Package ‘scAnnotatR’

February 22, 2024

Type Package
Title Pretrained learning models for cell type prediction on single cell RNA-sequencing data
Version 1.8.0
Description The package comprises a set of pretrained machine learning models to predict basic immune cell types. This enables all users to quickly get a first annotation of the cell types present in their dataset without requiring prior knowledge. scAnnotatR also allows users to train their own models to predict new cell types based on specific research needs.
License MIT + file LICENSE
Encoding UTF-8
biocViews SingleCell, Transcriptomics, GeneExpression, SupportVectorMachine, Classification, Software
Imports dplyr, ggplot2, caret, ROCR, pROC, data.tree, methods, stats, e1071, ape, kernlab, AnnotationHub, utils
Suggests knitr, rmarkdown, scRNAseq, testthat
VignetteBuilder knitr
Depends R (>= 4.1), Seurat, SingleCellExperiment, SummarizedExperiment
LazyData true
RoxygenNote 7.2.3
URL https://github.com/grisslab/scAnnotatR
BugReports https://github.com/grisslab/scAnnotatR/issues/new
git_url https://git.bioconductor.org/packages/scAnnotatR
git_branch RELEASE_3_18
git_last_commit 5367b84
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-02-21
caret_model

Description

Returns the caret model of the scAnnotatR object

Usage

caret_model(classifier)

Arguments

  classifier    scAnnotatR object

Value

Classifier is the object returned by caret SVM learning process. More information about the caret package: https://topepo.github.io/caret/
Examples

data("tirosh_mel80_example")
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "B cells", tag_slot = 'active.ident')
caret_model(classifier_b)

Description

Returns the cell type for the given classifier.

Usage

cell_type(classifier)

## S4 replacement method for signature 'scAnnotatR'

```r
cell_type(classifier) <- value
```

Arguments

- `classifier`: scAnnotatR object. The object is returned from the train_classifier function.
- `value`: the new cell type

Value

- `cell_type` of object

  scAnnotatR object with the new cell type

Examples

```r
data("tirosh_mel80_example")
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "B cells", tag_slot = 'active.ident')
caret_model(classifier_b)
```
```r
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "B cells", tag_slot = 'active.ident')
cell_type(classifier_b) <- "B cell"
```

---

**cell_type**<-  

*Setter for cell_type. Change cell type of a classifier*

---

**Description**

Setter for cell_type. Change cell type of a classifier

**Usage**

`cell_type(classifier) <- value`

**Arguments**

- `classifier` the classifier whose cell type will be changed
- `value` the new cell type

**Value**

the classifier with the new cell type

---

**checkObjectValidity**  

*Internal functions of scAnnotatR package*

---

**Description**

Check if a scAnnotatR object is valid

Train a classifier for a new cell type If cell type has a parent, only available for scAnnotatR object as parent cell classifying model.

Train a classifier for a new cell type If cell type has a parent, only available for scAnnotatR object as parent cell classifying model.

Train a classifier for a new cell type from expression matrix and tag If cell type has a parent, only available for scAnnotatR object as parent cell classifying model.

Preprocess Seurat object to produce expression matrix, tag, parent cell tag.

Preprocess Seurat object to produce expression matrix, tag, parent cell tag.

Testing process when test object is of type Seurat

Testing process when test object is of type SCE

Testing process from matrix and tag

This function ensures that parent classifiers are also selected.
checkObjectValidity

Usage

checkObjectValidity(object)

checkCellTypeValidity(cell_type)

checkMarkerGenesValidity(marker_genes)

checkParentValidity(parent)

checkPThresValidity(p_thres)

checkCaretModelValidity(caret_model)

parent(classifier) <- value

## S4 replacement method for signature 'scAnnotatR'
parent(classifier) <- value

caret_model(classifier) <- value

## S4 replacement method for signature 'scAnnotatR'
caret_model(classifier) <- value

marker_genes(classifier) <- value

## S4 replacement method for signature 'scAnnotatR'
marker_genes(classifier) <- value

train_classifier_seurat(
    train_obj,
    cell_type,
    marker_genes,
    parent_cell = NA_character_,
    parent_classifier = NULL,
    path_to_models = "default",
    zscore = TRUE,
    seurat_tag_slot,
    seurat_parent_tag_slot = "predicted_cell_type",
    seurat_assay,
    seurat_slot,
    ambiguous_chars
)

train_classifier_sce(
    train_obj,
    cell_type,
    marker_genes,
    parent_cell = NA_character_,
    }
parent_classifier = NULL, 
path_to_models = "default", 
zscore = TRUE, 
sce_tag_slot, 
sce_parent_tag_slot = "predicted_cell_type", 
sce_assay, 
ambiguous_chars = NULL 
)

train_classifier_from_mat( 
    mat, 
    tag, 
    cell_type, 
    marker_genes, 
    parent_tag, 
    parent_cell, 
    parent_classifier, 
    path_to_models, 
    zscore, 
    ambiguous_chars = NULL 
)

preprocess_seurat_object( 
    seurat_obj, 
    seurat_assay, 
    seurat_slot, 
    seurat_tag_slot, 
    seurat_parent_tag_slot 
)

preprocess_sce_object(sce_obj, sce_assay, sce_tag_slot, sce_parent_tag_slot)

test_classifier_seurat( 
    test_obj, 
    classifier, 
    target_cell_type = NULL, 
    parent_classifier = NULL, 
    path_to_models = "default", 
    zscore = TRUE, 
    seurat_tag_slot, 
    seurat_parent_tag_slot = "predicted_cell_type", 
    seurat_assay, 
    seurat_slot, 
    ambiguous_chars = NULL 
)

test_classifier_sce( 
    test_obj,
checkObjectValidity

classifier,
target_cell_type = NULL,
parent_classifier = NULL,
path_to_models = "default",
zscore = TRUE,
sce_tag_slot,
sce_parent_tag_slot = "predicted_cell_type",
sce_assay,
ambiguous_chars = NULL
)

test_classifier_from_mat(
  mat,
tag,
classifier,
parent_tag,
target_cell_type,
parent_classifier,
path_to_models,
zscore,
ambiguous_chars = NULL
)

classify_cells_seurat(
  classify_obj,
classifiers = NULL,
cell_types = "all",
chunk_size = 5000,
path_to_models = "default",
ignore_ambiguous_result = FALSE,
cluster_slot,
seurat_assay,
seurat_slot
)

classify_cells_sce(
  classify_obj,
classifiers = NULL,
cell_types = "all",
chunk_size = 5000,
path_to_models = "default",
ignore_ambiguous_result = FALSE,
sce_assay,
cluster_slot = NULL
)

balance_dataset(mat, tag)
checkObjectValidity

train_func(mat, tag)

transform_to_zscore(mat)

subset_models(model_list, model_names)

select_marker_genes(mat, marker_genes)

check_parent_child_coherence(
    mat,
    tag,
    pos_parent,
    parent_cell,
    cell_type,
    target_cell_type
)

filter_cells(mat, tag, ambiguous_chars = NULL)

construct_tag_vect(tag, cell_type)

process_parent_classifier(
    mat,
    parent_tag,
    parent_cell_type,
    parent_classifier,
    path_to_models,
    zscore
)

make_prediction(mat, classifier, pred_cells, ignore_ambiguous_result = TRUE)

simplify_prediction(meta.data, full_pred, classifiers)

verify_parent(mat, classifier, meta.data)

test_performance(mat, classifier, tag)

classify_clust(clusts, most_probable_cell_type)

download_data_file(verbosе = FALSE)

Arguments

- **object**: The request classifier to check.
- **cell_type**: name of cell type
- **marker_genes**: list of selected marker genes
- **parent**: Classifier parent to check.
checkObjectValidity

- **p_thres**: Classifier probability threshold to check.
- **caret_model**: Classifier to check.
- **classifier**: The new classifier
- **class**: Classifier
- **value**: the new classifier
- **train_obj**: SCE object
- **parent_cell**: name of parent cell type
- **parent_classifier**: scAnnotatR object corresponding to classification model for the parent cell type
- **path_to_models**: path to databases, or by default
- **zscore**: boolean indicating the transformation of gene expression in object to zscore or not
- **seurat_tag_slot**: string, name of annotation slot indicating cell tag/label in the testing object. Strings indicating cell types are expected in this slot. Expected values are string (A-Z, a-z, 0-9, no special character accepted) or binary/logical, 0/"no"/F/FALSE: not being new cell type, 1/"yes"/T/TRUE: being new cell type.
- **seurat_parent_tag_slot**: string, name of tag slot in cell meta data indicating pre-assigned/predicted parent cell type. Default field is "predicted_cell_type". The slot must contain only string values.
- **seurat_assay**: name of assay to use in Seurat object
- **seurat_slot**: type of expression data to use in Seurat object. Some available types are: "counts", "data" and "scale.data".
- **ambiguous_chars**: Vector of character (sequences) that if contained within a cell type mark this cell type as being ambiguous. If NULL default values are used. Characters with a meaning in REGEX must be enclosed by []. E.g. "[+]", "[.]", "-". "[+]", "[.]", " and ", " or ", "_or_", "-or-", "[(]", "[)]". "ambiguous"
- **sce_tag_slot**: string, name of annotation slot indicating cell tag/label in the testing object. Strings indicating cell types are expected in this slot. Expected values are string (A-Z, a-z, 0-9, no special character accepted) or binary/logical, 0/"no"/F/FALSE: not being new cell type, 1/"yes"/T/TRUE: being new cell type.
- **sce_parent_tag_slot**: string, name of tag slot in cell meta data indicating pre-assigned/predicted parent cell type. Default field is "predicted_cell_type". The slot must contain only string values.
- **sce_assay**: name of assay to use in SCE object
- **mat**: expression matrix
- **tag**: tag of data
- **parent_tag**: vector, named list indicating pre-assigned/predicted parent cell type
- **seurat_obj**: Seurat object
- **sce_obj**: Seurat object
- **test_obj**: SCE object used for testing
checkObjectValidity

target_cell_type
alternative cell types (in case of testing classifier)
classify_obj
the SCE object containing cells to be classified
classifiers
classifiers
cell_types
list of cell types containing models to be used for classification, only applicable if the models have been saved to package.
chunk_size
size of data chunks to be predicted separately. This option is recommended for large datasets to reduce running time. Default value at 5000, because smaller datasets can be predicted rapidly.
ignore_ambiguous_result
whether ignore ambiguous result
cluster_slot
name of slot in meta data containing cluster information, in case users want to have additional cluster-level prediction
model_list
A list of models
model_names
The names of the models to retain
pos_parent
a vector indicating parent classifier prediction
parent_cell_type
name of parent cell type
pred_cells
a whole prediction for all cells
meta.data
object meta data
full_pred
full prediction
clusts
cluster info
most_probable_cell_type
predicted cell type
verbose
logical indicating downloading the file or not

Value

TRUE if the classifier is valid or the reason why it is not
TRUE if the cell type is valid or the reason why it is not.
TRUE if the marker.genes is valid or the reason why it is not.
TRUE if the parent is valid or the reason why it is not.
TRUE if the p_thres is valid or the reason why it is not.
the classifier with the new parent.
scAnnotatR object with the new parent
the classifier with the new core caret model.
scAnnotatR object with the new trained classifier.
the classifier with the new marker genes
scAnnotatR object with the new marker genes.
scAnnotatR object
**classify_cells**

**Description**

Classify cells from multiple models
Usage

classify_cells(
    classify_obj,
    assay,
    slot = NULL,
    classifiers = NULL,
    cell_types = "all",
    chunk_size = 5000,
    path_to_models = "default",
    ignore_ambiguous_result = FALSE,
    cluster_slot = "clusters"
)

Arguments

classify_obj  the object containing cells to be classified
assay         name of assay to use in classify_object
slot          type of expression data to use in classify_object. For Seurat object, some available types are: "counts", "data" and "scale.data".
classifiers   list of classification models. The model is obtained from train_classifier function or available in current working space. Users may test the model using test_classifier before using this function. If classifiers contain classifiers for subcell types, classifiers for parent cell type must be indicated first in order to be applied before children classifiers. If classifiers is NULL, the method will use all classifiers in database.
cell_types    list of cell types containing models to be used for classification, only applicable if the models have been saved to package.
chunk_size    size of data chunks to be predicted separately. This option is recommended for large datasets to reduce running time. Default value at 5000, because smaller datasets can be predicted rapidly.
path_to_models path to the folder containing the list of models. As default value, the pretrained models in the package will be used. If user has trained new models, indicate the folder containing the new_models.rda file.
ignore_ambiguous_result return all ambiguous predictions (multiple cell types) to empty When this parameter turns to TRUE, most probably predicted cell types will be ignored.
cluster_slot  name of slot in meta data containing cluster information, in case users want to have additional cluster-level prediction

Value

the input object with new slots in cells meta data New slots are: predicted_cell_type, most_probable_cell_type, slots in form of [cell_type]_p, [cell_type]_class, and clust_pred (if cluster_slot was provided).
**Examples**

```r
# load small example dataset
data("tirosd_mel80_example")

# train one classifier for one cell type, for ex, B cell
# define genes to use to classify this cell type
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")

# train the classifier
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "b cells", tag_slot = 'active.ident')

# do the same thing with other cell types, for example, T cells
selected_marker_genes_T = c("CD4", "CD8A", "CD8B")
set.seed(123)
classifier_t <- train_classifier(train_obj = tirosd_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_T,
cell_type = "T cells", tag_slot = 'active.ident')

# create a list of classifiers
classifier_ls <- list(classifier_b, classifier_t)

# classify cells with list of classifiers
seurat.obj <- classify_cells(classify_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', classifiers = classifier_ls)
```

---

**delete_model**  
Delete model/branch from package

**Description**
Delete model/branch from package

**Usage**
deleate_model(cell_type, path_to_models = tempdir())

**Arguments**
cell_type  
string indicating the cell type of which the model will be removed from package
Attention: deletion of a parent model will also delete all of child model.

path_to_models  
path to the folder containing the list of models in which the to-be-deleted model is.

**Value**
no return value, but the model is deleted from database
Examples

# load small example dataset
data("tirosh_mel80_example")

# train a classifier
set.seed(123)
selected_marker_genes_T = c("CD4", "CD8A", "CD8B")
classifier_t <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_T,
cell_type = "t cells", tag_slot = 'active.ident')

# save a classifier to system
save_new_model(new_model = classifier_t, path_to_models = tempdir(),
include.default = FALSE)

# delete classifier from system
delete_model("t cells", path_to_models = tempdir())

load_models Load classifiers from databases

Description

Load classifiers from databases

Usage

load_models(path_to_models)

Arguments

path_to_models path to databases, or by default

Value

list of classifiers

marker_genes marker_genes

Description

Returns the set of marker genes for the given classifier.

Usage

marker_genes(classifier)
parent

Arguments

| classifier | scAnnotatR object |

Value

Applied marker genes of object

Examples

data("tirosh_mel80_example")
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "B cells", tag_slot = 'active.ident')
marker_genes(classifier_b)

parent

Description

Returns the parent of the cell type corresponding to the given classifier.

Usage

parent(classifier)

Arguments

| classifier | scAnnotatR object |

Value

Parent model of object

Examples

data("tirosh_mel80_example")
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "B cells", tag_slot = 'active.ident')
parent(classifier_b)
### plant_tree

**Plant tree from list of models**

**Description**

Plant tree from list of models

**Usage**

```r
plant_tree(path_to_models = "default")
```

**Arguments**

- `path_to_models`: list of models. If not provided, list of default pretrained models in the package will be used.

**Value**

tree structure and plot of tree

**Examples**

```r
# to create the tree of classifiers
# (in this example, based on default classifiers)
plant_tree()
```

### plot_roc_curve

**Plot roc curve**

**Description**

Plot roc curve

**Usage**

```r
plot_roc_curve(test_result)
```

**Arguments**

- `test_result`: result of test_classifier function

**Value**

ggplot2 roc curve
Examples

```r
# load small example dataset
data("tirosis_mel80_example")

# train a classifier, for ex: B cell
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = "RNA", slot = "counts", marker_genes = selected_marker_genes_B,
cell_type = "b cells", tag_slot = "active.ident")

classifier_b_test <- test_classifier(classifier = classifier_b,
test_obj = tirosh_mel80_example, assay = "RNA", slot = "counts",
tag_slot = "active.ident", target_cell_type = c("B cell"))

# run plot curve on the test result
roc_curve <- plot_roc_curve(test_result = classifier_b_test)
```

---

### Description

Returns the probability threshold for the given classifier.

### Usage

```r
p_thres(classifier)
```

### Arguments

- `classifier` scAnnotatR object. The object is returned from the `train_classifier` function.
- `value` the new threshold

### Value

Predicting probability threshold of object scAnnotatR object with the new threshold.
Examples

data("tirosh_mel80_example")
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
    cell_type = "B cells", tag_slot = 'active.ident')
p_thres(classifier_b)

data("tirosh_mel80_example")
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
    cell_type = "B cells", tag_slot = 'active.ident')
classifier_b_test <- test_classifier(classifier = classifier_b,
    test_obj = tirosh_mel80_example, assay = 'RNA', slot = 'counts',
    tag_slot = 'active.ident')
# assign a new threshold probability for prediction
p_thres(classifier_b) <- 0.4

---

### p_thres<-

**Setter for predicting probability threshold**

**Description**

Setter for predicting probability threshold

**Usage**

p_thres(classifier) <- value

**Arguments**

- **classifier** the classifier whose predicting probability threshold will be changed
- **value** the new threshold

**Value**

classifier with the new threshold.
Description

Save a model to the package

Usage

save_new_model(new_model, include.default = TRUE, path_to_models = tempdir())

Arguments

new_model new model to be added into the classification tree
include.default whether include the default models of the package in the list of new trained models or not. If users further want to classify cells, they can only use default pretrained model list or their new model list. Including default models in new trained models helps users using both of them once. In addition, default pretrained models of the package cannot be changed or removed. This can be done with the new trained model list.
path_to_models path to the folder containing the list of new models.

Value

no return value, but the model is now saved to database

Examples

# load small example dataset
data("tirosh_mel80_example")

# train classifier
selected_marker_genes_T = c("CD4", "CD8A", "CD8B")
set.seed(123)
classifier_t <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_T,
cell_type = "t cells", tag_slot = 'active.ident')

# save the trained classifier to system
# test classifier can be used before this step
# Note: We do not include the default models here to runtime of the example
save_new_model(new_model = classifier_t, path_to_models = tempdir(),
include.default = FALSE)

# verify if new model has been saved
print(names(load(file.path(tempdir(), "new_models.rda"))))
delete_model("t cells")
scAnnotatR

scAnnotatR class.

Description

This class is returned by the train_classifier function. Generally, scAnnotatR objects are never created directly.

Usage

scAnnotatR(cell_type, caret_model, marker_genes, p_thres, parent)

Arguments

- cell_type: character. Name of the cell type.
- caret_model: list. Trained model returned by caret train function.
- marker_genes: vector/character containing marker genes used for the training.
- p_thres: numeric. Probability threshold for the cell type to be assigned for a cell.
- parent: character. Parent cell type.

Value

A scAnnotatR object.

Slots

- cell_type: character. Name of the cell type.
- caret_model: list. Trained model returned by caret train function.
- marker_genes: vector/character containing marker genes used for the training.
- p_thres: numeric. Probability threshold for the cell type to be assigned for a cell.
- parent: character. Parent cell type.

Examples

# load small example dataset
data("tirosh_mel80_example")

# train a classifier, for ex: B cell
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = "RNA", slot = "counts", marker_genes = selected_marker_genes_B,
cell_type = "B cells", tag_slot = "active.ident")

classifier_b
### Description

Show object

### Usage

```r
## S4 method for signature 'scAnnotatR'
show(object)
```

### Arguments

- `object`: scAnnotatR object

### Value

print to console information about the object

### Examples

```r
data("tirosh_mel80_example")
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "B cells", tag_slot = 'active.ident')
classifier_b
```

----

test_classifier  

---

### Description

Testing process.
Usage

test_classifier(
    classifier,
    test_obj,
    assay,
    slot = NULL,
    tag_slot,
    target_cell_type = NULL,
    parent_classifier = NULL,
    parent_tag_slot = "predicted_cell_type",
    path_to_models = "default",
    zscore = TRUE,
    ambiguous_chars = NULL
)

## S4 method for signature 'scAnnotatR'

test_classifier(
    classifier,
    test_obj,
    assay,
    slot = NULL,
    tag_slot,
    target_cell_type = NULL,
    parent_classifier = NULL,
    parent_tag_slot = "predicted_cell_type",
    path_to_models = "default",
    zscore = TRUE,
    ambiguous_chars = NULL
)

Arguments

classifier scAnnotatR classification model

test_obj object that can be used for testing

assay name of assay to use in test_object

slot type of expression data to use in test_object. For Seurat object, some available types are: "counts", "data" and "scale.data". Ignore this if test_obj is SingleCellExperiment object.

tag_slot string, name of annotation slot indicating cell tag/label in the testing object. Strings indicating cell types are expected in this slot. Expected values are string (A-Z, a-z, 0-9, no special character accepted) or binary/logical, 0/no/F/FALSE: not being new cell type, 1/yes/T/TRUE: being new cell type.

target_cell_type vector indicating other cell types than cell labels that can be considered as the main cell type in classifier, for example, c("plasma cell", "b cell", "b cells", "activating b cell"). Default as NULL.
parent_classifier

`scAnnotatR` object corresponding to classification model for the parent cell type

parent_tag_slot

string, name of tag slot in cell meta data indicating pre-assigned/predicted parent cell type. Default field is "predicted_cell_type". The slot must contain only string values.

path_to_models

path to the folder containing the list of models. As default, the pretrained models in the package will be used. If user has trained new models, indicate the folder containing the new_models.rda file.

zscore

boolean, whether gene expression is transformed to zscore

ambiguous_chars

List of characters to indicate ambiguous cells. For more details see `filter_cells`.

Value

result of testing process in form of a list, including predicted values, prediction accuracy at a probability threshold, and roc curve information.

Note

Only one cell type is expected for each cell. Ambiguous cell type, such as: "T cells/NK cells/ILC", will be ignored. Subtypes used in testing model for parent cell types can be indicated as parent cell type, or can be indicated in target_cell_type. For example, when testing for B cells, plasma cells can be annotated as B cells, or target_cell_type is set c(“plasma cells”).

Examples

# load small example dataset
data("tirosh_mel80_example")

# train the classifier
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "b cells", tag_slot = 'active.ident')

# test the classifier, target cell type can be in other formats or
# alternative cell type that can be considered as the classified cell type
classifier_b_test <- test_classifier(classifier = classifier_b,
test_obj = tirosh_mel80_example, assay = 'RNA', slot = 'counts',
tag_slot = 'active.ident', target_cell_type = c("B cell"))
classifier_b_test
**tirosh_mel80_example**  
*A Seurat Object Sample*

**Description**

An example Seurat object shipped with the package as an example data. The expression data was originally from the dataset GSE72056, with samples corresponding to patient CY80. The Seurat object was then adapted to be used in *scAnnotatR*.

**Usage**

```r
tirosh_mel80_example
```

**Format**

a *Seurat* object

**Author(s)**

Itay Tirosh, 2016-04-05

**Source**

WEIZMANN INSTITUTE OF SCIENCE

---

**train_classifier**  
*Train cell type classifier*

**Description**

Train a classifier for a new cell type. If cell type has a parent, only available for *scAnnotatR* object as parent cell classifying model.

**Usage**

```r
train_classifier(
  train_obj,
  assay,
  slot = NULL,
  cell_type,
  marker_genes,
  tag_slot,
  parent_cell = NA_character_,
  parent_tag_slot = "predicted_cell_type",
  parent_classifier = NULL,
  path_to_models = "default",
)```
train_classifier

zscore = TRUE,
ambiguous_chars = NULL
)

Arguments

train_obj object that can be used for training the new model. Seurat object or SingleCellExperiment object is supported. If the training model has parent, parent_tag_slot may have been indicated. This field would have been filled out automatically if user prece-dently run classify_cells function. If no (predicted) cell type annotation provided, the function can be run if 1- parent_cell or 2- parent_classifier is pro-

assay name of assay to use in training object.

slot type of expression data to use in training object, omitted if train_obj is SingleCellExperiment object.

cell_type string indicating the name of the subtype This must exactly match cell tag/label if cell tag/label is a string.

marker_genes list of marker genes used for the new training model

tag_slot string, name of slot in cell meta data indicating cell tag/label in the training object. Strings indicating cell types are expected in this slot. For Seurat object, default value is "active.ident". Expected values are string (A-Z, a-z, 0-9, no special character accepted) or binary/logical, 0/no/F/FALSE: not being new cell type, 1/yes/T/TRUE: being new cell type.

parent_cell string indicated the name of the parent cell type, if parent cell type classifier has already been saved in model database. Adjust path_to_models for exact database.

parent_tag_slot string, name of a slot in cell meta data indicating assigned/predicted cell type. Default is "predicted_cell_type". This slot would have been filled automatically if user have called classify_cells function. The slot must contain only string values.

parent_classifier classification model for the parent cell type

path_to_models path to the folder containing the model database. As default, the pretrained models in the package will be used. If user has trained new models, indicate the folder containing the new_models.rda file.

zscore whether gene expression in train_obj is transformed to zscore

ambiguous_chars List of characters to indicate ambiguous cells. For more details see filter_cells.

Value

scAnnotatR object
Note

Only one cell type is expected for each cell in object. Ambiguous cell type, such as: "T cells/NK cells/ILC", will be ignored from training. Subtypes used in training model for parent cell types must be indicated as parent cell type. For example, when training for B cells, plasma cells must be annotated as B cells in order to be used.

Examples

```r
# load small example dataset
data("tirosh_mel80_example")

# this dataset already contains pre-defined cell labels
table(Seurat::Idents(tirosh_mel80_example))

# define genes to use to classify this cell type (B cells in this example)
selected_marker_genes_B = c("CD19", "MS4A1", "CD79A")

# train the classifier, the "cell_type" argument must match
# the cell labels in the data, except upper/lower case
set.seed(123)
classifier_b <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', marker_genes = selected_marker_genes_B,
cell_type = "b cells", tag_slot = 'active.ident')

# classify cell types using B cell classifier,
# a test classifier process may be used before applying the classifier
tirosh_mel80_example <- classify_cells(classify_obj = tirosh_mel80_example,
classifiers = c(classifier_b), assay = 'RNA', slot = 'counts')

# tag all cells that are plasma cells (random example here)
tirosh_mel80_example[["plasma_cell_tag"]]
<- c(rep(1, 80), rep(0, 400))

# set new marker genes for the subtype
p_marker_genes = c("SDC1", "CD19", "CD79A")

# train the classifier, the "B cell" classifier is used as parent.
# This means, only cells already classified as "B cells" will be evaluated.
# the "tag_slot" parameter tells the classifier to use this cell meta data
# for the training process.
set.seed(123)
plasma_classifier <- train_classifier(train_obj = tirosh_mel80_example,
assay = 'RNA', slot = 'counts', cell_type = 'Plasma cell',
marker_genes = p_marker_genes, tag_slot = 'plasma_cell_tag',
parent_classifier = classifier_b)
```
Index

* datasets
  tirosh_mel80_example, 24

balance_dataset (checkObjectValidity), 4

caret_model, 2
caret_model<- (checkObjectValidity), 4
caret_model<-, scAnnotatR-method (checkObjectValidity), 4
cell_type, 3
cell_type<-, 4
cell_type<-, scAnnotatR-method (cell_type), 3
check_parent_child_coherence (checkObjectValidity), 4
checkCaretModelValidity (checkObjectValidity), 4
checkCellTypeValidity (checkObjectValidity), 4
checkMarkerGenesValidity (checkObjectValidity), 4
checkObjectValidity, 4
checkParentValidity (checkObjectValidity), 4
checkPTThresValidity (checkObjectValidity), 4
classify_cells, 11
classify_cells_sce (checkObjectValidity), 4
classify_cells_seurat (checkObjectValidity), 4
classify_clust (checkObjectValidity), 4
construct_tag_vect (checkObjectValidity), 4

delete_model, 13
download_data_file (checkObjectValidity), 4

filter_cells (checkObjectValidity), 4

load_models, 14

make_prediction (checkObjectValidity), 4
marker_genes, 14
marker_genes<- (checkObjectValidity), 4
marker_genes<-, scAnnotatR-method (checkObjectValidity), 4
p_thres, 17
p_thres<-, 18
p_thres<-, scAnnotatR-method (p_thres), 17
parent, 15
parent<- (checkObjectValidity), 4
parent<-, scAnnotatR-method (checkObjectValidity), 4
plant_tree, 16
plot_roc_curve, 16
preprocess_sce_object (checkObjectValidity), 4
preprocess_seurat_object (checkObjectValidity), 4
process_parent_classifier (checkObjectValidity), 4
save_new_model, 19
scAnnotatR, 2, 4, 9–11, 20, 23–25
select_marker_genes (checkObjectValidity), 4
Seurat, 24, 25
show, scAnnotatR-method, 21
simplify_prediction (checkObjectValidity), 4
SingleCellExperiment, 22, 25
subset_models (checkObjectValidity), 4
test_classifier, 21
test_classifier, scAnnotatR-method (test_classifier), 21
test_classifier_from_mat (checkObjectValidity), 4

train_classifier, 20, 24

train_classifier_from_mat (checkObjectValidity), 4

train_classifier_sce (checkObjectValidity), 4

train_classifier_seurat (checkObjectValidity), 4

train_func (checkObjectValidity), 4

transform_to_zscore (checkObjectValidity), 4

verify_parent (checkObjectValidity), 4