Package ‘pencal’

January 15, 2021

Title Penalized Regression Calibration (PRC)

Version 0.2.2

Description Computes the penalized regression calibration (PRC) method, that allows to predict survival using high-dimensional longitudinal predictors. PRC is described in Signorelli et al. (in review, arXiv preprint: <arXiv:2101.04426>)).

License GPL-3

URL https://mirkosignorelli.github.io/r

Depends R (>= 4.0.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports foreach, doParallel, glmnet, nlme, survival, MASS, stats, survcomp, survivalROC, dplyr

Suggests ptmixed, survminer, knitr, rmarkdown

NeedsCompilation no

Author Mirko Signorelli [aut, cre, cph] (<https://orcid.org/0000-0002-8102-3356>), Pietro Spitali [ctb], Roula Tsonaka [ctb]

Maintainer Mirko Signorelli <msignorelli.rpackages@gmail.com>

Repository CRAN

Date/Publication 2021-01-15 10:50:06 UTC

R topics documented:

`fitted_prclmm` *A fitted PRC LMM*

Description

This list contains a fitted PRC LMM, where the CBOCP is computed using 50 cluster bootstrap samples. It is used to reduce the computing time in the example of the function `performance_prclmm`

Usage

```
data(fitted_prclmm)
```

Format

A list comprising step 2 and step 3 as obtained during the estimation of the PRC LMM

Author(s)

Mirko Signorelli

References

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

See Also

`performance_prclmm`

Examples

```
data(fitted_prclmm)
ls(fitted_prclmm)
```

fit_lmms *Step 1 of PRC-LMM (estimation of the linear mixed models)*

Description

This function performs the first step for the estimation of the PRC-LMM model proposed in Signorelli et al. (2020, in review)

Usage

```
fit_lmms(y.names, fixefs, ranefs, long.data, surv.data, t.from.base,
  n.boots = 0, n.cores = 1, verbose = TRUE)
```

Arguments

`y.names`	character vector with the names of the response variables which the LMMs have to be fitted to
`fixefs`	fixed effects formula for the model, example: `~ time`
`ranefs`	random effects formula for the model, specified using the representation of random effect structures of the `R` package `nlme`
`long.data`	a data frame with the longitudinal predictors, comprehensive of a variable called `id` with the subject ids
`surv.data`	a data frame with the survival data and (if relevant) additional baseline covariates. `surv.data` should at least contain a subject id (called `id`), the time to event outcome (`time`), and binary event variable (`event`)
`t.from.base`	name of the variable containing time from baseline in `long.data`
`n.boots`	number of bootstrap samples to be used in the cluster bootstrap optimism correction procedure (CBOCP). If 0, no bootstrapping is performed
`n.cores`	number of cores to use to parallelize the computation of the CBOCP procedure. If `ncores = 1` (default), no parallelization is done. Pro tip: you can use `parallel::detectCores()` to check how many cores are available on your computer
`verbose`	if `TRUE` (default and recommended value), information on the ongoing computations is printed in the console

Value

A list containing the following objects:

- `call.info`: a list containing the following function call information: `call`, `y.names`, `fixefs`, `ranefs`;
- `lmm.fits.orig`: a list with the LMMs fitted on the original dataset (it should comprise as many LMMs as the elements of `y.names` are);
- `df.sanitized`: a sanitized version of the supplied `long.data` dataframe, without the longitudinal measurements that are taken after the event or after censoring;

- n.boots: number of bootstrap samples;
- boot.ids: a list with the ids of bootstrapped subjects (when n.boots > 0);
- lmms.fits.boot: a list of lists, which contains the LMMs fitted on each bootstrapped datasets (when n.boots > 0).

Author(s)

Mirko Signorelli

References

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

See Also

simulate_prclmm_data, summarize_lmms (step 2), fit_prclmm (step 3), performance_prclmm

Examples

```
# generate example data
set.seed(1234)
p = 4 # number of longitudinal predictors
simdata = simulate_prclmm_data(n = 100, p = p, p.relev = 2,
             seed = 123, t.values = c(0, 0.2, 0.5, 1, 1.5, 2))

# specify options for cluster bootstrap optimism correction
# procedure and for parallel computing
do.bootstrap = FALSE
# IMPORTANT: set do.bootstrap = TRUE to compute the optimism correction!
n.boots = ifelse(do.bootstrap, 100, 0)
parallelize = FALSE
# IMPORTANT: set parallelize = TRUE to speed computations up!
if (!parallelize) n.cores = 1
if (parallelize) {
   # identify number of available cores on your machine
   n.cores = parallel::detectCores()
   if (is.na(n.cores)) n.cores = 1
}

# step 1 of PRC-LMM: estimate the LMMs
y.names = paste('marker', 1:p, sep = '')
step1 = fit_lmms(y.names = y.names,
                 fixefs = ~ age, ranefs = ~ age | id,
                 long.data = simdata$long.data,
                 surv.data = simdata$surv.data,
                 t.from.base = t.from.base,
                 n.boots = n.boots, n.cores = n.cores)
```

fit_prclmm *Step 3 of PRC-LMM (estimation of the penalized Cox model(s))*

Description

This function performs the third step for the estimation of the PRC-LMM model proposed in Signorelli et al. (2020, in review)

Usage

```
fit_prclmm(object, surv.data, baseline.covs = NULL, penalty = "ridge",
  standardize = TRUE, pfac.base.covs = 0, n.alpha.elnet = 11,
  n.folds.elnet = 5, n.cores = 1, verbose = TRUE)
```

Arguments

`object`	the output of step 2 of the PRC-LMM procedure, as produced by the `summarize_lmms` function
`surv.data`	a data frame with the survival data and (if relevant) additional baseline covariates. `surv.data` should at least contain a subject id (called `id`), the time to event outcome (`time`), and binary event variable (`event`)
`baseline.covs`	a formula specifying the variables (e.g., baseline age) in `surv.data` that should be included as baseline covariates in the penalized Cox model. Example: `baseline.covs = '~ baseline.age'`. Default is `NULL`
`penalty`	the type of penalty function used for regularization. Default is `'ridge'`, other possible values are `'elasticnet'` and `'lasso'`
`standardize`	logical argument: should the predicted random effects be standardized when included in the penalized Cox model? Default is `TRUE`
`pfac.base.covs`	a single value, or a vector of values, indicating whether the baseline covariates (if any) should be penalized (1) or not (0). Default is `pfac.base.covs = 0` (no penalization of all baseline covariates)
`n.alpha.elnet`	number of alpha values for the two-dimensional grid of tuning parameteres in elasticnet. Only relevant if `penalty = 'elasticnet'`. Default is 11, so that the resulting alpha grid is c(1, 0.95, 0.90, ..., 0.05, 0)
`n.folds.elnet`	number of folds to be used for the selection of the tuning parameter in elasticnet. Only relevant if `penalty = 'elasticnet'`. Default is 5
`n.cores`	number of cores to use to parallelize the computation of the cluster bootstrap optimism correction procedure. If `ncores = 1` (default), no parallelization is done. Pro tip: you can use `parallel::detectCores()` to check how many cores are available on your computer
`verbose`	if `TRUE` (default and recommended value), information on the ongoing computations is printed in the console

Value

A list containing the following objects:

- `call`: the function call
- `pcox.orig`: the penalized Cox model fitted on the original dataset;
- `surv.data`: the supplied survival data (ordered by subject id)
- `n.boots`: number of bootstrap samples;
- `boot.ids`: a list with the ids of bootstrapped subjects (when `n.boots` > `0`);
- `pcox.boot`: a list where each element is a fitted penalized Cox model for a given bootstrap sample (when `n.boots` > `0`).

Author(s)

Mirko Signorelli

References

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

See Also

`fit_lmms` (step 1), `summarize_lmms` (step 2), `performance_prclmm`

Examples

```
# generate example data
set.seed(1234)
p = 4 # number of longitudinal predictors
simdata = simulate_prclmm_data(n = 100, p = p, p.relev = 2,
             seed = 123, t.values = c(0, 0.2, 0.5, 1, 1.5, 2))

# specify options for cluster bootstrap optimism correction
# procedure and for parallel computing
do.bootstrap = FALSE
# IMPORTANT: set do.bootstrap = TRUE to compute the optimism correction!
n.boots = ifelse(do.bootstrap, 100, 0)
parallelize = FALSE
# IMPORTANT: set parallelize = TRUE to speed computations up!
if (!parallelize) n.cores = 1
if (parallelize) {
   # identify number of available cores on your machine
   n.cores = parallel::detectCores()
   if (is.na(n.cores)) n.cores = 1
}

# step 1 of PRC-LMM: estimate the LMMs
y.names = paste('marker', 1:p, sep = '')
step1 = fit_lmms(y.names = y.names,
```

```
                 fixefs = ~ age, ranefs = ~ age | id,
                 long.data = simdata$long.data,
                 surv.data = simdata$surv.data,
                 t.from.base = t.from.base,
                 n.boots = n.boots, n.cores = n.cores)

# step 2 of PRC-LMM: compute the summaries
# of the longitudinal outcomes
step2 = summarize_lmms(object = step1, n.cores = n.cores)

# step 3 of PRC-LMM: fit the penalized Cox models
step3 = fit_prclmm(object = step2, surv.data = simdata$surv.data,
                   baseline.covs = ~ baseline.age,
                   penalty = 'ridge', n.cores = n.cores)
```

`performance_prclmm` *Predictive performance of the PRC-LMM model*

Description

This function computes the naive and optimism-corrected measures of performance (C index and time-dependent AUC) for the PRC-LMM model proposed in Signorelli et al. (2020, in review). The optimism correction is computed based on a cluster bootstrap optimism correction procedure (CBOCP)

Usage

```
performance_prclmm(step2, step3, times = 1, n.cores = 1, verbose = TRUE)
```

Arguments

`step2`	the output of `summarize_lmms` (step 2 of the estimation of the PRC-LMM model)
`step3`	the output of `fit_prclmm` (step 3 of the estimation of the PRC-LMM model)
`times`	numeric vector with the time points at which to estimate the time-dependent AUC
`n.cores`	number of cores to use to parallelize the computation of the CBOCP procedure. If `ncores = 1` (default), no parallelization is done. Pro tip: you can use `parallel::detectCores()` to check how many cores are available on your computer
`verbose`	if `TRUE` (default and recommended value), information on the ongoing computations is printed in the console

Value

A list containing the following objects:

- `call`: the function call;

- concordance: a data frame with the naive and optimism-corrected estimates of the concordance (C) index;
- tdAUC: a data frame with the naive and optimism-corrected estimates of the time-dependent AUC at the desired time points.

Author(s)

Mirko Signorelli

References

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

See Also

fit_lmms (step 1), summarize_lmms (step 2), fit_prclmm (step 3)

Examples

```
data(fitted_prclmm)

parallelize = FALSE
# IMPORTANT: set parallelize = TRUE to speed computations up!
if (!parallelize) n.cores = 1
if (parallelize) {
   # identify number of available cores on your machine
   n.cores = parallel::detectCores()
   if (is.na(n.cores)) n.cores = 1
}

# compute the performance measures
perf = performance_prclmm(fitted_prclmm$step2, fitted_prclmm$step3,
          times = c(0.5, 1, 1.5, 2), n.cores = n.cores)

# concordance index:
perf$concordance
# time-dependent AUC:
perf$tdAUC
```

---

simulate_prclmm_data   *Simulate data that can be used to fit the PRC-LMM model*

Description

This function allows to simulate a survival outcome from longitudinal predictors. Specifically, the longitudinal predictors are simulated from linear mixed models (LMMs), and the survival outcome from a Weibull model where the time to event depends on the random effects from the LMMs. It is an implementation of the simulation method used in Signorelli et al. (2020, in review)

Usage

```
simulate_prclmm_data(n = 100, p = 10, p.relev = 4, lambda = 0.2,
  nu = 2, seed = 1, base.age.range = c(3, 5), cens.range = c(0.5, 10),
  t.values = c(0, 0.5, 1, 2))
```

Arguments

`n`	sample size
`p`	number of longitudinal outcomes
`p.relev`	number of longitudinal outcomes that are associated with the survival outcome (min: 1, max: p)
`lambda`	Weibull location parameter, positive
`nu`	Weibull scale parameter, positive
`seed`	random seed (defaults to 1)
`base.age.range`	range for age at baseline (set it equal to c(0, 0) if you want all subjects to enter the study at the same age)
`cens.range`	range for censoring times
`t.values`	vector specifying the time points at which longitudinal measurements are collected (NB: for simplicity, this function assumes a balanced designed; however, `pencal` is designed to work both with balanced and with unbalanced designs!)

Value

A list containing the following elements:

- a dataframe `long.data` with data on the longitudinal predictors, comprehensive of a subject id (`id`), baseline age (`base.age`),time from baseline (`t.from.base`) and the longitudinal biomarkers;
- a dataframe `surv.data` with the survival data: a subject id (`id`), baseline age (`baseline.age`), the time to event outcome (`time`) and a binary vector (`event`) that is 1 if the event is observed, and 0 in case of right-censoring;
- `perc.cens` the proportion of censored individuals in the simulated dataset.

Author(s)

Mirko Signorelli

References

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

Examples

```
# generate example data
simdata = simulate_prclmm_data(n = 20, p = 10,
                        p.relev = 4, seed = 1)
# view the longitudinal markers:
library(ptmixed)
make.spaghetti(x = age, y = marker1,
               id = id, group = id,
               data = simdata$long.data,
               legend.inset = - 1)
# proportion of censored subjects
simdata$censoring.prop
# visualize KM estimate of survival
library(survival)
surv.obj = Surv(time = simdata$surv.data$time,
                event = simdata$surv.data$event)
kaplan <- survfit(surv.obj ~ 1,
                  type="kaplan-meier")
plot(kaplan)
```

`simulate_t_weibull` *Generate survival data from a Weibull model*

Description

This function implements the algorithm proposed by Bender et al. (2005) to simulate survival times from a Weibull model

Usage

```
simulate_t_weibull(n, lambda, nu, X, beta, seed = 1)
```

Arguments

`n`	sample size
`lambda`	Weibull location parameter, positive
`nu`	Weibull scale parameter, positive
`X`	design matrix (n rows, p columns)
`beta`	p-dimensional vector of regression coefficients associated to X
`seed`	random seed (defaults to 1)

Value

A vector of survival times

Author(s)

Mirko Signorelli

References

Bender, R., Augustin, T., & Blettner, M. (2005). Generating survival times to simulate Cox proportional hazards models. Statistics in medicine, 24(11), 1713-1723.

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

Examples

```
# generate example data
set.seed(1)
n = 50
X = cbind(matrix(1, n, 1),
   matrix(rnorm(n*9, sd = 0.7), n, 9))
beta = rnorm(10, sd = 0.7)
times = simulate_t_weibull(n = n, lambda = 1, nu = 2,
   X = X, beta = beta)
hist(times, 20)
```

---

summarize_lmms *Step 2 of PRC-LMM (computation of the predicted random effects)*

Description

This function performs the second step for the estimation of the PRC-LMM model proposed in Signorelli et al. (2020, in review)

Usage

```
summarize_lmms(object, n.cores = 1, verbose = TRUE)
```

Arguments

`object`	a list of objects as produced by `fit_lmms`
`n.cores`	number of cores to use to parallelize the computation of the cluster bootstrap optimism correction procedure. If `ncores = 1` (default), no parallelization is done. Pro tip: you can use `parallel::detectCores()` to check how many cores are available on your computer
`verbose`	if `TRUE` (default and recommended value), information on the ongoing computations is printed in the console

Value

A list containing the following objects:

- `call`: the function call
- `ranef.orig`: a matrix with the predicted random effects computed for the original data;
- `n.boots`: number of bootstrap samples;
- `boot.ids`: a list with the ids of bootstrapped subjects (when `n.boots` > `0`);
- `ranef.boot.train`: a list where each element is a matrix that contains the predicted random effects for each bootstrap sample (when `n.boots` > `0`);
- `ranef.boot.valid`: a list where each element is a matrix that contains the predicted random effects on the original data, based on the lmm fitted on the cluster bootstrap samples (when `n.boots` > `0`);

Author(s)

Mirko Signorelli

References

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

See Also

`fit_lmms` (step 1), `fit_prclmm` (step 3), `performance_prclmm`

Examples

```
# generate example data
set.seed(1234)
p = 4 # number of longitudinal predictors
simdata = simulate_prclmm_data(n = 100, p = p, p.relev = 2,
             seed = 123, t.values = c(0, 0.2, 0.5, 1, 1.5, 2))

# specify options for cluster bootstrap optimism correction
# procedure and for parallel computing
do.bootstrap = FALSE
# IMPORTANT: set do.bootstrap = TRUE to compute the optimism correction!
n.boots = ifelse(do.bootstrap, 100, 0)
parallelize = FALSE
# IMPORTANT: set parallelize = TRUE to speed computations up!
if (!parallelize) n.cores = 1
if (parallelize) {
   # identify number of available cores on your machine
   n.cores = parallel::detectCores()
   if (is.na(n.cores)) n.cores = 1
}
```

```
# step 1 of PRC-LMM: estimate the LMMs
y.names = paste('marker', 1:p, sep = '')
step1 = fit_lmms(y.names = y.names,
                 fixefs = ~ age, ranefs = ~ age | id,
                 long.data = simdata$long.data,
                 surv.data = simdata$surv.data,
                 t.from.base = t.from.base,
                 n.boots = n.boots, n.cores = n.cores)

# step 2 of PRC-LMM: compute the summaries
# of the longitudinal outcomes
step2 = summarize_lmms(object = step1, n.cores = n.cores)
```

`survpred_prclmm` *Compute predictive survival probabilities from PRC-LMM*

Description

This function computes the predictive survival probabilities for the PRC-LMM model proposed in Signorelli et al. (2020, in review)

Usage

```
survpred_prclmm(step2, step3, times = 1)
```

Arguments

`step2`	the output of `summarize_lmms` (step 2 of the estimation of the PRC-LMM model)
`step3`	the output of `fit_prclmm` (step 3 of the estimation of the PRC-LMM model)
`times`	numeric vector with the time points at which to estimate the time-dependent AUC

Value

A data frame with the predicted survival probabilities computed at the supplied time points

Author(s)

Mirko Signorelli

References

Signorelli, M., Spitali, P., Al-Khalili Szigyarto, C, The MARK-MD Consortium, Tsonaka, R. (2021). Penalized regression calibration: a method for the prediction of survival outcomes using complex longitudinal and high-dimensional data. arXiv preprint: arXiv:2101.04426.

See Also

`fit_lmms` (step 1), `summarize_lmms` (step 2), `fit_prclmm` (step 3)

Examples

```
# generate example data
set.seed(1234)
p = 4 # number of longitudinal predictors
simdata = simulate_prclmm_data(n = 100, p = p, p.relev = 2,
             seed = 123, t.values = c(0, 0.2, 0.5, 1, 1.5, 2))

# step 1 of PRC-LMM: estimate the LMMs
y.names = paste('marker', 1:p, sep = '')
step1 = fit_lmms(y.names = y.names,
                 fixefs = ~ age, ranefs = ~ age | id,
                 long.data = simdata$long.data,
                 surv.data = simdata$surv.data,
                 t.from.base = t.from.base,
                 n.boots = 0)

# step 2 of PRC-LMM: compute the summaries
# of the longitudinal outcomes
step2 = summarize_lmms(object = step1)

# step 3 of PRC-LMM: fit the penalized Cox models
step3 = fit_prclmm(object = step2, surv.data = simdata$surv.data,
                   baseline.covs = ~ baseline.age,
                   penalty = 'ridge')

# predict survival probabilities at times 1, 2, 3
surv.probs = survpred_prclmm(step2, step3, times = 1:3)
head(surv.probs)
```

Index