Package ‘sleev’

February 1, 2022

Type Package

Title Semiparametric Likelihood Estimation with Errors in Variables

Version 1.0.1

Description
Efficient regression analysis under general two-phase sampling, where Phase I includes error-prone data and Phase II contains validated data on a subset.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.1.2

Depends Rcpp (>= 1.0.7), R (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Suggests knitr, lme4, MASS, rmarkdown, splines, testthat

NeedsCompilation yes

Author Sarah Lotspeich [aut],
Ran Tao [aut, cre],
Joey Sherrill [prg]

Maintainer Ran Tao <r.tao@vanderbilt.edu>

Repository CRAN

Date/Publication 2022-02-01 07:50:07 UTC

R topics documented:

cv_linear2ph ................................................................. 2
linear2ph ................................................................. 4
logistic2ph ............................................................... 7

Index 11
cv_linear2ph

Performs cross-validation to calculate the average predicted log likelihood for the linear2ph function. This function can be used to select the B-spline basis that yields the largest average predicted log likelihood.

Description

Performs cross-validation to calculate the average predicted log likelihood for the linear2ph function. This function can be used to select the B-spline basis that yields the largest average predicted log likelihood.

Usage

cv_linear2ph(
    Y_unval = NULL,
    Y = NULL,
    X_unval = NULL,
    X = NULL,
    Z = NULL,
    Bspline = NULL,
    data = NULL,
    nfolds = 5,
    MAX_ITER = 2000,
    TOL = 1e-04,
    verbose = FALSE
)

Arguments

Y_unval Specifies the column of the error-prone outcome that is continuous. Subjects with missing values of Y_unval are omitted from the analysis. This argument is required.

Y Specifies the column that stores the validated value of Y_unval in the second phase. Subjects with missing values of Y are considered as those not selected in the second phase. This argument is required.

X_unval Specifies the columns of the error-prone covariates. Subjects with missing values of X_unval are omitted from the analysis. This argument is required.

X Specifies the columns that store the validated values of X_unval in the second phase. Subjects with missing values of X are considered as those not selected in the second phase. This argument is required.

Z Specifies the columns of the accurately measured covariates. Subjects with missing values of Z are omitted from the analysis. This argument is optional.

Bspline Specifies the columns of the B-spline basis. Subjects with missing values of Bspline are omitted from the analysis. This argument is required.
cv_linear2ph

data Specifies the name of the dataset. This argument is required.
nfolds Specifies the number of cross-validation folds. The default value is 5. Although nfolds can be as large as the sample size (leave-one-out cross-validation), it is not recommended for large datasets. The smallest value allowable is 3.
MAX_ITER Specifies the maximum number of iterations in the EM algorithm. The default number is 2000. This argument is optional.
TOL Specifies the convergence criterion in the EM algorithm. The default value is 1E-4. This argument is optional.
verbose If TRUE, then show details of the analysis. The default value is FALSE.

Value
avg_pred_loglike Stores the average predicted log likelihood.
pred_loglike Stores the predicted log likelihood in each fold.
converge Stores the convergence status of the EM algorithm in each run.

Examples
rho = 0.3
p = 0.3
n = 100
n2 = 40
alpha = 0.3
beta = 0.4

### generate data
simX = rnorm(n)
epsilon = rnorm(n)
simY = alpha+beta*simX+epsilon
error = MASS::mvrnorm(n, mu=c(0,0), Sigma=matrix(c(1, rho, rho, 1), nrow=2))

simS = rbinom(n, 1, p)
simu = simS*error[,2]
simW = simS*error[,1]
simY_tilde = simY+simW
simX_tilde = simX+simU

id_phase2 = sample(n, n2)
simY[-id_phase2] = NA
simX[-id_phase2] = NA

# cubic basis
nsieves = c(5, 10)
pred_loglike = rep(NA, length(nsieves))
for (i in 1:length(nsieves)) {
    nsieve = nsieves[i]
    Bsplit = splines::bs(simX_tilde, df=nsieve, degree=3,
                         Boundary.knots=range(simX_tilde), intercept=TRUE)
colnames(Bspline) = paste("bs", 1:nsieve, sep="")
# cubic basis

data = data.frame(Y_tilde=simY_tilde, X_tilde=simX_tilde, Y=simY, X=simX, Bspline)
### generate data

res = cv_linear2ph(Y="Y", X="X", Y_unval="Y_tilde", X_unval="X_tilde",
                   Bspline=colnames(Bspline), data=data, nfolds = 5)
pred_loglike[i] = res$avg_pred_loglik
}
data.frame(nsieves, pred_loglike)

linear2ph

Sieve maximum likelihood estimator (SMLE) for two-phase linear regression problems

Description

Performs efficient semiparametric estimation for general two-phase measurement error models when there are errors in both the outcome and covariates.

Usage

linear2ph(
  Y_unval = NULL,
  Y = NULL,
  X_unval = NULL,
  X = NULL,
  Z = NULL,
  Bspline = NULL,
  data = NULL,
  hn_scale = 1,
  noSE = FALSE,
  TOL = 1e-04,
  MAX_ITER = 1000,
  verbose = FALSE
)

Arguments

Y_unval Column name of the error-prone or unvalidated continuous outcome. Subjects with missing values of Y_unval are omitted from the analysis. This argument is required.

Y Column name that stores the validated value of Y_unval in the second phase. Subjects with missing values of Y are considered as those not selected in the second phase. This argument is required.
Specify the columns of the error-prone covariates. Subjects with missing values of X_unval are omitted from the analysis. This argument is required.

X Specifies the columns that store the validated values of X_unval in the second phase. Subjects with missing values of X are considered as those not selected in the second phase. This argument is required.

Z Specifies the columns of the accurately measured covariates. Subjects with missing values of Z are omitted from the analysis. This argument is optional.

Bspline Specifies the columns of the B-spline basis. Subjects with missing values of Bspline are omitted from the analysis. This argument is required.

data Specifies the name of the dataset. This argument is required.

hn_scale Specifies the scale of the perturbation constant in the variance estimation. For example, if hn_scale = 0.5, then the perturbation constant is $0.5n^{-1/2}$, where $n$ is the first-phase sample size. The default value is 1. This argument is optional.

noSE If TRUE, then the variances of the parameter estimators will not be estimated. The default value is FALSE. This argument is optional.

TOL Specifies the convergence criterion in the EM algorithm. The default value is 1E-4. This argument is optional.

MAX_ITER Maximum number of iterations in the EM algorithm. The default number is 1000. This argument is optional.

verbose If TRUE, then show details of the analysis. The default value is FALSE.

Value

coefficients Stores the analysis results.

sigma Stores the residual standard error.

covariance Stores the covariance matrix of the regression coefficient estimates.

converge In parameter estimation, if the EM algorithm converges, then converge = TRUE. Otherwise, converge = FALSE.

converge_cov In variance estimation, if the EM algorithm converges, then converge_cov = TRUE. Otherwise, converge_cov = FALSE.

References


See Also

cv_linear2ph() to calculate the average predicted log likelihood of this function.
Examples

\[
\begin{align*}
\rho &= -0.3 \\
p &= 0.3 \\
\text{hn\_scale} &= 1 \\
\text{nsieve} &= 20 \\
n &= 100 \\
n2 &= 40 \\
\alpha &= 0.3 \\
\beta &= 0.4 \\
\text{set.seed}(12345) \\
\text{### generate data} \\
\text{simX} &= \text{rnorm}(n) \\
\epsilon &= \text{rnorm}(n) \\
\text{simY} &= \alpha + \beta \cdot \text{simX} + \epsilon \\
\text{error} &= \text{MASS::mvrnorm}(n, \text{mu}=c(0,0), \text{Sigma}=\text{matrix}(c(1, \rho, \rho, 1), \text{nrow}=2)) \\
\text{simS} &= \text{rbinom}(n, 1, p) \\
\text{simU} &= \text{simS} \cdot \text{error}[,2] \\
\text{simW} &= \text{simS} \cdot \text{error}[,1] \\
\text{simY\_tilde} &= \text{simY} + \text{simW} \\
\text{simX\_tilde} &= \text{simX} + \text{simU} \\
\text{id\_phase2} &= \text{sample}(n, n2) \\
\text{simY}[\text{-id\_phase2}] &= \text{NA} \\
\text{simX}[\text{-id\_phase2}] &= \text{NA} \\
\text{### histogram basis} \\
\text{Bspline} &= \text{matrix}(\text{NA}, \text{nrow}=n, \text{ncol}=\text{nsieve}) \\
\text{cut\_x\_tilde} &= \text{cut}(\text{simX\_tilde}, \text{breaks}=\text{quantile(\text{simX\_tilde}, \text{probs}=\text{seq}(0, 1, 1/\text{nsieve})), \text{include.lowest} = \text{TRUE}}) \\
\text{for} \; (i \; \text{in} \; 1:\text{nsieve}) \{ \\
\text{Bspline}[,i] &= \text{as.numeric(\text{cut\_x\_tilde} == \text{names(table(\text{cut\_x\_tilde}))}[i])} \\
\text{\}} \\
\text{colnames(Bspline)} &= \text{paste("bs", 1:nsieve, sep="")} \\
\text{### linear basis} \\
\text{Bspline} &= \text{splines::bs(\text{simX\_tilde}, \text{df}=\text{nsieve}, \text{degree}=1, \text{Boundary.knots}=\text{range(\text{simX\_tilde})}, \text{intercept}=\text{TRUE})} \\
\text{colnames(Bspline)} &= \text{paste("bs", 1:nsieve, sep="")} \\
\text{### quadratic basis} \\
\text{Bspline} &= \text{splines::bs(\text{simX\_tilde}, \text{df}=\text{nsieve}, \text{degree}=2, \text{Boundary.knots}=\text{range(\text{simX\_tilde})}, \text{intercept}=\text{TRUE})} \\
\text{colnames(Bspline)} &= \text{paste("bs", 1:nsieve, sep="")} \\
\text{### cubic basis} \\
\end{align*}
\]
logistic2ph

Bspline = splines::bs(simX_tilde, df=nsieve, degree=3, Boundary.knots=range(simX_tilde), intercept=TRUE)
colnames(Bspline) = paste("bs", 1:nsieve, sep="")
# cubic basis
data = data.frame(Y_tilde=simY_tilde, X_tilde=simX_tilde, Y=simY, X=simX, Bspline)
res = linear2ph(Y="Y", X="X", Y_unval="Y_tilde", X_unval="X_tilde",
               Bspline=colnames(Bspline), data=data, hn_scale=0.1)

logistic2ph

Sieve maximum likelihood estimator (SMLE) for two-phase logistic regression problems

Description

This function returns the sieve maximum likelihood estimators (SMLE) for the logistic regression model from Lotspeich et al. (2021).

Usage

logistic2ph(
  Y_unval = NULL,
  Y = NULL,
  X_unval = NULL,
  X = NULL,
  Z = NULL,
  Bspline = NULL,
  data = NULL,
  theta_pred = NULL,
  gamma_pred = NULL,
  initial_lr_params = "Zeros",
  hn_scale = 1,
  noSE = FALSE,
  TOL = 1e-04,
  MAX_ITER = 1000,
  verbose = FALSE
)

Arguments

Y_unval
Column name of the error-prone or unvalidated continuous outcome. Subjects with missing values of Y_unval are omitted from the analysis. If Y_unval is null, the outcome is assumed to be error-free.

Y
Column name that stores the validated value of Y_unval in the second phase. Subjects with missing values of Y are considered as those not selected in the second phase. This argument is required.
X_unval
Column name(s) with the unvalidated predictors. If X_unval and X are null, all predictors are assumed to be error-free.

X
Column name(s) with the validated predictors. If X_unval and X are NULL, all predictors are assumed to be error-free.

Z
(Optional) Column name(s) with additional error-free covariates.

Bspline
Vector of column names containing the B-spline basis functions.

data
A dataframe with one row per subject containing columns: Y_unval, Y, X_unval, X, Z, and Bspline.

theta_pred
Vector of columns in data that pertain to the predictors in the analysis model.

gamma_pred
Vector of columns in data that pertain to the predictors in the outcome error model.

initial_lr_params
Initial values for parametric model parameters. Choices include (1) "Zeros" (non-informative starting values) or (2) "Complete-data" (estimated based on validated subjects only)

hn_scale
Size of the perturbation used in estimating the standard errors via profile likelihood. If none is supplied, default is \(hn\_scale = 1\).

noSE
Indicator for whether standard errors are desired. Defaults to noSE = FALSE.

TOL
Tolerance between iterations in the EM algorithm used to define convergence.

MAX_ITER
Maximum number of iterations in the EM algorithm. The default number is 1000. This argument is optional.

verbose
If TRUE, then show details of the analysis. The default value is FALSE.

Value

coeff
dataframe with final coefficient and standard error estimates (where applicable) for the analysis model.

outcome_err_coeff
dataframe with final coefficient estimates for the outcome error model.

Bspline_coeff
dataframe with final B-spline coefficient estimates (where applicable).

vcov
variance-covariance matrix for coeff (where applicable).

converged
indicator of EM algorithm convergence for parameter estimates.

se_converged
indicator of standard error estimate convergence.

converged_msg
(where applicable) description of non-convergence.

iterations
number of iterations completed by EM algorithm to find parameter estimates.

od_loglik_at_conv
value of the observed-data log-likelihood at convergence.

References

Examples

set.seed(918)

# Set sample sizes ---------------------------------------------
N <- 1000 # Phase-I = N
n <- 250 # Phase-II/audit size = n

# Generate true values Y, Xb, Xa ------------------------------
Xa <- rbinom(n = N, size = 1, prob = 0.25)
Xb <- rbinom(n = N, size = 1, prob = 0.5)
Y <- rbinom(n = N, size = 1, prob = (1 + exp(-(-0.65 - 0.2 * Xb - 0.1 * Xa))) ^ (-1))

# Generate error-prone Xb* from error model P(Xb*|Xb,Xa) ----
sensX <- specX <- 0.75
delta0 <- - log(specX / (1 - specX))
delta1 <- - delta0 - log((1 - sensX) / sensX)
Xbstar <- rbinom(n = N, size = 1,
                  prob = (1 + exp(- (delta0 + delta1 * Xb + 0.5 * Xa))) ^ (-1))

# Generate error-prone Y* from error model P(Y*|Xb*,Y,Xb,Xa)
sensY <- 0.95
specY <- 0.90
theta0 <- - log(specY / (1 - specY))
theta1 <- - theta0 - log((1 - sensY) / sensY)
Ystar <- rbinom(n = N, size = 1,
                prob = (1 + exp(- (theta0 - 0.2 * Xbstar + theta1 * Y - 0.2 * Xb - 0.1 * Xa))) ^ (-1))

## V is a TRUE/FALSE vector where TRUE = validated --------
V <- seq(1, N) %in% sample(x = seq(1, N), size = n, replace = FALSE)

# Build dataset -----------------------------------------------
sdat <- cbind(Y, Xb, Ystar, Xbstar, Xa)
# Make Phase-II variables Y, Xb NA for unaudited subjects ---
sdat[, c("Y", "Xb")][!V] <- NA

# Fit models -----------------------------------------------
## Naive model -----------------------------------------
naive <- glm(Ystar ~ Xbstar + Xa, family = "binomial", data = data.frame(sdat))

## Generalized raking --------------------------------------
### Taken from: https://github.com/T0ngChen/multiwave/blob/master/sim.r
inf.fun <- function(fit) {
  dm <- model.matrix(fit)
  Ihat <- (t(dm) %*% (dm * fit$fitted.values * (1 - fit$fitted.values))) / nrow(dm)
  ## influence function
  infl <- (dm * resid(fit, type = "response")) %*% solve(Ihat)
  infl
}
naive_infl <- inf.fun(naive) # error-prone influence functions based on naive model
colnames(naive_infl) <- paste0("if", 1:3)

# Add naive influence functions to sdat -----------------------------------------------
sdat <- cbind(id = 1:N, sdat, naive_infl)

### Construct B-spline basis -------------------------------
### Since Xb* and Xa are both binary, reduces to indicators --
nsieve <- 4
B <- matrix(0, nrow = N, ncol = nsieve)
B[which(Xa == 0 & Xbstar == 0), 1] <- 1
B[which(Xa == 0 & Xbstar == 1), 2] <- 1
B[which(Xa == 1 & Xbstar == 0), 3] <- 1
B[which(Xa == 1 & Xbstar == 1), 4] <- 1
colnames(B) <- paste0("bs", seq(1, nsieve))
sdat <- cbind(sdat, B)
smle <- logistic2ph(Y_unval = "Ystar",
  Y = "Y",
  X_unval = "Xbstar",
  X = "Xb",
  Z = "Xa",
  Bspline = colnames(B),
  data = sdat,
  noSE = FALSE,
  MAX_ITER = 1000,
  TOL = 1E-4)
Index

cv_linear2ph, 2
cv_linear2ph(), 5

linear2ph, 4
logistic2ph, 7