Spatial Heatmap Vignette

Jianhai Zhang (jzhan067@ucr.edu; zhang.jianhai@hotmail.com), advisor: Prof. Dr. Thomas Girke (thomas.girke@ucr.edu)

Last update: 07 September, 2018

Contents

Maintainer: Jianhai Zhang (jzhan067@ucr.edu; zhang.jianhai@hotmail.com)

Package Description

This package can generate interactive table matrix, interactive spatial heatmap, interactive matrix heatmap, interactive network based on given a data matrix and an associated SVG image. It requires the following packages pre-installed: shiny, grImport, XML, rsvg, ggplot2, DT, data.table, grid.Extra, plotly, ggdendro, WGCNA, visNetwork, flashClust, genefilter.

Details

This Shiny App based Spatial Heatmap can be used for interactive visualisation as long as a data matrix and an associated SVG image are provided. In the following the instructions are given with a gene expression matrix and an associated root tissue image in SVG format. This app has three main functionalities. First, it generates spatial tissue heatmaps where user defined tissue regions are coloured by the expression profile of a gene of interest. The gene expression information is uploaded as a table matrix and the associated tissue image is uploaded as an SVG format. Second, the app computes gene network modules based on gene expression profiles across tissue samples. It uses a matrix heatmap to visualize the expression of a chosen gene in the context of the corresponding gene network module the chosen gene belongs to. Third, the app dispalys the network module from the second section in the form of an interactive network graph. The network module identification is computationally demanding for large gene expression matrix (e.g.: $>$ 10,000 genes), so to make this app more widely applicable the "local mode" is developed for processing large data matrix. If the data matrix is small (e.g.: $<$ 10,000 genes), the "online mode" can be used. If the app times out at some time, then users want to refresh the page in their web browser.

Input

At first, users need to select a mode under "Select a work mode" in the left "Input" menu. The "Default" is most convenient for users to test the app, since this option relies on pre-uploaded files and users do not need to upload any files at all. The "Compute locally" should be selected if users have a large gene expression file (e.g.: $>$ 10,000 genes) while the "Compute online" can be selected if users have a small gene expression file (e.g.: $<$ 10,000 genes). In "Step 1: upload an svg file" and "Step 2: upload a gene expression file", users are asked to upload the svg file and associated gene expression file respectively. Details about how to properly format and associate custom SVG images with expression tables are provided here.

The "Step 3: is column or row gene?" option specifies if column or row is gene in the gene expression table, and "Step 4: separator" specifies the separator among the expression values. "Step 5: Color scheme" allows users to input colour components to construct colour scale for gene expression levels. Colours must be sepatated by comma, e.g. the default is "green,blue,purple,yellow,red".

In the gene matrix, the dimension names are gene IDs and sample/conditions. The sample/condition names MUST be fomatted this way: a sample name is followed by double underscore then the condition, such as "sample name__condition name". The meta data (e.g. gene annotation) can also be included in parallel with sample/condition. In the names of sample/condition and meta data, only letters, digits, single underscore, dots are allowed. The example SVG image and associated gene expression matrix can be downloaded in the instruction page of this app, which can be uploaded directly for testing.

The "Compute locally" option is designed for large gene expression data (e.g.: $>$ 10,000 genes), since gene network modules are identified with the R package WGCNA and the computation of topological overlap matrix (TOM) is time comsuming for large expression matrix. To maintain good performance, this process needs to be performed on user's local computer. The tutorial of how to compute locally is provided below.

The "Compute online" option is designed for small gene expression data (e.g.: $<$ 10,000 genes). The first two items under this option filter genes according to a proportion that a gene's expression values exceed a threthold A across all samples. Only the genes exceeding the specified proportion will be maintained.

The third and fourth items filter genes according to the coefficient of variation (CV). Only the genes with CV between the two specified values are maintained. The genes passing all these criteria are retained for downstream analysis. To save time, the TOM computation is only performed once when the matrix heatmap is displayed for the first time, but if the gene expression martix or its filter parameters are changed, the TOM will be re-computed.

The “Minmum module size” sets the minimum module size in gene module indentification. In “Network type”, “Signed” means both positive and negative adjacency between genes are maintained in network module identification while “Unsigned” takes the absolute values of negative adjacency.

Expression Matrix & Spatial Heatmap

The gene expression data is represented as an interactive table under “Expression Matrix”, where row names are gene IDs and column names are samples/conditions and meta data (the dimension names of the original table are adjusted internally in “Step 3: is column or row gene?”). Users can sort the expression values for a sample or search for a particular gene by its ID. Users can select multiple genes in the table to display corresponding spatial tissue heatmaps. In each such spatial heatmap, the gene expression levels are represented by colours for each sample under each condition. In the “Spatial Heatmap” section, users can customise the dimension and layout of the spatial tissue heatmaps, or scale the expression values by gene or sample. The scaled values are only used for matrix heatmap display, not for downstream module identifications.

Matrix Heatmap & Network

In the “Matrix Heatmap & Network” section, all gene IDs chosen in “Expression Matrix” are listed under “Select a gene to display matrix heatmap & network”. After a gene is selected from this list, the gene module containing the selected gene would be displayed in the form of interactive matrix heatmap, where the rows and columns are sorted by hierarchical clustering dendrograms and the chosen gene is tagged by a red rectangle. To explore the results, the matrix heatmap has several interactive features. For instance, users can zoom in and out by drawing a rectangle and by double clicking the image, respectively.

The gene network modules are identified at two alternative sensitivities levels (3, 2). From 3 to 2, the sensitivity decreases and results in more modules with smaller sizes. The “Select a module splitting sensitivity level” option allows users to choose which level to use for displaying the iteractive matrix heatmap and network.

The module in “Matrix Heatmap” is displayed as an interactive network. Nodes and edges mean genes and adjacency between genes respectively. Gene colours from blue, green to red mean gene connectivity is increasing while the edge length is inversely proportional to gene adjacency. If too many edges (e.g.: > 300) are displayed in this network, the app can possibly get stuck. So the “Input an adjacency threshold to display the adjacency network” option sets a threthold to filer out some edges. Only edges above the threthold are displayed in the network. Under the threthold option, the app outputs the total number of remaining edges resulting from each input adjacency threthold. If it is not too large (e.g.: < 300), users can check “Yes” under “Display or not?”, then the network can be displayed smoothly. 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 gene around. All the gene IDs in the network module are listed in “Select by id” according to gene connectivity in decreasing order. The selected gene ID is appended “_selected“, which can be easily identified from the list. By clicking an ID in this list, users can identify the corresponding gene in the network.

Example

#Get the path of SVG file and data matrix.
svg.path <- system.file("extdata/example", "test_final.svg", package = "spatialHeatmap")
data.path <- system.file("extdata/example", "gene_expr_test.txt", package = "spatialHeatmap")

#Local work mode for large data matrix.

## Filter the data matrix. The resulting data matrix "processed_data.txt" is automatically saved in the directory "local_mode_result", which should be uploaded to the app on web browser.
exp <- filter.data(data=data.path, sep="\t", isRowGen=T, c(0, 0), c(0.1, 10000), "processed_data")

## Compute the adjacency matrix and identify modules. The resulting adjacency file "adj.txt" and "mod.txt" are also automatically saved in "local_mode_result", which should be uploaded to the app on web browser.
adj_mod <- adj.mod(data=exp, type="signed", minSize=20)

# Lauch of the app.
spatial.hm.all().

# Plot the spatial heatmaps on R console.
spatial.hm(svg=svg.path, data=data.path, sep="\t", isRowGene=T, pOA=c(0.1, 3), CV=c(0.1, 1000), ID=c("ATMG00580", "ATCG01060"), colour=c("green", "blue", "purple", "yellow", "red"), width=1, height=1, sub.title.size=11, layout="gene", ncol=3)

Reference

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