Package 'Allspice'

January 20, 2023

Type Package

Title RNA-Seq Profile Classifier

Version 1.0.7

Date 2023-01-20

Author Ville-Petteri Makinen [aut, cre]

Maintainer Ville-Petteri Makinen <vpmakine@gmail.com>

Description We developed a lightweight machine learning tool for RNA profiling of acute lymphoblastic leukemia (ALL), however, it can be used for any problem where multiple classes need to be identified from multi-dimensional data. The methodology is described in Makinen V-P, Rehn J, Breen J, Yeung D, White DL (2022) Multi-cohort transcriptomic subtyping of B-cell acute lymphoblastic leukemia, International Journal of Molecular Sciences 23:4574, <doi:10.3390/ijms23094574>. The classifier contains optimized mean profiles of the classes (centroids) as observed in the training data, and new samples are matched to these centroids using the shortest Euclidean distance. Centroids derived from a dataset of 1,598 ALL patients are included, but users can train the models with their own data as well. The output includes both numerical and visual presentations of the classification results. Samples with mixed features from multiple classes or atypical values are also identified.

License GPL (>= 2)

Imports methods

VignetteBuilder knitr, rmarkdown

Suggests knitr, rmarkdown

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2023-01-20 03:40:02 UTC

R topics documented:

`assemble<-,Asset-method`
Finish Asset contents

Description

Trains a new classification model.

Usage

```
assemble(obj) <- value
```

Arguments

`obj`	An object of the class Asset.
`value`	A list that containts training data, see details.

Details

The value argument must contain three named elements: `title`, `dat` and `bits`. Optional `predictors` and `covariates` elements can also be included.

The title is a descriptive identifier for the asset that will be displayed by the Classifier object in `report()`.

The `dat` element is a matrix that contains the training samples. The variables are organized into named rows and the samples into named columns. Non-finite values are not allowed.

The `predictors` element contains the names of the input variables that should be used for training the model. If empty, all inputs are used for automatic feature selection and subsequent training steps.

The `covariates` element contains additional information for constructing the final classification models. Unlike the data matrix, variables are organized into named columns and the samples are stored as rows. Non-finite values are not allowed.

The `bits` element contains labels for category memberships. Three formats are supported: 1) a character vector of named elements that contains non-empty strings, 2) a matrix or a data frame with row names and a single column of non-empty values, and 3) a matrix or a data frame with multiple columns that contain binary values where 1s indicate category membership (the name of the column is the name of the category). Overlapping categories are allowed.

The final asset is assembled in six steps. First, the training data are standardized and normalized. Second, input variables are sorted according to their univariate classification performance. Third, redundant features are excluded by testing the sorted variables for mutual correlations; this produces an optimized listing of non-redundant features that are the most predictive of the category labels. Fourth, mean centroids are calculated for each category. Fifth, training samples are matched to their nearest centroids and the distances collected as preliminary predictor scores. Lastly, logistic regression models are fitted to the preliminary scores, covariates and category labels to enable the calculation of standardized predictor scores for new data.

Value

Updates the Asset object.

Examples

```
# Prepare training data.
simu <- bcellALL(200)
materials <- list(title="Simutypes")
materials$dat <- simu$counts
materials$covariates <- simu$metadata[,c("MALE","AGE")]
materials$bits <- simu$metadata["SUBTYPE",drop=FALSE]

# Assemble classification asset.
bALL <- asset()
assemble(bALL) <- materials

# Export asset into a new folder.
tpath <- tempfile()
export(bALL, folder = tpath)

# Create a classifier.
cls <- classifier(tpath, verbose = FALSE)

# Classify new samples.
simu <- bcellALL(5)
covariates(cls) <- simu$metadata
profiles(cls) <- simu$counts
primary <- predictions(cls)[[1]]
print(primary[,c("LABEL","PROX","EXCL")])
```

asset *Constructor for Asset*

Description

Creates a new Asset object.

Usage

```
asset(folder = NULL, verbose = TRUE)
```

Arguments

folder	Path to a folder that contains the necessary files for a classification asset.
verbose	If TRUE, accessed asset items (file names) are printed on screen.

Value

Returns an Asset object.

Examples

```
# Set up an ALL subtyping asset.
folder <- system.file("subtypes", package="Allspice")
a <- asset(folder)
```

bcellALL *Simulated data*

Description

Simulated data of B-cell acute lymphoblastic leukemia.

Usage

```
bcellALL(n = 200, contamination = 0.05)
```

Arguments

n	Number of samples.
contamination	Proportion of samples with randomly shuffled values.

Value

Returns a list with two elements: `counts` contains gene RNA read counts and `metadata` contains age and sex information and the generating subtype label.

Examples

```
# Simulate B-cell ALL samples.
simu <- bcellALL(5)
print(head(simu$counts))
print(simu$metadata)
```

`classifier` *Constructor for Classifier*

Description

Creates a new Classifier object.

Usage

```
classifier(..., verbose = TRUE)
```

Arguments

`...`	Any number of paths to folders that contain assets (see `asset()`).
`verbose`	If TRUE, information about the imported assets is printed on screen.

Details

The first input argument will set the primary asset, and the others will be considered secondary assets.

Value

Returns a Classifier object.

Examples

```
# Set up an ALL classifier object.
cls <- classifier()
```

`classify,Asset-method` *Data classification*

Description

Assigns category labels to new data.

Usage

```
classify(obj, dat, covariates)
```

Arguments

`obj`	An object of the class Asset.
`dat`	A matrix that containts variables as rows and samples as columns.
`covariates`	A data.frame or matrix that containts samples as rows and covariates as columns.

Details

The input data will be automatically normalized and standardized using the internal asset parameters, see `configuration()` for details.

Value

Returns a data frame that contains predicted category labels and performance indicators. The column 'CATEG' contains the final predictions, including "Unclassified" or "Ambiguous" for samples that could not be reliably classified. The columns 'MATCH.1st' and 'MATCH.2nd' contain the first and second best matching categories, respectively.

The column 'BIOMRK' contains a standardized biomarker score that indicates how similar the sample is with respect to the best-matching category. The column 'PROX' tells the likelihood of how likely it is that the observed biomarker score would have been generated by a training sample from the best-matching category (balanced group sizes). The column 'EXCL' tells the likelihood that the sample does not share characteristic features with any other category.

The returned data frame also has the attribute "biomarkers" that contains biomarker scores for all categories.

Examples

```
# Import ALL subtyping asset.
base <- system.file(package = "Allspice")
folder <- file.path(base, "subtypes")
a <- asset(folder)

# Simulated data.
simu <- bcellALL(5)

# Predict categories.
```

```
res <- classify(a, dat = simu$counts, covariates = simu$metadata)
print(res[,c("LABEL","PROX","EXCL")])
```

`configuration<-,Asset-method`
Asset configuration

Description

Set internal parameters for an Asset object.

Usage

```
configuration(obj) <- value
configuration(obj)
```

Arguments

`obj`	An object of the class Asset.
`value`	A numeric vector with named elements.

Details

Element names from the input are compared with the internal list of parameters. Those that match will be updated.

Normalization parameters include 'norm' (if set to 0, normalization is not performed), 'nonzero.min' (the minimum data value considered larger than zero) and 'nonzero.ratio' (minimum ratio of non-zero values to include a variable in the output). See `normalize()` for additional details.

Standardization parameters include 'standard' (if set to 0, standardization is not performed) and 'logarithm' (if set to 0, data values are used without taking the logarithm). See `standardize()` for additional details.

Feature selection parameters include 'ninput.max' (the maximum number of features to be used for classification) and 'rrinput.max' (the maximum correlation r-squared to be allowed between features). See `assemble()` for details on selecting non-redundant inputs.

Classification parameters include 'probability.min' (minimum probability rating considered for reliable classification) and 'exclusivity.min' (minimum exclusivity for non-ambiguous classification). See `classify()` for additional details.

Value

Updates the Asset object.

Examples

```
# Change asset configuration.
a <- asset()
print(configuration(a))
configuration(a) <- c(nonzero.min=0, nonzero.ratio=0)
print(configuration(a))
```

`covariates<-,Classifier-method`
Sample covariate data

Description

Add covariate data into the classifier.

Usage

```
covariates(obj) <- value
```

Arguments

`obj`	An object of the class Classifier.
`value`	A numeric vector or a matrix.

Details

If the input is a vector, the elements must be named and these names will be used to identify variables.

If the input is a matrix, it must have named rows and named columns that will be matched with sample identities in `profiles()`.

Value

Updates the Classifier object. Any previous data are discarded.

Examples

```
# Simulated data.
simu <- bcellALL(5)

# Predict subtypes without covariates.
cls <- classifier(verbose = FALSE)
profiles(cls) <- simu$counts
primary <- predictions(cls)[[1]]
print(primary[,c("LABEL","PROX","EXCL")])

# Predict subtypes with covariates.
cls <- classifier(verbose = FALSE)
```

```
covariates(cls) <- simu$metadata
profiles(cls) <- simu$counts
primary <- predictions(cls)[[1]]
print(primary[,c("LABEL","PROX","EXCL")])
```

`export,Asset-method` *Data classification*

Description

Assigns category labels to new data.

Usage

```
export(obj, folder)
```

Arguments

`obj`	An object of the class Asset.
`folder`	Path to a folder that will contain the asset files.

Value

Returns the names of the files that were saved in the folder.

Examples

```
# Import ALL subtyping asset.
base <- system.file(package = "Allspice")
folder <- file.path(base, "subtypes")
a <- asset(folder)

# Export asset into a new folder.
tpath <- tempfile()
fnames <- export(a, folder = tpath)
print(dir(tpath))
```

information,Classifier-method *Classifier contents*

Description

Information about the assets of a classifier

Usage

```
information(obj)
```

Arguments

obj	An object of the class Classifier.

Value

A list with three elements: `covariates` is a data frame that contains the variable names that are included in the classification assets, `configuration` contains the analysis settings for each asset and `categories` contains the labels and visual attributes for the assets.

Examples

```
# Show the contents of the b-cell ALL classifier.
cls <- classifier(verbose=FALSE)
info <- information(cls)
print(info$covariates)
print(info$configuration)
print(head(info$categories))
print(tail(info$categories))
```

nomenclature<-,Asset-method *Variable names*

Description

Conversion table between variable naming schemes.

Usage

```
nomenclature(obj) <- value
```

Arguments

obj	An object of the class Asset.
value	A data frame, see details.

Details

The data frame should contain character columns of variable names with each column representing a spefic naming scheme from which the variables are translated into the row names of the data frame.

For example, to convert gene symbols into Ensemble codes, a data frame with the gene symbols as row names and one column of ensemble codes is needed.

Value

Updates the Asset object.

Examples

```
# Import nomenclature from a system file.
base <- system.file(package = "Allspice")
fname <- file.path(base, "subtypes", "nomenclature.txt")
info <- read.delim(fname, stringsAsFactors = FALSE)

# Set ENSEMBLE identities as row names.
rownames(info) <- info$ENSEMBL
info$ENSEMBL <- NULL
print(head(info))

# Create a new asset and set nomenclature.
a <- asset()
nomenclature(a) <- info

# Prepare training data.
simu <- bcellALL(200)
materials <- list(title="Simutypes")
materials$dat <- simu$counts
materials$covariates <- simu$metadata[,c("MALE","AGE")]
materials$bits <- simu$metadata[,"SUBTYPE",drop=FALSE]

# Assemble classification asset.
assemble(a) <- materials

# Check that nomenclature was set.
simu <- bcellALL(5)
expres <- normalize(a, dat = simu$counts)
print(head(simu$counts))
print(head(expres))
```

`normalize,Asset-method` *Sample normalization*

Description

Adjust scale differences between samples.

Usage

```
normalize(obj, dat)
```

Arguments

`obj`	An object of the class Asset.
`dat`	A matrix that containts variables as rows and samples as columns.

Details

The normalization pipeline comprises three steps. First, variable names are checked against the internal nomenclature and converted to the internal naming scheme where necessary (see `nomenclature()`). Second, variables that are present in the internal normalization reference are imputed with reference values if not available in the data. Third, the data are normalized according to the DESeq2 algortihm (Love MI, Huber W & Anders S, Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2, Genome Biol 15, 550, 2014).

Value

Returns a matrix in the same format as the input.

Examples

```
# Import ALL subtyping asset.
base <- system.file(package = "Allspice")
folder <- file.path(base, "subtypes")
a <- asset(folder)

# Simulated data.
simu <- bcellALL(5)

# Normalize RNA read counts.
expres <- normalize(a, dat = simu$counts)
print(head(simu$counts))
print(head(expres))
```

predictions,Classifier-method
Category predictions

Description

Classifies samples based on their profiles.

Usage

```
predictions(obj)
```

Arguments

obj	An object of the class Classifier.

Details

Use the functions covariates() and profiles() to import data into the classifier.

Value

Returns a list of data frames that contain the output from each classification asset within the classifier. See classify() for details on the result items.

Examples

```
# Simulated data.
simu <- bcellALL(5)

# Predict subtypes.
cls <- classifier(verbose = FALSE)
covariates(cls) <- simu$metadata
profiles(cls) <- simu$counts
pred <- predictions(cls)
print(pred[[1]][,c("LABEL","PROX","EXCL")])
print(pred[[2]][,c("LABEL","PROX","EXCL")])
print(pred[[3]][,c("LABEL","PROX","EXCL")])
```

profiles<-,Classifier-method
Sample data profiles

Description

Analyse new data using classification assets.

Usage

```
profiles(obj) <- value
```

Arguments

obj	An object of the class Classifier.
value	A numeric vector or a matrix where samples are organized into columns and variables into rows.

Details

If the input is a vector, the elements must be named and these names will be used to identify variables.

If the input is a matrix, it must have sample identities as column names and variables identified by row names.

Value

Updates the Classifier object. Any previous data are discarded.

Examples

```
# Simulated data.
simu <- bcellALL(5)

# Predict subtypes.
cls <- classifier(verbose = FALSE)
covariates(cls) <- simu$metadata
profiles(cls) <- simu$counts
primary <- predictions(cls)[[1]]
print(primary[,c("LABEL","PROX","EXCL")])
```

report,Classifier-method
Visualization of results

Description

Creates a visual report of the classification results.

Usage

```
report(obj, name, file = NULL)
```

Arguments

obj	An object of the class Classifier.
name	Name of the sample to be shown.
file	Name of the output file.

Details

The function generates a Scalable Vector Graphics figure that shows the results from each classification asset within the Classifier. The report will highlight the predicted category label and quality metrics for the primary asset and bar charts for the fits to categories in all assests.

If no file name is provided, the report is plotted on the current device, however, note that best visual outcomes are achieved by plotting in a file, especially with classifiers with three or more assets.

Value

Returns the name of the output file.

Examples

```
# Simulated data.
simu <- bcellALL(5)
keys <- colnames(simu$counts)

# Predict subtypes.
cls <- classifier(verbose = FALSE)
covariates(cls) <- simu$metadata
profiles(cls) <- simu$counts

# Show visual report by name.
dev.new()
report(cls, name = keys[3])

# Show visual report by sample index.
dev.new()
report(cls, name = 3)
```

scores,Classifier-method *Category scores*

Description

Classification scores for samples based on their profiles.

Usage

```
scores(obj)
```

Arguments

obj	An object of the class Classifier.

Details

Use the functions covariates() and profiles() to import data into the classifier.

Value

Returns a list of data frames that contain the output from each classification asset within the classifier.

Examples

```
# Simulated data.
simu <- bcellALL(5)

# Predict subtypes.
cls <- classifier(verbose = FALSE)
covariates(cls) <- simu$metadata
profiles(cls) <- simu$counts
z <- scores(cls)
print(z[[1]][,1:5])
print(z[[2]][,1:5])
print(z[[3]][,1:5])
```

standardize,Asset-method
Data standardization

Description

Standardize scale and location of variables.

Usage

```
standardize(obj, dat, trim = FALSE)
```

Arguments

obj	An object of the class Asset.
dat	A matrix that containts variables as rows and samples as columns.
trim	If true, returns only variables used as input features for classification.

Details

If the asset is so configured, the data are first transformed by log(x + 1). Values are processed with the mean and standard deviation that were calculated from the training data when the asset was assembled. The mean is subtracted and the values divided by SD. To control for outliers, extreme values are compressed by the t-distribution with 50 degrees of freedom.

Value

Returns a matrix in the same format as the input.

Examples

```
# Import ALL subtyping asset.
base <- system.file(package = "Allspice")
folder <- file.path(base, "subtypes")
a <- asset(folder)

# Simulated data.
simu <- bcellALL(5)

# Standardize RNA read counts.
expres <- normalize(a, dat = simu$counts)
zscores <- standardize(a, dat = expres)
print(head(simu$counts))
print(head(expres))
print(head(zscores))
```

visuals<-,Asset-method *Visual attributes*

Description

Attach text and color attributes to categories.

Usage

```
visuals(obj) <- value
```

Arguments

obj	An object of the class Asset.
value	A character vector or a data frame, see details.

Details

If the input value is a character vector, the elements are stored as the category text and names of the elements are stored as category names.

If the input value is a data frame, the column 'LABEL' is used as the text and the row names as the names of the categories. Additional columns may include 'COLOR', 'COLOR.dark' and 'COLOR.light' that must contain strings of color names or hexadecimal codes as produced by `rgb()`. In absence of color data, the function assigns automatic colors.

Value

Updates the Asset object.

Examples

```
# Create a new asset and set nomenclature.
a <- asset()

# Set category labels with automatic colors.
labels <- paste("Category", 1:8)
names(labels) <- paste0("cat", 1:8)
visuals(a) <- labels
print(a@categories)

# Add color information.
info <- data.frame(stringsAsFactors = FALSE,
    LABEL = labels, COLOR = "red")
rownames(info) <- names(labels)
visuals(a) <- info
print(a@categories)
```

Index