Package 'iSEE'

October 16, 2018

Title Interactive SummarizedExperiment Explorer

Version 1.0.1

Date 2018-05-02

Maintainer Federico Marini <marinif@uni-mainz.de>

Description Provides functions for creating an interactive Shiny-based graphical user interface for exploring data stored in SummarizedExperiment objects, including row- and column-level metadata. Particular attention is given to single-cell data in a SingleCellExperiment object with visualization of dimensionality reduction results.

Depends R (>= 3.5), SummarizedExperiment, SingleCellExperiment

Imports methods, BiocGenerics, S4Vectors, utils, stats, shiny, shinydashboard, shinyAce, shinyjs, DT, rintrojs, ggplot2, colourpicker, igraph, vipor, mgcv, reshape2, rentrez, AnnotationDbi, graphics, grDevices, viridisLite, cowplot, scales, dplyr

Suggests testthat, BiocStyle, knitr, rmarkdown, scRNAseq, scater, Rtsne, irlba, RColorBrewer, viridis, org.Mm.eg.db, htmltools

Enhances ExperimentHub

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

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

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

License MIT + file LICENSE

VignetteBuilder knitr

RoxygenNote 6.0.1

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

git_branch RELEASE_3_7

git_last_commit 6cc698d

git_last_commit_date 2018-05-02

Date/Publication 2018-10-15

Author Charlotte Soneson [aut, cre],
Aaron Lun [aut],
Federico Marini [aut],
Kevin Rue-Albrecht [aut]

R topics documented:

annotateEnsembl 2
annotateEntrez 3
defaults 4
ExperimentColorMap-class 9
isColorMapCompatible 11
iSEE 12
iSEE-pkg 14
modeGating 15
subsetPointsByGrid 16
synchronizeAssays 17

Index **20**

`annotateEnsembl` *Annotation via ENSEMBL database*

Description

Annotation facility for displaying additional information on selected genes, based on the data retrieved from the ENSEMBL database

Usage

```
annotateEnsembl(se, orgdb, keytype, rowdata_col = NULL,
  ens_species = gsub(" ", "_", species(orgdb)))
```

Arguments

`se`	An object that is coercible to SingleCellExperiment.
`orgdb`	An OrgDb object, as a basis for the annotation. Typical values can be `org.Hs.eg.db` for human, `org.Mm.eg.db` for mouse, and so on. The corresponding package has to be available and loaded before calling this function
`keytype`	The keytype that matches the IDs used in the `se` object. Can be one of the values of `keytypes(org.XX.eg.db)`, typically being "SYMBOL", "ENSEMBL", or "ENTREZID"
`rowdata_col`	A character string, corresponding to one of the columns (if present) in `rowData(se)`. Defaults to NULL, which corresponds to having the ids of the features as row names of the `se` object itself.
`ens_species`	Character string containing the species name, coded as in the ENSEMBL database and browser. This is for example "Homo_sapiens" for human, and "Mus_musculus" for mouse. Defaults to `species(orgdb)`, with the whitespace replaced by underscore

Value

A function to be used as the value of the annotFun parameter of iSEE. This function itself returns a HTML tag object with the content extracted from the call, with parameteres the se object, and the row_index corresponding to the feature of interest.

Examples

```
library(scRNAseq)
data(allen)
sce <- as(allen, "SingleCellExperiment")
library(org.Mm.eg.db)
myfun <- annotateEnsembl(sce,org.Mm.eg.db,"SYMBOL")
myfun(sce, 4242)

# to be used when launching the app itself ----

app <- iSEE(sce, annotFun = annotateEnsembl(sce,org.Mm.eg.db,"SYMBOL"))
if (interactive()) {
  shiny::runApp(app, port = 1234)
}
```

---

annotateEntrez *Annotation via Entrez database*

Description

Annotation facility for displaying additional information on selected genes, based on the data retrieved from the ENTREZ database

Usage

```
annotateEntrez(se, orgdb, keytype, rowdata_col = NULL)
```

Arguments

se	An object that is coercible to SingleCellExperiment.
orgdb	An OrgDb object, as a basis for the annotation. Typical values can be org.Hs.eg.db for human, org.Mm.eg.db for mouse, and so on. The corresponding package has to be available and loaded before calling this function.
keytype	The keytype that matches the IDs used in the se object. Can be one of the values of keytypes(org.XX.eg.db), typically being "SYMBOL", "ENSEMBL", or "ENTREZID"
rowdata_col	A character string, corresponding to one of the columns (if present) in rowData(se). Defaults to NULL, which corresponds to having the ids of the features as row names of the se object itself.

Value

A function to be used as the value of the annotFun parameter of iSEE. This function itself returns a HTML tag object with the content extracted from the call, with parameteres the se object, and the row_index corresponding to the feature of interest.

Examples

```
library(scRNAseq)
data(allen)
sce <- as(allen, "SingleCellExperiment")
library(org.Mm.eg.db)
annotateEntrez(sce,org.Mm.eg.db,"SYMBOL")
myfun <- annotateEntrez(sce,org.Mm.eg.db,"SYMBOL")
## Not run:
# Requires a working internet connection
myfun(sce, 4242)

## End(Not run)

# to be used when launching the app itself ----

app <- iSEE(sce, annotFun = annotateEntrez(sce,org.Mm.eg.db,"SYMBOL"))
if (interactive()) {
  shiny::runApp(app, port = 1234)
}
```

---

defaults *Parameter defaults*

Description

Create default settings for various panels in the iSEE interface.

Usage

```
redDimPlotDefaults(se, number)

featAssayPlotDefaults(se, number)

colDataPlotDefaults(se, number)

rowStatTableDefaults(se, number)

rowDataPlotDefaults(se, number)

heatMapPlotDefaults(se, number)
```

Arguments

se	A SummarizedExperiment object.
number	An integer scalar, specifying the maximum number of panels of the corresponding type that can be added to the interface.

Value

A DataFrame containing default settings for various parameters of each panel.

Reduced dimension plot parameters

`Type`: Integer, which entry of `reducedDims(se)` should be shown? Defaults to 1, i.e., the first entry. Alternatively, a string can be supplied containing the name of the reduced dimension field, if `reducedDims(se)` has names.

`XAxis`: Integer, which component should be shown on the x-axis? Defaults to 1.

`YAxis`: Integer, which component should be shown on the y-axis? Defaults to 2.

Feature assay plot parameters

`YAxisFeatName`: Integer, the index of the feature for which to show the expression on the y-axis if `YAxis="Feature name"`. Defaults to 1, i.e., the first feature in `se`. Alternatively, a string can be supplied containing the name of the feature, i.e., the row name.

`YAxisRowTable`: Character, what row statistic table should be used to choose a feature to display on the y-axis? Any setting will override `YAxisFeatName` upon initialization of the app. Defaults to `"---"`, which means that no table will be used.

`Assay`: Integer, which assay should be used to supply the expression values shown on the y-axis? Defaults to 1, i.e., the first assay in `se`. Alternatively, a string can also be supplied containing the name of the assay, if `assays(se)` has names.

`XAxis`: Character, what type of variable should be shown on the x-axis? Defaults to `"None"`, but can also be `"Row table"`, `"Column data"` or `"Feature name"`.

`XAxisColData`: Character, what column of `colData(se)` should be shown on the x-axis if `XAxis="Column data"`? Defaults to the first entry of `colData(se)`.

`XAxisFeatName`: Integer, the index of the feature for which to show the expression on the x-axis if `XAxis="Feature name"`. Defaults to 1, i.e., the first feature in `se`. Alternatively, a string can be supplied containing the name of the feature.

`XAxisRowTable`: Character, which row statistic table should be used to choose a feature to put on the x-axis if `XAxis="Row table"`? Any setting will override `XAxisFeatName` upon initialization of the app. Defaults to `"---"`, which means that no table will be used.

Column data plot parameters

`YAxis`: Character, which column of `colData(se)` should be shown on the y-axis? Defaults to the first entry of `colData(se)`.

`XAxis`: Character, what type of variable should be shown on the x-axis? Defaults to `"None"`, but can also be `"Column data"`.

`XAxisColData`: Character, which column of `colData(se)` should be shown on the x-axis if `XAxis="Column data"`? Defaults to the first entry of `colData(se)`.

Row data plot parameters

`YAxis`: Character, which column of `rowData(se)` should be shown on the y-axis? Defaults to the first entry of `colData(se)`.

`XAxis`: Character, what variable should be shown on the x-axis? Defaults to `"None"`, but can also be `"Row table"`, `"Row data"` or `"Feature name"`.

`XAxisRowData`: Character, which column of `rowData(se)` should be shown on the x-axis if `XAxis="Row data"`? Defaults to the first entry of `rowData(se)`.

Coloring parameters (for points)

`ColorBy`: Character, what type of data should be used for coloring? Defaults to `"None"`, but can also be `"Row table"`, `"Feature name"` or `"Column data"`.

`ColorByDefaultColor`: String specifying the default point colour when `ColorBy="None"`. Defaults to `"black"`.

`ColorByFeatName`: Integer, the index of the feature to use for colouring based on expression, if `ColorBy="Feature name"`? Defaults to 1, i.e., the first feature in `se`. Alternatively, a string can be supplied containing the name of the feature.

`ColorByRowTable`: Character, which row statistic table should be used to choose a feature to color by, if `ColorBy="Feature name"`? Any setting will override `ColorByFeatName` upon initialization of the app. Defaults to `"---"`, which means that no table will be used.

For the plots where each point represents a sample (i.e., all plots except for heatmaps and row data plots), the following additional options apply:

`ColorByColData`: Character, which column of `colData(se)` should be used for colouring if `ColorBy="Column data"`? Defaults to the first entry of `colData(se)`.

`ColorByFeatNameAssay`: Integer, which assay should be used to supply the expression values for colouring if `ColorBy="Feature name"`? Defaults to 1, i.e., the first assay in `se`. Alternatively, a string can also be supplied containing the name of the assay, if `assays(se)` has names.

For plots where each point represents a feature (i.e., row data plots), the following additional options apply:

`ColorByRowData`: Character, which column of `rowData(se)` should be used for colouring if `ColorBy="Row data"`? Defaults to the first entry of `rowData(se)`.

`ColorByFeatNameColor`: String specifying the colour to be used to highlight the selected feature from the text if `ColorBy="Feature name"`. Defaults to `"red"`.

Other plot parameters (for point-based plots)

`DataBoxOpen`: Logical, should the data parameter box be open upon initialization? Defaults to `FALSE`.

`VisualBoxOpen`: Logical, should the visual parameter box be open upon initialization? Defaults to `FALSE`.

`VisualChoices`: A list containing one character vector, specifying the visual box parameters to be shown upon initialization. This defaults to `list("Color")` to show only the coloring parameters, but the internal vector can also contain `"Points"` and `"Other"`.

`PointSize`: Numeric, the size of the points. Defaults to 1.

`PointAlpha`: Numeric, what level of transparency should be used for the points? Ignored when `SelectEffect="Transparent"` and the transmitting plot has a non-`NULL` selection of points. Defaults to 1.

`Downsample`: Logical, indicating whether downsampling of overlapping points should be performed. Defaults to `FALSE`.

`SampleRes`: Numeric, specifying the downsampling resolution, i.e., the granularity at which points are considered to overlap. Higher values result in a more stringent definition of overlaps, and thus less downsampling. Defaults to 200.

`FontSize`: Numeric, size of the font. Defaults to 1.

`LegendPosition`: String specifying the legend position. Defaults to `"Bottom"` but can also be `"Right"`.

`ZoomData`: A list containing numeric vectors of length 4, containing values with names `"xmin"`, `"xmax"`, `"ymin"` and `"ymax"`. These define the zoom window on the x- and y-axes. Each element of the list defaults to `NULL`, i.e., no zooming is performed.

Row statistics table parameters

`Selected`: Integer, containing the index of the row to be initially selected. Defaults to the first row, i.e., 1. Alternatively, a string can be supplied containing the row name.

`Search`: Character, containing the initial value of the search field. Defaults to an empty string.

`SearchColumns`: A list containing character vectors of length equal to the number of columns in `rowData(se)`, specifying the initial value of the search field for each column. All entries default to an empty string.

Heatmap parameters

The features/rows to be used in the construction of the heatmap are specified with:

`FeatName`: List of length equal to the number of panels. Each list entry corresponds to a panel and should be an integer vector with the indices of the feature(s) for which to show the expression in the heatmap. Defaults to `1L` for each panel, i.e., the first feature in `se`. Alternatively, a character vector can be supplied containing the names of the features.

`Assay`: Integer, which assay should be used to supply the expression values shown on the y-axis? Defaults to 1, i.e., the first assay in `se`. Alternatively, a string can also be supplied containing the name of the assay, if `assays(se)` has names.

`FeatNameSource`: Character, which other panel should be used to choose the features to show in the heatmap? Defaults to `"---"`, which means that no panel is used for feature selection.

`FeatNameBoxOpen`: Logical, should the feature selection box be open upon initialization? Defaults to `FALSE`.

The column metadata variables control the ordering of the samples in the heatmap. They can be controlled with:

`ColData`: List of length equal to the number of panels. Each list entry corresponds to a panel and should contain a character vector specifying the field(s) of `colData(se)` that should be used to order the samples in the heatmap. Note that these fields will also appear as annotation bars. Each character vector defaults to the first entry of `colData(se)`.

`ColDataBoxOpen`: Logical, should the column data selection box be open upon initialization? Defaults to `FALSE`.

A variety of parameters are available to control the color scale of the heatmap. They can be specified with:

`ColorBoxOpen`: Logical, should the color selection panel for the heatmap be open upon initialization? Defaults to `FALSE`.

`CenterScale`: List of length equal to the number of panels. Each list entry corresponds to a panel and contains a character vector specifying whether each row of expression values should be mean-centered and/or scaled to unit variance. Defaults to `"Centered"`. Users can set it to `c("Centered", "Scaled")` to obtain mean-centered and unit-scaled rows.

`Lower`: Numeric, what should be the lower bound of the color scale for the values in the heatmap? All values below this threshold will be shown in the same color. Defaults to -Inf, meaning that the lowest value in the data matrix will be used.

`Upper`: Numeric, what should be the upper bound of the color scale for the values in the heatmap? All values above this threshold will be shown in the same color. Defaults to Inf, meaning that the highest value in the data matrix will be used.

`ColorScale`: Character, what color scale (in the form low-mid-high) should be used to color the heatmap when values are centered? Defaults to `"purple-black-yellow"`.

Finally, the `ZoomData` field for heatmaps should contain an integer vector of consecutive indices to zoom into from the full heatmap. This vector will subset the entries in `FeatName` for a given panel. This defaults to `NULL`, i.e., all specified features in `FeatName` are shown.

Point selection parameters

For the point-based plots, the following options apply:

`SelectBoxOpen`: Logical, should the point selection parameter box be open upon initialization? Defaults to `FALSE`.

`SelectByPlot`: Character, which other plot should be used for point selection in the current plot? Defaults to `"---"`, which means that no plot is used for point selection.

`SelectEffect`: Character, what is the effect of receiving point selection information? Can be `"Restrict"`, where only the selected points are shown; `"Color"`, where the selected points have a different color; or `"Transparent"`, where all points other than those selected are made transparent. Defaults to `"Transparent"`.

`SelectColor`: Character, what color should be used for the selected points when `SelectEffect="Color"`? Defaults to `"red"`.

`SelectAlpha`: Numeric, what level of transparency should be used for the unselected points when `SelectEffect="Transparent"`? This should lie in [0, 1], where 0 is fully transparent and 1 is fully opaque. Defaults to 0.1.

Note that point selection cannot occur between row data plots and the other plot types. This is because each point in a row data plot is a feature, while each point represents a sample in the other plots. Point selection can only occur between plots of the same point type.

For the row statistics tables, the following options apply:

`SelectBoxOpen`: Logical, should the point selection parameter box be open upon initialization? Defaults to `FALSE`.

`SelectByPlot`: Character, which other plot should be used to select features in the current table? Defaults to `"---"`, which means that no plot is used for point selection.

Only row data plots can be used for selecting points to supply to tables.

Examples

```
library(scRNAseq)
data(allen)
class(allen)

library(scater)
sce <- as(allen, "SingleCellExperiment")
counts(sce) <- assay(sce, "tophat_counts")
sce <- normalize(sce)
sce <- runPCA(sce)
sce
```

```
redDimPlotDefaults(sce, number=5)
featAssayPlotDefaults(sce, number=5)
colDataPlotDefaults(sce, number=5)
rowStatTableDefaults(sce, number=5)
rowDataPlotDefaults(sce, number=5)
heatMapPlotDefaults(sce, number=5)
```

`ExperimentColorMap-class` *ExperimentColorMap class*

Description

ExperimentColorMap class

Usage

```
ExperimentColorMap(assays = list(), colData = list(), rowData = list(),
  all_discrete = list(assays = NULL, colData = NULL, rowData = NULL),
  all_continuous = list(assays = NULL, colData = NULL, rowData = NULL),
  global_discrete = NULL, global_continuous = NULL, ...)
```

Arguments

`assays`	List of color maps for `assays`.
`colData`	List of color maps for `colData`.
`rowData`	List of color maps for `rowData`.
`all_discrete`	Discrete color maps applied to all undefined `assays`, `colData`, and `rowData`, respectively.
`all_continuous`	Continuous color maps applied to all undefined `assays`, `colData`, and `rowData`, respectively.
`global_discrete`	Discrete color maps applied to all undefined discrete covariates.
`global_continuous`	Continuous color maps applied to all undefined discrete covariates.
`...`	additional arguments passed on to the `ExperimentColorMap` constructor

Details

Color maps must all be functions that take at least one argument: the number of (named) colours to return as a `character` vector. This argument may be ignored in the body of the color map function to produce constant color maps.

Value

An object of class `ExperimentColorMap`

Accessors

In the following code snippets, `x` is an `ExperimentColorMap` object. If the color map can not immediately be found in the appropriate slot, `discrete` is a `logical(1)` that indicates whether the default color map returned should be discrete `TRUE` or continuous (`FALSE`, default).

`assayColorMap(x, i, ..., discrete=FALSE)`**:** Get an `assays` colormap.

`colDataColorMap(x, i, ..., discrete=FALSE)`**:** Get a `colData` colormap.

`rowDataColorMap(x, i, ..., discrete=FALSE)`**:** Get a `rowData` colormap.

Setters

In the following code snippets, `x` is an `ExperimentColorMap` object, and .

`assayColorMap(x, i, ...) <- value`**:** Set an `assays` colormap.

`colDataColorMap(x, i, ...) <- value`**:** Set a `colData` colormap.

`rowDataColorMap(x, i, ...) <- value`**:** Set a `rowData` colormap.

`assay(x, i, ...) <- value`**:** Alias. Set an `assays` colormap.

Examples

```
# Example color maps ----

count_colors <- function(n){
  c("black","brown","red","orange","yellow")
}
fpkm_colors <- viridis::inferno
tpm_colors <- viridis::plasma

qc_color_fun <- function(n){
  qc_colors <- c("forestgreen", "firebrick1")
  names(qc_colors) <- c("Y", "N")
  return(qc_colors)
}

# Constructor ----

ecm <- ExperimentColorMap(
    assays = list(
        counts = count_colors,
        tophat_counts = count_colors,
        cufflinks_fpkm = fpkm_colors,
        rsem_tpm = tpm_colors
    ),
    colData = list(
        passes_qc_checks_s = qc_color_fun
    )
)

# Accessors ----

# assay color maps
assayColorMap(ecm, "logcounts") # [undefined --> default]
assayColorMap(ecm, "counts")
```

```
assayColorMap(ecm, "cufflinks_fpkm")
assay(ecm, "cufflinks_fpkm") # alias

# colData color maps
colDataColorMap(ecm, "passes_qc_checks_s")
colDataColorMap(ecm, "undefined")

# rowData color maps
rowDataColorMap(ecm, "undefined")

# generic accessors
assays(ecm)
assayNames(ecm)

# Setters ----

assayColorMap(ecm, "counts") <- function(n){c("blue","white","red")}
assay(ecm, 1) <- function(n){c("blue","white","red")}

colDataColorMap(ecm, "passes_qc_checks_s") <- function(n){NULL}
rowDataColorMap(ecm, "undefined") <- function(n){NULL}
```

`isColorMapCompatible` *Check compatibility between ExperimentColorMap and SummarizedExperiment objects*

Description

This function compares a pair of ExperimentColorMap and SingleCellExperiment objects, and examines whether all of the `assays`, `colData`, and `rowData` defined in the ExperimentColorMap object exist in the SingleCellExperiment object.

Usage

```
isColorMapCompatible(ecm, se, error = FALSE)
```

Arguments

`ecm`	An ExperimentColorMap.
`se`	A SingleCellExperiment.
`error`	A logical value that indicates whether an informative error should be thrown, describing why the two objects are not compatible.

Value

A logical value that indicates whether a given pair of ExperimentColorMap and SummarizedExperiment objects are compatible. If `error=TRUE`, an informative error is thrown, rather than returing `FALSE`.

Author(s)

Kevin Rue-Albrecht

Examples

```
# Example color maps ----

count_colors <- function(n){
  c("black","brown","red","orange","yellow")
}

qc_color_fun <- function(n){
  qc_colors <- c("forestgreen", "firebrick1")
  names(qc_colors) <- c("Y", "N")
  return(qc_colors)
}

ecm <- ExperimentColorMap(
    assays = list(
        tophat_counts = count_colors
    ),
    colData = list(
        passes_qc_checks_s = qc_color_fun
    )
)

# Example SingleCellExperiment ----

library(scRNAseq)
data(allen)
library(scater)
sce <- as(allen, "SingleCellExperiment")

# Test for compatibility ----

isColorMapCompatible(ecm, sce)
```

`iSEE` *iSEE: interactive SummarizedExperiment/SingleCellExperiment Explorer*

Description

Interactive and reproducible visualization of data contained in a SummarizedExperiment/SingleCellExperiment, using a Shiny interface.

Usage

```
iSEE(se, redDimArgs = NULL, colDataArgs = NULL, featAssayArgs = NULL,
  rowStatArgs = NULL, rowDataArgs = NULL, heatMapArgs = NULL,
  redDimMax = 5, colDataMax = 5, featAssayMax = 5, rowStatMax = 5,
  rowDataMax = 5, heatMapMax = 5, initialPanels = NULL, annotFun = NULL,
  colormap = ExperimentColorMap(), tour = NULL, appTitle = NULL,
  runLocal = TRUE)
```

Arguments

se	An object that is coercible to SingleCellExperiment.
redDimArgs	A DataFrame similar to that produced by redDimPlotDefaults, specifying initial parameters for the plots.
colDataArgs	A DataFrame similar to that produced by colDataPlotDefaults, specifying initial parameters for the plots.
featAssayArgs	A DataFrame similar to that produced by featAssayPlotDefaults, specifying initial parameters for the plots.
rowStatArgs	A DataFrame similar to that produced by rowStatTableDefaults, specifying initial parameters for the plots.
rowDataArgs	A DataFrame similar to that produced by rowDataPlotDefaults, specifying initial parameters for the plots.
heatMapArgs	A DataFrame similar to that produced by heatMapPlotDefaults, specifying initial parameters for the plots.
redDimMax	An integer scalar specifying the maximum number of reduced dimension plots in the interface.
colDataMax	An integer scalar specifying the maximum number of column data plots in the interface.
featAssayMax	An integer scalar specifying the maximum number of feature assay plots in the interface.
rowStatMax	An integer scalar specifying the maximum number of row statistics tables in the interface.
rowDataMax	An integer scalar specifying the maximum number of row data plots in the interface.
heatMapMax	An integer scalar specifying the maximum number of heatmaps in the interface.
initialPanels	A DataFrame specifying which panels should be created at initialization. This should contain a Name character field and may have optional Width and Height integer fields, see Details.
annotFun	A function, constructed in the form of annotateEntrez or annotateEnsembl. This function is built in a way to generate itself a function supposed to accept two parameters, se and row_index, to have a unified yet flexible interface to generate additional information to display for the selected genes of interest.
colormap	An ExperimentColorMap object that defines custom color maps to apply to individual assays, colData, and rowData covariates.
tour	A data.frame with the content of the interactive tour to be displayed after starting up the app. Defaults to NULL. More information is provided in the details.
appTitle	A string indicating the title to be displayed in the app. If not provided, the app displays the version info of iSEE. Users can specify this to provide a compact description of the dataset, or the PubMedID for the input data.
runLocal	A logical indicating whether the app is to be run locally or remotely on a server, which determines how documentation will be accessed.

Details

Users can pass default parameters via DataFrame objects in redDimArgs and featAssayArgs. Each object can contain some or all of the expected fields (see redDimPlotDefaults). Any missing fields will be filled in with the defaults.

The number of maximum plots for each type of plot is set to the larger of `*Max` and `nrow(*Args)`. Users can specify any number of maximum plots, though increasing the number will increase the time required to render the interface.

The `initialPanels` argument specifies the panels to be created upon initializing the interface. This should be a DataFrame containing a `Name` field specifying the identity of the panel, e.g., `"Reduced dimension plot 1"`, `"Row statistics table 2"`. The trailing number should not be greater than the number of maximum plots of that type. Users can also define the `Width` field, specifying the width of each panel from 2 to 12 (values will be coerced inside this range); and the `Height` field, specifying the height of each panel from 400 to 1000 pixels. By default, one panel of each type (where possible) will be generated, with height of 500 and width of 4.

The `tour` argument needs to be provided in a form compatible with the format expected by the `rintrojs` package, where the variables `element` and `intro` have to be present. See more information at https://github.com/carlganz/rintrojs#usage regarding the values that the `element` can assume for proper displaying.

Value

A Shiny App is launched for interactive data exploration of the SummarizedExperiment/SingleCellExperiment object

Examples

```
library(scRNAseq)
data(allen)
class(allen)

# Example data ----

library(scater)
sce <- as(allen, "SingleCellExperiment")
counts(sce) <- assay(sce, "tophat_counts")
sce <- normalize(sce)

sce <- runPCA(sce)
sce <- runTSNE(sce)
rowData(sce)$ave_count <- rowMeans(counts(sce))
rowData(sce)$n_cells <- rowSums(counts(sce)>0)
sce

# launch the app itself ----

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

`iSEE-pkg` *iSEE: interactive SummarizedExperiment/SingleCellExperiment Explorer*

Description

iSEE is a Bioconductor package that provides an interactive Shiny-based graphical user interface for exploring data stored in SummarizedExperiment objects, including row- and column-level metadata. Particular attention is given to single-cell data in a SingleCellExperiment object with visualization of dimensionality reduction results, e.g., from principal components analysis (PCA) or t-distributed stochastic neighbour embedding (t-SNE)

Author(s)

Aaron Lun <alun@wehi.edu.au>

Charlotte Soneson <charlotte.soneson@uzh.ch>

Federico Marini <marinif@uni-mainz.de>

Kevin Rue-Albrecht <kevin.rue-albrecht@kennedy.ox.ac.uk>

---

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

Description

App pre-configured to link multiple feature assay plots

Usage

```
modeGating(se, features, featAssayMax = max(2, nrow(features)), ...,
  plot_width = 4)
```

Arguments

se	An object that coercible to SingleCellExperiment
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.
featAssayMax	Maximal number of feature assay plots in the app.
...	Additional arguments passed to iSEE.
plot_width	The grid width of linked plots (numeric vector of length either 1 or equal to nrow(features)

Value

A Shiny App preconfigured with multiple chain-linked feature expression plots is launched for interactive data exploration of the SingleCellExperiment / SummarizedExperiment object

Examples

```
library(scRNAseq)
data(allen)
class(allen)

# Example data ----

library(scater)
```

```
sce <- as(allen, "SingleCellExperiment")
counts(sce) <- assay(sce, "tophat_counts")
sce <- normalize(sce)


# Select top variable genes ----

plot_count <- 6
rv <- rowVars(logcounts(sce))
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, featAssayMax = 6)
if (interactive()) {
  shiny::runApp(app, port = 1234)
}
```

`subsetPointsByGrid` *Subset points for faster plotting*

Description

Subset points using a grid-based system, to avoid unnecessary rendering when plotting.

Usage

```
subsetPointsByGrid(X, Y, resolution = 200)
```

Arguments

`X`	A numeric vector of x-coordinates for all points.
`Y`	A numeric vector of y-coordinates for all points, of the same length as `X`.
`resolution`	A positive integer specifying the number of bins on each axis of the grid.

Details

This function will define a grid of the specified resolution across the plot. Each point is allocated to a grid location (i.e., pair of bins on the x- and y-axes). If multiple points are allocated to a given location, only the last/right-most point is retained. This mimics the fact that plotting will overwrite earlier points with later points. In this manner, we can avoid unnecessary rendering of earlier points that would not show up anyway.

Note that the `resolution` should be a positive integer less than the square root of the maximum integer size, and will be coerced into the valid range if necessary. This enables fast calculation of the grid locations for all points.

For plots where `X` and `Y` are originally categorical, use the jittered versions as input to this function.

Value

A logical vector indicating which points should be retained.

Author(s)

Aaron Lun

Examples

```
X <- rnorm(100000)
Y <- X + rnorm(100000)

summary(subsetPointsByGrid(X, Y, resolution=100))

summary(subsetPointsByGrid(X, Y, resolution=200))

summary(subsetPointsByGrid(X, Y, resolution=1000))
```

---

synchronizeAssays *Synchronize assay colormaps to match those in a SummarizedExperiment*

Description

This function returns an updated ExperimentColorMap in which colormaps in the `assays` slot are ordered to match the position of their corresponding assay in the SingleCellExperiment object. Assays in the SingleCellExperiment that do not have a match in the ExperimentColorMap are assigned the appropriate default colormap.

Usage

```
synchronizeAssays(ecm, se)
```

Arguments

`ecm`	An ExperimentColorMap.
`se`	A SingleCellExperiment.

Details

It is highly recommended to name *all* assays in both ExperimentColorMap and SummarizedExperiment prior to calling this function, as this will facilitate the identification of matching assays between the two objects. In most cases, unnamed colormaps will be dropped from the new ExperimentColorMap object.

The function supports three main situations:

- If *all* assays in the SingleCellExperiment are named, this function will populate the `assays` slot of the new ExperimentColorMap with the name-matched colormap from the input ExperimentColorMap, if available. Assays in the SingleCellExperiment that do not have a colormap defined in the ExperimentColorMap are assigned the appropriate default colormap.

- If *all* assays in the SingleCellExperiment are unnamed, this function requires that the ExperimentColorMap supplies a number of assay colormaps *identical* to the number of assays in the SingleCellExperiment object. In that case, the ExperimentColorMap object will be returned *as is*.
- If only a subset of assays in the SingleCellExperiment are named, this function will ignore unnamed colormaps in the ExperimentColorMap; It will populate the `assays` slot of the new ExperimentColorMap with the name-matched colormap from the input ExperimentColorMap, if available. Assays in the SingleCellExperiment that are unnamed, or that do not have a colormap defined in the ExperimentColorMap are assigned the appropriate default colormap.

Value

An ExperimentColorMap with color maps in the `assay` slot synchronized to match the position of the corresponding assay in the SingleCellExperiment.

Author(s)

Kevin Rue-Albrecht

Examples

```
# Example ExperimentColorMap ----

count_colors <- function(n){
  c("black","brown","red","orange","yellow")
}
fpkm_colors <- viridis::inferno

ecm <- ExperimentColorMap(
    assays = list(
        counts = count_colors,
        tophat_counts = count_colors,
        cufflinks_fpkm = fpkm_colors,
        rsem_counts = count_colors,
        orphan = count_colors,
        orphan2 = count_colors,
        count_colors,
        fpkm_colors
    )
)

# Example SingleCellExperiment ----

library(scRNAseq)
data(allen)
library(scater)
sce <- as(allen, "SingleCellExperiment")
counts(sce) <- assay(sce, "tophat_counts")
sce <- normalize(sce)
sce <- runPCA(sce)
sce <- runTSNE(sce)

# Example ----

ecm_sync <- synchronizeAssays(ecm, sce)
```

Index

annotateEnsembl, 2, *13*
annotateEntrez, 3, *13*
assay,ExperimentColorMap,character-method
(ExperimentColorMap-class), 9
assay,ExperimentColorMap,numeric-method
(ExperimentColorMap-class), 9
assayColorMap
(ExperimentColorMap-class), 9
assayColorMap,ExperimentColorMap,character-method
(ExperimentColorMap-class), 9
assayColorMap,ExperimentColorMap,numeric-method
(ExperimentColorMap-class), 9
assayColorMap<-
(ExperimentColorMap-class), 9
assayColorMap<-,ExperimentColorMap,character-method
(ExperimentColorMap-class), 9
assayColorMap<-,ExperimentColorMap,numeric-method
(ExperimentColorMap-class), 9
assayNames,ExperimentColorMap-method
(ExperimentColorMap-class), 9
assayNames<-,ExperimentColorMap,ANY-method
(ExperimentColorMap-class), 9
assays,ExperimentColorMap-method
(ExperimentColorMap-class), 9
assays<-,ExperimentColorMap,list-method
(ExperimentColorMap-class), 9

class:ExperimentColorMap
(ExperimentColorMap-class), 9
colData,ExperimentColorMap-method
(ExperimentColorMap-class), 9
colData<-,ExperimentColorMap,ANY-method
(ExperimentColorMap-class), 9
colDataColorMap
(ExperimentColorMap-class), 9
colDataColorMap,ExperimentColorMap,character-method
(ExperimentColorMap-class), 9
colDataColorMap<-
(ExperimentColorMap-class), 9
colDataColorMap<-,ExperimentColorMap,character-method
(ExperimentColorMap-class), 9
colDataPlotDefaults, *13*
colDataPlotDefaults *(*defaults*)*, 4

defaults, 4

ExperimentColorMap, *11*, *13*, *17, 18*
ExperimentColorMap
(ExperimentColorMap-class), 9
ExperimentColorMap-class, 9

featAssayPlotDefaults, *13*
featAssayPlotDefaults *(*defaults*)*, 4

heatMapPlotDefaults, *13*
heatMapPlotDefaults *(*defaults*)*, 4

isColorMapCompatible, 11
iSEE, 12, *13*, *15*
iSEE-pkg, 14
iSEE-pkg-package *(*iSEE-pkg*)*, 14
modeGating, 15

redDimPlotDefaults, *13*
redDimPlotDefaults *(*defaults*)*, 4
rowData,ExperimentColorMap-method
(ExperimentColorMap-class), 9
rowData<-,ExperimentColorMap,ANY-method
(ExperimentColorMap-class), 9
rowDataColorMap
(ExperimentColorMap-class), 9
rowDataColorMap,ExperimentColorMap,character-method
(ExperimentColorMap-class), 9
rowDataColorMap<-
(ExperimentColorMap-class), 9
rowDataColorMap<-,ExperimentColorMap,character-method
(ExperimentColorMap-class), 9
rowDataPlotDefaults, *13*
rowDataPlotDefaults *(*defaults*)*, 4
rowStatTableDefaults, *13*
rowStatTableDefaults *(*defaults*)*, 4

SingleCellExperiment, *2, 3*, *11*, *13–15*, *17*
subsetPointsByGrid, 16
SummarizedExperiment, *14, 15*
synchronizeAssays, 17