Package: svycoxme (via r-universe)

May 19, 2026

Title Mixed-Effects Cox Models for Complex Samples

Version 1.0.0.9000

Description Mixed-effect proportional hazards models for multistage stratified, cluster-sampled, unequally weighted survey samples. Provides variance estimation by Taylor series linearisation or replicate weights.

Depends R (>= 4.1.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports survey, coxme, survival, Rcpp, lme4, Matrix, future, parallelly

Roxygen list(markdown = TRUE)

LinkingTo Rcpp

Suggests knitr, rmarkdown, future.apply, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL `https://github.com/bdrayton/svycoxme`

BugReports `https://github.com/bdrayton/svycoxme/issues`

Config/pak/sysreqs cmake make

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

Date/Publication 2025-08-22 10:25:01 UTC

RemoteUrl https://github.com/bdrayton/svycoxme

RemoteRef HEAD

RemoteSha 097450de4dd6102d07293745f368c74bda438bce

Contents

---

pop *Synthetic population data*

Description

This is made-up time-to-event data with properties that make it useful for testing and demonstrating `svycoxme` functions. There is a single level of clustering, identified with group_id, and the X covariates depend on Z covariates.

Usage

```
pop
```

Format

A data frame with 20,000 rows and 10 columns:

X1 Observation-level $N(\mu_{X1}, 1)$ distributed covariate where $\mu_{X1} = 0.5 * (Z1 + 1)$

X2 Cluster-level $N(\mu_{X2}, 1)$ distributed covariate where $\mu_{X2} = 0.5 * (Z2 + Z3)$

X3 Cluster-level binary covariate where $Pr(X3 = 1) = Z3$

Z1 Stratum membership. Takes the values 1 to 5

Z2 cluster-level N(0,1) distributed covariate

Z3 cluster-level Uniform(0,1) distributed covariate

stat_time Event or Censoring time

stat Event/Censoring indicator. Event=1; Censoring=0

group_id Unique cluster ID

obs_id Unique observation ID

sampled Sampling indicator. Is this observation in samp_srcs?

`residuals.coxme` *Calculate residuals for a 'coxme' fit*

Description

Calculates score, dfbeta, or dfbetas residuals for a mixed-effects proportional hazards model. Only fixed-effect components are calculated; see Details.

Usage

```
## S3 method for class 'coxme'
residuals(
  object,
  data,
  type = c("score", "dfbeta", "dfbetas"),
  weighted = (type %in% c("dfbeta", "dfbetas")),
  include_re = FALSE,
  ...
)
```

Arguments

`object`	an object inheriting from class `coxme`. This includes the output from `coxme` and `svycoxme` functions.
`data`	the data used to generate `object`.
`type`	character string indicating the type of residual desired. Possible values are "score", "dfbeta"', "dfbetas".
`weighted`	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
`include_re`	logical flag indicating if residuals for random effects should be returned. This flag is currently ignored; see Details.
`...`	other unused arguments.

Details

An observation's contribution to the score vector includes values for every fixed and random effect in the fitted model. In many cases, the number of random effects will be large, and most residuals will be zero. Until efficient sparse computation is implemented, it is too expensive computationally and on memory to calculate the random effect residual terms, so they are excluded. This is likely to change, and the parameter `include_re` is include for future expansion.

Value

A matrix of residuals. The score residuals are each observation's contribution to the score vector. Two transformations of this are often more useful: dfbeta is the approximate change in the coefficient vector if that observation were dropped, and dfbetas is the approximate change in the coefficients, scaled by the standard error for the coefficients.

Examples

```
fit1 <- coxme::coxme(survival::Surv(stat_time, stat) ~ X1 + X2 + X3 + (1 | group_id),
                     data = samp_srcs)
dfbeta_res <- resid(fit1, data = samp_srcs, type = "dfbeta")

head(dfbeta_res)
```

`samp_srcs` *Simple random cluster sample of 100 clusters from synthetic population data, pop.*

Description

This is made-up time-to-event data with properties that make it useful for testing and demonstrating `svycoxme` functions. There is a single level of clustering, identified with group_id, and the X covariates depend on Z covariates.

Usage

```
samp_srcs
```

Format

A data frame with 20,000 rows and 10 columns:

X1 Observation-level $N(\mu_{X1}, 1)$ distributed covariate where $\mu_{X1} = 0.5 * (Z1 + 1)$.

X2 Cluster-level $N(\mu_{X2}, 1)$ distributed covariate where $\mu_{X2} = 0.5 * (Z2 + Z3)$.

X3 Cluster-level binary covariate where $Pr(X3 = 1) = Z3$.

Z1 Stratum membership. Takes the values 1 to 5.

Z2 cluster-level N(0,1) distributed covariate.

Z3 cluster-level Uniform(0,1) distributed covariate.

stat_time Event or Censoring time.

stat Event/Censoring indicator. Event=1; Censoring=0.

group_id Unique cluster ID.

obs_id Unique observation ID.

fpc Total number of clusters in the population.

weight Observation-level inverse probability of selection weight.

`svycoxme` *Survey-weighted mixed-effects Cox models*

Description

Fit a mixed-effect proportional hazards model to data from a complex design.

Usage

```
svycoxme(
  formula,
  design,
  subset = NULL,
  rescale = TRUE,
  control = coxme::coxme.control(),
  ...
)

## S3 method for class 'DBIsvydesign'
svycoxme(
  formula,
  design,
  subset = NULL,
  rescale = TRUE,
  control = coxme::coxme.control(),
  ...
)

## S3 method for class 'survey.design'
svycoxme(
  formula,
  design,
  subset = NULL,
  rescale = TRUE,
  control = coxme::coxme.control(),
  ...
)

## S3 method for class 'svyrep.design'
svycoxme(
  formula,
  design,
  subset = NULL,
  rescale = TRUE,
  control = coxme::coxme.control(),
  multicore = getOption("survey.multicore"),
  return.replicates = FALSE,
```

```
  ...
)
```

Arguments

`formula`	Model formula.
`design`	`survey.design` object. It must contain all variables in the formula.
`subset`	Expression to select a subpopulation.
`rescale`	Rescale weights to improve numerical stability.
`control`	Optional list of `coxme` control options. See `coxme.control` for details.
`...`	Other arguments passed to `coxme`.
`multicore`	For replicate weight designs. Should parallel processing be used?
`return.replicates`	For replicate weight designs. Should replicates be returned?

Details

Parallel processing is done with `future_lapply`. Future planning is left to the user, e.g. using `plan` before the call to `svycoxme`. Note that `svycoxme.DBIsvydesign` has not been implemented yet.

Value

An object of class `svycoxme`.

Examples

```
des <- survey::svydesign(ids = ~group_id, weights = ~weight, data = samp_srcs)
fit1 <- svycoxme(survival::Surv(stat_time, stat) ~ X1 + X2 + X3 + (1 | group_id),
                 design = des)
summary(fit1)

# with replicate weights (only 10 replicates are used to reduce CPU time)
repdes <- survey::as.svrepdesign(des, type = "bootstrap", replicates = 10)
fit2 <- svycoxme(survival::Surv(stat_time, stat) ~ X1 + X2 + X3 + (1 | group_id),
                 design = repdes)
summary(fit2)

# use multicore processing (`n_cores = 2` to comply with CRAN policy). Otherwise,
# something like, `floor(parallelly::availableCores() * 0.8)`, could be used.

n_cores = 2
future::plan("multicore", workers = n_cores)
fit3 <- svycoxme(survival::Surv(stat_time, stat) ~ X1 + X2 + X3 + (1 | group_id),
                 design = repdes, multicore = TRUE)
all.equal(coef(fit2), coef(fit3))
future::plan("sequential")
```

Index