Package ‘survivalSL’

July 2, 2025

Type Package

Title Super Learner for Survival Prediction from Censored Data

Version 0.98

Depends R (>= 4.0.0), splines, survival

Imports date, graphics, MASS, glmnet, caret, flexsurv, randomForestSRC, hdnom, survivalPLANN, dplyr, rpart, methods

Description Several functions and S3 methods to construct a super learner in the presence of censored times-to-event and to evaluate its prognostic capacities.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

BugReports `https://github.com/foucher-y/survivalSL/issues`

Author Yohann Foucher [aut, cre] (ORCID: <`https://orcid.org/0000-0003-0330-7457`>), Camille Sabathe [aut] (ORCID: <`https://orcid.org/0000-0001-9353-691X`>)

Maintainer Yohann Foucher <`yohann.foucher@univ-poitiers.fr`>

Repository CRAN

Date/Publication 2025-07-02 16:10:02 UTC

Contents

`dataDIVAT2` *A Sample from the DIVAT Data Bank.*

Description

A data frame with 1912 French kidney transplant recipients from the DIVAT cohort.

Usage

```
data(dataDIVAT2)
```

Format

A data frame with the 4 following variables:

- `age` This numeric vector provides the age of the recipient at the transplantation (in years).
- `hla` This numeric vector provides the indicator of transplantations with at least 4 HLA incompatibilities between the donor and the recipient (1 for high level and 0 otherwise).
- `retransplant` This numeric vector provides the indicator of re-transplantation (1 for more than one transplantation and 0 for first kidney transplantation).

`ecd` The Expended Criteria Donor (1 for transplantations from ECD and 0 otherwise). ECD are defined by widely accepted criteria, which includes donors older than 60 years of age or 50-59 years of age with two of the following characteristics: history of hypertension, cerebrovascular accident as the cause of death or terminal serum creatinine higher than 1.5 mg/dL.

`times` This numeric vector is the follow up times of each patient.

`failures` This numeric vector is the event indicator (0=right censored, 1=event). An event is considered when return in dialysis or patient death with functioning graft is observed.

Source

URL: www.divat.fr

References

Le Borgne F, Giraudeau B, Querard AH, Giral M and Foucher Y. Comparisons of the performances of different statistical tests for time-to-event analysis with confounding factors: practical illustrations in kidney transplantation. Statistics in medicine. 30;35(7):1103-16, 2016. <doi:10.1002/sim.6777>

Examples

```
data(dataDIVAT2)

# Compute the non-adjusted Hazard Ratio related to the ECD versus SCD
cox.ecd<-coxph(Surv(times, failures) ~ ecd, data=dataDIVAT2)
summary(cox.ecd) # Hazard Ratio = 1.97
```

`dataDIVAT3` *A Sample from the DIVAT Data Bank.*

Description

A data frame with 4267 French kidney transplant recipients.

Usage

```
data(dataDIVAT3)
```

Format

A data frame with 4267 observations for the 8 following variables.

`ageR` This numeric vector represents the age of the recipient (in years)

`sexeR` This numeric vector represents the gender of the recipient (1=men, 0=female)

`year.tx` This numeric vector represents the year of the transplantation

`ante.diab` This numeric vector represents the diabetes statute (1=yes, 0=no) `pra` This numeric vector represents the pre-graft immunization using the panel reactive antibody (1=detectable, 0=undetectable)

`ageD` This numeric vector represents the age of the donor (in years)

`death.time` This numeric vector represents the follow up time in days (until death or censoring)

`death` This numeric vector represents the death indicator at the follow-up end (1=death, 0=alive)

Source

URL: www.divat.fr

References

Le Borgne et al. Standardized and weighted time-dependent ROC curves to evaluate the intrinsic prognostic capacities of a marker by taking into account confounding factors. Manuscript submitted. Stat Methods Med Res. 27(11):3397-3410, 2018. <doi: 10.1177/ 0962280217702416.>

Examples

```
data(dataDIVAT3)

### a short summary of the recipient age at the transplantation
summary(dataDIVAT3$ageR)

### Kaplan and Meier estimation of the recipient survival
plot(survfit(Surv(death.time/365.25, death) ~ 1, data = dataDIVAT3),
 xlab="Post transplantation time (in years)", ylab="Patient survival",
 mark.time=FALSE)
```

`dataOFSEP` *A Simulated Sample from the OFSEP Cohort.*

Description

A data frame with 1300 simulated French patients with multiple sclerosis from the OFSEP cohort. The baseline is 1 year after the initiation of the first-line treatment.

Usage

```
data(dataOFSEP)
```

Format

A data frame with 1300 observations for the 3 following variables:

`time` This numeric vector represents the follow up time in years (until disease progression or censoring)

`event` This numeric vector represents the disease progression indicator at the follow-up end (1=progression, 0=censoring) `age` This numeric vector represents the patient age (in years) at baseline.

`duration` This numeric vector represents the disease duration (in days) at baseline.

`period` This numeric vector represents the calendar period: 1 in-between 2014 and 2018, and 0 otherwise.

`gender` This numeric vector represents the gender: 1 for women.

`relapse` This numeric vector represents the diagnosis of at least one relapse since the treatment initiation : 1 if at leat one event, and 0 otherwise.

`edss` This vector of character string represents the EDSS level: "miss" for missing, "low" for EDSS between 0 to 2, and "high" otherwise.

`t1` This vector of character string represents the new gadolinium-enhancing T1 lesion: "missing", "0" or "1+" for at least 1 lesion.

`t2` This vector of character string represents the new T2 lesions: "no" or "yes".

`rio` This numeric vector represents the modified Rio score.

Examples

```
data(dataOFSEP)

### Kaplan and Meier estimation of the disease progression free survival
plot(survfit(Surv(time, event) ~ 1, data = dataOFSEP),
     ylab="Disease progression free survival",
     xlab="Time after the first anniversary of the first-line treatment in years")
```

`LIB_AFTgamma` *Library of the Super Learner for an Accelerated Failure Time (AFT) Model with a Gamma Distribution*

Description

Fit an AFT parametric model with a gamma distribution.

Usage

```
LIB_AFTgamma(formula, data)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

Details

The model is obtained by using the `dist="gamma"` in the `flexsurvreg` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Jackson, C. (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_AFTgamma(formula=formula, data=dataDIVAT2[1:200,])

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_AFTggamma` *Library of the Super Learner for an Accelerated Failure Time (AFT) Model with a Generalized Gamma Distribution*

Description

Fit an AFT parametric model with a generalized gamma distribution.

Usage

```
LIB_AFTggamma(formula, data)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

Details

The model is obtained by using the `dist="gengamma"` in the `flexsurvreg` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Jackson, C. (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

Examples

```
data("dataDIVAT2")


# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_AFTggamma(formula=formula, data=dataDIVAT2[1:200,])

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_AFTllogis` *Library of the Super Learner for an Accelerated Failure Time (AFT) Model with a Log Logistic Distribution*

Description

Fit an AFT parametric model with a log logistic distribution.

Usage

```
LIB_AFTllogis(formula, data)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

Details

The model is obtained by using the `dist="llogis"` in the `flexsurvreg` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Jackson, C. (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

Examples

```
data("dataDIVAT2")


# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_AFTllogis(formula=formula, data=dataDIVAT2[1:200,])

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_AFTweibull` *Library of the Super Learner for an Accelerated Failure Time (AFT) Model with a Weibull Distribution*

Description

Fit an AFT parametric model with a Weibull distribution.

Usage

```
LIB_AFTweibull(formula, data)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

Details

The model is obtained by using the `dist="weibull"` in the `flexsurvreg` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Jackson, C. (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

Examples

```
data("dataDIVAT2")


# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_AFTweibull(formula=formula, data=dataDIVAT2[1:200,])

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_COXaic` *Library of the Super Learner for a Cox Model with Selected Covariates*

Description

Fit a Cox regression for a selection of covariate.

Usage

```
LIB_COXaic(formula, data, penalty=NULL)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the predictoes on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.
`penalty`	A numerical vector with a length equals to the number of predictors. It allows the integration of covariates into the final model, i.e. with no selection: the value 0 to force the covariate in the model, 1 otherwise. If `NULL`, all covariates undergo the selection process.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_COXaic(formula=formula, data=dataDIVAT2[1:200,])
```

```
# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_COXall` *Library of the Super Learner for Cox Regression*

Description

Fit a Cox regression for all covariates to be used in the super learner.

Usage

```
LIB_COXall(formula, data)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

Details

The Cox regression is obtained by using the `survival` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Terry M. Therneau (2021). A Package for Survival Analysis in R. R package version 3.2-13, https://CRAN.R-project.org/package=survival.

Examples

```
data("dataDIVAT2")


# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
data<-dataDIVAT2[1:200,]
model <- LIB_COXall(formula=formula, data=data)

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_COXen` *Library of the Super Learner for Elastic Net Cox Regression*

Description

Fit an elastic net Cox regression for fixed values of the regularization parameters.

Usage

```
LIB_COXen(formula, data, penalty=NULL, alpha, lambda)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.
`penalty`	A numerical vector that allows the covariates not to be penalized. We give the value 0 if we do not want the covariate to be penalized otherwise 1. If `NULL`, all covariates are penalized.
`alpha`	The value of the regularization parameter alpha for penalizing the partial likelihood.
`lambda`	The value of the regularization parameter lambda for penalizing the partial likelihood.

Details

The elastic net Cox regression is obtained by using the `glmnet` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lignes

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_COXen(formula=formula, data=dataDIVAT2[1:200,], lambda=.1, alpha=.1)

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_COXlasso` *Library of the Super Learner for Lasso Cox Regression*

Description

Fit a Lasso Cox regression for a fixed value of the regularization parameter.

Usage

```
LIB_COXlasso(formula, data, penalty=NULL, lambda)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

`penalty`	A numerical vector with a length equals to the number of predictors. It allows the integration of covariates into the final model, i.e. with no selection: the value 0 to force the covariate in the model, 1 otherwise. If `NULL`, all covariates undergo the selection process.
`lambda`	The value of the regularization parameter lambda for penalizing the partial likelihood.

Details

The Lasso Cox regression is obtained by using the `glmnet` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_COXlasso(formula=formula, data=dataDIVAT2[1:200,], lambda=.1)

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_COXridge` *Library of the Super Learner for Ridge Cox Regression*

Description

Fit a ridge Cox regression for a fixed value of the regularization parameter.

Usage

```
LIB_COXridge(formula, data, penalty=NULL, lambda)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.
`penalty`	A numerical vector with a length equals to the number of predictors. It allows the integration of covariates into the final model, i.e. with no selection: the value 0 to force the covariate in the model, 1 otherwise. If `NULL`, all covariates undergo the selection process.
`lambda`	The value of the regularization parameter lambda for penalizing the partial likelihood.

Details

The ridge Cox regression is obtained by using the `glmnet` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_COXridge(formula=formula, data=dataDIVAT2[1:200,], lambda=.1)

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_PHexponential` *Library of the Super Learner for a Proportional Hazards (PH) Model with an Exponential Distribution*

Description

Fit a PH model with an Exponential distribution.

Usage

```
LIB_PHexponential(formula, data)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

Details

The model is obtained by using the `dist="exp"` in the `flexsurvreg` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Jackson, C. (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_PHexponential(formula=formula, data=dataDIVAT2[1:200,])

# The predicted survival of the first subject of the training sample
```

```
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_PHgompertz` *Library of the Super Learner for an Proportional Hazards (PH) Model with a Gompertz Distribution*

Description

Fit a PH parametric model with a Gompertz distribution.

Usage

```
LIB_PHgompertz(formula, data)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

Details

The model is obtained by using the `dist="gompertz"` in the `flexsurvreg` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Jackson, C. (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_PHgompertz(formula=formula, data=dataDIVAT2[1:200,])

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_PHspline` *Library of the Super Learner for an Survival Regression using the Royston/Parmar Spline Model*

Description

Fit an PH model with a survival function is modelled as a natural cubic spline function.

Usage

```
LIB_PHspline(formula,
data, k)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.
`k`	Number of knots.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Jackson, C. (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_PHspline(formula=formula, data=dataDIVAT2[1:200,], k=2)

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_PLANN` *Library of the Super Learner for Survival Neural Network Based on the PLANN Method*

Description

Fit a neural network based on the partial logistic regression.

Usage

```
LIB_PLANN(formula, data, inter, size, decay,
          maxit, MaxNWts, maxtime=NULL)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.
`inter`	The length of the intervals.
`size`	The number of units in the hidden layer.
`decay`	The parameter for weight decay.
`maxit`	The maximum number of iterations.
`MaxNWts`	The maximum allowable number of weights.
`maxtime`	A numeric value with the maximum prognostic time. If `NULL`, the maximum prognostic time is the maximum value of database times + 1.

Details

This function is based is based on the `survivalPLANN` from the related package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Biganzoli E, Boracchi P, Mariani L, and et al. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. Stat Med, 17:1169-86, 1998.

Examples

```
data("dataDIVAT2")

# The neural network based from the first 300 individuals of the data base

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_PLANN(formula, data=dataDIVAT2[1:300,],
  inter=0.5, size=32, decay=0.01, maxit=100, MaxNWts=10000, maxtime=NULL)

# The predicted survival of the first subject of the training sample

plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`LIB_RSF` *Library of the Super Learner for Survival Random Survival Forest*

Description

Fit survival random forest tree for given values of the regularization parameters.

Usage

```
LIB_RSF(formula, data, nodesize, mtry, ntree, seed=NULL)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.
`nodesize`	The value of the node size.

`mtry`	The number of variables randomly sampled as candidates at each split.
`ntree`	The number of trees.
`seed`	A random seed to ensure reproducibility during bootstrap sampling. If `NULL`, a seed is randomly assigned.

Details

The survival random forest tree is obtained by using the `randomForestSRC` package.

Value

`formula`	The formula object used for model construction.
`model`	The estimated model.
`data`	The data frame used for learning.
`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

References

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

# The estimation of the model

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_RSF(formula, data=dataDIVAT2, nodesize=10,
  mtry=2, ntree=100, seed=NULL)

# The predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`metrics` *Metrics to Evaluate the Prognostic Capacities*

Description

Compute several metrics to evaluate the prognostic capacities with time-to-event data.

Usage

```
metrics(metric, formula=NULL, data=NULL,
survivals.matrix=NULL, hazards.matrix=NULL, prediction.times=NULL,
object=NULL, pro.time=NULL, ROC.precision=seq(.01, .99, by=.01))
```

Arguments

`metric`	The metric to compute. See details.
`formula`	The formula used to build the survivals.matrix.
`data`	A data frame for in which to look for the variables related to the status of the follow-up time.
`survivals.matrix`	A matrix with the predictions of survivals of each subject (lines) for each prognostic times (columns).
`hazards.matrix`	A matrix with the predictions of hazards of each subject (lines) for each prognostic times (columns).
`prediction.times`	A vector of numeric values with the times of the `predictions` (same length than the number of columns of `prediction.matrix`).
`pro.time`	This optional value of prognostic time represents the maximum delay for which the capacity of the variable is evaluated. The same unit than the one used in the argument times. Not used for the following metrics: "ll", "ibs", and "ibll". Default value is the time at which half of the subjects are still at risk.
`object`	An object of type `libsl`, is `NULL` by default. When a value is assigned to it, the other parameters automatically inherit the formula, predictions, times of predictions and data from the object. When the object is set to `NULL`, the parameters `formula`, `survivals.matrix`, `prediction.times` and `data` must be defined.
`ROC.precision`	An optional argument with the percentiles (between 0 and 1) of the prognostic variable used for computing each point of the time dependent ROC curve. Only used when `metric="auc"`. 0 (min) and 1 (max) are not allowed. By default, the precision is `seq(.01,.99,.01)`.

Details

The following metrics can be used: "bs" for the Brier score at the prognostic time `pro.time`, "p_ci" and "uno_ci" for the concordance index at the prognostic time `pro.time` (Pencina and Uno versions), "ll" for the log-likelihood, "ibs" for the integrated Brier score up to the last observed time in the training data, "ibll" for the integrated binomial log-likelihood up to the last observed time in the training data, "ribs" for the restricted integrated Brier score up to the prognostic time `pro.time`, "ribll" for the restricted integrated binomial log-likelihood up to the prognostic time `pro.time`, "bll" for the binomial log-likelihood, "auc" for the area under the time-dependent ROC curve up to the prognostic time `pro.time`.

Value

A numeric value with the metric estimation.

Examples

```
data("dataDIVAT2")


# The estimation of the model

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_COXridge(formula, data=dataDIVAT2, lambda=.1)

# The apparent AUC

metrics(metric="auc", object=model)

# The integrated Brier score up to 10 years post-transplanation

metrics(metric="ribs", object=model, pro.time=10)
```

`plot.libsl` *Calibration Plot*

Description

A calibration plot of an object of the class `libsl` (library of survival super learner).

Usage

```
## S3 method for class 'libsl'
plot(x, n.groups=5, pro.time=NULL,
newdata=NULL, ...)
```

Arguments

`x`	An object returned by a library of survival super learner.
`n.groups`	A numeric value with the number of groups by their class probabilities. The default is 5.
`pro.time`	The prognostic time at which the calibration plot of the survival probabilities.
`newdata`	An optional data frame containing the new sample for validation with covariate values, follow-up times, and event status. The default value is `NULL`, the calibration plot is performed from the same subjects of the training sample.
`...`	Additional arguments affecting the plot.

Details

The plot represents the observed survival and the related 95% confidence intervals, which are respectively estimated by the Kaplan and Meier estimator and the Greenwood formula, against the mean of the predictive values for individuals stratified into groups of the same size according to the percentiles. The identity line is usually included for reference.

Value

No return value for this S3 method.

See Also

`plot.default`

Examples

```
data("dataDIVAT2")

# The estimation of the model

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
data=dataDIVAT2[1:150,]
model <- LIB_COXall(formula, data=data)

# The calibration plot from the validation sample of 150 patients
plot(model, n.groups=5, pro.time=12, col=3,
     xlab="Predicted 12-year survival", ylab="Observed 12-year survival",
     newdata=dataDIVAT2[151:300,])
```

`plot.sltime` *Calibration Plot for Super Learner*

Description

A calibration plot of a Super Learner obtained by the function `survivalSL`.

Usage

```
## S3 method for class 'sltime'
plot(x, method="sl", n.groups=5, pro.time=NULL, newdata=NULL,
 ...)
```

Arguments

`x`	An object returned by the function `survivalSL`.
`method`	A character string with the name of the algorithm included in the SL for which the calibration plot is performed. The default is "sl" for the Super Learner.
`n.groups`	A numeric value with the number of groups by their class probabilities. The default is 5.
`pro.time`	The prognostic time at which the calibration plot of the survival probabilities.
`newdata`	An optional data frame containing the new sample for validation with covariate values, follow-up times, and event status. The default value is `NULL`, the calibration plot is performed from the same subjects of the training sample.
`...`	Additional arguments affecting the plot.

Details

The plot represents the observed survival and the related 95% confidence intervals, which are respectively estimated by the Kaplan and Meier estimator and the Greenwood formula, against the mean of the predictive values for individuals stratified into groups of the same size according to the percentiles. The identity line is usually included for reference.

Value

No return value for this S3 method.

See Also

`plot.default`

Examples

```
data("dataDIVAT2")


#The outcome model base on a Super Learner from the first 150 individuals of the data base


formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

sl1 <- survivalSL(formula, data=dataDIVAT2[1:150,],
                  methods=c("LIB_AFTgamma", "LIB_PHgompertz"), metric="auc",  cv=3)

# The calibration plot from the validation sample of 150 patients
plot(sl1, method="sl", n.groups=5,
pro.time=12, col=2,
     xlab="Predicted 12-year survival",
     ylab="Observed 12-year survival",
     newdata=dataDIVAT2[151:300,])
```

`predict.libsl` *Prediction from an Flexible Parametric Model*

Description

Predict the survival based on a model or algorithm from an object of the class `libsl`.

Usage

```
## S3 method for class 'libsl'
predict(object, newdata, newtimes, ...)
```

Arguments

`object`	An object of the class `libsl`.
`newdata`	An optional data frame containing covariate values at which to produce predicted values. The default value is `NULL`, the predicted values are computed for the subjects of the training sample.
`newtimes`	The times at which to produce predicted values. The default value is `NULL`, the predicted values are computed for the observed times in the training data frame.
`...`	For future methods.

Value

`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	A matrix with the predictions of survivals of each subject (lines) for each observed time (columns).

Examples

```
data("dataDIVAT2")

# The estimation of the model from the first 200 lines

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_PHgompertz(formula, data=dataDIVAT2[1:200,])

# Predicted survival for 2 new subjects
pred <- predict(model,
  newdata=data.frame(age=c(52,52), hla=c(0,1), retransplant=c(1,1), ecd=c(0,1)))

plot(y=pred$predictions[1,], x=pred$times, xlab="Time (years)", ylab="Predicted survival",
     col=1, type="l", lty=1, lwd=2, ylim=c(0,1))

lines(y=pred$predictions[2,], x=pred$times, col=2, type="l", lty=1, lwd=2)

legend("bottomright", col=c(1,2), lty=1, lwd=2, c("Subject #1", "Subject #2"))
```

`predict.sltime` *Prediction from a Super Learner for Censored Outcomes*

Description

Predict the survival of new observations based on an SL by using the `survivalSL` function.

Usage

```
## S3 method for class 'sltime'
predict(object, newdata, newtimes, ...)
```

Arguments

`object`	An object returned by the function `survivalSL`.
`newdata`	An optional data frame containing covariate values at which to produce predicted values. The default value is `NULL`, the predicted values are computed for the subjects of the training sample.
`newtimes`	The times at which to produce predicted values. The default value is `NULL`, the predicted values are computed for the observed times in the training data frame.
`...`	For future methods.

Value

`predictions`	A list of matrix with the predictions of survivals of each subject (lines) for each observed time (columns) for each model used for the superlearner construction and the superlearner itself.
`times`	A vector of numeric values with the times of the `predictions`.

See Also

`survivalSL`.

Examples

```
data("dataDIVAT2")

# The training of the super learner from the first 150 individuals of the data base


formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

sl1 <- survivalSL(formula, data=dataDIVAT2[1:150,],
             method=c("LIB_COXridge", "LIB_AFTggamma"), metric="auc", pro.time = 12, cv=3)

# Individual prediction for 2 new subjects
pred <- predict(sl1,
  newdata=data.frame(age=c(52,52),
  hla=c(0,1), retransplant=c(1,1), ecd=c(0,1)))

plot(y=pred$predictions$sl[1,], x=pred$times,
xlab="Time (years)",
ylab="Predicted survival",
col=1, type="l", lty=1, lwd=2, ylim=c(0,1))

lines(y=pred$predictions$sl[2,], x=pred$times,
col=2, type="l", lty=1, lwd=2)

legend("bottomright", col=c(1,2), lty=1, lwd=2,
c("Subject #1", "Subject #2"))
```

`print.libsl` *S3 Method for Printing an 'libsl' Object*

Description

Print the model or algorithm.

Usage

```
## S3 method for class 'libsl'
print(x, ...)
```

Arguments

`x`	A 'libsl' object.
`...`	For future methods.

Value

No return value for this S3 method.

Examples

```
data("dataDIVAT2")

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
model <- LIB_AFTgamma(formula,  data=dataDIVAT2[1:100,])

print(model)
```

`print.sltime` *S3 Method for Printing an 'sltime' Object*

Description

Print the contribution of learners included in the super learner.

Usage

```
## S3 method for class 'sltime'
print(x,  digits=7, ...)
```

Arguments

`x`	An object returned by the function `survivalSL`.
`digits`	An optional integer for the number of digits to print when printing numeric values.
`...`	For future methods.

Value

No return value for this S3 method.

Examples

```
data("dataDIVAT2")

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

sl1 <- survivalSL(formula, data=dataDIVAT2[1:150,],
         method=c("LIB_COXridge", "LIB_AFTggamma"),
         metric="auc", pro.time = 12, cv=3)

print(sl1, digits=4)
```

`summary.libsl` *Summaries of a Learner*

Description

Return predictive performances of a model or algorithm obtained by a library of the class `libsl`.

Usage

```
## S3 method for class 'libsl'
summary(object, newdata=NULL, ROC.precision=seq(.01,.99,.01),
      digits=7, pro.time=NULL, ...)
```

Arguments

`object`	An object returned by a library of the class `libsl`.
`newdata`	An optional data frame containing the new sample for validation with covariate values, follow-up times, and event status. The default value is `NULL`, the summary is performed on the same subjects of the training sample.
`ROC.precision`	An optional argument with the percentiles (between 0 and 1) of the prognostic variable used for computing each point of the time dependent ROC curve. 0 (min) and 1 (max) are not allowed. By default, the precision is `seq(.01,.99,.01)`.
`digits`	An optional integer for the number of digits to print when printing numeric values.

`pro.time`	This optional value of prognostic time represents the maximum delay for which the capacity of the variable is evaluated. The same unit than the one used in the argument times. Not used for the following metrics: "ll", "ibs", and "ibll". Default value is the time at which half of the subjects are still at risk.
`...`	Additional arguments affecting the summary which are passed from `libsl` by default.

Details

The following metrics can be used : "bs" for the Brier score at the prognostic time `pro.time`, "p_ci" and "uno_ci" for the concordance index at the prognostic time `pro.time` (Pencina and Uno versions), "ll" for the log-likelihood, "ibs" for the integrated Brier score up to the last observed time in the training data, "ibll" for the integrated binomial log-likelihood up to the last observed time in the training data, "ribs" for the restricted integrated Brier score up to the prognostic time `pro.time`, "ribll" for the restricted integrated binomial log-likelihood up to the prognostic time `pro.time`, "bll" for the binomial log-likelihood, "auc" for the area under the time-dependent ROC curve up to the prognostic time `pro.time`.

Value

`metrics`	A data frame containing the computed predictive performance metrics.
`library`	The name of the library used for model training.
`pro.time`	The prognostic time used for evaluation.
`ROC.precision`	The precision values used for the ROC curve computation.

Examples

```
data("dataDIVAT2")

# The training of the gompertz  model with the first 400 patients

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

data<-dataDIVAT2[1:400,]
model <- LIB_PHgompertz(formula, data=data)

# The prognostic capacities from the same training sample
summary(model)

# The prognostic capacities from a validation of the next 150 patients
# (up to 4 years for several indicators)

#newdata<-dataDIVAT2[401:550,]
#summary(model, pro.time=4, newdata=newdata)
```

`summary.sltime` *Summaries of a Super Learner*

Description

Return goodness-of-fit indicators of a Super Learner obtained by the function `survivalSL`.

Usage

```
## S3 method for class 'sltime'
summary(object, newdata=NULL,  method="sl",
ROC.precision=seq(.01,.99,.01), digits=7, pro.time=NULL, ...)
```

Arguments

`object`	An object returned by the function `survivalSL`.
`method`	A character string with the name of the algorithm included in the SL for which the summary is performed. The default is "sl" for the Super Learner.
`newdata`	An optional data frame containing the new sample for validation with covariate values, follow-up times, and event status. The default value is `NULL`, the summary is performed on the same subjects of the training sample.
`ROC.precision`	An optional argument with the percentiles (between 0 and 1) of the prognostic variable used for computing each point of the time dependent ROC curve. 0 (min) and 1 (max) are not allowed. By default, the precision is `seq(.01,.99,.01)`.
`digits`	An optional integer for the number of digits to print when printing numeric values.
`pro.time`	This optional value of prognostic time represents the maximum delay for which the capacity of the variable is evaluated. The same unit than the one used in the argument times. Not used for the following metrics: "ll", "ibs", and "ibll". Default value is the time at which half of the subjects are still at risk.
`...`	Additional arguments affecting the summary which are passed from `libsl` by default.

Details

The following metrics can be used : "bs" for the Brier score at the prognostic time `pro.time`, "p_ci" and "uno_ci" for the concordance index at the prognostic time `pro.time` (Pencina and Uno versions), "ll" for the log-likelihood, "ibs" for the integrated Brier score up to the last observed time in the training data, "ibll" for the integrated binomial log-likelihood up to the last observed time in the training data, "ribs" for the restricted integrated Brier score up to the prognostic time `pro.time`, "ribll" for the restricted integrated binomial log-likelihood up to the prognostic time `pro.time`, "bll" for the binomial log-likelihood, "auc" for the area under the time-dependent ROC curve up to the prognostic time `pro.time`.

Value

`metrics`	A data frame containing the computed predictive performance metrics.
`library`	The name of the library used for model training.
`pro.time`	The prognostic time used for evaluation.
`ROC.precision`	The precision values used for the ROC curve computation.

See Also

`survivalSL`.

Examples

```
#data("dataDIVAT2")

#formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

#sl1 <- survivalSL(formula, method=c("LIB_AFTgamma", "LIB_PHgompertz"
#,"LIB_AFTllogis"),  metric="auc",
  #data=dataDIVAT2[1:100,],
  #pro.time = 12,
  #cv=3)

# The prognostic capacities from the same training sample
#summary(sl1)
```

`survivalSL` *Super Learner for Censored Outcomes*

Description

This function allows to compute a Super Learner (SL) to predict survival outcomes.

Usage

```
survivalSL(formula, data, methods, metric="auc", penalty=NULL,
cv=10, param.tune=NULL, pro.time=NULL,
optim.local.min=FALSE, ROC.precision=seq(.01,.99,.01),
param.weights.fix=NULL, param.weights.init=NULL,
seed=NULL, optim.method="Nelder-Mead", maxit=1000,
show_progress=TRUE)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame whose columns correspond to the variables present in the formula.

`methods`	A vector of characters with the names of the algorithms included in the SL. At least two algorithms have to be included.
`metric`	The loss function or metric used to estimate the weights of the algorithms in the SL. See details.
`penalty`	A numerical vector that allows the integration of covariates into the final model after selection (It concerns `"LIB_COXaic"`.) or/and allows the covariates not to be penalized (It concerns : `"LIB_COXen"` `"LIB_COXlasso"` and `"LIB_COXridge"`.). We give the value 0 if we want to force the covariate in the model or/and not to be penalized otherwise 1. If `NULL`, all covariates undergo the selection process or/and penalization process.
`cv`	The number of splits for cross-validation. The default value is 10.
`param.tune`	A list with a length equals to the number of algorithms included in `methods`. If `NULL`, the tunning parameters are estimated (see details).
`pro.time`	This optional value of prognostic time represents the maximum delay for which the capacity of the variable is evaluated. The same unit than the one used in the argument times. Not used for the following metrics: "ll", "ibs", and "ibll". Default value is the time at which half of the subjects are still at risk.
`optim.local.min`	An optional logical value. If `TRUE`, the optimization is performed twice to better ensure the estimation of the weights. If `FALSE` (default value), the optimization is performed once.
`ROC.precision`	The percentiles (between 0 and 1) of the prognostic variable used for computing each point of the time dependent ROC curve. Only used when `metric="auc"`. 0 (min) and 1 (max) are not allowed. By default: `seq(.01,.99,.01)`.
`param.weights.fix`	A vector with the parameters of the multinomial logistic regression which generates the weights of the algorithms declared in `methods`. When completed, the related parameters are not estimated. The default value is NULL: the parameters are estimated by a `cv`-fold cross-validation. See details.
`param.weights.init`	A vector with the initial values of the parameters of the multinomial logistic regression which generates the weights of the algorithms declared in `methods`. The default value is NULL: the initial values are equaled to 0. See details.
`seed`	A random seed to ensure reproducibility. If `NULL`, a seed is randomly assigned.
`optim.method`	The optimization method used to estimate the weights. It can be either `"SANN"` or `"Nelder-Mead"`. By default we use `Nelder-Mead`.
`maxit`	The number of iterations during the weight optimization process. By default, it is set to 1000.
`show_progress`	Parameter to display the progress bar. By default, it is set to `TRUE`.

Details

Each object of the list declared in `param.tune` must have the same name than the names of the `methods` included in the SL. If `param.tune = NULL`, `survivalSL` has already predefined default grids of tunning parameters for each algorithm in this case. The final tunning parameters are chosen thanks to cv-fold cross-validation (except for `LIB_RSF`, which uses the Out Of Bag observations to select the best hyperparameters based on the optimal value of the chosen metric). The following metrics can be used : "bs" for the Brier score at the prognostic time `pro.time`, "p_ci" and "uno_ci" for the concordance index at the prognostic time `pro.time` (Pencina and Uno versions), "ll" for the log-likelihood, "ibs" for the integrated Brier score up to the last observed time in the training data, "ibll" for the integrated binomial log-likelihood up to the last observed time in the training data, "ribs" for the restricted integrated Brier score up to the prognostic time `pro.time`, "ribll" for the restricted integrated binomial log-likelihood up to the prognostic time `pro.time`, "bll" for the binomial log-likelihood, "auc" for the area under the time-dependent ROC curve up to the prognostic time `pro.time`.

The following learners are available:

Names	Description	Package
`"LIB_AFTgamma"`	Gamma-distributed AFT model	flexsurv
`"LIB_AFTggamma"`	Generalized Gamma-distributed AFT model	flexsurv
`"LIB_AFTweibull"`	Weibull-distributed AFT model	flexsurv
`"LIB_PHexponential"`	Exponential-distributed PH model	flexsurv
`"LIB_PHgompertz"`	Gompertz-distributed PH model	flexsurv
`"LIB_PHspline"`	Spline-based PH model	flexsurv
`"LIB_COXall"`	Usual Cox model	survival
`"LIB_COXaic"`	Cox model with AIC-based forward selection	MASS
`"LIB_COXen"`	Elastic Net Cox model	glmnet
`"LIB_COXlasso"`	Lasso Cox model	glmnet
`"LIB_COXridge"`	Ridge Cox model	glmnet
`"LIB_RSF"`	Survival Random Forest	randomForestSRC
`"LIB_PLANN"`	Survival Neural Network	survivalPLANN

The following loss functions for the estimation of the super learner weigths are available (`metric`):

- Area under the ROC curve (`"auc"`)
- Pencina concordance index (`"p_ci"`)
- Uno concordance index (`"uno_ci"`)
- Brier score (`"bs"`)
- Binomial log-likelihood (`"bll"`)
- Integrated Brier score (`"ibs"`)
- Integrated binomial log-likelihood (`"ibll"`)
- Restricted integrated Brier score (`"ribs"`)
- Restricted integrated binomial log-Likelihood (`"ribll"`)
- Log-Likelihood (`"ll"`)

Value

`times`	A vector of numeric values with the times of the `predictions`.
`predictions`	It corresponds to a matrix with the survival predictions related to the SL.

`FitALL`	It corresponds to a list of matrix with the survival predictions related to each of the learner used for the SL construction.
`formula`	The formula object used for the SL construction.
`data`	The data frame used for learning.
`ROC.precision`	The percentiles (between 0 and 1) of the prognostic variable used for computing each point of the time dependent ROC curve.
`cv`	The number of splits for cross-validation.
`methods`	A vector of characters with the names of the algorithms included in the SL.
`pro.time`	The maximum delay for which the capacity of the variable is evaluated.
`models`	A list with the estimated models/algorithms included in the SL.
`weights`	A list composed by two vectors: the regressions `coefficients` of the logistic multinomial regression and the resulting weights' `values`.
`metric`	A list composed by two vectors: the loss function used to estimate the weights of the algorithms in the SL and its cross validation value.
`param.tune`	The estimated tunning parameters.
`seed`	The random seed used.
`optim.method`	The optimization method used.

References

Polley E and van der Laanet M. Super Learner In Prediction. http://biostats.bepress.com. 2010.

Examples

```
data("dataDIVAT2")

# The Super Learner based from the first 200 individuals of the data base

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

sl1 <- survivalSL(formula=formula, data=dataDIVAT2[1:200,],
                  methods=c("LIB_AFTgamma", "LIB_PHgompertz"))

# Individual prediction
pred <- predict(sl1, newdata=data.frame(age=c(52,52), hla=c(0,1),
retransplant=c(1,1), ecd=c(0,1)))

plot(y=pred$predictions$sl[1,], x=pred$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))

lines(y=pred$predictions$sl[2,], x=pred$times, col=2, type="l", lty=1, lwd=2)

legend("topright", col=c(1,2), lty=1, lwd=2, c("Subject #1", "Subject #2"))
```

`tuneCOXen` *Tune Elastic Net Cox Regression*

Description

This function finds the optimal lambda and alpha parameters for an elastic net Cox regression.

Usage

```
tuneCOXen(formula, data, penalty=NULL,
 cv=10, parallel=FALSE, alpha=seq(.1,.9,.1), lambda=NULL, seed=NULL)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame for training the model with the same covariates as in the formula.
`penalty`	A numerical vector that allows the covariates not to be penalized. We give the value 0 if we do not want the covariate to be penalized otherwise 1. If `NULL`, all covariates are penalized.
`cv`	The value of the number of folds. The default value is 10.
`parallel`	If `TRUE`, use parallel `foreach` to fit each fold. The default is `FALSE`.
`alpha`	The values of the regularization parameter alpha optimized over.
`lambda`	The values of the regularization parameter lambda optimized over.
`seed`	A random seed to ensure reproducibility during the cv process. If `NULL`, a seed is randomly assigned.

Details

The function runs the `cv.glmnet` function of the `glmnet` package.

Value

`optimal`	The values of lambda and alpha that gives the minimum cross-validated deviance.
`results`	The data frame with the cross-validated deviance for each lambda and alpha values.

References

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
tune.model <- tuneCOXen(formula=formula, data=dataDIVAT2, cv=5,
  alpha=seq(.1, 1, by=.1), lambda=seq(.1, 1, by=.1))

tune.model$optimal$lambda # the estimated lambda value

# The estimation of the training modelwith the corresponding lambda value
model <- LIB_COXen(formula, data=dataDIVAT2,
  alpha=tune.model$optimal$alpha,
  lambda=tune.model$optimal$lambda)

# The resulted predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`tuneCOXlasso` *Tune Lasso Cox Regression*

Description

This function finds the optimal lambda parameter for a Lasso Cox regression.

Usage

```
tuneCOXlasso(formula, data, penalty=NULL,
 cv=10, parallel=FALSE, lambda=NULL, seed=NULL)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame for training the model with the same covariates as in the formula.
`penalty`	A numerical vector that allows the covariates not to be penalized. We give the value 0 if we do not want the covariate to be penalized otherwise 1. If `NULL`, all covariates are penalized.
`cv`	The value of the number of folds. The default value is 10.
`parallel`	If `TRUE`, use parallel `foreach` to fit each fold. The default is `FALSE`.
`lambda`	The values of the regularization parameter lambda optimized over.
`seed`	A random seed to ensure reproducibility during the cv process. If `NULL`, a seed is randomly assigned.

Details

The function runs the `cv.glmnet` function of the `glmnet` package.

Value

`optimal`	The value of lambda that gives the minimum cross-validated deviance.
`results`	The data frame with the cross-validated deviance for each lambda value.

References

Simon et al. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

tune.model <- tuneCOXlasso(formula=formula, data=dataDIVAT2,
  cv=5, lambda=seq(0, 10, by=.1))

tune.model$optimal$lambda # the estimated lambda value

# The estimation of the training modelwith the corresponding lambda value
model <- LIB_COXlasso(formula, data=dataDIVAT2,
  lambda=tune.model$optimal$lambda)

# The resulted predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`tuneCOXridge` *Tune Ridge Cox Regression*

Description

This function finds the optimal lambda parameter for a ridge Cox regression.

Usage

```
tuneCOXridge(formula, data, penalty=NULL,
 cv=10, parallel=FALSE, lambda=NULL, seed=NULL)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame for training the model with the same covariates as in the formula.
`penalty`	A numerical vector that allows the covariates not to be penalized. We give the value 0 if we do not want the covariate to be penalized otherwise 1. If `NULL`, all covariates are penalized.
`cv`	The value of the number of folds. The default value is 10.
`parallel`	If `TRUE`, use parallel `foreach` to fit each fold. The default is `FALSE`.
`lambda`	The values of the regularization parameter lambda optimized over.
`seed`	A random seed to ensure reproducibility during the cv process. If `NULL`, a seed is randomly assigned.

Details

The function runs the `cv.glmnet` function of the `glmnet` package.

Value

`optimal`	The value of lambda that gives the minimum cross-validated deviance.
`results`	The data frame with the cross-validated deviance for each lambda value.

References

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011) Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, Journal of Statistical Software, Vol. 39(5), 1-13, https://www.jstatsoft.org/v39/i05/

Examples

```
data("dataDIVAT2")

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

tune.model <- tuneCOXridge(formula=formula, data=dataDIVAT2,
  cv=5, lambda=seq(0, 10, by=.1))

tune.model$optimal$lambda # the estimated lambda value

# The estimation of the training modelwith the corresponding lambda value
model <- LIB_COXridge(formula, data=dataDIVAT2,
  lambda=tune.model$optimal$lambda)

# The resulted predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

`tunePHspline` *Tune a Survival Regression using the Royston/Parmar Spline Model*

Description

This function finds the optimal number of knots of the spline function.

Usage

```
tunePHspline(formula,
data, cv=10, metric="auc", k=1:4, pro.time=NULL,
seed=NULL, ROC.precision=seq(.01, .99, by=.01))
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame for training the model with the same covariates as in the formula.
`cv`	The value of the number of folds. The default value is 10.
`metric`	The loss function or metric. See details. Default metric is Area Under ROC (`"auc"`).
`k`	The number of knots optimized over.
`pro.time`	This optional value of prognostic time represents the maximum delay for which the capacity of the variable is evaluated. The same unit than the one used in the argument times. Not used for the following metrics: "ll", "ibs", and "ibll". Default value is the time at which half of the subjects are still at risk.
`seed`	A random seed to ensure reproducibility during the cv process. If `NULL`, a seed is randomly assigned.
`ROC.precision`	The percentiles (between 0 and 1) of the prognostic variable used for computing each point of the time dependent ROC curve. Only used when `metric="auc"`. 0 (min) and 1 (max) are not allowed. By default: `seq(.01,.99,.01)`.

Details

The function runs the `flexsurvspline` function of the `flexsurv` package. The following metrics can be used : "bs" for the Brier score at the prognostic time `pro.time`, "p_ci" and "uno_ci" for the concordance index at the prognostic time `pro.time` (Pencina and Uno versions), "ll" for the log-likelihood, "ibs" for the integrated Brier score up to the last observed time in the training data, "ibll" for the integrated binomial log-likelihood up to the last observed time in the training data, "ribs" for the restricted integrated Brier score up to the prognostic time `pro.time`, "ribll" for the restricted integrated binomial log-likelihood up to the prognostic time `pro.time`, "bll" for the binomial log-likelihood, "auc" for the area under the time-dependent ROC curve up to the prognostic time `pro.time`.

Value

`optimal`	The value of `k` that optimizes the cross-validated value of the metric/loss function.
`results`	The data frame with the cross-validated value of the metric/loss function according to `k`.

References

Royston, P. and Parmar, M. (2002). Flexible parametric proportional-hazards and proportional odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. Statistics in Medicine 21(1):2175-2197. doi: 10.1002/sim.1203

Examples

```
data("dataDIVAT2")

# The estimation of the hyperparameters on the first 150 patients

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
tune.model <- tunePHspline(formula=formula,
data=dataDIVAT2[1:150,], cv=3, metric="auc",
k=1:2, pro.time=NULL,seed=123,
ROC.precision=seq(.01, .99, by=.02))

# the estimated nodesize value

 tune.model$optimal
 tune.model$results
```

`tunePLANN` *Tune a Survival Neural Network Based on the PLANN Method*

Description

This function finds the optimal inter, size, decay, maxit, and MaxNWts parameters for the survival neural network.

Usage

```
tunePLANN(formula, data, cv=10, inter=1, size=c(2, 4, 6, 8, 10),
decay=c(0.001, 0.01, 0.02, 0.05), maxit=100, MaxNWts=10000, maxtime=NULL,
seed=NULL,metric="auc", pro.time=NULL,
ROC.precision=seq(.01, .99, by=.01))
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame for training the model with the same covariates as in the formula.
`cv`	The value of the number of folds. The default value is 10.
`metric`	The loss function or metric. See details. Default metric is Area Under ROC (`"auc"`).
`inter`	The length of the intervals.
`size`	The number of units in the hidden layer.
`decay`	The parameter for weight decay.
`maxit`	The maximum number of iterations.
`MaxNWts`	The maximum allowable number of weights.
`maxtime`	A numeric value with the maximum prognostic. If `NULL`, the maximum prognostic time is the highest time observed in the data + 1.
`pro.time`	This optional value of prognostic time represents the maximum delay for which the capacity of the variable is evaluated. The same unit than the one used in the argument times. Not used for the following metrics: "ll", "ibs", and "ibll". Default value is the time at which half of the subjects are still at risk.
`seed`	A random seed to ensure reproducibility during the cv process. If `NULL`, a seed is randomly assigned.
`ROC.precision`	The percentiles (between 0 and 1) of the prognostic variable used for computing each point of the time dependent ROC curve. Only used when `metric="auc"`. 0 (min) and 1 (max) are not allowed. By default: `seq(.01,.99,.01)`.

Details

The function runs the `flexsurvspline` function of the `flexsurv` package. The following metrics can be used : "bs" for the Brier score at the prognostic time `pro.time`, "p_ci" and "uno_ci" for the concordance index at the prognostic time `pro.time` (Pencina and Uno versions), "ll" for the log-likelihood, "ibs" for the integrated Brier score up to the last observed time in the training data, "ibll" for the integrated binomial log-likelihood up to the last observed time in the training data, "ribs" for the restricted integrated Brier score up to the prognostic time `pro.time`, "ribll" for the restricted integrated binomial log-likelihood up to the prognostic time `pro.time`, "bll" for the binomial log-likelihood, "auc" for the area under the time-dependent ROC curve up to the prognostic time `pro.time`.

Value

`optimal`	The value of `inter`, `size`, `decay`, `maxit`, and `MaxNWts` that optimizes the cross-validated value of the metric/loss function.
`results`	The data frame with the cross-validated value of the metric/loss function according to `inter`, `size`, `decay`, `maxit`, and `MaxNWts`.

References

Biganzoli E, Boracchi P, Mariani L, and et al. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. Stat Med, 17:1169-86, 1998.

Examples

```
data("dataDIVAT2")

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd
tune.model <- tunePLANN(formula=formula, data=dataDIVAT2[1:150,],
cv=5, inter=c(2,1), size=c(16,32), decay=0.01, maxit=100, MaxNWts=1000,
maxtime=NULL,seed=123,metric="auc",
pro.time=NULL,ROC.precision=seq(.01, .99, by=.01))

tune.model$optimal # the optimal hyperparameters

tune.model$results
```

`tuneRSF` *Tune a Survival Random Forest*

Description

This function finds the optimal nodesize, mtry, and ntree parameters for a survival random forest tree.

Usage

```
tuneRSF(formula, data, nodesize=c(2, 4, 6, 10, 20, 30, 50, 100),
   mtry, ntree=500, seed=NULL)
```

Arguments

`formula`	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	A data frame for training the model with the same covariates as in the formula.
`nodesize`	The values of the node size optimized over.
`mtry`	The numbers of variables randomly sampled as candidates at each split optimized over.
`ntree`	The numbers of trees optimized over.
`seed`	A random seed to ensure reproducibility during the bootstrapping process. If `NULL`, a seed is randomly assigned.

Details

The function runs the `tune.rfsrc` function of the `randomForestSRC` package.

Value

`optimal`	The value of lambda that gives the minimum mean cross-validated error.
`results`	The data frame with the mean cross-validated errors for each lambda values.

References

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Examples

```
data("dataDIVAT2")

formula<-Surv(times,failures) ~ age + hla + retransplant + ecd

tune.model <- tuneRSF(formula, data=dataDIVAT2,
  nodesize=c(100, 250, 500), mtry=1, ntree=100)

tune.model$optimal # the estimated nodesize value

# The estimation of the training modelwith the corresponding lambda value
model <- LIB_RSF(formula, data=dataDIVAT2,
  nodesize=tune.model$optimal$nodesize, mtry=1, ntree=100)

# The resulted predicted survival of the first subject of the training sample
plot(y=model$predictions[1,], x=model$times, xlab="Time (years)",
ylab="Predicted survival", col=1, type="l", lty=1, lwd=2, ylim=c(0,1))
```

Index