

# Package ‘dsp’

May 12, 2026

**Type** Package

**Title** Dynamic Shrinkage Process and Change Point Detection

**Version** 1.4.0

**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 (2024) <[doi:10.48550/arXiv.2408.11315](https://doi.org/10.48550/arXiv.2408.11315)>.

**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>>),  
 Jason B. Cho [aut],  
 David S. Matteson [aut]

**Maintainer** Toryn Schafer <toryn27@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-05-12 19:00:12 UTC

## Contents

abco . . . . .	4
btf . . . . .	5
btf0 . . . . .	7
btf_bspline . . . . .	9
btf_bspline0 . . . . .	11
btf_reg . . . . .	12
btf_sparse . . . . .	14
build_Q . . . . .	17
build_XtX . . . . .	17
computeDIC_ASV . . . . .	18
credBands . . . . .	18
dsp_fit . . . . .	19
dsp_spec . . . . .	21
ergMean . . . . .	22
fit_ASV . . . . .	23
fit_paramsASV . . . . .	24
fit_paramsASV_n . . . . .	25
generate_ly2hat . . . . .	26
getARpXmat . . . . .	26
getEffSize . . . . .	27
getNonZeros . . . . .	27
initCholReg_spam . . . . .	28
initChol_spam . . . . .	28
initDHS . . . . .	29
initEvol0 . . . . .	29
initEvolParams . . . . .	30
initSV . . . . .	30
init_paramsASV . . . . .	31
init_paramsASV_n . . . . .	32
invlogit . . . . .	32
logit . . . . .	33
ncind . . . . .	33

plot.dsp . . . . .	34
predict.dsp . . . . .	37
sampleAR1 . . . . .	38
sampleBTF . . . . .	39
sampleBTF_bspline . . . . .	40
sampleBTF_reg . . . . .	41
sampleBTF_reg_backfit . . . . .	41
sampleBTF_sparse . . . . .	42
sampleDSP . . . . .	43
sampleEvol0 . . . . .	44
sampleEvolParams . . . . .	45
sampleFastGaussian . . . . .	46
sampleLogVolMu . . . . .	46
sampleLogVolMu0 . . . . .	47
sampleLogVols . . . . .	48
sampleSVparams . . . . .	49
sampleSVparams0 . . . . .	49
sample_j_wrap . . . . .	50
sample_mat_c . . . . .	50
simBaS . . . . .	51
simRegression . . . . .	52
simRegression0 . . . . .	53
simUnivariate . . . . .	54
spec_dsp . . . . .	55
summary.dsp . . . . .	55
t_create_loc . . . . .	56
t_initEvolParams_no . . . . .	57
t_initEvolZeta_ps . . . . .	57
t_initSV . . . . .	58
t_sampleAR1 . . . . .	58
t_sampleBTF . . . . .	59
t_sampleEvolParams . . . . .	60
t_sampleEvolZeta_ps . . . . .	61
t_sampleLogVolMu . . . . .	61
t_sampleLogVols . . . . .	62
t_sampleR_mh . . . . .	63
t_sampleSVparams . . . . .	64
uni.slice . . . . .	65

**Description**

Run the MCMC sampler for ABCO with a penalty on first ( $D = 1$ ), or second ( $D = 2$ ) differences of the conditional expectation. The penalty utilizes the dynamic horseshoe prior on the evolution errors. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

**Usage**

```
abco(
  y,
  D = 1,
  useAnom = TRUE,
  obsSV = "const",
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  mcmc_params = list("mu", "omega", "yhat", "evol_sigma_t2", "r", "zeta", "obs_sigma_t2",
    "zeta_sigma_t2", "dhs_phi", "dhs_mean", "h", "h_smooth"),
  verbose = TRUE,
  D_asv = 1,
  evol_error_asv = "HS",
  nugget_asv = TRUE
)
```

**Arguments**

<code>y</code>	the T vector of time series observations
<code>D</code>	degree of differencing ( $D = 1$ , or $D = 2$ )
<code>useAnom</code>	logical; if TRUE, include an anomaly component in the observation equation
<code>obsSV</code>	Options for modeling the error variance. It must be one of the following: <ul style="list-style-type: none"> <li>• <code>const</code>: Constant error variance for all time points.</li> <li>• <code>SV</code>: Stochastic Volatility model.</li> <li>• <code>ASV</code>: Adaptive Stochastic Volatility model.</li> </ul>
<code>nsave</code>	number of MCMC iterations to record
<code>nburn</code>	number of MCMC iterations to discard (burnin)
<code>nskip</code>	number of MCMC iterations to skip between saving iterations, i.e., save every ( <code>nskip + 1</code> )th draw
<code>mcmc_params</code>	named list of parameters for which we store the MCMC output; must be one or more of: <ul style="list-style-type: none"> <li>• <code>"mu"</code> (conditional mean)</li> </ul>

	<ul style="list-style-type: none"> <li>• "omega" (Dth difference of mu)</li> <li>• "yhat" (posterior predictive distribution)</li> <li>• "evol_sigma_t2" (evolution error variance)</li> <li>• "obs_sigma_t2" (observation error variance)</li> <li>• "zeta_sigma_t2" (outlier error variance)</li> <li>• "dhs_phi" (DHS AR(1) coefficient)</li> <li>• "dhs_mean" (DHS AR(1) unconditional mean)</li> <li>• "h" (log variances or log of "obs_sigma_t2". Only used when obsSV = "ASV")</li> <li>• "h_smooth" (smooth estimate of log variances. Only used when obsSV = "ASV" and nugget_asv = TRUE)</li> </ul>
verbose	logical; should R report extra information on progress?
D_asv	integer; degree of differencing (0, 1, or 2) for the ASV model. Only used when obsSV = "ASV".
evol_error_asv	character; evolution error distribution for the ASV model. Must be one of the five options used in evol_error. Only used when obsSV = "ASV".
nugget_asv	logical; if TRUE, fits the nugget variant of the ASV model. Only used when obsSV = "ASV".

### Value

A named list of the nsave MCMC samples for the parameters named in mcmc\_params

---

btf

*MCMC Sampler for Bayesian Trend Filtering*

---

### Description

Run the MCMC for Bayesian trend filtering with a penalty on zeroth ( $D = 0$ ), first ( $D = 1$ ), or second ( $D = 2$ ) differences of the conditional expectation. The penalty is determined by the prior on the evolution errors, which include:

- the dynamic horseshoe prior ('DHS');
- the static horseshoe prior ('HS');
- the Bayesian lasso ('BL');
- the normal stochastic volatility model ('SV');
- the normal-inverse-gamma prior ('NIG').

In each case, the evolution error is a scale mixture of Gaussians. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

**Usage**

```

btf(
  y,
  evol_error = "DHS",
  D = 2,
  obsSV = "const",
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  mcmc_params = list("mu", "ypred", "evol_sigma_t2", "obs_sigma_t2", "dhs_phi",
    "dhs_mean", "h", "h_smooth"),
  computeDIC = TRUE,
  verbose = TRUE,
  D_asv = 1,
  evol_error_asv = "HS",
  nugget_asv = TRUE
)

```

**Arguments**

y	the T x 1 vector of time series observations
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior; the default), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
D	degree of differencing (D = 0, D = 1, or D = 2)
obsSV	Options for modeling the error variance. It must be one of the following: <ul style="list-style-type: none"> <li>• const: Constant error variance for all time points.</li> <li>• SV: Stochastic Volatility model.</li> <li>• ASV: Adaptive Stochastic Volatility model.</li> </ul> for the observation error variance
nsave	number of MCMC iterations to record
nburn	number of MCMC iterations to discard (burnin)
nskip	number of MCMC iterations to skip between saving iterations, i.e., save every (nskip + 1)th draw
mcmc_params	named list of parameters for which we store the MCMC output; must be one or more of: <ul style="list-style-type: none"> <li>• "mu" (conditional mean)</li> <li>• "ypred" (posterior predictive distribution)</li> <li>• "evol_sigma_t2" (evolution error variance)</li> <li>• "obs_sigma_t2" (observation error variance)</li> <li>• "dhs_phi" (DHS AR(1) coefficient)</li> <li>• "dhs_mean" (DHS AR(1) unconditional mean)</li> <li>• "h" (log variances or log of "obs_sigma_t2". Only used when obsSV = "ASV")</li> </ul>

	<ul style="list-style-type: none"> <li>• "h_smooth" (smooth estimate of log variances. Only used when obsSV = "ASV" and nugget_asv = TRUE)</li> </ul>
computeDIC	logical; if TRUE, compute the deviance information criterion DIC and the effective number of parameters $p_d$
verbose	logical; should R report extra information on progress?
D_asv	integer; degree of differencing (0, 1, or 2) for the ASV model. Only used when obsSV = "ASV".
evol_error_asv	character; evolution error distribution for the ASV model. Must be one of the five options used in evol_error. Only used when obsSV = "ASV".
nugget_asv	logical; if TRUE, fits the nugget variant of the ASV model. Only used when obsSV = "ASV".

### Value

A named list of the nsave MCMC samples for the parameters named in mcmc\_params

### 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.

---

btf0

*MCMC Sampler for Bayesian Trend Filtering:  $D = 0$*

---

### Description

Run the MCMC for Bayesian trend filtering with a penalty on the conditional expectation. The penalty is determined by the prior on the evolution errors, which include:

- the dynamic horseshoe prior ('DHS');
- the static horseshoe prior ('HS');
- the Bayesian lasso ('BL');
- the normal stochastic volatility model ('SV');
- the normal-inverse-gamma prior ('NIG').

In each case, the evolution error is a scale mixture of Gaussians. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

**Usage**

```

btf0(
  y,
  evol_error = "DHS",
  obsSV = "const",
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  mcmc_params = list("mu", "ypred", "evol_sigma_t2", "obs_sigma_t2", "dhs_phi",
    "dhs_mean", "h", "h_smooth"),
  computeDIC = TRUE,
  verbose = TRUE,
  D_asv = 1,
  evol_error_asv = "HS",
  nugget_asv = TRUE
)

```

**Arguments**

y	the T x 1 vector of time series observations
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
obsSV	Options for modeling the error variance. It must be one of the following: <ul style="list-style-type: none"> <li>• const: Constant error variance for all time points.</li> <li>• SV: Stochastic Volatility model.</li> <li>• ASV: Adaptive Stochastic Volatility model.</li> </ul>
nsave	number of MCMC iterations to record
nburn	number of MCMC iterations to discard (burnin)
nskip	number of MCMC iterations to skip between saving iterations, i.e., save every (nskip + 1)th draw
mcmc_params	named list of parameters for which we store the MCMC output; must be one or more of: <ul style="list-style-type: none"> <li>• "mu" (conditional mean)</li> <li>• "ypred" (posterior predictive distribution)</li> <li>• "evol_sigma_t2" (evolution error variance)</li> <li>• "obs_sigma_t2" (observation error variance)</li> <li>• "dhs_phi" (DHS AR(1) coefficient)</li> <li>• "dhs_mean" (DHS AR(1) unconditional mean)</li> <li>• "h" (log variances or log of "obs_sigma_t2". Only used when obsSV = "ASV")</li> <li>• "h_smooth" (smooth estimate of log variances. Only used when obsSV = "ASV" and nugget_asv = TRUE)</li> </ul>
computeDIC	logical; if TRUE, compute the deviance information criterion DIC and the effective number of parameters p <sub>d</sub>

verbose	logical; should R report extra information on progress?
D_asv	integer; degree of differencing (0, 1, or 2) for the ASV model. Only used when obsSV = "ASV".
evol_error_asv	character; evolution error distribution for the ASV model. Must be one of the five options used in evol_error. Only used when obsSV = "ASV".
nugget_asv	logical; if TRUE, fits the nugget variant of the ASV model. Only used when obsSV = "ASV".

**Value**

A named list of the nsave MCMC samples for the parameters named in mcmc\_params

**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.

---

btf\_bspline

*MCMC Sampler for B-spline Bayesian Trend Filtering*


---

**Description**

Run the MCMC for B-spline fitting with a Bayesian trend filtering model on the coefficients, i.e., a penalty on zeroth ( $D=0$ ), first ( $D=1$ ), or second ( $D=2$ ) differences of the B-spline basis coefficients. The penalty is determined by the prior on the evolution errors, which include:

- the dynamic horseshoe prior ('DHS');
- the static horseshoe prior ('HS');
- the Bayesian lasso ('BL');
- the normal stochastic volatility model ('SV');
- the normal-inverse-gamma prior ('NIG').

In each case, the evolution error is a scale mixture of Gaussians. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

**Usage**

```
btf_bspline(
  y,
  times = NULL,
  num_knots = NULL,
  evol_error = "DHS",
  D = 2,
  nsave = 1000,
  nburn = 1000,
```

```

nskip = 4,
mcmc_params = list("mu", "ypred", "beta", "evol_sigma_t2", "obs_sigma_t2", "dhs_phi",
  "dhs_mean"),
computeDIC = TRUE,
verbose = TRUE
)

```

### Arguments

<code>y</code>	the $T \times 1$ vector of time series observations
<code>times</code>	the $T \times 1$ vector of observation points; if NULL, assume equally spaced
<code>num_knots</code>	the number of knots; if NULL, use the default of $\max(20, \min(\text{ceiling}(T/4), 150))$
<code>evol_error</code>	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
<code>D</code>	degree of differencing ( $D = 0$ , $D = 1$ , or $D = 2$ )
<code>nsave</code>	number of MCMC iterations to record
<code>nburn</code>	number of MCMC iterations to discard (burn-in)
<code>nskip</code>	number of MCMC iterations to skip between saving iterations, i.e., save every $(\text{nskip} + 1)$ th draw
<code>mcmc_params</code>	named list of parameters for which we store the MCMC output; must be one or more of: <ul style="list-style-type: none"> <li>• "mu" (conditional mean)</li> <li>• "beta" (B-spline basis coefficients)</li> <li>• "ypred" (posterior predictive distribution)</li> <li>• "evol_sigma_t2" (evolution error variance)</li> <li>• "obs_sigma_t2" (observation error variance)</li> <li>• "dhs_phi" (DHS AR(1) coefficient)</li> <li>• "dhs_mean" (DHS AR(1) unconditional mean)</li> </ul>
<code>computeDIC</code>	logical; if TRUE, compute the deviance information criterion DIC and the effective number of parameters $p_d$
<code>verbose</code>	logical; should R report extra information on progress?

### Value

A named list of the `nsave` MCMC samples for the parameters named in `mcmc_params`

### 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.

The primary advantages of `btf_bspline` over `btf` are

1. Unequally-spaced points are handled automatically and
2. Computations are linear in the number of basis coefficients, which may be substantially fewer than the number of time points.

---

btf\_bspline0

*MCMC Sampler for B-spline Bayesian Trend Filtering:  $D = 0$* 


---

### Description

Run the MCMC for B-spline fitting with a penalty the B-spline basis coefficients. The penalty is determined by the prior on the evolution errors, which include:

- the dynamic horseshoe prior ('DHS');
- the static horseshoe prior ('HS');
- the Bayesian lasso ('BL');
- the normal stochastic volatility model ('SV');
- the normal-inverse-gamma prior ('NIG').

In each case, the evolution error is a scale mixture of Gaussians. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

### Usage

```
btf_bspline0(
  y,
  times = NULL,
  num_knots = NULL,
  evol_error = "DHS",
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  mcmc_params = list("mu", "ypred", "beta", "evol_sigma_t2", "obs_sigma_t2", "dhs_phi",
    "dhs_mean"),
  computeDIC = TRUE,
  verbose = TRUE
)
```

### Arguments

y	the $T \times 1$ vector of time series observations
times	the $T \times 1$ vector of observation points; if NULL, assume equally spaced
num_knots	the number of knots; if NULL, use the default of $\max(20, \min(\text{ceiling}(T/4), 150))$
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)

nsave	number of MCMC iterations to record
nburn	number of MCMC iterations to discard (burn-in)
nskip	number of MCMC iterations to skip between saving iterations, i.e., save every (nskip + 1)th draw
mcmc_params	named list of parameters for which we store the MCMC output; must be one or more of: <ul style="list-style-type: none"> <li>• "mu" (conditional mean)</li> <li>• "beta" (B-spline basis coefficients)</li> <li>• "ypred" (posterior predictive distribution)</li> <li>• "evol_sigma_t2" (evolution error variance)</li> <li>• "obs_sigma_t2" (observation error variance)</li> <li>• "dhs_phi" (DHS AR(1) coefficient)</li> <li>• "dhs_mean" (DHS AR(1) unconditional mean)</li> </ul>
computeDIC	logical; if TRUE, compute the deviance information criterion DIC and the effective number of parameters $p_d$
verbose	logical; should R report extra information on progress?

**Value**

A named list of the nsave MCMC samples for the parameters named in mcmc\_params

**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.

The primary advantages of btf\_bspline over btf are

1. Unequally-spaced points are handled automatically and
2. Computations are linear in the number of basis coefficients, which may be substantially fewer than the number of time points.

---

btf\_reg

*MCMC Sampler for Bayesian Trend Filtering: Regression*


---

**Description**

Run the MCMC for Bayesian trend filtering regression with a penalty on first (D=1) or second (D=2) differences of each dynamic regression coefficient. The penalty is determined by the prior on the evolution errors, which include:

- the dynamic horseshoe prior ('DHS');
- the static horseshoe prior ('HS');
- the Bayesian lasso ('BL');

- the normal stochastic volatility model ('SV');
- the normal-inverse-gamma prior ('NIG').

In each case, the evolution error is a scale mixture of Gaussians. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

### Usage

```
btf_reg(
  y,
  X = NULL,
  evol_error = "DHS",
  D = 1,
  obsSV = "const",
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  mcmc_params = list("mu", "ypred", "beta", "evol_sigma_t2", "obs_sigma_t2", "dhs_phi",
    "dhs_mean", "h", "h_smooth"),
  use_backfitting = FALSE,
  computeDIC = TRUE,
  verbose = TRUE,
  D_asv = 1,
  evol_error_asv = "HS",
  nugget_asv = TRUE
)
```

### Arguments

<code>y</code>	the $T \times 1$ vector of time series observations
<code>X</code>	the $T \times p$ matrix of time series predictors
<code>evol_error</code>	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
<code>D</code>	degree of differencing ( $D = 1$ or $D = 2$ )
<code>obsSV</code>	Options for modeling the error variance. It must be one of the following: <ul style="list-style-type: none"> <li>• <code>const</code>: Constant error variance for all time points.</li> <li>• <code>SV</code>: Stochastic Volatility model.</li> <li>• <code>ASV</code>: Adaptive Stochastic Volatility model.</li> </ul>
<code>nsave</code>	number of MCMC iterations to record
<code>nburn</code>	number of MCMC iterations to discard (burn-in)
<code>nskip</code>	number of MCMC iterations to skip between saving iterations, i.e., save every $(nskip + 1)$ th draw
<code>mcmc_params</code>	named list of parameters for which we store the MCMC output; must be one or more of: <ul style="list-style-type: none"> <li>• "mu" (conditional mean)</li> </ul>

- "ypred" (posterior predictive distribution)
- "beta" (dynamic regression coefficients)
- "evol\_sigma\_t2" (evolution error variance)
- "obs\_sigma\_t2" (observation error variance)
- "dhs\_phi" (DHS AR(1) coefficient)
- "dhs\_mean" (DHS AR(1) unconditional mean)
- "h" (log variances or log of "obs\_sigma\_t2". Only used when obsSV = "ASV")
- "h\_smooth" (smooth estimate of log variances. Only used when obsSV = "ASV" and nugget\_asv = TRUE)

use_backfitting	logical; if TRUE, use backfitting to sample the predictors $j=1,\dots,p$ (faster, but usually less MCMC efficient)
computeDIC	logical; if TRUE, compute the deviance information criterion DIC and the effective number of parameters $p_d$
verbose	logical; should R report extra information on progress?
D_asv	integer; degree of differencing (0, 1, or 2) for the ASV model. Only used when obsSV = "ASV".
evol_error_asv	character; evolution error distribution for the ASV model. Must be one of the five options used in evol_error. Only used when obsSV = "ASV".
nugget_asv	logical; if TRUE, fits the nugget variant of the ASV model. Only used when obsSV = "ASV".

**Value**

A named list of the `nsave` MCMC samples for the parameters named in `mcmc_params`

**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.

---

btf\_sparse

---

*Run the MCMC for sparse Bayesian trend filtering*


---

**Description**

Sparse Bayesian trend filtering has two penalties: (1) a penalty on the first ( $D = 1$ ) or second ( $D = 2$ ) differences of the conditional expectation and (2) a penalty on the conditional expectation, i.e., shrinkage to zero.

**Usage**

```

btf_sparse(
  y,
  evol_error = "DHS",
  zero_error = "DHS",
  D = 2,
  obsSV = "const",
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  mcmc_params = list("mu", "ypred", "evol_sigma_t2", "obs_sigma_t2", "zero_sigma_t2",
    "dhs_phi", "dhs_mean", "dhs_phi_zero", "dhs_mean_zero", "h", "h_smooth"),
  computeDIC = TRUE,
  verbose = TRUE,
  D_asv = 1,
  evol_error_asv = "HS",
  nugget_asv = TRUE
)

```

**Arguments**

y	the $T \times 1$ vector of time series observations
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
zero_error	the shrinkage-to-zero distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
D	degree of differencing ( $D = 1$ , or $D = 2$ )
obsSV	Options for modeling the error variance. It must be one of the following: <ul style="list-style-type: none"> <li>• const: Constant error variance for all time points.</li> <li>• SV: Stochastic Volatility model.</li> <li>• ASV: Adaptive Stochastic Volatility model.</li> </ul>
nsave	number of MCMC iterations to record
nburn	number of MCMC iterations to discard (burnin-in)
nskip	number of MCMC iterations to skip between saving iterations, i.e., save every (nskip + 1)th draw
mcmc_params	named list of parameters for which we store the MCMC output; must be one or more of: <ul style="list-style-type: none"> <li>• "mu" (conditional mean)</li> <li>• "ypred" (posterior predictive distribution)</li> <li>• "evol_sigma_t2" (evolution error variance)</li> <li>• "zero_sigma_t2" (shrink-to-zero error variance)</li> <li>• "obs_sigma_t2" (observation error variance)</li> </ul>

	<ul style="list-style-type: none"> <li>• "dhs_phi" (DHS AR(1) coefficient for evolution error)</li> <li>• "dhs_mean" (DHS AR(1) unconditional mean for evolution error)</li> <li>• "dhs_phi_zero" (DHS AR(1) coefficient for shrink-to-zero error)</li> <li>• "dhs_mean_zero" (DHS AR(1) unconditional mean for shrink-to-zero error)</li> <li>• "h" (log variances or log of "obs_sigma_t2". Only used when obsSV = "ASV")</li> <li>• "h_smooth" (smooth estimate of log variances. Only used when obsSV = "ASV" and nugget_asv = TRUE)</li> </ul>
computeDIC	logical; if TRUE, compute the deviance information criterion DIC and the effective number of parameters $p_d$
verbose	logical; should R report extra information on progress?
D_asv	integer; degree of differencing (0, 1, or 2) for the ASV model. Only used when obsSV = "ASV".
evol_error_asv	character; evolution error distribution for the ASV model. Must be one of the five options used in evol_error. Only used when obsSV = "ASV".
nugget_asv	logical; if TRUE, fits the nugget variant of the ASV model. Only used when obsSV = "ASV".

### Details

Each penalty is determined by a prior, which include:

- the dynamic horseshoe prior ('DHS');
- the static horseshoe prior ('HS');
- the Bayesian lasso ('BL');
- the normal stochastic volatility model ('SV');
- the normal-inverse-gamma prior ('NIG').

In each case, the prior is a scale mixture of Gaussians. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

### Value

A named list of the `nsave` MCMC samples for the parameters named in `mcmc_params`

### 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.

---

<code>build_Q</code>	<i>Compute the quadratic term in Bayesian trend filtering</i>
----------------------	---

---

**Description**

Compute the quadratic term arising in the full conditional distribution of a Bayesian trend filtering model with  $D = 1$  or  $D = 2$ . This function exploits the known  $D$ -banded structure of  $Q$  to compute the matrix directly, using objects in the Matrix package.

**Usage**

```
build_Q(obs_sigma_t2, evol_sigma_t2, D = 1)
```

**Arguments**

<code>obs_sigma_t2</code>	the $T \times 1$ vector of observation error variances
<code>evol_sigma_t2</code>	the $T \times 1$ vector of evolution error variances
<code>D</code>	the degree of differencing (one or two)

**Value**

Banded  $T \times T$  Matrix (object)  $Q$

---

<code>build_XtX</code>	<i>Compute <math>X'X</math></i>
------------------------	---------------------------------

---

**Description**

Build the  $T_p \times T_p$  matrix  $XtX$  using the Matrix() package

**Usage**

```
build_XtX(X)
```

**Arguments**

<code>X</code>	$T \times p$ matrix of predictors
----------------	-----------------------------------

**Value**

Block diagonal  $T_p \times T_p$  Matrix (object) where each  $p \times p$  block is `tcrossprod(matrix(X[t,]))`

**Note**

$X'X$  is a one-time computing cost. Special cases may have more efficient computing options, but the Matrix representation is important for efficient computations within the sampler.

---

computeDIC_ASV	<i>Function for calculating DIC and Pb (Bayesian measures of model complexity and fit by Spiegelhalter et al. 2002)</i>
----------------	---

---

**Description**

Function for calculating DIC and Pb (Bayesian measures of model complexity and fit by Spiegelhalter et al. 2002)

**Usage**

```
computeDIC_ASV(y, beta, post_sigma2, post_loglike)
```

**Arguments**

y	the T x 1 vector of time series observations.
beta	the known mean of the process. 0 by default.
post_sigma2	posterior samples of the variance, i.e. exp(h)
post_loglike	log likelihood based on the posterior sample.

**Value**

a list containing DIC and p\_d. Two options for estimating both DIC and p\_d, which are both included.

---

credBands	<i>Compute Simultaneous Credible Bands</i>
-----------	--

---

**Description**

Compute (1-alpha)\

**Usage**

```
credBands(sampFuns, alpha = 0.05)
```

**Arguments**

sampFuns	Nsims x m matrix of Nsims MCMC samples and m points along the curve
alpha	confidence level

**Value**

m x 2 matrix of credible bands; the first column is the lower band, the second is the upper band

**Note**

The input needs not be curves: the simultaneous credible "bands" may be computed for vectors. The resulting credible intervals will provide joint coverage at the  $(1-\alpha)\%$  level across all components of the vector.

---

 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,
  ...
)

## 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 <code>dsp_spec()</code> .
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
...	currently not used
x	object of class dsp from <code>dsp_fit()</code>

### 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
mcpair	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.
- yhat: Posterior predictive  $\hat{y}_t$ .
- evol\_sigma\_t2: Variance of the state variable.
- obs\_sigma\_t2: Observation variance  $\sigma_\epsilon^2$  (family = "gaussian").
- dhs\_phi, dhs\_mean: DHS hyperparameter draws (evol\_error = "DHS").
- h: Time-varying log-volatility component,  $\log(\text{obs\_sigma\_t2})$  (obsSV = "ASV").

Model-specific parameters include:

- changepoint (Gaussian):
  - omega:  $D$ -th differenced posterior mean draws,  $\mu_t$ .
  - zeta: anomaly component (when useAnom = TRUE).
  - zeta\_sigma\_t2: anomaly variance (when useAnom = TRUE).
  - r: 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 = 500, nburn = 500)

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"> <li>• "gaussian": Gaussian family.</li> <li>• "negbinomial": Negative binomial family.</li> </ul>
model	A character string specifying the model type: <ul style="list-style-type: none"> <li>• family = "gaussian":           <ul style="list-style-type: none"> <li>– "changepoint": Change point detection with Adaptive Bayesian Change-point analysis and local Outlier (ABCO),</li> </ul> </li> </ul>

- "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 = "changeoint")

print(model_spec)
```

---

 ergMean

---

*Compute the ergodic (running) mean.*


---

**Description**

Compute the ergodic (running) mean.

**Usage**

```
ergMean(x)
```

**Arguments**

x vector for which to compute the running mean

**Value**

A vector y with each element defined by  $y[i] = \text{mean}(x[1:i])$

**Description**

The penalty is determined by the prior on the evolution errors, which include:

- the dynamic horseshoe prior ('DHS');
- the static horseshoe prior ('HS');
- the Bayesian lasso ('BL');
- the normal-inverse-gamma prior ('NIG').

In each case, the evolution error is a scale mixture of Gaussians. Sampling is accomplished with a (parameter-expanded) Gibbs sampler, mostly relying on a dynamic linear model representation.

**Usage**

```
fit_ASV(
  y,
  beta = 0,
  evol_error = "DHS",
  D = 1,
  nsave = 1000,
  nburn = 1000,
  nskip = 4,
  mcmc_params = list("h", "logy2hat", "sigma2", "evol_sigma_t2", "dhs_phi", "dhs_mean"),
  nugget = FALSE,
  computeDIC = TRUE,
  verbose = TRUE
)
```

**Arguments**

<code>y</code>	the $T \times 1$ vector of time series observations.
<code>beta</code>	the mean of the observed process $y$ . If not provided, they are assumed to be 0.
<code>evol_error</code>	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
<code>D</code>	degree of differencing ( $D = 1$ , or $D = 2$ )
<code>nsave</code>	number of MCMC iterations to record
<code>nburn</code>	number of MCMC iterations to discard (burn-in)
<code>nskip</code>	number of MCMC iterations to skip between saving iterations, i.e., save every ( <code>nskip + 1</code> )th draw
<code>mcmc_params</code>	named list of parameters for which we store the MCMC output; must be one or more of:

- "h" (Log variance)
- "h\_smooth" (smooth estimate of log variances. Only used when nugget\_asv = TRUE)
- "logy2hat" (posterior predictive distribution of  $\log(y^2)$ )
- "sigma2" (Variance, i.e.  $\exp(h)$ )
- "evol\_sigma\_t2" (evolution error variance)
- "dhs\_phi" (DHS AR(1) coefficient)
- "dhs\_mean" (DHS AR(1) unconditional mean)

nugget	logical; if TRUE, fits the nugget variant of the ASV model
computeDIC	logical; if TRUE, compute the deviance information criterion DIC and the effective number of parameters $\rho_d$
verbose	logical; should R report extra information on progress?

**Value**

A named list of the nsave MCMC samples for the parameters named in mcmc\_params

**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.

---

fit\_paramsASV

*Helper function for Sampling parameters for ASV model*


---

**Description**

Helper function for Sampling parameters for ASV model

**Usage**

```
fit_paramsASV(data, sParams, evol_error, D)
```

**Arguments**

data	the $T \times 1$ vector of time series observations.
sParams	list from the previous run of fit_paramsASV function or init_paramsASV function
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
D	degree of differencing ( $D = 1$ , or $D = 2$ )

**Value**

a list containing 4 sets of parameters

- `s_p_error_term`: matrix containing mean and the variance from 10-component gaussian mixture (Omori et al. 2007)
- `s_mu`: a vector containing the posterior sample of log variance  $h$ ,
- `s_evolParams0`: a list containing posterior samples of parameters associated with the variance of first  $D$  observation of the log variance term,  $h$ .
- `s_evolParams`: a list containing posterior samples parameters associated with the variance of  $D$  to the last observations of the log variance term,  $h$ .

---

<code>fit_paramsASV_n</code>	<i>Helper function for Sampling parameters for ASV model with a nugget Effect</i>
------------------------------	---

---

**Description**

Helper function for Sampling parameters for ASV model with a nugget Effect

**Usage**

```
fit_paramsASV_n(data, sParams, evol_error, D)
```

**Arguments**

<code>data</code>	the $T \times 1$ vector of time series observations.
<code>sParams</code>	list from the previous run of <code>fit_paramsASV</code> function or <code>init_paramsASV</code> function
<code>evol_error</code>	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
<code>D</code>	degree of differencing ( $D = 1$ , or $D = 2$ )

**Value**

a list containing 4 sets of parameters

- `s_p_error_term`: matrix containing mean and the variance from 10-component gaussian mixture (Omori et al. 2007)
- `s_mu`: a vector containing the posterior sample of log variance  $h$ ,
- `s_evolParams0`: a list containing posterior samples of parameters associated with the variance of first  $D$  observation of the log variance term,  $h$ .
- `s_evolParams`: a list containing posterior samples parameters associated with the variance of  $D$  to the last observations of the log variance term,  $h$ .

---

generate_ly2hat	<i>Posterior predictive sampler on the transformed <math>y</math> (<math>\log(y^2)</math>)</i>
-----------------	--

---

**Description**

Posterior predictive sampler on the transformed  $y$  ( $\log(y^2)$ )

**Usage**

```
generate_ly2hat(h, p_error_term)
```

**Arguments**

h	the log variance term h
p_error_term	2 dimensional data frame containing mean and the variance from the 10 component Gaussian mixture in Omori et al 2007 paper.

**Value**

a vector containing posterior predictive on  $\log(y^2)$

---

getARpXmat	<i>Compute the design matrix X for AR(p) model</i>
------------	--

---

**Description**

Compute the design matrix X for AR(p) model

**Usage**

```
getARpXmat(y, p = 1, include_intercept = FALSE)
```

**Arguments**

y	(T x 1) vector of responses
p	order of AR(p) model
include_intercept	logical; if TRUE, first column of X is ones

---

getEffSize	<i>Summarize of effective sample size</i>
------------	---

---

**Description**

Compute the summary statistics for the effective sample size (ESS) across posterior samples for possibly many variables

**Usage**

```
getEffSize(postX)
```

**Arguments**

postX	An array of arbitrary dimension (nsims x ... x ...), where nsims is the number of posterior samples
-------	---

**Value**

Table of summary statistics using the function `summary()`.

---

getNonZeros	<i>Compute Non-Zeros (Signals)</i>
-------------	------------------------------------

---

**Description**

Estimate the location of non-zeros (signals) implied by horseshoe-type thresholding.

**Usage**

```
getNonZeros(post_evol_sigma_t2, post_obs_sigma_t2 = NULL)
```

**Arguments**

post_evol_sigma_t2	the Nsims x T or Nsims x T x p matrix/array of posterior draws of the evolution error variances.
post_obs_sigma_t2	the Nsims x 1 or Nsims x T matrix of posterior draws of the observation error variances.

**Details**

Thresholding is based on  $\kappa[t] > 1/2$ , where  $\kappa = 1/(1 + \text{evol\_sigma\_t2}/\text{obs\_sigma\_t2})$ , `evol_sigma_t2` is the evolution error variance, and `obs_sigma_t2` is the observation error variance. In particular, the decision rule is based on the posterior mean of  $\kappa$ .

**Value**

A vector (or matrix) of indices identifying the signals according to the horseshoe-type thresholding rule.

**Note**

The thresholding rule depends on whether the prior variance for the state variable  $\mu$  (i.e., `evol_sigma_t2`) is scaled by the observation standard deviation, `obs_sigma_t2`. Explicitly, if  $\mu[t] \sim N(0, \text{evol\_sigma\_t2}[t])$  then the correct thresholding rule is based on  $\kappa = 1/(1 + \text{evol\_sigma\_t2}/\text{obs\_sigma\_t2})$ . However, if  $\mu[t] \sim N(0, \text{evol\_sigma\_t2}[t]*\text{obs\_sigma\_t2}[t])$  then the correct thresholding rule is based on  $\kappa = 1/(1 + \text{evol\_sigma\_t2})$ . The latter case may be implemented by omitting the input for `post_obs_sigma_t2` (or setting it to NULL).

---

<code>initCholReg_spam</code>	<i>Compute initial Cholesky decomposition for TVP Regression</i>
-------------------------------	--

---

**Description**

Computes the Cholesky decomposition for the quadratic term in the (Gaussian) posterior of the TVP regression coefficients. The sparsity pattern will not change during the MCMC, so we can save computation time by computing this up front.

**Usage**

```
initCholReg_spam(obs_sigma_t2, evol_sigma_t2, XtX, D = 1)
```

**Arguments**

<code>obs_sigma_t2</code>	the $T \times 1$ vector of observation error variances
<code>evol_sigma_t2</code>	the $T \times p$ matrix of evolution error variances
<code>XtX</code>	the $Tp \times Tp$ matrix of $X'X$ (one-time cost; see <code>?build_XtX</code> )
<code>D</code>	the degree of differencing (one or two)

---

<code>initChol_spam</code>	<i>Compute initial Cholesky decomposition for Bayesian Trend Filtering</i>
----------------------------	--

---

**Description**

Computes the Cholesky decomposition for the quadratic term in the (Gaussian) posterior of the Bayesian Trend Filtering coefficients. The sparsity pattern will not change during the MCMC, so we can save computation time by computing this up front.

**Usage**

```
initChol_spam(nT, D = 1)
```

**Arguments**

nT	number of time points
D	degree of differencing (D = 1 or D = 2)

---

initDHS	<i>Initialize the evolution error variance parameters</i>
---------	---

---

**Description**

Compute initial values for evolution error variance parameters under the dynamic horseshoe prior

**Usage**

```
initDHS(omega)
```

**Arguments**

omega	T x p matrix of evolution errors
-------	----------------------------------

**Value**

List of relevant components: the T x p evolution error SD `sigma_wt`, the T x p log-volatility `ht`, the p x 1 log-vol unconditional mean(s) `dhs_mean`, the p x 1 log-vol AR(1) coefficient(s) `dhs_phi`, the T x p log-vol innovation SD `sigma_eta_t` from the PG priors, the p x 1 initial log-vol SD `sigma_eta_0`, and the mean of log-vol means `dhs_mean0` (relevant when  $p > 1$ )

---

initEvol0	<i>Initialize the parameters for the initial state variance</i>
-----------	---

---

**Description**

The initial state SDs are assumed to follow half-Cauchy priors,  $C+(0,A)$ , where the SDs may be common or distinct among the states.

**Usage**

```
initEvol0(mu0, commonSD = TRUE)
```

**Arguments**

mu0	p x 1 vector of initial values (undifferenced)
commonSD	logical; if TRUE, use common SDs (otherwise distinct)

**Details**

This function initializes the parameters for a PX-Gibbs sampler.

**Value**

List of relevant components: the  $p \times 1$  evolution error SD  $\sigma_{w0}$ , the  $p \times 1$  parameter-expanded RV's  $\text{px\_sigma\_w0}$ , and the corresponding global scale parameters  $\sigma_{00}$  and  $\text{px\_sigma\_00}$  (ignore if commonSD)

---

initEvolParams	<i>Initialize the evolution error variance parameters</i>
----------------	---

---

**Description**

Compute initial values for evolution error variance parameters under the various options: dynamic horseshoe prior ('DHS'), horseshoe prior ('HS'), Bayesian lasso ('BL'), normal stochastic volatility ('SV'), or normal-inverse-gamma prior ('NIG').

**Usage**

```
initEvolParams(omega, evol_error = "DHS")
```

**Arguments**

omega	T x p matrix of evolution errors
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), or 'NIG' (normal-inverse-gamma prior)

**Value**

List of relevant components:  $\sigma_{wt}$ , the  $T \times p$  matrix of evolution standard deviations, and additional parameters associated with the DHS and HS priors.

---

initSV	<i>Initialize the stochastic volatility parameters</i>
--------	--

---

**Description**

Compute initial values for normal stochastic volatility parameters. The model assumes an AR(1) for the log-volatility.

**Usage**

```
initSV(omega)
```

**Arguments**

omega	T x p matrix of errors
-------	------------------------

**Value**

List of relevant components: `sigma_wt`, the  $T \times p$  matrix of standard deviations, and additional parameters (unconditional mean, AR(1) coefficient, and standard deviation).

---

<code>init_paramsASV</code>	<i>Helper function for initializing parameters for ASV model</i>
-----------------------------	--

---

**Description**

Helper function for initializing parameters for ASV model

**Usage**

```
init_paramsASV(data, evol_error, D)
```

**Arguments**

<code>data</code>	the $T \times 1$ vector of time series observations.
<code>evol_error</code>	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
<code>D</code>	degree of differencing ( $D = 1$ , or $D = 2$ )

**Value**

a list containing 4 sets of parameters

- `s_p_error_term`: matrix containing mean and the variance from 10-component gaussian mixture (Omori et al. 2007)
- `s_mu`: a vector containing the posterior sample of log variance  $h$ ,
- `s_evolParams0`: a list containing posterior samples of parameters associated with the variance of first  $D$  observation of the log variance term,  $h$ .
- `s_evolParams`: a list containing posterior samples parameters associated with the variance of  $D$  to the last observations of the log variance term,  $h$ .

---

init_paramsASV_n	<i>Helper function for initializing parameters for ASV model with a nugget effect</i>
------------------	---

---

**Description**

Helper function for initializing parameters for ASV model with a nugget effect

**Usage**

```
init_paramsASV_n(data, evol_error, D)
```

**Arguments**

data	the $T \times 1$ vector of time series observations.
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), 'BL' (Bayesian lasso), or 'NIG' (normal-inverse-gamma prior)
D	degree of differencing ( $D = 1$ , or $D = 2$ )

**Value**

a list containing 4 sets of parameters

- s\_p\_error\_term: matrix containing mean and the variance from 10-component gaussian mixture (Omori et al. 2007)
- s\_mu: a vector containing the posterior sample of log variance  $h$ ,
- s\_evolParams0: a list containing posterior samples of parameters associated with the variance of first  $D$  observation of the log variance term,  $h$ .
- s\_evolParams: a list containing posterior samples parameters associated with the variance of  $D$  to the last observations of the log variance term,  $h$ .

---

invlogit	<i>Compute the inverse log-odds</i>
----------	-------------------------------------

---

**Description**

Compute the inverse log-odds

**Usage**

```
invlogit(x)
```

**Arguments**

`x` scalar or vector for which to compute the (componentwise) inverse log-odds

**Value**

A scalar or vector of values in (0,1)

---

logit	<i>Compute the log-odds</i>
-------	-----------------------------

---

**Description**

Compute the log-odds

**Usage**

logit(x)

**Arguments**

`x` scalar or vector in (0,1) for which to compute the (componentwise) log-odds

**Value**

A scalar or vector of log-odds

---

ncind	<i>Sample components from a discrete mixture of normals</i>
-------	---

---

**Description**

Sample  $Z$  from  $1, 2, \dots, k$ , with  $P(Z=i)$  proportional to  $q_i N(\mu_i, \text{sig}_i^2)$ .

**Usage**

ncind(y, mu, sig, q)

**Arguments**

`y` vector of data  
`mu` vector of component means  
`sig` vector of component standard deviations  
`q` vector of component weights

**Value**

Sample from  $\{1, \dots, k\}$

---

plot.dsp

---

*Plot the Bayesian trend filtering fitted values*


---

### Description

Plot the BTF posterior mean of the conditional expectation with posterior credible intervals (point-wise and joint), the observed data, and true curves (if known)

### Usage

```
## S3 method for class 'dsp'
plot(
  x,
  type,
  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 <code>dsp_fit()</code> .
type	character string giving the parameter name to visualize; must be one of the entries in <code>x\$mcmc_output</code> .
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 $T \times 1$ 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 = 500, nburn = 500)
# 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 <code>dsp_fit()</code>
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`, a numeric vector of indices that correspond to indices of the observed data. If `cp_prop = TRUE`, a list containing:

- 'cp\_t': a numeric vector of indices that correspond to indices of the observed data.
- 'cp\_prop': a numeric vector of length (T - D) with the pointwise proportion of samples where the increment exceeds the threshold.

If no proportions exceed `cp_thres`, then the vector will be a length 0 integer vector.

**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 = "changeoint",
                      D = 1, useAnom = TRUE)
mcmc_output = dsp_fit(y, model_spec = model_spec, nsave = 500, nburn = 500)
predict(mcmc_output)

```

---

sampleAR1

*Sample the AR(1) coefficient(s)*


---

### Description

Compute one draw of the AR(1) coefficient in a model with Gaussian innovations and time-dependent innovation variances. In particular, we use the sampler for the log-volatility AR(1) process with the parameter-expanded Polya-Gamma sampler. The sampler also applies to a multivariate case with independent components.

### Usage

```
sampleAR1(h_yc, h_phi, h_sigma_eta_t, prior_dhs_phi = NULL)
```

### Arguments

h_yc	the $T \times p$ matrix of centered log-volatilities (i.e., the log-vols minus the unconditional means <code>dhs_mean</code> )
h_phi	the $p \times 1$ vector of previous AR(1) coefficient(s)
h_sigma_eta_t	the $T \times p$ matrix of log-vol innovation standard deviations
prior_dhs_phi	the parameters of the prior for the log-volatility AR(1) coefficient <code>dhs_phi</code> ; either NULL for uniform on $[-1, 1]$ or a 2-dimensional vector of (shape1, shape2) for a Beta prior on $[(dhs\_phi + 1)/2]$

### Value

$p \times 1$  vector of sampled AR(1) coefficient(s)

### Note

For the standard AR(1) case,  $p = 1$ . However, the function applies more generally for sampling  $p > 1$  independent AR(1) processes (jointly).

---

sampleBTF	<i>Sampler for first or second order random walk (RW) Gaussian dynamic linear model (DLM)</i>
-----------	---

---

**Description**

Compute one draw of the  $T \times 1$  state variable  $\mu$  in a DLM using back-band substitution methods. This model is equivalent to the Bayesian trend filtering (BTF) model, assuming appropriate (shrinkage/sparsity) priors for the evolution errors.

**Usage**

```
sampleBTF(
  y,
  obs_sigma_t2,
  evol_sigma_t2,
  D = 1,
  loc_obs = NULL,
  chol0 = NULL,
  prior_mean = NULL
)
```

**Arguments**

<code>y</code>	the $T \times 1$ vector of time series observations
<code>obs_sigma_t2</code>	the $T \times 1$ vector of observation error variances
<code>evol_sigma_t2</code>	the $T \times 1$ vector of evolution error variances
<code>D</code>	the degree of differencing (one or two)
<code>loc_obs</code>	list of the row and column indices to fill in a band-sparse matrix
<code>chol0</code>	(optional) the $m \times m$ matrix of initial Cholesky factorization; if NULL, use the Matrix package for sampling, otherwise use the spam package
<code>prior_mean</code>	optional (default is NULL); numeric $T \times 1$ vector specifying the prior mean of $\mu$

**Value**

$T \times 1$  vector of simulated states

**Note**

Missing entries (NAs) are not permitted in  $y$ . Imputation schemes are available.

---

sampleBTF_bspline	<i>Sampler for first or second order random walk (RW) Gaussian dynamic linear model (DLM)</i>
-------------------	---

---

### Description

Compute one draw of the  $p \times 1$  B-spline basis coefficients  $\beta$  in a DLM using back-band substitution methods. The coefficients are penalized with a prior on the  $D = 0$ ,  $D = 1$ , or  $D = 2$  differences. This model is equivalent to the Bayesian trend filtering (BTF) model applied to  $p \times 1$  vector of equally-spaced B-spline coefficients, with the basis matrix serving as a design matrix in the observation equation.

### Usage

```
sampleBTF_bspline(
  y,
  X,
  obs_sigma2,
  evol_sigma_t2,
  XtX_bands,
  Xty = NULL,
  D = 1
)
```

### Arguments

<code>y</code>	the $T \times 1$ vector of time series observations
<code>X</code>	the $T \times p$ basis matrix
<code>obs_sigma2</code>	the scalar observation error variance
<code>evol_sigma_t2</code>	the $p \times 1$ vector of evolution error variances
<code>XtX_bands</code>	list with 4 vectors consisting of the 4-bands of $XtX = \text{crossprod}(X)$ (one-time cost)
<code>Xty</code>	the $p \times 1$ matrix $\text{crossprod}(X,y)$ , which is a one-time cost (assuming no missing entries in $y$ )
<code>D</code>	the degree of differencing (zero, one, or two)

### Value

$p \times 1$  vector of simulated basis coefficients  $\beta$

### Note

Missing entries (NAs) are not permitted in  $y$ . Imputation schemes are available.

---

sampleBTF_reg	<i>Sampler for first or second order random walk (RW) Gaussian dynamic linear model (DLM)</i>
---------------	---

---

**Description**

Compute one draw of the  $T \times p$  state variable  $\beta$  in a DLM using back-band substitution methods. This model is equivalent to the Bayesian trend filtering (BTF) model applied to  $p$  dynamic regression coefficients corresponding to the design matrix  $X$ , assuming appropriate (shrinkage/sparsity) priors for the evolution errors.

**Usage**

```
sampleBTF_reg(y, X, obs_sigma_t2, evol_sigma_t2, XtX, D = 1, chol0 = NULL)
```

**Arguments**

$y$	the $T \times 1$ vector of time series observations
$X$	the $T \times p$ matrix of time series predictors
obs_sigma_t2	the $T \times 1$ vector of observation error variances
evol_sigma_t2	the $T \times p$ matrix of evolution error variances
$XtX$	the $Tp \times Tp$ matrix of $X'X$ (one-time cost; see ?build_XtX)
$D$	the degree of differencing (one or two)
chol0	(optional) the $m \times m$ matrix of initial Cholesky factorization; if NULL, use the Matrix package for sampling, otherwise use the spam package

**Value**

$T \times p$  matrix of simulated dynamic regression coefficients  $\beta$

**Note**

Missing entries (NAs) are not permitted in  $y$ . Imputation schemes are available.

---

sampleBTF_reg_backfit	<i>(Backfitting) Sampler for first or second order random walk (RW) Gaussian dynamic linear model (DLM)</i>
-----------------------	---

---

**Description**

Compute one draw of the  $T \times p$  state variable  $\beta$  in a DLM using back-band substitution methods. This model is equivalent to the Bayesian trend filtering (BTF) model applied to  $p$  dynamic regression coefficients corresponding to the design matrix  $X$ , assuming appropriate (shrinkage/sparsity) priors for the evolution errors. The sampler here uses a backfitting method that draws each predictor  $j=1, \dots, p$  conditional on the other predictors (and coefficients), which leads to a faster  $O(Tp)$  algorithm. However, the MCMC may be less efficient.

**Usage**

```
sampleBTF_reg_backfit(y, X, beta, obs_sigma_t2, evol_sigma_t2, D = 1)
```

**Arguments**

`y` the  $T \times 1$  vector of time series observations  
`X` the  $T \times p$  matrix of time series predictors  
`beta` the  $T \times p$  matrix of previous dynamic regression coefficients  
`obs_sigma_t2` the  $T \times 1$  vector of observation error variances  
`evol_sigma_t2` the  $T \times p$  matrix of evolution error variances  
`D` the degree of differencing (one or two)

**Value**

$T \times p$  matrix of simulated dynamic regression coefficients `beta`

**Note**

Missing entries (NAs) are not permitted in `y`. Imputation schemes are available.

---

sampleBTF_sparse	<i>Sampler for first or second order random walk (RW) Gaussian dynamic linear model (DLM) with additional shrinkage to zero</i>
------------------	---

---

**Description**

Compute one draw of the  $T \times 1$  state variable `mu` in a DLM using back-band substitution methods. This model is equivalent to the Bayesian trend filtering (BTF) model, assuming appropriate (shrinkage/sparsity) priors for the evolution errors, with an additional shrinkage-to-zero prior.

**Usage**

```
sampleBTF_sparse(  
  y,  
  obs_sigma_t2,  
  evol_sigma_t2,  
  zero_sigma_t2,  
  D = 1,  
  chol0 = NULL  
)
```

**Arguments**

y	the T x 1 vector of time series observations
obs_sigma_t2	the T x 1 vector of observation error variances
evol_sigma_t2	the T x 1 vector of evolution error variances
zero_sigma_t2	the T x 1 vector of shrink-to-zero variances
D	the degree of differencing (one or two)
chol0	(optional) the m x m matrix of initial Cholesky factorization; if NULL, use the Matrix package for sampling, otherwise use the spam package

**Value**

T x 1 vector of simulated states

**Note**

Missing entries (NAs) are not permitted in y. Imputation schemes are available.

---

sampleDSP

*Sample the dynamic shrinkage process parameters*

---

**Description**

Compute one draw for each of the parameters in the dynamic shrinkage process for the special case in which the shrinkage parameter  $\kappa \sim \text{Beta}(\alpha, \beta)$  with  $\alpha = \beta$ . The primary example is the dynamic horseshoe process with  $\alpha = \beta = 1/2$ .

**Usage**

```
sampleDSP(
  omega,
  evolParams,
  sigma_e = 1,
  loc = NULL,
  prior_dhs_phi = c(10, 2),
  alphaPlusBeta = 1
)
```

**Arguments**

omega	T x p matrix of evolution errors
evolParams	list of parameters to be updated (see Value below)
sigma_e	the observation error standard deviation; for (optional) scaling purposes
loc	list of the row and column indices to fill in a band-sparse matrix

prior_dhs_phi	the parameters of the prior for the log-volatility AR(1) coefficient dhs_phi; either NULL for uniform on [-1,1] or a 2-dimensional vector of (shape1, shape2) for a Beta prior on [(dhs_phi + 1)/2]
alphaPlusBeta	For the symmetric prior kappa ~ Beta(alpha, beta) with alpha=beta, specify the sum [alpha + beta]

**Value**

List of relevant components:

- the  $T \times p$  evolution error standard deviations sigma\_wt,
- the  $T \times p$  log-volatility ht, the  $p \times 1$  log-vol unconditional mean(s) dhs\_mean,
- the  $p \times 1$  log-vol AR(1) coefficient(s) dhs\_phi,
- the  $T \times p$  log-vol innovation standard deviations sigma\_eta\_t from the Polya-Gamma priors,
- the  $p \times 1$  initial log-vol SD sigma\_eta\_0,
- and the mean of log-vol means dhs\_mean0 (relevant when  $p > 1$ )

**Note**

The priors induced by prior\_dhs\_phi all imply a stationary (log-) volatility process.

---

sampleEvol0

*Sample the parameters for the initial state variance*

---

**Description**

The initial state SDs are assumed to follow half-Cauchy priors,  $C+(0,A)$ , where the SDs may be common or distinct among the states.

**Usage**

```
sampleEvol0(mu0, evolParams0, commonSD = FALSE, A = 1)
```

**Arguments**

mu0	$p \times 1$ vector of initial values (undifferenced)
evolParams0	list of relevant components (see below)
commonSD	logical; if TRUE, use common SDs (otherwise distinct)
A	prior scale parameter from the half-Cauchy prior, $C+(0,A)$

**Details**

This function samples the parameters for a PX-Gibbs sampler.

**Value**

List of relevant components: the  $p \times 1$  evolution error SD sigma\_w0 and the  $p \times 1$  parameter-expanded RV's px\_sigma\_w0

---

sampleEvolParams      *Sampler evolution error variance parameters*

---

### Description

Compute one draw of evolution error variance parameters under the various options:

- dynamic horseshoe prior ('DHS');
- horseshoe prior ('HS');
- normal-inverse-gamma prior ('NIG').

### Usage

```
sampleEvolParams(
  omega,
  evolParams,
  sigma_e = 1,
  evol_error = "DHS",
  loc = NULL
)
```

### Arguments

omega	T x p matrix of evolution errors
evolParams	list of parameters pertaining to each evol_error type to be updated
sigma_e	the observation error standard deviation; for (optional) scaling purposes
evol_error	the evolution error distribution; must be one of 'DHS' (dynamic horseshoe prior), 'HS' (horseshoe prior), or 'NIG' (normal-inverse-gamma prior)
loc	list of the row and column indices to fill in a band-sparse matrix

### Value

List of relevant components in evolParams: sigma\_wt, the T x p matrix of evolution standard deviations, and additional parameters associated with the DHS and HS priors.

### Note

The list evolParams is specific to each evol\_error type, but in each case contains the evolution error standard deviations sigma\_wt.

To avoid scaling by the observation standard deviation sigma\_e, simply use sigma\_e = 1 in the functional call.

---

sampleFastGaussian	<i>Sample a Gaussian vector using the fast sampler of BHATTACHARYA et al.</i>
--------------------	---

---

**Description**

Sample from  $N(\mu, \Sigma)$  where  $\Sigma = \text{solve}(\text{crossprod}(\Phi) + \text{solve}(D))$  and  $\mu = \Sigma * \text{crossprod}(\Phi, \alpha)$ :

**Usage**

```
sampleFastGaussian(Phi, Ddiag, alpha)
```

**Arguments**

Phi	n x p matrix (of predictors)
Ddiag	p x 1 vector of diagonal components (of prior variance)
alpha	n x 1 vector (of data, scaled by variance)

**Value**

Draw from  $N(\mu, \Sigma)$ , which is p x 1, and is computed in  $O(n^2 * p)$

**Note**

Assumes D is diagonal, but extensions are available

---

sampleLogVolMu	<i>Sample the AR(1) unconditional means</i>
----------------	---

---

**Description**

Compute one draw of the unconditional means in an AR(1) model with Gaussian innovations and time-dependent innovation variances. In particular, we use the sampler for the log-volatility AR(1) process with the parameter-expanded Polya-Gamma sampler. The sampler also applies to a multivariate case with independent components.

**Usage**

```
sampleLogVolMu(h, h_mu, h_phi, h_sigma_eta_t, h_sigma_eta_0, h_log_scale = 0)
```

**Arguments**

h	the $T \times p$ matrix of log-volatilities
h_mu	the $p \times 1$ vector of previous means
h_phi	the $p \times 1$ vector of AR(1) coefficient(s)
h_sigma_eta_t	the $T \times p$ matrix of log-vol innovation standard deviations
h_sigma_eta_0	the standard deviations of initial log-vols
h_log_scale	prior mean from scale mixture of Gaussian (Polya-Gamma) prior, e.g. $\log(\sigma_e^2)$ or dhs_mean0

**Value**

a list containing

- the sampled mean(s) dhs\_mean and
- the sampled precision(s) dhs\_mean\_prec\_j from the Polya-Gamma parameter expansion

---

sampleLogVolMu0	<i>Sample the mean of AR(1) unconditional means</i>
-----------------	---

---

**Description**

Compute one draw of the mean of unconditional means in an AR(1) model with Gaussian innovations and time-dependent innovation variances (for  $p > 1$ ). More generally, the sampler applies to the "mean" parameter (on the log-scale) for a Polya-Gamma parameter expanded hierarchical model.

**Usage**

```
sampleLogVolMu0(h_mu, h_mu0, dhs_mean_prec_j, h_log_scale = 0)
```

**Arguments**

h_mu	the $p \times 1$ vector of means
h_mu0	the previous mean of unconditional means
dhs_mean_prec_j	the $p \times 1$ vector of precisions (from the Polya-Gamma parameter expansion)
h_log_scale	prior mean from scale mixture of Gaussian (Polya-Gamma) prior, e.g. $\log(\sigma_e^2)$

**Value**

The sampled mean parameter dhs\_mean0

**Note**

This sampler is particularly for  $p > 1$  and the setting in which we want hierarchical shrinkage effects, e.g. predictor- and time-dependent shrinkage, predictor-dependent shrinkage, and global shrinkage, with a natural hierarchical ordering.

---

sampleLogVols

*Sample the latent log-volatilities*


---

### Description

Compute one draw of the log-volatilities using a discrete mixture of Gaussians approximation to the likelihood (see Omori, Chib, Shephard, and Nakajima, 2007) where the log-vols are assumed to follow an AR(1) model with time-dependent innovation variances. More generally, the code operates for  $p$  independent AR(1) log-vol processes to produce an efficient joint sampler in  $O(Tp)$  time.

### Usage

```
sampleLogVols(
  h_y,
  h_prev,
  h_mu,
  h_phi,
  h_sigma_eta_t,
  h_sigma_eta_0,
  loc = NULL
)
```

### Arguments

<code>h_y</code>	the $T \times p$ matrix of data, which follow independent SV models
<code>h_prev</code>	the $T \times p$ matrix of the previous log-vols
<code>h_mu</code>	the $p \times 1$ vector of log-vol unconditional means
<code>h_phi</code>	the $p \times 1$ vector of log-vol AR(1) coefficients
<code>h_sigma_eta_t</code>	the $T \times p$ matrix of log-vol innovation standard deviations
<code>h_sigma_eta_0</code>	the $p \times 1$ vector of initial log-vol innovation standard deviations
<code>loc</code>	list of the row and column indices to fill in a band-sparse matrix

### Value

$T \times p$  matrix of simulated log-vols

### Note

For Bayesian trend filtering,  $p = 1$ . More generally, the sampler allows for  $p > 1$  but assumes (contemporaneous) independence across the log-vols for  $j = 1, \dots, p$ .

---

sampleSVparams	<i>Sampler for the stochastic volatility parameters</i>
----------------	---

---

**Description**

Compute one draw of the normal stochastic volatility parameters. The model assumes an AR(1) for the log-volatility.

**Usage**

```
sampleSVparams(omega, svParams)
```

**Arguments**

omega	T x p matrix of errors
svParams	list of parameters to be updated

**Value**

List of relevant components in svParams: sigma\_wt, the T x p matrix of standard deviations, and additional parameters associated with SV model.

---

sampleSVparams0	<i>Sampler for the stochastic volatility parameters using same functions as DHS prior</i>
-----------------	---

---

**Description**

Compute one draw of the normal stochastic volatility parameters. The model assumes an AR(1) for the log-volatility.

**Usage**

```
sampleSVparams0(omega, svParams)
```

**Arguments**

omega	T x p matrix of errors
svParams	list of parameters to be updated

**Value**

List of relevant components in svParams: sigma\_wt, the T x p matrix of standard deviations, and additional parameters associated with SV model.

---

sample_j_wrap	<i>Sampling from 10-component Gaussian Mixture component described in Omori et al. 2007</i>
---------------	---

---

**Description**

Samples from the conditional posterior distribution of the  $\log(\chi^2)$  distribution by approximating it with the mixture Gaussian distribution described in Omori et al. 2007.

**Usage**

```
sample_j_wrap(Td, obs = NULL)
```

**Arguments**

Td	length of the vector
obs	Td x 1 vector for the data.

**Value**

Dataframe containing the posterior samples: mean and variance for the mixture component.

**Note**

When the obs is not not specified, the components are samples from the prior distribution.

---

sample_mat_c	<i>Wrapper function for C++ call for sample mat, check pre-conditions to prevent crash</i>
--------------	--

---

**Description**

Wrapper function for C++ call for sample mat, check pre-conditions to prevent crash

**Usage**

```
sample_mat_c(row_ind, col_ind, mat_val, mat_l, num_inp, linht, rd, D)
```

**Arguments**

row_ind	list of the row indices to fill in the bandsparse matrix
col_ind	list of the columns indices to fill in the bandsparse matrix
mat_val	list of the values to fill in the bandsparse matrix
mat_l	dimension of the band-sparse matrix
num_inp	number of non-zero elements in the bandsparse matrix
linht	T-D vector of linear term in the sampler
rd	T-D vector of standard normal noise samples
D	the degree of differencing for changepoint

---

 simBaS

---

*Compute Simultaneous Band Scores (SimBaS)*


---

**Description**

Compute simultaneous band scores (SimBaS) from Meyer et al. (2015, Biometrics). SimBaS uses MC(MC) simulations of a function of interest to compute the minimum alpha such that the joint credible bands at the alpha level do not include zero. This quantity is computed for each grid point (or observation point) in the domain of the function.

**Usage**

```
simBaS(sampFuns)
```

**Arguments**

sampFuns	Nsims x m matrix of Nsims MCMC samples and m points along the curve
----------	---

**Value**

m x 1 vector of simBaS

**Note**

The input needs not be curves: the simBaS may be computed for vectors to achieve a multiplicity adjustment.

The minimum of the returned value,  $P_{\text{simBaS}_t}$ , over the domain  $t$  is the Global Bayesian P-Value (GBP) for testing whether the function is zero everywhere.

---

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  $\beta_{\text{true}}$
- the true function  $\mu_{\text{true}}$
- the true observation standard deviation  $\sigma_{\text{true}}$

### Note

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

---

simRegression0	<i>Simulate noisy observations from a dynamic regression model</i>
----------------	--

---

### Description

Simulates data from a time series regression with dynamic regression coefficients. The dynamic regression coefficients are selected using the options from the `simUnivariate()` function in the `wmtsa` package.

### Usage

```
simRegression0(
  signalNames = c("bumps", "blocks"),
  nT = 200,
  RSNR = 10,
  p_0 = 5,
  include_intercept = TRUE,
  scale_all = TRUE,
  include_plot = TRUE,
  ar1 = 0
)
```

### Arguments

<code>signalNames</code>	vector of strings matching the "name" argument in the <code>simUnivariate()</code> function, e.g. "bumps" or "doppler"
<code>nT</code>	number of points
<code>RSNR</code>	root-signal-to-noise ratio
<code>p_0</code>	number of true zero regression terms to include
<code>include_intercept</code>	logical; if TRUE, the first column of X is 1's
<code>scale_all</code>	logical; if TRUE, scale all regression coefficients to [0,1]
<code>include_plot</code>	logical; if TRUE, include a plot of the simulated data and the true curve
<code>ar1</code>	the AR(1) coefficient for the predictors X; default is zero for iid N(0,1) predictors

### 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 number of predictors is  $p = \text{length}(\text{signalNames}) + p_0$ .

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

---

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)
```

---

spec_dsp	<i>Compute the spectrum of an AR(p) model</i>
----------	---

---

**Description**

Compute the spectrum of an AR(p) model

**Usage**

```
spec_dsp(ar_coefs, sigma_e, n.freq = 500)
```

**Arguments**

ar_coefs	(p x 1) vector of AR(p) coefficients
sigma_e	observation standard deviation
n.freq	number of frequencies at which to evaluate the spectrum

**Value**

A (n.freq x 2) matrix where the first column is the frequencies and the second column is the spectrum evaluated at that frequency

---

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 = 500, nburn = 500)

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

---

t\_create\_loc

*Initializer for location indices for filling in band-sparse matrix*


---

**Description**

Create row and column indices for locations of symmetric band-sparse matrix. Starts with the locations of the diagonal, proceed with upper-diagonals, followed by lower-diagonals.

**Usage**

```
t_create_loc(len, D)
```

**Arguments**

len	length of the diagonal of the band-sparse matrix
D	number of super-diagonals to include for the band-sparse

**Value**

a list containing

- the row indices r and
- the column indices c

---

t\_initEvolParams\_no     *Initialize the evolution error variance parameters*

---

**Description**

Compute initial values for evolution error variance parameters under the dynamic horseshoe prior

**Usage**

t\_initEvolParams\_no(y, D, omega)

**Arguments**

y	the T vector of time series observations
D	degree of differencing (D = 1, or D = 2)
omega	T vector of evolution errors

**Value**

List of relevant components: sigma\_wt, the T vector of evolution standard deviations, and additional parameters associated with the DHS priors.

---

t\_initEvolZeta\_ps     *Initialize the anomaly component parameters*

---

**Description**

Compute initial values for either a horseshoe prior or horseshoe+ prior for the anomaly component.

**Usage**

t\_initEvolZeta\_ps(zeta)

**Arguments**

zeta	T vector of initial estimates.
------	--------------------------------

**Value**

List of relevant components: sigma\_wt, the T vector of standard deviations, and additional parameters for inverse gamma priors (shape and scale).

---

t_initSV	<i>Initialize the stochastic volatility parameters</i>
----------	--

---

**Description**

Compute initial values for normal stochastic volatility parameters. The model assumes an AR(1) for the log-volatility.

**Usage**

```
t_initSV(omega)
```

**Arguments**

omega	T vector of errors
-------	--------------------

**Value**

List of relevant components: sigma\_wt, the T vector of standard deviations, and additional parameters (unconditional mean, AR(1) coefficient, and standard deviation).

---

t_sampleAR1	<i>Sample the TAR(1) coefficients</i>
-------------	---------------------------------------

---

**Description**

Compute one draw of the TAR(1) coefficients in a model with Gaussian innovations and time-dependent innovation variances. In particular, we use the sampler for the log-volatility TAR(1) process with the parameter-expanded Polya-Gamma sampler. The sampler also applies to a multivariate case with independent components.

**Usage**

```
t_sampleAR1(h_yc, h_phi, h_phi2, h_sigma_eta_t, h_st, prior_dhs_phi = NULL)
```

**Arguments**

h_yc	the T vector of centered log-volatilities (i.e., the log-vols minus the unconditional means dhs_mean)
h_phi	the 1 vector of previous AR(1) coefficient(s)
h_phi2	the 1 vector of previous penalty coefficient(s)
h_sigma_eta_t	the T vector of log-vol innovation standard deviations
h_st	the T vector of indicators on whether each time-step exceed the estimated threshold
prior_dhs_phi	the parameters of the prior for the log-volatility AR(1) coefficient dhs_phi; either NULL for uniform on [-1,1] or a 2-dimensional vector of (shape1, shape2) for a Beta prior on [(dhs_phi + 1)/2]

**Value**

2 vector of sampled TAR(1) coefficient(s)

---

t_sampleBTF	<i>Sampler for first or second order random walk (RW) Gaussian dynamic linear model (DLM)</i>
-------------	---

---

**Description**

Compute one draw of the T state variable  $\mu$  in a DLM using back-band substitution methods. This model is equivalent to the Bayesian trend filtering (BTF) model, assuming appropriate (shrinkage/sparsity) priors for the evolution errors.

**Usage**

```
t_sampleBTF(y, obs_sigma_t2, evol_sigma_t2, D = 1, loc_obs)
```

**Arguments**

y	the T x 1 vector of time series observations
obs_sigma_t2	the T x 1 vector of observation error variances
evol_sigma_t2	the T x 1 vector of evolution error variances
D	the degree of differencing (one or two)
loc_obs	list of the row and column indices to fill in a band-sparse matrix

**Value**

T x 1 vector of simulated states

**Note**

Missing entries (NAs) are not permitted in y. Imputation schemes are available.

---

t\_sampleEvolParams      *Sample the thresholded dynamic shrinkage process parameters*

---

### Description

Compute one draw for each of the parameters in the thresholded dynamic shrinkage process for the special case in which the shrinkage parameter  $\kappa \sim \text{Beta}(\alpha, \beta)$  with  $\alpha = \beta = 1/2$ .

### Usage

```
t_sampleEvolParams(
  omega,
  evolParams,
  D = 1,
  sigma_e = 1,
  lower_b,
  upper_b,
  loc,
  prior_dhs_phi = c(20, 1),
  alphaPlusBeta = 1
)
```

### Arguments

omega	T vector of evolution errors
evolParams	list of parameters to be updated (see Value below)
D	the degree of differencing (one or two)
sigma_e	the observation error standard deviation; for (optional) scaling purposes
lower_b	the lower bound in the uniform prior of the threshold variable
upper_b	the upper bound in the uniform prior of the threshold variable
loc	list of the row and column indices to fill in a band-sparse matrix
prior_dhs_phi	the parameters of the prior for the log-volatility AR(1) coefficient dhs_phi; either NULL for uniform on [-1,1] or a 2-dimensional vector of (shape1, shape2) for a Beta prior on [(dhs_phi + 1)/2]
alphaPlusBeta	For the symmetric prior $\kappa \sim \text{Beta}(\alpha, \beta)$ with $\alpha = \beta$ , specify the sum [alpha + beta]

### Value

List of relevant components:

- the T evolution error standard deviations sigma\_wt,
- the T log-volatility ht,

- the 1 log-vol unconditional mean(s) dhs\_mean,
- the 1 log-vol AR(1) coefficient(s) dhs\_phi,
- the 1 log-vol correction coefficient(s) dhs\_phi2,
- the T log-vol innovation standard deviations sigma\_eta\_t from the Polya-Gamma priors,
- the 1 initial log-vol SD sigma\_eta\_0,
- the 1 threshold parameter r

**Note**

The priors induced by prior\_dhs\_phi all imply a stationary (log-) volatility process.

---

t\_sampleEvolZeta\_ps     *Sampler for the anomaly component parameters*

---

**Description**

Compute one draw of the anomaly component parameters.

**Usage**

```
t_sampleEvolZeta_ps(omega, evolParams)
```

**Arguments**

omega	T vector of errors
evolParams	list of parameters to be updated

**Value**

List of relevant components in evolParams: sigma\_wt, the T vector of standard deviations, and additional parameters for inverse gamma priors (shape and scale).

---

t\_sampleLogVolMu     *Sample the TAR(1) unconditional means*

---

**Description**

Compute one draw of the unconditional means in an TAR(1) model with Gaussian innovations and time-dependent innovation variances. In particular, we use the sampler for the log-volatility TAR(1) process with the parameter-expanded Polya-Gamma sampler. The sampler also applies to a multivariate case with independent components.

**Usage**

```
t_sampleLogVolMu(
  h,
  h_mu,
  h_phi,
  h_phi2,
  h_sigma_eta_t,
  h_sigma_eta_0,
  h_st,
  h_log_scale = 0
)
```

**Arguments**

h	the T vector of log-volatilities
h_mu	the 1 vector of previous means
h_phi	the 1 vector of AR(1) coefficient(s)
h_phi2	the 1 vector of previous penalty coefficient(s)
h_sigma_eta_t	the T vector of log-vol innovation standard deviations
h_sigma_eta_0	the standard deviations of initial log-vols
h_st	the T vector of indicators on whether each time-step exceed the estimated threshold
h_log_scale	prior mean from scale mixture of Gaussian (Polya-Gamma) prior, e.g. $\log(\sigma_e^2)$ or dhs_mean0

**Value**

the sampled mean(s) dhs\_mean

---

t_sampleLogVols	<i>Sample the latent log-volatilities</i>
-----------------	---

---

**Description**

Compute one draw of the log-volatilities using a discrete mixture of Gaussians approximation to the likelihood (see Omori, Chib, Shephard, and Nakajima, 2007) where the log-vols are assumed to follow an TAR(1) model with time-dependent innovation variances. More generally, the code operates for  $p$  independent TAR(1) log-vol processes to produce an efficient joint sampler in  $O(Tp)$  time.

**Usage**

```
t_sampleLogVols(
  h_y,
  h_prev,
  h_mu,
  h_phi,
  h_phi2,
  h_sigma_eta_t,
  h_sigma_eta_0,
  h_st,
  loc
)
```

**Arguments**

h_y	the T vector of data, which follow independent SV models
h_prev	the T vector of the previous log-vols
h_mu	the 1 vector of log-vol unconditional means
h_phi	the 1 vector of log-vol AR(1) coefficients
h_phi2	the 1 vector of previous penalty coefficient(s)
h_sigma_eta_t	the T vector of log-vol innovation standard deviations
h_sigma_eta_0	the 1 vector of initial log-vol innovation standard deviations
h_st	the T vector of indicators on whether each time-step exceed the estimated threshold
loc	list of the row and column indices to fill in the band-sparse matrix in the sampler

**Value**

T x p vector of simulated log-vols

**Note**

For Bayesian trend filtering,  $p = 1$ . More generally, the sampler allows for  $p > 1$  but assumes (contemporaneous) independence across the log-vols for  $j = 1, \dots, p$ .

---

t\_sampleR\_mh

*Sample the threshold parameter*


---

**Description**

Compute one draw of the threshold parameter in the TAR(1) model with Gaussian innovations and time-dependent innovation variances. The sampler utilizes metropolis hasting to draw from uniform prior.

**Usage**

```
t_sampleR_mh(
  h_yc,
  h_phi,
  h_phi2,
  h_sigma_eta_t,
  h_sigma_eta_0,
  h_st,
  h_r,
  lower_b,
  upper_b,
  omega,
  D
)
```

**Arguments**

h_yc	the T vector of centered log-volatilities (i.e., the log-vols minus the unconditional means dhs_mean)
h_phi	the 1 vector of previous AR(1) coefficient(s)
h_phi2	the 1 vector of previous penalty coefficient(s)
h_sigma_eta_t	the T vector of log-vol innovation standard deviations
h_sigma_eta_0	the 1 vector of initial log-vol innovation standard deviations
h_st	the T vector of indicators on whether each time-step exceed the estimated threshold
h_r	1 the previous draw of the threshold parameter
lower_b	the lower bound in the uniform prior of the threshold variable
upper_b	the upper bound in the uniform prior of the threshold variable
omega	T vector of evolution errors
D	the degree of differencing (one or two)

**Value**

the sampled threshold value r

---

t\_sampleSVparams

*Sampler for the stochastic volatility parameters*


---

**Description**

Compute one draw of the normal stochastic volatility parameters. The model assumes an AR(1) for the log-volatility.

**Usage**

```
t_sampleSVparams(omega, svParams)
```

**Arguments**

omega	T vector of errors
svParams	list of parameters to be updated

**Value**

List of relevant components in svParams: sigma\_wt, the T vector of standard deviations, and additional parameters associated with SV model.

---

uni.slice

*Univariate Slice Sampler from Neal (2008)*


---

**Description**

Compute a draw from a univariate distribution using the code provided by Radford M. Neal. The documentation below is also reproduced from Neal (2008).

**Usage**

```
uni.slice(x0, g, w = 1, m = Inf, lower = -Inf, upper = +Inf, gx0 = NULL)
```

**Arguments**

x0	Initial point
g	Function returning the log of the probability density (plus constant)
w	Size of the steps for creating interval (default 1)
m	Limit on steps (default infinite)
lower	Lower bound on support of the distribution (default -Inf)
upper	Upper bound on support of the distribution (default +Inf)
gx0	Value of g(x0), if known (default is not known)

**Value**

The point sampled, with its log density attached as an attribute.

**Note**

The log density function may return -Inf for points outside the support of the distribution. If a lower and/or upper bound is specified for the support, the log density function will not be called outside such limits.

# Index

abco, 4

btf, 5  
btf0, 7  
btf\_bspline, 9  
btf\_bspline0, 11  
btf\_reg, 12  
btf\_sparse, 14  
build\_Q, 17  
build\_XtX, 17

computeDIC\_ASV, 18  
credBands, 18

dsp\_fit, 19  
dsp\_fit(), 20, 34, 37, 55  
dsp\_spec, 21  
dsp\_spec(), 19, 22

ergMean, 22

fit\_ASV, 23  
fit\_paramsASV, 24  
fit\_paramsASV\_n, 25

generate\_ly2hat, 26  
getARpxmat, 26  
getEffSize, 27  
getNonZeros, 27  
graphics::hist(), 35  
graphics::par(), 35  
graphics::plot(), 35

init\_paramsASV, 31  
init\_paramsASV\_n, 32  
initChol\_spam, 28  
initCholReg\_spam, 28  
initDHS, 29  
initEvol0, 29  
initEvolParams, 30  
initSV, 30

invlogit, 32

logit, 33

ncind, 33

plot.dsp, 34  
predict.dsp, 37  
print.dsp(dsp\_fit), 19  
print.dsp\_spec(dsp\_spec), 21

quantile(), 55

sample\_j\_wrap, 50  
sample\_mat\_c, 50  
sampleAR1, 38  
sampleBTF, 39  
sampleBTF\_bspline, 40  
sampleBTF\_reg, 41  
sampleBTF\_reg\_backfit, 41  
sampleBTF\_sparse, 42  
sampleDSP, 43  
sampleEvol0, 44  
sampleEvolParams, 45  
sampleFastGaussian, 46  
sampleLogVolMu, 46  
sampleLogVolMu0, 47  
sampleLogVols, 48  
sampleSVparams, 49  
sampleSVparams0, 49  
simBaS, 51  
simRegression, 52  
simRegression0, 53  
simUnivariate, 54  
spec\_dsp, 55  
summary.dsp, 55

t\_create\_loc, 56  
t\_initEvolParams\_no, 57  
t\_initEvolZeta\_ps, 57  
t\_initSV, 58

t\_sampleAR1, [58](#)  
t\_sampleBTF, [59](#)  
t\_sampleEvolParams, [60](#)  
t\_sampleEvolZeta\_ps, [61](#)  
t\_sampleLogVolMu, [61](#)  
t\_sampleLogVols, [62](#)  
t\_sampleR\_mh, [63](#)  
t\_sampleSVparams, [64](#)  
  
uni.slice, [65](#)