Package 'ebTobit'

May 4, 2024

Type Package

Title Empirical Bayesian Tobit Matrix Estimation

Version 1.0.2

Date 2024-05-03

Description Estimation tools for multidimensional Gaussian means using empirical Bayesian g-modeling. Methods are able to handle fully observed data as well as left-, right-, and interval-censored observations (Tobit likelihood); descriptions of these methods can be found in Barbehenn and Zhao (2023) <doi:10.48550/arXiv.2306.07239>. Additional, lower-level functionality based on Kiefer and Wolfowitz (1956) <doi:10.1214/aoms/1177728066> and Jiang and Zhang (2009) <doi:10.1214/08-AOS638> is provided that can be used to accelerate many empirical Bayes and nonparametric maximum likelihood problems.

License GPL-3

URL https://github.com/barbehenna/ebTobit

BugReports https://github.com/barbehenna/ebTobit/issues

Imports Rcpp (>= 1.0.10), RcppParallel, stats

Suggests REBayes

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.3.0)

NeedsCompilation yes

Author Alton Barbehenn [aut, cre] (<https://orcid.org/0009-0000-3364-7204>), Sihai Dave Zhao [aut]

Maintainer Alton Barbehenn <altonbarbehenn@gmail.com>

Repository CRAN

Date/Publication 2024-05-03 23:00:02 UTC

R topics documented:

`BileAcid` *Bile Acid Data*

Description

A bile acid data set taken from Lei et al. (2018) doi:10.1096/fj.201700055R via Wei et al. (2018) doi:10.1371/journal.pcbi.1005973 (corresponding GitHub repository: https://github.com/WandeRum/GSimp). The values in `BileAcid` can be assumed to be independent log-normal measurements.

Usage

```
BileAcid
```

Format

A data frame with 198 rows and 34 variables. Each row is a patient id and each column is an bile acid measurement.

ConvexDual *Convex Optimization of the Kiefer-Wolfowitz NPMLE*

Description

This method only works if there is a working installation of `REBayes` available. See the `REBayes` package and corresponding papers for more implementation details.

Usage

```
ConvexDual(A, ...)
```

Arguments

`A`	numeric matrix likelihoods
`...`	further arguments passed to `Rmosek` such as `rtol`

Details

The matrix `A` is structured as follows: A_ij = P(X_i | theta = t_j), where X_i is the i'th observation and t_j is the j'th set of parameters/grid-point.

Value

a vector containing the fitted prior

ConvexPrimal *Convex Optimization of the Kiefer-Wolfowitz NPMLE*

Description

This method only works if there is a working installation of `REBayes` available. See the `REBayes` package and corresponding papers for more implementation details.

Usage

```
ConvexPrimal(A, ...)
```

Arguments

`A`	numeric matrix likelihoods
`...`	further arguments passed to `Rmosek` such as `rtol`

Details

The matrix A is structured as follows: A_ij = P(X_i | theta = t_j), where X_i is the i'th observation and t_j is the j'th set of parameters/grid-point.

Value

a vector containing the fitted prior

---

ebTobit *Empirical Bayes Matrix Estimation under a Tobit Likelihood*

Description

Fit and estimate the nonparametric maximum likelihood estimator in R^p (p >= 1) when the likelihood is Gaussian and possibly interval censored. If p = 1, then L, R, and gr may be vectors (they are immediately converted into matrices internally).

Usage

```
ebTobit(
  L,
  R = L,
  gr = (R + L)/2,
  s1 = 1,
  algorithm = "EM",
  pos_lik = TRUE,
  ...
)
```

Arguments

L	n x p matrix of lower bounds on observations
R	n x p matrix of upper bounds on observations
gr	m x p matrix of grid points
s1	a single numeric standard deviation or an n x p matrix of standard deviations
algorithm	method to fit prior, either a function or function name
pos_lik	boolean indicating whether to lower-bound the likelihood matrix with .Machine$double.xmin (default: TRUE); helps avoid possible divide-by-zero errors in algorithm
...	further arguments passed into fitting method such as rtol and maxiter, see for example EM

Details

Each observation is stored in a pair of matrices, L and R. If L_ij = R_ij then a direct measurement X_ij ~ N(theta, s1^2) is made; if L_ij < R_ij then the measurement is censored so that L_ij < X_ij < R_ij.

To use a custom fitting algorithm, define a function MyAlg that takes in an n x m likelihood matrix: P_ij = P(L_i, R_i | theta = t_j) and returns a vector of estimated prior weights for t_j. Once MyAlg is defined, fit the prior by using algorithm = "MyAlg" or use the function itself algorithm = MyAlg.

Alternative fitting algorithms "ConvexPrimal"and "ConvexDual" have been (wrappers of REBayes::KWPrimal and REBayes::KWDual, respectively) included and can be used if MOSEK and REBayes are properly installed.

Value

a fitted ebTobit object containing at least the prior weights, corresponding grid/support points, and likelihood matrix relating the grid to the observations

Examples

```
set.seed(1)
n <- 100
p <- 5
r <- 2
U.true <- matrix(stats::rexp(n*r), n, r)
V.true <- matrix(sample(x = c(1,4,7),
                        size = p*r,
                        replace = TRUE,
                        prob = c(0.7, 0.2, 0.1)),
                 p, r)
TH <- tcrossprod(U.true, V.true)
X <- TH + matrix(stats::rnorm(n*p), n, p)

# fit uncensored method
fit1 <- ebTobit(X)

# fit left-censored method
ldl <- 1 # lower and upper detection limits
udl <- Inf
L <- ifelse(X < ldl, 0, ifelse(X <= udl, X, udl))
R <- ifelse(X < ldl, ldl, ifelse(X <= udl, X, Inf))
fit2 <- ebTobit(L, R)
```

Description

Compute the nonparametric maximum likelihood estimate given a likelihood matrix. The matrix A is structured so that A_{ij} = f(X_i | theta_j) for some grid of potential parameter values theta_1, ..., theta_p and observations X_1, ..., X_n. The parameters, theta_j, can be multidimensional because all that is required is the likelihood. Convergence is achieved when the relative improvements of the log-likelihood is below the provided tolerance level.

Usage

```
EM(A, maxiter = 10000L, rtol = 1e-06)
```

Arguments

A	numeric matrix likelihoods
maxiter	early stopping condition
rtol	convergence tolerance: abs(loss_new - loss_old)/abs(loss_old)

Value

the estimated prior distribution (a vector of masses corresponding to the columns of A)

Examples

```
set.seed(1)
t = sample(c(0,5), size = 100, replace = TRUE)
x = t + stats::rnorm(100)
gr = seq(from = min(x), to = max(x), length.out = 50)
A = stats::dnorm(outer(x, gr, "-"))
EM(A)
## Not run:
# compare to solution from rmosek (requires additional library installation):
all.equal(
    REBayes::KWPrimal(A = A, d = rep(1, 50), w = rep(1/100, 100))$f,
    EM(A, maxiter = 1e+6, rtol = 1e-16), # EM alg converges slowly
    tolerance = 0.01
)

## End(Not run)
```

---

fitted.ebTobit *Fitted Estimates of an ebTobit object*

Description

Compute either the posterior mean (default) or posterior L1 mediod which corresponds to the posterior median in one-dimension.

Usage

```
## S3 method for class 'ebTobit'
fitted(object, method = "mean", ...)
```

Arguments

object	an object inheriting from class ebTobit
method	either "mean", "L1mediod", or "mode" corresponding to the methods: posterior_*.ebTobit()
...	not used

Value

matrix containing the posterior estimates for measurements in the fit empirical Bayes model object

---

is.ebTobit *Validate ebTobit Object*

Description

Validate ebTobit Object

Usage

```
is.ebTobit(object)
```

Arguments

object	any R object

Value

boolean: TRUE if the object is a valid ebTobit object

---

likMat *Helper Function - generate likelihood matrix*

Description

Compute a matrix L whose entries are L[i,k] = P(L_i, R_i | theta = t_k) for observations (L_i, R_i) and grid of means t_k.

Usage

```
likMat(L, R, gr, s1)
```

Arguments

L	n x p matrix of lower bounds
R	n x p matrix of upper bounds
gr	m x p matrix of candidate means
s1	n x p matrix of standard deviations

Value

the n x m likelihood matrix under partial interval censoring

Examples

```
# set-up
n = 100; m = 50; p = 5
gr = matrix(stats::rnorm(m*p), m, p)
L = R = matrix(stats::rnorm(n*p), n, p)
s1 = matrix(1, n, p)
missing.idx = sample.int(n = n*p, size = p*p)
L[missing.idx] = L[missing.idx] - stats::runif(p, 0, 1)

# R solution
lik = matrix(nrow = n, ncol = m)
for (i in 1:n) {
    for(k in 1:m) {
        lik[i,k] = prod(ifelse(
            L[i,] == R[i,],
            stats::dnorm(L[i,]-gr[k,], sd = s1[i,]),
         stats::pnorm(R[i,]-gr[k,], sd = s1[i,]) - stats::pnorm(L[i,]-gr[k,], sd = s1[i,])
        ))
    }
}

# Compare R to RcppParallel method
all.equal(lik, likMat(L, R, gr, s1))
```

---

lik_GaussianPIC *Helper Function - generate likelihood for pair (L,R) and mean gr*

Description

Compute P(L_i, R_i | theta = t_k) for observations (L_i, R_i) and grid of mean t_k.

Usage

```
lik_GaussianPIC(L, R, gr, s1)
```

Arguments

L	numeric vector of lower bounds
R	numeric vector of upper bounds
gr	numeric vector of means
s1	numeric vector of standard deviations

Value

the likelihood under partial interval censoring

Examples

```
# set-up
p = 15
gr = stats::rnorm(p)
L = R = stats::rnorm(p)
missing.idx = sample.int(n = p, size = p/5)
L[missing.idx] = L[missing.idx] - stats::runif(length(missing.idx), 0, 1)
R[missing.idx] = R[missing.idx] + stats::runif(length(missing.idx), 0, 1)

# R solution
lik = prod(ifelse(
           L == R,
           stats::dnorm(L-gr),
           stats::pnorm(R-gr) - stats::pnorm(L-gr)))

# Compare R to RcppParallel method
all.equal(lik, lik_GaussianPIC(L, R, gr, rep(1,p)))
```

---

logLik.ebTobit *Marginal Log-likelihood of an ebTobit object*

Description

Marginal Log-likelihood of an ebTobit object

Usage

```
## S3 method for class 'ebTobit'
logLik(object, ...)
```

Arguments

object	an object inheriting from class ebTobit
...	not used

Value

log likelihood for the fitted empirical Bayes model in object

`new_ebTobit` *Create a new ebTobit object*

Description

Validate the provided elements and populate the object. Current methods require that `gr` is numeric for that calculation of posterior statistics (mean and mediod).

Usage

```
new_ebTobit(prior, gr, lik)
```

Arguments

`prior`	numeric vector of non-negative weights (sums to one)
`gr`	numeric matrix of support points
`lik`	numeric matrix of likelihoods

Value

an `EBayesMat` object containing at least the prior weights, corresponding grid/support points, and likelihood matrix relating the grid to the observations

`posterior_L1mediod.ebTobit`
Compute the Posterior L1 Mediod of an ebTobit object

Description

The posterior L1 mediod is defined as \arg\min_y E |y - t|_1 where the expectation is taken over the posterior t|X=x. Here the posterior L1 mediod is evaluated for each of the observations used to fit `object`.

Usage

```
posterior_L1mediod.ebTobit(object)
```

Arguments

`object`	an object inheriting from class `ebTobit`

Value

numeric matrix of posterior L1 mediods for the fitted empirical Bayes model in `object`

`posterior_mean.ebTobit` *Compute Posterior Mean of an ebTobit object*

Description

Compute Posterior Mean of an ebTobit object

Usage

```
posterior_mean.ebTobit(object)
```

Arguments

`object`	an object inheriting from class `ebTobit`

Value

numeric matrix of posterior means for the fitted empirical Bayes model in `object`

`posterior_mode.ebTobit` *Compute Posterior Mode of an ebTobit object*

Description

Compute Posterior Mode of an ebTobit object

Usage

```
posterior_mode.ebTobit(object)
```

Arguments

`object`	an object inheriting from class `ebTobit`

Value

numeric matrix of posterior modes for the fitted empirical Bayes model in `object`

predict.ebTobit *Fitted Estimates of an ebTobit object*

Description

Compute either the posterior mean (default) or posterior L1 mediod which corresponds to the posterior median in one-dimension.

Usage

```
## S3 method for class 'ebTobit'
predict(object, L, R = L, s1 = 1, method = "mean", ...)
```

Arguments

object	an object inheriting from class ebTobit
L	n x p matrix of lower bounds on observations
R	n x p matrix of upper bounds on observations
s1	a single numeric standard deviation or an n x p matrix of standard deviations
method	either "mean", "L1mediod", or "mode" corresponding to the methods: posterior_*.ebTobit()
...	not used

Value

matrix of posterior estimates for new observations under the provided, pre-fit empirical Bayes model object

tobit_sd *Fit Tobit Standard Deviation via Maximum Likelihood*

Description

Fit the matrix of standard deviations given censored observations current mean estimates. Currently there are four models for S implemented: global, column-specific, row-specific, and rank-1.

Usage

```
tobit_sd(
  L,
  R,
  mu = matrix(colMeans(L + R)/2, nrow(L), ncol(L), byrow = TRUE),
  sd.structure = "global",
  interval = c(1e-04, 100),
  tol = .Machine$double.eps^0.25,
  maxiter = 1000
)
```

Arguments

L	matrix of lower bounds on observations (n x p)
R	matrix of upper bounds on observations (n x p)
mu	matrix of known means (n x p)
sd.structure	structure imposed on noise level estimates, must be one of: "global", "column", "row", or "rank1"
interval	a vector containing the end-points of the interval defining the convex search space (default: c(1e-4, 1e+2))
tol	the desired accuracy
maxiter	early stopping condition

Value

matrix of maximum likelihood estimates for each observation's standard deviation (n x p)

Examples

```
set.seed(1)
n = 100; p = 5; r = 2
U.true = matrix(stats::rexp(n*r), n, r)
V.true = matrix(sample(x = c(1,4,7),
                       size = p*r,
                       replace = TRUE,
                       prob = c(0.7, 0.2, 0.1)),
                p, r)
TH = tcrossprod(U.true, V.true)
X = TH + matrix(stats::rnorm(n*p, sd = 1), n, p)
ldl <- 0.1 # lower detection limit, known to be non-negative
L <- ifelse(X < ldl, 0, X)
R <- ifelse(X < ldl, ldl, X)

tobit_sd(L, R, mu = TH)
tobit_sd(L, R, mu = TH, sd.structure = "column")
```

---

tobit_sd_mle *Maximum Likelihood Estimator for a Single Standard Deviation Parameter*

Description

Use standard numerical optimization methods to maximize the log-likelihood of the given problem. If all of the data is passed in, this method computes the global estimate of standard deviation. By passing in a subset of the data, more specific estimates can be made (ex column-specific standard deviations).

Usage

```
tobit_sd_mle(
  L,
  R,
  mu = matrix(mean(L + R)/2, nrow(L), ncol(L)),
  interval = c(1e-04, 100),
  tol = .Machine$double.eps^0.25
)
```

Arguments

L	matrix of lower bounds on observations (n x p)
R	matrix of upper bounds on observations (n x p)
mu	matrix of known means (n x p)
interval	a vector containing the end-points of the interval defining the convex search space (default: c(1e-4, 1e+2))
tol	the desired accuracy

Value

a list containing estimate (maximum) and log-likelihood (objective)

Examples

```
set.seed(1)
n = 100; p = 5; r = 2
U.true = matrix(stats::rexp(n*r), n, r)
V.true = matrix(sample(x = c(1,4,7),
                       size = p*r,
                       replace = TRUE,
                       prob = c(0.7, 0.2, 0.1)),
                p, r)
TH = tcrossprod(U.true, V.true)
X = TH + matrix(stats::rnorm(n*p, sd = 1), n, p)
ldl <- 0.1 # lower detection limit, known to be non-negative
L <- ifelse(X < ldl, 0, X)
R <- ifelse(X < ldl, ldl, X)
tobit_sd_mle(L, R, mu = TH)
```

Index