Package 'rfcox'

May 22, 2025

Type Package

Title Robust functional Cox regression Model

Version 1.0

Depends R (>= 3.5.0), fda, MASS, survival, coxrobust

Imports expm, fda.usc, goffda, pcaPP

LazyLoad yes

ByteCompile TRUE

Encoding UTF-8

Maintainer Ufuk Beyaztas <ufukbeyaztas@gmail.com>

Description

Functions for implementing robust estimation methods for functional Cox regression model.

License GPL-3

R topics documented:

`data_generation` *Data Generation for Robust Functional Cox Regression Simulations*

Description

Generates synthetic data for evaluating robust functional Cox regression models, including both functional and scalar covariates, censoring, and contamination mechanisms.

Usage

```
data_generation(n = 1000, j = 101, p = 4, gamma0 = rep(0.5, p),
                h0t = 1, cenrate = 0.5, tau = 1.55, mev = 0.5,
                out.p = 0)
```

Arguments

n	Number of observations to generate.
j	Number of grid points used to discretize the functional domain \([0,1]\).
p	Number of scalar covariates.
gamma0	True regression coefficients for the scalar covariates.
h0t	Baseline hazard rate (assumed constant).
cenrate	Target censoring rate.
tau	Upper bound for censoring time sampling (via uniform distribution).
mev	Measurement error variance added to functional predictors.
out.p	Proportion of observations to be contaminated as outliers.

Details

This function simulates data under a robust functional Cox regression framework with one functional covariate and multiple scalar covariates. The functional predictor is generated from a truncated Karhunen–Loeve expansion with correlated components, while scalar covariates follow a multivariate Gaussian distribution with specified correlation.

The true coefficient function is a smooth nonlinear combination of sinusoidal and Gaussian terms, reflecting rich temporal effects. Survival times are drawn from an exponential distribution with a hazard dependent on both the functional covariate (via numerical integration) and scalar predictors. Right-censoring is imposed via independent uniform random times.

Measurement error is introduced into the functional predictor to reflect realistic noise. Additionally, contamination can be injected via out.p to simulate outlier scenarios, affecting both the functional trajectories and survival times. Outlier functional data are perturbed using a magnitude-shift Ornstein–Uhlenbeck process, while survival times are replaced with extreme early or late values.

Value

A list with the following components:

time	Observed time-to-event or censoring (numeric vector of length n).
event	Event indicator (1 = event, 0 = censored).
Xt	Noisy functional predictor matrix (dimension n x j).
Xt_true	True smooth functional predictor matrix (before noise).
Z	Scalar covariate matrix (dimension n x p).
beta	True coefficient function evaluated on the grid points.
gamma	True scalar coefficients.
gp	Grid points over domain [0,1].

Note

This function is primarily intended for simulation studies examining robustness properties in functional Cox regression modeling. For robust estimation methods, see the corresponding functions using RFPCA and M-type partial likelihood.

Author(s)

Gizel Bakicierler Sezer and Ufuk Beyaztas

Examples

```
# Generate synthetic data with default settings
sim_data <- data_generation()
```

`rfcox` *Robust Functional Cox Regression Model with Scalar and Functional Covariates*

Description

Performs robust estimation of the functional Cox regression model (RFLCRM) by combining projection-pursuit-based robust functional principal component analysis (RFPCA) with an M-type robust partial likelihood estimator. This approach is designed to handle outliers in both functional predictors and survival responses.

Usage

```
rfcox(time, status, X, Z, trunc = 0.9, nb = NULL, gp = NULL,
      f.weight = c("linear", "quadratic", "exponential"))
```

Arguments

`time`	A numeric vector of observed event or censoring times.
`status`	A numeric vector of event indicators: 1 for event, 0 for censored.
`X`	A matrix of functional predictors (dimension `n` x `j`), observed over a common grid.
`Z`	A matrix of scalar covariates (dimension `n` x `p`).
`trunc`	Truncation parameter for the robust M-estimator used in the Cox regression step.
`nb`	Number of basis functions to be used in RFPCA.
`gp`	Grid points over the functional domain. If `NULL`, defaults to `seq(0,1,length.out = j)`.
`f.weight`	A character string specifying the weight function used in the robust Cox model. One of `"linear"`, `"quadratic"`, or `"exponential"`.

Details

This function implements the robust functional linear Cox regression model (RFLCRM), which integrates two stages:

1. *Robust Functional Principal Component Analysis (RFPCA)*: The functional predictors are decomposed using a projection-pursuit approach that identifies principal components by maximizing robust dispersion measures.
2. *Robust Partial Likelihood Estimation*: The Cox model is then fitted using an M-type estimator that incorporates a robust weighting function to downweight extreme observations in both covariates and survival times.

Value

A list with components:

bhat	Estimated functional coefficient evaluated over the grid points.
gammahat	Estimated scalar coefficients.
concordance	Concordance index for model predictive performance.
model	Fitted robust Cox model object.
rfpca	List of RFPCA outputs: basis evaluations, scores, and eigenfunctions.

Note

Requires the **coxrobust** package for robust estimation and `getRPCA()` for robust FPCA.

Author(s)

Gizel Bakicierler Sezer and Ufuk Beyaztas

References

Bali, J. L., Boente, G., Tyler, D. E., and Wang, J.-L. (2011). Robust functional principal components: A projection-pursuit approach. *Annals of Statistics*, **39**(6), 2852-2882.
Bednarski, T. (1993). Robust estimation in Cox's proportional hazards model. *Scandinavian Journal of Statistics*, **20**(2), 189-204.

See Also

data_generation, rfcox_predict, coxr

Examples

```
# Simulate data
sim_data <- data_generation(n = 500, out.p = 0.05)

# Fit robust functional Cox model
fit <- rfcox(time = sim_data$time, status = sim_data$event,
             X = sim_data$Xt, Z = sim_data$Z, gp = sim_data$gp)
```

---

rfcox_predict *Prediction for Robust Functional Cox Regression Models*

Description

Generates linear predictors and computes the concordance index for new survival data using a fitted robust functional Cox regression model.

Usage

```
rfcox_predict(object, time, status, X, Z)
```

Arguments

object	An object returned by rfcox, containing the fitted robust model and associated RFPCA components.
time	A numeric vector of observed survival or censoring times for the new data.
status	A numeric vector of event indicators for the new data: 1 for event, 0 for censored.
X	A matrix of functional predictors for new observations (dimension: n x j).
Z	A matrix of scalar covariates for new observations (dimension: n x p).

Details

This function performs prediction using a robust functional Cox regression model fitted via rfcox. The procedure includes:

1. Projecting new functional data onto the robust FPCA space from training,
2. Combining FPCA scores with scalar covariates,
3. Predicting risk scores (linear predictors) using the fitted robust Cox model,
4. Calculating the concordance index based on new survival outcomes.

This ensures that prediction is robust to outliers in both functional and scalar covariates.

Value

A list with the following components:

predictions	A numeric vector of linear predictors for the test observations.
concordance	Concordance index comparing predictions to observed survival times and status.

Note

Requires a fitted model object from rfcox. The function uses getRPCA_test for projecting new functional data.

Author(s)

Gizel Bakicierler Sezer and Ufuk Beyaztas

Examples

```
# Simulate training and test data
train_data <- data_generation(n = 500, out.p = 0.05)
test_data <- data_generation(n = 1000)

# Fit the model
fit <- rfcox(time = train_data$time, status = train_data$event,
             X = train_data$Xt, Z = train_data$Z, gp = train_data$gp)

# Predict on test data
pred_out <- rfcox_predict(object = fit,
                          time = test_data$time,
                          status = test_data$event,
                          X = test_data$Xt,
                          Z = test_data$Z)
```

Index