Package 'adverSCarial'

March 26, 2023

Title adverSCarial, generate and analyze the vulnerability of scRNA-seq classifiers to adversarial attacks

Version 0.99.0

Description

adverSCarial is an R Package designed for generating and analyzing the vulnerability of scRNA-seq
classifiers to adversarial attacks. The package is versatile and provides a format for integrating any type of classifier. It offers functions for studying and generating two types of attacks,
min change attack and max change attack. The min change attack involves making a small modification
to the input to alter the classification. The max change attack involves making a large modification to the input without changing its classification.
The package provides a comprehensive solution for evaluating the robustness of scRNA-seq classifiers
against adversarial attacks.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

biocViews Software, SingleCell, Transcriptomics, Classification

Suggests knitr, RUnit, BiocGenerics, TENxPBMCData

Imports gtools, stringr, randomForest

VignetteBuilder knitr

Author Ghislain FIEVET <ghislain.fievet@gmail.com>

Maintainer Ghislain FIEVET <ghislain.fievet@gmail.com>

R topics documented:

`advGridMinChange` *Grid search of min change adversarial attack. Tries each combination on a cluster, given a list of genes and a list of modifications.*

Description

Grid search of min change adversarial attack. Tries each combination on a cluster, given a list of genes and a list of modifications.

Usage

```
advGridMinChange(
  exprs,
  clusters,
  target,
  classifier,
  genes,
  modifications = list(list("perc1"), list("perc99")),
  return_first_found = FALSE,
  verbose = FALSE,
  iamsure = FALSE
)
```

Arguments

`exprs`	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.
`clusters`	a list of the clusters to which the cells belong
`target`	the name of the cluster to modify
`classifier`	a classifier in the suitable format
`genes`	the list of genes to study
`modifications`	the list of the modifications to study
`return_first_found`	set to TRUE to return result when a the first misclassification is found
`verbose`	logical, set to TRUE to activate verbose mode
`iamsure`	logical, prevents from expansive calculations when `genes` list is too long, set to `TRUE` to run anyway.

Value

dataframe results of the classification of all the grid combinations

Examples

```
MyClassifier <- function(expr, clusters, target) {
   c("T cell", 0.9)
}
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

advGridMinChange(rna_expression, clusters_id, "T cell",
MyClassifier, genes=genes,
modifications = list(list("perc1"), list("perc99")))
```

---

advMaxChange *Find a max change adversarial attack. It finds the longer list of genes you can modify on a cluster without changing its classification.*

Description

Find a max change adversarial attack. It finds the longer list of genes you can modify on a cluster without changing its classification.

Usage

```
advMaxChange(
  exprs,
  clusters,
  target,
  classifier,
  excl_genes = c(),
  genes = c(),
  adv_method = "perc99",
  adv_fixed_value = 3,
  adv_fct = NULL,
  max_split_size = 1,
  verbose = FALSE
)
```

Arguments

exprs	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.
clusters	a list of the clusters to which the cells belong

target	the name of the cluster to modify
classifier	a classifier in the suitable format
excl_genes	a list of genes to exclude from the analysis
genes	a list of genes in case you want to limit the attack on a subset of genes
adv_method	the name of the method to use
adv_fixed_value	the numeric value to use in case of adv_method=fixed
adv_fct	the function to use in case adv_method belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct
max_split_size	max size of dichotomic slices.
verbose	logical, set to TRUE to activate verbose mode

Value

a list of genes you can modify on a cluster without modifying its classification

Examples

```
MyClassifier <- function(expr, clusters, target) {
   c("T cell", 0.9)
}
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

advMaxChange(rna_expression, clusters_id,
"T cell", MyClassifier, adv_method="perc99")
```

---

advMinChange

Find a one gene min change adversarial attack list. A one gene min change adversarial attack refers to the modification of a single gene within a cluster, leading to a change in its classification. The function returns a list of genes/new classification.

Description

Find a one gene min change adversarial attack list. A one gene min change adversarial attack refers to the modification of a single gene within a cluster, leading to a change in its classification. The function returns a list of genes/new classification.

Usage

```
advMinChange(
  exprs,
  clusters,
  target,
  classifier,
  excl_genes = c(),
  genes = c(),
  adv_method = "perc99",
  adv_fixed_value = 3,
  adv_fct = NULL,
  first_dichot = 100,
  max_split_size = 1,
  return_first_found = FALSE,
  change_type = "any",
  verbose = FALSE
)
```

Arguments

exprs	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.
clusters	a list of the clusters to which the cells belong
target	the name of the cluster to modify
classifier	a classifier in the suitable format
excl_genes	a list of genes to exclude from the analysis
genes	a list of genes in case you want to limit the attack on a subset of genes
adv_method	the name of the method to use
adv_fixed_value	the numeric value to use in case of adv_method=fixed
adv_fct	the function to use in case adv_method belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct
first_dichot	the initial number of slices before the dichotomic search
max_split_size	max size of dichotomic slices
return_first_found	set to TRUE to return result when a the first misclassification is found
change_type	any consider each misclassification, not_na consider each misclassification but NA.
verbose	logical, set to TRUE to activate verbose mode

Value

a list of genes/new classification tuples

Examples

```
MyClassifier <- function(expr, clusters, target) {
   c("T cell", 0.9)
}
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

advMinChange(rna_expression, clusters_id,
"T cell", MyClassifier, adv_method="perc99")
```

`advModifications` *Returns a modified RNA expression matrix, for a given cluster, for a given modification.*

Description

Returns a modified RNA expression matrix, for a given cluster, for a given modification.

Usage

```
advModifications(
  exprs,
  genes,
  clusters,
  target,
  adv_method = "perc99",
  adv_fixed_value = 3,
  adv_fct = NULL,
  verbose = FALSE
)
```

Arguments

`exprs`	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.
`genes`	the list of genes to modify
`clusters`	a list of the clusters to which the cells belong
`target`	the name of the cluster to modify
`adv_method`	the name of the method to use
`adv_fixed_value`	the numeric value to use in case of adv_method=`fixed`
`adv_fct`	the function to use in case adv_method belongs to the following list: `full_row_fct`, `target_row_fct`, `target_matrix_fct`, `full_matrix_fct`
`verbose`	logical, set to TRUE to activate verbose mode

Value

the matrix or a dataframe exprs modified on asked genes with the specified modification

Examples

```
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

advModifications(rna_expression, genes, clusters_id,
"T cell", adv_method="perc99")
```

`advRandWalkMinChange` *Random walk search of min change adversarial attack. Step 1 is to find a seed by trying random combinations of genes and modifications on a cluster until the classification is altered. Step 2 is to perform a random walk search to reduce the number of genes needed to change the classification."*

Description

Random walk search of min change adversarial attack. Step 1 is to find a seed by trying random combinations of genes and modifications on a cluster until the classification is altered. Step 2 is to perform a random walk search to reduce the number of genes needed to change the classification."

Usage

```
advRandWalkMinChange(
  exprs,
  clusters,
  target,
  classifier,
  genes,
  modifications = list(list("perc1"), list("perc99")),
  first_batch = 100,
  walk_length = 100,
  step_change_ratio = 0.2,
  while_max_count = 10000,
  change_type = "any",
  verbose = FALSE
)
```

Arguments

`exprs`	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.

clusters	a list of the clusters to which the cells belong
target	the name of the cluster to modify
classifier	a classifier in the suitable format
genes	the list of genes to study
modifications	the list of the modifications to study
first_batch	the maximum number of try in step 1
walk_length	the maximum number of try in step 2
step_change_ratio	ratio of parameters change in new walk step
while_max_count	the maximum number of try when looking for new combination of parameters
change_type	any consider each misclassification, not_na consider each misclassification but NA.
verbose	logical, set to TRUE to activate verbose mode

Value

dataframe results of the classification of all the grid combinations

Examples

```
MyClassifier <- function(expr, clusters, target) {
   c("T cell", 0.9)
}
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

advRandWalkMinChange(rna_expression, clusters_id, "T cell",
MyClassifier, genes=genes,
modifications = list(list("perc1"), list("perc99")))

# Stop at first attack discovery, whitout going into the walk
# parameter search.
advRandWalkMinChange(rna_expression, clusters_id, "T cell",
MyClassifier, genes=genes,
modifications = list(list("perc1"), list("perc99")), walk_length=0)
```

---

matrixFromSCE *Returns the RNA expression matrix from a SingleCellExperiment with unique hgnc gene names in columns*

Description

Returns the RNA expression matrix from a SingleCellExperiment with unique hgnc gene names in columns

Usage

```
matrixFromSCE(sce)
```

Arguments

sce	SingleCellExperiment object to convert

Value

the RNA expression matrix from a SingleCellExperiment with unique hgnc gene names in columns

Examples

```
library(TENxPBMCData)

pbmc <- TENxPBMCData(dataset = "pbmc3k")
mat_rna <- matrixFromSCE(pbmc)
```

---

maxChangeOverview *Run an approximation of advMaxChange on all clusters for a given modification. The precision of the approximation can be controlled using the* max_split_size *parameter, with lower values resulting in greater precision but longer processing time. The purpose of this approximation is to determine which clusters are most susceptible to max change adversarial attacks.*

Description

Run an approximation of advMaxChange on all clusters for a given modification. The precision of the approximation can be controlled using the max_split_size parameter, with lower values resulting in greater precision but longer processing time. The purpose of this approximation is to determine which clusters are most susceptible to max change adversarial attacks.

Usage

```
maxChangeOverview(
  exprs,
  clusters,
  classifier,
  excl_genes = c(),
  genes = c(),
  modifications = list(list("perc1"), list("perc99")),
  adv_method = "perc99",
  adv_fixed_value = 3,
  adv_fct = NULL,
  max_split_size = 100,
  verbose = FALSE
)
```

Arguments

exprs	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.
clusters	a list of the clusters to which the cells belong
classifier	a classifier in the suitable format
excl_genes	a list of genes to exclude from the analysis
genes	a list of genes in case you want to limit the analysis on a subset of genes
modifications	the list of the modifications to study
adv_method	the name of the method to use
adv_fixed_value	the numeric value to use in case of adv_method=fixed
adv_fct	the function to use in case adv_method belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct
max_split_size	max size of dichotomic slices.
verbose	logical, set to TRUE to activate verbose mode

Value

a list of genes/new classification tuples

Examples

```
MyClassifier <- function(expr, clusters, target) {
   c("T cell", 0.9)
}
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

maxChangeOverview(rna_expression, clusters_id,
MyClassifier, modifications = list(list("perc1"), list("perc99")))
```

---

minChangeOverview *Run an approximation of advMinChange on all clusters for a given modification. The precision of the approximation can be controlled using the* max_split_size *parameter, with lower values resulting in greater precision but longer processing time. The purpose of this approximation is to determine which clusters are most susceptible to min change adversarial attacks.*

Description

Run an approximation of advMinChange on all clusters for a given modification. The precision of the approximation can be controlled using the max_split_size parameter, with lower values resulting in greater precision but longer processing time. The purpose of this approximation is to determine which clusters are most susceptible to min change adversarial attacks.

Usage

```
minChangeOverview(
  exprs,
  clusters,
  classifier,
  excl_genes = c(),
  genes = c(),
  modifications = list(list("perc1"), list("perc99")),
  adv_method = "perc99",
  adv_fixed_value = 3,
  adv_fct = NULL,
  first_dichot = 100,
  max_split_size = 100,
  change_type = "any",
  verbose = FALSE
)
```

Arguments

exprs	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.
clusters	a list of the clusters to which the cells belong
classifier	a classifier in the suitable format
excl_genes	a list of genes to exclude from the analysis
genes	a list of genes in case you want to limit the analysis on a subset of genes
modifications	the list of the modifications to study
adv_method	the name of the method to use
adv_fixed_value	the numeric value to use in case of adv_method=fixed
adv_fct	the function to use in case adv_method belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct
first_dichot	the initial number of slices before the dichotomic search
max_split_size	max size of dichotomic slices.
change_type	any consider each misclassification, not_na consider each misclassification but NA.
verbose	logical, set to TRUE to activate verbose mode

Value

a list of genes/new classification tuples

Examples

```
MyClassifier <- function(expr, clusters, target) {
   c("T cell", 0.9)
```

```
}
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

minChangeOverview(rna_expression, clusters_id,
MyClassifier, modifications = list(list("perc1"), list("perc99")))
```

`predictWithNewValue` *Returns a classification and an odd value from a RNA expression matrix, for given genes, for a given cluster, for a given modification.*

Description

Returns a classification and an odd value from a RNA expression matrix, for given genes, for a given cluster, for a given modification.

Usage

```
predictWithNewValue(
  exprs,
  genes,
  clusters,
  target,
  classifier,
  adv_method = "perc99",
  adv_fixed_value = 3,
  adv_fct = NULL,
  verbose = FALSE
)
```

Arguments

`exprs`	a matrix or dataframe of numeric RNA expression, cells are rows and genes are columns.
`genes`	the list of genes to modify
`clusters`	a list of the clusters to which the cells belong
`target`	the name of the cluster to modify
`classifier`	a classifier in the suitable format
`adv_method`	the name of the method to use
`adv_fixed_value`	the numeric value to use in case of adv_method=`fixed`
`adv_fct`	the function to use in case adv_method belongs to the following list: `full_row_fct`, `target_row_fct`, `target_matrix_fct`, `full_matrix_fct`
`verbose`	logical, set to TRUE to activate verbose mode

Value

a vector of the classification, and the associated odd

Examples

```
MyClassifier <- function(expr, clusters, target) {
   c("T cell", 0.9)
}
rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
     CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

predictWithNewValue(rna_expression, genes, clusters_id,
"T cell", MyClassifier, adv_method="perc99")
```

RFClassifier *Example cell type classifier for the pbmc3k dataset*

Description

Example cell type classifier for the pbmc3k dataset

Usage

```
RFClassifier(expr, clusters, target)
```

Arguments

expr	RNA expression matrix
clusters	list of clusters to which each cell belongs
target	name of the cell cluster to classify

Value

a vector with the classification, and the odd

Examples

```
library(TENxPBMCData)

pbmc <- TENxPBMCData(dataset = "pbmc3k")
mat_rna <- matrixFromSCE(pbmc)
cell_types <- system.file("extdata",
    "pbmc3k_cell_types.tsv",
    package = "adverSCarial"
)
cell_types <- read.table(cell_types, sep = "\t")$cell_type
```

```
RFClassifier(mat_rna, cell_types, "DC")
```

`sceConvertToHGNC` *Returns a SingleCellExperiment object keeping unique HGNC gene*

Description

Returns a SingleCellExperiment object keeping unique HGNC gene

Usage

```
sceConvertToHGNC(sce)
```

Arguments

`sce`	SingleCellExperiment object to convert

Value

the SingleCellExperiment object keeping unique HGNC gene

Examples

```
library(TENxPBMCData)

pbmc <- TENxPBMCData(dataset = "pbmc3k")
hgnc_pbmc <- sceConvertToHGNC(pbmc)
```

Index