Package ‘scGate’

April 23, 2024

Type Package

Title Marker-Based Cell Type Purification for Single-Cell Sequencing

Data

Version 1.6.2

Description
A common bioinformatics task in single-cell data analysis is to purify a cell type or cell population of interest from heterogeneous datasets. ‘scGate’ automatizes marker-based purification of specific cell populations, without requiring training data or reference gene expression profiles. Briefly, ‘scGate’ takes as input: i) a gene expression matrix stored in a ‘Seurat’ object and ii) a “gating model” (GM), consisting of a set of marker genes that define the cell population of interest. The GM can be as simple as a single marker gene, or a combination of positive and negative markers. More complex GMs can be constructed in a hierarchical fashion, akin to gating strategies employed in flow cytometry. ‘scGate’ evaluates the strength of signature marker expression in each cell using the rank-based method ‘UCell’, and then performs k-nearest neighbor (kNN) smoothing by calculating the mean ‘UCell’ score across neighboring cells. kNN-smoothing aims at compensating for the large degree of sparsity in scRNA-seq data. Finally, a universal threshold over kNN-smoothed signature scores is applied in binary decision trees generated from the user-provided gating model, to annotate cells as either “pure” or “impure”, with respect to the cell population of interest. See the related publication Andreatta et al. (2022) <doi:10.1093/bioinformatics/btac141>.

biocViews

Depends R (>= 4.3.0)

Imports Seurat (>= 4.0.0), UCell (>= 2.6.0), dplyr, stats, utils, methods, patchwork, ggridges, reshape2, ggplot2, BiocParallel

Suggests ggpaint, partykit, knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/carmonalab/scGate

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

License GPL-3

Encoding UTF-8

LazyData true
combine_scGate_multiclass

Combine scGate annotations

Description

If a single-cell dataset has precomputed results for multiple scGate models, combined them in multi-class annotation

Usage

```r
combine_scGate_multiclass(
  obj,
  prefix = "is.pure_",
  scGate_classes = NULL,
  min_cells = 1,
  multi.asNA = FALSE,
  out_column = "scGate_multi"
)
```
gating_model

Description

Generate an scGate model from scratch or edit an existing one

Usage

```r
gating_model(
  model = NULL,
  level = 1,
  name,
  signature,
  positive = TRUE,
```
genes.blacklist.default

Blocklist of genes for dimensionality reduction

Description

A list of signatures, for mouse and human. These include cell cycling, heat-shock genes, mitochondrial genes, and other genes classes, that may confound the identification of cell types. These are used internally by scGate and excluded from the calculation of dimensional reductions (PCA).

Format

A list of signatures
**Description**

Download, update or load local version of the scGate model database. These are stored in a GitHub repository, from where you can download specific versions of the database.

**Usage**

```r
get_scGateDB(
  destination = tempdir(),
  force_update = FALSE,
  version = "latest",
  branch = c("master", "dev"),
  verbose = FALSE,
  repo_url = "https://github.com/carmonalab/scGate_models"
)
```

**Arguments**

- `destination`:
  Destination path for storing the DB. The default is `tempdir()`; if you wish to edit locally the models and link them to the current project, set this parameter to a new directory name, e.g. `scGateDB`.

- `force_update`:
  Whether to update an existing database.

- `version`:
  Specify the version of the scGate_models database (e.g. 'v0.1'). By default downloads the latest available version.

- `branch`:
  Branch of the scGate model repository, either 'master' (default) or 'dev' for the latest models.

- `verbose`:
  Display progress messages.

- `repo_url`:
  URL path to scGate model repository database.

**Details**

Models for scGate are dataframes where each line is a signature for a given filtering level. A database of models can be downloaded using the function `get_scGateDB`. You may directly use the models from the database, or edit one of these models to generate your own custom gating model.

**Value**

A list of models, organized according to the folder structure of the database. See the examples below.

**See Also**

`scGate load_scGate_model`
Examples

scGate.model.db <- get_scGateDB()
# To see a specific model, browse the list of models:
scGate.model.db$human$generic$Myeloid

get_testing_data  Download sample data

Description

Helper function to obtain some sample data

Usage

get_testing_data(version = "hsa.latest", destination = tempdir())

Arguments

version  Which sample dataset
destination  Save to this directory

Value

A list of datasets that can be used to test scGate

Examples

testing.datasets <- get_testing_data(version = "hsa.latest")

load_scGate_model  Load a single scGate model

Description

Loads a custom scGate model into R. For the format of these models, have a look or edit one of the default models obtained with get_scGateDB

Usage

load_scGate_model(model_file, master.table = "master_table.tsv")
**performance.metrics**

**Arguments**

- `model_file` scGate model file, in .tsv format.
- `master.table` File name of the master table (in repo_path folder) that contains cell type signatures.

**Value**

A scGate model in dataframe format, which can be given as input to the `scGate` function.

**See Also**

- `scGate`
- `get_scGateDB`

**Examples**

```r
dir <- tempdir() # this may also be set to your working directory
models <- get_scGateDB(destination=dir)
# Original or edited model
model.path <- paste0(dir,"/scGate_models-master/human/generic/Bcell_scGate_Model.tsv")
master.path <- paste0(dir,"/scGate_models-master/human/generic/master_table.tsv")
my.model <- load_scGate_model(model.path, master.path)
my.model
```

---

**performance.metrics**  Performance metrics

**Description**

Evaluate model performance for binary tasks

**Usage**

```r
performance.metrics(actual, pred, return_contingency = FALSE)
```

**Arguments**

- `actual` Logical or numeric binary vector giving the actual cell labels.
- `pred` Logical or numeric binary vector giving the predicted cell labels.
- `return_contingency` Logical indicating if contingency table must be returned.

**Value**

Prediction performance metrics (Precision, Recall, MCC) between actual and predicted cell type labels.
Examples

```r
results <- performance.metrics(actual= sample(c(1,0),20,replace=TRUE),
   pred = sample(c(1,0),20,replace=TRUE,prob = c(0.65,0.35) ) )
```

---

**plot_levels**  
*Plot scGate filtering results by level*

**Description**

Fast plotting of gating results over each model level.

**Usage**

```r
plot_levels(obj, pure.col = "green", impure.col = "gray")
```

**Arguments**

- `obj`  
  Gated Seurat object output of scGate filtering function
- `pure.col`  
  Color code for pure category
- `impure.col`  
  Color code for impure category

**Value**

UMAP plots with 'Pure'/Impure' labels for each level of the scGate model

**Examples**

```r
scGate.model.db <- get_scGateDB()
model <- scGate.model.db$human$generic$Myeloid
# Apply scGate with this model
data(query.seurat)
query.seurat <- scGate(query.seurat, model=model,
   reduction="pca", save.levels=TRUE)
library(patchwork)
pll <- plot_levels(query.seurat)
wrap_plots(pll)
```
**plot_tree**  

**Plot model tree**

**Description**

View scGate model as a decision tree (require ggparty package)

**Usage**

```r
plot_tree(model, box.size = 8, edge.text.size = 4)
```

**Arguments**

- `model`: A scGate model to be visualized
- `box.size`: Box size
- `edge.text.size`: Edge text size

**Value**

A plot of the model as a decision tree. At each level, green boxes indicate the 'positive' (accepted) cell types, red boxes indicate the 'negative' cell types (filtered out). The final pure population is the bottom right subset in the tree.

**Examples**

```r
library(ggparty)
models <- get_scGateDB()
plot_tree(models$human$generic$Tcell)
```

---

**plot_UCell_scores**  

**Plot UCell scores by level**

**Description**

Show distribution of UCell scores for each level of a given scGate model

**Usage**

```r
plot_UCell_scores(
  obj,
  model,
  overlay = 5,
  pos.thr = 0.2,
  neg.thr = 0.2,
  ncol = NULL,
  combine = TRUE
)
```
Arguments

obj          Gated Seurat object (output of scGate)
model        scGate model used to identify a target population in obj
overlay      Degree of overlay for ggridges
pos.thr      Threshold for positive signatures used in scGate model (set to NULL to disable)
eg.neg.thr   Threshold for negative signatures used in scGate model (set to NULL to disable)
ncol         Number of columns in output object (passed to wrap_plots)
combine      Whether to combine plots into a single object, or to return a list of plots

Value

Returns a density plot of UCell scores for the signatures in the scGate model, for each level of the model
Either a plot combined by patchwork (combine=T) or a list of plots (combine=F)

Examples

scGate.model.db <- get_scGateDB()
model <- scGate.model.db$human$generic$Tcell
# Apply scGate with this model
data(query.seurat)
query.seurat <- scGate(query.seurat, model=model,
   reduction="pca", save.levels=TRUE)
# View UCell score distribution
plot_UCell_scores(query.seurat, model)

query.seurat            Toy dataset to test the package

Description

A downsampled version (300 cells) of the single-cell dataset by Zilionis et al. (2019) <doi:10.1016/j.immuni.2019.03.009>, with precalculated PCA and UMAP reductions.

Format

A Seurat object
scGate

Filter single-cell data by cell type

Description

Apply scGate to filter specific cell types in a query dataset

Usage

scGate(
  data,
  model,
  pos.thr = 0.2,
  neg.thr = 0.2,
  assay = NULL,
  slot = "data",
  ncores = 1,
  BPPARAM = NULL,
  seed = 123,
  keep.ranks = FALSE,
  reduction = c("calculate", "pca", "umap", "harmony"),
  min.cells = 30,
  nfeatures = 2000,
  pca.dim = 30,
  param_decay = 0.25,
  maxRank = 1500,
  output.col.name = "is.pure",
  k.param = 30,
  smooth.decay = 0.1,
  smooth.up.only = FALSE,
  genes.blacklist = "default",
  return.CellOntology = TRUE,
  multi.asNA = FALSE,
  additional.signatures = NULL,
  save.levels = FALSE,
  verbose = FALSE,
 progressbar = T
)

Arguments

data Seurat object containing a query data set - filtering will be applied to this object
model A single scGate model, or a list of scGate models. See Details for this format
pos.thr Minimum UCell score value for positive signatures
neg.thr Maximum UCell score value for negative signatures
assay Seurat assay to use
slot  Data slot in Seurat object to calculate UCell scores
ncores  Number of processors for parallel processing
BPPARAM  A [BiocParallel::bpparam()] object that tells scGate how to parallelize. If provided, it overrides the 'ncores' parameter.
seed  Integer seed for random number generator
keep.ranks  Store UCell rankings in Seurat object. This will speed up calculations if the same object is applied again with new signatures.
reduction  Dimensionality reduction to use for knn smoothing. By default, calculates a new reduction based on the given assay; otherwise you may specify a precalculated dimensionality reduction (e.g. in the case of an integrated dataset after batch-effect correction)
min.cells  Minimum number of cells to cluster or define cell types
nfeatures  Number of variable genes for dimensionality reduction
pca.dim  Number of principal components for dimensionality reduction
param_decay  Controls decrease in parameter complexity at each iteration, between 0 and 1. param_decay == 0 gives no decay, increasingly higher param_decay gives increasingly stronger decay
maxRank  Maximum number of genes that UCell will rank per cell
output.col.name  Column name with 'pure/impure' annotation
k.param  Number of nearest neighbors for knn smoothing
smooth.decay  Decay parameter for knn weights: (1-decay)^n
smooth.up.only  If TRUE, only let smoothing increase signature scores
genes.blacklist  Genes blacklisted from variable features. The default loads the list of genes in scGate::genes.blacklist.default; you may deactivate blacklisting by setting genes.blacklist=NULL
return.CellOntology  If TRUE Cell ontology name and id are returned as additional metadata columns when running multiple models.
multi.asNA  How to label cells that are "Pure" for multiple annotations: "Multi" (FALSE) or NA (TRUE)
additional.signatures  A list of additional signatures, not included in the model, to be evaluated (e.g. a cycling signature). The scores for this list of signatures will be returned but not used for filtering.
save.levels  Whether to save in metadata the filtering output for each gating model level
verbose  Verbouse output
progressbar  Whether to show a progressbar or not
Details

Models for scGate are data frames where each line is a signature for a given filtering level. A database of models can be downloaded using the function `get_scGateDB`. You may directly use the models from the database, or edit one of these models to generate your own custom gating model.

Multiple models can also be evaluated at once, by running scGate with a list of models. Gating for each individual model is returned as metadata, with a consensus annotation stored in `scGate_multi` metadata field. This allows using scGate as a multi-class classifier, where only cells that are "Pure" for a single model are assigned a label, cells that are "Pure" for more than one gating model are labeled as "Multi", all others cells are annotated as NA.

Value

A new metadata column `is.pure` is added to the query Seurat object, indicating which cells passed the scGate filter. The `active.ident` is also set to this variable.

See Also

`load_scGate_model` `get_scGateDB` `plot_tree`

Examples

```r
### Test using a small toy set
data(query.seurat)
# Define basic gating model for B cells
my_scGate_model <- gating_model(name = "Bcell", signature = c("MS4A1"))
query.seurat <- scGate(query.seurat, model = my_scGate_model, reduction="pca")
table(query.seurat$is.pure)
### Test with larger datasets
library(Seurat)
testing.datasets <- get_testing_data(version = "hsa.latest")
seurat_object <- testing.datasets[["JerbyArnon"]]
# Download pre-defined models
models <- get_scGateDB()
seurat_object <- scGate(seurat_object, model=models$human$generic$PanBcell)
DimPlot(seurat_object)
seurat_object_filtered <- subset(seurat_object, subset=is.pure="Pure")

### Run multiple models at once
models <- get_scGateDB()
model.list <- list("Bcell" = models$human$generic$Bcell, 
                   "Tcell" = models$human$generic$Tcell)
seurat_object <- scGate(seurat_object, model=model.list)
DimPlot(seurat_object, group.by = "scGate_multi")
```
Description

Wrapper for fast model testing on 3 sampled datasets

Usage

test_my_model(
  model,
  testing.version = "hsa.latest",
  custom.dataset = NULL,
  target = NULL,
  plot = TRUE
)

Arguments

model scGate model in data.frame format

testing.version Character indicating the version of testing tatasets to be used. By default "hsa-
latest" will be used. It will be ignored if a custom dataset is provided (in Seurat format).

custom.dataset Seurat object to be used as a testing dataset. For testing purposes, metadata
seurat object must contain a column named 'cell_type' to be used as a gold standard. Also a set of positive targets must be provided in the target variable.

target Positive target cell types. If default testing version is used this variable must be a
character indicating one of the available target models ('immune','Lymphoid','Myeloid','Tcell','Bcell','CD8T','CD4T','NK','MoMacDC','Plasma_cell','PanBcell'). If a custom dataset is provided
in Seurat format, this variable must be a vector of positive cell types in your
data. The last case also require that such labels were named as in your cell_type
meta.data column.

plot Whether to return plots to device

Value

Returns performance metrics for the benchmarking datasets, and optionally plots of the predicted
cell type labels in reduced dimensionality space.

Examples

scGate.model.db <- get_scGateDB()
# Browse the list of models and select one:
model.panBcell <- scGate.model.db$human$generic$PanBcell
# Test the model with available testing datasets
panBcell.performance <- test_my_model(model.panBcell, target = "PanBcell")

model.Myeloid <- scGate.model.db$human$generic$Myeloid

myeloid.performance <- test_my_model(model.Myeloid, target = "Myeloid")
Index

combine_scGate_multiclass, 2

gating_model, 3

genes.blacklist.default, 4

get_scGateDB, 5, 6, 7, 13

get_testing_data, 6

load_scGate_model, 5, 6, 13

performance.metrics, 7

plot_levels, 8

plot_tree, 9, 13

plot_UCell_scores, 9

query.seurat, 10

scGate, 4, 5, 7, 11

test_my_model, 14