Package: ropper (via r-universe)

July 22, 2026

Type Package

Title Ranking-Optimized Population Posterior Expected Percentiles

Version 0.2

Date 2026-07-11

Maintainer Nicholas Henderson <nchender@umich.edu>

Imports stats, utils

Description Implements the empirical Bayes ranking method described in Henderson and Hartman (2025) <doi:10.48550/arXiv.2511.16530>. The data structure of interest is a collection of cluster-specific effect size estimates, associated standard errors for these estimates, and cluster-level covariates. Estimates of the rankings of the cluster-specific effect sizes after adjusting for cluster-level covariates are generated.

License GPL-2

Encoding UTF-8

Depends R (>= 3.5)

LazyData true

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

Date/Publication 2026-07-11 05:22:18 UTC

RemoteUrl https://github.com/nchenderson/ropper

RemoteRef HEAD

RemoteSha bd51e7e2bfff2ddb2917c88e6405cdde185b2863

Contents

BLUP *Posterior expected means*

Description

Computes estimates of cluster-level posterior expected means given an estimate of the regression coefficients and an estimate of the random effects variance. For a fixed value of the random effects variance, the collection of posterior expected mean estimates is commonly referred to as the "best linear unbiased predictor", or BLUP.

Usage

```
BLUP(beta.hat, Y, X, ses, tau.sq)
```

Arguments

`beta.hat`	The 1 x p vector of regression coefficient estimates.
`Y`	The K x 1 vector of cluster-specific effect estimates.
`X`	The K x p design matrix containing cluster-specific covariates.
`ses`	The K x 1 vector sampling standard errors squared.
`tau.sq`	Estimate of the random effects variance. This must be a scalar greater than 0.

Details

The BLUP for cluster k is defined as $E(v_k|Y_k) = B_k(Y_k - X_k^T\hat{\beta})$, where $\hat{\beta}$ is the vector of regression coefficient estimates and B_k is defined as $B_k = \tau^2/(\tau^2 + ses_k)$

Value

A K x 1 vector of posterior mean estimates.

Author(s)

Nicholas C. Henderson and Nicholas Hartman

See Also

See also PEPP.

Examples

```
#########################################################
###  Example of using PEPP with simulated data  #####
#########################################################
n <- 50
tau.sq <- 0.5
cluster.sizes <- 1 + rpois(n, lambda=5)

xx <- rnorm(n)
theta.vec <- rnorm(n, sd=sqrt(tau.sq))
## True fixed-effect function is pnorm(x)
mu.vec <- pnorm(xx) + theta.vec
yy <- mu.vec + rnorm(n)/sqrt(cluster.sizes)

## Fit model where it is assumed that fixed-effect
## function is beta0 + beta1*x
XX <- cbind(rep(1, n), xx)
## define standard errors squared
stderrs_sq <- 1/cluster.sizes


## Estimate regression coefficients by maximum likelihood
## estimation assuming that tau.sq = 0.5
beta.mle <- solve(crossprod(XX, (stderrs_sq + 0.5)*XX),
                  crossprod(XX, (stderrs_sq + 0.5)*yy))
## Extract the posterior means associated with the maximum
## likelihood estimate:
blup.mle <- BLUP(beta.mle, Y=yy, X=XX, ses=stderrs_sq,
                 tau.sq=0.5)

## Estimate regression coefficients using ROPPER assuming
## that tau.sq = 0.5
rfit <- ropper(Y=yy, X=XX, ses=stderrs_sq, tau.sq=0.5)

## Extract posterior means associated with the ROPPER estimate
blup.rop <- BLUP(rfit$coefficients, Y=yy, X=XX,
                 ses=stderrs_sq, tau.sq=0.5)

plot(blup.mle, blup.rop,
     xlab="MLE Posterior Means",
     ylab="ROPPER Posterior Means")
```

KnnTausqEst *Nearest neighbors estimation of random effects variance*

Description

Computes an estimate of the random effects variance using a nonparametric, k nearest neighbors approach. In this approach, the standard errors for the cluster-level effect estimates are assumed to be fixed.

Usage

```
KnnTausqEst(Y, X, ses, trim.quant = 0, k = 1)
```

Arguments

Y	The K x 1 vector of cluster-specific effect estimates.
X	The K x p design matrix containing cluster-specific covariates.
ses	The K x 1 vector sampling standard errors squared.
trim.quant	The proportion of extreme estimates to exclude when estimating the random effects variance. Cluster-specific estimates lower than the trim.quant quantile will be set equal to the trim.quant quantile, and estimates larger than the 1 - trim.quant quantile will be set equal to the 1 - trim.quant quantile. This must be a number greater than or equal to 0 and less than 0.5.
k	The number of neighbors to include in the k nearest neighbors procedure. This should be an integer both greater than or equal to 1 and less than or equal to K.

Details

Calculation of the nearest neighbor estimator of the random effects variance involves a random split of the data into a training set D_1 and a test set D_2 of approximately equal sizes. The one-nearest neighbor estimator (i.e., when k=1) of τ^2 is then calculated as $\hat{\tau}^2_{1-NN} = \frac{1}{K}\sum_{j=1}^{K} Y_j^2 - \frac{1}{K_T}\sum_{j\in D_2} Y_j Y_j' - \frac{1}{K_T}\sum_{j\in D_2} ses_j^2$, where K_T is the number of observations in the test set D_2, and Y_j' is the effect estimate associated with the nearest neighbor $\mathbf{x}_j'$ of $\mathbf{x}_j$ in the training set.

Value

The estimate of the random effects variance. A vector of length 1.

Author(s)

Nicholas C. Henderson and Nicholas Hartman

References

Devroye, L., Györfi, L., Lugosi, G., & Walk, H. (2018). A nearest neighbor estimate of the residual variance. Electronic Journal of Statistics, 12(1), 1752–1778. doi:10.1214/18EJS1438

See Also

See also REMLTausqEst, ropper

Examples

```
############################################################
###  Example of kNN estimation with simulated data  #####
############################################################
n <- 500   ## 500 units
## True tau.sq is 0.5
tau.sq <- 0.5
```

```
cluster.sizes <- 1 + rpois(n, lambda=5)

xx <- rnorm(n)
theta.vec <- rnorm(n, sd=sqrt(tau.sq))

## True fixed-effect function is pnorm(x)
mu.vec <- pnorm(xx) + theta.vec
yy <- mu.vec + rnorm(n)/sqrt(cluster.sizes)

## Fit model where it is assumed that fixed-effect
## function is beta0 + beta1*x
XX <- cbind(rep(1, n), xx)
## define standard errors squared
stderrs_sq <- 1/cluster.sizes

## Run Knn estimation with a 1 nearest neighbor approach
tausq_hat_knn <- KnnTausqEst(Y=yy, X=XX, ses=stderrs_sq, k=1)
tausq_hat_knn
```

`mlb2025` *Baseball Winning Percentages in 2025*

Description

Data on the number of team wins during the 2025 major league baseball (MLB) season. The data includes the following additional team characteristics: payroll, strength of schedule, and ballpark difficulty.

Usage

```
data("mlb2025")
```

Format

A data frame with 29 observations on the following 8 variables.

`Team` a character vector containing team names.

`Wins` a numeric vector giving the number of total wins over the 2025 MLB season.

`Losses` a numeric vector giving the number of total losses over the 2025 MLB season.

`WinProp` a numeric vector with the proportion of wins for each team.

`StdErrSq` a numeric vector with the standard errors squared associated with the win proportions.

`Payroll` a numeric vector containing the projected payroll (in millions) at the start of the 2025 baseball season.

`SOS` a numeric vector measuring the 2025 strength of schedule.

`ParkFactors` a numeric vector measuring the overall difficulty of each team's home ballpark. Park difficulty is averaged over the 2024-2026 seasons.

Details

All MLB teams except the Athletics were included in this data.

Source

Total wins, total losses, and 2025 strength of schedule were obtained from https://www.baseball-reference.com/. The park factor index was retrieved from https://baseballsavant.mlb.com/. Payroll projections for 2025 were obtained from https://blogs.fangraphs.com/the-2025-payrolls-and-beyond/.

Examples

```
data(mlb2025)
str(mlb2025)

## Fit model with Payroll, SOS, ParkFactors as covariates
X1 <- model.matrix(~ Payroll + SOS + ParkFactors,
                   data=mlb2025)
rank_mod <- ropper(Y = mlb2025$WinProp,
                   X=X1,
                   ses=mlb2025$StdErrSq)

## Look at estimated regression coefficients
rank_mod$coef

## Look at ROPPER percentiles estimates
## Rank teams according to best percentiles
mlb_estperc <- data.frame(Team=mlb2025$Team,
                      percentile=round(rank_mod$pepp, 4))
mlb_estperc <- mlb_estperc[order(-mlb_estperc$percentile),]

head(mlb_estperc)
```

PEPP *Posterior expected population percentiles*

Description

Computes estimates of cluster-level posterior expected population percentiles given an estimate of the regression coefficients and an estimate of the random effects variance.

Usage

```
PEPP(beta.hat, Y, X, ses, tau.sq)
```

Arguments

`beta.hat`	The 1 x p vector of regression coefficient estimates.
`Y`	The K x 1 vector of cluster-specific effect estimates.
`X`	The K x p design matrix containing cluster-specific covariates.
`ses`	The K x 1 vector sampling standard errors squared.
`tau.sq`	Estimate of the random effects variance. This must be a scalar greater than 0.

Details

The posterior expected population percentile for cluster k is defined as $E(\rho_k|Y_k) = \Phi[V_k(Y_k - X_k^T\hat{\beta})]$, where Φ is the cumulative distribution of a standard normal distribution, $\hat{\beta}$ is the vector of regression coefficient estimates, and V_k is defined as $V_k = \tau^2/[(\tau^2 + ses_k)\sqrt{2ses_k + \tau^2}]$. The parameter ρ_k is the "population percentile" which is defined as $\rho_k = \Phi(v_k/\tau)$, where v_k is the random effect for the cluster-k effect estimate Y_k.

Value

A K x 1 vector of posterior expected population percentile estimates.

Author(s)

Nicholas C. Henderson and Nicholas Hartman

References

Henderson, N. C., & Hartman, N. (2025). Targeted Parameter Estimation for Robust Empirical Bayes Ranking. *arXiv preprint*. doi:10.48550/arXiv.2511.16530

See Also

See also `ropper`, `BLUP`.

Examples

```
#########################################################
###  Example of using PEPP with simulated data  #####
#########################################################
n <- 50
tau.sq <- 0.5
cluster.sizes <- 1 + rpois(n, lambda=5)

xx <- rnorm(n)
theta.vec <- rnorm(n, sd=sqrt(tau.sq))
## True fixed-effect function is pnorm(x)
mu.vec <- pnorm(xx) + theta.vec
yy <- mu.vec + rnorm(n)/sqrt(cluster.sizes)

## Fit model where it is assumed that fixed-effect
## function is beta0 + beta1*x
```

```
XX <- cbind(rep(1, n), xx)
## define standard errors squared
stderrs_sq <- 1/cluster.sizes


## Estimate regression coefficients by maximum likelihood
## estimation assuming that tau.sq = 0.5
beta.mle <- solve(crossprod(XX, (stderrs_sq + 0.5)*XX),
                  crossprod(XX, (stderrs_sq + 0.5)*yy))
## Extract the PEPP associated with the maximum
## likelihood estimate:
pepp.mle <- PEPP(beta.mle, Y=yy, X=XX, ses=stderrs_sq,
                 tau.sq=0.5)

## Get PEPP using ROPPER
rfit <- ropper(Y=yy, X=XX, ses=stderrs_sq)
pepp_rop1 <- rfit$pepp
## This should match using PPEP with ROPPER estimates of
## regression coefficients and tau.sq
pepp_rop2 <- PEPP(rfit$coefficients, Y=yy, X=XX,
                  ses=stderrs_sq, tau.sq=rfit$tau.sq.hat)

all.equal(pepp_rop1, pepp_rop2)
```

`REMLTausqEst` *Restricted maximum likelihood estimation of random effects variance*

Description

Computes an estimate of the random effects variance by maximizing the restricted maximum likelihood objective function. The standard errors for the cluster-level effect estimates are assumed to be fixed.

Usage

```
REMLTausqEst(Y, X, ses, trim.quant = 0)
```

Arguments

`Y`	The K x 1 vector of cluster-specific effect estimates.
`X`	The K x p design matrix containing cluster-specific covariates.
`ses`	The K x 1 vector sampling standard errors squared.
`trim.quant`	The proportion of extreme estimates to exclude when estimating the random effects variance. Cluster-specific estimates lower than the `trim.quant` quantile will be set equal to the `trim.quant` quantile, and estimates larger than the 1 - `trim.quant` quantile will be set equal to the 1 - `trim.quant` quantile. This must be a number greater than or equal to 0 and less than 0.5.

Details

Estimate of the random effects variance τ^2 is found by maximizing the restricted maximum likelihood objective function $-\log\det(\mathbf{W})-\log\det(\mathbf{X}^T\mathbf{W}^{-1}\mathbf{X})+\mathbf{Y}^T\mathbf{P}\mathbf{Y}$ with respect to τ^2. Here, $\mathbf{Y}$ is the K x 1 vector of cluster-level effect estimates, $\mathbf{W}$ is the K x K diagonal matrix with diagonal elements $\tau^2+ses_k^2$, and $\mathbf{P}$ is the K x K matrix defined as $\mathbf{P}=\mathbf{W}^{-1}-\mathbf{W}^{-1}\mathbf{X}(\mathbf{X}^T\mathbf{W}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}^{-1}$

Value

The estimate of the random effects variance. A vector of length 1.

Author(s)

Nicholas C. Henderson and Nicholas Hartman

References

Jiang, J., & Nguyen, T. (2021). Linear and Generalized Linear Mixed Models and Their Applications (2nd ed.). Springer. doi:10.1007/9781071612828

See Also

See also ropper,

Examples

```
############################################################
###  Example of REML estimation with simulated data  #####
############################################################
n <- 500   ## 500 units
## True tau.sq is 0.5
tau.sq <- 0.5
cluster.sizes <- 1 + rpois(n, lambda=5)

xx <- rnorm(n)
theta.vec <- rnorm(n, sd=sqrt(tau.sq))

mu.vec <- xx + theta.vec
yy <- mu.vec + rnorm(n)/sqrt(cluster.sizes)

## Fit model where it is assumed that fixed-effect
## function is beta0 + beta1*x
XX <- cbind(rep(1, n), xx)
## define standard errors squared
stderrs_sq <- 1/cluster.sizes

tausq_hat_reml <- REMLTausqEst(Y=yy, X=XX, ses=stderrs_sq)
tausq_hat_reml
```

ropper *Ranking-targeted estimation of covariate-adjusted rankings*

Description

Computes ranking of cluster-specific estimates after performing a covariate adjustment.

Usage

```
ropper(Y, X, ses, tau.sq = c("REML", "kNN"), H=1,
       opt.method=c("MM", "optim"), trim.quant=0,
       control = list())
```

Arguments

Y	The K x 1 vector of cluster-specific effect estimates.
X	The K x p design matrix containing cluster-specific covariates.
ses	The K x 1 vector sampling standard errors squared.
tau.sq	Method for estimating the random effects variance. This can be either the restricted maximum likelihood estimation (REML) REML or k-nearest neighbors kNN.
H	The degree of the approximation used for the unbiased risk estimator. This must be an integer greater than or equal to 1.
opt.method	The optimization method used for computing the regression coefficient estimates. This can be either an MM algorithm or the use of the optim function.
trim.quant	The proportion of extreme estimates to exclude when estimating the random effects variance. Cluster-specific estimates lower than the trim.quant quantile will be set equal to the trim.quant quantile, and estimates larger than the 1 - trim.quant quantile will be set equal to the 1 - trim.quant quantile. This must be a number greater than or equal to 0 and less than 0.5.
control	A list of control parameters specifying any changes to default values of the optimization procedures. Full names of control list elements must be specified, otherwise, user-specifications are ignored. See *Details*.

Details

Default values of control are: maxiter=500, tol=1e-06

maxiter An integer denoting the maximum limit on the number of MM iterations or optim iterations. Default value is 500.

tol A small, positive scalar that determines when iterations should be terminated. Iterations are terminated when $||\beta_{k+1} - \beta_k|| <$ tol. Default is 1.e-06.

Value

A list with the following components:

`coefficients`	The length-p vector of ranking-targeted regression coefficient estimates
`pepp`	The length-K vector of estimated posterior expected population percentiles
`objfn.track`	A vector containing the value of the objective function at each iteration
`tau.sq.hat`	The estimate of the random effects variance

Author(s)

Nicholas C. Henderson and Nicholas Hartman

References

Henderson, N. C., & Hartman, N. (2025). Targeted Parameter Estimation for Robust Empirical Bayes Ranking. *arXiv preprint*. doi:10.48550/arXiv.2511.16530.

Examples

```
#########################################################
###  Example of using ropper with simulated data  #####
#########################################################
n <- 50
tau.sq <- 0.5
cluster.sizes <- 1 + rpois(n, lambda=5)

set.seed(2507)
xx <- rnorm(n)
theta.vec <- rnorm(n, sd=sqrt(tau.sq))
## True fixed-effect function is pnorm(x)
mu.vec <- pnorm(xx) + theta.vec
yy <- mu.vec + rnorm(n)/sqrt(cluster.sizes)

## Fit model where it is assumed that fixed-effect
## function is beta0 + beta1*x
XX <- cbind(rep(1, n), xx)
## define standard errors squared
stderrs_sq <- 1/cluster.sizes

rfit <- ropper(Y=yy, X=XX, ses=stderrs_sq)

## Regression coefficient estimates
rfit$coefficients

## ROPPER percentile estimates for first 5 units
rfit$pepp[1:5]

## Plot true random effect rankings vs. ROPPer rankings:
true_ranks <- rank(theta.vec)
ropper_ranks <- rank(rfit$pepp)
```

```
plot(true_ranks, ropper_ranks, las=1,
     xlab="True RE Ranking", ylab="ROPPER Ranking")
abline(0, 1)

######
## Example of using ropper with optim for optimization
## instead of MM
######

rfit.optim <- ropper(Y=yy, X=XX, ses=stderrs_sq,
                     opt.method="optim")

## Compare coefficients obtained from the two optimization methods
round(cbind(rfit$coefficients, rfit.optim$coefficients), 4)

## Compare first 5 percentiles obtained from the two optimization methods
round(cbind(rfit$pepp[1:5], rfit.optim$pepp[1:5]), 4)
```

Index