Package ‘UCell’

March 21, 2024

Type Package

Title Rank-based signature enrichment analysis for single-cell data

Version 2.6.2

Description UCell is a package for evaluating gene signatures in single-cell datasets. UCell signature scores, based on the Mann-Whitney U statistic, are robust to dataset size and heterogeneity, and their calculation demands less computing time and memory than other available methods, enabling the processing of large datasets in a few minutes even on machines with limited computing power. UCell can be applied to any single-cell data matrix, and includes functions to directly interact with SingleCellExperiment and Seurat objects.

Depends R(>= 4.2.0)

Imports methods, data.table(>= 1.13.6), Matrix, stats, BiocParallel, BiocNeighbors, SingleCellExperiment, SummarizedExperiment

Suggests Seurat, scater, scRNAseq, reshape2, patchwork, ggplot2, BiocStyle, knitr, rmarkdown

biocViews SingleCell, GeneSetEnrichment, Transcriptomics, GeneExpression, CellBasedAssays

VignetteBuilder knitr

BugReports https://github.com/carmonalab/UCell/issues


License GPL-3 + file LICENSE

Encoding UTF-8

LazyData FALSE

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

git_url https://git.bioconductor.org/packages/UCell
git_branch RELEASE_3_18
git_last_commit 06be2df
git_last_commit_date 2023-11-06
**AddModuleScore_UCell**

**Calculate module enrichment scores from single-cell data (Seurat interface)**

**Description**

Given a Seurat object, calculates module/signature enrichment scores at single-cell level using the Mann-Whitney U statistic. UCell scores are normalized U statistics (between 0 and 1), and they are mathematically related to the Area under the ROC curve (see Mason and Graham)

**Usage**

```r
AddModuleScore_UCell(
  obj,
  features,
  maxRank = 1500,
  chunk.size = 1000,
  BPPARAM = NULL,
  ncores = 1,
  storeRanks = FALSE,
)```

---

**R topics documented:**

- AddModuleScore_UCell .................................................. 2
- calculate_Uscore .......................................................... 4
- check_genes ................................................................. 5
- check_signature_names .................................................. 6
- data_to_ranks_data_table ................................................ 6
- knn_smooth_scores ........................................................ 7
- rankings2Uscore ........................................................... 7
- sample.matrix ............................................................. 8
- ScoreSignatures_UCell .................................................... 9
- SmoothKNN.Seurat ......................................................... 10
- split_data.matrix ........................................................ 13
- StoreRankings_UCell ...................................................... 14
- UCell ................................................................. 15
- u_stat ................................................................. 16
- u_stat_signature_list .................................................... 17

**Index**

18
AddModuleScore_UCell

w_neg = 1,
assay = NULL,
slot = "counts",
ties.method = "average",
force.gc = FALSE,
name = "_UCell")

Arguments

obj Seurat object
features A list of signatures, for example: list(Tcell_signature = c("CD2", "CD3E", "CD3D"), Myeloid_signature = c("SPI1", "FCER1G", "CSF1R")) You can also specify positive and negative gene sets by adding a + or - sign to genes in the signature; see an example below
maxRank Maximum number of genes to rank per cell; above this rank, a given gene is considered as not expressed.
chunk.size Number of cells to be processed simultaneously (lower size requires slightly more computation but reduces memory demands)
BPPARAM A BiocParallel::bpparam() object that tells UCell how to parallelize. If provided, it overrides the ncores parameter.
ncores Number of processors to parallelize computation. If BPPARAM = NULL, the function uses BiocParallel::MulticoreParam(workers=ncores)
storeRanks Store ranks matrix in Seurat object ('UCellRanks' assay) for fast subsequent computations. This option may demand large amounts of RAM.
w_neg Weight on negative genes in signature. e.g. w_neg=1 weighs equally up- and down-regulated genes, w_neg=0.5 gives 50% less importance to negative genes
assay Pull out data from this assay of the Seurat object (if NULL, use DefaultAssay(obj))
slot Pull out data from this slot of the Seurat object (will become "layer" in Seurat v5)
ties.method How ranking ties should be resolved - passed on to data.table::frank
force.gc Explicitly call garbage collector to reduce memory footprint
name Name tag that will be appended at the end of each signature name, "_UCell" by default (e.g. signature score in meta data will be named: Myeloid_signature_UCell)

Details

In contrast to Seurat’s AddModuleScore, which is normalized by binning genes of similar expression at the population level, UCell scores depend only on the gene expression ranks of individual cell, and therefore they are robust across datasets regardless of dataset composition.

Value

Returns a Seurat object with module/signature enrichment scores added to object meta data; each score is stored as the corresponding signature name provided in features followed by the tag given in name (or "_UCell" by default)
Examples

```r
library(UCell)
gene.sets <- list(Tcell = c("CD2","CD3E","CD3D"),
    Myeloid = c("SPI1","FCER1G","CSF1R"))
data(sample.matrix)
obj <- Seurat::CreateSeuratObject(sample.matrix)

obj <- AddModuleScore_UCell(obj,features = gene.sets)
head(obj[[[]]])

## Using positive and negative gene sets
gene.sets <- list()
gene.sets$Tcell_gd <- c("TRDC+","TRGC1+","TRGC2+","TRDV1+",
    "TRAC-","TRBC1-","TRBC2-")
gene.sets$NKcell <- c("FGFbp2+","SPON2+","KLRF1+",
    "FGCGR3A+","CD3E-","CD3G-")
obj <- AddModuleScore_UCell(obj, features = gene.sets, name=NULL)
head(obj$NKcell)
```

calculate_Uscore | Calculate rankings and scores for query data and given signature set

Description

Calculate rankings and scores for query data and given signature set

Usage

```r
calculate_Uscore(
    matrix,
    features,
    maxRank = 1500,
    chunk.size = 1000,
    BPPARAM = NULL,
    ncores = 1,
    w_neg = 1,
    ties.method = "average",
    storeRanks = FALSE,
    force.gc = FALSE,
    name = "_UCell"
)
```

Arguments

- **matrix**: Input data matrix
- **features**: List of signatures
- **maxRank**: Rank cutoff (1500)
check_genes

chunk.size  Cells per sub-matrix (1000)
BPPARAM    A BioParallel object to instruct UCell how to parallelize
ncores     Number of cores to use for parallelization
w_neg      Weight on negative signatures
ties.method How to break ties, for data.table::frankv method ("average")
storeRanks Store ranks? (FALSE)
force.gc   Force garbage collection? (FALSE)
name       Suffix for metadata columns ("_UCell")

Value

A list of signature scores

check_genes  Check genes

Description

Check if all genes in signatures are found in data matrix - otherwise add zero counts in data-matrix to complete it

Usage

check_genes(matrix, features)

Arguments

matrix  Input data matrix
features List of genes that must be present (otherwise they are added)

Value

Same input matrix, extended to comprise any missing genes
check_signature_names  Check signature names and add standard names is missing

---

Description
Check signature names and add standard names is missing

Usage
check_signature_names(features)

Arguments
- features  List of signatures for scoring

Value
The input list of signatures, with standard names if provided un-named

---

data_to_ranks_data_table
Calculate per-cell feature rankings

---

Description
Calculate per-cell feature rankings

Usage
data_to_ranks_data_table(data, ties.method = "average")

Arguments
- data  Expression data matrix
- ties.method  How to break ties (passed on to data.table::frankv)

Value
A data.table of ranks
**knn_smooth_scores**

*Smoothing scores by KNN*

**Description**

Smoothing scores by KNN

**Usage**

```
knn_smooth_scores(matrix = NULL, nn = NULL, decay = 0.1, up.only = FALSE)
```

**Arguments**

- `matrix`: Input data matrix
- `nn`: A nearest neighbor object returned by `BiocNeighbors::findKNN`
- `decay`: Exponential decay for nearest neighbor weight: (1-decay)^n
- `up.only`: If set to TRUE, smoothed scores will only be allowed to increase by smoothing

**Value**

A dataframe of knn-smoothed scores

---

**rankings2Uscore**

*Get signature scores from pre-computed rank matrix*

**Description**

Get signature scores from pre-computed rank matrix

**Usage**

```
rankings2Uscore(
    ranks_matrix, features, chunk.size = 1000, w_neg = 1, BPPARAM = NULL, ncores = 1, force.gc = FALSE, name = "_UCell"
)
```
sample.matrix

**Arguments**

- **ranks_matrix**: A rank matrix
- **features**: List of signatures
- **chunk.size**: How many cells per matrix chunk
- **w_neg**: Weight on negative signatures
- **BPPARAM**: A BioParallel object to instruct UCell how to parallelize
- **ncores**: How many cores to use for parallelization?
- **force.gc**: Force garbage collection to recover RAM? (FALSE)
- **name**: Name suffix for metadata columns ("_UCell")

**Value**

A list of signature scores

---

**sample.matrix**

*Sample dataset to test UCell installation*

**Description**

A sparse matrix (class "dgCMatrix") of single-cell transcriptomes (scRNA-seq) for 600 cells and 20729 genes. Single-cell UMI counts were normalized using a standard log-normalization: counts for each cell were divided by the total counts for that cell and multiplied by 10,000, then natural-log transformed using log1p.

This a subsample of T cells from the large scRNA-seq PBMC dataset published by Hao et al. and available as UMI counts at [https://atlas.fredhutch.org/data/nygc/multimodal/pbmc_multimodal.h5seurat](https://atlas.fredhutch.org/data/nygc/multimodal/pbmc_multimodal.h5seurat)

**Usage**

`sample.matrix`

**Format**

A sparse matrix of 600 cells and 20729 genes.

**Source**

ScoreSignatures_UCell  Calculate module enrichment scores from single-cell data

Description

Given a gene vs. cell matrix, calculates module/signature enrichment scores on single-cell level using Mann-Whitney U statistic. UCell scores are normalized U statistics (between 0 and 1), and they are mathematically related to the Area under the ROC curve (see Mason and Graham). These scores only depend on the gene expression ranks of individual cell, and therefore they are robust across datasets regardless of dataset composition.

Usage

ScoreSignatures_UCell(
  matrix = NULL,
  features,
  precalc.ranks = NULL,
  maxRank = 1500,
  w_neg = 1,
  name = "_UCell",
  assay = "counts",
  chunk.size = 1000,
  BPPARAM = NULL,
  ncores = 1,
  ties.method = "average",
  force.gc = FALSE
)

Arguments

- **matrix**: Input matrix, either stored in a `SingleCellExperiment` object or as a raw matrix. `dgCMatrix` format supported.
- **features**: A list of signatures, for example: `list(Tcell_signature = c("CD2", "CD3E", "CD3D"), Myeloid_signature = c("SPI1", "FCER1G", "CSF1R"))`. You can also specify positive and negative gene sets by adding a `+` or `-` sign to genes in the signature; see an example below.
- **precalc.ranks**: If you have pre-calculated ranks using `StoreRankings_UCell`, you can specify the pre-calculated ranks instead of the gene vs. cell matrix.
- **maxRank**: Maximum number of genes to rank per cell; above this rank, a given gene is considered as not expressed. Note: this parameter is ignored if precalc.ranks are specified.
- **w_neg**: Weight on negative genes in signature. e.g. `w_neg=1` weighs equally up- and down-regulated genes, `w_neg=0.5` gives 50% less importance to negative genes.
- **name**: Name suffix appended to signature names.
- **assay**: The sce object assay where the data is to be found.
chunk.size  Number of cells to be processed simultaneously (lower size requires slightly more computation but reduces memory demands)

BPPARAM  A `BiocParallel::bpparam()` object that tells UCell how to parallelize. If provided, it overrides the ncores parameter.

ncores  Number of processors to parallelize computation. If `BPPARAM = NULL`, the function uses `BiocParallel::MulticoreParam(workers=ncores)`

ties.method  How ranking ties should be resolved - passed on to `data.table::frank`

force.gc  Explicitly call garbage collector to reduce memory footprint

Value  Returns input SingleCellExperiment object with UCell scores added to altExp

Examples

```r
library(UCell)

# Using sparse matrix
data(sample.matrix)
gene.sets <- list( Tcell_signature = c("CD2","CD3E","CD3D"),
                   Myeloid_signature = c("SPI1","FCER1G","CSF1R"))
scores <- ScoreSignatures_UCell(sample.matrix, features=gene.sets)
head(scores)

# Using sce object
library(SingleCellExperiment)
data(sample.matrix)
my.sce <- SingleCellExperiment(list(counts=sample.matrix))
gene.sets <- list( Tcell_signature = c("CD2","CD3E","CD3D"),
                   Myeloid_signature = c("SPI1","FCER1G","CSF1R"))
my.sce <- ScoreSignatures_UCell(my.sce, features=gene.sets)
altExp(my.sce, 'UCell')
```

SmoothKNN.Seurat  Smooth signature scores by kNN

Description

This function performs smoothing of single-cell scores by weighted average of the k-nearest neighbors. It can be useful to 'impute' scores by neighboring cells and partially correct data sparsity. While this function has been designed to smooth UCell scores, it can be applied to any numerical metadata contained in SingleCellExperiment or Seurat objects.
Usage

```r
## S3 method for class 'Seurat'
SmoothKNN(
  obj = NULL,
  signature.names = NULL,
  reduction = "pca",
  k = 10,
  decay = 0.1,
  up.only = FALSE,
  BNPARAM = AnnoyParam(),
  BPPARAM = SerialParam(),
  suffix = "_kNN",
  assay = NULL,
  slot = "data",
  sce.expname = NULL,
  sce.assay = NULL
)

## S3 method for class 'SingleCellExperiment'
SmoothKNN(
  obj = NULL,
  signature.names = NULL,
  reduction = "PCA",
  k = 10,
  decay = 0.1,
  up.only = FALSE,
  BNPARAM = AnnoyParam(),
  BPPARAM = SerialParam(),
  suffix = "_kNN",
  assay = NULL,
  slot = "data",
  sce.expname = c("UCell", "main"),
  sce.assay = NULL
)
```

SmoothKNN(
  obj = NULL,
  signature.names = NULL,
  reduction = "pca",
  k = 10,
  decay = 0.1,
  up.only = FALSE,
  BNPARAM = AnnoyParam(),
  BPPARAM = SerialParam(),
  suffix = "_kNN",
  assay = NULL,
  slot = "data",
  sce.expname = c("UCell", "main"),
  sce.assay = NULL
)
sce.assay = NULL
)

Arguments

obj Input object - either a SingleCellExperiment object or a Seurat object.
signature.names The names of the signatures (or any numeric metadata column) for which to calculate kNN-smoothed scores
reduction Which dimensionality reduction to use for kNN smoothing. It must be already present in the input object.
k Number of neighbors for kNN smoothing
decay Exponential decay for nearest neighbor weight: (1-decay)^n
up.only If set to TRUE, smoothed scores will only be allowed to increase by smoothing
BNPARAM A BiocNeighborParam object specifying the algorithm to use for kNN calculation.
BPPARAM A BiocParallel::bpparam() object for parallel computing, e.g. MulticoreParam or SnowParam
suffix Suffix to append to metadata columns for the new knn-smoothed scores
assay For Seurat objects only - do smoothing on expression data from this assay. When NULL, only looks in metadata
slot For Seurat objects only - do smoothing on expression data from this slot
sce.expname For sce objects only - which experiment stores the signatures to be smoothed. Set to 'main' for smoothing gene expression stored in the main sce experiment.
sce.assay For sce objects only - pull data from this assay

Value

An augmented obj with the smoothed signatures. If obj is a Seurat object, smoothed signatures are added to metadata; if obj is a SingleCellExperiment object, smoothed signatures are returned in a new altExp. See the examples below.

Examples

#### Using Seurat ####
library(Seurat)
gene.sets <- list(Tcell = c("CD2","CD3E","CD3D"),
                    Myeloid = c("SPI1","FCER1G","CSF1R"))
data(sample.matrix)
obj <- Seurat::CreateSeuratObject(sample.matrix)
# Calculate UCell scores
obj <- AddModuleScore_UCell(obj,features = gene.sets, name=NULL)
# Run PCA
obj <- FindVariableFeatures(obj) |> NormalizeData() |> ScaleData() |> RunPCA()
# Smooth signatures
obj <- SmoothKNN(obj, reduction="pca", signature.names=names(gene.sets))
head(obj[[1]])
```r
### Using SingleCellExperiment ###
library(SingleCellExperiment)
library(scater)
data(sample.matrix)
sce <- SingleCellExperiment(list(counts=sample.matrix))
gene.sets <- list(Tcell = c("CD2","CD3E","CD3D"),
                  Myeloid = c("SPI1","FCER1G","CSF1R"))
# Calculate UCell scores
sce <- ScoreSignatures_UCell(sce, features=gene.sets, name=NULL)
# Run PCA
sce <- logNormCounts(sce)
sce <- runPCA(sce, scale=TRUE, ncomponents=20)
# Smooth signatures
sce <- SmoothKNN(sce, reduction="PCA", signature.names=names(gene.sets))
# See results
altExp(sce, 'UCell')
asstays(altExp(sce, 'UCell'))
# Plot on UMAP
sce <- runUMAP(sce, dimred="PCA")
plotUMAP(sce, colour_by = "Tcell_kNN", by_exprs_values = "UCell_kNN")
```

---

**split_data.matrix**

Split data matrix into smaller sub-matrices (‘chunks’)

**Description**

Split data matrix into smaller sub-matrices (‘chunks’)

**Usage**

`split_data.matrix(matrix, chunk.size = 1000)`

**Arguments**

- `matrix` Input data matrix
- `chunk.size` How many cells to include in each sub-matrix

**Value**

A list of sub-matrices, each with size n_features x chunk_size
StoreRankings_UCell  

*Calculate and store gene rankings for a single-cell dataset*

**Description**

Given a gene vs. cell matrix, calculates the rankings of expression for all genes in each cell.

**Usage**

```r
StoreRankings_UCell(
  matrix,  
  maxRank = 1500, 
  chunk.size = 1000, 
  BPPARAM = NULL, 
  ncores = 1, 
  assay = "counts", 
  ties.method = "average", 
  force.gc = FALSE
)
```

**Arguments**

- **matrix**: Input matrix, either stored in a `SingleCellExperiment` object or as a raw matrix. `dgCMatrix` format supported.
- **maxRank**: Maximum number of genes to rank per cell; above this rank, a given gene is considered as not expressed.
- **chunk.size**: Number of cells to be processed simultaneously (lower size requires slightly more computation but reduces memory demands).
- **BPPARAM**: A `BiocParallel::bpparam()` object that tells UCell how to parallelize. If provided, it overrides the `ncores` parameter.
- **ncores**: Number of processors to parallelize computation. If `BPPARAM = NULL`, the function uses `BiocParallel::MulticoreParam(workers=ncores)`.
- **assay**: Assay where the data is to be found (for input in 'sce' format).
- **ties.method**: How ranking ties should be resolved - passed on to `data.table::frank`.
- **force.gc**: Explicitly call garbage collector to reduce memory footprint.

**Details**

While `ScoreSignatures_UCell` can be used 'on the fly' to evaluate signatures in a query dataset, it requires recalculating gene ranks at every execution. If you have a large dataset and plan to experiment with multiple signatures, evaluating the same dataset multiple times, this function allows you to store pre-calculated ranks so they do not have to be recomputed every time. Pre-calculated ranks can then be applied to the function `ScoreSignatures_UCell` to evaluate gene signatures in a significantly faster way on successive iterations.
Value

Returns a sparse matrix of pre-calculated ranks that can be used multiple times to evaluate different signatures.

Examples

```r
library(UCell)
data(sample.matrix)
ranks <- StoreRankings_UCell(sample.matrix)
ranks[1:5,1:5]
gene.sets <- list( Tcell_signature = c("CD2","CD3E","CD3D"),
                     Myeloid_signature = c("SPI1","FCER1G","CSF1R"))
scores <- ScoreSignatures_UCell(features=gene.sets, precalc.ranks=ranks)
head(scores)
```

---

**UCell**

*UCell: Robust and scalable single-cell gene signature scoring*

---

**Description**

UCell is an R package for scoring gene signatures in single-cell datasets. UCell scores, based on the Mann-Whitney U statistic, are robust to dataset size and heterogeneity, and their calculation demands relatively less computing time and memory than most other methods, enabling the processing of large datasets (> $10^5$ cells). UCell can be applied to any cell vs. gene data matrix, and includes functions to directly interact with Seurat and SingleCellExperiment objects.

**UCell functions**

- `ScoreSignatures_UCell` Calculate module enrichment scores from single-cell data. Given a gene vs. cell matrix (either as sparse matrix or stored in a SingleCellExperiment object), it calculates module/signature enrichment scores. This score depends only on the gene activity ranks of individual cell, and therefore is robust across datasets.

- `AddModuleScore_UCell` A wrapper for UCell to interact directly with Seurat objects. Given a Seurat object and a set of signatures, it calculates enrichment scores on single-cell level and returns them into the meta.data of the input Seurat object.

- `StoreRankings_UCell` Calculates and stores gene rankings for a single-cell dataset. Given a gene vs. cell matrix and a set of signatures, it calculates the rankings of expression for all genes in each cell. It can then be applied to the function `ScoreSignatures_UCell` to evaluate gene signatures on the gene expression ranks of individual cells.

- `SmoothKNN` Perform signature score smoothing using a weighted average of the scores of the first k nearest neighbors (kNN). It can be useful to 'impute' scores by neighboring cells and partially correct data sparsity. While this function has been designed to smooth UCell scores, it can be applied to any numerical metadata contained in SingleCellExperiment or Seurat objects.
Gene signatures

UCell evaluates the strength of gene signatures (or gene sets) in individual cells of your dataset. You may specify positive and negative (up- or down-regulated) genes in signatures. See the examples below:

```r
markers <- list()
markers$Tcell_CD4 <- c("CD4","CD40LG")
markers$Tcell_CD8 <- c("CD8A","CD8B")
markers$Tcell_Treg <- c("FOXP3","IL2RA")
markers$Tcell_gd <- c("TRDC+","TRGC1+","TRGC2+",
                       "TRDV1+","TRAC-","TRBC1-","TRBC2-")
markers$Tcell_NK <- c("FGFBP2+","SPON2+","KLRF1+",
                      "FCGR3A+","CD3E-","CD3G-")
```

If you don’t specify +/- for genes, they are assumed to be all as a positive set. The UCell score is calculated as:

\[
U = \max(0, U^+ - w_{neg} \cdot U^-)
\]

where \(U^+\) and \(U^-\) are respectively the UCell scores for the positive and negative set, and \(w_{neg}\) is a weight on the negative set. When no negative set of genes is present, \(U = U^+\)

References


---

**u_stat**

*Calculate Mann Whitney U from a vector of ranks*

**Description**

Calculate Mann Whitney U from a vector of ranks

**Usage**

```r
u_stat(rank_value, maxRank = 1000, sparse = FALSE)
```

**Arguments**

- `rank_value`: A vector of ranks
- `maxRank`: Max number of features to include in ranking
- `sparse`: Whether the vector of ranks is in sparse format

**Value**

Normalized AUC (as U statistic) for the vector
u_stat_signature_list

Calculate U scores for a list of signatures, given a rank matrix

Description

Calculate U scores for a list of signatures, given a rank matrix

Usage

u_stat_signature_list(
  sig_list,
  ranks_matrix,
  maxRank = 1000,
  sparse = FALSE,
  w_neg = 1
)

Arguments

- **sig_list**: A list of signatures
- **ranks_matrix**: Matrix of pre-computed ranks
- **maxRank**: Max number of features to include in ranking, for u_stat function
- **sparse**: Whether the vector of ranks is in sparse format
- **w_neg**: Weight on negative signatures

Value

A matrix of U scores
Index

* datasets
  sample.matrix, 8
AddModuleScore_UCell, 2
BiocNeighborParam, 12
BiocNeighbors::findKNN, 7
BiocParallel::bpparam(), 3, 10, 12, 14
calculate_Uscore, 4
check_genes, 5
check_signature_names, 6
data.table::frank, 3, 10, 14
data_to_ranks_data_table, 6
knn_smooth_scores, 7
MulticoreParam, 12
rankings2Uscore, 7
sample.matrix, 8
ScoreSignatures_UCell, 9, 14
SingleCellExperiment, 9, 12, 14
SmoothKNN(SmoothKNN.Seurat), 10
SmoothKNN.Seurat, 10
SnowParam, 12
split_data.matrix, 13
StoreRankings_UCell, 9, 14
u_stat, 16
u_stat_signature_list, 17
UCell, 15