Package ‘MLSeq’

April 23, 2016

Type Package

Title Machine learning interface for RNA-Seq data

Version 1.8.1

Date 2016-02-29

Author Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Maintainer Gokmen Zararsiz <gokmenzararsiz@hotmail.com>

Depends R (>= 3.0.0), caret, DESeq2, Biobase, limma, randomForest, edgeR

VignetteBuilder knitr

Suggests knitr, e1071, kernlab, earth, ellipse, fastICA, gam, ipred, klaR, MASS, mda, mgcv, mlbench, nnet, party, pls, pROC, proxy, RANN, spls, affy

Imports methods

Collate class.R generics.R methods.R classify.R predictClassify.R

biocViews Sequencing, RNASeq, Classification, Clustering

Description This package applies several machine learning methods, including SVM, bagSVM, Random Forest and CART, to RNA-Seq data.

License GPL(>=2)

NeedsCompilation no

R topics documented:

`MLSeq-package` *Machine Learning Interface for RNA-Seq data*

Description

This package applies several machine learning methods, including SVM, bagSVM, Random Forest and CART, to RNA-Seq data.

Details

Package:	MLSeq
Type:	Package
License:	GPL (>= 2)

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Maintainer: Gokmen Zararsiz <`gokmenzararsiz@erciyes.edu.tr`>

`cervical` *Cervical Cancer Data*

Description

Cervical cancer data measures the expressions of 714 miRNAs of human samples. There are 29 tumor and 29 non-tumor cervical samples and these two groups are treated as two separete classes.

Usage

```
data(cervical)
```

Format

A data frame with 58 observations on the following 715 variables.

Source

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2880020/#supplementary-material-sec

References

Witten, D., et al. (2010) Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls. BMC Biology, 8:58

Examples

```
data(cervical)
```

`classify` *Fitting Classification Models to Sequencing Data*

Description

This function fits classification algorithms to sequencing data and measures model performances using various statistics

Usage

```
classify(data, method = c(”svm”, ”bagsvm”, ”randomforest”, ”cart”), normalize = c(”deseq”, ”none”, ”t
deseqTransform = c(”vst”, ”voomCPM”), cv = 5, rpt = 10, B = 100, ref=NULL, ...)
```

Arguments

`data` DESeqDataSet instance

`method` There are four methods available to perform classification: `svm`: support vector machines using radial-based kernel function, `bagsvm`: support vector machines with bagging ensemble, `randomForest`: random forest algorithm, `cart`: classification and regression trees algorithm.

`normalize` Normalization of count data for classification. `none`: Normalization is not applied. Count data is used for classification. `deseq`: deseq normalization. `tmm`: Trimmed mean of `M` values.

`deseqTransform` Transformation method applied after normalization.`vst`: variance stabilizing transformation. `voomCPM`: voom transformation (log of counts-per-million).

`cv` Number of cross-validation folds.

`rpt` Number of complete sets of folds for computation.

`B` Number of bootstrap samples for `bagsvm` method.

`ref` User defined reference class. Default is `NULL`.

`...` Optional arguments for `train()` function from `caret` package.

Details

In RNA-Seq studies, normalization is used to adjust between-sample differences for further analysis. In this package, "deseq" and "tmm" normalization methods are available. "deseq" estimates the size factors by dividing each sample by the geometric means of the transcript counts. "tmm" trims the lower and upper side of the data by log fold changes to minimize the log-fold changes between the samples and by absolute intensity. After normalization, it is useful to transform the data for classification. `MLSeq` package has "voomCPM" and "vst" transformation methods. "voomCPM" transformation applies a logarithmic transformation (log-cpm) to normalized count data. Second transformation method is the "vst" transformation and this approach uses an error modeling and the concept of variance stabilizing transformations to estimate the mean-dispersion relationship of data.

For model validation, k-fold cross-validation ("cv" option in `MLSeq` package) is a widely used technique. Using this technique, training data is randomly splitted into k non-overlapping and equally sized subsets. A classification model is trained on (k-1) subsets and tested in the remaining subsets. `MLSeq` package also has the repeat option as "rpt" to obtain more generalizable models. Giving a number of m repeats, cross validation concept is applied m times.

For more details, see the vignette.

Value

`model`	fitted classification model
`method`	used classification method
`normalization`	used normalization method
`deseqTransform`	deseq transformation if `deseq` normalization is used
`confusionMat`	cross-tabulation of observed and predicted classes and corresponding statistics
`ref`	reference class

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

References

Kuhn M. (2008). Building predictive models in R using the caret package. Journal of Statistical Software, (http://www.jstatsoft.org/v28/i05/).

Anders S. Huber W. (2010). Differential expression analysis for sequence count data. Genome Biology, 11:R106

Witten DM. (2011). Classification and clustering of sequencing data using a poisson model. The Annals of Applied Statistics, 5(4), 2493:2518.

Charity WL. et al. (2014) Voom: precision weights unlock linear model analysis tools for RNA-Seq read counts, Genome Biology, 15:R29, doi:10.1186/gb-2014-15-2-r29

Witten D. et al. (2010) Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls. BMC Biology, 8:58

Robinson MD, Oshlack A (2010). A scaling normalization method for differential expression analysis of RNA-Seq data. Genome Biology, 11:R25, doi:10.1186/gb-2010-11-3-r25

See Also

predictClassify

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c("N","T"),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])

# train set in S4 class
data.trainS4 = DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 = DESeq(data.trainS4, fitType="local")

# Classification and Regression Tree (CART) Classification
cart = classify(data = data.trainS4, method = "cart", normalize = "deseq", deseqTransform = "vst", cv = 5, rpt = 3
cart

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = "randomforest", normalize = "deseq", deseqTransform = "vst", cv = 5, r
rf
```

confusionMat-methods *Accessors for the 'confusionMat' slot of an MLSeq object*

Description

Confusion matrix for the trained model using classify function.

Usage

```
  ## S4 method for signature 'MLSeq'
confusionMat(object)
```

Arguments

`object` an `MLSeq` object

Details

`confusionMat` slot stores information about cross-tabulation of observed and predicted classes and corresponding statistics such as accuracy rate, sensitivity, specifity, etc.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c(”N”,”T”),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])

# train set in S4 class
data.trainS4 = DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 = DESeq(data.trainS4, fitType=”local”)

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = ”randomforest”, normalize = ”deseq”, deseqTransform = ”vst”, cv = 5, r

confusionMat(rf)
```

---

deseqTransform-methods

Accessors for the ’deseqTransform’ slot of an MLSeq object

Description

Used transformation method for the trained model using `classify` function.

Usage

```
  ## S4 method for signature 'MLSeq'
deseqTransform(object)
```

Arguments

`object`	an `MLSeq` object

Details

`deseqTransform` slot stores the name of the transformation method either "vst" or "voomCPM"

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c(”N”,”T”),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])

# train set in S4 class
data.trainS4 = DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 = DESeq(data.trainS4, fitType=”local”)

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = ”randomforest”, normalize = ”deseq”, deseqTransform = ”vst”, cv = 5, r

deseqTransform(rf)
```

method-methods *Accessors for the 'method' slot of an MLSeq object*

Description

Used classification method for the trained model using `classify` function.

Usage

```
  ## S4 method for signature 'MLSeq'
method(object)
```

Arguments

`object`	an `MLSeq` object

Details

`method` slot stores the name of the classification method as "svm", support vector machines using radial-based kernel function; "bagsvm", support vector machines with bagging ensemble; "randomForest", random forest algorithm and "cart", classification and regression trees algorithm.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c("N","T"),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])

# train set in S4 class
data.trainS4 = DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 = DESeq(data.trainS4, fitType="local")

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = "randomforest", normalize = "deseq", deseqTransform = "vst", cv = 5, r

method(rf)
```

`MLSeq-class` *MLSeq object*

Description

For classification, this is the main class for the `MLSeq` package.

Objects from the Class

Objects can be created by calls of the form `new("ClassifySeq", ...)`.

This type of objects is created as a result of `classify` function of `MLSeq` package. It is then used in `predictClassify` function for predicting the class labels of new samples.

Slots

`method`**:** stores the name of used classification method in the classification model

`deseqTransform`**:** stores the name of used transformation method in the classification model

`normalization`**:** stores the name of used normalization method in the classification model

`confusionMat`**:** stores the information of classification performance results

`trained`**:** stores the information about training process and model parameters that used in the corresponding model

ref stores user defined reference class

Note

An `MLSeq` class stores the results of `classify` function and offers further slots that are populated during the analysis. The slot `confusionMat` stores the information of classification performance results. These results contain the classification table and several statistical measures including accuracy rate, sensitivity, specifity, positive and negative predictive rates, etc. `method`, `normalization` and `deseqTransform` slots store the name of used classification method, normalization method and transformation method in the classification model respectively. Lastly, the slot `trained` stores the information about training process and model parameters that used in the corresponding model.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Examples

```
# See the vignette
```

`normalization-methods` *Accessors for the 'normalization' slot of an MLSeq object*

Description

Used normalization method for the trained model using `classify` function.

Usage

```
  ## S4 method for signature 'MLSeq'
normalization(object)
```

Arguments

`object`	an `MLSeq` object

Details

`normalization` slot stores the name of the normalization method "deseq", "none" or "tmm"

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c("N","T"),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])
```

```
# train set in S4 class
data.trainS4 = DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 = DESeq(data.trainS4, fitType="local")

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = "randomforest", normalize = "deseq", deseqTransform = "vst", cv = 5, r

normalization(rf)
```

predictClassify *Extract Predictions From* `classify()` *objects*

Description

This function predicts the class labels of test data for a given model.

Usage

```
predictClassify(model, test.data)
```

Arguments

`model`	a model of `MLSeq` class
`test.data`	a `DESeqDataSet` instance of new observations.

Details

`predictClassify` function gives a vector of predicted classes of data set. This vector is in `factor` class.

Value

`predicted`	a vector of predicted classes of test data. See details.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

References

Kuhn M. (2008). Building predictive models in R using the caret package. Journal of Statistical Software, (http://www.jstatsoft.org/v28/i05/).

Anders S. Huber W. (2010). Differential expression analysis for sequence count data. Genome Biology, 11:R106

Witten DM. (2011). Classification and clustering of sequencing data using a poisson model. The Annals of Applied Statistics, 5(4), 2493:2518.

Charity WL. et al. (2014) Voom: precision weights unlock linear model analysis tools for RNA-seq read counts, Genome Biology, 15:R29, doi:10.1186/gb-2014-15-2-r29

Witten D. et al. (2010) Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls. BMC Biology, 8:58

Robinson MD, Oshlack A (2010). A scaling normalization method for differential expression analysis of RNA-Seq data. Genome Biology, 11:R25, doi:10.1186/gb-2010-11-3-r25

See Also

classify

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c("N","T"),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 <- DESeq(data.trainS4, fitType="local")

# test set
data.test = data[,ind]
data.test = as.matrix(data.test + 1)
classts = data.frame(condition=class[ind,])

# test set in S4
```

```
data.testS4 = DESeqDataSetFromMatrix(countData = data.test,
colData = classts, formula(~ condition))
data.testS4 = DESeq(data.testS4, fitType="local")

## Number of repeats (rpt) might change model accuracies ##

# Classification and Regression Tree (CART) Classification
cart = classify(data = data.trainS4, method = "cart", normalize = "deseq", deseqTransform = "vst", cv = 5, rpt = 3
cart

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = "randomforest", normalize = "deseq", deseqTransform = "vst", cv = 5, r
rf

# predicted classes of test samples for SVM method
pred.cart = predictClassify(cart, data.testS4)
pred.cart

# predicted classes of test samples for RF method
pred.rf = predictClassify(rf, data.testS4)
pred.rf
```

ref-methods *Accessors for the 'ref' slot of an MLSeq object*

Description

The reference class category for the trained model using `classify` function.

Usage

```
  ## S4 method for signature 'MLSeq'
ref(object)
```

Arguments

`object`	an `MLSeq` object

Details

Reference class category is important while calculating the statistical measures for the confusion matrix obtained from classification models.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c("N","T"),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])

# train set in S4 class
data.trainS4 = DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 = DESeq(data.trainS4, fitType="local")

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = "randomforest", normalize = "deseq", deseqTransform = "vst", cv = 5, r

ref(rf)
```

trained-methods *Accessors for the 'trained' slot of an MLSeq object*

Description

Details about the training model information which is obtained `classify` function.

Usage

```
  ## S4 method for signature 'MLSeq'
trained(object)
```

Arguments

`object`	an `MLSeq` object

Details

`trained` slot stores information about the training process such as optimum model parameters and resampling properties on the fitted classification model.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Turgay Unver, Ahmet Ozturk

Examples

```
data(cervical)

data = cervical[c(1:150),]  # a subset of cervical data with first 150 features.

class = data.frame(condition=factor(rep(c("N","T"),c(29,29))))# defining sample classes.

n = ncol(data)  # number of samples
p = nrow(data)  # number of features

nTest = ceiling(n*0.2)  # number of samples for test set (20% test, 80% train).
ind = sample(n,nTest,FALSE)

# train set
data.train = data[,-ind]
data.train = as.matrix(data.train + 1)
classtr = data.frame(condition=class[-ind,])

# train set in S4 class
data.trainS4 = DESeqDataSetFromMatrix(countData = data.train,
colData = classtr, formula(~ condition))
data.trainS4 = DESeq(data.trainS4, fitType="local")

# Random Forest (RF) Classification
rf = classify(data = data.trainS4, method = "randomforest", normalize = "deseq", deseqTransform = "vst", cv = 5, r

trained(rf)
```

Index