panelColor(x)` will return the specified default color for this panel class.
- `.allowableXAxisChoices(x,se)` returns a character vector specifying the acceptable log-fold change-related variables in `rowData(se)` that can be used as choices for the x-axis, see `?.getAcceptableLogFCFields`.
- `.allowableYAxisChoices(x,se)` returns a character vector specifying the acceptable p-value-related variables in `rowData(se)` that can be used as choices for the y-axis, see `?.getAcceptablePValueFie`
- `.hideInterface(x,field)` will return `TRUE` for `field="XAxis"`, otherwise it will call the RowDataPlot method.
- `.fullName(x)` will return `"Volcano plot"`.

For monitoring reactive expressions:

- `.createObservers(x,se,input,session,pObjects,rObjects)` sets up observers for all new slots described above, as well as in the parent classes via the RowDataPlot method.

For creating the plot:

- `.generateDotPlotData(x,envir)` will create a data.frame of row metadata variables in `envir`. This should contain negative log-transformed p-values on the y-axis and log-fold changes on the x-axis, in addition to an extra field specifying whether or not the feature was considered to be significantly up or down. The method will return the commands required to do so as well as a list of labels.

- `.prioritizeDotPlotData(x,envir)` will create variables in `envir` marking the priority of points. Significant features receive higher priority (i.e., are plotted over their non-significant counterparts) and are less aggressively downsampled when `Downsample=TRUE`. The method will return the commands required to do this as well as a logical scalar indicating that rescaling of downsampling resolution is performed.
- `.colorByNoneDotPlotField(x)` will return a string specifying the field of the data.frame (generated by `.generateDotPlotData`) containing the significance information. This is to be used for coloring when `ColorBy="None"`.
- `.colorByNoneDotPlotScale(x)` will return a string containing a **ggplot2** command to add a default color scale when `ColorBy="None"`.
- `.generateDotPlot(x,labels,envir)` returns a list containing `plot` and `commands`, using the inital ColumnDataPlot ggplot and adding vertical lines demarcating the log-fold change threshold.

Author(s)

Aaron Lun

See Also

RowDataPlot, for the base class.

Examples

```
# Making up some results:
se <- SummarizedExperiment(matrix(rnorm(10000), 1000, 10))
rownames(se) <- paste0("GENE_", seq_len(nrow(se)))
rowData(se)$PValue <- runif(nrow(se))
rowData(se)$LogFC <- rnorm(nrow(se))
rowData(se)$AveExpr <- rnorm(nrow(se))

if (interactive()) {
    iSEE(se, initial=list(VolcanoPlot()))
}
```

Index