Package ‘ebnm’

October 13, 2022

Encoding UTF-8
Type Package
Version 1.0-9
Date 2022-03-02
Title Solve the Empirical Bayes Normal Means Problem

URL https://github.com/stephenslab/ebnm

BugReports https://github.com/stephenslab/ebnm/issues

Description Provides simple, fast, and stable functions to fit the normal means model using empirical Bayes. For available models and details, see function ebnm(). A comprehensive introduction to the package is provided by Willwerscheid and Stephens (2021) <arXiv:2110.00152>.

License GPL (>= 3)

NeedsCompilation no

Depends R (>= 3.3.0)

Imports stats, methods, ashr, mixsqp, truncnorm, trust, horseshoe, deconvolveR, magrittr, rlang, dplyr, ggplot2

Suggests testthat, numDeriv, REBayes

RoxygenNote 7.1.2

Author Willwerscheid Jason [aut], Stephens Matthew [aut], Carbonetto Peter [aut, cre], Goldstein Andrew [ctb]

Maintainer Carbonetto Peter <peter.carbonetto@gmail.com>

Repository CRAN

Date/Publication 2022-03-08 20:10:02 UTC
R topics documented:

ebnm ................................................................. 2
ebnm_point_normal ............................................... 6
gammamix .......................................................... 13
horseshoe ........................................................... 14
laplacemix ........................................................... 14
plot.ebnm ............................................................ 15
print.ebnm ........................................................... 16

Index 17

ebnm Solve the EBNM problem

Description
Solves the empirical Bayes normal means (EBNM) problem using a specified family of priors. For a comprehensive introduction to the package, see the paper cited in References below.

Usage

```r
ebnm(
  x,
  s = 1,
  prior_family = c("point_normal", "point_laplace", "point_exponential", "normal",
                   "horseshoe", "normal_scale_mixture", "unimodal", "unimodal_symmetric",
                   "unimodal_nonnegative", "unimodal_nonpositive", "npmle", "deconvolver", "ash"),
  mode = 0,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
  output = output_default(),
  optmethod = NULL,
  control = NULL,
  ...)
```

output_default()

output_all()

Arguments

- **x** A vector of observations. Missing observations (NAs) are not allowed.
- **s** A vector of standard errors (or a scalar if all are equal). Standard errors may not be exactly zero, and missing standard errors are not allowed. Two prior families have additional restrictions: when horseshoe priors are used, errors must be...
homoskedastic; and since function `deconv` in package `deconvolveR` takes z-scores, all standard errors must be equal to 1 when the "deconvolver" family is used.

**prior_family**
A character string that specifies the prior family \( G \). See "Details" below.

**mode**
A scalar specifying the mode of the prior \( g \) or "estimate" if the mode is to be estimated from the data. This parameter is ignored by the NPMLE and `deconvolveR` prior family.

**scale**
A scalar or vector specifying the scale parameter(s) of the prior or "estimate" if the scale parameters are to be estimated from the data. The interpretation of `scale` depends on the prior family. For normal and point-normal families, it is a scalar specifying the standard deviation of the normal component. For point-Laplace and point-exponential families, it is a scalar specifying the scale parameter of the Laplace or exponential component. For the horseshoe family, it corresponds to \( s\tau \) in the usual parametrization of the horseshoe distribution. For the NPMLE and `deconvolveR` prior family, it is a scalar specifying the distance between support points. For all other prior families, which are implemented using the function `ash` in package `ashr`, it is a vector specifying the parameter \( \text{mixsd} \) to be passed to `ash` or "estimate" if \( \text{mixsd} \) is to be chosen by `ebnm`. (Note that `ebnm` chooses \( \text{mixsd} \) differently from `ashr`. To use the `ashr` grid, set `scale = "estimate"` and pass in `gridmult` as an additional parameter. See `ash` for defaults and details.)

**g_init**
The prior distribution \( g \). Usually this is left unspecified (NULL) and estimated from the data. However, it can be used in conjunction with `fix_g = TRUE` to fix the prior (useful, for example, to do computations with the "true" \( g \) in simulations). If `g_init` is specified but `fix_g = FALSE`, `g_init` specifies the initial value of \( g \) used during optimization. For non-parametric priors, this has the side effect of fixing the mode and `scale` parameters. If `g_init` is supplied, it should be an object of class `normalmix` for normal, point-normal, scale mixture of normals, and `deconvolveR` prior families, as well as for the NPMLE; class `laplacemix` for point-Laplace families; class `gammamix` for point-exponential families; class `horseshoe` for horseshoe families; and class `unimix` for unimodal_families.

**fix_g**
If TRUE, fix the prior \( g \) at `g_init` instead of estimating it.

**output**
A character vector indicating which values are to be returned. Function `output_default()` provides the default return values, while `output_all()` lists all possible return values. See "Value" below.

**optmethod**
A string specifying which optimization function is to be used. Options include "nlm", "lbfgsb" (which calls `optim` with method = "L-BFGS-B"), and "trust" (which calls into package `trust`). Other options are "nohess_nlm", "nograd_nlm", and "nograd_lbfgsb", which use numerical approximations rather than exact expressions for the Hessian and (for the latter two) the gradient. The default option is "nohess_nlm". Since all non-parametric families rely upon external packages, this parameter is only available for parametric families (point-normal, point-Laplace, point-exponential, and normal).

**control**
A list of control parameters to be passed to the optimization function. `optimize` is used for normal and horseshoe prior families, while `nlm` is used for parametric families unless parameter `optmethod` specifies otherwise. `nlm` is also used for
the deconvolver prior family. For ash families (including scale mixtures of normals, the NPMLE, and all unimodal families), function mixsqp in package mixsqp is the default.

Additional parameters. When a unimodal prior family is used, these parameters are passed to function ash in package arhs. When the "deconvolver" family is used, they are passed to function deconv in package deconvolver. Although it does not call into arhs, the scale mixture of normals family accepts parameter gridmult for purposes of comparison. When gridmult is set, an arhs-style grid will be used instead of the default ebnm grid. In all other cases, additional parameters are ignored.

Details

Given vectors of data $x$ and standard errors $s$, ebnm solves the "empirical Bayes normal means" (EBNM) problem for various choices of prior family. The model is

$$x_j | \theta_j, s_j \sim N(\theta_j, s_j^2)$$

$$\theta_j \sim g \in G,$$

where $g$, which is referred to as the "prior distribution" for $\theta$, is to be estimated from among some specified family of prior distributions $G$. Several options for $G$ are implemented, some parametric and others non-parametric; see below for examples.

Solving the EBNM problem involves two steps. First, $g \in G$ is estimated via maximum marginal likelihood:

$$\hat{g} := \arg \max_{g \in G} L(g),$$

where

$$L(g) := \prod_j \int p(x_j | \theta_j, s_j) g(d\theta_j).$$

Second, posterior distributions $p(\theta_j | x_j, s_j, \hat{g})$ and/or summaries such as posterior means and posterior second moments are computed.

Implemented prior families include:

- point_normal: The family of mixtures where one component is a point mass at $\mu$ and the other is a normal distribution centered at $\mu$.
- point_laplace: The family of mixtures where one component is a point mass at zero and the other is a double-exponential distribution.
- point_exponential: The family of mixtures where one component is a point mass at zero and the other is a (nonnegative) exponential distribution.
- normal: The family of normal distributions.
- horseshoe: The family of horseshoe distributions.
- normal_scale_mixture: The family of scale mixtures of normals.
- unimodal: The family of all unimodal distributions.
- unimodal_symmetric: The family of symmetric unimodal distributions.
- unimodal_nonnegative: The family of unimodal distributions with support constrained to be greater than the mode.
unimodal_nonpositive  The family of unimodal distributions with support constrained to be less than the mode.

npmle  The family of all distributions.

dehover  A non-parametric exponential family with a natural spline basis. Like npmle, there is no unimodal assumption, but whereas npmle produces spiky estimates for \( g \), deconvolver estimates are much more regular. See deconvolver-package for details and references.

Value

An ebnm object. Depending on the argument to output, the object is a list containing elements:

data  A data frame containing the observations \( x \) and standard errors \( s \).

posterior  A data frame of summary results (posterior means, standard deviations, second moments, and local false sign rates).

fitted_g  The fitted prior \( \hat{g} \) (an object of class normalmix, laplacemix, gammamix, unimix, or horseshoe).

log_likelihood  The optimal log likelihood attained, \( L(\hat{g}) \).

posterior_sampler  A function that can be used to produce samples from the posterior. For all prior families other than the horseshoe, the sampler takes a single parameter nsamp, the number of posterior samples to return per observation. Since ebnm_horseshoe returns an MCMC sampler, it additionally takes parameter burn, the number of burn-in samples to discard.

Functions

• output_default: Lists the default return values.

• output_all: Lists all valid return values.

References


See Also

A plotting method is available for ebnm objects: see plot.ebnm.

Calling functions ebnm_point_normal, ebnm_point_laplace, ebnm_point_exponential, ebnm_normal, ebnm_horseshoe, ebnm_normal_scale_mixture, ebnm_unimodal, ebnm_unimodal_symmetric, ebnm_unimodal_nonnegative, ebnm_unimodal_nonpositive, ebnm_npmle, ebnm_deconvolver, and ebnm_ash is equivalent to calling ebnm with prior_family set accordingly.

Examples

```r
theta <- c(rep(0, 100), rexp(100))
s <- 1
x <- theta + rnorm(200, 0, s)

# The following are equivalent:
pn.res <- ebnm(x, s, prior_family = "point_normal")
```
ebnm_point_normal

Solve the EBNM problem using a specified family of priors

Description

Each of the functions listed below solves the empirical Bayes normal means (EBNM) problem using a specified family of priors. Calling function ebnm_xxx is equivalent to calling function ebnm with argument prior_family = "xxx". For details about the model, see ebnm or the paper cited in References below.

Usage

```
ebnm_point_normal(
  x,
  s = 1,
  mode = 0,
```
ebnm_point_normal

    scale = "estimate",
    g_init = NULL,
    fix_g = FALSE,
    output = output_default(),
    optmethod = NULL,
    control = NULL

)  
ebnm_point_laplace(
    x,
    s = 1,
    mode = 0,
    scale = "estimate",
    g_init = NULL,
    fix_g = FALSE,
    output = output_default(),
    optmethod = NULL,
    control = NULL

)  
ebnm_point_exponential(
    x,
    s = 1,
    mode = 0,
    scale = "estimate",
    g_init = NULL,
    fix_g = FALSE,
    output = output_default(),
    optmethod = NULL,
    control = NULL

)  
ebnm_normal(
    x,
    s = 1,
    mode = 0,
    scale = "estimate",
    g_init = NULL,
    fix_g = FALSE,
    output = output_default(),
    optmethod = NULL,
    control = NULL

)  
ebnm_horseshoe(
    x,
    s = 1,
    scale = "estimate",
ebnm_point_normal

    g_init = NULL,
    fix_g = FALSE,
    output = output_default(),
    control = NULL
  )

ebnm_normal_scale_mixture(  
  x,
  s = 1,
  mode = 0,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
  output = output_default(),
  control = NULL,
  ...
)

ebnm_unimodal(  
  x,
  s = 1,
  mode = 0,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
  output = output_default(),
  control = NULL,
  ...
)

ebnm_unimodal_symmetric(  
  x,
  s = 1,
  mode = 0,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
  output = output_default(),
  control = NULL,
  ...
)

ebnm_unimodal_nonnegative(  
  x,
  s = 1,
  mode = 0,
  scale = "estimate",
  g_init = NULL,
ebnm_point_normal

  fix_g = FALSE,
  output = output_default(),
  control = NULL,
  ...
)

ebnm_unimodal_nonpositive(
  x,
  s = 1,
  mode = 0,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
  output = output_default(),
  control = NULL,
  ...
)

ebnm_ash(
  x,
  s = 1,
  mode = 0,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
  output = output_default(),
  control = NULL,
  ...
)

ebnm_npmle(
  x,
  s = 1,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
  output = output_default(),
  optmethod = NULL,
  control = NULL,
  ...
)

ebnm_deconvolver(
  x,
  s = 1,
  scale = "estimate",
  g_init = NULL,
  fix_g = FALSE,
output = output_default(),
control = NULL,
...)

Arguments

x     A vector of observations. Missing observations (NAs) are not allowed.
s     A vector of standard errors (or a scalar if all are equal). Standard errors may not
be exactly zero, and missing standard errors are not allowed. Two prior families
have additional restrictions: when horseshoe priors are used, errors must be
homoskedastic; and since function deconv in package deconvolverR takes z-
scores, all standard errors must be equal to 1 when the "deconvolver" family is
used.
mode   A scalar specifying the mode of the prior g or "estimate" if the mode is to
be estimated from the data. This parameter is ignored by the NPMLE and
deconvolverR prior family.
scale  A scalar or vector specifying the scale parameter(s) of the prior or
"estimate" if the scale parameters are to be estimated from the data. The interpretation
of scale depends on the prior family. For normal and point-normal families,
it is a scalar specifying the standard deviation of the normal component. For
point-Laplace and point-exponential families, it is a scalar specifying the scale
parameter of the Laplace or exponential component. For the horseshoe family,
it corresponds to sτ in the usual parametrization of the horseshoe distribu-
tion. For the NPMLE and deconvolverR prior family, it is a scalar specifying
the distance between support points. For all other prior families, which are im-
plemented using the function ash in package ashr, it is a vector specifying the
parameter mixsd to be passed to ash or "estimate" if mixsd is to be chosen by
ebnm. (Note that ebnm chooses mixsd differently from ashr. To use the ashr
grid, set scale = "estimate" and pass in gridmult as an additional parameter.
See ash for defaults and details.)
g_init The prior distribution g. Usually this is left unspecified (NULL) and estimated
from the data. However, it can be used in conjunction with fix_g = TRUE to fix the
prior (useful, for example, to do computations with the "true" g in simulations).
If g_init is specified but fix_g = FALSE, g_init specifies the initial value of
g used during optimization. For non-parametric priors, this has the side effect
of fixing the mode and scale parameters. If g_init is supplied, it should be an
object of class normalmix for normal, point-normal, scale mixture of normals,
and deconvolverR prior families, as well as for the NPMLE; class laplacemix
for point-Laplace families; class gammamix for point-exponential families; class
horseshoe for horseshoe families; and class unimix for unimodal families.
fix_g   If TRUE, fix the prior g at g_init instead of estimating it.
output A character vector indicating which values are to be returned. Function output_default()
provides the default return values, while output_all() lists all possible return
values. See "Value" below.
optmethod A string specifying which optimization function is to be used. Options in-
clude "nlm", "lbfgsb" (which calls optim with method = "L-BFGS-B"), and
"trust" (which calls into package trust). Other options are "nohess_nlm", "nograd_nlm", and "nograd_lbfgsb", which use numerical approximations rather than exact expressions for the Hessian and (for the latter two) the gradient. The default option is "nohess_nlm". Since all non-parametric families rely upon external packages, this parameter is only available for parametric families (point-normal, point-Laplace, point-exponential, and normal).

**control** A list of control parameters to be passed to the optimization function. `optimize` is used for normal and horseshoe prior families, while `nlm` is used for parametric families unless parameter `optmethod` specifies otherwise. `nlm` is also used for the deconvolveR prior family. For ash families (including scale mixtures of normals, the NPMLE, and all unimodal_ families), function `mixsqp` in package `mixsqp` is the default.

Additional parameters. When a unimodal_ prior family is used, these parameters are passed to function `ash` in package `ashr`. When the "deconvolver" family is used, they are passed to function `deconv` in package `deconvolveR`. Although it does not call into `ashr`, the scale mixture of normals family accepts parameter `gridmult` for purposes of comparison. When `gridmult` is set, an `ashr`-style grid will be used instead of the default `ebnm` grid. In all other cases, additional parameters are ignored.

**Details**

Implemented prior families include:

- **ebnm_point_normal** The family of mixtures where one component is a point mass at \( \mu \) and the other is a normal distribution centered at \( \mu \).
- **ebnm_point_laplace** The family of mixtures where one component is a point mass at zero and the other is a double-exponential distribution.
- **ebnm_point_exponential** The family of mixtures where one component is a point mass at zero and the other is a (nonnegative) exponential distribution.
- **ebnm_normal** The family of normal distributions.
- **ebnm_horseshoe** The family of horseshoe distributions.
- **ebnm_normal_scale_mixture** The family of scale mixtures of normals.
- **ebnm_unimodal** The family of all unimodal distributions.
- **ebnm_unimodal_symmetric** The family of symmetric unimodal distributions.
- **ebnm_unimodal_nonnegative** The family of unimodal distributions with support constrained to be greater than the mode.
- **ebnm_unimodal_nonpositive** The family of unimodal distributions with support constrained to be less than the mode.
- **ebnm_npmle** The family of all distributions.
- **ebnm_deconvolver** A non-parametric exponential family with a natural spline basis. Like `npmle`, there is no unimodal assumption, but whereas `npmle` produces spiky estimates for \( g \), deconvolver estimates are much more regular. See `deconvolveR-package` for details and references.
Value

An `ebnm` object. Depending on the argument to output, the object is a list containing elements:

- **data** A data frame containing the observations `x` and standard errors `s`.
- **posterior** A data frame of summary results (posterior means, standard deviations, second moments, and local false sign rates).
- **fitted_g** The fitted prior \( \hat{g} \) (an object of class `normalmix`, `laplacemix`, `gammamix`, `unimix`, or `horseshoe`).
- **log_likelihood** The optimal log likelihood attained, \( L(\hat{g}) \).
- **posterior_sampler** A function that can be used to produce samples from the posterior. For all prior families other than the horseshoe, the sampler takes a single parameter `nsamp`, the number of posterior samples to return per observation. Since `ebnm_horseshoe` returns an MCMC sampler, it additionally takes parameter `burn`, the number of burn-in samples to discard.

References


See Also

`ebnm`

Examples

```r
theta <- c(rep(0, 100), rexp(100))
s <- 1
x <- theta + rnorm(200, 0, s)
# The following are equivalent:
pn.res <- ebnm(x, s, prior_family = "point_normal")
pn.res <- ebnm_point_normal(x, s)
# Inspect results:
pn.res$log_likelihood
plot(pn.res)
# Fix the scale parameter:
pl.res <- ebnm_point_laplace(x, s, scale = 1)
pl.res$fitted_g$scale
# Estimate the mode:
normal.res <- ebnm_normal(x, s, mode = "estimate")
normal.res$fitted_g$mean
# Use an initial g (this fixes mode and scale for ash priors):
normalmix.res <- ebnm_normal_scale_mixture(x, s, g_init = pn.res$fitted_g)
# Fix g and get different output:
g_init <- pn.res$fitted_g
```
gammamix

Constructor for gammamix class

Description

Creates a finite mixture of gamma distributions.

Usage

gammamix(pi, shape, scale, shift = rep(0, length(pi)))

Arguments

pi      A vector of mixture proportions.
shape   A vector of shape parameters.
scale   A vector of scale parameters.
shift   A vector of shift parameters.

Value

An object of class gammamix (a list with elements pi, shape, scale, and shift, described above).
horseshoe  Constructor for horseshoe class

Description

Creates a horseshoe prior (see Carvalho, Polson, and Scott (2010)). The horseshoe is usually parametrized as $\theta_i \sim N(0, s^2 \tau^2 \lambda_i^2)$, $\lambda_i \sim \text{Cauchy}^+(0, 1)$, with $s^2$ the variance of the error distribution. We use a single parameter scale, which corresponds to $s\tau$ and thus does not depend on the error distribution.

Usage

horseshoe(scale)

Arguments

scale  The scale parameter (must be a scalar).

Value

An object of class horseshoe (a list with a single element scale, described above).

laplacemix  Constructor for laplacemix class

Description

Creates a finite mixture of Laplace distributions.

Usage

laplacemix(pi, mean, scale)

Arguments

pi  A vector of mixture proportions.
mean  A vector of means.
scale  A vector of scale parameters.

Value

An object of class laplacemix (a list with elements pi, mean, and scale, described above).
plot.ebnm

Plot an ebnm object

Description

Given a fitted ebnm object, produces a plot of posterior means vs. observations.

Usage

```
## S3 method for class 'ebnm'
plot(x, remove_abline = FALSE, ...)
```

Arguments

- `x` The fitted ebnm object.
- `remove_abline` To better illustrate shrinkage effects, the plot will include the line \( y = x \) by default. If `remove_abline = TRUE`, then this line will not be drawn.
- `...` Additional parameters to be passed to ggplot2 function `geom_point`.

Details

An object of class `ggplot` is returned, so that the plot can be customized in the usual ggplot2 fashion.

Value

A ggplot object.

Examples

```r
theta <- c(rep(0, 100), rexp(100))
s <- 1
x <- theta + rnorm(200, 0, s)
ebnm.res <- ebnm(x, s)
plot(ebnm.res)

# Customize plot:
library(ggplot2)
plot(ebnm.res, color = "blue", remove_abline = TRUE) +
  theme_bw() +
  labs(x = "Simulated data")
```
Print an ebnm object

Description

The print method for class ebnm.

Usage

```r
## S3 method for class 'ebnm'
print(x, ...)
```

Arguments

- `x` The fitted ebnm object.
- `...` Not used. Included for consistency as an S3 method.
Index

ash, 3, 4, 10, 11

decov, 3, 4, 10, 11

ebnm, 2, 6, 12
ebnm_ash, 5
ebnm_ash (ebnm_point_normal), 6
ebnm_deconvolver, 5
ebnm_deconvolver (ebnm_point_normal), 6
ebnm_horseshoe, 5
ebnm_horseshoe (ebnm_point_normal), 6
ebnm_normal, 5
ebnm_normal (ebnm_point_normal), 6
ebnm_normal_scale_mixture, 5
ebnm_normal_scale_mixture (ebnm_point_normal), 6
ebnm_npmle, 5
ebnm_npmle (ebnm_point_normal), 6
ebnm_point_exponential
  (ebnm_point_normal), 6
ebnm_point_laplace, 5
ebnm_point_laplace (ebnm_point_normal), 6
ebnm_point_normal, 5, 6
ebnm_unimodal, 5
ebnm_unimodal (ebnm_point_normal), 6
ebnm_unimodal_nonnegative, 5
ebnm_unimodal_nonnegative (ebnm_point_normal), 6
ebnm_unimodal_nonpositive, 5
ebnm_unimodal_nonpositive (ebnm_point_normal), 6
ebnm_unimodal_symmetric, 5
ebnm_unimodal_symmetric (ebnm_point_normal), 6

gammamix, 3, 5, 10, 12, 13
gem_point, 15
ggplot2, 15

horseshoe, 3–5, 10–12, 14

laplacemix, 3, 5, 10, 12, 14

mixsqp, 4, 11

nlm, 3, 11

normalmix, 3, 5, 10, 12

optimize, 3, 11

output_all (ebnm), 2

output_default (ebnm), 2

plot.ebnm, 5, 15

print.ebnm, 16

unimix, 3, 5, 10, 12