Package ‘MLBC’

May 19, 2025

Version 0.1.0

Title Bias Correction Methods for Models Using Synthetic Data

Description Implements three bias-correction techniques (additive bias correction, multiplicative bias correction, and one-step estimation via Template Model Builder (TMB)) based on Battaglia et al. (2025 <doi:10.48550/arXiv.2402.15585>) to improve inference using synthetic data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports TMB

LinkingTo TMB, RcppEigen

Suggests roxygen2

NeedsCompilation yes

Author Konrad Kurczynski [aut, cre],
Timothy Christensen [aut]

Maintainer Konrad Kurczynski <konrad.kurczynski@yale.edu>

Repository CRAN

Date/Publication 2025-05-19 09:00:02 UTC

Contents

ols — *Ordinary least squares (and heteroskedastic-robust SEs)*

Description

Ordinary least squares (and heteroskedastic-robust SEs)

Usage

```
ols(Y, X, se = TRUE)
```

Arguments

`Y`	numeric response
`X`	numeric design matrix
`se`	logical; return SEs?

Value

list(coef, vcov, sXX) or list(coef, sXX)

ols_bca — *Additive bias-corrected OLS estimator*

Description

Computes the additive bias correction (BCA) for an OLS regression when the primary regressor is measured by an ML/AI method.

Usage

```
ols_bca(Y, Xhat, fpr, m, intercept = TRUE)
```

Arguments

`Y`	Numeric vector of responses.
`Xhat`	Numeric matrix of regressors excluding the intercept. The first column **must** be the ML-generated variable to correct.
`fpr`	Numeric. Estimated false-positive rate of the generated regressor.
`m`	Integer. Size of the validation (labeled) sample used to estimate `fpr`.
`intercept`	Logical; if `TRUE`, an intercept column of 1's is prepended.

Value

An object of class `mlbc_fit` and subclass `mlbc_bca`, a list with elements

- `coef`: Numeric vector of bias-corrected coefficients (intercept first, if requested).
- `vcov`: Variance–covariance matrix of those coefficients.

References

Battaglia, Christensen, Hansen, and Sacher (2025). "Inference for Regression with Variables Generated by AI or Machine Learning".

See Also

ols_bcm for the multiplicative correction.

Examples

```
# unlabeled:
Nunl      <- 1e4
Xtrue_unl <- rbinom(Nunl, 1, 0.2)
Xhat_unl  <- ifelse(runif(Nunl) < 0.1, 1, Xtrue_unl)
Y_unl     <- 5 + 2 * Xtrue_unl + rnorm(Nunl)

# small labeled sample to get fpr:
nval      <- 100
Xtrue_val <- rbinom(nval, 1, 0.2)
Xhat_val  <- ifelse(runif(nval) < 0.1, 1, Xtrue_val)
Y_val     <- 5 + 2 * Xtrue_val + rnorm(nval)
fpr_hat   <- mean(Xhat_val == 1 & Xtrue_val == 0)

# now do additive correction, with intercept
fit_bca <- ols_bca(
  Y        = Y_unl,
  Xhat     = matrix(Xhat_unl, ncol = 1, dimnames = list(NULL, "Xhat")),
  fpr      = fpr_hat,
  m        = nval,
  intercept= TRUE
)
print(fit_bca)
```

`ols_bcm` *Multiplicative bias-corrected OLS estimator*

Description

Computes the multiplicative bias correction (BCM) for an OLS regression when the primary regressor is measured by an ML/AI method.

Usage

```
ols_bcm(Y, Xhat, fpr, m, intercept = TRUE)
```

Arguments

Y	Numeric vector (or one-column matrix) of responses.
Xhat	Numeric matrix of regressors; **the first column** must be the ML-generated regressor whose bias we're correcting, and remaining columns are any additional "true" controls.
fpr	Numeric scalar. Estimated false-positive rate of the generated regressor (proportion of ML positives that are actually negatives).
m	Integer. Size of the **validation/labeled** subsample used to estimate `fpr` — i.e.\ the number of observations where you observe both the ML prediction (`Xhat`) and the true regressor.
intercept	logical, `TRUE` by default.

Value

An object of class `mlbc_fit` (and subclass `mlbc_bcm`) with two components:

- `coef`: Numeric vector of bias-corrected regression coefficients.
- `vcov`: Variance-covariance matrix for those coefficients.

References

Battaglia, Christensen, Hansen, and Sacher (2025). "Inference for Regression with Variables Generated by AI or Machine Learning".

See Also

ols_bca for the additive correction.

Examples

```
# generate data
Nunl     <- 10000
Xtrue_unl<- rbinom(Nunl, 1, 0.2)
Xhat_unl <- ifelse(runif(Nunl) < 0.1, 1, Xtrue_unl)
Y_unl     <- 5 + 2*Xtrue_unl + rnorm(Nunl)
#estimate the false-positive rate
nval     <- 100
Xtrue_val<- rbinom(nval, 1, 0.2)
Xhat_val <- ifelse(runif(nval) < 0.1, 1, Xtrue_val)
Y_val     <- 5 + 2*Xtrue_val + rnorm(nval)
fpr_hat  <- mean(Xhat_val==1 & Xtrue_val==0)
fit_bcm <- ols_bcm(Y_unl,
                  Xhat = matrix(Xhat_unl, ncol=1),
                  fpr = fpr_hat,
                  m   = nval,
```

```
                    intercept = TRUE)
summary(fit_bcm)
```

one_step *One-step estimator for unlabeled data (multi-dist)*

Description

Fits the one-step estimator by maximizing the unlabeled likelihood via TMB, automatically differentiating the objective, gradient, and Hessian.

Usage

```
one_step(
  Y,
  Xhat,
  homoskedastic = FALSE,
  distribution = c("normal", "t", "laplace", "gamma", "beta"),
  nu = 4,
  gshape = 2,
  gscale = 1,
  ba = 2,
  bb = 2,
  intercept = TRUE
)
```

Arguments

Y	Numeric response vector.
Xhat	Numeric matrix of regressors *excluding* the intercept. The **first** column must be the ML-generated regressor to correct.
homoskedastic	Logical; if `TRUE`, assume a single error variance.
distribution	Character: one of `"normal"`, `"t"`, `"laplace"`, `"gamma"`, or `"beta"`. Specifies which conditional density to use for residuals in the likelihood estimation.
nu	Numeric; degrees of freedom (only used if `distribution = "t"`).
gshape, gscale	Numeric; shape & scale for Gamma (only if `distribution = "gamma"`).
ba, bb	Numeric; alpha & beta for Beta (only if `distribution = "beta"`).
intercept	Logical; if `TRUE`, an intercept column of 1's is prepended.

Value

An object of class `mlbc_fit` and subclass `mlbc_onestep` with:

- `coef`: Named numeric vector of estimated coefficients.
- `cov` : Variance–covariance matrix.

References

Battaglia, Christensen, Hansen, and Sacher (2025). "Inference for Regression with Variables Generated by AI or Machine Learning".

Examples

```
set.seed(2025)

# 1) Simulate “unlabeled” data
n     <- 200
p     <- 0.3
Xtrue <- rbinom(n, 1, p)
# ML regressor with 10% false positives
Xhat  <- ifelse(runif(n) < 0.10, 1 - Xtrue, Xtrue)
Y     <- 1 + 2 * Xtrue + rnorm(n)

# 2) Small validation set to estimate fpr
m        <- 50
Xval_t   <- rbinom(m, 1, p)
Xval_h   <- ifelse(runif(m) < 0.10, 1 - Xval_t, Xval_t)
fpr_hat  <- mean(Xval_h == 1 & Xval_t == 0)

# 3) One-step TMB estimator (Normal), with intercept
fit <- one_step(
  Y            = Y,
  Xhat         = matrix(Xhat, ncol = 1, dimnames = list(NULL, "Xhat")),
  homoskedastic = FALSE,
  distribution  = "normal",
  intercept     = TRUE
)
print(fit)
```

Index