Package ‘dualKS’

April 14, 2017

Type Package

Title Dual KS Discriminant Analysis and Classification

Version 1.34.0

Date 2008-04-25

Author Eric J. Kort, Yarong Yang

Maintainer Eric J. Kort <ericjkort@gmail.com>,
             Yarong Yang <yarong.yang@ndsu.edu>

Imports graphics

Depends R (>= 2.6.0), Biobase (>= 1.15.0), affy, methods

Description This package implements a Kolmogorov Smirnov rank-sum based
algorithm for training (i.e. discriminant analysis--identification
of genes that discriminate between classes) and classification
of gene expression data sets. One of the chief strengths of
this approach is that it is amenable to the ``multiclass'' problem.
That is, it can discriminate between more than 2 classes.

predictClass.R rKS.R rKSScan.R

License LGPL (>= 2.0)

biocViews Microarray, Classification

NeedsCompilation no

R topics documented:

dualKS-package .................................................. 2
dks .......................................................... 2
DKSClassifier .................................................. 3
dksClassify .................................................. 3
dksCustomClass ............................................. 4
DKSGeneScores ............................................... 5
DKSGeneSet .................................................. 6
dksPerm ....................................................... 6
DKSPredicted ................................................ 8
dksSelectGenes ............................................. 9
dksTrain ..................................................... 10
This package implements a Kolmogorov Smirnov rank-sum based algorithm for training (i.e. discriminant analysis—identification of genes that discriminate between classes) and classification of gene expression data sets. One of the chief strengths of this approach is that it is amenable to the "multiclass" problem. That is, it can discriminate between more than 2 classes.

Details

Package: dualKS
Type: Package
Version: 1.0
Date: 2008-04-18
License: GPL-3

Use of this package requires an ExpressionSet, pre-processed to your liking, containing gene expression data you wish to train on to identify discriminant genes. Optionally, a second ExpressionSet may be employed for subsequent testing or "diagnosis".

If the ExpressionSet does not contain the appropriate phenoData specifying the classes for each sample, a factor containing this information will also be required.

See the examples for further details.

Author(s)

Eric J. Kort Maintainer: <Eric.Kort@vai.org>

Description

This data set contains an ExpressionSet comprised of a subset of data from the Gene Expression Omnibus (GEO) data set GDS2126 (Rheumatoid arthritis: synovial tissues). The data set has been severely pruned down to allow quick execution of the examples associated with this package.

Usage

data("dks")
**DKSClassifier**

**Format**
MIAME formatted data as an ExpressionSet.

**Source**

---

**DKSClassifier**

*Class to contain gene set signatures.*

**Description**
The function dksSelectGenes returns an object of class DKSClassifier that contains two DKSGeneSet objects: one for each signature (upregulated and downregulated), needed for subsequent classification. This way, the gene scoring does not need to be repeated for classifiers based on different numbers of genes (see DKSGeneScores and dksTrain).

**Author(s)**
Eric J. Kort

**See Also**
DKSGeneScores-class, dksTrain.

---

**dksClassify**

*Predict classes for gene expression sets.*

**Description**
Kolmogorov-Smirnov rank sum scoring will be used to assign one or more samples to one of two or more classes based on previously defined gene signatures (see dksTrain).

**Usage**
dksClassify(eset, classifier, rescale=FALSE, method="kort")

**Arguments**
- **eset**
  An ExpressionSet or matrix containing the gene expression data for the samples to be classified.
- **classifier**
  An DKSClassifier produced by dksSelectGenes describing the gene expression signature for each class.
- **rescale**
  If TRUE, scores for each class will be mean centered and normalized to remove arbitrary differences in scale and baseline value between signatures for different classes.
- **method**
  Two methods are supported. The 'kort' method returns the maximum of the running sum. The 'yang' method returns the sum of the maximum and the minimum of the running sum, thereby penalizing classes that are highly enriched in a subset of genes of a given signature, but highly down regulated in another subset of that same signature.
dksCustomClass

Value
An object of class `DKSPredicted` containing the class to which each sample in the eset was assigned as well as other information. This object has its own summary and show functions useful for displaying this information in a user friendly format.

Author(s)
Eric J. Kort, Yarong Yang

See Also
`dksTrain`, `dksSelectGenes`, `dksClassify`, `DKSGeneScores`, `DKSPredicted`, `DKSClassifier`

Examples
```
data("dks")
tr <- dksTrain(eset, 1, "up")
c1 <- dksSelectGenes(tr, 100)
pr <- dksClassify(eset, c1, rescale=FALSE)
summary(pr, pData(eset)[,1])
show(pr)
plot(pr, actual=pData(eset)[,1])
```

---

dksCustomClass

Create a classification object from predefined gene signature.

Description
This utility function will build a `DKSClassifier` object from your own list of gene ids for use by `dksClassify`. This is useful if you want to use the classification functionality of this package, but already have gene signatures you want to use (as opposed to generating them with `dksTrain`.

Usage
```
dksCustomClass(upgenes=NULL, upclass=NULL, downgenes=NULL, downclass=NULL)
```

Arguments
- `upgenes`: A vector of gene ids that are upregulated in the classes of interest. Note that upgenes and downgenes need not both be specified, but at least one must be.
- `upclass`: A vector or factor of classes specifying which signature each gene in `upgenes` belongs to. (Note that `upgenes` can describe an arbitrary number of classes, and the order of `upgenes` is not important so long as the position of ids in `upgenes` and the position of classes in `upclass` correspond to one another).
- `downgenes`: A vector of gene ids that are downregulated in the classes of interest. Note that `upgenes` and `downgenes` need not both be specified, but at least one must be.
- `downclass`: A vector or factor of classes specifying which signature each gene in `downgenes` belongs to. (Note that `downgenes` can describe an arbitrary number of classes, and the order of `downgenes` is not important so long as the position of ids in `downgenes` and the position of classes in `downclass` correspond to one another).
Value

An object of class \texttt{DKSClassifier}.

Author(s)

Eric J. Kort, Yarong Yang

See Also

dksTrain, dksSelectGenes, dksClassify, DKSGeneScores, DKSPredicted, DKSClassifier

Examples

data("dks")
up <- rownames(exprs(eset))[1:300]
cls <- factor(rep(c(1,2,3), 100))
classifier <- dksCustomClass(upgenes=up, upclass=cls)

---

\textbf{Description}

The function \texttt{dksTrain} returns an object of class \texttt{DKSGeneScore} to hold the analysis results for subsequent classifier extraction and classification. This way, the gene scoring does not need to be repeated for classifiers based on different numbers of genes.

\textbf{Creating Objects}

\texttt{new("DKSGeneScore",gscores.up=new("matrix"),gscores.down=new("matrix"))}

\textbf{Slots}

- \texttt{gscores.up}: A matrix of gene scores, one column per class
- \texttt{gscores.down}: A matrix of gene scores, one column per class

Author(s)

Eric J. Kort

See Also

\texttt{DKSPredicted-class, dksTrain}.
DKSGeneSet

Class to contain gene signatures.

Description

This class contains a vector of genes, and a corresponding factor of classes that indicates to which signature each gene belongs. The function `dksSelectGenes` extracts gene signatures of a specified size from an object of class `DKSGeneScores`.

calculates a score for each possible class (as determined by the `DKSGeneSet` passed to it) for each sample in the test set passed to the function. It then determines which class each sample most likely belongs to based on which of those scores is the largest. All this information is useful after the analysis, and so it is retained in the return object of class `DKSPredicted`.

Slots

- `genes`: A vector of gene identifiers. Order is not important.
- `classes`: A factor specifying which class (signature) each gene belongs to.

Author(s)

Eric J. Kort

See Also

`DKSGeneScores-class`, `DKSClassifier-class`, `dksSelectGenes`.

dksPerm

Estimate significance of signature scores.

Description

The distribution of Kolmogorov Rank Sum scores generated by this package depends on a variety of factors including the size of the signature and the total number of genes measured in each sample. For a given classifier, this function bootstraps an approximate distribution for the scores and then identifies optimum parameters for the gamma distribution that best fits the bootstrap distribution. The corresponding gamma probability function is then returned, allowing p-values for one or more scores to be readily computed.

Usage

```r
   dksPerm(eset, class, n=100, samples=100, type="up", rescale=FALSE, verbose=FALSE, method="kort")
```
**Arguments**

- **eset**: An ExpressionSet or matrix containing the gene expression data to be used for bootstrapping.
- **class**: A factor with two or more levels indicating which class each sample in the expression set belongs OR an integer indicating which column of pData(eset) contains this information.
- **n**: The number of genes per class to use in the bootstrap signature.
- **samples**: The number of bootstrap samples to generate. A value of at least 1000 gives good results, but may take a while.
- **type**: One of "up", "down", or "both". See `dksTrain`.
- **rescale**: Logical indicating whether scores should be rescaled to range c(0,1).
- **verbose**: Set to TRUE if you want more evidence of progress while data is being processed. Set to FALSE if you want your CPU cycles to be used on analysis and not printing messages.
- **method**: One of either 'kort' or 'yang'. Should match that used for `dksTrain`.

**Value**

A function (1-pgamma(x, ...)) with the appropriate parameters preset based on log likelihood maximization relative to the bootstrapped distribution.

**Note**

All arguments should match those used by `dksClassify`, otherwise the estimated p-values will not meaningfully describe the distribution of scores generated by that function.

**Author(s)**

Eric J. Kort

**See Also**

`dksTrain`, `dksSelectGenes`, `dksClassify`, `DKSGeneScores`, `DKSPredicted`, `DKSClassifier`

**Examples**

```r
data("dks")
p.value <- dksPerm(eset, 1, samples=25)
# this is not nearly enough samples, but will suffice for
# the demonstration. See the vignette for more informative
# example.
p.value(250)
p.value(1500)
```
DKSPredicted

Class to contain classification results from dksClassify

Description

The function \texttt{dksClassify} calculates a score for each possible class (as determined by the \texttt{DKSGeneSet} passed to it) for each sample in the test set passed to the function. It then determines which class each sample most likely belongs to based on which of those scores is the largest. All this information is useful after the analysis, and so it is retained in the return object of class \texttt{DKSPredicted}.

Creating Objects

\begin{verbatim}
new("DKSPredicted", samples=new("vector"), predictedClass=new("factor"), predictedScore=new("vector"), scoreMatrix=new("matrix"))
\end{verbatim}

Slots

\begin{itemize}
  \item \texttt{samples}: A vector of sample names.
  \item \texttt{predictedClass}: A factor specifying which class each sample belongs to based on the classification.
  \item \texttt{predictedScore}: A vector giving the score corresponding to the assigned class for each sample.
  \item \texttt{scoreMatrix}: The complete score matrix for future reference.
\end{itemize}

Methods

Standard generic methods:

\begin{itemize}
  \item \texttt{plot(DKSPredicted)}: Plot scores for each sample and class sorted, in turn, by each class.
  \item \texttt{plot(DKSPredicted, missing)}: Plot scores for each sample and class sorted, in turn, by each class.
  \item \texttt{summary(DKSPredicted)}: Display summary information for the classification.
  \item \texttt{summary(DKSPredicted, factor)}: Display summary information for the classification including the percent correctly classified (as specified by the factor).
  \item \texttt{show(DKSPredicted)}: Displays a table of scores and predicted classes for each sample.
\end{itemize}

Author(s)

Eric J. Kort

See Also

\texttt{DKSGeneScores-class, DKSClassifier-class, dksClassify}. 
**dksSelectGenes**  
Extract gene signatures from a DKSGeneScores object.

**Description**

The DKSGeneScores returned by `dksTrain` holds the rank data for all the genes in the original `ExpressionSet`. However, generally only the top n genes for each class are desired for classification. Rather than needing to re-run `dksTrain` every time a signature of different size (n) is desired, you simply extract that top n genes from this object using `dksSelectGenes`.

**Usage**

```r
dksSelectGenes(data, n)
```

**Arguments**

- `data` An object of class `DKSGeneScores`, typically generated by `dksTrain`
- `n` The number of genes, per class, to include in the classification signature.

**Value**

An object of class `DKSGeneScores`

**Author(s)**

Eric J. Kort, Yarong Yang

**See Also**

`dksTrain`, `dksSelectGenes`, `dksClassify`, `DKSGeneScores`, `DKSPredicted`, `DKSClassifier`

**Examples**

```r
data("dks")
tr <- dksTrain(eset, 1, "up")
c1 <- dksSelectGenes(tr, 100)
pr <- dksClassify(eset, c1)
summary(pr, pData(eset)[,1])
show(pr)
plot(pr, actual=pData(eset)[,1])
```
Perform Dual KS Discriminant Analysis

Description

This function will perform dual KS discriminant analysis on a training set of gene expression data (in the form of an ExpressionSet) and a vector of classes describing which of (two or more) classes each column of data corresponds to. Genes will be be ranked based on the degree to which they are upregulated or downregulated in each class, or both. Discriminant gene signatures are then extracted using dksSelectGenes and applied to new samples with dksClassify.

Usage

dksTrain(eset, class, type = "up", verbose=FALSE, weights=FALSE, logweights=TRUE, method='kort')

Arguments

esetGene expression data in the form of an ExpressionSet or matrix
classA factor with two or more levels indicating which class each sample in the expression set belongs OR an integer indicating which column of pData(eset) contains this information.
typeOne of "up", "down", or "both" indicating whether you want to analyze and classify based on up or down regulated genes, or both (note that classification of samples based on down regulated genes from single color experiments should be expected to work well due to the noise at low expression levels. Therefore, 'down', or 'both' should only be used for two color experiments or one color data that has been converted to ratios based on some reference sample(s).)
verboseSet to TRUE if you want more evidence of progress while data is being processed. Set to FALSE if you want your CPU cycles to be used on analysis and not printing messages.
weightsValue determines whether and how genes are weighted when building the signatures. See details.
logweightsShould the weights be log10 transformed prior to applying?
methodTwo methods are supported. The 'kort' method returns the maximum of the running sum. The 'yang' method returns the sum of the maximum and the minimum of the running sum, thereby penalizing genes that are highly enriched in a subset of samples of a given class, but highly down regulated in another subset of that same class.

Details

This function calculates the Kolmogorov-Smirnov rank sum statistic for each gene and each level of 'class'. The highest scoring genes can then be extracted for use in classification.

If weights=FALSE, signatures are defined based on the ranks of members of each class when sorted on each gene. Those genes for which a given class has the highest rank when sorting samples by those genes will be included in the classifier, with no regard to the absolute expression level of those genes. This is the classic KS statistic.

Very discriminant genes identified in this way may or may not be the highest expressed genes. The result is that signatures identified in this way have arbitrary "baseline" values. This may lead to
**dksWeights**

Calculate gene weights based on average expression.

**Description**

Prior to selecting genes it may be desirable to calculate weights for each gene so that some genes are more likely than others to be included in the gene signature all other things being equal. This function will calculate an N x M weight matrix for N genes in data and M unique classes in class. The weights are based on mean expression of each gene in each class such that genes that are highly expressed on average in a given class will be weighted more highly when scoring genes for that class.

The resulting weight matrix can be bassed to **dksTrain** as the weights argument.

**Usage**

\[
dksWeights(eset, class)
\]
Arguments

eset  An ExpressionSet or matrix containing the gene expression data to be used for bootstrapping.

class  A factor with two or more levels indicating which class each sample in the expression set belongs OR an integer indicating which column of pData(eset) contains this information.

Value

An N x M matrix containing the weights for each gene and each class.

Note

There are many metrics the user might want to use for weighting. This convenience function just implements one of the most obvious ones. The user can provide his/her own N x M weight matrix to dksTrain. The weight matrix calculated by this function will be calculated on the fly if the weights is set to TRUE when calling dksTrain. However, it multiple calls to dksTrain are being made (for example when performing some type of optimization or validation), it will save a lot of time if the weight matrix is pre-calculated by a call to this function and the resulting matrix supplied directly to dksTrain rather than having it re-calculate the weight matrix every time.

Author(s)

Eric J. Kort

See Also

dksTrain, dksSelectGenes, dksClassify, DKSGeneScores, DKSPredicted, DKSClassifier

Examples

data("dks")
wt <- dksWeights(eset, 1)
str(wt)

---

KS  Calculate Kolmogorov Smirnov rank sum scores.

Description

This function calculates the degree to which a subset of genes (i.e. a "signature") is biased in the ordered list of all genes. The function is typically used internally by dksClassify, but the user may want to call it directly to inspect the running sums.

Usage

KS(data, geneset, decreasing=TRUE, method="kort")
Arguments

- **data**: A vector of gene expression data. The data need not be sorted, as the function will sort it itself.
- **geneset**: A DKSGeneSet object, such as one of the slots of the DKSClassifier returned by link{dksClassify}.
- **decreasing**: Indicates which way data should be sorted. If TRUE, the degree of upregulation will be scored. If FALSE, the degree of down regulation will be scored.
- **method**: Two methods are supported. The ‘kort’ method returns the maximum of the running sum. The ‘yang’ method returns the sum of the maximum and the minimum of the running sum, thereby penalizing classes that are highly enriched in a subset of genes of a given signature, but highly down regulated in another subset of that same signature.

Value

- **runningSums**: A matrix with 1 row per gene and 1 column per signature. The value is the running sum of the KS metric at each point along the sorted list of genes. The maximum of this column vector corresponds to the KS score for the corresponding signature.
- **ksScores**: A named vector giving the KS score for each signature.

Author(s)

Eric J. Kort, Yarong Yang

See Also

dksTrain, dksSelectGenes, dksClassify, DKSGeneScores, DKSPredicted, DKSClassifier

Examples

data("dks")
tr <- dksTrain(eset, 1, "both")
cl <- dksSelectGenes(tr, 100)
sc <- KS(exprs(eset)[,1], cl@genes.up)
plot(sc$runningSums[,1], type='l')
Index

*Topic classes
  DKSClassifier, 3
  DKSGeneScores, 5
  DKSGeneSet, 6
  DKSPredicted, 8

*Topic classified
  dksClassify, 3
  dksCustomClass, 4
  dksPerm, 6
  dksSelectGenes, 9
  dksTrain, 10
  dksWeights, 11
  KS, 12

*Topic datasets
  dks, 2

*Topic package
  dualKS-package, 2

plot,DKSPredicted,missing-method (DKSPredicted), 8
plot,DKSPredicted-method (DKSPredicted), 8
pv.f(dks), 2
pvr.f(dks), 2
show,DKSPredicted-method (DKSPredicted), 8
summary,DKSPredicted-method (DKSPredicted), 8

class:DKSClassifier (DKSClassifier), 3
class:DKSGeneScores (DKSGeneScores), 5
class:DKSGeneSet (DKSGeneSet), 6
class:DKSPredicted (DKSPredicted), 8
dks, 2
DKSClassifier, 3, 4, 5, 7, 9, 11–13
DKSClassifier-class (DKSClassifier), 3
dksClassify, 3, 4, 5, 7–9, 11–13
dksCustomClass, 4
DKSGeneScores, 3–5, 5, 6, 7, 9, 11–13
DKSGeneScores-class (DKSGeneScores), 5
DKSGeneSet, 3, 6, 13
DKSGeneSet-class (DKSGeneSet), 6
dksPerm, 6
DKSPredicted, 4, 5, 7, 8, 9, 11–13
DKSPredicted-class (DKSPredicted), 8
dksSelectGenes, 3–7, 9, 11–13
dksTrain, 3–5, 7, 9, 10, 11–13
dksWeights, 11
dualKS (dualKS-package), 2
dualKS-package, 2

eset (dks), 2
KS, 12