Package ‘Landmarking’

January 20, 2025

Title Analysis using Landmark Models

Version 1.0.0

Description The landmark approach allows survival predictions to be
updated dynamically as new measurements from an individual are recorded.
The idea is to set predefined time points, known as ``landmark times",
and form a model at each landmark time using only the individuals in the
risk set. This package allows the longitudinal data to be modelled
either using the last observation carried forward or linear mixed
effects modelling. There is also the option to model competing risks,
either through cause-specific Cox regression or Fine-Gray regression.
To find out more about the methods in this package, please see
<https://isobelbarrott.github.io/Landmarking/articles/Landmarking>.

License GPL (>= 2)

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

Imports nlme, riskRegression, dplyr, pec, methods, prodlim, stats,
survival, mstate, ggplot2,

RoxygenNote 7.1.2

Suggests testthat (>= 3.0.0), knitr, rmarkdown, JM

VignetteBuilder knitr

Depends R (>= 2.10)

URL https://github.com/isobelbarrott/Landmarking/

BugReports https://github.com/isobelbarrott/Landmarking/issues

Config/testthat/edition 3

NeedsCompilation no

Author Isobel Barrott [aut, cre],
Jessica Barrett [aut],
Ruth Keogh [ctb],
Michael Sweeting [ctb],
David Stevens [ctb]

Maintainer Isobel Barrott <isobel.barrott@gmail.com>

Repository CRAN

Date/Publication 2022-02-15 20:00:07 UTC

Contents

`add_cv_number` *Assign a k-fold cross-validation number*

Description

Randomly assigns a k-fold cross-validation number to each individual in a dataset.

Usage

```
add_cv_number(data_long, individual_id, k, seed = 1)
```

Arguments

`data_long`	Data frame in long format i.e. there may be more than one row per individual
`individual_id`	Character string specifying the column name in `data_long` which contains the individual identifiers
`k`	Integer specifying the number of folds for cross-validation. An alternative to setting parameter `cross_validation_df` for performing cross-validation; if both are missing no cross-validation is used.
`seed`	The value of the seed (default is 1)

Details

This function randomly divides the `n` individual IDs into `k` groups, each with `n/k` members (or as close to this number as possible).

Value

Data frame `data_long` updated to contain a new column `cross_validation_number` indicating the fold to which the individual has been assigned.

Author(s)

Isobel Barrott <`isobel.barrott@gmail.com`>

Examples

```
data(data_repeat_outcomes)
data_repeat_outcomes <- add_cv_number(data_long = data_repeat_outcomes,
                                      individual_id = "id",
                                      k = 10)
```

`data_repeat_outcomes` *Simulated repeat measurement and time-to-event data*

Description

A simulated dataset which is a combination of longitudinal (repeat measurement) data and time-to-event data.This dataset contains simulated data from 3000 patients.

The longitudinal (repeat measurement) data is formed using an LME model, whose parameters were based on CVD risk assessments recorded at primary care practices in New Zealand.

A LME model was fitted to this dataset and values of `sbp_stnd` and `tchdl_stnd` were estimated at landmark age 60. These values (along with the other baseline covariates) were used to simulate time-to-event data from a cause specific model with parameters based on CVD events of patients at primary care practices in New Zealand.

Usage

```
data_repeat_outcomes
```

Format

A dataset with 9048 rows and 14 columns:

id Patient ID

smoking Smoking status, 0 indicates the patient has never smoked, 1 indicates the patient has quit smoking, and 2 indicates the patient is a current smoker

diabetes Diabetes status, 0 indicates the patient is not diagnosed with diabetes, and 1 indicates the patient is diagnosed with diabetes

ethnicity Ethnicity, one of five ethnicities

deprivation Deprivation score, quintiles

index An index indicating assessment number for a patient

sbp_stnd Standardised systolic blood pressure

tchdl_stnd Standardised total cholesterol to HDL ratio

end_of_study_age Age that individual left the study, either the age at event (CVD or death) or age at end of study (1st Jan 2010)

response_time_tchdl_stnd Age that total cholesterol to HDL ratio was recorded

response_time_sbp_stnd Age that systolic blood pressure was recorded, this is the same as the date that the fixed measures were recorded

start_time Age the individual entered follow-up

event_status Event status, 0 indicates censoring, 1 indicates CVD event, and 2 indicates death from other causes

event_time Event time

`fit_LME_landmark` *Fit a landmarking model using a linear mixed effects (LME) model for the longitudinal submodel*

Description

This function performs the two-stage landmarking analysis.

Usage

```
fit_LME_landmark(
  data_long,
  x_L,
  x_hor,
  fixed_effects,
  random_effects,
  fixed_effects_time,
  random_effects_time,
  individual_id,
  k,
  cross_validation_df,
  random_slope_in_LME = TRUE,
  random_slope_as_covariate = TRUE,
  standardise_time = FALSE,
  lme_control = nlme::lmeControl(),
  event_time,
  event_status,
  survival_submodel = c("standard_cox", "cause_specific", "fine_gray"),
  b
)
```

Arguments

`data_long`	Data frame or list of data frames each corresponding to a landmark age `x_L` (each element of the list must be named the value of `x_L` it corresponds to). Each data frame contains repeat measurements data and time-to-event data in long format.
`x_L`	Numeric specifying the landmark time(s)
`x_hor`	Numeric specifying the horizon time(s)
`fixed_effects`	Vector of character strings specifying the column names in `data_long` which correspond to the fixed effects
`random_effects`	Vector of character strings specifying the column names in `data_long` which correspond to the random effects
`fixed_effects_time`	Character string specifying the column name in `data_long` which contains the time at which the fixed effects were recorded
`random_effects_time`	Vector of character strings specifying the column names in `data_long` which contain the times at which repeat measures were recorded. This should either be length 1 or the same length as `random_effects`. In the latter case the order of elements must correspond to the order of elements in `random_effects`.
`individual_id`	Character string specifying the column name in `data_long` which contains the individual identifiers
`k`	Integer specifying the number of folds for cross-validation. An alternative to setting parameter `cross_validation_df` for performing cross-validation; if both are missing no cross-validation is used.
`cross_validation_df`	List of data frames containing the cross-validation fold each individual is assigned to. Each data frame in the list should be named according to the landmark time `x_L` that they correspond. Each data frame should contain the columns `individual_id` and a column `cross_validation_number` which contains the cross-validation fold of the individual. An alternative to setting parameter `k` for performing cross-validation; if both are missing no cross-validation is used.
`random_slope_in_LME`	Boolean indicating whether to include a random slope in the LME model
`random_slope_as_covariate`	Boolean indicating whether to include the random slope estimate from the LME model as a covariate in the survival submodel.
`standardise_time`	Boolean indicating whether to standardise the time variables (`fixed_effects_time` and `random_effects_time`) by subtracting the mean and dividing by the standard deviation (see Details section for more information)
`lme_control`	Object created using `nlme::lmeControl()`, which will be passed to the `control` argument of the `lme` function
`event_time`	Character string specifying the column name in `data_long` which contains the event time

event_status	Character string specifying the column name in data_long which contains the event status (where 0=censoring, 1=event of interest, if there are competing events these are labelled 2 or above).
survival_submodel	Character string specifying which survival submodel to use. Three options: the standard Cox model i.e. no competing risks ("standard_cox"), the cause-specific regression model ("cause_specific"), or the Fine Gray regression model ("fine_gray")
b	Integer specifying the number of bootstrap samples to take when calcluating standard error of c-index and Brier score

Details

Firstly, this function selects the individuals in the risk set at the landmark time x_L. Specifically, the individuals in the risk set are those that have entered the study before the landmark age (there is at least one observation for each of the fixed_effects andrandom_effects on or before x_L) and exited the study on after the landmark age (event_time is greater than x_L).

Secondly, if the option to use cross validation is selected (using either parameter k or cross_validation_df), then an extra column cross_validation_number is added with the cross-validation folds. If parameter k is used, then the function add_cv_number randomly assigns these folds. For more details on this function see ?add_cv_number. If the parameter cross_validation_df is used, then the folds specified in this data frame are added. If cross-validation is not selected then the landmark model is fit to the entire group of individuals in the risk set (this is both the training and test dataset).

Thirdly, the landmark model is then fit to each of the training data. There are two parts to fitting the landmark model: using the longitudinal data and using the survival data. Using the longitudinal data is the first stage and is performed using fit_LME_longitudinal. See ?fit_LME_longitudinal more for information about this function. Using the survival data is the second stage and is performed using fit_survival_model. See ?fit_survival_model more for information about this function.

Fourthly, the performance of the model is then assessed on the set of predictions from the entire set of individuals in the risk set by calculating Brier score and C-index. This is performed using get_model_assessment. See ?get_model_assessment more for information about this function.

Value

List containing containing information about the landmark model at each of the landmark times. Each element of this list is named the corresponding landmark time, and is itself a list containing elements: data, model_longitudinal, model_LME, model_LME_standardise_time, model_survival, and prediction_error.

data has one row for each individual in the risk set at x_L and contains the value of the fixed_effects using the LOCF approach and predicted values of the random_effects using the LME model at the landmark time x_L. It also includes the predicted probability that the event of interest has occurred by time x_hor, labelled as "event_prediction". There is one row for each individual.

model_longitudinal indicates that the longitudinal approach is LME.

model_LME contains the output from the lme function from package nlme. For a model using cross-validation, model_LME contains a list of outputs with each element in the list corresponds to a different cross-validation fold. prediction_error contains a list indicating the c-index and Brier score at time `x_hor` and their standard errors if parameter `b` is used. For more information on how the prediction error is calculated please see `?get_model_assessment` which is the function used to do this within `fit_LME_landmark`.

`model_LME_standardise_time` contains a list of two objects `mean_response_time` and `sd_response_time` if the parameter `standardise_time=TRUE` is used. This is the mean and standard deviation use to normalise times when fitting the LME model.

`model_survival` contains the outputs from the survival submodel functions, including the estimated parameters of the model. For a model using cross-validation, `model_survival` will contain a list of outputs with each element in the list corresponding to a different cross-validation fold. `model_survival` contains the outputs from the function used to fit the survival submodel, including the estimated parameters of the model. For a model using cross-validation, `model_survival` contains a list of outputs with each element in the list corresponding to a different cross-validation fold. For more information on how the survival model is fitted please see `?fit_survival_model` which is the function used to do this within `fit_LME_landmark`.

`prediction_error` contains a list indicating the c-index and Brier score at time `x_hor` and their standard errors if parameter `b` is used.

Author(s)

Isobel Barrott <`isobel.barrott@gmail.com`>

Examples

```
library(Landmarking)
data(data_repeat_outcomes)
data_model_landmark_LME <-
  fit_LME_landmark(
    data_long = data_repeat_outcomes,
    x_L = c(60, 61),
    x_hor = c(65, 66),
    k = 10,
    fixed_effects = c("ethnicity", "smoking", "diabetes"),
    fixed_effects_time = "response_time_sbp_stnd",
    random_effects = c("sbp_stnd", "tchdl_stnd"),
    random_effects_time = c("response_time_sbp_stnd", "response_time_tchdl_stnd"),
    individual_id = "id",
    standardise_time = TRUE,
    lme_control = nlme::lmeControl(maxIter = 100, msMaxIter = 100),
    event_time = "event_time",
    event_status = "event_status",
    survival_submodel = "cause_specific"
  )
```

`fit_LME_longitudinal` *Fit a landmarking model using a linear mixed effects (LME) model for the longitudinal submodel*

Description

This function is a helper function for `fit_LME_landmark`.

Usage

```
fit_LME_longitudinal(
  data_long,
  x_L,
  fixed_effects,
  random_effects,
  fixed_effects_time,
  random_effects_time,
  standardise_time = FALSE,
  random_slope_in_LME = TRUE,
  random_slope_as_covariate = FALSE,
  cv_name = NA,
  individual_id,
  lme_control = nlme::lmeControl()
)
```

Arguments

`data_long`	Data frame containing repeat measurement data and time-to-event data in long format.
`x_L`	Numeric specifying the landmark time(s)
`fixed_effects`	Vector of character strings specifying the column names in `data_long` which correspond to the fixed effects
`random_effects`	Vector of character strings specifying the column names in `data_long` which correspond to the random effects
`fixed_effects_time`	Character string specifying the column name in `data_long` which contains the time at which the fixed effects were recorded
`random_effects_time`	Vector of character strings specifying the column names in `data_long` which contain the times at which repeat measures were recorded. This should either be length 1 or the same length as `random_effects`. In the latter case the order of elements must correspond to the order of elements in `random_effects`.
`standardise_time`	Boolean indicating whether to standardise the time variable by subtracting the mean and dividing by the standard deviation (see Details section for more information)
`random_slope_in_LME`	Boolean indicating whether to include a random slope in the LME model
`random_slope_as_covariate`	Boolean indicating whether to include the random slope estimate from the LME model as a covariate in the survival submodel.

`cv_name`	Character string specifying the column name in `data_long` that indicates cross-validation fold
`individual_id`	Character string specifying the column name in `data_long` which contains the individual identifiers
`lme_control`	Object created using `nlme::lmeControl()`, which will be passed to the `control` argument of the `lme` function

Details

For an individual i, the LME model can be written as

$$Y_i = X_i\beta + Z_iU_i + \epsilon_i$$

where

- Y_i is the vector of outcomes at different time points for the individual
- X_i is the matrix of covariates for the fixed effects at these time points
- β is the vector of coefficients for the fixed effects
- Z_i is the matrix of covariates for the random effects
- U_i is the matrix of coefficients for the random effects
- ϵ_i is the error term, typically from N(0, σ)

By using an LME model to fit repeat measures data we can allow measurements from the same individuals to be more similar than measurements from different individuals. This is done through the random intercept and/or random slope.

Extending this model to the case where there are multiple random effects, denoted k, we have

$$Y_{ik} = X_{ik}\beta_k + Z_{ik}U_{ik} + \epsilon_{ik}$$

Using this model we can allow a certain covariance structure within the random effects term U_{ik}, for example a sample from the multivariate normal (MVN) distribution $MVN(0, \Sigma_u)$. This covariance structure means the value of one random effects variable informs about the value of the other random effects variables, leading to more accurate predictions and allowing there to be missing data in the random effects variables.

The function `fit_LME_landmark` uses a unstructured covariance for the random effects when fitting the LME model (i.e. no constraints are imposed on the values). To fit the LME model the function `lme` from the package `nlme` is used. The fixed effects are calculated as the LOCF for the variables `fixed_effects` at the landmark age `x_L` and the random effects are those stated in `random_effects` and at times `random_effects_time`. The random intercept is always included in the LME model. Additionally, the random slope can be included in the LME model using the parameter `random_slope_in_LME=TRUE`. The model is used to predict the values of the random effects at the landmark time `x_L`, and these are used as predictors in the survival model along with the LOCF values of the fixed effects. Additionally, the estimated value of the random slope can be included as predictors in the survival model using the parameter `random_slope_as_covariate=TRUE`.

It is important to distinguish between the validation set and the development set for fitting the LME model. The development set includes all the repeat measurements (including those after the landmark age x_L). Conversely, the validation set only includes the repeat measurements recorded up until and including the landmark age x_L.

There is an important consideration about fitting the linear mixed effects model. As the variable random_effects_time gets further from 0, the random effects coefficients get closer to 0. This causes computational issues as the elements in the covariance matrix of the random effects, Σ_u, are constrained to be greater than 0. Using parameter standard_time=TRUE can prevent this issue by standardising the time variables to ensure that the random_effects_time values are not too close to 0.

The LOCF values for the fixed effects and the prediction of the random effects at the landmark age are used as the covariates for the survival submodel, in addition to the estimated random slopes if option random_effects_as_covariate is selected.

Value

List containing elements: data_longitudinal, model_longitudinal, model_LME, and model_LME_standardise_time.

data_longitudinal has one row for each individual in the risk set at x_L and contains the value of the covariates at the landmark time x_L of the fixed_effects using the LOCF model and random_effects using the LME model.

model_longitudinal indicates that the LME approach is used.

model_LME contains the output from the lme function from package nlme. For a model using cross-validation, model_LME contains a list of outputs with each element in the list corresponds to a different cross-validation fold.

model_LME_standardise_time contains a list of two objects mean_response_time and sd_response_time if the parameter standardise_time=TRUE is used. This is the mean and standard deviation used to normalise times when fitting the LME model.

---

fit_LOCF_landmark — *Fit a landmark model using a last observation carried forward (LOCF) method for the longitudinal data*

Description

This function performs the two-stage landmarking analysis. In the first stage, the longitudinal submodel is fitted using the LOCF method and in the second stage the survival submodel is fitted.

Usage

```
fit_LOCF_landmark(
  data_long,
  x_L,
  x_hor,
  covariates,
  covariates_time,
  k,
  cross_validation_df,
  individual_id,
  event_time,
  event_status,
  survival_submodel = c("standard_cox", "cause_specific", "fine_gray"),
  b
)
```

Arguments

`data_long`	Data frame or list of data frames each corresponding to a landmark age `x_L` (each element of the list must be named the value of `x_L` it corresponds to). Each data frame contains repeat measurements data and time-to-event data in long format.
`x_L`	Numeric specifying the landmark time(s)
`x_hor`	Numeric specifying the horizon time(s)
`covariates`	Vector of character strings specifying the column names in `data_long` which correspond to the covariates
`covariates_time`	Vector of character strings specifying the column names in `data_long` which contain the times at which `covariates` were recorded. This should either be length 1 or the same length as `covariates`. In the latter case the order of elements must correspond to the order of elements in `covariates`.
`k`	Integer specifying the number of folds for cross-validation. An alternative to setting parameter `cross_validation_df` for performing cross-validation; if both are missing no cross-validation is used.
`cross_validation_df`	List of data frames containing the cross-validation fold each individual is assigned to. Each data frame in the list should be named according to the landmark time `x_L` that they correspond. Each data frame should contain the columns `individual_id` and a column `cross_validation_number` which contains the cross-validation fold of the individual. An alternative to setting parameter `k` for performing cross-validation; if both are missing no cross-validation is used.
`individual_id`	Character string specifying the column name in `data_long` which contains the individual identifiers
`event_time`	Character string specifying the column name in `data_long` which contains the event time
`event_status`	Character string specifying the column name in `data_long` which contains the event status (where 0=censoring, 1=event of interest, if there are competing events these are labelled 2 or above).
`survival_submodel`	Character string specifying which survival submodel to use. Three options: the standard Cox model i.e. no competing risks (`"standard_cox"`), the cause-specific regression model (`"cause_specific"`), or the Fine Gray regression model (`"fine_gray"`)
`b`	Integer specifying the number of bootstrap samples to take when calcluating standard error of c-index and Brier score

Details

Firstly, this function selects the individuals in the risk set at the landmark time `x_L`. Specifically, the individuals in the risk set are those that have entered the study before the landmark age (there is at least one observation for each of the `covariates` on or before `x_L`) and exited the study on after the landmark age (`event_time` is greater than `x_L`).

Secondly, if the option to use cross validation is selected (using either parameter `k` or `cross_validation_df`), then an extra column `cross_validation_number` is added with the cross-validation folds. If parameter `k` is used, then the function `add_cv_number` randomly assigns these folds. For more details on this function see `?add_cv_number`. If the parameter `cross_validation_df` is used, then the folds specified in this data frame are added. If cross-validation is not selected then the landmark model is fit to the entire group of individuals in the risk set (this is both the training and test dataset).

Thirdly, the landmark model is then fit to each of the training data. There are two parts to fitting the landmark model: using the longitudinal data and using the survival data. Using the longitudinal data is the first stage and is performed using `fit_LOCF_longitudinal`. See `?fit_LOCF_longitudinal` more for information about this function. Using the survival data is the second stage and is performed using `fit_survival_model`. See `?fit_survival_model` more for information about this function.

Fourthly, the performance of the model is then assessed on the set of predictions from the entire set of individuals in the risk set by calculating Brier score and C-index. This is performed using `get_model_assessment`. See `?get_model_assessment` more for information about this function.

Value

List containing containing information about the landmark model at each of the landmark times. Each element of this list is named the corresponding landmark time, and is itself a list containing elements: `data`, `model_longitudinal`, `model_survival`, and `prediction_error`.

`data` has one row for each individual in the risk set at `x_L` and contains the value of the `covariates` at the landmark time `x_L` using the LOCF approach. It also includes the predicted probability that the event of interest has occurred by time `x_hor`, labelled as `"event_prediction"`. There is one row for each individual.

`model_longitudinal` indicates that the longitudinal approach is LOCF.

`model_survival` contains the outputs from the function used to fit the survival submodel, including the estimated parameters of the model. For a model using cross-validation, `model_survival` contains a list of outputs with each element in the list corresponding to a different cross-validation fold. For more information on how the survival model is fitted please see `?fit_survival_model` which is a function used within `fit_LOCF_landmark`.

`prediction_error` contains a list indicating the c-index and Brier score at time `x_hor` and their standard errors if parameter `b` is used. For more information on how the prediction error is calculated please see `?get_model_assessment` which is the function used to do this within `fit_LOCF_landmark`.

Author(s)

Isobel Barrott `<isobel.barrott@gmail.com>`

Examples

```
library(Landmarking)
data(data_repeat_outcomes)
data_model_landmark_LOCF <-
   fit_LOCF_landmark(
     data_long = data_repeat_outcomes,
     x_L = c(60, 61),
     x_hor = c(65, 66),
     covariates =
       c("ethnicity", "smoking", "diabetes", "sbp_stnd", "tchdl_stnd"),
     covariates_time =
       c(rep("response_time_sbp_stnd", 4), "response_time_tchdl_stnd"),
     k = 10,
     individual_id = "id",
     event_time = "event_time",
     event_status = "event_status",
     survival_submodel = "cause_specific"
   )
```

`fit_LOCF_longitudinal` *Find the last observation carried forward (LOCF) values for covariates in a dataset*

Description

This function is a helper function for `fit_LOCF_landmark`.

Usage

```
fit_LOCF_longitudinal(
  data_long,
  x_L,
  covariates,
  covariates_time,
  cv_name = NA,
  individual_id
)
```

Arguments

`data_long`	Data frame in long format i.e. there may be more than one row per individual
`x_L`	Numeric specifying the landmark time(s)
`covariates`	Vector of character strings specifying the column names in `data_long` which correspond to the covariates
`covariates_time`	Vector of character strings specifying the column names in `data_long` which contain the times at which `covariates` were recorded. This should either be length 1 or the same length as `covariates`. In the latter case the order of elements must correspond to the order of elements in `covariates`.

cv_name	Character string specifying the column name in data_long that indicates cross-validation fold
individual_id	Character string specifying the column name in data_long which contains the individual identifiers

Details

This function extracts the LOCF value for each of the covariates in data_long up to (and including) time x_L.

Value

List containing data_longitudinal, model_longitudinal, and call.

data_longitudinal has one row for each individual in data_long and contains the LOCF value of covariates at the landmark time x_L.

model_longitudinal indicates that the LOCF approach is used.

call contains the call of the function.

Author(s)

Isobel Barrott <isobel.barrott@gmail.com>

---

fit_survival_model *Fit a survival sub-model*

Description

This function is a helper function for fit_LOCF_landmark_model and fit_LME_landmark_model.

Usage

```
fit_survival_model(
  data,
  individual_id,
  cv_name = NA,
  covariates,
  event_time,
  event_status,
  survival_submodel = c("standard_cox", "cause_specific", "fine_gray"),
  x_hor
)
```

Arguments

`data`	Data frame containing covariates and time-to-event data, one row for each individual.
`individual_id`	Character string specifying the column name in `data` which contains the individual identifiers
`cv_name`	Character string specifying the column name in `data` that indicates cross-validation fold. If no cross-validation is needed, set this parameter to `NA`.
`covariates`	Vector of character strings specifying the column names in `data_long` which correspond to the covariates
`event_time`	Character string specifying the column name in `data` which contains the event time
`event_status`	Character string specifying the column name in `data` which contains the event status (where 0=censoring, 1=event of interest, if there are competing events these are labelled 2 or above). Events at time x_hor should be labelled censored.
`survival_submodel`	Character string specifying which survival submodel to use. Three options: the standard Cox model i.e. no competing risks (`"standard_cox"`), the cause-specific regression model (`"cause_specific"`), or the Fine Gray regression model (`"fine_gray"`)
`x_hor`	Numeric specifying the horizon time(s)

Details

For the survival submodel, there are three choices of model:

- the standard Cox model, this is a wrapper function for `coxph` from the package `survival`
- the cause-specific model, this is a wrapper function for `CSC` from package `riskRegression`
- the Fine Gray model, this is a wrapper function for `FGR` from package `riskRegression`

The latter two models estimate the probability of the event of interest in the presence of competing events.

For both the c-index and Brier score calculations, inverse probability censoring weighting (IPCW) is used to create weights which account for the occurrence of censoring. The censoring model assumes for this function is the Kaplan Meier model, i.e. censoring occurs independently of covariates.

Value

List containing `data_survival` and `model_survival`

`data_survival` contains the predicted risk of event by the horizon time `x_hor`.

`model_survival` contains the outputs from the function used to fit the survival submodel, including the estimated parameters of the model. For a model using cross-validation, `model_survival` contains a list of outputs with each element in the list corresponding to a different cross-validation fold.

Author(s)

Isobel Barrott `<isobel.barrott@gmail.com>`

get_model_assessment *Compute C-index and Brier score*

Description

Performs model assessment by computing the C-index and Brier score at time `x_hor`. There is the option to calculate their standard errors using bootstraping.

Usage

```
get_model_assessment(
  data,
  individual_id,
  event_prediction,
  event_status,
  event_time,
  x_hor,
  b
)
```

Arguments

`data`	Data frame containing survival outcomes and the event predictions from the model, there should be one row for each individual
`individual_id`	Character string specifying the column name in `data` which contains the individual identifiers
`event_prediction`	Character string specifying the column name in `data` containing the predicted probability of the event of interest
`event_status`	Character string specifying the column name in `data` which contains the event status (where 0=censoring, 1=event of interest, if there are competing events these are labelled 2 or above). Events at time `x_hor` should be labelled censored.
`event_time`	Character string specifying the column name in `data` which contains the event time.
`x_hor`	Numeric specifying the horizon time(s)
`b`	Integer specifying the number of bootstrap samples to take when calcluating standard error of c-index and Brier score

Details

There are two factors in assessing the performance of a prediction model; its discrimination and its calibration. The c-index is a commonly used metric which assesses discrimination, this refers to the ability of the model to separate individuals into those that will have an event and those that will not. The c-index at a horizon time `x_hor` looks at the pairs of individuals where one individual has the event at a time T and the other has not had the event at time T. It is calculated as the proportion of these pairs where their relative risk prediction agrees with the actual outcomes for the two individuals. This is extended to the competing risks case by comparing individuals where one had the event of interest at time T and the other individual either did not experience the event before this time T or experienced a competing event.

The Brier score gives an indication of the calibration of a model (and its discrimination to an extent), this refers to the agreement between the risk prediction and the outcome. The Brier score is calculated as the average mean squared error of the predicted risk and the event outcome (where an event is 1 and not experiencing the event is 0). This is extended to the competing risks case by including the competing risk events as not experiencing the event.

For both the c-index and Brier score calculations, inverse probability censoring weighting (IPCW) is used to create weights which account for the occurence of censoring. The censoring model assumes for this function is the Kaplan Meier model, i.e. censoring occurs independently of covariates.

The c-index is calculated using the `cindex` function in package `pec`. The Brier score is calculated using `pec` function in package `pec`.

Value

List containing C-index, Brier score and their standard errors

Author(s)

Isobel Barrott <`isobel.barrott@gmail.com`>

Examples

```
## Not run:
library(Landmarking)
data(data_repeat_outcomes)
data_model_landmark_LOCF <-
  fit_LOCF_landmark(
    data_long = data_repeat_outcomes,
    x_L = c(60, 61),
    x_hor = c(65, 66),
    covariates =
      c("ethnicity", "smoking", "diabetes", "sbp_stnd", "tchdl_stnd"),
    covariates_time =
      c(rep("response_time_sbp_stnd", 4), "response_time_tchdl_stnd"),
    k = 10,
    individual_id = "id",
    event_time = "event_time",
    event_status = "event_status",
    survival_submodel = "cause_specific"
  )
get_model_assessment(data = data_model_landmark_LOCF[["60"]]$data,
  individual_id = "id",
  event_prediction = "event_prediction",
  event_status = "event_status",
  event_time = "event_time",
  x_hor = 65,
  b = 10)
## End(Not run)
```

mixoutsamp *Calculate point estimates from a linear mixed effects (LME) model for new data*

Description

This function allows the user to make out-of-sample predictions from an LME model.

Usage

```
mixoutsamp(model, newdata)
```

Arguments

`model`	Object of class `lme` containing the fitted LME model
`newdata`	Data frame containing data for which to make predictions. The response variable should be set to NA for the rows of the data the user wishes to make predictions for. The columns in the data should have the same names as those used to fit the model. The variables should also be of the same type as in the data used to fit the mixed model (numeric, factor etc).

Value

List containing `preddata` and `random`. Data frame `preddata` is a version of `newdata` updated to contain columns corresponding to the fixed effects values (`fixed`), random effects values (`random`), and fitted values (`fitted`). Data frame `random` contains the values of random effects components for each individual.

Author(s)

This code was originally written by Ruth Keogh (London School of Hygiene and Tropical Medicine) which can be viewed at github.com/ruthkeogh/landmark_CF. There have been further contributions from Jessica Barrett (MRC Biostatistics Unit, University of Cambridge), David Stevens (University of Liverpool), and Mike Sweeting (University of Leicester).

plot.landmark *Create a calibration plot*

Description

Creates a calibration plot for the landmark model fitted by `fit_LME_landmark_model` or `fit_LOCF_landmark_model`. This function plots the observed frequencies of the event of interest against the predicted probabilities of the event of interest.

Usage

```
## S3 method for class 'landmark'
plot(x, x_L, n, x_lims, y_lims, ...)
```

Arguments

`x`	Object inheriting the class `landmark`, this should be the output from either `fit_LME_landmark_model` or `fit_LOCF_landmark_model`. It should contain a list of landmark models corresponding to different landmark times `x_L`.
`x_L`	Numeric specifying the landmark time. This indicates which landmark model in `x` to use.
`n`	Numeric specifying the number of bins to use.
`x_lims`	Vector of length 2 specifying the limits of the x axes
`y_lims`	Vector of length 2 specifying the limits of the y axes
`...`	Arguments passed to `ggplot2::labs` to modify axis, legend, and plot labels

Details

This function bins the predicted probabilities of the event of interest into `n` bins. The event of interest is the event with `event_status=1` when fitting the landmark model. For each of the `n` sets of individuals, the Aalen-Johansen estimator is fit to that set and used to calculate the risk of an event at the horizon time. The predictions (from the landmark model) and the observed frequencies (from the Aalen-Johansen estimator) are plotted against each other. For a perfect prediction model, the points will be plotted along the y=x line.

Value

Calibration plot showing the value of predicted probabilities against observed frequencies, with a y=x line.

Examples

```
library(Landmarking)
data(data_repeat_outcomes)
data_model_landmark_LOCF <-
  fit_LOCF_landmark(
    data_long = data_repeat_outcomes,
    x_L = c(60, 61),
    x_hor = c(65, 66),
    covariates =
      c("ethnicity", "smoking", "diabetes", "sbp_stnd", "tchdl_stnd"),
    covariates_time =
      c(rep("response_time_sbp_stnd", 4), "response_time_tchdl_stnd"),
    k = 10,
    individual_id = "id",
    event_time = "event_time",
    event_status = "event_status",
    survival_submodel = "cause_specific"
  )
```

```
plot(x=data_model_landmark_LOCF,x_L=60,n=5)
plot(x=data_model_landmark_LOCF,x_L=61,n=5)
```

`predict.landmark` *Predict the risk of an event for a new individual using the landmark model*

Description

This function predicts the risk of an event for new data using the landmark model fitted by `fit_LME_landmark` or `fit_LOCF_landmark`. The 'event' is defined as event for which `event_status` is 1.

Usage

```
## S3 method for class 'landmark'
predict(object, x_L, x_hor, newdata, cv_fold = NA, ...)
```

Arguments

`object`	Object inheriting the class `landmark`, this should be the output from either `fit_LME_landmark` or `fit_LOCF_landmark`. It should contain a list of landmark models corresponding to different landmark times `x_L`.
`x_L`	Numeric specifying the landmark time. This indicates which landmark model in `object` to use.
`x_hor`	Numeric specifying the horizon time. The function assesses the risk of event before this time.
`newdata`	Data frame containing new data to return the risk prediction of the event of interest. The data should be in in long format and the columns must contain the covariates and time variables that are used to fit the model. For the LME model this the variables `fixed_effects`, `random_effects`, `fixed_effects_time`, and `random_effects_time`. For the LOCF model this is `covariates` and `covariates_time`.
`cv_fold`	If cross validation is used to fit `fit_LME_landmark` or `fit_LOCF_landmark`, then the cross validation fold to use when making risk predictions needs to be specified.
`...`	Arguments passed on to `riskRegression::predictRisk`

Value

Data frame `newdata` updated to contained a new column `event_prediction`

Examples

```
library(Landmarking)
data(data_repeat_outcomes)
data_model_landmark_LOCF <-
  fit_LOCF_landmark(
    data_long = data_repeat_outcomes,
    x_L = c(60, 61),
    x_hor = c(65, 66),
    covariates =
      c("ethnicity", "smoking", "diabetes", "sbp_stnd", "tchdl_stnd"),
    covariates_time =
      c(rep("response_time_sbp_stnd", 4), "response_time_tchdl_stnd"),
    k = 10,
    individual_id = "id",
    event_time = "event_time",
    event_status = "event_status",
    survival_submodel = "cause_specific"
  )
 newdata <-
   rbind(
     data.frame(
       id = c(3001, 3001, 3001),
       response_time_sbp_stnd = c(57, 58, 59),
       smoking = c(0, 0, 0),
       diabetes = c(0, 0, 0),
       ethnicity = c("Indian", "Indian", "Indian"),
       sbp_stnd = c(0.45, 0.87, 0.85),
       tchdl_stnd = c(-0.7, 0.24, 0.3),
       response_time_tchdl_stnd = c(57, 58, 59)
     )
   )
 predict(object=data_model_landmark_LOCF,x_L=60,x_hor=62,newdata=newdata,cv_fold=1)
```

`return_ids_with_LOCF` *Select individuals in a dataset with a last observation carried forward (LOCF) at a landmark time*

Description

To fit the LOCF model, all individuals must have at least one non-`NA` entry by landmark time `x_L` for all covariates. This function selects these individuals and removes the other rows.

Usage

```
return_ids_with_LOCF(
  data_long,
  individual_id,
  x_L,
  covariates,
  covariates_time
)
```

Arguments

`data_long`	Data frame with repeated measurements data in long format
`individual_id`	Character string specifying the column name in `data_long` which contains the individual identifiers
`x_L`	Numeric specifying the landmark time(s)
`covariates`	Vector of character strings specifying the column names in `data_long` which correspond to the covariates
`covariates_time`	Vector of character strings specifying the column names in `data_long` which contain the times at which `covariates` were recorded. This should either be length 1 or the same length as `covariates`. In the latter case the order of elements must correspond to the order of elements in `covariates`.

Details

Individuals have a LOCF if there is a non-`NA` entry for each of the covariates in `covariates` up until (not including) time `x_L`.

Value

List of data frames which correspond to each landmark time `x_L`. Each data frame is an updated version of `data_long` which contains only rows of individuals with a LOCF at age `x_L`, other rows are removed.

Author(s)

Isobel Barrott <`isobel.barrott@gmail.com`>

Examples

```
library(Landmarking)
data(data_repeat_outcomes)
data_repeat_outcomes <-
  return_ids_with_LOCF(
    data_long = data_repeat_outcomes,
    individual_id = "id",
    covariates =
      c("ethnicity", "smoking", "diabetes", "sbp_stnd", "tchdl_stnd"),
    covariates_time =
      c(rep("response_time_sbp_stnd", 4), "response_time_tchdl_stnd"),
    x_L = c(60,61)
  )
```

Index