Package: smashr (via r-universe)

May 15, 2026

Encoding UTF-8

Type Package

Maintainer Peter Carbonetto <pcarbo@uchicago.edu>

Title Smoothing by Adaptive Shrinkage

Version 1.3-12

Date 2025-12-09

Description Fast, wavelet-based Empirical Bayes shrinkage methods for signal denoising, including smoothing Poisson-distributed data and Gaussian-distributed data with possibly heteroskedastic error. The algorithms implement the methods described Z. Xing, P. Carbonetto & M. Stephens (2021) <https://jmlr.org/papers/v22/19-042.html>.

License GPL (>= 3)

Copyright file COPYRIGHTS

Depends R (>= 3.1.1),

Imports utils, stats, data.table, caTools, wavethresh, ashr, Rcpp (>= 1.1.0)

Suggests knitr, rmarkdown, MASS, EbayesThresh, testthat

LinkingTo Rcpp

NeedsCompilation yes

LazyData true

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

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

VignetteBuilder knitr

RoxygenNote 7.3.1

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

Date/Publication 2025-12-09 16:43:52 UTC

RemoteUrl https://github.com/stephenslab/smashr

RemoteRef HEAD

RemoteSha e06efa41a3a523833fc0ee743a34cb02da0f26a7

Contents

glm.approx . . . 2
reflect . . . 3
reverse.pwave . . . 4
sd_estimate_gasser_etal . . . 5
smash . . . 5
smash.gaus . . . 8
smash.poiss . . . 10
smashr . . . 11
ti.thresh . . . 12
treas . . . 14

Index **15**

`glm.approx` *Model fitting using weighted least squares or a GLM approach.*

Description

Fit the model specified in documentation, using either a weighted least squares approach or a generalized linear model approach, with some modifications. This function fits many "simple" logistic regressions (ie zero or one covariate) simultaneously, allowing for the possibility of small sample sizes with low or zero counts. In addition, an alternative model in the form of a weighted least squares regression can also be fit in place of a logistic regression.

Usage

```
glm.approx(
  x,
  g = NULL,
  minobs = 1,
  pseudocounts = 0.5,
  all = FALSE,
  eps = 1e-08,
  center = FALSE,
  repara = FALSE,
  forcebin = FALSE,
  lm.approx = FALSE,
  disp = c("add", "mult"),
  bound = 0.02
)
```

Arguments

x	A matrix of N (# of samples) by 2*B (B: # of WCs or, more precisely, of different scales and locations in multi-scale space); Two consecutive columns correspond to a particular scale and location; The first column (the second column) contains

	# of successes (# of failures) for each sample at the corresponding scale and location.
g	A vector of covariate values. Can be a factor (2 groups only) or quantitative. For a 2-group categorical covariate, provide g as a 0-1 factor instead of a 0-1 numeric vector for faster computation.
minobs	Minimum number of non-zero required for each model to be fitted (otherwise NA is returned for that model).
pseudocounts	A number to be added to counts when counts are zero (or possibly extremely small).
all	Bool, if TRUE pseudocounts are added to all entries, if FALSE (default) pseudocounts are added only to cases when either number of successes or number of failures (but not both) is 0.
eps	Small positive number added to counts to improve numerical stability of the computations.
center	Bool, indicating whether to center g. If g is a 2-group categorical variable and centering is desired, use center=TRUE instead of treating g as numeric and centering manually to avoid slower computation.
repara	Bool, indicating whether to reparameterize alpha and beta so that their likelihoods can be factorized.
forcebin	Bool, if TRUE don't allow for overdipersion. Defaults to TRUE if nsig=1, and FALSE otherwise.
lm.approx	Bool, indicating whether a WLS alternative should be used. Defaults to FALSE.
disp	A string, can be either "add" or "mult", indicating the form of overdispersion assumed when lm.approx=TRUE.
bound	Numeric, indicates the threshold of the success vs failure ratio below which pseudocounts will be added.

Value

A matrix of 2 (or 5 if g is provided) by T (# of WCs); Each row contains alphahat (1st row), standard error of alphahat (2nd), betahat (3rd), standard error of betahat (4th), covariance between alphahat and betahat (5th) for each WC.

`reflect` *Reflect and extend a vector.*

Description

Extends the vector to have length a power of 2 (if not already a power of 2) and then reflects it about its right end.

Usage

```
reflect(x)
```

Arguments

x	An n-vector.

Details

The vector x is first reflected about both its left and right ends, by (roughly) the same amount each end, to make its length a power of 2 (if the length of x is already a power of 2 this step is skipped). Then the resulting vector is reflected about its right end to create a vector that is both circular (left and right ends are the same) and a power of 2.

Value

A list with two list elements: "x" containing the extended-and-reflected signal; and "idx" containing the indices of the original signal.

---

reverse.pwave *Reverse wavelet transform a set of probabilities in TItable format for Poisson data.*

Description

Reverse wavelet transform a set of probabilities in TItable format for Poisson data.

Usage

```
reverse.pwave(est, lp, lq = NULL)
```

Arguments

est	An n-vector. Usually a constant vector with each element equal to the estimated log(total rate).
lp	A J by n matrix of probabilities.
lq	A J by n matrix of complementary probabilities.

Value

Reconstructed signal in the original data space.

sd_estimate_gasser_etal

Estimate homoskedastic standard deviation from nonparamatric regression.

Description

Estimate homoskedastic standard deviation from nonparamatric regression.

Usage

```
sd_estimate_gasser_etal(x)
```

Arguments

x	The data.

Details

Uses formula (3) from Brown and Levine (2007), Annals of Statistics, who attribute it to Gasser et al.

Value

An estimate of the standard deviation

smash

Estimate the underlying mean or intensity function from Gaussian or Poisson data, respectively.

Description

This is a wrapper function for smash.gaus or smash.poiss as appropriate. For details see smash.gaus and smash.poiss.

Usage

```
smash(x, model = NULL, ...)
```

Arguments

x	A vector of observations (assumed equally spaced).
model	Specifies the model (Gaussian or Poisson). Can be NULL, in which case the Poisson model is assumed if x consists of integers, and the Gaussian model is assumed otherwise. One of 'gaus' or 'poiss' can also be specified to fit a specific model.
...	Additional arguments passed to smash.gaus or smash.poiss.

Details

Performs nonparametric regression on univariate Poisson or Gaussian data using wavelets. Unlike many wavelet methods the data do not need to have length that is a power of 2 or be cyclic – the functions internally deal with these issues. For the Poisson case, the data are assumed to be i.i.d. from an underlying inhomogeneous mean function that is "smooth". Similarly for the Gaussian case, the data are assumed to be independent with an underlying smooth mean function. In the Gaussian case, the variances are allowed vary, but are similarly "spatially structured" as with the mean function. The functions `smash.gaus` and `smash.pois` perform smoothing for Gaussian and Poisson data respectively.

Value

See `smash.gaus` or `smash.poiss` for details.

Examples

```
# First Gaussian example
# ----------------------
# In this first example, the length of the signal is a power of 2.
#
# Create the baseline mean function. (The "spikes" function is used
# as an example here.)
set.seed(2)
n <- 2^9
t <- 1:n/n
spike.f <- function (x) (0.75 * exp(-500 * (x - 0.23)^2) +
  1.5 * exp(-2000 * (x - 0.33)^2) + 3 * exp(-8000 * (x - 0.47)^2) +
  2.25 * exp(-16000 * (x - 0.69)^2) + 0.5 * exp(-32000 * (x - 0.83)^2))
mu.s <- spike.f(t)

# Scale the signal to be between 0.2 and 0.8
mu.t <- (1 + mu.s)/5
plot(mu.t,type = "l")

# Create the baseline variance function. (The function V2 from Cai &
# Wang (2008) is used here.)
var.fn <- (1e-04 + 4 * (exp(-550 * (t - 0.2)^2) +
                        exp(-200 * (t - 0.5)^2) +
                        exp(-950 * (t - 0.8)^2)))/1.35
plot(var.fn,type = "l")

# Set the signal-to-noise ratio.
rsnr    <- sqrt(5)
sigma.t <- sqrt(var.fn)/mean(sqrt(var.fn)) * sd(mu.t)/rsnr^2

# Simulate an example dataset.
X.s <- rnorm(n,mu.t,sigma.t)

# Run smash (Gaussian version is run since observations are not
# counts).
mu.est <- smash(X.s)
```

```
# Plot the true mean function as well as the estimated one.
plot(mu.t,type = "l")
lines(mu.est,col = 2)

# First Poisson example
# ---------------------
# Scale the signal to be non-zero and to have a low average intensity.
mu.t <- 0.01 + mu.s

# Simulate an example dataset.
X.s <- rpois(n,mu.t)

# Run smash (the Poisson version is run since observations are counts).
mu.est <- smash(X.s)

# Plot the true mean function as well as the estimated one.
plot(mu.t,type = "l")
lines(mu.est,col = 2)

# Second Gaussian example
# -----------------------
# In this second example, we demonstrate that smash also works even
# when the signal length (n) is not a power of 2.
n    <- 1000
t    <- 1:n/n
mu.s <- spike.f(t)

# Scale the signal to be between 0.2 and 0.8.
mu.t <- (1 + mu.s)/5

# Create the baseline variance function.
var.fn  <- (1e-04 + 4 * (exp(-550 * (t - 0.2)^2) +
                         exp(-200 * (t - 0.5)^2) +
                         exp(-950 * (t - 0.8)^2)))/1.35
sigma.t <- sqrt(var.fn)/mean(sqrt(var.fn)) * sd(mu.t)/rsnr^2

# Simulate an example dataset.
X.s <- rnorm(n,mu.t,sigma.t)

# Run smash.
mu.est <- smash(X.s)

# Plot the true mean function as well as the estimated one.
plot(mu.t,type = "l")
lines(mu.est,col = 2)

# Second Poisson example
# ----------------------
# The Poisson version of smash also works with signals that are not
# exactly of length 2^J for some integer J.
#
# Scale the signal to be non-zero and to have a low average intensity.
```

```
mu.t <- 0.01 + mu.s

# Simulate an example dataset
X.s <- rpois(n,mu.t)

# Run smash (Poisson version is run since observations are counts).
mu.est <- smash(X.s)

# Plot the true mean function as well as the estimated one.
plot(mu.t,type = "l")
lines(mu.est,col = 2)
```

---

smash.gaus *Estimate underlying mean function from noisy Gaussian data.*

Description

This function performs non-parametric regression (signal denoising) using Empirical Bayes wavelet-based methods.

Usage

```
smash.gaus(
  x,
  sigma = NULL,
  v.est = FALSE,
  joint = FALSE,
  v.basis = FALSE,
  post.var = FALSE,
  filter.number = 1,
  family = "DaubExPhase",
  return.loglr = FALSE,
  jash = FALSE,
  SGD = TRUE,
  weight = 0.5,
  min.var = 1e-08,
  ashparam = list(),
  homoskedastic = FALSE,
  reflect = FALSE
)
```

Arguments

x	A vector of observations. Reflection is done automatically if length of x is not a power of 2.
sigma	A vector of standard deviations. Can be provided if known or estimated beforehand.

v.est	Boolean indicating if variance estimation should be performed instead.
joint	Boolean indicating if results of mean and variance estimation should be returned together.
v.basis	Boolean indicating if the same wavelet basis should be used for variance estimation as mean estimation. If false, defaults to Haar basis for variance estimation (this is much faster than other bases).
post.var	Boolean indicating if the posterior variance should be returned for the mean and/or variance estiamtes.
filter.number	Choice of wavelet basis to be used, as in wavethresh.
family	Choice of wavelet basis to be used, as in wavethresh.
return.loglr	Boolean indicating if a logLR should be returned.
jash	Indicates if the prior from method JASH should be used. This will often provide slightly better variance estimates (especially for nonsmooth variance functions), at the cost of computational efficiency. Defaults to FALSE.
SGD	Boolean indicating if stochastic gradient descent should be used in the EM. Only applicable if jash=TRUE.
weight	Optional parameter used in estimating overall variance. Only works for Haar basis. Defaults to 0.5. Setting this to 1 might improve variance estimation slightly.
min.var	The minimum positive value to be set if the variance estimates are non-positive.
ashparam	A list of parameters to be passed to ash; default values are set by function setAshParam.gaus.
homoskedastic	indicates whether to assume constant variance (if v.est is true)
reflect	A logical indicating if the signals should be reflected.

Details

We assume that the data come from the model $Y_t = \mu_t + \epsilon_t$ for $t = 1, ..., T$, where μ_t is an underlying mean, assumed to be spatially structured (or treated as points sampled from a smooth continous function), and $\epsilon_t \sim N(0, \sigma_t)$, and are independent. Smash provides estimates of μ_t and σ_t^2 (and their posterior variances if desired).

Value

By default smash.gaus simply returns a vector of estimated means. However, if more outputs are requested (eg if return.loglr or v.est is TRUE) then the output is a list with one or more of the following elements:

mu.res	A list with the mean estimate, its posterior variance if post.var is TRUE, the logLR if return.loglr is TRUE, or a vector of mean estimates if neither post.var nor return.loglr are TRUE.

If v.est is TRUE, then smash.gaus returns the following:

var.res	A list with the variance estimate, its posterior variance if post.var is TRUE, or a vector of variance estimates if post.var is FALSE In addition, if joint is TRUE, then both mu.res and var.res are returned.

Examples

```
n=2^10
t=1:n/n
spike.f = function(x) (0.75*exp(-500*(x-0.23)^2) +
  1.5*exp(-2000*(x-0.33)^2) + 3*exp(-8000*(x-0.47)^2) +
  2.25*exp(-16000*(x-0.69)^2)+0.5*exp(-32000*(x-0.83)^2))
mu.s = spike.f(t)

# Gaussian case
mu.t = (1+mu.s)/5
plot(mu.t,type='l')
var.fn = (0.0001 + 4*(exp(-550*(t-0.2)^2) + exp(-200*(t-0.5)^2) +
  exp(-950*(t-0.8)^2)))/1.35
plot(var.fn,type='l')
rsnr=sqrt(5)
sigma.t=sqrt(var.fn)/mean(sqrt(var.fn))*sd(mu.t)/rsnr^2
X.s=rnorm(n,mu.t,sigma.t)
mu.est=smash.gaus(X.s)
plot(mu.t,type='l')
lines(mu.est,col=2)
```

`smash.poiss` *Estimate the underlying intensity for Poisson counts.*

Description

Main smoothing procedure for Poisson data. Takes a univariate inhomogeneous Poisson process and estimates its mean intensity.

Usage

```
smash.poiss(
  x,
  post.var = FALSE,
  log = FALSE,
  reflect = FALSE,
  glm.approx.param = list(),
  ashparam = list(),
  cxx = TRUE,
  lev = 0
)
```

Arguments

`x`	A vector of Poisson counts (reflection is done automatically if length of `x` is not a power of 2).
`post.var`	Boolean, indicates if the posterior variance should be returned.

log	bool, determines if smoothed signal is returned on log scale or not
reflect	A logical indicating if the signals should be reflected.
glm.approx.param	A list of parameters to be passed to glm.approx; default values are set by function setGlmApproxParam.
ashparam	A list of parameters to be passed to ash; default values are set by function setAshParam.poiss.
cxx	bool, indicates if C++ code should be used to create TI tables.
lev	integer from 0 to J-1, indicating primary level of resolution. Should NOT be used (ie shrinkage is performed at all resolutions) unless there is good reason to do so.

Details

We assume that the data come from the model $Y_t \sim Pois(\mu_t)$ for $t = 1, ..., T$, where μ_t is the underlying intensity, assumed to be spatially structured (or treated as points sampled from a smooth continous function). The Y_t are assumed to be independent. Smash provides estimates of μ_t (and its posterior variance if desired).

Value

smash.poiss returns the mean estimate by default, with the posterior variance as an additional component if post.var is TRUE.

Examples

```
n=2^10
t=1:n/n
spike.f = function(x) (0.75*exp(-500*(x-0.23)^2) +
  1.5*exp(-2000*(x-0.33)^2) + 3*exp(-8000*(x-0.47)^2) +
  2.25*exp(-16000*(x-0.69)^2)+0.5*exp(-32000*(x-0.83)^2))
mu.s=spike.f(t)
mu.t=0.01+mu.s
X.s=rpois(n,mu.t)
mu.est=smash.poiss(X.s)
plot(mu.t,type='l')
lines(mu.est,col=2)
```

Description

This package performs nonparametric regression on univariate Poisson or Gaussian data using multi-scale methods. For the Poisson case, the data x is a vector, with $x_j \sim Poi(\mu_j)$ where the mean vector μ is to be estimated. For the Gaussian case, the data x are a vector with $x_j \sim N(\mu_j, \sigma_j^2)$. Where the mean vector μ and variance vector σ^2 are to be estimated. The primary assumption is that μ is spatially structured, so $\mu_j - \mu_{j+1}$ will often be small (that is, roughly, μ is smooth). Also σ is spatially structured in the Gaussian case (or, optionally, σ is constant, not depending on j).

Details

The function `smash` provides a minimal interface to perform simple smoothing. It is actually a wrapper to `smash.gaus` and `smash.poiss` which provide more options for advanced use. The only required input is a vector of length 2^J for some integer J. Other options include the possibility of returning the posterior variances, specifying a wavelet basis (default is Haar, which performs well in general due to the fact that smash uses the translation-invariant transform)

Author(s)

Matthew Stephens and Zhengrong Xing

See Also

Useful links:

- https://github.com/stephenslab/smashr
- Report bugs at https://github.com/stephenslab/smashr/issues

---

`ti.thresh` *TI thresholding with heteroskedastic errors.*

Description

TI thresholding with heteroskedastic errors.

Usage

```
ti.thresh(
  x,
  sigma = NULL,
  method = "smash",
  filter.number = 1,
  family = "DaubExPhase",
  min.level = 3,
  ashparam = list()
)
```

Arguments

x	The data. Should be a vector of length a power of 2.
sigma	The standard deviation function. Can be provided if known or estimated beforehand.
method	The method to estimate the variance function. Can be 'rmad' for running MAD as described in Gao (1997), or 'smash'.
filter.number	The wavelet basis to be used.
family	The wavelet basis to be used.
min.level	The primary resolution level.
ashparam	Passed as the "ashparam" argument in the call to smash.gaus.

Details

The 'rmad' option effectively implements the procedure described in Gao (1997), while the 'smash' option first estimates the variance function using package smash and then performs thresholding given this variance function.

Value

returns a vector of mean estimates

References

Gao, Hong-Ye (1997) Wavelet shrinkage estimates for heteroscedastic regression models. MathSoft, Inc.

Examples

```
n=2^10
t=1:n/n
spike.f = function(x) (0.75*exp(-500*(x-0.23)^2) +
  1.5*exp(-2000*(x-0.33)^2) +
  3*exp(-8000*(x-0.47)^2) +
  2.25*exp(-16000*(x-0.69)^2) +
  0.5*exp(-32000*(x-0.83)^2))
mu.s = spike.f(t)

# Gaussian case
# -------------
mu.t=(1+mu.s)/5
plot(mu.t,type='l')
var.fn = (0.0001+4*(exp(-550*(t-0.2)^2) +
  exp(-200*(t-0.5)^2) + exp(-950*(t-0.8)^2)))/1.35
plot(var.fn,type='l')
rsnr=sqrt(5)
sigma.t=sqrt(var.fn)/mean(sqrt(var.fn))*sd(mu.t)/rsnr^2
X.s=rnorm(n,mu.t,sigma.t)
mu.est.rmad<-ti.thresh(X.s,method='rmad')
mu.est.smash<-ti.thresh(X.s,method='smash')
```

```
plot(mu.t,type='l')
lines(mu.est.rmad,col=2)
lines(mu.est.smash,col=4)
```

`treas` *Three-month treasury bill data.*

Description

Yields of the three-month Treasury Bill from the secondary market rates, on Fridays. The secondary market rates are annualised using a 360-day year of bank interest and are quoted on a discount basis. The data consist of 1735 weekly observations, from 5 January 1962 to 31 March 1995.

Usage

```
data(treas)
```

Format

Data are stored in numeric (floating-point) vector, `treas`, with 1735 entries.

References

J. Fan and Q. Yao (1998). Efficient estimation of conditional variance functions in stochastic regression. *Biometrika* **85**, 645–660.

Examples

```
# See smashr vignette.
```

Index

∗ **datasets**
treas, 14

glm.approx, 2

reflect, 3
reverse.pwave, 4

sd_estimate_gasser_etal, 5
smash, 5, *12*
smash.gaus, *5, 6*, 8, *12, 13*
smash.poiss, *5, 6*, 10, *12*
smashr, 11
smashr-package *(*smashr*)*, 11

ti.thresh, 12
treas, 14