Package 'spatialHeatmap'

October 18, 2022

Type Package

Title spatialHeatmap

Version 2.2.0

Description The spatialHeatmap package provides functionalities for visualizing cell-, tissue- and organ-specific data of biological assays by coloring the corresponding spatial features defined in anatomical images according to a numeric color key.

License Artistic-2.0

Encoding UTF-8

biocViews Spatial, Visualization, Microarray, Sequencing, GeneExpression, DataRepresentation, Network, Clustering, GraphAndNetwork, CellBasedAssays, ATACSeq, DNASeq, TissueMicroarray, SingleCell, CellBiology, GeneTarget

VignetteBuilder knitr

Suggests knitr, rmarkdown, BiocStyle, BiocSingular, RUnit, BiocGenerics, ExpressionAtlas, DT, Biobase, GEOquery, shinyWidgets, shinyjs, htmltools, shinyBS, sortable

Depends R (>= 3.5.0)

Imports av, BiocParallel, BiocFileCache, data.table, DESeq2, distinct, edgeR, WGCNA, flashClust, htmlwidgets, genefilter, ggplot2, ggdendro, grImport, grid, gridExtra, gplots, igraph, HDF5Array, limma, methods, magick, Matrix, rsvg, shiny, dynamicTreeCut, grDevices, graphics, ggplotify, parallel, plotly, pROC, rols, rappdirs, reshape2, scater, scuttle, scran, stats, SummarizedExperiment, SingleCellExperiment, shinydashboard, S4Vectors, utils, visNetwork, UpSetR, xml2, yaml

BugReports https://github.com/jianhaizhang/spatialHeatmap/issues

URL https://github.com/jianhaizhang/spatialHeatmap

RoxygenNote 7.1.2

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

git_branch RELEASE_3_15

git_last_commit b145080

git_last_commit_date 2022-04-26

Date/Publication 2022-10-18

Author Jianhai Zhang [aut, trl, cre],
Jordan Hayes [aut],
Le Zhang [aut],
Bing Yang [aut],
Wolf Frommer [aut],
Julia Bailey-Serres [aut],
Thomas Girke [aut]

Maintainer Jianhai Zhang <jzhan067@ucr.edu>

R topics documented:

`spatialHeatmap-package`

spatialHeatmap Spatial Heatmap, Matrix Heatmap, Network

Description

The spatialHeatmap package provides functionalities for visualizing cell-, tissue- and organ-specific data of biological assays by coloring the corresponding spatial features defined in anatomical images according to a numeric color key.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

The spatialHeatmap package provides functionalities for visualizing cell-, tissue- and organ-specific data of biological assays by coloring the corresponding spatial features defined in anatomical images according to a numeric color key. The color scheme used to represent the assay values can be customized by the user. This core functionality is called a spatial heatmap plot. It is enhanced with nearest neighbor visualization tools for groups of measured items (e.g. gene modules) sharing related abundance profiles, including matrix heatmaps combined with hierarchical clustering dendrograms and network representations. The functionalities of spatialHeatmap can be used either in a command-driven mode from within R or a graphical user interface (GUI) provided by a Shiny App that is also part of this package. While the R-based mode provides flexibility to customize and automate analysis routines, the Shiny App includes a variety of convenience features that will appeal to many biologists. Moreover, the Shiny App has been designed to work on both local computers as well as server-based deployments (e.g. cloud-based or custom servers) that can be accessed remotely as a centralized web service for using spatialHeatmap's functionalities with community and/or private data.

As anatomical images the package supports both tissue maps from public repositories and custom images provided by the user. In general any type of image can be used as long as it can be provided in SVG (Scalable Vector Graphics) format, where the corresponding spatial features have been defined (see aSVG below). The numeric values plotted onto a spatial heatmap are usually quantitative measurements from a wide range of profiling technologies, such as microarrays, next generation sequencing (e.g. RNA-Seq and scRNA-Seq), proteomics, metabolomics, or many other small- or large-scale experiments. For convenience, several preprocessing and normalization methods for the most common use cases are included that support raw and/or preprocessed data. Currently, the main application domains of the spatialHeatmap package are numeric data sets and spatially mapped images from biological and biomedical areas. Moreover, the package has been designed to also work with many other spatial data types, such a population data plotted onto geographic maps. This high level of flexibility is one of the unique features of spatialHeatmap. Related software tools for biological applications in this field are largely based on pure web applications (Winter et al. 2007; Waese et al. 2017) or local tools (Maag 2018; Muschelli, Sweeney, and Crainiceanu 2014) that typically lack customization functionalities. These restrictions limit users to utilizing pre-existing expression data and/or fixed sets of anatomical image collections. To close this gap for biological use cases, we have developed spatialHeatmap as a generic R/Bioconductor package for plotting quantitative values onto any type of spatially mapped images in a programmable environment and/or in an intuitive to use GUI application.

Author(s)

NA Author: NA Jianhai Zhang (PhD candidate at Genetics, Genomics and Bioinformatics, University of California, Riverside), Dr. Thomas Girke (Professor at Department of Botany and Plant Sciences, University of California, Riverside) Maintainer: NA Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>.

References

https://www.w3schools.com/graphics/svg_intro.asp

https://shiny.rstudio.com/tutorial/

https://shiny.rstudio.com/articles/datatables.html

https://rstudio.github.io/DT/010-style.html

https://plot.ly/r/heatmaps/

https://www.gimp.org/tutorials/

https://inkscape.org/en/doc/tutorials/advanced/tutorial-advanced.en.html

http://www.microugly.com/inkscape-quickguide/

https://cran.r-project.org/web/packages/visNetwork/vignettes/Introduction-to-visNetwork.html

https://github.com/ebi-gene-expression-group/anatomogram/tree/master/src/svg

Winter, Debbie, Ben Vinegar, Hardeep Nahal, Ron Ammar, Greg V Wilson, and Nicholas J Provart. 2007. "An 'Electronic Fluorescent Pictograph' Browser for Exploring and Analyzing Large-Scale Biological Data Sets." PLoS One 2 (8): e718

Waese, Jamie, Jim Fan, Asher Pasha, Hans Yu, Geoffrey Fucile, Ruian Shi, Matthew Cumming, et al. 2017. "EPlant: Visualizing and Exploring Multiple Levels of Data for Hypothesis Generation in Plant Biology." Plant Cell 29 (8): 1806–21

Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angelica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9

Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas

Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8

McCarthy, Davis J., Chen, Yunshun, Smyth, and Gordon K. 2012. "Differential Expression Analysis of Multifactor RNA-Seq Experiments with Respect to Biological Variation." Nucleic Acids Research 40 (10): 4288–97

Maag, Jesper L V. 2018. "Gganatogram: An R Package for Modular Visualisation of Anatograms and Tissues Based on Ggplot2." F1000Res. 7 (September): 1576

Muschelli, John, Elizabeth Sweeney, and Ciprian Crainiceanu. 2014. "BrainR: Interactive 3 and 4D Images of High Resolution Neuroimage Data." R J. 6 (1): 41–48

Morgan, Martin, Valerie Obenchain, Jim Hester, and Hervé Pagès. 2018. SummarizedExperiment: SummarizedExperiment Container

Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2017). shiny: Web Application Framework for R. R package version 1.0.3. https://CRAN.R-project.org/package=shiny

Winston Chang and Barbara Borges Ribeiro (2017). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.6.1. https://CRAN.R-project.org/package=shinydashboard

Paul Murrell (2009). Importing Vector Graphics: The grImport Package for R. Journal of Statistical Software, 30(4), 1-37. URL http://www.jstatsoft.org/v30/i04/.

Jeroen Ooms (2017). rsvg: Render SVG Images into PDF, PNG, PostScript, or Bitmap Arrays. R package version 1.1. https://CRAN.R-project.org/package=rsvg

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Yihui Xie (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2. https://CRAN.R-project.org/package=DT

Baptiste Auguie (2016). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.2.1. https://CRAN.R-project.org/package=gridExtra

Andrie de Vries and Brian D. Ripley (2016). ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'. R package version 0.1-20. https://CRAN.R-project.org/package=ggdendro

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 doi:10.1186/1471-2105-9-559

Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/.

Simon Urbanek and Jeffrey Horner (2015). Cairo: R graphics device using cairo graphics library for creating high-quality bitmap (PNG, JPEG, TIFF), vector (PDF, SVG, PostScript) and display (X11 and Win32) output. R package version 1.5-9. https://CRAN.R-project.org/package=Cairo

R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Duncan Temple Lang and the CRAN Team (2017). XML: Tools for Parsing and Generating XML Within R and S-Plus. R package version 3.98-1.9. https://CRAN.R-project.org/package=XML

Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec and Pedro Despouy (NA). plotly: Create Interactive Web Graphics via 'plotly.js'. https://plot.ly/r, https://cpsievert.github.io/plotly_book/, https://github.com/ropensci/plotly.

Matt Dowle and Arun Srinivasan (2017). data.table: Extension of 'data.frame'. R package version 1.10.4. https://CRAN.R-project.org/package=data.table

R. Gentleman, V. Carey, W. Huber and F. Hahne (2017). genefilter: genefilter: methods for filtering genes from high-throughput experiments. R package version 1.58.1.

Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/.

Almende B.V., Benoit Thieurmel and Titouan Robert (2017). visNetwork: Network Visualization using 'vis.js' Library. R package version 2.0.1. https://CRAN.R-project.org/package=visNetwork

Lori Shepherd and Martin Morgan (2020). BiocFileCache: Manage Files Across Sessions. R package version 1.12.1.

See Also

norm_data, aggr_rep, filter_data, spatial_hm, submatrix, adj_mod, matrix_hm, network, return_feature, update_feature, shiny_shm, custom_shiny

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as a
## "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.

## Set up toy data.

# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann); df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be made
# based on the experiment design, which is accessible through the accession number "E-MTAB-6769"
# in the R package ExpressionAtlas. An example targets file is included in this package and
```

```
# accessed below.
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".

library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

## As conventions, raw sequencing count data should be normalized, aggregated, and filtered to
## reduce noise.

# Normalize count data.
# The normalizing function "calcNormFactors" (McCarthy et al. 2012) with default settings is used.
df.nor.chk <- norm_data(data=df.chk, norm.fun='CNF', log2.trans=TRUE)
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', log2.trans=TRUE)
# Aggregate count data.
# Aggregate "sample__condition" replicates in toy data1.
df.aggr.chk <- aggr_rep(data=df.nor.chk, aggr='mean')
df.aggr.chk[1:3, ]
# Aggregate "sample_condition" replicates in toy data2, where "sample" is "organism_part" and
# "condition" is "age".
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='organism_part', con.factor='age', aggr='mean')
assay(se.aggr.chk)[1:3, 1:3]
# Filter out genes with low counts and low variance. Genes with counts over 5 (log2 unit) in at
# least 1% samples (pOA), and coefficient of variance (CV) between 0.2 and 100 are retained.
# Filter toy data1.
df.fil.chk <- filter_data(data=df.aggr.chk, pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
# Filter toy data2.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='organism_part', con.factor='age',
pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)

## Spatial heatmaps.

# To make spatial heatmaps, a pair of formatted data and pre-annotated SVG (aSVG) file are
# required. If the data is a "data frame", the formatting is to use the naming scheme
# "sample__condition" in column names. If "SummarizedExperiment", the "sample" and "condition"
# replicates should be defined in the "colData" slot. In the aSVG, each spatial feature has a
# unique identifier. The numeric values are mapped to spatial features and translated into
# colors according to their identifiers programatically. The mapped images are called spatial
# heatmaps.

# The following shows how to download the corresponding pre-annotated aSVG file from the EBI
# SVG repository based on above tissues and species involved, i.e. c('heart', 'brain') and
# c('gallus') respectively. See the function "return_feature" for details. An empty directory
# is recommended so as to avoid overwriting existing SVG files. Here "tmp.dir" is used.

# To meet the package building requirements, the code of querying aSVG remotely is not evaluated.
# The matching aSVG "gallus_gallus.svg" is included in this package and accessed.
```

```
# Make an empty directory "tmp.dir" if not exist.
tmp.dir <- paste0(normalizePath(tempdir(check=TRUE), winslash="/", mustWork=FALSE), '/shm')
# Remote aSVG repos.
data(aSVG.remote.repo)
tmp.dir <- normalizePath(tempdir(check=TRUE), winslash="/", mustWork=FALSE)
tmp.dir.ebi <- paste0(tmp, '/ebi.zip')
tmp.dir.shm <- paste0(tmp, '/shm.zip')
# Download the remote aSVG repos as zip files. According to Bioconductor's
# requirements, downloadings are not allowed inside functions, so the repos are
# downloaded before calling "return_feature".
download.file(aSVG.remote.repo$ebi, tmp.dir.ebi)
download.file(aSVG.remote.repo$shm, tmp.dir.shm)
remote <- list(tmp.dir.ebi, tmp.dir.shm)

# Query aSVGs from remote repos.
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir=tmp.dir,
match.only=FALSE, remote=remote)
feature.df
# The path of matching aSVG.
svg.chk <- paste0(tmp.dir, '/gallus_gallus.svg')


# Get the matching aSVG path from the package.
svg.chk <- system.file("extdata/shinyApp/example", "gallus_gallus.svg",
package="spatialHeatmap")

# Plot spatial heatmaps on gene "ENSGALG00000019846". In the middle are spatial heatmaps. Only
# aSVG features with matching countparts in data are colored. On the right is the legend plot,
# only the matching features are labeled.
# Toy data1.
spatial_hm(svg.path=svg.chk, data=df.fil.chk, ID='ENSGALG00000019846', height=0.4,
legend.r=1.9, sub.title.size=7, ncol=3)
# Save spaital heatmaps as HTML and video files by assigning "tmp.dir" to "out.dir".

tmp.dir <- paste0(normalizePath(tempdir(check=TRUE), winslash="/", mustWork=FALSE), '/shm')
spatial_hm(svg.path=svg.chk, data=df.fil.chk, ID='ENSGALG00000019846', height=0.4, legend.r=1.9,
sub.title.size=7, ncol=3, out.dir=tmp.dir)

# Toy data2.
spatial_hm(svg.path=svg.chk, data=se.fil.chk, ID='ENSGALG00000019846', legend.r=1.9,
legend.nrow=2, sub.title.size=7, ncol=3)

# When plot spatial heatmaps, the data can also come as as a simple vector. The following
# gives an example on a vector of 3 random values.
# Random values.
vec <- sample(1:100, 3)
# Name the vector slots. The last name is assumed as a random sample without a matching
# feature in aSVG.
names(vec) <- c('brain', 'heart', 'notMapped')
vec
# Plot.
```

```
spatial_hm(svg.path=svg.chk, data=vec, ID='geneX', height=0.6, legend.r=1.5, ncol=1)

# Plot spatial heatmaps on aSVGs of two Arabidopsis thaliana development stages.

# Make up a random numeric data frame.
df.test <- data.frame(matrix(sample(x=1:100, size=50, replace=TRUE), nrow=10))
colnames(df.test) <- c('shoot_totalA__condition1', 'shoot_totalA__condition2',
'shoot_totalB__condition1', 'shoot_totalB__condition2', 'notMapped')
rownames(df.test) <- paste0('gene', 1:10) # Assign row names
df.test[1:3, ]

# aSVG of development stage 1.
svg1 <- system.file("extdata/shinyApp/example", "arabidopsis.thaliana_organ_shm1.svg",
package="spatialHeatmap")
# aSVG of development stage 2.
svg2 <- system.file("extdata/shinyApp/example", "arabidopsis.thaliana_organ_shm2.svg",
package="spatialHeatmap")
# Spatial heatmaps.
spatial_hm(svg.path=c(svg1, svg2), data=df.test, ID=c('gene1'), height=0.8, legend.r=1.6,
preserve.scale=TRUE)

## If users want to use custom identifiers for spatial features in the aSVG file, the function
# "update_feature" should be used. For illustration purpose, the aSVG "gallus_gallus.svg" in
# this package is copied to 'tmp.dir' as example.


# Make an empty directory "tmp.dir" if not exist.
tmp.dir <- paste0(normalizePath(tempdir(check=TRUE), winslash="/", mustWork=FALSE), '/shm')
# Make a copy of "gallus_gallus.svg".
file.copy(from=svg.chk, to=tmp.dir, overwrite=FALSE)
# Remote aSVG repos.
data(aSVG.remote.repo)
tmp.dir <- normalizePath(tempdir(check=TRUE), winslash="/", mustWork=FALSE)
tmp.dir.ebi <- paste0(tmp, '/ebi.zip')
tmp.dir.shm <- paste0(tmp, '/shm.zip')
# Download the remote aSVG repos as zip files. According to Bioconductor's
# requirements, downloadings are not allowed inside functions, so the repos are
# downloaded before calling "return_feature".
download.file(aSVG.remote.repo$ebi, tmp.dir.ebi)
download.file(aSVG.remote.repo$shm, tmp.dir.shm)
remote <- list(tmp.dir.ebi, tmp.dir.shm)

# Query "gallus_gallus.svg" in remote repos.
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir=tmp.dir,
match.only=TRUE, remote=remote)
feature.df

# New features.
ft.new <- c('BRAIN', 'HEART')
# Add new features to the first column.
feature.df.new <- cbind(featureNew=ft.new, feature.df)
feature.df.new
# Update features.
```

```
update_feature(df.new=feature.df.new, dir=tmp.dir)



## Matrix heatmap

# The matrix heatmap and following network are supplements to the core feature of spatial
# heatmap. First, nearest neighbors are selected for each target gene according to correlation
# (default) or distance measure independently. There are three alternative parameters used for
# the selection: "p" is the proportion of top nearest neighbors, "n" is the number of top
# nearest neighbors, and "v" is a specific cutoff value for correlation or distance. Then
# target genes and their nearest neighbors are hierarchically clustered and visualized in
# static or interactive matrix heatmap, where target genes are labeled by black lines. If the
# data is "SummarizedExperiment", the argument "ann" is the column name of gene annotation in
# "rowData" slot. It is only relevant if users want to see annotation when mousing over a node
# in the interactive network below, so it is optional. Here "ann='ann'" is set and the
# corresponding annotation is appended to selected nearest neighbors.

# Select nearest neighbors for target genes 'ENSGALG00000019846' and 'ENSGALG00000000112'.
df.sub.mat <- submatrix(data=df.fil.chk, ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), p=0.1)
se.sub.mat <- submatrix(data=se.fil.chk, ann='ann', ID=c('ENSGALG00000019846',
'ENSGALG00000000112'), p=0.1)

# In the following, "df.sub.mat" and "se.sub.mat" is used in the same way, so only
# "se.sub.mat" illustrated.

# The subsetted matrix is partially shown below.
se.sub.mat[c('ENSGALG00000019846', 'ENSGALG00000000112'), c(1:2, 63)]

# Static matrix heatmap.
matrix_hm(ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), data=se.sub.mat, angleCol=80,
angleRow=35, cexRow=0.8, cexCol=0.8, margin=c(8, 10), static=TRUE,
arg.lis1=list(offsetRow=0.01, offsetCol=0.01))

# Interactive matrix heatmap.
 matrix_hm(ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), data=se.sub.mat,
angleCol=80, angleRow=35, cexRow=0.8, cexCol=0.8, margin=c(8, 10), static=FALSE,
arg.lis1=list(offsetRow=0.01, offsetCol=0.01))

## Network

# Network analysis with WGCNA (Langfelder and Horvath 2008) is applied on the subsetted matix
# visualized in the matrix heatmap. The gene module containing a specifc target gene is
# visualized in static and interactive network graphs. Briefly, a correlation matrix or
# distance matrix is computed on all genes in matrix heatmap, and transformed to an adjacency
# matrix and topological overlap matrix (TOM) sequentially, which are advanced measures to
# quantify coexpression similarity. Then network modules are identified by hierarchinally
# clustering the TOM-transformed dissimilarity matrix 1-TOM, which are clusters of genes with
# highly similar coexpression profiles. The module containing a target gene is finally
# displayed as network graphs. Refer to function "adj_mod" for details.

# Adjacency matrix and module identification
```

```
# The modules are identified by "adj_mod". It returns a list containing an adjacency matrix and
# a data frame of module assignment.
adj.mod <- adj_mod(data=se.sub.mat)

# The adjacency matrix is a measure of co-expression similarity between genes, where larger
# value denotes more similarity.
adj.mod[['adj']][1:3, 1:3]

# The modules are identified at two alternative sensitivity levels (ds=2 or 3). From 2 to 3,
# more modules are identified but module sizes are smaller. The two sets of module assignment
# are returned in a data frame. The first column is ds=2 while the second is ds=3. The numbers
# in each column are module labels, where "0" indicates genes not assigned to any module.
adj.mod[['mod']][1:3, ]

# Static network. In the graph, nodes are genes and edges are adjacencies between genes. The
# thicker edge denotes higher adjacency (co-expression similarity) while larger node indicates
# higher gene connectivity (sum of a gene's adjacency with all its direct neighbors). The target
# gene is labeled by "_target". The node connectivity increases from "turquoise" to "violet",
# and the adjacency increases from "yellow" to "blue".
network(ID="ENSGALG00000019846", data=se.sub.mat, adj.mod=adj.mod, adj.min=0.7,
vertex.label.cex=1.5, vertex.cex=4, static=TRUE)

# Interactive network. Same with static mode, the target gene ID is appended "_target".
 network(ID="ENSGALG00000019846", data=se.sub.mat, adj.mod=adj.mod, static=FALSE)

## Shiny App

# In additon to generating spatial heatmaps and corresponding gene context plots from R,
# spatialHeatmap includes a Shiny App (https://shiny.rstudio.com/) that provides access to the
# same functionalities from an intuitive-to-use web browser interface. Apart from being very
# user-friendly, this App conveniently organizes the results of the entire visualization
# workflow in a single browser window with options to adjust the parameters of the individual
# components interactively. This app is launched by the function "shiny_shm" without any
# parameters. Upon launched, the app automatically displays a pre-formatted example.
shiny_shm()

# The gene expression data and aSVG image files are uploaded to the Shiny App as tabular
# text (e.g. in CSV or TSV format) and SVG file, respectively. To also allow users to upload
# gene expression data stored in "SummarizedExperiment" objects, one can export them from R
# to a tabular file with the "filter_data" function. In this function call, the user sets a
# desired directory path under "dir" (see below). Within this directory the tabular file will
# be written to "customData.txt" in TSV format. The column names in the exported tabular file
# preserve the experimental design information from the "colData" slot by concatenating the
# corresponding sample and condition information separated by double underscores. An example
# of this format is shown in below.

# To interactively view functional descriptions by moving the cursor over network nodes, the
# corresponding annotation column needs to be present in the "rowData" slot and its column
# name assigned to the "ann" argument. In the exported tabular file the extra annotation
# column is appended to the expression matrix.
 se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='organism_part',
con.factor='age', pOA=c(0.01, 5), CV=c(0.2, 100), dir='./'); assay(se.fil.chk)[1:3, 1:3]
```

```
# The Shiny app can be customized by including user-provided default examples and default
# parameters. See the fucntion "custom_shiny" for details.
```

---

adj_mod *Compute Adjacency Matrix and Identify Modules*

Description

The objective is to explore target items (gene, protein, metabolite, *etc*) in context of their neighbors sharing highly similar abundance profiles in a more advanced approach than matrix_hm. This advanced approach is the **WGCNA** algorithm (Langfelder and Horvath 2008; Ravasz et al. 2002). It takes the assay matrix subsetted by submatrix as input and splits the items into network modules, *i.e.* groups of items showing most similar coexpression profiles.

Usage

```
adj_mod(
  data,
  assay.na = NULL,
  type = "signed",
  power = if (type == "distance") 1 else 6,
  arg.adj = list(),
  TOMType = "unsigned",
  arg.tom = list(),
  method = "complete",
  minSize = 15,
  arg.cut = list(),
  dir = NULL
)
```

Arguments

data	The subsetted data matrix returned by the function submatrix, where rows are assayed items and columns are samples/conditions.
assay.na	Applicable when data is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is NULL.
type	The network type, one of "unsigned", "signed", "signed hybrid", "distance". Correlation and distance are transformed as follows: for type="unsigned", adjacency=|cor|^power; for type="signed", adjacency=(0.5 * (1+cor))^power; for type="signed hybrid", if cor>0 adjacency=cor^power, otherwise adjacency=0; and for type="distance", adjacency=(1-(dist/max(dist))^2)^power. Refer to **WGCNA** (Langfelder and Horvath 2008) for more details.
power	A numeric of soft thresholding power for generating the adjacency matrix. The default is 1 for type=='distance' and 6 for other network types.

`arg.adj` A list of additional arguments passed to `adjacency`, *e.g.* `list(corFnc='cor')`. The default is an empty list `list()`.

`TOMType` one of `"none"`, `"unsigned"`, `"signed"`, `"signed Nowick"`, `"unsigned 2"`, `"signed 2"` and `"signed Nowick 2"`. If `"none"`, adjacency will be used for clustering. See `TOMsimilarityFromExpr` for details.

`arg.tom` A list of additional arguments passed to `TOMsimilarity`, *e.g.* `list(verbose=1)`. The default is an empty list `list()`.

`method` the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of `"ward"`, `"single"`, `"complete"`, `"average"`, `"mcquitty"`, `"median"` or `"centroid"`.

`minSize` The expected minimum module size. The default is 15. Refer to **WGCNA** for more details.

`arg.cut` A list of additional arguments passed to `cutreeHybrid`, *e.g.* `list(verbose=2)`. The default is an empty list `list()`.

`dir` The directory to save the results. In this directory, a folder "customComputedData" is created automatically, where the adjacency matrix and module assignments are saved as TSV-format files "adj.txt" and "mod.txt" respectively. This argument should be the same with the `dir` in `submatrix` so that the "sub_matrix.txt" generated in `submatrix` is saved in the same folder. This argument is designed since the computation is intensive for large data matrix (*e.g.* > 10,000 genes). Therefore, to avoid system crash when using the Shiny app (see `shiny_shm`), "adj.txt" and "mod.txt" can be computed in advance and then uploaded to the app. In addition, the saved files can be used repetitively and therefore avoid repetitive computation. The default is NULL and no file is saved. This argument is used only when the "customComputedData" is chosen in the Shiny app. The large matrix issue could be resolved by increasing the subsetting strigency to get smaller matrix in `submatrix` in most cases. Only in rare cases users cannot avoid very large subsetted matrix, this argument is recommended.

Value

A list containing the adjacency matrix and module assignment, which should be provided to `network`. The module assignment is a data frame. The first column is `ds=2` while the second is `ds=3` (see the "Details" section). The numbers in each column are module labels, where "0" means items not assigned to any modules. If `dir` is specified, both adjacency matrix and module assignment are automatically saved in the folder "customComputedData" as "adj.txt" and "mod.txt" respectively, which can be uploaded under "customComputedData" in the Shiny app (see `shiny_shm`).

Details

To identify modules, first a correlation matrix is computed using distance or correlation-based similarity metrics. Second, the obtained matrix is transformed into an adjacency matrix defining the connections among items. Third, the adjacency matrix is used to calculate a topological overlap matrix (TOM) where shared neighborhood information among items is used to preserve robust connections, while removing spurious connections. Fourth, the distance transformed TOM is used for hierarchical clustering. To maximize time performance, the hierarchical clustering is performed with the flashClust package (Langfelder and Horvath 2012). Fifth, network modules are

identified with the dynamicTreeCut package (Langfelder, Zhang, and Steve Horvath 2016). Its ds (deepSplit) argument can be assigned integer values from 0 to 3, where higher values increase the stringency of the module identification process. Since this is a coexpression analysis, variables of sample/condition should be at least 5. Otherwise, identified modules are not reliable. These procedures are wrapped in adj_mod for convenience. The result is a list containing the adjacency matrix and the final module assignments stored in a data.frame. Since the interactive network feature (see network) used in the downstream visualization performs best on smaller modules, only modules obtained with stringent ds settings (here ds=2 and ds=3) are returned.

Author(s)

Jianhai Zhang <zhang.jianhai@hotmail.com; jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 doi:10.1186/1471-2105-9-559
Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
Peter Langfelder, Bin Zhang and with contributions from Steve Horvath (2016). dynamicTreeCut: Methods for Detection of Clusters in Hierarchical Clustering Dendrograms. R package version 1.63-1. https://CRAN.R-project.org/package=dynamicTreeCut
Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. “Gene Expression Across Mammalian Organ Development.” Nature 571 (7766): 505–9
Ravasz, E, A L Somera, D A Mongru, Z N Oltvai, and A L Barabási. 2002. “Hierarchical Organization of Modularity in Metabolic Networks.” Science 297 (5586): 1551–5.

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as a
## "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.
## Set up toy data.

# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
```

```
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be
# made based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in this
# package and accessed below.
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

## As conventions, raw sequencing count data should be normalized, aggregated, and filtered to
## reduce noise.

# Normalize count data.
# The normalizing function "calcNormFactors" (McCarthy et al. 2012) with default settings
# is used.
df.nor.chk <- norm_data(data=df.chk, norm.fun='CNF', log2.trans=TRUE)
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', log2.trans=TRUE)
# Aggregate count data.
# Aggregate "sample__condition" replicates in toy data1.
df.aggr.chk <- aggr_rep(data=df.nor.chk, aggr='mean')
df.aggr.chk[1:3, ]
# Aggregate "sample_condition" replicates in toy data2, where "sample" is "organism_part" and
# "condition" is "age".
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='organism_part', con.factor='age',
aggr='mean')
assay(se.aggr.chk)[1:3, 1:3]
# Filter out genes with low counts and low variance. Genes with counts over 5 (log2 unit) in
# at least 1% samples (pOA), and coefficient of variance (CV) between 0.2 and 100 are retained.
# Filter toy data1.
df.fil.chk <- filter_data(data=df.aggr.chk, pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
# Filter toy data2.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='organism_part', con.factor='age',
```

```
pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)

## Select nearest neighbors for target genes 'ENSGALG00000019846' and 'ENSGALG00000000112',
## which are usually genes visualized in spatial heatmaps.
# Toy data1.
df.sub.mat <- submatrix(data=df.fil.chk, ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), p=0.1)
# Toy data2.
se.sub.mat <- submatrix(data=se.fil.chk, ann='ann', ID=c('ENSGALG00000019846',
'ENSGALG00000000112'), p=0.1)

# In the following, "df.sub.mat" and "se.sub.mat" is used in the same way, so only
# "se.sub.mat" illustrated.

# The subsetted matrix is partially shown below.
se.sub.mat[c('ENSGALG00000019846', 'ENSGALG00000000112'), c(1:2, 63)]
## Adjacency matrix and module identification
# The modules are identified by "adj_mod". It returns a list containing an adjacency matrix and
# a data frame of module assignment.
adj.mod <- adj_mod(data=se.sub.mat)
# The adjacency matrix is a measure of co-expression similarity between genes, where larger
# value denotes higher similarity.
adj.mod[['adj']][1:3, 1:3]
# The modules are identified at two alternative sensitivity levels (ds=2 or 3). From 2 to 3,
# more modules are identified but module sizes are smaller. The two sets of module assignment
# are returned in a data frame. The first column is ds=2 while the second is ds=3. The numbers
# in each column are module labels, where "0" means genes not assigned to any module.
adj.mod[['mod']][1:3, ]
```

`aggr_rep` *Aggregate "Sample__Condition" Replicates in Data Matrix*

Description

This function aggregates "sample__condition" (see `data` argument) replicates by mean or median. The input data is either a `data.frame` or `SummarizedExperiment`.

Usage

```
aggr_rep(data, assay.na = NULL, sam.factor, con.factor, aggr = "mean")
```

Arguments

`data`	An object of `data.frame` or `SummarizedExperiment`. In either case, the columns and rows should be sample/conditions and assayed items (*e.g.* genes, proteins, metabolites) respectively. If `data.frame`, the column names should follow the naming scheme "sample__condition". The "sample" is a general term and stands for cells, tissues, organs, *etc*., where the values are measured. The "condition" is also a general term and refers to experiment treatments applied to "sample" such as drug dosage, temperature, time points, *etc*. If certain samples are not

expected to be colored in "spatial heatmaps" (see spatial_hm), they are not required to follow this naming scheme. In the downstream interactive network (see network), if users want to see node annotation by mousing over a node, a column of row item annotation could be optionally appended to the last column.
In the case of SummarizedExperiment, the assays slot stores the data matrix. Similarly, the rowData slot could optionally store a data frame of row item antation, which is only relevant to the interactive network. The colData slot usually contains a data frame with one column of sample replicates and one column of condition replicates. It is crucial that replicate names of the same sample or condition must be identical. *E.g.* If sampleA has 3 replicates, "sampleA", "sampleA", "sampleA" is expected while "sampleA1", "sampleA2", "sampleA3" is regarded as 3 different samples. If original column names in the assay slot already follow the "sample__condition" scheme, then the colData slot is not required at all.
In the function spatial_hm, this argument can also be a numeric vector. In this vector, every value should be named, and values expected to color the "spatial heatmaps" should follow the naming scheme "sample__condition".
In certain cases, there is no condition associated with data. Then in the naming scheme of data frame or vector, the "__condition" part could be discarded. In SummarizedExperiment, the "condition" column could be discarded in colData slot.
Note, regardless of data class the double underscore is a special string that is reserved for specific purposes in "spatialHeatmap", and thus should be avoided for naming feature/samples and conditions.
In the case of spatial-temporal data, there are three factors: samples, conditions, and time points. The naming scheme is slightly different and includes three options: 1) combine samples and conditions to make the composite factor "sampleCondition", then concatenate the new factor and times with double underscore in between, *i.e.* "sampleCondition__time"; 2) combine samples and times to make the composite factor "sampleTime", then concatenate the new factor and conditions with double underscore in between, *i.e.* "sampleTime__condition"; or 3) combine all three factors to make the composite factor "sampleTimeCondition" without double underscore. See the vignette for more details by running browseVignettes('spatialHeatmap') in R.

assay.na — Applicable when data is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is NULL.

sam.factor — The column name corresponding to samples in the colData of SummarizedExperiment. If the original column names in the assay slot already follows the scheme "sample__condition", then the colData slot is not required and accordingly this argument could be NULL.

con.factor — The column name corresponding to conditions in the colData of SummarizedExperiment. Could be NULL if column names of in the assay slot already follows the scheme "sample__condition", or no condition is associated with the data.

aggr — Aggregate "sample__condition" replicates by "mean" or "median". The default is "mean". If the data argument is a SummarizedExperiment, the "sample__condition" replicates are internally formed by connecting samples and conditions with "__" in colData slot, and are subsequently replace the original col-

umn names in assay slot. If no condition specified to con.factor, the data are aggregated by sample replicates. If "none", no aggregation is applied.

Value

The returned value is the same class with the input data, a data.frame or SummarizedExperiment. In either case, the column names of the data matrix follows the "sample__condition" scheme.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
McCarthy, Davis J., Chen, Yunshun, Smyth, and Gordon K. 2012. "Differential Expression Analysis of Multifactor RNA-Seq Experiments with Respect to Biological Variation." Nucleic Acids Research 40 (10): 4288–97
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9
Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on developments of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as a
## "data frame" input to exemplify data with simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.

## Set up toy data.

# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]
```

```
# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be made
# based on the experiment design, which is accessible through the accession number "E-MTAB-6769"
# in the R package ExpressionAtlas. An example targets file is included in this package and
# accessed below.
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

# Aggregate "sample_condition" replicates in toy data1.
df.aggr.chk <- aggr_rep(data=df.chk, aggr='mean')
df.aggr.chk[1:3, ]

# Aggregate "sample_condition" replicates in toy data2, where "sample" is "organism_part" and
# "condition" is "age".
se.aggr.chk <- aggr_rep(data=se.chk, sam.factor='organism_part', con.factor='age', aggr='mean')
assay(se.aggr.chk)[1:3, 1:3]
```

`aSVG.remote.repo` *A list of URLs of remote aSVG repos*

Description

A list of URLs of remote aSVG repos, *i.e. EBI anatomogram and spatialHeatmap_aSVG_Repository*.

Usage

```
data(aSVG.remote.repo)
```

Format

A list.

Source

EBI anatomogram spatialHeatmap_aSVG_Repository

References

https://github.com/ebi-gene-expression-group/anatomogram/tree/master/src/svg https://github.com/jianhaizhang/spatialHeat

Examples

```
data(aSVG.remote.repo)
aSVG.remote.repo
```

`auc_bar` *Bar plot of AUCs from validating data sets*

Description

After coclustering optimization, visualize AUCs generated by optimal parameter settings on validating data sets.

Usage

```
auc_bar(
  df.auc,
  auc = "auc",
  thr = 0.5,
  bar.width = 0.8,
  spd.sel = NULL,
  title = NULL,
  key.title = NULL,
  x.agl = 80,
  x.vjust = 0.6
)
```

Arguments

`df.auc`	The `data.frame` of AUCs generated by optimal parameter settings on validating data sets, which is returned by `cocluster`.
`auc`	The column name of AUCs in `df.auc`.
`thr`	The AUC threshold, which will be labeled in the bar plot.
`bar.width`	Width of a single bar.
`spd.sel`	A character vector of selected `spd.set`, usually the common `spd.set` displaying desired AUCs across multiple validating data sets.
`title`	The title of composite violin plots.
`key.title`	The title of legend.
`x.agl, x.vjust`	The angle and vertical position of x-axis text.

Value

An object of ggplot.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Examples

```
# Create parameter settings and corresponding aucs.
df.auc <- data.frame(auc=runif(n=5, min=1e-12, max=.99))
ran <- seq(0.2, 0.8, 0.1)
df.spd <- data.frame(sim=sample(ran, 5, replace=FALSE), sim.p=sample(ran, 5, replace=FALSE), dim=sample(seq(5, 40
df.spd$spd.set <- paste0('s', df.spd$sim, 'p', df.spd$sim.p, 'd', df.spd$dim)

df.auc <- cbind(df.spd, df.auc)
# Plot the aucs.
auc_bar(df.auc, auc='auc', thr=0.5)
# See package vignette by calling "browseVignettes('spatialHeatmap')".
```

---

auc_stat *Extract AUC statistics in coclustering optimization*

Description

Extract AUC statistics of target parameters in coclustering optimization.

Usage

```
auc_stat(
  wk.dir,
  tar.par = "norm",
  total.min = 500,
  true.min = 300,
  aucs = round(seq(0.5, 0.9, 0.1), 1)
)
```

Arguments

wk.dir	The working directory of coclustering.
tar.par	The target parameter in optimization, one of norm, filter, dimred, graph, and spd.set corresponding to normalization methods, filtering parameter sets, dimensionality reduction methods, graph-building methods, and sim/sim.p/dim sets respectively.

total.min, true.min, aucs
: Cutoffs to extract AUCs. AUCs over a cutoff and having total bulk tissue assignments above total and true assignments above true.min are extracted. The default is total.min=500, true.min=300, round(seq(0.5, 0.9, 0.1), 1). Each AUC cuttoff in aucs is used independently.

Value

An nested list.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

Examples

```
# To obtain reproducible results, always start a new R session and set a fixed seed for Random Number Generator at th
set.seed(10)

# Example bulk data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Li et al 2016).
blk <- readRDS(system.file("extdata/cocluster/data", "bulk_cocluster.rds", package="spatialHeatmap"))

# Example single cell data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Shahan et al 20
sc10 <- readRDS(system.file("extdata/cocluster/data", "sc10_cocluster.rds", package="spatialHeatmap"))
sc11 <- readRDS(system.file("extdata/cocluster/data", "sc11_cocluster.rds", package="spatialHeatmap"))


# These example data are already pre-processed. To demonstrate the optimization process the pre-processing steps ar

# Inital filtering before normalization.
blk <- filter_data(data=blk, pOA=c(0.2, 15), CV=c(1.5, 100)); dim(blk)

fil.init <- filter_cell(lis=list(sc10=sc10, sc11=sc11), bulk=blk, gen.rm='^ATCG|^ATCG', min.cnt=1, p.in.cell=0.3,

# Normalization.
# sum.factor.
norm.fct <- norm_multi(dat.lis=fil.init, cpm=FALSE)
# sum.factor + CPM.
norm.cpm <- norm_multi(dat.lis=fil.init, cpm=TRUE)

# Secondary filtering.
# Filtering parameter sets.
df.par.fil <- data.frame(p=c(0.1, 0.2, 0.3, 0.4), A=rep(1, 4), cv1=c(0.1, 0.2, 0.3, 0.4), cv2=rep(100, 4), min.cnt=
df.par.fil

# Filtered results are saved in "opt_res".
if (!dir.exists('opt_res')) dir.create('opt_res')
fct.fil.all <- filter_iter(bulk=norm.fct$bulk, cell.lis=list(sc10=norm.fct$sc10, sc11=norm.fct$sc11), df.par.fil

cpm.fil.all <- filter_iter(bulk=norm.cpm$bulk, cell.lis=list(sc10=norm.cpm$sc10, sc11=norm.cpm$sc11), df.par.fil
```

```
# Matching table between bulk tissues and single cells.
match.pa <- system.file("extdata/cocluster/data", "match_arab_root_cocluster.txt", package="spatialHeatmap")
df.match.arab <- read.table(match.pa, header=TRUE, row.names=1, sep='\t')
df.match.arab[1:3, ]

# Optimization.
# Check parallelization guide.
coclus_opt(wk.dir='opt_res', parallel.info=TRUE, dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.

# The first-level parallel computing relies on the slurm scheduler (https://slurm.schedmd.com/documentation.html),
file.copy(system.file("extdata/cocluster", "slurm.tmpl", package="spatialHeatmap"), './slurm.tmpl')

# The first- and second-level parallelizations are set 3 and 2 respectively.
library(BiocParallel)
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# If slurm is not available, parallelize the optimization only at the second-level through 2 workers.
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# The performaces of parameter settings are measured by AUC values in ROC curve. The following demonstrates how to vi

# Extract AUCs and other parameter settings for filtering parameter sets.
df.lis.fil <- auc_stat(wk.dir='opt_res', tar.par='filter', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.fil$df.auc.mean[1:3, ]
```

---

auc_violin *Plot extracted AUCs by parameter settings*

Description

In coclustering optimization, visualize extracted AUCs by each parameter settings in violin plots.

Usage

```
auc_violin(
  df.lis,
  xlab,
  ylab = "AUC",
  nrow = 3,
  title = NULL,
  key.title = NULL,
  lgd.key.size = 0.05
)
```

Arguments

df.lis	The nested list of extracted aucs returned by auc_stat.
xlab, ylab	The x and y axis labels in the violin plots.

nrow	The numbers of rows of all the violin plots.
title	The title of composite violin plots.
key.title	The title of legend.
lgd.key.size	The size of legend keys.

Value

An object of ggplot.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

Examples

```
# To obtain reproducible results, always start a new R session and set a fixed seed for Random Number Generator at th
set.seed(10)

# Example bulk data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Li et al 2016).
blk <- readRDS(system.file("extdata/cocluster/data", "bulk_cocluster.rds", package="spatialHeatmap"))

# Example single cell data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Shahan et al 20
sc10 <- readRDS(system.file("extdata/cocluster/data", "sc10_cocluster.rds", package="spatialHeatmap"))
sc11 <- readRDS(system.file("extdata/cocluster/data", "sc11_cocluster.rds", package="spatialHeatmap"))


# These example data are already pre-processed. To demonstrate the optimization process the pre-processing steps ar

# Inital filtering before normalization.
blk <- filter_data(data=blk, pOA=c(0.2, 15), CV=c(1.5, 100)); dim(blk)

fil.init <- filter_cell(lis=list(sc10=sc10, sc11=sc11), bulk=blk, gen.rm='^ATCG|^ATCG', min.cnt=1, p.in.cell=0.3,

# Normalization.
# sum.factor.
norm.fct <- norm_multi(dat.lis=fil.init, cpm=FALSE)
# sum.factor + CPM.
norm.cpm <- norm_multi(dat.lis=fil.init, cpm=TRUE)

# Secondary filtering.
# Filtering parameter sets.
df.par.fil <- data.frame(p=c(0.1, 0.2, 0.3, 0.4), A=rep(1, 4), cv1=c(0.1, 0.2, 0.3, 0.4), cv2=rep(100, 4), min.cnt=
df.par.fil
```

```
# Filtered results are saved in "opt_res".
if (!dir.exists('opt_res')) dir.create('opt_res')
fct.fil.all <- filter_iter(bulk=norm.fct$bulk, cell.lis=list(sc10=norm.fct$sc10, sc11=norm.fct$sc11), df.par.fil

cpm.fil.all <- filter_iter(bulk=norm.cpm$bulk, cell.lis=list(sc10=norm.cpm$sc10, sc11=norm.cpm$sc11), df.par.fil

# Matching table between bulk tissues and single cells.
match.pa <- system.file("extdata/cocluster/data", "match_arab_root_cocluster.txt", package="spatialHeatmap")
df.match.arab <- read.table(match.pa, header=TRUE, row.names=1, sep='\t')
df.match.arab[1:3, ]

# Optimization.
# Check parallelization guide.
coclus_opt(wk.dir='opt_res', parallel.info=TRUE, dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.

# The first-level parallel computing relies on the slurm scheduler (https://slurm.schedmd.com/documentation.html),
file.copy(system.file("extdata/cocluster", "slurm.tmpl", package="spatialHeatmap"), './slurm.tmpl')

# The first- and second-level parallelizations are set 3 and 2 respectively.
library(BiocParallel)
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# If slurm is not available, parallelize the optimization only at the second-level through 2 workers.
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# The performaces of parameter settings are measured by AUC values in ROC curve. The following demonstrates how to vi

# Extract AUCs and other parameter settings for filtering parameter sets.
df.lis.fil <- auc_stat(wk.dir='opt_res', tar.par='filter', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.fil$df.auc.mean[1:3, ]

# Mean AUCs by each filtering settings and AUC cutoff.
mean_auc_bar(df.lis.fil[[1]], bar.width=0.07, title='Mean AUCs by filtering settings')

# All AUCs by each filtering settings and AUC cutoff.
auc_violin(df.lis=df.lis.fil, xlab='Filtering settings')
```

`cluster_cell` *Cluster single cells or combination of single cells and bulk*

Description

In co-clustering, cluster only single cell data or combination of single cell and bulk data. The cluster assignments are stored in the `label` column of `colData` slot of `SingleCellExperiment`.

Usage

```
cluster_cell(
  data,
```

```
  prop = 0.1,
  min.dim = 5,
  max.dim = 50,
  pca = FALSE,
  graph.meth = "knn",
  dimred = "PCA"
)
```

Arguments

data	The normalized single cell data or nomalized combination of single cell and bulk data at log2 scale in form of SingleCellExperiment, dgCMatrix, matrix, or data.frame.
prop	Numeric scalar specifying the proportion of genes to report as highly variable genes (HVGs). The default is 0.1.
min.dim, max.dim	Integer scalars specifying the minimum (min.dim) and maximum (max.dim) number of (principle components) PCs to retain respectively in denoisePCA. The default is min.dim=5, max.dim=50.
pca	Logical, if TRUE only the data with reduced dimentionality by PCA is returned and no clustering is performed. The default is FALSE and clustering is performed after dimensionality reduction.
graph.meth	Method to build a nearest-neighbor graph, snn (see buildSNNGraph) or knn (default, see buildKNNGraph). The clusters are detected by first creating a nearest neighbor graph using snn or kn then partitioning the graph.
dimred	A string of PCA (default) or UMAP specifying which reduced dimensionality to use in creating a nearest neighbor graph. Internally, before building a nearest neighbor graph the data dimensionalities are reduced by PCA and UMAP respectively.

Value

A list of normalized single cell and bulk data.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Morgan M, Obenchain V, Hester J, Pagès H (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24.0, https://bioconductor.org/packages/SummarizedExperiment. Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x. Lun ATL, McCarthy DJ, Marioni JC (2016). "A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor." F1000Res., 5, 2122. doi: 10.12688/f1000research.9501.2.

McCarthy DJ, Campbell KR, Lun ATL, Willis QF (2017). "Scater: pre-processing, quality control, normalisation and visualisation of single-cell RNA-seq data in R." Bioinformatics, 33, 1179-1186. doi: 10.1093/bioinformatics/btw777. Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. https://igraph.org

Examples

```
library(scran); library(scuttle)
sce <- mockSCE(); sce <- logNormCounts(sce)
# Modelling the variance.
var.stats <- modelGeneVar(sce)
sce <- denoisePCA(sce, technical=var.stats, subset.row=rownames(var.stats))

sce.clus <- cluster_cell(data=sce, prop=0.1, min.dim=5, max.dim=50, graph.meth='snn', dimred='PCA')
# Clusters.
table(colData(sce.clus)$label)


# See details in function "cocluster" by running "?cocluster".
```

`cocluster` *Coclustering bulk and single cell data in a single run*

Description

Cluster single cells, refine single cell clusters, and cocluster bulk and single cell data. Accept multiple parameter settings combinations in form of `data.frame`, and allows for parallelization.

Usage

```
cocluster(
  bulk,
  cell,
  df.match,
  df.para = NULL,
  sc.dim.min = 10,
  max.dim = 50,
  sim = 0.2,
  sim.p = 0.8,
  dim = 12,
  graph.meth = "knn",
  dimred = "PCA",
  sim.meth = "spearman",
  return.all = FALSE,
 multi.core.par = MulticoreParam(workers = 1, stop.on.error = FALSE, log = FALSE),
  verbose = TRUE,
  file = NULL
)
```

Arguments

bulk	Normalized and filtered bulk data at log2 scale in form of matrix, data.frame, SingleCellExperiment, or SummarizedExperiment.
cell	Normalized and filtered bulk data at log2 scale in form of matrix, data.frame, SingleCellExperiment, or SummarizedExperiment.
df.match	The data.frame specifying matching between cells and true bulk.
df.para	A data.frame of paramerter settings in coclustering, where the parameter names are the column names and one row denotes one combination of settings. Missing parameters in the data.frame are replaced by their default settings internally. E.g. In df.para = data.frame(sim = 0.2, sim.p = 0.8, dim = 12, graph.meth = 'knn')), the default settings of sc.dim.min = 10; max.dim=50; dimred='PCA'; sim.meth='spearman' are used internally. If multiple settings combinations are contained in multiple rows respectively, parallelization can be enabled through multi.core.par.
sc.dim.min	Integer scalar specifying the minimum number of (principle components) PCs to retain in denoisePCA when clustering single cells without bulk data. The default is 10.
max.dim	Integer scalar specifying the maximum number of (principle components) PCs to retain in denoisePCA when clustering single cells without bulk data and coclustering single cells and bulk data. The default is 50.
sim	Both are numeric scalars, ranging from 0 to 1. sim is a similarity (Spearman or Pearson correlation coefficient) cutoff between cells and sim.p is a proportion cutoff. In a certain cell cluster, cells having similarity >= sim with other cells in the same cluster at proportion >= sim.p would remain. Otherwise, they are discarded.
sim.p	Both are numeric scalars, ranging from 0 to 1. sim is a similarity (Spearman or Pearson correlation coefficient) cutoff between cells and sim.p is a proportion cutoff. In a certain cell cluster, cells having similarity >= sim with other cells in the same cluster at proportion >= sim.p would remain. Otherwise, they are discarded.
dim	Integer scalar specifying the minimum number of (principle components) PCs to retain in denoisePCA when coclustering single cells and bulk data. The default is 12.
graph.meth	Method to build a nearest-neighbor graph, snn (see buildSNNGraph) or knn (default, see buildKNNGraph). The clusters are detected by first creating a nearest neighbor graph using snn or kn then partitioning the graph.
dimred	A string of PCA (default) or UMAP specifying which reduced dimensionality to use in creating a nearest neighbor graph. Internally, before building a nearest neighbor graph the data dimensionalities are reduced by PCA and UMAP respectively.
sim.meth	Method to calculate similarities between bulk and cells in each cocluster when assigning bulk to cells. spearman (default) or pearson.
return.all	Logical. If TRUE, single cell data after refining cluster, roc object and the data.frame to create the roc during coclustering are returned in a nested list.

If FALSE (default), the df.para table including coclustering statisctics are returned. auc denotes area under the curve. true and total indicate the total true assignments of bulk tissues and total assignments (true and false) respectively. thr, spec, and sens refer to the optimal similarity threshold between bulk and cells in coclusters, specificity, and sensitivity corresponding to thr respectively.

multi.core.par — The parallelization settings. Default is MulticoreParam(workers=1, stop.on.error=FALSE, log=FALSE, logdir=NULL). See MulticoreParam.

verbose — Logical. If TRUE (default), intermediate messages are printed.

file — A file name without extension to save the table of parameter settings and coclustering statisctics if return.all = FALSE. The table is saved by saveRDS with extension .rds. Default is NULL and no file is saved.

Value

A nested list or a table of coclustering results.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Morgan M, Wang J, Obenchain V, Lang M, Thompson R, Turaga N (2021). BiocParallel: Bioconductor facilities for parallel evaluation. R package version 1.28.3, https://github.com/Bioconductor/BiocParallel. R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/. Morgan M, Obenchain V, Hester J, Pagès H (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24.0, https://bioconductor.org/packages/SummarizedExperiment. Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x. Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez and Markus Müller (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics, 12, p. 77. DOI: 10.1186/1471-2105-12-77 <http://www.biomedcentral.com/1471-2105/12/77/> Vacher CM, Lacaille H, O'Reilly JJ, Salzbank J et al. Placental endocrine function shapes cerebellar development and social behavior. Nat Neurosci 2021 Oct;24(10):1392-1401. PMID: 34400844. Ortiz C, Navarro JF, Jurek A, Märtin A et al. Molecular atlas of the adult mouse brain. Sci Adv 2020 Jun;6(26):eabb3446. PMID: 32637622 R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Examples

```
# To obtain reproducible results, always start a new R session and set a fixed seed for Random Number Generator at th
set.seed(10)

# Example bulk data of mouse brain for coclustering (Vacher et al 2021).
blk.mus.pa <- system.file("extdata/shinyApp/example", "bulk_mouse_cocluster.txt", package="spatialHeatmap")
```

```
blk.mus <- as.matrix(read.table(blk.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
blk.mus[1:3, 1:5]

# Example single cell data for coclustering (Ortiz et al 2020).
sc.mus.pa <- system.file("extdata/shinyApp/example", "cell_mouse_cocluster.txt", package="spatialHeatmap")
sc.mus <- as.matrix(read.table(sc.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
sc.mus[1:3, 1:5]

# Initial filtering.
blk.mus <- filter_data(data=blk.mus, sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 5), CV=c(0.2, 100), dir=NULL)
dim(blk.mus)
mus.lis <- filter_cell(lis=list(sc.mus=sc.mus), bulk=blk.mus, gen.rm=NULL, min.cnt=1, p.in.cell=0.5, p.in.gen=0.1

# Normalization: bulk and single cell are combined and normalized, then separated.
mus.lis.nor <- norm_multi(dat.lis=mus.lis, cpm=FALSE)

# Secondary filtering.
library(SingleCellExperiment)
blk.mus.fil <- filter_data(data=logcounts(mus.lis.nor$bulk), sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 0.5), C
dim(blk.mus.fil)

mus.lis.fil <- filter_cell(lis=list(sc.mus=logcounts(mus.lis.nor$sc.mus)), bulk=blk.mus.fil, gen.rm=NULL, min.cn

# The aSVG file of mouse brain.
svg.mus <- system.file("extdata/shinyApp/example", "mus_musculus.brain.svg", package="spatialHeatmap")
# Spatial features.
feature.df <- return_feature(svg.path=svg.mus)

# Matching table indicating true bulk tissues of each cell type and corresponding SVG bulk (spatial feature).
df.match.mus.pa <- system.file("extdata/shinyApp/example", "match_mouse_brain_cocluster.txt", package="spatialHe
df.match <- read.table(df.match.mus.pa, header=TRUE, row.names=1, sep='\t')
df.match

# The SVG bulk tissues are in the aSVG file.
df.match$SVGBulk %in% feature.df$feature

# Cluster single cells.
clus.sc <- cluster_cell(data=mus.lis.fil$sc.mus, min.dim=10, max.dim=50, graph.meth='knn', dimred='PCA')
# Cluster labels are stored in "label" column in "colData".
colData(clus.sc)[1:3, ]

# Refine cell clusters.
cell.refined <- refine_cluster(clus.sc, sim=0.2, sim.p=0.8, sim.meth='spearman')

# Include matching information in "colData".
cell.refined <- true_bulk(cell.refined, df.match)
colData(cell.refined)[1:3, ]

# Cocluster bulk and single cells.
roc.lis <- coclus_roc(bulk=mus.lis.fil$bulk, cell.refined=cell.refined, df.match=df.match, min.dim=12, max.dim=5

# The colustering results. "predictor" is the similarity between bulk and cells within a co-cluster. "index" is the
roc.lis$df.roc[1:3, ]
```

```
# ROC curve created according to "roc.lis$df.roc".
plot(roc.lis$roc.obj, print.auc=TRUE)
# Incorporate "cell.refined" in "roc.lis" for downstream use in co-visualization.
res.lis <- c(list(cell.refined=cell.refined), roc.lis)

# The processes of clustering single cells, refining cell clusters, and coclustering bulk and single cells can be pe
library(BiocParallel)
res.lis <- cocluster(bulk=mus.lis.fil$bulk, cell=mus.lis.fil$sc.mus, df.match=df.match, df.para=NULL, sim=0.2, si
res.lis <- res.lis[[1]]

names(res.lis)

# "cocluster" accepts multiple combinations of parameter settings provided in a data frame, and coclustering on thes
# Multiple combinations of parameter settings. If some parameters are not specified in this table such as "graph.met
df.par <- data.frame(sim=c(0.2, 0.3), sim.p=c(0.8, 0.7), dim=c(12, 13))

# The computation is parallelized on 2 cpu cores by "multi.core.par".
library(BiocParallel)
res.multi <- cocluster(bulk=mus.lis.fil$bulk, cell=mus.lis.fil$sc.mus, df.match=df.match, df.para=df.par, sc.dim

# The results of auto-matching through coclustering can be tailored through "Lasso Select" on the convenience Shiny
df.desired.bulk <- NULL
# Example of desired bulk downloaded from convenience Shiny app.
desired.blk.pa <- system.file("extdata/shinyApp/example", "selected_cells_with_desired_bulk.txt", package="spati
df.desired.bulk <- read.table(desired.blk.pa, header=TRUE, row.names=1, sep='\t')
df.desired.bulk[1:3, ]

# Desired bulk manually definded by x-y coordinates ranges.
plot_dim(res.lis$cell.refined, dim='PCA', color.by='cell', x.break=seq(-10, 10, 2), y.break=seq(-10, 10, 2))

df.desired.bulk <- data.frame(x.min=c(2, 6), x.max=c(4, 10), y.min=c(6, 8), y.max=c(8, 10), desiredSVGBulk=c('cere
df.desired.bulk

# Extract cells with true bulk assignments. If "df.desired.bulk" is provided, the corrresponding assignments are in
sce.lis <- sub_asg(res.lis=res.lis, df.desired.bulk=df.desired.bulk, df.match=df.match, true.only=TRUE)

# Aggregate the extracted cells and the aggregated abundance profiles are used to color matching bulk tissues in the
sce.aggr <- aggr_rep(data=sce.lis$cell.sub, assay.na='logcounts', sam.factor='SVGBulk', con.factor=NULL, aggr='m

# Co-visualize bulk and single cells without abundance profiles.
shm.lis1 <- spatial_hm(svg.path=svg.mus, data=sce.aggr, ID=c('Adcy1'), legend.nrow=4, sce.dimred=sce.lis$cell.re

# Co-visualize bulk and single cells with abundance profiles of gene "Adcy1".
shm.lis2 <- spatial_hm(svg.path=svg.mus, data=sce.aggr, ID=c('Adcy1'), legend.nrow=4, sce.dimred=sce.lis$cell.re
```

`coclus_opt` *Optimizing coclustering process with two levels of parallelizations available.*

Description

Optimizing coclustering process with two levels of parallelizations available.

Usage

```
coclus_opt(
  wk.dir,
  parallel.info = FALSE,
  sc.dim.min = 10,
  max.dim = 50,
  dimred = c("PCA", "UMAP"),
  graph.meth = c("knn", "snn"),
  sim = seq(0.2, 0.8, by = 0.1),
  sim.p = seq(0.2, 0.8, by = 0.1),
  dim = seq(5, 40, by = 1),
  df.match,
  sim.meth = "spearman",
 batch.par = BatchtoolsParam(workers = 1, cluster = "slurm", template = "slurm.tmpl",
    RNGseed = 100, stop.on.error = FALSE, log = TRUE, logdir = file.path(wk.dir,
    "batch_log")),
 multi.core.par = MulticoreParam(workers = 1, RNGseed = NULL, stop.on.error = FALSE,
    log = TRUE, logdir = file.path(wk.dir, "multi_core_log")),
  verbose = TRUE
)
```

Arguments

wk.dir	Working directory. Must be the same with that in filter_iter, since the filtered data will be used automatically.
parallel.info	Logical. If FALSE (default), coclustering optimization is performed. If TRUE, parallelization guide is returned and no optimization. Users are advised to check the guide when editing the slurm template.
sc.dim.min	Integer scalar specifying the minimum number of (principle components) PCs to retain in denoisePCA when clustering single cells without bulk data. The default is 10.
max.dim	Integer scalar specifying the maximum number of (principle components) PCs to retain in denoisePCA when clustering single cells without bulk data and coclustering single cells and bulk data. The default is 50.
dimred	Dimensionality redeuction methods to optimize: c('PCA', 'UMAP').
graph.meth	Graph-building methods in coclustering: c('knn', 'snn') (see buildKNNGraph), buildSNNGraph) respectively). The clusters are detected by first creating a nearest neighbor graph using snn or knn then partitioning the graph.
sim, sim.p	Used when refining cell clusters. Both are numeric scalars, ranging from 0 to 1. sim is a similarity (Spearman or Pearson correlation coefficient) cutoff between cells and sim.p is a proportion cutoff. In a certain cell cluster, cells having similarity >= sim with other cells in the same cluster at proportion >= sim.p

	would remain. Otherwise, they are discarded. The default of both is `seq(0.2, 0.8, by=0.1)` and can be customized.
`dim`	Number of principle components (PCs, equivalent to genes) in combined bulk and single cell data. Used as the minimum number of PCs to retain in `denoisePCA` when coclustering bulk and single cells. The default is `seq(5, 40, by=1)`, and can be customized.
`df.match`	The `data.frame` specifying matching between cells and true bulk.
`sim.meth`	Method to calculate similarities between bulk and cells in each cocluster when assigning bulk to cells. `spearman` (default) or `pearson`.
`batch.par`	The parameters for first-level parallelization. See `BatchtoolsParam`. It works with the "slurm" scheduler, so "slurm" needs to be installed. If `NULL`, the first-level parallelization is skipped.
`multi.core.par`	The parameters for second-level parallelization. See `MulticoreParam`.
`verbose`	Logical. If `TRUE` (default), intermediate messages are printed.

Value

A list of normalized single cell and bulk data.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Li, S., Yamada, M., Han, X., Ohler, U., and Benfey, P. N. (November, 2016) High-Resolution Expression Map of the Arabidopsis Root Reveals Alternative Splicing and lincRNA Regulation. Dev. Cell, 39(4), 508–522. Shahan, R., Hsu, C.-W., Nolan, T. M., Cole, B. J., Taylor, I. W., Vlot, A. H. C., Benfey, P. N., and Ohler, U. (June, 2020) A single cell Arabidopsis root atlas reveals developmental trajectories in wild type and cell identity mutants. Morgan M, Wang J, Obenchain V, Lang M, Thompson R, Turaga N (2021). BiocParallel: Bioconductor facilities for parallel evaluation. R package version 1.28.3, https://github.com/Bioconductor/BiocParallel.

Examples

```
# To obtain reproducible results, always start a new R session and set a fixed seed for Random Number Generator at th
set.seed(10)

# Example bulk data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Li et al 2016).
blk <- readRDS(system.file("extdata/cocluster/data", "bulk_cocluster.rds", package="spatialHeatmap"))

# Example single cell data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Shahan et al 20
sc10 <- readRDS(system.file("extdata/cocluster/data", "sc10_cocluster.rds", package="spatialHeatmap"))
sc11 <- readRDS(system.file("extdata/cocluster/data", "sc11_cocluster.rds", package="spatialHeatmap"))


# These example data are already pre-processed. To demonstrate the optimization process the pre-processing steps ar
```

```
# Inital filtering before normalization.
blk <- filter_data(data=blk, pOA=c(0.2, 15), CV=c(1.5, 100)); dim(blk)

fil.init <- filter_cell(lis=list(sc10=sc10, sc11=sc11), bulk=blk, gen.rm='^ATCG|^ATCG', min.cnt=1, p.in.cell=0.3,

# Normalization.
# sum.factor.
norm.fct <- norm_multi(dat.lis=fil.init, cpm=FALSE)
# sum.factor + CPM.
norm.cpm <- norm_multi(dat.lis=fil.init, cpm=TRUE)

# Secondary filtering.
# Filtering parameter sets.
df.par.fil <- data.frame(p=c(0.1, 0.2, 0.3, 0.4), A=rep(1, 4), cv1=c(0.1, 0.2, 0.3, 0.4), cv2=rep(100, 4), min.cnt=
df.par.fil

# Filtered results are saved in "opt_res".
if (!dir.exists('opt_res')) dir.create('opt_res')
fct.fil.all <- filter_iter(bulk=norm.fct$bulk, cell.lis=list(sc10=norm.fct$sc10, sc11=norm.fct$sc11), df.par.fil

cpm.fil.all <- filter_iter(bulk=norm.cpm$bulk, cell.lis=list(sc10=norm.cpm$sc10, sc11=norm.cpm$sc11), df.par.fil

# Matching table between bulk tissues and single cells.
match.pa <- system.file("extdata/cocluster/data", "match_arab_root_cocluster.txt", package="spatialHeatmap")
df.match.arab <- read.table(match.pa, header=TRUE, row.names=1, sep='\t')
df.match.arab[1:3, ]

# Optimization.
# Check parallelization guide.
coclus_opt(wk.dir='opt_res', parallel.info=TRUE, dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.

# The first-level parallel computing relies on the slurm scheduler (https://slurm.schedmd.com/documentation.html),
file.copy(system.file("extdata/cocluster", "slurm.tmpl", package="spatialHeatmap"), './slurm.tmpl')

# The first- and second-level parallelizations are set 3 and 2 respectively.
library(BiocParallel)
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# If slurm is not available, parallelize the optimization only at the second-level through 2 workers.
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# The performaces of parameter settings are measured by AUC values in ROC curve. The following demonstrates how to vi

# Extract AUCs and other parameter settings for filtering parameter sets.
df.lis.fil <- auc_stat(wk.dir='opt_res', tar.par='filter', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.fil$df.auc.mean[1:3, ]

# Mean AUCs by each filtering settings and AUC cutoff.
mean_auc_bar(df.lis.fil[[1]], bar.width=0.07, title='Mean AUCs by filtering settings')

# All AUCs by each filtering settings and AUC cutoff.
auc_violin(df.lis=df.lis.fil, xlab='Filtering settings')
```

```
# Optimal filtering settings: fil1, fil2, fil3
df.par.fil[c(1, 2, 3), ]

# Extract AUCs and other parameter settings for normalization methods.
df.lis.norm <- auc_stat(wk.dir='opt_res', tar.par='norm', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9, 0
df.lis.norm$df.auc.mean[1:3, ]

# Mean AUCs by each normalization method and AUC cutoff.
mean_auc_bar(df.lis.norm[[1]], bar.width=0.07, title='Mean AUCs by normalization methods')

# All AUCs by each normalization method and AUC cutoff.
auc_violin(df.lis=df.lis.norm, xlab='Normalization methods')

# Optimal normalization method: fct (computeSumFactors).

# Extract AUCs and other parameter settings for graph-building methods.
df.lis.graph <- auc_stat(wk.dir='opt_res', tar.par='graph', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.graph$df.auc.mean[1:3, ]

# Mean AUCs by each graph-building method and AUC cutoff.
mean_auc_bar(df.lis.graph[[1]], bar.width=0.07, title='Mean AUCs by graph-building methods')

# All AUCs by each graph-building method and AUC cutoff.
auc_violin(df.lis=df.lis.graph, xlab='Graph-building methods')

# Optimal graph-building methods: knn (buildKNNGraph).

# Extract AUCs and other parameter settings for dimensionality reduction methods.
df.lis.dimred <- auc_stat(wk.dir='opt_res', tar.par='dimred', total.min=500, true.min=300, aucs=round(seq(0.5, 0.
df.lis.dimred$df.auc.mean[1:3, ]

# Mean AUCs by each dimensionality reduction method and AUC cutoff.
mean_auc_bar(df.lis.dimred[[1]], bar.width=0.07, title='Mean AUCs by dimensionality reduction methods')

# All AUCs by each dimensionality reduction method and AUC cutoff.
auc_violin(df.lis=df.lis.dimred, xlab='Dimensionality reduction')

# Optimal dimensionality reduction method: pca (denoisePCA).

# Extract AUCs and other parameter settings for spd.sets.
df.lis.spd <- auc_stat(wk.dir='opt_res', tar.par='spd.set', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.spd$auc0.5$df.frq[1:3, ]

# All AUCs of top spd.sets ranked by frequency.
spd_auc_violin(df.lis=df.lis.spd, n=5, xlab='spd.sets', x.vjust=0.6)

# Optimal spd.sets.
df.spd.top <- rbind(df.lis.spd[[1]]$df.frq[1:5, ], df.lis.spd[[2]]$df.frq[1:5, ], df.lis.spd[[3]]$df.frq[1:5, ],

df.spd.top$spd.set <- paste0('s', df.spd.top$sim, 'p', df.spd.top$sim.p, 'd', df.spd.top$dim)
df.spd.top <- subset(df.spd.top, !duplicated(spd.set))
df.spd.top[1:3, c('sim', 'sim.p', 'dim', 'spd.set')]
```

coclus_roc *Co-cluster bulk and single cell data Calculate ROC/AUC for the combined bulk and single cell data*

Description

Co-cluster bulk and refined single cell data and assign bulk to single cells. Since the identities of bulk tissues and single cells are labeled, ROC/AUC are calculated to evaluate the co-clustering performance.

Usage

```
coclus_roc(
  bulk,
  cell.refined,
  df.match,
  min.dim = 10,
  max.dim = 50,
  graph.meth = "snn",
  dimred = "PCA",
  sim.meth = "spearman"
)
```

Arguments

bulk	The normalized and filtered bulk data in form of SingleCellExperiment, matrix (log2-scale), or data.frame (log2-scale).
cell.refined	The refined cell data in form of SingleCellExperiment, which is returned by refine_cluster.
df.match	The data.frame specifying matching between cells and true bulk.
min.dim	Integer scalars specifying the minimum (min.dim) and maximum (max.dim) number of (principle components) PCs to retain respectively in denoisePCA. The default is min.dim=5, max.dim=50.
max.dim	Integer scalars specifying the minimum (min.dim) and maximum (max.dim) number of (principle components) PCs to retain respectively in denoisePCA. The default is min.dim=5, max.dim=50.
graph.meth	Method to build a nearest-neighbor graph, snn (see buildSNNGraph) or knn (default, see buildKNNGraph). The clusters are detected by first creating a nearest neighbor graph using snn or kn then partitioning the graph.
dimred	A string of PCA (default) or UMAP specifying which reduced dimensionality to use in creating a nearest neighbor graph. Internally, before building a nearest neighbor graph the data dimensionalities are reduced by PCA and UMAP respectively.
sim.meth	Method to calculate similarities between bulk and cells in each cocluster when assigning bulk to cells. spearman (default) or pearson.

Value

A list of roc object and the data frame to create the roc.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x.

Examples

```
# Example bulk data of mouse brain for coclustering (Vacher et al 2021).
blk.mus.pa <- system.file("extdata/shinyApp/example", "bulk_mouse_cocluster.txt", package="spatialHeatmap")
blk.mus <- as.matrix(read.table(blk.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
blk.mus[1:3, 1:5]

# Example single cell data for coclustering (Ortiz et al 2020).
sc.mus.pa <- system.file("extdata/shinyApp/example", "cell_mouse_cocluster.txt", package="spatialHeatmap")
sc.mus <- as.matrix(read.table(sc.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
sc.mus[1:3, 1:5]

# Initial filtering.
blk.mus <- filter_data(data=blk.mus, sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 5), CV=c(0.2, 100), dir=NULL)
dim(blk.mus)
mus.lis <- filter_cell(lis=list(sc.mus=sc.mus), bulk=blk.mus, gen.rm=NULL, min.cnt=1, p.in.cell=0.5, p.in.gen=0.1

# Normalization: bulk and single cell are combined and normalized, then separated.
mus.lis.nor <- norm_multi(dat.lis=mus.lis, cpm=FALSE)

# Secondary filtering.
library(SingleCellExperiment)
blk.mus.fil <- filter_data(data=logcounts(mus.lis.nor$bulk), sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 0.5), C
dim(blk.mus.fil)

mus.lis.fil <- filter_cell(lis=list(sc.mus=logcounts(mus.lis.nor$sc.mus)), bulk=blk.mus.fil, gen.rm=NULL, min.cn

# The aSVG file of mouse brain.
svg.mus <- system.file("extdata/shinyApp/example", "mus_musculus.brain.svg", package="spatialHeatmap")
# Spatial features.
feature.df <- return_feature(svg.path=svg.mus)

# Matching table indicating true bulk tissues of each cell type and corresponding SVG bulk (spatial feature).
df.match.mus.pa <- system.file("extdata/shinyApp/example", "match_mouse_brain_cocluster.txt", package="spatialHe
df.match <- read.table(df.match.mus.pa, header=TRUE, row.names=1, sep='\t')
df.match
```

```
# The SVG bulk tissues are in the aSVG file.
df.match$SVGBulk %in% feature.df$feature

# Cluster single cells.
clus.sc <- cluster_cell(data=mus.lis.fil$sc.mus, min.dim=10, max.dim=50, graph.meth='knn', dimred='PCA')
# Cluster labels are stored in "label" column in "colData".
colData(clus.sc)[1:3, ]

# Refine cell clusters.
cell.refined <- refine_cluster(clus.sc, sim=0.2, sim.p=0.8, sim.meth='spearman')

# Include matching information in "colData".
cell.refined <- true_bulk(cell.refined, df.match)
colData(cell.refined)[1:3, ]

# Cocluster bulk and single cells.
roc.lis <- coclus_roc(bulk=mus.lis.fil$bulk, cell.refined=cell.refined, df.match=df.match, min.dim=12, max.dim=5
```

`com_factor` *Combine Factors in Targets File*

Description

This is a helper function for data/aSVGs involving three or more factors such as sample, time, condition. It combine factors in targets file to make composite factors.

Usage

```
com_factor(se, target, factors2com, sep = ".", factor.new)
```

Arguments

`se`	A `SummarizedExperiment` object.
`target`	A `data.frame` object of targets file.
`factors2com`	A character vector of column names or a numeric vector of column indeces in the targets file. Entries in these columns are combined.
`sep`	The separator in the combined factors. One of _, and . (default).
`factor.new`	The column name of the new combined factors.

Value

If `se` is provided, a `SummarizedExperiment` object is returned, where the `colData` slot contains the new column of combined factors. Otherwise, a`data.frame` object is returned, where the new column of combined factors is appended.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Narsai, Reena, David Secco, Matthew D Schultz, Joseph R Ecker, Ryan Lister, and James Whelan. 2017. "Dynamic and Rapid Changes in the Transcriptome and Epigenome During Germination and in Developing Rice (Oryza Sativa) Coleoptiles Under Anoxia and Re-Oxygenation." Plant J. 89 (4): 805–24

Examples

```
clp.tar <- system.file('extdata/shinyApp/example/target_coleoptile.txt', package='spatialHeatmap')
target.clp <- read_fr(clp.tar)
target.clp <- com_factor(target=target.clp, factors2com=c('organism_part', 'age'), factor.new='samTime')
```

---

custom_shiny *Create Customized Shiny App of Spaital Heatmap*

Description

This function creates customized Shiny App with user-provided data, aSVG files, and default parameters. Default settings are defined in the "config.yaml" file in the "config" folder of the app, and can be edited directly in a yaml file editor.

Usage

```
custom_shiny(
  ...,
  lis.par = NULL,
  lis.par.tmp = FALSE,
  lis.dld.single = NULL,
  lis.dld.mul = NULL,
  lis.dld.st = NULL,
  example = TRUE,
  app.dir = "."
)
```

Arguments

...	Separate lists of paired data matrix and aSVG files, which are included as default datasets in the Shiny app. Each list must have three elements with name slots of "name", "data", and "svg" respectively. For example, list(name='dataset1', data='./data1.txt', svg='./root_shm.svg'). The "name" element (*e.g.* 'dataset1') is listed under "Step 1: data sets" in the app, while "data" and "svg" are the paths of data matrix and aSVG files. If multiple aSVGs (*e.g.* growth stages) are

included in one list, the respective paths are stored in a vector in the "svg" slot (see example below). After calling this function, the data and aSVGs are copied to the "example" folder in the app. See detailed examples below.

`lis.par` A list of default parameters of the Shiny app. See `lis.par.tmp` . Default is NULL, which means default parameters are adopted.

`lis.par.tmp` Logical, TRUE (default) or FALSE. If TRUE the template of default paramter list is returned, and users can set customized default values then assign this list to `lis.par` . Note, only the existing values in the list can be changed while the hierarchy of the list should be preserved. Otherwise, it cannot be recognized by the internal program.

`lis.dld.single` A list of paired data matrix and single aSVG file, which would be downloadable on the app for testing. The list should have two elements with name slots of "data" and "svg" respectively, which are the paths of the data matrix and aSVG file repectively. After the function call, the specified data and aSVG are copied to the "example" folder in the app. Note the two name slots should not be changed. *E.g.* `list(data='./data_download.txt', svg='./root_download_shm.svg')`.

`lis.dld.mul` A list of paired data matrix and multiple aSVG files, which would be downloadable on the app for testing. The multiple aSVG files could be multiple growth stages of a plant. The list should have two elements with name slots of "data" and "svg" respectively, which are the paths of the data matrix and aSVG files repectively. The data and aSVG should only include the spatial dimension, no temporal dimension. After the function call, the specified data and aSVGs are copied to the "example" folder in the app. Note the two name slots should not be changed. *E.g.* list(data='./data_download.txt', svg=c('./root_young_download_shm.svg', './root_old_download_shm.svg')).

`lis.dld.st` A list of paired data matrix and single aSVG file, which would be downloadable on the app for testing. The list should have two elements with name slots of "data" and "svg" respectively, which are the paths of the data matrix and aSVG file repectively. Compared with `lis.dld.single`, the only difference is the data and aSVG include spatial and temporal dimension. See the example section for details. After the function call, the specified data and aSVG are copied to the "example" folder in the app. Note the two name slots should not be changed. *E.g.* `list(data='./data_download.txt', svg='./root_download_shm.svg')`.

`example` Logical, TRUE or FALSE. If TRUE (default), the default examples in "spatialHeatmap" package are included in the app as well as those provided to `...` by users.

`app.dir` The directory to create the Shiny app. Default is current work directory `..`

Value

If `lis.par.tmp==TRUE`, the template of default paramter list is returned. Otherwise, a customized Shiny app is generated in the path of `app.dir`.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Jeremy Stephens, Kirill Simonov, Yihui Xie, Zhuoer Dong, Hadley Wickham, Jeffrey Horner, reikoch, Will Beasley, Brendan O'Connor and Gregory R. Warnes (2020). yaml: Methods to Convert R Data to YAML and Back. R package version 2.2.1. https://CRAN.R-project.org/package=yaml
Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2017). shiny: Web Application Framework for R. R package version 1.0.3. https://CRAN.R-project.org/package=shiny

Examples

```
# The examples build on pre-packaged examples in spatialHeatmap.

# Get one data path and one aSVG path and assembly them into a list for creating default dataset.
data.path1 <- system.file('extdata/shinyApp/example/expr_arab.txt', package='spatialHeatmap')
svg.path1 <- system.file('extdata/shinyApp/example/arabidopsis.thaliana_shoot_shm.svg',
package='spatialHeatmap')
# The list with name slots of "name", "data", and "svg".
lis.dat1 <- list(name='shoot', data=data.path1, svg=svg.path1)

# Get the paths of spatiotemporal data and aSVG files and assembly them into a list for
# creating default dataset.
data.path.st <- system.file('extdata/shinyApp/example/expr_coleoptile_samTimeCon.txt',
package='spatialHeatmap')
svg.path.st <- system.file('extdata/shinyApp/example/oryza.sativa_coleoptile.ANT_shm.svg',
package='spatialHeatmap')
# The list with name slots of "name", "data", and "svg".
lis.dat.st <- list(name='spatiotemporal', data=data.path.st, svg=svg.path.st)

# Get one data path and two aSVG paths and assembly them into a list for creating default
# dataset, which include two growth stages.
data.path2 <- system.file('extdata/shinyApp/example/random_data_multiple_aSVGs.txt',
package='spatialHeatmap')
svg.path2.1 <- system.file('extdata/shinyApp/example/arabidopsis.thaliana_organ_shm1.svg',
package='spatialHeatmap')
svg.path2.2 <- system.file('extdata/shinyApp/example/arabidopsis.thaliana_organ_shm2.svg',
package='spatialHeatmap')
# The list with name slots of "name", "data", and "svg", where the two aSVG paths are stored
# in a vector in "svg".
lis.dat2 <- list(name='growthStage', data=data.path2, svg=c(svg.path2.1, svg.path2.2))

# Get one data path and one aSVG path and assembly them into a list for creating downloadable
# dataset.
data.path.dld1 <- system.file('extdata/shinyApp/example/expr_arab.txt',
package='spatialHeatmap')
svg.path.dld1 <- system.file('extdata/shinyApp/example/arabidopsis.thaliana_organ_shm.svg',
package='spatialHeatmap')
# The list with name slots of "data", and "svg".
lis.dld.single <- list(name='organ', data=data.path.dld1, svg=svg.path.dld1)
# For demonstration purpose, the same data and aSVGs are used to make the list for creating
# downloadable dataset of two growth stages.
lis.dld.mul <- list(data=data.path2, svg=c(svg.path2.1, svg.path2.2))

# For demonstration purpose, the same spatiotemporal data and aSVG are used to create the
```

```
# downloadable spatiotemporal dataset.
lis.dld.st <- list(data=data.path.st, svg=svg.path.st)

# Retrieve the default parameters.
lis.par <- custom_shiny(lis.par.tmp=TRUE)
# Change default values.
lis.par$shm.img['color', ] <- 'yellow,orange,blue'
# The default dataset to show upon the app is launched.
lis.par$default.dataset <- 'shoot'

if (!dir.exists('~/test_shiny')) dir.create('~/test_shiny')
# Create custom Shiny app by feeding this function these datasets and parameters.
custom_shiny(lis.dat1, lis.dat2, lis.dat.st, lis.par=lis.par, lis.dld.single=lis.dld.single,
lis.dld.mul=lis.dld.mul, lis.dld.st=lis.dld.st, app.dir='~/test_shiny')
# Lauch the app.
shiny::runApp('~/test_shiny/shinyApp')


# The customized Shiny app is able to take database backend as well. Examples are
# demonstrated in the function "write_hdf5".
```

`deg.table` *A table of differentialy-expressed genes (DEGs) detected by different methods*

Description

A table of up- and down-DEGs detected by different methods such as edgeR, limma, DEseq2.

Usage

```
data(deg.table)
```

Format

A table.

Source

ExpressionAtlas E-MTAB-6769

References

Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9

Examples

```
data(deg.table)
deg.table[1:2, ]
```

deg_ovl *Plot Overlap of Spatially-Enriched Genes Across Methods*

Description

In spatial_enrich, the spatially-enriched genes are detected within each method (edgeR, limma, DESeq2, distinct). This function plot the overlap of these detected genes across methods in form of upset plot (Nils, 2019) and overlap matrix.

Usage

```
deg_ovl(
  lis.up.down,
  type = "up",
  plot = "upset",
  order.by = "degree",
  nintersects = 40,
  point.size = 3,
  line.size = 1,
  mb.ratio = c(0.6, 0.4),
  text.scale = 1.5
)
```

Arguments

lis.up.down	The list of all up- and down-regulated genes organized by methods (edgeR, limma, DESeq2, distinct), which comes from the returned value by spatial_enrich.
type	One of up (default) or down, which refers to up- or down-regulated genes.
plot	One of upset (default) or matrix, which corresponds to upset plot or overlap matrix in the output plot.
order.by	How the intersections in the matrix should be ordered by. Options include frequency (entered as "freq"), degree, or both in any order.
nintersects	Number of intersections to plot. If set to NA, all intersections will be plotted.
point.size	Size of points in matrix plot
line.size	Width of lines in matrix plot
mb.ratio	Ratio between matrix plot and main bar plot (Keep in terms of hundredths)
text.scale	Numeric, value to scale the text sizes, applies to all axis labels, tick labels, and numbers above bar plot. Can be a universal scale, or a vector containing individual scales in the following format: c(intersection size title, intersection size tick labels, set size title, set size tick labels, set names, numbers above bars)

Value

An upset plot or matrix plot, which displays overlap of spatially-enriched genes across methods.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9 Nils Gehlenborg (2019). UpSetR: A More Scalable Alternative to Venn and Euler Diagrams for Visualizing Intersecting Sets. R package version 1.4.0. https://CRAN.R-project.org/package=UpSetR

See Also

spatial_enrich

Examples

```
data(lis.deg.up.down)
# Overlap of up-regulated brain-specific genes across methods.
deg_ovl(lis.deg.up.down, type='up', plot='upset')
deg_ovl(lis.deg.up.down, type='up', plot='matrix')
# Overlap of down-regulated brain-specific genes across methods.
deg_ovl(lis.deg.up.down, type='down', plot='upset')
deg_ovl(lis.deg.up.down, type='down', plot='matrix')
# See detailed examples in the function spatial_enrich.
```

---

desired_bulk_shiny *Integrated Shiny App*

Description

Integrated Shiny App

Usage

```
desired_bulk_shiny()
```

Value

A web browser based Shiny app.

Details

No argument is required, this function launches the Shiny app directly.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

https://www.w3schools.com/graphics/svg_intro.asp

https://shiny.rstudio.com/tutorial/

https://shiny.rstudio.com/articles/datatables.html

https://rstudio.github.io/DT/010-style.html

https://plot.ly/r/heatmaps/

https://www.gimp.org/tutorials/

https://inkscape.org/en/doc/tutorials/advanced/tutorial-advanced.en.html

http://www.microugly.com/inkscape-quickguide/

https://cran.r-project.org/web/packages/visNetwork/vignettes/Introduction-to-visNetwork.html

Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2017). shiny: Web Application Framework for R. R package version 1.0.3. https://CRAN.R-project.org/package=shiny

Winston Chang and Barbara Borges Ribeiro (2017). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.6.1. https://CRAN.R-project.org/package=shinydashboard

Paul Murrell (2009). Importing Vector Graphics: The grImport Package for R. Journal of Statistical Software, 30(4), 1-37. URL http://www.jstatsoft.org/v30/i04/

Jeroen Ooms (2017). rsvg: Render SVG Images into PDF, PNG, PostScript, or Bitmap Arrays. R package version 1.1. https://CRAN.R-project.org/package=rsvg

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Yihui Xie (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2. https://CRAN.R-project.org/package=DT

Baptiste Auguie (2016). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.2.1. https://CRAN.R-project.org/package=gridExtra

Andrie de Vries and Brian D. Ripley (2016). ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'. R package version 0.1-20. https://CRAN.R-project.org/package=ggdendro

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 doi:10.1186/1471-2105-9-559

Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/

Simon Urbanek and Jeffrey Horner (2015). Cairo: R graphics device using cairo graphics library for creating high-quality bitmap (PNG, JPEG, TIFF), vector (PDF, SVG, PostScript) and display (X11 and Win32) output. R package version 1.5-9. https://CRAN.R-project.org/package=Cairo

R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/

Duncan Temple Lang and the CRAN Team (2017). XML: Tools for Parsing and Generating XML Within R and S-Plus. R package version 3.98-1.9. https://CRAN.R-project.org/package=XML

Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec and Pedro Despouy (NA). plotly: Create Interactive Web Graphics via 'plotly.js'. https://plot.ly/r, https://cpsievert.github.io/plotly_book/, https://github.com/ropensci/plotly

Matt Dowle and Arun Srinivasan (2017). data.table: Extension of 'data.frame'. R package version 1.10.4. https://CRAN.R-project.org/package=data.table

R. Gentleman, V. Carey, W. Huber and F. Hahne (2017). genefilter: genefilter: methods for filtering genes from high-throughput experiments. R package version 1.58.1.

Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/

Almende B.V., Benoit Thieurmel and Titouan Robert (2017). visNetwork: Network Visualization using 'vis.js' Library. R package version 2.0.1. https://CRAN.R-project.org/package=visNetwork

Examples

```
desired_bulk_shiny()
```

`edit_tar` *Edit Targets Files*

Description

Replace existing entries in a chosen column of a targets file with desired ones.

Usage

```
edit_tar(df.tar, column, old, new, sub.row)
```

Arguments

`df.tar`	The data frame of a targets file.
`column`	The column to edit, either the column name or an integer of the column index.
`old`	A vector of existing entries to replace, where the length must be the same with `new`.
`new`	A vector of desired entries to replace that in `old`, where each entry corresponds to a counterpart in `old` respectively.
`sub.row`	A vector of integers corresponding to target rows for editing, or a vector of TRUE and FALSE corresponding to each row. Default is all rows in the targets file.

Value

A `data.frame`.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Mustroph, Angelika, M Eugenia Zanetti, Charles J H Jang, Hans E Holtan, Peter P Repetti, David W Galbraith, Thomas Girke, and Julia Bailey-Serres. 2009. "Profiling Translatomes of Discrete Cell Populations Resolves Altered Cellular Priorities During Hypoxia in Arabidopsis." Proc Natl Acad Sci U S A 106 (44): 18843–8

Examples

```
sh.tar <- system.file('extdata/shinyApp/example/target_arab.txt', package='spatialHeatmap')
target.sh <- read_fr(sh.tar)
target.sh.new <- edit_tar(df.tar=target.sh, column='conditions', old=c('control', 'hypoxia'),
new=c('C', 'H'), sub.row=c(1:12))
```

---

filter_cell *Filter single cell data separately in a list*

Description

Filter single cell data separately in a list and take overlap genes between all single cell and bulk data. The bulk data are not filtered and are only used to obtain overlap genes.

Usage

```
filter_cell(
  lis,
  bulk = NULL,
  gen.rm = NULL,
  min.cnt = 1,
  p.in.cell = 0.4,
  p.in.gen = 0.2,
  verbose = TRUE
)
```

Arguments

lis	A named list of single cell data in form of data.frame, SingleCellExperiment, or SummarizedExperiment.
bulk	The bulk data in form of data.frame or SummarizedExperiment. They are only used to obtain common genes with all single cell data and not filtered. The default is NULL.

gen.rm	A regular expression of gene identifiers in single cell data to remove before filtering. E.g. mitochondrial, chloroplast and protoplasting-induced genes (^ATCG|^ATCG). The default is NULL.
min.cnt	The min count of gene expression. The default is 1.
p.in.cell	The min proportion of counts above min.cnt in a cell. The default is 0.4.
p.in.gen	The min proportion of counts above min.cnt in a gene. The default is 0.2.
verbose	Logical. If TRUE (default), intermediate messages are printed.

Value

A list of filtered single cell data and bulk data, which have common genes.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24.0. https://bioconductor.org/packages/SummarizedExperiment Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." _Nature Methods_, *17*, 137-145. <URL: https://www.nature.com/articles/s41592-019-0654-x> Douglas Bates and Martin Maechler (2021). Matrix: Sparse and Dense Matrix Classes and Methods. R package version 1.4-0. https://CRAN.R-project.org/package=Matrix Vacher CM, Lacaille H, O'Reilly JJ, Salzbank J et al. Placental endocrine function shapes cerebellar development and social behavior. Nat Neurosci 2021 Oct;24(10):1392-1401. PMID: 34400844. Ortiz C, Navarro JF, Jurek A, Märtin A et al. Molecular atlas of the adult mouse brain. Sci Adv 2020 Jun;6(26):eabb3446. PMID: 32637622

Examples

```
# Example bulk data of mouse brain for coclustering (Vacher et al 2021).
blk.mus.pa <- system.file("extdata/shinyApp/example", "bulk_mouse_cocluster.txt", package="spatialHeatmap")
blk.mus <- as.matrix(read.table(blk.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
blk.mus[1:3, 1:5]
# Example single cell data for coclustering (Ortiz et al 2020).
sc.mus.pa <- system.file("extdata/shinyApp/example", "cell_mouse_cocluster.txt", package="spatialHeatmap")
sc.mus <- as.matrix(read.table(sc.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
sc.mus[1:3, 1:5]
# Initial filtering.
blk.mus <- filter_data(data=blk.mus, sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 5), CV=c(0.2, 100), dir=NULL)
dim(blk.mus)
mus.lis <- filter_cell(lis=list(sc.mus=sc.mus), bulk=blk.mus, gen.rm=NULL, min.cnt=1, p.in.cell=0.5, p.in.gen=0.1
```

`filter_data` *Filter the Data Matrix*

Description

This function is designed to filter the numeric data in class of "data.frame" or "SummarizedExperiment". The filtering builds on two functions `pOverA` and `cv` from the package **genefilter** (Gentleman et al. 2018).

Usage

```
filter_data(
  data,
  assay.na = NULL,
  pOA = c(0, 0),
  CV = c(-Inf, Inf),
  top.CV = 1,
  ann = NULL,
  sam.factor = NULL,
  con.factor = NULL,
  dir = NULL,
  verbose = TRUE
)
```

Arguments

`data`	An object of `data.frame` or `SummarizedExperiment`. In either case, the columns and rows should be sample/conditions and assayed items (*e.g.* genes, proteins, metabolites) respectively. If `data.frame`, the column names should follow the naming scheme "sample__condition". The "sample" is a general term and stands for cells, tissues, organs, *etc*., where the values are measured. The "condition" is also a general term and refers to experiment treatments applied to "sample" such as drug dosage, temperature, time points, *etc*. If certain samples are not expected to be colored in "spatial heatmaps" (see `spatial_hm`), they are not required to follow this naming scheme. In the downstream interactive network (see `network`), if users want to see node annotation by mousing over a node, a column of row item annotation could be optionally appended to the last column. In the case of `SummarizedExperiment`, the `assays` slot stores the data matrix. Similarly, the `rowData` slot could optionally store a data frame of row item antation, which is only relevant to the interactive network. The `colData` slot usually contains a data frame with one column of sample replicates and one column of condition replicates. It is crucial that replicate names of the same sample or condition must be identical. *E.g.* If sampleA has 3 replicates, "sampleA", "sampleA", "sampleA" is expected while "sampleA1", "sampleA2", "sampleA3" is regarded as 3 different samples. If original column names in the `assay` slot already follow the "sample__condition" scheme, then the `colData` slot is not required at all.

In the function spatial_hm, this argument can also be a numeric vector. In this vector, every value should be named, and values expected to color the "spatial heatmaps" should follow the naming scheme "sample__condition".
In certain cases, there is no condition associated with data. Then in the naming scheme of data frame or vector, the "__condition" part could be discarded. In SummarizedExperiment, the "condition" column could be discarded in colData slot.
Note, regardless of data class the double underscore is a special string that is reserved for specific purposes in "spatialHeatmap", and thus should be avoided for naming feature/samples and conditions.
In the case of spatial-temporal data, there are three factors: samples, conditions, and time points. The naming scheme is slightly different and includes three options: 1) combine samples and conditions to make the composite factor "sampleCondition", then concatenate the new factor and times with double underscore in between, *i.e.* "sampleCondition__time"; 2) combine samples and times to make the composite factor "sampleTime", then concatenate the new factor and conditions with double underscore in between, *i.e.* "sampleTime__condition"; or 3) combine all three factors to make the composite factor "sampleTimeCondition" without double underscore. See the vignette for more details by running browseVignettes('spatialHeatmap') in R.

assay.na — Applicable when data is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is NULL.

pOA — It specifies parameters of the filter function pOverA from the package **genefilter** (Gentleman et al. 2018), where genes with expression values larger than "A" in at least the proportion of "P" samples are retained. The input is a vector of two numbers with the first being "P" and the second being "A". The default is c(0, 0), which means no filter is applied.
E.g. c(0.1, 2) means genes with expression values over 2 in at least 10% of all samples are kept.

CV — It specifies parameters of the filter function cv from the package **genefilter** (Gentleman et al. 2018), which filters genes according to the coefficient of variation (CV). The input is a vector of two numbers that specify the CV range. The default is c(-Inf, Inf), which means no filtering is applied.
E.g. c(0.1, 5) means genes with CV between 0.1 and 5 are kept.

top.CV — The proportion of top coefficient of variations (CVs), which ranges from 0 to 1. Only row items with CVs in this proportion are kept. *E.g.* if the proportion is 0.7, only row items with CVs ranked in the top 70% are retained. Default is 1, which means all items are retained. Note this argument takes precedence over CV.

ann — The column name of row item (gene, proteins, *etc*.) annotation in the rowData slot of SummarizedExperiment. The default is NULL. In filter_data, this argument is only relevant if dir is specified, while in network it is only relevant if users want to see annotation when mousing over a node.

sam.factor — The column name corresponding to samples in the colData of SummarizedExperiment. If the original column names in the assay slot already follows the scheme "sam-

ple__condition", then the colData slot is not required and accordingly this argument could be NULL.

con.factor — The column name corresponding to conditions in the colData of SummarizedExperiment. Could be NULL if column names of in the assay slot already follows the scheme "sample__condition", or no condition is associated with the data.

dir — The directory path where the filtered data matrix is saved as a TSV-format file "customData.txt", which is ready to upload to the Shiny app launched by shiny_shm. In the "customData.txt", the rows are assayed items and column names are in the syntax "sample__condition". If gene annotation is provided to ann, it is appended to "customData.txt". The default is NULL and no file is saved. This argument is used only when the data is stored in SummarizedExperiment and need to be uploaded to the "customData" in the Shiny app.

verbose — TRUE or FALSE. If TRUE (default), the summary of statistics is printed.

Value

The returned value is the same class with the input data, a data.frame or SummarizedExperiment. In either case, the column names of the data matrix follows the "sample__condition" scheme. If dir is specified, the filtered data matrix is saved in a TSV-format file "customData.txt".

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Gentleman, R, V Carey, W Huber, and F Hahne. 2018. "Genefilter: Methods for Filtering Genes from High-Throughput Experiments." http://bioconductor.uib.no/2.7/bioc/html/genefilter.html

Matt Dowle and Arun Srinivasan (2017). data.table: Extension of 'data.frame'. R package version 1.10.4. https://CRAN.R-project.org/package=data.table

Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1

R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/

Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas

Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8

Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9

Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as
## a "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.

## Set up toy data.

# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the naming scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be
# made based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in
# this package and accessed below.
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

# Filter out genes with low counts and low variance. Genes with counts over 5 (log2 unit) in
# at least 1% samples (pOA), and coefficient of variance (CV) between 0.2 and 100 are retained.
# Filter toy data1.
df.fil.chk <- filter_data(data=df.chk, pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
# Filter toy data2.
se.fil.chk <- filter_data(data=se.chk, sam.factor='organism_part', con.factor='age',
pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
```

`filter_iter` *Iteratively filter bulk and single data according to parameters in a data frame.*

Description

In secondary filtering of coclustering optimization, iteratively filter normalized bulk and single data (in a list) according to parameter combinations in a data frame.

Usage

```
filter_iter(
  bulk,
  cell.lis,
  df.par.fil,
  gen.rm = NULL,
  norm.meth,
  wk.dir,
  verbose = TRUE
)
```

Arguments

`bulk`	Normalized bulk data at log2-scale returned by `norm_multi`.
`cell.lis`	Normalized single cell data at log2-scale in a named list returned by `norm_multi`.
`df.par.fil`	A `data.frame` of filtering parameter settings that are passed to `filter_data` and `filter_cell` respectively. E.g. `df.par.fil <- data.frame(p=c(0.2, 0.3), A=rep(1, 4), cv1=c(0.2, 0.3), cv2=rep(100, 4), min.cnt=rep(1, 4), p.in.cell=c(0.25, 0.3), p.in.gen=c(0.05, 0.1))`.
`gen.rm`	A regular expression of gene identifiers in single cell data to remove before filtering. E.g. mitochondrial, chloroplast and protoplasting-induced genes (`^ATCG|^ATCG`). The default is `NULL`.
`norm.meth`	Methods used to normalize bulk and single cell data. One of `fct` and `cpm`, standing for `computeSumFactors` only and further normalized by counts per million (cpm) respectively. No actual normalization is performed, only used in file names when saving filtered results.
`wk.dir`	The work directory where filtered data are saved in ".rds" files by `saveRDS`.
`verbose`	Logical. If `TRUE` (default), intermediate messages are printed.

Value

Filtered data are save in `wk.dir`.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

Examples

```
# Example bulk data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Li et al 2016).
blk <- readRDS(system.file("extdata/cocluster/data", "bulk_cocluster.rds", package="spatialHeatmap"))

# Example single cell data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Shahan et al 20
sc10 <- readRDS(system.file("extdata/cocluster/data", "sc10_cocluster.rds", package="spatialHeatmap"))
sc11 <- readRDS(system.file("extdata/cocluster/data", "sc11_cocluster.rds", package="spatialHeatmap"))


# These example data are already pre-processed. To demonstrate the optimization process the pre-processing steps ar

# Inital filtering before normalization.
blk <- filter_data(data=blk, pOA=c(0.2, 15), CV=c(1.5, 100)); dim(blk)

fil.init <- filter_cell(lis=list(sc10=sc10, sc11=sc11), bulk=blk, gen.rm='^ATCG|^ATCG', min.cnt=1, p.in.cell=0.3,

# Normalization.
# sum.factor.
norm.fct <- norm_multi(dat.lis=fil.init, cpm=FALSE)
# sum.factor + CPM.
norm.cpm <- norm_multi(dat.lis=fil.init, cpm=TRUE)

# Secondary filtering.
# Filtering parameter sets.
df.par.fil <- data.frame(p=c(0.1, 0.2, 0.3, 0.4), A=rep(1, 4), cv1=c(0.1, 0.2, 0.3, 0.4), cv2=rep(100, 4), min.cnt=
df.par.fil

# Filtered results are saved in "opt_res".
if (!dir.exists('opt_res')) dir.create('opt_res')
fct.fil.all <- filter_iter(bulk=norm.fct$bulk, cell.lis=list(sc10=norm.fct$sc10, sc11=norm.fct$sc11), df.par.fil

cpm.fil.all <- filter_iter(bulk=norm.cpm$bulk, cell.lis=list(sc10=norm.cpm$sc10, sc11=norm.cpm$sc11), df.par.fil
```

---

lis.deg.up.down *A nested list of differentialy-expressed genes (DEGs) detected by different methods*

Description

A nested list of up- and down-DEGs detected by different methods such as edgeR, limma, DEseq2.

Usage

```
data(lis.deg.up.down)
```

Format

A nested list.

Source

ExpressionAtlas E-MTAB-6769

References

Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9

Examples

```
data(lis.deg.up.down)
lis.deg.up.down$up.lis$edgeR.up[1:5]
```

`matrix_hm` *Matrix Heatmap*

Description

This function visualizes the input assayed items (gene, protein, metabolite, *etc*) in context of their nearest neighbors, which are subsetted by `submatrix`. The visualization is in form of static or interactive matrix heatmap, where rows and columns are sorted by hierarchical clustering dendrograms and the row of target items are tagged by two lines. In the interactive heatmap, users can zoom in and out by drawing a rectangle and by double clicking the image, respectively.

Usage

```
matrix_hm(
  ID,
  data,
  assay.na = NULL,
  scale = "no",
  col = c("yellow", "orange", "red"),
  main = NULL,
  title.size = 10,
  cexCol = 1,
  cexRow = 1,
  angleCol = 45,
  angleRow = 45,
  sep.color = "black",
  sep.width = 0.02,
  static = TRUE,
  margin = c(10, 10),
  arg.lis1 = list(),
  arg.lis2 = list()
)
```

Arguments

ID	A vector of target item identifiers in the data.
data	The subsetted data matrix returned by the function submatrix, where rows are assayed items and columns are samples/conditions.
assay.na	Applicable when data is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is NULL.
scale	One of "row", "column", or "no", corresponding to scale the heatmap by row, column, or no scale respectively. Default is "no".
col	A character vector of color ingredients for constructing the color scale. The default is c('yellow', 'orange', 'red').
main	The title of the matrix heatmap.
title.size	A numeric value of the title size.
cexCol	A numeric value of column name size. Default is 1.
cexRow	A numeric value of row name size. Default is 1.
angleCol	The angle of column names. The default is 45.
angleRow	The angle of row names. The default is 45.
sep.color	The color of the two lines labeling the row of ID. The default is "black".
sep.width	The width of two lines labeling the row of ID. The default is 0.02.
static	Logical, TRUE returns the static visualization and FALSE returns the interactive.
margin	A vector of two numbers, specifying bottom and right margins respectively. The default is c(10, 10).
arg.lis1	A list of additional arguments passed to the heatmap.2 function from "gplots" package. *E.g.* list(xlab='sample', ylab='gene'). The default is an empty list.
arg.lis2	A list of additional arguments passed to the ggplot function from "ggplot2" package. The default is an empty list.

Value

A static image or an interactive instance lauched on the web browser.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
Andrie de Vries and Brian D. Ripley (2016). ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'. R package version 0.1-20. https://CRAN.R-project.org/package=ggdendro
H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Carson Sievert (2018) plotly for R. https://plotly-book.cpsievert.me
Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 doi:10.1186/1471-2105-9-559
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
Gregory R. Warnes, Ben Bolker, Lodewijk Bonebakker, Robert Gentleman, Wolfgang Huber Andy Liaw, Thomas Lumley, Martin Maechler, Arni Magnusson, Steffen Moeller, Marc Schwartz and Bill Venables (2019). gplots: Various R Programming Tools for Plotting Data. R package version 3.0.1.1. https://CRAN.R-project.org/package=gplots
Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. “Gene Expression Across Mammalian Organ Development.” Nature 571 (7766): 505–9

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as
## a "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.
## Set up toy data.

# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be
# made based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in
# this package and accessed below.
```

```
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

## As conventions, raw sequencing count data should be normalized, aggregated, and filtered
## to reduce noise.

# Normalize count data.
# The normalizing function "calcNormFactors" (McCarthy et al. 2012) with default settings
# is used.
df.nor.chk <- norm_data(data=df.chk, norm.fun='CNF', log2.trans=TRUE)
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', log2.trans=TRUE)
# Aggregate count data.
# Aggregate "sample__condition" replicates in toy data1.
df.aggr.chk <- aggr_rep(data=df.nor.chk, aggr='mean')
df.aggr.chk[1:3, ]
# Aggregate "sample_condition" replicates in toy data2, where "sample" is "organism_part"
# and "condition" is "age".
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='organism_part', con.factor='age',
aggr='mean')
assay(se.aggr.chk)[1:3, 1:3]
# Filter out genes with low counts and low variance. Genes with counts over 5 (log2 unit) in
# at least 1% samples (pOA), and coefficient of variance (CV) between 0.2 and 100 are retained.
# Filter toy data1.
df.fil.chk <- filter_data(data=df.aggr.chk, pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
# Filter toy data2.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='organism_part', con.factor='age',
pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)

## Select nearest neighbors for target genes 'ENSGALG00000019846' and 'ENSGALG00000000112',
## which are usually genes visualized in spatial heatmaps.
# Toy data1.
df.sub.mat <- submatrix(data=df.fil.chk, ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), p=0.1)
# Toy data2.
se.sub.mat <- submatrix(data=se.fil.chk, ann='ann', ID=c('ENSGALG00000019846',
'ENSGALG00000000112'), p=0.1)

# In the following, "df.sub.mat" and "se.sub.mat" is used in the same way, so only
# "se.sub.mat" illustrated.

# The subsetted matrix is partially shown below.
se.sub.mat[c('ENSGALG00000019846', 'ENSGALG00000000112'), c(1:2, 63)]

## Matrix heatmap.
# Static matrix heatmap.
matrix_hm(ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), data=se.sub.mat, angleCol=80,
```

```
angleRow=35, cexRow=0.8, cexCol=0.8, margin=c(8, 10), static=TRUE,
arg.lis1=list(offsetRow=0.01, offsetCol=0.01))
# Interactive matrix heatmap.
 matrix_hm(ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), data=se.sub.mat,
angleCol=80, angleRow=35, cexRow=0.8, cexCol=0.8, margin=c(8, 10), static=FALSE,
arg.lis1=list(offsetRow=0.01, offsetCol=0.01))
# In case the interactive heatmap is not automatically opened, run the following code snippet.
# It saves the heatmap as an HTML file according to the value assigned to the "file" argument.

mhm <- matrix_hm(ID=c('ENSGALG00000019846', 'ENSGALG00000000112'), data=se.sub.mat,
angleCol=80, angleRow=35, cexRow=0.8, cexCol=0.8, margin=c(8, 10), static=FALSE,
arg.lis1=list(offsetRow=0.01, offsetCol=0.01))
htmlwidgets::saveWidget(widget=mhm, file='mhm.html', selfcontained=FALSE)
browseURL('mhm.html')
```

`mean_auc_bar` *Plot mean of extracted AUCs by parameter settings*

Description

In coclustering optimization, visualize means of extracted AUCs of each parameter settings by each AUC cutoff in bar plots.

Usage

```
mean_auc_bar(
  df.auc,
  parameter = "parameter",
  auc.thr = "auc.thr",
  mean = "mean",
  bar.width = 0.8,
  title = NULL,
  key.title = NULL,
  lgd.key.size = 0.05
)
```

Arguments

`df.auc`	The `data.frame` of mean AUCs in the nested list of extracted aucs returned by `auc_stat`.
`parameter`	The coloumn name of parameters in `df.auc`.
`auc.thr`	The coloumn name of AUC cutoffs in `df.auc`.
`mean`	The coloumn name of mean AUCs in `df.auc`.
`bar.width`	Width of a single bar.
`title`	The title of composite violin plots.
`key.title`	The title of legend.
`lgd.key.size`	The size of legend keys.

Value

An object of ggplot.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Examples

```
# To obtain reproducible results, always start a new R session and set a fixed seed for Random Number Generator at th
set.seed(10)

# Example bulk data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Li et al 2016).
blk <- readRDS(system.file("extdata/cocluster/data", "bulk_cocluster.rds", package="spatialHeatmap"))

# Example single cell data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Shahan et al 20
sc10 <- readRDS(system.file("extdata/cocluster/data", "sc10_cocluster.rds", package="spatialHeatmap"))
sc11 <- readRDS(system.file("extdata/cocluster/data", "sc11_cocluster.rds", package="spatialHeatmap"))


# These example data are already pre-processed. To demonstrate the optimization process the pre-processing steps ar

# Inital filtering before normalization.
blk <- filter_data(data=blk, pOA=c(0.2, 15), CV=c(1.5, 100)); dim(blk)

fil.init <- filter_cell(lis=list(sc10=sc10, sc11=sc11), bulk=blk, gen.rm='^ATCG|^ATCG', min.cnt=1, p.in.cell=0.3,

# Normalization.
# sum.factor.
norm.fct <- norm_multi(dat.lis=fil.init, cpm=FALSE)
# sum.factor + CPM.
norm.cpm <- norm_multi(dat.lis=fil.init, cpm=TRUE)

# Secondary filtering.
# Filtering parameter sets.
df.par.fil <- data.frame(p=c(0.1, 0.2, 0.3, 0.4), A=rep(1, 4), cv1=c(0.1, 0.2, 0.3, 0.4), cv2=rep(100, 4), min.cnt=
df.par.fil

# Filtered results are saved in "opt_res".
if (!dir.exists('opt_res')) dir.create('opt_res')
fct.fil.all <- filter_iter(bulk=norm.fct$bulk, cell.lis=list(sc10=norm.fct$sc10, sc11=norm.fct$sc11), df.par.fil

cpm.fil.all <- filter_iter(bulk=norm.cpm$bulk, cell.lis=list(sc10=norm.cpm$sc10, sc11=norm.cpm$sc11), df.par.fil

# Matching table between bulk tissues and single cells.
match.pa <- system.file("extdata/cocluster/data", "match_arab_root_cocluster.txt", package="spatialHeatmap")
df.match.arab <- read.table(match.pa, header=TRUE, row.names=1, sep='\t')
```

```
df.match.arab[1:3, ]

# Optimization.
# Check parallelization guide.
coclus_opt(wk.dir='opt_res', parallel.info=TRUE, dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.

# The first-level parallel computing relies on the slurm scheduler (https://slurm.schedmd.com/documentation.html),
file.copy(system.file("extdata/cocluster", "slurm.tmpl", package="spatialHeatmap"), './slurm.tmpl')

# The first- and second-level parallelizations are set 3 and 2 respectively.
library(BiocParallel)
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# If slurm is not available, parallelize the optimization only at the second-level through 2 workers.
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# The performaces of parameter settings are measured by AUC values in ROC curve. The following demonstrates how to vi

# Extract AUCs and other parameter settings for filtering parameter sets.
df.lis.fil <- auc_stat(wk.dir='opt_res', tar.par='filter', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.fil$df.auc.mean[1:3, ]

# Mean AUCs by each filtering settings and AUC cutoff.
mean_auc_bar(df.lis.fil[[1]], bar.width=0.07, title='Mean AUCs by filtering settings')
```

---

network *Visualize a Target Assayed Item in a Network Graph*

Description

This function exhibits a target assayed item (gene, protein, metabolite, *etc*) in the context of corresponding network module as static or interactive network graphs. See function adj_mod for module identification. In the network graph, nodes are items and edges are adjacencies (coexpression similarities) between items. The thicker edge denotes higher adjacency between nodes while larger node indicates higher connectivity (sum of a node's adjacencies with all its direct neighbours).
In the interactive mode, there is an interactive color bar to denote node connectivity. The color ingredients can only be separated by comma, semicolon, single space, dot, hypen, or, underscore. *E.g.* "yellow,orange,red", which means node connectivity increases from yellow to red. If too many edges (*e.g.*: > 500) are displayed, the app may get crashed, depending on the computer RAM. So the "Adjacency threshold" option sets a threthold to filter out weak edges. Meanwhile, the "Maximun edges" limits the total of shown edges. In case a very low adjacency threshold is choosen and introduces too many edges that exceed the Maximun edges, the app will internally increase the adjacency threshold until the edge total is within the Maximun edges, which is a protection against too many edges. The adjacency threshold of 1 produces no edges, in this case the app wil internally decrease this threshold until the number of edges reaches the Maximun edges. If adjacency threshold of 0.998 is selected and no edge is left, this app will also internally update the edges to 1 or 2. To maintain acceptable performance, users are advised to choose a stringent threshold (*e.g.* 0.9) initially, then decrease the value gradually. The interactive feature allows users to zoom in and

out, or drag a node around. All the node IDs in the network module are listed in "Select by id" in decreasing order according to node connectivity. The input item ID is appended "_target" as a label. By clicking an ID in this list, users can identify the corresponding node in the network. If the input data has item annotations, then the annotation can be seen by hovering the cursor over a node.

Usage

```
network(
  ID,
  data,
  assay.na = NULL,
  adj.mod,
  ds = "3",
  adj.min = 0,
  con.min = 0,
  node.col = c("turquoise", "violet"),
  edge.col = c("yellow", "blue"),
  vertex.label.cex = 1,
  vertex.cex = 3,
  edge.cex = 10,
  layout = "circle",
  main = NULL,
  static = TRUE,
  ...
)
```

Arguments

ID	A target item identifier.
data	The subsetted data matrix returned by the function submatrix, where rows are assayed items and columns are samples/conditions.
assay.na	Applicable when data is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is NULL.
adj.mod	The two-component list returned by adj_mod with the adjacency matrix and module assignment respectively.
ds	One of "2" or "3", the module splitting sensitivity level. The former indicates larger but less modules while the latter denotes smaller but more modules. Default is "3". See function adj_mod for details.
adj.min	Minimum adjacency between nodes, edges with adjacency below which will be removed. Default is 0. Applicable to static network.
con.min	Minimun connectivity of a node, nodes with connectivity below which will be removed. Default is 0. Applicable to static network.
node.col	A vector of color ingredients for constructing node color scale in the static image. The default is c("turquoise", "violet"), where node connectivity increases from "turquoise" to "violet".

edge.col	A vector of color ingredients for constructing edge color scale in the static image. The default is c("yellow", "blue"), where edge adjacency increases from "yellow" to "blue".
vertex.label.cex	The size of node label in the static and interactive networks. The default is 1.
vertex.cex	The size of node in the static image. The default is 3.
edge.cex	The size of edge in the static image. The default is 10.
layout	The layout of the network in static image, either "circle" or "fr". The "fr" stands for force-directed layout algorithm by Fruchterman and Reingold. The default is "circle".
main	The title in the static image. Default is NULL.
static	Logical, TRUE returns a static network while FALSE returns an interactive network.
...	Other arguments passed to the generic function plot.default, *e.g.*: asp=1.

Value

A static or interactive network graph.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. http://igraph.org
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2018). shiny: Web Application Framework for R. R package version 1.1.0. https://CRAN.R-project.org/package=shiny
Winston Chang and Barbara Borges Ribeiro (2018). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.7.1. https://CRAN.R-project.org/package=shinydashboard
Almende B.V., Benoit Thieurmel and Titouan Robert (2018). visNetwork: Network Visualization using 'vis.js' Library. R package version 2.0.4. https://CRAN.R-project.org/package=visNetwork
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as
## a "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.

## Set up toy data.

# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be made
# based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in this
# package and accessed below.
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

## As conventions, raw sequencing count data should be normalized, aggregated, and filtered to
## reduce noise.

# Normalize count data.
# The normalizing function "calcNormFactors" (McCarthy et al. 2012) with default settings
# is used.
df.nor.chk <- norm_data(data=df.chk, norm.fun='CNF', log2.trans=TRUE)
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', log2.trans=TRUE)
# Aggregate count data.
```

```
# Aggregate "sample__condition" replicates in toy data1.
df.aggr.chk <- aggr_rep(data=df.nor.chk, aggr='mean')
df.aggr.chk[1:3, ]
# Aggregate "sample_condition" replicates in toy data2, where "sample" is "organism_part" and
# "condition" is "age".
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='organism_part', con.factor='age',
aggr='mean')
assay(se.aggr.chk)[1:3, 1:3]
# Filter out genes with low counts and low variance. Genes with counts over 5 (log2 unit) in
# at least 1% samples (pOA), and coefficient of variance (CV) between 0.2 and 100 are retained.
# Filter toy data1.
df.fil.chk <- filter_data(data=df.aggr.chk, pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
# Filter toy data2.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='organism_part', con.factor='age',
pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)

## Select nearest neighbors for target genes 'ENSGALG00000019846' and 'ENSGALG00000000112',
## which are usually genes visualized in spatial heatmaps.
# Toy data1.
df.sub.mat <- submatrix(data=df.fil.chk, ID=c('ENSGALG00000019846', 'ENSGALG00000000112'),
p=0.1)
# Toy data2.
se.sub.mat <- submatrix(data=se.fil.chk, ann='ann', ID=c('ENSGALG00000019846',
'ENSGALG00000000112'), p=0.1)

# In the following, "df.sub.mat" and "se.sub.mat" is used in the same way, so only
# "se.sub.mat" illustrated.

# The subsetted matrix is partially shown below.
se.sub.mat[c('ENSGALG00000019846', 'ENSGALG00000000112'), c(1:2, 63)]
## Adjacency matrix and module identification
# The modules are identified by "adj_mod". It returns a list containing an adjacency matrix
# and a data frame of module assignment.
adj.mod <- adj_mod(data=se.sub.mat)
# The adjacency matrix is a measure of co-expression similarity between genes, where larger
# value denotes higher similarity.
adj.mod[['adj']][1:3, 1:3]
# The modules are identified at two alternative sensitivity levels (ds=2 or 3). From 2 to 3,
# more modules are identified but module sizes are smaller. The two sets of module assignment
# are returned in a data frame. The first column is ds=2 while the second is ds=3. The numbers
# in each column are module labels, where "0" means genes not assigned to any module.
adj.mod[['mod']][1:3, ]
# Static network. In the graph, nodes are genes and edges are adjacencies between genes.
# The thicker edge denotes higher adjacency (co-expression similarity) while larger node
# indicates higher gene connectivity (sum of a gene's adjacency with all its direct neighbors).
# The target gene is labeled by "_target".
network(ID="ENSGALG00000019846", data=se.sub.mat, adj.mod=adj.mod, adj.min=0.7,
vertex.label.cex=1.5, vertex.cex=4, static=TRUE)
# Interactive network. The target gene ID is appended "_target".
 network(ID="ENSGALG00000019846", data=se.sub.mat, adj.mod=adj.mod, static=FALSE)
```

norm_data *Normalize Sequencing Count Matrix*

Description

This function normalizes sequencing count data. It accepts the count matrix and sample metadata (optional) in form of `SummarizedExperiment` or `data.frame`. In either class, the columns and rows of the count matix should be sample/conditions and genes respectively.

Usage

```
norm_data(
  data,
  assay.na = NULL,
  norm.fun = "CNF",
  parameter.list = NULL,
  log2.trans = TRUE,
  data.trans
)
```

Arguments

`data`	An object of `data.frame` or `SummarizedExperiment`. In either case, the columns and rows should be sample/conditions and assayed items (*e.g.* genes, proteins, metabolites) respectively. If `data.frame`, the column names should follow the naming scheme "sample__condition". The "sample" is a general term and stands for cells, tissues, organs, *etc*., where the values are measured. The "condition" is also a general term and refers to experiment treatments applied to "sample" such as drug dosage, temperature, time points, *etc*. If certain samples are not expected to be colored in "spatial heatmaps" (see `spatial_hm`), they are not required to follow this naming scheme. In the downstream interactive network (see `network`), if users want to see node annotation by mousing over a node, a column of row item annotation could be optionally appended to the last column. In the case of `SummarizedExperiment`, the `assays` slot stores the data matrix. Similarly, the `rowData` slot could optionally store a data frame of row item antation, which is only relevant to the interactive network. The `colData` slot usually contains a data frame with one column of sample replicates and one column of condition replicates. It is crucial that replicate names of the same sample or condition must be identical. *E.g.* If sampleA has 3 replicates, "sampleA", "sampleA", "sampleA" is expected while "sampleA1", "sampleA2", "sampleA3" is regarded as 3 different samples. If original column names in the `assay` slot already follow the "sample__condition" scheme, then the `colData` slot is not required at all. In the function `spatial_hm`, this argument can also be a numeric vector. In this vector, every value should be named, and values expected to color the "spatial heatmaps" should follow the naming scheme "sample__condition".

In certain cases, there is no condition associated with data. Then in the naming scheme of `data frame` or `vector`, the "__condition" part could be discarded. In `SummarizedExperiment`, the "condition" column could be discarded in `colData` slot.
Note, regardless of data class the double underscore is a special string that is reserved for specific purposes in "spatialHeatmap", and thus should be avoided for naming feature/samples and conditions.
In the case of spatial-temporal data, there are three factors: samples, conditions, and time points. The naming scheme is slightly different and includes three options: 1) combine samples and conditions to make the composite factor "sampleCondition", then concatenate the new factor and times with double underscore in between, *i.e.* "sampleCondition__time"; 2) combine samples and times to make the composite factor "sampleTime", then concatenate the new factor and conditions with double underscore in between, *i.e.* "sampleTime__condition"; or 3) combine all three factors to make the composite factor "sampleTimeCondition" without double underscore. See the vignette for more details by running `browseVignettes('spatialHeatmap')` in R.

`assay.na` Applicable when `data` is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is `NULL`.

`norm.fun` One of the normalizing functions: "CNF", "ESF", "VST", "rlog", "none". Specifically, "CNF" stands for `calcNormFactors` from edgeR (McCarthy et al. 2012), and "EST", "VST", and "rlog" is equivalent to `estimateSizeFactors`, `varianceStabilizingTransformation`, and `rlog` from DESeq2 respectively (Love, Huber, and Anders 2014). If "none", no normalization is applied. The default is "CNF" and the output data is processed by `cpm` (Counts Per Million). The parameters of each normalization function are provided through `parameter.list`.

`parameter.list` A list of parameters for each normalizing function assigned in `norm.fun`. The default is NULL and `list(method='TMM')`, `list(type='ratio')`, `list(fitType='parametric', blind=TRUE)`, `list(fitType='parametric', blind=TRUE)` is internally set for "CNF", "ESF", "VST", "rlog" respectively. Note the slot name of each element in the list is required. *E.g.* `list(method='TMM')` is expected while `list('TMM')` would cause errors.
Complete parameters of "CNF": https://www.rdocumentation.org/packages/edgeR/versions/3.14.0/topics/calcNormFactors
Complete parameters of "ESF": https://www.rdocumentation.org/packages/DESeq2/versions/1.12.3/topics/estimateSizeFactors
Complete parameters of "VST": https://www.rdocumentation.org/packages/DESeq2/versions/1.12.3/topics/varianceStabilizingTransformation
Complete parameters of "rlog": https://www.rdocumentation.org/packages/DESeq2/versions/1.12.3/topics/rlog

`log2.trans` Logical, TRUE or FALSE. If TRUE (default) and the selected normalization method does not use log2 scale by default ("ESF"), the output data is log2-transformed after normalization. If FALSE and the selected normalization method uses log2 scale by default ("VST", "rlog"), the output data is 2-exponent transformed after normalization.

`data.trans` This argument is deprecated and replaced by `log2.trans`. One of "log2", "exp2", and "none", corresponding to transform the count matrix by "log2", "2-based exponent", and "no transformation" respecitvely. The default is "none".

Value

If the input data is `SummarizedExperiment`, the retured value is also a `SummarizedExperiment` containing normalized data matrix and metadata (optional). If the input data is a `data.frame`, the returned value is a `data.frame` of normalized data and metadata (optional).

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
McCarthy, Davis J., Chen, Yunshun, Smyth, and Gordon K. 2012. "Differential Expression Analysis of Multifactor RNA-Seq Experiments with Respect to Biological Variation." Nucleic Acids Research 40 (10): 4288–97
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
McCarthy, Davis J., Chen, Yunshun, Smyth, and Gordon K. 2012. "Differential Expression Analysis of Multifactor RNA-Seq Experiments with Respect to Biological Variation." Nucleic Acids Research 40 (10): 4288–97
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. “Gene Expression Across Mammalian Organ Development.” Nature 571 (7766): 505–9

See Also

`calcNormFactors` in edgeR, and `estimateSizeFactors`, `varianceStabilizingTransformation`, `rlog` in DESeq2.

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as
## a "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.

## Set up toy data.
```

```
# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk)

# Normalize raw count data. The normalizing function "calcNormFactors" (McCarthy et al. 2012)
# with default settings is used.
df.nor.chk <- norm_data(data=df.chk, norm.fun='CNF', log2.trans=TRUE)
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', log2.trans=TRUE)
```

`norm_multi` *Normalize one or multiple count data sets.*

Description

Normalize count data of single cell and bulk provided in a `list` in co-clustering. The single cell and bulk data are combined, normalized and subsequently separated. The input single cell and bulk data are replaced by normalized data respectively.

Usage

```
norm_multi(dat.lis, cpm = FALSE, count.kp = FALSE)
```

Arguments

`dat.lis`	A named `list` containing count data of single cell and bulk, which are in form of `matrix`, `data.frame`, `dgCMatrix`, or `SingleCellExperiment`.
`cpm`	Logical. The count data are first normalized by `computeSumFactors`. If `TRUE`, the data is further normalized by counts per million (cpm). The default is `FALSE`.
`count.kp`	Logical. If `FALSE` (default), the count data is discarded and only log2-scale data are kept.

Value

A list of normalized single cell and bulk data.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x. Lun ATL, McCarthy DJ, Marioni JC (2016). "A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor." F1000Res., 5, 2122. doi: 10.12688/f1000research.9501.2. McCarthy DJ, Campbell KR, Lun ATL, Willis QF (2017). "Scater: pre-processing, quality control, normalisation and visualisation of single-cell RNA-seq data in R." Bioinformatics, 33, 1179-1186. doi: 10.1093/bioinformatics/btw777. Douglas Bates and Martin Maechler (2021). Matrix: Sparse and Dense Matrix Classes and Methods. R package version 1.4-0. https://CRAN.R-project.org/package=Matrix Morgan M, Obenchain V, Hester J, Pagès H (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24. 0, https://bioconductor.org/packages/SummarizedExpe
Vacher CM, Lacaille H, O'Reilly JJ, Salzbank J et al. Placental endocrine function shapes cerebellar development and social behavior. Nat Neurosci 2021 Oct;24(10):1392-1401. PMID: 34400844.
Ortiz C, Navarro JF, Jurek A, Märtin A et al. Molecular atlas of the adult mouse brain. Sci Adv 2020 Jun;6(26):eabb3446. PMID: 32637622

Examples

```
# Example bulk data of mouse brain for coclustering (Vacher et al 2021).
blk.mus.pa <- system.file("extdata/shinyApp/example", "bulk_mouse_cocluster.txt", package="spatialHeatmap")
blk.mus <- as.matrix(read.table(blk.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
blk.mus[1:3, 1:5]

# Example single cell data for coclustering (Ortiz et al 2020).
sc.mus.pa <- system.file("extdata/shinyApp/example", "cell_mouse_cocluster.txt", package="spatialHeatmap")
sc.mus <- as.matrix(read.table(sc.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
sc.mus[1:3, 1:5]

# Initial filtering.
blk.mus <- filter_data(data=blk.mus, sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 5), CV=c(0.2, 100), dir=NULL)
dim(blk.mus)
mus.lis <- filter_cell(lis=list(sc.mus=sc.mus), bulk=blk.mus, gen.rm=NULL, min.cnt=1, p.in.cell=0.5, p.in.gen=0.1

# Normalization: bulk and single cell are combined and normalized, then separated.
mus.lis.nor <- norm_multi(dat.lis=mus.lis, cpm=FALSE)
```

plot_dim *Plotting single cells in reduced dimensionalities*

Description

Plotting single cells in reduced dimensionalities

Usage

```
plot_dim(sce, dim, color.by, row.sel = NULL, x.break = NULL, y.break = NULL)
```

Arguments

sce	A SingleCellExperiment object with reduced dimensions seen by reducedDimNames(sce).
dim	One of PCA, UMAP, TSNE, the method for reducing dimensionality.
color.by	One of the column names in the colData slot of sce.
row.sel	A numeric vector of row numbers in the colData slot of sce. The cells corresponding to these rows are highlighted and plotted on top of other cells.
x.break, y.break	Two numeric vectors for x, y axis breaks respectively. E.g. seq(-10, 10, 2). The default is NULL.

Value

An object of ggplot.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. Morgan M, Obenchain V, Hester J, Pagès H (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24.0, https://bioconductor.org/packages/SummarizedExperiment. Lun ATL, McCarthy DJ, Marioni JC (2016). "A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor." F1000Res., 5, 2122. doi: 10.12688/f1000research.9501.2. McCarthy DJ, Campbell KR, Lun ATL, Willis QF (2017). "Scater: pre-processing, quality control, normalisation and visualisation of single-cell RNA-seq data in R." Bioinformatics, 33, 1179-1186. doi: 10.1093/bioinformatics/btw777.

Examples

```
library(scran); library(scuttle)
sce <- mockSCE(); sce <- logNormCounts(sce)
# Modelling the variance.
var.stats <- modelGeneVar(sce)
sce <- denoisePCA(sce, technical=var.stats, subset.row=rownames(var.stats))
plot_dim(sce, dim='PCA', color.by='Cell_Cycle')
# See function "cocluster" by running "?cocluster".
```

`profile_gene` *Plot Gene Expression Profiles in a Data Frame*

Description

Plot Gene Expression Profiles in a Data Frame

Usage

```
profile_gene(
  data,
  scale = "none",
  x.title = "Sample/conditions",
  y.title = "Value",
  text.size = 15,
  text.angle = 45
)
```

Arguments

`data`	A data frame, where rows are genes and columns are features/conditions.
`scale`	The way to to scale the data. If `none` (default), no scaling. If `row`, the data is scalaed independently. If `all`, all the data is scaled as a whole.
`x.title, y.title`	X-axis title and Y-axis title respectively.
`text.size`	The size of axis title and text.
`text.angle`	The angle of axis text.

Value

An image of ggplot.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/.

See Also

`spatial_enrich`

Examples

```
data(deg.table)
# Line graph of selected gene expression profile.
profile_gene(deg.table[1, ])
# See detailed examples in the function "spatial_enrich".
```

`random_para` *Generate random combinations of parameter settings*

Description

In coclustering, generate random combinations of parameter settings for validating optimal settings.

Usage

```
random_para(
  fil.set,
  norm,
  dimred,
  graph.meth,
  sim = round(seq(0.2, 0.8, by = 0.1), 1),
  sim.p = round(seq(0.2, 0.8, by = 0.1), 1),
  dim = seq(5, 40, by = 1),
  df.spd.opt
)
```

Arguments

`fil.set`	A character vector of filtering parameter set. E.g. `c('fil3', 'fil4')`.
`norm`	A character vector of normalization methods. E.g. `c('cpm')`.
`dimred`	A character vector of dimesionality reduction methods. E.g. `c('umap')`.
`graph.meth`	A character vector of graph-building methods. E.g. `c('knn', 'snn')`.
`sim`	Both are numeric scalars, ranging from 0 to 1. `sim` is a similarity (Spearman or Pearson correlation coefficient) cutoff between cells and `sim.p` is a proportion cutoff. In a certain cell cluster, cells having similarity >= `sim` with other cells in the same cluster at proportion >= `sim.p` would remain. Otherwise, they are discarded.

sim.p	Both are numeric scalars, ranging from 0 to 1. sim is a similarity (Spearman or Pearson correlation coefficient) cutoff between cells and sim.p is a proportion cutoff. In a certain cell cluster, cells having similarity >= sim with other cells in the same cluster at proportion >= sim.p would remain. Otherwise, they are discarded.
dim	Integer scalar specifying the minimum number of (principle components) PCs to retain in denoisePCA when coclustering single cells and bulk data. The default is 12.
df.spd.opt	A data.frame of optimized spd.set settings. These settings are avoided in the output random settings.

Value

A data.frame.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

Examples

```
df.spd.opt <- data.frame(sim=c(0.2, 0.4, 0.3), sim.p=c(0.8, 0.6, 0.7), dim=c(12, 14, 13))
df.para.rdn <- random_para(fil.set=c('fil3', 'fil4'), norm='cpm', dimred='umap', graph.meth=c('knn', 'snn'), sim=
```

---

read_cache *Read R Objects from Cache*

Description

Read R Objects from Cache

Usage

```
read_cache(dir, name, info = FALSE)
```

Arguments

dir	The directory path where cached data are located. It should be the path returned by save_cache.
name	The name of the object to retrieve, which is one of the entries in the "rname" column returned by setting info=TRUE.
info	Logical, TRUE or FALSE. If TRUE (default), the information of all tracked files in cache is returned in a table.

Value

An R object retrieved from the cache.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Lori Shepherd and Martin Morgan (2020). BiocFileCache: Manage Files Across Sessions. R package version 1.12.1.

Examples

```
# Save the object "iris" in the default cache "~/.cache/shm".
cache.pa <- save_cache(dir=NULL, overwrite=TRUE, iris)
# Retrieve "iris".
iris1 <- read_cache(cache.pa, 'iris')
```

`read_fr` *Import Data from Tabular Files*

Description

This function reads data from a tabular file, which is a wrapper of fread. If the tabular file contains both character and numeric columns, it is able to maintain the character or numeric attribute for each column in the returned data frame. In addition, it is able to detect separators automatically.

Usage

```
read_fr(input, header = TRUE, sep = "auto", fill = TRUE, check.names = FALSE)
```

Arguments

`input`	The file path.
`header`	One of TRUE, FALSE, or "auto". Default is TRUE. Does the first data line contain column names, according to whether every non-empty field on the first data line is type character? If "auto" or TRUE is supplied, any empty column names are given a default name.
`sep`	The separator between columns. Defaults to the character in the set `[,\t |;:]` that separates the sample of rows into the most number of lines with the same number of fields. Use `NULL` or `""` to specify no separator; i.e. each line a single character column like `base::readLines` does.
`fill`	Logical (default is TRUE). If TRUE then in case the rows have unequal length, blank fields are implicitly filled.
`check.names`	default is `FALSE`. If `TRUE` then the names of the variables in the `data.table` are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by `make.names`) so that they are, and also to ensure that there are no duplicates.

Value

A data frame.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Matt Dowle and Arun Srinivasan (2019). data.table: Extension of 'data.frame'. R package version 1.12.8. https://CRAN.R-project.org/package=data.table

Examples

```
sh.tar <- system.file('extdata/shinyApp/example/target_arab.txt', package='spatialHeatmap')
target.sh <- read_fr(sh.tar); target.sh[60:63, ]
```

---

read_hdf5 *Read Data from the Shiny App Database*

Description

This function is used to extract data from the Shiny App Database "data_shm.tar".

Usage

```
read_hdf5(file, prefix)
```

Arguments

file	The path of "data_shm.tar" generated by write_hdf5.
prefix	A vector of data set identifiers such as c('expr_arab', 'expr_chicken', 'df_pair'). The vector elements must come from the "data" column in the pairing table that is made when calling write_hdf5.

Value

A list of data set and/or the pairing table.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/ Hervé Pagès (2020). HDF5Array: HDF5 backend for DelayedArray objects. R package version 1.16.1.

Examples

```
## The examples below demonstrate 1) how to dump Expression Atlas data set into the Shiny database;
## 2) how to dump GEO data set into the Shiny database; 3) how to include aSVGs of multiple
## development stages; 4) how to read the database; 5) how to create customized Shiny app with
## the database.

# 1. Dump data from Expression Atlas into "data_shm.tar" using ExpressionAtlas package (Keays 2019).

# The chicken data derived from an RNA-seq analysis on developments of 7 chicken organs under 9
# time points (Cardoso-Moreira et al. 2019) is chosen as example.
# The following searches the Expression Atlas for expression data from 'heart' and 'gallus'.
library(ExpressionAtlas)
cache.pa <- '~/.cache/shm' # The path of cache.
all.chk <- read_cache(cache.pa, 'all.chk') # Retrieve data from cache.
if (is.null(all.chk)) { # Save downloaded data to cache if it is not cached.
 all.chk <- searchAtlasExperiments(properties="heart", species="gallus")
 save_cache(dir=cache.pa, overwrite=TRUE, all.chk)
}

all.chk[3, ]
rse.chk <- read_cache(cache.pa, 'rse.chk') # Read data from cache.
if (is.null(rse.chk)) { # Save downloaded data to cache if it is not cached.
  rse.chk <- getAtlasData('E-MTAB-6769')[[1]][[1]]
  save_cache(dir=cache.pa, overwrite=TRUE, rse.chk)
}
# The downloaded data is stored in "SummarizedExperiment" by default (SE, M. Morgan et al. 2018).
# The experiment design is described in the "colData" slot. The following returns first three rows.
colData(rse.chk)[1:3, ]
# In the "colData" slot, it is required to define the "sample" and "condition" columns respectively.
# Both "sample" and "condition" are general terms. The former refers to entities where the numeric
# data are measured such as cell organelles, tissues, organs, ect. while the latter denotes
# experimental treatments such as drug dosages, gender, trains, time series, PH values, ect. In the
# downloaded data, the two columns are not explicitly defined, so "organism_part" and "age" are
# selected and renamed as "sample" and "condition" respectively.
colnames(colData(rse.chk))[c(6, 8)] <- c('condition', 'sample'); colnames(colData(rse.chk))
# The raw RNA-Seq count are preprocessed with the following steps: (1) normalization,
# (2) aggregation of replicates, and (3) filtering of reliable expression data. The details of
# these steps are explained in the pacakge vignette.
 browseVignettes('spatialHeatmap')
se.nor.chk <- norm_data(data=rse.chk, norm.fun='ESF', log2.trans=TRUE) # Normalization
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='sample', con.factor='condition',
aggr='mean') # Replicate agggregation using mean
# Genes are filtered out if not meet these criteria: expression values are at least 5 in at least
# 1% of all samples, coeffient of variance is between 0.6 and 100.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='sample', con.factor='condition',
```

```
pOA=c(0.01, 5), CV=c(0.6, 100), dir=NULL)
# The aSVG file corresponding with the data is pre-packaged and copied to a temporary directory.
dir.svg <- paste0(tempdir(check=TRUE), '/svg_shm') # Temporary directory.
if (!dir.exists(dir.svg)) dir.create(dir.svg)
# Path of the aSVG file.
svg.chk <- system.file("extdata/shinyApp/example", 'gallus_gallus.svg', package="spatialHeatmap")
file.copy(svg.chk, dir.svg, overwrite=TRUE) # Copy the aSVG file.

# 2. Dump data from GEO into "data_shm.tar" using GEOquery package (S. Davis and Meltzer 2007).

# The Arabidopsis thaliana (Arabidopsis) data from an microarray assay of hypoxia treatment on
# Arabidopsis root and shoot cell types (Mustroph et al. 2009) is selected as example.
# The data set is downloaded with the accession number "GSE14502". It is stored in ExpressionSet
# container (W. Huber et al. 2015) by default, and then converted to a SummarizedExperiment object.
library(GEOquery)
gset <- read_cache(cache.pa, 'gset') # Retrieve data from cache.
if (is.null(gset)) { # Save downloaded data to cache if it is not cached.
  gset <- getGEO("GSE14502", GSEMatrix=TRUE, getGPL=TRUE)[[1]]
  save_cache(dir=cache.pa, overwrite=TRUE, gset)
}
se.sh <- as(gset, "SummarizedExperiment") # Converted to SummarizedExperiment
# The gene symbol identifiers are extracted from the rowData component to be used as row names.
rownames(se.sh) <- make.names(rowData(se.sh)[, 'Gene.Symbol'])
# A slice of the experimental design in colData slot is shown. Both the samples and conditions
# are contained in the "title" column. The samples are indicated by promoters: pGL2 (root
# atrichoblast epidermis), pCO2 (root cortex meristematic zone), pSCR (root endodermis),
# pWOL (root vasculature), etc., and conditions are control and hypoxia.
colData(se.sh)[60:63, 1:4]
# Since the samples and conditions need to be listed in two independent columns, like the the
# chicken data above, a targets file is recommended to separate samples and conditions. The main
# reason to choose this Arabidopdis data is to illusrate the usage of targets file when necessary.
# A pre-packaged targets file is accessed and partially shown below.
sh.tar <- system.file('extdata/shinyApp/example/target_arab.txt', package='spatialHeatmap')
target.sh <- read_fr(sh.tar); target.sh[60:63, ]
# Load custom the targets file into colData slot.
colData(se.sh) <- DataFrame(target.sh)
# This data set was already normalized with the RMA algorithm (Gautier et al. 2004). Thus, the
# pre-processing steps are restricted to aggregation of replicates and filtering of reliably
# expressed genes.
# Replicate agggregation using mean
se.aggr.sh <- aggr_rep(data=se.sh, sam.factor='samples', con.factor='conditions', aggr='mean')
se.fil.arab <- filter_data(data=se.aggr.sh, sam.factor='samples', con.factor='conditions',
pOA=c(0.03, 6), CV=c(0.30, 100), dir=NULL) # Filtering of genes with low intensities and variance

# Similarly, the aSVG file corresponding to this data is pre-packaged and copied to the same
# temporary directory.
svg.arab <- system.file("extdata/shinyApp/example", 'arabidopsis.thaliana_organ_shm.svg',
package="spatialHeatmap")
file.copy(svg.arab, dir.svg, overwrite=TRUE)

# 3. The random data and aSVG files of two development stages of Arabidopsis organs.

# The gene expression data is randomly generated and pre-packaged.
```

```
pa.growth <- system.file("extdata/shinyApp/example", 'random_data_multiple_aSVGs.txt',
package="spatialHeatmap")
dat.growth <- read_fr(pa.growth); dat.growth[1:3, ]
# Paths of the two corresponsing aSVG files.
svg.arab1 <- system.file("extdata/shinyApp/example", 'arabidopsis.thaliana_organ_shm1.svg',
package="spatialHeatmap")
svg.arab2 <- system.file("extdata/shinyApp/example", 'arabidopsis.thaliana_organ_shm2.svg',
package="spatialHeatmap")
# Copy the two aSVG files to the same temporary directory.
file.copy(c(svg.arab1, svg.arab2), dir.svg, overwrite=TRUE)

# 4. Include aSVG templates of raster images.

pa.leaf <- system.file("extdata/shinyApp/example", 'dat_overlay.txt',
package="spatialHeatmap")
dat.leaf <- read_fr(pa.leaf); dat.leaf[1:2, ]
# Paths of the two aSVG files.
svg.leaf1 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm1.svg',
package="spatialHeatmap")
svg.leaf2 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm2.svg',
package="spatialHeatmap")
# Paths of the two corresponsing raster images of templates.
tmp.leaf1 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm1.png',
package="spatialHeatmap")
tmp.leaf2 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm2.png',
package="spatialHeatmap")
# Copy the two aSVG and two template files to the same temporary directory.
file.copy(c(svg.leaf1, svg.leaf2, tmp.leaf1, tmp.leaf2), dir.svg, overwrite=TRUE)

# Make the pairing table, which describes matchings between the data and image files.
df.pair <- data.frame(name=c('chicken', 'arab', 'growth', 'leaf'), data=c('expr_chicken', 'expr_arab',
'random_data_multiple_aSVGs', 'leaf'), aSVG=c('gallus_gallus.svg', 'arabidopsis.thaliana_organ_shm.svg',
'arabidopsis.thaliana_organ_shm1.svg;arabidopsis.thaliana_organ_shm2.svg',
'maize_leaf_shm1.svg;maize_leaf_shm1.png;maize_leaf_shm2.svg;maize_leaf_shm2.png'))
# Note that multiple aSVGs should be concatenated by comma, semicolon, or single space.
df.pair

# Organize the data and pairing table in a list, and create the database.
dat.lis <- list(df_pair=df.pair, expr_chicken=se.fil.chk, expr_arab=se.fil.arab,
random_data_multiple_aSVGs=dat.growth, leaf=dat.leaf)
# Create the database in a temporary directory "db_shm".
dir.db <- paste0(tempdir(check=TRUE), '/db_shm') # Temporary directory.

if (!dir.exists(dir.db)) dir.create(dir.db)
write_hdf5(dat.lis=dat.lis, dir=dir.db, svg.dir=dir.svg, replace=TRUE)

# 4. Read data and/or pairing table from "data_shm.tar".
dat.lis1 <- read_hdf5(paste0(dir.db, '/data_shm.tar'), names(dat.lis))
```

reduce_rep *Reduce sample replicates*

Description

In an expression profile matrix such as RNA-seq count table, where columns and rows are samples and biological molecules respectively, reduce sample replicates according to sum of correlation coefficients (Pearson, Spearman, Kendall).

Usage

```
reduce_rep(dat, n = 3, sim.meth = "pearson")
```

Arguments

dat	Abundance matrix in form of data.frame or matrix, where columns and rows are samples and biological molecules respectively. For example, gene exression matrix generated in RNA-seq.
n	An integer, the max number of replicates to keep per sample (e.g. tissue type). Within each sample, pairwise correlations are calculated among all replicates, and the correlations between one replicate and other relicates are summed. The replicates with top n largest sums are retained in each sample.
sim.meth	One of pearson (default), kendall, or spearman, indicating which correlation coefficient method to use for calculating similarities between replicates.

Value

A matrix.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

Examples

```
# Random abundance matrix.
dat <- matrix(rnorm(100), nrow=10)
# Two samples, each has 5 replicates.
colnames(dat) <- c(rep('sampleA', 5), rep('sampleB', 5))
rownames(dat) <- paste0('gene', seq_len(nrow(dat)))
reduce_rep(dat)
```

refine_cluster *Refine single cell clusters*

Description

In each cell cluster, the pairwise Spearman or Pearson correlation coefficients (similarities) are calculated between cells. Cells having similarities over `sim` with other cells in the same cluster at proportion over `sim.p` remain, and other cells are filtered out. The resulting clusters are more homogeneous.

Usage

```
refine_cluster(
  sce.clus,
  sim = 0.2,
  sim.p = 0.8,
  sim.meth = "spearman",
  verbose = TRUE
)
```

Arguments

`sce.clus`	The single cell data in form of `SummarizedExperiment`, where cluster assignments are stored in the `label` column in `colData` slot.
`sim, sim.p`	Both are numeric scalars, ranging from 0 to 1. `sim` is a similarity (Spearman or Pearson correlation coefficient) cutoff between cells and `sim.p` is a proportion cutoff. In a certain cell cluster, cells having similarity >= `sim` with other cells in the same cluster at proportion >= `sim.p` would remain. Otherwise, they are discarded.
`sim.meth`	Method to compute similarities between cells, `spearman` or `pearson`. The `logcount` values in `sce.clus` are used.
`verbose`	Logical. If `TRUE` (default), intermediate messages are printed.

Value

A `SummarizedExperiment` object with some cells discarded.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Morgan M, Obenchain V, Hester J, Pagès H (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24.0, https://bioconductor.org/packages/SummarizedExperiment.
mezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x. Lun ATL, McCarthy DJ, Marioni JC (2016). "A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor." F1000Res., 5, 2122. doi: 10.12688/f1000research.9501.2.
McCarthy DJ, Campbell KR, Lun ATL, Willis QF (2017). "Scater: pre-processing, quality control, normalisation and visualisation of single-cell RNA-seq data in R." Bioinformatics, 33, 1179-1186. doi: 10.1093/bioinformatics/btw777.

Examples

```
library(scran); library(scuttle)
sce <- mockSCE(); sce <- logNormCounts(sce)
# Modelling the variance.
var.stats <- modelGeneVar(sce)
sce <- denoisePCA(sce, technical=var.stats, subset.row=rownames(var.stats))

sce.clus <- cluster_cell(data=sce, prop=0.1, min.dim=5, max.dim=50, graph.meth='snn', dimred='PCA')
# Clusters.
table(colData(sce.clus)$label)

cell.refined <- refine_cluster(sce.clus, sim=0.5, sim.p=0.8, sim.meth='spearman', verbose=TRUE)


# See details in function "cocluster" by running "?cocluster".
```

---

return_feature *Return aSVG Files Relevant to Target Features*

Description

This function parses a collection of aSVG files and returns those containing target features in a data frame. Successful spatial heatmap plotting requires the aSVG features of interest have matching samples (cells, tissues, *etc*) in the data. To meet this requirement, the returned features could be used to replace target sample counterparts in the data. Alternatively, the target samples in the data could be used to replace matching features in the aSVG through function `update_feature`. Refer to function `spatial_hm` for more details on aSVG files.

Usage

```
return_feature(
  feature,
  species,
  keywords.any = TRUE,
  remote = NULL,
```

```
  dir = NULL,
  svg.path = NULL,
  desc = FALSE,
  match.only = TRUE,
  return.all = FALSE
)
```

Arguments

feature	A vector of target feature keywords (case insentitive), which is used to select aSVG files from a collection. *E.g.* c('heart', 'brain'). If NA or NULL, all features of all SVG files matching species are returned.
species	A vector of target species keywords (case insentitive), which is used to select aSVG files from a collection. *E.g.* c('gallus'). If NA or NULL, all SVG files in dir are queried.
keywords.any	Logical, TRUE or FALSE. Default is TRUE. The internal searching is case-insensitive. The space, dot, hypen, semicolon, comma, forward slash are treated as separators between words and not counted in searching. If TRUE, every returned hit contains at least one word in the feature vector and at least one word in the species vector, which means all the possible hits are returned. *E.g.* "prefrontal cortex" in "homo_sapiens.brain.svg" would be returned if feature=c('frontal') and species=c('homo'). If FALSE, every returned hit contains at least one exact element in the feature vector and all exact elements in the species vector. *E.g.* "frontal cortex" rather than "prefrontal cortex" in "homo_sapiens.brain.svg" would be returned if feature=c('frontal cortex') and species=c('homo sapiens', 'brain').
remote	Logical, FALSE or TRUE. If TRUE (default), the remote EBI aSVG repository https://github.com/ebi-gene-expression-group/anatomogram/tree/master/src/svg and spatialHeatmap aSVG Repository https://github.com/jianhaizhang/spatialHeatmap_aSVG_Repository developed in this project are queried.
dir	The directory path of aSVG files. If remote is TRUE, the returned aSVG files are saved in this directory. Note existing aSVG files with same names as returned ones are overwritten. If remote is FALSE, user-provided (local) aSVG files should be saved in this directory for query. Default is NULL.
svg.path	The path of a specific aSVG file. If the provided aSVG file exists, only features of this file are returned and there will be no querying process. Default is NULL.
desc	Logical, FALSE or TRUE. Default is FALSE. If TRUE, the feature descriptions from the R package "rols" (Laurent Gatto 2019) are added. If too many features are returned, this process takes a long time.
match.only	Logical, TRUE or FALSE. If TRUE (default), only target features are returned. If FALSE, all features in the matching aSVG files are returned, and the matching features are moved on the top of the data frame.
return.all	Logical, FALSE or TRUE. Default is FALSE. If TRUE, all features together with all respective aSVG files are returned, regardless of feature and species.

Value

A data frame containing information on target features and aSVGs.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Laurent Gatto (2019). rols: An R interface to the Ontology Lookup Service. R package version 2.14.0. http://lgatto.github.com/rols/
Hadley Wickham, Jim Hester and Jeroen Ooms (2019). xml2: Parse XML. R package version 1.2.2. https://CRAN.R-project.org/package=xml2
R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505-9

Examples

```
# This function is able to work on the EBI aSVG repository directly: https://github.com/
# ebi-gene-expression-group/anatomogram/tree/master/src/svg. The following shows how to
# download a chicken aSVG containing spatial features of 'brain' and 'heart'. An empty
# directory is recommended so as to avoid overwriting existing SVG files.
# Here "~/test" is used.


# Make an empty directory "~/test" if not exist.
if (!dir.exists('~/test')) dir.create('~/test')
# Remote aSVG repos.
data(aSVG.remote.repo)
tmp.dir <- normalizePath(tempdir(check=TRUE), winslash="/", mustWork=FALSE)
tmp.dir.ebi <- paste0(tmp.dir, '/ebi.zip')
tmp.dir.shm <- paste0(tmp.dir, '/shm.zip')
# Download the remote aSVG repos as zip files. According to Bioconductor's
# requirements, downloadings are not allowed inside functions, so the repos are
# downloaded before calling "return_feature".
download.file(aSVG.remote.repo$ebi, tmp.dir.ebi)
download.file(aSVG.remote.repo$shm, tmp.dir.shm)
remote <- list(tmp.dir.ebi, tmp.dir.shm)
# Query the remote aSVG repos.
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir='~/test',
match.only=FALSE, remote=remote)
feature.df
# The path of downloaded aSVG.
svg.chk <- '~/test/gallus_gallus.svg'


# The spatialHeatmap package has a small aSVG collection and can be used to demonstrate the
# local query.
# Get the path of local aSVGs from the package.
svg.dir <- system.file("extdata/shinyApp/example", package="spatialHeatmap")
# Query the local aSVG repo. The "species" argument is set NULL on purpose so as to illustrate
```

```
# how to select the target aSVG among all matching aSVGs.
feature.df <- return_feature(feature=c('heart', 'brain'), species=NULL, dir=svg.dir,
match.only=FALSE, remote=NULL)
# All matching aSVGs.
unique(feature.df$SVG)
# Select the target aSVG of chicken.
subset(feature.df, SVG=='gallus_gallus.svg')
```

---

save_cache *Save R Objects in Cache*

Description

Save R Objects in Cache

Usage

```
save_cache(dir = NULL, overwrite = TRUE, ...)
```

Arguments

dir	The directory path to save the cached data. Default is NULL and the cached data is stored in ~/.cache/shm.
overwrite	Logical, TRUE or FALSE. Default is TRUE and data in the cache with the same name of the object in ... will be overwritten.
...	A single R object to be cached.

Value

The directory path of the cache.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Lori Shepherd and Martin Morgan (2020). BiocFileCache: Manage Files Across Sessions. R package version 1.12.1.

Examples

```
# Save the object "iris" in the default cache "~/.cache/shm".
cache.pa <- save_cache(dir=NULL, overwrite=TRUE, iris)
```

shiny_shm *Integrated Shiny App*

Description

In additon to generating spatial heatmaps and corresponding item (genes, proteins, metabolites, *etc.*) context plots from R, spatialHeatmap includes a Shiny App (https://shiny.rstudio.com/) that provides access to the same functionalities from an intuitive-to-use web browser interface. Apart from being very user-friendly, this App conveniently organizes the results of the entire visualization workflow in a single browser window with options to adjust the parameters of the individual components interactively. Upon launched, the app automatically displays a pre-formatted example. To use this app, the data matrix (*e.g.* gene expression matrix) and aSVG image are uploaded as tabular text (*e.g.* in CSV or TSV format) and SVG file, respectively. To also allow users to upload data matrix stored in `SummarizedExperiment` objects, one can export them from R to a tabular file with the `filter_data` function. In this function call, the user sets a desired directory path under `dir`. Within this directory the tabular file will be written to "customComputedData/sub_matrix.txt" in TSV format. The column names in the exported tabular file preserve the experimental design information from the `colData` slot by concatenating the corresponding sample and condition information separated by double underscores. To interactively view functional descriptions by moving the cursor over network nodes, the corresponding annotation column needs to be present in the `rowData` slot and its column name assigned to the `ann` argument. In the exported tabular file the extra annotation column is appended to the expression matrix. See function `filter_data` for details. If the subsetted data matrix in the Matrix Heatmap is too large, *e.g.* >10,000 rows, the "customComputedData" under "Step 1: data sets" is recommended. Since this subsetted matrix is fed to the Network, and the internal computation of adjacency matrix and module identification would be intensive. In order to protect the app from crash, the intensive computation should be performed outside the app, then upload the results under "customComputedData". When using "customComputedData", the data matrix to upload is the subsetted matrix "sub_matrix.txt" generated with `submatrix`, which is a TSV-tabular text file. The adjacency matrix and module assignment to upload are "adj.txt" and "mod.txt" generated in function `adj_mod` respectively. Note, "sub_matrix.txt", "adj.txt", and "mod.txt" are downstream to the same call on `filter_data`, so the three files should not be mixed between different filtering when uploading. See the instruction page in the app for details. The large matrix issue could be resolved by increasing the subsetting strigency to get smaller matrix in `submatrix` in most cases. Only in rare cases users cannot avoid very large subsetted matrix, the "customComputedData" is recommended.

Usage

```
shiny_shm()
```

Value

A web browser based Shiny app.

Details

No argument is required, this function launches the Shiny app directly.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

https://www.w3schools.com/graphics/svg_intro.asp

https://shiny.rstudio.com/tutorial/

https://shiny.rstudio.com/articles/datatables.html

https://rstudio.github.io/DT/010-style.html

https://plot.ly/r/heatmaps/

https://www.gimp.org/tutorials/

https://inkscape.org/en/doc/tutorials/advanced/tutorial-advanced.en.html

http://www.microugly.com/inkscape-quickguide/

https://cran.r-project.org/web/packages/visNetwork/vignettes/Introduction-to-visNetwork.html

Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2017). shiny: Web Application Framework for R. R package version 1.0.3. https://CRAN.R-project.org/package=shiny

Winston Chang and Barbara Borges Ribeiro (2017). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.6.1. https://CRAN.R-project.org/package=shinydashboard

Paul Murrell (2009). Importing Vector Graphics: The grImport Package for R. Journal of Statistical Software, 30(4), 1-37. URL http://www.jstatsoft.org/v30/i04/

Jeroen Ooms (2017). rsvg: Render SVG Images into PDF, PNG, PostScript, or Bitmap Arrays. R package version 1.1. https://CRAN.R-project.org/package=rsvg

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Yihui Xie (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2. https://CRAN.R-project.org/package=DT

Baptiste Auguie (2016). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.2.1. https://CRAN.R-project.org/package=gridExtra

Andrie de Vries and Brian D. Ripley (2016). ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'. R package version 0.1-20. https://CRAN.R-project.org/package=ggdendro

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 doi:10.1186/1471-2105-9-559

Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/

Simon Urbanek and Jeffrey Horner (2015). Cairo: R graphics device using cairo graphics library for creating high-quality bitmap (PNG, JPEG, TIFF), vector (PDF, SVG, PostScript) and display (X11 and Win32) output. R package version 1.5-9. https://CRAN.R-project.org/package=Cairo

R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/

Duncan Temple Lang and the CRAN Team (2017). XML: Tools for Parsing and Generating XML Within R and S-Plus. R package version 3.98-1.9. https://CRAN.R-project.org/package=XML

Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec and Pedro Despouy (NA). plotly: Create Interactive Web Graphics via 'plotly.js'. https://plot.ly/r, https://cpsievert.github.io/plotly_book/, https://github.com/ropensci/plotly

Matt Dowle and Arun Srinivasan (2017). data.table: Extension of 'data.frame'. R package version 1.10.4. https://CRAN.R-project.org/package=data.table

R. Gentleman, V. Carey, W. Huber and F. Hahne (2017). genefilter: genefilter: methods for filtering genes from high-throughput experiments. R package version 1.58.1.

Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/

Almende B.V., Benoit Thieurmel and Titouan Robert (2017). visNetwork: Network Visualization using 'vis.js' Library. R package version 2.0.1. https://CRAN.R-project.org/package=visNetwork

Examples

```
shiny_shm()
```

`spatial_enrich` *Identify Spatial Feature-Specifcally Expressed Genes*

Description

This functionality is an extension of the spatial heatmap. It identifies spatial feature-specifically expressed genes and thus enables the spatial heatmap to visualize feature-specific profiles. The spatial features include cellular compartments, tissues, organs, *etc*. The function compares the target feature with all other selected features in a pairwise manner. The genes significantly up- or down-regulated in the target feature across all pairwise comparisons are denoted final target feature-specifcally expressed genes. The underlying methods include edgeR (Robinson et al, 2010), limma (Ritchie et al, 2015), DESeq2 (Love et al, 2014), distinct (Tiberi et al, 2020). The feature-specific genes are first detected with each method and can be summarized across methods.
In addition to feature-specific genes, this function is also able to identify genes specifically expressed in certain condition or in composite factor. The latter is a combination of multiple expermental factors. `E.g.` the spatiotemporal factor is a combination of feature and time points.

Usage

```
spatial_enrich(
  data,
  methods = c("edgeR", "limma"),
  norm = "TMM",
  log2.trans.dis = TRUE,
  log2.fc = 1,
  p.adjust = "BH",
  fdr = 0.05,
  aggr = "mean",
  log2.trans.aggr = TRUE
)
```

Arguments

data	A SummarizedExperiment object, which is returned by sub_data. The colData slot is required to contain at least two columns of "features" and "factors" respectively. The rowData slot can optionally contain a column of discriptions of each gene and the column name should be metadata.
methods	One or more of edgeR, limma, DESeq2, distinct. The default is c('edgeR', 'limma').
norm	The normalization method (TMM, RLE, upperquartile, none) in edgeR. The default is TMM. Details: https://www.rdocumentation.org/packages/edgeR/versions/3.14.0/topics/calcNormF
log2.trans.dis	Logical, only applicable when distinct is in methods. The default is TRUE, and the count data is transformed to log-2 scale.
log2.fc	The log2-fold change cutoff. The default is 1.
p.adjust	The method (holm, hochberg, hommel, bonferroni, BH, BY, fdr, none) to adjust p values in multiple hypothesis testing. The default is BH.
fdr	The FDR cutoff. The default is 0.05.
aggr	One of mean (default), median. The method to aggregated replicates in the data frame of feature-specific genes.
log2.trans.aggr	Logical. If TRUE (default), the aggregated data (see aggr) is transformed to log2-scale, included in the returned data frame of feature-specific genes, and would be further used in the spatial heatmaps.

Value

A nested list containing the feature-specific genes summarized across methods within methods.

Author(s)

Jianhai Zhang <jianhai.zhang@email.ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
Robinson MD, McCarthy DJ and Smyth GK (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26, 139-140
Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7), e47.
Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550 (2014)

Simone Tiberi and Mark D. Robinson. (2020). distinct: distinct: a method for differential analyses via hierarchical permutation tests. R package version 1.2.0. https://github.com/SimoneTiberi/distinct

Examples

```
## In the following examples, the toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, it is
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769".

library(SummarizedExperiment)

## Set up toy data.

# Access toy data.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data. It should be made
# based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in this
# package and accessed below.
# Access the count table.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data in "SummarizedExperiment".
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene metadata, but not required. Only the
# column named "metadata" will be recognized.
# Pseudo row metadata.
metadata <- paste0('meta', seq_len(nrow(count.chk))); metadata[1:3]
rowData(se.chk) <- DataFrame(metadata=metadata)

# Subset the data by selected features (brain, heart, kidney) and factors (day10, day12).
data.sub <- sub_data(data=se.chk, feature='organism_part', features=c('brain', 'heart',
'kidney'), factor='age', factors=c('day10', 'day12'), com.by='feature', target='brain')

## As conventions, raw sequencing count data should be normalized and filtered to
## reduce noise. Since normalization will be performed in spatial enrichment, only filtering
## is required.

# Filter out genes with low counts and low variance. Genes with counts over 5 in
# at least 10% samples (pOA), and coefficient of variance (CV) between 3.5 and 100 are
# retained.
data.sub.fil <- filter_data(data=data.sub, sam.factor='organism_part', con.factor='age',
pOA=c(0.1, 5), CV=c(0.7, 100), dir=NULL)
```

```
# Identify brain-specifically expressed genes relative to heart and kidney, where day10 and
# day12 are treated as replicates.
deg.lis <- spatial_enrich(data.sub.fil)
# All up- and down-regulated genes in brain across methods. On the right is the data after
# replicates aggregated, and will be used in the spatial heatmaps.
deg.lis$deg.table[1:3, ]
# Up-regulated genes detected by edgeR.
deg.lis$lis.up.down$up.lis$edgeR.up[1:5]
# The aSVG path.
svg.chk <- system.file("extdata/shinyApp/example", "gallus_gallus.svg",
package="spatialHeatmap")
# Plot one brain-specific gene in spatial heatmap.
spatial_hm(svg.path=svg.chk, data=deg.lis$deg.table, ID=deg.lis$deg.table$gene[1], legend.r=1.9, legend.nrow=2,
# Overlap of up-regulated brain-specific genes across methods.
deg_ovl(deg.lis$lis.up.down, type='up', plot='upset')
deg_ovl(deg.lis$lis.up.down, type='up', plot='matrix')
# Overlap of down-regulated brain-specific genes across methods.
deg_ovl(deg.lis$lis.up.down, type='down', plot='upset')
deg_ovl(deg.lis$lis.up.down, type='down', plot='matrix')
# Line graph of gene expression profile.
profile_gene(deg.lis$deg.table[1, ])
```

`spatial_hm` *Plot Spatial Heatmaps*

Description

The input are a pair of annotated SVG (aSVG) file and formatted data (`vector`, `data.frame`, `SummarizedExperiment`). In the former, spatial features are represented by shapes and assigned unique identifiers, while the latter are numeric values measured from these spatial features and organized in specific formats. In biological cases, aSVGs are anatomical or cell structures, and data are measurements of genes, proteins, metabolites, *etc*. in different samples (*e.g.* cells, tissues). Data are mapped to the aSVG according to identifiers of assay samples and aSVG features. Only the data from samples having matching counterparts in aSVG features are mapped. The mapped features are filled with colors translated from the data, and the resulting images are termed spatial heatmaps. Note, "sample" and "feature" are two equivalent terms referring to cells, tissues, organs *etc.* where numeric values are measured. Matching means a target sample in data and a target spatial feature in aSVG have the same identifier.

This function is designed as much flexible as to achieve optimal visualization. For example, subplots of spatial heatmaps can be organized by gene or condition for easy comparison, in multi-layer anotomical structures selected tissues can be set transparent to expose burried features, color scale is customizable to highlight difference among features. This function also works with many other types of spatial data, such as population data plotted to geographic maps.

Usage

```
spatial_hm(
  svg.path,
  data,
```

```
assay.na = NULL,
sam.factor = NULL,
con.factor = NULL,
ID,
sce.dimred = NULL,
dimred = "PCA",
tar.cell = "matched",
tar.bulk,
profile = FALSE,
cell.group = NULL,
tmp.path = NULL,
charcoal = FALSE,
alpha.overlay = 1,
lay.shm = "gene",
ncol = 2,
col.com = c("yellow", "orange", "red"),
col.bar = "selected",
sig.thr = c(NA, NA),
cores = NA,
bar.width = 0.08,
bar.title.size = 0,
trans.scale = NULL,
ft.trans = NULL,
tis.trans = ft.trans,
lis.rematch = NULL,
legend.r = 0.2,
sub.title.size = 11,
sub.title.vjust = 2,
legend.plot = "all",
ft.legend = "identical",
bar.value.size = 10,
legend.plot.title = "Legend",
legend.plot.title.size = 11,
legend.ncol = NULL,
legend.nrow = NULL,
legend.position = "bottom",
legend.direction = NULL,
legend.key.size = 0.02,
legend.text.size = 12,
angle.text.key = NULL,
position.text.key = NULL,
legend.2nd = FALSE,
position.2nd = "bottom",
legend.nrow.2nd = NULL,
legend.ncol.2nd = NULL,
legend.key.size.2nd = 0.03,
legend.text.size.2nd = 10,
angle.text.key.2nd = 0,
```

```
  position.text.key.2nd = "right",
  dim.lgd.pos = "bottom",
  dim.lgd.nrow = 2,
  dim.lgd.text.size = 8,
  add.feature.2nd = FALSE,
  label = FALSE,
  label.size = 4,
  label.angle = 0,
  hjust = 0,
  vjust = 0,
  opacity = 1,
  key = TRUE,
  line.size = 0.2,
  line.color = "grey70",
  relative.scale = NULL,
  verbose = TRUE,
  out.dir = NULL,
  animation.scale = 1,
  selfcontained = FALSE,
  video.dim = "640x480",
  res = 500,
  interval = 1,
  framerate = 1,
  bar.width.vdo = 0.1,
  legend.value.vdo = NULL,
  ...
)
```

Arguments

svg.path	The path of aSVG file(s). *E.g.*: system.file("extdata/shinyApp/example", "gallus_gallus.svg", package="spatialHeatmap"). Multiple aSVGs are also accepted, such as aSVGs depicting organs development across mutiple times. In this case, the aSVGs should be indexed with suffixes "_shm1", "_shm2", ..., such as "arabidopsis.thaliana_organ_shm1.svg", "arabidopsis.thaliana_organ_shm2.svg", and the paths of these aSVGs be provided in a character vector. See return_feature for details on how to directly download aSVGs from the EBI aSVG repository https://github.com/ebi-gene-expression-group/anatomogram/tree/master/src/svg and spatialHeatmap aSVG Repository https://github.com/jianhaizhang/spatialHeatmap_aSVG_Repository developed in this project.
data	An object of data.frame or SummarizedExperiment. In either case, the columns and rows should be sample/conditions and assayed items (*e.g.* genes, proteins, metabolites) respectively. If data.frame, the column names should follow the naming scheme "sample__condition". The "sample" is a general term and stands for cells, tissues, organs, *etc*., where the values are measured. The "condition" is also a general term and refers to experiment treatments applied to "sample" such as drug dosage, temperature, time points, *etc*. If certain samples are not

expected to be colored in "spatial heatmaps" (see spatial_hm), they are not required to follow this naming scheme. In the downstream interactive network (see network), if users want to see node annotation by mousing over a node, a column of row item annotation could be optionally appended to the last column.
In the case of SummarizedExperiment, the assays slot stores the data matrix. Similarly, the rowData slot could optionally store a data frame of row item antation, which is only relevant to the interactive network. The colData slot usually contains a data frame with one column of sample replicates and one column of condition replicates. It is crucial that replicate names of the same sample or condition must be identical. *E.g.* If sampleA has 3 replicates, "sampleA", "sampleA", "sampleA" is expected while "sampleA1", "sampleA2", "sampleA3" is regarded as 3 different samples. If original column names in the assay slot already follow the "sample__condition" scheme, then the colData slot is not required at all.
In the function spatial_hm, this argument can also be a numeric vector. In this vector, every value should be named, and values expected to color the "spatial heatmaps" should follow the naming scheme "sample__condition".
In certain cases, there is no condition associated with data. Then in the naming scheme of data frame or vector, the "__condition" part could be discarded. In SummarizedExperiment, the "condition" column could be discarded in colData slot.
Note, regardless of data class the double underscore is a special string that is reserved for specific purposes in "spatialHeatmap", and thus should be avoided for naming feature/samples and conditions.
In the case of spatial-temporal data, there are three factors: samples, conditions, and time points. The naming scheme is slightly different and includes three options: 1) combine samples and conditions to make the composite factor "sampleCondition", then concatenate the new factor and times with double underscore in between, *i.e.* "sampleCondition__time"; 2) combine samples and times to make the composite factor "sampleTime", then concatenate the new factor and conditions with double underscore in between, *i.e.* "sampleTime__condition"; or 3) combine all three factors to make the composite factor "sampleTimeCondition" without double underscore. See the vignette for more details by running browseVignettes('spatialHeatmap') in R.

assay.na
: Applicable when data is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is NULL.

sam.factor
: The column name corresponding to samples in the colData of SummarizedExperiment. If the original column names in the assay slot already follows the scheme "sample__condition", then the colData slot is not required and accordingly this argument could be NULL.

con.factor
: The column name corresponding to conditions in the colData of SummarizedExperiment. Could be NULL if column names of in the assay slot already follows the scheme "sample__condition", or no condition is associated with the data.

ID
: A character vector of assyed items (*e.g.* genes, proteins) whose abudance values are used to color the aSVG.

sce.dimred
: A SingleCellExperiment with reduced dimentionalities such as PCA, UMAP, TSNE.

dimred	One of PCA, UMAP, TSNE, specifying which reduced dimensionality to use in co-visualization of bulk and single data.
tar.cell	Applicable in co-visualizing bulk and single cell data through manual matching. Identifiers of target cell groups to show in embedding plot, which are defined in cell.group. The default is matched and only cell groups in the matching list will have legends in the embedding plot.
tar.bulk	Applicable in co-visualizing bulk and single cell data through auto-matching (coclustering). One of the SVGBulk entries in the matching data.frame. Cells correctly assigned to this bulk tissue is highlighted, while other cells corresponding to this bulk but have false or no bulk assignments are colored black.
profile	Logical, applicable in co-visualizing bulk and single cell data through auto-matching (coclustering). If TRUE, one or multiple biological molecule (e.g. gene) identifiers need to be assigned to ID, and their abundance proifles are included in the co-visualization plot. If FALSE (default), only the bulk tissue in tar.bulk and matching cells are highlighted in the the co-visualization plot without abundance proifles.
cell.group	Applicable in co-visualizing bulk and single cell data through manual matching. A column name in colData such as cluster (auto-generated), label (user-defined). Cells are divided into clusters by cell groups in this column and these clusters are matched to bulk tissues.
tmp.path	The path of the template image in the form of raster/bitmap. The template is used to create aSVGs and can be overlaid with spatial heatmaps.
charcoal	Logical, if TRUE the template image will be turned black and white.
alpha.overlay	The opacity of top-layer spatial heatmaps if a template image is added at the bottom layer. The default is 1.
lay.shm	One of "gene", "con", or "none". If "gene", spatial heatmaps are organized by genes proteins, or metabolites, *etc.* and conditions are sorted whithin each gene. If "con", spatial heatmaps are organized by the conditions/treatments applied to experiments, and genes are sorted winthin each condition. If "none", spaital heatmaps are organized by the gene order in ID and conditions follow the order they appear in data.
ncol	An integer of the number of columns to display the spatial heatmaps, which does not include the legend plot.
col.com	A character vector of the color components used to build the color scale. The default is c('yellow', 'orange', 'red').
col.bar	One of "selected" or "all", the former uses values of ID to build the color scale while the latter uses all values from the data. The default is "selected".
sig.thr	A two-numeric vector of the signal thresholds (the range of the color bar). The first and the second element will be the minmun and maximum threshold in the color bar respectively. Signals/values above the max or below min will be assigned the same color as the max or min respectively. The default is c(NA, NA) and the min and max signals in the data will be used. If one needs to change only max or min, the other should be NA.
cores	The number of CPU cores for parallelization, relevant for aSVG files with size larger than 5M. The default is NA, and the number of used cores is 1 or 2 depending on the availability.

`bar.width` The width of color bar that ranges from 0 to 1. The default is 0.08.

`bar.title.size` A numeric of color bar title size. The default is 0.

`trans.scale` One of "log2", "exp2", "row", "column", or NULL, which means transform the data by "log2" or "2-base expoent", scale by "row" or "column", or no manipuation respectively. This argument should be used if colors across samples cannot be distinguished due to low variance or outliers.

`ft.trans` A character vector of tissue/spatial feature identifiers that will be set transparent. *E.g* c("brain", "heart"). This argument is used when target features are covered by overlapping features and the latter should be transparent.

`tis.trans` This argument is deprecated and replaced by `ft.trans`.

`lis.rematch` A list for rematching features. In each slot, the slot name is an existing feature in the data, and the slot contains a vector of features in aSVG that will be rematched to the feature in the slot name. *E.g.* `list(featureData1 = c('featureSVG1', 'featureSVG2'), featureData2 = c('featureSVG3'))`, where features `c('featureSVG1', 'featureSVG2')`, `c('featureSVG3')` in the aSVG are rematched to features `'featureData1'`, `'featureData2'` in data, respectively.

`legend.r` A numeric (between -1 and 1) to adjust the legend plot size. The default is 0.2.

`sub.title.size` A numeric of the subtitle font size of each individual spatial heatmap. The default is 11.

`sub.title.vjust` A numeric of vertical adjustment for subtitle. The default is 2.

`legend.plot` A vector of suffix(es) of aSVG file name(s) such as `c('shm1', 'shm2')`. Only aSVG(s) whose suffix(es) are assigned to this arugment will have a legend plot on the right. The default is `all` and each aSVG will have a legend plot. If NULL, no legend plot is shown.

`ft.legend` One of "identical", "all", or a character vector of tissue/spatial feature identifiers from the aSVG file. The default is "identical" and all the identical/matching tissues/spatial features between the data and aSVG file are colored in the legend plot. If "all", all tissues/spatial features in the aSVG are shown. If a vector, only the tissues/spatial features in the vector are shown.

`bar.value.size` A numeric of value size in the color bar y-axis. The default is 10.

`legend.plot.title` The title of the legend plot. The default is 'Legend'.

`legend.plot.title.size` The title size of the legend plot. The default is 11.

`legend.ncol` An integer of the total columns of keys in the legend plot. The default is NULL. If both `legend.ncol` and `legend.nrow` are used, the product of the two arguments should be equal or larger than the total number of shown spatial features.

`legend.nrow` An integer of the total rows of keys in the legend plot. The default is NULL. It is only applicable to the legend plot. If both `legend.ncol` and `legend.nrow` are used, the product of the two arguments should be equal or larger than the total number of matching spatial features.

`legend.position` the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)

legend.direction
: layout of items in legends ("horizontal" or "vertical")

legend.key.size
: A numeric of the legend key size ("npc"), applicable to the legend plot. The default is 0.02.

legend.text.size
: A numeric of the legend label size, applicable to the legend plot. The default is 12.

angle.text.key
: A value of key text angle in legend plot. The default is NULL, equivalent to 0.

position.text.key
: The position of key text in legend plot, one of "top", "right", "bottom", "left". Default is NULL, equivalent to "right".

legend.2nd
: Logical, TRUE or FALSE. If TRUE, the secondary legend is added to each spatial heatmap, which are the numeric values of each matching spatial features. The default its FALSE. Only applies to the static image.

position.2nd
: The position of the secondary legend. One of "top", "right", "bottom", "left", or a two-component numeric vector. The default is "bottom". Applies to the static image and video.

legend.nrow.2nd
: An integer of rows of the secondary legend keys. Applies to the static image and video.

legend.ncol.2nd
: An integer of columns of the secondary legend keys. Applies to the static image and video.

legend.key.size.2nd
: A numeric of legend key size. The default is 0.03. Applies to the static image and video.

legend.text.size.2nd
: A numeric of the secondary legend text size. The default is 10. Applies to the static image and video.

angle.text.key.2nd
: A value of angle of key text in the secondary legend. Default is 0. Applies to the static image and video.

position.text.key.2nd
: The position of key text in the secondary legend, one of "top", "right", "bottom", "left". Default is "right". Applies to the static image and video.

dim.lgd.pos
: The legend position in the dimensionality reduction plot. The default is bottom.

dim.lgd.nrow
: The number of legend rows in the dimensionality reduction plot. The default is 2.

dim.lgd.text.size
: The size of legend text in the dimensionality reduction plot. The default is 8.

add.feature.2nd
: Logical TRUE or FALSE. Add feature identifiers to the secondary legend or not. The default is FALSE. Applies to the static image.

label
: Logical. If TRUE, spatial features having matching samples are labeled by feature identifiers. The default is FALSE. It is useful when spatial features are labeled by similar colors.

label.size	The size of spatial feature labels in legend plot. The default is 4.
label.angle	The angle of spatial feature labels in legend plot. Default is 0.
hjust	The value to horizontally adjust positions of spatial feature labels in legend plot. Default is 0.
vjust	The value to vertically adjust positions of spatial feature labels in legend plot. Default is 0.
opacity	The transparency of colored spatial features in legend plot. Default is 1. If 0, features are totally transparent.
key	Logical. The default is TRUE and keys are added in legend plot. If label is TRUE, the keys could be removed.
line.size	The thickness of each shape outline in the aSVG is maintained in spatial heatmaps, *i.e.* the stroke widths in Inkscape. This argument is the extra thickness added to all outlines. Default is 0.2 in case stroke widths in the aSVG are 0.
line.color	A character of the shape outline color. Default is "grey70".
relative.scale	A numeric to adjust the relative sizes between multiple aSVGs. Applicable only if multiple aSVG paths is assigned to svg.path. Default is NULL and all aSVGs have the same size.
verbose	Logical, FALSE or TRUE. If TRUE the samples in data not colored in spatial heatmaps are printed to R console. Default is TRUE.
out.dir	The directory to save interactive spatial heatmaps as independent HTML files and videos. Default is NULL, and the HTML files and videos are not saved.
animation.scale	A numeric to scale the spatial heatmap size in the HTML files. The default is 1, and the height is 550px and the width is calculated according to the original aspect ratio in the aSVG file.
selfcontained	Whether to save the HTML as a single self-contained file (with external resources base64 encoded) or a file with external resources placed in an adjacent directory.
video.dim	A single character of the dimension of video frame in form of 'widthxheight', such as '1920x1080', '1280x800', '320x568', '1280x1024', '1280x720', '320x480', '480x360', '600x600', '800x600', '640x480' (default). The aspect ratio of spatial heatmaps are decided by width and height.
res	Resolution of the video in dpi.
interval	The time interval (seconds) between spatial heatmap frames in the video. Default is 1.
framerate	An integer of video framerate in frames per seconds. Default is 1. Larger values make the video smoother.
bar.width.vdo	The color bar width in video, between 0 and 1.
legend.value.vdo	Logical TRUE or FALSE. If TRUE, the numeric values of matching spatial features are added to video legend. The default is NULL.
...	additional element specifications not part of base ggplot2. In general, these should also be defined in the element tree argument.

Value

An image of spatial heatmap(s), a three-component list of the spatial heatmap(s) in `ggplot` format, a data frame of mapping between assayed samples and aSVG features, and a data frame of feature attributes.

Details

See the package vignette (`browseVignettes('spatialHeatmap')`).

Author(s)

Jianhai Zhang <jianhai.zhang@email.ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

https://www.gimp.org/tutorials/
https://inkscape.org/en/doc/tutorials/advanced/tutorial-advanced.en.html
http://www.microugly.com/inkscape-quickguide/ Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
Jeroen Ooms (2018). rsvg: Render SVG Images into PDF, PNG, PostScript, or Bitmap Arrays. R package version 1.3. https://CRAN.R-project.org/package=rsvg
R. Gentleman, V. Carey, W. Huber and F. Hahne (2017). genefilter: genefilter: methods for filtering genes from high-throughput experiments. R package version 1.58.1
Paul Murrell (2009). Importing Vector Graphics: The grImport Package for R. Journal of Statistical Software, 30(4), 1-37. URL http://www.jstatsoft.org/v30/i04/
Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. RL https://www.R-project.org/
https://github.com/ebi-gene-expression-group/anatomogram/tree/master/src/svg
Yu, G., 2020. ggplotify: Convert Plot to 'grob' or 'ggplot' Object. R package version 0.0.5.URLhttps://CRAN.R-project.org/package=ggplotify30
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
Guangchuang Yu (2020). ggplotify: Convert Plot to 'grob' or 'ggplot' Object. R package version 0.0.5. https://CRAN.R-project.org/package=ggplotify
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9 Marques A et al. (2016). Oligodendrocyte heterogeneity in the mouse juvenile and adult central nervous system. Science 352(6291), 1326-1329. Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x.

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as
## a "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.

## Set up toy data.

# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be made
# based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in this
# package and accessed below.
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

## As conventions, raw sequencing count data should be normalized, aggregated, and filtered to
## reduce noise.

# Normalize count data.
# The normalizing function "calcNormFactors" (McCarthy et al. 2012) with default settings
# is used.
df.nor.chk <- norm_data(data=df.chk, norm.fun='CNF', log2.trans=TRUE)
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', log2.trans=TRUE)
# Aggregate count data.
```

```
# Aggregate "sample__condition" replicates in toy data1.
df.aggr.chk <- aggr_rep(data=df.nor.chk, aggr='mean')
df.aggr.chk[1:3, ]
# Aggregate "sample_condition" replicates in toy data2, where "sample" is "organism_part" and
# "condition" is "age".
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='organism_part', con.factor='age',
aggr='mean')
assay(se.aggr.chk)[1:3, 1:3]
# Filter out genes with low counts and low variance. Genes with counts over 5 (log2 unit) in
# at least 1% samples (pOA), and coefficient of variance (CV) between 0.2 and 100 are retained.
# Filter toy data1.
df.fil.chk <- filter_data(data=df.aggr.chk, pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
# Filter toy data2.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='organism_part', con.factor='age',
pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)

## Spatial heatmaps.

# The target chicken aSVG is downloaded from the EBI aSVG repository
# (https://github.com/ebi-gene-expression-group/anatomogram/tree/master/src/svg) directly with
# function "return_feature". It is included in this package and accessed as below. Details on
# how this aSVG is selected are documented in function "return_feature".
svg.chk <- system.file("extdata/shinyApp/example", "gallus_gallus.svg",
package="spatialHeatmap")
# Plot spatial heatmaps on gene "ENSGALG00000019846".
# Toy data1.
spatial_hm(svg.path=svg.chk, data=df.fil.chk, ID='ENSGALG00000019846', height=0.4,
legend.r=1.9, sub.title.size=7, ncol=3)
# Save spaital heatmaps as HTML and video files by assigning "out.dir" "~/test".

if (!dir.exists('~/test')) dir.create('~/test')
spatial_hm(svg.path=svg.chk, data=df.fil.chk, ID='ENSGALG00000019846', height=0.4,
legend.r=1.9, sub.title.size=7, ncol=3, out.dir='~/test')

# Toy data2.
spatial_hm(svg.path=svg.chk, data=se.fil.chk, ID='ENSGALG00000019846', legend.r=1.9,
legend.nrow=2, sub.title.size=7, ncol=3)

# The data can also come as as a simple named vector. The following gives an example on a
# vector of 3 random values.
# Random values.
vec <- sample(1:100, 3)
# Name the vector. The last name is assumed as a random sample without a matching feature
# in aSVG.
names(vec) <- c('brain', 'heart', 'notMapped')
vec
# Plot.
spatial_hm(svg.path=svg.chk, data=vec, ID='geneX', height=0.6, legend.r=1.5, ncol=1)

# Plot spatial heatmaps on aSVGs of two Arabidopsis thaliana development stages.

# Make up a random numeric data frame.
df.test <- data.frame(matrix(sample(x=1:100, size=50, replace=TRUE), nrow=10))
```

```
colnames(df.test) <- c('shoot_totalA__condition1', 'shoot_totalA__condition2',
'shoot_totalB__condition1', 'shoot_totalB__condition2', 'notMapped')
rownames(df.test) <- paste0('gene', 1:10) # Assign row names
df.test[1:3, ]
# aSVG of development stage 1.
svg1 <- system.file("extdata/shinyApp/example", "arabidopsis.thaliana_organ_shm1.svg",
package="spatialHeatmap")
# aSVG of development stage 2.
svg2 <- system.file("extdata/shinyApp/example", "arabidopsis.thaliana_organ_shm2.svg",
package="spatialHeatmap")
# Spatial heatmaps.
spatial_hm(svg.path=c(svg1, svg2), data=df.test, ID=c('gene1'), height=0.8, legend.r=1.6,
preserve.scale=TRUE)

# Multiple development stages can also be arranged in a single aSVG image, but the
# samples, stages, and conditions should be formatted in different ways. See the vignette
# for details by running "browseVignette('spatialHeatmap')" in R.
# Overlay real images with spatial heatmaps.

# The first real image used as a template to create an aSVG.
tmp.pa1 <- system.file('extdata/shinyApp/example/maize_leaf_shm1.png',
package='spatialHeatmap')
# The first aSVG created with the first real image.
svg.pa1 <- system.file('extdata/shinyApp/example/maize_leaf_shm1.svg',
package='spatialHeatmap')
# The second real image used as a template to create an aSVG.
tmp.pa2 <- system.file('extdata/shinyApp/example/maize_leaf_shm2.png',
package='spatialHeatmap')
# The second aSVG created with the second real image.
svg.pa2 <- system.file('extdata/shinyApp/example/maize_leaf_shm2.svg',
package='spatialHeatmap')

# The data table.
dat.overlay <- read_fr(system.file('extdata/shinyApp/example/dat_overlay.txt',
package='spatialHeatmap'))

# Plot spatial heatmaps on top of real images.
spatial_hm(svg.path=c(svg.pa1, svg.pa2), data=dat.overlay, tmp.path=c(tmp.pa1, tmp.pa2),
charcoal=FALSE, ID=c('gene1'), alpha.overlay=0.5)


# Co-visualizing single cell and bulk tissues through manual matching.
# Example single cell data from mouse brain (Marques et al. (2016)).
sce.manual.pa <- system.file("extdata/shinyApp/example", "sce_manual_mouse.rds", package="spatialHeatmap")
sce.manual <- readRDS(sce.manual.pa)
# The following are simplified steps on single cell data analysis. Details are available at http://bioconductor.org

# Quality control
library(scuttle)
stats <- perCellQCMetrics(sce.manual, subsets=list(Mt=rowData(sce.manual)$featureType=='mito'), threshold=1)
sub.fields <- 'subsets_Mt_percent'
ercc <- 'ERCC' %in% altExpNames(sce.manual)
if (ercc) sub.fields <- c('altexps_ERCC_percent', sub.fields)
```

```
qc <- perCellQCFilters(stats, sub.fields=sub.fields, nmads=3)

# Discard unreliable cells.
colSums(as.matrix(qc))
sce.manual <- sce.manual[, !qc$discard]

# Normalization
library(scran); set.seed(1000)
clusters <- quickCluster(sce.manual)
sce.manual <- computeSumFactors(sce.manual, cluster=clusters)
sce.manual <- logNormCounts(sce.manual)

# Variance modelling.
df.var <- modelGeneVar(sce.manual)
top.hvgs <- getTopHVGs(df.var, prop = 0.1, n = 3000)

# Dimensionality reduction.
library(scater)
sce.manual <- denoisePCA(sce.manual, technical=df.var, subset.row=top.hvgs)
sce.manual <- runTSNE(sce.manual, dimred="PCA")
sce.manual <- runUMAP(sce.manual, dimred = "PCA")

# Clustering.
library(igraph)
snn.gr <- buildSNNGraph(sce.manual, use.dimred="PCA")
# Cell clusters.
cluster <- paste0('clus', cluster_walktrap(snn.gr)$membership)
table(cluster)
# Cell cluster/group assignments need to store in "colData" slot of "SingleCellExperiment". Cell clusters/groups pr
cdat <- colData(sce.manual)
lab.lgc <- 'label' %in% make.names(colnames(cdat))
if (lab.lgc) {
  cdat <- cbind(cluster=cluster, colData(sce.manual))
  idx <- colnames(cdat) %in% c('cluster', 'label')
 cdat <- cdat[, c(which(idx), which(!idx))]
} else cdat <- cbind(cluster=cluster, colData(sce.manual))
colnames(cdat) <- make.names(colnames(cdat))
colData(sce.manual) <- cdat; cdat[1:3, ]


plotUMAP(sce.manual, colour_by="label")
plotUMAP(sce.manual, colour_by="cluster")


# The aSVG file of mouse brain.
svg.mus <- system.file("extdata/shinyApp/example", "mus_musculus.brain.svg", package="spatialHeatmap")
# Spatial features to match with single cell clusters.
feature.df <- return_feature(svg.path=svg.mus)
feature.df$feature

# The single cells can be matched to bulk tissues according to cluster assignments in the "label" or "cluster" colum
# Matching according to cell clusters in the "label" column in "colData", which are the cell sources provided in the
unique(colData(sce.manual)$label)
```

```
# Aggregate cells by cell clusters defined in the "label" column.
sce.manual.aggr <- aggr_rep(sce.manual, assay.na='logcounts', sam.factor='label', con.factor='expVar', aggr='mea
# Manually create the matching list.
lis.match <- list(hypothalamus=c('hypothalamus'), cortex.S1=c('cerebral.cortex'))
# Co-visualization through manual mathcing: label.
shm.lis <- spatial_hm(svg.path=svg.mus, data=sce.manual.aggr, ID=c('St18'), height=0.7, legend.r=1.5, legend.key.

# Matching according to cell clusters in the "cluster" column in "colData".
unique(colData(sce.manual)$cluster)
# Aggregate cells by cell clusters defined in the "label" column.
sce.manual.aggr <- aggr_rep(sce.manual, assay.na='logcounts', sam.factor='cluster', con.factor=NULL, aggr='mean'
# Manually create the matching list.
lis.match <- list(clus1=c('hypothalamus'), clus3=c('cerebral.cortex', 'midbrain'))
# Co-visualization through manual mathcing: cluster.
shm.lis <- spatial_hm(svg.path=svg.mus, data=sce.manual.aggr, ID=c('St18'), height=0.7, legend.r=1.5, legend.key.
```

---

spd_auc_violin *Violin plot of extracted AUCs by top spd.sets*

Description

In coclustering optimization, visualize extracted AUCs by top spd.sets ranked by frequency in violin plots.

Usage

```
spd_auc_violin(
  df.lis,
  n = 5,
  ylab = "AUC",
  xlab,
  x.agl = 45,
  x.vjust = 0.6,
  nrow = 3,
  title = NULL,
  key.title = NULL,
  lgd.key.size = 0.03
)
```

Arguments

df.lis	The nested list of extracted aucs returned by auc_stat.
n	Number of top spd.set ranked by frequencies to plot.
xlab, ylab	The x and y axis labels in the violin plots.
x.agl, x.vjust	Angle and vertical position to adjust x-axis text.
nrow	The numbers of rows of all the violin plots.
title	The title of composite violin plots.

key.title	The title of legend.
lgd.key.size	The size of legend keys.

Value

An object of ggplot.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

Examples

```
# To obtain reproducible results, always start a new R session and set a fixed seed for Random Number Generator at th
set.seed(10)

# Example bulk data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Li et al 2016).
blk <- readRDS(system.file("extdata/cocluster/data", "bulk_cocluster.rds", package="spatialHeatmap"))

# Example single cell data of Arabidopsis thaliana (Arabidopsis) root for coclustering optimization (Shahan et al 20
sc10 <- readRDS(system.file("extdata/cocluster/data", "sc10_cocluster.rds", package="spatialHeatmap"))
sc11 <- readRDS(system.file("extdata/cocluster/data", "sc11_cocluster.rds", package="spatialHeatmap"))


# These example data are already pre-processed. To demonstrate the optimization process the pre-processing steps ar

# Inital filtering before normalization.
blk <- filter_data(data=blk, pOA=c(0.2, 15), CV=c(1.5, 100)); dim(blk)

fil.init <- filter_cell(lis=list(sc10=sc10, sc11=sc11), bulk=blk, gen.rm='^ATCG|^ATCG', min.cnt=1, p.in.cell=0.3,

# Normalization.
# sum.factor.
norm.fct <- norm_multi(dat.lis=fil.init, cpm=FALSE)
# sum.factor + CPM.
norm.cpm <- norm_multi(dat.lis=fil.init, cpm=TRUE)

# Secondary filtering.
# Filtering parameter sets.
df.par.fil <- data.frame(p=c(0.1, 0.2, 0.3, 0.4), A=rep(1, 4), cv1=c(0.1, 0.2, 0.3, 0.4), cv2=rep(100, 4), min.cnt=
df.par.fil

# Filtered results are saved in "opt_res".
if (!dir.exists('opt_res')) dir.create('opt_res')
fct.fil.all <- filter_iter(bulk=norm.fct$bulk, cell.lis=list(sc10=norm.fct$sc10, sc11=norm.fct$sc11), df.par.fil
```

```
cpm.fil.all <- filter_iter(bulk=norm.cpm$bulk, cell.lis=list(sc10=norm.cpm$sc10, sc11=norm.cpm$sc11), df.par.fil

# Matching table between bulk tissues and single cells.
match.pa <- system.file("extdata/cocluster/data", "match_arab_root_cocluster.txt", package="spatialHeatmap")
df.match.arab <- read.table(match.pa, header=TRUE, row.names=1, sep='\t')
df.match.arab[1:3, ]

# Optimization.
# Check parallelization guide.
coclus_opt(wk.dir='opt_res', parallel.info=TRUE, dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.

# The first-level parallel computing relies on the slurm scheduler (https://slurm.schedmd.com/documentation.html),
file.copy(system.file("extdata/cocluster", "slurm.tmpl", package="spatialHeatmap"), './slurm.tmpl')

# The first- and second-level parallelizations are set 3 and 2 respectively.
library(BiocParallel)
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# If slurm is not available, parallelize the optimization only at the second-level through 2 workers.
opt <- coclus_opt(wk.dir='opt_res', dimred=c('PCA', 'UMAP'), graph.meth=c('knn', 'snn'), sim=seq(0.2, 0.4, by=0.1

# The performaces of parameter settings are measured by AUC values in ROC curve. The following demonstrates how to vi

# Extract AUCs and other parameter settings for filtering parameter sets.
df.lis.fil <- auc_stat(wk.dir='opt_res', tar.par='filter', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.fil$df.auc.mean[1:3, ]

# Mean AUCs by each filtering settings and AUC cutoff.
mean_auc_bar(df.lis.fil[[1]], bar.width=0.07, title='Mean AUCs by filtering settings')

# All AUCs by each filtering settings and AUC cutoff.
auc_violin(df.lis=df.lis.fil, xlab='Filtering settings')

# Optimal filtering settings: fil1, fil2, fil3
df.par.fil[c(1, 2, 3), ]

# Extract AUCs and other parameter settings for normalization methods.
df.lis.norm <- auc_stat(wk.dir='opt_res', tar.par='norm', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9, 0
df.lis.norm$df.auc.mean[1:3, ]

# Mean AUCs by each normalization method and AUC cutoff.
mean_auc_bar(df.lis.norm[[1]], bar.width=0.07, title='Mean AUCs by normalization methods')

# All AUCs by each normalization method and AUC cutoff.
auc_violin(df.lis=df.lis.norm, xlab='Normalization methods')

# Optimal normalization method: fct (computeSumFactors).

# Extract AUCs and other parameter settings for graph-building methods.
df.lis.graph <- auc_stat(wk.dir='opt_res', tar.par='graph', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.graph$df.auc.mean[1:3, ]
```

```
# Mean AUCs by each graph-building method and AUC cutoff.
mean_auc_bar(df.lis.graph[[1]], bar.width=0.07, title='Mean AUCs by graph-building methods')

# All AUCs by each graph-building method and AUC cutoff.
auc_violin(df.lis=df.lis.graph, xlab='Graph-building methods')

# Optimal graph-building methods: knn (buildKNNGraph).

# Extract AUCs and other parameter settings for dimensionality reduction methods.
df.lis.dimred <- auc_stat(wk.dir='opt_res', tar.par='dimred', total.min=500, true.min=300, aucs=round(seq(0.5, 0.
df.lis.dimred$df.auc.mean[1:3, ]

# Mean AUCs by each dimensionality reduction method and AUC cutoff.
mean_auc_bar(df.lis.dimred[[1]], bar.width=0.07, title='Mean AUCs by dimensionality reduction methods')

# All AUCs by each dimensionality reduction method and AUC cutoff.
auc_violin(df.lis=df.lis.dimred, xlab='Dimensionality reduction')

# Optimal dimensionality reduction method: pca (denoisePCA).

# Extract AUCs and other parameter settings for spd.sets.
df.lis.spd <- auc_stat(wk.dir='opt_res', tar.par='spd.set', total.min=500, true.min=300, aucs=round(seq(0.5, 0.9,
df.lis.spd$auc0.5$df.frq[1:3, ]

# All AUCs of top spd.sets ranked by frequency.
spd_auc_violin(df.lis=df.lis.spd, n=5, xlab='spd.sets', x.vjust=0.6)
```

`submatrix` *Subset Target Assayed Items and Their Nearest Neighbors*

Description

Given a vector of target assayed items (gene, protein, metabolite, *etc*), this function selects nearest neighbors for every target item independently, which share most similar abundance profiles with the targets. The selection is based on correlation or distance matrix computed by `cor` or `dist` from the "stats" package respectively. One of three alternative arguments `p`, `n`, `v` sets a cutoff for the selection.

Usage

```
submatrix(
  data,
  assay.na = NULL,
  ann = NULL,
  ID,
  p = 0.3,
  n = NULL,
  v = NULL,
```

```
  fun = "cor",
  cor.absolute = FALSE,
  arg.cor = list(method = "pearson"),
  arg.dist = list(method = "euclidean"),
  dir = NULL
)
```

Arguments

data	A "data.frame", "SummarizedExperiment", or "SingleCellExperiment" object returned by the function filter_data, where the columns and rows of the data matrix are samples/conditions and assayed items (*e.g.* genes, proteins) respectively. Since this function builds on coexpression analysis, variables of sample/condition should be at least 5. Otherwise, the results are not reliable.
assay.na	Applicable when data is "SummarizedExperiment" or "SingleCellExperiment", where multiple assays could be stored. The name of target assay to use. The default is NULL.
ann	Applicable when data is "SummarizedExperiment" or "SingleCellExperiment". The column name corresponding to row item annotation in the rowData slot. Default is NULL.
ID	A vector of target item identifiers.
p	The proportion of top items with most similar expression profiles with the target items. Only items within this proportion are returned. Default is 0.3. It applies to each target item independently, and selected items of each target are returned together.
n	An integer of top items with most similar expression profiles with the target items. Only items within this number are returned. Default is NULL. It applies to each target independently, and selected items of each target are returned together.
v	A numeric of correlation (-1 to 1) or distance (>=0) threshold to select items sharing the most similar expression profiles with the target items. If fun='cor', only items with correlation coefficient larger than v are returned. If fun='dist', only items with distance less than v are returned. Default is NULL. It applies to each target independently, and selected items of each target are returned together.
fun	The function to calculate similarity/distance measure, 'cor' or 'dist', corresponding to cor or dist from the "stats" package respectively. Default is 'cor'.
cor.absolute	Logical, TRUE or FALSE. Use absolute values or not. Only applies to fun='cor'. Default is FALSE, meaning the correlation coefficient preserves the negative sign when selecting items.
arg.cor	A list of arguments passed to cor in the "stats" package. Default is list(method="pearson").
arg.dist	A list of arguments passed to dist in the "stats" package. Default is list(method="euclidean").

dir — The directory where the folder "customComputedData" is created automatically to save the subsetted matrix as a TSV-format file "sub_matrix.txt", which is ready to upload to the Shiny app launched by shiny_shm. In the "sub_matrix.txt", the rows are assayed items and column names are in the syntax "sample__condition". This argument should be the same with the dir in adj_mod so that the files "adj.txt" and "mod.txt" generated by adj_mod are saved in the same folder "customComputedData". The default is NULL and no file is saved. This argument is used only when using the "customComputedData" in the Shiny app.

Value

The subsetted matrix of target items and their nearest neighbors.

Author(s)

Jianhai Zhang <zhang.jianhai@hotmail.com; jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 doi:10.1186/1471-2105-9-559
Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. Journal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
Peter Langfelder, Bin Zhang and with contributions from Steve Horvath (2016). dynamicTreeCut: Methods for Detection of Clusters in Hierarchical Clustering Dendrograms. R package version 1.63-1. https://CRAN.R-project.org/package=dynamicTreeCut
Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. “Gene Expression Across Mammalian Organ Development.” Nature 571 (7766): 505–9

Examples

```
## In the following examples, the 2 toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, they are
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769". Toy data1 is used as
## a "data frame" input to exemplify data of simple samples/conditions, while toy data2 as
## "SummarizedExperiment" to illustrate data involving complex samples/conditions.

## Set up toy data.
```

```
# Access toy data1.
cnt.chk.simple <- system.file('extdata/shinyApp/example/count_chicken_simple.txt',
package='spatialHeatmap')
df.chk <- read.table(cnt.chk.simple, header=TRUE, row.names=1, sep='\t', check.names=FALSE)
# Columns follow the namig scheme "sample__condition", where "sample" and "condition" stands
# for organs and time points respectively.
df.chk[1:3, ]

# A column of gene annotation can be appended to the data frame, but is not required.
ann <- paste0('ann', seq_len(nrow(df.chk))); ann[1:3]
df.chk <- cbind(df.chk, ann=ann)
df.chk[1:3, ]

# Access toy data2.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data2. It should be made
# based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in this
# package and accessed below.
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data2 corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data2 in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene annotation, but not required.
rowData(se.chk) <- DataFrame(ann=ann)

## As conventions, raw sequencing count data should be normalized, aggregated, and filtered to
## reduce noise.

# Normalize count data.
# The normalizing function "calcNormFactors" (McCarthy et al. 2012) with default settings
# is used.
df.nor.chk <- norm_data(data=df.chk, norm.fun='CNF', log2.trans=TRUE)
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', log2.trans=TRUE)
# Aggregate count data.
# Aggregate "sample__condition" replicates in toy data1.
df.aggr.chk <- aggr_rep(data=df.nor.chk, aggr='mean')
df.aggr.chk[1:3, ]
# Aggregate "sample_condition" replicates in toy data2, where "sample" is "organism_part" and
# "condition" is "age".
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='organism_part', con.factor='age',
aggr='mean')
assay(se.aggr.chk)[1:3, 1:3]
# Filter out genes with low counts and low variance. Genes with counts over 5 (log2 unit) in at
# least 1% samples (pOA), and coefficient of variance (CV) between 0.2 and 100 are retained.
# Filter toy data1.
```

```
df.fil.chk <- filter_data(data=df.aggr.chk, pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)
# Filter toy data2.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='organism_part', con.factor='age',
pOA=c(0.01, 5), CV=c(0.2, 100), dir=NULL)

## Select nearest neighbors for target genes 'ENSGALG00000019846' and 'ENSGALG00000000112',
## which are usually genes visualized in spatial heatmaps.
# Toy data1.
df.sub.mat <- submatrix(data=df.fil.chk, ID=c('ENSGALG00000019846', 'ENSGALG00000000112'),
p=0.1)
# Toy data2.
se.sub.mat <- submatrix(data=se.fil.chk, ann='ann', ID=c('ENSGALG00000019846',
'ENSGALG00000000112'), p=0.1)

# In the following, "df.sub.mat" and "se.sub.mat" is used in the same way, so only
# "se.sub.mat" illustrated.

# The subsetted matrix is partially shown below.
se.sub.mat[c('ENSGALG00000019846', 'ENSGALG00000000112'), c(1:2, 63)]
```

---

sub_asg *Subset the bulk-cell assignments*

Description

Subset the bulk-cell assignments according to a threshold, which is a similarity value between bulk and cells.

Usage

```
sub_asg(
  res.lis,
  thr = 0,
  df.desired.bulk = NULL,
  df.match = NULL,
  true.only = TRUE
)
```

Arguments

res.lis	The result list of coclustering, which is the output of `tests` and comprises three slots `sce`, `roc.obj`, `df.roc`.
thr	The threshold for subsetting bulk-cell assignments, which is a similarity value (Pearson's or Spearman's correlation coefficient) between bulk and cells. Only bulk-cell assignments with similarity values above the thresold would remain. The default is 0.

df.desired.bulk
: A "data.frame" of desired bulk for some cells. The cells could be specified by providing x-y axis ranges in an embedding plot ("UMAP", "PCA", "TSNE") returned by plot_dim. E.g. df.desired.bulk <- data.frame(x.min=c(4, -6), x.max=c(5, -5), y.min=c(-2.5, 2), y.max=c(-2, 2.5), desiredSVGBulk=c('CORT', 'STELE'), dimred='UMAP'), where columns x.min, x.max, y.min, y.max, desiredSVGBulk, dimred are required. In this example, cells located in 4 <= x <= 5 and -2.5 <= y <= -2 in the "UMAP" plot are assigned "STELE", and cells located in -6 <= x <= -5 and 2 <= y <= 2.5 in the "UMAP" plot are assigned "CORT".
Alternatively, the "data.frame" could be downloaded from the Shiny app launched by desired_bulk_shiny.
df.desired.bulk and df.match together are used to tailor the co-clustering results. That is to say additional true bulk-cell assignments are created and included in the final assignments. If these assignments conflict with the co-colustering results the latter would be overwritten.

df.match
: The ground-truth matching between cells and bulk. See the example of data(df.match).

true.only
: Logical. If TRUE, only the true assignments are returned in the subset SingleCellExperiment. This argument affects the values for plottting SHMs.

Value

A SingleCellExperiment of remaining bulk-cell assignments.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Amezquita R, Lun A, Becht E, Carey V, Carpp L, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pages H, Smith M, Huber W, Morgan M, Gottardo R, Hicks S (2020). "Orchestrating single-cell analysis with Bioconductor." Nature Methods, 17, 137–145. https://www.nature.com/articles/s41592-019-0654-x Morgan M, Obenchain V, Hester J, Pagès H (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24.0, https://bioconductor.org/packages/SummarizedExperiment.

Examples

```
# To obtain reproducible results, always start a new R session and set a fixed seed for Random Number Generator at th
set.seed(10)

# Example bulk data of mouse brain for coclustering (Vacher et al 2021).
blk.mus.pa <- system.file("extdata/shinyApp/example", "bulk_mouse_cocluster.txt", package="spatialHeatmap")
blk.mus <- as.matrix(read.table(blk.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
blk.mus[1:3, 1:5]

# Example single cell data for coclustering (Ortiz et al 2020).
sc.mus.pa <- system.file("extdata/shinyApp/example", "cell_mouse_cocluster.txt", package="spatialHeatmap")
sc.mus <- as.matrix(read.table(sc.mus.pa, header=TRUE, row.names=1, sep='\t', check.names=FALSE))
sc.mus[1:3, 1:5]
```

```
# Initial filtering.
blk.mus <- filter_data(data=blk.mus, sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 5), CV=c(0.2, 100), dir=NULL)
dim(blk.mus)
mus.lis <- filter_cell(lis=list(sc.mus=sc.mus), bulk=blk.mus, gen.rm=NULL, min.cnt=1, p.in.cell=0.5, p.in.gen=0.1

# Normalization: bulk and single cell are combined and normalized, then separated.
mus.lis.nor <- norm_multi(dat.lis=mus.lis, cpm=FALSE)

# Secondary filtering.
library(SingleCellExperiment)
blk.mus.fil <- filter_data(data=logcounts(mus.lis.nor$bulk), sam.factor=NULL, con.factor=NULL, pOA=c(0.1, 0.5), C
dim(blk.mus.fil)

mus.lis.fil <- filter_cell(lis=list(sc.mus=logcounts(mus.lis.nor$sc.mus)), bulk=blk.mus.fil, gen.rm=NULL, min.cn

# The aSVG file of mouse brain.
svg.mus <- system.file("extdata/shinyApp/example", "mus_musculus.brain.svg", package="spatialHeatmap")
# Spatial features.
feature.df <- return_feature(svg.path=svg.mus)

# Matching table indicating true bulk tissues of each cell type and corresponding SVG bulk (spatial feature).
df.match.mus.pa <- system.file("extdata/shinyApp/example", "match_mouse_brain_cocluster.txt", package="spatialHe
df.match <- read.table(df.match.mus.pa, header=TRUE, row.names=1, sep='\t')
df.match

# The SVG bulk tissues are in the aSVG file.
df.match$SVGBulk %in% feature.df$feature

# Cluster single cells.
clus.sc <- cluster_cell(data=mus.lis.fil$sc.mus, min.dim=10, max.dim=50, graph.meth='knn', dimred='PCA')
# Cluster labels are stored in "label" column in "colData".
colData(clus.sc)[1:3, ]

# Refine cell clusters.
cell.refined <- refine_cluster(clus.sc, sim=0.2, sim.p=0.8, sim.meth='spearman')

# Include matching information in "colData".
cell.refined <- true_bulk(cell.refined, df.match)
colData(cell.refined)[1:3, ]

# Cocluster bulk and single cells.
roc.lis <- coclus_roc(bulk=mus.lis.fil$bulk, cell.refined=cell.refined, df.match=df.match, min.dim=12, max.dim=5

# The colustering results. "predictor" is the similarity between bulk and cells within a co-cluster. "index" is the
roc.lis$df.roc[1:3, ]
# ROC curve created according to "roc.lis$df.roc".
plot(roc.lis$roc.obj, print.auc=TRUE)
# Incorporate "cell.refined" in "roc.lis" for downstream use in co-visualization.
res.lis <- c(list(cell.refined=cell.refined), roc.lis)

# The processes of clustering single cells, refining cell clusters, and coclustering bulk and single cells can be pe
library(BiocParallel)
```

```
res.lis <- cocluster(bulk=mus.lis.fil$bulk, cell=mus.lis.fil$sc.mus, df.match=df.match, df.para=NULL, sim=0.2, si
res.lis <- res.lis[[1]]

names(res.lis)

# "cocluster" accepts multiple combinations of parameter settings provided in a data frame, and coclustering on thes
# Multiple combinations of parameter settings. If some parameters are not specified in this table such as "graph.met
df.par <- data.frame(sim=c(0.2, 0.3), sim.p=c(0.8, 0.7), dim=c(12, 13))

# The computation is parallelized on 2 cpu cores by "multi.core.par".
res.multi <- cocluster(bulk=mus.lis.fil$bulk, cell=mus.lis.fil$sc.mus, df.match=df.match, df.para=df.par, sc.dim

# The results of auto-matching through coclustering can be tailored through "Lasso Select" on the convenience Shiny
df.desired.bulk <- NULL
# Example of desired bulk downloaded from convenience Shiny app.
desired.blk.pa <- system.file("extdata/shinyApp/example", "selected_cells_with_desired_bulk.txt", package="spati
df.desired.bulk <- read.table(desired.blk.pa, header=TRUE, row.names=1, sep='\t')
df.desired.bulk[1:3, ]

# Desired bulk manually definded by x-y coordinates ranges.
plot_dim(res.lis$cell.refined, dim='PCA', color.by='cell', x.break=seq(-10, 10, 2), y.break=seq(-10, 10, 2))

df.desired.bulk <- data.frame(x.min=c(2, 6), x.max=c(4, 10), y.min=c(6, 8), y.max=c(8, 10), desiredSVGBulk=c('cere
df.desired.bulk

# Extract cells with true bulk assignments. If "df.desired.bulk" is provided, the corrresponding assignments are in
sce.lis <- sub_asg(res.lis=res.lis, df.desired.bulk=df.desired.bulk, df.match=df.match, true.only=TRUE)
```

---

sub_data *Subset Target Data for Spatial Enrichment*

Description

This function subsets the target spatial features (*e.g.* cells, tissues, organs) and factors (*e.g.* experimental treatments, time points) for the subsequent spatial enrichment.

Usage

```
sub_data(
  data,
  feature,
  features = NULL,
  factor,
  factors = NULL,
  com.by = "feature",
  target = NULL
)
```

Arguments

data	A SummarizedExperiment object. The colData slot is required to contain at least two columns of "features" and "factors" respectively. The rowData slot can optionally contain a column of discriptions of each gene and the column name should be metadata.
feature	The column name of "features" in the colData slot.
features	A vector of at least two selected features for spatial enrichment, which come from the feature column. The default is NULL and the first two features will be selected. If all, then all features will be selected.
factor	The column name of "factors" in the colData slot.
factors	A vector of at least two selected factors for spatial enrichment, which come from the factor column. The default is NULL and the first two factors will be selected. If all, then all factors will be selected.
com.by	One of feature, factor, feature.factor. If feature, pairwise comparisons will be perfomed between the selected features and the factors will be treated as replicates. If factor, pairwise comparisons will be perfomed between the selected factors and the features will be treated as replicates. If feature.factor, the selected features and factors will be concatenated by __ and pairwise comparisons will be perfomed between the "feature__factor" entities. The default is feature. The corresponding column will be moved to the first in the colData slot and be recognized in the spatial enrichment process.
target	A single-component vector of the target for spatial enrichment. If com.by='feature', the target will be one of the entries in features. If com.by='factor', the target will be one of the entries in factors. If com.by='feature.factor', the target will be one of the concatenated features and factors. *E.g.* features=c('brain', 'kidney'), factors=c('control', 'drug'), the target could be one of c('brain__control', 'brain__drug', 'kidney__control', 'kidney__drug'). The default is NULL, and the first entity in features is selected, since the default com.by is feature. A target column will be included in the colData slot and will be recognized in spatial enrichment.

Value

A subsetted SummarizedExperiment object.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505–9
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Martin Morgan, Valerie Obenchain, Jim Hester and Hervé Pagès (2018). SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1

Examples

```
## In the following examples, the toy data come from an RNA-seq analysis on development of 7
## chicken organs under 9 time points (Cardoso-Moreira et al. 2019). For conveninece, it is
## included in this package. The complete raw count data are downloaded using the R package
## ExpressionAtlas (Keays 2019) with the accession number "E-MTAB-6769".

## Set up toy data.

# Access toy data.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]

# A targets file describing samples and conditions is required for toy data. It should be made
# based on the experiment design, which is accessible through the accession number
# "E-MTAB-6769" in the R package ExpressionAtlas. An example targets file is included in this
# package and accessed below.
# Access the count table.
cnt.chk <- system.file('extdata/shinyApp/example/count_chicken.txt', package='spatialHeatmap')
count.chk <- read.table(cnt.chk, header=TRUE, row.names=1, sep='\t')
count.chk[1:3, 1:5]
# Access the example targets file.
tar.chk <- system.file('extdata/shinyApp/example/target_chicken.txt', package='spatialHeatmap')
target.chk <- read.table(tar.chk, header=TRUE, row.names=1, sep='\t')
# Every column in toy data corresponds with a row in targets file.
target.chk[1:5, ]
# Store toy data in "SummarizedExperiment".
library(SummarizedExperiment)
se.chk <- SummarizedExperiment(assay=count.chk, colData=target.chk)
# The "rowData" slot can store a data frame of gene metadata, but not required. Only the
# column named "metadata" will be recognized.
# Pseudo row metadata.
metadata <- paste0('meta', seq_len(nrow(count.chk))); metadata[1:3]
rowData(se.chk) <- DataFrame(metadata=metadata)

## As conventions, raw sequencing count data should be normalized and filtered to
## reduce noise. Since normalization will be performed in spatial enrichment, only filtering
## is required before subsetting the data.

# Filter out genes with low counts and low variance. Genes with counts over 5 in
# at least 10% samples (pOA), and coefficient of variance (CV) between 3.5 and 100 are
# retained.
se.fil.chk <- filter_data(data=se.chk, sam.factor='organism_part', con.factor='age',
pOA=c(0.1, 5), CV=c(3.5, 100), dir=NULL)
# Subset the data.
data.sub <- sub_data(data=se.fil.chk, feature='organism_part', features=c('brain', 'heart',
'kidney'), factor='age', factors=c('day10', 'day12'), com.by='feature', target='brain')
```

---

true_bulk *Assign true bulk to cells in* colData *slot.*

Description

In co-clustering, assign true bulk to cells in colData slot.

Usage

```
true_bulk(sce, df.match)
```

Arguments

sce	A SingleCellExperiment of clustered single cell data.
df.match	The matching table between cells and true bulk.

Value

A SingleCellExperiment object.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Morgan M, Obenchain V, Hester J, Pagès H (2021). SummarizedExperiment: SummarizedExperiment container. R package version 1.24. 0, https://bioconductor.org/packages/SummarizedExperiment.

Examples

```
# Matching table.
match.mus.brain.pa <- system.file("extdata/shinyApp/example", "match_mouse_brain_cocluster.txt", package="spatia
df.match.mus.brain <- read.table(match.mus.brain.pa, header=TRUE, row.names=1, sep='\t')
df.match.mus.brain

# Create random data matrix.
df.random <- matrix(rexp(30), nrow=5)
dimnames(df.random) <- list(paste0('gene', seq_len(nrow(df.random))), c('cere', 'cere', 'hipp', 'hipp', 'corti.su

library(SingleCellExperiment); library(S4Vectors)
cell.refined <- SingleCellExperiment(assays=list(logcounts=df.random), colData=DataFrame(cell=colnames(df.rando

cell.refined <- true_bulk(cell.refined, df.match.mus.brain)
colData(cell.refined)

# See detailed example in the "cocluster" function by running "?cocluster".
```

update_feature *Update aSVG Spatial Features*

Description

Successful spatial heatmap plotting requires the aSVG features of interest have matching samples (cells, tissues, *etc*) in the data. If this requirement is not fulfiled, either the sample identifiers in the data or the spatial feature identifiers in the aSVG should be changed. This function is designed to replace existing feature identifiers, stroke (outline) widths, and/or feature colors in aSVG files with user-provided entries.

Usage

```
update_feature(df.new, dir)
```

Arguments

df.new	The custom feature identifiers, stroke (outline) widths, and/or feature colors, should be included in the data frame returned by return_feature as independent columns, and the corresponding column names should be "featureNew", "strokeNew", and "colorNew" respectively in order to be recognized. To color the corresponding features, the identifiers in "featureNew" should be the same with matching sample identifiers. The numeric values in "strokeNew" would be the outline widths of corresponding features. The colors in "colorNew" would be the default colors for highlighting target features in the legend plot.
dir	The directory path where the aSVG files to update. It should be the same with dir in return_feature.

Value

Nothing is returned. The aSVG files of interest in dir are updated with provided attributes, and are ready to use in function spatial_hm.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

Hadley Wickham, Jim Hester and Jeroen Ooms (2019). xml2: Parse XML. R package version 1.2.2. https://CRAN.R-project.org/package=xml2
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. "Gene Expression Across Mammalian Organ Development." Nature 571 (7766): 505-9
Gregory R. Warnes, Ben Bolker, Lodewijk Bonebakker, Robert Gentleman, Wolfgang Huber, Andy Liaw, Thomas Lumley, Martin Maechler, Arni Magnusson, Steffen Moeller, Marc Schwartz and

Bill Venables (2020). gplots: Various R Programming Tools for Plotting Data. R package version 3.0.3. https://CRAN.R-project.org/package=gplots

Examples

```
# The following shows how to download a chicken aSVG containing spatial features of 'brain'
# and 'heart' from the EBI aSVG repository directly
# (https://github.com/ebi-gene-expression-group/anatomogram/tree/master/src/svg). An empty
# directory is recommended so as to avoid overwriting existing SVG files with the same names.
# Here "~/test" is used.

# Remote aSVG repos.
data(aSVG.remote.repo)
tmp.dir <- normalizePath(tempdir(check=TRUE), winslash="/", mustWork=FALSE)
tmp.dir.ebi <- paste0(tmp.dir, '/ebi.zip')
tmp.dir.shm <- paste0(tmp.dir, '/shm.zip')

# Download the remote aSVG repos as zip files. According to Bioconductor's
# requirements, downloadings are not allowed inside functions, so the repos are
# downloaded before calling "return_feature".
download.file(aSVG.remote.repo$ebi, tmp.dir.ebi)
download.file(aSVG.remote.repo$shm, tmp.dir.shm)
remote <- list(tmp.dir.ebi, tmp.dir.shm)
# Make an empty directory "~/test" if not exist.
if (!dir.exists('~/test')) dir.create('~/test')
# Query the remote aSVG repos.
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir='~/test',
match.only=TRUE, remote=remote)
feature.df

# New features, stroke widths, colors.
ft.new <- c('BRAIN', 'HEART')
stroke.new <- c(0.05, 0.1)
col.new <- c('green', 'red')
# Include new features, stroke widths, colors to the feature data frame.
feature.df.new <- cbind(featureNew=ft.new, strokeNew=stroke.new, colorNew=col.new, feature.df)
feature.df.new

# Update features.
update_feature(df.new=feature.df.new, dir='~/test')
```

`write_hdf5` *Construct Database for the Shiny App*

Description

This is a convenience function for constructing the database backend in the Shiny app (shiny_shm). The data to store in the database should be in the class of "data.frame" or "SummarizedExperiment" and should be formatted according to the conventions in the "data" argument of spatial_hm. After

formatted, all these data should be arranged in a list and each data slot should have a unique name such as "expr_arab", "expr_chicken", *etc.*.
In addition, a pairing data frame describing the matching relationship between the data and aSVG files must also be included in the list with the exclusive slot name "df_pair". This data frame should contain at least three columns: name, data, aSVG. The name column includes concise description of each data-aSVG pair, and entries in this column will be listed under "Step 1: data sets" on the Shiny app. The data column contains slot names of all data in the list ("expr_arab", "expr_chicken", *etc.*), and the aSVG column includes the aSVG file names corresponding to each data respectively such as "gallus_gallus.svg", *etc.* If one data is related to multiple aSVG files (*e.g.* multiple development stages), these aSVGs should be concatenated by comma, space, or semicolon, *e.g.* "arabidopsis.thaliana_organ_shm1.svg;arabidopsis.thaliana_organ_shm2.svg". Inclusion of other columns providing metadata of the data and aSVGs are optional, which is up to the users.
After calling this function, all the data including "df_pair" in the list are saved into independent DHF5 databases, and all the DHF5 databases are finally compressed in the file "data_shm.tar". Accordingly, all the corresponding aSVG files listed in the "df_pair" should be compressed in another "tar" file such as "aSVG.tar". If the directory path containing the aSVG files are assigned to `svg.dir`, all the SVG files in the diretory are compressed in "aSVGs.tar" automatically. The two tar files compose the database in the Shiny app and should be placed in the "example" folder in the app or uploaded on the user interface.

Usage

```
write_hdf5(
  dat.lis,
  dir = "./data_shm",
  replace = FALSE,
  chunkdim = NULL,
  level = NULL,
  verbose = FALSE,
  svg.dir = NULL
)
```

Arguments

`dat.lis`	A list of data of class "data.frame" or "SummarizedExperiment", where every data should have a unique slot name such as "expr_arab", "expr_chicken", *etc.*. In addition to the data, a pairing data frame describing pairing between the data and aSVG files must be included under the exclusive slot name "df_pair". This data frame has three required columns: the "name" column includes concise names of the data-aSVG pair, the "data" column contains all slot names of the data ("expr_arab", "expr_chicken", *etc.*) and the "aSVG" column contains the aSVG file names corresponding to each data. If one data is related to multiple aSVG files (*e.g.* multiple development stages), these aSVGs should be concatenated by comma, space, or semicolon, *e.g.* "arabidopsis.thaliana_organ_shm1.svg;arabidopsis.thaliana_organ_shm2.svg". The metadata of data and aSVGs could be optionally included in extra columns.
`dir`	The directory path to save the "data_shm.tar" file. Default is `./data_shm`.

replace	If data with the same slot names in dat.lis are already saved in dir, should the dir be emptied? Default is FALSE. If TRUE, the existing content in dir will be lost.
chunkdim	The dimensions of the chunks and the compression level to use for writing the assay data to disk. Passed to the internal calls to writeHDF5Array. See ?writeHDF5Array for more information.
level	The dimensions of the chunks and the compression level to use for writing the assay data to disk. Passed to the internal calls to writeHDF5Array. See ?writeHDF5Array for more information.
verbose	Set to TRUE to make the function display progress. In the case of saveHDF5SummarizedExperiment(), verbose is set to NA by default, in which case verbosity is controlled by DelayedArray:::get_verbose_block_processing(). Setting verbose to TRUE or FALSE overrides this.
svg.dir	The directory path of aSVG files listed in "df_pair". If provded, all SVG files in the directory are compressed in "aSVGs.tar" and saved in dir. Default is NULL, which requires users to compress the aSVGs in a tar file.

Value

A file of "data_shm.tar" is save in dir. If svg.dir is assigned a valid value, all relevant SVG files are compressed in "aSVGs.tar" in dir.

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
Dr. Thomas Girke <thomas.girke@ucr.edu>

References

SummarizedExperiment: SummarizedExperiment container. R package version 1.10.1
R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
Hervé Pagès (2020). HDF5Array: HDF5 backend for DelayedArray objects. R package version 1.16.1.
Mustroph, Angelika, M Eugenia Zanetti, Charles J H Jang, Hans E Holtan, Peter P Repetti, David W Galbraith, Thomas Girke, and Julia Bailey-Serres. 2009. "Profiling Translatomes of Discrete Cell Populations Resolves Altered Cellular Priorities During Hypoxia in Arabidopsis." Proc Natl Acad Sci U S A 106 (44): 18843–8
Davis, Sean, and Paul Meltzer. 2007. "GEOquery: A Bridge Between the Gene Expression Omnibus (GEO) and BioConductor." Bioinformatics 14: 1846–7
Gautier, Laurent, Leslie Cope, Benjamin M. Bolstad, and Rafael A. Irizarry. 2004. "Affy—analysis of Affymetrix GeneChip Data at the Probe Level." Bioinformatics 20 (3). Oxford, UK: Oxford University Press: 307–15. doi:10.1093/bioinformatics/btg405
Keays, Maria. 2019. ExpressionAtlas: Download Datasets from EMBL-EBI Expression Atlas
Huber, W., V. J. Carey, R. Gentleman, S. An ders, M. Carlson, B. S. Carvalho, H. C. Bravo, et al.

2015. “Orchestrating High-Throughput Genomic Analysis Wit H Bioconductor.” Nature Methods 12 (2): 115–21. http://www.nature.com/nmeth/journal/v12/n2/full/nmeth.3252.html
Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2." Genome Biology 15 (12): 550. doi:10.1186/s13059-014-0550-8
McCarthy, Davis J., Chen, Yunshun, Smyth, and Gordon K. 2012. "Differential Expression Analysis of Multifactor RNA-Seq Experiments with Respect to Biological Variation." Nucleic Acids Research 40 (10): 4288–97
Cardoso-Moreira, Margarida, Jean Halbert, Delphine Valloton, Britta Velten, Chunyan Chen, Yi Shao, Angélica Liechti, et al. 2019. “Gene Expression Across Mammalian Organ Development.” Nature 571 (7766): 505–9

Examples

```
## The examples below demonstrate 1) how to dump Expression Atlas data set into the Shiny database;
## 2) how to dump GEO data set into the Shiny database; 3) how to include aSVGs of multiple
## development stages; 4) how to read the database; 5) how to create customized Shiny app with
## the database.

# 1. Dump data from Expression Atlas into "data_shm.tar" using ExpressionAtlas package (Keays 2019).

# The chicken data derived from an RNA-seq analysis on developments of 7 chicken organs under 9
# time points (Cardoso-Moreira et al. 2019) is chosen as example.
# The following searches the Expression Atlas for expression data from ‘heart’ and ‘gallus’.
library(ExpressionAtlas)
cache.pa <- '~/.cache/shm' # The path of cache.
all.chk <- read_cache(cache.pa, 'all.chk') # Retrieve data from cache.
if (is.null(all.chk)) { # Save downloaded data to cache if it is not cached.
 all.chk <- searchAtlasExperiments(properties="heart", species="gallus")
 save_cache(dir=cache.pa, overwrite=TRUE, all.chk)
}

all.chk[3, ]
rse.chk <- read_cache(cache.pa, 'rse.chk') # Read data from cache.
if (is.null(rse.chk)) { # Save downloaded data to cache if it is not cached.
  rse.chk <- getAtlasData('E-MTAB-6769')[[1]][[1]]
  save_cache(dir=cache.pa, overwrite=TRUE, rse.chk)
}
# The downloaded data is stored in "SummarizedExperiment" by default (SE, M. Morgan et al. 2018).
# The experiment design is described in the "colData" slot. The following returns first three rows.
colData(rse.chk)[1:3, ]
# In the "colData" slot, it is required to define the "sample" and "condition" columns respectively.
# Both "sample" and "condition" are general terms. The former refers to entities where the numeric
# data are measured such as cell organelles, tissues, organs, ect. while the latter denotes
# experimental treatments such as drug dosages, gender, trains, time series, PH values, ect. In the
# downloaded data, the two columns are not explicitly defined, so "organism_part" and "age" are
# selected and renamed as "sample" and "condition" respectively.
colnames(colData(rse.chk))[c(6, 8)] <- c('condition', 'sample'); colnames(colData(rse.chk))
# The raw RNA-Seq count are preprocessed with the following steps: (1) normalization,
# (2) aggregation of replicates, and (3) filtering of reliable expression data. The details of
# these steps are explained in the pacakge vignette.
```

```
 browseVignettes('spatialHeatmap')
se.nor.chk <- norm_data(data=rse.chk, norm.fun='ESF', log2.trans=TRUE) # Normalization
se.aggr.chk <- aggr_rep(data=se.nor.chk, sam.factor='sample', con.factor='condition',
aggr='mean') # Replicate agggregation using mean
# Genes are filtered out if not meet these criteria: expression values are at least 5 in at least
# 1% of all samples, coeffient of variance is between 0.6 and 100.
se.fil.chk <- filter_data(data=se.aggr.chk, sam.factor='sample', con.factor='condition',
pOA=c(0.01, 5), CV=c(0.6, 100), dir=NULL)
# The aSVG file corresponding with the data is pre-packaged and copied to a temporary directory.
dir.svg <- paste0(tempdir(check=TRUE), '/svg_shm') # Temporary directory.
if (!dir.exists(dir.svg)) dir.create(dir.svg)
# Path of the aSVG file.
svg.chk <- system.file("extdata/shinyApp/example", 'gallus_gallus.svg', package="spatialHeatmap")
file.copy(svg.chk, dir.svg, overwrite=TRUE) # Copy the aSVG file.

# 2. Dump data from GEO into "data_shm.tar" using GEOquery package (S. Davis and Meltzer 2007).

# The Arabidopsis thaliana (Arabidopsis) data from an microarray assay of hypoxia treatment on
# Arabidopsis root and shoot cell types (Mustroph et al. 2009) is selected as example.
# The data set is downloaded with the accession number "GSE14502". It is stored in ExpressionSet
# container (W. Huber et al. 2015) by default, and then converted to a SummarizedExperiment object.
library(GEOquery)
gset <- read_cache(cache.pa, 'gset') # Retrieve data from cache.
if (is.null(gset)) { # Save downloaded data to cache if it is not cached.
  gset <- getGEO("GSE14502", GSEMatrix=TRUE, getGPL=TRUE)[[1]]
  save_cache(dir=cache.pa, overwrite=TRUE, gset)
}
se.sh <- as(gset, "SummarizedExperiment") # Converted to SummarizedExperiment
# The gene symbol identifiers are extracted from the rowData component to be used as row names.
rownames(se.sh) <- make.names(rowData(se.sh)[, 'Gene.Symbol'])
# A slice of the experimental design in colData slot is shown. Both the samples and conditions
# are contained in the "title" column. The samples are indicated by promoters: pGL2 (root
# atrichoblast epidermis), pCO2 (root cortex meristematic zone), pSCR (root endodermis),
# pWOL (root vasculature), etc., and conditions are control and hypoxia.
colData(se.sh)[60:63, 1:4]
# Since the samples and conditions need to be listed in two independent columns, like the the
# chicken data above, a targets file is recommended to separate samples and conditions. The main
# reason to choose this Arabidopdis data is to illusrate the usage of targets file when necessary.
# A pre-packaged targets file is accessed and partially shown below.
sh.tar <- system.file('extdata/shinyApp/example/target_arab.txt', package='spatialHeatmap')
target.sh <- read_fr(sh.tar); target.sh[60:63, ]
# Load custom the targets file into colData slot.
colData(se.sh) <- DataFrame(target.sh)
# This data set was already normalized with the RMA algorithm (Gautier et al. 2004). Thus, the
# pre-processing steps are restricted to aggregation of replicates and filtering of reliably
# expressed genes.
# Replicate agggregation using mean
se.aggr.sh <- aggr_rep(data=se.sh, sam.factor='samples', con.factor='conditions', aggr='mean')
se.fil.arab <- filter_data(data=se.aggr.sh, sam.factor='samples', con.factor='conditions',
pOA=c(0.03, 6), CV=c(0.30, 100), dir=NULL) # Filtering of genes with low intensities and variance

# Similarly, the aSVG file corresponding to this data is pre-packaged and copied to the same
# temporary directory.
```

```
svg.arab <- system.file("extdata/shinyApp/example", 'arabidopsis.thaliana_organ_shm.svg',
package="spatialHeatmap")
file.copy(svg.arab, dir.svg, overwrite=TRUE)

# 3. The random data and aSVG files of two development stages of Arabidopsis organs.

# The gene expression data is randomly generated and pre-packaged.
pa.growth <- system.file("extdata/shinyApp/example", 'random_data_multiple_aSVGs.txt',
package="spatialHeatmap")
dat.growth <- read_fr(pa.growth); dat.growth[1:3, ]
# Paths of the two corresponsing aSVG files.
svg.arab1 <- system.file("extdata/shinyApp/example", 'arabidopsis.thaliana_organ_shm1.svg',
package="spatialHeatmap")
svg.arab2 <- system.file("extdata/shinyApp/example", 'arabidopsis.thaliana_organ_shm2.svg',
package="spatialHeatmap")
# Copy the two aSVG files to the same temporary directory.
file.copy(c(svg.arab1, svg.arab2), dir.svg, overwrite=TRUE)

# 4. Include aSVG templates of raster images.

pa.leaf <- system.file("extdata/shinyApp/example", 'dat_overlay.txt',
package="spatialHeatmap")
dat.leaf <- read_fr(pa.leaf); dat.leaf[1:2, ]
# Paths of the two aSVG files.
svg.leaf1 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm1.svg',
package="spatialHeatmap")
svg.leaf2 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm2.svg',
package="spatialHeatmap")
# Paths of the two corresponsing raster images of templates.
tmp.leaf1 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm1.png',
package="spatialHeatmap")
tmp.leaf2 <- system.file("extdata/shinyApp/example", 'maize_leaf_shm2.png',
package="spatialHeatmap")
# Copy the two aSVG and two template files to the same temporary directory.
file.copy(c(svg.leaf1, svg.leaf2, tmp.leaf1, tmp.leaf2), dir.svg, overwrite=TRUE)

# Make the pairing table, which describes matchings between the data and image files.
df.pair <- data.frame(name=c('chicken', 'arab', 'growth', 'leaf'), data=c('expr_chicken', 'expr_arab',
'random_data_multiple_aSVGs', 'leaf'), aSVG=c('gallus_gallus.svg', 'arabidopsis.thaliana_organ_shm.svg',
'arabidopsis.thaliana_organ_shm1.svg;arabidopsis.thaliana_organ_shm2.svg',
'maize_leaf_shm1.svg;maize_leaf_shm1.png;maize_leaf_shm2.svg;maize_leaf_shm2.png'))
# Note that multiple aSVGs should be concatenated by comma, semicolon, or single space.
df.pair

# Organize the data and pairing table in a list, and create the database.
dat.lis <- list(df_pair=df.pair, expr_chicken=se.fil.chk, expr_arab=se.fil.arab,
random_data_multiple_aSVGs=dat.growth, leaf=dat.leaf)
# Create the database in a temporary directory "db_shm".
dir.db <- paste0(tempdir(check=TRUE), '/db_shm') # Temporary directory.

if (!dir.exists(dir.db)) dir.create(dir.db)
write_hdf5(dat.lis=dat.lis, dir=dir.db, svg.dir=dir.svg, replace=TRUE)
```

```
# 4. Read data and/or pairing table from "data_shm.tar".
dat.lis1 <- read_hdf5(paste0(dir.db, '/data_shm.tar'), names(dat.lis))


# 5. Create customized Shiny app with the database.

if (!dir.exists('~/test_shiny')) dir.create('~/test_shiny')
lis.tar <- list(data=paste0(dir.db, '/data_shm.tar'), svg=paste0(dir.db, '/aSVGs.tar'))
custom_shiny(lis.tar, app.dir='~/test_shiny')
# Run the app.
shiny::runApp('~/test_shiny/shinyApp')


# Except "SummarizedExperiment", the database also accepts data in form of "data.frame". In that
# case, the columns should follow the naming scheme "sample__condition", i.e. a sample and a
# condition are concatenated by double underscore. The details are seen in the "data" argument
# of the function "spatial_hm".
# The following takes the Arabidopsis data as example.
df.arab <- assay(se.fil.arab); df.arab[1:3, 1:3]
# The new data list.
dat.lis2 <- list(df_pair=df.pair, expr_chicken=se.fil.chk, expr_arab=df.arab,
random_data_multiple_aSVGs=dat.growth)

# If the data does not have an corresponding aSVG or vice versa, in the pairing table the slot
# of missing data or aSVG should be filled with "none". In that case, on the Shiny user
# interface, users will be prompted to select an aSVG for the unpaired data or select a data
# for the unpaired aSVG.
# For example, if the aSVG "arabidopsis.thaliana_organ_shm.svg" has no matching data, the
# pairing table should be made like below.
df.pair1 <- data.frame(name=c('chicken', 'arab', 'growth'), data=c('expr_chicken', 'none',
'random_data_multiple_aSVGs'), aSVG=c('gallus_gallus.svg', 'arabidopsis.thaliana_organ_shm.svg',
'arabidopsis.thaliana_organ_shm1.svg;arabidopsis.thaliana_organ_shm2.svg'))
df.pair1
# The new data list.
dat.lis3 <- list(df_pair=df.pair, expr_chicken=se.fil.chk, none='none',
random_data_multiple_aSVGs=dat.growth)
```

Index