Package ‘beyondWhittle’

March 13, 2024

Type Package

Title Bayesian Spectral Inference for Time Series

Version 1.2.1

Date 2024-03-12

Maintainer Renate Meyer <renate.meyer@auckland.ac.nz>

Description Implementations of Bayesian parametric, nonparametric and semiparametric procedures for univariate and multivariate time series. The package is based on the methods presented in C. Kirch et al (2018) <doi:10.1214/18-BA1126>, A. Meier (2018) <https://opendata.uni-halle.de//handle/1981185920/13470> and Y. Tang et al (2023) <arXiv:2303.11561>. It was supported by DFG grants KI 1443/3-1 and KI 1443/3-2.

License GPL (>= 3)

Imports ltsa (>= 1.4.6), Rcpp (>= 0.12.5), MASS, forecast

LinkingTo Rcpp, RcppArmadillo, BH

RoxygenNote 7.2.2

NeedsCompilation yes

Encoding UTF-8

Author Alexander Meier [aut],
Claudia Kirch [aut],
Matthew C. Edwards [aut],
Renate Meyer [aut, cre],
Yifu Tang [aut]

Repository CRAN

Date/Publication 2024-03-13 19:30:02 UTC

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

R topics documented:

`beyondWhittle-package` *Bayesian spectral inference for time series*

Description

Bayesian parametric, nonparametric and semiparametric procedures for spectral density inference of univariate (locally) stationary time series and multivariate stationary time series

Details

The package contains several methods (parametric, nonparametric and semiparametric) for Bayesian spectral density inference. The main algorithms to fit the models for univariate stationary time series are:

- gibbs_ar: Parametric, autoregressive (AR) model

- gibbs_np: Nonparametric model with Whittle's likelihood and Bernstein-Dirichlet prior from Choudhuri et al (2007)
- gibbs_npc: Semiparametric model with corrected AR likelihood and Bernstein-Dirichlet prior from Kirch et al (2018)

The package also contains the following models for multivariate stationary time series:

- gibbs_var: Parametric, vector autoregressive (VAR) model
- gibbs_vnp: Nonparametric model with Whittle's likelihood and Bernstein-Hpd-Gamma prior from Meier (2018)

The main function for univariate locally stationary time series is:

- gibbs_bdp_dw: Nonparametric model with BDP-DW approach from Tang et al (2023)

as well as some useful utility functions. To get started, it is recommended to consider the examples and documentation of the functions listed above. The work was supported by DFG grants KI 1443/3-1 and KI 1443/3-2.

Author(s)

Claudia Kirch, Renate Meyer, Matthew C. Edwards, Alexander Meier, Yifu Tang

Maintainer: Renate Meyer <renate.meyer@auckland.ac.nz>

References

N. Choudhuri, S. Ghosal and A. Roy (2004) *Bayesian estimation of the spectral density of a time series* JASA <doi:10.1198/016214504000000557>

C. Kirch, M. C. Edwards, A. Meier and R. Meyer (2018) *Beyond Whittle: Nonparametric Correction of a Parametric Likelihood with a Focus on Bayesian Time Series Analysis* Bayesian Analysis <doi:10.1214/18-BA1126>

A. Meier (2018) *A matrix Gamma process and applications to Bayesian analysis of multivariate time series* PhD thesis, OvGU Magdeburg <doi:10.25673/13407>

Y. Tang, C. Kirch, J. E. Lee and R. Meyer (2023) *Bayesian nonparametric spectral analysis of locally stationary processes* ArXiv preprint <arXiv:2303.11561>

`bayes_factor` *a generic method for bdp_dw_result class*

Description

a generic method for bdp_dw_result class

Usage

```
bayes_factor(obj)
```

Arguments

`obj`	object of class bdp_dw_result

`bayes_factor.bdp_dw_result` *Extracting the Bayes factor of k1=1 from bdp_dw_result class*

Description

Extracting the Bayes factor of k1=1 from bdp_dw_result class

Usage

```
## S3 method for class 'bdp_dw_result'
bayes_factor(obj)
```

Arguments

`obj`	object of class bdp_dw_result

Value

the estimated Bayes factor of k1=1

`bdp_dw_bayes_factor_k1` *Estimating the Bayes factor of hypothesis "k1 = 1".*

Description

Estimating the Bayes factor of hypothesis "k1 = 1".

Usage

```
bdp_dw_bayes_factor_k1(post_sample, precision = 1000)
```

Arguments

`post_sample`	the posterior sample generated by bdp_dw_mcmc.
`precision`	a positive integer specifying the number of terms used in approximating the normalizing constant of the prior probability mass function of k1. Default 1000.

Value

The Savage-Dickey estimate of the Bayes factor and its theoretical upper bound. c.f. section 3.3 of Tang et al. (2023).

References

Tang et al. (2023) *Bayesian nonparametric spectral analysis of locally stationary processes* ArXiv preprint <arXiv:2303.11561>

bdp_dw_est_post_stats *Calculating the estimated posterior mean, median and credible region (tv-PSD)*

Description

Calculating the estimated posterior mean, median and credible region (tv-PSD)

Usage

```
bdp_dw_est_post_stats(post_sample, rescaled_time, freq, unif_CR = FALSE)
```

Arguments

`post_sample` the posterior sample generated by bdp_dw_mcmc.

`rescaled_time, freq`
numeric vectors forming a rectangular grid on which the estimated tv-PSD is evaluated.

`unif_CR` a Boolean value (default FALSE) indicating whether to calculate the uniform credible region rescaled_time must be in $[0, 1]$ and freq must be in $[0, \pi]$.

Value

list containing the following fields:

`tvpsd.mean,tvpsd.median`
posterior mean and pointwise posterior median (matrices of dimension length(rescaled_time) by length(freq))

`tvpsd.p05,tvpsd.p95`
90 percent pointwise credibility interval

`tvpsd.u05,tvpsd.u95`
90 percent uniform credibility interval if unif_CR = TRUE. Otherwise NA

bdp_dw_mcmc_params_gen *Generate a list of values for MCMC algorithm*

Description

Generate a list of values for MCMC algorithm

Usage

```
bdp_dw_mcmc_params_gen(
  Ntotal = 110000,
  burnin = 60000,
  thin = 10,
  adaptive.batchSize = 50,
  adaptive.targetAcceptanceRate = 0.44
)
```

Arguments

`Ntotal`	total number of iterations to run the Markov chain
`burnin`	number of initial iterations to be discarded
`thin`	thinning number (for post-processing of the posterior sample)
`adaptive.batchSize`	the batch size for the adaptive MCMC algorithm for sampling tau
`adaptive.targetAcceptanceRate`	the target acceptance rate for the adaptive MCMC algorithm for sampling tau

Value

A list of MCMC parameter values

---

bdp_dw_prior_params_gen *Generate a list of parameter values in prior elicitation*

Description

Generate a list of parameter values in prior elicitation

Usage

```
bdp_dw_prior_params_gen(
  M = 1,
  g0.alpha = 1,
  g0.beta = 1,
  k1.theta = 0.01,
  k2.theta = 0.01,
  tau.alpha = 0.001,
  tau.beta = 0.001,
  k1max = 100,
  k2max = 100,
  L = 20,
  bernstein1_l = 0.1,
  bernstein1_r = 0.9,
```

```
  bernstein2_l = 0.1,
  bernstein2_r = 0.9
)
```

Arguments

`M`	DP base measure constant (> 0)
`g0.alpha, g0.beta`	parameters of Beta base measure of DP
`k1.theta`	prior parameter for polynomial corresponding to rescaled time (propto exp(-k1.theta*k1*log(k1)))
`k2.theta`	prior parameter for polynomial corresponding to rescaled frequency (propto exp(-k2.theta*k2*log(k2)))
`tau.alpha, tau.beta`	prior parameters for tau (inverse gamma)
`k1max`	upper bound of the degrees of Bernstein polynomial corresponding to rescaled time (for pre-computation of basis functions)
`k2max`	upper bound of the degrees of Bernstein polynomial corresponding to rescaled frequency (for pre-computation of basis functions)
`L`	truncation parameter of DP in stick breaking representation
`bernstein1_l, bernstein1_r`	left and right truncation of Bernstein polynomial basis functions for rescaled time, 0<=bernstein1_l<bernstein1_r<=1
`bernstein2_l, bernstein2_r`	left and right truncation of Bernstein polynomial basis functions for rescaled frequency, 0<=bernstein2_l<bernstein2_r<=1

Value

A list of prior parameter values

`fourier_freq` *Fourier frequencies*

Description

Fourier frequencies on [0,pi], as defined by 2*pi*j/n for j=0,...,floor(n/2).

Usage

```
fourier_freq(n)
```

Arguments

`n`	integer

Value

numeric vector of length floor(n/2)+1

`gibbs_ar` *Gibbs sampler for an autoregressive model with PACF parametrization.*

Description

Obtain samples of the posterior of a Bayesian autoregressive model of fixed order.

Usage

```
gibbs_ar(
  data,
  ar.order,
  Ntotal,
  burnin,
  thin = 1,
  print_interval = 500,
  numerical_thresh = 1e-07,
  adaption.N = burnin,
  adaption.batchSize = 50,
  adaption.tar = 0.44,
  full_lik = F,
  rho.alpha = rep(1, ar.order),
  rho.beta = rep(1, ar.order),
  sigma2.alpha = 0.001,
  sigma2.beta = 0.001
)
```

Arguments

`data`	numeric vector; NA values are interpreted as missing values and treated as random
`ar.order`	order of the autoregressive model (integer >= 0)
`Ntotal`	total number of iterations to run the Markov chain
`burnin`	number of initial iterations to be discarded
`thin`	thinning number (postprocessing)
`print_interval`	Number of iterations, after which a status is printed to console
`numerical_thresh`	Lower (numerical pointwise) bound for the spectral density
`adaption.N`	total number of iterations, in which the proposal variances (of rho) are adapted
`adaption.batchSize`	batch size of proposal adaption for the rho_i's (PACF)

`adaption.tar` target acceptance rate for the rho_i's (PACF)

`full_lik` logical; if TRUE, the full likelihood for all observations is used; if FALSE, the partial likelihood for the last n-p observations

`rho.alpha, rho.beta`
prior parameters for the rho_i's: 2*(rho-0.5)~Beta(rho.alpha,rho.beta), default is Uniform(-1,1)

`sigma2.alpha, sigma2.beta`
prior parameters for sigma2 (inverse gamma)

Details

Partial Autocorrelation Structure (PACF, uniform prior) and the residual variance sigma2 (inverse gamma prior) is used as model parametrization. The DIC is computed with two times the posterior variance of the deviance as effective number of parameters, see (7.10) in the referenced book by Gelman et al. Further details can be found in the simulation study section in the referenced paper by C. Kirch et al. For more information on the PACF parametrization, see the referenced paper by Barndorff-Nielsen and Schou.

Value

list containing the following fields:

`rho` matrix containing traces of the PACF parameters (if p>0)

`sigma2` trace of sigma2

`DIC` a list containing the numeric value `DIC` of the Deviance Information Criterion (DIC) and the effective number of parameters `ENP`

`psd.median,psd.mean`
psd estimates: (pointwise) posterior median and mean

`psd.p05,psd.p95`
pointwise credibility interval

`psd.u05,psd.u95`
uniform credibility interval

`lpost` trace of log posterior

References

C. Kirch et al. (2018) *Beyond Whittle: Nonparametric Correction of a Parametric Likelihood With a Focus on Bayesian Time Series Analysis* Bayesian Analysis <doi:10.1214/18-BA1126>

A. Gelman et al. (2013) *Bayesian Data Analysis, Third Edition*

O. Barndorff-Nielsen and G. Schou On the parametrization of autoregressive models by partial autocorrelations Journal of Multivariate Analysis (3),408-419 <doi:10.1016/0047-259X(73)90030-4>

Examples

```
## Not run:

##
## Example 1: Fit an AR(p) model to sunspot data:
##

# Use this variable to set the AR model order
p <- 2

data <- sqrt(as.numeric(sunspot.year))
data <- data - mean(data)

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_ar(data=data, ar.order=p, Ntotal=10000, burnin=4000, thin=2)

# Plot spectral estimate, credible regions and periodogram on log-scale
plot(mcmc, log=T)


##
## Example 2: Fit an AR(p) model to high-peaked AR(1) data
##

# Use this variable to set the AR model order
p <- 1

n <- 256
data <- arima.sim(n=n, model=list(ar=0.95))
data <- data - mean(data)
omega <- fourier_freq(n)
psd_true <- psd_arma(omega, ar=0.95, ma=numeric(0), sigma2=1)

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_ar(data=data, ar.order=p, Ntotal=10000, burnin=4000, thin=2)

# Compare estimate with true function (green)
plot(mcmc, log=F, pdgrm=F, credib="uniform")
lines(x=omega, y=psd_true, col=3, lwd=2)

# Compute the Integrated Absolute Error (IAE) of posterior median
cat("IAE=", mean(abs(mcmc$psd.median-psd_true)[-1]) , sep="")

## End(Not run)
```

---

gibbs_bdp_dw *BDP-DW method: performing posterior sampling and calculating statistics based on the posterior samples*

Description

BDP-DW method: performing posterior sampling and calculating statistics based on the posterior samples

Usage

```
gibbs_bdp_dw(
  data,
  m,
  likelihood_thinning = 1,
  monitor = TRUE,
  print_interval = 100,
  unif_CR = FALSE,
  res_time,
  freq,
  Ntotal = 110000,
  burnin = 60000,
  thin = 10,
  adaptive.batchSize = 50,
  adaptive.targetAcceptanceRate = 0.44,
  M = 1,
  g0.alpha = 1,
  g0.beta = 1,
  k1.theta = 0.01,
  k2.theta = 0.01,
  tau.alpha = 0.001,
  tau.beta = 0.001,
  k1max = 100,
  k2max = 100,
  L = 20,
  bernstein1_l = 0.1,
  bernstein1_r = 0.9,
  bernstein2_l = 0.1,
  bernstein2_r = 0.9
)
```

Arguments

`data`	time series that needs to be analyzed
`m`	window size needed to calculate moving periodogram.
`likelihood_thinning`	the thinning factor of the dynamic Whittle likelihood.
`monitor`	a Boolean value (default TRUE) indicating whether to display the real-time status
`print_interval`	If monitor = TRUE, then this value indicates number of iterations after which a status is printed to console; If monitor = FALSE, it does not have any effect

`unif_CR` a Boolean value (default FALSE) indicating whether to calculate the uniform credible region

`res_time, freq` a set of grid lines in $[0, 1]$ and $[0, \pi]$, respectively, specifying where to evaluate the estimated tv-PSD

`Ntotal` total number of iterations to run the Markov chain

`burnin` number of initial iterations to be discarded

`thin` thinning number (for post-processing of the posterior sample)

`adaptive.batchSize` the batch size for the adaptive MCMC algorithm for sampling tau

`adaptive.targetAcceptanceRate` the target acceptance rate for the adaptive MCMC algorithm for sampling tau

`M` DP base measure constant (> 0)

`g0.alpha, g0.beta` parameters of Beta base measure of DP

`k1.theta` prior parameter for polynomial corresponding to rescaled time (propto exp(-k1.theta*k1*log(k1)))

`k2.theta` prior parameter for polynomial corresponding to rescaled frequency (propto exp(-k2.theta*k2*log(k2)))

`tau.alpha, tau.beta` prior parameters for tau (inverse gamma)

`k1max` upper bound of the degrees of Bernstein polynomial corresponding to rescaled time (for pre-computation of basis functions)

`k2max` upper bound of the degrees of Bernstein polynomial corresponding to rescaled frequency (for pre-computation of basis functions)

`L` truncation parameter of DP in stick breaking representation

`bernstein1_l, bernstein1_r` left and right truncation of Bernstein polynomial basis functions for rescaled time, 0<=bernstein1_l<bernstein1_r<=1

`bernstein2_l, bernstein2_r` left and right truncation of Bernstein polynomial basis functions for rescaled frequency, 0<=bernstein2_l<bernstein2_r<=1

Value

list containing the following fields:

`k1,k2,tau,V,W1,W2` posterior traces of PSD parameters

`lpost` traces log posterior

`tim` total run time

`bf_k1` Savage-Dickey estimate of Bayes factor of hypothesis k1=1

`tvpsd.mean,tvpsd.median` posterior mean and pointwise posterior median (matrices of dimension length(rescaled_time) by length(freq))

`tvpsd.p05,tvpsd.p95`
: 90 percent pointwise credibility interval

`tvpsd.u05,tvpsd.u95`
: 90 percent uniform credibility interval if unif_CR = TRUE. Otherwise NA

References

Tang et al. (2023) *Bayesian nonparametric spectral analysis of locally stationary processes* ArXiv preprint <arXiv:2303.11561>

Examples

```
## Not run:

##
## Example: Applying BDP-DW method to a multi-peaked tvMA(1) process
##

# set seed
set.seed(200)
# set the length of time series
len_d <- 1500
# generate data from DGP LS2c defined in Section 4.2.2 of Tang et al. (2023).
# see also ?sim_tvarma12
sim_data <- sim_tvarma12(len_d = 1500, dgp = "LS2", innov_distribution = "c")
# specify grid-points at which the tv-PSD is evaluated
res_time <- seq(0, 1, by = 0.005); freq <- pi * seq(0, 1, by = 0.01)
# calculate the true tv-PSD of DGP LS2c at the pre-specified grid
true_tvPSD <- psd_tvarma12(rescaled_time = res_time, freq = freq, dgp = "LS2")
# plot the true tv-PSD
# type ?plot.bdp_dw_tv_psd for more info
plot(true_tvPSD)

# If you run the example be aware that this may take several minutes
print("This example may take some time to run")
result <- gibbs_bdp_dw(data = sim_data,
m = 50,
likelihood_thinning = 2,
rescaled_time = res_time,
 freq = freq)

# extract bayes factor and examine posterior summary
bayes_factor(result)
summary(result)

# compare estimate with true function
# type ?plot.bdp_dw_result for more info
par(mfrow = c(1,2))

plot(result, which = 1,
zlim = range(result$tvpsd.mean, true_tvPSD$tv_psd)
)
```

```
plot(true_tvPSD,
zlim = range(result$tvpsd.mean, true_tvPSD$tv_psd),
main = "true tv-PSD")

par(mfrow = c(1,1))


## End(Not run)
```

---

gibbs_np *Gibbs sampler for Bayesian nonparametric inference with Whittle likelihood*

Description

Obtain samples of the posterior of the Whittle likelihood in conjunction with a Bernstein-Dirichlet prior on the spectral density.

Usage

```
gibbs_np(
  data,
  Ntotal,
  burnin,
  thin = 1,
  print_interval = 100,
  numerical_thresh = 1e-07,
  M = 1,
  g0.alpha = 1,
  g0.beta = 1,
  k.theta = 0.01,
  tau.alpha = 0.001,
  tau.beta = 0.001,
  kmax = 100 * coars + 500 * (!coars),
  trunc_l = 0.1,
  trunc_r = 0.9,
  coars = F,
  L = max(20, length(data)^(1/3))
)
```

Arguments

`data`	numeric vector; NA values are interpreted as missing values and treated as random
`Ntotal`	total number of iterations to run the Markov chain
`burnin`	number of initial iterations to be discarded
`thin`	thinning number (postprocessing)

`print_interval` Number of iterations, after which a status is printed to console

`numerical_thresh`
: Lower (numerical pointwise) bound for the spectral density

`M` DP base measure constant (> 0)

`g0.alpha, g0.beta`
: parameters of Beta base measure of DP

`k.theta` prior parameter for polynomial degree k (propto exp(-k.theta*k*log(k)))

`tau.alpha, tau.beta`
: prior parameters for tau (inverse gamma)

`kmax` upper bound for polynomial degree of Bernstein-Dirichlet mixture (can be set to Inf, algorithm is faster with kmax<Inf due to pre-computation of basis functions, but values 500<kmax<Inf are very memory intensive)

`trunc_l, trunc_r`
: left and right truncation of Bernstein polynomial basis functions, 0<=trunc_l<trunc_r<=1

`coars` flag indicating whether coarsened or default bernstein polynomials are used (see Appendix E.1 in Ghosal and van der Vaart 2017)

`L` truncation parameter of DP in stick breaking representation

Details

Further details can be found in the simulation study section in the references papers.

Value

list containing the following fields:

`psd.median,psd.mean`
: psd estimates: (pointwise) posterior median and mean

`psd.p05,psd.p95`
: pointwise credibility interval

`psd.u05,psd.u95`
: uniform credibility interval

`k,tau,V,W` posterior traces of PSD parameters

`lpost` trace of log posterior

References

C. Kirch et al. (2018) *Beyond Whittle: Nonparametric Correction of a Parametric Likelihood With a Focus on Bayesian Time Series Analysis* Bayesian Analysis <doi:10.1214/18-BA1126>

N. Choudhuri et al. (2004) *Bayesian Estimation of the Spectral Density of a Time Series* JASA <doi:10.1198/016214504000000557>

S. Ghosal and A. van der Vaart (2017) *Fundamentals of Nonparametric Bayesian Inference* <doi:10.1017/9781139029834>

Examples

```
## Not run:

##
## Example 1: Fit the NP model to sunspot data:
##

data <- sqrt(as.numeric(sunspot.year))
data <- data - mean(data)

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_np(data=data, Ntotal=10000, burnin=4000, thin=2)

# Plot spectral estimate, credible regions and periodogram on log-scale
plot(mcmc, log=T)


##
## Example 2: Fit the NP model to high-peaked AR(1) data
##

n <- 256
data <- arima.sim(n=n, model=list(ar=0.95))
data <- data - mean(data)
omega <- fourier_freq(n)
psd_true <- psd_arma(omega, ar=0.95, ma=numeric(0), sigma2=1)

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_np(data=data, Ntotal=10000, burnin=4000, thin=2)

# Compare estimate with true function (green)
plot(mcmc, log=F, pdgrm=F, credib="uniform")
lines(x=omega, y=psd_true, col=3, lwd=2)

# Compute the Integrated Absolute Error (IAE) of posterior median
cat("IAE=", mean(abs(mcmc$psd.median-psd_true)[-1]) , sep="")

## End(Not run)
```

---

gibbs_npc — *Gibbs sampler for Bayesian semiparametric inference with the corrected AR likelihood*

Description

Obtain samples of the posterior of the corrected autoregressive likelihood in conjunction with a Bernstein-Dirichlet prior on the correction.

Usage

```
gibbs_npc(
  data,
  ar.order,
  Ntotal,
  burnin,
  thin = 1,
  print_interval = 100,
  numerical_thresh = 1e-07,
  adaption.N = burnin,
  adaption.batchSize = 50,
  adaption.tar = 0.44,
  full_lik = F,
  rho.alpha = rep(1, ar.order),
  rho.beta = rep(1, ar.order),
  eta = T,
  M = 1,
  g0.alpha = 1,
  g0.beta = 1,
  k.theta = 0.01,
  tau.alpha = 0.001,
  tau.beta = 0.001,
  trunc_l = 0.1,
  trunc_r = 0.9,
  coars = F,
  kmax = 100 * coars + 500 * (!coars),
  L = max(20, length(data)^(1/3))
)
```

Arguments

- `data` numeric vector; NA values are interpreted as missing values and treated as random
- `ar.order` order of the autoregressive model (integer > 0)
- `Ntotal` total number of iterations to run the Markov chain
- `burnin` number of initial iterations to be discarded
- `thin` thinning number (postprocessing)
- `print_interval` Number of iterations, after which a status is printed to console
- `numerical_thresh` Lower (numerical pointwise) bound for the spectral density
- `adaption.N` total number of iterations, in which the proposal variances (of rho) are adapted
- `adaption.batchSize` batch size of proposal adaption for the rho_i's (PACF)
- `adaption.tar` target acceptance rate for the rho_i's (PACF)
- `full_lik` logical; if TRUE, the full likelihood for all observations is used; if FALSE, the partial likelihood for the last n-p observations

`rho.alpha, rho.beta`
: prior parameters for the rho_i's: 2*(rho-0.5)~Beta(rho.alpha,rho.beta), default is Uniform(-1,1)

`eta`
: logical variable indicating whether the model confidence eta should be included in the inference (eta=T) or fixed to 1 (eta=F)

`M`
: DP base measure constant (> 0)

`g0.alpha, g0.beta`
: parameters of Beta base measure of DP

`k.theta`
: prior parameter for polynomial degree k (propto exp(-k.theta*k*log(k)))

`tau.alpha, tau.beta`
: prior parameters for tau (inverse gamma)

`trunc_l, trunc_r`
: left and right truncation of Bernstein polynomial basis functions, 0<=trunc_l<trunc_r<=1

`coars`
: flag indicating whether coarsened or default bernstein polynomials are used (see Appendix E.1 in Ghosal and van der Vaart 2017)

`kmax`
: upper bound for polynomial degree of Bernstein-Dirichlet mixture (can be set to Inf, algorithm is faster with kmax<Inf due to pre-computation of basis functions, but values 500<kmax<Inf are very memory intensive)

`L`
: truncation parameter of DP in stick breaking representation

Details

Partial Autocorrelation Structure (PACF, uniform prior) and the residual variance sigma2 (inverse gamma prior) is used as model parametrization. A Bernstein-Dirichlet prior for c_eta with base measure Beta(g0.alpha, g0.beta) is used. Further details can be found in the simulation study section in the referenced paper by Kirch et al. For more information on the PACF parametrization, see the referenced paper by Barndorff-Nielsen and Schou.

Value

list containing the following fields:

`psd.median,psd.mean`
: psd estimates: (pointwise) posterior median and mean

`psd.p05,psd.p95`
: pointwise credibility interval

`psd.u05,psd.u95`
: uniform credibility interval

`k,tau,V,W`
: posterior traces of nonparametric correction

`rho`
: posterior trace of model AR parameters (PACF parametrization)

`eta`
: posterior trace of model confidence eta

`lpost`
: trace of log posterior

References

C. Kirch et al. (2018) *Beyond Whittle: Nonparametric Correction of a Parametric Likelihood With a Focus on Bayesian Time Series Analysis* Bayesian Analysis <doi:10.1214/18-BA1126>

S. Ghosal and A. van der Vaart (2017) *Fundamentals of Nonparametric Bayesian Inference* <doi:10.1017/9781139029834>

O. Barndorff-Nielsen and G. Schou On the parametrization of autoregressive models by partial autocorrelations Journal of Multivariate Analysis (3),408-419 <doi:10.1016/0047-259X(73)90030-4>

Examples

```
## Not run:

##
## Example 1: Fit a nonparametrically corrected AR(p) model to sunspot data:
##

# Use this variable to set the AR model order
p <- 2

data <- sqrt(as.numeric(sunspot.year))
data <- data - mean(data)

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_npc(data=data, ar.order=p, Ntotal=10000, burnin=4000, thin=2)

# Plot spectral estimate, credible regions and periodogram on log-scale
plot(mcmc, log=T)


##
## Example 2: Fit a nonparametrically corrected AR(p) model to high-peaked AR(1) data
##

# Use this variable to set the autoregressive model order
p <- 1

n <- 256
data <- arima.sim(n=n, model=list(ar=0.95))
data <- data - mean(data)
omega <- fourier_freq(n)
psd_true <- psd_arma(omega, ar=0.95, ma=numeric(0), sigma2=1)

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_npc(data=data, ar.order=p, Ntotal=10000, burnin=4000, thin=2)

# Compare estimate with true function (green)
plot(mcmc, log=F, pdgrm=F, credib="uniform")
lines(x=omega, y=psd_true, col=3, lwd=2)
```

```
# Compute the Integrated Absolute Error (IAE) of posterior median
cat("IAE=", mean(abs(mcmc$psd.median-psd_true)[-1]) , sep="")

## End(Not run)
```

---

gibbs_var *Gibbs sampler for vector autoregressive model.*

Description

Obtain samples of the posterior of a Bayesian VAR model of fixed order. An independent Normal-Inverse-Wishart prior is employed.

Usage

```
gibbs_var(
  data,
  ar.order,
  Ntotal,
  burnin,
  thin = 1,
  print_interval = 500,
  full_lik = F,
  beta.mu = rep(0, ar.order * ncol(data)^2),
  beta.Sigma = 10000 * diag(ar.order * ncol(data)^2),
  Sigma.S = 1e-04 * diag(ncol(data)),
  Sigma.nu = 1e-04
)
```

Arguments

`data`	numeric matrix; NA values are interpreted as missing values and treated as random
`ar.order`	order of the autoregressive model (integer >= 0)
`Ntotal`	total number of iterations to run the Markov chain
`burnin`	number of initial iterations to be discarded
`thin`	thinning number (postprocessing)
`print_interval`	Number of iterations, after which a status is printed to console
`full_lik`	logical; if TRUE, the full likelihood for all observations is used; if FALSE, the partial likelihood for the last n-p observations
`beta.mu`	prior mean of beta vector (normal)
`beta.Sigma`	prior covariance matrix of beta vector
`Sigma.S`	prior parameter for the innovation covariance matrix, symmetric positive definite matrix
`Sigma.nu`	prior parameter for the innovation covariance matrix, nonnegative real number

Details

See Section 2.2.3 in Koop and Korobilis (2010) or Section 6.2 in Meier (2018) for further details

Value

list containing the following fields:

- `beta` — matrix containing traces of the VAR parameter vector beta
- `Sigma` — trace of innovation covariance Sigma
- `psd.median,psd.mean` — psd estimates: (pointwise, componentwise) posterior median and mean
- `psd.p05,psd.p95` — pointwise credibility interval
- `psd.u05,psd.u95` — uniform credibility interval, see (6.5) in Meier (2018)
- `lpost` — trace of log posterior

References

G. Koop and D. Korobilis (2010) *Bayesian Multivariate Time Series Methods for Empirical Macroeconomics* Foundations and Trends in Econometrics <doi:10.1561/0800000013>

A. Meier (2018) *A Matrix Gamma Process and Applications to Bayesian Analysis of Multivariate Time Series* PhD thesis, OvGU Magdeburg <https://opendata.uni-halle.de//handle/1981185920/13470>

Examples

```
## Not run:

##
## Example 1: Fit a VAR(p) model to SOI/Recruitment series:
##

# Use this variable to set the VAR model order
p <- 5

data <- cbind(as.numeric(astsa::soi-mean(astsa::soi)),
              as.numeric(astsa::rec-mean(astsa::rec)) / 50)
data <- apply(data, 2, function(x) x-mean(x))

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_var(data=data, ar.order=p, Ntotal=10000, burnin=4000, thin=2)

# Plot spectral estimate, credible regions and periodogram on log-scale
plot(mcmc, log=T)



##
```

```
## Example 2: Fit a VAR(p) model to VMA(1) data
##

# Use this variable to set the VAR model order
p <- 5

n <- 256
ma <- rbind(c(-0.75, 0.5), c(0.5, 0.75))
Sigma <- rbind(c(1, 0.5), c(0.5, 1))
data <- sim_varma(model=list(ma=ma), n=n, d=2)
data <- apply(data, 2, function(x) x-mean(x))

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_var(data=data, ar.order=p, Ntotal=10000, burnin=4000, thin=2)

# Plot spectral estimate, credible regions and periodogram on log-scale
plot(mcmc, log=T)

## End(Not run)
```

---

gibbs_vnp *Gibbs sampler for multivaiate Bayesian nonparametric inference with Whittle likelihood*

Description

Obtain samples of the posterior of the multivariate Whittle likelihood in conjunction with an Hpd AGamma process prior on the spectral density matrix.

Usage

```
gibbs_vnp(
  data,
  Ntotal,
  burnin,
  thin = 1,
  print_interval = 100,
  numerical_thresh = 1e-07,
  adaption.N = burnin,
  adaption.batchSize = 50,
  adaption.tar = 0.44,
  eta = ncol(data),
  omega = ncol(data),
  Sigma = 10000 * diag(ncol(data)),
  k.theta = 0.01,
  kmax = 100 * coars + 500 * (!coars),
  trunc_l = 0.1,
  trunc_r = 0.9,
```

```
  coars = F,
  L = max(20, length(data)^(1/3))
)
```

Arguments

`data`	numeric matrix; NA values are interpreted as missing values and treated as random
`Ntotal`	total number of iterations to run the Markov chain
`burnin`	number of initial iterations to be discarded
`thin`	thinning number (postprocessing)
`print_interval`	Number of iterations, after which a status is printed to console
`numerical_thresh`	Lower (numerical pointwise) bound for the eigenvalues of the spectral density
`adaption.N`	total number of iterations, in which the proposal variances (of r and U) are adapted
`adaption.batchSize`	batch size of proposal adaption
`adaption.tar`	target acceptance rate for adapted parameters
`eta`	AGamma process parameter, real number > ncol(data)-1
`omega`	AGamma process parameter, positive constant
`Sigma`	AGamma process parameter, Hpd matrix
`k.theta`	prior parameter for polynomial degree k (propto exp(-k.theta*k*log(k)))
`kmax`	upper bound for polynomial degree of Bernstein-Dirichlet mixture (can be set to Inf, algorithm is faster with kmax<Inf due to pre-computation of basis functions, but values 500<kmax<Inf are very memory intensive)
`trunc_l, trunc_r`	left and right truncation of Bernstein polynomial basis functions, 0<=trunc_l<trunc_r<=1
`coars`	flag indicating whether coarsened or default bernstein polynomials are used (see Appendix E.1 in Ghosal and van der Vaart 2017)
`L`	truncation parameter of Gamma process

Details

A detailed description of the method can be found in Section 5 in Meier (2018).

Value

list containing the following fields:

`r,x,U`	traces of the AGamma process parameters
`k`	posterior trace of polynomial degree
`psd.median,psd.mean`	psd estimates: (pointwise, componentwise) posterior median and mean

`psd.p05,psd.p95`
: pointwise credibility interval

`psd.u05,psd.u95`
: uniform credibility interval, see (6.5) in Meier (2018)

`lpost`
: trace of log posterior

References

A. Meier (2018) *A Matrix Gamma Process and Applications to Bayesian Analysis of Multivariate Time Series* PhD thesis, OvGU Magdeburg <https://opendata.uni-halle.de//handle/1981185920/13470>

Examples

```
## Not run:

##
## Example: Fit multivariate NP model to SOI/Recruitment series:
##

data <- cbind(as.numeric(astsa::soi-mean(astsa::soi)),
              as.numeric(astsa::rec-mean(astsa::rec)) / 50)
data <- apply(data, 2, function(x) x-mean(x))

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_vnp(data=data, Ntotal=10000, burnin=4000, thin=2)

# Visualize results
plot(mcmc, log=T)


##
## Example 2: Fit multivariate NP model to VMA(1) data
##

n <- 256
ma <- rbind(c(-0.75, 0.5), c(0.5, 0.75))
Sigma <- rbind(c(1, 0.5), c(0.5, 1))
data <- sim_varma(model=list(ma=ma), n=n, d=2)
data <- apply(data, 2, function(x) x-mean(x))

# If you run the example be aware that this may take several minutes
print("example may take some time to run")
mcmc <- gibbs_vnp(data=data, Ntotal=10000, burnin=4000, thin=2)

# Plot spectral estimate, credible regions and periodogram on log-scale
plot(mcmc, log=T)

## End(Not run)
```

local_moving_FT_zigzag

Calculate the moving Fourier transform ordinates

Description

Calculate the moving Fourier transform ordinates

Usage

```
local_moving_FT_zigzag(x, m, thinning_factor)
```

Arguments

`x`	A numeric vector containing time series.
`m`	A positive integer indicating the size of window.
`thinning_factor`	Selected from "1, 2, 3".

Value

A list containing the moving Fourier transform and corresponding time grid.

References

Y. Tang et al. (2023) *Bayesian nonparametric spectral analysis of locally stationary processes* ArXiv preprint <arXiv:2303.11561>

Examples

```
set.seed(1); x <- rnorm(1500)
local_moving_FT_zigzag(x, 50, 1)
```

pacf_to_ar

Convert partial autocorrelation coefficients to AR coefficients.

Description

Convert partial autocorrelation coefficients to AR coefficients.

Usage

```
pacf_to_ar(pacf)
```

Arguments

`pacf`	numeric vector of partial autocorrelations in (-1,1)

Details

See Section 2 in Kirch et al (2018) or Section III in Barndorff-Nielsen and Schou (1973) for further details

Value

numeric vector of autoregressive model coefficients

References

C. Kirch et al Supplemental material of *Beyond Whittle: Nonparametric Correction of a Parametric Likelihood With a Focus on Bayesian Time Series Analysis* Bayesian Analysis <doi:10.1214/18-BA1126SUPP>

O. Barndorff-Nielsen and G. Schou On the parametrization of autoregressive models by partial autocorrelations Journal of Multivariate Analysis (3),408-419 <doi:10.1016/0047-259X(73)90030-4>

See Also

acf2AR, ARMAacf

`plot.bdp_dw_result` *Plot method for bdp_dw_result class*

Description

Plot method for bdp_dw_result class

Usage

```
## S3 method for class 'bdp_dw_result'
plot(
  x,
  which = 1:4,
  ask = prod(par("mfcol")) < length(which) && dev.interactive(),
  col = hcl.colors(200, "Blue-Red 3"),
  ...
)
```

Arguments

`x`	object of class bdp_dw_result
`which`	if a subset of the plots is required, specify a subset of the numbers 1:6. 1 indicates posterior mean, 2 indicates posterior median, 3 for lower bound of pointwise 90 percent credible region, 4 for upper bound of pointewise 90 percent credible region, 5 indicates lower bound of uniform 90 percent credible region, 6 indicates upper bound of uniform 90 percent credible region.
`ask`	logical; if TRUE, the user is asked before each plot.
`col`	choice of color, default hcl.colors(200, "Blue-Red 3").
`...`	other parameters to be passed through to `image.default`

`plot.bdp_dw_tv_psd` *Plot method for bdp_dw_tv_psd class*

Description

Plot method for bdp_dw_tv_psd class

Usage

```
## S3 method for class 'bdp_dw_tv_psd'
plot(x, col = hcl.colors(200, "Blue-Red 3"), ...)
```

Arguments

`x`	an object of class bdp_dw_tv_psd
`col`	choice of color, default hcl.colors(200, "Blue-Red 3").
`...`	further arguments to be parsed to `image.default`

Details

Visualizes the spectral density function of time-varying

plot.gibbs_psd *Plot method for gibbs_psd class*

Description

Plot method for gibbs_psd class

Usage

```
## S3 method for class 'gibbs_psd'
plot(x, pdgrm = T, credib = "both", log = T, ...)
```

Arguments

x	an object of class gibbs_psd
pdgrm	bool flag indicating whether periodogram is visualized or not
credib	string indicating which credible regions are visualized. Possible values are "pointwise", "uniform", "both" and "none".
log	logical value to determine if the individual spectra are visualized on a log scale
...	further arguments to be parsed to `plot.default`

Details

Visualizes the spectral density estimate (pointwise posterior median), along with the periodogram and credibility regions. If the data has missing values, the periodogram is computed with a linearly interpolated version of the data using na.interp.

print.bdp_dw_result *Print method for bdp_dw_result class*

Description

Print method for bdp_dw_result class

Usage

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

Arguments

x	object of class bdp_dw_result
...	not in use

print.gibbs_psd — *Print method for gibbs_psd class*

Description

Print method for gibbs_psd class

Usage

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

Arguments

x	object of class gibbs_psd
...	not in use

psd_arma — *ARMA(p,q) spectral density function*

Description

Evaluate the ARMA(p,q) spectral density at some frequencies freq in [0,pi), Note that no test for model stationarity is performed.

Usage

```
psd_arma(freq, ar, ma, sigma2 = 1)
```

Arguments

freq	numeric vector of frequencies to evaluate the psd, 0 <= freq < pi
ar	autoregressive coefficients of ARMA model (use numeric(0) for empty AR part)
ma	moving average coefficients of ARMA model (use numeric(0) for empty MA part)
sigma2	the model innovation variance

Details

See section 4.4 in the referenced book

Value

numeric vector of the (real-valued) spectral density values

References

P. J. Brockwell and R. Davis (1996) *Time Series: Theory and Methods (Second Edition)*

`psd_tvarma12` *time-varying spectral density function of the tvARMA(1,2) processes for illustrations*

Description

time-varying spectral density function of the tvARMA(1,2) processes for illustrations

Usage

```
psd_tvarma12(
  rescaled_time,
  freq,
  dgp = NULL,
  a1 = function(u) {
     rep(0, length(u))
 },
  b1 = function(u) {
     rep(0, length(u))
 },
  b2 = function(u) {
     rep(0, length(u))
 }
)
```

Arguments

`rescaled_time, freq`	numeric vectors forming a rectangular grid on which the tv-PSD is evaluated.
`dgp`	optional: the tv-ARMA models demonstrated in section 4.2 of Tang et al. (2023). Should be chosen from "LS1", "LS2" and "LS3". See section Details.
`a1, b1, b2`	If dgp is not supplied, these arguments can be used to specify customized tv-ARMA process (up to order(1,2)). See Details. rescaled_time must be in $[0, 1]$ and freq must be in $[0, \pi]$.

Details

See sim_tvarma12 for the precise definition of a tvARMA(1,2) process. The time-varying spectral density function of this process is defined as

$$f(u,\lambda) = \frac{1}{2\pi}\frac{1 + b_1^2(u) + b_2^2(u) + 2b_1(u)(b_2(u)+1)\cos(\lambda) + 2b_2(u)\cos(2\lambda)}{1 + a_1^2(u) - 2a_1(u)\cos(\lambda)}, \quad (u,\lambda) \in [0,1]\times[0,\pi],$$

where u is called rescaled time and λ is called frequency.

For dgp = "LS1", it is a tvMA(2) process (MA order is 2) with

$$a_1(u) = 0, b_1(u) = 1.122(1 - 1.178\sin(\pi/2u)), b_2(u) = -0.81.$$

For dgp = "LS2", it is a tvMA(1) process (MA order is 1) with

$$a_1(u) = 0, b_1(u) = 1.1\cos\left(1.5 - \cos\left(4\pi u\right)\right), b_2(u) = 0.$$

For dgp = "LS3", it is a tvAR(1) process (MA order is 0) with

$$a_1(u) = 1.2u - 0.6, b_1(u) = 0, b_2(u) = 0.$$

Value

a matrix of dimension length(rescaled_time) by length(freq).

References

Tang et al. (2023) *Bayesian nonparametric spectral analysis of locally stationary processes* ArXiv preprint <arXiv:2303.11561>

Examples

```
## Not run:
res_time <- seq(0, 1, by = 0.005); freq <- pi*seq(0, 1, by = 0.01)
true_tvPSD <- psd_tvarma12(rescaled_time = res_time, freq = freq, dgp = "LS2")
plot(true_tvPSD)

## End(Not run)
```

---

psd_varma   *VARMA(p,q) spectral density function*

Description

Evaluate the VARMA(p,q) spectral density at some frequencies freq in [0,pi). Note that no test for model stationarity is performed.

Usage

```
psd_varma(
  freq,
  ar = matrix(nrow = nrow(Sigma), ncol = 0),
  ma = matrix(nrow = nrow(Sigma), ncol = 0),
  Sigma
)
```

Arguments

`freq`	numeric vector of frequencies to evaluate the psd, 0 <= freq < pi
`ar`	autoregressive coeffient matrix (d times p*d) of VARMA model, defaults to empty VAR component
`ma`	moving average coeffient matrix (d times p*d) of VARMA model, defaults to empty VAR component
`Sigma`	positive definite innovation covariance matrix (d times d)

Details

See section 11.5 in the referenced book

Value

an array containing the values of the varma psd matrix at freq

References

P. J. Brockwell and R. Davis (1996) *Time Series: Theory and Methods (Second Edition)*

`rmvnorm` *Simulate from a Multivariate Normal Distribution*

Description

Produces one or more samples from the specified multivariate normal distribution.

Usage

```
rmvnorm(n, d, mu = rep(0, d), Sigma = diag(d), ...)
```

Arguments

`n`	sample size
`d`	dimensionality
`mu`	mean vector
`Sigma`	covariance matrix
`...`	further arguments to be parsed to

Details

This is a simple wrapper function based on mvrnorm, to be used within sim_varma

Value

If n=1 a vector of length d, otherwise an n by d matrix with one sample in each row.

`scree_type_ar` *Negative log AR likelihood values for scree-type plots*

Description

(Approximate) negative maximum log-likelihood for for different autoregressive orders to produce scree-type plots.

Usage

```
scree_type_ar(data, order.max, method = "yw")
```

Arguments

`data`	numeric vector of data
`order.max`	maximum autoregressive order to consider
`method`	character string giving the method used to fit the model, to be forwarded to `stats::ar`

Details

By default, the maximum likelihood is approximated by the Yule-Walker method, due to numerical stabililty and computational speed. Further details can be found in the simulation study section in the referenced paper.

Value

a data frame containing the autoregressive orders `p` and the corresponding negative log likelihood values `nll`

References

C. Kirch et al. (2018) *Beyond Whittle: Nonparametric Correction of a Parametric Likelihood With a Focus on Bayesian Time Series Analysis* Bayesian Analysis <doi:10.1214/18-BA1126>

Examples

```
## Not run:

###
### Interactive visual inspection for the sunspot data
###

data <- sqrt(as.numeric(sunspot.year))
data <- data <- data - mean(data)

screeType <- scree_type_ar(data, order.max=15)
```

```
# Determine the autoregressive order by an interactive visual inspection of the scree-type plot
plot(x=screeType$p, y=screeType$nll, type="b")
p_ind <- identify(x=screeType$p, y=screeType$nll, n=1, labels=screeType$p)
print(screeType$p[p_ind])

## End(Not run)
```

`sim_tvarma12` *simulate from the tvARMA(1,2) process for illustration*

Description

simulate from the tvARMA(1,2) process for illustration

Usage

```
sim_tvarma12(
  len_d,
  dgp = NULL,
  ar_order = 1,
  ma_order = 2,
  a1 = NULL,
  b1 = NULL,
  b2 = NULL,
  innov_distribution = NULL,
  wn = NULL
)
```

Arguments

`len_d` a positive integer indicating the length of the simulated process.

`dgp` optional: the tv-ARMA models demonstrated in section 4.2 of Tang et al. (2023). Should be chosen from "LS1", "LS2" and "LS3". See section Details.

`ar_order, ma_order, a1, b1, b2`
If dgp is not supplied, these arguments can be used to specify customized tv-ARMA process (up to order(1,2)). See details.

`innov_distribution`
optional: the distributions of innovation used in section 4.2.2 of Tang et al. (2023) . Should be chosen from "a", "b", "c". "a" denotes standard normal distribution, "b" indicates standardized Student-t distribution with degrees of freedom 4 and "c" denotes standardized Pareto distribution with scale 1 and shape 4.

`wn` If innov_distribution is not specified, one may supply its own innovation sequence. Please make sure the length of wn is at least the sum of len_d and the MA order of the process. If ma_order is specified, then MA order is exactly ma_order. If dgp is specified, the MA order of "LS1", "LS2" and "LS3" can be found in section Details below.

Details

This function simulates from the following time-varying Autoregressive Moving Average model with order (1,2):

$$X_{t,T} = a_1(t/T)X_{t-1,T} + w_t + b_1(t/T)w_{t-1} + b_2(t/T)w_{t-2}, \quad t = 1, 2, \cdots, T,$$

where T is the length specified and $\{w_t\}$ are a sequence of i.i.d. random variables with mean 0 and standard deviation 1.

For dgp = "LS1", it is a tvMA(2) process (MA order is 2) with

$$a_1(u) = 0, b_1(u) = 1.122(1 - 1.178\sin(\pi/2u)), b_2(u) = -0.81.$$

For dgp = "LS2", it is a tvMA(1) process (MA order is 1) with

$$a_1(u) = 0, b_1(u) = 1.1\cos\left(1.5 - \cos\left(4\pi u\right)\right), b_2(u) = 0.$$

For dgp = "LS3", it is a tvAR(1) process (MA order is 0) with

$$a_1(u) = 1.2u - 0.6, b_1(u) = 0, b_2(u) = 0.$$

Value

a numeric vector of length len_d simulated from the given process.

References

Tang et al. (2023) *Bayesian nonparametric spectral analysis of locally stationary processes* ArXiv preprint <arXiv:2303.11561>

Examples

```
## Not run:
sim_tvarma12(len_d = 1500,
dgp = "LS2",
innov_distribution = "a") # generate from LS2a

sim_tvarma12(len_d = 1500,
dgp = "LS2",
wn = rnorm(1502)) # again, generate from LS2a

sim_tvarma12(len_d = 1500,
ar_order = 0,
ma_order = 1,
b1 = function(u){1.1*cos(1.5 - cos(4*pi*u))},
innov_distribution = "a") # again, generate from LS2a

sim_tvarma12(len_d = 1500,
ar_order = 0,
ma_order = 1,
b1 = function(u){1.1*cos(1.5 - cos(4*pi*u))},
```

```
wn = rnorm(1502)) # again, generate from LS2a

## End(Not run)
```

---

sim_varma *Simulate from a VARMA model*

Description

Simulate from a Vector Autoregressive Moving Average (VARMA) model. Note that no test for model stationarity is performed.

Usage

```
sim_varma(model, n, d, rand.gen = rmvnorm, burnin = 10000, ...)
```

Arguments

model	A list with component `ar` and/or `ma` giving the VAR and VMA coefficients respectively. An empty list gives an VARMA(0, 0) model, that is white noise.
n	sample size
d	positive integer for the dimensionality
rand.gen	random vector generator, function of type rand.gen(n, d, ...)
burnin	length of burnin period (initial samples that are discarded)
...	further arguments to be parsed to `rand.gen`

Value

If n=1 a vector of length d, otherwise an n by d matrix with one sample in each row.

See Also

arima.sim to simulate from univariate ARMA models

Examples

```
## Not run:
# Example: Draw from bivariate normal VAR(2) model
ar <- rbind(c(.5, 0, 0, 0), c(0, -.3, 0, -.5))
Sigma <- matrix(data=c(1, .9, .9, 1), nrow=2, ncol=2)
x <- sim_varma(n=256, d=2, model=list(ar=ar))
plot.ts(x)

## End(Not run)
```

summary.bdp_dw_result *Summary method for bdp_dw_result class*

Description

Summary method for bdp_dw_result class

Usage

```
## S3 method for class 'bdp_dw_result'
summary(object, ...)
```

Arguments

`object`	object of class bdp_dw_result
`...`	not in use

summary.gibbs_psd *Summary method for gibbs_psd class*

Description

Summary method for gibbs_psd class

Usage

```
## S3 method for class 'gibbs_psd'
summary(object, ...)
```

Arguments

`object`	object of class gibbs_psd
`...`	not in use

Index