Package 'dynamichazard'

November 26, 2017

Type Package

Title Dynamic Hazard Models using State Space Models

Version 0.5.1

Description Contains functions that lets you fit dynamic hazard models using state space models. The first implemented model is described in Fahrmeir (1992) <doi:10.1080/01621459.1992.10475232> and Fahrmeir (1994) <doi:10.1093/biomet/81.2.317>. Extensions hereof are available where the Extended Kalman filter is replaced by an unscented Kalman filter and other options including particle filters.

License GPL-2

LazyData TRUE

LinkingTo Rcpp, RcppArmadillo

Imports parallel, Rcpp (>= 0.12.6), boot, stringr, data.table

Depends R (>= 3.3.2), stats, graphics, utils, survival

RoxygenNote 6.0.1

Suggests testthat, knitr, rmarkdown, timereg, captioner, biglm, httr, mgcv, shiny, formatR, R.rsp, speedglm, dichromat, colorspace

VignetteBuilder knitr, R.rsp

BugReports https://github.com/boennecd/dynamichazard/issues

SystemRequirements C++11

URL https://github.com/boennecd/dynamichazard

Encoding UTF-8

NeedsCompilation yes

Author Benjamin Christoffersen [cre, aut],
Alan Miller [cph],
Anthony Williams [cph],
Boost developers [cph],
R-core [cph]

Maintainer Benjamin Christoffersen <boennecd@gmail.com>

Repository CRAN

Date/Publication 2017-11-25 23:38:48 UTC

R topics documented:

ddFixed . . . 2
ddhazard . . . 3
ddhazard_app . . . 6
ddhazard_boot . . . 7
get_risk_obj . . . 8
get_survival_case_weights_and_data . . . 9
hatvalues.ddhazard . . . 11
hds . . . 12
logLik.ddhazard . . . 13
logLik.PF_clouds . . . 14
PF_EM . . . 14
plot.ddhazard . . . 16
plot.ddhazard_space_errors . . . 17
plot.PF_clouds . . . 18
plot.PF_EM . . . 19
predict.ddhazard . . . 19
print.ddhazard_boot . . . 21
print.summary.ddhazard . . . 21
residuals.ddhazard . . . 22
static_glm . . . 23

Index **25**

`ddFixed` *Auxiliary functions for fixed effects*

Description

Functions used in formula of `ddhazard` for time-invariant effects. `ddFixed_intercept` is only used for the intercept.

Usage

```
ddFixed(object)

ddFixed_intercept(object)
```

Arguments

`object`	expression that would be used in formula. E.g. `x` or `poly(x, degree = 3)`.

Examples

```
# we can get a time-invariant effect of `x1` by
## Not run:
ddhazard(Surv(stop, event) ~ ddFixed(x1), data)

## End(Not run)

# all of the calls below will yield the same result with a time-invariant
# intercept:
## Not run:
ddhazard(Surv(stop, event) ~ ddFixed_intercept() + x1, data)
ddhazard(Surv(stop, event) ~ -1 + ddFixed_intercept() + x1, data)
ddhazard(Surv(stop, event) ~ ddFixed_intercept(what_ever) + x1, data)

## End(Not run)
```

---

ddhazard *Fitting dynamic hazard models*

Description

Function to fit dynamic hazard models using state space models.

Usage

```
ddhazard(formula, data, model = "logit", by, max_T, id, a_0, Q_0, Q = Q_0,
  order = 1, weights, control = list(), verbose = F)
```

Arguments

formula	coxph like formula with Surv(tstart, tstop, event) on the left hand site of ~.
data	data.frame or environment containing the outcome and co-variates.
model	"logit" or "exponential" for the logistic link function in the first case or for the continuous time model in the latter case.
by	interval length of the bins in which parameters are fixed.
max_T	end of the last interval interval.
id	vector of ids for each row of the in the design matrix.
a_0	vector a_0 for the initial coefficient vector for the first iteration (optional). Default is estimates from static model (see static_glm).
Q_0	covariance matrix for the prior distribution.
Q	initial covariance matrix for the state equation.
order	order of the random walk.
weights	weights to use if e.g. a skewed sample is used.
control	list of control variables (see the control section below).
verbose	TRUE if you want status messages during execution.

Details

This function can be used to estimate survival models where the regression parameters follows a given order random walk. The order is specified by the `order` argument. 1. and 2. order random walks is implemented. The regression parameters are updated at time `by`, 2`by`, ..., `max_T`. See the `vignette("ddhazard", "dynamichazard")` for details.

All filter methods needs a state covariance matrix `Q_0` and state vector `a_0`. An estimate from a time-invariant model is used for `a_0` if it is not supplied (the same model you would get from `static_glm`). A diagonal matrix with large entries is recommended for `Q_0`. What is large dependents on the data set and `model`. Further, a covariance matrix for the first iteration `Q` is needed. The `Q` and `a_0` are estimated with an EM-algorithm.

The model is specified through the `model` argument. The logistic model is where outcomes are binned into the intervals. Be aware that there can be "loss" of information due to binning. It is key for the logit model that the `id` argument is provided if individuals in the data set have time-varying co-variates. The the exponential model use an exponential model for the arrival times where there is no "loss" information due to binning.

It is recommended to see the Shiny app demo for this function by calling `ddhazard_app()`.

Value

A list with class `ddhazard`. The list contains

`formula` the passed formula.

`call` the matched call.

`state_vecs` 2D matrix with the estimated state vectors (regression parameters) in each bin.

`state_vars` 3D array with smoothed variance estimates for each state vector.

`lag_one_cov` 3D array with lagged correlation matrix for each for each change in the state vector. Only present when the model is logit and the method is EKF.

`n_risk` the number of observations in each interval.

`times` the interval borders.

`risk_set` the object from `get_risk_obj` if saved.

`data` the `data` argument if saved.

`weights` `weights` used in estimation if saved.

`id` ids used to match rows in `data` to individuals.

`order` order of the random walk.

`F_` matrix which map from one state vector to the next.

`method` method used in the E-step.

`est_Q_0` `TRUE` if `Q_0` was estimated in the EM-algorithm.

`family` Rcpp `Module` with C++ functions used for estimation given the `model` argument.

`discrete_hazard_func` the hazard function corresponding to the `model` argument.

`terms` the `terms` object used.

`has_fixed_intercept` `TRUE` if the model has a time-invariant intercept.

`xlev` a record of the levels of the factors used in fitting.

Control

The control argument allows you to pass a list to select additional parameters. See vignette("ddhazard", "dynamichaz for more information. Unspecified elements of the list will yield default values.

method set to the method to use in the E-step. Either "EKF" for the Extended Kalman Filter, "UKF" for the Unscented Kalman Filter, "SMA" for the sequential posterior mode approximation method or "GMA" for the global mode approximation method. "EKF" is the default.

LR learning rate.

NR_eps tolerance for the Extended Kalman filter. Default is NULL which means that no extra iteration is made in the correction step.

alpha hyper parameter α in the unscented Kalman Filter.

beta hyper parameter β in the unscented Kalman Filter.

kappa hyper parameter κ in the unscented Kalman Filter.

n_max maximum number of iteration in the EM-algorithm.

eps tolerance parameter for the EM-algorithm.

est_Q_0 TRUE if you want the EM-algorithm to estimate Q_0. Default is FALSE.

save_risk_set TRUE if you want to save the list from get_risk_obj used to estimate the model. It may be needed for later calls to e.g., residuals, plot and logLike.

save_data TRUE if you want to keep the data argument. It may be needed for later calls to e.g., residuals, plot and logLike.

denom_term term added to denominators in either the EKF or UKF.

n_threads maximum number of threads to use.

fixed_parems_start starting value for fixed terms.

fixed_terms_method the method used to estimate the fixed effects. Either 'M_step' or 'E_step' for estimation in the M-step or E-step respectively.

Q_0_term_for_fixed_E_step the diagonal value of the initial covariance matrix, Q_0, for the fixed effects if fixed effects are estimated in the E-step.

eps_fixed_parems tolerance used in the M-step of the Fisher's scoring algorithm for the fixed effects.

permu TRUE if the risk sets should be permutated before computation. This is TRUE by default for posterior mode approximation method and FALSE for all other methods.

posterior_version the implementation version of the posterior approximation method. Either "woodbury" or "cholesky".

GMA_max_rep maximum number of iterations in the correction step if method = 'GMA'.

GMA_NR_eps tolerance for the convergence criteria for the relative change in the norm of the coefficients in the correction step if method = 'GMA'.

References

Fahrmeir, Ludwig. *Dynamic modelling and penalized likelihood estimation for discrete time survival data*. Biometrika 81.2 (1994): 317-330.

Durbin, James, and Siem Jan Koopman. *Time series analysis by state space methods*. No. 38. Oxford University Press, 2012.

See Also

plot, residuals, predict, static_glm, ddhazard_app, ddhazard_boot

Examples

```
# example with first order model
library(dynamichazard)
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3600,
 Q_0 = diag(1, 2), Q = diag(1e-4, 2), by = 50,
 control = list(method = "GMA"))
plot(fit)

# example with second order model
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3600,
 Q_0 = diag(1, 4), Q = diag(1e-4, 2), by = 50,
 control = list(method = "GMA"),
 order = 2)
plot(fit)
```

---

ddhazard_app *ddhazard demo*

Description

ddhazard_app runs a shiny app with demonstration of models.

Usage

```
ddhazard_app(quietly = F, ...)
```

Arguments

quietly	TRUE if no messages should be printed when the app is run.
...	starting values for the shiny app.

Details

Runs a shiny app where you try different model specifications on simulated data.

Examples

```
## Not run:
dynamichazard::ddhazard_app()
dynamichazard::ddhazard_app(seed = 1, more_options = TRUE)

## End(Not run)
```

ddhazard_boot *Bootstrap for* ddhazard

Description

See the vignette `vignette("Bootstrap_illustration", "dynamichazard")`. The `do_stratify_with_event` may be useful when either cases or non-cases are very rare to ensure that the model estimation succeeds.

Usage

```
ddhazard_boot(ddhazard_fit, strata, unique_id, R = 100,
  do_stratify_with_event = F, do_sample_weights = F,
  LRs = ddhazard_fit$control$LR * 2^(0:(-4)), print_errors = F)
```

Arguments

`ddhazard_fit`	returned object from a ddhazard call.
`strata`	strata to sample within. These need to be on an individual by individual basis and not rows in the design matrix.
`unique_id`	unique ids where entries match entries of `strata`.
`R`	number of bootstrap estimates.
`do_stratify_with_event`	`TRUE` if sampling should be by strata of whether the individual has an event. An interaction factor will be made if `strata` is provided.
`do_sample_weights`	`TRUE` if weights should be sampled instead of individuals.
`LRs`	learning rates in decreasing order which will be used to estimate the model.
`print_errors`	`TRUE` if errors should be printed when estimations fails.

Value

An object like from the boot function.

See Also

ddhazard, plot

Examples

```
## Not run:
library(dynamichazard)
set.seed(56219373)
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3000,
 Q_0 = diag(1, 2), Q = diag(1e-4, 2), by = 100,
```

```
 control = list(method = "GMA"))
bt <- ddhazard_boot(fit, R = 999)
plot(fit, ddhazard_boot = bt, level = .9)

## End(Not run)
```

`get_risk_obj` *Risk set on an equidistant distant grid*

Description

Get the risk set at each bin over an equidistant distant grid.

Usage

```
get_risk_obj(Y, by, max_T, id, is_for_discrete_model = T, n_threads = 1,
  min_chunk = 5000)
```

Arguments

`Y`	vector of outcome variable returned from `Surv`.
`by`	length of each bin.
`max_T`	last observed time.
`id`	vector with ids where entries match with outcomes `Y`.
`is_for_discrete_model`	`TRUE` if the model outcome is discrete. For example, a logit model is discrete whereas what is is referred to as the exponential model in this package is a dynamic model.
`n_threads`	set to a value greater than one to use `mclapply` to find the risk object.
`min_chunk`	minimum chunk size of ids to use when parallel version is used.

Value

a list with the following elements

`risk_sets` list of lists with one for each bin. Each of the sub lists have indices that corresponds to the entries of `Y` that are at risk in the bin.

`min_start` start time of the first bin.

`I_len` length of each bin.

`d` number of bins.

`is_event_in` indices for which bin an observation `Y` is an event. `-1` if the individual does not die in any of the bins.

`is_for_discrete_model` value of `is_for_discrete_model` argument.

Examples

```
# small toy example with time-varying covariates
dat <- data.frame(
 id     = c(1, 1, 2, 2),
 tstart = c(0, 4, 0, 2),
 tstop  = c(4, 6, 2, 4),
 event  = c(0, 1, 0, 0))

with(dat, get_risk_obj(Surv(tstart, tstop, event), by = 1, max_T = 6, id = id))
```

---

get_survival_case_weights_and_data

Get `data.frame` *for discrete time survival models*

Description

Function used to get `data.frame` with weights for a static fit for survivals.

Usage

```
get_survival_case_weights_and_data(formula, data, by, max_T, id, init_weights,
  risk_obj, use_weights = T, is_for_discrete_model = T, c_outcome = "Y",
  c_weights = "weights", c_end_t = "t")
```

Arguments

`formula`	`coxph` like formula with `Surv(tstart, tstop, event)` on the left hand site of ~.
`data`	`data.frame` or environment containing the outcome and co-variates.
`by`	interval length of the bins in which parameters are fixed.
`max_T`	end of the last interval interval.
`id`	vector of ids for each row of the in the design matrix.
`init_weights`	weights for the rows in `data`. Useful e.g., with skewed sampling.
`risk_obj`	a pre-computed result from a `get_risk_obj`. Will be used to skip some computations.
`use_weights`	`TRUE` if weights should be used. See details.
`is_for_discrete_model`	`TRUE` if the model is for a discrete hazard model is used like the logistic model.
`c_outcome, c_weights, c_end_t`	alternative names to use for the added columns described in the return section. Useful if you already have a column named `Y`, `t` or `weights`.

Details

This function is used to get the data.frame for e.g. a glm fit that is comparable to a ddhazard fit in the sense that it is a static version. For example, say that we bin our time periods into (0,1], (1,2] and (2,3]. Next, consider an individual who dies at time 2.5. He should be a control in the the first two bins and should be a case in the last bin. Thus the rows in the final data frame for this individual is c(Y = 1, ..., weights = 1) and c(Y = 0, ..., weights = 2) where Y is the outcome, ... is the co-variates and weights is the weights for the regression. Consider another individual who does not die and we observe him for all three periods. Thus, he will yield one row with c(Y = 0, ..., weights = 3).

This function use similar logic as the ddhazard for individuals with time varying co-variates (see the vignette vignette("ddhazard", "dynamichazard") for details).

If use_weights = FALSE then the two previously mentioned individuals will yield three rows each. The first individual will have c(Y = 0, t = 1, ..., weights = 1), c(Y = 0, t = 2, ..., weights = 1), c(Y = 1, t = 3, ..., weights = 1) while the latter will have three rows c(Y = 0, t = 1, ..., weights = 1), c(Y = 0, t = 2, ..., weights = 1), c(Y = 0, t = 3, ..., weights = 1). This kind of data frame is useful if you want to make a fit with e.g. gam function in the mgcv package as described en Tutz et. al (2016).

Value

Returns a data.frame where the following is added (column names will differ if you specified them): column Y for the binary outcome, column weights for weights of each row and additional rows if applicable. A column t is added for the stop time of the bin if use_weights = FALSE.

References

Tutz, Gerhard, and Matthias Schmid. *Nonparametric Modeling and Smooth Effects*. Modeling Discrete Time-to-Event Data. Springer International Publishing, 2016. 105-127.

See Also

ddhazard, static_glm

Examples

```
library(dynamichazard)
# small toy example with time-varying covariates
dat <- data.frame(
 id     = c(   1,    1, 2,     2),
 tstart = c(   0,    4, 0,     2),
 tstop  = c(   4,    6, 2,     6),
 event  = c(   0,    1, 0,     0),
 x1     = c(1.09, 1.29, 0, -1.16))

get_survival_case_weights_and_data(
 Surv(tstart, tstop, event) ~ x1, dat, by = 1, id = dat$id)$X
get_survival_case_weights_and_data(
 Surv(tstart, tstop, event) ~ x1, dat, by = 1, id = dat$id,
 use_weights = FALSE)$X
```

`hatvalues.ddhazard` *Hat values for* `ddhazard` *object*

Description

Computes hat-"like" values from usual L2 penalized binary regression.

Usage

```
## S3 method for class 'ddhazard'
hatvalues(model, ...)
```

Arguments

`model`	a fit from `ddhazard`.
`...`	not used.

Details

Computes hat-"like" values in each interval for each individual at risk in the interval. See the `vignette("ddhazard", "dynamichazard")` vignette for details.

Value

A list of matrices. Each matrix has three columns: the hat values, the row number of the original data point and the id the row belongs to.

See Also

`ddhazard`

Examples

```
library(dynamichazard)
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3000,
 Q_0 = diag(1, 2), Q = diag(1e-4, 2), by = 100,
 control = list(method = "GMA"))
hvs <- hatvalues(fit)
head(hvs[[1]])
head(hvs[[2]])
```

hds *Hard Drive failures.*

Description

A data set containing hard drive failures data from Backblaze in the start-stop format used in the `survival` package.

Usage

```
hds
```

Format

A `data.frame` with the following columns:

serial_number Serial number for the hard disk which the row belongs to.

model hard disk model.

manufacturer manufacturer of the hard disk model.

tstart,tstop start and stop times on the SMART 9 attribute scale.

fails 1 if the hard disk fails at `tstop`.

size_tb hard disk size in terabytes.

smart_x the raw SMART attribute x value. E.g., `smart_12` is the power cycle count.

smart_x_bin 1 if the SMART attribute x value is non-zero.

..._cumsum cumulative sum of the prefix ....

n_fails number of failures in the original data. Hard disk should only fail once but this is not the case in the raw data.

n_records number of records in the original source.

min_date,max_date first and last date in the original source.

min_hours,max_hours smallest and largest value of the SMART 9 attribute in the original source.

Details

Details about the the SMART attributes can be found on https://en.wikipedia.org/wiki/S.M.A.R.T.. As stated in the original source

"Reported stats for the same SMART stat can vary in meaning based on the drive manufacturer and the drive model. Make sure you are comparing apples-to-apples as drive manufacturers don't generally disclose what their specific numbers mean."

There are some notes on https://en.wikipedia.org/wiki/S.M.A.R.T. regarding which attributes that have vendor specific raw value. Further,

"The values in the files are the values reported by the drives. Sometimes, those values are out of whack. For example, in a few cases the RAW value of SMART 9 (Drive life in hours) reported a

value that would make a drive 10+ years old, which was not possible. In other words, it's a good idea to have bounds checks when you process the data."

Last value carried forward have been used for all covariates. See this github page for the processing steps https://github.com/boennecd/backblaze_survival_analysis_prep.

Source

Raw data from https://www.backblaze.com/b2/hard-drive-test-data.html. Data have been processed to get a start-stop `data.frame` format.

`logLik.ddhazard` *Log likelihood of mean path of* `ddhazard` *object*

Description

Computes the log likelihood of (a potentially new) data set given the estimated:

$$E_\theta(\alpha_1|y_{1:d}), E_\theta(\alpha_2|y_{1:d}), ..., E_\theta(\alpha_d|y_{1:d})$$

of the `ddhazard` object. Note that this is not the log likelihood of the observed data given the outcome.

Usage

```
## S3 method for class 'ddhazard'
logLik(object, data = NULL, id, ...)
```

Arguments

`object`	an object of class `ddhazard`.
`data`	new data to evaluate the likelihood for.
`id`	the individual identifiers as in `ddhazard`.
`...`	unused.

Examples

```
library(dynamichazard)
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3600,
 Q_0 = diag(1, 2), Q = diag(1e-4, 2), by = 50,
 control = list(method = "GMA"))
logLik(fit)
```

logLik.PF_clouds *Log likelihood of a* PF_clouds *object*

Description

Computes the log likelihood using the forward filter clouds. See the `vignette("Particle_filtering", "dynamichazard"` for details.

Usage

```
## S3 method for class 'PF_clouds'
logLik(object, ...)
```

Arguments

object	an object of class PF_clouds.
...	unused.

Value

The log-likelihood value given the observed data and set of parameter used when simulating the clouds.

PF_EM *EM estimation with particle filters*

Description

Method to estimate the hyper parameters with an EM algorithm.

Usage

```
PF_EM(formula, data, model = "logit", by, max_T, id, a_0, Q_0, Q, order = 1,
  control = list(), trace = 0)
```

Arguments

formula	coxph like formula with Surv(tstart, tstop, event) on the left hand site of ~.
data	data.frame or environment containing the outcome and co-variates.
model	either 'logit' for binary outcomes or 'exponential' for piecewise constant exponential distributed arrival times.
by	interval length of the bins in which parameters are fixed.
max_T	end of the last interval interval.

id	vector of ids for each row of the in the design matrix.
a_0	vector a_0 for the initial coefficient vector for the first iteration (optional). Default is estimates from static model (see static_glm).
Q_0	covariance matrix for the prior distribution.
Q	initial covariance matrix for the state equation.
order	order of the random walk.
control	list of control variables (see the control section below).
trace	argument to get progress information. Zero will yield no info and larger integer values will yield incrementally more information.

Details

See vignette("Particle_filtering", "dynamichazard") for details.

Value

An object of class PF_EM.

Control

The control argument allows you to pass a list to select additional parameters. See vignette("Particle_filtering", "
for details. Unspecified elements of the list will yield default values.

method method for forward, backward and smoothing filter.

smoother smoother to use.

N_fw_n_bw number of particles to use in forward and backward filter.

N_first number of particles to use at time 0 and time $d + 1$.

N_smooth number of particles to use in particle smoother.

eps convergence threshold in EM method.

n_max maximum number of iterations of the EM algorithm.

n_threads maximum number threads to use in the computations.

forward_backward_ESS_threshold required effective sample size to not re-sample in the particle filters.

seed seed to set at the start of every EM iteration.

Examples

```
#####
# Fit model with lung data set from survival
# Warning: this has a longer computation time

## Not run:
library(dynamichazard)
.lung <- lung[!is.na(lung$ph.ecog), ]
set.seed(43588155)
pf_fit <- PF_EM(
```

```
 Surv(time, status == 2) ~ ph.ecog + age,
 data = .lung, by = 50, id = 1:nrow(.lung),
 Q_0 = diag(1, 3), Q = diag(1, 3),
 max_T = 800,
 control = list(
   N_fw_n_bw = 500,
   N_first = 2500,
   N_smooth = 2500,
   n_max = 50,
   n_threads = parallel::detectCores()),
 trace = 1)

# Plot state vector estimates
plot(pf_fit, cov_index = 1)
plot(pf_fit, cov_index = 2)
plot(pf_fit, cov_index = 3)

# Plot log-likelihood
plot(pf_fit$log_likes)
## End(Not run)

#####
# Can be compared with this example from ?coxph in R 3.4.1. Though, the above
# only has a linear effect for age

## Not run:
cox <- coxph(
 Surv(time, status) ~ ph.ecog + tt(age), data= .lung,
 tt=function(x,t,...) pspline(x + t/365.25))
cox
## End(Not run)
```

`plot.ddhazard` *Plots for* `ddhazard` *object*

Description

Plot of estimated state space variables from a `ddhazard` fit.

Usage

```
## S3 method for class 'ddhazard'
plot(x, xlab = "Time", ylab = "Hazard", type = "cov",
  plot_type = "l", cov_index, ylim, col = "black", add = F,
  do_alter_mfcol = T, level = 0.95, ddhazard_boot, ...)
```

Arguments

x	result of `ddhazard` call.

xlab, ylab, ylim, col	arguments to override defaults set in the function.
type	type of plot. Currently, only "cov" is available for plot of the state space parameters.
plot_type	the type argument passed to plot.
cov_index	the index (indices) of the state space parameter(s) to plot.
add	FALSE if you want to make a new plot.
do_alter_mfcol	TRUE if the function should alter par(mfcol) in case that cov_index has more than one element.
level	level (fraction) for confidence bounds.
ddhazard_boot	object from a ddhazard_boot call which confidence bounds will be based on and where bootstrap samples will be printed with a transparent color.
...	arguments passed to plot.default or lines depending on the value of add.

Details

Creates a plot of state variables or adds state variables to a plot with indices cov_index. Pointwise 1.96 std. confidence intervals are provided with the smoothed co-variance matrices from the fit.

Examples

```
library(dynamichazard)
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3600,
 Q_0 = diag(1, 2), Q = diag(1e-4, 2), by = 50,
 control = list(method = "GMA"))
plot(fit)
plot(fit, cov_index = 2)
```

---

plot.ddhazard_space_errors

State space error plot

Description

Plot function for state space errors from ddhazard fit.

Usage

```
## S3 method for class 'ddhazard_space_errors'
plot(x, mod, cov_index = NA, t_index = NA,
  p_cex = par()$cex * 0.2, pch = 16, ylab = "Std. state space error",
  x_tick_loc = NA, x_tick_mark = NA, xlab = "Time", ...)
```

Arguments

x	result of residuals with a 'type' argument which yields state space errors.
mod	the ddhazard result used in the residuals call.
cov_index	the indices of state vector errors to plot. Default is to use all.
t_index	the bin indices to plot. Default is to use all bins.
p_cex	cex argument for the points
pch, ylab, xlab	arguments to override defaults set in the function.
x_tick_loc, x_tick_mark	at and labels arguments passed to axis.
...	arguments passed to plot.default.

---

plot.PF_clouds *Plot of clouds from a* PF_clouds *object*

Description

Plots mean curve along with quantiles through time for the forward, backward or smoothed clouds.

Usage

```
## S3 method for class 'PF_clouds'
plot(x, y, type = c("smoothed_clouds", "forward_clouds",
  "backward_clouds"), ylim, add = FALSE, qlvls = c(0.05, 0.5, 0.95),
  pch = 4, lty = 1, ..., cov_index)
```

Arguments

x	an object of class PF_clouds.
y	unused.
type	parameter to specify which cloud to plot.
ylim	ylim passed to matplot.
add	TRUE if a new plot should not be made.
qlvls	vector of quantile levels to be plotted.
pch	pch argument for the quantile points.
lty	lty argument for the mean curves.
...	unused.
cov_index	indices of the state vector to plot. All are plotted if this argument is omitted.

Value

List with quantile levels and mean curve.

plot.PF_EM *Plot for a* PF_EM *object*

Description

Short hand to call plot.PF_clouds.

Usage

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

Arguments

x	an object of class PF_EM.
y	unused.
...	arguments to plot.PF_clouds.

Value

See plot.PF_clouds

predict.ddhazard *Predict method for* ddhazard

Description

Predict method for ddhazard.

Usage

```
## S3 method for class 'ddhazard'
predict(object, new_data, type = c("response", "term"),
  tstart = "start", tstop = "stop", use_parallel = F, sds = F,
  max_threads = getOption("ddhazard_max_threads"), ...)
```

Arguments

object	result of a ddhazard call.
new_data	new data to base predictions on.
type	either "response" for predicted probability of death or "term" for predicted terms in the linear predictor.
tstart	name of the start time column in new_data. It must be on the same time scale as the tstart used in the Surv(tstart, tstop, event) in the formula passed to ddhazard.

tstop	same as tstart for the stop argument.
use_parallel	TRUE if computation for type = "response" should be computed in parallel with the mcmapply. Notice the limitation in the help page of mcmapply.
sds	TRUE if point wise standard deviation should be computed. Convenient if you use functions like ns and you only want one term per term in the right hand site of the formula used in ddhazard.
max_threads	maximum number of threads to use. -1 if it should be determined by a call to detectCores.
...	not used.

Term

The result with type = "term" is a list with the following elements

terms is a 3D array. The first dimension is the number of bins, the second dimension is rows in new_data and the last dimension is the state space terms.

sds similar to terms for the point wise confidence intervals using the smoothed co-variance matrices.

fixed_terms vector of the fixed effect terms for each observation.

Response

The result with type = "response" is a list with the elements below. The function check if there are columns in new_data which's names match tstart and tstop. If not, then each row in new data will get a predicted probability of dying in every bin.

fits fitted probability of dying.

istart vector with the index of the first bin the elements in fits is in.

istop vector with the index of the last bin the elements in fits is in.

Examples

```
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3600,
 Q_0 = diag(1, 2), Q = diag(1e-4, 2), by = 50,
 control = list(method = "GMA"))
predict(fit, type = "response", new_data =
 data.frame(time = 0, status = 2, bili = 3))
predict(fit, type = "term", new_data =
 data.frame(time = 0, status = 2, bili = 3))
```

print.ddhazard_boot *Summary statistics for a* ddhazard_boot *object*

Description

Arguments have the same effects as for an object from a boot call. See print.

Usage

```
## S3 method for class 'ddhazard_boot'
print(x, digits = getOption("digits"),
  index = 1L:ncol(boot.out$t), ...)
```

Arguments

x	returned object from a ddhazard_boot call.
digits	the number of digits to be printed in the summary statistics.
index	indices indicating for which elements of the bootstrap output summary statistics are required.
...	not used.

See Also

ddhazard_boot

print.summary.ddhazard *Summarizing Dynamic Hazard Models Fits*

Description

The sd printed for time-varying effects are point-wise standard deviations from the smoothed covariance matrices.

Usage

```
## S3 method for class 'summary.ddhazard'
print(x, digits = getOption("digits"), ...)

## S3 method for class 'ddhazard'
summary(object, var_indices = 1:ncol(object$state_vecs),
  max_print = 10, ...)
```

Arguments

x	object returned from summary.ddhazard.
digits	number of digits to print.
...	not used.
object	object returned from ddhazard.
var_indices	variable indices to print for time-varying effects.
max_print	maximum number of time points to print coefficients at.

---

residuals.ddhazard *Residuals method for* ddhazard

Description

Residuals method for the result of a ddhazard call.

Usage

```
## S3 method for class 'ddhazard'
residuals(object, type = c("std_space_error",
  "space_error", "pearson", "raw"), data = NULL, ...)
```

Arguments

object	result of ddhazard call.
type	type of residuals. Four possible values: "std_space_error", "space_error", "pearson" and "raw". See the sections below for details.
data	data.frame with data for the Pearson or raw residuals. This is only needed if the data set is not saved with the object. Must be the same data set used in the initial call to ddhazard.
...	not used.

Pearson and raw residuals

Is the result of a call with a type argument of either "pearson" or "raw" for Pearson residuals or raw residuals. Returns a list with class "ddhazard_residual" with the following elements.

residuals list of residuals for each bin. Each element of the list contains a 2D array where the rows corresponds to the passed data and columns are the residuals (residuals), estimated probability of death (p_est), outcome (Y) and row number in the initial data set (row_num). The data rows will only have a residuals in a given risk list if they are at risk in that risk set.

type the type of residual.

State space errors

Is the result of a call with a type argument of either "std_space_error" or "space_error". The former is for standardized residuals while the latter is non-standardized. Returns a list with class. "ddhazard_space_errors" with the following elements:

residuals 2D array with either standardized or non-standardized state space errors. The row are bins and the columns are the parameters in the regression.

standardize TRUE if standardized state space errors.

Covariances 3D array with the smoothed co-variance matrix for each set of the state space errors.

Examples

```
library(dynamichazard)
fit <- ddhazard(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3600,
 Q_0 = diag(1, 2), Q = diag(1e-4, 2), by = 50,
 control = list(method = "GMA"))
resids <- residuals(fit, type = "pearson")$residuals
head(resids[[1]])
head(resids[[2]])
```

---

static_glm *Static glm fit*

Description

Method to fit a static model corresponding to a ddhazard fit. The method uses weights to ease the memory requirements. See get_survival_case_weights_and_data for details on weights.

Usage

```
static_glm(formula, data, by, max_T, ..., id, family = "logit", model = F,
  weights, risk_obj = NULL, speedglm = F, only_coef = FALSE, mf,
  method_use = c("glm", "speedglm", "parallelglm"),
  n_threads = getOption("ddhazard_max_threads"))
```

Arguments

formula	coxph like formula with Surv(tstart, tstop, event) on the left hand site of ~.
data	data.frame or environment containing the outcome and co-variates.
by	interval length of the bins in which parameters are fixed.
max_T	end of the last interval interval.
...	arguments passed to glm or speedglm. If only_coef = TRUE then the arguments are passed to glm.control if glm is used.

id	vector of ids for each row of the in the design matrix.
family	"logit" or "exponential" for a static equivalent model of ddhazard.
model	TRUE if you want to save the design matrix used in glm.
weights	weights to use if e.g. a skewed sample is used.
risk_obj	a pre-computed result from a get_risk_obj. Will be used to skip some computations.
speedglm	depreciated.
only_coef	TRUE if only coefficients should be returned. This will only call the speedglm.wfit or glm.fit which will be faster.
mf	model matrix for regression. Needed when only_coef = TRUE
method_use	method to use for estimation. glm uses glm.fit, speedglm uses speedglm.wfit and parallelglm uses a parallel C++ version glm.fit which only gives the coefficients.
n_threads	number of threads to use when method_use is "parallelglm".

Value

The returned list from the glm call or just coefficients depending on the value of only_coef.

Examples

```
library(dynamichazard)
fit <- static_glm(
 Surv(time, status == 2) ~ log(bili), pbc, id = pbc$id, max_T = 3600,
 by = 50)
fit$coefficients
```

Index

∗Topic **datasets**
hds, 12

boot, *7*, *21*

coxph, *3*, *9*, *14*, *23*

ddFixed, 2
ddFixed_intercept *(*ddFixed*)*, 2
ddhazard, *2*, 3, *7*, *10, 11*, *13*, *16–20*, *22–24*
ddhazard_app, *4*, *6*, 6
ddhazard_boot, *6*, 7, *17*, *21*
detectCores, *20*

gam, *10*
get_risk_obj, *4, 5*, 8, *9*, *24*
get_survival_case_weights_and_data, 9, *23*
glm, *23, 24*
glm.control, *23*
glm.fit, *24*

hatvalues.ddhazard, 11
hds, 12

logLik.ddhazard, 13
logLik.PF_clouds, 14

matplot, *18*
mclapply, *8*
mcmapply, *20*
Module, *4*

ns, *20*

PF_EM, 14
plot, *6, 7*
plot.ddhazard, 16
plot.ddhazard_space_errors, 17
plot.default, *17, 18*
plot.PF_clouds, 18, *19*
plot.PF_EM, 19
predict, *6*
predict.ddhazard, 19
print, *21*
print.ddhazard_boot, 21
print.summary.ddhazard, 21

residuals, *6*, *18*
residuals.ddhazard, 22

speedglm, *23, 24*
speedglm.wfit, *24*
static_glm, *3, 4*, *6*, *10*, *15*, 23
summary.ddhazard *(*print.summary.ddhazard*)*, 21
Surv, *3*, *8, 9*, *14*, *19*, *23*

terms, *4*