Package 'iterativeBMAsurv'

May 9, 2026

Type Package

Title The Iterative Bayesian Model Averaging (BMA) Algorithm For Survival Analysis

Version 1.71.0

Date 2008-5-19

Author Amalia Annest, University of Washington, Tacoma, WA Ka Yee Yeung, University of Washington, Seattle, WA

Maintainer Ka Yee Yeung <kayee@u.washington.edu>

Description The iterative Bayesian Model Averaging (BMA) algorithm for survival analysis is a variable selection method for applying survival analysis to microarray data.

Depends BMA, leaps, survival, splines

Imports graphics, grDevices, stats, survival, utils

License GPL (>= 2)

URL http://expression.washington.edu/ibmasurv/protected

biocViews Microarray

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

git_branch devel

git_last_commit e0e1d61

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-08

Contents

iterativeBMAsurv-package 2
crossVal 4
imageplot.iterate.bma.surv 6
iterateBMAsurv.train 7
iterateBMAsurv.train.predict.assess 10
iterateBMAsurv.train.wrapper 13
iterativeBMAsurv-internal 15
predictBicSurv 16
predictiveAssessCategory 17

printTopGenes 19
singleGeneCoxph 21
testCens 22
testData 23
testSurv 23
trainCens 24
trainData 25
trainSurv 25

Index **27**

`iterativeBMAsurv-package`

The Iterative Bayesian Model Averaging (BMA) algorithm for survival analysis

---

Description

The iterative Bayesian Model Averaging (BMA) algorithm for survival analysis is a variable selection method for applying survival analysis to microarray data..

Details

Package:	iterativeBMAsurv
Type:	Package
Version:	0.1.0
Date:	2008-3-24
License:	GPL version 2 or higher

The function `iterateBMAsurv.train` selects relevant variables by iteratively applying the `bic.surv` function from the `BMA` package until all variables in the training data are exhausted. The variables are assumed to be pre-sorted by rank when this function is called. The function `iterateBMAsurv.train.wrapper` acts as a wrapper for `iterateBMAsurv.train`, returning the names of the selected variables and an object of class `bic.surv` if the iterations exhaust all variables in the training set (-1 otherwise). Again, the variables are assumed to be pre-sorted by rank, so calling this function allows users to experiment with different univariate ranking measures. The function `iterateBMAsurv.train.predict.assess` combines the training, prediction, and assessment phases. It returns a list consisting of the numbers of selected genes and models from the training phase, the predicted risk scores of the test samples, and the overall survival analysis statistics indicating the difference between risk groups (p-value, chi-square statistic, and variance matrix). It also writes a Kaplan-Meier survival analysis curve to file, which serves as a pictorial nonparametric estimator of the difference between risk groups. The variables are not assumed to be pre-sorted by rank when this function is called. `iterateBMAsurv.train.predict.assess` calls `singleGeneCoxph`, which ranks the genes based on their log likelihood scores using Cox Proportional Hazards Regression. `iterateBMAsurv.train.predict.assess` calls `iterateBMAsurv.train.wrapper` in its training phase, so if Cox Proportional Hazards Regression is the desired univariate ranking algorithm, then calling this function with the training and testing sets is all that is necessary for a complete survival analysis run. The function `crossVal` performs k runs of n-fold cross validation on a training data set, where k and n are specified by the user. `crossVal` calls `iterateBMAsurv.train.predict.assess` during each fold, so Cox Proportional Hazards Regression is the univariate ranking measure for this function.

Author(s)

Ka Yee Yeung, University of Washington, Seattle, WA, Amalia Annest, University of Washington, Tacoma, WA

Maintainer: Ka Yee Yeung <kayee@u.washington.edu> Amalia Annest <amanu@u.washington.edu>

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train, iterateBMAsurv.train.wrapper, iterateBMAsurv.train.predict.assess, singleGeneCoxph, predictiveAssessCategory, crossVal, trainData, trainSurv, trainCens, testData, testSurv, testCens

Examples

```
library (BMA)
library (iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)
data(testData)
data(testSurv)
data(testCens)

## Use p=10 genes and nbest=5 for fast computation
ret.bma <- iterateBMAsurv.train.predict.assess (train.dat=trainData, test.dat=testData, surv.time.train=trai

## Extract the statistics from this survival analysis run
number.genes <- ret.bma$nvar
number.models <- ret.bma$nmodel
evaluate.success <- ret.bma$statistics

## Perform 1 run of 2-fold cross validation on the training set, using p=10 genes and nbest=5 for fast computatio
cv <- crossVal(exset=trainData, survTime=trainSurv, censor=trainCens, diseaseType="DLBCL", noFolds=2, noRuns=
```

crossVal *Cross Validation for Iterative Bayesian Model Averaging*

Description

This function performs k runs of n-fold cross validation on a training dataset for survival analysis on microarray data, where k and n are specified by the user.

Usage

```
crossVal(exset, survTime, censor, diseaseType="cancer", nbest=10, maxNvar=25, p=100, cutPoint=50, v
```

Arguments

exset	Data matrix for the training set where columns are variables and rows are observations. In the case of gene expression data, the columns (variables) represent genes, while the rows (observations) represent samples. The data is not assumed to be pre-sorted by rank.
survTime	Vector of survival times for the patient samples in the training set. Survival times are assumed to be presented in uniform format (e.g., months or days), and the length of this vector should be equal to the number of rows in exset.
censor	Vector of censor data for the patient samples in the training set. In general, 0 = censored and 1 = uncensored. The length of this vector should equal the number of rows in exset and the number of elements in survTime.
diseaseType	String denoting the type of disease in the training dataset (used for writing to file). Default is 'cancer'.
nbest	A number specifying the number of models of each size returned to bic.surv in the BMA package. The default is 10.
maxNvar	A number indicating the maximum number of variables used in each iteration of bic.surv from the BMA package. The default is 25.
p	A number indicating the maximum number of top univariate genes used in the iterative bic.surv algorithm. This number is assumed to be less than the total number of genes in the training data. A larger p usually requires longer computational time as more iterations of the bic.surv algorithm are potentially applied. The default is 100.
cutPoint	Threshold percent for separating high- from low-risk groups. The default is 50.
verbose	A boolean variable indicating whether or not to print interim information to the console. The default is FALSE.
noFolds	A number specifying the desired number of folds in each cross validation run. The default is 10.
noRuns	A number specifying the desired number of cross validation runs. The default is 10.

Details

This function performs k runs of n-fold cross validation, where k and n are specified by the user through the `noRuns` and `noFolds` arguments respectively. For each run of cross validation, the training set, survival times, and censor data are re-ordered according to a random permutation. For each fold of cross validation, $1/n$th of the data is set aside to act as the validation set. In each fold, the `iterateBMAsurv.train.predict.assess` function is called in order to carry out a complete run of survival analysis. This means the univariate ranking measure for this cross validation function is Cox Proportional Hazards Regression; see `iterateBMAsurv.train.wrapper` to experiment with alternate univariate ranking methods. With each run of cross validation, the survival analysis statistics are saved and written to file.

Value

The output of this function is a series of files giving information on cross validation results. The file beginning with 'foldresults' contains information for every fold in the form of a 2 x 2 table indicating the number of test samples in each category (high-risk or censored, high-risk or uncensored, low-risk or censored, low-risk or uncensored). This file also gives the accumulated percentage of uncensored statistic from each run. The file beginning with 'runresults' gives the total number of test samples assigned to each category along with percentage uncensored across the entire run. The end of this file contains this same information, averaged across all runs. The file beginning with 'stats' gives the statistics from each fold, including the p-value, chi-square statistic, and variance matrix. Finally, the file beginning with 'avg_p_value_chi_square' gives the overall means and standard deviations of the p-values and chi-square statistics across all runs and all folds.

Note

The `BMA` package is required. Also, smaller training sets may lead to cross validation folds where all test samples are assigned to one risk group or all samples are in the same censor category (all samples are either censored or uncensored). In this case, the fold is skipped, and cross validation proceeds from the next fold. This particular error will be evidenced by a missing fold result in the output files. All averages will be calculated as if this fold had never occurred.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

`iterateBMAsurv.train.predict.assess` `iterateBMAsurv.train.wrapper`, `iterateBMAsurv.train`, `singleGeneCoxph`, `predictBicSurv`, `predictiveAssessCategory`, `trainData`, `trainSurv`, `trainCens`

Examples

```
library (BMA)
library(iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)

## Perform 1 run of 2-fold cross validation on the training set, using p=10 genes and nbest=5 for fast computatio
cv <- crossVal (exset=trainData, survTime=trainSurv, censor=trainCens, diseaseType="DLBCL", noRuns=1, noFolds

## Upon completion of this function, all relevant output files will be in the working R directory.
```

`imageplot.iterate.bma.surv`

An image plot visualization tool

Description

Create a visualization of the models and variables selected by the iterative BMA algorithm.

Usage

```
imageplot.iterate.bma.surv (bicreg.out, color="default", ...)
```

Arguments

`bicreg.out`	An object of type 'bicreg', 'bic.glm' or 'bic.surv'
`color`	The color of the plot. The value 'default' uses the current default R color scheme for image. The value 'blackandwhite' produces a black and white image.
`...`	Other parameters to be passed to the image and axis functions.

Details

This function is a modification of the `imageplot.bma` function from the `BMA` package. The difference is that variables (genes) with `probne0` equal to 0 are removed before plotting. The arguments of this function are identical to those in `imageplot.bma`.

Value

An heatmap-style image, with the BMA selected variables on the vertical axis, and the BMA selected models on the horizontal axis. The variables (genes) are sorted in descreasing order of the posterior probability that the variable is not equal to 0 (`probne0`) from top to bottom. The models are sorted in descreasing order of the model posterior probability (`postprob`) from left to right.

Note

The `BMA` package is required.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Clyde, M. (1999) Bayesian Model Averaging and Model Search Strategies (with discussion). In Bayesian Statistics 6. J.M. Bernardo, A.P. Dawid, J.O. Berger, and A.F.M. Smith eds. Oxford University Press, pages 157-185.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train.wrapper, iterateBMAsurv.train.predict.assess, trainData, trainSurv, trainCens

Examples

```
library (BMA)
library (iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)

## Training phase: select relevant genes
## Assumes the training data is in sorted order with the desired number of genes
ret.bic.surv <- iterateBMAsurv.train.wrapper (x=trainData, surv.time=trainSurv, cens.vec=trainCens)

## Produce an image plot to visualize the selected genes and models
imageplot.iterate.bma.surv (ret.bic.surv$obj)
```

---

iterateBMAsurv.train *Iterative Bayesian Model Averaging: training*

Description

Survival analysis and variable selection on microarray data. This is a multivariate technique to select a small number of relevant variables (typically genes) to perform survival analysis on microarray data. This function performs the training phase. It repeatedly calls `bic.surv` from the `BMA` package until all variables are exhausted. The variables in the dataset are assumed to be pre-sorted by rank.

Usage

```
iterateBMAsurv.train (x, surv.time, cens.vec, curr.mat, stopVar=0, nextVar, nbest=10, maxNvar=25, m
```

Arguments

`x`	Data matrix where columns are variables and rows are observations. The variables (columns) are assumed to be sorted using a univariate measure. In the case of gene expression data, the columns (variables) represent genes, while the rows (observations) represent samples.

surv.time	Vector of survival times for the patient samples. Survival times are assumed to be presented in uniform format (e.g., months or days), and the length of this vector should be equal to the number of rows in x.
cens.vec	Vector of censor data for the patient samples. In general, 0 = censored and 1 = uncensored. The length of this vector should equal the number of rows in x and the number of elements in surv.time.
curr.mat	Matrix of independent variables in the active bic.surv window. There can be at most maxNvar variables in the window at any given time.
stopVar	0 to continue iterations, 1 to stop iterations (default 0)
nextVar	Integer placeholder indicating the next variable to be brought into the active bic.surv window.
nbest	A number specifying the number of models of each size returned to bic.surv in the BMA package. The default is 10.
maxNvar	A number indicating the maximum number of variables used in each iteration of bic.surv from the BMA package. The default is 25.
maxIter	A number indicating the maximum iterations of bic.surv. The default is 200000.
thresProbne0	A number specifying the threshold for the posterior probability that each variable (gene) is non-zero (in percent). Variables (genes) with such posterior probability less than this threshold are dropped in the iterative application of bic.surv. The default is 1 percent.
verbose	A boolean variable indicating whether or not to print interim information to the console. The default is FALSE.
suff.string	A string for writing to file.

Details

The training phase consists of first ordering all the variables (genes) by a univariate measure such as Cox Proportional Hazards Regression, and then iteratively applying the bic.surv algorithm from the BMA package. In the first application of the bic.surv algorithm, the top maxNvar univariate ranked genes are used. After each application of the bic.surv algorithm, the genes with probne0 < thresProbne0 are dropped, and the next univariate ordered genes are added to the active bic.surv window.

Value

On the last iteration of bic.surv, four items are returned:

curr.mat	A vector containing the names of the variables (genes) from the final iteration of bic.surv
.	
stopVar	The ending value of stopVar after all iterations.
nextVar	The ending value of nextVar after all iterations.
	An object of class bic.surv resulting from the last iteration of bic.surv. The object is a list consisting of the following components:
	namesx the names of the variables in the last iteration of bic.surv.
	postprob the posterior probabilities of the models selected.
	label labels identifying the models selected.

bic values of BIC for the models.

size the number of independent variables in each of the models.

which a logical matrix with one row per model and one column per variable indicating whether that variable is in the model.

probne0 the posterior probability that each variable is non-zero (in percent).

postmean the posterior mean of each coefficient (from model averaging).

postsd the posterior standard deviation of each coefficient (from model averaging).

condpostmean the posterior mean of each coefficient conditional on the variable being included in the model.

condpostsd the posterior standard deviation of each coefficient conditional on the variable being included in the model.

mle matrix with one row per model and one column per variable giving the maximum likelihood estimate of each coefficient for each model.

se matrix with one row per model and one column per variable giving the standard error of each coefficient for each model.

reduced a logical indicating whether any variables were dropped before model averaging.

dropped a vector containing the names of those variables dropped before model averaging.

call the matched call that created the bma.lm object.

Note

The BMA package is required.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train.wrapper, iterateBMAsurv.train.predict.assess, singleGeneCoxph, predictBicSurv, trainData, trainSurv, trainCens, testData

Examples

```
library(BMA)
library(iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)
data(testData)
```

```
## Training data should be pre-sorted before beginning

## Initialize the matrix for the active bic.surv window with variables 1 through maxNvar
maxNvar <- 25
curr.mat <- trainData[, 1:maxNvar]
nextVar <- maxNvar + 1

## Training phase: select relevant genes using nbest=5 for fast computation
ret.bic.surv <- iterateBMAsurv.train (x=trainData, surv.time=trainSurv, cens.vec=trainCens, curr.mat, stopVar

# Apply bic.surv again using selected genes
ret.bma <- bic.surv (x=ret.bic.surv$curr.mat, surv.t=trainSurv, cens=trainCens, nbest=5, maxCol=(maxNvar+1))

## Get the matrix for genes with probne0 > 0
ret.gene.mat <- ret.bic.surv$curr.mat[ret.bma$probne0 > 0]

## Get the gene names from ret.gene.mat
selected.genes <- dimnames(ret.gene.mat)[[2]]

## Show the posterior probabilities of selected models
ret.bma$postprob

## Get the subset of test data with the genes from the last iteration of
## 'bic.surv'
curr.test.dat <- testData[, selected.genes]

## Compute the predicted risk scores for the test samples
y.pred.test <- apply (curr.test.dat, 1, predictBicSurv, postprob.vec=ret.bma$postprob, mle.mat=ret.bma$mle)
```

`iterateBMAsurv.train.predict.assess`
Iterative Bayesian Model Averaging: training, prediction, assessment

---

Description

Survival analysis and variable selection on microarray data. This is a multivariate technique to select a small number of relevant variables (typically genes) to perform survival analysis on microarray data. This function performs the training, prediction, and assessment steps. The data is not assumed to be pre-sorted by rank before this function is called.

Usage

```
iterateBMAsurv.train.predict.assess (train.dat, test.dat, surv.time.train, surv.time.test, cens.ve
```

Arguments

`train.dat`	Data matrix for the training set where columns are variables and rows are observations. In the case of gene expression data, the columns (variables) represent genes, while the rows (observations) represent samples. If Cox Proportional Hazards Regression is the desired univariate ranking measure, the data does not need to be pre-sorted. To use a different univariate ranking measure, see `iterateBMAsurv.train.wrapper`.

test.dat	Data matrix for the test set where columns are variables and rows are observations. In the case of gene expression data, the columns (variables) represent genes, while the rows (observations) represent samples. The test set should contain the same variables as the training set.
surv.time.train	Vector of survival times for the patient samples in the training set. Survival times are assumed to be presented in uniform format (e.g., months or days), and the length of this vector should be equal to the number of rows in train.dat.
surv.time.test	Vector of survival times for the patient samples in the test set. Survival times are assumed to be presented in uniform format (e.g., months or days), and the length of this vector should be equal to the number of rows in test.dat.
cens.vec.train	Vector of censor data for the patient samples in the training set. In general, 0 = censored and 1 = uncensored. The length of this vector should equal the number of rows in train.dat and the number of elements in surv.time.train.
cens.vec.test	Vector of censor data for the patient samples in the test set. In general, 0 = censored and 1 = uncensored. The length of this vector should equal the number of rows in train.dat and the number of elements in surv.time.test.
p	A number indicating the maximum number of top univariate genes used in the iterative bic.surv algorithm. This number is assumed to be less than the total number of genes in the training data. A larger p usually requires longer computational time as more iterations of the bic.surv algorithm are potentially applied. The default is 100.
nbest	A number specifying the number of models of each size returned to bic.surv in the BMA package. The default is 10.
maxNvar	A number indicating the maximum number of variables used in each iteration of bic.surv from the BMA package. The default is 25.
maxIter	A number indicating the maximum iterations of bic.surv. The default is 200000.
thresProbne0	A number specifying the threshold for the posterior probability that each variable (gene) is non-zero (in percent). Variables (genes) with such posterior probability less than this threshold are dropped in the iterative application of bic.surv. The default is 1 percent.
cutPoint	Threshold percent for separating high- from low-risk groups. The default is 50.
verbose	A boolean variable indicating whether or not to print interim information to the console. The default is FALSE.
suff.string	A string for writing to file.

Details

This function consists of the training phase, the prediction phase, and the assessment phase. The training phase first orders all the variables (genes) by a univariate measure called Cox Proportional Hazards Regression, and then iteratively applies the bic.surv algorithm from the BMA package. The prediction phase uses the variables (genes) selected in the training phase to predict the risk scores of the patient samples in the test set. In the assessment phase, the risk scores are used to designate each test sample as either high-risk or low-risk based on the user-designated cutPoint. Prediction accuracy is measured by the p-value difference between groups as calculated through the central chi-square distribution. In addition, a Kaplan-Meier Survival Analysis Curve illustrating the difference between risk groups is written to file in the working R directory. If Cox Proportional Hazards Regression is the desired univariate ranking algorithm, then calling this function with the training and testing sets is all that is necessary for a complete survival analysis run.

Value

If all samples are assigned to a single risk group or all samples are in the same censor category, an error message is printed and a boolean variable `success` is returned as FALSE.If both risk groups are present in the patient test samples, a Kaplan-Meier Survival Analysis Curve is written to file, and a list with 6 components is returned:

`nvar`	The number of variables selected by the last iteration of `bic.surv`.
`nmodel`	The number of models selected by the last iteration of `bic.surv`.
`ypred`	The predicted risk scores on the test samples.
`result.table`	A 2 x 2 table indicating the number of test samples in each category (high-risk/censored, high-risk/uncensored, low-risk/censored, low-risk/uncensored).
`statistics`	An object of class `survdiff` that contains the statistics from survival analysis, including the variance matrix, chi-square statistic, and p-value.
`success`	A boolean variable returned as TRUE if both risk groups are present in the patient test samples.

Note

The `BMA` package is required.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

`iterateBMAsurv.train.wrapper`, `iterateBMAsurv.train`, `singleGeneCoxph`, `predictBicSurv`, `predictiveAssessCategory`, `trainData`, `trainSurv`, `trainCens`, `testData`, `testSurv`, `testCens`

Examples

```
library (BMA)
library(iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)
data(testData)
data(testSurv)
data(testCens)

## Use p=10 genes and nbest=5 for fast computation
ret.bma <- iterateBMAsurv.train.predict.assess (train.dat=trainData, test.dat=testData, surv.time.train=trai

## Extract the statistics from this survival analysis run
```

```
number.genes <- ret.bma$nvar
number.models <- ret.bma$nmodel
evaluate.success <- ret.bma$statistics
```

`iterateBMAsurv.train.wrapper`
Iterative Bayesian Model Averaging: training

Description

This function is a wrapper for `iterateBMAsurv.train`, which repeatedly calls `bic.surv` from the `BMA` package until all variables are exhausted. At the point when this function is called, the variables in the dataset are assumed to be pre-sorted by rank.

Usage

```
iterateBMAsurv.train.wrapper (x, surv.time, cens.vec, nbest=10,
    maxNvar=25, maxIter=200000, thresProbne0=1, verbose=FALSE, suff.string="")
```

Arguments

`x`	Data matrix where columns are variables and rows are observations. The variables (columns) are assumed to be sorted using a univariate measure. In the case of gene expression data, the columns (variables) represent genes, while the rows (observations) represent samples.
`surv.time`	Vector of survival times for the patient samples. Survival times are assumed to be presented in uniform format (e.g., months or days), and the length of this vector should be equal to the number of rows in x.
`cens.vec`	Vector of censor data for the patient samples. In general, 0 = censored and 1 = uncensored. The length of this vector should equal the number of rows in x and the number of elements in surv.time.
`nbest`	A number specifying the number of models of each size returned to `bic.surv` in the `BMA` package. The default is 10.
`maxNvar`	A number indicating the maximum number of variables used in each iteration of `bic.surv` from the `BMA` package. The default is 25.
`maxIter`	A number indicating the maximum iterations of `bic.surv`. The default is 200000.
`thresProbne0`	A number specifying the threshold for the posterior probability that each variable (gene) is non-zero (in percent). Variables (genes) with such posterior probability less than this threshold are dropped in the iterative application of `bic.surv`. The default is 1 percent.
`verbose`	A boolean variable indicating whether or not to print interim information to the console. The default is FALSE.
`suff.string`	A string for writing to file.

Details

In this wrapper function for `iterateBMAsurv.train`, the variables are assumed to be sorted, and `bic.surv` is called repeatedly until all the variables have been exhausted. In the first application of the `bic.surv` algorithm, the top `maxNvar` univariate ranked genes are used. After each application of the `bic.surv` algorithm, the genes with `probne0` < `thresProbne0` are dropped, and the next univariate ordered genes are added to the `bic.surv` window. The function `iterateBMAsurv.train.predict.assess` calls `SingleGeneCoxph` before calling this function. Using this function directly, users can experiment with alternative univariate measures.

Value

If `maxIter` is reached or the iterations stop before all variables are exhausted, -1 is returned. If all variables are exhausted, two items are returned:

`curr.names` A vector containing the names of the variables (genes) from the last iteration of `bic.surv`

.

`obj` An object of class `bic.surv` returned by the last iteration of `bic.surv`. The object of class `bic.surv` is a list consisting of the following components:

namesx the names of the variables in the last iteration of `bic.surv`.

postprob The posterior probabilities of the models selected.

label Labels identifying the models selected.

bic Values of BIC for the models.

size The number of independent variables in each of the models.

which A logical matrix with one row per model and one column per variable indicating whether that variable is in the model.

probne0 The posterior probability that each variable is non-zero (in percent).

postmean The posterior mean of each coefficient (from model averaging).

postsd The posterior standard deviation of each coefficient (from model averaging).

condpostmean The posterior mean of each coefficient conditional on the variable being included in the model.

condpostsd The posterior standard deviation of each coefficient conditional on the variable being included in the model.

mle Matrix with one row per model and one column per variable giving the maximum likelihood estimate of each coefficient for each model.

se Matrix with one row per model and one column per variable giving the standard error of each coefficient for each model.

reduced A logical indicating whether any variables were dropped before model averaging.

dropped A vector containing the names of those variables dropped before model averaging.

call The matched call that created the bma.lm object.

Note

The `BMA` package is required.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train.predict.assess, iterateBMAsurv.train, predictiveAssessCategory, singleGeneCoxph, trainData, trainSurv, trainCens

Examples

```
library (BMA)
library(iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)

## Training data should be pre-sorted before beginning

## Run iterative bic.surv, using nbest=5 for fast computation
ret.list <- iterateBMAsurv.train.wrapper (x=trainData, surv.time=trainSurv, cens.vec=trainCens, nbest=5)

## Extract the 'bic.surv' object
ret.bma <- ret.list$obj

## Extract the names of the genes from the last iteration of 'bic.surv'
gene.names <- ret.list$curr.names
```

---

iterativeBMAsurv-internal

Internal functions for iterativeBMAsurv

Description

Internal functions for iterativeBMAsurv, not meant to be called directly.

Usage

```
iterateBMAinit (x, maxNvar = 25)
assignRiskGroup (x, quantile.cutPoint)
crossVal.init (exset, noFolds, noRuns, nbest, maxNvar, p, cutPoint)
crossVal.run (exset, survTime, censor, diseaseType, nbest, maxNvar, p, cutPoint, rp.mat, overall.av
crossVal.fold (cvpass, survtimepass, censorpass, diseaseType, nbest, maxNvar, p, cutPoint, lastSamp
```

```
crossVal.tabulate (results, single.run.result.mat)
crossVal.final.calc (overall.average.results, noRuns)
imageplot.bma.mod (bicreg.out, color = "default", ...)
```

---

predictBicSurv *Predicted patient risk scores from iterative Bayesian Model Averaging*

Description

This function predicts the risk scores for patient samples in the test set.

Usage

```
predictBicSurv(newdata.vec, postprob.vec, mle.mat)
```

Arguments

newdata.vec	A vector consisting of the data from a test sample.
postprob.vec	A vector consisting of the posterior probability of each BMA selected model.
mle.mat	A matrix with one row per model and one column per variable giving the maximum likelihood estimate of each coefficient for each bic.surv selected model.

Details

The function begins by computing the risk score of each model k in the selected set of models M. The risk score for a model k = sum (coefficient in model k * corresponding expression level in newdata.vec). The function then predicts a patient risk score by summing the product of the posterior probability of model k and the risk score of model k over all models in M. In other words, predicted patient risk score = sum (postprob model k * risk score model k). This function is called by iterateBMAsurv.train.predict.assess.

Value

A real number representing the predicted risk score of a given patient sample.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train.predict.assess, iterateBMAsurv.train.wrapper, predictiveAssessCategory, trainData, trainSurv, trainCens, testData

Examples

```
library (BMA)
library (iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)
data(testData)

## Training phase: select relevant genes. Assume the training data is sorted
## and includes the desired number of top-ranked genes.
ret.list <- iterateBMAsurv.train.wrapper (x=trainData, surv.time=trainSurv, cens.vec=trainCens, nbest=5)
ret.bma <- ret.list$obj

## Get the selected genes with probne0 > 0
selected.genes <- ret.list$curr.names[ret.bma$probne0 > 0]

## Get the subset of test data with the genes from the last iteration of bic.surv
curr.test.dat <- testData [, selected.genes]

## Compute the predicted risk scores for the test samples
y.pred.test <- apply (curr.test.dat, 1, predictBicSurv, postprob.vec=ret.bma$postprob, mle.mat=ret.bma$mle)
```

---

predictiveAssessCategory

Risk Groups: assignment of patient test samples

Description

This function assigns a risk group (high-risk or low-risk) to each patient sample in the test set based on the value of the patient's predicted risk score. The `cutPoint` between high-risk and low-risk is designated by the user.

Usage

```
predictiveAssessCategory (y.pred.test, y.pred.train, cens.vec.test, cutPoint=50)
```

Arguments

`y.pred.test`	A vector containing the predicted risk scores of the test samples.
`y.pred.train`	A vector containing the computed risk scores of the training samples.
`cens.vec.test`	A vector of censor data for the patient samples in the test set. In general, 0 = censored and 1 = uncensored.
`cutPoint`	Threshold percent for separating high- from low-risk groups. The default is 50.

Details

This function begins by using the computed risk scores of the training set (`y.pred.train`) to define a real-number empirical cutoff point between high- and low-risk groups. The cutoff point is determined by the percentile `cutPoint` as designated by the user. The predicted risk scores from the test samples are then matched against this cutoff point to determine whether they belong in the high-risk or the low-risk category.

Value

A list consisting of 2 components:

assign.risk	A 2 x 2 table indicating the number of test samples in each category (high-risk/censored, high-risk/uncensored, low-risk/censored, low-risk/uncensored).
groups	A list of all patient samples in the test set with their corresponding 'High-risk' or 'Low-risk' designations.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train.predict.assess, predictBicSurv, singleGeneCoxph, printTopGenes, trainData, trainSurv, trainCens, testData, testSurv, testCens,

Examples

```
library(BMA)
library(iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)
data(testData)
data(testSurv)
data(testCens)

## Training should be pre-sorted before beginning

## Initialize the matrix for the active bic.surv window with variables 1 through maxNvar
maxNvar <- 25
curr.mat <- trainData[, 1:maxNvar]
nextVar <- maxNvar + 1

## Training phase: select relevant genes, using nbest=5 for fast computation
ret.bic.surv <- iterateBMAsurv.train (x=trainData, surv.time=trainSurv, cens.vec=trainCens, curr.mat, stopVar

# Apply bic.surv again using selected genes
ret.bma <- bic.surv (x=ret.bic.surv$curr.mat, surv.t=trainSurv, cens=trainCens, nbest=5, maxCol=(maxNvar+1))

## Get the matrix for genes with probne0 > 0
ret.gene.mat <- ret.bic.surv$curr.mat[ret.bma$probne0 > 0]

## Get the gene names from ret.gene.mat
selected.genes <- dimnames(ret.gene.mat)[[2]]
```

```
## Show the posterior probabilities of selected models
ret.bma$postprob

## Get the subset of test data with the genes from the last iteration of 'bic.surv'
curr.test.dat <- testData[, selected.genes]

## Compute the predicted risk scores for the test samples
y.pred.test <- apply (curr.test.dat, 1, predictBicSurv, postprob.vec=ret.bma$postprob, mle.mat=ret.bma$mle)

## Compute the risk scores in the training set
y.pred.train <- apply (trainData[, selected.genes], 1, predictBicSurv, postprob.vec=ret.bma$postprob, mle.mat

## Assign risk categories for test samples
ret.table <- predictiveAssessCategory (y.pred.test, y.pred.train, testCens, cutPoint=50)

## Extract risk group vector and risk group table
risk.list <- ret.table$groups
risk.table <- ret.table$assign.risk

## Create a survival object from the test set
mySurv.obj <- Surv(testSurv, testCens)

## Extract statistics including p-value and chi-square
stats <- survdiff(mySurv.obj ~ unlist(risk.list))

## The entire block of code above can be executed simply by calling
## 'iterateBMAsurv.train.predict.assess'
```

`printTopGenes` *Write a training set including the top-ranked G variables from a sorted matrix to file*

Description

This function takes a matrix of rank-ordered variables and writes a training set containing the top G variables in the matrix to file.

Usage

```
printTopGenes (retMatrix, numGlist=c(10, 30, 50, 100, 500, 1000, ncol(trainData)), trainData, myPre
```

Arguments

`retMatrix`	A three-column matrix where the first column contains the sorted variable names (the top log-ranked variable appears first), the second column contains the original index of the variables, and the third column contains the variable ranking from 1 to ncol(trainData).
`numGlist`	A list of values for the desired number of top-ranked variables to be written to file. A separate file will be written for each number G in the list, containing genes 1:G (default = c(10, 30, 50, 100, 500, 1000, ncol(trainData))).

trainData	Data matrix where columns are variables and rows are observations. In the case of gene expression data, the columns (variables) represent genes, while the rows (observations) represent patient samples.
myPrefix	A string prefix for the filename (default = 'sorted_topCoxphGenes_').

Details

This function is called by iterateBMAsurv.train.predict.assess. It is meant to be used in conjunction with singleGeneCoxph, as the retMatrix argument is returned by singleGeneCoxph.

Value

A file or files consisting of the training data sorted in descending order by the top-ranked G variables (one file for each G in numGList).

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train.predict.assess, singleGeneCoxph, trainData, trainSurv, trainCens,

Examples

```
library(BMA)
library(iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)

## Start by ranking and sorting the genes; in this case we use the Cox Proportional Hazards Model
sorted.genes <- singleGeneCoxph(trainData, trainSurv, trainCens)

## Write top 100 genes to file
sorted.top.genes <- printTopGenes(retMatrix=sorted.genes, 100, trainData)

## The file, 'sorted_topCoxphGenes_100', is now in the working R directory.
```

`singleGeneCoxph` *Univariate Cox Proportional Hazards Model for selecting top log-ranked predicitve variables*

Description

This is a univariate technique to rank variables by their predictive relevance for use in survival analysis on microarray data. The log likelihood is computed for each indiviual variable, where a larger log likelihood value indicates a higher rank.

Usage

```
singleGeneCoxph(trainData, survData, censoredData)
```

Arguments

`trainData`	Data matrix where columns are variables and rows are observations. In the case of gene expression data, the columns (variables) represent genes, while the rows (observations) represent patient samples.
`survData`	Vector of survival times for the patient samples. Survival times are assumed to be presented in uniform format (e.g., months or days), and the length of this vector should be equal to the number of rows in trainData.
`censoredData`	Vector of censor data for the patient samples. In general, 0 = censored and 1 = uncensored. The length of this vector should equal the number of rows in trainData and the number of elements in survData.

Details

This function is called by `iterateBMAsurv.train.predict.assess`.

Value

This function returns a sorted three-column matrix of the training data variables. The first column gives the variable names with the top log-ranked variable appearing first. The second column gives the original indexes of the variables, and the third column gives the rank of the variables from 1 through ncol(trainData). The matrix is also written to file in the working R directory under the filename 'sorted_loglik.txt'.

References

Annest, A., Yeung, K.Y., Bumgarner, R.E., and Raftery, A.E. (2008). Iterative Bayesian Model Averaging for Survival Analysis. Manuscript in Progress.

Cox, D. (1972). Regression Models and Life Tables. Journal of the Royal Statistical Society Series B 34: 187-220.

Raftery, A.E. (1995). Bayesian model selection in social research (with Discussion). Sociological Methodology 1995 (Peter V. Marsden, ed.), pp. 111-196, Cambridge, Mass.: Blackwells.

Volinsky, C., Madigan, D., Raftery, A., and Kronmal, R. (1997) Bayesian Model Averaging in Proprtional Hazard Models: Assessing the Risk of a Stroke. Applied Statistics 46: 433-448.

Yeung, K.Y., Bumgarner, R.E. and Raftery, A.E. (2005) Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. Bioinformatics 21: 2394-2402.

See Also

iterateBMAsurv.train.predict.assess, printTopGenes, trainData, trainSurv, trainCens

Examples

```
library(BMA)
library(iterativeBMAsurv)
data(trainData)
data(trainSurv)
data(trainCens)

sorted.genes <- singleGeneCoxph(trainData, trainSurv, trainCens)

## Write top 100 genes to file
sorted.top.genes <- printTopGenes(retMatrix=sorted.genes, 100, trainData)

## The file, 'sorted_topCoxphGenes_100', is now in the working R directory.
```

---

testCens *Sample Test Data for the Iterative BMA Algorithm for Survival Analysis*

Description

This is an diffuse large B-cell lymphoma (DLBCL) dataset from Rosenwald et al. (2002). This is a vector of censor information for 36 test samples. In this vector, 0 = censored and 1 = uncensored.

Usage

```
data(testCens)
```

Format

The vector is called testCens.

Source

Full dataset: http://llmpp.nih.gov/DLBCL/.

References

Rosenwald, A., Wright, G., Wing, C., Connors, J., Campo, E. et al. (2002). The Use of Molecular Profiling to Predict Survival After Chemotherapy for Diffuse Large-B-Cell Lymphoma. The New England Journal of Medicine, 346(25), 1937-1947.

testData *Sample Test Data for the Iterative BMA Algorithm for Survival Analysis*

Description

This is an adapted diffuse large B-cell lymphoma (DLBCL) dataset from Rosenwald et al. (2002). This data matrix consists of the expression levels from 36 DLBCL samples (rows), and 100 top univariate genes (columns). This dataset is used as a sample test set in our examples.

Usage

```
data(trainData)
```

Format

The data matrix is called `testData`. Each entry in the matrix represents the expression level of one gene from a DLBCL sample.

Details

We started with the full expression data from Rosenwald et al. (2002), which is available along with corresponding patient information at their supplemental website http://llmpp.nih.gov/DLBCL/. We selected a subset of the 80 test samples, and then performed Cox Proportional Hazards Regression to obtain the 100 genes with the highest log likelihood. The filtered dataset consists of 36 test samples, and it is available at our website http://expression.washington.edu/ibmasurv/protected.

Source

Full dataset: http://llmpp.nih.gov/DLBCL/.

References

Rosenwald, A., Wright, G., Wing, C., Connors, J., Campo, E. et al. (2002). The Use of Molecular Profiling to Predict Survival After Chemotherapy for Diffuse Large-B-Cell Lymphoma. The New England Journal of Medicine, 346(25), 1937-1947.

testSurv *Sample Test Data for the Iterative BMA Algorithm for Survival Analysis*

Description

This is an diffuse large B-cell lymphoma (DLBCL) dataset from Rosenwald et al. (2002). This is a vector of survival times for 36 test samples. All survival times are given in years.

Usage

```
data(testSurv)
```

Format

The vector is called `testSurv`.

Source

Full dataset: http://llmpp.nih.gov/DLBCL/.

References

Rosenwald, A., Wright, G., Wing, C., Connors, J., Campo, E. et al. (2002). The Use of Molecular Profiling to Predict Survival After Chemotherapy for Diffuse Large-B-Cell Lymphoma. The New England Journal of Medicine, 346(25), 1937-1947.

`trainCens` *Sample Training Data for the Iterative BMA Algorithm for Survival Analysis*

Description

This is an diffuse large B-cell lymphoma (DLBCL) dataset from Rosenwald et al. (2002). This is a vector of censor information for 65 training samples. In this vector, 0 = censored and 1 = uncensored.

Usage

```
data(trainCens)
```

Format

The vector is called `trainCens`.

Source

Full dataset: http://llmpp.nih.gov/DLBCL/.

References

Rosenwald, A., Wright, G., Wing, C., Connors, J., Campo, E. et al. (2002). The Use of Molecular Profiling to Predict Survival After Chemotherapy for Diffuse Large-B-Cell Lymphoma. The New England Journal of Medicine, 346(25), 1937-1947.

trainData — *Sample Training Data for the Iterative BMA Algorithm for Survival Analysis*

Description

This is an adapted diffuse large B-cell lymphoma (DLBCL) dataset from Rosenwald et al. (2002). This data matrix consists of the expression levels from 65 DLBCL samples (rows), and 100 top univariate genes (columns). This dataset is used as a sample training set in our examples.

Usage

```
data(trainData)
```

Format

The data matrix is called `trainData`. Each entry in the matrix represents the expression level of one gene from a DLBCL sample.

Details

We started with the full expression data from Rosenwald et al. (2002), which is available along with corresponding patient information at their supplemental website http://llmpp.nih.gov/DLBCL/. We selected a subset of the 160 training samples, and then performed Cox Proportional Hazards Regression to obtain the 100 genes with the highest log likelihood. The filtered dataset consists of 65 training samples, and it is available at our website http://expression.washington.edu/ibmasurv/protected.

Source

Full dataset: http://llmpp.nih.gov/DLBCL/.

References

Rosenwald, A., Wright, G., Wing, C., Connors, J., Campo, E. et al. (2002). The Use of Molecular Profiling to Predict Survival After Chemotherapy for Diffuse Large-B-Cell Lymphoma. The New England Journal of Medicine, 346(25), 1937-1947.

trainSurv — *Sample Training Data for the Iterative BMA Algorithm for Survival Analysis*

Description

This is an diffuse large B-cell lymphoma (DLBCL) dataset from Rosenwald et al. (2002). This is a vector of survival times for 65 training samples. All survival times are given in years.

Usage

```
data(trainSurv)
```

Format

The vector is called `trainSurv`.

Source

Full dataset: http://llmpp.nih.gov/DLBCL/.

References

Rosenwald, A., Wright, G., Wing, C., Connors, J., Campo, E. et al. (2002). The Use of Molecular Profiling to Predict Survival After Chemotherapy for Diffuse Large-B-Cell Lymphoma. The New England Journal of Medicine, 346(25), 1937-1947.

Index

∗ **datasets**
testCens, 22
testData, 23
testSurv, 23
trainCens, 24
trainData, 25
trainSurv, 25
∗ **internal**
iterativeBMAsurv-internal, 15
∗ **multivariate**
iterateBMAsurv.train, 7
iterateBMAsurv.train.predict.assess, 10
iterateBMAsurv.train.wrapper, 13
iterativeBMAsurv-package, 2
∗ **print**
printTopGenes, 19
∗ **regression**
singleGeneCoxph, 21
∗ **survival**
crossVal, 4
imageplot.iterate.bma.surv, 6
iterateBMAsurv.train, 7
iterateBMAsurv.train.predict.assess, 10
iterateBMAsurv.train.wrapper, 13
iterativeBMAsurv-package, 2
predictBicSurv, 16
predictiveAssessCategory, 17
∗ **univar**
printTopGenes, 19
singleGeneCoxph, 21

assignRiskGroup *(iterativeBMAsurv-internal)*, 15

crossVal, *3*, 4
crossVal.final.calc *(iterativeBMAsurv-internal)*, 15
crossVal.fold *(iterativeBMAsurv-internal)*, 15
crossVal.init *(iterativeBMAsurv-internal)*, 15
crossVal.run *(iterativeBMAsurv-internal)*, 15
crossVal.tabulate *(iterativeBMAsurv-internal)*, 15

imageplot.bma.mod *(iterativeBMAsurv-internal)*, 15
imageplot.iterate.bma.surv, 6
iterateBMAinit *(iterativeBMAsurv-internal)*, 15
iterateBMAsurv.train, *3*, *5*, 7, *12*, *15*
iterateBMAsurv.train.predict.assess, *3*, *5*, *7*, *9*, 10, *15, 16*, *18*, *20*, *22*
iterateBMAsurv.train.wrapper, *3*, *5*, *7*, *9*, *12*, 13, *16*
iterativeBMAsurv *(iterativeBMAsurv-package)*, 2
iterativeBMAsurv-internal, 15
iterativeBMAsurv-package, 2

predictBicSurv, *5*, *9*, *12*, 16, *18*
predictiveAssessCategory, *3*, *5*, *12*, *15, 16*, 17
printTopGenes, *18*, 19, *22*

singleGeneCoxph, *3*, *5*, *9*, *12*, *15*, *18*, *20*, 21

testCens, *3*, *12*, *18*, 22
testData, *3*, *9*, *12*, *16*, *18*, 23
testSurv, *3*, *12*, *18*, 23
trainCens, *3*, *5*, *7*, *9*, *12*, *15, 16*, *18*, *20*, *22*, 24
trainData, *3*, *5*, *7*, *9*, *12*, *15, 16*, *18*, *20*, *22*, 25
trainSurv, *3*, *5*, *7*, *9*, *12*, *15, 16*, *18*, *20*, *22*, 25