Package 'spatialHeatmap'

February 22, 2021

Type Package

Title spatialHeatmap

Version 1.0.0

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

VignetteBuilder knitr

Suggests knitr, rmarkdown, BiocStyle, RUnit, BiocGenerics, data.table, ExpressionAtlas, DT, reshape2, Biobase, GEOquery, shinyWidgets

Depends

Imports av, DESeq2, edgeR, WGCNA, flashClust, htmlwidgets, genefilter, ggplot2, ggdendro, grImport, grid, gridExtra, gplots, igraph, rsvg, shiny, dynamicTreeCut, grDevices, graphics, ggplotify, plotly, rols, stats, SummarizedExperiment, shinydashboard, utils, visNetwork, methods, xml2, yaml

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

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

RoxygenNote 7.1.0

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

git_branch RELEASE_3_12

git_last_commit c4cba0e

git_last_commit_date 2020-10-27

Date/Publication 2021-02-21

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

See Also

norm_data, aggr_rep, filter_data, spatial_hm, submatrix, adj_mod, matrix_hm, network, return_feature, update_feature, shiny_all, 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', data.trans='log2')
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', data.trans='log2')
# 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')
# Query aSVGs from remote.
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir=tmp.dir,
match.only=FALSE, remote=TRUE)
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)
# Query "gallus_gallus.svg".
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir=tmp.dir,
match.only=TRUE, remote=TRUE)
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(feature=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_all" without any
# parameters. Upon launched, the app automatically displays a pre-formatted example.
shiny_all()

# 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,
  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.
`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_all), "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_all).

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', data.trans='log2')
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', data.trans='log2')
# 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, 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.

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 column 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; 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/
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

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]
```

`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,
  custom = TRUE,
  custom.computed = TRUE,
  example = FALSE,
  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 or FALSE. Default is 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 default 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. 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')).
`custom`	Logical, TRUE or FALSE. If TRUE (default), the "customData" option under "Step 1: data sets" is included, which allows to upload datasets from local computer.
`custom.computed`	Logical, TRUE or FALSE. If TRUE (default), the "customComputdData" option

under "Step 1: data sets" is included, which allows to upload computed datasets from local computer. See adj_mod.

example — Logical, TRUE or FALSE. If TRUE, 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 pre-packaged examples are used for illustration purpose.
# 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 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, which include two growth stages.
lis.dld.mul <- list(data=data.path2, svg=c(svg.path2.1, svg.path2.2))
```

```
# 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.par=lis.par, lis.dld.single=lis.dld.single,
lis.dld.mul=lis.dld.mul, app.dir='~/test_shiny')
# Lauch the app.
shiny::runApp('~/test_shiny/shinyApp')
```

`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,
  pOA = c(0, 0),
  CV = c(-Inf, Inf),
  ann = NULL,
  sam.factor,
  con.factor,
  dir = NULL
)
```

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.

`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.

`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 "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.

`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_all`. 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.

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

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)

# 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)
```

`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,
  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.
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', data.trans='log2')
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', data.trans='log2')
# 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')
```

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,
  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 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.
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', data.trans='log2')
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', data.trans='log2')
# 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, norm.fun = "CNF", parameter.list = NULL, data.trans = "none")
```

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.
`norm.fun`	One of the normalizing functions: "CNF", "ESF", "VST", "rlog". 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". 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

data.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', data.trans='log2')
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', data.trans='log2')
```

---

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 = TRUE,
  dir = 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').
species	A vector of target species keywords (case insentitive), which is used to select aSVG files from a collection. *E.g.* c('gallus').
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.
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')
# Query the remote EBI aSVG repo.
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir='~/test',
match.only=FALSE, remote=TRUE)
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=FALSE)
# All matching aSVGs.
unique(feature.df$SVG)
# Select the target aSVG of chicken.
subset(feature.df, SVG=='gallus_gallus.svg')
```

`shiny_all` *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_all()
```

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_all()
```

`spatial_hm` *Create 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,
  sam.factor = NULL,
  con.factor = NULL,
  ID,
  lay.shm = "gene",
  ncol = 2,
  col.com = c("yellow", "orange", "red"),
  col.bar = "selected",
  bar.width = 0.08,
  legend.width = 1,
  bar.title.size = 0,
  trans.scale = NULL,
  tis.trans = NULL,
  width = 1,
  height = 1,
  legend.r = 1,
  sub.title.size = 11,
  legend.plot = "all",
  sam.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",
  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",
  preserve.scale = TRUE,
  verbose = TRUE,
  out.dir = NULL,
  anm.width = 650,
  anm.height = 550,
  selfcontained = FALSE,
  video.dim = "640x480",
  res = 500,
  interval = 1,
  framerate = 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.

`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.

`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".

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

`legend.width` The width of legend plot that ranges from 0 to 1 (default).

`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.

`tis.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.

`width` A numeric of overall width of all subplots, between 0 and 1. The default is 1.

`height` A numeric of overall height of all subplots, between 0 and 1. The default is 1.

`legend.r` A numeric to adjust the dimension of the legend plot. The default is 1. The larger, the higher ratio of width to height.

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

`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. Only applicable if multiple aSVG files are provided to `svg.path`.

`sam.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 indicated 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.

`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` A numeric of the shape outline size. Default is 0.2.

line.color	A character of the shape outline color. Default is "grey70".
preserve.scale	Logical, TRUE or FALSE. If TRUE (default), the relative dimensions of multiple aSVGs are preserved. Only applicable if multiple aSVG files are provided to svg.path. The original dimension (width/height) is specified in the top-most node "svg" in the aSVG file.
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.
anm.width	The width of spatial heatmaps in HTML files. Default is 650.
anm.height	The height of spatial heatmaps in HTML files. Default is 550.
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.
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 two-component list of the spatial heatmap(s) in ggplot format and a data frame of mapping between assayed samples and aSVG features.

Details

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

Author(s)

Jianhai Zhang <jzhan067@ucr.edu; zhang.jianhai@hotmail.com>
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

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', data.trans='log2')
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', data.trans='log2')
# 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)
```

`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,
  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 or SummarizedExperiment 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.
ann	Applies to data argument of SummarizedExperiment. 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_all. 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', data.trans='log2')
se.nor.chk <- norm_data(data=se.chk, norm.fun='CNF', data.trans='log2')
# 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)]
```

---

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. This function is designed to replace existing features in aSVG files with user-provided features. Note this function treats the first column in the feature data frame as user-provided features, so custom features must be the first column.

Usage

```
update_feature(feature, dir)
```

Arguments

`feature`	The data frame returned by `return_feature` with the user-provided features added as the first column.
`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 new features, 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

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.


# Make an empty directory "~/test" if not exist.
if (!dir.exists('~/test')) dir.create('~/test')
# Query the remote EBI aSVG repo.
feature.df <- return_feature(feature=c('heart', 'brain'), species=c('gallus'), dir='~/test',
match.only=TRUE, remote=TRUE)
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(feature=feature.df.new, dir='~/test')
```

Index