Package 'dynamichazard'

September 16, 2017

Type Package

Title Dynamic Hazard Models using State Space Models

Version 0.4.0

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 2017-09-16 18:57:46 UTC

R topics documented:

`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(n)
```

Arguments

`object`	Expression that would be used in formula. E.g. `x` or `poly(x, degree = 3)`
`n`	Number of rows in the data frame the data for estimation

Examples

```
# All these call with give the same result where
# 'data' is a hypothetical data frame. We can get a
# time-invariant estimate for 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(1) + x1, data)
ddhazard(Surv(stop, event)
    ~ -1 + ddFixed(1) + x1, data)
ddhazard(Surv(stop, event)
    ~ ddFixed(rep(1, nrow(data))) + x1, data)
ddhazard(Surv(stop, event)
    ~ -1 + ddFixed(rep(1, nrow(data))) + x1, data)
ddhazard(Surv(stop, event)
    ~ ddFixed_intercept(nrow(data)) + x1, data)
ddhazard(Surv(stop, event)
    ~ -1 + ddFixed_intercept(nrow(data)) + x1, data)

## End(Not run)
```

`ddhazard` *Function to fit dynamic discrete hazard models*

Description

Function to fit dynamic discrete 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"`, `"exp_clip_time_w_jump"`, `"exp_clip_time"` or `"exp_bin"` for the discrete time function using the logistic link function in the first case or for the continuous time model with different estimation method in the three latter cases (see the ddhazard vignette for details of the methods)

by	Interval length of the bins in which parameters are fixed
max_T	End of the last interval. The last stop time with an event is selected if the parameter is omitted
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 a binary regression 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, 2by, ..., max_T. See the vignette 'ddhazard' for more details

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

The model is specified through the model argument. See the model in the argument above for details. 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 models 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 fahrmeier_94. 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

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

`order` Order of the random walk

`F_` Matrix with that map transition 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

`hazard_func` Hazard function

`hazard_first_deriv` First derivative of the hazard function with respect to the linear predictor

Control

The `control` argument allows you to pass a `list` to select additional parameters. See the vignette 'ddhazard' for more information on hyper parameters. 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 for the Extended Kalman filter

`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 call to `residuals`, `plot` and `logLike`. Can be set to `FALSE` to save memory

`save_data` `TRUE` if you want to save the list `data` argument. It may be needed for later call to `residuals`, `plot` and `logLike`. Can be set to `FALSE` to save memory

`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`

`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 'Bootstrap illustration'. 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 sample 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 returned from the boot function

See Also

ddhazard, plot

get_risk_obj *Get the risk set at each bin over an equal distance grid*

Description

Get the risk set at each bin over an equal distance 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
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/FALSE for whether the model outcome is discrete. For example, a logit model is discrete whereas what is coined an 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

`get_survival_case_weights_and_data` *Static GLM fit for survival models*

Description

Function used to get design matrix and weights for a static fit for survivals models where observations are binned into intervals

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`	Length of each intervals that cases are binned into
`max_T`	The end time of the last bin
`id`	The id for each row in `data`. This is important when variables are time varying
`init_weights`	Weights for the rows `data`. Useful with skewed sampling and will be used when computing the final weights
`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 like the logistic model. Affects how deaths are included when individuals have time varying coefficients
`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 "ddhazard" for details)

If use_weights = FALSE then the two 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) (see reference)

Value

Returns a data frame with the design matrix from the formula 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

---

hatvalues.fahrmeier_94

Hat values for ddhazard

Description

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

Usage

```
## S3 method for class 'fahrmeier_94'
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 ddhazard 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

`logLik.fahrmeier_94` *Log likelihood of smoothed state vector of a* `fahrmeier_94` *object*

Description

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

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

from the `fahrmeier_94` class object. Note that this is not the log likelihood of the observed data given the outcome.

Usage

```
## S3 method for class 'fahrmeier_94'
logLik(object, data = NULL, id, ...)
```

Arguments

`object`	an object of class `fahrmeier_94`.
`data`	new data to evaluate the likelihood for.
`id`	the individual identifiers as in ddhazard.
`...`	unused.

`logLik.PF_clouds` *Log-Likelihood of a* `PF_clouds` *object*

Description

Computes the log-likelihood using the forward filter clouds. See the particle_filter vignette 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.

---

model.frame.ddformula *model.frame and model.matrix for ddformula*

Description

Functions added to handle fixed (time-invariant) intercept coefficient for ddhazard. model.frame.ddformula always has drop.unused.levels = FALSE regardless of the input.

Usage

```
## S3 method for class 'ddformula'
model.frame(formula, data = NULL, subset = NULL,
  na.action = na.fail, xlev = NULL, ...)

## S3 method for class 'ddformula'
model.matrix(object, data = environment(object),
  contrasts.arg = NULL, xlev = NULL, ...)
```

Arguments

formula	Same as model.frame.default
data	Same as model.matrix.default
subset	Same as model.frame.default
na.action	Same as model.frame.default
xlev	Same as model.frame.default
...	Unused
object	Same as model.matrix.default
contrasts.arg	Same as model.matrix.default

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. The last stop time with an event is selected if the parameter is omitted
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 an larger integer values will yield incrementally more information.

Details

See the particle_filter vignette for details.

Value

An object of class PF_EM.

Control

The `control` argument allows you to pass a `list` to select additional parameters. See the vignette 'ddhazard' for more information on hyper parameters. Unspecified elements of the list will yield default values

`method` Method for forward, backward and smoothing filter. See the particle_filter vignette for details.

`smoother` Smoother to use. See the particle_filter vignette for details.

`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 threhshold 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.fahrmeier_94` *Plots for* `ddhazard`

Description

Plot to illustrate the estimate state space variables from a `ddhazard` fit

Usage

```
## S3 method for class 'fahrmeier_94'
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` 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

plot.fahrmeier_94_SpaceErrors *State space error plot*

Description

Plot function for state space errors from ddhazard fit

Usage

```
## S3 method for class 'fahrmeier_94_SpaceErrors'
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 for 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 which is likely what you want if the state space errors are standardized
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

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 cloud.

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.fahrmeier_94 *Predict function for the result of* ddhazard

Description

Predict function for the result of ddhazard

Usage

```
## S3 method for class 'fahrmeier_94'
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 corresponds to 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 parallel package
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 of 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 of `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

---

`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.fahrmeier_94 *Print function for* ddhazard *result*

Description

The sd printed for time-varying effects are point-wise standard deviations from either the filter with smoothing

Usage

```
## S3 method for class 'fahrmeier_94'
print(x, var_indices = 1:ncol(x$state_vecs),
  time_indices = 1:nrow(x$state_vecs), digits = getOption("digits"), ...)
```

Arguments

x	Object returned from ddhazard
var_indices	Variable indices to print for time-varying effects
time_indices	Time intervals to print for time-varying effects
digits	Number of digits to print
...	Not used

residuals.fahrmeier_94 *Residuals for* ddhazard

Description

Residuals function for the result of a ddhazard fit

Usage

```
## S3 method for class 'fahrmeier_94'
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 Pearson or raw residuals
...	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 `"fahrmeier_94_res"` 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 dataset (`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 `"fahrmeier_94_SpaceErrors"` 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

---

`static_glm` *Function to make a static glm fit*

Description

Function to make a static glm fit

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`	Length of each intervals that cases are binned into
`max_T`	The end time of the last bin
`...`	arguments passed to `glm` or `speedglm`. If `only_coef = TRUE` then the arguments are passed to `glm.control` if `glm` is used
`id`	The id for each row in `data`. This is important when variables are time varying

family	"logit" or "exponential" for the static equivalent model of ddhazard
model	TRUE if you want to save the design matrix used in glm
weights	weights if a skewed sample or similar 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".

Details

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

Value

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

Index