Package: ctsGE (via r-universe)

October 15, 2026

Title Clustering of Time Series Gene Expression data

Version 1.39.0

Description Methodology for supervised clustering of potentially many predictor variables, such as genes etc., in time series datasets Provides functions that help the user assigning genes to predefined set of model profiles.

Depends R (>= 3.2)

Imports ccaPP, ggplot2, limma, reshape2, shiny, stats, stringr, utils

Suggests BiocStyle, dplyr, DT, GEOquery, knitr, pander, rmarkdown, testthat

URL https://github.com/michalsharabi/ctsGE

BugReports https://github.com/michalsharabi/ctsGE/issues

License GPL-2

LazyData true

RoxygenNote 5.0.1

VignetteBuilder knitr

biocViews ImmunoOncology, GeneExpression, Transcription, DifferentialExpression, GeneSetEnrichment, Genetics, Bayesian, Clustering, TimeCourse, Sequencing, RNASeq

Config/pak/sysreqs cmake make libicu-dev libuv1-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-04-28 12:43:56 UTC

RemoteUrl https://github.com/bioc/ctsGE

RemoteRef HEAD

RemoteSha 61143f9f761d1f60b1234f8d78c39041373afbc5

Contents

ClustIndexes 2
ctsGEShinyApp 3
index 4
PlotIndexesClust 5
PreparingTheIndexes 6
readTSGE 7

Index **9**

`ClustIndexes` *Clustering the indexes applying K-means*

Description

Clustering each index, that was predifined by `PreparingTheIndexes`, with `kmeans`.

Usage

```
ClustIndexes(x, scaling = TRUE)
```

Arguments

`x`	list of expression data and their indexes after running `PreparingTheIndexes`
`scaling`	Boolean parameter, does the data should be standardized before clustered. Default = TRUE

Details

The clustering is done with K-means. To choose an optimal k for K-means clustering, the Elbow method was applied, this method looks at the percentage of variance explained as a function of the number of clusters: the chosen number of clusters should be such that adding another cluster does not give much better modeling of the data. First, the ratio of the within-cluster sum of squares (WSS) to the total sum of squares (TSS) is computed for different values of k (i.e., 1, 2, 3 ...). The WSS, also known as sum of squared error (SSE), decreases as k gets larger. The Elbow method chooses the k at which the SSE decreases abruptly. This happens when the computed value of the WSS-to-TSS ratio first drops from 0.2.

Running `kmeans` and calculating the optimal k for each one of the indexes in the data could take a long time. To shorten the procedure the user can skip this step altogether and directly view a specific index and its clusters by running either the `PlotIndexesClust` or the `ctsGEShinyApp` function.

By default data is standardize before clustering,for clustering the raw counts set the **scaling** parameter to FALSE.

Value

list object is returned as output, with the relative culstered indexes table in *object$ClusteredIdxTable*, and the number of clusters for each index in *object$optimalK*

See Also

kmeans, PlotIndexesClust

Examples

```
data_dir <- system.file("extdata", package = "ctsGE")
files <- dir(path=data_dir,pattern = "\\.xls$")
rts <- readTSGE(files, path = data_dir,
labels = c("0h","6h","12h","24h","48h","72h"), skip = 10625 )
prts <- PreparingTheIndexes(rts)

tsCI <- ClustIndexes(prts)

head(tsCI$ClusteredIdxTable) #the table with the clusterd indexes
head(tsCI$optimalK) #the table with the number of clusters for each index
```

---

ctsGEShinyApp *GUI for interactive exploration of gene expression data.*

Description

Produce and launch Shiny app for interactive exploration of gene expression data. For more information about shiny apps *http://shiny.rstudio.com/*

Usage

```
ctsGEShinyApp(rts, min_cutoff = 0.5, max_cutoff = 0.7, mad.scale = TRUE,
  title = NULL)
```

Arguments

rts	list of an expression data that made by readTSGE
min_cutoff	A numeric the lower limit range to calculate the optimal cutoff for the data, default to 0.5 *See* PreparingTheIndexes.
max_cutoff	A numeric the upper limit range to calculate the optimal cutoff for the data, default to 0.7 *See* PreparingTheIndexes.
mad.scale	A boolean defaulting to TRUE as to what method of scaling to use. Default median-base scaling. FALSE, mean-base scaling
title	Character, the title at the header panel. default to NULL.

Details

The 'ctsGEShinyApp' function takes the ctsGE object and opens an html page as a GUI. On the web page, the user chooses the profile to visualize and the number of clusters (k parameter for K-means) to show. The line graph of the profile separated into the clusters will show in the main panel, and a list of the genes and their expressions will also be available. The tables and figures can be downloaded.

Value

Creates a shiny application and opens a shinyapp.io web page

See Also

shiny::ShinyApp

Examples

```
## Not run:
data_dir <- system.file("extdata", package = "ctsGE")
files <- dir(path=data_dir,pattern = "\\.xls$")
rts <- readTSGE(files, path = data_dir,
labels = c("0h","6h","12h","24h","48h","72h") )
ctsGEShinyApp(rts)
## End(Not run)
```

---

index *Indexing function*

Description

Takes a numeric vector and return an expression index (i.e., a sequence of 1,-1, and 0)

Usage

```
index(x, cutoff)
```

Arguments

x	A numeric
cutoff	A numeric, dermine the threshold for indexing

Details

The function defines limits around the center (median or mean), +/- cutoff value in median absolute deviation (MAD) or standard deviation (SD) units respectively.The user defines a parameter cutoff that determines the limits around the gene-expression center. Then the function calculates the index value at each time point according to:

1. **0:** standardized value is within the limits (+/- cutoff)
2. **1:** standardized value exceeds the upper limit (+ cutoff)
3. **-1:** standardized value exceeds the lower limit (- cutoff)

Value

Gene expression index

See Also

PreparingTheIndexes

Examples

```
rawCounts <-
c(103.5, 75.1, 97.3, 27.12, 34.83, 35.53, 40.59, 30.84, 16.39, 29.29)

(sCounts <- scale(rawCounts)[,1])# standardized mean-base scaling

cutoff <- seq(0.2,2,0.1)    # different cutoff produce different indexes

for(i in cutoff){print(index(sCounts,i))}
```

PlotIndexesClust *Graphic visualization of an index*

Description

The function generates graphs and tables of a specific index and its clusters. The user decides whether to supply the k or let the function calculate the k for the selected index

Usage

```
PlotIndexesClust(x, idx, k = NULL, scaling = TRUE)
```

Arguments

x	list of expression data and their indexes after running PreparingTheIndexes
idx	A character, the index to plot (e.g., for 8 time points "11100-1-1-1")
k	A numeric, number of clusters. If not given the function will calculate what is the optimal k for the index.
scaling	A boolean, default to TRUE, does the data should be standardized before clustered with K-means.

Value

A list with two objects:

1. Table of of a specific index and its clusters
2. Gene expression pattern graphs for each one of the clusters

See Also

ggplot, kmeans, ClustIndexes

Examples

```
data_dir <- system.file("extdata", package = "ctsGE")
files <- dir(path=data_dir,pattern = "\\.xls$")
rts <- readTSGE(files, path = data_dir,
labels = c("0h","6h","12h","24h","48h","72h"), skip = 10625 )
prts <- PreparingTheIndexes(rts)
pp <- PlotIndexesClust(prts,idx="00101-1")
pp$graphs # plots the line graphs
```

PreparingTheIndexes *Define an expression index for each gene*

Description

Reads the table of genes expression and return an expression index for each gene.

Usage

```
PreparingTheIndexes(x, min_cutoff = 0.5, max_cutoff = 0.7,
  mad.scale = TRUE)
```

Arguments

x	list of an expression data that made by readTSGE
min_cutoff	A numeric the lower limit range to calculate the optimal cutoff for the data, default to 0.5 *See Details*.
max_cutoff	A numeric the upper limit range to calculate the optimal cutoff for the data, default to 0.7 *See Details*.
mad.scale	A boolean defaulting to TRUE as to what method of scaling to use. Default median-base scaling. FALSE, mean-base scaling.

Details

1. First, the expression matrix is standardized. The function default standardizing method is a median-based scaling; alternatively, a mean-based scaling can be used. The new scaled values represent the distance of each gene at a certain time point from its center, median or mean, in median absolute deviation (MAD) units or standard deviation (SD) units, respectively.

2. The function compute the cutoff value following the idea that the clustering will be performed on small gene groups, an optimal cutoff value will be one that will minimize the number of genes in each group, i.e., generate index groups of equal size. The chi-squared values will be generate for

each cutoff value (from min_cutoff to max_cutoff parameter in increments of 0.05) the cutoff that generate the lowest chi-squared is chosen.

3. Next, the standardized values are converted to index values that indicate whether gene expression is above, below or within the limits around the center of the time series, i.e., **1 / -1 / 0**, respectively. The cutoff parameter determines the limits around the gene-expression center. Then the function calculates the index value at each time point according to:

1. **0:** standardized value is within the limits (+/- cutoff)
2. **1:** standardized value exceeds the upper limit (+ cutoff)
3. **-1:** standardized value exceeds the lower limit (- cutoff)

Value

list object is returned as output with the relative standarization table in *object$scaled*, and the indexes table in *object$index*

See Also

scale index

Examples

```
data_dir <- system.file("extdata", package = "ctsGE")
files <- dir(path=data_dir,pattern = "\\.xls$")
rts <- readTSGE(files, path = data_dir,
labels = c("0h","6h","12h","24h","48h","72h"), skip = 10625 )
prts <- PreparingTheIndexes(rts)
prts$cutoff # the optimal cutoff
```

---

readTSGE *Read and merge a set of files containing gene expression data*

Description

Reads and merges a set of text files containing normalized gene expression data

Usage

```
readTSGE(files, path = NULL, columns = c(1, 2), labels = NULL,
  desc = NULL, ...)
```

Arguments

files	character vector of filenames, or alternative a named list of tables for each time point.
path	character string giving the directory containing the files. The default is the current working directory.
columns	numeric vector stating which two columns contain the tag names and counts, respectively
labels	character vector giving short names to associate with the libraries.
desc	character vector with genes description (annotation),default to NULL Defaults to the file names.
...	other are passed to read.delim

Details

As input, the ctsGE package expects normalized expression table, where rows are genes and columns are samples Each file is assumed to contained digital gene expression data for one sample (or library), with transcript or gene identifiers in the first column and expression values in the second column. Transcript identifiers are assumed to be unique and not repeated in any one file. By default, the files are assumed to be tab-delimited and to contain column headings. The function forms the union of all transcripts and creates one big table with zeros where necessary. When reading the normalized expression values the function check whether there are rows that their median absolute deviation (MAD) value equal to zero and remove these rows. This step is important in order to continue to the next step of indexing the data. The function will output a message of how many genes were remove.

Value

A list with four objects:

1. expression matrix
2. samples names
3. tags - genes name
4. timePoints - number of time points

Examples

```
## Read all .txt files from current working directory
data_dir <- system.file("extdata", package = "ctsGE")
files <- dir(path=data_dir,pattern = "\\.xls$")

# reading only 2000 genes
rts <- readTSGE(files, path = data_dir,
 labels = c("0h","6h","12h","24h","48h","72h"), skip = 10625 )
```

Index

ClustIndexes, 2, *6*
ctsGEShinyApp, *2*, 3

ggplot, *6*

index, 4, *7*

kmeans, *2, 3*, *6*

PlotIndexesClust, *2, 3*, 5
PreparingTheIndexes, *2, 3*, *5*, 6

readTSGE, 7

scale, *7*