Package 'NormExpression'

March 5, 2018

Type Package

Title Normalize Gene Expression Data using Evaluated Methods

Version 0.1.0

Author Zhenfeng Wu , Shan Gao

Maintainer Shan Gao <gao_shan@mail.nankai.edu.cn>

Description It provides a framework and a fast and simple way for researchers to evaluate methods (particularly some data-driven methods or their own methods) and then select a best one for data normalization in the gene expression analysis, based on the consistency of metrics and the consistency of datasets.
Zhenfeng Wu, Weixiang Liu, Xiufeng Jin, Deshui Yu, Hua Wang, Gustavo Glusman, Max Robinson, Lin Liu, Jishou Ruan and Shan Gao (2018) <doi:10.1101/251140>.

License Artistic-2.0

Encoding UTF-8

LazyData true

NeedsCompilation no

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2018-03-05 18:38:58 UTC

R topics documented:

---

bkRNA18 *bkRNA18*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
data("bkRNA18")
```

Format

A data frame with 57955 observations on the following 18 variables.

`col36l6_1` a numeric vector

`col38l6_3` a numeric vector

`col39l6_5` a numeric vector

`col40l6_7` a numeric vector

`col44l6_9` a numeric vector

`col45l6_11` a numeric vector

col47l6_13 a numeric vector
col48l6_97 a numeric vector
col52l6_17 a numeric vector
col36l6_2 a numeric vector
col38l6_4 a numeric vector
col39l6_6 a numeric vector
col40l6_8 a numeric vector
col44l6_10 a numeric vector
col45l6_12 a numeric vector
col47l6_14 a numeric vector
col48l6_98 a numeric vector
col52l6_18 a numeric vector

Examples

```
data(bkRNA18)
## maybe str(bkRNA18) ; plot(bkRNA18) ...
```

---

bkRNA18_factors *bkRNA18_factors*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
data("bkRNA18_factors")
```

Format

A data frame with 18 observations on the following 13 variables.

HG7 a numeric vector
ERCC a numeric vector
TN a numeric vector
TC a numeric vector
CR a numeric vector
NR a numeric vector
DESeq a numeric vector
UQ a numeric vector
TMM a numeric vector
TU a numeric vector
NCS a numeric vector
ES a numeric vector
GAPDH a numeric vector

Examples

```
data(bkRNA18_factors)
## maybe str(bkRNA18_factors) ; plot(bkRNA18_factors) ...
```

---

calcFactorRLE *calcFactorRLE*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
calcFactorRLE(data, p = p)
```

Arguments

data

p

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, p = p)
{
    gm <- exp(rowMeans(.log(data), na.rm = TRUE))
    apply(data, 2, function(u) quantile((u/gm)[u != 0], na.rm = TRUE,
        p = p))
  }
```

---

calcFactorUpperquartile
calcFactorUpperquartile

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
calcFactorUpperquartile(data, lib.size, p = p)
```

Arguments

data

lib.size

p

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, lib.size, p = p)
{
    y <- t(t(data)/lib.size)
    f <- apply(y, 2, function(x) quantile(x[x != 0], p = p))
  }
```

---

calcFactorWeighted *calcFactorWeighted*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
calcFactorWeighted(obs, ref, libsize.obs, libsize.ref, logratioTrim,
sumTrim, doWeighting, Acutoff)
```

Arguments

obs

ref

libsize.obs

libsize.ref

logratioTrim

sumTrim

doWeighting

Acutoff

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (obs, ref, libsize.obs = NULL, libsize.ref = NULL, logratioTrim = 0.3,
    sumTrim = 0.05, doWeighting = TRUE, Acutoff = -1e+10)
{
    if (all(obs == ref))
        return(1)
    obs <- as.numeric(obs)
    ref <- as.numeric(ref)
    if (is.null(libsize.obs))
        nO <- sum(obs)
    else nO <- libsize.obs
    if (is.null(libsize.ref))
        nR <- sum(ref)
    else nR <- libsize.ref
    logR <- log2((obs/nO)/(ref/nR))
    absE <- (log2(obs/nO) + log2(ref/nR))/2
    v <- (nO - obs)/nO/obs + (nR - ref)/nR/ref
    fin <- is.finite(logR) & is.finite(absE) & (absE > Acutoff)
    logR <- logR[fin]
    absE <- absE[fin]
    v <- v[fin]
    n <- length(logR)
    loL <- floor(n * logratioTrim) + 1
    hiL <- n + 1 - loL
    loS <- floor(n * sumTrim) + 1
    hiS <- n + 1 - loS
    keep <- (rank(logR) >= loL & rank(logR) <= hiL) & (rank(absE) >=
        loS & rank(absE) <= hiS)
    if (doWeighting) {
        2^(sum(logR[keep]/v[keep], na.rm = TRUE)/sum(1/v[keep],
            na.rm = TRUE))
    }
    else {
        2^(mean(logR[keep], na.rm = TRUE))
    }
  }
```

`change_colours` *change_colours*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
change_colours(p, palette, type)
```

Arguments

p

palette

type

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (p, palette, type)
{
    n <- nlevels(p$data[[deparse(p$mapping$group)]])
    tryCatch(as.character(palette), error = function(e) stop("be vector",call. = FALSE))
    if (n > length(palette))
        stop("Not enough colours in palette.")
    if (missing(type))
        type <- grep("colour|fill", names(p$layers[[1]]$mapping),
            value = TRUE)[1]
    pal <- function(n) palette[seq_len(n)]
    p + discrete_scale(type, "foo", pal)
  }
```

CV2AUCVC *CV2AUCVC*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
CV2AUCVC(data, cvResolution = 0.005)
```

Arguments

data

cvResolution

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, cvResolution = 0.005)
{
    cv_cutoff <- NULL
    uniform_genes_counts <- NULL
    for (i in seq(0, 1, cvResolution)) {
        cv_cutoff <- c(cv_cutoff, i)
        gene_number <- length(which(data <= i))
        uniform_genes_counts <- c(uniform_genes_counts, gene_number)
    }
    getArea(cv_cutoff, uniform_genes_counts)
  }
```

`estimateSizeFactorsForMatrix` *estimateSizeFactorsForMatrix*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
estimateSizeFactorsForMatrix(data, p = p)
```

Arguments

`data`

`p`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, p = p)
{
    loggeomeans <- rowMeans(.log(data), na.rm = TRUE)
    apply(data, 2, function(cnts) exp(quantile(.log(cnts) - loggeomeans,
        na.rm = TRUE, p = p)))
  }
```

filteredZero *filteredZero*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
filteredZero(data, nonzeroRatio)
```

Arguments

```
data
nonzeroRatio
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, nonzeroRatio)
{
    nozeroCount <- apply(data, 1, function(x) length(which(x !=
        0)))
    geneIndex <- which(nozeroCount >= ncol(data) * nonzeroRatio)
    return(geneIndex)
  }
```

findGenes *findGenes*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
findGenes(g, qlower = NULL, qupper = NULL, pre_ratio = NULL)
```

Arguments

```
g
qlower
qupper
pre_ratio
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (g, qlower = NULL, qupper = NULL, pre_ratio = NULL)
{
    gene_name <- rownames(g)
    g <- unlist(g)
    seen <- which(g >= qlower & g <= qupper)
    counts <- length(seen)
    if (counts >= pre_ratio * length(g)) {
        gene_name
    }
  }
```

---

gatherCors *gatherCors*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
gatherCors(data, cor_method = c("spearman", "pearson", "kendall"),
HG7 = NULL, ERCC = NULL, TN = NULL, TC = NULL, CR = NULL, NR = NULL,
DESeq = NULL, UQ = NULL, TMM = NULL, TU = NULL, GAPDH = NULL,
pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65, rounds = 1e+06)
```

Arguments

data

cor_method

HG7

ERCC

TN

TC

CR

NR

DESeq

UQ

TMM

TU

GAPDH

pre_ratio

lower_trim

upper_trim

rounds

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, cor_method = c("spearman", "pearson", "kendall"),
    HG7 = NULL, ERCC = NULL, TN = NULL, TC = NULL, CR = NULL,
    NR = NULL, DESeq = NULL, UQ = NULL, TMM = NULL, TU = NULL,
    GAPDH = NULL, pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65,
    rounds = 1e+06)
{
    methodsList <- list(HG7 = HG7, ERCC = ERCC, TN = TN, TC = TC,
        CR = CR, NR = NR, DESeq = DESeq, UQ = UQ, TMM = TMM,
        TU = TU, GAPDH = GAPDH)
    specifiedMethods <- methodsList[!unlist(lapply(methodsList,
        is.null))]
    numMethod <- length(specifiedMethods)
    method_range <- seq(1, numMethod, 1)
    ubq_genes <- identifyUbq(data, pre_ratio = pre_ratio, lower_trim = lower_trim,
        upper_trim = upper_trim, min_ubq = 100)
    cor_value_method <- NULL
    for (j in method_range) {
        norm.matrix <- getNormMatrix(data, specifiedMethods[[j]])
        dataUse2Cor <- norm.matrix[ubq_genes, ]
        cor.result <- getCor(dataUse2Cor, method = cor_method,
            rounds = rounds)
        cor_vm <- cbind(cor.result, rep(names(specifiedMethods)[j],
            times = round(rounds)))
        cor_value_method <- rbind(cor_value_method, cor_vm)
    }
    colnames(cor_value_method) <- c("Value", "Methods")
    return(cor_value_method)
  }
```

---

gatherCors4Matrices *gatherCors4Matrices*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
gatherCors4Matrices(..., raw_matrix, cor_method = c("spearman", "pearson", "kendall"),
pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65, rounds = 1e+06)
```

Arguments

...	
raw_matrix	
cor_method	
pre_ratio	
lower_trim	
upper_trim	
rounds	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (..., raw_matrix, cor_method = c("spearman", "pearson",
    "kendall"), pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65,
    rounds = 1e+06)
{
    matrices <- list(...)
    numMethod <- length(matrices)
    method_range <- seq(1, numMethod, 1)
    ubq_genes <- identifyUbq(raw_matrix, pre_ratio = pre_ratio,
        lower_trim = lower_trim, upper_trim = upper_trim, min_ubq = 100)
    cor_value_method <- NULL
    for (j in method_range) {
        dataUse2Cor <- matrices[[j]][ubq_genes, ]
        cor.result <- getCor(dataUse2Cor, method = cor_method,
            rounds = rounds)
        cor_vm <- cbind(cor.result, rep(names(matrices)[j], times = round(rounds)))
        cor_value_method <- rbind(cor_value_method, cor_vm)
    }
    colnames(cor_value_method) <- c("Value", "Methods")
    return(cor_value_method)
  }
```

gatherCVs *gatherCVs*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
gatherCVs(data,nonzeroRatio,HG7,ERCC,TN,TC,CR,NR,
DESeq,UQ,TMM,TU,GAPDH,cvNorm,cvResolution)
```

Arguments

data

nonzeroRatio

HG7

ERCC

TN

TC

CR

NR

DESeq

UQ

TMM

TU

GAPDH

cvNorm

cvResolution

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, nonzeroRatio = NULL, HG7 = NULL, ERCC = NULL,
    TN = NULL, TC = NULL, CR = NULL, NR = NULL, DESeq = NULL,
    UQ = NULL, TMM = NULL, TU = NULL, GAPDH = NULL, cvNorm = TRUE,
    cvResolution = 0.005)
{
    if (is.null(nonzeroRatio)) {
        stop("Please provide nonzeroRatio!")
```

```
    }
    methodsList <- list(HG7 = HG7, ERCC = ERCC, TN = TN, TC = TC,
        CR = CR, NR = NR, DESeq = DESeq, UQ = UQ, TMM = TMM,
        TU = TU, GAPDH = GAPDH)
    specifiedMethods <- methodsList[!unlist(lapply(methodsList,
        is.null))]
    numMethod <- length(specifiedMethods)
    method_range_tmp <- seq(1, numMethod, 1)
    cv_range_tmp <- seq(0, 1, cvResolution)
    method_range_times <- length(cv_range_tmp)
    cv_range_times <- length(method_range_tmp)
    method_range <- rep(method_range_tmp, each = round(method_range_times))
    cv_range <- rep(cv_range_tmp, times = round(cv_range_times))
    nozeroIndex <- filteredZero(data, nonzeroRatio = nonzeroRatio)
    for (j in method_range_tmp) {
        norm.matrix <- getNormMatrix(data, specifiedMethods[[j]])
        dataUse2CV <- norm.matrix[nozeroIndex, ]
        cv.result <- getCV(dataUse2CV, cvNorm = cvNorm)
        assign(paste(names(specifiedMethods)[j], ".cv", sep = ""),
            cv.result)
    }
    cv_uniform <- NULL
    cv_uniform_all <- mapply(function(i, j) {
        cv.result <- paste(names(specifiedMethods)[j], ".cv",
            sep = "")
        gene_number <- length(which(get(cv.result) <= i))
        cv_uniform_row <- c(i, gene_number, names(specifiedMethods)[j])
        rbind(cv_uniform, cv_uniform_row)
    }, cv_range, method_range)
    cv_uniform_all <- t(cv_uniform_all)
    colnames(cv_uniform_all) <- c("Cutoff", "Counts", "Methods")
    return(cv_uniform_all)
}
```

---

gatherCVs4Matrices *gatherCVs4Matrices*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
gatherCVs4Matrices(..., raw_matrix, nonzeroRatio , cvNorm , cvResolution = 0.005)
```

Arguments

```
...
raw_matrix
```

nonzeroRatio

cvNorm

cvResolution

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (..., raw_matrix, nonzeroRatio = NULL, cvNorm = TRUE,
    cvResolution = 0.005)
{
    if (is.null(nonzeroRatio)) {
        stop("Please provide nonzeroRatio!")
    }
    matrices <- list(...)
    matrices_name <- names(matrices)
    numMethod <- length(matrices)
    method_range_tmp <- seq(1, numMethod, 1)
    cv_range_tmp <- seq(0, 1, cvResolution)
    method_range_times <- length(cv_range_tmp)
    cv_range_times <- length(method_range_tmp)
    method_range <- rep(method_range_tmp, each = round(method_range_times))
    cv_range <- rep(cv_range_tmp, times = round(cv_range_times))
    nozeroIndex <- filteredZero(raw_matrix, nonzeroRatio = nonzeroRatio)
    for (j in method_range_tmp) {
        dataUse2CV <- matrices[[j]][nozeroIndex, ]
        cv.result <- getCV(dataUse2CV, cvNorm = cvNorm)
        assign(paste(matrices_name[j], ".cv", sep = ""), cv.result)
    }
    cv_uniform <- NULL
    cv_uniform_all <- mapply(function(i, j) {
        cv.result <- paste(matrices_name[j], ".cv", sep = "")
        gene_number <- length(which(get(cv.result) <= i))
        cv_uniform_row <- c(i, gene_number, matrices_name[j])
        rbind(cv_uniform, cv_uniform_row)
    }, cv_range, method_range)
    cv_uniform_all <- t(cv_uniform_all)
    colnames(cv_uniform_all) <- c("Cutoff", "Counts", "Methods")
    return(cv_uniform_all)
  }
```

---

gatherFactors *gatherFactors*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
gatherFactors(data,
methods = c("HG7", "ERCC", "TN", "TC", "CR", "NR", "DESeq", "UQ", "TMM", "TU"),
HG7.size = NULL, ERCC.size = NULL, TN.size = NULL, TC.size = NULL,
CR.size = NULL, NR.size = NULL, pre_ratio = 0.5,
lower_trim = 0.05, upper_trim = 0.65, min_ubq = 100)
```

Arguments

data

methods

HG7.size

ERCC.size

TN.size

TC.size

CR.size

NR.size

pre_ratio

lower_trim

upper_trim

min_ubq

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, methods = c("HG7", "ERCC", "TN", "TC", "CR",
    "NR", "DESeq", "UQ", "TMM", "TU"), HG7.size = NULL, ERCC.size = NULL,
    TN.size = NULL, TC.size = NULL, CR.size = NULL, NR.size = NULL,
    pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65, min_ubq = 100)
{
    method1 <- as.list(methods)
    numMethod <- length(method1)
    method_range <- seq(1, numMethod, 1)
    for (i in method_range) {
        if (method1[[i]] == "HG7" || method1[[i]] == "ERCC" ||
            method1[[i]] == "TN" || method1[[i]] == "TC" || method1[[i]] ==
            "CR" || method1[[i]] == "NR") {
            size.name <- paste(method1[[i]], ".size", sep = "")
            out.name1 <- paste(method1[[i]], ".factors", sep = "")
            if (is.null(size.name)) {
                stop("Please provide", size.name, "!")
            }
            else {
```

```
                assign(out.name1, getFactors(data, method = "sizefactor",
                  lib.size = get(size.name)))
            }
        }
        if (method1[[i]] == "DESeq" || method1[[i]] == "RLE" ||
            method1[[i]] == "UQ" || method1[[i]] == "TMM") {
            out.name2 <- paste(method1[[i]], ".factors", sep = "")
            assign(out.name2, getFactors(data, method = method1[[i]]))
        }
        if (method1[[i]] == "TU") {
            TU.factors <- getFactors(data, method = "TU", pre_ratio = pre_ratio,
                lower_trim = lower_trim, upper_trim = upper_trim,
                min_ubq = min_ubq)
        }
    }
    factors.list <- NULL
    for (m in methods) {
        m.factors <- paste(m, ".factors", sep = "")
        factors.list <- c(factors.list, m.factors)
    }
    factors.result <- NULL
    for (i in method_range) {
        factors.result <- cbind(factors.result, get(factors.list[i]))
    }
    colnames(factors.result) <- methods
    return(factors.result)
}
```

---

getArea *getArea*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getArea(x, y)
```

Arguments

x

y

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

```
## The function is currently defined as
function (x, y)
{
    x <- x/max(x)
    y <- y/max(y)
    if (!(is.numeric(x) || is.complex(x)) || !(is.numeric(y) ||
        is.complex(y))) {
        stop("Arguments 'x' and 'y' must be real or complex vectors.")
    }
    if (length(x) != length(y)) {
        stop("The length of two input vectors should be equal!")
    }
    m <- length(x)
    n <- 2 * m
    xp <- c(x, x[m:1])
    yp <- c(numeric(m), y[m:1])
    p1 <- sum(xp[1:(n - 1)] * yp[2:n]) + xp[n] * yp[1]
    p2 <- sum(xp[2:n] * yp[1:(n - 1)]) + xp[1] * yp[n]
    return(0.5 * (p1 - p2))
  }
```

---

getAUCVC *getAUCVC*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getAUCVC(data, nonzeroRatio = NULL, cvNorm = TRUE, cvResolution = 0.005)
```

Arguments

data
nonzeroRatio
cvNorm
cvResolution

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, nonzeroRatio = NULL, cvNorm = TRUE, cvResolution = 0.005)
{
    nozeroIndex <- filteredZero(data, nonzeroRatio = nonzeroRatio)
```

```
  dataUse2CV <- data[nozeroIndex, ]
  cv.result <- getCV(dataUse2CV, cvNorm = cvNorm)
  CV2AUCVC(cv.result, cvResolution = cvResolution)
}
```

---

getAUCVCs *getAUCVCs*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getAUCVCs(..., nonzeroRatio = NULL, cvNorm = TRUE, cvResolution = 0.005)
```

Arguments

...	
nonzeroRatio	
cvNorm	
cvResolution	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (..., nonzeroRatio = NULL, cvNorm = TRUE, cvResolution = 0.005)
{
    matrices <- list(...)
    numMethod <- length(matrices)
    method_range <- seq(1, numMethod, 1)
    result <- NULL
    for (i in method_range) {
        AUCVC.result <- getAUCVC(matrices[[i]], nonzeroRatio = nonzeroRatio,
            cvNorm = cvNorm, cvResolution = cvResolution)
        result <- c(result, AUCVC.result)
        names(result)[i] <- names(matrices)[i]
    }
    sorted_AUCVCs <- sort(result, decreasing = TRUE)
    return(sorted_AUCVCs)
  }
```

getCor *getCor*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getCor(data, method = c("spearman", "pearson", "kendall"), rounds = 1e+06)
```

Arguments

data	
method	
rounds	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, method = c("spearman", "pearson", "kendall"),
    rounds = 1e+06)
{
    sp_result <- NULL
    method <- match.arg(method)
    for (i in 1:rounds) {
        rg1 <- sample(1:nrow(data), size = 1)
        rg2 <- sample(1:nrow(data), size = 1)
        while (rg1 == rg2) {
            rg2 <- sample(1:nrow(data), size = 1)
        }
        gene1 <- unlist(data[rg1, ])
        gene2 <- unlist(data[rg2, ])
        sp_value <- cor(gene1, gene2, method = method)
        sp_result <- c(sp_result, sp_value)
    }
    return(sp_result)
  }
```

getCorMedians *getCorMedians*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getCorMedians(data)
```

Arguments

```
data
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data)
{
    if (!is.data.frame(data))
        data <- data.frame(data)
    if (is.factor(data$Value))
        data$Value <- as.numeric(as.character(data$Value))
    sorted_result <- sort(tapply(data$Value, data$Methods, median),
        decreasing = FALSE)
    return(sorted_result)
  }
```

getCV *getCV*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getCV(data, cvNorm = TRUE)
```

Arguments

```
data
cvNorm
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, cvNorm = TRUE)
{
    if (!is.matrix(data))
        data <- as.matrix(data)
    if (cvNorm) {
        rawCV <- apply(data, 1, function(x) {
            sd(log2(x[x != 0]))/mean(log2(x[x != 0]))
        })
        (rawCV - min(rawCV))/(max(rawCV) - min(rawCV))
    }
    else {
        apply(data, 1, function(x) {
            sd(x)/mean(x)
        })
    }
  }
```

---

getFactors *getFactors*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getFactors(data, method = c("sizefactor", "DESeq", "RLE", "UQ", "TMM", "TU"),
lib.size = NULL, pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65, min_ubq = 100)
```

Arguments

data

method

lib.size

pre_ratio

lower_trim

upper_trim

min_ubq

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, method = c("sizefactor", "DESeq", "RLE", "UQ",
    "TMM", "TU"), lib.size = NULL, pre_ratio = 0.5, lower_trim = 0.05,
    upper_trim = 0.65, min_ubq = 100)
{
    if (!is.matrix(data))
        data <- as.matrix(data)
    if (any(is.na(data)))
        stop("NA counts not permitted")
    if (is.null(lib.size))
        libsize <- colSums(data)
    else libsize <- lib.size
    if (any(is.na(libsize)))
        stop("NA libsizes not permitted")
    method <- match.arg(method)
    i <- apply(data <= 0, 1, all)
    if (any(i))
        data <- data[!i, , drop = FALSE]
   f <- switch(method, sizefactor = 1e+06/libsize, DESeq = 1/estimateSizeFactorsForMatrix(data,
        p = 0.5), RLE = calcFactorRLE(data, p = 0.5)/libsize,
        UQ = calcFactorUpperquartile(data, lib.size = libsize,
            p = 0.75), TMM = {
            fq <- calcFactorUpperquartile(data = data, lib.size = libsize,
                p = 0.75)
            refColumn <- which.min(abs(fq - mean(fq)))
            if (length(refColumn) == 0 | refColumn < 1 | refColumn >
                ncol(data)) refColumn <- 1
            f <- rep(NA, ncol(data))
            for (i in 1:ncol(data)) {
                f[i] <- calcFactorWeighted(obs = data[, i], ref = data[,
                 refColumn], libsize.obs = libsize[i], libsize.ref = libsize[refColumn],
                  logratioTrim = 0.3, sumTrim = 0.05, doWeighting = TRUE,
                  Acutoff = -1e+10)
            }
            f
        }, TU = {
            if (!is.data.frame(data)) data <- data.frame(data)
            ubq_genes <- identifyUbq(data, lower_trim = lower_trim,
                upper_trim = upper_trim, pre_ratio = pre_ratio,
                min_ubq = min_ubq)
            ubq_sums <- colSums(data[ubq_genes, ])
            mean(ubq_sums)/ubq_sums
        }, )
    if (method == "RLE" || method == "UQ" || method == "TMM") {
        f <- 1e+06/libsize/f
    }
    norm.factors <- f/exp(mean(base::log(f)))
```

```
  round(norm.factors, digits = 5)
}
```

---

getNormMatrix *getNormMatrix*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
getNormMatrix(data, norm.factors)
```

Arguments

data

norm.factors

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, norm.factors)
{
    data * matrix(rep(norm.factors, dim(data)[1]), nrow = dim(data)[1],
        ncol = length(norm.factors), byrow = T)
  }
```

---

gridAUCVC *gridAUCVC*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
gridAUCVC(data, dataType = c("bk", "sc"), HG7 = NULL, ERCC = NULL, TN = NULL,
TC = NULL, CR = NULL, NR = NULL, DESeq = NULL, UQ = NULL, TMM = NULL, TU = 0,
GAPDH = NULL, nonzeroRatios = c(0.7, 0.8, 0.9, 1), cvNorm = TRUE, cvResolution = 0.005)
```

Arguments

data

dataType

HG7

ERCC

TN

TC

CR

NR

DESeq

UQ

TMM

TU

GAPDH

nonzeroRatios

cvNorm

cvResolution

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, dataType = c("bk", "sc"), HG7 = NULL, ERCC = NULL,
    TN = NULL, TC = NULL, CR = NULL, NR = NULL, DESeq = NULL,
    UQ = NULL, TMM = NULL, TU = 0, GAPDH = NULL, nonzeroRatios = c(0.7,
        0.8, 0.9, 1), cvNorm = TRUE, cvResolution = 0.005)
{
    grid_result <- NULL
    if (length(TU) == 1 && TU == 1) {
        colnames_paraMatrix <- c("nonzeroRatio", "pre_ratio",
            "lower_trim", "upper_trim")
        write.table(t(as.matrix(colnames_paraMatrix)), file = "bestPara.txt",
            sep = "\t", row.names = FALSE, col.names = FALSE)
    }
    for (i in nonzeroRatios) {
        if (dataType == "sc") {
            if ((ncol(data) * i) <= 100) {
                cat("nonzeroRatio:", i, " is too small!\n")
                stop("We suggest that the minimal counts of
                nonzero samples should be greater than 100!")
            }
        }
        result <- nonzeroRatio2AUCVC(data = data, dataType = dataType,
```

```
            HG7 = HG7, ERCC = ERCC, TN = TN, TC = TC, CR = CR,
            NR = NR, DESeq = DESeq, UQ = UQ, TMM = TMM, TU = TU,
            GAPDH = GAPDH, nonzeroRatio = i, cvNorm = cvNorm,
            cvResolution = cvResolution)
        nonzeroM <- matrix(i, 1, 1, TRUE)
        colnames(nonzeroM) <- "NonzeroRatio"
        grid_record <- cbind(nonzeroM, result)
        grid_result <- rbind(grid_result, grid_record)
    }
    return(grid_result)
  }
```

---

gridAUCVC4Matrices *gridAUCVC4Matrices*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
gridAUCVC4Matrices(..., nonzeroRatios = NULL, cvNorm = TRUE, cvResolution = 0.005)
```

Arguments

...	
nonzeroRatios	
cvNorm	
cvResolution	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (..., nonzeroRatios = NULL, cvNorm = TRUE, cvResolution = 0.005)
{
    if (is.null(nonzeroRatios)) {
        stop("Please provide nonzeroRatios!")
    }
    matrices <- list(...)
    numMethod <- length(matrices)
    grid_result <- NULL
    for (i in nonzeroRatios) {
        result.sorted <- getAUCVCs(..., nonzeroRatio = i, cvNorm = cvNorm,
            cvResolution = cvResolution)
        grid_record <- c(i, result.sorted)
```

```
        names(grid_record)[1] <- "NonzeroRatio"
        grid_result <- c(grid_result, names(grid_record), grid_record)
    }
    grid_result2 <- matrix(grid_result, ncol = numMethod + 1,
        byrow = TRUE)
    return(grid_result2)
  }
```

---

identifyUbq *identifyUbq*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
identifyUbq(data, pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65, min_ubq = 100)
```

Arguments

data	
pre_ratio	
lower_trim	
upper_trim	
min_ubq	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, pre_ratio = 0.5, lower_trim = 0.05, upper_trim = 0.65,
    min_ubq = 100)
{
    qlower <- apply(data, 2, function(x) quantile(x[x != 0],
        p = lower_trim))
    qupper <- apply(data, 2, function(x) quantile(x[x != 0],
        p = upper_trim))
    ubq_genes <- NULL
    for (i in 1:nrow(data)) {
        genes_finded <- findGenes(data[i, ], qlower = qlower,
            qupper = qupper, pre_ratio = pre_ratio)
        ubq_genes <- c(ubq_genes, genes_finded)
    }
    if (length(ubq_genes) < min_ubq) {
```

```
        cat("Parameters range", lower_trim, "-", upper_trim,
            "...identified too few ubiquitous genes (", length(ubq_genes),
            "), trying range 5-95  instead", "\n")
        ubq_genes <- identifyUbqRepeat(data, pre_ratioC = pre_ratio,
            lower_trimC = 0.05, upper_trimC = 0.95)
    }
    return(ubq_genes)
  }
```

---

identifyUbqRepeat *identifyUbqRepeat*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
identifyUbqRepeat(data, pre_ratioC = NULL, lower_trimC = NULL, upper_trimC = NULL)
```

Arguments

data	
pre_ratioC	
lower_trimC	
upper_trimC	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, pre_ratioC = NULL, lower_trimC = NULL, upper_trimC = NULL)
{
    qlower <- apply(data, 2, function(x) quantile(x[x != 0],
        p = lower_trimC))
    qupper <- apply(data, 2, function(x) quantile(x[x != 0],
        p = upper_trimC))
    ubq_genes <- NULL
    for (i in 1:nrow(data)) {
        genes_finded <- findGenes(data[i, ], qlower = qlower,
            qupper = qupper, pre_ratio = pre_ratioC)
        ubq_genes <- c(ubq_genes, genes_finded)
    }
    return(ubq_genes)
  }
```

nonzeroRatio2AUCVC *nonzeroRatio2AUCVC*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
nonzeroRatio2AUCVC(data, dataType = c("bk", "sc"),
HG7 = NULL, ERCC = NULL, TN = NULL, TC = NULL, CR = NULL, NR = NULL, DESeq = NULL,
UQ = NULL, TMM = NULL, TU = 0, GAPDH = NULL, nonzeroRatio = NULL, cvNorm = TRUE,
cvResolution = 0.005)
```

Arguments

data

dataType

HG7

ERCC

TN

TC

CR

NR

DESeq

UQ

TMM

TU

GAPDH

nonzeroRatio

cvNorm

cvResolution

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, dataType = c("bk", "sc"), HG7 = NULL, ERCC = NULL,
    TN = NULL, TC = NULL, CR = NULL, NR = NULL, DESeq = NULL,
    UQ = NULL, TMM = NULL, TU = 0, GAPDH = NULL, nonzeroRatio = NULL,
```

```
    cvNorm = TRUE, cvResolution = 0.005)
{
    nozeroIndex <- filteredZero(data, nonzeroRatio = nonzeroRatio)
    methodsList <- list(HG7 = HG7, ERCC = ERCC, TN = TN, TC = TC,
        CR = CR, NR = NR, DESeq = DESeq, UQ = UQ, TMM = TMM,
        TU = TU, GAPDH = GAPDH)
    specifiedMethods <- methodsList[!unlist(lapply(methodsList,
        is.null))]
    if (length(TU) == 1 && TU == 0) {
        specifiedMethods$TU <- NULL
    }
    if (length(TU) == 1 && TU == 1) {
        if (dataType == "bk") {
            optimalPara <- optTU(data, nonzeroRatio = nonzeroRatio,
                pre_ratio_range = c(1, 1), prResolution = 0.1,
                lower_range = c(0.05, 0.4), upper_range = c(0.6,
                  0.95), qResolution = 0.05, min_ubq = 1000,
                cvNorm = cvNorm, cvResolution = cvResolution)
        }
        else {
            optimalPara <- optTU(data, nonzeroRatio = nonzeroRatio,
                pre_ratio_range = c(0.2, 0.6), prResolution = 0.1,
                lower_range = c(0.05, 0.4), upper_range = c(0.6,
                  0.95), qResolution = 0.05, min_ubq = 100, cvNorm = cvNorm,
                cvResolution = cvResolution)
        }
        optimalPara <- as.matrix(optimalPara)
        lower_trim <- optimalPara["lower", 1]
        upper_trim <- optimalPara["upper", 1]
        pre_ratio <- optimalPara["ratio", 1]
        para <- c(nonzeroRatio, pre_ratio, lower_trim, upper_trim)
        names(para)[1] <- "nonzeroRatio"
        paraMatrix <- t(as.matrix(para))
        write.table(paraMatrix, file = "bestPara.txt", sep = "\t",
            row.names = FALSE, col.names = FALSE, append = TRUE)
        TU.factors <- getFactors(data, method = "TU", lower_trim = lower_trim,
            upper_trim = upper_trim, pre_ratio = pre_ratio, min_ubq = 100)
        norm.matrix <- getNormMatrix(data, TU.factors)
        dataUse2CV <- norm.matrix[nozeroIndex, ]
        cv.result <- getCV(dataUse2CV, cvNorm = cvNorm)
        TU.AUCVC <- CV2AUCVC(cv.result, cvResolution = cvResolution)
        specifiedMethods$TU <- NULL
    }
    numMethod <- length(specifiedMethods)
    if (numMethod >= 1) {
        method_range <- seq(1, numMethod, 1)
        for (i in method_range) {
            norm.matrix <- getNormMatrix(data, specifiedMethods[[i]])
            dataUse2CV <- norm.matrix[nozeroIndex, ]
            cv.result <- getCV(dataUse2CV, cvNorm = cvNorm)
            assign(names(specifiedMethods)[i], CV2AUCVC(cv.result,
                cvResolution = cvResolution))
        }
```

```
        AUCVC.result <- NULL
        for (i in method_range) {
            AUCVC.result <- cbind(AUCVC.result, get(names(specifiedMethods)[i]))
        }
        colnames(AUCVC.result) <- names(specifiedMethods)
        if (length(TU) == 1 && TU == 1) {
            AUCVC.result <- cbind(AUCVC.result, TU.AUCVC)
            colnames(AUCVC.result) <- c(names(specifiedMethods),
                "TU")
        }
    }
    if (numMethod == 0 && TU == 0)
        stop("Please specify at least one method!")
    if (numMethod == 0 && TU == 1) {
        AUCVC.result <- as.matrix(TU.AUCVC)
        colnames(AUCVC.result) <- "TU"
    }
    return(AUCVC.result)
}
```

---

optTU *optTU*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
optTU(data, nonzeroRatio = NULL, pre_ratio_range = c(0.2, 0.6), prResolution = 0.1,
lower_range = c(0.05, 0.4), upper_range = c(0.6, 0.95),
qResolution = 0.05, min_ubq = 100, cvNorm = TRUE, cvResolution = 0.005)
```

Arguments

data

nonzeroRatio
pre_ratio_range

prResolution

lower_range

upper_range

qResolution

min_ubq

cvNorm

cvResolution

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, nonzeroRatio = NULL, pre_ratio_range = c(0.2,
    0.6), prResolution = 0.1, lower_range = c(0.05, 0.4), upper_range = c(0.6,
    0.95), qResolution = 0.05, min_ubq = 100, cvNorm = TRUE,
    cvResolution = 0.005)
{
    if (is.null(nonzeroRatio)) {
        stop("Please provide nonzeroRatios!")
    }
    pre_ratio_times <- (pre_ratio_range[2] - pre_ratio_range[1] +
        prResolution) * 10
    lower_times <- (upper_range[2] - upper_range[1] + qResolution)/qResolution
    lower_range_tmp <- rep(seq(lower_range[1], lower_range[2],
        qResolution), each = round(lower_times))
    lower_range2 <- rep(lower_range_tmp, times = round(pre_ratio_times))
    upper_times <- (lower_range[2] - lower_range[1] + qResolution)/qResolution
    upper_range_tmp <- rep(seq(upper_range[1], upper_range[2],
        qResolution), times = round(upper_times))
    upper_range2 <- rep(upper_range_tmp, times = round(pre_ratio_times))
    lower_upper_tmp_len <- length(lower_range_tmp)
    pre_ratio_range2 <- rep(seq(pre_ratio_range[1], pre_ratio_range[2],
        0.1), each = round(lower_upper_tmp_len))
    nozeroIndex <- filteredZero(data, nonzeroRatio = nonzeroRatio)
    all_aucvc <- mapply(function(lower_trim, upper_trim, pre_ratio) {
        factors.TU <- getFactors(data, method = "TU", lower_trim = lower_trim,
            upper_trim = upper_trim, pre_ratio = pre_ratio, min_ubq = min_ubq)
        norm.TU <- getNormMatrix(data, factors.TU)
        dataUse2CV <- norm.TU[nozeroIndex, ]
        cv.TU <- getCV(dataUse2CV, cvNorm = cvNorm)
        TU.AUCVC <- CV2AUCVC(cv.TU, cvResolution = cvResolution)
        return(c(TU.AUCVC = TU.AUCVC, lower = lower_trim, upper = upper_trim,
            ratio = pre_ratio))
    }, lower_range2, upper_range2, pre_ratio_range2)
    all_aucvc2 <- t(all_aucvc)
    max_index <- which(max(all_aucvc2[, "TU.AUCVC"]) == all_aucvc2[,
        "TU.AUCVC"])
    return(all_aucvc2[max_index, ])
  }
```

---

plotCors *plotCors*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
plotCors(data, methods = c("None", "HG7", "ERCC", "TN", "TC", "CR", "NR", "DESeq",
"UQ", "TMM", "TU"), legend.position = c(0.15, 0.56))
```

Arguments

data	
methods	
legend.position	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, methods = c("None", "HG7", "ERCC", "TN", "TC",
    "CR", "NR", "DESeq", "UQ", "TMM", "TU"), legend.position = c(0.15,
    0.56))
{
    if (!is.data.frame(data))
        data <- data.frame(data)
    if (is.factor(data$Value))
        data$Value <- as.numeric(as.character(data$Value))
    data$Methods <- factor(data$Methods, levels = methods, labels = methods)
    change_colours(ggplot(data = data, aes(x = Value, y = ..count../sum(..count..))) +
        geom_freqpoly(aes(group = Methods, color = Methods),
            size = 3, bins = 50) + xlab("Spearman correlation") +
      ylab("Fraction of gene pairs") + theme_bw() + theme(panel.grid.minor = element_blank(),
        axis.title.x = element_text(size = 48), axis.title.y = element_text(size = 48),
        axis.text.x = element_text(size = 38), axis.text.y = element_text(size = 38),
        legend.text = element_text(size = 39), legend.title = element_text(size = 43),
        legend.position = legend.position, legend.background = element_blank(),
        legend.key = element_blank(), legend.key.height = unit(1.8,
            "cm"), plot.margin = unit(c(0.5, 1, 0.5, 0.5), "cm")) +
        scale_x_continuous(expand = c(0.01, 0.01), breaks = round(seq(-1,
            1, 0.25), 2)) + scale_y_continuous(expand = c(0.01,
        0)) + guides(color = guide_legend(title = NULL)), c("olivedrab",
        "blue", "red", "violet", "orange", "yellow", "magenta",
        "peru", "black", "maroon", "lightblue", "darkslateblue",
        "seashell4", "tan2", "darkgreen", "springgreen"))
  }
```

---

plotCVs *plotCVs*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
plotCVs(data, methods = c("None", "HG7", "ERCC", "TN", "TC", "CR", "NR",
"DESeq", "UQ", "TMM", "TU"), legend.position = c(0.85, 0.48))
```

Arguments

data	
methods	
legend.position	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, methods = c("None", "HG7", "ERCC", "TN", "TC",
    "CR", "NR", "DESeq", "UQ", "TMM", "TU"), legend.position = c(0.85,
    0.48))
{
    if (!is.data.frame(data))
        data <- data.frame(data)
    if (is.factor(data$Cutoff))
        data$Cutoff <- as.numeric(as.character(data$Cutoff))
    if (is.factor(data$Counts))
        data$Counts <- as.numeric(as.character(data$Counts))
    data$Methods <- factor(data$Methods, levels = methods, labels = methods)
    change_colours(ggplot(data = data, aes(x = Cutoff, y = Counts)) +
        geom_line(aes(group = Methods, color = Methods), size = 3) +
        xlab("Normalized CV cutoff") + ylab("Number of uniform genes") +
        theme_bw() + theme(panel.grid.minor = element_blank(),
        axis.title.x = element_text(size = 48), axis.title.y = element_text(size = 48),
        axis.text.x = element_text(size = 38), axis.text.y = element_text(size = 38),
        legend.text = element_text(size = 39), legend.title = element_text(size = 43),
        legend.position = legend.position, legend.background = element_blank(),
        legend.key = element_blank(), legend.key.height = unit(1.8,
            "cm"), plot.margin = unit(c(0.5, 0.5, 0.5, 0.5),
            "cm")) + scale_x_continuous(breaks = seq(0, 1, 0.2)) +
        scale_y_continuous() + guides(color = guide_legend(title = NULL)),
        c("olivedrab", "blue", "red", "violet", "orange", "yellow",
            "magenta", "peru", "black", "maroon", "lightblue",
            "darkslateblue", "seashell4", "tan2", "darkgreen",
            "springgreen"))
  }
```

plotHC *plotHC*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
plotHC(data, method = c("spearman", "pearson", "kendall"), mar = c(9, 1, 0, 20))
```

Arguments

data

method

mar

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (data, method = c("spearman", "pearson", "kendall"),
    mar = c(9, 1, 0, 20))
{
    if (!is.data.frame(data))
        data <- data.frame(data)
    method <- match.arg(method)
    hc <- hclust(as.dist(1 - cor(data, method = method)))
    dend <- as.dendrogram(hc)
    dend <- dend %>% set("labels_cex", 6.5) %>% set("branches_lwd",
        6.5)
    par(mar = mar, mgp = c(10, 5, 0), cex.axis = 6)
    plot(dend, horiz = TRUE)
    axis(side = 1, lwd = 8)
  }
```

scRNA663 *scRNA663*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
data("scRNA663")
```

Format

A data frame with 57955 observations on the following 663 variables.

`col36l_1` a numeric vector

`col36l_2` a numeric vector

`col36l_3` a numeric vector

`col36l_4` a numeric vector

`col36l_5` a numeric vector

`col36l_6` a numeric vector

`col36l_7` a numeric vector

`col36l_8` a numeric vector

`col36l_9` a numeric vector

`col36l_10` a numeric vector

`col36l_11` a numeric vector

`col36l_12` a numeric vector

`col36l_13` a numeric vector

`col36l_14` a numeric vector

`col36l_15` a numeric vector

`col36l_16` a numeric vector

`col36l_17` a numeric vector

`col36l_18` a numeric vector

`col36l_19` a numeric vector

`col36l_20` a numeric vector

`col36l_21` a numeric vector

`col36l_22` a numeric vector

`col36l_23` a numeric vector

`col36l_24` a numeric vector

`col36l_25` a numeric vector

`col36l_26` a numeric vector

`col36l_27` a numeric vector

`col36l_28` a numeric vector

`col36l_29` a numeric vector

`col36l_30` a numeric vector

`col36l_31` a numeric vector

`col36l_32` a numeric vector

col36l_33 a numeric vector

col36l_34 a numeric vector

col36l_35 a numeric vector

col36l_36 a numeric vector

col36l_37 a numeric vector

col36l_38 a numeric vector

col36l_39 a numeric vector

col36l_40 a numeric vector

col36l_41 a numeric vector

col36l_42 a numeric vector

col36l_43 a numeric vector

col36l_44 a numeric vector

col36l_45 a numeric vector

col36l_46 a numeric vector

col36l_47 a numeric vector

col36l_48 a numeric vector

col36l_49 a numeric vector

col36l_50 a numeric vector

col36l_51 a numeric vector

col36l_52 a numeric vector

col36l_53 a numeric vector

col36l_54 a numeric vector

col36l_55 a numeric vector

col36l_56 a numeric vector

col36l_57 a numeric vector

col36l_58 a numeric vector

col36l_59 a numeric vector

col36l_60 a numeric vector

col36l_61 a numeric vector

col36l_62 a numeric vector

col36l_63 a numeric vector

col36l_64 a numeric vector

col36l_65 a numeric vector

col36l_66 a numeric vector

col36l_67 a numeric vector

col36l_68 a numeric vector

col36l_69 a numeric vector

col36l_70 a numeric vector

col36l_71 a numeric vector

col38l_1 a numeric vector

col38l_2 a numeric vector

col38l_6 a numeric vector

col38l_7 a numeric vector

col38l_8 a numeric vector

col38l_10 a numeric vector

col38l_11 a numeric vector

col38l_12 a numeric vector

col38l_13 a numeric vector

col38l_14 a numeric vector

col38l_15 a numeric vector

col38l_16 a numeric vector

col38l_17 a numeric vector

col38l_19 a numeric vector

col38l_20 a numeric vector

col38l_21 a numeric vector

col38l_22 a numeric vector

col38l_23 a numeric vector

col38l_24 a numeric vector

col38l_25 a numeric vector

col38l_26 a numeric vector

col38l_27 a numeric vector

col38l_28 a numeric vector

col38l_29 a numeric vector

col38l_30 a numeric vector

col38l_31 a numeric vector

col38l_33 a numeric vector

col38l_34 a numeric vector

col38l_35 a numeric vector

col38l_36 a numeric vector

col38l_37 a numeric vector

col38l_39 a numeric vector

col38l_40 a numeric vector

col38l_41 a numeric vector

col38l_42 a numeric vector

col38l_43 a numeric vector
col38l_46 a numeric vector
col38l_47 a numeric vector
col38l_48 a numeric vector
col38l_49 a numeric vector
col38l_52 a numeric vector
col38l_55 a numeric vector
col38l_56 a numeric vector
col38l_57 a numeric vector
col38l_59 a numeric vector
col38l_60 a numeric vector
col38l_61 a numeric vector
col38l_62 a numeric vector
col38l_64 a numeric vector
col38l_65 a numeric vector
col38l_66 a numeric vector
col38l_67 a numeric vector
col38l_68 a numeric vector
col38l_69 a numeric vector
col38l_70 a numeric vector
col38l_72 a numeric vector
col39l1_47 a numeric vector
col39l1_48 a numeric vector
col39l1_49 a numeric vector
col39l1_50 a numeric vector
col39l1_51 a numeric vector
col39l1_52 a numeric vector
col39l1_53 a numeric vector
col39l1_54 a numeric vector
col39l1_55 a numeric vector
col39l1_56 a numeric vector
col39l1_57 a numeric vector
col39l1_58 a numeric vector
col39l1_59 a numeric vector
col39l1_60 a numeric vector
col39l1_61 a numeric vector
col39l1_62 a numeric vector

col39l1_63 a numeric vector
col39l1_64 a numeric vector
col39l1_69 a numeric vector
col39l1_70 a numeric vector
col39l1_71 a numeric vector
col39l1_72 a numeric vector
col39l1_73 a numeric vector
col39l1_74 a numeric vector
col39l1_75 a numeric vector
col39l1_76 a numeric vector
col39l1_77 a numeric vector
col39l1_78 a numeric vector
col39l1_79 a numeric vector
col39l1_80 a numeric vector
col39l1_81 a numeric vector
col39l1_82 a numeric vector
col39l1_83 a numeric vector
col39l1_84 a numeric vector
col39l1_85 a numeric vector
col39l1_86 a numeric vector
col39l1_87 a numeric vector
col39l1_88 a numeric vector
col39l1_89 a numeric vector
col39l1_90 a numeric vector
col39l1_91 a numeric vector
col39l1_92 a numeric vector
col39l1_93 a numeric vector
col39l1_94 a numeric vector
col39l1_95 a numeric vector
col39l1_96 a numeric vector
col39l2_20 a numeric vector
col39l2_21 a numeric vector
col39l2_22 a numeric vector
col39l2_23 a numeric vector
col39l2_24 a numeric vector
col39l2_25 a numeric vector
col39l2_26 a numeric vector

col39l2_27 a numeric vector
col39l2_28 a numeric vector
col39l2_29 a numeric vector
col39l2_30 a numeric vector
col39l2_31 a numeric vector
col39l2_32 a numeric vector
col39l2_33 a numeric vector
col39l2_34 a numeric vector
col39l2_35 a numeric vector
col39l2_36 a numeric vector
col39l2_37 a numeric vector
col39l2_38 a numeric vector
col39l2_39 a numeric vector
col39l2_40 a numeric vector
col39l2_41 a numeric vector
col39l2_42 a numeric vector
col39l2_43 a numeric vector
col39l2_44 a numeric vector
col39l2_45 a numeric vector
col39l2_46 a numeric vector
col39l2_47 a numeric vector
col39l2_48 a numeric vector
col39l2_49 a numeric vector
col39l2_50 a numeric vector
col39l2_52 a numeric vector
col39l2_53 a numeric vector
col39l2_54 a numeric vector
col39l2_55 a numeric vector
col39l2_56 a numeric vector
col39l2_57 a numeric vector
col39l2_58 a numeric vector
col39l2_59 a numeric vector
col39l2_60 a numeric vector
col39l2_61 a numeric vector
col39l2_62 a numeric vector
col39l2_63 a numeric vector
col39l2_64 a numeric vector

col39l2_65 a numeric vector
col39l2_66 a numeric vector
col39l2_67 a numeric vector
col39l2_68 a numeric vector
col39l2_69 a numeric vector
col39l2_70 a numeric vector
col39l2_72 a numeric vector
col39l2_73 a numeric vector
col39l2_74 a numeric vector
col39l3_31 a numeric vector
col39l3_32 a numeric vector
col39l3_33 a numeric vector
col39l3_34 a numeric vector
col39l3_35 a numeric vector
col39l3_36 a numeric vector
col39l3_38 a numeric vector
col39l3_39 a numeric vector
col39l3_40 a numeric vector
col39l3_41 a numeric vector
col39l3_42 a numeric vector
col39l3_43 a numeric vector
col39l3_44 a numeric vector
col39l3_45 a numeric vector
col39l3_46 a numeric vector
col39l3_47 a numeric vector
col39l3_48 a numeric vector
col39l3_49 a numeric vector
col39l3_50 a numeric vector
col39l3_51 a numeric vector
col39l3_52 a numeric vector
col39l3_53 a numeric vector
col39l3_54 a numeric vector
col39l3_55 a numeric vector
col39l3_56 a numeric vector
col39l3_57 a numeric vector
col39l3_58 a numeric vector
col39l3_59 a numeric vector

col39l3_60 a numeric vector
col39l3_61 a numeric vector
col39l3_62 a numeric vector
col39l3_63 a numeric vector
col39l3_64 a numeric vector
col39l3_65 a numeric vector
col39l3_66 a numeric vector
col39l3_67 a numeric vector
col39l3_68 a numeric vector
col39l3_69 a numeric vector
col39l3_70 a numeric vector
col39l3_71 a numeric vector
col39l3_72 a numeric vector
col39l3_73 a numeric vector
col39l3_74 a numeric vector
col39l3_75 a numeric vector
col39l3_76 a numeric vector
col39l3_77 a numeric vector
col39l3_78 a numeric vector
col39l3_79 a numeric vector
col39l3_80 a numeric vector
col39l3_81 a numeric vector
col39l3_82 a numeric vector
col39l3_83 a numeric vector
col39l3_85 a numeric vector
col40l_1 a numeric vector
col40l_2 a numeric vector
col40l_3 a numeric vector
col40l_4 a numeric vector
col40l_5 a numeric vector
col40l_6 a numeric vector
col40l_7 a numeric vector
col40l_8 a numeric vector
col40l_9 a numeric vector
col40l_10 a numeric vector
col40l_11 a numeric vector
col40l_12 a numeric vector

col40l_13 a numeric vector
col40l_14 a numeric vector
col40l_15 a numeric vector
col40l_16 a numeric vector
col40l_17 a numeric vector
col40l_18 a numeric vector
col40l_19 a numeric vector
col40l_20 a numeric vector
col40l_21 a numeric vector
col40l_22 a numeric vector
col40l_23 a numeric vector
col40l_24 a numeric vector
col40l_25 a numeric vector
col40l_26 a numeric vector
col40l_27 a numeric vector
col40l_28 a numeric vector
col40l_29 a numeric vector
col40l_30 a numeric vector
col40l_31 a numeric vector
col40l_32 a numeric vector
col40l_33 a numeric vector
col40l_34 a numeric vector
col40l_35 a numeric vector
col40l_37 a numeric vector
col40l_38 a numeric vector
col40l_39 a numeric vector
col40l_40 a numeric vector
col40l_41 a numeric vector
col40l_42 a numeric vector
col40l_44 a numeric vector
col40l_45 a numeric vector
col40l_46 a numeric vector
col40l_47 a numeric vector
col40l_48 a numeric vector
col40l_49 a numeric vector
col40l_50 a numeric vector
col44l1_1 a numeric vector

col44l1_2 a numeric vector

col44l1_3 a numeric vector

col44l1_4 a numeric vector

col44l1_8 a numeric vector

col44l1_11 a numeric vector

col44l1_12 a numeric vector

col44l1_13 a numeric vector

col44l1_14 a numeric vector

col44l1_15 a numeric vector

col44l1_16 a numeric vector

col44l1_17 a numeric vector

col44l1_18 a numeric vector

col44l1_19 a numeric vector

col44l1_20 a numeric vector

col44l1_24 a numeric vector

col44l1_28 a numeric vector

col44l1_29 a numeric vector

col44l1_30 a numeric vector

col44l1_32 a numeric vector

col44l1_33 a numeric vector

col44l1_36 a numeric vector

col44l1_38 a numeric vector

col44l1_40 a numeric vector

col44l1_41 a numeric vector

col44l1_42 a numeric vector

col44l1_43 a numeric vector

col44l1_47 a numeric vector

col44l1_48 a numeric vector

col44l1_50 a numeric vector

col44l1_53 a numeric vector

col44l1_58 a numeric vector

col44l1_59 a numeric vector

col44l1_60 a numeric vector

col44l1_64 a numeric vector

col44l1_66 a numeric vector

col44l1_67 a numeric vector

col44l1_68 a numeric vector

col44l1_69 a numeric vector
col44l1_70 a numeric vector
col44l1_71 a numeric vector
col44l1_73 a numeric vector
col44l1_74 a numeric vector
col44l1_75 a numeric vector
col44l1_76 a numeric vector
col44l1_77 a numeric vector
col44l1_79 a numeric vector
col44l1_80 a numeric vector
col44l1_82 a numeric vector
col44l1_83 a numeric vector
col44l1_84 a numeric vector
col44l1_85 a numeric vector
col44l1_86 a numeric vector
col44l1_87 a numeric vector
col44l1_89 a numeric vector
col44l1_90 a numeric vector
col44l1_91 a numeric vector
col44l1_92 a numeric vector
col44l1_93 a numeric vector
col44l1_94 a numeric vector
col44l2_1 a numeric vector
col44l2_2 a numeric vector
col44l2_3 a numeric vector
col44l2_4 a numeric vector
col44l2_5 a numeric vector
col44l2_6 a numeric vector
col44l2_9 a numeric vector
col44l2_10 a numeric vector
col44l2_12 a numeric vector
col44l2_14 a numeric vector
col44l2_17 a numeric vector
col44l2_18 a numeric vector
col44l2_19 a numeric vector
col44l2_20 a numeric vector
col44l2_23 a numeric vector

col44l2_26 a numeric vector
col44l2_27 a numeric vector
col44l2_28 a numeric vector
col44l2_30 a numeric vector
col44l2_31 a numeric vector
col44l2_32 a numeric vector
col44l2_35 a numeric vector
col44l2_36 a numeric vector
col44l2_37 a numeric vector
col44l2_38 a numeric vector
col44l2_40 a numeric vector
col44l7_67 a numeric vector
col44l7_68 a numeric vector
col44l7_69 a numeric vector
col44l7_70 a numeric vector
col44l7_71 a numeric vector
col44l7_75 a numeric vector
col44l7_76 a numeric vector
col44l7_77 a numeric vector
col44l7_78 a numeric vector
col44l7_79 a numeric vector
col44l7_80 a numeric vector
col44l7_82 a numeric vector
col44l7_83 a numeric vector
col44l7_84 a numeric vector
col44l7_85 a numeric vector
col44l7_86 a numeric vector
col44l7_87 a numeric vector
col44l7_88 a numeric vector
col44l7_89 a numeric vector
col44l7_90 a numeric vector
col44l7_91 a numeric vector
col44l7_92 a numeric vector
col44l7_93 a numeric vector
col44l7_94 a numeric vector
col44l7_95 a numeric vector
col44l7_96 a numeric vector

col44l8_17 a numeric vector
col44l8_19 a numeric vector
col44l8_21 a numeric vector
col44l8_22 a numeric vector
col44l8_23 a numeric vector
col44l8_24 a numeric vector
col44l8_25 a numeric vector
col44l8_26 a numeric vector
col44l8_28 a numeric vector
col44l8_29 a numeric vector
col44l8_31 a numeric vector
col44l8_32 a numeric vector
col44l8_33 a numeric vector
col44l8_34 a numeric vector
col44l8_43 a numeric vector
col44l8_45 a numeric vector
col44l8_46 a numeric vector
col44l8_47 a numeric vector
col44l8_48 a numeric vector
col44l8_50 a numeric vector
col44l8_51 a numeric vector
col44l8_52 a numeric vector
col44l8_53 a numeric vector
col44l8_54 a numeric vector
col44l8_55 a numeric vector
col44l8_56 a numeric vector
col44l8_58 a numeric vector
col44l8_59 a numeric vector
col44l8_60 a numeric vector
col44l8_61 a numeric vector
col44l8_62 a numeric vector
col44l8_63 a numeric vector
col44l8_67 a numeric vector
col44l8_68 a numeric vector
col44l8_70 a numeric vector
col44l8_71 a numeric vector
col44l8_72 a numeric vector

col44l8_73 a numeric vector
col44l8_74 a numeric vector
col44l8_76 a numeric vector
col44l8_78 a numeric vector
col44l8_81 a numeric vector
col44l8_85 a numeric vector
col44l8_86 a numeric vector
col44l8_88 a numeric vector
col44l8_94 a numeric vector
col45l2_42 a numeric vector
col45l2_43 a numeric vector
col45l2_45 a numeric vector
col45l2_47 a numeric vector
col45l2_48 a numeric vector
col45l2_50 a numeric vector
col45l2_51 a numeric vector
col45l2_52 a numeric vector
col45l2_54 a numeric vector
col45l2_55 a numeric vector
col45l2_56 a numeric vector
col45l2_57 a numeric vector
col45l2_58 a numeric vector
col45l2_59 a numeric vector
col45l2_60 a numeric vector
col45l2_61 a numeric vector
col45l2_62 a numeric vector
col45l2_63 a numeric vector
col45l2_64 a numeric vector
col45l2_65 a numeric vector
col45l2_66 a numeric vector
col45l2_67 a numeric vector
col45l2_68 a numeric vector
col45l2_69 a numeric vector
col45l2_70 a numeric vector
col45l2_71 a numeric vector
col45l2_76 a numeric vector
col45l2_77 a numeric vector

col45l2_78 a numeric vector
col45l2_81 a numeric vector
col45l2_83 a numeric vector
col45l2_84 a numeric vector
col45l2_85 a numeric vector
col45l2_88 a numeric vector
col45l2_89 a numeric vector
col45l2_90 a numeric vector
col45l2_91 a numeric vector
col45l2_92 a numeric vector
col45l2_93 a numeric vector
col45l2_94 a numeric vector
col45l2_95 a numeric vector
col45l7_24 a numeric vector
col45l7_25 a numeric vector
col45l8_2 a numeric vector
col45l8_3 a numeric vector
col45l8_4 a numeric vector
col45l8_5 a numeric vector
col45l8_6 a numeric vector
col45l8_7 a numeric vector
col45l8_8 a numeric vector
col45l8_9 a numeric vector
col45l8_11 a numeric vector
col45l8_12 a numeric vector
col45l8_13 a numeric vector
col45l8_14 a numeric vector
col45l8_35 a numeric vector
col47l7_26 a numeric vector
col47l7_27 a numeric vector
col47l7_28 a numeric vector
col47l7_29 a numeric vector
col47l7_30 a numeric vector
col47l7_31 a numeric vector
col47l7_32 a numeric vector
col47l7_33 a numeric vector
col47l7_34 a numeric vector

col47l7_35 a numeric vector
col47l7_36 a numeric vector
col47l7_37 a numeric vector
col47l7_38 a numeric vector
col47l7_41 a numeric vector
col47l7_42 a numeric vector
col47l7_44 a numeric vector
col47l7_45 a numeric vector
col47l7_47 a numeric vector
col47l7_48 a numeric vector
col47l7_49 a numeric vector
col47l7_50 a numeric vector
col47l7_51 a numeric vector
col47l7_54 a numeric vector
col47l7_57 a numeric vector
col47l7_58 a numeric vector
col47l7_59 a numeric vector
col47l7_60 a numeric vector
col47l7_63 a numeric vector
col47l7_64 a numeric vector
col47l7_65 a numeric vector
col47l7_66 a numeric vector
col48l6_2 a numeric vector
col48l6_4 a numeric vector
col48l6_5 a numeric vector
col48l6_6 a numeric vector
col48l6_7 a numeric vector
col48l6_8 a numeric vector
col48l6_9 a numeric vector
col48l6_10 a numeric vector
col48l6_11 a numeric vector
col48l6_12 a numeric vector
col48l6_13 a numeric vector
col48l6_14 a numeric vector
col48l6_16 a numeric vector
col48l6_17 a numeric vector
col48l6_18 a numeric vector

col48l6_19 a numeric vector
col48l6_20 a numeric vector
col48l6_21 a numeric vector
col48l6_22 a numeric vector
col48l6_24 a numeric vector
col48l6_26 a numeric vector
col48l6_27 a numeric vector
col48l6_28 a numeric vector
col48l6_30 a numeric vector
col48l6_31 a numeric vector
col48l6_34 a numeric vector
col48l6_35 a numeric vector
col48l6_36 a numeric vector
col48l6_39 a numeric vector
col48l6_40 a numeric vector
col48l6_42 a numeric vector
col48l6_43 a numeric vector
col48l6_44 a numeric vector
col48l6_46 a numeric vector
col48l6_47 a numeric vector
col48l6_48 a numeric vector
col48l6_50 a numeric vector
col48l6_51 a numeric vector
col48l6_52 a numeric vector
col48l6_53 a numeric vector
col48l6_54 a numeric vector
col48l6_55 a numeric vector
col48l6_56 a numeric vector
col48l6_58 a numeric vector
col48l6_60 a numeric vector
col48l6_61 a numeric vector
col48l6_62 a numeric vector
col48l6_63 a numeric vector
col48l6_64 a numeric vector
col48l6_65 a numeric vector
col48l6_66 a numeric vector
col48l6_67 a numeric vector

col48l6_68 a numeric vector
col48l6_69 a numeric vector
col48l6_70 a numeric vector
col48l6_71 a numeric vector
col48l6_72 a numeric vector
col48l6_73 a numeric vector
col48l6_74 a numeric vector
col48l6_75 a numeric vector
col48l6_76 a numeric vector
col48l6_77 a numeric vector
col48l6_78 a numeric vector
col48l6_79 a numeric vector
col48l6_80 a numeric vector
col48l6_81 a numeric vector
col48l6_82 a numeric vector
col48l6_85 a numeric vector
col48l6_86 a numeric vector
col48l6_88 a numeric vector
col48l6_90 a numeric vector
col48l6_92 a numeric vector
col48l6_93 a numeric vector
col48l6_94 a numeric vector
col48l6_95 a numeric vector
col48l6_96 a numeric vector
col48l7_1 a numeric vector
col48l7_2 a numeric vector
col48l7_3 a numeric vector
col48l7_4 a numeric vector
col48l7_5 a numeric vector
col48l7_7 a numeric vector
col48l7_9 a numeric vector
col48l7_11 a numeric vector
col48l7_12 a numeric vector
col48l7_13 a numeric vector
col48l7_15 a numeric vector
col48l7_16 a numeric vector
col48l7_18 a numeric vector
col48l7_20 a numeric vector
col48l7_21 a numeric vector

Examples

```
data(scRNA663)
## maybe str(scRNA663) ; plot(scRNA663) ...
```

---

scRNA663_factors *scRNA663_factors*

Description

Please refer to the file /inst/doc/readme.pdf.

Usage

```
data("scRNA663_factors")
```

Format

A data frame with 663 observations on the following 12 variables.

`HG7` a numeric vector

`ERCC` a numeric vector

`TN` a numeric vector

`TC` a numeric vector

`CR` a numeric vector

`NR` a numeric vector

`DESeq` a numeric vector

`UQ` a numeric vector

`TMM` a numeric vector

`TU` a numeric vector

`NCS` a numeric vector

`ES` a numeric vector

Examples

```
data(scRNA663_factors)
## maybe str(scRNA663_factors) ; plot(scRNA663_factors) ...
```

Index