Package: BioMark (via r-universe)

August 24, 2024

Type Package

Title Find Biomarkers in Two-Class Discrimination Problems

Version 0.4.5

Author Ron Wehrens, Pietro Franceschi

Maintainer Ron Wehrens <`ron.wehrens@gmail.com`>

Description Variable selection methods are provided for several classification methods: the lasso/elastic net, PCLDA, PLSDA, and several t-tests. Two approaches for selecting cutoffs can be used, one based on the stability of model coefficients under perturbation, and the other on higher criticism.

License GPL (>= 2)

Depends pls, glmnet, MASS, st (>= 1.1.6)

LazyLoad yes

NeedsCompilation no

Date/Publication 2015-09-07 18:17:37

Repository https://rwehrens.r-universe.dev

RemoteUrl https://github.com/cran/BioMark

RemoteRef HEAD

RemoteSha bf21eb0c772c2313fd7f3b120ead307668133dc2

Contents

---

aux.biom *Auxiliary functions in the biomarker package*

Description

These functions return coefficient sizes for a variety of modelling methods. Not to be called directly by the user - use function `get.biom` for that.

Usage

```
pcr.coef(X, Y, ncomp, scale.p, ...)
pcr.stab(X, Y, ncomp, scale.p,
           segments = NULL, variables = NULL, ...)
pls.coef(X, Y, ncomp, scale.p, ...)
pls.stab(X, Y, ncomp, scale.p,
           segments = NULL, variables = NULL, ...)
vip.coef(X, Y, ncomp, scale.p, ...)
vip.stab(X, Y, ncomp, scale.p,
         segments = NULL, variables = NULL, ...)
lasso.coef(X, Y, scale.p,
           lasso.opt = biom.options()$lasso,...)
lasso.stab(X, Y, scale.p,
           segments = NULL, variables = NULL, ...)
shrinkt.coef(X, Y, scale.p, ...)
shrinkt.stab(X, Y, scale.p,
             segments = NULL, variables = NULL, ...)
studentt.coef(X, Y, scale.p, ...)
studentt.stab(X, Y, scale.p,
              segments = NULL, variables = NULL, ...)
pval.pcr(X, Y, ncomp, scale.p, npermut)
pval.plsvip(X, Y, ncomp, scale.p, npermut, smethod)
```

Arguments

`X`	Data matrix. Usually the number of columns (variables) is (much) larger than the number of rows (samples).
`Y`	Class indication. Either a factor, or a numeric vector.
`ncomp`	Number of latent variables to use in PCR and PLS (VIP) modelling. In function `get.biom` this may be a vector; in all other functions it should be one number. Default: 2.

`scale.p`	Scaling. This is performed individually in every crossvalidation iteration, and can have a profound effect on the results. Default: "none". Other possible choices: "auto" for autoscaling, "pareto" for pareto scaling, "log" and "sqrt" for log and square root scaling, respectively.
`segments`	matrix where each column indicates a set of samples to be left out of the analysis.
`variables`	indices of variables to be used in the analysis.
`lasso.opt`	optional arguments to the `glmnet` function, in the form of a list.
`...`	Further arguments for modelling functions. Often used to catch unused arguments.
`npermut`	Number of permutations to use in the calculation of the p values.
`smethod`	Either "both", "pls", or "vip" - indicates what coefficients to convert to p values. Both are derived from PLS models so it is much more efficient to calculate them together.

Value

The functions ending in `coef` return t-statistics or model coefficients for all variables. The functions ending in `stab` return these statistics in a matrix, one column per segment. The functions starting with `pval` convert model coefficients or VIP statistics into p values, using permutation resampling.

Author(s)

Ron Wehrens

See Also

`get.biom`, `glmnet`, `scalefun`

`biom.options` *Set or return options for stability-based biomarker selection*

Description

A function to set options for stability-based biomarker selection in the **BioMark** package, or to return the current options.

Usage

```
biom.options(..., reset = FALSE)
```

Arguments

`...`	a single list, a single character vector, or any number of named arguments (*name = value*).
`reset`	logical: if TRUE all options are set to their factory defaults.

Details

If called with no arguments, or with an empty list as the single argument, `biom.options` returns the current options.

If called with a character vector as the single argument, a list with the arguments named in the vector are returned.

If called with a non-empty list as the single arguments, the list elements should be named, and are treated as named arguments to the function.

Otherwise, `biom.options` should be called with one or more named arguments *name = value*. For each argument, the option named *name* will be given the value *value*.

The options are saved in an envirtonment variable `.biom.Options`, and remain in effect until the end of the session. If the environment is saved upon exit, they will be remembered in the next session.

The recognised options are:

max.seg Maximal number of jackknife iterations. Default: 100.

oob.size, oob.fraction Size of the out-of-bag fraction, either given as an absolute number (oob.size) or as a fraction. Default is to leave out ten percent. If oob.size is given explicitly, it takes precedence over oob.fraction. Default: oob.fraction = .1.

variable.fraction Use 1 to always include all variables - use a smaller fraction to have a different random subset of all variables in each iteration (stability-based identification). Default: .7.

ntop The number of "top" coefficients taken into account in stability-based biomarker identification. If a variable appears consistently among the `ntop` biggest coefficients, it is said to be stable. If ntop is a number between 0 and 1, it is taken to indicate the fraction of variables to be included in the model. Default: 10.

min.present The minimal fraction of times a variable should be in the top list to be considered as a potential biomarker (stability-based identification). Setting this argument to 0 will lead to a list containing all coefficients that were present in the top list at least once - a value of 1 only returns those variables that are selected in every iteration. Default: .1.

nset The number of permutations to establish null distributions for PCR, PLS and VIP statistics in the Higher-Criticism approach. Default: 10,000.

fmethods All biomarker selection methods available within BioMark. Currently equal to `c("studentt", "shrinkt", "pcr","pls", "vip", "lasso"`.

univ.methods The names of the univariate biomarker selection methods currently known to BioMark. Currently equal to `c("studentt", "shrinkt")`

HCalpha The default of the alpha parameter in the HC method. Value: 0.1.

lasso a list of arguments passed to the underlying `glmnet` function, such as `family`, `nlambda`, `alpha`, `lambda`, or `lambda.min.ratio`. For binary classification, the "binomial" family is the default, but the most similar setting compared to the other methods in the package is `family = "gaussian"`. For choices other than the default, a warning is printed to the screen.

Value

A list with the (possibly changed) options. If any named argument (or list element) was provided, the list is returned invisibly.

Side Effects

If any named argument (or list element) was provided, `biom.options` updates the elements of the option list `.biom.Options$options`.

Note

This function is based on the `pls.options` function in package **pls**.

Author(s)

Ron Wehrens

See Also

`glmnet`

Examples

```
## Return current options:
biom.options()
biom.options("max.seg")

## Set options:
biom.options(max.seg = 100, oob.fraction = .2)
biom.options(lasso = list(alpha = .75, nlambda = 50))
biom.options()
## the next line removes some options - for these, glmnet defaults will be used
biom.options(lasso = list(alpha = .9, family = "binomial"))

## Restore factory settings:
biom.options(reset = TRUE)
```

`gen.data` *Simulate data sets*

Description

The functions `gen.data` and `gen.data2` generate one or more two-class data matrices where the first `nbiom` variables are changed in the treatment class. The aim is to provide an easy means to evaluate the performance of biomarker identification methods. Function `gen.data` samples from a multivariate normal distribution; `gen.data2` generates spiked data either by adding differences to the first columns, or by multiplying with factors given by the user. Note that whereas `gen.data` will provide completely new simulated data, both for the control and treatment classes, `gen.data2` essentially only changes the biomarker part of the treated class.

Usage

```
gen.data(ncontrol, ntreated = ncontrol, nvar, nbiom = 5, group.diff = 0.5,
         nsimul = 100, means = rep(0, nvar), cormat = diag(nvar))
gen.data2(X, ncontrol, nbiom, spikeI,
          type = c("multiplicative", "additive"),
          nsimul = 100, stddev = .05)
```

Arguments

`ncontrol, ntreated`	Numbers of objects in the two classes. If only ncontrol is given, the two classes are assumed to be of equal size, or, in the case of `gen.data2`, the remainder of the samples are taken to be the treatment samples.
`nvar`	Number of variables.
`nbiom`	Number of biomarkers, i.e. the number of variables to be changed in the treatment class compared to the control class. The variables that are changed are always the first variables in the data matrix.
`group.diff`	group difference; the average difference between values of the biomarkers in the two classes.
`nsimul`	Number of data sets to simulate.
`means`	Mean values of all variables, a vector.
`cormat`	Correlation matrix to be used in the simulation. Default is the identity matrix.
`X`	Experimental data matrix, without group differences.
`spikeI`	A vector of at least three different numbers, used to generate new values for the biomarker variables in the treated class.
`type`	Whether to use multiplication (useful when simulating cases where things like "twofold differences" are relevant), or addition (in the case of absolute differences in the treatment and control groups).
`stddev`	Additional noise: in every simulation, normally distributed noise with a standard deviation of `stddev * mean(spikeI)` will be added to `spikeI` before generating the actual simulated data.

Details

The `spikeI` argument in function `gen.data2` provides the numbers that will be used to artificially "spike" the biomarker variables, either by multiplication (the default) or by addition. To obtain approximate two-fold differences, for example, one could use `spikeI = c(1.8, 2.0, 2.2)`. At least three different values should be given since in most cases more than one set will be simulated and we require different values in the biomarker variables.

Value

A list with the following elements:

`X`	An array of dimension `nobj1 + nobj2` times `nvar` times `nsimul`.
`Y`	The class vector.

`n.biomarkers`	The number of biomarkers.

Note that the biomarkers are always in the first `nbiom` columns of the data matrix.

Author(s)

Ron Wehrens

Examples

```
## Not run:
X <- gen.data(10, nvar = 200)
names(X)
dim(X$X)

set.seed(7)
simdat <- gen.data(10, nvar = 1200, nbiom = 22, nsimul = 1,
                   group.diff = 2)
simdat.stab <- get.biom(simdat$X[,,1], simdat$Y, fmethod = "all",
                        type = "stab", ncomp = 3, scale.p = "auto")
## show LASSO success
traceplot(simdat.stab, lty = 1, col = rep(2:1, c(22, 1610)))

data(SpikePos)
real.markers <- which(SpikePos$annotation$found.in.standards > 0)
X.no.diff <- SpikePos$data[1:20, -real.markers]

set.seed(7)
simdat2 <- gen.data2(X.no.diff, ncontrol = 10, nbiom = 22,
                     spikeI = c(1.2, 1.4, 2), nsimul = 1)
simdat2.stab <- get.biom(simdat2$X[,,1], simdat$Y,
                         fmethod = "all", type = "stab", ncomp = 3,
                         scale.p = "auto")
## show LASSO success
traceplot(simdat2.stab, lty = 1, col = rep(2:1, c(22, 1610)))

## End(Not run)
```

`get.biom` *Get biomarkers discriminating between two classes*

Description

Biomarkers can be identified in several ways: the classical way is to look at those variables with large model coefficients or large t statistics. One other is based on the higher criticism approach (HC), and the third possibility assesses the stability of these coefficients under subsampling of the data set.

Usage

```
get.biom(X, Y, fmethod = "all", type = c("stab", "HC", "coef"),
         ncomp = 2, biom.opt = biom.options(), scale.p = "auto",
         ...)
## S3 method for class 'BMark'
coef(object, ...)
## S3 method for class 'BMark'
print(x, ...)
## S3 method for class 'BMark'
summary(object, ...)
```

Arguments

`X`	Data matrix. Usually the number of columns (variables) is (much) larger than the number of rows (samples).
`Y`	Class indication. For classification with two or more factors a factor; a numeric vector will be interpreted as a regression situation, which can only be tackled by `fmethod = "lasso"`.
`fmethod`	Modelling method(s) employed. The default is to use `"all"`, which will test all methods in the current `biom.options$fmethods` list. Note that from version 0.4.0, `"plsda"` and `"pclda"` are no longer in the list of methods - they have been replaced by `"pls"` and `"pcr"`, respectively. For compatibility reasons, using the old terms will not lead to an error but only a warning.
`type`	Whether to use coefficient size as a criterion (`"coef"`), `"stab"` or `"HC"`.
`ncomp`	Number of latent variables to use in PCR and PLS (VIP) modelling. In function `get.biom` this may be a vector; in all other functions it should be one number. Default: 2.
`biom.opt`	Options for the biomarker selection - a list with several named elements. See `biom.options`.
`scale.p`	Scaling. This is performed individually in every crossvalidation iteration, and can have a profound effect on the results. Default: "auto" (autoscaling). Other possible choices: "none" for no scaling, "pareto" for pareto scaling, "log" and "sqrt" for log and square root scaling, respectively.
`object, x`	A BMark object.
`...`	Further arguments for modelling functions. Often used to catch unused arguments.

Value

Function `get.biom` returns an object of class "BMark", a list containing an element for every `fmethod` that is selected, as well as an element `info`. The individual elements contain information depending on the type chosen: for `type == "coef"`, the only element returned is a matrix containing coefficient sizes. For `type == "HC"` and `type == "stab"`, a list is returned containing elements `biom.indices`, and either `pvals` (for `type == "HC"`) or `fraction.selected` (for `type == "stab"`). Element `biom.indices` contains the indices of the selected variables, and can be extracted using function `selection`. Element `pvals` contains the p values used to perform HC thresholding;

these are presented in the original order of the variables, and can be obtained directly from e.g. t statistics, or from permutation sampling. Element `fraction.selected` indicates in what fraction of the stability selection iterations a particular variable has been selected. The more often it has been selected, the more stable it is as a biomarker. Generic function `coef.biom` extracts model coefficients, p values or stability fractions for types `"coef"`, `"HC"` and `"stab"`, respectively.

Author(s)

Ron Wehrens

See Also

`biom.options`, `get.segments`, `selection`, `scalefun`

Examples

```
## Real apple data (small set)
data(spikedApples)
apple.coef <- get.biom(X = spikedApples$dataMatrix,
                       Y = factor(rep(1:2, each = 10)),
                       ncomp = 2:3, type = "coef")
coef.sizes <- coef(apple.coef)
sapply(coef.sizes, range)

## stability-based selection
set.seed(17)
apple.stab <- get.biom(X = spikedApples$dataMatrix,
                       Y = factor(rep(1:2, each = 10)),
                       ncomp = 2:3, type = "stab")
selected.variables <- selection(apple.stab)
unlist(sapply(selected.variables, function(x) sapply(x, length)))
## Ranging from more than 70 for pcr, approx 40 for pls and student t,
## to 0-29 for the lasso
unlist(sapply(selected.variables,
              function(x) lapply(x, function(xx, y) sum(xx %in% y),
              spikedApples$biom)))
## TPs (stab): all find 5/5, except pcr.2 and the lasso with values for lambda
## larger than 0.0484

unlist(sapply(selected.variables,
              function(x) lapply(x, function(xx, y) sum(!(xx %in% y)),
              spikedApples$biom)))
## FPs (stab): PCR finds most FPs (approx. 60), other latent-variable
## methods approx 40, lasso allows for the optimal selection around
## lambda = 0.0702

## regression example
data(gasoline) ## from the pls package
gasoline.stab <- get.biom(gasoline$NIR, gasoline$octane,
                          fmethod = c("pcr", "pls", "lasso"), type = "stab")
```

```
## Not run:
## Same for HC-based selection
## Warning: takes a long time!
apple.HC <- get.biom(X = spikedApples$dataMatrix,
                     Y = factor(rep(1:2, each = 10)),
                     ncomp = 2:3, type = "HC")
sapply(apple.HC[names(apple.HC) != "info"],
       function(x, y) sum(x$biom.indices %in% y),
       spikedApples$biom)
sapply(apple.HC[names(apple.HC) != "info"],
       function(x, y) sum(!(x$biom.indices %in% y)),
       spikedApples$biom)

## End(Not run)
```

`get.segments` *Subsampling segments*

Description

Provides combinations of samples to be left out in subsampling, with a maximum given by parameter `max.seg`.

Usage

```
get.segments(i1, i2 = NULL, oob.size = 1, max.seg = 100)
```

Arguments

`i1`	either an index vector for objects in class 1, or a classification vector (factor, or numeric), from which the indices of both classes can be derived.
`i2`	if non-NULL, vector indexing objects in class 2.
`oob.size`	number of samples to be left out in every iteration. If one (the default), this corresponds to LOO subsampling.
`max.seg`	maximal number of segments to return. If null, all possible combinations are returned – this option is only possible if oob.size equals 1. If oob.size is larger, max.seg must be defined since the number of possibilities becomes too large for even very small numbers of objects.

Value

Returns a matrix where the columns contain the numbers of the samples to be left out in the respective iterations.

Author(s)

Ron Wehrens

See Also

get.biom

Examples

```
i1 <- seq(1, 10, by = 2)
i2 <- seq(2, 15, by = 2)
get.segments(i1, i2)
get.segments(i1, i2, max.seg = 10)
get.segments(i1, i2, oob.size = 2, max.seg = 10)

I <- rep(1:2, c(5,6))
get.segments(I)
get.segments(I, max.seg = 15)
```

HCthresh *Biomarker thresholding by Higher Criticism*

Description

Higher Criticism (HC) is a second-level significance testing approach to determine which variables in a multivariate set show significant differences in two classes. Function HCthresh selects those p values that are significantly different from what would be expected from their uniform distribution under the null hypothesis.

Usage

```
HCthresh(pvec, alpha = 0.1, plotit = FALSE)
```

Arguments

pvec	Vector of p values.
alpha	Parameter of the HC approach: the maximal fraction of differentially expressed p values.
plotit	Logical, whether or not a plot should be produced.

Details

In HC, one tests the deviation of the expected behaviour of p values under a null distribution. Function HCthresh implements the approach by Donoho and Jin to find out which of these correspond to real differences. The prerequisites are that the true biomarkers are rare (consist of only a small fraction of all variables) and weak (are not able to discriminate between the two classes all by themselves).

Value

A vector containing the ordered indices of the p values satisfying the HC criterion.

Author(s)

Ron Wehrens

References

David Donoho and Jiashun Jin: Higher criticism thresholding: Optimal feature selection when useful features are rare and weak. *PNAS* 108:14790-14795 (2008).

Ron Wehrens and Pietro Franceschi: Thresholding for Biomarker Selection in Multivariate Data using Higher Criticism. Mol. Biosystems (2012). In press. DOI: 10.1039/C2MB25121C

See Also

`get.biom` for general approaches to obtain biomarkers based on multivariate discriminant methods and t statistics

Examples

```
data(spikedApples)
bms <- get.biom(spikedApples$dataMatrix, rep(0:1, each = 10),
                type = "coef", fmethod = "studentt")
bms.pvalues <- 2 * (1 - pt(abs(bms[[1]]), 18))
sum(bms.pvalues < .05)                          ## 15
sum(p.adjust(bms.pvalues, method = "fdr") < .05) ## 4
signif.bms <- HCthresh(bms.pvalues, plotit = TRUE)
length(signif.bms)                              ## 11
```

ROC *ROC curves*

Description

Functions for making, plotting and analysing ROC curves.

Usage

```
ROC(TestResult, ...)
## Default S3 method:
ROC(TestResult, D, take.abs = TRUE, ...)
## S3 method for class 'ROC'
plot(x, type = "b", null.line = TRUE,
xlab = "False Pos. Rate", ylab = "True Pos. Rate",
xlim = c(0, 1), ylim = c(0, 1), main = "ROC", ...)
## S3 method for class 'ROC'
points(x, ...)
## S3 method for class 'ROC'
lines(x, ...)
## S3 method for class 'ROC'
```

```
identify(x, labels = NULL, ..., digits = 1)
## S3 method for class 'ROC'
print(x, ...)
roc.value(found, true, totalN)
AUC(x, max.mspec)
```

Arguments

`TestResult`	Typically regression coefficients or t statistics. Note that when p values are used directly, the least significant values would be selected first. In this case one should use 1/p.
`D`	True, known, differences, either expressed as a vector of 0 and 1 of the same length as `TestResult` or as a vector of indices.
`take.abs`	Logical, indicating whether to take absolute values of the test statistic.
`x`	An object of class ROC.
`type`, `xlab`, `ylab`, `xlim`, `ylim`, `main`, `labels`, `digits`	Standard arguments to functions like `plot` and `identify`.
`null.line`	Logical, whether to draw the line y = x, corresponding to random guessing.
`max.mspec`	Maximal value of the True Positive Rate to consider in AUC calculations. Setting this to a value smaller than one (which is the default) leads to a partial AUC value, which may in many cases be more useful.
`found`	The indices of the coefficients identified with a biomarker identification method.
`true`	The indices of the true biomarkers.
`totalN`	The total number of variables to choose from.
`...`	Further arguments, especially useful in the plotting functions.

Value

Function `ROC` returns a list with elements:

1. sensSensitivity, or True Positive Rate (TPR).
2. mspec1 - Specificity, or False Positive Rate (FPR).
3. testlevels of the test statistic.
4. callFunction call.

Function `roc.value` returns a list with elements `sens` and `mspec`, i.e., one point on a ROC curve.

Function `AUC` returns the area under the curve, measured up to the value of `max.mspec` - if the latter is smaller than 1, it is a partial AUC curve.

Author(s)

Ron Wehrens

References

T. Lumley: ROC curves - in Programme's Niche, R News 4/1, June 2004.

Examples

```
data(spikedApples)
apple.coef <- get.biom(X = spikedApples$dataMatrix,
                       Y = rep(1:2, each = 10),
                       fmethod = "vip",
                       ncomp = 3, type = "coef")

## ROC curve for all VIP values, ordered according to size
true.biom <- (1:ncol(spikedApples$dataMatrix) %in% spikedApples$biom)
vip.roc <- ROC(apple.coef$vip, true.biom)
plot(vip.roc)

## Add stability-based selection point
apple.stab <- get.biom(X = spikedApples$dataMatrix,
                       Y = rep(1:2, each = 10),
                       fmethod = "vip",
                       ncomp = 3, type = "stab")
stab.roc <- roc.value(apple.stab$vip[[1]]$biom.indices,
                      spikedApples$biom,
                      totalN = ncol(spikedApples$dataMatrix))
points(stab.roc, col = "red", pch = 19, cex = 1.5)

## Not run:
## Add HC-based selection point
## Attention: takes approx. 2 minutes on my PC
apple.HC <- get.biom(X = spikedApples$dataMatrix,
                     Y = rep(1:2, each = 10),
                     fmethod = "vip",
                     ncomp = 3, type = "HC")
HC.roc <- roc.value(apple.HC$vip$biom.indices,
                    spikedApples$biom,
                    totalN = ncol(spikedApples$dataMatrix))
points(HC.roc, col = "blue", pch = 19, cex = 1.5)

## End(Not run)
```

`scalefun` *Different forms of scaling*

Description

Function providing different forms of scaling in disciminant analysis - the resulting data matrix is mean-centered after the scaling. Modelled after functions in the st package.

Usage

```
scalefun(sc.p = c("none", "log", "sqrt", "pareto", "auto"))
```

Arguments

`sc.p`	Type of scaling. A pass-through option, performing only mean-centering, is provided by argument "none".

Value

A matrix. The function performs the required scaling, and mean-centers the result.

Author(s)

Ron Wehrens

Examples

```
X <- gen.data(5, nvar = 9, nsimul = 1)
FUN <- scalefun(sc.p = "pareto")
FUN(X$X[,,1])
```

`selection` *Accessor function to the selected variables of a BioMark object*

Description

Convenience function to get the indices of the selection in a BioMark object.

Usage

```
selection(object, ...)
```

Arguments

`object`	An object of class `BioMark`.
`...`	Further arguments, currently ignored.

Value

A vector containing the indices of the selected variables.

See Also

`get.biom`

Examples

```
## stability-based selection
set.seed(17)
data(spikedApples)
apple.stab <- get.biom(X = spikedApples$dataMatrix,
                       Y = factor(rep(1:2, each = 10)),
                       ncomp = 2:3, type = "stab")
selected.variables <- selection(apple.stab)
```

`SpikedApple` *Spike-in metabolomics data for apple extracts*

Description

Data from a spike-in experiment for apple extracts. Twenty apple extracts are divided in two groups, one control, and one spike-in group. The control group is measured without any spiking - the spike-in group is spiked with nine chemical compounds in three different combinations of concentrations. The data provide the experimental data of the forty apple extracts in lists `SpikePos` and `SpikeNeg` for positive and negative ionization, respectively, and in two separate data.frames (`pos.markers` and `neg.markers`) contains information of the features of the standards, i.e., the spike-in compounds.

Usage

```
data(SpikePos)
data(SpikeNeg)
```

Format

`SpikePos` and `SpikeNeg` are lists with three elements:

data Data matrix, describing for each of the forty injections the intensity of the features (columns). Column names consist of a combination of retention time (in seconds) and m/z values, and are sorted on retention time.

classes Class labels for the forty injections (control, or group1, 2 or 3).

annotation Matrix, containing for each of the features XCMS and CAMERA information, such as mz, rt, number of times a feature is identified in the control or spike-in samples, possible isotope or adduct annotation, and whether or not the feature is identified in the standards (the spike-in data).

In addition, `pos.markers` and `neg.markers` contain the information of the standards, i.e. the compounds that are spiked in. These data.frames describe in their rows single features identified with XCMS and CAMERA, using the same settings as the experimental apple data, and have the following columns:

comp The (short) name of the spiked-in compound giving rise to this particular feature.

mz, rt, isotope, adduct Feature information, similar to the information in the `annotation` fields in `SpikePos` and `SpikeNeg`.

feature.nr The number of the corresponding feature in either `SpikePos` or `SpikeNeg`.

group1, group2, group3 Approximate spiking levels for the three groups. A value of 1.0 corresponds to an increase that is roughly equal to the naturally occuring concentration in apple. Exceptions are trans-resveratrol and cyanidin-3-galactoside, both not naturally occuring. These two compounds have been spiked in at one constant level which gives features of comparable size.

Details

This is the complete data set, from which `spikedApples` is a subset, basically presenting the control and group1 information with hand-picked spike-in features. The data in SpikePos and SpikeNeg use CAMERA grouping to automatically determine which features are corresponding to which spike-in compounds. Raw data in CDF format are available from the MetaboLights repository.

Author(s)

Pietro Franceschi

Source

http://www.ebi.ac.uk/metabolights/MTBLS59

P. Franceschi, D. Masuero, U. Vrhovsek, F. Mattivi and R. Wehrens: A benchmark spike-in data set for biomarker identification in metabolomics. J. Chemom. 26, 16-24 (2012).

See Also

spikedApples

Examples

```
data(SpikePos)
plot(SpikePos$annotation[,c("rt", "mz")],
     xlab = "Time (s)", ylab = "m/z",
     main = "Positive ionization mode")
points(pos.markers[!is.na(pos.markers$feature.nr), c("rt", "mz")],
       pch = 19, col = 2)

data(SpikeNeg)
plot(SpikeNeg$annotation[,c("rt", "mz")],
     xlab = "Time (s)", ylab = "m/z",
     main = "Negative ionization mode")
points(neg.markers[!is.na(neg.markers$feature.nr), c("rt", "mz")],
       pch = 19, col = 2)
```

spikedApples *Metabolomics data on spiked apples*

Description

An data set of LC-MS features, obtained from twenty apples. The last ten apples are spiked with known compounds. This set provides a test case for biomarker selection methods: the task is to retrieve the true biomarker variables. The raw LC-MS data have been converted to CDF format and processed with XCMS to obtain the features.

Usage

```
data(spikedApples)
```

Format

The format is a list of four elements:

mz the m/z values of the features (rounded)

rt the retention times of the features

dataMatrix the intensities of the features in the individual samples

biom the indices of the "true" biomarkers

Author(s)

Pietro Franceschi

References

P. Franceschi, D. Masuero, U. Vrhovsek, F. Mattivi and R. Wehrens: A benchmark spike-in data set for biomarker identification in metabolomics. J. Chemom. 26, 16-24 (2012)

R. Wehrens, P. Franceschi, U. Vrhovsek and F. Mattivi. Stability-based biomarker selection. Analytica Chimica Acta (2011), 705, 15-23. http://dx.doi.org/10.1016/j.aca.2011.01.039.

Examples

```
data(spikedApples)
## show features identified in all apples
plot(spikedApples$rt, spikedApples$mz,
     xlab = "Retention time (s)", ylab = "m/z",
     main = "Spiked apples - subset")
```

`traceplot`	*Plot the coefficient or stability trace for the lasso/elastic net biomarker selection.*

Description

The function plots the coefficient or stability traces for the lasso element of a BioMark object.

Usage

```
traceplot(object, ...)
```

Arguments

`object`	An object of class `BioMark`.
`...`	Further plotting arguments.

Author(s)

Ron Wehrens

References

N. Meinshausen and P. Buhlmann: Stability Selection. J. R. Statist. Soc. B 72, 417-473 (2010)

See Also

`get.biom`

Examples

```
library(BioMark)
data(spikedApples)

mycols <- rep("gray", ncol(spikedApples$dataMatrix))
mycols[spikedApples$biom] <- "red"
myltys <- rep(2, ncol(spikedApples$dataMatrix))
myltys[spikedApples$biom] <- 1

par(mfrow = c(1,2))
apple.coef <- get.biom(X = spikedApples$dataMatrix,
                       Y = factor(rep(0:1, each = 10)),
                       ncomp = 2:3, type = "coef")
traceplot(apple.coef, col = mycols, lty = myltys)

apple.stab <- get.biom(X = spikedApples$dataMatrix,
                       Y = factor(rep(0:1, each = 10)),
                       fmethod = c("vip","lasso"), type = "stab")
traceplot(apple.stab, col = mycols, lty = myltys)
```

Index