Package ‘survJamda’

November 5, 2015

Type Package

Title Survival Prediction by Joint Analysis of Microarray Gene
Expression Data

Version 1.1.4

Date 2015-11-01

Author Haleh Yasrebi

Maintainer Haleh Yasrebi<hyasrebi@yahoo.com>

Description Microarray gene expression data can be analyzed individually or jointly using merging methods or meta-analysis to predict patients' survival and risk assessment.

Depends survival, survivalROC,ecodist,survcomp,survJamda.data

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2015-11-05 20:33:44

R topics documented:

survJamda-package *Survival Prediction by Joint Analysis of Microarray Gene Expression Data*

Description

Prediction of survival and risk assessment of patients using joint analysis of microarray gene expression data.

Details

Package:	survJamda
Type:	Package
Version:	1.1.4
Date:	2015-11-01
Depends:	survival, survivalROC, ecodist, survcomp, survJamda.data
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Haleh Yasrebi

Maintainer: Haleh Yasrebi <hyasrebi@yahoo.com>

References

Haleh Yasrebi, Comparative study of joint analysis of microarray gene expression data in survival prediction and risk assessment of breast cancer patients, Brief Bioinform, 2015 doi:10.1093/bib/bbv092, PMID:26504096.

Yasrebi H. SurvJamda: an R package to predict patients' survival and risk assessment using joint analysis of microarray gene expression data. Bioinformatics. 2011 Apr 15;27(8):1168-9. doi: 10.1093/bioinformatics/btr103. Epub 2011 Mar 2. PubMed PMID: 21367873.

Yasrebi H., Sperisen P., Praz V., Bucher P., Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?, PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431, 2009, PMID: 21367873.

Yasrebi, H. Prediction of survival and risk assessment using joint analysis of microarray gene expression data, PhD thesis, EPFL, Switzerland, doi:10.5075/Thesis 4494, 2010, PMID: 19851466.

See Also

`coxph`, `survivalROC`, `corgen`, `p.adjust`, `concordance.index`, `sbrier.score2proba`

`aprior` *Calculate empirical hyper-prior values*

Description

Calculate empirical hyper-prior values

Usage

```
aprior(gamma.hat)
```

Arguments

`gamma.hat`	Estimate of additive batch effect.

Value

Empirical hyper-prior values of Bayesian model.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: `http://dx.doi.org/10.1093/biostatistics/kxj037`.

See Also

`ComBat`,`bprior`

Beta.NA *Fit the L/S model in the presence of missing data values*

Description

Fit the L/S model in the presence of missing data values

Usage

```
Beta.NA(y, X)
```

Arguments

y	Product of design matrix and matrix of gene expression data.
X	Matrix of gene expression data.

Value

Vector of Regression coefficients in L/S model fitting.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: http://dx.doi.org/10.1093/biostatistics/kxj037.

See Also

ComBat

bprior — *Calculate empirical hyper-prior values of Bayesian model*

Description

Calculate empirical hyper-prior values of Bayesian model

Usage

```
bprior(gamma.hat)
```

Arguments

`gamma.hat`	Estimate of additive batch effect

Value

Empirical hyper-prior values of Bayesian model.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: `http://dx.doi.org/10.1093/biostatistics/kxj037`.

See Also

`ComBat`,`aprior`

`build.design` *Initiation to build the design matrix*

Description

Initiation to build the design matrix.

Usage

```
build.design(vec, des = NULL, start = 2)
```

Arguments

`vec`	Vector of batches in the sample info matrix.
`des`	Initial value of design matrix
`start`	Starting index of design matrix.

Value

Design matrix

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644.

`cal.cox.coef` *Cox coefficient calculation.*

Description

Calculate the Cox coefficients of covariates.

Usage

```
cal.cox.coef (gnExpMat, survivaltime, censor)
```

Arguments

`gnExpMat` Matrix of gene expression data.

`survivaltime` Vector of survival time.

`censor` Vector of censoring status. 1 = event occurred, 0 = censored.

Value

Vector of Cox coefficients.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

`calPerformance.auc.plot`

Assess the performance obtained from the merged data set by independent validation

---

Description

Identify a gene signature and reduce the gene set in the training and testing sets accordingly.

Arguments

`lst` List of two objects, the gene expression data matrix and a list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`train.ind` Training set index.

`test.ind` Testing set index.

`file.name` The name of the expression file used as the testing set.

`col` Color of ROC curve.

`method` A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function.

`normalization` The normalization method, Z-score2, Z-score1 or ComBat.

`time.dep` An integer 0 or 1, 1 to plot time-dependent ROC curves for different time points and 0 for no plot

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`calPerformance.merge.indep` *Assess performance derived from the merged data set by independent validation*

Description

Identify a gene signature from the merged data set and reduce of the gene set in the training and testing sets accordingly. The performance of the gene signature is performed by independent validation.

Arguments

`lst`	List of two objects, the gene expression data matrix and a list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`train.ind`	Index of training set.
`test.ind`	Index of testing set.
`method`	A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.
`gn.nb`	Number of genes to select for gene signature when `method`="none".
`perf.eval`	A string taking one the values, "auc", "cindex", "bsc".
`normalization`	A character string specifying the normalization method, "zscore1", "zscore2" or "combat".

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If `perf.eval` == "auc", time-dependent AUC and hazard ratio are used as the measure of performance, `perf.eval` == "cindex", concordance index defined in the `survcomp` package or `perf.eval` == "bsc", brier score defined in the `survcomp` package is used.

Value

AUC, HR(CI) and p-value.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

`calPerformance.meta` *Meta analysis of survival data*

Description

Analyze jointly the data set by the inverse normal method (Hedges and Olkin, 1985).

Usage

```
calPerformance.meta(common.gene, zstat, i, j, geno.files, surv.data, method)
```

Arguments

`common.gene`	A vector of character strings containing the names of the genes common to all data sets.
`zstat`	A list containing the combined Z-scores of the data sets composing the training set.

`i` A vector of character strings containing the names of the data sets composing the training set.

`j` A character string specifying the name the data set used as the testing set.

`geno.files` A vector of character strings containing the names of gene expression files.

`surv.data` A list of two vectors, survival time and censoring status.In the censoring status vector, 1 = event occurred, 0 = censored.

`method` A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function.

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-value not greater than 0.05 are retained if `method` != "none".

Value

AUC, HR(CI) and p-value.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

L. V. Hedges and I. Olkin. Statistical Methods for Meta-Analysis. Academic Press,January 1985. ISBN 0123363802.

`calPerformance.single.indep` *Performance assessment on single data sets using independent validation*

Description

Assess the performance of the gene signatures on single data sets in pair-wise manner.

Usage

```
calPerformance.single.indep(lst1, lst2, method,gn.nb,perf.eval)
```

Arguments

`lst1` A list of two objects, (i) the gene expression data and (ii) the list of survival time and censoring status of the data set used as the training set. In the censoring status vector, 1 = event occurred, 0 = censored.

`lst2` A list of two objects, (i) the gene expression data and (ii) the list of survival time and censoring status of an independent data set used as the testing set. In the censoring status vector, 1 = event occurred, 0 = censored.

`method` A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.

`gn.nb` Number of genes to select for gene signature when `method`="none".

`perf.eval` A string taking one the values, "auc", "cindex", "bsc".

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-`gn.nb` ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If `perf.eval` == "auc", time-dependent AUC and hazard ratio are used as the measure of performance, `perf.eval` == "cindex", concordance index defined in the `survcomp` package or `perf.eval` == "bsc", brier score defined in the `survcomp` package is used.

Value

AUC, HR(CI) and p-value.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

ci.gm *Confidence interval of a Geometric mean*

Description

Calculate the Confidence Interval (CI) of a geometric mean.

Usage

```
ci.gm(x)
```

Arguments

x	Vector of numeric values. For example, a vector of HRs.

Value

Vector of the CI of a geometric mean.

Author(s)

Haleh Yasrebi

See Also

gm

Examples

```
v = c(1.5,2.5,7,4)
ci.gm(v)

## The function is currently defined as
function(x){
          gm1 = mean(log(x), na.rm = T)
          cil = exp(gm1-(1.96*(sd(log(x), na.rm = T)/sqrt(length(x)))))
          ciupp = exp(gm1+(1.96*(sd(log(x), na.rm = T)/sqrt(length(x)))))
          vec = c(round(cil,2), round(ciupp,2))
          return (vec)
}
```

comb.surv.censor *Merge survival times and censoring status.*

Description

Merge vectors of survival time and censoring status of different data sets for joint analysis.

Usage

```
comb.surv.censor(geno.files,index,surv.data)
```

Arguments

`geno.files`	Vector of character strings containing the names of expression files.
`index`	Index of the data files in `geno.files` to be combined.
`surv.data`	List of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

Value

A list of two vectors, combined survival time and censoring status.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

ComBat *ComBat-adjusted microarray gene expression data*

Description

Compute ComBat-adjusted microarray gene expression data.

Usage

```
ComBat(expression_xls, sample_info_file, type = "txt", write = TRUE,
covariates = "all", par.prior = TRUE, filter = FALSE, skip = 0,
prior.plots = TRUE)
```

Arguments

`expression_xls` A character string specifying gene expression file.

`sample_info_file`
A character string specifying sample file.

`type` A character string specifying the type of the file, "txt" or "csv".

`write` A Boolean variable indicating whether the output (adjusted data) should be written into a file.

`covariates` A vector of integers or "all" if all covariates should be used. `covariates=all` will use all of the columns in your sample info file in the modeling (except array/sample name), if you only want use a some of the columns in your sample info file, specify these columns here as a vector (you must include the Batch column in this list).

`par.prior` A Boolean character indicating whether the parametric adjustment should be applied.

`filter` A Boolean variable indicating whether presence/absence call is used in the gene expression file.

`skip` An integer value indicating the number of columns that contain the gene names. skip = 1 implies the first expression values start from column 2.

`prior.plots` A Boolean variable indicating whether the prior plots should be given where black is a kernel density estimate of the batch effects. Quantile-quantile plots are also included. If the red and black lines do not match up well, use the nonparametric adjustment.

Value

Matrix of adjusted expression data.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644.

`combat.likelihood` *Likelihood function.*

Description

Likelihood function.

Usage

```
combat.likelihood(x, g.hat, d.hat)
```

Arguments

`x`	Matrix of gene expression data.
`g.hat`	Estimated additive batch effect.
`d.hat`	Estimated multiplicative batch effect.

Value

Likelihood estimate.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: `http://dx.doi.org/10.1093/biostatistics/kxj037`.

See Also

`ComBat`

compute.combat *Initiate ComBat adjustment*

Description

Call ComBat function for ComBat-adjustment of microarray gene expression data.

Usage

```
compute.combat(fileGeno, fileSample)
```

Arguments

fileGeno	A character string specifying the name of the gene expression file. Genes are in columns and samples are in rows. Column names should contain the gene names.
fileSample	A character string specifying the name of the file containing the sample or array names of gene expression data and batch ID.

Value

ComBat-adjusted gene expression data matrix. Genes are organized in rows and samples are organized in columns.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

cross.val.combat *Cross validation with ComBat adjustment*

Description

Assess the performance of the gene signatures derived from the merged data set adjusted by ComBat in cross-validation.

Usage

```
cross.val.combat(x, y, censor, batchID, method, gn.nb, plot.roc, ngroup, iter)
```

Arguments

`x` Matrix of gene expression data.

`y` Vector of survival time.

`censor` Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`batchID` Vector containing the batch ID of the data set `x`. The batch ID of the data sets composing the matrix `x` should be in the same order of the component data sets. For a given data set, the batch id can be an integer or the name of the data set. The batch id must be the same for all samples or arrays of a data set.

`method` A character string specifying the feature selection method: "none" for top-ranking (top-100 ranking by default) or one of the adjusting methods specified by the p.adjust function.

`gn.nb` An integer variable specifying the number of genes to select. The default is 100.

`plot.roc` An integer specifying whether the ROC curves should be plotted or not (1 or 0).

`ngroup` An integer variable specifying the number of cross-validation folds. The default is 10.

`iter` An integer variable specifying the current number of iteration.

Details

If the user wants to apply his own feature selection method, he should define his function with the same number of parameters as the defined feature selection function of the package, i.e. `featureselection`.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

ROC curves are the plots of the mean of true positives (sensitivity) and the mean of false positives (1-specificity) over `ngroup` folds of cross-validation.

Value

Arithmetic mean of AUC +/- standard deviation and geometric mean of HR(CI) generated from cross-validation.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

See Also

`iter.crossval.combat`

`cross.val.surv` *Cross validation with or without Z-score normalization*

Description

Assess the performance of the gene signatures derived from a single or merged data set by cross-validation.

Usage

```
cross.val.surv(x, y, censor, ngroup, iter, method, zscore, gn.nb,
gn.nb.display, plot.roc)
```

Arguments

`x` Matrix of gene expression data.

`y` Vector of survival time.

`censor` Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`ngroup` An integer specifying the number of cross-validation folds. The default is 10.

`iter` An integer specifying the current number of iteration.

`method` A character string specifying the feature selection method: "none" for top-ranking (top-100 ranking by default) or one of the adjusting methods specified by the p.adjust function.

`zscore` An integer specifying whether Z-score normalization should be applied or not (1 or 0). 1 if the `data` is a merged data set and 0 if `data` is a single data set.

`gn.nb` An integer specifying the number of genes to select. The default is 100.

`gn.nb.display` An integer specifying the number of selected genes to display.

`plot.roc` An integer specifying whether the ROC curves should be plotted or not (1 or 0).

Details

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If the user wants to apply his own feature selection method, he should define his function with the same number of parameters as the defined feature selection function of the package, i.e. `featureselection`.

ROC curves are the plots of the mean of true positives (sensitivity) and the mean of false positives (1-specificity) over `ngroup` folds of cross-validation.

Value

AUC and HR generated from cross-validation.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

`design.mat` *Build a design matrix*

Description

Build a design matrix for ComBat adjustment.

Usage

```
design.mat(saminfo)
```

Arguments

`saminfo`	Matrix of sample information.

Value

Design matrix.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: `http://dx.doi.org/10.1093/biostatistics/kxj037`.

det.batchID *Determine the batch ID of data sets.*

Description

Determine the batch ID of data sets for ComBat.

Usage

```
det.batchID(geno.files)
```

Arguments

`geno.files`	A vector of character containing the names of gene expression data files.

Value

A vector of integers specifying the batch ID of data sets. The integers from 1 to the number specifying the length of `geno.files` are set as the batch ID of the data sets named in `geno.files` as follows: 1 to the first expression file name in `geno.files`, 2 to the second expression file name in `geno.files`, ... and the integer specifying the length of `geno.files` to the last expression file in `geno.files`, respectively.

Author(s)

Haleh Yasrebi

det.set.ind *Determine the indices of the training or testing set.*

Description

Determine the indices of the training or testing set.

Usage

```
det.set.ind(geno.files,train, i)
```

Arguments

`geno.files`	A vector of character containing the names of gene expression data files.
`train`	Integer variable specifying whether the returned indices are for the training set or testing set (1 or 0).
`i`	Integer variable specifying the indices of the file in `geno.files`.

Value

A vector containing the indices of the required set.

Author(s)

Haleh Yasrebi

`det.set.meta` *Split data for meta analysis.*

Description

Split data into the training and testing sets for meta analysis.

Usage

```
det.set.meta(i, j, geno.files, surv.data, method)
```

Arguments

`i`	A vector of character strings consisting of the names of the expression files used for the training set.
`j`	A character string specifying the name of the expression file used for the testing set.
`geno.files`	A vector of character strings consisting of the names of the expression files.
`surv.data`	A list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`method`	A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

Value

None.

Author(s)

Haleh Yasrebi

`detFileName` *Determine the name of a file.*

Description

Determine the name of a file.

Usage

```
detFileName(file.name)
```

Arguments

`file.name`	A character string specifying the name of the expression file to display.

Value

`file.name` in upper case (in the case of a "gse" file) or without any change.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`eval.merge.simulate` *Performance evaluation by merging two simulated independent data sets*

Description

Simulate two data sets, merge them and evaluate the performance of the gene signature derived from the merged data set in 10 iterations of 10-fold cross-validation. The data sets are combined into one set, split into the training and testing sets which are then normalized by Z-score normalization.

Usage

```
eval.merge.simulate(d1, d2, tot.genes, gene.nb, zscore)
```

Arguments

`d1` Matrix of gene expression data of the first simulated data sets.

`d2` Matrix of gene expression data of the second simulated data set.

`tot.genes` Number of total genes.

`gene.nb` Number of true survival genes to identify.

`zscore` An integer (1 or 0) specifying whether to apply Z-score normalization or not.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

See Also

`proc.simulate`

`eval.subset` *Performance evaluation derived from a subset of a data set*

Description

Select a subset of a single data set and split it into the training and testing sets. Generate a gene signature from the training set and evaluate its performance on the testing set.

Usage

```
eval.subset(x, y, censor, iter, method, gn.nb, train.nb)
```

Arguments

`x` Matrix of gene expression data.

`y` Vector of survival time.

`censor` Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`iter` An integer specifying the current iteration.

`method` A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.

`gn.nb` An integer specifying the number of genes to select.

`train.nb` An integer specifying the sample size of the training set.

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes were ranked based on their likelihood ratio P-value and the top-`gn.nb` ranked genes with the smallest P-values were retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

Value

AUC and HR.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`excl.missing` *Exclude missing samples*

Description

Exclude samples with missing survival times.

Usage

```
excl.missing(mat,phyno)
```

Arguments

`mat`	Matrix of gene expression data.
`phyno`	A list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

Value

A list of two objects; (i) matrix of gene expression data of the patients with no missing survival times. (ii) the list of two vectors, survival time and censoring status of patients with no missing survival time time points.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

---

excl.missing.single.indep *Exclude missing samples prior to independent validation*

Description

Exclude samples with missing survival time points prior to the application of independent validation to single data sets.

Usage

```
excl.missing.single.indep(geno.files,ind,surv.data, common.gene)
```

Arguments

`geno.files`	A vector of character strings containing the names of the expression files.
`ind`	Index of expression files in `geno.files` to combine.
`surv.data`	The list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`common.gene`	A vector of character strings containing the names of the genes common to all data sets.

Value

A list of two objects, (i) Matrix of gene expression data of the patients with no missing survival time. (ii) The list of two vectors, survival time and censoring status of the patients with no missing survival time points.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`excl.samples` *Exclude samples*

Description

Time-dependent ROC curves could be plotted based on different survival time points at which an event has occurred. To this end, censored patients should be excluded so that the plot could be based on the time points at which an event has occurred.

Usage

```
excl.samples(test.ind, surv, censor)
```

Arguments

`test.ind`	Vector of testing set index.
`surv`	Vector of survival time.
`censor`	Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

Value

Index of survival time points of the patients in the testing set who have experienced an event.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`featureselection` *Apply a feature selection*

Description

Apply univariate Cox regression and rank the genes based on the Cox p-value.

Usage

```
featureselection(gnExpMat, survivaltime, censor, method = "none",gn.nb)
```

Arguments

`gnExpMat`	Matrix of gene expression data.
`survivaltime`	Vector of survival time.
`censor`	Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`method`	A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.
`gn.nb`	Number of genes to select for gene signature when `method`="none".

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-`gn.nb` ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

Value

A list of two vectors, the Cox coefficients and Cox p-values.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

`featureselection.meta` *Feature selection for meta analysis*

Description

Apply univariate Cox regression and aggregate gene Z-scores.

Usage

```
featureselection.meta(gnExpMat, survivaltime, censor)
```

Arguments

`gnExpMat` Matrix of gene expression data.

`survivaltime` Vector of survival time.

`censor` Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

Value

Vector of gene Z-scores.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`filter.absent` *Filter absent calls*

Description

Filter data based on presence/absence call.

Usage

```
filter.absent(x, pct)
```

Arguments

`x` Matrix of gene expression data.

`pct` Number of presence calls.

Value

A Boolean variable specifying the presence or absence call.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: http://dx.doi.org/10.1093/biostatistics/kxj037.

See Also

ComBat

---

generate.survival.data

Generate survival data.

Description

Generate survival data following a Weibull model with specified parameters such as Cox coefficients and correlation among genes. Then, identify a gene signature and assess its performance in cross-validation.

Usage

```
generate.survival.data(gene.nb, tot.genes, sample.nb, beta.init,
correlation, shape, scale)
```

Arguments

gene.nb	The number of genes to select.
tot.genes	The total number of genes.
sample.nb	The total number of samples.
beta.init	Initial values for beta or Cox coefficients. The values between +/-0.5 to +/-3 are good choices.
correlation	Correlation among genes. The value should be between 0 and 1.
shape	Shape parameter of the Weibull model. Select a value between 1 and 5.
scale	Scale parameter of the Weibull model. Select a value between 1 and 5.

Value

A list of three objects, (i) Matrix of simulated gene expression data, (ii) Vector of survival time and (iii) Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

See Also

`proc.simulate`

`gm` *Geometric Mean*

Description

Calculates the geometric mean of a vector.

Usage

```
gm(x)
```

Arguments

`x`	Vector of numeric values.

Value

Geometric mean of `x`.

Author(s)

Haleh Yasrebi

See Also

`ci.gm`

groups.cv *Split a data set for cross-validation*

Description

Define the folds of cross-validation.

Usage

```
groups.cv(n, ngroup, censor)
```

Arguments

`n`	Sample size of a data set.
`ngroup`	Number of folds of cross-validation.
`censor`	Vector of censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

Details

The function avoids allocating only censored patients to the testing set. At least one patient having experienced an event is needed for the applicability of the Cox proportional hazard model.

Value

The folds of cross-validation.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

init.plot *Start plotting*

Description

Plot the coordinates with the main title and axis labels.

Usage

```
init.plot(file.name)
```

Arguments

`file.name`	A character string specifying the name of the expression file to display.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`int.eprior` *Integration function to find nonparametric adjustments*

Description

Monte Carlo integration function to find the nonparametric adjustments

Usage

```
int.eprior(sdat, g.hat, d.hat)
```

Arguments

`sdat`	Matrix of standardized gene expression data.
`g.hat`	Estimated additive batch effect.
`d.hat`	Estimated multiplicative batch effect.

Value

Matrix with two columns containing the estimated additive and multiplicative batch effects.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: http://dx.doi.org/10.1093/biostatistics/kxj037.

See Also

ComBat

inv.normal *Apply the inverse normal method.*

Description

Apply the inverse normal method (Hedges and Olkin, 1985). For each data set, combine the Z-scores or Z-tests and divide them by the square root of the sample size of the data set.

Usage

```
inv.normal(i, zstat)
```

Arguments

i	Vector of integer variables containing the indices of data sets.
zstat	List of numeric vectors representing Z-tests or Z-scores.

Value

Vector of combined Z-tests.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

L. V. Hedges and I. Olkin. Statistical Methods for Meta-Analysis. Academic Press,January 1985. ISBN 0123363802.

`it.sol` *Iterative solution for Empirical Bayesian method.*

Description

Iterative solution for Empirical Bayesian method.

Usage

```
it.sol(sdat, g.hat, d.hat, g.bar, t2, a, b, conv = 1e-04)
```

Arguments

`sdat` Standardized matrix of gene expression data.

`g.hat` Estimated additive batch effect.

`d.hat` Estimated multiplicative batch effect.

`g.bar` Mean of `g.hat`.

`t2` Variance of the rows of `g.hat`.

`a` Value of the function `aprior` applied to multiplicative batch effect.

`b` Value of the function `bprior` applied to multiplicative batch effect.

`conv` Convergence threshold.

Value

Matrix of estimated additive and multiplicative batch effects.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson.

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, Jan ary2007. ISSN 1465-4644. doi: `http://dx.doi.org/10.1093/biostatistics/kxj037`.

See Also

`ComBat`,`aprior`,`bprior`

iter.crossval *Performance assessment of gene signatures by cross-validation.*

Description

Assess the performance of a gene signature derived from a single or merged data set by ten iterations of cross validation.

Usage

```
iter.crossval(data, surv, censor, ngroup = 10, plot.roc = 0, method = "none",
zscore = 0, gn.nb = 100, gn.nb.display = 0)
```

Arguments

`data`	Matrix of gene expression data.
`surv`	Vector of survival time.
`censor`	Vector of censoring status. 1 = event occurred, 0 = censored.
`ngroup`	An integer specifying the number of cross-validation folds.
`plot.roc`	An integer specifying whether the ROC curves should be plotted or not (1 or 0).
`method`	A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function
`zscore`	An integer specifying whether Z-score normalization should be applied or not (1 or 0). 1 if `data` is a merged data set or 0 if `data` is a single data set.
`gn.nb`	An integer specifying the number of genes to select for gene signature.
`gn.nb.display`	An integer specifying the number of selected genes to display.

Details

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If the user wants to apply his own feature selection method, he should define his function with the same number of parameters as the defined feature selection function of the package, i.e. `featureselection`.

ROC curves are the plots of the mean of true positives (sensitivity) and the mean of false positives (1-specificity) over `ngroup` folds of cross-validation.

Value

Mean of AUC +/- standard deviation of AUC, geometric mean of HR(CI).

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

See Also

`iter.crossval.combat`,

Examples

```
## Single data set
data(gse4335)
data(gse4335pheno)

#And run the following script:
#iter.crossval(gse4335, gse4335pheno[,6], gse4335pheno[,5])

## To observe the frequency of the CYB5D1 gene selection, run the following script:
#iter.crossval(gse4335, gse4335pheno[,6], gse4335pheno[,5], gn.nb =1, gn.nb.display = 1)

## Merged data set
data(gse4335)
data(gse4335pheno)

data(gse1992)
data(gse1992pheno)

common.gene = intersect(colnames(gse4335), colnames(gse1992))

data = rbind(gse4335[,common.gene], gse1992[,common.gene])
surv = c(gse4335pheno[,6],gse1992pheno[,19])
censor = c(gse4335pheno[,5],gse1992pheno[,18])

#And run the following script:
#iter.crossval(data, surv,censor, zscore=1)

## The function is currently defined as
function(data,surv,censor,ngroup=10,plot.roc=0,method=”none”,zscore=0,gn.nb=100,gn.nb.display=0){

         require(survival)
         require(survivalROC)

         res = NULL

         file.name=deparse(substitute(data))
         if (plot.roc)
         init.plot(file.name)

         data =data[!is.na(surv),]
         censor= censor[!is.na(surv)]
         surv= surv[!is.na(surv)]
```

```
            cat ("Iteration\tAUC\tHR(CI)\t\tP-val\n")
            for (i in 1:ngroup){
                    new.lst = cross.val.surv(data, surv, censor,ngroup, i, method, zscore,
                    gn.nb,gn.nb.display,plot.roc,p.list)
                    res = rbind (res, new.lst)
            }

            if(ngroup != length(surv)){
                    cat ("Avg AUC+/-SD\tHR(CI)\n")
                    if (plot.roc)
                            legend (0.55,0.1, legend = paste("AUC+/-SD =", sprintf("%.2f",
                             as.numeric(mean(res[,1],na.rm = T))), "+/-", sprintf("%.2f",
                             sd (res[,1],na.rm = T)), sep = " "),bty = "n")
                    cat (sprintf("%.2f",as.numeric(mean(res[,1], na.rm = T))),  "+/-",
                    sprintf("%.2f",sd (res[,1],na.rm = T)), "\t", gm(res[,2]),
                    "(", sprintf("%.2f",ci.gm(res[,2])[1]), "-", sprintf("%.2f",
                    ci.gm(res[,2])[2]), ")\n", sep = "")
            }
    }
```

iter.crossval.combat *Merge data set by ComBat within cross-validation.*

Description

Assess the performance of the gene signatures derived from the merged data set adjusted by ComBat by ten iterations of cross-validation.

Usage

```
iter.crossval.combat(data, surv, censor, batchID, ngroup = 10, plot.roc = 0,
 method = "none", gn.nb = 100)
```

Arguments

data	Matrix of gene expression data.
surv	Vector of survival times.
censor	Vector of censoring status. 1 = event occurred, 0 = censored.
batchID	For a given data set, the batch id can be an integer or the name of the data set. The batch id must be the same for all samples or arrays of a data set.
ngroup	An integer specifying the number of cross-validation folds.
plot.roc	A integer (0 or 1) indicating whether the ROC curves should be plotted.
method	A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.
gn.nb	An integer specifying the number of genes to select.

Details

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If the user wants to apply his own feature selection method, he should define his function with the same number of parameters as the defined feature selection function of the package, i.e. `featureselection`.

ROC curves are the plots of the mean of true positives (sensitivity) and the mean of false positives (1-specificity) over `ngroup` folds of cross-validation.

Value

Arithmetic mean of AUC +/- standard deviation (AUC) and geometric mean of HR(CI).

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

See Also

`iter.crossval`

Examples

```
require(survJamda.data)

data(gse4335)
data(gse4335pheno)

data(gse1992)
data(gse1992pheno)

common.gene = intersect(colnames(gse4335), colnames(gse1992))

data = rbind(gse4335[,common.gene], gse1992[,common.gene])
surv = c(gse4335pheno[,6],gse1992pheno[,19])
censor = c(gse4335pheno[,5],gse1992pheno[,18])

# An integer is used as batchID
batchID = rep(1,nrow(gse4335))
batchID = c(batchID,rep(2,nrow(gse1992)))

#Or the name of the data sets is used as batch ID
#batchID = rep("gse4335",nrow(gse4335))
#batchID = c(batchID,rep("gse1992",nrow(gse1992)))

#And run the following script:
```

```
#iter.crossval.combat(data, surv,censor, batchID)

## The function is currently defined as
function (data, surv, censor, batchID, ngroup=10, plot.roc = 0, method = "none",
gn.nb = 100){

        require(survival)
        require(survivalROC)

        if(!exists("batchID"))
                stop("\rSet batchID", call.=FALSE)

        niter = ifelse(ngroup == length(surv), 1,10)
        res = NULL

        file.name=deparse(substitute(data))
        if (plot.roc)
                init.plot(file.name)

        data =data[!is.na(surv),]
        censor= censor[!is.na(surv)]
        surv= surv[!is.na(surv)]

        cat ("Iteration\tAUC\tHR(CI)\t\tP-val\n")
        for (i in 1:niter){
                new.lst = cross.val.combat(data, surv, censor,method = "none",
                gn.nb, plot.roc, ngroup, i)
                res = rbind (res, new.lst)
        }

        if(ngroup != length(surv)){
                cat ("Avg AUC+/-SD\tHR(CI)\n")
                if (plot.roc)
                legend (0.55,0.1, legend = paste("AUC+/-SD =", sprintf("%.2f",
                as.numeric(mean(res[,1],na.rm = TRUE))), "+/-", sprintf("%.2f",
                sd (res[,1],na.rm = TRUE)), sep = " "), bty = "n")

                cat (sprintf("%.2f",as.numeric(mean(res[,1], na.rm = TRUE))),
                "+/-", sprintf("%.2f",sd (res[,1],na.rm = TRUE)), "\t",
                gm(res[,2]), "(", sprintf("%.2f",ci.gm(res[,2])[1]), "-",
                sprintf("%.2f",ci.gm(res[,2])[2]), ")\n", sep = "")
        }
}
```

iter.subset *Performance evaluation by subsetting data sets in 100 iterations*

Description

A data set can be split to different subsets to determine if the performance derived from its subsets is improved by the increase of sample size. Each subset can then be split 100 times into the independent training and testing sets. The sample size of the training set is set by the user (20,50,...,up to 2/3 of the complete set) and the remaining samples are used for the testing set. A gene signature will be derived from the training set and assessed on the testing set.

The performance obtained from the larger subsets and ultimately, the complete set is more likely higher than the performance generated from the smaller subsets. If it is not the case, the performance improvement might have been retained by factors such as heterogeneity with respect to patient's cohort or tumor characteristics.

Usage

```
iter.subset(data, surv, censor, method = "none", gn.nb = 50, train.nb = 100)
```

Arguments

`data` Matrix of gene expression data.

`surv` Vector of survival times.

`censor` Vector of censoring status. 1 = event occurred, 0 = censored.

`method` A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.

`gn.nb` An integer specifying the number of genes to select.

`train.nb` An integer specifying the sample size of the training set.

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-`gn.nb` ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

Value

Mean of AUC +/- standard deviation of AUC, geometric mean of HR (CI).

Author(s)

Haleh Yasrebi

Examples

```
data(gse4335)
data(gse4335pheno)
#The following script might be lengthy
#iter.subset(gse4335, gse4335pheno[,6],gse4335pheno[,5])
## The function is currently defined as
function (data, surv, censor,method = "none", gn.nb = 50, train.nb = 100){
        require (survival)
        require (survivalROC)

        data =data[!is.na(surv),]
        censor= censor[!is.na(surv)]
        surv= surv[!is.na(surv)]

        res = NULL
        iteration.nb = 100

        cat ("Iteration\tAUC\tHR(CI)\t\tP-val\n")

        for (i in 1:iteration.nb){
                new.lst = eval.subset(data, surv, censor,i, method, gn.nb, train.nb)
                res = rbind (res, new.lst)
        }

        cat ("Avg AUC+/-SD\tHR(CI)\n")

        cat (sprintf("%.2f",mean(res[,1], na.rm = T)),  "+/-",
        sprintf("%.2f",sd (res[,1],na.rm = T)), "\t",
        sprintf("%.2f",gm(res[,2])), "(",
        sprintf("%.2f",ci.gm(res[,2])[1]), "-",
        sprintf("%.2f",ci.gm(res[,2])[2]), ")\n",
        sep = "")
}
```

L *Likelihood function.*

Description

Likelihood function.

Usage

```
L(x, g.hat, d.hat)
```

Arguments

`x`	Matrix of gene expression data.
`g.hat`	Estimated additive batch effect.
`d.hat`	Estimated multiplicative batch effect.

Value

Likelihood estimate.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: http://dx.doi.org/10.1093/biostatistics/kxj037.

`list.batch` *Make a list of data batches.*

Description

Append the batches of data sets.

Usage

```
list.batch(saminfo)
```

Arguments

`saminfo`	Matrix of sample information.

Value

List of batches.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January 2007. ISSN 1465-4644.

See Also

`ComBat`

main.merge.indep.valid *Performance assessment of merged data sets by independent validation*

Description

Assess the performance of survival prediction derived from the merged data sets by independent validation.

Usage

```
main.merge.indep.valid(geno.files,surv.data,gn.nb=100,method = "none",
 normalization = "zscore1", perf.eval = "auc")
```

Arguments

`geno.files`	Vector of character strings containing the names of gene expression files.
`surv.data`	List of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`gn.nb`	Number of genes to select for gene signature when `method`="none".
`method`	A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.
`normalization`	A character string, "combat", "zscore1" or "zscore2".
`perf.eval`	A string taking one the values, "auc", "cindex", "bsc".

Details

In Z-score1 normalization, all data sets are Z-score normalized separately and then, the data sets composing the training set are merged together. The remaining set is used as the testing set. This process is continued S times (S being the number of data sets) until all data sets are used in the training and testing sets.

In Z-score2 normalization, the data sets are selected for the training and testing sets. Suppose there are S data sets. Then, in S iteration, S-1 data sets are used for the training set and the remaining set used as the testing set. This process is continued until all data sets are used in the training and testing sets. In each iteration, the data sets composing the training set are first merged together and the merged data set is then Z-score normalized. The testing set is independently adjusted by Z-score normalization.

If the user wants to apply his own feature selection method, he should define his function with the same number of parameters as the defined feature selection function of the package, i.e. `featureselection`.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If `perf.eval` == "auc", time-dependent AUC and hazard ratio are used as the measure of performance, `perf.eval` == "cindex", concordance index defined in the `survcomp` package or `perf.eval` == "bsc", brier score defined in the `survcomp` package is used.

Value

AUC, HR(CI) and p-value.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

Examples

```
require(survJamda.data)

data(gse4335)
data(gse3143)
data(gse1992)

data(gse4335pheno)
data(gse3143pheno)
data(gse1992pheno)

geno.files = c("gse4335", "gse3143","gse1992")
surv.data = list(c(gse4335pheno[,6],gse3143pheno[,4],gse1992pheno[,19]),
                 c(gse4335pheno[,5],gse3143pheno[,3],gse1992pheno[,18]))
#The following script might  take some time
#main.merge.indep.valid(geno.files,surv.data)

function(geno.files,surv.data,method = "none", normalization= "zscore1", perf.eval = "auc")
{
        require(survival)
        require(survivalROC)

        if (length(geno.files) < 3)
                stop ("\rThere should be minimum 3 data sets", call. = FALSE)

        if(normalization == "combat")
                batchID = det.batchID()

        if (!is.element(normalization, c("zscore","combat")))
                stop("\rnormalization = \"zscore\" or normalization = \"combat\"",
                call.=FALSE)

       common.gene = colnames(get(geno.files[1]))
       for (i in 2:length(geno.files))
              common.gene = intersect(common.gene, colnames(get(geno.files[i])))

       curr_set = 1:length(geno.files)

       for (y in curr_set){
            x = setdiff(curr_set, y)
            prep = get(paste("prep",normalization, sep = ""))
            lst = prep(common.gene,geno.files,surv.data,x,y)

            if (normalization == "zscore1" || normalization == "combat")
                  splitMerged.indep (geno.files,lst, x, y, method, perf.eval)
            else
                  splitZscore2.merge.indep (common.gene,geno.files,surv.data,
                  lst, x, y, method, perf.eval)
      }
}
```

`main.process` *main.process*

Description

Plot time-dependent ROC curves based on different time points.

Usage

```
main.process(common.gene, geno.files, surv.data, method = "none", time.dep)
```

Arguments

`common.gene` Vector of character strings containing the names of the genes common to all data sets.

`geno.files` Vector of character strings containing the names of expression files.

`surv.data` The list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`method` A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function

`time.dep` An integer 0 or 1, 1 to plot time-dependent ROC curves for different time points and 0 for no plot

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

Value

ROC curves plot and AUC values on the plot.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

---

main.single.indep.valid

Independent validation of the performance of the gene signatures derived from single data sets.

Description

Assess the performance of the gene signatures derived from the single data sets in pair-wise manner.

Usage

```
main.single.indep.valid(geno.files, surv.data,normalization = "zscore",
 method = "none",gn.nb=100, perf.eval = "auc")
```

Arguments

`geno.files`	A vector of character strings containing the names of expression files.
`surv.data`	The list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`normalization`	A character string, "combat" or "zscore".
`method`	A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.
`gn.nb`	Number of genes to select for gene signature when `method`="none".
`perf.eval`	A string taking one the values, "auc", "cindex", "bsc".

Details

If the user wants to apply his own feature selection method, he should define his function with the same number of parameters as the defined feature selection function of the package, i.e. `featureselection`.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If `perf.eval` == "auc", time-dependent AUC and hazard ratio are used as the measure of performance, `perf.eval` == "cindex", concordance index defined in the `survcomp` package or `perf.eval` == "bsc", brier score defined in the `survcomp` package is used.

Value

AUC, HR(CI) and p-value.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

See Also

znorm, ComBat

Examples

```
require(survJamda.data)

data(gse4335)
data(gse3143)
data(gse1992)

data(gse4335pheno)
data(gse3143pheno)
data(gse1992pheno)

geno.files = c("gse4335", "gse3143","gse1992")
surv.data = list(c(gse4335pheno[,6],gse3143pheno[,4],gse1992pheno[,19]),
                 c(gse4335pheno[,5],gse3143pheno[,3],gse1992pheno[,18]))

#The following script might  take some time
#main.single.indep.valid(geno.files, surv.data)
## The function is currently defined as
function(geno.files, surv.data,normalization = "zscore", method = "none", perf.eval = "auc")
{
       require(survival)
       require(survivalROC)

       if (!is.element(normalization, c("zscore","combat")))
                stop("\rnormalization = \"zscore\" or normalization = \"combat\"",
                call.=FALSE)

       if(normalization == "combat")
                batchID = det.batchID()

        for (i in 1:length(geno.files)){
              for (j in 1:length(geno.files))
                     if (i != j){
                            common.gene = intersect(colnames(get(geno.files[i])),
                            colnames(get(geno.files[j])))
                            ds1 = excl.missing.single.indep(geno.files[i],
                            surv.data,common.gene)
                            ds2 = excl.missing.single.indep(geno.files[j],
                            surv.data,common.gene)
```

```
                                if (normalization == "combat")
                                      mat = prepcombat.single.indep(ds1$mat,ds2$mat)
                                else
                                      mat = prepzscore(ds1$mat,ds2$mat)

                                i.adj = mat[1:nrow(ds1$mat),]
                                j.adj = mat[(nrow(ds1$mat)+1):nrow(mat),]
                                cat ("Train data set: ", geno.files[j], " Test data set: ",
                                geno.files[i], "\n")
                          calPerformance.single.indep(list(mat=j.adj,phyno=ds2$phyno,perf.eval),
                                list(mat=i.adj,phyno=ds1$phyno), method=method,perf.eval)
                        }
        }
}
```

meta.main *Meta analysis of survival data.*

Description

Meta analysis of microarray gene expression data for survival prediction.

Usage

```
meta.main(geno.files, surv.data, method = "none")
```

Arguments

geno.files	A vector of character strings containing the names of expression files.
surv.data	The list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
method	A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if method != "none".

Value

AUC, HR(CI) and p-value.

Author(s)

Haleh Yasrebi

Examples

```
require(survJamda.data)

data(gse4335)
data(gse3143)
data(gse1992)

data(gse4335pheno)
data(gse3143pheno)
data(gse1992pheno)

geno.files = c("gse4335", "gse3143","gse1992")
surv.data = list(c(gse4335pheno[,6],gse3143pheno[,4],gse1992pheno[,19]),
c(gse4335pheno[,5],gse3143pheno[,3],gse1992pheno[,18]))

#The following script might take some time
#meta.main(geno.files, surv.data)

## The function is currently defined as
function(geno.files,surv.data, method = "none")
{
        options(warn=-1)
        curr_set = 1:length(geno.files)
        for (y in curr_set){
                x = setdiff(curr_set, y)
                data.set.meta (x, y, geno.files,surv.data, method)
        }
}
```

`plot.roc.curves` *Plot ROC curves of the testing set normalized by a joint analysis method.*

Description

Plot ROC curves of the testing set normalized by a joint analysis method, Z-score2, Z-score1 or ComBat.

Arguments

`surv`	The vector of survival times of the testing set.
`censor`	The vector of censoring status of the testing set.
`lp`	The vector of biomarkers derived from the testing set.
`test`	The matrix of the testing set.

`file.name`	The name of the file of the testing set.
`col`	The color of the ROC curve.
`normalization`	The normalization method, Z-score2, Z-score1 or ComBat.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

Haleh Yasrebi, Comparative study of joint analysis of microarray gene expression data in survival prediction and risk assessment of breast cancer patients, Brief Bioinform, 2015 doi:10.1093/bib/bbv092.

`plot.time.dep` *Plot time-dependent ROC curves from 0 to 120 months.*

Description

Plot time-dependent ROC curves for the testing set from 0 to 120 months. As the clinical trials are carried out up to 10 years, the maximum time point is limited to 120 months.

Arguments

`surv`	The vector of survival times of the testing set.
`censor`	The vector of censoring status of the testing set.
`lp`	The vector of biomarkers derived from the testing set.
`test`	The matrix of the testing set.
`file.name`	The name of the file of the testing set.
`col`	The color of the ROC curve.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

Haleh Yasrebi, Comparative study of joint analysis of microarray gene expression data in survival prediction and risk assessment of breast cancer patients, Brief Bioinform, 2015 doi:10.1093/bib/bbv092.

`plotROC` *Plot ROC curves related to different time points.*

Description

Plot time-dependent AUC based on different survival time points.

Arguments

`test.ind`	Index of testing set.
`all.surv`	Vector of combined survival times.
`all.censor`	Vector of combined censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`lp`	Vector of linear predictor scores or markers. `lp` is the sum of gene expression values weighted by the Cox coefficients.
`file.name`	Vector of character strings containing the names of expression files.
`col`	Color of ROC curve.
`normalization`	The normalization method, Z-score2, Z-score1 or ComBat.
`time.dep`	An integer 0 or 1, 1 to plot time-dependent ROC curves for different time points and 0 for no plot

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

pool.zscores *Combine data for meta analysis.*

Description

Combine the expression data, survival data (survival time and censoring status) and gene Z-scores for meta analysis.

Usage

```
pool.zscores(common.gene, s, geno.files, surv.data)
```

Arguments

`common.gene`	A vector of character strings containing the name of the genes common to the data sets composing the training set.
`s`	A vector of integers specifying the index of the expression files composing the training set.
`geno.files`	A vector of character strings consisting of the names of the expression files.
`surv.data`	A list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

postmean *Estimated additive batch effect*

Description

Estimated additive batch effect

Usage

```
postmean(g.hat, g.bar, n, d.star, t2)
```

Arguments

g.hat Theoretical estimated additive batch effect.

g.bar Mean of g.hat

n A vector containing the sum of the rows of standardized gene expression data.

d.star Multiplicative batch effect.

t2 Variance of the rows of g.hat.

Value

Empirical estimated additive batch effect.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: http://dx.doi.org/10.1093/biostatistics/kxj037.

See Also

ComBat, it.sol

postvar *Estimated multiplicative batch effect*

Description

Estimated multiplicative batch effect

Usage

```
postvar(sum2, n, a, b)
```

Arguments

sum2 Batch sample variance.

n A vector containing the sum of the rows of standardized gene expression data.

a Value of aprior function applied to multiplicative batch effect.

b Value of bprior function applied to multiplicative batch effect.

Value

Estimated multiplicative batch effect

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: `http://dx.doi.org/10.1093/biostatistics/kxj037`.

See Also

`ComBat`,`aprior`,`bprior`

`pred.time.indep.valid` *Prediction of survival time by independent validation.*

Description

Identify the gene list common to all single data sets and invoke the subsequent function `main.process`.

Usage

```
pred.time.indep.valid(geno.files, surv.data, method = "none", time.dep = 0)
```

Arguments

`geno.files`	A vector of character strings containing the names of gene expression files.
`surv.data`	A list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`method`	A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function.
`time.dep`	An integer 0 or 1, 1 to plot time-dependent ROC curves for different time points and 0 for no plot

Value

None.

Author(s)

Haleh Yasrebi

Examples

```
require(survJamda.data)

data(gse4335)
data(gse3143)
data(gse1992)

data(gse4335pheno)
data(gse3143pheno)
data(gse1992pheno)

geno.files = c(”gse4335”,”gse1992”,”gse3143”)
surv.data = list(c(gse4335pheno[,6],gse1992pheno[,19],gse3143pheno[,4]),
                 c(gse4335pheno[,5],gse1992pheno[,18],gse3143pheno[,3]))
#pred.time.indep.valid(geno.files, surv.data)

## The function is currently defined as
function(geno.files, surv.data)
{
common.gene = colnames(get(geno.files[1]))
for (i in 2:length(geno.files))
common.gene = intersect(common.gene, colnames(get(geno.files[i])))

par (mfrow = c(1,length(geno.files)))
par(oma=c(2,2,length(geno.files),2))

main.process (common.gene, geno.files, surv.data)
  }
```

---

prepcombat   *Combination of data sets prior to the application of ComBat.*

Description

Combine the gene expression data, survival time and censoring status of at least two data sets prior to the application of ComBat.

Usage

```
prepcombat(common.gene, geno.files, surv.data, batchID, x, y)
```

Arguments

`common.gene` Vector of character strings specifying the names of the genes common to all single data sets.

`geno.files` Vector of character strings specifying the names of gene expression files.

`surv.data` The list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`batchID` For a given data set, the batch id can be an integer or the name of the data set. The batch id must be the same for all samples or arrays of a data set.

`x` Vector of character strings specifying the names of gene expression files composing the training set.

`y` A vector of character string specifying the name of gene expression file used as the testing set.

Value

A list of two objects, (i) ComBat-adjusted gene expression data and (ii) the list of two vectors, the merged survival time and censoring status.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`prepcombat.single.indep` *Pair-wise combination of single data sets prior to the application of ComBat and independent validation.*

Description

Combine in a pair-wise manner the gene expression values, survival time and censoring status of two single data sets prior to the application of ComBat.

Usage

```
prepcombat.single.indep(ds1, ds2,i, j, batchID)
```

Arguments

`ds1`	Matrix of gene expression data of one of the single data sets.
`ds2`	Matrix of gene expression data of the other single data sets.
`i`	An integer or character string specifying the batch ID of the data set `ds1`
`j`	An integer or character string specifying the batch ID of the data set `ds2`
`batchID`	The batch id can be an integer for a given data set or the name of a data set. The batch id must be the same for all samples or arrays of a data set.

Value

ComBat-adjusted merged gene expression data.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`prepzscore` *Z-score normalization.*

Description

Take two data sets, apply Z-score normalization to each and combine the two normalized data sets.

Usage

```
prepzscore(i, j)
```

Arguments

`i`	Matrix of gene expression data of the one of the two single data sets.
`j`	Matrix of gene expression data of the other single data set.

Value

Matrix of Z-score normalized merged data set.

Author(s)

Haleh Yasrebi

Examples

```
require(survJamda.data)

data(gse4335)
data(gse1992)

common.gene = intersect(colnames(gse4335),colnames(gse1992))
#m = prepzscore(gse4335[,common.gene],gse1992[,common.gene])

## The function is currently defined as
function (i, j)
{
        i = scale(t(scale(t(i))))
        j = scale(t(scale(t(j))))
        mat = rbind(i,j)
        return(mat)
}
```

---

prepzscore1 *Apply Z-score1 normalization.*

Description

Apply Z-score1 normalization before combining the data sets together. Each data sets is Z-score normalized separately and then, the data sets composing the training sets are combined together. The testing set is Z-score normalized independently and separately from the training set.

Usage

```
prepzscore1(common.gene, geno.files, surv.data, x, y)
```

Arguments

common.gene	A vector of character strings containing the name of the genes common to the data sets composing the training set.
geno.files	A vector of character strings containing the names of expression files.
surv.data	The list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
x	A vector of indices of the names of the expression files composing the training set.
y	Index of the name of expression file used as the testing set.

Value

A list of two objects, (i) the matrix of combined normalized gene expression data and (ii) a list of two vectors, the combined survival times and censoring status.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

See Also

znorm

---

prepzscore2 *Apply Z-score2 normalization.*

Description

Apply Z-score2 normalization. First, combine the data sets composing the training set and then, apply Z-score normalization to the merged data set.

Usage

```
prepzscore2(common.gene, geno.files, surv.data, x, y)
```

Arguments

`common.gene`	Vector of character strings containing the names of the genes common to the all data sets.
`geno.files`	A vector of character strings containing the names of gene expression files.
`surv.data`	The list of two vectors, survival time and censoring status related to the training set. In the censoring status vector, 1 = event occurred, 0 = censored.
`x`	A vector of indices of the names of data files composing the training set.
`y`	Index of the name of data file used as the testing set.

Value

A list of two objects related to the training set, (i) the matrix of Z-score normalized merged gene expression data and (ii) a list of two vectors, the combined survival time and censoring status.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

proc.simulate *Simulate survival data.*

Description

Simulation of survival data using a Weibull model. Two scenarios could be considered: Simulate gene expression values with and without correlation among genes. The aim is to determine if the prediction performance derived from the merged data set is mediated by the correlation among genes.

Usage

```
proc.simulate(tot.genes = 100, correlation = 0, gene.nb = 50, sample.nb = 400,
 beta.init = 0.5, shape = 1, scale = 1)
```

Arguments

`tot.genes`	Number of total genes.
`correlation`	Correlation among genes (between 0 and 1).
`gene.nb`	Number of true survival genes to identify.
`sample.nb`	Sample size.
`beta.init`	Initial value for the Cox coefficients.
`shape`	Shape parameter of the Weibull model.
`scale`	Scale parameter of the Weibull model.

Details

`beta` should not be close to zero. Otherwise, the true genes cannot be identified. Values between +/- 0.2 and +/- 3 would be good choices. Shape and scale should be selected between 1 and 2 (integer, preferably).

Value

Mean of AUC +/- standard deviation(AUC), geometric mean of HR (CI).

Warning

The user must enter a value for beta, scale and shape different from the initial values when (s)he is asked to do so. Otherwise, the old values are used for the second data set which will make the second data set have the same distribution as the first data set which is not desired.

Author(s)

Haleh Yasrebi

Examples

```
#using the default parameters, run the following script:
#proc.simulate()

#other values to be used:
#correlation = 0.8
#number of genes: 10,1000,5000

## The function is currently defined as
function(tot.genes = 100, correlation = 0, gene.nb = 50, sample.nb = 400,
beta.init = 0.5, shape = 1, scale = 1)
{
       require(survival)
       require(survivalROC)
       require(ecodist)

       d1 = generate.survival.data (gene.nb, tot.genes,sample.nb,beta.init, correlation,
       shape, scale)
       old.beta.init = beta.init

       cat ("Enter a different value for beta for the second data set:
       Example: beta.init = 0.1\n")
       beta.init = as.numeric(readline())
       old.shape = shape
       old.scale = scale

       cat ("Enter a different value for shape for the second data set.
       Enter only a numeric value. Example: 2\n")
       shape = as.numeric(readline())
       if (!is.numeric(shape))
              shape = old.shape

       cat ("Enter a different value for scale for the second data set.
              Enter only a numeric value. Example: 1.5\n")
       scale = as.numeric(readline())
       if (!is.numeric(scale))
              scale = old.scale

       d2 = generate.survival.data(gene.nb, tot.genes,sample.nb,beta.init,
       correlation,shape, scale)

       eval.merge.simulate(d1,d2,tot.genes, gene.nb, zscore = 1)
}
```

---

shuffle.samples *Shuffle samples.*

Description

To ensure the applicability of Cox regression, the function splits the samples randomly and assigns at least one deceased or relapsed patient to the training and testing sets.

Usage

```
shuffle.samples(n, censor, train.nb)
```

Arguments

`n` Sample size of the complete data set.

`censor` Vector of censoring status. 1 = event occurred, 0 = censored.

`train.nb` Number of samples in the training set.

Value

List of two vectors, the indices of the training set and the indices of the testing set.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`splitMerged.auc.plot` *Determine the indices of the training and testing sets.*

Description

Determine the indices of the training and testing sets prior to the plot of ROC curves.

Usage

```
splitMerged.auc.plot(geno.files, lst, i, j, col, method, time.dep)
```

Arguments

`geno.files` A vector of character strings containing the names of the expression files.

`lst` A list of two objects, (i) gene expression data and (ii) list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`i` A vector of character strings containing the names of the expression files used for the training set.

`j` A character string specifying the name of the expression file used as the testing set.

`col` Color of ROC curve.

`method` A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function

`time.dep` An integer 0 or 1, 1 to plot time-dependent ROC curves for different time points and 0 for no plot

Details

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if method != "none".

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

---

splitMerged.indep *Merge the data sets by ComBat or Z-score1 normalization and apply independent validation.*

Description

The data sets are either adjusted by Z-score1 normalization or ComBat. In Z-score1 normalization, all data sets are first Z-score normalized and then, merged together. The selection of data sets for the training and testing sets is performed before the application of independent validation.

Usage

```
splitMerged.indep(geno.files,lst, i, j, method,gn.nb,perf.eval,normalization)
```

Arguments

geno.files	a vector of character containing the names of gene expression data files.
lst	A list of two objects, (i) gene expression data and (ii) list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
i	Index of the names of the expression files composing the training set.
j	Index of the name of the expression file used as the testing set.
method	A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting method specifying by the p.adjust function
gn.nb	Number of genes to select for gene signature when method="none".
perf.eval	A string taking one the values, "auc", "cindex", "bsc".
normalization	A character string specifying the normalization method, "zscore1", "zscore2" or "combat".

Details

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If `perf.eval` == "auc", time-dependent AUC and hazard ratio are used as the measure of performance, `perf.eval` == "cindex", concordance index defined in the `survcomp` package or `perf.eval` == "bsc", brier score defined in the `survcomp` package is used.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

References

Yasrebi H, Sperisen P, Praz V, Bucher P, 2009 Can Survival Prediction Be Improved By Merging Gene Expression Data Sets?. PLoS ONE 4(10): e7431. doi:10.1371/journal.pone.0007431.

`splitZscore2.auc.plot` *Z-score2 normalization prior to AUC plot.*

Description

For independent validation, merge survival time and censoring status of the data sets composing the training set and apply the Z-score normalization prior to the plot of AUC.

Usage

```
splitZscore2.auc.plot(common.gene, geno.files, surv.data, lst, i, j,
 col, method, time.dep)
```

Arguments

`common.gene`	A vector of character strings containing the names of the genes common to the all data sets.
`geno.files`	A vector of character strings containing the names of gene expression files.
`surv.data`	A list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.
`lst`	The list of two objects, (i) matrix of gene expression data and (ii) list of two vectors, survival time and censoring status.

`i` Index of the file names composing the training set.

`j` Index of the file name used as the testing set.

`col` Color of ROC curve.

`method` A character string specifying the feature selection method: "none" for top-100 ranking or one of the adjusting methods specified by the p.adjust function.

`time.dep` An integer 0 or 1, 1 to plot time-dependent ROC curves for different time points and 0 for no plot

Details

Z-score2 normalization is performed as follows: First, the data sets are selected for the training and testing sets. Suppose there are S data set. Then, in S iteration, S-1 data sets are selected for the training set and the remaining set selected as the testing set until all data sets are used in the training and testing sets. The data sets composing the training set are merged together and the merged data set is then Z-score normalized. The testing set is independently adjusted by Z-score normalization.

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

`splitZscore2.merge.indep`

Merge data sets by Z-score2 normalization and assess the performance by independent validation.

Description

Z-score2 normalization is performed as follows: First, the data sets are selected for the training and testing sets. Suppose there are S data set. Then, in S iteration, S-1 data sets selected as the training set and the remaining set as the testing set until all data sets are used in the training and testing sets. The data sets composing the training set are merged together and the merged data set is then Z-score normalized. The testing set is independently adjusted by Z-score normalization.

Usage

```
splitZscore2.merge.indep(common.gene, geno.files, surv.data, lst, i, j,
method,gn.nb,perf.eval, normalization)
```

Arguments

`common.gene` A vector of character strings containing the names of the genes common to the datasets composing the training set.

`geno.files` A vector of character strings containing the names of the expression files.

`surv.data` A list of two vectors, survival time and censoring status. In the censoring status vector, 1 = event occurred, 0 = censored.

`lst` The list of two objects, the gene expression data and `surv.data`.

`i` Index of the names of the expression files composing the training set.

`j` Index of the name of the expression file used as the testing set.

`method` A character string specifying the feature selection method: "none" for top-ranking or one of the adjusting methods specified by the p.adjust function.

`gn.nb` Number of genes to select for gene signature when `method`="none".

`perf.eval` A string taking one the values, "auc", "cindex", "bsc".

`normalization` A character string specifying the normalization method, "zscore1", "zscore2" or "combat".

Details

In top-ranking, genes are selected based on univariate Cox P-value ranking using the coxph function in the R survival package. In this feature selection method, the genes are ranked based on their likelihood ratio P-value and the top-100 ranked genes with the smallest P-values are retained as the gene signature.

The p.adjust function in the R stats package is used and all adjusted p-values not greater than 0.05 are retained if `method` != "none".

If `perf.eval` == "auc", time-dependent AUC and hazard ratio are used as the measure of performance, `perf.eval` == "cindex", concordance index defined in the `survcomp` package or `perf.eval` == "bsc", brier score defined in the `survcomp` package is used.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

trim.dat *Trim the data.*

Description

Trim the data of extra columns with array names 'X' or start with 'X.'

Usage

```
trim.dat(dat)
```

Arguments

`dat`	Matrix of gene expression data.

Value

Matrix of gene expression data trimmed as specified in Description section.

Warning

This function is not called by the user directly.

Author(s)

WE Johnson

References

W. Johnson E., L. Chen, Rabinovic, and A. Adjusting batch effects in microarray expression data using Empirical Bayes methods. Biostatistics, 8(1):118-127, January2007. ISSN 1465-4644. doi: http://dx.doi.org/10.1093/biostatistics/kxj037.

writeGeno *Reformat gene expression data for ComBat.*

Description

Reformat gene expression data for ComBat.

Usage

```
writeGeno(x, fileName)
```

Arguments

`x`	Matrix of gene expression data to be adjusted.
`fileName`	A character string specifying the name of the file in which the adjusted data should be saved.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

See Also

`writeSamples`,`ComBat`

`writeSamples` *Write batch samples for ComBat.*

Description

Create a file for the batch IDs of the data sets.

Usage

```
writeSamples(x, batchID, fileName)
```

Arguments

`x`	Matrix of gene expression data.
`batchID`	A vector containing the batch IDs of the data set `x`. The batch ID of the data sets composing the matrix `x` should be in the same order of the component data sets.
`fileName`	A character string specifying the name of the file to be created.

Details

This function writes two columns in a file: Array.name and Batch. The Array.name column contains the array or sample ID which are the row names of the matrix `x`. The batch id in the second column can be an integer or the name of the data set. The batch id must be the same for all samples or arrays of a data set.

Value

None.

Warning

This function is not called by the user directly.

Author(s)

Haleh Yasrebi

See Also

`writeGeno`,`ComBat`

`znorm` *Matrix Z-score normalization.*

Description

Z-score normalization of a matrix

Usage

```
znorm(m)
```

Arguments

`m`	Matrix of gene expression data.

Value

Z-score normalized matrix of gene expression data `m`.

Author(s)

Haleh Yasrebi

References

Larsen, R. J. and Marx, M. L. (2000). An Introduction to Mathematical Statistics and Its Applications (3rd Edition), Prentice Hall.ISBN 0139223037.

Examples

```
require(survJamda.data)

data(gse4335)
data(gse3143)

common.gene = intersect(colnames(gse3143), colnames(gse4335))
m = znorm(rbind(gse3143[,common.gene],gse4335[,common.gene]))

## The function is currently defined as
function(m)
{
        m = scale(t(scale(t(m))))
        return(m)
}
```

Index