Package 'COTAN'

August 11, 2022

Type Package

Title COexpression Tables ANalysis

Version 1.0.0

Description Statistical and computational method to analyze the co-expression of gene pairs at single cell level. It provides the foundation for single-cell gene interactome analysis. The basic idea is studying the zero UMI counts' distribution instead of focusing on positive counts; this is done with a generalized contingency tables framework. COTAN can effectively assess the correlated or anti-correlated expression of gene pairs. It provides a numerical index related to the correlation and an approximate p-value for the associated independence test. COTAN can also evaluate whether single genes are differentially expressed, scoring them with a newly defined global differentiation index. Moreover, this approach provides ways to plot and cluster genes according to their co-expression pattern with other genes, effectively helping the study of gene interactions and becoming a new tool to identify cell-identity marker genes.

URL https://github.com/seriph78/COTAN

BugReports https://github.com/seriph78/COTAN/issues

Depends R (>= 4.1)

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Imports dplyr, methods, grDevices, Matrix, ggplot2, ggrepel, stats, parallel, tibble, tidyr, basilisk, reticulate, ComplexHeatmap, circlize, grid, scales, utils, rlang, Rfast

Suggests testthat (>= 3.0.0), spelling, knitr, data.table, R.utils, tidyverse, rmarkdown, htmlwidgets, MASS, factoextra, Rtsne, plotly, dendextend, BiocStyle, cowplot

Config/testthat/edition 3

SystemRequirements python

Language en-US

biocViews SystemsBiology, Transcriptomics, GeneExpression, SingleCell

VignetteBuilder knitr

LazyData false

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

git_branch RELEASE_3_15

git_last_commit b6a43f1

git_last_commit_date 2022-04-26

Date/Publication 2022-08-11

Author Galfrè Silvia Giulia [aut, cre]
(<https://orcid.org/0000-0002-2770-0344>),
Morandin Francesco [aut] (<https://orcid.org/0000-0002-2022-2300>),
Pietrosanto Marco [aut] (<https://orcid.org/0000-0001-5129-6065>),
Cremisi Federico [aut] (<https://orcid.org/0000-0003-4925-2703>),
Helmer-Citterich Manuela [aut]
(<https://orcid.org/0000-0001-9530-7504>)

Maintainer Galfrè Silvia Giulia <silvia.galfre@uniroma1.it>

R topics documented:

`add.row.to.meta` *add.row.to.meta*

Description

This function is used to add a line of information to the information data frame (metadata).

Usage

```
add.row.to.meta(object, text.line)

## S4 method for signature 'scCOTAN'
add.row.to.meta(object, text.line)
```

Arguments

`object`	a COTAN object
`text.line`	an array containing the information

Value

the updated COTAN object

Examples

```
data("ERCC.cotan")
text <- c("Test", "This is a test")
ERCC.cotan <- add.row.to.meta(ERCC.cotan, text)
get.metadata(ERCC.cotan)
```

automatic.COTAN.object.creation
automatic.COTAN.object.creation

Description

automatic.COTAN.object.creation

Usage

```
automatic.COTAN.object.creation(
  df,
  out_dir,
  GEO,
  sc.method,
  cond,
  mt = FALSE,
  mt_prefix = "^mt",
  cores = 1
)

## S4 method for signature 'data.frame'
automatic.COTAN.object.creation(
  df,
  out_dir,
  GEO,
  sc.method,
  cond,
  mt = FALSE,
  mt_prefix = "^mt",
  cores = 1
)
```

Arguments

df	dataframe with the row counts
out_dir	directory for the output
GEO	GEO or other code that identify the dataset
sc.method	Type of single cell RNA-seq method used
cond	A string that will identify the sample or condition. It will be part of the final file name.
mt	A boolean (default F). If T mitochondrial genes will be kept in the analysis, otherwise they will be removed.
mt_prefix	is the prefix that identify the mitochondrial genes (default is the mouse prefix: "^mt")
cores	number of cores to be used

Value

It return the COTAN object. It will also store it directly in the output directory

Examples

```
data("raw.dataset")
obj <- automatic.COTAN.object.creation(df= raw.dataset,
out_dir =  tempdir(),
GEO = "test_GEO",
sc.method = "test_method",
cond = "test")
```

---

clean *clean*

Description

Main function that can be used to check and clean the dataset. It also produce (using the function fun_linear) and store the estimators for nu and lambda. It also fill the raw.norm (raw / nu) and n_cell (the initial number of cells in the dataset)

Usage

```
clean(object)

## S4 method for signature 'scCOTAN'
clean(object)
```

Arguments

object	COTAN object

Value

a list of objects containing: "cl1" is the first cell cluster, "cl2" is the second cell cluster, "pca.cell.2" is a ggplot2 cell pca plot, "object" is the COTAN object with saved the estimated lambda and mu, "mu_estimator", "D" "pca_cells" pca numeric data.

Examples

```
data("ERCC.cotan")
ttm  <- clean(ERCC.cotan)
```

`cotan_analysis` *cotan_analysis*

Description

This is the main function that estimates the a vector to store all the negative binomial dispersion factors. It need to be run after `clean`

Usage

```
cotan_analysis(object, cores = 1)

## S4 method for signature 'scCOTAN'
cotan_analysis(object, cores = 1)
```

Arguments

`object`	A COTAN object
`cores`	number of cores to use. Default is 11.

Value

a COTAN object

Examples

```
data("ERCC.cotan")
ERCC.cotan  <- cotan_analysis(ERCC.cotan)
```

`drop.genes.cells` *drop.genes.cells*

Description

This finction remove an array of genes and/or cells from the original object raw matrix.

Usage

```
drop.genes.cells(object, genes = c(), cells = c())

## S4 method for signature 'scCOTAN'
drop.genes.cells(object, genes = c(), cells = c())
```

Arguments

object	a COTAN object
genes	an array of gene names
cells	an array of cell names

Value

the original object but with the raw matrix without the indicated cells and/or genes.

Examples

```
data("ERCC.cotan")
genes.to.rem = names(get.genes(ERCC.cotan)[1:10])
cells.to.rem = names(get.cell.size(ERCC.cotan)[1:10])
ERCC.cotan = drop.genes.cells(ERCC.cotan,genes.to.rem,cells.to.rem )
```

ERCC.cotan *COTAN object*

Description

The COTAN object for the ERCC dataset.

Usage

```
ERCC.cotan
```

Format

A structure with:

raw the raw dataset: 88 fake genes for 1015 fake cells

raw.norm raw divided for nu

coex

nu UDE

lambda average gene expression

a

hk genes expressed in all cells

n_cells final number of cells

meta meta data

Source

https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/ercc?

ERCCraw *Raw ERCC dataset*

Description

ERCC dataset

Usage

```
ERCCraw
```

Format

A data frame

Source

ERCC

est.min.parameters *est.min.parameters*

Description

This function estimates the nu and a parameters to minimize the...

Usage

```
est.min.parameters(object, par)

## S4 method for signature 'scCOTAN'
est.min.parameters(object, par)
```

Arguments

object	A COTAN object
par	A vector of inizial a and log(nu)

Value

vector

Examples

```
data("ERCC.cotan")
```

extract.coex *extract.coex*

Description

This function extract a complete or a prtial coex matrix from the COTAN object.

Usage

```
extract.coex(object, genes = "all")

## S4 method for signature 'scCOTAN'
extract.coex(object, genes = "all")
```

Arguments

object	A COTAN object
genes	A vector of gene names. These will be on the columns of the final data frame. By defaults the function will use all genes.

Value

the coex values data frame

Examples

```
data("ERCC.cotan")
coex <- extract.coex(ERCC.cotan)
```

fun_linear *fun_linear*

Description

Internal function to estimate the cell efficiency

Usage

```
fun_linear(object)
```

Arguments

object	COTAN object

Value

a list of object (dist_cells, to_clust, pca_cells, t_to_clust, mu_estimator, object)

get.a *get.a*

Description

This function extract the a array.

Usage

```
get.a(object)

## S4 method for signature 'scCOTAN'
get.a(object)
```

Arguments

object	A COTAN object

Value

the a array.

Examples

```
data("ERCC.cotan")
get.a(ERCC.cotan)[1:10]
```

get.cell.number *get.cell.number*

Description

This function extract number of analysed cells.

Usage

```
get.cell.number(object)

## S4 method for signature 'scCOTAN'
get.cell.number(object)
```

Arguments

object	A COTAN object

Value

the cell number value.

Examples

```
data("ERCC.cotan")
get.cell.number(ERCC.cotan)
```

`get.cell.size` *get.cell.size*

Description

This finction extract the cell raw libray size.

Usage

```
get.cell.size(object)

## S4 method for signature 'scCOTAN'
get.cell.size(object)
```

Arguments

`object`	A COTAN object

Value

an array with the library sizes

Examples

```
data("ERCC.cotan")
get.cell.size(ERCC.cotan)[1:10]
```

`get.coex` *get.coex*

Description

This function estimates and stores the coex matrix in the coex field. It need to be run after `cotan_analysis`

Usage

```
get.coex(object)

## S4 method for signature 'scCOTAN'
get.coex(object)
```

Arguments

`object`	A COTAN object

Value

It returns a COTAN object

Examples

```
data("ERCC.cotan")
obj <- get.coex(ERCC.cotan)
```

`get.constitutive.genes`
get.constitutive.genes This function return the genes expressed in all cells in the dataset.

Description

get.constitutive.genes This function return the genes expressed in all cells in the dataset.

Usage

```
get.constitutive.genes(object)

## S4 method for signature 'scCOTAN'
get.constitutive.genes(object)
```

Arguments

`object`	A COTAN object

Value

an array containing all genes expressed in all cells

Examples

```
data("ERCC.cotan")
get.constitutive.genes(ERCC.cotan)
```

`get.expected.ct` *get.expected.ct*

Description

This function is used to get the expected contingency table for a given pair of genes.

Usage

```
get.expected.ct(object, g1, g2)

## S4 method for signature 'scCOTAN'
get.expected.ct(object, g1, g2)
```

Arguments

`object`	The cotan object
`g1`	A gene
`g2`	The other gene

Value

A contingency table as dataframe

Examples

```
data("ERCC.cotan")
g1 <- get.genes(ERCC.cotan)[sample(length(get.genes(ERCC.cotan)), 1)]
g2 <- get.genes(ERCC.cotan)[sample(length(get.genes(ERCC.cotan)), 1)]
while (g1 %in% get.constitutive.genes(ERCC.cotan)) {
g1 <- get.genes(ERCC.cotan)[sample(length(get.genes(ERCC.cotan)), 1)]
}

while (g2 %in% get.constitutive.genes(ERCC.cotan)) {
   g2 <- get.genes(ERCC.cotan)[sample(length(get.genes(ERCC.cotan)), 1)]
}
get.expected.ct(object = ERCC.cotan, g1 = g1, g2 = g2)
```

get.GDI *get.GDI*

Description

This function produce a dataframe with the GDI for each genes.

Usage

```
get.GDI(object, type = "S")

## S4 method for signature 'scCOTAN'
get.GDI(object, type = "S")
```

Arguments

object	A COTAN object
type	Type of statistic to be used. Default is "S": Pearson's chi-squared test statistics. "G" is G-test statistics

Value

A dataframe

Examples

```
data("ERCC.cotan")
quant.p <- get.GDI(ERCC.cotan)
```

get.gene.coexpression.space *get.gene.coexpression.space*

Description

To make the GDI more specific, it may be desirable to restrict the set of genes against which GDI is computed to a selected subset V with the recommendation to include a consistent fraction of cell-identity genes, and possibly focusing on markers specific for the biological question of interest (for instance neural cortex layering markers). In this case we denote it as local differentiation index (LDI) relative to V.

Usage

```
get.gene.coexpression.space(object, n.genes.for.marker = 25, primary.markers)

## S4 method for signature 'scCOTAN'
get.gene.coexpression.space(object, n.genes.for.marker = 25, primary.markers)
```

Arguments

object	The COTAN object.
n.genes.for.marker	The number of genes correlated with the primary markers that we want to consider. By default this is 25.
primary.markers	A vector of primary marker names.

Value

A dataframe

Examples

```
data("ERCC.cotan")
df <- get.gene.coexpression.space(ERCC.cotan, n.genes.for.marker = 10,
primary.markers=get.genes(ERCC.cotan)[sample(length(get.genes(ERCC.cotan)),5)])
```

---

get.genes *get.genes*

Description

This function extract all genes in the dataset.

Usage

```
get.genes(object)

## S4 method for signature 'scCOTAN'
get.genes(object)
```

Arguments

object	A COTAN object

Value

a gene array

Examples

```
data("ERCC.cotan")
get.genes(ERCC.cotan)[1:10]
```

`get.lambda` *get.lambda*

Description

This function extract the lambda array (mean expression for each gene).

Usage

```
get.lambda(object)

## S4 method for signature 'scCOTAN'
get.lambda(object)
```

Arguments

`object`	A COTAN object

Value

the lambda array.

Examples

```
data("ERCC.cotan")
get.lambda(ERCC.cotan)[1:10]
```

`get.metadata` *get.metadata*

Description

This function extract the meta date stored for the dataset.

Usage

```
get.metadata(object)

## S4 method for signature 'scCOTAN'
get.metadata(object)
```

Arguments

`object`	A COTAN object

Value

the metatdata dataframe

Examples

```
data("ERCC.cotan")
get.metadata(ERCC.cotan)
```

`get.normdata` *get.normdata*

Description

This function extract the normalized count table.

Usage

```
get.normdata(object)

## S4 method for signature 'scCOTAN'
get.normdata(object)
```

Arguments

`object`	A COTAN object

Value

the normalized count dataframe (divided by nu).

Examples

```
data("ERCC.cotan")
get.normdata(ERCC.cotan)[1:10,1:10]
```

get.nu *get.nu*

Description

This function extract the nu array.

Usage

```
get.nu(object)

## S4 method for signature 'scCOTAN'
get.nu(object)
```

Arguments

object	A COTAN object

Value

the nu array.

Examples

```
data("ERCC.cotan")
get.nu(ERCC.cotan)[1:10]
```

get.observed.ct *get.observed.ct*

Description

This function is used to get the observed contingency table for a given pair of genes.

Usage

```
get.observed.ct(object, g1, g2)

## S4 method for signature 'scCOTAN'
get.observed.ct(object, g1, g2)
```

Arguments

object	The cotan object
g1	A gene
g2	The other gene

Value

A contingency table as dataframe

Examples

```
data("ERCC.cotan")
g1 <- get.genes(ERCC.cotan)[sample(length(get.genes(ERCC.cotan)), 1)]
g2 <- get.genes(ERCC.cotan)[sample(length(get.genes(ERCC.cotan)), 1)]
get.observed.ct(object = ERCC.cotan, g1 = g1, g2 = g2)
```

---

get.pval *get.pval*

Description

This function computes the p-values for genes in the COTAN object. It can be used genome-wide or setting some specific genes of interest. By default it computes the p-values using the S statistics (χ^2)

Usage

```
get.pval(object, gene.set.col = c(), gene.set.row = c(), type_stat = "S")

## S4 method for signature 'scCOTAN'
get.pval(object, gene.set.col = c(), gene.set.row = c(), type_stat = "S")
```

Arguments

object	a COTAN object
gene.set.col	an array of genes. It will be put in columns. If left empty the function will do it genome-wide.
gene.set.row	an array of genes. It will be put in rows. If left empty the function will do it genome-wide.
type_stat	By default it computes the S (χ^2)

Value

a p-value matrix

Examples

```
data("ERCC.cotan")
ERCC.cotan <- get.pval(ERCC.cotan,type_stat="S")
```

`get.rawdata` *get.rawdata*

Description

This function extract the raw count table.

Usage

```
get.rawdata(object)

## S4 method for signature 'scCOTAN'
get.rawdata(object)
```

Arguments

`object`	A COTAN object

Value

the raw count dataframe

Examples

```
data("ERCC.cotan")
get.rawdata(ERCC.cotan)[1:10,1:10]
```

`get.subset` *get.subset*

Description

This function estract the row data for a subset of cluesters.

Usage

```
get.subset(object, cluster.names)

## S4 method for signature 'scCOTAN'
get.subset(object, cluster.names)
```

Arguments

`object`	A COTAN object
`cluster.names`	A cluster names array

Value

the subset raw count dataframe

Examples

```
data("ERCC.cotan")
```

---

initRaw *initRaw*

Description

It starts to fill some fields of cotan object.

Usage

```
initRaw(object, GEO, sc.method = "10X", cond)

## S4 method for signature 'scCOTAN'
initRaw(object, GEO, sc.method = "10X", cond)
```

Arguments

object	the dataframe containing the raw data: it should be a data.frame, not a matrix.
GEO	a code reporting the GEO identification or other specific dataset code
sc.method	a string reporting the method used for the sequencing
cond	a string reporting the specific sample condition or time point

Value

A COTAN object to store all information

Examples

```
data("raw.dataset")
obj <- new("scCOTAN", raw = raw.dataset)
obj <- initRaw(obj, GEO="code" , sc.method="10X",cond = "mouse dataset")
```

mat2vec_rfast *mat2vec_rfast*

Description

mat2vec_rfast

Usage

```
mat2vec_rfast(mat)

## S4 method for signature 'matrix'
mat2vec_rfast(mat)
```

Arguments

mat	a symmetric matrix with all genes as row and column names

Value

a list formed by two arrays: "genes" with the gene names and "values" with all unique values.

Examples

```
mat <- matrix(0,nrow = 10, ncol = 10)
mat <- Rfast::lower_tri.assign(mat,c(1:55),diag = TRUE)
mat <- Rfast::upper_tri.assign(mat,v = Rfast::upper_tri(Rfast::transpose(mat)))
v <- mat2vec_rfast(mat)
```

plot_GDI *plot_GDI*

Description

This function directly evaluate and plot the GDI for a sample.

Usage

```
plot_GDI(object, cond, type = "S")

## S4 method for signature 'scCOTAN'
plot_GDI(object, cond, type = "S")
```

Arguments

object	A COTAN object
cond	A string corresponding to the condition/sample (it is used only for the title)
type	Type of statistic to be used. Default is "S": Pearson's chi-squared test statistics. "G" is G-test statistics

Value

A ggplot2 object

Examples

```
data("ERCC.cotan")
plot_GDI(ERCC.cotan, cond = "ERCC")
```

---

plot_general.heatmap *plot_general.heatmap*

Description

This function is used to plot an heatmap made using only some genes, as markers, and collecting all other genes correlated with these markers with a p-value smaller than the set threshold. Than all relations are plotted. Primary markers will be plotted as groups of rows. Markers list will be plotted as columns.

Usage

```
plot_general.heatmap(
  prim.markers = c("Satb2", "Bcl11b", "Cux1", "Fezf2", "Tbr1"),
  markers.list = c(),
  dir,
  condition,
  p_value = 0.001,
  symmetric = TRUE
)

## S4 method for signature 'ANY'
plot_general.heatmap(
  prim.markers = c("Satb2", "Bcl11b", "Cux1", "Fezf2", "Tbr1"),
  markers.list = c(),
  dir,
  condition,
  p_value = 0.001,
  symmetric = TRUE
)
```

Arguments

prim.markers	A set of genes plotted as rows.
markers.list	A set of genes plotted as columns.
dir	The directory where the COTAN object is stored.
condition	The prefix for the COTAN object file.
p_value	The p-value threshold
symmetric	A boolean: default F. If T the union of prim.markers and marker.list is sets as both rows and column genes

Value

A ggplot2 object

Examples

```
data("ERCC.cotan")
data_dir <- tempdir()
saveRDS(ERCC.cotan, file = file.path(data_dir,"ERCC.cotan.RDS"))
# some genes
primary.markers <- c("ERCC-00154","ERCC-00156","ERCC-00164")
#a example of named list of different gene set
gene.sets.list <- list("primary.markers"=primary.markers,
                   "2.R" = c("ERCC-00170","ERCC-00158"),
                   "3.S"=c("ERCC-00160","ERCC-00162"))
plot_general.heatmap(prim.markers = primary.markers, p_value = 0.05, markers.list =gene.sets.list ,
condition ="ERCC" ,dir = paste0(data_dir,"/"))
```

---

plot_heatmap *plot_heatmap*

Description

This is the function that create the heatmap of one or more COTAN object.

Usage

```
plot_heatmap(p_val.tr = 0.05, df_genes, sets, conditions, dir)

## S4 method for signature 'ANY'
plot_heatmap(p_val.tr = 0.05, df_genes, sets, conditions, dir)
```

Arguments

p_val.tr	p-value threshold. Default is 0.05
df_genes	this is a list of gene array. The first array will define genes in the columns.
sets	This is a numeric array indicating from which fields of the previous list will be considered
conditions	An array of prefixes indicating the different files.
dir	The directory in which are all COTAN files (corresponding to the previous prefixes)

Value

a ggplot object

Examples

```
data("ERCC.cotan")
data_dir <- tempdir()
saveRDS(ERCC.cotan, file = file.path(data_dir,"ERCC.cotan.RDS"))
# some genes
primary.markers <- c("ERCC-00154","ERCC-00156","ERCC-00164")
#a example of named list of different gene set
gene.sets.list <- list("primary.markers"=primary.markers,
                   "2.R" = c("ERCC-00170","ERCC-00158"),
                   "3.S"=c("ERCC-00160","ERCC-00162"))
plot_heatmap(p_v = 0.05, df_genes =gene.sets.list ,
sets =c(2,3) ,conditions =c("ERCC") ,dir = paste0(data_dir,"/"))
```

---

raw.dataset *Raw sample dataset*

Description

A subsample of a real sc-RNAseq dataset

Usage

```
raw.dataset
```

Format

A data frame with 2000 genes and 815 cells:

Source

GEO GSM2861514

scCOTAN-class *scCOTAN-class*

Description

Define my COTAN structure

Value

the object class

Slots

`raw` ANY. To store the raw data matrix

`raw.norm` ANY. To store the raw data matrix divided for the cell efficiency estimated (nu)

`coex` ANY. The coex matrix (sparce)

`nu` vector.

`lambda` vector.

`a` vector.

`hk` vector.

`n_cells` numeric.

`meta` data.frame.

`yes_yes` ANY.

`clusters` vector.

`cluster_data` data.frame.

Examples

```
data("ERCCraw")
obj <- new("scCOTAN",raw = data)
```

vec2mat_rfast *vec2mat_rfast*

Description

vec2mat_rfast

Usage

```
vec2mat_rfast(x, genes = "all")

## S4 method for signature 'list'
vec2mat_rfast(x, genes = "all")
```

Arguments

x	a list formed by two arrays: "genes" with the gene names and "values" with all unique values.
genes	a vector with all wanted genes or "all". By default is equal to "all" in this way it recreate the entire coex dataframe.

Value

a dataframe

Examples

```
v <- list("genes" = paste0("gene.",c(1:10)),"values"=c(1:55))
genes <- c("gene.3","gene.4","gene.7")
df <- vec2mat_rfast(v,genes)
df2 <- vec2mat_rfast(v)
```

Index