Package 'RNentropy'

October 12, 2022

Type Package

Title Entropy Based Method for the Detection of Significant Variation in Gene Expression Data

Version 1.2.3

Depends R (>= 2.10)

Date 2022-04-05

Maintainer Federico Zambelli <federico.zambelli@unimi.it>

Description An implementation of a method based on information theory devised for the identification of genes showing a significant variation of expression across multiple conditions. Given expression estimates from any number of RNA-Seq samples and conditions it identifies genes or transcripts with a significant variation of expression across all the conditions studied, together with the samples in which they are over- or under-expressed. Zambelli et al. (2018) <doi:10.1093/nar/gky055>.

License GPL-3

RoxygenNote 6.0.1

NeedsCompilation no

Author Federico Zambelli [cre] (<https://orcid.org/0000-0003-3487-4331>), Giulio Pavesi [aut] (<https://orcid.org/0000-0001-5705-6249>)

Repository CRAN

Date/Publication 2022-04-13 09:52:35 UTC

R topics documented:

RNentropy-package *Entropy Based Method for the Detection of Significant Variation in Gene Expression Data*

Description

An implementation of a method based on information theory devised for the identification of genes showing a significant variation of expression across multiple conditions. Given expression estimates from any number of RNA-Seq samples and conditions it identifies genes or transcripts with a significant variation of expression across all the conditions studied, together with the samples in which they are over- or under-expressed. Zambelli et al. (2018) <doi:10.1093/nar/gky055>.

Author(s)

NA

Maintainer: Federico Zambelli <federico.zambelli@unimi.it>

References

doi = 10.1093/nar/gky055 doi = 10.1007/978-1-0716-1307-8_6

Examples

```
#load expression values and experiment design
data("RN_Brain_Example_tpm", "RN_Brain_Example_design")
#compute statistics and p-values (considering only a subset of genes due to
#examples running time limit of CRAN).
Results <- RN_calc(RN_Brain_Example_tpm[1:10000,], RN_Brain_Example_design)
#select only genes with significant changes of expression
Results <- RN_select(Results)
#Compute the Point Mutual information Matrix
Results <- RN_pmi(Results)

#load expression values and experiment design
data("RN_BarresLab_FPKM", "RN_BarresLab_design")
#compute statistics and p-values (considering only a subset of genes due to
#examples running time limit of CRAN)
Results_B <- RN_calc(RN_BarresLab_FPKM[1:10000,], RN_BarresLab_design)
#select only genes with significant changes of expression
Results_B <- RN_select(Results_B)
#Compute the Point Mutual information matrix
Results_B <- RN_pmi(Results_B)
```

`RNentropy` *RNentropy*

Description

This function runs RNentropy on a file containing normalized expression values.

Usage

```
RNentropy(file, tr.col, design, header = TRUE, skip = 0, skip.col = NULL,
col.names = NULL)
```

Arguments

`file` Expression data file. It must contain a column with univocal identifiers for each row (typically transcript or gene ids).

`tr.col` The column # (1 based) in the file containing the univocal identifiers (usually transcript or gene ids).

`design` The RxC design matrix where R (rows) corresponds to the number of numeric columns (samples) in 'file' and C (columns) to the number of conditions. It must be a binary matrix with one and only one '1' for every row, corresponding to the condition (column) for which the sample corresponding to the row has to be considered a biological ot technical replicate. See the example 'RN_Brain_Example_design' for the design matrix of 'RN_Brain_Example_tpm' which has three replicates for three conditions (three rows) for a total of nine samples (nine rows).

`header` Set this to FALSE if 'file' does not have an header row. If header is TRUE the read.table function used by RNentropy will try to assign names to each column using the header. Default = TRUE

`skip` Number of rows to skip at the beginning of 'file'. Rows beginning with a '#' token will be skipped even if skip is set to 0.

`skip.col` Columns that will not be imported from 'file' (1 based) and not included in the subsequent analysis. Useful if you have more than annotation column, e.g. gene name in the first, transcript ID in the second columns.

`col.names` Assign names to the imported columns. The number of names must correspond to the columns effectively imported (thus, no name for the skipped ones). Also, tr.col is not considered an imported column. Useful to assign sample names to expression columns.

Value

Refer to the help for RN_calc.

Author(s)

Giulio Pavesi - Dep. of Biosciences, University of Milan

Federico Zambelli - Dep. of Biosciences, University of Milan

Examples

```
#load expression values and experiment design
data("RN_Brain_Example_tpm", "RN_Brain_Example_design")
#compute statistics and p-values (considering only a subset of genes due to
#examples running time limit of CRAN)
Results <- RN_calc(RN_Brain_Example_tpm[1:10000,], RN_Brain_Example_design)
#select only genes with significant changes of expression
Results <- RN_select(Results)


## The function is currently defined as
function (file, tr.col, design, header = TRUE, skip = 0, skip.col = NULL,
    col.names = NULL)
{
    TABLE <- read.table(file, row.names = tr.col, header = header,
        skip = skip, blank.lines.skip = TRUE, comment.char = "#")
    if (!is.null(skip.col)) {
        TABLE <- .RN_delete_col(TABLE, tr.col, skip.col)
    }
    if (!is.null(col.names)) {
        colnames(TABLE) <- col.names
    }
    return(RN_calc(TABLE, design))
  }
```

RN_BarresLab_design *RN_BarresLab_design*

Description

The experiment design matrix for RN_BarresLab_design

Usage

```
data("RN_Brain_Example_design")
```

Format

The format is: num[1:9,1:3]

Details

A binary matrix where conditions correspond to columns and samples to rows. In this example there are seven conditions (cell types) and seven samples with the expression measured by mean FPKM.

Examples

```
data(RN_Brain_Example_design)
```

`RN_BarresLab_FPKM` *RN_BarresLab_FPKM*

Description

Data expression values (mean FPKM) for different cerebral cortex cell from Zhang et al: "An RNA-Seq transcriptome and splicing database of glia, neurons, and vascular cells of the cerebral cortex", J Neurosci 2014 Sep. RN_BarresLab_design is the corresponding design matrix.

Usage

```
data("RN_BarresLab_FPKM")
```

Format

A data frame with 12978 observations on the following 7 variables.

`Astrocyte` a numeric vector

`Neuron` a numeric vector

`OPC` a numeric vector

`NFO` a numeric vector

`MO` a numeric vector

`Microglia` a numeric vector

`Endothelial` a numeric vector

Source

Zhang Y, Chen K, Sloan SA, Bennett ML, Scholze AR, O'Keeffe S, Phatnani HP, Guarnieri P, Caneda C, Ruderisch N, Deng S, Liddelow SA, Zhang C, Daneman R, Maniatis T, Barres BA, Wu JQ. An RNA-sequencing transcriptome and splicing database of glia, neurons, and vascular cells of the cerebral cortex. J Neurosci. 2014 Sep 3;34(36):11929-47. doi: 10.1523/JNEUROSCI.1860-14.2014. Erratum in: J Neurosci. 2015 Jan 14;35(2):846-6. PubMed PMID: 25186741; PubMed Central PMCID: PMC4152602.

Data were downloaded from https://web.stanford.edu/group/barres_lab/brain_rnaseq.html

Examples

```
data(RN_BarresLab_FPKM)
```

`RN_Brain_Example_design` *RN_Brain_Example_design*

Description

The example design matrix for RN_Brain_Example_tpm

Usage

```
data("RN_Brain_Example_design")
```

Format

The format is: num[1:9,1:3]

Details

A binary matrix where conditions correspond to the columns and samples to the rows. In this example there are three conditions (individuals) and nine samples, with three replicates for each condition.

Examples

```
data(RN_Brain_Example_design)
```

`RN_Brain_Example_tpm` *RN_Brain_Example_tpm*

Description

Example data expression values from brain of three different individuals, each with three replicates. RN_Brain_Example_design is the corresponding design matrix.

Usage

```
data("RN_Brain_Example_tpm")
```

Format

A data frame with 78699 observations on the following 9 variables.

`GENE` a factor vector

`BRAIN_2_1` a numeric vector

`BRAIN_2_2` a numeric vector

`BRAIN_2_3`	a numeric vector
`BRAIN_3_1`	a numeric vector
`BRAIN_3_2`	a numeric vector
`BRAIN_3_3`	a numeric vector
`BRAIN_1_1`	a numeric vector
`BRAIN_1_2`	a numeric vector
`BRAIN_1_3`	a numeric vector

Examples

```
data(RN_Brain_Example_tpm)
```

RN_calc *RN_calc*

Description

Computes both global and local p-values, and returns the results in a list containing for each gene the original expression values and the associated global and local p-values (as -log10(p-value)).

Usage

```
RN_calc(X, design)
```

Arguments

`X` data.frame with expression values. It may contain additional non numeric columns (eg. a column with gene names).

`design` The RxC design matrix where R (rows) corresponds to the number of numeric columns (samples) in 'file' and C (columns) to the number of conditions. It must be a binary matrix with one and only one '1' for every row, corresponding to the condition (column) for which the sample corresponding to the row has to be considered a biological ot technical replicate. See the example 'RN_Brain_Example_design' for the design matrix of 'RN_Brain_Example_tpm' which has three replicates for three conditions (three rows) for a total of nine samples (nine rows). design defaults to a square matrix of independent samples (diagonal = 1, everything else = 0)

Value

`gpv` -log10 of the global p-values

`lpv` -log10 of the local p-values

`c_like` results formatted as in the output of the C++ implementation of RNentropy.

`res` The results data.frame with the original expression values and the associated -log10 of global and local p-values.

`design` the experimental design matrix

Author(s)

Giulio Pavesi - Dep. of Biosciences, University of Milan

Federico Zambelli - Dep. of Biosciences, University of Milan

Examples

```
data("RN_Brain_Example_tpm", "RN_Brain_Example_design")
#compute statistics and p-values (considering only a subset of genes due to
#examples running time limit of CRAN)
Results <- RN_calc(RN_Brain_Example_tpm[1:10000,], RN_Brain_Example_design)

## The function is currently defined as

function(X, design = NULL)
{
  if(is.null(design))
  {
    design <- .RN_default_design(sum(sapply(X, is.numeric)))
  }

Results <- list(expr = X, design = design)

GPV <- RN_calc_GPV(X, bind = FALSE)
LPV <- RN_calc_LPV(X, design = design, bind = FALSE)

TABLE = cbind(X,'---',GPV,'---',LPV)

Results$gpv <- GPV
Results$lpv <- LPV
Results$c_like <- TABLE
Results$res <- cbind(X, GPV, LPV)

return(Results)
}
```

RN_calc_GPV *RN_calc_GPV*

Description

This function calculates global p-values from expression data, represented as -log10(p-values).

Usage

```
RN_calc_GPV(X, bind = TRUE)
```

Arguments

X	data.frame with expression values. It may contain additional non numeric columns (eg. a column with gene names)
bind	See Value (Default: TRUE)

Value

If bind is TRUE the function returns a data.frame with the original expression values from 'X' and an attached column with the -log10() of the global p-value, otherwise only the numeric vector of -log10(p-values) is returned.

Author(s)

Giulio Pavesi - Dep. of Biosciences, University of Milan

Federico Zambelli - Dep. of Biosciences, University of Milan

Examples

```
data("RN_Brain_Example_tpm")
GPV <- RN_calc_GPV(RN_Brain_Example_tpm)

## The function is currently defined as
function (X, bind = TRUE)
{
    rnums <- sapply(X, is.numeric)
    GL_LPV <- apply(X[rnums], 1, ".RN_calc_GPV_row")
    if (bind) {
        GPV <- cbind(X, GL_LPV)
        return(GPV)
    }
    else {
        return(GL_LPV)
    }
  }
```

RN_calc_LPV *RN_calc_LPV*

Description

This function calculates local p-values from expression data, output as -log10(p-value).

Usage

```
RN_calc_LPV(X, design, bind = TRUE)
```

Arguments

X	data.frame with expression values. It may contain additional non numeric columns (eg. a column with gene names)
design	The design matrix. Refer to the help for RN_calc for further info.
bind	See Value (Default = TRUE)

Value

If bind is TRUE the function returns a data.frame with the original expression values from data.frame X and attached columns with the -log10(p-values) computed for each sample, otherwise only the matrix of -log10(p-values) is returned.

Author(s)

Giulio Pavesi - Dep. of Biosciences, University of Milan

Federico Zambelli - Dep. of Biosciences, University of Milan

Examples

```
data("RN_Brain_Example_tpm", "RN_Brain_Example_design")
LPV <- RN_calc_LPV(RN_Brain_Example_tpm, RN_Brain_Example_design)
## The function is currently defined as

function(X, design = NULL, bind = TRUE)
{
  if(is.null(design))
  {
    design <- .RN_default_design(sum(sapply(X, is.numeric)))
  }

rnums <- sapply(X, is.numeric)

.RN_design_check(X, design, rnums)
RL <- .RN_get_replicate_list(design)

l1 <- rep("LOC_LPV", length(X[rnums]))
l2 <- colnames(X[rnums])
PV <- apply(X[rnums],  1, '.RN_calc_LPV_row', RL = RL)
PV <- t(PV)
colnames(PV) <- paste(l1, l2, sep="_")

if(bind)
{
return (cbind(X, PV))
}
else
return (PV)
}
```

RN_pmi *Compute point mutual information matrix for the experimental conditions.*

Description

Compute point mutual information for experimental conditions from the overxexpressed genes identified by RN_select.

Usage

```
RN_pmi(Results)
```

Arguments

`Results`	The output of RNentropy, RN_calc or RN_select. If RN_select has not already run on the results it will be invoked by RN_pmi using default arguments.

Value

The original input containing

`gpv`	-log10 of the Global p-values
`lpv`	-log10 of the Local p-values
`c_like`	a table similar to the one you obtain running the C++ implementation of RNentropy.
`res`	The results data.frame containing the original expression values together with the -log10 of Global and Local p-values.
`design`	The experimental design matrix.
`selected`	Transcripts/genes with a corrected Global p-value lower than gpv_t. Each condition N gets a condition_N column which values can be -1,0,1 or NA. 1 means that all the replicates of this condition seems to be consistently over-expressed w.r.t the overall expression of the transcript in all the conditions (that is, all the replicates of condition N have a positive Local p-value <= lpv_t). -1 means that all the replicates of this condition seems to be consistently under-expressed w.r.t the overall expression of the transcript in all the conditions (that is, all the replicates of condition N have a negative Local p-value and abs(Local p-values) <= lpv_t). 0 means that one or more replicates have an abs(Local p-value) > lpv_t. NA means that the Local p-values of the replicates are not consistent for this condition.

And two new matrices:

`pmi`	Point mutual information matrix of the conditions.
`npmi`	Normalized point mutual information matrix of the conditions.

Author(s)

Giulio Pavesi - Dep. of Biosciences, University of Milan

Federico Zambelli - Dep. of Biosciences, University of Milan

Examples

```
data("RN_Brain_Example_tpm", "RN_Brain_Example_design")
#compute statistics and p-values (considering only a subset of genes due to
#examples running time limit of CRAN)
Results <- RN_calc(RN_Brain_Example_tpm[1:10000,], RN_Brain_Example_design)
Results <- RN_select(Results)
Results <- RN_pmi(Results)

## The function is currently defined as
RN_pmi <- function(Results)
{
  if(is.null(Results$selected))
    Results <- RN_select(Results)

  Results$pmi <- matrix(nrow = ncol(Results$design),
                        ncol = ncol(Results$design))

  colnames(Results$pmi) <- colnames(Results$design)
  rownames(Results$pmi) <- colnames(Results$design)

  Results$npmi <- Results$pmi

  colshift <- ncol(Results$selected) - ncol(Results$design)

  for(x in 1:nrow(Results$pmi))
  {
    for(y in 1:nrow(Results$pmi))
    {
      if(x > y)
      {
        Results$pmi[x,y] <- Results$pmi[y,x]
        Results$npmi[x,y] <- Results$npmi[y,x]
        next
      } else
      {
        sum_x <- sum(Results$selected[,x+colshift] == 1, na.rm = TRUE)
        sum_y <- sum(Results$selected[,y+colshift] == 1, na.rm = TRUE)
        sum_xy <- sum(Results$selected[,x+colshift] == 1 &
                        Results$selected[,y+colshift] == 1, na.rm = TRUE)
        freq_x <- sum_x / nrow(Results$selected)
        freq_y <- sum_y / nrow(Results$selected)
        freq_xy <- sum_xy / nrow(Results$selected)

        h_xy <- log2(1/freq_xy)

        Results$pmi[x,y] <- log2(freq_xy / (freq_x * freq_y))
        Results$npmi[x,y] <- Results$pmi[x,y] / h_xy
      }
    }
  }

  return (Results)
  }
```

RN_select *Select transcripts/genes with significant p-values.*

Description

Select transcripts with global p-value lower than an user defined threshold and provide a summary of over- or under-expression according to local p-values.

Usage

```
RN_select(Results, gpv_t = 0.01, lpv_t = 0.01, method = "BH")
```

Arguments

`Results` The output of RNentropy or RN_calc.

`gpv_t` Threshold for global p-value. (Default: 0.01)

`lpv_t` Threshold for local p-value. (Default: 0.01)

`method` Multiple test correction method. Available methods are the ones of p.adjust. Type p.adjust.methods to see the list. Default: BH (Benjamini & Hochberg)

Value

The original input containing

`gpv` -log10 of the global p-values

`lpv` -log10 of the local p-values

`c_like` results formatted as in the output of the C++ implementation of RNentropy.

`res` The results data.frame containing the original expression values together with the -log10 of global and local p-values.

`design` The experimental design matrix.

and a new dataframe

`selected` Transcripts/genes with a corrected global p-value lower than gpv_t. For each condition it will contain a column where values can be -1,0,1 or NA. 1 means that all the replicates of this condition have expression value higher than the average and local p-value <= lpv_t (thus the corresponding gene will be over-expressed in this condition). -1 means that all the replicates of this condition have expression value lower than the average and local p-value <= lpv_t (thus the corresponding gene will be under-expressed in this condition). 0 means that at least one of the replicates has a local p-value > lpv_t. NA means that the local p-values of the replicates are not consistent for this condition, that is, at least one replicate results to be over-expressed and at least one results to be under-expressed.

Author(s)

Giulio Pavesi - Dep. of Biosciences, University of Milan

Federico Zambelli - Dep. of Biosciences, University of Milan

Examples

```
data("RN_Brain_Example_tpm", "RN_Brain_Example_design")
#compute statistics and p-values (considering only a subset of genes due to
#examples running time limit of CRAN)
Results <- RN_calc(RN_Brain_Example_tpm[1:10000,], RN_Brain_Example_design)
Results <- RN_select(Results)

## The function is currently defined as
function (Results, gpv_t = 0.01, lpv_t = 0.01, method = "BH")
{
    lpv_t <- -log10(lpv_t)
    gpv_t <- -log10(gpv_t)
    Results$gpv_bh <- -log10(p.adjust(10^-Results$gpv, method = method))
    true_rows <- (Results$gpv_bh >= gpv_t)
    design_b <- t(Results$design > 0)
    Results$lpv_sel <- data.frame(row.names = rownames(Results$lpv)[true_rows])
    for (d in seq_along(design_b[, 1])) {
        col <- apply(Results$lpv[true_rows, ], 1, ".RN_select_lpv_row",
            design_b[d, ], lpv_t)
        Results$lpv_sel <- cbind(Results$lpv_sel, col)
        colnames(Results$lpv_sel)[length(Results$lpv_sel)] <- paste("condition",
            d, sep = "_")
    }
    lbl <- Results$res[, !sapply(Results$res, is.numeric)]
    Results$selected <- cbind(lbl[true_rows], Results$gpv[true_rows],
Results$gpv_bh[true_rows], Results$lpv_sel)
    colnames(Results$selected) <- c(names(which(!sapply(Results$res,
is.numeric))), "GL_LPV", "Corr. GL_LPV", colnames(Results$lpv_sel))
    Results$selected <- Results$selected[order(Results$selected[,3], decreasing=TRUE),]
    Results$lpv_sel <- NULL
    return(Results)
  }
```

Index