Package 'asuri'

May 14, 2026

Date 2024-04-03

Type Package

Title Analysis of SUrvival and RIsk prediction in patients based on gene signatures

Version 1.0.0

Description
The ASURI (Analysis of SUrvival and patients RIsk prediction based on gene signatures) package discovers marker genes that are related to risk prediction capabilities and to a clinical variable of interest. It uses two main steps, including subsampling glmnet and unicox. The package implements robust functions to discover survival markers related to a clinical phenotype and to predict a risk score, allowing to study the patient's risk based on the gene signatures. Several plots are provided to visualise the relevance of the genes, the risk score, and patient stratification, as well as a robust version of the Kaplan-Meier curves.

License LGPL-3 + file LICENSE

Depends R (>= 4.5.0), stats, methods

Imports SummarizedExperiment, spsUtil, lubridate, survival, glmnet, siggenes, survcomp, scales, ROCR, ggplot2, grDevices, graphics, utils, Rdpack

Suggests BiocStyle, knitr, formatR, rmarkdown, magick, devtools

biocViews Software, StatisticalMethod, WorkflowStep, GeneExpression, Microarray, DifferentialExpression, GenePrediction, Regression, Survival, ExonArray, MultipleComparison

URL https://github.com/jdelasrivas-lab/asuri

BugReports https://github.com/jdelasrivas-lab/asuri/issues

BiocType Software

ZipData TRUE

RoxygenNote 7.3.3

NeedsCompilation no

RdMacros Rdpack

Encoding UTF-8

VignetteBuilder knitr

LazyData false

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

git_branch RELEASE_3_23

git_last_commit 9c8ad97

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-05-14

Author Alberto Berral-Gonzalez [aut, cre, ctb] (ORCID: <https://orcid.org/0000-0001-8388-6051>), María Sanchez-Martin [aut, ctb] (ORCID: <https://orcid.org/0009-0006-5549-6113>), Santiago Bueno-Fortes [aut, ctb] (ORCID: <https://orcid.org/0000-0001-7141-2860>), Natalia Alonso-Moreda [aut, ctb] (ORCID: <https://orcid.org/0000-0002-2347-5127>), Jose Manuel Sanchez-Santos [aut, ctb] (ORCID: <https://orcid.org/0000-0002-7434-7598>), Manuel Martin-Merino Acera [aut, ctb] (ORCID: <https://orcid.org/0000-0002-5914-2332>), Javier De Las Rivas [aut, ctb] (ORCID: <https://orcid.org/0000-0002-0984-9946>)

Maintainer Alberto Berral-Gonzalez <aberralgonzalez@gmail.com>

Contents

`asuri` *asuri:* **A***nalysis of disease* **SU***rvival and patient* **RI***sk prediction based on gene signatures*

Description

The **asuri** package provides functions to discovers marker genes that are related to risk prediction capabilities and to a clinical variable of interest. It uses two main steps, including subsampling glmnet and unicox. The package implements robust functions to discover survival markers related to a clinical phenotype and to predict a risk score, allowing to study the patient's risk based on the gene signatures. Several plots are provided to visualise the relevance of the genes, the risk score, and patient stratification, as well as a robust version of the Kaplan-Meier curves.

Value

This page does not return any value. Provides general usage information.

Main functions

- `prefilterSAM`: Bootstrapped differential expression based on SAM.
- `genePheno`: Analyze gene-phenotype associations.
- `geneSurv`: Compute survival analysis for genes.
- `patientRisk`: Calculate patient risk scores.
- `predict_PatientRisk`: Predict patient risk for new data.
- `predict_SurvCurve`: Predict survival curves for patients.
- Plotting functions: `plotBoxplot`, `plotProbClass`, `plotLogRank`, `plotSigmoid`, `plotLambda`, `plotBetas`, `plotKM`.

`ex_genePheno` *ex_genePheno*

Description

Result of running the genePheno function.

Usage

```
data(ex_genePheno)
```

Format

A 'List' object with:

- genes: A list of genes ranked according to the degree of association with the clinical or phenotypic variable tested.
- listCoeff: A list with the beta regression coefficients and the AUC score for each bootstrap iteration.
- stability: Gene selection probability estimated by bootstrap (the number of times discovered over "n" iterations).
- betasMedian: Median of the beta coefficients over the B replicates.
- betasMean: Mean of the beta coefficients over the B replicates.
- betasTable: Table of genes ordered by decreasing value of the stability coefficient. Contains several metrics: the stability index, the mean and the median of the beta coefficients.

Value

Load a 'List' object.

See Also

genePheno

---

ex_patientRisk *ex_patientRisk*

Description

Result of running the patientRisk function.

Usage

```
data(ex_patientRisk)
```

Format

A 'List' containing the following elements:

- cv_risk_score: Risk score prediction for the training set using a double nested crossvalidated strategy.
- cv_normalized_risk: Normalized risk score in the interval (0,100).
- table_genes_selected: Data frame with the following columns: The names for the genes selected by the Cox regression, the beta coefficients for the optimal multivariate Cox regression fitted to the training set, the Hazard Ratio for each gene and the p-value for the univariate log-rank statistical test. Genes are shown by descending order of the HR index.

- `table_genes_selected_extended`: Table with the same format as table_genes_selected. A search for local minima within a 5% range of the selected minimum is performed. The goal is expanding the list of significant genes to improve biological interpretability, since the lasso penalty drastically reduces the number of significant genes.
- `model.optimalLambda`: The fitted model for the optimal regularization parameter.
- `groups`: Vector of classification of patients in two risk groups, high (2) or low (1).
- `riskThresholds`: Thresholds that allows to stratify the test patients in three groups according to the predicted risk score: low, intermediate and high risk.
- `range.risk`: Range of the unscaled risk score in the training set.
- `list.models`: List of models tested for different values of the regularization parameter.
- `evaluation.models`: Data frame that provides several metrics for each model evaluated. The lambda column provides the regularization parameter for the multivariate Cox regression adjusted, the number_features gives the number of genes selected by this model, c.index and se.c.index the concordance index and the standard deviation for the risk prediction and finally, the p_value_c.index and the logrank_p_value give the p-values for the the concordance index and the log-rank statistics respectively. Models are shown by ascending order of the log-rank p-value and the best one is marked with two asterisks.
- `betasplot`: Dataset used to create the plot of genes ranked according to the regression coefficients in the multivariate Cox model.
- `plot_values`: A list containing Kaplan-Meier fit results, logrank p-value, and hazard ratio.
- `membership_prob`: If method "class.probs" is selected a table with two columns is returned. The first one is the probability of classification to the low risk group while the second one is the membership probability to the high risk group.

Value

Load a ‘List‘ object.

See Also

`patientRisk`

`ex_predictPatientRisk` *ex_predictPatientRisk*

Description

Test matrix created in the PredictPatientRisk example

Usage

```
data(ex_predictPatientRisk)
```

Format

A 'data.frame' object with:

data: A matrix of normalized gene expression values (77 genes × 1 samples).

Value

Load a 'data.frame' object.

See Also

predict_PatientRisk

---

ex_prefilterSAM *ex_prefilterSAM*

Description

Result of running the prefilterSAM function.

Usage

```
data(ex_prefilterSAM)
```

Format

A 'character' vector object with the list of obtained genes.

Value

Load a 'character' vector object.

See Also

prefilterSAM

genePheno *Identify Predictive Genes for a Phenotype*

Description

This function implements robust algorithms to obtain a list of genes associated to a given clinical variable. It is based on the elastic net algorithm and the robustness and reproducibility of the subset of genes is improved using a bootstrap strategy combined with ensemble methods.

Usage

```
genePheno(
  seData,
  DEgenes,
  vectorGroups,
  vectorSampleID,
  iter = 100,
  numberOfFolds = 5,
  verbose = TRUE
)
```

Arguments

seData	SummarizedExperiment object with the normalized expression data and the phenotypic data in colData.
DEgenes	Vector containing the genes to be used. Expected to be in the same format as the rows of the assay(seData). Usually this vector is the result of running prefilterSAM().
vectorGroups	Clinical variable or phenotypic variable tested. It must be provided as a numeric binary vector.
vectorSampleID	Vector containing the sample names in the same order as in assay(seData).
iter	Number of bootstrap iterations (default: 100, should be changed if the function takes too long to execute).
numberOfFolds	Number of folds to implement nested cross-validation. By default 5.
verbose	Logical. Show progress bar.

Details

This function implements a robust version of the elastic net algorithm proposed by Tibshirani (Tibshirani et al., 2009). This algorithm considers a penalty term to avoid overfitting that is a convex combination of the L₂ norm (ridge regression) and L₁ (Lasso regression). When the alpha parameter is 1, the regularization term perfoms similarly to Lasso and minimizes the number of non-null coefficients. If a subset of features are slightly correlated Lasso selects only one of them randomly. To avoid this extreme behavior the alpha parameter is set up to 0.75 that includes more relevant variables than Lasso and improves the prediction accuracy. Besides, this

choice will help to improve the stability and to reduce the variance in the feature selection process. In order to improve the robustness and reproducibility of the gene signature discovered, a bootstrap strategy is implemented. The patients are resampled with replacement giving rise to B replicates. For each replicate, a gene signature is obtained using double nested cross-validation to avoid overfitting. The final gene list is built as an ensemble of lists, considereing several metrics that evaluate the stability, the robustness and the predictive power of each gene. See (Martinez-Romero et al., 2018) for more details.

Value

A list containing the following elements:

- `genes`: A list of genes ranked according to the degree of association with the clinical or phenotypic variable tested.
- `listCoeff`: A list with the beta regression coefficients and the AUC score for each bootstrap iteration.
- `stability`: Gene selection probability estimated by bootstrap (the number of times discovered over "n" iterations).
- `betasMedian`: Median of the beta coefficients over the B replicates.
- `betasMean`: Mean of the beta coefficients over the B replicates.
- `betasTable`: Table of genes ordered by decreasing value of the stability coefficient. Contains several metrics: the stability index, the mean and the median of the beta coefficients.

References

- Martinez-Romero J, Bueno-Fortes S, Martín-Merino M, Molina ARD, De Las Rivas J (2018). "Survival marker genes of colorectal cancer derived from consistent transcriptomic profiling." *BMC Genomics*, **19**(8), 45–60. ISSN 14712164. doi:10.1186/S1286401851939. https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5193-9.
- Bueno-Fortes S, Berral-Gonzalez A, Sánchez-Santos JM, Martin-Merino M, De Las Rivas J (2023). "Identification of a gene expression signature associated with breast cancer survival and risk that improves clinical genomic platforms." *Bioinformatics Advances*, **3**(1). ISSN 26350041. doi:10.1093/BIOADV/VBAD037. https://dx.doi.org/10.1093/bioadv/vbad037.

Examples

```
data(seBRCA)

# prefilterSAM ---
data(ex_prefilterSAM)

# genePheno ---
vectorSampleID <- rownames(SummarizedExperiment::colData(seBRCA))
vectorGroups <- SummarizedExperiment::colData(seBRCA)$ER.IHC |> as.numeric()

ex_genePheno <- genePheno(seBRCA, ex_prefilterSAM, vectorGroups, vectorSampleID,
                          iter = 25)

# NOTE: For consistent results with the vignettes and example data, use
```

```
# default parameters (e.g., iter = 100).
```

---

geneSurv *Kaplan-Meier Survival Analysis Based on Gene Expression or Risk Score*

Description

This function analyzes the ability of a gene to mark survival based on a robust version of the KM curves. The robust K-M estimator is obtained by a bootstrap strategy.

Usage

```
geneSurv(
  seData,
  time,
  status,
  geneName,
  boxplot = TRUE,
  iter = 100,
  type = c("exprs", "risk"),
  cut_time = 10,
  verbose = TRUE
)
```

Arguments

seData	SummarizedExperiment object with the normalized expression data and the phenotypic data in colData. Phenotypic colData must contain the samples name in the first column and two columns with the time and the status.
time	SummarizedExperiment colData column name containing the survival time in years for each sample in numeric format.
status	SummarizedExperiment colData column name containing the status (censored 0 and not censored 1) for each sample.
geneName	A character string with the name of the gene being analyzed.
boxplot	A logical value indicating whether to generate a boxplot of gene expression by survival group (default = TRUE).
iter	The number of iterations (bootstrap resampling) for calculating optimal group cutoffs (default = 100).
type	Defines if the KM curve groups are computed using risk ("risk") or gene expression (default "exprs").
cut_time	A numeric value specifying the cutoff time (in years) for survival analysis. All events beyond this time are treated as censored (default = 10 years).
verbose	Logical. Show progress bar.

Details

This function improves the stability and robustness of the K-M estimator using a bootstrap strategy. Patients are resampled with replacement giving rise to B replicates. The K-M estimator is obtained based on the replicates as well as the confidence intervals. The patients are stratified in two risk groups by an expression threshold that optimizes the log-rank statistics, that is the separability between the Kaplan-Meier curves for each group. This function implements a novel method to find the optimal threshold avoiding the problems of instability and unbalanced classes that suffer other implementations. Besides, a membership probability for each risk group is estimated from the classification of each sample in the replicates. This membership probability allow us to reclassify patients around the gene expression threshold in a more robust way. The function provides a robust estimation of the log-rank p-value and the Hazard ratio that allow us to evaluate the ability of a given gene to mark survival.

Value

Depending on the type run, the output changes:

- For `type = exprs`, a Kaplan-Meier plot based on expression groups, a differential expression boxplot and a plot with the membership probability for each risk group. Additionally, an object with the following components:
  - `geneName`: A character string with the selected name of the gene to analyze.
  - `patientExpr`: The expression level of each patient for the gene.
  - `patientClass`: Vector of group classification according to the gene expression level: 2 = high expression, and 1 = low expression level.
  - `patientClassProbality`: Vector of membership probabilities for the classification.
  - `wilcox.pvalue`: The p-value from the Wilcoxon test comparing the two expression groups.
  - `plot_values`: A list containing Kaplan-Meier fit results, log-rank p-value, and hazard ratio.
- For `type = risk`, a Kaplan-Meier plot based on risk groups. Additionally, an object with the following components:
  - `geneName`: A character string with the selected name of the gene to analyze.
  - `patientExpr`: The expression level of each patient for the gene.
  - `risk_score_predicted`: A numeric vector of predicted relative risk scores for each patient.
  - `plot_values`: A list containing Kaplan-Meier fit results, log-rank p-value, and hazard ratio.

References

- Martinez-Romero J, Bueno-Fortes S, Martín-Merino M, Molina ARD, De Las Rivas J (2018). "Survival marker genes of colorectal cancer derived from consistent transcriptomic profiling." *BMC Genomics*, **19**(8), 45–60. ISSN 14712164. doi:10.1186/S1286401851939. https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5193-9.
- Bueno-Fortes S, Berral-Gonzalez A, Sánchez-Santos JM, Martin-Merino M, De Las Rivas J (2023). "Identification of a gene expression signature associated with breast cancer survival and risk that improves clinical genomic platforms." *Bioinformatics Advances*, **3**(1). ISSN 26350041. doi:10.1093/BIOADV/VBAD037. https://dx.doi.org/10.1093/bioadv/vbad037.

Examples

```
data(seBRCA)
time <- "time"
status <- "status"
geneName <- "ESR1"
# The TIME value must be transformed to YEARS
# The gene expression vector must be provided with the NAMES of each sample,
# that should match the time and status NAMES.
set.seed(5)
outputKM <- geneSurv(seBRCA, time, status, geneName, type = "exprs")

# Generate the plots again
## Plots for c(type = exprs)
plotBoxplot(outputKM)
plotProbClass(outputKM)
plotKM(outputKM)

# If we instead consider to run the function as *type* = risk

geneName <- "BRCA1"
set.seed(5)
outputKM.TP53 <- geneSurv(seBRCA, time, status, geneName, type = "risk")

## Plots for c(type = risk)
plotKM(outputKM.TP53)
```

---

patientRisk *patientRisk*

Description

This function selects a subset of good risk markers and estimates a multivariate risk score based on the UNICOX algorithm. The patients are stratified into two or more prognostic groups based on the risk score. The Cox regression is trained using a ten-fold double nested crossvalidation strategy to avoid overfitting.

Usage

```
patientRisk(
  seData,
  selectedGenes,
  time,
  status,
  group.vector,
  method = NULL,
  nboot = 50,
  cut_time = 10
)
```

Arguments

seData	SummarizedExperiment object with the normalized expression data and the phenotypic data in colData. Phenotypic colData must contain the samples name in the first column and two columns with time and status.
selectedGenes	Vector containing the genes to be used. Expected to be in the same format as the rows of the assay(seData). Usually this vector is the result of running prefilterSAM().
time	SummarizedExperiment colData column name containing the survival time in years for each sample in numeric format.
status	SummarizedExperiment colData column name containing the status (censored 0 and not censored 1) for each sample.
group.vector	A numeric vector specifying predefined risk groups for the patients. This is optional.
method	A character string specifying the method for defining risk groups, the default method is "class.probs". Possible options are: - "min.pval": Define risk groups based on the minimum p-value. - "med.pval": Define risk groups based on the median p-value. - "class.probs": Defines risk groups based on the classification probabilities from the model.
nboot	An integer specifying the number of bootstrap iterations for risk score calculation. Default is 50.
cut_time	A numeric value specifying the cutoff time (in years) for survival analysis. All events beyond this time are treated as censored (default = 10 years).

Details

A multivariate Cox regression is trained to select a subset of genes significantly associated with the risk and to estimate a risk score based on these risk markers. The algorithm considered is based on UNICOX, a regularized multivariate Cox regression model (see Tibshirani et al., 2009 for more details). In this predictor, the variables are penalized individually using an L_1 norm term which allow us to keep more relevant genes correlated with risk than in Lasso. The Lasso model selects only one representative gene randomly from the set of correlated genes. The optimal value for the lambda parameter as well as the risk score are estimated using a double nested crossvalidation strategy. Finally, the risk score allow us to stratify the whole set of patients according to their risks. Three algorithms are implemented to estimate the optimal threshold that classifies the patients in risk groups. "min.pval" determines the optimal threshold by minimization of the log-rank p-value statistics, that is by maximization of the separability between the K-M curves for the high and low risk groups, see (Martinez-Romero et al., 2018). When several local minima arise this may be sample dependent and unstable. To avoid this problem, "med.pval" estimates the optimal threshold as the median of the lower 10th percentile logrank p-values. The lower 10th percentile selects the smallest values from the p-value distribution corresponding to intermediate risk patients that are on the boundary between both groups. This interval is more robust than a single minimum and provides good experimental results for a large variety of problems tested. The median threshold in this interval may change from one iteration to another because the distribution of p-values for patients with intermediate risk may change due to sample variations. Finally, "class.probs" implements a bootstrap strategy for the patients corresponding to the lower 10th percentile p-values and estimates a robust threshold to stratify the patients. It estimates also a membership probability of classification.

Value

A list containing the following elements:

- `cv_risk_score`: Risk score prediction for the training set using a double nested crossvalidated strategy.
- `cv_normalized_risk`: Normalized risk score in the interval (0,100).
- `table_genes_selected`: Data frame with the following columns: The names for the genes selected by the Cox regression, the beta coefficients for the optimal multivariate Cox regression fitted to the training set, the Hazard Ratio for each gene and the p-value for the univariate log-rank statistical test. Genes are shown by descending order of the HR index.
- `table_genes_selected_extended`: Table with the same format as table_genes_selected. A search for local minima within a 5% range of the selected minimum is performed. The goal is expanding the list of significant genes to improve biological interpretability, since the lasso penalty drastically reduces the number of significant genes.
- `model.optimalLambda`: The fitted model for the optimal regularization parameter.
- `groups`: Vector of classification of patients in two risk groups, high (2) or low (1).
- `riskThresholds`: Thresholds that allows to stratify the test patients in three groups according to the predicted risk score: low, intermediate and high risk.
- `range.risk`: Range of the unscaled risk score in the training set.
- `list.models`: List of models tested for different values of the regularization parameter.
- `evaluation.models`: Data frame that provides several metrics for each model evaluated. The lambda column provides the regularization parameter for the multivariate Cox regression adjusted, the number_features gives the number of genes selected by this model, c.index and se.c.index the concordance index and the standard deviation for the risk prediction and finally, the p_value_c.index and the logrank_p_value give the p-values for the the concordance index and the log-rank statistics respectively. Models are shown by ascending order of the log-rank p-value and the best one is marked with two asterisks.
- `betasplot`: Dataset used to create the plot of genes ranked according to the regression coefficients in the multivariate Cox model.
- `plot_values`: A list containing Kaplan-Meier fit results, logrank p-value, and hazard ratio.
- `membership_prob`: If method "class.probs" is selected a table with two columns is returned. The first one is the probability of classification to the low risk group while the second one is the membership probability to the high risk group.

References

- Martinez-Romero J, Bueno-Fortes S, Martín-Merino M, Molina ARD, De Las Rivas J (2018). "Survival marker genes of colorectal cancer derived from consistent transcriptomic profiling." *BMC Genomics*, **19**(8), 45–60. ISSN 14712164. doi:10.1186/S1286401851939. https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5193-9.
- Bueno-Fortes S, Berral-Gonzalez A, Sánchez-Santos JM, Martin-Merino M, De Las Rivas J (2023). "Identification of a gene expression signature associated with breast cancer survival and risk that improves clinical genomic platforms." *Bioinformatics Advances*, **3**(1). ISSN 26350041. doi:10.1093/BIOADV/VBAD037. https://dx.doi.org/10.1093/bioadv/vbad037.

Examples

```
data(seBRCA)

# genePheno ---
data(ex_genePheno)

# Survival times should be provided in YEARS
time <- 'time'
status <- 'status'
geneList <- names(ex_genePheno$genes)

set.seed(5)
ex_patientRisk <- patientRisk(seBRCA, geneList, time, status,
                              method = "class.probs",
                              nboot = 10)

# NOTE: For consistent results with the vignettes and example data, use
# default parameters (e.g., nboot = 50).

# Generate the plots again
# plotLogRank(ex_patientRisk)
# plotSigmoid(ex_patientRisk)
# plotLambda(ex_patientRisk)
# plotBetas(ex_patientRisk)
# plotKM(ex_patientRisk)
```

---

plotBetas *Plot the ordered beta values for patientRisk()*

Description

This function is called internally by the patientRisk function to generate the beta values of the model for each gene and the corresponding stability p-value. It also can be used after the function has been run to generate the plots again without running patientRisk().

Usage

```
plotBetas(pr_result)
```

Arguments

pr_result	Returning object from running patientRisk function

Value

Plot of the beta values of the model for each gene from patientRisk()

See Also

patientRisk for more information about the analysis

Examples

```
data(seBRCA)

# genePheno ---
data(ex_genePheno)

# Survival times should be provided in YEARS
time <- 'time'
status <- 'status'
geneList <- names(ex_genePheno$genes)

set.seed(5)
ex_patientRisk <- patientRisk(seBRCA, geneList, time, status,
                              method = "class.probs",
                              nboot = 10)

# NOTE: For consistent results with the vignettes and example data, use
# default parameters (e.g., nboot = 50).

# Generate the plots again
plotBetas(ex_patientRisk)
```

---

plotBoxplot *Plot a Boxplot for geneSurv()*

Description

This function is called internally by the geneSurv function to generate the expression boxplot. It also can be used after the function has been run to generate the plots again without running geneSurv().

Usage

```
plotBoxplot(gs_result)
```

Arguments

gs_result	Returning object from running geneSurv function

Value

Boxplot with the expression values from geneSurv()

See Also

geneSurv for more information about the analysis

Examples

```
data(seBRCA)
time <- "time"
status <- "status"
geneName <- "ESR1"
# The TIME value must be transformed to YEARS
# The gene expression vector must be provided with the NAMES of each sample,
# that should match the time and status NAMES.
set.seed(5)
outputKM <- geneSurv(seBRCA, time, status, geneName, type = "exprs")

# Generate the plots again
## Plots for c(type = exprs)
plotBoxplot(outputKM)
```

---

plotKM *Plot the KM curves for geneSurv() and patientRisk()*

Description

This function is called internally by the patientRisk function to generate the Kaplan-Meier curves. It also can be used after the function has been run to generate the plots again without running geneSurv() or patientRisk().

Usage

```
plotKM(
  result,
  col.surv = NULL,
  col.ci = NULL,
  par.bot = 6,
  par.left = 10,
  par.top = 2,
  par.right = 4,
  y.just.legend = 0.5,
  x.title.adj = 1.75,
  mark = 3,
  simple = FALSE,
  xaxis.at = c(0:10),
  xaxis.lab = xaxis.at,
  lty.surv = 1,
  lty.ci = 3,
  lwd = 4,
  lwd.surv = 4,
  lwd.ci = 4,
  group.names = "",
  group.order = seq_along(km$n),
```

```
  extra.left.margin = 4,
  label.n.at.risk = FALSE,
  draw.lines = TRUE,
  cex.axis = 1.25,
  main = "",
  xlim = c(0, 10),
  ylim = c(0, 1),
  grid = TRUE,
  lty.grid = 1,
  lwd.grid = 1,
  col.grid = grey(0.9),
  legend = TRUE,
  loc.legend = "bottomleft",
  add = FALSE,
  ...
)
```

Arguments

result	Returning object from running genesurv or patientRisk function
col.surv	Color for the survival curve lines
col.ci	Color for the confidence interval lines
par.bot	Bottom margin of the plot
par.left	Left margin of the plot
par.top	Top margin of the plot
par.right	Right margin of the plot
y.just.legend	Vertical adjustment for the legend
x.title.adj	Horizontal adjustment for the x-axis title
mark	Symbol used for censoring marks
simple	Logical; if TRUE, use a simplified plot style
xaxis.at	Positions of ticks on the x-axis
xaxis.lab	Labels for the x-axis ticks
lty.surv	Line type for survival curves
lty.ci	Line type for confidence intervals
lwd	Overall line width
lwd.surv	Line width for survival curves
lwd.ci	Line width for confidence intervals
group.names	Names of groups in the plot
group.order	Order of groups to display
extra.left.margin	Additional left margin space
label.n.at.risk	Logical; whether to label number at risk

draw.lines	Logical; whether to draw the survival lines
cex.axis	Magnification of axis annotation
main	Main title of the plot
xlim	Limits of the x-axis
ylim	Limits of the y-axis
grid	Logical; whether to draw grid lines
lty.grid	Line type for grid lines
lwd.grid	Line width for grid lines
col.grid	Color for the grid lines
legend	Logical; whether to include a legend
loc.legend	Location of the legend (e.g., "bottomleft")
add	Logical; if TRUE, add to an existing plot
...	Additional arguments passed to plotting functions

Value

Plot of Kaplan-Meier for each risk/expression group from geneSurv() or patientRisk()

See Also

geneSurv or patientRisk for more information about the analysis

Examples

```
# Genesurv
data(seBRCA)
time <- "time"
status <- "status"
geneName <- "ESR1"
# The TIME value must be transformed to YEARS
# The gene expression vector must be provided with the NAMES of each sample,
# that should match the time and status NAMES.
set.seed(5)
outputKM <- geneSurv(seBRCA, time, status, geneName, type = "exprs")

# Generate the plots again
## Plots for c(type = exprs)
plotKM(outputKM)

# PatientRisk
data(seBRCA)

# genePheno ---
data(ex_genePheno)

# Survival times should be provided in YEARS
time <- 'time'
```

```
status <- 'status'
geneList <- names(ex_genePheno$genes)

set.seed(5)
ex_patientRisk <- patientRisk(seBRCA, geneList, time, status,
                              method = "class.probs",
                              nboot = 10)

# NOTE: For consistent results with the vignettes and example data, use
# default parameters (e.g., nboot = 50).

# Generate the plots again
plotKM(ex_patientRisk)
```

`plotLambda` *Plot a lambda distribution for patientRisk()*

Description

This function is called internally by the patientRisk function to generate the p-value distribution for each lambda. It also can be used after the function has been run to generate the plots again without running patientRisk().

Usage

```
plotLambda(pr_result)
```

Arguments

`pr_result`	Returning object from running patientRisk function

Value

Plot of the p-value distribution for each lambda from patientRisk()

See Also

`patientRisk` for more information about the analysis

Examples

```
data(seBRCA)

# genePheno ---
data(ex_genePheno)

# Survival times should be provided in YEARS
time <- 'time'
status <- 'status'
```

```
geneList <- names(ex_genePheno$genes)

set.seed(5)
ex_patientRisk <- patientRisk(seBRCA, geneList, time, status,
                              method = "class.probs",
                              nboot = 10)

# NOTE: For consistent results with the vignettes and example data, use
# default parameters (e.g., nboot = 50).

# Generate the plots again
plotLambda(ex_patientRisk)
```

---

plotLogRank *Plot the logrank p-value for each patient splitting for patientRisk()*

Description

This function is called internally by the patientRisk function to generate the logrank p-value distribution for each patient splitting. It also can be used after the function has been run to generate the plots again without running patientRisk().

Usage

```
plotLogRank(pr_result)
```

Arguments

pr_result	Returning object from running patientRisk function

Value

Plot of the log-rank p-value distribution from geneSurv()

See Also

patientRisk for more information about the analysis

Examples

```
data(seBRCA)

# genePheno ---
data(ex_genePheno)

# Survival times should be provided in YEARS
time <- 'time'
status <- 'status'
geneList <- names(ex_genePheno$genes)
```

```
set.seed(5)
ex_patientRisk <- patientRisk(seBRCA, geneList, time, status,
                              method = "class.probs",
                              nboot = 10)

# NOTE: For consistent results with the vignettes and example data, use
# default parameters (e.g., nboot = 50).

# Generate the plots again
plotLogRank(ex_patientRisk)
```

`plotProbClass` *Plot a class probability distribution for geneSurv()*

Description

This function is called internally by the geneSurv function to generate the probability class distribution of the patients. It also can be used after the function has been run to generate the plots again without running geneSurv().

Usage

```
plotProbClass(gs_result)
```

Arguments

`gs_result`	Returning object from running geneSurv function

Value

Plot of the class probability distribution across samples for geneSurv()

See Also

`geneSurv` for more information about the analysis

Examples

```
data(seBRCA)
time <- "time"
status <- "status"
geneName <- "ESR1"
# The TIME value must be transformed to YEARS
# The gene expression vector must be provided with the NAMES of each sample,
# that should match the time and status NAMES.
set.seed(5)
outputKM <- geneSurv(seBRCA, time, status, geneName, type = "exprs")
```

```
# Generate the plots again
## Plots for c(type = exprs)
plotProbClass(outputKM)
```

---

plotSigmoid *Plot the ordered risk distribution for patientRisk()*

Description

This function is called internally by the patientRisk function to generate the ordered risk distribution for the patients. It also can be used after the function has been run to generate the plots again without running patientRisk().

Usage

```
plotSigmoid(pr_result)
```

Arguments

pr_result	Returning object from running patientRisk function

Value

Plot of the ordered risk distribution from patientRisk()

See Also

patientRisk for more information about the analysis

Examples

```
data(seBRCA)

# genePheno ---
data(ex_genePheno)

# Survival times should be provided in YEARS
time <- 'time'
status <- 'status'
geneList <- names(ex_genePheno$genes)

set.seed(5)
ex_patientRisk <- patientRisk(seBRCA, geneList, time, status,
                              method = "class.probs",
                              nboot = 10)

# NOTE: For consistent results with the vignettes and example data, use
# default parameters (e.g., nboot = 50).
```

```
# Generate the plots again
plotSigmoid(ex_patientRisk)
```

`predict_PatientRisk` *Predict Patient Risk Based on Gene Expression Data*

Description

Function to predict the risk for new patients considering the gene expression for a subset of genes and the multivariate Cox regression model trained by function patientRisk, it may be used to predict over a single patient.

Usage

```
predict_PatientRisk(model.fit, mExpr.testData)
```

Arguments

`model.fit`	A list containing the pre-fitted model and necessary parameters for risk prediction, including the optimal lambda value, risk thresholds, and plot values.
`mExpr.testData`	A data frame representing the gene expression data of test patients, where each row is a gene and each column is a sample.

Details

A risk score is estimated for new patients considering the optimal regularized multivariate Cox regression model trained by the function patientRisk. The risk score is normalized to be interpretable in the scale (0-100). The function generates a risk plot for new patients and stratifies them in three risk groups (low, intermediate, high) considering the thresholds learned by function patientRisk.

Value

A list containing the following elements:

- `risk_score`: A vector with the unscaled risk score for new patients estimated by a multivariate Cox regression model.
- `scaled_risk_score`: A vector with the risk score for the new patients scaled to be interpretable in the range 0-100.
- `plot_values`: A list containing information for visualizing the sigmoid curve and risk thresholds.

References

- Martinez-Romero J, Bueno-Fortes S, Martín-Merino M, Molina ARD, De Las Rivas J (2018). "Survival marker genes of colorectal cancer derived from consistent transcriptomic profiling." *BMC Genomics*, **19**(8), 45–60. ISSN 14712164. doi:10.1186/S1286401851939. https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5193-9.
- Bueno-Fortes S, Berral-Gonzalez A, Sánchez-Santos JM, Martin-Merino M, De Las Rivas J (2023). "Identification of a gene expression signature associated with breast cancer survival and risk that improves clinical genomic platforms." *Bioinformatics Advances*, **3**(1). ISSN 26350041. doi:10.1093/BIOADV/VBAD037. https://dx.doi.org/10.1093/bioadv/vbad037.

Examples

```
data(seBRCA)

# prefilterSAM ---
data(ex_prefilterSAM)

# genePheno ---
data(ex_genePheno)

# patientRisk ---
data(ex_patientRisk)

# Simulate expression data
num_samples <- 20
geneList <- names(ex_genePheno$genes)
set.seed(5)
mExprs_testData <- matrix(rnorm(length(geneList) * num_samples,
                          mean = 10, sd = 3),
                          nrow = length(geneList), ncol = num_samples)

# Assign row names (genes) and column names (samples)
rownames(mExprs_testData) <- geneList
colnames(mExprs_testData) <- paste0("Sample", 1:num_samples)

set.seed(5)
risk_prediction_validation_set <- predict_PatientRisk(
                                                ex_patientRisk,
                                                mExprs_testData)


# Example for single patient prediction: Patient fourth is selected.
ex_predictPatientRisk <- data.frame(mExprs_testData[, 4])
colnames(ex_predictPatientRisk) <- colnames(mExprs_testData)[4]
# Risk prediction for the optimal subset of genes selected by patientRisk()
set.seed(5)
risk_prediction_one_patient <- predict_PatientRisk(
                                                ex_patientRisk,
                                                ex_predictPatientRisk)
```

predict_SurvCurve *Predict Survival Curve for a Test Patient Using Cox Model Predictions*

Description

This function computes the survival curve for a test patient based on their predicted risk score using a Cox proportional hazards model. It estimates the baseline survival function from training data and predicts the survival probability for the test patient at specified times.

Usage

```
predict_SurvCurve(
  cox_pred_training,
  mSurv,
  cox_pred_test,
  eval_surv_times = NULL
)
```

Arguments

cox_pred_training	A numeric vector representing the predicted risk scores (or log-risk scores) from the Cox model for the training set. It can be obtained from *predict_PatientRisk* function.
mSurv	A data frame with two columns: "time" representing survival times, and "status" representing the event status (1 for event, 0 for censored) for the training dataset.
cox_pred_test	A numeric vector of length 1 representing the predicted risk score (or log-risk score) for the test patient. It can be obtained from *predict_PatientRisk* function.
eval_surv_times	A numeric vector of times at which the survival curve should be evaluated. If NULL (default), the times will be taken from the training data up to the maximum event time.

Details

This function calculates the baseline survival curve using the training data by first estimating the hazard function and cumulative hazard function. Then, it computes the survival probability for the test patient using the baseline survival function and the test patient's risk score. If the test patient's risk is associated with a hazard ratio greater than 1, the survival curve will decrease more rapidly. If 'eval_surv_times' is provided, the curve is evaluated at those specific times; otherwise, the function uses the survival times from the training data.

Value

A list containing:

- `eval_times`: The times at which the survival curve is evaluated.
- `S_0_t`: The baseline survival function evaluated at the times in `eval_surv_times`.
- `S_test_patient`: The survival curve for the test patient at the times in `eval_surv_times`.

References

- Martinez-Romero J, Bueno-Fortes S, Martín-Merino M, Molina ARD, De Las Rivas J (2018). "Survival marker genes of colorectal cancer derived from consistent transcriptomic profiling." *BMC Genomics*, **19**(8), 45–60. ISSN 14712164. doi:10.1186/S1286401851939. https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5193-9.
- Bueno-Fortes S, Berral-Gonzalez A, Sánchez-Santos JM, Martin-Merino M, De Las Rivas J (2023). "Identification of a gene expression signature associated with breast cancer survival and risk that improves clinical genomic platforms." *Bioinformatics Advances*, **3**(1). ISSN 26350041. doi:10.1093/BIOADV/VBAD037. https://dx.doi.org/10.1093/bioadv/vbad037.

Examples

```
data(seBRCA)

# prefilterSAM ---
data(ex_prefilterSAM)

# genePheno ---
data(ex_genePheno)

# patientRisk ---
data(ex_patientRisk)

# predictPatientRisk ---
data(ex_predictPatientRisk)


#COX prediction for the training set
mPheno <- SummarizedExperiment::colData(seBRCA)
mExprs <- SummarizedExperiment::assay(seBRCA)
geneList <- names(ex_genePheno$genes)

set.seed(5)
mExprSelectedGenes <- mExprs[rownames(mExprs) %in% geneList,]

cox_pred_training <- predict_PatientRisk(ex_patientRisk,
                                         mExprSelectedGenes)
cox_pred_training$risk_score
#COX prediction for the test patient
set.seed(5)
cox_pred_test <- predict_PatientRisk(ex_patientRisk,
                                     ex_predictPatientRisk)
cox_pred_test$risk_score
#Survival curve estimation
eval_surv_times <- seq(0, max(mPheno$time), by = 0.1)
set.seed(5)
```

```
surv_curv_cox <- predict_SurvCurve(cox_pred_training$risk_score,
                                   mPheno[, c(2, 3)],
                                   cox_pred_test$risk_score,
                                   eval_surv_times)
```

`prefilterSAM` *Pre-filter Genes using SAM (Significance Analysis of Microarrays)*

Description

This function performs pre-filtering of genes based on Significance Analysis of Microarrays (SAM). It runs a bootstrapping procedure to select significant genes based on the False Discovery Rate (FDR) and allows filtering by the percentage of times a gene is selected across multiple iterations.

Usage

```
prefilterSAM(
  seData,
  groupsVector,
  FDRfilter = 0.05,
  iter = 100,
  percentageFilter = 80,
  verbose = TRUE
)
```

Arguments

`seData`	SummarizedExperiment object with the normalized expression data and the phenotypic data in colData.
`groupsVector`	A binary vector indicating group assignment the samples.
`FDRfilter`	A numeric value indicating the FDR threshold for selecting significant genes. Default is 0.05.
`iter`	The number of iterations for bootstrapping. Default is 100.
`percentageFilter`	A numeric value indicating the percentage of iterations a gene must appear in to be considered significant. Default is 80.
`verbose`	Logical. Show progress bar.

Details

This function implements SAM (Schwender H., 2022) robust diferential expression analysis based on bootstrap . It helps to remove noisy genes reducing the computational complexity of further analysis. The function uses a bootstrapping approach, where in each iteration a random sample is drawn from the input data (with replacement), and the SAM algorithm is applied to select significant genes. The genes that appear as significant in a specified percentage of iterations (controlled by ‘percentageFilter‘) are retained.

Value

An ordered vector with the names of differentially expressed genes between the categories of the grouping vector. A list of DE genes ordered by SAM d.value and filtered by percentageFilter.

References

- Schwender H (2025). *siggenes: Multiple Testing using SAM and Efron's Empirical Bayes Approaches*. doi:10.18129/B9.bioc.siggenes. R package version 1.82.0, https://bioconductor.org/packages/siggenes.
- Martinez-Romero J, Bueno-Fortes S, Martín-Merino M, Molina ARD, De Las Rivas J (2018). "Survival marker genes of colorectal cancer derived from consistent transcriptomic profiling." *BMC Genomics*, **19**(8), 45–60. ISSN 14712164. doi:10.1186/S1286401851939. https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5193-9.
- Bueno-Fortes S, Berral-Gonzalez A, Sánchez-Santos JM, Martin-Merino M, De Las Rivas J (2023). "Identification of a gene expression signature associated with breast cancer survival and risk that improves clinical genomic platforms." *Bioinformatics Advances*, **3**(1). ISSN 26350041. doi:10.1093/BIOADV/VBAD037. https://dx.doi.org/10.1093/bioadv/vbad037.

Examples

```
data(seBRCA)

# Bootstrapped differential expression based on SAM.
# Parameters: FDR = 0.05, iter = 100, percentage filter = 80 %
# CAUTION: if the data have a high number of genes this function will take
# several minutes to compute.

groupsVector <- SummarizedExperiment::colData(seBRCA)$ER.IHC

set.seed(5)
ex_prefilterSAM <- prefilterSAM(seBRCA, groupsVector, iter = 25)

# NOTE: For consistent results with the vignettes and example data, use
# default parameters (e.g., iter = 100).
```

---

seBRCA *Example SummarizedExperiment object*

Description

A 'SummarizedExperiment' object created from gene expression and phenotypic data included in the ASURI package. The dataset included in ASURI as a stud case is intended to facilitate the use of the package. The dataset was obtained from several GSE series from GEO database corresponding to breast cancer (BRCC) samples, where expression and survival time were reported (Bueno *et al.*, 2023). Each sample corresponds to genome-wide expression profiles of BRCC primary tumor samples hybridized on the transcriptomic platform: *Affymetrix* *HGU133 Plus2.0*; which is a high-density microarray expression platform containing 594,000 oligo-nucleotide probes (organized in

probe-sets) that we mapped to ENSEMBL genes using the *hgu133plus2hsensgcdf* CDF package, obtained from BRAINARRAY (hgu133plus2hsensgcdf_24.0.0.tar.gz.) The expression signal of the samples was normalized using fRMA (McCall,MN *et al.*, 2010), or RMA (Gautier,L *et al.*, 2004) and Combat (McCall,MN *et al.*, 2010), as described in Bueno *et al.*, 2023. It integrates 1070 genes measured across 200 breast cancer samples with associated clinical annotations.

Usage

```
data(seBRCA)
```

Format

A 'SummarizedExperiment' object with:

assay: A matrix of normalized gene expression values (1070 genes × 200 samples).

colData: A DataFrame with phenotypic variables for 200 samples.

Value

Load a 'SummarizedExperiment' object.

Source

Derived from GEO datasets as described in Bueno *et al.*, 2023].

References

- Schwender H (2025). *siggenes: Multiple Testing using SAM and Efron's Empirical Bayes Approaches*. doi:10.18129/B9.bioc.siggenes. R package version 1.82.0, https://bioconductor.org/packages/siggenes.
- Martinez-Romero J, Bueno-Fortes S, Martín-Merino M, Molina ARD, De Las Rivas J (2018). "Survival marker genes of colorectal cancer derived from consistent transcriptomic profiling." *BMC Genomics*, **19**(8), 45–60. ISSN 14712164. doi:10.1186/S1286401851939. https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5193-9.
- Bueno-Fortes S, Berral-Gonzalez A, Sánchez-Santos JM, Martin-Merino M, De Las Rivas J (2023). "Identification of a gene expression signature associated with breast cancer survival and risk that improves clinical genomic platforms." *Bioinformatics Advances*, **3**(1). ISSN 26350041. doi:10.1093/BIOADV/VBAD037. https://dx.doi.org/10.1093/bioadv/vbad037.
- McCall,MN, Bolstad,BM, Irizarry,RA (2010). "Frozen robust multiarray analysis (fRMA)." *Biostatistics*, **11**(2), 242–253. doi:10.1093/biostatistics/kxp059.
- Gautier,L, Cope,L, Bolstad,BM, Irizarry,RA (2004). "affy—analysis of Affymetrix GeneChip data at the probe level." *Bioinformatics*, **20**(3), 307–315. ISSN 1367-4803. doi:10.1093/bioinformatics/btg405.
- Kupfer P, Guthke R, Pohlers D, Huber,R, Koczan D, Kinne RW (2012). "Batch correction of microarray data substantially improves the identification of genes differentially expressed in rheumatoid arthritis and osteoarthritis." *BMC Medical Genomics*, **5**, 23. doi:10.1186/1755-8794523.

See Also

SummarizedExperiment

Index