Package 'dynamicLM'

December 23, 2024

Type Package

Title Dynamic w-year risk predictions from landmark time points

Version 1.0.0

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 7.3.2

Depends prodlim (>= 2019.11.13),
R (>= 2.10),
riskRegression (>= 2024.6.13),
survival (>= 2.44.1)

Imports data.table,
glmnet,
graphics,
stats,
utils

Suggests latex2exp,
msm (>= 1.6.9),
pec (>= 2021.10.11)

LazyData true

Contents

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 = c("linear", "quadratic"),
  func_lms = c("linear", "quadratic"),
  keep = TRUE
)
```

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/vector of strings or list of functions specifying which covariate-landmark interactions to include. If `x` are covariates and `t` are landmarks then "linear" (`x, x*t`), "quadratic" (`x, x*t^2`), "log" (`x, log(1 + x)`), or or "exp" (`x, exp(x)`) can be specified. A custom list of functions can be specified. For example, `list(function(t) t, function(t) exp(20*t))` will, for each covariate, create `x, x*t, exp(20*t)`.
`func_lms`	A list of functions to use for transformations of the landmark times input similarly to `func_covars`, either as a string/ vector of strings or a custom list of functions.
`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_LM1, ..., var_LMi are added; one column for each interaction given in func_covars is added (length(func_covars) == i).

Transformations of the LM column are added and labelled as LM1, ..., LMj; one column for each interaction given in func_lms is added (length(func_lms) == j).

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

See Also

`stack_data()`, `dynamic_lm()`

Examples

```
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)
```

`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 = TRUE,
  main,
  ...
)
```

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.
`...`	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).

See Also

`score()`, `pec::calPlot()`

Examples

```
## Not run:
# Internal validation
par(mfrow = c(2, 2), pty = "s")
outlist <- calplot(list("Model1" = supermodel),
                   method = "quantile", q = 5,  # 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(2, 2), pty = "s")
outlist <- calplot(list("Model1" = supermodel),
                   method = "quantile", q = 5,
                   split.method = "bootcv", B = 10, # 10 bootstraps
                   ylim = c(0, 0.4), xlim = c(0, 0.4))

# External validation
# a) newdata is a dataframe
newdata <- relapse[relapse$T_txgiven == 0, ]
newdata$age <- newdata$age.at.time.0
newdata$LM <- 0
par(mfrow = c(1, 1))
cal <- calplot(list("Model1" = supermodel), data = newdata, lms = "LM",
               method = "quantile", q = 5, ylim = c(0, 0.1), xlim = c(0, 0.1))

# b) newdata is a landmark dataset
par(mfrow = c(2, 2), pty = "s")
lmdata_new <- lmdata
cal <- calplot(list("Model1" = supermodel), data = lmdata_new,
               method = "quantile", q = 10, ylim = c(0, 0.4), xlim = c(0, 0.4))

## 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.

`CSC.fixed.coefs` *Altered code from `riskRegression` of the cause-specific Cox model to fit a CSC model with given coefficients.*

Description

Altered code from `riskRegression` of the cause-specific Cox model to fit a CSC model with given coefficients.

Usage

```
CSC.fixed.coefs(formula, data, cause, cause.specific.coefs, ...)
```

Arguments

`formula`	Formula to fit the model
`data`	Data on which to which
`cause`	Main cause of interest
`cause.specific.coefs`	Coefficients that each model should be fit with
`...`	Additional arguments to coxph.

Value

CSC model

References

- `riskRegression` package: https://cran.r-project.org/web/packages/riskRegression/index.html `cv.pen_lm` *Cross-validation for a penalized Cox or cause-specific Cox landmark supermodel*

Description

Fit by calling `glmnet::cv.glmnet()`. As in `cv.glmnet`, k-fold cross validation is performed. This produces a plot and returns optimal values for `lambda`, the penalization parameter. Input can be as typically done for `cv.glmnet` in the form of `x` and `y` which are a matrix and response object or with a landmark super dataset specifying the dependent columns in `y`.

Usage

```
cv.pen_lm(
  x,
  y,
  id_col,
  alpha = 1,
  nfolds = 10,
  type.measure = "deviance",
  seed = NULL,
  foldid = NULL,
  ...
)
```

Arguments

`x`	An "LMdataframe", which can be created by running `stack_data()` and `add_interactions()`.
`y`	Optional, a vector of column names of the data stored in `lmdata` that are to be used as dependent variables. If not specified, it is assumed that all non-response variables are the dependent variables.
`id_col`	Column name or index that identifies individuals in data. Used to ensure individuals appear in the same cross-validation sets.
`alpha`	The elastic net mixing parameter: Lies between 0 and 1. At 1, the penalty is the LASSO penalty, and at 0, the penalty is the ridge penalty. The default is 1.
`nfolds`	Number of folds in k-fold cross validation. Default is 10.
`type.measure`	Loss for cross-validation. Currently the only option is "deviance" which is the partial-likelihood for the Cox model. If using cause-specific Cox models, this is evaluated on each model separately.
`seed`	Set a seed.
`foldid`	Optional, specify which fold each individual is in.
`...`	Additional arguments to `cv.glmnet()`.

Value

An object of class cv.pen_lm. This is a list of cv.glmnet objects (one for each cause-specific Cox model or a list of length one for a regular Cox model). The object also has attributes `survival.type` (`competing.risk` or `survival`) and `lmdata` and `xcols` which store the inputs if given. Functions `print()` and `plot()` exist for the object. To make predictions, see `dynamic_lm.cv.pen_lm()`.

See Also

print.cv.pen_lm(), plot.cv.pen_lm(), dynamic_lm.cv.pen_lm()

Examples

```
## Not run:
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")
lmdata <- add_interactions(lmdata, func_covars = c("linear", "quadratic"),
                           func_lms = c("linear", "quadratic"))

# use all covariates
cv_model <- cv.pen_lm(lmdata, alpha = 1)
print(cv_model, all_causes = TRUE)

par(mfrow = c(1, 2))
plot(cv_model, all_causes = TRUE)

# only use a subset of covariates
cv_model1 <- cv.pen_lm(lmdata, y = c("male", "male_LM1", "male_LM2",
                                     "stage", "stage_LM1", "stage_LM2"))

## End(Not run)
```

---

dynamic_lm — *Fit a dynamic Cox or cause-specific Cox landmark supermodel with or without regularization.*

Description

To fit Cox or cause-specific Cox models without regularization see:

- dynamic_lm.LMdataframe() for use on a stacked landmark dataset
- dynamic_lm.data.frame() for use on a dataframe

To fit penalized Cox or cause-specific Cox models see:

- dynamic_lm.pen_lm() without cross-validation
- dynamic_lm.cv.pen_lm() with cross-validation

Usage

```
dynamic_lm(...)
```

Arguments

...	Arguments to pass to dynamic_lm()

Value

A fitted landmark supermodel object which has components:

- model: fitted model
- type: as input
- w, func_covars, func_lms, lm_covs, all_covs, outcome: as in `lmdata`
- LHS: the survival outcome
- linear.predictors: the vector of linear predictors, one per subject. Note that this vector has not been centered.

If the model is unpenalized (class "LMcoxph" or "LMCSC") it has additional components:

- 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.

If the model is penalized (class "penLMcoxph" or "penLMCSC") it has additional components:

- lambda: the values of lambda for which this model has been fit.

See Also

`dynamic_lm.LMdataframe()`, `dynamic_lm.data.frame()`, `dynamic_lm.pen_lm()`, `dynamic_lm.cv.pen_lm()`

`dynamic_lm.cv.pen_lm` *Fit a penalized cross-validated coxph or CSC super model*

Description

Use one value of `lambda` to fit a model from which predictions can be made.

Usage

```
## S3 method for class 'cv.pen_lm'
dynamic_lm(object, lambda = "lambda.min", x = FALSE, ...)
```

Arguments

`object`	A fitted object of class "cv.pen_lm". This can be created by calling `cv.pen_lm` using arguments `lmdata` and `xcols`.
`lambda`	Value of the penalty parameter `lambda` to fit a model. Default is "lambda.min"; "lambda.1se" can also be used or a specific value can be input. For cause-specific Cox super models, this must be a list or vector of values: one for each cause.
`x`	Logical value. If set to true, `lmdata` is stored in the returned object. This is required for internal validation.
`...`	Additional arguments to pass to `survival::coxph()` or `riskRegression::CSC()`

Details

The Breslow method is used for handling ties, as we use the `glmnet` package which does the same.

Value

An object of class "penLMcoxph" or "penLMCSC" 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.
- lambda: the values of lambda for which this model has been fit.

`dynamic_lm.data.frame` *Fit a dynamic Cox or cause-specific Cox landmark supermodel to a dataframe.*

Description

Note that it is recommended to rather use `stack_data()` and `add_interactions()` to create an object of class LMdataframe rather than directly calling `dynamic_lm()` on a dataframe to ensure the data has the correct form.

Usage

```
## S3 method for class 'data.frame'
dynamic_lm(
  lmdata,
  formula,
  type = "coxph",
  method = "breslow",
  func_covars,
  func_lms,
  lm_col,
  outcome,
  w,
  lm_covs,
  cluster,
  x = FALSE,
  ...
)
```

Arguments

`lmdata`	A dataframe that should be a stacked dataset across landmark times.
`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. See details for further information. 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 `survival::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. If omitted, extracted from `formula`.
`x`	Logical value. If set to true, `lmdata` is stored in the returned object. This is required for internal validation.
`...`	Arguments given to coxph or CSC.

Details

For standard survival data (one event and possible censoring), use `type = "coxph"` and a a formula with left-hand side (LHS) of the form `Surv(LM, Time, event)`. For competing risks (multiple events and possible censoring), use `type = "CSC"` and a LHS of the form `Hist(Time, event, LM)`

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 input.
- LHS: the survival outcome
- 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.

`dynamic_lm.formula` *Fit a dynamic Cox or cause-specific Cox landmark supermodel*

Description

Fit a dynamic Cox or cause-specific Cox landmark supermodel

Usage

```
## S3 method for class 'formula'
dynamic_lm(formula, lmdata, type, ...)
```

Arguments

`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. See details for further information. Note that transformations (e.g., `x1*x2`) cannot be used in the formula and factors/categorical variables must first be made into dummy variables.
`lmdata`	An object of class "LMdataframe", this can be created by running `stack_data()` and `add_interactions()`
`type`	"coxph" or "CSC"/"CauseSpecificCox"
`...`	Arguments given to coxph or CSC.

Details

For standard survival data (one event and possible censoring), use `type = "coxph"` and a a formula with left-hand side (LHS) of the form `Surv(LM, Time, event)`. For competing risks (multiple events and possible censoring), use `type = "CSC"` and a LHS of the form `Hist(Time, event, LM)`. This form is kept to ensure compatibility with the original dynamicLM library, although in later versions, the formula is the second argument.

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 survival outcome
- 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.

`dynamic_lm.LMdataframe` *Fit a dynamic Cox or cause-specific Cox landmark supermodel to a stacked landmark dataset*

Description

Fit a dynamic Cox or cause-specific Cox landmark supermodel to a stacked landmark dataset

Usage

```
## S3 method for class 'LMdataframe'
dynamic_lm(
  lmdata,
  formula,
  type = "coxph",
  method = "breslow",
  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. See details for further information. 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 `survival::coxph()`.
`cluster`	Variable which clusters the observations (for e.g., identifies repeated patient IDs), for the purposes of a robust variance. If omitted, extracted from `formula`.
`x`	Logical value. If set to true, `lmdata` is stored in the returned object. This is required for internal validation.
`...`	Arguments given to coxph or CSC.

Details

For standard survival data (one event and possible censoring), use `type = "coxph"` and a a formula with left-hand side (LHS) of the form `Surv(LM, Time, event)`. For competing risks (multiple events and possible censoring), use `type = "CSC"` and a LHS of the form `Hist(Time, event, LM)`

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 survival outcome
- 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

```
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")
lmdata <- add_interactions(lmdata, func_covars = c("linear", "quadratic"),
                           func_lms = c("linear", "quadratic"))

# for competing risk data (in this example)
formula <- "Hist(Time, event, LM) ~ male + male_LM1 + male_LM2 +
            stage + stage_LM1 + stage_LM2 + bmi + bmi_LM1 + bmi_LM2 +
            treatment + treatment_LM1 + treatment_LM2 + LM1 + LM2 + cluster(ID)"
supermodel <- dynamic_lm(lmdata, as.formula(formula), "CSC", x = TRUE)

#' \dontrun{
# for survival data
formula <- "Surv(LM, Time, event) ~
            age + age_LM1 + age_LM2 + male + male_LM1 + male_LM2 +
            stage + stage_LM1 + stage_LM2 + bmi + bmi_LM1 + bmi_LM2 +
            treatment + treatment_LM1 + treatment_LM2 + LM1 + LM2 + cluster(ID)"
supermodel <- dynamic_lm(lmdata, as.formula(formula), "coxph")
}

print(supermodel)

coef(supermodel)

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

---

dynamic_lm.pen_lm   *Fit a penalized coxph or CSC supermodel for a specific coefficient*

Description

Use one value of `lambda` to fit a model from which predictions can be made.

Usage

```
## S3 method for class 'pen_lm'
dynamic_lm(object, lambda, x = FALSE, ...)
```

Arguments

`object`	A fitted object of class "pen_lm". This can be created by calling `pen_lm()` using arguments `lmdata` and `xcols`.
`lambda`	Value of the penalty parameter `lambda` at which to fit a model. For cause-specific Cox super models, this must be a list or vector of values: one for each cause.

`x`	Logical value. If set to true, `lmdata` is stored in the returned object. This is required for internal validation.
`...`	Additional arguments to pass to `survival::coxph()` or `riskRegression::CSC()`

Details

The Breslow method is used for handling ties, as we use the `glmnet` package which does the same.

Value

An object of class "penLMcoxph" or "penLMCSC" with components:

- model: fitted model
- type: as input
- w, func_covars, func_lms, lm_covs, all_covs, outcome: as in lmdata.
- LHS: the survival outcome
- linear.predictors: the vector of linear predictors, one per subject. Note that this vector has not been centered.
- lambda: the values of lambda for which this model has been fit.
- LHS: the survival outcome
- args: arguments used to call model fitting
- pen_args: arguments used to call the penalized model
- 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.

`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,
  left.open = FALSE,
  split.data
)
```

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".
`left.open`	Boolean (default = FALSE), indicating if the intervals for the time-varying covariates are open on the left (and closed on the right) or vice-versa.
`split.data`	List of data split according to ID. Allows for faster computation.

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.
- The dynpred package (https://cran.r-project.org/web/packages/dynpred/index.html), in particular, the code for cutLM.

See Also

`stack_data()`

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"))
lm12 <- get_lm_data(relapse, outcome, lm = 12, horizon = 60, covs = covars,
                    format = "long", id = "ID", rtime = "T_txgiven")
head(lm12)
```

```
## End(Not run)
```

---

pen_lm — *Compute the regularization path of coefficients for a Cox or cause-specific Cox landmark supermodel with lasso or elasticnet penalization.*

Description

Fit by calling [glmnet::glmnet()]. As in glmnet, the model is fit via penalized maximum likelihood to produce a regularization path at a grid of values for the regularization parameter lambda. Input can be as typically done for glmnet in the form of x and y which are a matrix and response object or with a landmark super dataset specifying the dependent columns in y.

Usage

```
pen_lm(x, y, alpha = 1, ...)
```

Arguments

x	An "LMdataframe", which can be created by running stack_data() and add_interactions().
y	Optional, a vector of column names of the data stored in lmdata that are to be used as dependent variables. If not specified, it is assumed that all non-response variables are the dependent variables.
alpha	The elastic net mixing parameter: Lies between 0 and 1. At 1, the penalty is the LASSO penalty, and at 0, the penalty is the ridge penalty. The default is 1.
...	Additional arguments passed to glmnet().

Value

An object of class pen_lm. This is a list of glmnet objects (one for each cause-specific Cox model or a list of length one for a regular Cox model). The object also has attributes survival.type (competing.risk or survival) and lmdata and xcols which store the inputs if given. Functions print and plot exist for the object. To make predictions, see dynamic_lm.pen_lm() and predict.dynamicLM().

See Also

print.pen_lm(), plot.pen_lm(), dynamic_lm.pen_lm()

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")

# use all covariates
path <- pen_lm(lmdata, alpha = 0)
print(path, all_causes = TRUE)

par(mfrow = c(1, 2))
plot(path, all_causes = TRUE)

# only use a subset of covariates
path1 <- pen_lm(lmdata, y = c("male", "male_LM1", "male_LM2",
                              "stage", "stage_LM1", "stage_LM2"))

## End(Not run)
```

`plot.coefs` *Generic function to plot coefficients*

Description

Can plot positive and negative coefficients in two separate plots or the same. X-axes are the same if separate plots are used.

Usage

```
## S3 method for class 'coefs'
plot(
  x,
  single_plot = TRUE,
  max_coefs = NULL,
  col = "blue",
  xlab = "Coefficient value",
  ...
)
```

Arguments

`x`	(Named) Vector of coefficients
`single_plot`	Logical, defaults to TRUE. A single plot for both positive and negative coefficients, or two separate plots.
`max_coefs`	Default is to plot all coefficients. If specified, gives the maximum number of coefficients to plot.
`col`	Fill color for the barplot.
`xlab`	x-axis Label
`...`	Additional arguments to barplot.

`plot.cv.pen_lm` *Plot cross-validation curve created by `cv.pen_lm()`, analogous to plotting from `cv.glmnet()`*

Description

The cross-validation curve is plotted as a function of the lambda values used. Upper and lower standard deviation is plotted too.

Usage

```
## S3 method for class 'cv.pen_lm'
plot(
  x,
  all_causes = FALSE,
  silent = FALSE,
  label = FALSE,
  sign.lambda = 1,
  se.bands = TRUE,
  all_causes_title = TRUE,
  ...
)
```

Arguments

`x`	a fitted `cv.pen_lm()` object
`all_causes`	if `pen_lm()` fit a cause-specific Cox model, set TRUE to plot coefficient profile plots for each model.
`silent`	Set TRUE to hide messages.
`label`	Set TRUE to label the curves by variable index numbers.
`sign.lambda`	Plot against `log(lambda)` (default) or its negative if set to -1.
`se.bands`	Logical. If TRUE, shading is produced to show stand-error bands. Defaults to TRUE.
`all_causes_title`	If `all_causes` is set to TRUE, includes a title with the cause. Defaults to TRUE.
`...`	additional graphical parameters

Details

If the model is a survival model (i.e., no competing risks), then the output is the same as a call to `cv.glmnet` would produce. For competing risks, the default is only to plot the cross-validation curve for the cause of interest (first cause) Further events can be examined by setting `all_causes = TRUE`.

`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 = "Landmark time",
  ylab,
  ylim,
  main,
  ...
)
```

Arguments

`x`	A fitted supermodel
`covars`	Vector or list of strings indicating the variables to plot (note these must be given without time interaction).
`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
`discrete_grid`	Defaults to 0.1, how to discretize the grid for plotting

Details

See our GitHub for example code

Value

Plots for each variable in covars showing the dynamic hazard ratio

Examples

```
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")
lmdata <- add_interactions(lmdata, func_covars = c("linear", "quadratic"),
                           func_lms = c("linear", "quadratic"))
formula <- "Hist(Time, event, LM) ~ male + male_LM1 + male_LM2 +
            stage + stage_LM1 + stage_LM2 + bmi + bmi_LM1 + bmi_LM2 +
            treatment + treatment_LM1 + treatment_LM2 + LM1 + LM2 + cluster(ID)"
supermodel <- dynamic_lm(lmdata, as.formula(formula), "CSC", x = TRUE)

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

par(mfrow = c(1, 2))
plot(supermodel,
     covars = c("stage", "bmi"), # subset of covariates to plot
     logHR = FALSE,              # plot HR instead of log HR
     conf_int = FALSE,           # do not plot confidence intervals
     main = c("HR of stage", "HR of BMI"))
```

---

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, main, ...)
```

Arguments

`x`	An object of class "LMcalibrationPlot" output from `calplot()`
`main`	Optional title to override default.
`...`	Other arguments to pass to pass to plot

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

Description

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

Usage

```
## S3 method for class 'LMScore'
plot(
  x,
  metrics,
  contrasts = FALSE,
  landmarks = TRUE,
  summary = TRUE,
  se = TRUE,
  add_pairwise_contrasts = FALSE,
  cutoff_contrasts = 0.05,
  pairwise_heights,
  width,
  loc,
  xlab,
  ylab,
  pch,
  ylim,
  xlim,
  main,
  font.main = 1,
  col = NULL,
  cex = 1,
  length = 0.1,
  legend = TRUE,
  legend.title = NULL,
  auc = TRUE,
  brier = TRUE,
  ...
)
```

Arguments

`x`	An object of class "LMScore" output from `score()`
`metrics`	One or both of "AUC" and "Brier"
`contrasts`	Plot the difference between metrics. Default is FALSE and plots the metrics themselves.
`landmarks`	Plot time-dependent metrics. Default is TRUE.
`summary`	Plot the summary metric. Default is TRUE.

`se`	To include point wise confidence intervals. Default is TRUE.
`add_pairwise_contrasts`	If plotting summary metrics (`summary = TRUE`, `landmarks = FALSE`) set this argument TRUE to include the p-values of significant pairwise contrasts. In this case, arguments `pairwise_heights` and `width` must be set. The argument `cutoff_contrasts` is optional, specifying the significance cutoff.
`cutoff_contrasts`	If `add_pairwise_contrasts`, sets the signifance level of which tests are considered significant (numeric, default is 0.05).
`pairwise_heights`	If `add_pairwise_contrasts`, sets the height at which the p-values are plotted. Given as a vector of heights.
`width`	If `add_pairwise_contrasts`, the width of the ends of the contrast bars as a numeric value.
`loc`	Location for legend.
`xlab`, `ylab`, `pch`, `ylim`, `xlim`, `main`, `font.main`, `col`, `cex`	graphical parameters
`length`	The width of the ends of the error bars.
`legend`	Include a legend or not. Default is TRUE.
`legend.title`	Title of the legend. No title by default.
`auc`	Plot the AUC or not (if available). Default is TRUE.
`brier`	Plot the Brier Score or not (if available). Default is TRUE.
`...`	Additional arguments to `plot()`

plot.penLMcoxph — *Plot the non-zero coefficients of a penalized Cox landmark supermodel or the dynamic log-hazard ratios*

Description

Can plot positive and negative coefficients in two separate plots or the same. X-axes are the same if separate plots are used.

Usage

```
## S3 method for class 'penLMcoxph'
plot(
  x,
  single_plot = TRUE,
  max_coefs = NULL,
  col = "blue",
  xlab = "Coefficient value",
  HR = FALSE,
  covars = NULL,
  ...
)
```

Arguments

x	a penalized Cox supermodel - created by calling dynamic_lm() on an object created from pen_lm() or cv.pen_lm().
single_plot	Logical, defaults to TRUE. A single plot for both positive and negative coefficients, or two separate plots.
max_coefs	Default is to plot all coefficients. If specified, gives the maximum number of coefficients to plot.
col	Fill color for the barplot.
xlab	x-axis Label
HR	Plot the hazard ratio? Default is FALSE. See plot.dynamicLM() for additional arguments.
covars	If HR is TRUE, a vector or list of strings indicating the variables to plot (note these must be given without time interaction). Defaults to all non-zero variables.
...	Additional arguments to barplot or to plot.dynamicLM().

Details

If plotting the log hazard ratios, check plot.dynamicLM() to see further arguments.

---

plot.penLMCSC — *Plot the non-zero coefficients of a penalized cause-specific Cox landmark supermodel or the dynamic log-hazard ratios*

Description

Can plot positive and negative coefficients in two separate plots or the same. X-axes are the same if separate plots are used. If plotting the log hazard ratios, check plot.dynamicLM() to see further arguments.

Usage

```
## S3 method for class 'penLMCSC'
plot(
  x,
  single_plot = TRUE,
  max_coefs = NULL,
  all_causes = FALSE,
  HR = FALSE,
  covars = NULL,
  col = "blue",
  xlab = "Coefficient value",
  ...
)
```

Arguments

`x`	a penalized cause-specific Cox supermodel - created by calling `dynamic_lm()` on an object created from `pen_lm()` or `cv.pen_lm()`.
`single_plot`	Logical, defaults to TRUE. A single plot for both positive and negative coefficients, or two separate plots.
`max_coefs`	Default is to plot all coefficients. If specified, gives the maximum number of coefficients to plot.
`all_causes`	Logical, default is FALSE. Plot coefficients for all cause-specific models.
`HR`	Plot the hazard ratio? Default is FALSE. See `plot.dynamicLM()` for additional arguments.
`covars`	If HR is TRUE, a vector or list of strings indicating the variables to plot (note these must be given without time interaction). Defaults to all variables.
`col`	Fill color for the barplot.
`xlab`	x-axis Label
`...`	Additional arguments to barplot or to `plot.dynamicLM()`.

Details

If plotting the log hazard ratios, check `plot.dynamicLM()` to see further arguments.

`plot.pen_lm` *Plot the coefficient path created by calling `pen_lm()`, analogous to plotting from `glmnet()`*

Description

As in the glmnet package, produces a coefficient profile plot of the coefficient paths.

Usage

```
## S3 method for class 'pen_lm'
plot(
  x,
  xvar = "norm",
  all_causes = FALSE,
  silent = FALSE,
  label = FALSE,
  all_causes_title = TRUE,
  ...
)
```

Arguments

`x`	a fitted `pen_lm()` object
`xvar`	As in `glmnet()`: "What is on the X-axis. `"norm"` plots against the L1-norm of the coefficients, `"lambda"` against the log-lambda sequence, and `"dev"` against the percent deviance explained."

`all_causes`	if pen_lm fit a cause-specific Cox model, set TRUE to plot coefficient profile plots for each model.
`silent`	Set TRUE to hide messages.
`label`	Set TRUE to label the curves by variable index numbers.
`all_causes_title`	If `all_causes` is set to TRUE, includes a title with the cause. Defaults to TRUE.
`...`	additional graphical parameters

Details

If the model is a survival model (i.e., no competing risks), then the output is the same as a call to `glmnet` would produce. For competing risks, the default is only to plot the coefficient profile plot for the cause of interest (first cause) Further events can be examined by setting `all_causes = TRUE`.

`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 = FALSE,
  silence = FALSE,
  pch,
  lty,
  lwd,
  col,
  main,
  xlab,
  ylab,
  xlim,
  ylim = c(0, 1),
  x.legend,
  y.legend,
  las = 1,
  ...
)
```

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.
`las`	the style of axis labels
`...`	Additional arguments passed to plot

Details

See our GitHub for example code

Value

Single plot of the absolute w-year risk of individuals

Examples

```
data(relapse)

# select patients whose risk we want to plot
idx <- relapse$ID %in% c("ID1007", "ID1301")

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)

# Prediction time points (how smooth the plot will be)
x <- seq(0, 18, by = 1)

# Stack landmark datasets
dat <- stack_data(relapse[idx, ], outcome, x, w, covars, format = "long",
                  id = "ID", rtime = "T_txgiven")$data
dat$age <- dat$age.at.time.0 + dat$LM / 12 # age is in years and LM is in months
head(dat)

plotrisk(supermodel, dat, format = "long", ylim = c(0, 1), #0.7),
         x.legend = "bottomright")
```

`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 = FALSE,
  silence = FALSE,
  complete = TRUE,
  ...
)
```

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.
`...`	Unimplemented for now.

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.

See Also

`stack_data()`, `add_interactions()`, `dynamic_lm()`, `score()`, `calplot()`

Examples

```
data(relapse)
outcome <- list(time = "Time", status = "event")
covars <- list(fixed = c("male", "stage", "bmi"),
               varying = c("treatment"))
w <- 60; lms <- c(0, 6, 12, 18)
lmdata <- stack_data(relapse, outcome, lms, w, covars, format = "long",
                     id = "ID", rtime = "T_txgiven")
lmdata <- add_interactions(lmdata, func_covars = c("linear", "quadratic"),
                           func_lms = c("linear", "quadratic"))

formula <- "Hist(Time, event, LM) ~ male + male_LM1 + male_LM2 +
            stage + stage_LM1 + stage_LM2 + bmi + bmi_LM1 + bmi_LM2 +
            treatment + treatment_LM1 + treatment_LM2 + LM1 + LM2 + cluster(ID)"
supermodel <- dynamic_lm(lmdata, as.formula(formula), "CSC", x = TRUE)

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

`print.cv.pen_lm` *Print the output from calling cv.pen_lm()*, `Similar to printing the output of printing a` *cv.glmnet‘ object, print a cross-validated penalized cause-specific Cox supermodel*

Description

Print the output from calling cv.pen_lm(), `Similar to printing the output of printing a` cv.glmnet‘ object, print a cross-validated penalized cause-specific Cox supermodel

Usage

```
## S3 method for class 'cv.pen_lm'
print(x, all_causes = FALSE, silent = FALSE, digits = 3, ...)
```

Arguments

`x`	a cv.pen_lm object
`all_causes`	if `cv.pen_lm` fit a cause-specific Cox model, set TRUE to print a summary of the glmnet path for each model.
`silent`	Set TRUE to hide messages.
`digits`	Number of significant digits to include
`...`	additional print arguments

Details

If the model is a survival model (i.e., no competing risks), then the output is the same as a call to glmnet would produce. For competing risks, the default is only to print the output for the cause of interest (first cause). Further events can be examined by setting `all_causes = TRUE`.

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008). Regularization Paths for Generalized Linear Models via Coordinate Descent

`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, landmarks = TRUE, summary = TRUE, ...)
```

Arguments

`x`	Object of class LMScore
`digits`	Number of significant digits to include
`landmarks`	Print the time-dependent metrics at individual landmarks. Default is TRUE.
`summary`	Print the summary metrics of the models if they have been calculated. Default is TRUE.
`...`	Arguments passed to print.

Value

Printed output.

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

Description

Print function for object of class penLMcoxph

Usage

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

Arguments

`x`	Object of class penLMcoxph
`verbose`	Logical, if verbose print func_covars, func_lms, w, end_time and type.
`...`	Arguments passed to print.

Value

Printed output.

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

Description

Print function for object of class penLMCSC

Usage

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

Arguments

`x`	Object of class penLMCSC
`cause`	Print the model for a given cause. If left out, all models are printed.
`verbose`	Logical, if verbose print func_covars, func_lms, w, end_time and type.
`...`	Arguments passed to print.

Value

Printed output.

`print.pen_lm` *Print the output from calling* `pen_lm()`

Description

Similar to the output of printing a`glmnet` object, print a summary of the glmnet path at each step along the path.

Usage

```
## S3 method for class 'pen_lm'
print(x, all_causes = FALSE, silent = FALSE, digits = 3, ...)
```

Arguments

`x`	a pen_lm object
`all_causes`	if `pen_lm()` fit a cause-specific Cox model, set TRUE to print a summary of the glmnet path for each model.
`silent`	Set TRUE to hide messages.
`digits`	Number of significant digits to include
`...`	additional print arguments

Details

If the model is a survival model (i.e., no competing risks), then the output is the same as a call to glmnet would produce. For competing risks, the default is only to print the output for the cause of interest (first cause). Further events can be examined by setting `all_causes = TRUE`.

As in glmnet, "A three-column matrix with columns `Df`, `%Dev` and `Lambda` is printed. The `Df` column is the number of nonzero coefficients (Df is a reasonable name only for lasso fits). `%Dev` is the percent deviance explained (relative to the null deviance)."

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008). Regularization Paths for Generalized Linear Models via Coordinate Descent

---

`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,
  se.fit = TRUE,
  conf.int = 0.95,
  contrasts = TRUE,
  split.method = "none",
  B = 1,
  M,
  summary = TRUE,
  cores = 1,
  seed,
  cause,
  silent = TRUE,
  ...
)
```

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".
`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).
`contrasts`	If TRUE, perform model comparison tests.
`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.
`summary`	Compute the summary metrics (average of the time-dependent metrics). By default is TRUE.
`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)
`...`	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 landmark times is sparse, some 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. Currently ignores these bootstraps and calculates metrics from the bootstrap samples that worked.

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 list with entries `AUC` and `Brier` if "auc" and "brier" were included as metrics respectively and `AUC_Summary` and/or `Brier_summary` if `summary` is not null. Each will have entries:

- `score`: data.table containing the metric
- `contrasts`: data.table containing model comparisons

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 (using model)
scores <- score(list("Model1" = supermodel))
print(scores)

par(mfrow=c(1, 4))
plot(scores)

# Internal validation (using predictions)
p1 <- predict(supermodel)
scores <- score(list("Model1" = p1))
print(scores)


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

# External validation
# a) newdata is a dataframe
newdata <- relapse[relapse$T_txgiven == 0, ]
newdata$age <- newdata$age.at.time.0
newdata$LM <- 0
score(list("Model1" = supermodel), data = newdata, lms = "LM")

# b) newdata is a landmark dataset
lmdata_new <- lmdata
score(list("Model1" = supermodel), data = lmdata_new)

## 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,
  left.open = FALSE
)
```

Arguments

`data`	Data frame from which to construct landmark super dataset
`outcome`	A list with items time and status, containing 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. If missing, all columns that are not the outcome, rtime or id column are set to be time-varying covariates.
`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".
`left.open`	Boolean (default = FALSE), indicating if the intervals for the time-varying covariates are open on the left (and closed on the right) 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.

See Also

`add_interactions()`, `dynamic_lm()`

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)
```

`summary.cv.pen_lm` *Summarize cv.pen_lm objects: not yet implemented*

Description

Summarize cv.pen_lm objects: not yet implemented

Usage

```
## S3 method for class 'cv.pen_lm'
summary(object, ...)
```

Arguments

`object`	cv.pen_lm object
`...`	Additional arguments

`summary.dynamicLM` *Summarize dynamic_lm objects: not yet implemented*

Description

Summarize dynamic_lm objects: not yet implemented

Usage

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

Arguments

`object`	dynamicLM object
`...`	Additional arguments

`summary.pen_lm` *Summarize pen_lm objects: not yet implemented*

Description

Summarize pen_lm objects: not yet implemented

Usage

```
## S3 method for class 'pen_lm'
summary(object, ...)
```

Arguments

`object`	pen_lm object
`...`	Additional arguments

`summary_metric` *Obtain the summary metric from landmark-specific estimates*

Description

Obtain the summary metric from landmark-specific estimates

Usage

```
summary_metric(metric, df_t, df_c, df_iid, conf_int, object, id_col, B, se.fit)
```

Arguments

`metric`	"AUC" or "Brier"
`df_t`	time-dependent table of scores (AUC(s,t) or Brier(s,t)), i.e., a smaller table with entries for each landmark and model used for landmark-estimates and their standard error for example: tLM model times Brier se lower upper b 1: 0 Null model 59.9999 0.06302245 0.008262163 0.04682891 0.07921599 1 2: 0 dynamicLM 59.9999 0.06140585 0.007931126 0.04586112 0.07695057 1 3: 6 Null model 65.9999 0.08492919 0.011250007 0.06287958 0.10697880 1 4: 6 dynamicLM 65.9999 0.08126512 0.010849048 0.06000138 0.10252887 1 5: 12 Null model 71.9999 0.11151696 0.013936209 0.08420249 0.13883143 1 6: 12 dynamicLM 71.9999 0.10554550 0.013541907 0.07900385 0.13208715 1
`df_c`	table of contrast of scores. for example: tLM times model reference delta.Brier se lower upper p 1: 0 59.9999 dynamicLM Null model -0.001616602 0.00119091 -0.003950743 0.0007175397 0.1746381
`df_iid`	df_iid contains the iid decomposition of the score, it is a larger table with entries for each individual, who appear multiple times for different landmarks and different models for example: tLM ID model times IF.Brier b 1: 0 1 0 59.9999 -5.853094e-02 1 2: 0 2 0 59.9999 -8.262274e-05 1 3: 0 3 0 59.9999 -5.852535e-02 1 ...
`conf_int`	Coverage level of the confidence interval.
`object`	Either fitted supermodel or risk predictions.
`id_col`	Column name of the ID (only needed if we incorporate weighted metrics)
`B`	Number of bootstrap replicates
`se.fit`	If FALSE or 0, no standard errors are calculated.

Value

An list with entries `score` and optionally `contrasts` if contrasts were calculated.

Index