Package 'BJM'

September 25, 2026

Title Backward Joint Model for the Dynamic Prediction of Both Time-to-Event and Longitudinal Outcomes

Version 0.2.0

Maintainer Wenhao Li <wenhaoli.jlu@gmail.com>

Description Provides tools to fit joint models of multivariate longitudinal data and time-to-event data for dynamic prediction. It allows the joint prediction of both future time-to-event outcomes and future longitudinal outcomes conditional on survival. The models accommodate irregularly measured longitudinal data and competing risks outcomes. The use of the backward joint model enables fast and efficient computation, especially for applications with large sample sizes and many longitudinal variables.

License MIT + file LICENSE

LazyData true

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0), survival

Imports nlme, mvtnorm, ggplot2, Matrix, parallel

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

Author Wenhao Li [aut, cre], Liang Li [aut]

NeedsCompilation no

Repository CRAN

Date/Publication 2026-09-25 02:30:02 UTC

Contents

`checkBandcountConvergence`

Check whether bandcount1/bandcount2/bandcount3 are large enough

---

Description

`dynamicPrediction()`/`dynamicPredictionBio()` approximate the integrals behind the predicted risk probabilities (and, for `dynamicPredictionBio()`, the predicted biomarker density) with a finite grid, controlled by `bandcount1`/`bandcount2`/ `bandcount3`. There is no universal correct value: too few grid points silently bias the answer, and too many just cost more time, and the right value depends on the data (e.g. how wide the follow-up range is). Rather than guess, or auto-loop until some tolerance is met – which multiplies runtime unpredictably, especially for `dynamicPredictionBio()`'s nested `bandcount2` x `bandcount3` grid – this runs the prediction once at the bandcount value(s) you supply, once more with those value(s) scaled up, and reports the largest relative change between the two, so you can see whether you have already converged or need to increase the checked bandcount(s) and re-run. It costs exactly 2 calls to `predict_fun`, regardless of how many times you invoke it.

Usage

```
checkBandcountConvergence(
  predict_fun,
  ...,
  bandcount_args,
  multiplier = 2,
  tol = 0.01
)
```

Arguments

predict_fun	The prediction function to check: dynamicPrediction or dynamicPredictionBio themselves (not a string, and not predictPlot()/riskPlot(), which return a plot rather than the underlying numeric predictions this function compares).
...	Arguments to forward to predict_fun, exactly as you would call it directly, except for the bandcount argument(s) being checked, which are supplied separately via bandcount_args. Any bandcount argument of predict_fun not named in bandcount_args must still be supplied here (it is held fixed at that value for both calls).
bandcount_args	A named list giving the bandcount value(s) to check, e.g. list(bandcount1 = 10, bandcount2 = 40). Every element is scaled by multiplier for the second call. To isolate which bandcount is driving instability, check one at a time (e.g. list(bandcount2 = 40), with bandcount3 passed as a fixed value via ...), rather than checking all of them together.
multiplier	Factor the checked bandcount value(s) are scaled by for the second call. Must be greater than 1. Defaults to 2 (doubling).
tol	Relative-change tolerance below which the result is reported as converged. Defaults to 0.01 (1%).

Value

An object of class "checkBandcountConvergence.BJM", a named list with elements:

base_bandcount The bandcount value(s) checked, as supplied in bandcount_args.

scaled_bandcount base_bandcount scaled by multiplier.

tol The relative-change tolerance used to decide converged.

by_field A named numeric vector giving the largest relative change, per comparable output field (e.g. risk_prob_1, Y_predict), between the base and scaled call.

max_rel_diff The largest value in by_field, or NA if there was no comparable output (e.g. horizon <= 0).

converged TRUE if max_rel_diff < tol, FALSE if not, NA if there was nothing to compare.

base_result The full result of calling predict_fun at base_bandcount.

scaled_result The full result of calling predict_fun at scaled_bandcount.

Printing the object summarizes these fields.

Examples

```
data(pbc3)

data_survival_fitting = pbc3[!duplicated(pbc3$id), ]
survival_fit_all = survivalSub(data_survival_fitting, Surv(years, status3) ~ age + sex, NULL)

long_sub_fixed = serBilir ~ year + age + sex + (years) + (years) * year
long_sub_random = ~ year | id
```

```
long_fit_all = longitudinalSub(pbc3[pbc3$status3 == 1, ], long_sub_fixed, long_sub_random)

trans = survivalTrans(c(1, 3, 5, 7))

data_predict_all = pbc3[pbc3$id == 2 & pbc3$year <= 3, ]

# Check bandcount1/bandcount2 together: are they both already large enough?
check = checkBandcountConvergence(
  dynamicPrediction, data_predict_all, long_fit_all, survival_fit_all,
  prediction_time = 3, horizon = 3, time_variable = "year",
  trans$survival_variable_all, trans$survival_trans_function,
  bandcount_args = list(bandcount1 = 10, bandcount2 = 10)
)
print(check)
```

`cmtPlot` *Plot conditional mean trajectories (CMT)*

Description

This function generates the Conditional Mean Trajectories (CMT) plot. All patients in this plot experience events at the same time point, specified by `condi_time2event`. Several evenly spaced time points between the baseline and `condi_time2event` are selected for plotting. Each point is calculated using the mean value of all patients' biomarker values at that time point. The interval between two time points is defined by `interval_time`

Usage

```
cmtPlot(
  data_plot_all,
  condi_time2event,
  event_type_variable,
  event_type,
  bio_variable,
  time_variable,
  survival_variable,
  interval_time = 1/12,
  id_variable = "id"
)
```

Arguments

`data_plot_all`	A `data.frame` that includes the biomarker used for plotting. It is utilized to plot conditional mean trajectories (CMT).

condi_time2event
: Conditional event time, indicating that all patients should have events at this time in the plot. If NULL, it defaults to the midpoint of the observed range of time_variable in data_plot_all.

event_type_variable
: Competing risks variable indicator name. Set to NULL if there are no competing risks.

event_type
: A vector containing the names of all event types.

bio_variable
: Name of the biomarker variable used for plotting.

time_variable
: The name of time variable in linear mixed model.

survival_variable
: Name of the time-to-event outcomes variable.

interval_time
: The time interval between two time points. Time points are plotted within the baseline to event time.

id_variable
: Name of the patient ID column in data_plot_all. Default is "id".

Value

Conditional mean trajectories plot.

Examples

```
# example without competing risks

data(pbc3)

pbc.cmt <- cmtPlot(data_plot_all = pbc3, condi_time2event = 5,
   event_type_variable = NULL, event_type = NULL,
   bio_variable = "serBilir", time_variable = "year",
   survival_variable = "years",
   interval_time = 1/12
)

pbc.cmt


# example with competing risks

data(pbc3)

data_plot_all = pbc3[!is.na(pbc3$status4),]

pbc.cmt.cr <- cmtPlot(data_plot_all, condi_time2event = 5,
   event_type_variable = 'status4', event_type = c("0", "1"),
   bio_variable = "albumin", time_variable = "year",
   survival_variable = "years",
   interval_time = 1/4
)

pbc.cmt.cr
```

dynamicPrediction *Dynamic prediction function*

Description

The time values in the prediction data subset must be less than the specified prediction_time which is the prediction time. The time points for longitudinal repeated measurements must not surpass the prediction time.

Usage

```
dynamicPrediction(
  data_predict_all,
  long_fit_all,
  survival_fit_all,
  prediction_time,
  horizon,
  time_variable,
  survival_variable_all,
  survival_trans_function,
  bandcount1 = "auto",
  bandcount2 = "auto"
)
```

Arguments

data_predict_all
: This involves a collection of data.frame objects for dynamic prediction, each corresponding to a distinct longitudinal outcome. These data frames should contain the variables specified in long_sub_fixed and long_sub_random. Utilizing a list structure allows for the incorporation of multiple longitudinal outcomes, each potentially following different measurement protocols. In instances where all longitudinal outcomes are recorded at identical time points across patients, a singular data.frame object may be used in a list. Alternatively, a single bare data.frame (not wrapped in a list) may be supplied directly; it is then reused for every longitudinal outcome. It is presumed that each data frame is structured in a long format.

long_fit_all
: Outputs from the model fitting process using the nlme package, encompassing the results and parameters obtained from the analysis.

survival_fit_all
: Results and parameters generated from the model fitting procedure, utilizing the coxph function. These outputs include the comprehensive findings and variables derived from the analysis.

prediction_time
: Time used to make the prediction.

`horizon` Prediction horizon.

`time_variable` The name of time variable in linear mixed model.

`survival_variable_all`
The name of the transformed time-to-event outcomes variable.

`survival_trans_function`
The transformation function used for time-to-event outcomes, in the order of `survival_variable_all`.

`bandcount1` The number of grid points spanning the prediction window, from `prediction_time` to `prediction_time + horizon` (the numerator of the risk probability). Larger values give a more accurate but slower estimate. Defaults to `"auto"` (see Details).

`bandcount2` The number of grid points spanning `[prediction_time, upper_bound]`, where `upper_bound` is set internally to twice the longest observed survival/censoring time among at-risk patients; this approximates integrating out to infinity for the denominator that normalizes the risk probability. A wider follow-up range needs a larger `bandcount2` to keep the grid spacing comparable. Defaults to `"auto"` (see Details).

Details

There is no universal correct value for `bandcount1`/`bandcount2`: as a practical check, double both and confirm the resulting risk probabilities barely change; if they do, keep doubling. By default (`bandcount1 = "auto"`, `bandcount2 = "auto"`), this doubling check is done for you: starting from small built-in values, both are doubled together, and the result is compared to the previous round, until the largest relative change in the risk probabilities drops below 1%, or 2 doublings have been tried (so at most 3 calls' worth of work). If it still has not converged by then, a warning reports this and the result at the largest value tried is returned anyway (not an error), so this never silently loops for an unbounded amount of time. Pass an explicit number for either argument to skip auto-tuning it and use a fixed value instead (as in previous package versions), or call `checkBandcountConvergence()` directly for more control over the tolerance and doubling count. See also `vignette("BJM-intro",package = "BJM")` for a worked example.

Value

An object of class `"dynamicPrediction.BJM"`, a named list with elements:

risk_prob_1 A vector of dynamically predicted probabilities, one per patient, of experiencing the (first) event within the prediction horizon. `0` when `horizon <= 0`.

risk_prob_2 When `survival_fit_all` was fit with competing risks, a vector of dynamically predicted probabilities, one per patient, of experiencing the competing event within the prediction horizon. `NULL` when there is no competing risk, or when `horizon <= 0`.

Examples

```
data(pbc3)

data_survival_fitting =  pbc3[!duplicated(pbc3$id), ]
```

```
form_marginal_surv = Surv(years, status3) ~ age + sex
form_conditional_cr = NULL

survival_fit_all = survivalSub(data_survival_fitting, form_marginal_surv,
                               form_conditional_cr)

long_sub_fixed = list(
  "long1" = serBilir ~ year + age + sex +  (years) + (years) * year,
  "long2" = prothrombin ~ year + age + sex + (years) + (years) * year,
  "long3" = albumin ~ year + age + age * year + sex + (years) + (years) * year,
  "long4" = alkaline ~ year + age + sex + (years) + (years) * year,
  "long5" = SGOT ~ year + age + sex + (years) + (years) * year,
  "long6" = platelets ~ year + age + sex + (years)  + (years) * year)

long_sub_random =list(
  "long1" =  ~ year| id,
  "long2" =  ~ year| id,
  "long3" =  ~ year| id,
  "long4" =  ~ year| id,
  "long5" =  ~ year| id,
  "long6" =  ~ year| id)

survival_variable_all = list(
  "Tyears1",  "Tyears2", "Tyears3", "Tyears4"
)

survival_trans_function = list(
  fun1 = function(x){abs(x - 1)},
  fun2 = function(x){abs(x - 3)},
  fun3 = function(x){abs(x - 5)},
  fun4 = function(x){abs(x - 7)}
)

# Complete case analysis
data_fit_all = list()
for(i in seq_len(length(long_sub_fixed))){
  data_fit_all[[i]] = pbc3[pbc3$status3 == 1, ]
}

# fitting longitudinal submodel
long_fit_all = longitudinalSub(data_fit_all, long_sub_fixed, long_sub_random)

i_PID = 2
data.raw.predict.1 = pbc3[pbc3$id == i_PID, ]

data_predict_all = list()
for(i in seq_len(length(long_sub_fixed))){
  data_predict_all[[i]] = data.raw.predict.1[data.raw.predict.1$year <= 3,]
}

# predict risk probability
risk.prob = dynamicPrediction(data_predict_all, long_fit_all, survival_fit_all,
                              prediction_time = 3,
```

```
horizon = 3, time_variable = "year",
survival_variable_all, survival_trans_function,
bandcount1 = 10, bandcount2 = 10)
```

---

dynamicPredictionBio *Dynamic prediction function for future biomarker*

Description

The time values in the prediction data subset must be less than the specified `prediction_time` which is the prediction time. The time points for longitudinal repeated measurements must not surpass the prediction time.

Usage

```
dynamicPredictionBio(
  bio_i,
  data_predict_all,
  long_fit_all,
  survival_fit_all,
  prediction_time,
  horizon,
  time_variable,
  survival_variable_all,
  survival_trans_function,
  bandcount2 = "auto",
  bandcount3 = "auto"
)
```

Arguments

`bio_i`	Biomarker used to do prediction
`data_predict_all`	This involves a collection of `data.frame` objects for dynamic prediction, each corresponding to a distinct longitudinal outcome. These data frames should contain the variables specified in `long_sub_fixed` and `long_sub_random`. Utilizing a list structure allows for the incorporation of multiple longitudinal outcomes, each potentially following different measurement protocols. In instances where all longitudinal outcomes are recorded at identical time points across patients, a singular `data.frame` object may be used in a `list`. Alternatively, a single bare `data.frame` (not wrapped in a list) may be supplied directly; it is then reused for every longitudinal outcome. It is presumed that each data frame is structured in a long format.

`long_fit_all` Outputs from the model fitting process using the `nlme` package, encompassing the results and parameters obtained from the analysis.

`survival_fit_all`
Results and parameters generated from the model fitting procedure, utilizing the `coxph` function. These outputs include the comprehensive findings and variables derived from the analysis.

`prediction_time`
Time used to make the prediction

`horizon` Prediction horizon

`time_variable` The name of time variable in linear mixed model.

`survival_variable_all`
The name of the transformed time-to-event outcomes variable.

`survival_trans_function`
The transformation function used for time-to-event outcomes, in the order of `survival_variable_all`.

`bandcount2` The number of grid points spanning `[prediction_time, upper_bound]`, where `upper_bound` is set internally to twice the longest observed survival/censoring time among at-risk patients; this approximates integrating out to infinity for the denominator that normalizes the predicted density. A wider follow-up range needs a larger `bandcount2` to keep the grid spacing comparable. Defaults to `"auto"` (see Details).

`bandcount3` The number of points in the candidate-biomarker-value grid (`Y_all`) used to build the predicted density (`Y_density`) and locate its mode (`Y_predict`). This controls the resolution of the density curve, not a time integral; increase it if the density looks jagged or `Y_predict` jumps erratically between nearby grid points. Defaults to `"auto"` (see Details).

Details

There is no universal correct value for `bandcount2`/`bandcount3`: as a practical check, double both and confirm the results barely change; if they do, keep doubling. By default (`bandcount2 = "auto"`, `bandcount3 = "auto"`), this doubling check is done for you: starting from small built-in values, both are doubled together, and the result is compared to the previous round, until the largest relative change in `Y_predict` drops below 1%, or 2 doublings have been tried (so at most 3 calls' worth of work). If it still has not converged by then, a warning reports this and the result at the largest value tried is returned anyway (not an error), so this never silently loops for an unbounded amount of time. Pass an explicit number for either argument to skip auto-tuning it and use a fixed value instead (as in previous package versions), or call `checkBandcountConvergence()` directly for more control over the tolerance and doubling count. See also `vignette("BJM-intro", package = "BJM")` for a worked example.

Value

An object of class `"dynamicPredictionBio.BJM"`, a named list with elements:

Y_predict A vector, one entry per patient, giving the MAP (most likely) predicted value of biomarker `bio_i` at `prediction_time + horizon`.

Y_density A probability matrix whose rows correspond to the candidate biomarker values in `Y_all` and whose columns correspond to individual patients; each entry is the dynamically predicted density of the biomarker taking that value.

Y_all The grid of candidate biomarker values used to build `Y_density`.

Examples

```
data(pbc3)

data_survival_fitting =  pbc3[!duplicated(pbc3$id), ]

form_marginal_surv = Surv(years, status3) ~ age + sex
form_conditional_cr = NULL

survival_fit_all = survivalSub(data_survival_fitting, form_marginal_surv,
                               form_conditional_cr)

long_sub_fixed = list(
  "long1" = serBilir ~ year + age + sex +  (years) + (years) * year,
  "long2" = prothrombin ~ year + age + sex + (years) + (years) * year,
  "long3" = albumin ~ year + age + age * year + sex + (years) + (years) * year,
  "long4" = alkaline ~ year + age + sex + (years) + (years) * year,
  "long5" = SGOT ~ year + age + sex + (years) + (years) * year,
  "long6" = platelets ~ year + age + sex + (years)  + (years) * year)

long_sub_random =list(
  "long1" =  ~ year| id,
  "long2" =  ~ year| id,
  "long3" =  ~ year| id,
  "long4" =  ~ year| id,
  "long5" =  ~ year| id,
  "long6" =  ~ year| id)

survival_variable_all = list(
  "Tyears1",  "Tyears2", "Tyears3", "Tyears4"
)

survival_trans_function = list(
  fun1 = function(x){abs(x - 1)},
  fun2 = function(x){abs(x - 3)},
  fun3 = function(x){abs(x - 5)},
  fun4 = function(x){abs(x - 7)}
)

# Complete case analysis
data_fit_all = list()
for(i in seq_len(length(long_sub_fixed))){
  data_fit_all[[i]] = pbc3[pbc3$status3 == 1, ]
}

# fitting longitudinal submodel
```

```
long_fit_all = longitudinalSub(data_fit_all, long_sub_fixed, long_sub_random)

i_PID = 2
data.raw.predict.1 = pbc3[pbc3$id == i_PID, ]

data_predict_all = list()
for(i in seq_len(length(long_sub_fixed))){
  data_predict_all[[i]] = data.raw.predict.1[data.raw.predict.1$year <= 3,]
}

Y_predict = dynamicPredictionBio(bio_i = 1, data_predict_all, long_fit_all,
                                 survival_fit_all, prediction_time = 3,
                                 horizon = 3, time_variable = "year",
                                 survival_variable_all, survival_trans_function,
                                 bandcount2 = 40, bandcount3 = 400)
```

`longitudinalSub` *The process involves estimating parameters for a multivariate linear mixed-effects model, which simultaneously analyzes multiple dependent variables that may be correlated. This approach incorporates both fixed effects, which are consistent across the population, and random effects, accounting for variations within groups or subjects. By fitting this model, one can assess the influence of predictor variables on several longitudinal outcomes while considering the inherent variability in the data due to random effects.*

Description

The process involves estimating parameters for a multivariate linear mixed-effects model, which simultaneously analyzes multiple dependent variables that may be correlated. This approach incorporates both fixed effects, which are consistent across the population, and random effects, accounting for variations within groups or subjects. By fitting this model, one can assess the influence of predictor variables on several longitudinal outcomes while considering the inherent variability in the data due to random effects.

Usage

```
longitudinalSub(data_fit_all, long_sub_fixed, long_sub_random)
```

Arguments

`data_fit_all`	This process requires a set of `data.frame` objects designated for model fitting, with each `data.frame` representing a separate longitudinal outcome. These `data.frame` objects must include the variables identified in `long_sub_fixed` and `long_sub_random`. The use of a `list` arrangement facilitates the inclusion of various longitudinal outcomes, which may adhere to different measurement

protocols. When all longitudinal outcomes are measured at the same time points for every patient, a single `data.frame` object can be in a list. It is assumed that every `data.frame` is organized in a long format.

`long_sub_fixed` This refers to a collection of formulas detailing the fixed effects portion for each longitudinal outcome. On the left side of each formula, the response variable is defined, while the right side outlines the fixed effect terms. Should only a single formula be provided - whether as a list with one item or as a standalone formula - it is inferred that a conventional univariate joint model is being constructed. Terms whose basis/contrasts depend on the data they are computed from – `poly()` in its default orthogonal mode, `splines::ns()`/ `splines::bs()`, and `factor()` – trigger a warning, because `dynamicPrediction()`/`dynamicPredictionBio()` rebuild the design matrix from a small, patient-specific slice of data at every point on the prediction grid, so the basis recomputed at prediction time can silently disagree with the one used to fit the model (or fail outright with too few distinct values). Prefer `poly(..., raw = TRUE)`, `I(x^2)`, `log()`, `sqrt()`, or other terms that do not depend on the surrounding data.

`long_sub_random`

A list of one-sided formulas that define the model for the random effects of each longitudinal outcome. The number of items in this `list` should match the length of `formLongFixed`.

Value

An object of class `"longitudinalSub.BJM"`, a named list with elements:

lfit A list of fitted univariate linear mixed models, one per longitudinal outcome, each obtained via `lme`.

Sigma_fit The estimated variance-covariance matrix of the random effects in the multivariate linear mixed model.

long_sub_fixed The `long_sub_fixed` argument, as supplied.

long_sub_random The `long_sub_random` argument, as supplied.

xlevels A list, one element per longitudinal outcome, of the factor levels observed in the full training data for that outcome. Used internally at prediction time so that `poly()`/ `splines::ns()`/`splines::bs()`/`factor()` terms in `long_sub_fixed` reuse the basis/contrasts fit at training time instead of recomputing one from a small, patient-specific slice of data.

Examples

```
long_sub_fixed = list(
  "long1" = serBilir ~ year + age + sex +  (years) + (years) * year,
  "long2" = prothrombin ~ year + age + sex + (years) + (years) * year,
  "long3" = albumin ~ year + age + age * year + sex + (years) + (years) * year,
  "long4" = alkaline ~ year + age + sex + (years) + (years) * year,
  "long5" = SGOT ~ year + age + sex + (years) + (years) * year,
  "long6" = platelets ~ year + age + sex + (years)  + (years) * year)

long_sub_random =list(
```

```
  "long1" =  ~ year| id,
  "long2" =  ~ year| id,
  "long3" =  ~ year| id,
  "long4" =  ~ year| id,
  "long5" =  ~ year| id,
  "long6" =  ~ year| id)

survival_variable_all = list(
  "Tyears1",  "Tyears2", "Tyears3", "Tyears4"
)

survival_trans_function = list(
  fun1 = function(x){abs(x - 1)},
  fun2 = function(x){abs(x - 3)},
  fun3 = function(x){abs(x - 5)},
  fun4 = function(x){abs(x - 7)}
)

# Complete case analysis
data_fit_all = list()
for(i in seq_len(length(long_sub_fixed))){
  data_fit_all[[i]] = pbc3[pbc3$status3 == 1, ]
}

# fitting longitudinal submodel
long_fit_all = longitudinalSub(data_fit_all, long_sub_fixed, long_sub_random)

# poly() in its default orthogonal mode, splines::ns()/bs(), and
# factor() trigger a warning (see the long_sub_fixed argument above),
# but are still safe to use: the terms/xlevels/contrasts fit on the
# full training data are cached and reused at prediction time, instead
# of being recomputed from each patient's small per-prediction slice.
long_fit_poly = longitudinalSub(
  pbc3[pbc3$status3 == 1, ],
  serBilir ~ year + poly(age, 2) + factor(sex) + years,
  ~ year | id)
```

---

pbc3 *Mayo Clinic primary biliary cirrhosis data used as example code*

Description

The dataset originates from the Mayo Clinic trial on primary biliary cirrhosis (PBC) of the liver, carried out from 1974 to 1984. It includes data from 424 PBC patients who were referred to the Mayo Clinic within this decade and met the eligibility requirements for a randomized placebo-controlled trial of D-penicillamine. However, only the initial 312 cases from the dataset were enrolled in the randomized trial. Thus, the dataset specifically pertains to these 312 patients, for whom the data is largely complete.

Usage

```
data(pbc3)
```

Format

A data frame with 1945 observations on the following 27 variables:

`id` patients identifier; in total there are 312 patients.

`years` number of years between registration and the earlier of death, transplantation, or study analysis time.

`status` a factor with levels `alive`, `transplanted` and `dead`.

`drug` a factor with levels `placebo` and `D-penicil`.

`age` at registration in years.

`sex` a factor with levels `male` and `female`.

`year` number of years between enrollment and this visit date, remaining values on the line of data refer to this visit.

`ascites` a factor with levels `No` and `Yes`.

`hepatomegaly` a factor with levels `No` and `Yes`.

`spiders` a factor with levels `No` and `Yes`.

`edema` a factor with levels `No edema` (i.e. no edema and no diuretic therapy for edema), `edema no diuretics` (i.e. edema present without diuretics, or edema resolved by diuretics), and `edema despite diuretics` (i.e. edema despite diuretic therapy).

`serBilir` serum bilirubin in mg/dl.

`serChol` serum cholesterol in mg/dl.

`albumin` albumin in mg/dl.

`alkaline` alkaline phosphatase in U/liter.

`SGOT` SGOT in U/ml.

`platelets` platelets per cubic ml/1000.

`prothrombin` prothrombin time in seconds.

`histologic` histologic stage of disease.

`status2` a numeric vector with the value 1 denoting if the patient was dead, and 0 if the patient was alive or transplanted.

`status3` a numeric vector with the value 1 denoting if the patient was dead or transplanted, and 0 if the patient was alive.

`status4` a numeric vector with the value 1 denoting if the patient was transplanted, and 0 if the patient was dead. Used for competing risks.

`status5` a numeric vector with the value 2 if the patient was transplanted, 1 denoting if the patient was dead, and 0 if the patient was alive. Used for competing risks with censored.

`Tyears1` a numeric vector with a transformed value of time-to-event outcome.

`Tyears2` a numeric vector with a transformed value of time-to-event outcome.

`Tyears3` a numeric vector with a transformed value of time-to-event outcome.

`Tyears4` a numeric vector with a transformed value of time-to-event outcome.

Source

pbc.

References

Fleming T, Harrington D. *Counting Processes and Survival Analysis*. 1991; New York: Wiley.

Therneau T, Grambsch P. *Modeling Survival Data: Extending the Cox Model*. 2000; New York: Springer-Verlag.

`predictPlot` *Plot of risk and future biomarker with density using dynamic prediction*

Description

This function gives the risk and biomarker prediction plot.

Usage

```
predictPlot(
  data_predict_all_one,
  long_fit_all,
  survival_fit_all,
  prediction_time = 4,
  horizon = seq(0, 3, 0.5),
  time_variable,
  survival_variable_all,
  survival_trans_function,
  bandcount1 = "auto",
  bandcount2 = "auto",
  bandcount3 = "auto",
  bio_his = 1,
  bio_pred = 1,
  density = 1,
  n_cores = 1
)
```

Arguments

`data_predict_all_one`
: This involves a collection of `data.frame` one object for dynamic prediction and making plots, each corresponding to a distinct longitudinal outcome. These data frames should contain the variables specified in `long_sub_fixed` and `long_sub_random`. Utilizing a list structure allows for the incorporation of multiple longitudinal outcomes, each potentially following different measurement protocols. In instances where all longitudinal outcomes are recorded at identical time points across patients, a singular `data.frame` object may be used in a `list`. Alternatively, a

single bare data.frame (not wrapped in a list) may be supplied directly; it is then reused for every longitudinal outcome. It is presumed that each data frame is structured in a long format.

long_fit_all Outputs from the model fitting process using the nlme package, encompassing the results and parameters obtained from the analysis.

survival_fit_all
Results and parameters generated from the model fitting procedure, utilizing the coxph function. These outputs include the comprehensive findings and variables derived from the analysis.

prediction_time
Time used to make the prediction.

horizon Prediction horizon.

time_variable The name of time variable in linear mixed model.

survival_variable_all
The name of the transformed time-to-event outcomes variable.

survival_trans_function
The transformation function used for time-to-event outcomes, in the order of survival_variable_all.

bandcount1 The number of grid points spanning the prediction window, from prediction_time to prediction_time + horizon. Larger values give a more accurate but slower estimate. Defaults to "auto", which resolves it once, before looping over horizon (using the largest requested horizon as a representative probe), by doubling from a built-in starting value until the predicted risk stabilizes; see dynamicPrediction's bandcount1 for details of that search. The resolved value is then reused, fixed, for every point in horizon – it is not re-searched on every iteration.

bandcount2 The number of grid points used to approximate integrating out to infinity when normalizing the predicted risk/density. A wider follow-up range needs a larger bandcount2 to keep the grid spacing comparable. Defaults to "auto"; resolved the same way as bandcount1 (jointly with it, when both are "auto").

bandcount3 The number of points in the candidate-biomarker-value grid used to build the predicted density curve; controls the resolution of the density, not a time integral. Defaults to "auto"; resolved the same way, but only when bio_pred is non-NULL (it is unused otherwise).

Pass explicit numbers instead of "auto" for full manual control, or use checkBandcountConvergence() (applied to dynamicPrediction()/ dynamicPredictionBio() directly) to inspect the convergence behavior yourself. See also vignette("BJM-intro", package = "BJM") for further guidance on choosing bandcount1/bandcount2/ bandcount3.

bio_his Which biomarker history will be plotted

bio_pred Indicator, predict future biomarker or not, if NULL do not predict

density Indicator, plot future biomarker density or not, if NULL do not plot

n_cores Number of CPU cores to use for computing the prediction at each point in horizon. Each point is computed independently, so this loop can be dispatched across cores. Defaults to 1 (serial execution; identical behavior/output to versions of this function without this argument). Values greater than 1 use parallel::mclapply(),

which relies on forking and is therefore only actually parallel on Unix-like systems (Linux, macOS); on Windows, `mclapply()` silently runs the iterations serially regardless of `n_cores` (a limitation of R's fork-based parallelism, not of this package). Parallel execution produces exactly the same numeric result as serial execution – only the order in which iterations are computed (not the order results are assembled in) changes.

Value

Plot of risk and future biomarker with density using dynamic prediction.

Examples

```
data(pbc3)

data_survival_fitting =  pbc3[!duplicated(pbc3$id), ]

form_marginal_surv = Surv(years, status3) ~ age + sex
form_conditional_cr = NULL

survival_fit_all = survivalSub(data_survival_fitting, form_marginal_surv,
                               form_conditional_cr)

long_sub_fixed = list(
  "long1" = serBilir ~ year + age + sex +  (years) + (years) * year,
  "long2" = prothrombin ~ year + age + sex + (years) + (years) * year,
  "long3" = albumin ~ year + age + age * year + sex + (years) + (years) * year,
  "long4" = alkaline ~ year + age + sex + (years) + (years) * year,
  "long5" = SGOT ~ year + age + sex + (years) + (years) * year,
  "long6" = platelets ~ year + age + sex + (years)  + (years) * year)

long_sub_random =list(
  "long1" =  ~ year| id,
  "long2" =  ~ year| id,
  "long3" =  ~ year| id,
  "long4" =  ~ year| id,
  "long5" =  ~ year| id,
  "long6" =  ~ year| id)

survival_variable_all = list(
  "Tyears1",  "Tyears2", "Tyears3", "Tyears4"
)

survival_trans_function = list(
  fun1 = function(x){abs(x - 1)},
  fun2 = function(x){abs(x - 3)},
  fun3 = function(x){abs(x - 5)},
  fun4 = function(x){abs(x - 7)}
)

# Complete case analysis
data_fit_all = list()
```

```
for(i in seq_len(length(long_sub_fixed))){
  data_fit_all[[i]] = pbc3[pbc3$status3 == 1, ]
}

# fitting longitudinal submodel
long_fit_all = longitudinalSub(data_fit_all, long_sub_fixed, long_sub_random)

i_PID = 2
data.raw.predict.plot = pbc3[pbc3$id == i_PID, ]
data_predict_all_one = list(data.raw.predict.plot, data.raw.predict.plot, data.raw.predict.plot,
                    data.raw.predict.plot, data.raw.predict.plot, data.raw.predict.plot)

# plot biomarker 1 history,  predict future biomarker

predictPlot(data_predict_all_one, long_fit_all, survival_fit_all,
            prediction_time = 5, bio_his = 1, bio_pred = 1,
            horizon = seq(0.5, 3.0, 0.5), time_variable = "year",
            survival_variable_all, survival_trans_function,
           bandcount1 = 10, bandcount2 = 10, bandcount3 = 200)
```

`print.dynamicPrediction.BJM` *Print method for* `dynamicPrediction.BJM` *objects*

Description

Automatically called when you type the result of `dynamicPrediction()` at the console.

Usage

```
## S3 method for class 'dynamicPrediction.BJM'
print(
  x,
  prediction_time = NULL,
  horizon = NULL,
  subject_ids = NULL,
  digits = 4,
  ...
)
```

Arguments

`x`	A `dynamicPrediction.BJM` object.
`prediction_time`	Landmark time (for display). Default `NULL`.

horizon	Prediction horizon (for display). Default NULL.
subject_ids	Optional subject ID labels.
digits	Decimal places. Default 4.
...	Additional arguments (currently unused).

Value

Invisibly returns x.

`print.dynamicPredictionBio.BJM` *Print method for* `dynamicPredictionBio.BJM` *objects*

Description

Automatically called when you type the result of `dynamicPredictionBio()` at the console.

Usage

```
## S3 method for class 'dynamicPredictionBio.BJM'
print(
  x,
  bio_i = NULL,
  long_fit_all = NULL,
  prediction_time = NULL,
  horizon = NULL,
  subject_ids = NULL,
  digits = 4,
  ...
)
```

Arguments

x	A `dynamicPredictionBio.BJM` object.
bio_i	Biomarker index (for label lookup). Default NULL.
long_fit_all	`longitudinalSub.BJM` object for name lookup.
prediction_time	Landmark time (for display). Default NULL.
horizon	Prediction horizon (for display). Default NULL.
subject_ids	Optional subject ID labels.
digits	Decimal places. Default 4.
...	Additional arguments (currently unused).

Value

Invisibly returns x.

print.longitudinalSub.BJM *Print method for* longitudinalSub.BJM *objects*

Description

Automatically called when you type long_fit_all at the console.

Usage

```
## S3 method for class 'longitudinalSub.BJM'
print(x, digits = 4, ...)
```

Arguments

x	A longitudinalSub.BJM object returned by longitudinalSub.
digits	Number of significant digits. Default is 4.
...	Additional arguments (currently unused).

Value

Invisibly returns x.

print.survivalSub.BJM *Print method for* survivalSub.BJM *objects*

Description

Automatically called when you type survival_fit_all or print(survival_fit_all) at the console. Displays a JMbayes2-style formatted summary of the survival sub-model.

Usage

```
## S3 method for class 'survivalSub.BJM'
print(x, digits = 4, ...)
```

Arguments

x	A survivalSub.BJM object returned by survivalSub.
digits	Number of significant digits. Default is 4.
...	Additional arguments (currently unused).

Value

Invisibly returns x.

Examples

```
data(pbc3)
data_survival_fitting <- pbc3[!duplicated(pbc3$id), ]
form_marginal_surv  <- Surv(years, status3) ~ age + sex
form_conditional_cr <- status4 ~ years + age + sex
survival_fit_all  <- survivalSub(data_survival_fitting,
                                 form_marginal_surv, form_conditional_cr)
survival_fit_all   # triggers print.survivalSub.BJM automatically
```

`printBJM` *Print both sub-models of a fitted backward joint model*

Description

Convenience function that prints the longitudinal and survival sub-model summaries together. Unlike `print.longitudinalSub.BJM` and `print.survivalSub.BJM`, this does not dispatch on a single `"BJM"`-classed object, because `longitudinalSub` and `survivalSub` are fit and returned separately; it simply prints both fit objects you already have.

Usage

```
printBJM(long_fit_all, survival_fit_all, digits = 4)
```

Arguments

`long_fit_all`	Output from `longitudinalSub`.
`survival_fit_all`	Output from `survivalSub`.
`digits`	Number of significant digits. Default is 4.

Value

Invisibly returns a named list with both fit objects.

`riskPlot` *Plot of risk using dynamic prediction*

Description

This function gives the risk prediction plot.

Usage

```
riskPlot(
  data_predict_all_pre,
  long_fit_all,
  survival_fit_all,
  prediction_time = NULL,
  bio_i = NULL,
  horizon,
  time_variable,
  survival_variable_all,
  survival_trans_function,
  bandcount1 = "auto",
  bandcount2 = "auto",
  n_cores = 1
)
```

Arguments

`data_predict_all_pre`
: This involves a collection of `data.frame` objects for dynamic prediction, each corresponding to a distinct longitudinal outcome. These data frames should contain the variables specified in `long_sub_fixed` and `long_sub_random`. Utilizing a list structure allows for the incorporation of multiple longitudinal outcomes, each potentially following different measurement protocols. In instances where all longitudinal outcomes are recorded at identical time points across patients, a singular `data.frame` object may be used in a `list`. Alternatively, a single bare `data.frame` (not wrapped in a list) may be supplied directly; it is then reused for every longitudinal outcome. It is presumed that each data frame is structured in a long format.

`long_fit_all`
: Outputs from the model fitting process using the `nlme` package, encompassing the results and parameters obtained from the analysis.

`survival_fit_all`
: Results and parameters generated from the model fitting procedure, utilizing the `coxph` function. These outputs include the comprehensive findings and variables derived from the analysis.

`prediction_time`
: Time used to make the prediction.

`bio_i`
: Biomarker used to do prediction.

`horizon`
: Prediction horizon.

`time_variable`
: The name of time variable in linear mixed model.

`survival_variable_all`
: The name of the transformed time-to-event outcomes variable.

`survival_trans_function`
: The transformation function used for time-to-event outcomes, in the order of `survival_variable_all`.

bandcount1 — The number of grid points spanning the prediction window, from prediction_time to prediction_time + horizon. Larger values give a more accurate but slower estimate. Defaults to "auto", which resolves it once, before looping over the landmark times, (using the first landmark time as a representative probe) by doubling from a built-in starting value until the predicted risk stabilizes; see dynamicPrediction's bandcount1 for details of that search. The resolved value is then reused, fixed, for every landmark time – it is not re-searched on every iteration.

bandcount2 — The number of grid points used to approximate integrating out to infinity when normalizing the predicted risk. A wider follow-up range needs a larger bandcount2 to keep the grid spacing comparable. Defaults to "auto"; resolved the same way as bandcount1 (jointly with it, when both are "auto").

Pass explicit numbers instead of "auto" for full manual control, or use checkBandcountConvergence() (applied to dynamicPrediction() directly) to inspect the convergence behavior yourself. See also vignette("BJM-intro", package = "BJM") for further guidance on choosing bandcount1/bandcount2.

n_cores — Number of CPU cores to use for computing the prediction at each landmark time in prediction_time. Each landmark time is computed independently, so this loop can be dispatched across cores. Defaults to 1 (serial execution; identical behavior/output to versions of this function without this argument). Values greater than 1 use parallel::mclapply(), which relies on forking and is therefore only actually parallel on Unix-like systems (Linux, macOS); on Windows, mclapply() silently runs the iterations serially regardless of n_cores (a limitation of R's fork-based parallelism, not of this package). Parallel execution produces exactly the same numeric result as serial execution – only the order in which iterations are computed (not the order results are assembled in) changes.

Value

Plot of risk using dynamic prediction.

---

summary.dynamicPrediction.BJM

Summary method for dynamicPrediction.BJM *objects*

Description

Like print but also shows mean, SD, and range of predicted risks.

Usage

```
## S3 method for class 'dynamicPrediction.BJM'
summary(
  object,
  prediction_time = NULL,
  horizon = NULL,
```

```
  subject_ids = NULL,
  digits = 4,
  ...
)
```

Arguments

object	A dynamicPrediction.BJM object.
prediction_time	Landmark time (for display). Default NULL.
horizon	Prediction horizon (for display). Default NULL.
subject_ids	Optional subject ID labels.
digits	Decimal places. Default 4.
...	Additional arguments (currently unused).

Value

Invisibly returns object.

---

summary.dynamicPredictionBio.BJM

Summary method for dynamicPredictionBio.BJM *objects*

Description

Like print but also shows distribution-level summaries across subjects.

Usage

```
## S3 method for class 'dynamicPredictionBio.BJM'
summary(
  object,
  bio_i = NULL,
  long_fit_all = NULL,
  prediction_time = NULL,
  horizon = NULL,
  subject_ids = NULL,
  digits = 4,
  ...
)
```

Arguments

object	A dynamicPredictionBio.BJM object.
bio_i	Biomarker index (for label lookup). Default NULL.
long_fit_all	longitudinalSub.BJM object for name lookup.
prediction_time	Landmark time (for display). Default NULL.
horizon	Prediction horizon (for display). Default NULL.
subject_ids	Optional subject ID labels.
digits	Decimal places. Default 4.
...	Additional arguments (currently unused).

Value

Invisibly returns object.

---

summary.longitudinalSub.BJM

Summary method for longitudinalSub.BJM *objects*

Description

Like print but adds per-outcome random-effects variance components and the full correlation matrix of D.

Usage

```
## S3 method for class 'longitudinalSub.BJM'
summary(object, digits = 4, ...)
```

Arguments

object	A longitudinalSub.BJM object.
digits	Number of significant digits. Default is 4.
...	Additional arguments (currently unused).

Value

Invisibly returns a list of per-outcome summary.lme objects.

summary.survivalSub.BJM *Summary method for* survivalSub.BJM *objects*

Description

Called via summary(survival_fit_all). Returns (and prints) an extended summary including baseline hazard range, BIC, and McFadden R2 for the competing-risks GLM.

Usage

```
## S3 method for class 'survivalSub.BJM'
summary(object, digits = 4, ...)
```

Arguments

object	A survivalSub.BJM object returned by survivalSub.
digits	Number of significant digits. Default is 4.
...	Additional arguments (currently unused).

Value

Invisibly returns a list with components cox_summary and (if competing risks) glm_summary.

Examples

```
data(pbc3)
data_survival_fitting <- pbc3[!duplicated(pbc3$id), ]
form_marginal_surv  <- Surv(years, status3) ~ age + sex
form_conditional_cr <- status4 ~ years + age + sex
survival_fit_all  <- survivalSub(data_survival_fitting,
                                 form_marginal_surv, form_conditional_cr)
summary(survival_fit_all)
```

survivalSub *Fitting survival sub-model*

Description

Fitting survival sub-model

Usage

```
survivalSub(data_survival_fitting, form_marginal_surv, form_conditional_cr)
```

Arguments

`data_survival_fitting`
: Input data containing survival outcomes and baseline covariates.

`form_marginal_surv`
: Survival input formats.

`form_conditional_cr`
: Competing risks input formats.

Value

An object of class `"survivalSub.BJM"`, a named list with elements:

coxph_fit The fitted `coxph` marginal survival model.

form_marginal_surv The `form_marginal_surv` formula, as supplied.

glm_fit The fitted `glm` competing-risks (event type) model, or `NULL` if `form_conditional_cr` was not supplied.

form_conditional_cr The `form_conditional_cr` formula, as supplied (or `NULL`).

Examples

```
data(pbc3)

data_survival_fitting =  pbc3[!duplicated(pbc3$id), ]

form_marginal_surv = Surv(years, status3) ~ age + sex
form_conditional_cr = NULL

survival_fit_all = survivalSub(data_survival_fitting, form_marginal_surv,
                               form_conditional_cr)
```

---

survivalTrans *Build a survival-time transform basis from cut points*

Description

`dynamicPrediction()`, `dynamicPredictionBio()`, `predictPlot()`, and `riskPlot()` all take a pair of arguments, `survival_variable_all`/`survival_trans_function`, that describe transformed basis variables of the (remaining) survival time; these can optionally be referenced in `long_sub_fixed` formulas to let the longitudinal sub-model depend flexibly on time-to-event. In every example in this package, that pair follows the same convention: variables named `"Tyears1"`, `"Tyears2"`, ... , each defined as the absolute distance from a fixed cut point (`function(x) abs(x - k)`). `survivalTrans()` builds exactly that pair from a plain vector of cut points, so you do not have to hand-write two parallel lists of matching names and closures. You remain free to construct `survival_variable_all`/`survival_trans_function` by hand for any other transform.

Usage

```
survivalTrans(cut_points, prefix = "Tyears")
```

Arguments

cut_points	A non-empty numeric vector of cut points, one per transformed basis variable.
prefix	Prefix used for the generated variable names in survival_variable_all ("Tyears" by default, giving "Tyears1", "Tyears2", ...).

Value

A named list with elements survival_variable_all and survival_trans_function, in the format expected by dynamicPrediction(), dynamicPredictionBio(), predictPlot(), and riskPlot().

Examples

```
trans <- survivalTrans(c(1, 3, 5, 7))
trans$survival_variable_all
trans$survival_trans_function[[1]](2)

# equivalent to hand-writing:
survival_variable_all <- list("Tyears1", "Tyears2", "Tyears3", "Tyears4")
survival_trans_function <- list(
  fun1 = function(x) abs(x - 1),
  fun2 = function(x) abs(x - 3),
  fun3 = function(x) abs(x - 5),
  fun4 = function(x) abs(x - 7)
)
```

Index