Package ‘dynamicLM’

July 6, 2023

Type Package

Title Dynamic w-year risk predictions from landmark time points

Version 0.3.0

Maintainer Anya Fries <afries@stanford.edu>

Description
The goal of dynamicLM is to provide a simple framework to make dynamic w-year risk predictions from landmark time points, allowing for competing risks and left and right censored data.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Depends dynpred (>= 0.1.2),
prodlim (>= 2019.11.13),
R (>= 2.10),
riskRegression (>= 2022.03.22),
survival (>= 2.44.1)

Imports data.table,
graphics,
stats,
utils

Suggests msm (>= 1.6.9),
pec (>= 2021.10.11)

LazyData true

R topics documented:

`add_interactions` *Add landmarking time interactions to a super dataset*

Description

The stacked dataset output is used as input to `dynamic_lm()` to fit a landmark supermodel for dynamic prediction.

Usage

```
add_interactions(lmdata, lm_covs, func_covars, func_lms, lm_col, keep = T)
```

Arguments

`lmdata`	An object of class "LMdataframe". This can be created by running `stack_data()`, or creating a stacked data set and storing it in a list with attributes outcome, w and end_time (see `stack_data()` for further description of outcome and w), end_time is the largest landmarking time.
`lm_covs`	Vector of strings indicating the columns (covariates) that are to have an interaction with the landmark times.
`func_covars`	Either a string (or vector of strings) specifying which covariate(x)-landmark(t) interactions to include. One or multiple of "linear" (`x, x*t`), "quadratic" (`x, x*t^2`), "log" (`x, log(1 + x)`), or or "exp" (`x, exp(x)`). Otherwise, a custom list of functions can be specified. For example, `list( function(t) t, function(t) exp(20*t))` will, for each covariate `x`, create `x, x*t, exp(20*t)`.
`func_lms`	Similar to `func_covars`: A list of functions to use for transformations of the landmark times. Either a string or vector of strings or a custom list of functions.
`lm_col`	Character string specifying the column name that indicates the landmark time point for a row. Obtained from `lmdata` if not input.
`keep`	Boolean value to indicate whether or not to keep the columns given by `lm_covs` without the time interactions. Default is TRUE.

Details

For each variable "var" in `lm_covs`, new columns var_1,...,var_i (length(func_covars) == i) are added; one column for each interaction given in func_covars is added.

Transformations of the LM column are added and labelled as LM_1,...,LM_j (length(func_lms) == j); one column for each interaction given in func_lms is added.

Value

An object of class "LMdataframe" which now also contains LM time-interactions. The object has the following components:

- w, outcome: as the input (obtained from lmdata)
- func_covars: as the input
- func_lms: as the input
- lm_covs: as the input
- all_covs: a list of the new columns added. This includes `lm_covs` if `keep` is TRUE.
- lm_col: as the input

Examples

```
## Not run:
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("age.at.time.0", "male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
# Choose covariates that will have time interaction
pred_covars <- c("age", "male", "stage", "bmi", "treatment")
# Stack landmark datasets
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")
# Update complex landmark-varying covariates
# note age is in years and LM is in months
lmdata$data$age <- lmdata$data$age.at.time.0 + lmdata$data$LM/12
# Add LM-time interactions
lmdata <- add_interactions(lmdata, pred_covars,
                           func_covars = c("linear", "quadratic"),
                           func_lms = c("linear", "quadratic"))
head(lmdata$data)

## End(Not run)
```

`calplot` *Calibration plots for dynamic risk prediction landmark models.*

Description

There are three ways to perform calibration: apparent/internal, bootstrapped, and external. Accordingly, the named list of prediction models must be as follows:

- For both apparent/internal calbration, objects output from `predict.dynamicLM()` for supermodels fit with `dynamic_lm()` may be used as input.
- In order to bootstrap, supermodels fit with `dynamic_lm()` may be used as input (note that the argument `x=TRUE` must be specified when fitting the model in `dynamic_lm()`).
- For external calibration, supermodels fit with `dynamic_lm()` are input along with new data in the `data` argument. This data can be a LMdataframe or a dataframe (in which case `lms` must be specified).

Usage

```
calplot(
  object,
  times,
  formula,
  data,
  lms,
  id_col = "ID",
  split.method = "none",
  B = 1,
  M,
  cores = 1,
  seed,
  regression_values = FALSE,
  cause,
  plot = T,
  main,
  sub = F,
  ...
)
```

Arguments

`object`	A named list of prediction models, where allowed entries are outputs from `predict.dynamicLM()` or supermodels from `dynamic_lm()` depending on the type of calibration.
`times`	Landmark times for which calibration must be plot. These must be a subset of landmark times used during the prediction
`formula`	A survival or event history formula (`Hist(...)`). The left If none is given, it is obtained from the prediction object.
`data`	Data for external validation. This can be an object of class LMdataframe (i.e., created by calling `stack_data()` and `add_interactions()`), or a data.frame. If it is a data.frame, argument `lms` must be specified.
`lms`	Landmark times corresponding to the patient entries in data. Only required if `data` is specified and is a dataframe. `lms` can be a string (indicating a column in data), a vector of length nrow(data), or a single value if all patient entries were obtained at the same landmark time.

`id_col`	Column name that identifies individuals in data. If omitted, it is obtained from the prediction object.
`split.method`	Defines the internal validation design as in `pec::calPlot()`. Options are currently "none" or "bootcv". "none": assess the model in the test data (`data` argument)/data it was "bootcv": `B` models are trained on bootstrap samples either drawn with size `M`. Models are then assessed in observations not in the sample.
`B`	Number of times bootstrapping is performed.
`M`	Subsample size for training in cross-validation. Entries not sampled
`cores`	To perform parallel computing, specifies the number of cores. (Not yet implemented)
`seed`	Optional, integer passed to set.seed. If not given or NA, no seed
`regression_values`	Default is FALSE. If set to TRUE, the returned list is appended by another list `regression_values`, which contains the intercept and slope of a linear regression of each model for each landmark time (i.e., each calibration plot). Note that perfect calibration has a slope of 1 and an intercept of 0.
`cause`	Cause of interest if considering competing risks. If left blank, this is inferred from object.
`plot`	If FALSE, do not plot the results, just return a plottable object. Default is TRUE.
`main`	Optional title to override default.
`sub`	If TRUE, add a subheading with the number of individuals at risk, Default is FALSE
`...`	Additional arguments to pass to calPlot (`pec` package). These arguments have been included for user flexibility but have not been tested and should be used with precaution.

Details

For both internal calibration and bootstrapping, it is assumed that all models in `object` are fit on the same data.

When collecting bootstrap samples, the same individuals are considered across landmarks. I.e., sample `M` unique individuals, train on the super dataset formed by these individuals, and validate on the individuals not sampled at the landmarks they remain alive (or that are given in `times`).

Note that only complete cases of data are considered (whatever type of calibration is performed).

A comment on the following message: "Dropping bootstrap b = X for model name due to unreliable predictions". As certain approximations are made, numerical overflow sometimes occurs in predictions for bootstrapped samples. To avoid potential errors, the whole bootstrap sample is dropped in this case. Note that input data should be complete otherwise this may occur unintentionally. Calibration plots are still produced excluding predictions made during the bootstrap resampling.

Value

List of plots of w-year risk, one entry per prediction/landmark time point. List has a component `$regression_values` (if argument regression_values is set to TRUE) which is a list of which contains the intercept and slope of a linear regression of each model for each landmark time (i.e., each calibration plot).

Examples

```
## Not run:
# Internal validation
par(mfrow=c(1,2),pty="s")
outlist <- calplot(list("Model_1" = supermodel),
                   times = c(0, 6),             # landmark times at which to plot
                   method = "quantile", q = 10, # method for calibration plot
                   regression_values = TRUE,    # output regression values
                   ylim = c(0, 0.4), xlim = c(0, 0.4)) # optional
outlist$regression_values

# Bootstrapping
# Remember to fit the supermodel with argument 'x = TRUE'
par(mfrow=c(1,2),pty="s")
outlist = calplot(list("Model_1" = supermodel),
                  times = c(0, 6),
                  method = "quantile", q=10,
                  split.method = "bootcv", B = 10, # 10 bootstraps
                  ylim = c(0, 0.4), xlim = c(0, 0.4))

# External validation
# Either input an object from predict as the object or a supermodel and
# "data" & "lms" argument
newdata <- relapse[relapse$T_txgiven == 0, ]
newdata$age <- newdata$age.at.time.0
newdata$LM <- 0
par(mfrow = c(1,1))
cal <- calplot(list("CSC" = supermodel), cause = 1, data = newdata, lms = "LM",
               method = "quantile", q = 10, ylim = c(0, 0.1), xlim = c(0, 0.1))

## End(Not run)
```

coef.dynamicLM *Get the coefficients of a fitted supermodel in dynamicLM*

Description

Get the coefficients of a fitted supermodel in dynamicLM

Usage

```
## S3 method for class 'dynamicLM'
coef(object, ...)
```

Arguments

`object`	Fitted supermodel
`...`	Other arguments to pass to `stats::coef()`

Value

Vector of coefficients for a Cox landmark supermodel or list of coefficients for each cause-specific model for a CSC landmark supermodel.

dynamic_lm — *Fit a coxph or CSC model to a landmark super dataset, i.e., fit a dynamic landmark supermodel*

Description

dynamic (dyn) landmark (lm) supermodel –> dynamic_lm

Usage

```
dynamic_lm(
  lmdata,
  formula,
  type = "coxph",
  method = "breslow",
  func_covars,
  func_lms,
  lm_col,
  outcome,
  w,
  lm_covs,
  cluster,
  x = FALSE,
  ...
)
```

Arguments

`lmdata`	An object of class "LMdataframe", this can be created by running `stack_data()` and `add_interactions()`
`formula`	The formula to be used, remember to include "+cluster(ID)" for the column that indicates the ID of the individual for robust error estimates. Note that transformations (e.g., `x1*x2`) cannot be used in the formula and factors/categorical variables must first be made into dummy variables.
`type`	"coxph" or "CSC"/"CauseSpecificCox"
`method`	A character string specifying the method for tie handling. Default is "breslow". More information can be found in coxph.
`func_covars`	A list of functions to use for interactions between LMs and covariates.
`func_lms`	A list of functions to use for transformations of the landmark times.
`lm_col`	Character string specifying the column name that indicates the landmark time point for a row.
`outcome`	List with items time and status, containing character strings identifying the names of time and status variables, respectively, of the survival outcome
`w`	Scalar, the value of the prediction window (ie predict w-year/other time period risk from the LM points)
`lm_covs`	Vector of strings indicating the columns that are to have a LM interaction
`cluster`	Variable which clusters the observations (for e.g., identifies repeated patient IDs), for the purposes of a robust variance.

x	Logical value. If set to true, the `lmdata` is stored in the returned object. This is required for internal validation.
...	Arguments given to coxph or CSC.

Value

An object of class "LMcoxph" or "LMCSC" with components:

- model: fitted model
- type: as input
- w, func_covars, func_lms, lm_covs, all_covs, outcome: as in `lmdata`
- LHS: the LHS of the input formula
- linear.predictors: the vector of linear predictors, one per subject. Note that this vector has not been centered.
- args: arguments used to call model fitting
- id_col: the cluster argument, often specifies the column with patient ID
- lm_col: column name that indicates the landmark time point for a row.

Examples

```
## Not run:
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("age.at.time.0", "male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
# Choose covariates that will have time interaction
pred_covars <- c("age", "male", "stage", "bmi", "treatment")
# Stack landmark datasets
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")

# Update complex landmark-varying covariates
# note age is in years and LM is in months
lmdata$data$age <- lmdata$data$age.at.time.0 + lmdata$data$LM/12
# Add LM-time interactions
lmdata <- add_interactions(lmdata, pred_covars,
                           func_covars = c("linear", "quadratic"),
                           func_lms = c("linear", "quadratic"))

formula <- "Hist(Time, event, LM) ~ age + male + stage + bmi + treatment +
           age_1 + age_2 + male_1 + male_2 + stage_1 + stage_2 + bmi_1 +
           bmi_2 + treatment_1 + treatment_2 + LM_1 + LM_2 + cluster(ID)"
supermodel <- dynamic_lm(lmdata, as.formula(formula), "CSC")
print(supermodel)

par(mfrow = c(2,3))
plot(supermodel)

## End(Not run)
```

`get_lm_data` *Build a landmark dataset*

Description

Build a landmark dataset

Usage

```
get_lm_data(
  data,
  outcome,
  lm,
  horizon,
  covs,
  format = c("wide", "long"),
  id,
  rtime,
  right = TRUE
)
```

Arguments

`data`	Data frame from which to construct landmark super dataset
`outcome`	A list with items time and status, containing character strings identifying the names of time and status variables, respectively, of the survival outcome
`lm`	The value of the landmark time point at which to construct the landmark dataset.
`horizon`	Scalar, the value of the prediction window (ie predict risk within time w landmark points)
`covs`	A list with items fixed and varying, containing character strings specifying column names in the data containing time-fixed and time-varying covariates, respectively.
`format`	Character string specifying whether the original data are in wide (default) or in long format.
`id`	Character string specifying the column name in data containing the subject id.
`rtime`	Character string specifying the column name in data containing the (running) time variable associated with the time-varying variables; only needed if format = "long".
`right`	Boolean (default = FALSE), indicating if the intervals for the time-varying covariates are closed on the right (and open on the left) or vice-versa.

Details

This function is based from `dynpred::cutLM()` with minor changes. The original function was authored by Hein Putter.

Value

A landmark dataset.

References

van Houwelingen HC, Putter H (2012). Dynamic Prediction in Clinical Survival Analysis. Chapman & Hall.

---

plot.dynamicLM *Plots the dynamic log-hazard ratio of a cox or CSC supermodel*

Description

Plots the dynamic log-hazard ratio of a cox or CSC supermodel

Usage

```
## S3 method for class 'dynamicLM'
plot(
  x,
  covars,
  conf_int = TRUE,
  cause,
  end_time,
  logHR = TRUE,
  extend = FALSE,
  silence = FALSE,
  xlab = "LM time",
  ylab,
  ylim,
  main,
  ...
)
```

Arguments

`x`	An object of class "LMcoxph" or "LMCSC", i.e. a fitted supermodel
`covars`	Vector or list of strings indicating the variables to plot (note these must be given without time interaction label, for e.g., as in the argument `lm_covs` in `add_interactions()`).
`conf_int`	Include confidence intervals or not, default is TRUE
`cause`	Cause of interest if considering competing risks
`end_time`	Final time point to plot HR, defaults to the last landmark point used in model fitting.
`logHR`	Boolean, if true plots the log of the hazard ratio, if false plots the hazard ratio. Default is TRUE.
`extend`	Argument to allow for HR to be plot at landmark times that are later than the LMs used in model fitting. Default is FALSE. If set to TRUE, the HR may be unreliable.
`silence`	silence the warning message when end_time > LMs used in fitting the model
`xlab`	x label for the plots
`ylab`	y label for the plots

`ylim`	y limit for the plots
`main`	Vector of strings indicating the title of each plot. Must be in the same order as covars.
`...`	Additional arguments passed to plot

Details

See our GitHub for example code

Value

Plots for each variable in covars showing the dynamic hazard ratio

---

`plot.LMcalibrationPlot` *Plot an object output from `calplot()`: plot the calibration plots.*

Description

Plot an object output from `calplot()`: plot the calibration plots.

Usage

```
## S3 method for class 'LMcalibrationPlot'
plot(x, ...)
```

Arguments

`x`	An object of class "LMcalibrationPlot" output from `calplot()`
`...`	Other arguments to pass to pass to plot

`plot.LMScore` *Plot an object output from `score()`: plot the landmark and time-dependent Brier and/or AUC of dynamic landmark supermodels.*

Description

Plot an object output from `score()`: plot the landmark and time-dependent Brier and/or AUC of dynamic landmark supermodels.

Usage

```
## S3 method for class 'LMScore'
plot(x, metrics, se = TRUE, xlab, ylab, legend, pch, ylim, xlim, ...)
```

Arguments

`x`	An object of class "LMScore" output from `score()`
`metrics`	One or both of "auc" and "brier"
`se`	Boolean, default TRUE. To include point wise confidence intervals.
`xlab, ylab, pch, ylim, xlim`	graphical parameters
`legend`	Location of legend
`...`	Additional arguments to `plot`

`plotrisk` *Plots the absolute risk of individuals for different LM points for an event of interest within a given window*

Description

Plots the absolute risk of individuals for different LM points for an event of interest within a given window

Usage

```
plotrisk(
  object,
  data,
  format,
  lm_col,
  id_col,
  w,
  cause,
  varying,
  end_time,
  extend = F,
  silence = F,
  pch,
  lty,
  lwd,
  col,
  main,
  xlab,
  ylab,
  xlim,
  ylim,
  x.legend,
  y.legend,
  ...
)
```

Arguments

`object`	Fitted landmark supermodel
`data`	Data frame of individuals from which to plot risk
`format`	Character string specifying whether the data are in wide (default) or in long format
`lm_col`	Character string specifying the column name in data containing the (running) time variable associated with the time-varying covariate(s); only needed if format="long"
`id_col`	Character string specifying the column name in data containing the subject id; only needed if format="long"
`w`	Prediction window, i.e., predict w-year (/month/..) risk from each of the tLMs. Defaults to the w used in model fitting. If w > than that used in model fitting, results are unreliable, but can be produced by setting extend=T.
`cause`	The cause we are looking at if considering competing risks
`varying`	Character string specifying column name in the data containing time-varying covariates; only needed if format="wide"
`end_time`	Final time point to plot risk
`extend`	Argument to allow for risk to be plot at landmark times that are later than the landmarks used in model fitting. Default is FALSE. If set to TRUE, risks may be unreliable.
`silence`	Silence the message when end_time > landmarks used in fitting the model
`pch`	Passed to points
`lty`	Vector with line style
`lwd`	Vector with line widths
`col`	Vector with colors
`main`	Title for the plot
`xlab`	Label for x-axis
`ylab`	Label for y-axis
`xlim`	Limits for the x-axis
`ylim`	Limits for the y-axis
`x.legend, y.legend`	The x and y co-ordinates to be used to position the legend. They can be specified by keyword or in any way which is accepted by xy.coords.
`...`	Additional arguments passed to plot

Details

See our GitHub for example code

Value

Single plot of the absolute w-year risk of individuals

predict.dynamicLM *Calculate w-year risk from a landmark time point*

Description

Calculate w-year risk from a landmark time point

Usage

```
## S3 method for class 'dynamicLM'
predict(
  object,
  newdata,
  lms,
  cause,
  w,
  extend = F,
  silence = F,
  complete = T,
  ...
)
```

Arguments

`object`	Fitted landmark supermodel
`newdata`	Either a dataframe of individuals to make predictions for or an object of class LMdataframe (e.g., created by calling `stack_data()` and `add_interactions()`). If it is a dataframe, it must contain the original covariates (i.e., without landmark interaction).
`lms`	landmark time points that correspond to the entries in `newdata`. Only required when `newdata` is a data.frame. `lms` is either a time point, a vector or character string. • For a single time point, w-year risk is predicted from this time for each data point. • For a vector, `lms` must have the same length as the number of rows of `newdata` (i.e., each data point is associated with one LM/prediction time point). • A character string indicates a column in `newdata`.
`cause`	Cause of interest for competing risks.
`w`	Prediction window, i.e., predict w-year (/month/..) risk from each of the `lms`. Defaults to the `w` used in model fitting. If `w` > than that used in model fitting, results are unreliable, but can be produced by setting `extend = T`.
`extend`	Argument to allow for predictions at landmark times that are later than those used in model fitting, or prediction windows greater than the one used in model fitting. Default is FALSE. If set to TRUE, predictions may be unreliable.
`silence`	Silence the warning message when extend is set to TRUE.
`complete`	Only make predictions for data entries with non-NA entries (i.e., non-NA predictions). Default is TRUE.
`...`	Unused

Value

An object of class "LMpred" with components:

- preds: a dataframe with columns LM and risk, each entry corresponds to one individual and prediction time point (landmark)
- w, type, LHS: as in the fitted super model
- data: the newdata given in input

References

van Houwelingen HC, Putter H (2012). Dynamic Prediction in Clinical Survival Analysis. Chapman & Hall.

Examples

```
## Not run:
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("age.at.time.0", "male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
# Choose covariates that will have time interaction
pred_covars <- c("age", "male", "stage", "bmi", "treatment")
# Stack landmark datasets
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")

# Update complex landmark-varying covariates
# note age is in years and LM is in months
lmdata$data$age <- lmdata$data$age.at.time.0 + lmdata$data$LM/12
# Add LM-time interactions
lmdata <- add_interactions(lmdata, pred_covars,
                           func_covars = c("linear", "quadratic"),
                           func_lms = c("linear", "quadratic"))

formula <- "Hist(Time, event, LM) ~ age + male + stage + bmi + treatment +
           age_1 + age_2 + male_1 + male_2 + stage_1 + stage_2 + bmi_1 +
           bmi_2 + treatment_1 + treatment_2 + LM_1 + LM_2 + cluster(ID)"
supermodel <- dynamic_lm(lmdata, as.formula(formula), "CSC")

p1 <- predict(supermodel)
head(p1$preds)

## End(Not run)
```

`print.LMcoxph` *Print function for object of class LMcoxph*

Description

Print function for object of class LMcoxph

Usage

```
## S3 method for class 'LMcoxph'
print(x, verbose = FALSE, ...)
```

Arguments

`x`	Object of class LMcoxph
`verbose`	Boolean, default is FALSE. Print further components.
`...`	Arguments passed to print.

Value

Printed output.

`print.LMCSC` *Print function for object of class LMCSC*

Description

Print function for object of class LMCSC

Usage

```
## S3 method for class 'LMCSC'
print(x, verbose = FALSE, cause, ...)
```

Arguments

`x`	Object of class LMCSC
`verbose`	Boolean, default is FALSE. Print further components.
`cause`	Print the model for a given cause. If left out, all models are printed.
`...`	Arguments passed to print.

Value

Printed output.

`print.LMdataframe` *Print function for object of class LMdataframe*

Description

Print function for object of class LMdataframe

Usage

```
## S3 method for class 'LMdataframe'
print(x, verbose = FALSE, ...)
```

Arguments

`x`	Object of class LMdataframe
`verbose`	Boolean, default is FALSE. Print further components.
`...`	Arguments passed to print.

Value

Printed output.

`print.LMpred` *Print function for object of class LMpred*

Description

Print function for object of class LMpred

Usage

```
## S3 method for class 'LMpred'
print(x, verbose = FALSE, ...)
```

Arguments

`x`	Object of class LMpred
`verbose`	Boolean, default is FALSE. Print further components.
`...`	Arguments passed to print.

Value

Printed output.

`print.LMScore` *Print function for object of class LMScore, i.e., output from* `score()`

Description

Print function for object of class LMScore, i.e., output from `score()`

Usage

```
## S3 method for class 'LMScore'
print(x, digits = 3, ...)
```

Arguments

`x`	Object of class LMScore
`digits`	Number of significant digits to include
`...`	Arguments passed to print.

Value

Printed output.

`relapse` *Time-to-event data of cancer relapse*

Description

Simple synthetic dataset containing the time-to-event of cancer relapse (event=1) with the competing risk in long-form with patient information.

Usage

```
relapse
```

Format

A data frame with 989 rows and 9 columns:

ID Patient ID

Time Time-to-event

event Event of interest (0=censoring, 1=relapse, 2,3=competing risks)

age.at.time.0 Patient's age at time of diagnosis

male Sex of patient, 1=male, 0=female

stage Cancer stage at diagnosis

bmi Patient's body mass index at diagnosis

treatment Patient's treatment status, treatment = 1 = on treatment, treament = 0 = patient is off treatment

T_txgiven Follow-up time, i.e., time at which updated treatment (tx) information was provided, which is equivalent to the time point at which the patient entry was created.

`riskScore` *Calcutes dynamic risk score at a time for an individual (helper to predict.dynamicLM)*

Description

Calcutes dynamic risk score at a time for an individual (helper to predict.dynamicLM)

Usage

```
riskScore(object, tLM, data, func_covars, func_lms)
```

Arguments

`object`	A coxph object
`tLM`	Landmarking time point at which to calculate risk score (time at which the prediction is made)
`data`	Dataframe (single row) of individual. Must contain the original covariates.
`func_covars`	A list of functions to use for interactions between LMs and covariates.
`func_lms`	A list of functions to use for transformations of the landmark times.

Value

Numeric risk score

`score` *Methods (time-dependent AUC and Brier Score) to score the predictive performance of dynamic risk prediction landmark models.*

Description

There are three ways to perform assess the predictive performance: apparent/internal, bootstrapped, and external. Accordingly, the named list of prediction models must be as follows:

- For both apparent/internal evaluation, objects output from `predict.dynamicLM()` or supermodels fit with `dynamic_lm()` may be used as input.
- In order to bootstrap, supermodels fit with `dynamic_lm()` may be used as input (note that the argument `x=TRUE` must be specified when fitting the model in `dynamic_lm()`).
- For external calibration, supermodels fit with `dynamic_lm()` are input along with new data in the `data` argument. This data can be a LMdataframe or a dataframe (in which case `lms` must be specified).

Usage

```
score(
  object,
  times,
  metrics = c("auc", "brier"),
  formula,
  data,
  lms = "LM",
  id_col = "ID",
  se.fit = TRUE,
  conf.int = 0.95,
  split.method = "none",
  B = 1,
  M,
  cores = 1,
  seed,
  cause,
  silent = T,
  na.rm = FALSE,
  ...
)
```

Arguments

`object`	A named list of prediction models, where allowed entries are outputs from `predict.dynamicLM()` or supermodels from `dynamic_lm()` depending on the type of calibration.
`times`	Landmark times for which calibration must be plot. These must be a subset of landmark times used during the prediction
`metrics`	Character vector specifying which metrics to apply. Choices are "auc" and "brier". Case matters.
`formula`	A survival or event history formula (`prodlim::Hist()`). The left hand side is used to compute the expected event status. If none is given, it is obtained from the prediction object.
`data`	Data for external validation.
`lms`	Landmark times corresponding to the patient entries in data. Only required if data is specified and is a dataframe. `lms` can be a string (indicating a column in data), a vector of length nrow(data), or a single value if all patient entries were obtained at the same landmark time.
`id_col`	Column name that identifies individuals in data. If omitted, it is obtained from the prediction object.
`se.fit`	If FALSE or 0, no standard errors are calculated.
`conf.int`	Confidence interval (CI) coverage. Default is 0.95. If bootstrapping, CIs are calculated from empirical quantiles. If not, for right censored data, they are calculated by the package riskRegression as in Blanche et al (references).
`split.method`	Defines the internal validation design. Options are currently "none" or "bootcv". "none": assess the model in the test data (`data` argument)/data it was trained on. "bootcv": `B` models are trained on boostrap samples either drawn with replacement of the same size as the original data or without replacement of size `M`. Models are then assessed in observations not in the sample.

`B`	Number of times bootstrapping is performed.
`M`	Subsample size for training in cross-validation. Entries not sampled in the M subsamples are used for validation.
`cores`	To perform parallel computing, specifies the number of cores. (Not yet implemented)
`seed`	Optional, integer passed to set.seed. If not given or NA, no seed is set.
`cause`	Cause of interest if considering competing risks. If left blank, this is inferred from object.
`silent`	Show any error messages when computing `score` for each landmark time (and potentially bootstrap iteration)
`na.rm`	Ignore bootstraps where there are errors (for example not enough datasamples) and calculate metrics on remaining values. This is not recommended. For example, if only one bootstrap sampling has enough data that live to the prediction window, the standard error will be zero.
`...`	Additional arguments to pass to `riskRegression::Score()`. These arguments have been included for user flexibility but have not been tested and should be used with precaution.

Details

For both internal evaluation and bootstrapping, it is assumed that all models in `object` are fit on the same data.

If data at late evaluation times is sparse, certain bootstrap samples may not have patients that live long enough to perform evaluation leading to the message "Upper limit of followup in bootstrap samples, was too low. Results at evaluation time(s) beyond these points could not be computed and are left as NA". In this case, consider only evaluating for earlier landmarks or performing prediction with a smaller window as data points are slim. If you wish to see which model/bootstrap/landmark times failed, set SILENT=FALSE. Set na.rm = TRUE ignores these bootstraps and calculate metrics from the bootstrap samples that worked (not recommended).

Another message may occur: "Dropping bootstrap b = X for model name due to unreliable predictions". As certain approximations are made, numerical overflow sometimes occurs in predictions for bootstrapped samples. To avoid potential errors, the whole bootstrap sample is dropped in this case. Note that input data should be complete otherwise this may occur unintentionally.

Value

An object of class "LMScore", which has components:

- `auct`: dataframe containing time-dependent AUC if "auc" was included as a metric
- `briert`: dataframe containing time-dependent Brier score if "brier" was included as a metric

References

Paul Blanche, Cecile Proust-Lima, Lucie Loubere, Claudine Berr, Jean- Francois Dartigues, and Helene Jacqmin-Gadda. Quantifying and comparing dynamic predictive accuracy of joint models for longitudinal marker and time-to-event in presence of censoring and competing risks. Biometrics, 71 (1):102–113, 2015.

P. Blanche, J-F Dartigues, and H. Jacqmin-Gadda. Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks. Statistics in Medicine, 32(30):5381–5397, 2013.

Examples

```
## Not run:
# Internal validation
scores <- score(list("Model1" = supermodel),
                times = c(0, 6)) # landmarks at which to provide calibration plots
scores

# Bootstrapping
# Remember to fit the supermodel with argument 'x = TRUE'
scores <- score(list("Model1" = supermodel),
                times = c(0, 6),
                split.method = "bootcv", B = 10) # 10 bootstraps
scores

par(mfrow=c(1,2))
plot(scores)

# External validation
# Either input an object from predict as the object or a supermodel and
# "data" & "lms" argument
newdata <- relapse[relapse$T_txgiven == 0, ]
newdata$age <- newdata$age.at.time.0
newdata$LM <- 0
score(list("CSC" = supermodel), cause = 1, data = newdata, lms = "LM")

## End(Not run)
```

`splc` *Time-to-event data of SPLC*

Description

Synthetic dataset containing the time-to-event of secondary primary lung cancer (SPLC) with competing risks of lung cancer death (cause 2) and other-cause death (cause 3) in long-form with patient information.

Usage

```
splc
```

Format

A data frame with 875 rows and 23 columns:

ID Patient ID

event Event of interest (0=censoring, 1=relapse, 2,3=competing risks)

Time Time-to-event

T.fup Follow-up time, i.e., time at which updated covariate information was provided. This is equivalent to the time point at which the patient entry was created.

age.ix Patient's age at time of diagnosis

male Sex of patient, 1 = male, 0 = female

fh Family history

ph Prior history

bmi Patient's body mass index at diagnosis

stage.ix Cancer stage at diagnosis (advanced/not)

surgery.ix Surgery (yes/no)

radiation.ix Radiation (yes/no)

chemo.ix Chemotherapy (yes/no)

smkstatus Smoking status. Former = 2, Current = 3

cigday Cigarettes per day.

packyears Number of pack years

quityears Number of quit years

hist_* Histology at diagnosis

`splc_test` *Time-to-event data of SPLC (test set)*

Description

Synthetic dataset containing the time-to-event of secondary primary lung cancer (SPLC) with competing risks of lung cancer death (cause 2) and other-cause death (cause 3) in long-form with patient information.

Usage

```
splc_test
```

Format

A data frame with 607 rows and 24 columns:

ID Patient ID

event Event of interest (0=censoring, 1=relapse, 2,3=competing risks)

Time Time-to-event

T.fup Follow-up time, i.e., time at which updated covariate information was provided. This is equivalent to the time point at which the patient entry was created.

age.ix Patient's age at time of diagnosis

male Sex of patient, 1 = male, 0 = female

fh Family history

ph Prior history

bmi Patient's body mass index at diagnosis

stage.ix Cancer stage at diagnosis (advanced/not)

surgery.ix Surgery (yes/no)

radiation.ix Radiation (yes/no)

chemo.ix Chemotherapy (yes/no)

smkstatus Smoking status. Former = 2, Current = 3

cigday Cigarettes per day.

packyears Number of pack years

quityears Number of quit years

hist_* Histology at diagnosis

stack_data *Build a stacked dataset from original dataset (wide or long format).*

Description

This stacked dataset output is used as input to `dynamic_lm()` to fit a landmark supermodel for dynamic prediction. Calling `add_interactions()` on the output before fitting the supermodel allows for landmark time interactions to be included.

Usage

```
stack_data(
  data,
  outcome,
  lms,
  w,
  covs,
  format = c("wide", "long"),
  id,
  rtime,
  right = FALSE
)
```

Arguments

`data`	Data frame from which to construct landmark super dataset
`outcome`	A list with items time and status, containing character strings identifying the names of time and status variables, respectively, of the survival outcome
`lms`	vector, the value of the landmark time points. This should be a range of points over the interval that prediction will be made. For example, if 5-year risk predictions are to be made over the first three years, this could be `c(0, 1.5, 3)`, `c(0, 1, 2, 3)` etc.
`w`	Scalar, the value of the prediction window (ie predict risk within time w landmark points)
`covs`	A list with items fixed and varying, containing character strings specifying column names in the data containing time-fixed and time-varying covariates, respectively.
`format`	Character string specifying whether the original data are in wide (default) or in long format.
`id`	Character string specifying the column name in data containing the subject id.

`rtime`	Character string specifying the column name in data containing the (running) time variable associated with the time-varying variables; only needed if format = "long".
`right`	Boolean (default = FALSE), indicating if the intervals for the time-varying covariates are closed on the right (and open on the left) or vice-versa.

Value

An object of class "LMdataframe". This the following components:

- data: containing the stacked data set, i.e., the outcome and the values of time-fixed and time-varying covariates taken at the landmark time points. The value of the landmark time point is stored in column LM.
- outcome: same as input
- w: same as input
- end_time: final landmarking point used in training
- lm_col: "LM", identifies the landmark time column.

Examples

```
## Not run:
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("age.at.time.0", "male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
# Stack landmark datasets
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")
head(lmdata$data)

## End(Not run)
```

Index