Package 'SigCheck'

November 12, 2026

Type Package

Title Check a gene signature's prognostic performance against random signatures, known signatures, and permuted data/metadata

Version 2.44.0

Author Rory Stark <bioconductor@starkhome.com> and Justin Norden

Maintainer Rory Stark <bioconductor@starkhome.com>

Description While gene signatures are frequently used to predict phenotypes (e.g. predict prognosis of cancer patients), it it not always clear how optimal or meaningful they are (cf David Venet, Jacques E. Dumont, and Vincent Detours' paper ``Most Random Gene Expression Signatures Are Significantly Associated with Breast Cancer Outcome''). Based on suggestions in that paper, SigCheck accepts a data set (as an ExpressionSet) and a gene signature, and compares its performance on survival and/or classification tasks against
a) random gene signatures of the same length;
b) known, related and unrelated gene signatures;
and c) permuted data and/or metadata.

License Artistic-2.0

LazyLoad yes

Depends R (>= 4.0.0), MLInterfaces, Biobase, e1071, BiocParallel, survival

Imports graphics, stats, utils, methods

Suggests BiocStyle, breastCancerNKI, qusage

biocViews GeneExpression, Classification, GeneSetEnrichment

git_url https://git.bioconductor.org/packages/SigCheck

git_branch RELEASE_3_23

git_last_commit 3d8182f

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

SigCheck-package . . . 2
classifyResults . . . 3
knownSignatures . . . 4
nkiResults . . . 5
sigCheck . . . 6
sigCheckAll . . . 9
sigCheckKnown . . . 11
SigCheckObject-class . . . 13
sigCheckPermuted . . . 15
sigCheckPlot . . . 17
sigCheckRandom . . . 19

Index **22**

`SigCheck-package` *Check a gene signature's survival and/or classification performance against random signatures, known signatures, and permuted data/metadata.*

Description

While gene signatures are frequently used to predict phenotypes (e.g. predict prognosis of cancer patients), it it not always clear how optimal or meaningful they are (cf David Venet, Jacques E. Dumont, and Vincent Detours' paper "Most Random Gene Expression Signatures Are Significantly Associated with Breast Cancer Outcome"). Based on suggestions in that paper, SigCheck accepts a data set (as an ExpressionSet) and a gene signature, and compares its performance on survival and/or classification tasks against a) random gene signatures of the same length; b) known, related and unrelated gene signatures; and c) permuted data and/or metadata.

Details

Package:	SigCheck
Type:	Package
Version:	2.0
Date:	2015-05-18
License:	Artistic-2.0

To use `SigCheck`, first create anew SigCheck object using the function `sigCheck`. This will establish the baseline performance of the signature. Next, either run specific checks, or use the high level function `sigCheckAll` to run all the core functions in turn. The three core functions enable 1) comparison of baseline performance against signatures composed of random genes (`sigCheckRandom`); 2) comparison of baseline performance against known, and mostly unrelated, gene signatures (`sigCheckKnown`); and 3) comparison of baseline performance against randomly permuted data and/or metadata (`sigCheckPermuted`).

At a minimum, SigCheck requires a data set (as an ExpressionSet), metadata indicating the membership of each sample in one of two classes, and a signature (a subset of features in the ExpressionSet). If survival data are available, survival analyses are carried out. Validation samples can be divided into two classes using one of the simple default methods (based on overall expression value or their first principal component). Alternatively, more sophisticated classification algorithms can be deployed, using the MLearn function from the MLInterfaces package to build a classifier (using link{smvI} by default). If no validation samples are specified, leave-one-out (LOO) cross-validation is utilized to build multiple classifiers, each predicting one sample. If no survival data are provided, signatures are evaluated based on classification performance.

Output of each check includes the distribution of random performance scores (either survival p-value or classification performance) and the ranking of the passed signature within this distribution. An empirical p-value calculation based on this rank is also returned indicating confidence that the performance of the signature being checked has unique power.

Author(s)

First version written by Justin Norden with Rory Stark at the University of Cambridge, Cancer Resaerch UK Cambridge Institute.

Second version, including all survival analysis, written by Rory Stark at CRUK-CI.

Maintainer: Rory Stark <rory.stark@cruk.cam.ac.uk>

References

Venet, David, Jacques E. Dumont, and Vincent Detours. "Most random gene expression signatures are significantly associated with breast cancer outcome." PLoS Computational Biology 7.10 (2011): e1002240.

`classifyResults` *Precomputed list of results for a calls to* `sigCheckRandom` *and* `sigCheckKnown` *using the* `breastCancerNKI` *dataset.*

Description

This object represents the results lists returned by calls to `sigCheckRandom` and `sigCheckKnown`. It is used by the vignette accompanying the `SigCheck` package as an example result. It was derived by running the code in the example below. It loads to object, `slassifyRandom` and `classifyKnown`.

Usage

```
data(classifyResults)
```

Examples

```
## Not run:
# This is how classifyResults is built
library(breastCancerNKI)
data(nki)
nki = nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)
check <- sigCheck(nki, classes="e.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
                  validationSamples=101:ncol(nki),
                  scoreMethod="classifier")

classifyRandom <- sigCheckRandom(check, iterations=1000)
classifyKnown <- sigCheckKnown(check)

## End(Not run)

# Example usage of classifyResults
data(classifyResults)
par(mfrow=c(1,2))
sigCheckPlot(classifyRandom, classifier=TRUE)
sigCheckPlot(classifyKnown, classifier=TRUE)
```

---

knownSignatures *Previously identified gene signatures for use in* sigCheckKnown

---

Description

Previously identified gene signature sets. These include three signatures sets from Venet et. al.

Usage

```
data(knownSignatures)
```

Format

The data object knownSignatures is a list of sets of gene signatures. Each set is a list of gene signatures. Each signature is a vector of gene names.

Gene signature sets include:

- "cancer": 48 gene signatures derived from cancer samples, from Venet et. al.
- "proliferation": 5 gene signatures comprising genes associated with cell proliferation, including a "super signature", from Venet et. al.
- "non.cancer": 3 gene signatures derived from non-cancer sources, from Venet et. al.

Details

These data are taken directly from the supplemental material for Venet et. al "Most random gene expression signatures are significantly associated with breast cancer outcome".

Note

Other signatures of interest can be downloaded at http://www.broad.mit.edu/gsea/downloads.jsp#msigdb and read in using the `read.gmt` function in the `qusage` package.

Source

http://www.ploscompbiol.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pcbi.1002240.s001

References

Venet, David, Jacques E. Dumont, and Vincent Detours. "Most random gene expression signatures are significantly associated with breast cancer outcome." PLoS Computational Biology 7.10 (2011): e1002240.

Examples

```
data(knownSignatures)
names(knownSignatures)
names(knownSignatures$cancer)
knownSignatures$cancer$VANTVEER
```

---

nkiResults *Precomputed list of results for a call to* `sigCheckAll` *using the* `breastCancerNKI` *dataset.*

Description

This object represents the results lists returned by a call to `sigCheckAll`. It is used by the vignette accompanying the `SigCheck` package as an example result. It was derived by running the code in the example below.

Usage

```
data(nkiResults)
```

Examples

```
## Not run:
# This is how nkiResults is built
library(breastCancerNKI)
data(nki)
nki = nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
```

```
                    signature=knownSignatures$cancer$VANTVEER,
                    annotation="HUGO.gene.symbol",
                    validationSamples=which(nki$series=="NKI2"))

nkiResults <- sigCheckAll(check,iterations=1000)

## End(Not run)

# Example usage of nkiResults
data(nkiResults)
sigCheckPlot(nkiResults)
```

---

sigCheck *Create a* SigCheckObject *and establish baseline performance.*

Description

Main constructor for a SigCheckObject. Also establishes baseline survival analysis and/or classification performance.

Usage

```
sigCheck(expressionSet, classes, survival, signature,
         annotation, validationSamples,
         scoreMethod="PCA1", threshold=median,
         classifierMethod=svmI, modeVal,
         survivalLabel, timeLabel,
         plotTrainingKM=TRUE, plotValidationKM=TRUE,
         impute=TRUE)
```

Arguments

expressionSet	An ExpressionSet object containing the data to be checked, including an expression matrix, feature labels, and samples. expressionSet can also be an existing SigCheckObject, in which case everything will be inherited from the passed object except the values for any specified parameters,
classes	Specifies which label is to be used to determine the prognostic categories (must be one of varLabels(expressionSet)). There should be only two unique values in expressionSet$classes.
survival	Specifies which label is to be used to determine survival times. (must be one of varLabels(expressionSet)). This may be missing if only classification is being checked.
signature	A vector of feature labels specifying which features comprise the signature to be checked. These feature labels should match values as specified in the annotation parameter. Alternatively, this can be a integer vector of feature indexes.

annotation Character string specifying which fvarLabels field should be used as the annotation. If missing, the row names of the expressionSet are used as the feature names.

validationSamples
Optional specification, as a vector of sample indices, of what samples in the expressionSet should be considered validation samples. If present, the main checks will be performed using only these samples. If a the scoreMethod parameter is equal to "classifier", the remaining samples will be used as a training set to construct a classifier that will be used to separate the training samples. If a classifier is used, and validationSamples is not specified, a leave-one-out (LOO) validation method will be used, where a separate classifier will be trained to classify each sample using the all the remaining samples.

scoreMethod specification of how the samples should be split into groups for survival analysis. If a character sting, one of the following values:

- "PCA1": default scoring method for separating validation samples into groups by taking the value of the first principal component of the expression values in the signature for each sample.
- "High": score used for separating validation samples into groups for each sample is the mean value over all the expression values in the signature for each sample.
- "classifer": score used for separating validation samples into groups is determined by a classifier specified in the classifierMethod parameter. If the survival parameter is specified, the classifier method must return a real-valued score for each predicted sample.

scoreMethod can also be a user-defined function that computes a score. The function should take a single parameter, an ExpressionSet, and return a vector of score, one for each row.

if the survival parameter is missing, scoreMethod value must be "classifier".

threshold specifies the threshold used for separating the validation samples into classed based on the score derived using scoreMethod. Can be either a function, (with default median) or a number between zero and one indicating a percentile. Validation samples will be divided into a group whose percentile scores are less than this value, and another group with percentile scores greater than or equal to this value. threshold may also be a vector of two percentiles, in which case samples will be divided into High, Low, and Mid groups. The survival p-value will be computed using only the high and low groups, with the mid group samples excluded.

classifierMethod
if the scoreMethod is equal to "classifer", this specifies what classifier to use. It is a MLInterfaces learnerSchema object indicating the machine learning method to use for classification. Default is svmI for linear Support Vector Machine classification. See MLearn for available methods.

modeVal specifies which of the two category values (one of the values implied by the classes parameter) should be considered as the default value when computing the performance of a "mode" classifier. Is missing, the actual mode (most commonly occurring) value of the training set will be used.

`survivalLabel` String to use in the Y-axis of any Kaplan-Meier plots generated, this indicates what aspect of survival is being predicted, such as time to recurrence or death.

`timeLabel` String to use in the X-axis of an Kaplan-Meier plots generated, this indicates the units of time, such as days or months to outcome event.

`plotTrainingKM` if the `survival` and `validationSamples` parameters are provided, a Kaplan-Meier plot can be plotted automatically for the training set samples if this is `TRUE`. A value of `FALSE` will suppress the plot being automatically generated.

`plotValidationKM` if the `survival` parameter is provided, a Kaplan-Meier plot can be plotted automatically for the validation set samples if this is `TRUE`. A value of `FALSE` will suppress the plot being automatically generated. Note that is the `validationSamples` parameter is missing, the resulting plot will be over all samples.

`impute` if `TRUE`, missing data values in the `expressionSet` will be imputed. If `FALSE`, any features with any missing values will be removed from the dataset.

Details

This function constructs a new `SigCheckObject` and carried out a baseline analysis, which will vary depending on which parameters are specified.

If the `survival` parameter is specified, a survival analysis is carried out. If the `validationSamples` parameter is specified, this will be done separately on the validation samples and the remaining (training/discovery) samples. The main result is a p-value indicating the confidence that the samples are separable into groups with distinct survival outcomes. This value is obtained using the `survdiff` function in the `survival` package (and applying `pchisq` to the `$chisq` component of the result). The samples are separated into groups using the `scoreMethod` and `threshold` parameters (and possibly the `classifierMethod` parameter).

If the `survival` parameter is not specified, then the `scoreMethod` parameter must be equal to `"classifier"`, and a pure classification analysis is completed (as was done in `SigCheck 1.0`). If the `validationSamples` parameter is specified, the remaining samples are used as a training set to construct a classifier that is used to classify the validation samples. If `validationSamples` is not specified, leave-one-out cross-validation is used whereby a separate classifier is trained to predict each sample using all of the others.

Value

If the baseline analysis can be completed, a `SigCheckObject` is returned.

Author(s)

Rory Stark with Justin Norden

See Also

`sigCheckAll`, `sigCheckRandom`, `sigCheckKnown`, `sigCheckPermuted`.

Examples

```
library(breastCancerNKI)
data(nki)
nki <- nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)

## survival analysis
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol")
check@survivalPval
check <- sigCheck(check, classes="e.dmfs", survival="t.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
                  scoreMethod="High", threshold=.33)
check@survivalPval

## survival analysis with separate training and validation using SVM
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
                  validationSamples=150:319,
                  scoreMethod="classifier")
check
```

---

sigCheckAll *Run a default set of checks on a gene signature.*

Description

High-level function for package SigCheck that runs a default set of checks against a predictive signature.

Usage

```
sigCheckAll(check,
            iterations=10, known="cancer",
            plotResults=TRUE, ...)
```

Arguments

check	A SigCheckObject, as returned by sigCheck.
iterations	Number of iterations to run to generate background distributions. This is how many random signatures the primary signature will be compared to, as well as how many of each type of permuted dataset will be generated for comparison.
known	Specification of a set of known (previously identified) signatures to compare to. See sigCheckKnown for more details.
plotResults	By default, plots of the results will be generated unless this is set or FALSE.
...	Extra parameters to pass through sigCheckPlot.

Details

This high-level function will run four checks, plot the results, and return a consolidated result set.

First, it calls sigCheckRandom to compare the performance of interations randomly selected signatures.

Next, it calls sigCheckKnown to check the performance of the signature against a database of signatures previously identified to discriminate in other domains.

Finally, two calls are made to sigCheckPermuted to check the performance of randomly permuted metadata and expression data. The first call permutes the survival data if they are available (toPermute="survival"); otherwise it permutes the category assignments (toPermute="categories") The second call permuted the expression value for each gene (permuting each row in the ExpressionSet, equivalent to toPermute="features").

If plotResults is TRUE, the results are plotted. If a classifier is involved, a set of four classification results are plotted in a 2x2 grid, showing how the classification performance of the main signature compares to that of a mode classifier and to the distribution of performance values observed for the random and known signature sets, as well as how it performs using the two type of permuted dataset. If survival data is available, another 2x2 grid is plotted showing how the baseline survival p-value compares to a p-value of 0.05 and to the distribution of p-values observed for the random and known signatures, as well as for the permuted data.

Value

A list containing four elements, each containing the result of a check.

- $checkRandom is the result list returned by sigCheckRandom.
- $checkKnown is the result list returned by sigCheckKnown.

  The third element of the result list will be one of the following:
- $checkPermutedSurvival is the result list returned by sigCheckPermuted with toPermute="survival".
- $checkPermutedCategories is the result list returned by sigCheckPermuted with toPermute="categories".

  The fourth element of the list will be:
- $checkPermutedFeatures is the result list returned by sigCheckPermuted with toPermute="features".

Author(s)

Rory Stark

References

Venet, David, Jacques E. Dumont, and Vincent Detours. "Most random gene expression signatures are significantly associated with breast cancer outcome." PLoS Computational Biology 7.10 (2011): e1002240.

See Also

sigCheck, sigCheckRandom, sigCheckPermuted, sigCheckKnown, sigCheckPlot

Examples

```
#Disable parallel so Bioconductor build won't hang
library(BiocParallel)
register(SerialParam())

library(breastCancerNKI)
data(nki)
nki <- nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)
ITERATIONS <- 5 # should be at least 20, 1000 for real checks

## survival analysis
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
                  validationSamples=150:319)

results <- sigCheckAll(check,iterations=ITERATIONS,
                       known=knownSignatures$cancer[1:20])


## classification analysis
check <- sigCheck(nki, classes="e.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
                  validationSamples=275:319,
                  scoreMethod="classifier")

results <- sigCheckAll(check,iterations=ITERATIONS,
                       known=knownSignatures$cancer[1:20])
```

`sigCheckKnown` *Check signature performance against a panel of known signatures.*

Description

Performance of a signature is compared to performance of a panel of known (previously identified) signature.

Usage

```
sigCheckKnown(check, known="cancer")
```

Arguments

check	A SigCheckObject, as returned by sigCheck.
known	Either a character string specifying which set of signatures to use from the included sets in knownSignatures, or a list of previously identified signatures to compare performance against. Each element in the list should be a vector

of feature labels. Default is to use the "cancer" signatures from the included k knownSignatures data set, taken from Venet et. al.

Details

Each specified known signature will be evaluated in the same manner as the primary signature. If survival data were supplied, a survival analysis will be carried out on the validation samples, and a p-value computed as a performance measure. If no survival data are available, the training samples will be used to train a classifier, and the performance score will be percentage of validation samples correctly classified. (If no validation samples are provided, leave-one-out cross validation will be used to calculate the classification performance for each known signature).

An empirical p-value will be computed based on the percentile rank of the performance of the primary signature compared to a null distribution of the performance of the known signatures.

Value

A result list with the following elements:

- `$checkType` is equal to `"Known"`.
- `$knownSigs` is the number of tests run (equal to the number of known signatures indicated where at least one gene matches a feature).
- `$rank` is the performance rank of the primary signature within the performance of the known signatures.
- `$checkPval` is the empirical p-value computed using the scores of the known signature as a null distribution. A value of zero indicates that no known signatures performed as good or better than the primary signature.
- `$survivalPval` represents the performance of the primary signature, if survival data were provided.
- `$survivalPvalsKnown` is a vector of performance scores (p-values) for each known signature on the validation samples, if survival data were provided.
- `$trainingPvalsKnown` is a vector of performance scores (p-values) for each known signature on the training samples, if survival data and separate validation samples were provided.
- `$sigPerformance` is the proportion of validation samples correctly classified by the primary signature if a classifier was used.
- `$modePerformance` is the proportion of validation samples correctly classified using a mode classifier.
- `$performanceKnown` is a vector of classification performance scores for each known signature, each indicating the proportion of validation samples correctly classified is a classifier was used.

Author(s)

Rory Stark

References

Venet, David, Jacques E. Dumont, and Vincent Detours. "Most random gene expression signatures are significantly associated with breast cancer outcome." PLoS Computational Biology 7.10 (2011): e1002240.

See Also

knownSignatures, sigCheck, sigCheckAll, sigCheckRandom, sigCheckPermuted, sigCheckPlot

Examples

```
#Disable parallel so Bioconductor build won't hang
library(BiocParallel)
register(SerialParam())

library(breastCancerNKI)
data(nki)
nki <- nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)

## survival analysis
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
                  validationSamples=150:319)

knownResult <- sigCheckKnown(check)
knownResult$checkPval
knownResult$survivalPvalsKnown[knownResult$survivalPvalsKnown <
                               knownResult$checkPval]
sigCheckPlot(knownResult)
```

SigCheckObject-class *Class* "SigCheckObject"

Description

The main object containing everything associated with an expression dataset and a gene signatures. Used for subsequent checks of the unique prognostic and/or classification performance of the signature. Based on an ExpressionSet object.

Objects from the Class

The preferred way to create a SigCheckObject is to use the sigCheck function.

Slots

checkType: Object of class "character" ~~

classes: Object of class "character" ~~

annotation: Object of class "character" ~~

survival: Object of class "character" ~~

signature: Object of class "vector" ~~

signatureLabels: Object of class "vector" ~~

validationSamples: Object of class "vector" ~~

survivalMethod: Object of class "character" ~~

threshold: Object of class "ANY" ~~

survivalScores: Object of class "numeric" ~~

survivalConfusionMatrix: Object of class "matrix" ~~

survivalClassificationScore: Object of class "numeric" ~~

survivalPval: Object of class "numeric", representing performance of the signature in dividing the samples into sets with distinct survival prognosis.

survivalTrainingScores: Object of class "numeric" ~~

survivalTrainingConfusionMatrix: Object of class "matrix" ~~

survivalTrainingClassificationScore: Object of class "numeric" ~~

survivalTrainingPval: Object of class "numeric" ~~

survivalLabel: Object of class "character" ~~

timeLabel: Object of class "character" ~~

classifierMethod: Object of class "learnerSchema" ~~

sigPerformance: Object of class "numeric" ~~

confusion: Object of class "matrix" ~~

modeVal: Object of class "character" ~~

modePerformance: Object of class "numeric" ~~

classifier: Object of class "classifierOutput" ~~

experimentData: Object of class "MIAME" ~~

assayData: Object of class "AssayData" ~~

phenoData: Object of class "AnnotatedDataFrame" ~~

featureData: Object of class "AnnotatedDataFrame" ~~

protocolData: Object of class "AnnotatedDataFrame" ~~

.__classVersion__: Object of class "Versions" ~~

Extends

Class "ExpressionSet", directly. Class "eSet", by class "ExpressionSet", distance 2. Class "VersionedBiobase", by class "ExpressionSet", distance 3. Class "Versioned", by class "ExpressionSet", distance 4.

Methods

No public methods, access slots directly if required.

Note

More methods and documentation coming soon...

Author(s)

Rory Stark

See Also

sigCheck

Examples

```
showClass("SigCheckObject")
```

---

sigCheckPermuted *Check signature performance against performance on randomly permuted data.*

Description

Performance of a signature on intact data is compared to performance on permuted data and/or metadata. Data may be permuted by by feature (expression values of each feature permuted across samples), samples (expression values of all features permuted within each sample). Metadata may be permuted by categories (permuted assignment of samples to classification categories) or survival (permuted assignment of survival times to samples).

Usage

```
sigCheckPermuted(check, toPermute="categories", iterations=10)
```

Arguments

check	A SigCheckObject, as returned by sigCheck.
toPermute	Character string or vector of strings indicating what should be permuted. Allowable values: • "features": the expression values for each feature will be permuted (permutation by row). • "samples": the expression values for each sample will be permuted (permutation by column). • "survival": the values in survival will be permuted. • "categories": the values in classes will be permuted.
iterations	The number of permuted dataset the primary signature will be compared to. This should be at least 1,000 to compute a meaningful empirical p-value for comparative performance.

Details

The primary signature will be evaluated against each permuted dataset in the same manner as for the intact dataset.

If survival data were supplied, a survival analysis will be carried out on the validation samples, and a p-value computed as a performance measure. If no survival data are available, the training samples will be used to train a classifier, and the performance score will be percentage of validation samples correctly classified. (If no validation samples are provided, leave-one-out cross validation will be used to calculate the classification performance for each permuted dataset).

An empirical p-value will be computed based on the percentile rank of the performance of the signature on the intact dataset compared to a null distribution of the performance of the signature on all the permuted datasets.

Value

A result list with the following elements:

- `$checkType` is equal to `"Permuted"`.
- `$permute` is equal to the passed value of `toPermute`.
- `$tests` is the number of tests run (equal to `iterations`.)
- `$rank` is the performance rank of the signature on the intact dataset compared to its performance in the permuted datasets.
- `$checkPval` is the empirical p-value computed using the performance scores of the signature on permuted datasets as a null distribution. A value of zero indicates that the signature did not perform better on any permuted datasets than it does using the intact data.
- `$survivalPval` represents the performance of the primary signature on the original dataset if survival data were provided.
- `$survivalPvalsPermuted` is a vector of performance scores (p-values) for each permuted dataset, if survival data were provided.
- `$trainingPvalsPermuted` is a vector of performance scores (p-values) for each permuted dataset, if survival data and separate validation samples were provided.
- `$sigPerformance` is the proportion of validation samples correctly classified using the intact dataset if a classifier was used.
- `$modePerformance` is the proportion of validation samples correctly classified in the intact dataset using a mode classifier.
- `$performancePermuted` is a vector of classification performance scores for each permuted dataset, each indicating the proportion of validation samples correctly classified if a classifier was used.

Author(s)

Rory Stark

See Also

`sigCheck`, `sigCheckAll`, `sigCheckRandom`, `sigCheckKnown`

Examples

```
#Disable parallel so Bioconductor build won't hang
library(BiocParallel)
register(SerialParam())

library(breastCancerNKI)
data(nki)
nki <- nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)
ITERATIONS <- 5 # should be at least 20, 1000 for real checks

## survival analysis
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
                  validationSamples=150:319)

par(mfrow=c(1,2))
permutedCategories <- sigCheckPermuted(check, toPermute="categories",
                                       iterations=ITERATIONS)
permutedCategories$checkPval
sigCheckPlot(permutedCategories)
permutedSurvival <- sigCheckPermuted(check, toPermute="survival",
                                     iterations=ITERATIONS)
permutedSurvival$checkPval
sigCheckPlot(permutedSurvival)
```

---

sigCheckPlot *Plot results of a signature check or set of checks*

Description

Plots results of a signature check, as returned by sigCheckRandom, sigCheckKnown, sigCheckPermuted, or sigCheckAll.

Usage

```
sigCheckPlot(checkResults, classifier=FALSE,
             title, nolegend=FALSE, ...)
```

Arguments

checkResults	The list returned by sigCheckRandom, sigCheckKnown, sigCheckPermuted, or sigCheck.
classifier	If a classifier was used in the original call to sigCheck, setting this to TRUE will result in a plot showing how the primary signature compares based on classification performance (rather than survival).
title	Title string for plot. If missing, a default plot title will be generated.

nolegend	IF TRUE, no legend will be included with the plot(s).
...	Additional arguments to be passed to the plot function.

Details

For results based on survival analysis, the background distribution of p-values (in 1-log10() format) derived from the check (either random signatures, known signatures, or performance using permuted data) is plotted. Up to two vertical red lines are also plotted: a solid red line representing the performance of the primary signature/data, and a dotted red line representing a p-value of 0.05. One or both of these may be missing if their performance falls outside the range of the background distributions.

For results based on classification performance, the x-axis represents the range of classification performance scores computed in the check, and the y-axis representing how many times that score was obtained. In addition, vertical lines are plotted representing the classification performance of the originally specified signature (solid red line) and the performance of a classifier that always predicts the mode value of the training samples (dotted red line).

If the results of sigCheckAll is passed in, all four results plots are generated in a 2x2 grid.

Value

none

Note

Better formance of the signature being checked results in the solid red line being to the right of the background distribution. For survival results, this indicates a lower-value. For classification results,this indicates superior classification performance.

Author(s)

Rory Stark with Justin Norden

See Also

sigCheck, sigCheckAll, sigCheckRandom, sigCheckKnown, sigCheckPermuted

Examples

```
#Disable parallel so Bioconductor build won't hang
library(BiocParallel)
register(SerialParam())

library(breastCancerNKI)
data(nki)
nki <- nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)
ITERATIONS <- 5 # should be at least 1000 for real checks

## survival analysis with separate training and validation using SVM
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
```

```
                    signature=knownSignatures$cancer$VANTVEER,
                    annotation="HUGO.gene.symbol",
                    validationSamples=250:319,
                    scoreMethod="classifier", threshold=.33)

results <- sigCheckRandom(check,iterations=ITERATIONS)
par(mfrow=c(1,2))
sigCheckPlot(results)
sigCheckPlot(results, classifier=TRUE)
```

`sigCheckRandom` *Check signature performance against signatures composed of randomly selected features*

Description

Performance of a signature is compared to performance of signatures composed of the same number of randomly-selected features.

Usage

```
sigCheckRandom(check, iterations=10)
```

Arguments

`check`	A `SigCheckObject`, as returned by `sigCheck`.
`iterations`	The number of random signatures the primary signature will be compared to. This should be at least 1,000 to compute a meaningful empirical p-value for comparative performance.

Details

sigCheckRandom will select `iterations` signatures, each consisting of the same number of features as are in the primary signature provided in the call to `sigCheck` that created the `SigCheckObject` sampled at random from all available features.

Each random signature will be evaluated in the same manner as the primary signature. If survival data were supplied, a survival analysis will be carried out on the validation samples, and a p-value computed as a performance measure. If no survival data are available, the training samples will be used to train a classifier, and the performance score will be percentage of validation samples correctly classified. (If no validation samples are provided, leave-one-out cross validation will be used to calculate the classification performance for each random signature).

An empirical p-value will be computed based on the percentile rank of the performance of the primary signature compared to a null distribution of the performance of the random signatures.

Value

A result list with the following elements:

- `$checkType` is equal to `"Random"`.
- `$tests` is the number of tests run (equal to `iterations`.)
- `$rank` is the performance rank of the primary signature within the performance of the random signatures.
- `$checkPval` is the empirical p-value computed using the scores of the random signature as a null distribution. A value of zero indicates that no random signatures performed as good or better than the primary signature.
- `$survivalPval` represents the performance of the primary, if survival data were provided.
- `$survivalPvalsRandom` is a vector of performance scores (p-values) for each random signature on the validation samples, if survival data were provided.
- `$trainingPvalsRandom` is a vector of performance scores (p-values) for each random signature on the training samples, if survival data and separate validation samples were provided.
- `$sigPerformance` is the proportion of validation samples correctly classified by the primary signature if a classifier was used.
- `$modePerformance` is the proportion of validation samples correctly classified using a mode classifier.
- `$performanceRandom` is a vector of classification performance scores for each random signature, each indicating the proportion of validation samples correctly classified if a classifier was used.

Author(s)

Rory Stark

See Also

sigCheck, sigCheckAll, sigCheckPermuted, sigCheckKnown, sigCheckPlot

Examples

```
#Disable parallel so Bioconductor build won't hang
library(BiocParallel)
register(SerialParam())

library(breastCancerNKI)
data(nki)
nki <- nki[,!is.na(nki$e.dmfs)]
data(knownSignatures)
ITERATIONS <- 5 # should be at least 20, 1000 for real checks

## survival analysis
check <- sigCheck(nki, classes="e.dmfs", survival="t.dmfs",
                  signature=knownSignatures$cancer$VANTVEER,
                  annotation="HUGO.gene.symbol",
```

```
                    validationSamples=150:319)

randomResult <- sigCheckRandom(check, iterations=ITERATIONS)
randomResult$checkPval
sigCheckPlot(randomResult)
```

Index

∗ **classes**
SigCheckObject-class, 13
∗ **datasets**
classifyResults, 3
knownSignatures, 4
nkiResults, 5
∗ **package**
SigCheck-package, 2

classifyResults, 3

eSet, *14*
ExpressionSet, *3*, *6, 7*, *10*, *13, 14*

fvarLabels, *7*

knownSignatures, 4, *11–13*

MLearn, *3*, *7*

nkiResults, 5

pchisq, *8*
plot, *18*

resultsNKI *(*knownSignatures*)*, 4

SigCheck, *3*, *5*, *9*
SigCheck *(*SigCheck-package*)*, 2
sigCheck, *2*, 6, *9–11*, *13*, *15–20*
SigCheck-package, 2
sigCheckAll, *2*, *5*, *8*, 9, *13*, *16–18*, *20*
sigCheckKnown, *2–4*, *8–10*, 11, *16–18*, *20*
SigCheckObject, *6*, *8, 9*, *11*, *13*, *15*, *19*
SigCheckObject *(*SigCheckObject-class*)*, 13
SigCheckObject-class, 13
sigCheckPermuted, *2*, *8*, *10*, *13*, 15, *17, 18*, *20*
sigCheckPlot, *9, 10*, *13*, 17, *20*
sigCheckRandom, *2, 3*, *8*, *10*, *13*, *16–18*, 19
survdiff, *8*
svmI, *7*

Versioned, *14*
VersionedBiobase, *14*