

Package ‘IRISFGM’

October 28, 2022

Type Package

Title Comprehensive Analysis of Gene Interactivity Networks Based on Single-Cell RNA-Seq

Version 1.5.0

Date 2020-12-22

Description Single-cell RNA-Seq data is useful in discovering cell heterogeneity and signature genes in specific cell populations in cancer and other complex diseases. Specifically, the investigation of functional gene modules (FGM) can help to understand gene interactive networks and complex biological processes. QUBIC2 is recognized as one of the most efficient and effective tools for FGM identification from scRNA-Seq data. However, its availability is limited to a C implementation, and its applicative power is affected by only a few downstream analyses functionalities. We developed an R package named IRIS-FGM (integrative scRNA-Seq interpretation system for functional gene module analysis) to support the investigation of FGMs and cell clustering using scRNA-Seq data. Empowered by QUBIC2, IRIS-FGM can identify co-expressed and co-regulated FGMs, predict types/clusters, identify differentially expressed genes, and perform functional enrichment analysis. It is noteworthy that IRIS-FGM also applies Seurat objects that can be easily used in the Seurat vignettes.

License GPL-2

Imports Rcpp (>= 1.0.0), MCL, anocva, Polychrome, RColorBrewer, colorspace, AnnotationDbi, ggplot2, org.Hs.eg.db, org.Mm.eg.db, pheatmap, AdaptGauss, DEsingle, DrImpute, Matrix, Seurat, SingleCellExperiment, clusterProfiler, ggpubr, ggraph, igraph, mixtools, scatter, scran, stats, methods, grDevices, graphics, utils, knitr

LinkingTo Rcpp

VignetteBuilder knitr

RoxygenNote 7.1.1

Encoding UTF-8

Collate 'Classes.R' 'generics.R' 'AddMeta.R' 'Bicluster.R' 'Bric.R' 'CellTypePrediction.R' 'DifferentialGene.R' 'DimensionReductionBasedOnLTMG.R' 'EnrichPathway.R' 'InputData.R' 'LTMGSCA.R' 'LTMG.R' 'Object.R' 'PlotHeatmap.R' 'PlotNetwork.R' 'PreprocessData.R' 'RcppExports.R' 'data.R'

biocViews Software, GeneExpression, SingleCell, Clustering, DifferentialExpression, Preprocessing, DimensionReduction, Visualization, Normalization, DataImport

Depends R (>= 4.1)

Suggests rmarkdown

git_url <https://git.bioconductor.org/packages/IRISFGM>

git_branch master

git_last_commit e54746b

git_last_commit_date 2022-04-26

Date/Publication 2022-10-28

Author Yuzhou Chang [aut, cre],
Qin Ma [aut],
Carter Allen [aut],
Dongjun Chung [aut]

Maintainer Yuzhou Chang <yuzhou.chang@osumc.edu>

R topics documented:

| | |
|---|----|
| .generateNetObject | 3 |
| .getBlock | 4 |
| .IDConvert | 5 |
| .runBiclustBaseOnDiscretization | 5 |
| .runBiclustBaseOnLTMG | 6 |
| .RunGO | 7 |
| .RunKEGG | 7 |
| .separateBic | 8 |
| AddMeta | 8 |
| BIC_LTMG | 9 |
| BIC_ZIMG | 9 |
| CalBinaryMultiSignal | 10 |
| CalBinarySingleSignal | 11 |
| CLUSTERING | 11 |
| CreateIRISFGMObject | 12 |
| DimReduce-class | 13 |
| DotPlotPathway | 13 |
| example_object | 14 |
| FindClassBasedOnMC | 14 |
| FindGlobalMarkers | 15 |
| FindMarker | 17 |
| Fit_LTMG | 18 |
| GetBinaryMultiSignal | 18 |
| GetBinarySingleSignal | 19 |
| GetLTMGmatrix | 20 |
| getMeta | 20 |
| Global_Zcut | 21 |

GRAPH 21

InverseMillsRatio 22

IRISFGM-class 22

KS_LTMG 23

KS_ZIMG 23

LogSeparateKRpkmNew 24

LogSeparateKRpkmNewLR 24

LTMG 25

MCL 25

MINUS 26

MIN_return 26

Pi_Zj_Zcut_new 27

PlotDimension 27

PlotHeatmap 28

PlotMarkerHeatmap 29

PlotMeta 30

PlotModuleNetwork 31

PlotNetwork 32

ProcessData 33

Pure_CDF 34

qubic 35

ReadFrom10X_folder 36

ReadFrom10X_h5 36

RunBicluster 37

RunClassification 38

RunDimensionReduction 40

RunDiscretization 41

RunLTMG 42

RunPathway 43

SC 44

SeparateKRpkmNew 45

SeparateKRpkmNew2 45

SeparateKRpkmNewLR 46

SeparateKRpkmNewLRPlus 46

SeparateKRpkmNewp 47

State_return 48

SubsetData 48

Index **50**

| | |
|---------------------------------|---|
| <code>.generateNetObject</code> | <i>GenerateNetObject Generate Net Object for the network use.</i> |
|---------------------------------|---|

Description

GenerateNetObject Generate Net Object for the network use.

Usage

```
.generateNetObject(object, N.bicluster = c(1, 5), method = "spearman")
```

Arguments

| | |
|-------------|---|
| object | Input IRIS-FGM object. |
| N.bicluster | Should be two integers indicating the number of two biclusters. |
| method | Should be a statistical method to calculate edge weight based on expression data. It can be either "Spearman" (default) or "Pearson." |

Value

get generateNetObject

| | |
|------------------|--------------------------|
| <i>.getBlock</i> | <i>getblock function</i> |
|------------------|--------------------------|

Description

get block from generated temporal files

Usage

```
.getBlock(object = NULL, keyword = "Conds")
```

Arguments

| | |
|---------|---|
| object | IRISFGM object |
| keyword | 'Conds' for co-regulatory or 'Genes' for co-expression gene |

Value

get blocks from biclustering results

```
.IDConvert          convert ID convert ID
```

Description

convert ID convert ID

Usage

```
.IDConvert(genes.use = NULL, species = NULL)
```

Arguments

genes.use Provide gene list
species which species is in the provided gene list

Value

converted ID

```
.runBiclusterBaseOnDiscretization  
                          Run Discretization Generate temporal discretized file
```

Description

Run Discretization Generate temporal discretized file

Usage

```
.runBiclusterBaseOnDiscretization(  
  object = NULL,  
  OpenDual = TRUE,  
  Extension = 1,  
  NumBlockOutput = 100,  
  BlockOverlap = 0.7,  
  BlockCellMin = 15  
)
```

Arguments

| | |
|----------------|---|
| object | input IRISFMG object |
| OpenDual | the flag using the lower bound of condition number. Default: 5 percent of the gene number in current bicluster. |
| Extension | consistency level of the block (0.5-1.0], the minimum ratio between the number of identical valid symbols in a column and the total number of rows in the output. Default: 1.0. |
| NumBlockOutput | number of blocks to report. Default: 100. |
| BlockOverlap | filtering overlapping blocks. Default: 0.7. |
| BlockCellMin | minimum column width of the block. Default: 15 columns. |

Value

It will perform discretization

`.runBiclusterBaseOnLTMG`

RunBicusterBaseOnLTMG

Description

Run bicluster based on LTMG

Usage

```
.runBiclusterBaseOnLTMG(
  object = NULL,
  OpenDual = FALSE,
  Extension = 1,
  NumBlockOutput = 100,
  BlockOverlap = 0.9,
  BlockCellMin = 15
)
```

Arguments

| | |
|----------------|---|
| object | input IRISFMG object |
| OpenDual | the flag using the lower bound of condition number. Default: 5 percent of the gene number in current bicluster. |
| Extension | consistency level of the block (0.5-1.0], the minimum ratio between the number of identical valid symbols in a column and the total number of rows in the output. Default: 1.0. |
| NumBlockOutput | number of blocks to report. Default: 100. |
| BlockOverlap | filtering overlapping blocks. Default: 0.7. |
| BlockCellMin | minimum column width of the block. Default: 15 columns. |

Value

It will return Biclustering results based on LTMG.

.RunGO *RunGO RunGO*

Description

RunGO RunGO

Usage

```
.RunGO(genes.use = NULL, species = "mouse")
```

Arguments

| | |
|-----------|--|
| genes.use | Provide gene list |
| species | which species is in the provided gene list |

Value

GO pathway enrichment analysis

.RunKEGG *RunKEGG RunKEGG*

Description

RunKEGG RunKEGG

Usage

```
.RunKEGG(genes.use = NULL, species = "mouse")
```

Arguments

| | |
|-----------|--|
| genes.use | Provide gene list |
| species | which species is in the provided gene list |

Value

KEGG results

| | |
|---------------------------|--|
| <code>.separateBic</code> | <i>separateBic separate biclusters</i> |
|---------------------------|--|

Description

`separateBic separate biclusters`

Usage

```
.separateBic(object = NULL)
```

Arguments

| | |
|---------------------|------------------------|
| <code>object</code> | Input IRIS-FGM object. |
|---------------------|------------------------|

Value

It will return a list for shoring gene and cell.

| | |
|---------|----------------|
| AddMeta | <i>AddMeta</i> |
|---------|----------------|

Description

This function can import cell annotation information to IRIS-FGM object.

Usage

```
AddMeta(object, ...)

.addMeta(object = NULL, meta.info = NULL)

## S4 method for signature 'IRISFGM'
AddMeta(object = NULL, meta.info = NULL)
```

Arguments

| | |
|------------------------|---|
| <code>object</code> | input IRIS-FGM object |
| <code>...</code> | other arguments passed to methods |
| <code>meta.info</code> | meta information table should be a data frame with rows representing cell and column representing different group condition |

Value

It will add meta information to IRISFGM.

Examples

```
x <- matrix(rnorm(100),ncol = 10)
colnames(x) <- paste0("cell",1:ncol(x))
rownames(x) <- paste0("gene",1:nrow(x))
my_meta <- data.frame(row.names = paste0("cell",1:ncol(x)),
cluster = c(rep("ClusterA",5),rep("ClusterB",5)))
object <- CreateIRISFGMObject(x)
object <- AddMeta(object,
meta.info = my_meta)
```

BIC_LTMG

BIC_LTMG BIC_LTMG

Description

BIC_LTMG BIC_LTMG

Usage

BIC_LTMG(y, rrr, Zcut)

Arguments

| | |
|------|----------------|
| y | input y |
| rrr | input vector |
| Zcut | input global z |

Value

BIC_LTMG

BIC_ZIMG

BIC_ZIMG fits different model BIC_ZIMG fits different model

Description

BIC_ZIMG fits different model BIC_ZIMG fits different model

Usage

BIC_ZIMG(y, rrr, Zcut)

Arguments

| | |
|------|--------------|
| y | input vector |
| rrr | input vector |
| Zcut | global zcut |

Value

BIC_ZIMG

CalBinaryMultiSignal *CalBinaryMultiSignal*

Description

This function is for calculating multisignal from LTMG signaling matrix.

Usage

```
CalBinaryMultiSignal(object)

.CalBinaryMultiSignal(object = NULL)

## S4 method for signature 'IRISFGM'
CalBinaryMultiSignal(object = NULL)
```

Arguments

| | |
|--------|----------------|
| object | Input IRIS-FGM |
|--------|----------------|

Value

It will return a binary matrix based on LTMG signal matrix.

Examples

```
data("example_object")
example_object <- CalBinaryMultiSignal(example_object)
```

CalBinarySingleSignal *CalBinarySingleSignal*

Description

CalBinarySingleSignal

CalBinarySingleSignal

Binarizing single signal function via distinguishing zero or non-zero value based on LTMG matrix

Usage

```
CalBinarySingleSignal(object)
```

```
.CalBinarySingleSignal(object = NULL)
```

```
## S4 method for signature 'IRISFGM'
```

```
CalBinarySingleSignal(object = NULL)
```

Arguments

object Input IRIS-FGM object

Value

It will return a binary matrix based on LTMG signal matrix.

Examples

```
data("example_object")
example_object <- CalBinarySingleSignal(example_object)
```

CLUSTERING

Packaging clustering method This function is used for choosing clustering method

Description

Packaging clustering method This function is used for choosing clustering method

Usage

```
CLUSTERING(Raw, blocks, method = "MCL", K = NULL)
```

Arguments

| | |
|--------|--|
| Raw | input raw discretized data which is chars file |
| blocks | input block identified by IRISFGM |
| method | chosse method, either MCL or SC. |
| K | number of cluster |

Value

clustering results

CreateIRISFGMObject *CreateIRISFGMObject*

Description

Create IRIS-FGM object

Arguments

| | |
|------------|--|
| x | Input expression matrix which should be a matrix or dataframe. |
| min.cell | each gene should be expressed by at least this many cell. |
| min.gene | each cell should express this many gene at least. |
| LTMGr | Automatically create LTMG object. |
| Biccluster | Automatically create Biccluster object. |

Details

CreateIRISFGMObject

Value

it should return a IRISFGM object of which structure can be found in tutorial.

Examples

```
x <- matrix(rnorm(100),ncol = 10)
colnames(x) <- paste0("cell",1:ncol(x))
rownames(x) <- paste0("gene",1:nrow(x))
object <- CreateIRISFGMObject(x)
```

| | |
|-----------------|--------------------------------|
| DimReduce-class | <i>Create DimReduce object</i> |
|-----------------|--------------------------------|

Description

Create DimReduce object

Slots

PCA ANY.
 UMAP ANY.
 TSNE ANY.

| | |
|----------------|-----------------------|
| DotPlotPathway | <i>DotPlotPathway</i> |
|----------------|-----------------------|

Description

Plot dotplot for enrichment pathway

Usage

```
DotPlotPathway(object, ...)

.dotPlotPathway(
  object = NULL,
  genes.source = c("CTS", "MC", "Bicluster"),
  showCategory = 20
)

## S4 method for signature 'IRISFGM'
DotPlotPathway(
  object = NULL,
  genes.source = c("CTS", "MC", "Bicluster"),
  showCategory = 20
)
```

Arguments

| | |
|--------------|---|
| object | Input IRIS-FGM object |
| ... | other arguments passed to methods |
| genes.source | Decide the plot source either "CTS", "MC" or "Bicluster." "CTS" means DEGs from DEsingle label, "MC" means DEGs from MC label, and "Bicluster" means using gene module from the selected bicluster. |
| showCategory | Show this number of pathway results. |

Value

This function will generate dot plot for pathway enrichment results.

Examples

```
data("example_object")
DotPlotPathway(example_object, genes.source = "module" )
```

| | |
|----------------|------------------------|
| example_object | <i>Example object.</i> |
|----------------|------------------------|

Description

The example data was pre-generated IRISFGM object and it was generated from the partial data from Yan's data which contain 90 human embryo cells.

Usage

```
data(example_object)
```

Format

A data frame with 970 rows (gene) and 90 columns(cell):

Raw_count slot for the original data

Processed_count slot for the preprocessed data which was generated by normalization or imputation ...

Source

https://bmb1.bmi.osumc.edu/downloadFiles/Yan_expression.txt

| | |
|--------------------|---------------------------|
| FindClassBasedOnMC | <i>FindClassBasedOnMC</i> |
|--------------------|---------------------------|

Description

This function is for performing Markov chain clustering regarding generated co-expression gene modules. This clustering method is working for relative small dataset. If you have a large dataset, We recommend you should use Seurat clustering wrapped in this IRISFGM package. See details [RunLTMG](#), [RunDimensionReduction](#), and [RunClassification](#).

Usage

```
FindClassBasedOnMC(object, ...)

.final(object = NULL, method = "MCL", K = 5)

## S4 method for signature 'IRISFGM'
FindClassBasedOnMC(object = NULL, method = "MCL", K = 5)
```

Arguments

| | |
|--------|---|
| object | input IRIS-FGM object |
| ... | other arguments passed to methods |
| method | using MCL(Markov Cluster) algorithm to predict clusters. There is alternative option which is 'SC.' (Unnormalized spectral clustering function. Uses Partitioning Around Medoids clustering instead of K-means.) |
| K | expected number of predicted clusters when you are using 'SC' method for cell clustering and this parameter does not work for 'MCL' |

Value

It will return cell clustering results based on MCL method.

Examples

```
data(example_object)
example_object<- RunLTMG(example_object,Gene_use = "200")
example_object <- CalBinaryMultiSignal(example_object)
# Due to generate intermedie files, please make sure to set working directory

example_object <- RunBiclustere(example_object,
                               DiscretizationModel = 'LTMG',
                               OpenDual = FALSE,
                               NumBlockOutput = 1000,
                               BlockOverlap = 0.7,
                               BlockCellMin = 15)
example_object <- FindClassBasedOnMC(example_object)
```

FindGlobalMarkers

FindGlobalMarkers

Description

FindGlobalMarkers

This function is for finding global marker FindGlobalMarkers is based on Seurat FindAllMarkers and the data from Tmp.seurat slots.

Usage

```
FindGlobalMarkers(object, ...)

.findglobalMarkers(
  object = NULL,
  idents = NULL,
  logfc.threshold = 0.25,
  test.use = "wilcox",
  only.pos = TRUE,
  random.seed = 1,
  min.pct = 0.1
)

## S4 method for signature 'IRISFGM'
FindGlobalMarkers(
  object = NULL,
  idents = NULL,
  logfc.threshold = 0.25,
  test.use = "wilcox",
  only.pos = TRUE,
  random.seed = 1,
  min.pct = 0.1
)
```

Arguments

| | |
|------------------------------|---|
| <code>object</code> | input IRIS-FGM object |
| <code>...</code> | other arguments passed to methods |
| <code>idents</code> | choose an idents for labelling cells |
| <code>logfc.threshold</code> | Limit testing to genes which show, on average, at least X-fold difference (log-scale) between the two groups of cells. Default is 0.25 Increasing <code>logfc.threshold</code> speeds up the function, but can miss weaker signals. |
| <code>test.use</code> | same as FindAllMarkers |
| <code>only.pos</code> | keep postive result |
| <code>random.seed</code> | set seed for reproducibility |
| <code>min.pct</code> | only test genes that are detected in a minimum fraction of <code>min.pct</code> cells in either of the two populations. Meant to speed up the function by not testing genes that are very infrequently expressed. Default is 0.1 |

Value

Output is a differentially expressed gene list

Examples

```
data("example_object")
markers <- FindGlobalMarkers(
  object = example_object,
  idents = "Seurat_r_0.8_k_20")
```

FindMarker

FindMarker

Description

FindMarker

Find marker based on DEsingle method Find marker based on DEsingle method

Usage

```
FindMarker(object, ...)

.findMarker(object, SimpleResult = TRUE, FDR = 0.05)

## S4 method for signature 'IRISFGM'
FindMarker(object, SimpleResult = TRUE, FDR = 0.05)
```

Arguments

| | |
|--------------|---|
| object | input IRISFGM object |
| ... | other arguments passed to methods |
| SimpleResult | marker gene only output log fold change (LFC), p-value, and adjusted p-value. |
| FDR | a number to specify the threshold of FDR, default by 0.05 |

Value

It will return differentially expressed gene based on DEsingle method.

Examples

```
data(example_object)
# It is an interactive function which requires user to provide the preferred cell labels.
# example_object <- FindMarker(example_object)
```

| | |
|----------|--------------------------|
| Fit_LTMG | <i>Fit_LTMG Fit_LTMG</i> |
|----------|--------------------------|

Description

Fit_LTMG Fit_LTMG

Usage

Fit_LTMG(x, n, q, k, err = 1e-10)

Arguments

| | |
|-----|--------------|
| x | input x |
| n | input n |
| q | input q |
| k | input k |
| err | random error |

Value

Fit_LTMG

| | |
|----------------------|-----------------------------|
| GetBinaryMultiSignal | <i>GetBinaryMultiSignal</i> |
|----------------------|-----------------------------|

Description

GetBinaryMultiSignal

GetBinaryMultiSignal

This function is for getting multisignal from LTMG signaling matrix.

Usage

GetBinaryMultiSignal(object, ...)

.GetBinaryMultiSignal(object = NULL)

```
## S4 method for signature 'IRISFGM'
GetBinaryMultiSignal(object = NULL)
```

Arguments

| | |
|--------|-----------------------------------|
| object | Input IRIS-FGM |
| ... | other arguments passed to methods |

Value

It will get a binary matrix based on LTMG signal matrix.

Examples

```
data(example_object)
Multisignal_matrix <- GetBinaryMultiSignal(example_object)
```

`GetBinarySingleSignal` *GetBinarySingleSignal*

Description

`GetBinarySingleSignal`

`GetBinarySingleSignal` Get binary Single Signal matrix

Usage

```
GetBinarySingleSignal(object, ...)

.GetBinarySingleSignal(object = NULL)

## S4 method for signature 'IRISFGM'
GetBinarySingleSignal(object = NULL)
```

Arguments

| | |
|---------------------|-----------------------------------|
| <code>object</code> | Input IRIS-FGM object |
| <code>...</code> | other arguments passed to methods |

Value

It will export the Binarized matrix based on LTMG signal matrix.

Examples

```
data(example_object)
SingleSignal_matrix <- GetBinarySingleSignal(example_object)
```

GetLTMGmatrix *GetLTMGmatrix*

Description

Get LTMG matrix

Usage

```
GetLTMGmatrix(object, ...)  
  
.GetLTMGmatrix(object)  
  
## S4 method for signature 'IRISFGM'  
GetLTMGmatrix(object)
```

Arguments

| | |
|--------|-----------------------------------|
| object | Input IRIS-FGM object |
| ... | other arguments passed to methods |

Value

It will return LTMG signal matrix.

Examples

```
data(example_object)  
LTMG_signalmatrix <- GetLTMGmatrix(example_object)
```

getMeta *getMeta*

Description

Obtain meta information from IRISFGM object

Usage

```
getMeta(object)  
  
.getmeta(object)  
  
## S4 method for signature 'IRISFGM'  
getMeta(object)
```

Arguments

object input IRISFGM object

Value

this function will return the meta information for IRISFGM object.

Examples

```
data(example_object)
meta_infor <- getMeta(example_object)
```

| | |
|-------------|--|
| Global_Zcut | <i>Global_Zcut create Zcut Global_Zcut create Zcut</i> |
|-------------|--|

Description

Global_Zcut create Zcut Global_Zcut create Zcut

Usage

Global_Zcut(MAT)

Arguments

MAT input matrix

Value

global_zcut

| | |
|-------|-----------------------|
| GRAPH | <i>generate graph</i> |
|-------|-----------------------|

Description

generate graph

Usage

GRAPH(blocks)

Arguments

blocks input blocks

Value

create a graph

| | |
|-------------------|--------------------|
| InverseMillsRatio | <i>Title Title</i> |
|-------------------|--------------------|

Description

Title Title

Usage

InverseMillsRatio(q, mean, sd)

Arguments

| | |
|------|------|
| q | q |
| mean | mean |
| sd | sd |

Value

InverseMillsRatio

| | |
|---------------|----------------|
| IRISFGM-class | <i>IRISFGM</i> |
|---------------|----------------|

Description

IRISFGM

Slots

Raw_count ANY
 Processed_count ANY
 MetaInfo ANY
 Discretization matrix.
 LTMG LTMGr.
 BiCluster Bicluster

| | |
|---------|------------------------|
| KS_LTMG | <i>KS_LTMG KS_LTMG</i> |
|---------|------------------------|

Description

KS_LTMG KS_LTMG

Usage

KS_LTMG(y, rrr, Zcut)

Arguments

| | |
|------|-------------------|
| y | input y |
| rrr | input vector |
| Zcut | input global zcut |

Value

KS_LTMG

| | |
|---------|------------------------|
| KS_ZIMG | <i>KS_ZIMG KS_ZIMG</i> |
|---------|------------------------|

Description

KS_ZIMG KS_ZIMG

Usage

KS_ZIMG(y, rrr, Zcut)

Arguments

| | |
|------|------------|
| y | input y |
| rrr | input rrr |
| Zcut | input Zcut |

Value

KS_ZIMG

LogSeparateKRpkmNew *Title Title*

Description

Title Title

Usage

LogSeparateKRpkmNew(x, n, q, k, err = 1e-10)

Arguments

| | |
|-----|-----|
| x | x |
| n | n |
| q | q |
| k | k |
| err | err |

Value

LogSeparateKRpkmNew

LogSeparateKRpkmNewLR *LogSeparateKRpkmNewLR*

Description

LogSeparateKRpkmNewLR

Usage

LogSeparateKRpkmNewLR(x, n, q, r, k = 2)

Arguments

| | |
|---|---|
| x | data, a List of NumericVectors |
| n | rounds |
| q | cutoff of the elements in x |
| r | maximum value of the standard diversion |
| k | number of peaks, should be 2 |

Value

LogSeparateKRpkmNewLR

LTMG

LTMG LTMG

Description

LTMG LTMG

Usage

LTMG(VEC, Zcut_G, k = 5)

Arguments

VEC input vector
Zcut_G input Zcut
k input k as gene regulatory signal

Value

return LTMG

MCL

MCL clustering MCL clustering

Description

MCL clustering MCL clustering

Usage

MCL(Raw, blocks)

Arguments

Raw input data
blocks input blocks

Value

MCL clustering results

MINUS

MINUS MINUS

Description

MINUS MINUS

Usage

MINUS(x, y)

Arguments

x input x

y input y

ValueMINUS

MIN_return

MIN_return MIN_return

Description

MIN_return MIN_return

Usage

MIN_return(x)

Arguments

x input vector

Value

MIN_return

| | | |
|----------------|-----------------------|-----------------------|
| Pi_Zj_Zcut_new | <i>Pi_Zj_Zcut_new</i> | <i>Pi_Zj_Zcut_new</i> |
|----------------|-----------------------|-----------------------|

Description

Pi_Zj_Zcut_new Pi_Zj_Zcut_new

Usage

```
Pi_Zj_Zcut_new(q, mean, sd, w10)
```

Arguments

| | |
|------|------|
| q | q |
| mean | mean |
| sd | sd |
| w10 | w10 |

Value

Pi_Zj_Zcut_new

| | |
|---------------|----------------------|
| PlotDimension | <i>PlotDimension</i> |
|---------------|----------------------|

Description

Generate Umap and it requires user to input cell label index on console window.

Usage

```
PlotDimension(object, ...)

.plotDimension(object, reduction = "umap", pt_size = 1, idents = NULL)

## S4 method for signature 'IRISFGM'
PlotDimension(object, reduction = "umap", pt_size = 1, idents = NULL)
```

Arguments

| | |
|-----------|--|
| object | Input IRIS-FGM Object |
| ... | other arguments passed to methods |
| reduction | Choose one of approaches for dimension reduction, including 'pca', 'tsne', 'umap'. |
| pt_size | Point size, default is 0. |
| idents | set current idents. |

Value

generate plot on umap space.

Examples

```
data("example_object")
PlotDimension(example_object,idents = "Seurat_r_0.8_k_20")
```

 PlotHeatmap

PlotHeatmap

Description

PlotHeatmap

plot heatmap based on bicluster

Usage

```
PlotHeatmap(object, ...)
```

```
.plotHeatmap(
  object = object,
  N.bicluster = c(1, 5),
  show.overlap = FALSE,
  show.annotation = FALSE,
  show.clusters = FALSE
)
```

```
## S4 method for signature 'IRISFGM'
```

```
PlotHeatmap(
  object = object,
  N.bicluster = c(1, 5),
  show.overlap = FALSE,
  show.annotation = FALSE,
  show.clusters = FALSE
)
```

Arguments

| | |
|-----------------|--|
| object | Input IRISFGM object |
| ... | other arguments passed to methods |
| N.bicluster | Number of biclusters. |
| show.overlap | Parameter (logic) indicates whether the figure shows the overlap part between two selected biclusters. |
| show.annotation | Parameter (logic) indicates whether to show annotation (biclusters number and cell cluster labels). |
| show.clusters | Parameter (logic) indicates whether to show the cell cluster label. |

Value

It will generate a heatmap based on selected two FGMs.

Examples

```
data(example_object)
PlotHeatmap(example_object,
N.biclust = c(1,20),
show.annotation = TRUE,
show.cluster =TRUE)
```

| | |
|-------------------|--|
| PlotMarkerHeatmap | <i>PlotMarkerHeatmap will visualize global marker This function will generate global marker gene heatmap</i> |
|-------------------|--|

Description

PlotMarkerHeatmap will visualize global marker This function will generate global marker gene heatmap

Usage

```
PlotMarkerHeatmap(
  Globalmarkers = NULL,
  object = NULL,
  idents = NULL,
  top.gene = 50,
  p.adj = 0.05,
  scale = "row",
  label.size = 1
)
```

Arguments

| | |
|---------------|--|
| Globalmarkers | output from FindGlobalMarkers |
| object | input IRISFGM object |
| idents | set current idents |
| top.gene | this number of genes will be used for generating heatmap |
| p.adj | adjusted pvalue cutoff for gene selection threshold |
| scale | character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default is "row" if symm false, and "none" otherwise. |
| label.size | Number to decide label size |

Value

heatmap

Examples

```
data("example_object")
markers <- FindGlobalMarkers(
  object = example_object,
  idents = "Seurat_r_0.8_k_20")
```

```
PlotMarkerHeatmap(
  Globalmarkers = markers,
  object = example_object,
  idents = "Seurat_r_0.8_k_20")
```

PlotMeta

PlotMeta

Description

This function can plot figure based on numebr total count information and this step is for the quality control. we shoud exclude extreme value in data.

Usage

```
PlotMeta(object, ...)

.plotMeta(object = NULL)

## S4 method for signature 'IRISFGM'
PlotMeta(object = NULL)
```

Arguments

| | |
|--------|-----------------------------------|
| object | input IRIS-FGM object |
| ... | other arguments passed to methods |

Value

It will generate two violin plots regarding number of RNA count and identified gene number.

Examples

```
x <- matrix(rnorm(100),ncol = 10)
colnames(x) <- paste0("cell",1:ncol(x))
rownames(x) <- paste0("gene",1:nrow(x))
my_meta <- data.frame(row.names = paste0("cell",1:ncol(x)),
cluster = c(rep("ClusterA",5),rep("ClusterB",5)))
object <- CreateIRISFGMObject(x)
object <- AddMeta(object,
meta.info = my_meta)
PlotMeta(object)
```

PlotModuleNetwork *PlotModuleNetwork*

Description

This function will visualize co-expression gene network based on selected two biclusters. The nodes represent the gene module network from the selected bicluster. The size of the nodes indicates the degree of presence. The thickness of edges indicates the value of the correlation coefficient.

Usage

```
PlotModuleNetwork(object, ...)

.plotmodulenetwork(
  object = NULL,
  method = "spearman",
  node.col = "orange",
  N.bicluster = c(1, 5),
  cutoff.neg = -0.8,
  cutoff.pos = 0.8,
  layout = "circle",
  node.label = TRUE,
  node.label.cex = 1
)

## S4 method for signature 'IRISFGM'
PlotModuleNetwork(
  object = NULL,
  method = "spearman",
  node.col = "orange",
  N.bicluster = c(1, 5),
  cutoff.neg = -0.8,
  cutoff.pos = 0.8,
  layout = "circle",
  node.label = TRUE,
  node.label.cex = 1
)
```

Arguments

| | |
|----------------|--|
| object | Input IRIS-FGM object. |
| ... | other arguments passed to methods |
| method | Should be a statistical method to calculate edge weight based on expression data. It can be either "Spearman" (default) or "Pearson." |
| node.col | Should be a color name (or color code) for nodes |
| N.bicluster | Should be the two numbers of biclusters. |
| cutoff.neg | Should be a cutoff to show a negative correlation between two nodes (default: -0.8). |
| cutoff.pos | Should be a cutoff to show a positive correlation between two nodes (default: 0.8). |
| layout | Should be one type of layouts to show nodes' arrangement, including 'linear', 'star', 'circle'(default), 'gem', 'dh', 'graphopt', 'grid', 'mds', 'randomly', 'fr', 'kk', 'drl', 'lgl'. |
| node.label | Should be logic to show the nodes' label (default: TRUE). |
| node.label.cex | Should be a number to control the label size. |

Value

It will generate co-expression network based on selected bicluster (can be one or multiple.)

Examples

```
data("example_object")
PlotModuleNetwork(object = example_object,
method = "spearman",
node.col = "black",
N.bicluster = c(1, 5),
cutoff.neg = -0.8,
cutoff.pos = 0.8,
layout = "circle",
node.label = TRUE,
node.label.cex = 1)
```

PlotNetwork

PlotNetwork

Description

PlotNetwork

PlotNetwork This function is to plot the network for selected biclusters to show the genes (or cells) overlapping relations.

Usage

```
PlotNetwork(object, ...)

## S4 method for signature 'IRISFGM'
PlotNetwork(
  object,
  edge.by = "gene",
  lay.out = "linear",
  N.bicluster = seq_len(20)
)
```

Arguments

| | |
|-------------|--|
| object | Input IRIS-FGM object. |
| ... | other arguments passed to methods |
| edge.by | Should be "cell" or by "gene," indicating nodes label. The default value is by "gene." |
| lay.out | Should be one type of layouts to show nodes' arrangement, including 'linear'(default), 'star', 'circle', 'gem', 'dh', 'graphopt', 'grid', 'mds', 'randomly', 'fr', 'kk', 'drl', 'lgl'. |
| N.bicluster | Should be two integers indicating the number of two biclusters. |

Value

It will generate a global network regarding overlapping genes or cells.

Examples

```
data(example_object)
PlotNetwork(example_object,
  N.bicluster =c(1:20))
```

 ProcessData

ProcessData

Description

Process data via normalization and imputation.

Usage

```
ProcessData(object, ...)

.processData(
  object = NULL,
  normalization = "cpm",
```

```

    library.size = 1e+05,
    IsImputation = FALSE
  )

  ## S4 method for signature 'IRISFGM'
  ProcessData(
    object = NULL,
    normalization = "cpm",
    library.size = 1e+05,
    IsImputation = FALSE
  )

```

Arguments

| | |
|----------------------------|--|
| <code>object</code> | Input IRISFGM object |
| <code>...</code> | other arguments passed to methods |
| <code>normalization</code> | two options: (1)library size normalization by using library size factor: 1e6, equal to CPM (count-per-million-reads), or (2) using 'scran' normalization method. |
| <code>library.size</code> | Sets the scale factor for cpm normalization |
| <code>IsImputation</code> | imputation method is provided by DrImpute. Default is FALSE. |

Value

It will processdata by normalization and imputation.

Examples

```

data("example_object")
example_object <- ProcessData(example_object,
normalization = "cpm")

```

Pure_CDF

Pure_CDF Pure_CDF

Description

Pure_CDF Pure_CDF

Usage

Pure_CDF(Vec)

Arguments

| | |
|------------------|--------------|
| <code>Vec</code> | input vector |
|------------------|--------------|

Value

Pure_CDF

| | |
|-------|--------------|
| qubic | <i>qubic</i> |
|-------|--------------|

Description

QUBIC Performs a QUalitative BIClustering.

Usage

```
qubic(  
  i = NULL,  
  N = FALSE,  
  R = FALSE,  
  Fa = FALSE,  
  d = FALSE,  
  D = FALSE,  
  C = FALSE,  
  n = FALSE,  
  q = 0.05,  
  f = 0.85,  
  k = 13,  
  c = 0.9,  
  o = 100  
)
```

Arguments

| | |
|----|-------|
| i | input |
| N | index |
| R | index |
| Fa | index |
| d | index |
| D | index |
| C | index |
| n | index |
| q | index |
| f | index |
| k | index |
| c | index |
| o | index |

Value

qubic results

ReadFrom10X_folder *ReadFrom10X_folder*

Description

This function provide a method for reading in a folder from 10X platform. In this folder, it should contain three files: barcode, matrix, and gene.

Usage

```
ReadFrom10X_folder(input.dir = NULL)

ReadFrom10X_folder(input.dir = NULL)

## S4 method for signature 'IRISFGM'
ReadFrom10X_folder(input.dir = NULL)
```

Arguments

| | |
|-----------|--|
| input.dir | Input directory. It should contain three file, including barcode file, feature file, and sparse counts matrix. |
| input | input data |

Value

The output from [Read10X](#)

ReadFrom10X_h5 *ReadFrom10X_h5*

Description

This function provide a method for reading in HDF5 file from 10X platform.

Usage

```
ReadFrom10X_h5(input = NULL, use.names = TRUE, unique.features = TRUE)

ReadFrom10X_h5(input = NULL, use.names = TRUE, unique.features = TRUE)

## S4 method for signature 'IRISFGM'
ReadFrom10X_h5(input = NULL, use.names = TRUE, unique.features = TRUE)
```

Arguments

| | |
|-----------------|--------------------------------|
| input | Input an HDF5 object |
| use.names | Use barcode, default is true |
| unique.features | use gene name, default is true |

Value

The output from [Read10X_h5](#)

It will return a gene expression matrix.

RunBicluster

RunBicluster

Description

This function will identify the Biclusters based on LTMG or Quantile normalization

Usage

```
RunBicluster(object, ...)  
  
.runBicluster(  
  object = NULL,  
  DiscretizationModel = "Quantile",  
  OpenDual = FALSE,  
  Extension = 1,  
  NumBlockOutput = 100,  
  BlockOverlap = 0.7,  
  BlockCellMin = 15  
)  
  
## S4 method for signature 'IRISFGM'  
RunBicluster(  
  object = NULL,  
  DiscretizationModel = "Quantile",  
  OpenDual = FALSE,  
  Extension = 1,  
  NumBlockOutput = 100,  
  BlockOverlap = 0.7,  
  BlockCellMin = 15  
)
```

Arguments

| | |
|---------------------|---|
| object | input IRIS-FGM object |
| ... | other arguments passed to methods |
| DiscretizationModel | use different discretization method, including 'Quantile' and 'LTMG.' |
| OpenDual | the flag using the lower bound of condition number. Default: 5 percent of the gene number in current bicluster. |
| Extension | consistency level of the block (0.5-1.0], the minimum ratio between the number of identical valid symbols in a column and the total number of rows in the output. Default: 1.0. |
| NumBlockOutput | number of blocks to report. Default: 100. |
| BlockOverlap | filtering overlapping blocks. Default: 0.7. |
| BlockCellMin | minimum column width of the block. Default: 15 columns. |

Value

It will generate a temporal file on local directory for processed data named 'tmp_expression.txt', discretized file named 'tmp_expression.txt.chars', and bicluster block named 'tmp_expression.txt.chars.block'.

Examples

```
# based on LTMG discretization
data("example_object")
example_object<- RunLTMG(example_object,Gene_use = "200")
example_object <- CalBinaryMultiSignal(example_object)
# Due to generate intermedie files, please make sure to set working directory

example_object <- RunBicluster(example_object,
                              DiscretizationModel = 'LTMG',
                              OpenDual = FALSE,
                              NumBlockOutput = 1000,
                              BlockOverlap = 0.7,
                              BlockCellMin = 15)
```

RunClassification *RunClassification*

Description

This function is based on Seurat package.

Usage

```
RunClassification(object, ...)

.runClassification(
  object,
  dims = seq_len(15),
  k.param = 20,
  resolution = 0.6,
  algorithm = 1
)

## S4 method for signature 'IRISFGM'
RunClassification(
  object,
  dims = seq_len(15),
  k.param = 20,
  resolution = 0.6,
  algorithm = 1
)
```

Arguments

| | |
|------------|--|
| object | input IRIS-FGM object. |
| ... | other arguments passed to methods |
| dims | Dimensions of reduction to use as input. |
| k.param | Defines k for the k-nearest neighbor algorithm. |
| resolution | Value of the resolution parameter, use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities. |
| algorithm | Algorithm for modularity optimization (1 = original Louvain algorithm; 2 = Louvain algorithm with multilevel refinement; 3 = SLM algorithm; 4 = Leiden algorithm). Leiden requires the leidenalg python. |

Value

It will generate cell type information.

Examples

```
data(example_object)
example_object <- RunClassification(example_object,
  dims = 1:15,
  k.param = 20,
  resolution = 0.6,
  algorithm = 1)
```

RunDimensionReduction *RunDimensionReduction*

Description

This function is based on the Seurat package to perform dimension reduction. The input matrix is the LTMG signaling matrix.

Usage

```
RunDimensionReduction(object, ...)  
  
.runDimensionReduction(  
  object,  
  mat.source = c("LTMG", "UMImatrix"),  
  reduction = "umap",  
  dims = seq_len(15),  
  perplexity = 15,  
  seed = 1  
)  
  
## S4 method for signature 'IRISFGM'  
RunDimensionReduction(  
  object,  
  mat.source = c("LTMG", "UMImatrix"),  
  reduction = "umap",  
  dims = seq_len(15),  
  perplexity = 15,  
  seed = 1  
)
```

Arguments

| | |
|------------|---|
| object | Input IRIS-FGM object. |
| ... | other arguments passed to methods |
| mat.source | choose source data for running this function either from LTMG signal matrix or from processed data. Values of this parameter are 'LTMG' and 'UMImatrix' |
| reduction | select a method for dimension reduction, including umap, tsne, and pca. |
| dims | select the number of PCs from PCA results to perform the following dimension reduction and cell clustering. |
| perplexity | Perplexity parameter as optimal number of neighbors. |
| seed | Set the seed of R's random number generator, which is useful for creating simulations or random objects that can be reproduced. |

Value

This function will generate pca, tsne, or umap dimension reduction results.

Examples

```
data(example_object)
example_obejct <- RunDimensionReduction(example_object,
  mat.source= 'LTMG',
  reduction = 'umap',
  dims = 1:15 ,
  perplexity = 15,
  seed = 1)
```

| | |
|-------------------|--------------------------|
| RunDiscretization | <i>RunDiscretization</i> |
|-------------------|--------------------------|

Description

Run discretization based on Quantile method

Usage

```
RunDiscretization(object, ...)

.runDiscretization(object = NULL, q = 0.06)

## S4 method for signature 'IRISFGM'
RunDiscretization(object = NULL, q = 0.06)
```

Arguments

| | |
|--------|--|
| object | input IRIS-FGM object |
| ... | other arguments passed to methods |
| q | quantile number which is used as discretized cutoff. The bigger q means more cells will be categorized into 1 in terms of binarizing one gene. |

Value

It will generate quantile based binary matrix.

Examples

```
data(example_object)
# Due to generate intermediate files, please make sure to set working directory

example_object <- RunDiscretization(example_object, q = 0.06)
```

RunLTMG

RunLTMG

Description

We will use Left-truncated Mixture Gaussian distribution to model the regulatory signal of each gene. Parameter, 'Gene_use', decides number of top high variant gene for LTMG modeling, and here we use all genes.

Usage

```
RunLTMG(object, ...)  
  
.RunLTMG(object, Gene_use = NULL, k = 5)  
  
## S4 method for signature 'IRISFGM'  
RunLTMG(object, Gene_use = NULL, k = 5)
```

Arguments

| | |
|----------|---|
| object | Input IRIS-FGM object |
| ... | other arguments passed to methods |
| Gene_use | using X numebr of top variant gene. input a number, recommend 2000. |
| k | Number of components. |

Value

it will return a LTMG signal matrix

Examples

```
# If you want to explore DEG, we recommend you should use top 2000 highly variant gene.  
data("example_object")  
example_object <- RunLTMG(example_object,  
Gene_use = "200",  
k = 5)
```

RunPathway

*RunPathway***Description**

This function will perform enrichment analysis based on a gene module or identified differentially expressed genes (DEG). This function is also depended on clusterProfiler, AnnotationDbi, org.Mm.eg.db, and org.Hs.eg.db package.

Usage

```
RunPathway(object, ...)

.runPathway(
  object = NULL,
  N.bicluster = c(10, 11, 12, 13),
  selected.gene.cutoff = 0.05,
  species = "Human",
  database = "GO",
  genes.source = c("CTS", "Bicluster")
)

## S4 method for signature 'IRISFGM'
RunPathway(
  object = NULL,
  N.bicluster = c(10, 11, 12, 13),
  selected.gene.cutoff = 0.05,
  species = "Human",
  database = "GO",
  genes.source = c("CTS", "Bicluster")
)
```

Arguments

| | |
|----------------------|--|
| object | Input IRIS-FGM object |
| ... | other arguments passed to methods |
| N.bicluster | Select the numebr of bicluster to perform this function. |
| selected.gene.cutoff | Set up a statistical significance cutoff for all identified DEGs. |
| species | You can choose either 'Human' or 'Mouse' |
| database | You can choose either 'GO' or 'KEGG' database |
| genes.source | You can choose a gene list source, either 'CTS' or 'Bicluster.' 'CTS' means from cell-type-specific DEGs, and 'Bicluster means using gene module from the selected bicluster.' |

Value

It will return a function enrichment analysis.

Examples

```
# To run this function based on the gene module from an identified bicluster use:
data("example_object")
# due to execute time for this function, please use the function below

object <- RunPathway(object = example_object,
  N.bicluster = 4,
  selected.gene.cutoff = 0.05,
  species = 'Human', database = 'GO', genes.source = 'Bicluster')
```

SC

spectral Clustering method spectral Clustering method

Description

spectral Clustering method spectral Clustering method

Usage

```
SC(Raw, blocks, K)
```

Arguments

| | |
|--------|--|
| Raw | input |
| blocks | input blocks |
| K | number of clusters; this parameter is used for SC clustering method. |

Value

spectral Clustering results

SeparateKRpkmNew *SeparateKRpkmNew SeparateKRpkmNew*

Description

SeparateKRpkmNew SeparateKRpkmNew

Usage

SeparateKRpkmNew(x, n, q, k, err = 1e-10)

Arguments

| | |
|-----|----------------------------------|
| x | data, example: x<-runif(100,0,1) |
| n | rounds |
| q | cutoff |
| k | k=1..5 |
| err | err |

Value

a matrix contains pi, mean and sd

SeparateKRpkmNew2 *SeparateKRpkmNew2 SeparateKRpkmNew2*

Description

SeparateKRpkmNew2 SeparateKRpkmNew2

Usage

SeparateKRpkmNew2(x, n, q, err = 1e-10)

Arguments

| | |
|-----|-----|
| x | x |
| n | n |
| q | q |
| err | err |

Value

SeparateKRpkmNew2

SeparateKRpkmNewLR *SeparateKRpkmNewLRPlus Calculate the LTMG_2LR for some genes
Calculate the LTMG_2LR for some genes*

Description

SeparateKRpkmNewLRPlus Calculate the LTMG_2LR for some genes Calculate the LTMG_2LR for some genes

Usage

SeparateKRpkmNewLR(x, n, q, r, s = 0.05, k = 2, err = 1e-10, M = Inf, m = -Inf)

Arguments

| | |
|-----|---|
| x | data, a List of NumericVector |
| n | rounds |
| q | cutoff of the elements in x |
| r | maximum value of the standard diversion |
| s | minimum value of the standard diversionzz |
| k | number of peaks, should be 2 |
| err | the upper bound on the absolute error |
| M | set to positive infinity |
| m | set to negative infinity |

Value

a matrix contains pi, mean and sd

SeparateKRpkmNewLRPlus *SeparateKRpkmNewLRPlus Calculate the LTMG_2LR for some genes
Calculate the LTMG_2LR for some genes*

Description

SeparateKRpkmNewLRPlus Calculate the LTMG_2LR for some genes Calculate the LTMG_2LR for some genes

Usage

```

SeparateKRpkmNewLRPlus(
  x,
  n,
  q,
  r,
  s = 0.05,
  k = 2,
  err = 1e-10,
  M = Inf,
  m = -Inf
)

```

Arguments

| | |
|-----|---|
| x | data, a List of NumericVector |
| n | rounds |
| q | cutoff of the elements in x |
| r | maximum value of the standard diversion |
| s | minimum value of the standard diversionzz |
| k | number of peaks, should be 2 |
| err | the upper bound on the absolute error |
| M | set to positive infinity |
| m | set to negative infinity |

Value

a matrix contains pi, mean and sd

| | |
|-------------------|--------------------|
| SeparateKRpkmNewp | <i>Title Title</i> |
|-------------------|--------------------|

Description

Title Title

Usage

```
SeparateKRpkmNewp(x, n, q, k, err = 1e-10)
```

Arguments

| | |
|-----|-----|
| x | x |
| n | n |
| q | q |
| k | k |
| err | err |

Value

SeparateKRpkmNewp

| | |
|--------------|----------------------------------|
| State_return | <i>State_return State_return</i> |
|--------------|----------------------------------|

Description

State_return State_return

Usage

State_return(x)

Arguments

| | |
|---|--------------|
| x | input vector |
|---|--------------|

Value

State_return

| | |
|------------|-------------------|
| SubsetData | <i>SubsetData</i> |
|------------|-------------------|

Description

This function is used for filtering out low-quality data based on previous result generated from [PlotMeta](#)

Usage

```
SubsetData(object, ...)

.subset_data(
  object,
  nFeature.upper = Inf,
  nFeature.lower = -Inf,
  Counts.upper = Inf,
  Counts.lower = -Inf
)

## S4 method for signature 'IRISFGM'
SubsetData(
  object,
  nFeature.upper = Inf,
  nFeature.lower = -Inf,
  Counts.upper = Inf,
  Counts.lower = -Inf
)
```

Arguments

| | |
|----------------|---|
| object | input IRIS-FGM object |
| ... | other arguments passed to methods |
| nFeature.upper | select upper limit for number of feature |
| nFeature.lower | select lower limit for number of feature |
| Counts.upper | select upper limit for number of UMI counts |
| Counts.lower | select lower limit for number of UMI counts |

Value

it will filter out some cell regarding threshold.

Examples

```
# Use Yan's data which posts on github tutorial
data("example_object")
example_object <- SubsetData(example_object,
  nFeature.upper=15000,
  nFeature.lower=8000,
  Counts.upper=700000,
  Counts.lower=400000)
```

Index

- * **datasets**
 - example_object, 14
 - .CalBinaryMultiSignal
 - (CalBinaryMultiSignal), 10
 - .CalBinarySingleSignal
 - (CalBinarySingleSignal), 11
 - .GetBinaryMultiSignal
 - (GetBinaryMultiSignal), 18
 - .GetBinarySingleSignal
 - (GetBinarySingleSignal), 19
 - .GetLTMGmatrix (GetLTMGmatrix), 20
 - .IDConvert, 5
 - .RunGO, 7
 - .RunKEGG, 7
 - .RunLTMG (RunLTMG), 42
 - .addMeta (AddMeta), 8
 - .dotPlotPathway (DotPlotPathway), 13
 - .final (FindClassBasedOnMC), 14
 - .findMarker (FindMarker), 17
 - .findGlobalMarkers (FindGlobalMarkers), 15
 - .generateNetObject, 3
 - .getBlock, 4
 - .getmeta (getMeta), 20
 - .plotDimension (PlotDimension), 27
 - .plotHeatmap (PlotHeatmap), 28
 - .plotMeta (PlotMeta), 30
 - .plotmodulenetwork (PlotModuleNetwork), 31
 - .processData (ProcessData), 33
 - .runBicluster (RunBicluster), 37
 - .runBiclusterBaseOnDiscretization, 5
 - .runBiclusterBaseOnLTMG, 6
 - .runClassification (RunClassification), 38
 - .runDimensionReduction
 - (RunDimensionReduction), 40
 - .runDiscretization (RunDiscretization), 41
 - .runPathway (RunPathway), 43
 - .separateBic, 8
 - .subset_data (SubsetData), 48
- AddMeta, 8
- AddMeta, IRISFGM-method (AddMeta), 8
- BIC_LTMG, 9
- BIC_ZIMG, 9
- CalBinaryMultiSignal, 10
- CalBinaryMultiSignal, IRISFGM-method (CalBinaryMultiSignal), 10
- CalBinarySingleSignal, 11
- CalBinarySingleSignal, IRISFGM-method (CalBinarySingleSignal), 11
- CLUSTERING, 11
- CreateIRISFGMObject, 12
- DimReduce-class, 13
- DotPlotPathway, 13
- DotPlotPathway, IRISFGM-method (DotPlotPathway), 13
- example_object, 14
- FindAllMarkers, 16
- FindClassBasedOnMC, 14
- FindClassBasedOnMC, IRISFGM-method (FindClassBasedOnMC), 14
- FindGlobalMarkers, 15
- FindGlobalMarkers, IRISFGM-method (FindGlobalMarkers), 15
- FindMarker, 17
- FindMarker, IRISFGM-method (FindMarker), 17
- Fit_LTMG, 18
- GetBinaryMultiSignal, 18
- GetBinaryMultiSignal, IRISFGM-method (GetBinaryMultiSignal), 18

- GetBinarySingleSignal, [19](#)
- GetBinarySingleSignal, IRISFGM-method (GetBinarySingleSignal), [19](#)
- GetLTMGmatrix, [20](#)
- GetLTMGmatrix, IRISFGM-method (GetLTMGmatrix), [20](#)
- getMeta, [20](#)
- getMeta, IRISFGM-method (getMeta), [20](#)
- Global_Zcut, [21](#)
- GRAPH, [21](#)

- InverseMillsRatio, [22](#)
- IRISFGM-class, [22](#)

- KS_LTMG, [23](#)
- KS_ZIMG, [23](#)

- LogSeparateKRpkmNew, [24](#)
- LogSeparateKRpkmNewLR, [24](#)
- LTMG, [25](#)

- MCL, [25](#)
- MIN_return, [26](#)
- MINUS, [26](#)

- Pi_Zj_Zcut_new, [27](#)
- PlotDimension, [27](#)
- PlotDimension, IRISFGM-method (PlotDimension), [27](#)
- PlotHeatmap, [28](#)
- PlotHeatmap, IRISFGM-method (PlotHeatmap), [28](#)
- PlotMarkerHeatmap, [29](#)
- PlotMeta, [30](#), [48](#)
- PlotMeta, IRISFGM-method (PlotMeta), [30](#)
- PlotModuleNetwork, [31](#)
- PlotModuleNetwork, IRISFGM-method (PlotModuleNetwork), [31](#)
- PlotNetwork, [32](#)
- PlotNetwork, IRISFGM-method (PlotNetwork), [32](#)
- ProcessData, [33](#)
- ProcessData, IRISFGM-method (ProcessData), [33](#)
- Pure_CDF, [34](#)

- qubic, [35](#)

- Read10X, [36](#)
- Read10X_h5, [37](#)
- ReadFrom10X_folder, [36](#)
- ReadFrom10X_folder, IRISFGM-method (ReadFrom10X_folder), [36](#)
- ReadFrom10X_h5, [36](#)
- ReadFrom10X_h5, IRISFGM-method (ReadFrom10X_h5), [36](#)
- RunBiccluster, [37](#)
- RunBiccluster, IRISFGM-method (RunBiccluster), [37](#)
- RunClassification, [14](#), [38](#)
- RunClassification, IRISFGM-method (RunClassification), [38](#)
- RunDimensionReduction, [14](#), [40](#)
- RunDimensionReduction, IRISFGM-method (RunDimensionReduction), [40](#)
- RunDiscretization, [41](#)
- RunDiscretization, IRISFGM-method (RunDiscretization), [41](#)
- RunLTMG, [14](#), [42](#)
- RunLTMG, IRISFGM-method (RunLTMG), [42](#)
- RunPathway, [43](#)
- RunPathway, IRISFGM-method (RunPathway), [43](#)

- SC, [44](#)
- SeparateKRpkmNew, [45](#)
- SeparateKRpkmNew2, [45](#)
- SeparateKRpkmNewLR, [46](#)
- SeparateKRpkmNewLRPlus, [46](#)
- SeparateKRpkmNewp, [47](#)
- State_return, [48](#)
- SubsetData, [48](#)
- SubsetData, IRISFGM-method (SubsetData), [48](#)