

Package ‘dsp’

October 16, 2026

Type Package

Title Dynamic Shrinkage Process and Change Point Detection

Version 1.7.0

Description Provides efficient Markov chain Monte Carlo (MCMC) algorithms for dynamic shrinkage processes, which extend global-local shrinkage priors to the time series setting by allowing shrinkage to depend on its own past. These priors yield locally adaptive estimates, useful for time series and regression functions with irregular features. The package includes full MCMC implementations for trend filtering using dynamic shrinkage on signal differences, producing locally constant or linear fits with adaptive credible bands. Also included are models with static shrinkage and normal-inverse-Gamma priors for comparison. Additional tools cover dynamic regression with time-varying coefficients and B-spline models with shrinkage on basis differences, allowing for flexible curve-fitting with unequally spaced data. Some support for heteroscedastic errors, outlier detection, and change point estimation. Methods in this package are described in Kowal et al. (2019) <[doi:10.1111/rssb.12325](https://doi.org/10.1111/rssb.12325)>, Wu et al. (2024) <[doi:10.1080/07350015.2024.2362269](https://doi.org/10.1080/07350015.2024.2362269)>, Schafer and Matteson (2024) <[doi:10.1080/00401706.2024.2407316](https://doi.org/10.1080/00401706.2024.2407316)>, and Cho and Matteson (2026) <[doi:10.1080/07350015.2025.2602846](https://doi.org/10.1080/07350015.2025.2602846)>.

License GPL (>= 3)

Depends R (>= 4.1.0)

Imports coda, fda, graphics, grDevices, Matrix, MCMCpack, methods, msm, pgdraw, Rcpp, RcppZiggurat, spam, progress, stats, stochvol, BayesLogit, truncdist, mgcv, purrr, rlang, lifecycle, glue

Suggests ggplot2, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

URL <https://github.com/schafert/dsp>

BugReports <https://github.com/schafert/dsp/issues>

NeedsCompilation yes

Author Daniel R. Kowal [aut, cph],
Haoxuan Wu [aut],
Toryn Schafer [aut, cre] (ORCID:
[<https://orcid.org/0000-0001-5594-7697>](https://orcid.org/0000-0001-5594-7697)),
Jason B. Cho [aut],
David S. Matteson [aut]

Maintainer Toryn Schafer <toryn27@gmail.com>

Repository CRAN

Date/Publication 2026-10-16 02:40:08 UTC

Contents

as.mcmc.dsp	2
dsp_fit	3
dsp_spec	6
is.dsp	7
plot.dsp	8
predict.dsp	11
simRegression	12
simUnivariate	13
summary.dsp	13
Index	15

as.mcmc.dsp	<i>Coerce a fitted dsp model to an mcmc object</i>
-------------	--

Description

Flattens the posterior draws stored in a dsp object into a single matrix of class `mcmc`, so that the diagnostic and plotting functions in **coda** can be applied to the fitted model.

Usage

```
## S3 method for class 'dsp'  
as.mcmc(x, pars, ...)
```

Arguments

- x an object of class dsp; see [dsp](#)
- pars character vector of parameter names to retain; defaults to all parameters named in x\$mcmc_output
- ... Further arguments to be passed to specific methods

Details

Each element of `x$mcmc_output` is stored with the posterior draws in the first dimension, so that a scalar parameter is a vector of length `nsave` and a time-varying parameter is an `nsave` by `T` matrix or an `nsave` by `T` by `J` array. Every element is flattened column by column and the results are bound into one matrix with `nsave` rows. Columns are named by indexing the original parameter, dropping the leading draw index. Column `"mu[3]"` therefore holds the draws of the mean function at time 3, and column `"beta[3, 2]"` holds the draws of the second regression coefficient at time 3.

The deviance information criterion and the effective number of parameters are posterior summaries rather than draws, and are dropped by the coercion. They remain available in `x$DIC`.

The sampler saves every `nskip + 1` iterations after burn-in, so the returned object records a thinning interval of `nskip + 1`, a starting iteration of `nburn + nskip + 1` and a final iteration of `nburn + nsave * (nskip + 1)`.

Value

An object of class `mcmc` with `nsave` rows and one column per scalar element of the retained parameters named by `pars`.

See Also

`dsp`, `summary.dsp()`, `coda::mcmc()`

Examples

```
set.seed(200)
signal <- c(rep(0, 50), rep(10, 50))
y <- signal + rnorm(100)

model_spec <- dsp_spec(family = "gaussian", model = "smoothing", D = 1)
fit <- dsp_fit(y, model_spec = model_spec, nsave = 200, nburn = 200)

draws <- coda::as.mcmc(fit, pars = c("dhs_phi", "dhs_mean"))
summary(draws)
coda::effectiveSize(draws)
```

Description

Wrapper function for fitting models with Dynamic Shrinkage Processes (DSP), including:

- Adaptive Bayesian Changepoint analysis and local Outlier (ABCO),
- Bayesian Trend Filter for Gaussian Data
- Time-varying Regression

- Bayesian Trend Filter with B-spline for irregularly spaced or functional time-series.
- Bayesian Smoothing for Count Data

Method for printing basic information about the MCMC sampling settings for the fitted model

Usage

```
dsp_fit(
  y,
  model_spec,
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  computeDIC = TRUE,
  verbose = TRUE,
  seed = NULL,
  ...
)

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

Arguments

y	a numeric vector of the $T \times 1$ vector of time series observations
model_spec	a list containing model specification generated from dsp_spec() .
nsave	integer scalar (default = 1000); number of MCMC iterations to record
nburn	integer scalar (default = 1000); number of MCMC iterations to discard (burn-in)
nskip	integer scalar (default = 4); number of MCMC iterations to skip between saving iterations, i.e., save every (nskip + 1)th draw
computeDIC	logical; if TRUE (default), compute the deviance information criterion DIC and the effective number of parameters p_d
verbose	logical; should extra information on progress be printed to the console? Defaults to FALSE
seed	optional integer; if supplied, seeds the random number generator for a reproducible fit. Equivalent to calling <code>set.seed()</code> beforehand, and applies to every family and model.
...	currently not used
x	object of class dsp from dsp_fit()

Details

A brief summary of the settings used to fit the model including number of iterations, burn in, and thinning rates.

Value

`dsp_fit` returns an object of class "dsp".

An object of class "dsp" is defined as a list containing at least the following components:

<code>mcmc_output</code>	a list of the <code>nsave</code> MCMC samples for the parameters named in <code>mcmc_params</code>
<code>DIC</code>	Deviance Information Criterion
<code>mcpair</code>	named vector of supplied <code>nsave</code> , <code>nburn</code> , and <code>nskip</code>
<code>model_spec</code>	the object supplied for <code>model_spec</code> argument

MCMC Output Parameters

Shared parameters across wrappers include:

- `mu`: Conditional mean.
- `ypred`: Posterior predictive $\tilde{y}_t$.
- `evol_sigma_t2`: Variance of the state variable.
- `obs_sigma_t2`: Observation variance σ_ϵ^2 (family = "gaussian").
- `dhs_phi`, `dhs_mean`: DHS hyperparameter draws (evol_error = "DHS").

Model-specific parameters include:

- **changepoint (Gaussian):**
 - `omega`: D -th differenced posterior mean draws, μ_t .
 - `zeta`: anomaly component (when `useAnom` = TRUE).
 - `zeta_sigma_t2`: anomaly variance (when `useAnom` = TRUE).
 - `gamma`: threshold parameter in the thresholded DHS log-volatility dynamics.
- **regression (Gaussian):**
 - `beta`: time-varying regression coefficient draws.
- **bspline (Gaussian):**
 - `beta`: B-spline basis coefficient draws.
- **smoothing (Negative Binomial):**
 - `r`: overdispersion parameter draws.

Note

The data `y` may contain NAs, which will be treated with a simple imputation scheme via an additional Gibbs sampling step. In general, rescaling `y` to have unit standard deviation is recommended to avoid numerical issues when family is "gaussian".

Examples

```
set.seed(200)
signal = c(rep(0, 50), rep(10, 50))
noise = rep(1, 100)
noise_var = rep(1, 100)
for (k in 2:100){
  noise_var[k] = exp(0.9*log(noise_var[k-1]) + rnorm(1, 0, 0.5))
  noise[k] = rnorm(1, 0, sqrt(noise_var[k])) }

y = signal + noise
model_spec = dsp_spec(family = "gaussian", model = "changepoint",
                      D = 1, useAnom = TRUE, obsSV = "SV")
mcmc_output = dsp_fit(y, model_spec = model_spec, nsave = 200, nburn = 200)

print(mcmc_output)
```

dsp_spec

Model Specification

Description

Method for creating dsp specification object prior to fitting.

Method for printing basic information about the model specification

Usage

```
dsp_spec(family, model, ...)

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

Arguments

- | | |
|--------|--|
| family | A character string specifying the model family. Must be one of: <ul style="list-style-type: none"> • "gaussian": Gaussian family. • "negbinomial": Negative binomial family. |
| model | A character string specifying the model type: <ul style="list-style-type: none"> • family = "gaussian": <ul style="list-style-type: none"> – "changepoint": Change point detection with Adaptive Bayesian Change-point analysis and local Outlier (ABCO), – "smoothing": Bayesian smoothing, – "regression": Time-varying regression, – "bspline": Bayesian smoothing with B-spline for irregularly spaced or functional time-series. • family = "negbinomial": |

- "smoothing": Bayesian smoothing.
- ... currently not used
- x object of class dsp_spec from [dsp_spec\(\)](#)

Value

A list containing the model specification.

Examples

```
model_spec <- dsp_spec(family = "gaussian",
                      model = "smoothing")

print(model_spec)
```

is.dsp

*Dynamic Shrinkage Process (dsp) Object***Description**

Check whether an object is a fitted dsp model.

Usage

```
is.dsp(object)
```

Arguments

object any R object

Details

A dsp object is a classified list created by [dsp_fit](#). It contains `mcmc_output`, a list of posterior draws whose matrix elements are `nsave` by `T` and whose vector elements are of length `nsave`; `DIC`, the deviance information criterion and effective number of parameters; `mcpair`, the number of saved draws, the burn-in and the thinning interval; and `model_spec`, the `dsp_spec` object describing the fitted model.

Value

Logical, whether object inherits from class dsp.

See Also

[as.mcmc.dsp\(\)](#), [summary.dsp\(\)](#)

plot.dsp

*Plot posterior summaries from a fitted dsp model***Description**

Visualize posterior samples for a selected parameter from a fitted dsp object. The parameter to plot is chosen with `type`, which must name one of the entries in `x$mcmc_output` and defaults to "mu". Depending on the dimension of the selected posterior samples, the function produces either a posterior density plot, a time-series plot with credible intervals, or a multi-panel time-series plot.

Usage

```
## S3 method for class 'dsp'
plot(
  x,
  type = "mu",
  true_values = NULL,
  times = NULL,
  y_obs = NULL,
  include_joint_bands = FALSE,
  alpha = 0.05,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  xlim = NULL,
  ylim = NULL,
  mar = NULL,
  par_args = list(),
  legend = TRUE,
  legend_cex = 1,
  legend_pt_cex = 2,
  nr = NULL,
  nc = NULL,
  cp_thres = 0.5,
  ...
)
```

Arguments

<code>x</code>	an object of class <code>dsp</code> from <code>dsp_fit()</code> .
<code>type</code>	character string giving the parameter name to visualize; must be one of the entries in <code>x\$mcmc_output</code> . Defaults to "mu".
<code>true_values</code>	optional ground-truth values to overlay on the plot. For scalar parameters, this should be a length-1 numeric value; for time-varying parameters, a $T \times 1$ vector; and for multi-parameter time-varying quantities, a $T \times p$ matrix matching the plotted parameter dimensions.

times	optional vector of observation points. If NULL, the function assumes T equally spaced points on $[0, 1]$.
y_obs	optional vector of observed data point of length T. Only for 2-dimensional parameters.
include_joint_bands	logical; if TRUE, include simultaneous credible bands in addition to pointwise credible intervals when available. Joint bands are currently supported only for time-varying parameters: zeta, omega, ypred, and mu for 2D outputs, and zeta, omega, ypred, mu, and beta for 3D outputs.
alpha	numeric credibility level used to construct posterior intervals and bands. Defaults to 0.05, corresponding to 95% intervals/bands.
xlab	optional x-axis label. If NULL, defaults to "t".
ylab	optional y-axis label. If NULL, defaults to type.
main	optional plot title. For multi-panel plots, this may be either a single title or a vector of titles of length equal to the number of panels.
xlim	optional x-axis limits passed to the plotting routine.
ylim	optional y-axis limits passed to the plotting routine.
mar	optional numeric vector of length 4 giving plot margins, passed to <code>graphics::par()</code> .
par_args	optional named list of additional graphical parameters passed to <code>graphics::par()</code> , such as <code>cex.axis</code> , <code>cex.lab</code> , <code>cex.main</code> , <code>las</code> , or <code>mgp</code> .
legend	logical; if TRUE, add a legend to the plot.
legend_cex	numeric scaling factor for legend text size.
legend_pt_cex	numeric scaling factor for legend symbol size.
nr	optional number of rows in the plotting layout for multi-panel (3-dimensional) parameters.
nc	optional number of columns in the plotting layout for multi-panel (3-dimensional) parameters.
cp_thres	(default 0.5) cutoff proportion for percentage of posterior samples exceeding the threshold needed to label a changepoint
...	additional graphical arguments passed to the main plotting call, such as <code>graphics::plot()</code> or <code>graphics::hist()</code> depending on the parameter dimension.

Details

The plotting behavior depends on the dimension of the posterior samples stored in `x$mcmc_output[[type]]`:

- **1D (scalar parameter):** A density-style summary is produced using a histogram with an overlaid kernel density estimate. The posterior mean and equal-tailed $(1 - \alpha)100\%$ credible interval are marked, and the true value is added if supplied through `true_values`.
- **2D (time-varying parameter):** A time-series plot is produced showing the posterior mean together with equal-tailed pointwise credible intervals. If `include_joint_bands = TRUE` and the parameter is one of `zeta`, `omega`, `ypred`, or `mu`, simultaneous credible bands are also displayed. If `true_values` is provided, the ground truth is overlaid.

- **3D (multi-parameter time-varying quantity):** A multi-panel collection of time-series plots is produced, one panel for each slice of the third dimension. Each panel shows the posterior mean, pointwise credible intervals, optional joint bands when supported, and optional ground-truth values. The panel layout is controlled by `nr` and `nc`; if omitted, a near-square layout is chosen automatically.

Axis labels, titles, plotting limits, margins, legend display, and additional graphical settings can be customized through `xlab`, `ylab`, `main`, `xlim`, `ylim`, `mar`, `legend`, `legend_cex`, `legend_pt_cex`, and `par_args`.

The x-axis values are taken from `times`. If `times` is not supplied, evenly spaced points on $[0, 1]$ are used. For differenced variance parameters such as `"evol_sigma_t2"` and `"zeta_sigma_t2"`, the initial time points associated with prior initialization are automatically removed before plotting.

For fitted changepoint models, changepoint annotations may be added when supported by the plotted parameter and the corresponding latent components are present in the MCMC output.

Value

No return value, called for side effects

References

Krivobokova, T., Kneib, T., and Claeskens, G. (2010). Simultaneous confidence bands for penalized spline estimators. *Journal of the American Statistical Association*, **105**(490), 852–863. doi:10.1198/jasa.2010.tm09165

Examples

```
set.seed(200)
signal = c(rep(0, 50), rep(10, 50))
noise = rep(1, 100)
noise_var = rep(1, 100)
for (k in 2:100){
  noise_var[k] = exp(0.9*log(noise_var[k-1]) + rnorm(1, 0, 0.5))
  noise[k] = rnorm(1, 0, sqrt(noise_var[k])) }

y = signal + noise
model_spec = dsp_spec(family = "gaussian", model = "changepoint",
                      D = 1, useAnom = TRUE, obsSV = "SV")
mcmc_output = dsp_fit(y, model_spec = model_spec, nsave = 200, nburn = 200)
# Estimated posterior mean vs ground truth
plot(mcmc_output, type = "mu", true_values = signal)
# Estimated innovation variance vs ground truth for illustration only
plot(mcmc_output, type = "obs_sigma_t2", true_values = noise^2)
```

predict.dsp

*Predict changepoints from the output of ABCO***Description**

Predict changepoints from the output of ABCO

Usage

```
## S3 method for class 'dsp'
predict(object, cp_thres = 0.5, cp_prop = FALSE, ...)
```

Arguments

object	object of class dsp from dsp_fit()
cp_thres	(default 0.5) cutoff proportion for percentage of posterior samples exceeding the threshold needed to label a changepoint
cp_prop	(default FALSE) logical flag determining if the posterior proportions of threshold exceedance is to be returned.
...	currently unused

Details

The changepoint model uses a thresholding mechanism with a latent indicator variable. This function calculates the proportion of samples where the increment exceeds the threshold.

Value

If `cp_prop = FALSE`, an integer vector containing the indices of detected changepoints in the observed data. If no changepoints are detected, an empty integer vector is returned.

If `cp_prop = TRUE`, a list containing: - `'cp_t'`: an integer vector containing the indices of detected changepoints. - `'cp_prop'`: a numeric vector of length $T - D$ containing the pointwise posterior proportions of threshold exceedance.

Examples

```
set.seed(200)
signal = c(rep(0, 50), rep(10, 50))
noise = rep(1, 100)
noise_var = rep(1, 100)
for (k in 2:100){
  noise_var[k] = exp(0.9*log(noise_var[k-1]) + rnorm(1, 0, 0.5))
  noise[k] = rnorm(1, 0, sqrt(noise_var[k])) }

y = signal + noise
model_spec = dsp_spec(family = "gaussian", model = "changepoint",
                      D = 1, useAnom = TRUE)
```

```
mcmc_output = dsp_fit(y, model_spec = model_spec, nsave = 200, nburn = 200)
predict(mcmc_output)
```

simRegression

Simulate noisy observations from a dynamic regression model

Description

Simulates data from a time series regression with dynamic regression coefficients. The dynamic regression coefficients are simulated as a Gaussian random walk, where jumps occur with a pre-specified probability sparsity. The coefficients are initialized by a $N(0,1)$ simulation.

Usage

```
simRegression(
  nT = 200,
  p = 20,
  p_0 = 15,
  sparsity = 0.05,
  RSNR = 5,
  ar1 = 0,
  include_plot = FALSE
)
```

Arguments

nT	number of time points
p	number of predictors (total)
p_0	number of true zero regression terms
sparsity	the probability of a jump (i.e., a change in the dynamic regression coefficient)
RSNR	root-signal-to-noise ratio
ar1	the AR(1) coefficient for the predictors X; default is zero for iid $N(0,1)$ predictors
include_plot	logical; if TRUE, include a plot of the simulated data and the true curve

Value

a list containing

- the simulated function y
- the simulated predictors X
- the simulated dynamic regression coefficients beta_true
- the true function mu_true
- the true observation standard deviation sigma_true

Note

The root-signal-to-noise ratio is defined as $RSNR = (sd \text{ of true function}) / (sd \text{ of noise})$.

simUnivariate	<i>Generate univariate signals of different type</i>
---------------	--

Description

Using code from the archived wmtsa package

Usage

```
simUnivariate(name, n = 1024, snr = Inf)
```

Arguments

- name character string of name of the test wavelet signal to be generated; one of "dirac", "kronecker", "heavisine", "bumps", "blocks", "doppler", "ramp", "cusp", "crease", "sing", "hisine", "losine", "linchirp", "twochirp", "quadchirp", "mishmash1", "mishmash2", "mishmash3", "levelshift", "jumpsine", "gauss", "patches", "linear", "quadratic", "cubic";
- n length of the series; defaults to 1024 points; increasing n infills the time series
- snr desired signal-to-noise ratio; default Inf corresponds to 0 noise

Value

A numeric vector the same length as n.

Examples

```
nms <- c("blocks", "linchirp", "mishmash1", "bumps")
z <- lapply(nms, simUnivariate)
```

summary.dsp	<i>Summarize DSP MCMC chains</i>
-------------	----------------------------------

Description

Summarize DSP MCMC chains

Usage

```
## S3 method for class 'dsp'
summary(object, pars, probs = c(0.025, 0.25, 0.5, 0.75, 0.975), ...)
```

Arguments

object	object of class dsp from <code>dsp_fit()</code>
pars	parameter names specified for summaries; currently defaults to all parameters named in <code>object\$mcmc_output</code>
probs	numeric vector of <code>quantile()</code> s requested for posterior summary of pars. Defaults to <code>c(0.025, 0.25, 0.50, 0.75, 0.975)</code>
...	currently not being used

Value

Returns a named list of the same length as `pars` where within each element of the list is a numeric matrix (vector parameters) or vector (scalar parameters). For matrices, each row is a time point (or dimension) of the parameter and each column is a named summary. The names are accessible with `colnames`. For vectors (scalar parameters), each element is a named summary.

Examples

```
set.seed(200)
signal = c(rep(0, 50), rep(10, 50))
noise = rep(1, 100)
noise_var = rep(1, 100)
for (k in 2:100){
  noise_var[k] = exp(0.9*log(noise_var[k-1]) + rnorm(1, 0, 0.5))
  noise[k] = rnorm(1, 0, sqrt(noise_var[k])) }

y = signal + noise
model_spec = dsp_spec(family = "gaussian", model = "changepoint",
                      D = 1, useAnom = TRUE, obsSV = "SV")
mcmc_output = dsp_fit(y, model_spec = model_spec, nsave = 200, nburn = 200)

summary_fit <- summary(mcmc_output)
summary_fit$mu[, "mean"]
summary_fit$evol_sigma_t2[, "mean"]
```

Index

`as.mcmc.dsp`, [2](#)
`as.mcmc.dsp()`, [7](#)

`coda::mcmc()`, [3](#)

`dsp`, [2](#), [3](#)
`dsp-class (is.dsp)`, [7](#)
`dsp_fit`, [3](#), [7](#)
`dsp_fit()`, [4](#), [8](#), [11](#), [14](#)
`dsp_spec`, [6](#)
`dsp_spec()`, [4](#), [7](#)

`graphics::hist()`, [9](#)
`graphics::par()`, [9](#)
`graphics::plot()`, [9](#)

`is.dsp`, [7](#)

`mcmc`, [2](#), [3](#)

`plot.dsp`, [8](#)
`predict.dsp`, [11](#)
`print.dsp (dsp_fit)`, [3](#)
`print.dsp_spec (dsp_spec)`, [6](#)

`quantile()`, [14](#)

`simRegression`, [12](#)
`simUnivariate`, [13](#)
`summary.dsp`, [13](#)
`summary.dsp()`, [3](#), [7](#)