Package: FunSurv (via r-universe)

March 18, 2025

Title Modeling Time-to-Event Data with Functional Predictors

Version 1.0.0

Description A collection of methods for modeling time-to-event data using both functional and scalar predictors. It implements functional data analysis techniques for estimation and inference, allowing researchers to assess the impact of functional covariates on survival outcomes, including time-to-single event and recurrent event outcomes.

Depends R (>= 3.5.0)

License MIT + file LICENSE

URL https://github.com/zifangkong/FunSurv

BugReports https://github.com/zifangkong/FunSurv/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports MFPCA, MASS, funData, Matrix, ggplot2, methods, reda (>= 0.5.0)

Config/pak/sysreqs libfftw3-dev

Repository https://zifangkong.r-universe.dev

RemoteUrl https://github.com/zifangkong/funsurv

RemoteRef HEAD

RemoteSha 303a3caca793da8d2eccd8d96f1c359970872342

Contents

`ar1_cor` *Construct an AR(1) correlation matrix*

Description

Construct an AR(1) correlation matrix

Usage

```
ar1_cor(n, rho)
```

Arguments

`n`	number of events for each subject
`rho`	autoregressive correlation

Value

A n by n matrix

Examples

```
## Generate AR(1) structure
ar1_cor(n = 5, rho = 0.3)

## first derivative of the AR(1) structure with respect to rho
dar1_cor.drho(n = 5, rho = 0.3)
```

`AR1_FRAILTY` *Fit a Functional Regression with AutoregressIve fraiLTY (FRAILTY) model for Recurrent Event Data*

Description

Jointly model longitudinal measurements and recurrent events, accommodating both scalar and functional predictors while capturing time-dependent correlations among events. The FRAILTY method employs a two-step estimation procedure. First, functional principal component analysis through conditional expectation (PACE) is applied to extract key temporal features from sparse, irregular longitudinal data. Second, the obtained scores are incorporated into a dynamic recurrent frailty model with an autoregressive structure to account for within-subject correlations across recurrent events. This function works only for univariate functional data.

Usage

```
AR1_FRAILTY(
  formula,
  sdat,
  fdat,
  para0 = c(0.5, 0.5),
  nbasis = 10,
  pve = 0.9,
  npc = NULL,
  makePD = FALSE,
  cov.weight.type = c("none", "counts"),
  iter.max = 50,
  eps = 1e-06
)
```

Arguments

`formula`	A formula, with the response on the left of a ~ operator being a `Recur` object as returned by function `Recur` in **reda**, and scalar covariates on the right.
`sdat`	A data frame containing subject IDs, time-to-event outcomes (starting time, end point, censoring time and event status), and scalar covariates
`fdat`	A data frame containing subject IDs, longitudinal measurements, and the corresponding time points for each measurement.
`para0`	A vector of initial values for θ^2 and auto-regressive coefficient ρ. Both default to 0.5.
`nbasis`	An integer, representing the number of B-spline basis functions used for estimation of the mean function and bivariate smoothing of the covariance surface. Defaults to `10` (cf. `fpca.sc` in **refund**).
`pve`	A numeric value between 0 and 1, the proportion of variance explained: used to choose the number of principal components. Defaults to `0.9` (cf. `fpca.sc` in **refund**).
`npc`	An integer, giving a prespecified value for the number of principal components. Defaults to `NULL`. If given, this overrides `pve` (cf. `fpca.sc` in **refund**).
`makePD`	Logical: should positive definiteness be enforced for the covariance surface estimate? Defaults to `FALSE` (cf. `fpca.sc` in **refund**).
`cov.weight.type`	The type of weighting used for the smooth covariance estimate. Defaults to `"none"`, i.e. no weighting. Alternatively, `"counts"` (corresponds to `fpca.sc` in **refund**) weights the pointwise estimates of the covariance function by the number of observation points.
`iter.max`	Maximum number of iterations for both inner iteration and outer iteration. Defaults to `50`.
`eps`	Tolerance criteria for a possible infinite coefficient value. Defaults to `1e-6`.

Details

Model specification:

Let T_{ij} denote the time of the jth event for subject i, and let C_i represent the censoring time. The observed event time, accounting for right censoring, is $\widetilde{T}_{ij} = \min(T_{ij}, C_i)$, and $\delta_{ij} = I(T_{ij} \leq C_i)$ serves as an indicator of whether the jth event for subject i is observed. The hazard function is specified as

$$h(t; \boldsymbol{Z}_i, X_i(\cdot)) = h_0(t - t_{i,j-1}) \exp(\eta_{ij}),$$

where $h_0(\cdot)$ is the baseline hazard function, and $\eta_{ij} = \boldsymbol{\alpha}^\top \boldsymbol{Z}_i + \int_{t_{i,j-1}}^{t} X_i(s)\beta(s)ds + v_{ij}$. Here, $t_{i,j-1}$ is the previous event time with $t_{i0} = 0$. $\boldsymbol{\alpha}$ is the fixed effect parameter associated with the time-invariant covariates $\boldsymbol{Z}_i$, and $\beta(t)$ is a time-varying coefficient that captures the effect of functional predictor $X_i(t)$ on the hazard rate of recurrent events.

Value

A funsurv object containing the following components:

`beta`	Estimation of coefficients of scalar covariates and FPC scores. Including estimated values, standard errors, and p-values
`beta_vcov`	Estimated variance-covariance of the estimates of beta
`eAR`	Estimation of variance components (θ^2 and ρ)
`eAR_vcov`	Estimated variance of estimates of θ^2 and ρ
`frailties`	Estimated frailty terms (random effects)
`basesurv`	Estimated baseline survival probability
`time`	Time points associated with baseline survival probability
`FPC`	Functional principal components

See Also

Recur

PACE

Examples

```
data(simDat)

fit <- AR1_FRAILTY(Recur(t_start %to% t_stop, id, status) ~ z1,
                   sdat = sdat, fdat = fdat, iter.max = 30)

summary(fit)
```

basesurv — *A function to obtain the baseline survival function*

Description

A function to obtain the baseline survival function

Usage

```
basesurv(object)
```

Arguments

`object`	A funsurv object

Value

A data frame including time and baseline survival

Examples

```
data(simDat)

fit <- AR1_FRAILTY(Recur(t_start %to% t_stop, id, status) ~ z1,
                   sdat = sdat, fdat = fdat, iter.max = 30)

plot(fit, what = "basesurv")
```

dar1_cor.drho — *First derivative of AR(1) correlation matrix with respect to the auto-regressive coefficient*

Description

First derivative of AR(1) correlation matrix with respect to the auto-regressive coefficient

Usage

```
dar1_cor.drho(n, rho)
```

Arguments

`n`	number of events for each subject
`rho`	autoregressive correlation

Value

A n by n inverse matrix

plot.funsurv *Plot method for 'funsurv' objects*

Description

Plot method for 'funsurv' objects

Usage

```
## S3 method for class 'funsurv'
plot(x, what = c("beta", "fpc", "basesurv"), ...)
```

Arguments

`x`	A funsurv object
`what`	A character string specifying what to be plotted. Use `what = "beta"` to plot the estimated $\beta(t)$. Use `what = "fpc"` to plot the functional principal components associated with the the longitudinal measurements. Use `what = "basesurv"` to plot the baseline survival probabilities.
`...`	additional graphical parameters to be passed to methods.

Value

A ggplot object ...

Examples

```
data(simDat)
fit <- AR1_FRAILTY(Recur(t_start %to% t_stop, id, status) ~ z1,
                   sdat = sdat, fdat = fdat, iter.max = 30)
plot(fit, what = "beta")
plot(fit, what = "fpc")
plot(fit, what = "basesurv")
```

simDat *Simulated datasets for demonstration*

Description

The dataset was generated based on the proposed model $h(t; \boldsymbol{Z}_i, X_i(\cdot)) = h_0(t - t_{i,j-1}) \exp(\eta_{ij})$, where $h_0(\cdot)$ is the baseline hazard function generated from a Weibull distribution. $\eta_{ij} = \boldsymbol{\alpha}^\top \boldsymbol{Z}_i + \int_{t_{i,j-1}}^{t} X_i(s)\beta(s)ds + v_{ij}$. $\boldsymbol{\alpha}$ is the fixed effect parameter associated with the time-invariant covariates $\boldsymbol{Z}_i$, and $\beta(t)$ is a time-varying coefficient that captures the effect of functional predictor $X_i(t)$ on the hazard rate of recurrent events. The simulated dataset is organized into two data frames: a survival data frame (`sdat`) and a functional data frame (`fdat`). The variables in each data frame are listed below:

Usage

```
data(simDat)
```

Format

A list with two data frame:

sdat Survival data; a data frame with xxx rows and xxx variables:

- **id** Subjects identification
- **event** A sequence of the number of events per subject
- **t_start** Event starting time
- **t_end** Event end time
- **censoring_time** Event censoring time
- **status** Event status: `status=1` if event is observed and `status=0` if event is censored
- **z1** A univariate scalar covariates. Can be extended to multiple scalar covariates

fdat Functional data; a data frame with xxx rows and xxx variables:

- **id** Subjects identification
- **time** Time points for each longitudinal measurement
- **x** Longitudinal measurements at distinct time points

Source

Simulated data