Package: dsp (via r-universe)

October 31, 2026

Type Package

Title Dynamic Shrinkage Process and Change Point Detection

Version 1.8.0

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>, Wu et al. (2024) <doi:10.1080/07350015.2024.2362269>, Schafer and Matteson (2024) <doi:10.1080/00401706.2024.2407316>, and Cho and Matteson (2026) <doi: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

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

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

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

Config/pak/sysreqs libgmp3-dev libssl-dev

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

Date/Publication 2026-10-30 22:09:30 UTC

RemoteUrl https://github.com/schafert/dsp

RemoteRef HEAD

RemoteSha 0bd18067e03c2f5bc84f50ee606e0bd889d8a7f3

Contents

as.mcmc.dsp . . . 2
credible_bands . . . 4
dsp-accessors . . . 6
dsp_fit . . . 7
dsp_spec . . . 10
is.dsp . . . 11
plot.dsp . . . 12
predict.dsp . . . 15
print.summary.dsp . . . 16
simRegression . . . 16
simUnivariate . . . 18
summary.dsp . . . 18

Index **21**

`as.mcmc.dsp` *Coerce a fitted dsp model to an mcmc object*

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

credible_bands *Simultaneous credible bands and band scores*

Description

Computes simultaneous credible bands for a time-varying parameter, and the simultaneous band scores (SimBaS) that test, at each time point, whether the parameter is zero.

Usage

```
credible_bands(object, pars = "mu", level = 0.95, times = NULL, ...)

## S3 method for class 'dsp_bands'
print(x, digits = 2, max_periods = 10, ...)
```

Arguments

object	an object of class dsp; see dsp
pars	character string naming one time-varying parameter; see par_names(). Defaults to "mu".
level	the credibility level of the bands. Defaults to 0.95.
times	optional vector of length T labelling the time points of the series, such as dates. Defaults to 1:T.
...	currently not used
x	an object of class dsp_bands
digits	number of significant digits for the global Bayesian p-value
max_periods	the largest number of periods to list per band

Details

The bands of Crainiceanu et al. (2007) have the form posterior mean plus or minus a multiple of the pointwise posterior standard deviation. The multiple is the level quantile, over the posterior draws, of the largest standardized absolute deviation from the posterior mean across all time points. The bands therefore contain the whole trajectory in a proportion level of the posterior draws, whereas the pointwise intervals from fitted.dsp() are computed at each time point separately.

The SimBaS of Meyer et al. (2015) at time t is the smallest α for which the $1-\alpha$ simultaneous band excludes zero at t. A small value is evidence that the parameter is nonzero at t, adjusted for testing all time points at once. The minimum over time, min(bands[, "simBaS"], na.rm = TRUE), is the global Bayesian p-value for the hypothesis that the parameter is zero at every time point.

The scores are informative only for a parameter for which zero is a value of interest. Examples are the increments omega, where a small score marks a time at which the mean function changes (for D = 1) or its slope changes (for D = 2), and the regression coefficients beta, where a small score marks a time at which the predictor has an effect. For the mean function mu of a series far from zero, every score is close to zero and carries no information. The changepoint model stores omega by default.

For omega, the test is distinct from the changepoint rule of predict.dsp(), which compares the increments with the threshold of the changepoint model.

Time points with no posterior variation, such as fixed initial values, are excluded when the bands are calibrated. At those points both bands equal the constant value and the score is NA.

Value

An object of class dsp_bands, a data frame with one row per time point and columns time, mean, lower, upper and simBaS. For a parameter with one band per predictor, such as the regression coefficients beta, a column predictor follows time. The time column is taken from times. The increments omega have T - D entries, which are labelled with the last T - D time points, so that each row refers to the time at which the increment ends. The print method reports the global Bayesian p-value and the periods in which the band excludes zero, by predictor. Subsetting with [returns a plain data frame.

References

Crainiceanu, C. M., Ruppert, D., Carroll, R. J., Joshi, A. and Goodner, B. (2007). Spatially adaptive Bayesian penalized splines with heteroscedastic errors. *Journal of Computational and Graphical Statistics*, 16(2), 265–288. doi:10.1198/106186007X208768

Meyer, M. J., Coull, B. A., Versace, F., Cinciripini, P. and Morris, J. S. (2015). Bayesian function-on-function regression for multilevel functional data. *Biometrics*, 71(3), 563–574. doi:10.1111/biom.12299

See Also

fitted.dsp() for pointwise intervals, plot.dsp() with include_joint_bands = TRUE to display the bands

Examples

```
set.seed(200)
y <- c(rep(0, 50), rep(10, 50)) + rnorm(100)
model_spec <- dsp_spec(family = "gaussian", model = "changepoint", D = 1)
fit <- dsp_fit(y, model_spec = model_spec, nsave = 200, nburn = 200)

# global Bayesian p-value and the periods in which the band excludes zero
bands <- credible_bands(fit, pars = "omega")
bands

# the pointwise values
head(as.data.frame(bands))

# time points at which the increment is nonzero at the joint 5% level
subset(bands, simBaS < 0.05)$time
```

dsp-accessors *Accessor functions for fitted dsp models*

Description

Extract information from a fitted dsp object without referring to its internal structure.

Usage

```
par_names(object)

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

DIC(object, ...)

## S3 method for class 'dsp'
DIC(object, ...)

get_spec(object)

## S3 method for class 'dsp'
nobs(object, ...)
```

Arguments

object	an object of class dsp; see dsp
pars	character string naming one parameter; see par_names() for the parameters available in object. Defaults to "mu", the mean function, which every model returns.
probs	numeric vector of probabilities for the posterior quantiles
...	currently not used

Details

par_names() returns the names of the parameters with stored posterior draws. These are the values accepted by the pars argument of fitted(), summary.dsp() and as.mcmc.dsp(), and by the type argument of plot.dsp().

fitted() returns pointwise posterior summaries of one parameter. For the posterior draws themselves, use as.mcmc.dsp().

DIC() returns the deviance information criterion under the two standard estimators of the effective number of parameters, p_D (Spiegelhalter et al. 2002) and p_V (Gelman et al. 2014). Lower values are preferred. We recommend p_D by default.

get_spec() returns the dsp_spec() object that the model was fitted with.

nobs() returns the number of time points T in the fitted series, including any missing observations, which the model imputes.

Value

`par_names()`: a character vector.

`fitted()`: for a time-varying parameter, a numeric matrix with one row per time point (or index) and columns `"mean"` followed by one column per element of `probs`, named as by `stats::quantile()`. A parameter with a further dimension, such as the regression coefficients `beta`, gives an array whose third dimension holds these summaries. A scalar parameter gives a named vector.

`DIC()`: a data frame with one row per estimator and columns `estimator`, `DIC` and `p_eff`, the effective number of parameters.

`get_spec()`: an object of class `dsp_spec`.

`nobs()`: an integer.

References

Spiegelhalter, D. J., Best, N. G., Carlin, B. P. and van der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B*, 64(4), 583–639.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A. and Rubin, D. B. (2014). *Bayesian Data Analysis*, 3rd edition. CRC Press.

See Also

`summary.dsp()`, `as.mcmc.dsp()`, `plot.dsp()`, `predict.dsp()`

Examples

```
set.seed(200)
y <- c(rep(0, 50), rep(10, 50)) + rnorm(100)
model_spec <- dsp_spec(family = "gaussian", model = "smoothing", D = 1)
fit <- dsp_fit(y, model_spec = model_spec, nsave = 200, nburn = 200)

par_names(fit)
head(fitted(fit))
DIC(fit)
get_spec(fit)
nobs(fit)
```

`dsp_fit` *MCMC Sampler for Models with Dynamic Shrinkage Processes*

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 x 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 set.seed() 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:

`mcmc_output`	a list of the `nsave` MCMC samples for the parameters named in `mcmc_params`
`DIC`	Deviance Information Criterion
`mcpar`	named vector of supplied nsave, nburn, and nskip
`model_spec`	the object supplied for model_spec 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. For $D > 0$, the first D entries are the variances of the initial states.
- `obs_sigma_t2`: Observation variance σ_ϵ^2 (`family = "gaussian"`). One draw per iteration when the variance is constant in time (`obsSV = "const"`, and always for `model = "bspline"`), otherwise one per time point.
- `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`), with T columns aligned with `y`. The anomaly is defined for $t > D$ only and is zero at the first D time points.
  - `zeta_sigma_t2`: variance of the horseshoe+ prior on the anomaly (when `useAnom = TRUE`), zero at the first D time points.
  - `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: • "gaussian": Gaussian family. • "negbinomial": Negative binomial family.
`model`	A character string specifying the model type: • `family` = "gaussian": – "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; mcpar, the number of saved draws, the burn-in and the thinning interval; and model_spec, the dsp_spec object describing the fitted model.

The accessor functions par_names(), fitted.dsp(), DIC(), get_spec() and nobs.dsp() extract these components without referring to the structure of the object.

Value

Logical, whether object inherits from class dsp.

See Also

dsp-accessors, 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 parameters returned by par_names() 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

x	an object of class dsp from dsp_fit().
type	character string giving the parameter name to visualize; must be one of the parameters returned by par_names(). Defaults to "mu".
true_values	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 x 1 vector; and for multi-parameter time-varying quantities, a T x 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 (Crainiceanu et al. 2007; see `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 `graphics::par()`.
`par_args`	optional named list of additional graphical parameters passed to `graphics::par()`, such as `cex.axis`, `cex.lab`, `cex.main`, `las`, or `mgp`.
`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 `graphics::plot()` or `graphics::hist()` 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 `"evol_sigma_t2"`, the first `D` entries, which are the variances of the initial states, are 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

Crainiceanu, C. M., Ruppert, D., Carroll, R. J., Joshi, A., and Goodner, B. (2007). Spatially adaptive Bayesian penalized splines with heteroscedastic errors. *Journal of Computational and Graphical Statistics*, **16**(2), 265–288. doi:10.1198/106186007X208768

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 containing the pointwise posterior proportions of threshold exceedance, aligned with the observed series so that cp_prop[cp_t] gives the proportions at the detected changepoints. The first D entries are NA, because no increment is defined there.

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)

cp <- predict(mcmc_output, cp_prop = TRUE)
cp$cp_prop[cp$cp_t]
```

`print.summary.dsp` *Print a summary of a fitted dsp model*

Description

Print a summary of a fitted dsp model

Usage

```
## S3 method for class 'summary.dsp'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	object of class `summary.dsp` from `summary.dsp()`
`digits`	number of significant digits to print
`...`	currently not used

Value

`x`, invisibly.

`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 of true function)/(sd of noise).

simUnivariate *Generate univariate signals of different type*

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 *Summarize a fitted dsp model*

Description

Produces a compact summary of a fitted model: the model specification, the MCMC settings, the deviance information criterion and one line per parameter.

Usage

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

Arguments

object	object of class dsp from dsp_fit()
pars	parameter names to summarize; defaults to all parameters returned by par_names()
probs	numeric vector of length two giving the probabilities of the posterior interval reported for scalar parameters. Defaults to c(0.025, 0.975)
ess	logical; if TRUE, also report the minimum and median pointwise effective sample size of each time-varying or vector parameter. Defaults to FALSE, because this requires one spectral estimate per time point and takes several seconds for long series.
...	currently not used

Details

Scalar parameters are summarized by their posterior mean, standard deviation, interval and effective sample size. A parameter indexed by time (or by basis function, or by predictor) is summarized in one line by its length and the range of its pointwise posterior means, and, if ess = TRUE, the minimum and median of its pointwise effective sample sizes. A regression coefficient array beta is reported one line per predictor. The first D entries of evol_sigma_t2, the variances of the initial states, are reported on their own line, labelled by their indices, since they have a different prior from the remaining evolution variances.

Effective sample sizes are computed by coda::effectiveSize(). Elements with no posterior variation, such as the fixed initial values of the evolution variances, are excluded from the minimum and median.

Pointwise posterior summaries of a time-varying parameter are returned by fitted.dsp(), and the posterior draws by as.mcmc.dsp(), for which the diagnostic functions in **coda** are available.

Value

An object of class summary.dsp, a list with components

model_spec the dsp_spec() object of the fitted model

mcpar the number of saved draws, the burn-in and nskip

nobs the number of time points

DIC the data frame returned by DIC(), or NULL if the DIC was not computed

scalars a matrix with one row per scalar parameter and columns mean, sd, the two quantiles and ess, or NULL if there are none

vectors a data frame with one row per non-scalar parameter and columns length, mean_min and mean_max, and, if ess = TRUE, ess_min and ess_median; or NULL if there are none

See Also

fitted.dsp(), DIC(), as.mcmc.dsp()

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(mcmc_output)

# pointwise summaries of the mean function
head(fitted(mcmc_output))
```

Index

as.mcmc.dsp, 2
as.mcmc.dsp(), *6, 7*, *11*, *19*

coda::effectiveSize(), *19*
coda::mcmc(), *3*
credible_bands, 4
credible_bands(), *13*

DIC *(*dsp-accessors*)*, 6
DIC(), *3*, *11*, *19*
dsp, *3, 4*, *6*
dsp-accessors, 6, *11*
dsp-class *(*is.dsp*)*, 11
dsp_fit, 7, *11*
dsp_fit(), *8*, *12*, *15*, *19*
dsp_spec, 10
dsp_spec(), *6*, *8*, *11*, *19*

fitted.dsp *(*dsp-accessors*)*, 6
fitted.dsp(), *4, 5*, *11*, *19*

get_spec *(*dsp-accessors*)*, 6
get_spec(), *11*
graphics::hist(), *13*
graphics::par(), *13*
graphics::plot(), *13*

is.dsp, 11

mcmc, *2, 3*

nobs.dsp *(*dsp-accessors*)*, 6
nobs.dsp(), *11*

par_names *(*dsp-accessors*)*, 6
par_names(), *3, 4*, *6*, *11, 12*, *19*
plot.dsp, 12
plot.dsp(), *5–7*
predict.dsp, 15
predict.dsp(), *5*, *7*
print.dsp *(*dsp_fit*)*, 7
print.dsp_bands *(*credible_bands*)*, 4
print.dsp_spec *(*dsp_spec*)*, 10
print.summary.dsp, 16

simRegression, 16
simUnivariate, 18
stats::quantile(), *7*
summary.dsp, 18
summary.dsp(), *3*, *6, 7*, *11*, *16*