**Seurat使用手册**

**1、Setup the Seurat Object**

We start by reading in the data. All features in Seurat have been configured to work with sparse matrices which results in significant memory and speed savings for Drop-seq/inDrop/10x data.

install.packages('Seurat')

library(Seurat)

library(dplyr)

library(Matrix)

#Load the dataset

mouse.data <- Read10X(data.dir="F:/HCA/sample3v3/outs/filtered\_gene\_bc\_matrices/mm10")

Initialize the Seurat object with the raw (non-normalized data). Min.cells means to keep all genes expressed in >= 3 cells. Min.genes means to keep all cells with at least 200 detected genes.

mouse <- CreateSeuratObject(raw.data = mouse.data, min.cells = 3, min.genes = 200, project = "10X\_mouse")

**Description**

Initializes the Seurat object and some optional filtering

**Usage**

CreateSeuratObject(raw.data, project = "SeuratProject", min.cells = 0,

min.genes = 0, is.expr = 0, normalization.method = NULL,

scale.factor = 10000, do.scale = FALSE, do.center = FALSE,

names.field = 1, names.delim = "\_", meta.data = NULL,

display.progress = TRUE, ...)

**Arguments**

|  |  |
| --- | --- |
| raw.data | Raw input data |
| project | Project name (string) |
| min.cells | Include genes with detected expression in at least this many cells. Will subset the raw.data matrix as well. To reintroduce excluded genes, create a new object with a lower cutoff. |
| min.genes | Include cells where at least this many genes are detected. |
| is.expr | Expression threshold for 'detected' gene. For most datasets, particularly UMI datasets, will be set to 0 (default). If not, when initializing, this should be set to a level based on pre-normalized counts (i.e. require at least 5 counts to be treated as expresesd) All values less than this will be set to 0 (though maintained in object@raw.data). |
| normalization.method | Method for cell normalization. Default is no normalization. In this case, run NormalizeData later in the workflow. As a shortcut, you can specify a normalization method (i.e. LogNormalize) here directly. |
| scale.factor | If normalizing on the cell level, this sets the scale factor. |
| do.scale | In object@scale.data, perform row-scaling (gene-based z-score). FALSE by default. In this case, run ScaleData later in the workflow. As a shortcut, you can specify do.scale = TRUE (and do.center = TRUE) here. |
| do.center | In object@scale.data, perform row-centering (gene-based centering) |
| names.field | For the initial identity class for each cell, choose this field from the cell's column name |
| names.delim | For the initial identity class for each cell, choose this delimiter from the cell's column name |
| meta.data | Additional metadata to add to the Seurat object. Should be a data frame where the rows are cell names, and the columns are additional metadata fields |
| display.progress | display progress bar for normalization and/or scaling procedure. |
| ... | Ignored |

**Value**

Returns a Seurat object with the raw data stored in object@raw.data. object@data, object@meta.data, object@ident, also initialized.

**2、QC and selecting cells for further analysis**

VlnPlot(object = mouse, features.plot = c("nGene", "nUMI"), nCol = 2)

GenePlot(object = mouse, gene1 = "nUMI", gene2 = "nGene")

mouse <- FilterCells(object = mouse, subset.names = "nGene",

low.thresholds = c(200), high.thresholds = c(2500))

**Description**

Creates a Seurat object containing only a subset of the cells in the original object. Takes either a list of cells to use as a subset, or a parameter (for example, a gene), to subset on.

**Usage**

FilterCells(object, subset.names, low.thresholds, high.thresholds,

cells.use = NULL)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| subset.names | Parameters to subset on. Eg, the name of a gene, PC1, a column name in object@meta.data, etc. Any argument that can be retreived using FetchData |
| low.thresholds | Low cutoffs for the parameters (default is -Inf) |
| high.thresholds | High cutoffs for the parameters (default is Inf) |
| cells.use | A vector of cell names to use as a subset |

**Value**

Returns a Seurat object containing only the relevant subset of cells

**3、Normalizing the data**

After removing unwanted cells from the dataset, the next step is to normalize the data. By default, we employ a global-scaling normalization method “LogNormalize” that normalizes the gene expression measurements for each cell by the total expression, multiplies this by a scale factor (10,000 by default), and log-transforms the result.

pbmc <- NormalizeData(object = pbmc, normalization.method = "LogNormalize",

scale.factor = 10000)

**Description**

Normalize data for a given assay

**Usage**

NormalizeData(object, assay.type = "RNA",

normalization.method = "LogNormalize", scale.factor = 10000,

display.progress = TRUE)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| assay.type | Type of assay to normalize for (default is RNA), but can be changed for multimodal analyses. |
| normalization.method | Method for normalization. Default is log-normalization (LogNormalize). More methods to be added very shortly. |
| scale.factor | Sets the scale factor for cell-level normalization |
| display.progress | display progress bar for scaling procedure. |

**Value**

Returns object after normalization. Normalized data is stored in data or scale.data slot, depending on the method

**4.Detection of variable genes across the single cells**

Seurat calculates highly variable genes and focuses on these for downstream analysis. FindVariableGenes calculates the average expression and dispersion for each gene, places these genes into bins, and then calculates a z-score for dispersion within each bin. This helps control for the relationship between variability and average expression. This function is unchanged from (Macosko et al.), but new methods for variable gene expression identification are coming soon. We suggest that users set these parameters to mark visual outliers on the dispersion plot, but the exact parameter settings may vary based on the data type, heterogeneity in the sample, and normalization strategy. The parameters here identify ~2,000 variable genes, and represent typical parameter settings for UMI data that is normalized to a total of 1e4 molecules.

pbmc <- FindVariableGenes(object = pbmc, mean.function = ExpMean, dispersion.function = LogVMR,

x.low.cutoff = 0.0125, x.high.cutoff = 3, y.cutoff = 0.5)

length(x = pbmc@var.genes)

**Description**

Identifies genes that are outliers on a 'mean variability plot'. First, uses a function to calculate average expression (mean.function) and dispersion (dispersion.function) for each gene. Next, divides genes into num.bin (deafult 20) bins based on their average expression, and calculates z-scores for dispersion within each bin. The purpose of this is to identify variable genes while controlling for the strong relationship between variability and average expression.

**Usage**

FindVariableGenes(object, mean.function = ExpMean,

dispersion.function = LogVMR, do.plot = TRUE, set.var.genes = TRUE,

x.low.cutoff = 0.1, x.high.cutoff = 8, y.cutoff = 1,

y.high.cutoff = Inf, num.bin = 20, do.recalc = TRUE,

sort.results = TRUE, do.cpp = TRUE, display.progress = TRUE, ...)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| mean.function | Function to compute x-axis value (average expression). Default is to take the mean of the detected (i.e. non-zero) values |
| dispersion.function | Function to compute y-axis value (dispersion). Default is to take the standard deviation of all values/ |
| do.plot | Plot the average/dispersion relationship |
| set.var.genes | Set object@var.genes to the identified variable genes (default is TRUE) |
| x.low.cutoff | Bottom cutoff on x-axis for identifying variable genes |
| x.high.cutoff | Top cutoff on x-axis for identifying variable genes |
| y.cutoff | Bottom cutoff on y-axis for identifying variable genes |
| y.high.cutoff | Top cutoff on y-axis for identifying variable genes |
| num.bin | Total number of bins to use in the scaled analysis (default is 20) |
| do.recalc | TRUE by default. If FALSE, plots and selects variable genes without recalculating statistics for each gene. |
| sort.results | If TRUE (by default), sort results in object@hvg.info in decreasing order of dispersion |
| do.cpp | Run c++ version of mean.function and dispersion.function if they exist. |
| display.progress | show progress bar for calculations |
| ... | Extra parameters to VariableGenePlot |

**5.Scaling the data and removing unwanted sources of variation**

Your single cell dataset likely contains ‘uninteresting’ sources of variation. This could include not only technical noise, but batch effects, or even biological sources of variation (cell cycle stage). To mitigate the effect of these signals, Seurat constructs linear models to predict gene expression based on user-defined variables. The scaled z-scored residuals of these models are stored in the scale.data slot, and are used for dimensionality reduction and clustering.

We can regress out cell-cell variation in gene expression driven by batch (if applicable), cell alignment rate (as provided by Drop-seq tools for Drop-seq data), the number of detected molecules, and mitochondrial gene expression. For cycling cells, we can also learn a ‘cell-cycle’ score (see example [HERE]) and regress this out as well. In this simple example here for post-mitotic blood cells, we regress on the number of detected molecules per cell as well as the percentage mitochondrial gene content.

pbmc <- ScaleData(object = pbmc, vars.to.regress = c("nUMI", "percent.mito"))

**Description**

Scales and centers genes in the dataset. If variables are provided in vars.to.regress, they are individually regressed against each gene, and the resulting residuals are then scaled and centered.

**Usage**

ScaleData(object, genes.use = NULL, data.use = NULL, vars.to.regress,

model.use = "linear", use.umi = FALSE, do.scale = TRUE,

do.center = TRUE, scale.max = 10, block.size = 1000,

min.cells.to.block = 3000, display.progress = TRUE, assay.type = "RNA",

do.cpp = TRUE, check.for.norm = TRUE)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| genes.use | Vector of gene names to scale/center. Default is all genes in object@data. |
| data.use | Can optionally pass a matrix of data to scale, default is object@data[genes.use, ] |
| vars.to.regress | Variables to regress out (previously latent.vars in RegressOut). For example, nUMI, or percent.mito. |
| model.use | Use a linear model or generalized linear model (poisson, negative binomial) for the regression. Options are 'linear' (default), 'poisson', and 'negbinom' |
| use.umi | Regress on UMI count data. Default is FALSE for linear modeling, but automatically set to TRUE if model.use is 'negbinom' or 'poisson' |
| do.scale | Whether to scale the data. |
| do.center | Whether to center the data. |
| scale.max | Max value to return for scaled data. The default is 10. Setting this can help reduce the effects of genes that are only expressed in a very small number of cells. If regressing out latent variables and using a non-linear model, the default is 50. |
| block.size | Default size for number of genes to scale at in a single computation. Increasing block.size may speed up calculations but at an additional memory cost. |
| min.cells.to.block | If object contains fewer than this number of cells, don't block for scaling calculations. |
| display.progress | Displays a progress bar for scaling procedure |
| assay.type | Assay to scale data for. Default is RNA. Can be changed for multimodal analyses. |
| do.cpp | By default (TRUE), most of the heavy lifting is done in c++. We've maintained support for our previous implementation in R for reproducibility (set this to FALSE) as results can change slightly due to differences in numerical precision which could affect downstream calculations. |
| check.for.norm | Check to see if data has been normalized, if not, output a warning (TRUE by default) |

**Details**

ScaleData now incorporates the functionality of the function formerly known as RegressOut (which regressed out given the effects of provided variables and then scaled the residuals). To make use of the regression functionality, simply pass the variables you want to remove to the vars.to.regress parameter.

Setting center to TRUE will center the expression for each gene by subtracting the average expression for that gene. Setting scale to TRUE will scale the expression level for each gene by dividing the centered gene expression levels by their standard deviations if center is TRUE and by their root mean square otherwise.

**Value**

Returns a seurat object with object@scale.data updated with scaled and/or centered data.

**6.Perform linear dimensional reduction**

Next we perform PCA on the scaled data. By default, the genes in object@var.genes are used as input, but can be defined using pc.genes. We have typically found that running dimensionality reduction on highly variable genes can improve performance. However, with UMI data - particularly after regressing out technical variables, we often see that PCA returns similar (albeit slower) results when run on much larger subsets of genes, including the whole transcriptome.

pbmc <- RunPCA(object = pbmc, pc.genes = pbmc@var.genes, do.print = TRUE, pcs.print = 1:5, genes.print = 5)

**Description**

Run a PCA dimensionality reduction. For details about stored PCA calculation parameters, seePrintPCAParams.

**Usage**

RunPCA(object, pc.genes = NULL, pcs.compute = 20, use.imputed = FALSE,

rev.pca = FALSE, weight.by.var = TRUE, do.print = TRUE,

pcs.print = 1:5, genes.print = 30, reduction.name = "pca",

reduction.key = "PC", assay.type = "RNA", seed.use = 42, ...)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| pc.genes | Genes to use as input for PCA. Default is object@var.genes |
| pcs.compute | Total Number of PCs to compute and store (20 by default) |
| use.imputed | Run PCA on imputed values (FALSE by default) |
| rev.pca | By default computes the PCA on the cell x gene matrix. Setting to true will compute it on gene x cell matrix. |
| weight.by.var | Weight the cell embeddings by the variance of each PC (weights the gene loadings if rev.pca is TRUE) |
| do.print | Print the top genes associated with high/low loadings for the PCs |
| pcs.print | PCs to print genes for |
| genes.print | Number of genes to print for each PC |
| reduction.name | dimensional reduction name, specifies the position in the object$dr list. pca by default |
| reduction.key | dimensional reduction key, specifies the string before the number for the dimension names. PC by default |
| assay.type | Data type, RNA by default. Can be changed for multimodal |
| seed.use | Set a random seed. By default, sets the seed to 42. Setting NULL will not set a seed. |
| ... | Additional arguments to be passed to IRLBA |

**Value**

Returns Seurat object with the PCA calculation stored in object@dr$pca.

**7.Determine statistically significant principal components**

To overcome the extensive technical noise in any single gene for scRNA-seq data, Seurat clusters cells based on their PCA scores, with each PC essentially representing a ‘metagene’ that combines information across a correlated gene set. Determining how many PCs to include downstream is therefore an important step.

PCElbowPlot(object = pbmc)

**Description**

Plots the standard deviations (or approximate singular values if running PCAFast) of the principle components for easy identification of an elbow in the graph. This elbow often corresponds well with the significant PCs and is much faster to run.

**Usage**

PCElbowPlot(object, num.pc = 20)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| num.pc | Number of PCs to plot |

**Value**

Returns ggplot object

**8.Cluster the cells**

Seurat now includes an graph-based clustering approach compared to (Macosko et al.). Importantly, the distance metric which drives the clustering analysis (based on previously identified PCs) remains the same. However, our approach to partioning the cellular distance matrix into clusters has dramatically improved. Our approach was heavily inspired by recent manuscripts which applied graph-based clustering approaches to scRNA-seq data [SNN-Cliq, Xu and Su, Bioinformatics, 2015] and CyTOF data [PhenoGraph, Levine et al., Cell, 2015]. Briefly, these methods embed cells in a graph structure - for example a K-nearest neighbor (KNN) graph, with edges drawn between cells with similar gene expression patterns, and then attempt to partition this graph into highly interconnected ‘quasi-cliques’ or ‘communities’. As in PhenoGraph, we first construct a KNN graph based on the euclidean distance in PCA space, and refine the edge weights between any two cells based on the shared overlap in their local neighborhoods (Jaccard distance). To cluster the cells, we apply modularity optimization techniques [SLM, Blondel et al., Journal of Statistical Mechanics], to iteratively group cells together, with the goal of optimizing the standard modularity function.

The FindClusters function implements the procedure, and contains a resolution parameter that sets the ‘granularity’ of the downstream clustering, with increased values leading to a greater number of clusters. We find that setting this parameter between 0.6-1.2 typically returns good results for single cell datasets of around 3K cells. Optimal resolution often increases for larger datasets. The clusters are saved in the object@ident slot.

pbmc <- FindClusters(object = pbmc, reduction.type = "pca", dims.use = 1:10,

resolution = 0.6, print.output = 0, save.SNN = TRUE)

**Description**

Identify clusters of cells by a shared nearest neighbor (SNN) modularity optimization based clustering algorithm. First calculate k-nearest neighbors and construct the SNN graph. Then optimize the modularity function to determine clusters. For a full description of the algorithms, see Waltman and van Eck (2013) *The European Physical Journal B*.

**Usage**

FindClusters(object, genes.use = NULL, reduction.type = "pca",

dims.use = NULL, k.param = 30, k.scale = 25, plot.SNN = FALSE,

prune.SNN = 1/15, print.output = TRUE, distance.matrix = NULL,

save.SNN = FALSE, reuse.SNN = FALSE, force.recalc = FALSE,

modularity.fxn = 1, resolution = 0.8, algorithm = 1, n.start = 100,

n.iter = 10, random.seed = 0, temp.file.location = NULL)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| genes.use | A vector of gene names to use in construction of SNN graph if building directly based on expression data rather than a dimensionally reduced representation (i.e. PCs). |
| reduction.type | Name of dimensional reduction technique to use in construction of SNN graph. (e.g. "pca", "ica") |
| dims.use | A vector of the dimensions to use in construction of the SNN graph (e.g. To use the first 10 PCs, pass 1:10) |
| k.param | Defines k for the k-nearest neighbor algorithm |
| k.scale | Granularity option for k.param |
| plot.SNN | Plot the SNN graph |
| prune.SNN | Sets the cutoff for acceptable Jaccard distances when computing the neighborhood overlap for the SNN construction. Any edges with values less than or equal to this will be set to 0 and removed from the SNN graph. Essentially sets the strigency of pruning (0 — no pruning, 1 — prune everything). |
| print.output | Whether or not to print output to the console |
| distance.matrix | Build SNN from distance matrix (experimental) |
| save.SNN | Saves the SNN matrix associated with the calculation in object@snn |
| reuse.SNN | Force utilization of stored SNN. If none store, this will throw an error. |
| force.recalc | Force recalculation of SNN. |
| modularity.fxn | Modularity function (1 = standard; 2 = alternative). |
| 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). |
| n.start | Number of random starts. |
| n.iter | Maximal number of iterations per random start. |
| random.seed | Seed of the random number generator. |
| temp.file.location | Directory where intermediate files will be written. Specify the ABSOLUTE path. |

**Value**

Returns a Seurat object and optionally the SNN matrix, object@ident has been updated with new cluster info

**9.Run Non-linear dimensional reduction (tSNE)**

Seurat continues to use tSNE as a powerful tool to visualize and explore these datasets. While we no longer advise clustering directly on tSNE components, cells within the graph-based clusters determined above should co-localize on the tSNE plot. This is because the tSNE aims to place cells with similar local neighborhoods in high-dimensional space together in low-dimensional space. As input to the tSNE, we suggest using the same PCs as input to the clustering analysis, although computing the tSNE based on scaled gene expression is also supported using the genes.use argument.

pbmc <- RunTSNE(object = pbmc, dims.use = 1:10, do.fast = TRUE)

TSNEPlot(object = pbmc)

**Description**

Run t-SNE dimensionality reduction on selected features. Has the option of running in a reduced dimensional space (i.e. spectral tSNE, recommended), or running based on a set of genes. For details about stored TSNE calculation parameters, see PrintTSNEParams.

**Usage**

RunTSNE(object, reduction.use = "pca", cells.use = NULL, dims.use = 1:5,

genes.use = NULL, seed.use = 1, do.fast = TRUE, add.iter = 0,

dim.embed = 2, distance.matrix = NULL, reduction.name = "tsne",

reduction.key = "tSNE\_", ...)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| reduction.use | Which dimensional reduction (e.g. PCA, ICA) to use for the tSNE. Default is PCA |
| cells.use | Which cells to analyze (default, all cells) |
| dims.use | Which dimensions to use as input features |
| genes.use | If set, run the tSNE on this subset of genes (instead of running on a set of reduced dimensions). Not set (NULL) by default |
| seed.use | Random seed for the t-SNE |
| do.fast | If TRUE, uses the Barnes-hut implementation, which runs faster, but is less flexible. TRUE by default. |
| add.iter | If an existing tSNE has already been computed, uses the current tSNE to seed the algorithm and then adds additional iterations on top of this |
| dim.embed | The dimensional space of the resulting tSNE embedding (default is 2). For example, set to 3 for a 3d tSNE |
| distance.matrix | If set, runs tSNE on the given distance matrix instead of data matrix (experimental) |
| reduction.name | dimensional reduction name, specifies the position in the object$dr list. tsne by default |
| reduction.key | dimensional reduction key, specifies the string before the number for the dimension names. tSNE\_ by default |
| ... | Additional arguments to the tSNE call. Most commonly used is perplexity (expected number of neighbors default is 30) |

**Value**

Returns a Seurat object with a tSNE embedding in object@dr$tsne@cell.embeddings

**10.Finding differentially expressed genes (cluster biomarkers)**

Seurat can help you find markers that define clusters via differential expression. By default, it identifes positive and negative markers of a single cluster (specified in ident.1), compared to all other cells. FindAllMarkers automates this process for all clusters, but you can also test groups of clusters vs. each other, or against all cells.

The min.pct argument requires a gene to be detected at a minimum percentage in either of the two groups of cells, and the thresh.test argument requires a gene to be differentially expressed (on average) by some amount between the two groups. You can set both of these to 0, but with a dramatic increase in time - since this will test a large number of genes that are unlikely to be highly discriminatory. As another option to speed up these computations, max.cells.per.ident can be set. This will downsample each identity class to have no more cells than whatever this is set to. While there is generally going to be a loss in power, the speed increases can be significiant and the most highly differentially expressed genes will likely still rise to the top.

cluster1.markers <- FindMarkers(object = pbmc, ident.1 = 1, min.pct = 0.25)

print(x = head(x = cluster1.markers, n = 5))

**Description**

Finds markers (differentially expressed genes) for identity classes

**Usage**

FindMarkers(object, ident.1, ident.2 = NULL, genes.use = NULL,

logfc.threshold = 0.25, test.use = "wilcox", min.pct = 0.1,

min.diff.pct = -Inf, print.bar = TRUE, only.pos = FALSE,

max.cells.per.ident = Inf, random.seed = 1, latent.vars = "nUMI",

min.cells = 3, pseudocount.use = 1, assay.type = "RNA", ...)

**Arguments**

|  |  |
| --- | --- |
| object | Seurat object |
| ident.1 | Identity class to define markers for |
| ident.2 | A second identity class for comparison. If NULL (default) - use all other cells for comparison. |
| genes.use | Genes to test. Default is to use all genes |
| logfc.threshold | 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 logfc.threshold speeds up the function, but can miss weaker signals. |
| test.use | Denotes which test to use. Available options are:   * "wilcox" : Wilcoxon rank sum test (default) * "bimod" : Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013) * "roc" : Standard AUC classifier * "t" : Student's t-test * "tobit" : Tobit-test for differential gene expression (Trapnell et al., Nature Biotech, 2014) * "poisson" : Likelihood ratio test assuming an underlying poisson distribution. Use only for UMI-based datasets * "negbinom" : Likelihood ratio test assuming an underlying negative binomial distribution. Use only for UMI-based datasets * "MAST : GLM-framework that treates cellular detection rate as a covariate (Finak et al, Genome Biology, 2015) * "DESeq2 : DE based on a model using the negative binomial distribution (Love et al, Genome Biology, 2014) |
| min.pct | only test genes that are detected in a minimum fraction of min.pct 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 |
| min.diff.pct | only test genes that show a minimum difference in the fraction of detection between the two groups. Set to -Inf by default |
| print.bar | Print a progress bar once expression testing begins (uses pbapply to do this) |
| only.pos | Only return positive markers (FALSE by default) |
| max.cells.per.ident | Down sample each identity class to a max number. Default is no downsampling. Not activated by default (set to Inf) |
| random.seed | Random seed for downsampling |
| latent.vars | Variables to test |
| min.cells | Minimum number of cells expressing the gene in at least one of the two groups |
| pseudocount.use | Pseudocount to add to averaged expression values when calculating logFC. 1 by default. |
| assay.type | Type of assay to fetch data for (default is RNA) |
| ... | Additional parameters to pass to specific DE functions |

**Details**

p-value adjustment is performed using bonferroni correction based on the total number of genes in the dataset. Other correction methods are not recommended, as Seurat pre-filters genes using the arguments above, reducing the number of tests performed. Lastly, as Aaron Lun has pointed out, p-values should be interpreted cautiously, as the genes used for clustering are the same genes tested for differential expression.

**Value**

Matrix containing a ranked list of putative markers, and associated statistics (p-values, ROC score, etc.)

<https://paste.ubuntu.com/p/fJFqg2zkpH/> 所有源代码附件

Fengchen

20180302