Package ‘beyondWhittle’

April 4, 2022

Type Package
Title Bayesian Spectral Inference for Stationary Time Series
Version 1.1.3
Date 2022-04-03
Maintainer Alexander Meier <meier.alexander@posteo.de>
License GPL (>= 3)
Imports ltsa (>= 1.4.6), Rcpp (>= 0.12.5), MASS, forecast
LinkingTo Rcpp, RcppArmadillo, BH
RoxygenNote 6.1.1
NeedsCompilation yes
Author Alexander Meier [aut, cre], Claudia Kirch [aut], Matthew C. Edwards [aut], Renate Meyer [aut]
Repository CRAN
Date/Publication 2022-04-04 11:00:02 UTC
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3

R topics documented:

beyondWhittle-package ........................................... 2
fourier_freq ....................................................... 3
gibbs_ar .......................................................... 3
gibbs_np .......................................................... 6
gibbs_npc .......................................................... 8
Bayesian spectral inference for stationary time series

Description

Bayesian parametric, nonparametric and semiparametric procedures for spectral density inference of univariate and multivariate 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 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 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)

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 grant KI 1443/3-1.

Author(s)

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

Maintainer: Alexander Meier <meier.alexander@posteo.de>
fourier_freq

Fourier frequencies

Description
Fourier frequencies on \([0, \pi]\), as defined by \(2\pi j/n\) for \(j=0, \ldots, \text{floor}(n/2)\).

Usage

\texttt{fourier_freq(n)}

Arguments

\begin{itemize}
  \item \texttt{n} integer
\end{itemize}

Value

device 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

\texttt{gibbs_ar(data, ar.order, Ntotal, burnin, thin = 1,}
\texttt{  print_interval = 500, numerical_thresh = 1e-07,}
\texttt{  adaption.N = burnin, adaption.batchSize = 50, adaption.tar = 0.44,}
\texttt{  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


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


Examples

```r
## 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_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


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

Examples

```r
## 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))
```
gibbs_npc <- gibbs_npc(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

```r
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

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

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

```r
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 &ge; 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


Examples
```r
## 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 multivariate 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
numeral_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

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)
```

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>


See Also
acf2AR, ARMAacf
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.gibbs_psd

**Print method for gibbs_psd class**

**Description**
Print method for gibbs_psd class

**Usage**
```r
## 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**
```r
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
Values

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

References


---

**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**

```r
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 coefficient matrix (d times p*d) of VARMA model, defaults to empty VAR component
- `ma` : moving average coefficient 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**

**rmvnorm**  
*Simulate from a Multivariate Normal Distribution*

**Description**

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

**Usage**

```r
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 `mvnorm`, 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**

```r
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 stability 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


Examples

```r
## Not run:

### Interactive visual inspection for the sunspot data

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

```r
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 a 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

```r
## 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.gibbs_psd

**Summary method for gibbs_psd class**

**Description**

Summary method for `gibbs_psd` class

**Usage**

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

**Arguments**

- **object**: object of class `gibbs_psd`
- **...**: not in use
Index

* package
  beyondWhittle-package, 2

acf2AR, 16
ar, 19
arima.sim, 21
ARMAacf, 16

beyondWhittle(beyondWhittle-package), 2
beyondWhittle-package, 2

fourier_freq, 3
gibbs_ar, 2, 3
gibbs_np, 2, 6
gibbs_npc, 2, 8
gibbs_var, 2, 11
gibbs_vnp, 2, 13

mvrnorm, 19

na.interp, 17

pacf_to_ar, 15
plot.gibbs_psd, 16
print.gibbs_psd, 17
psd arma, 17
psd varma, 18

rmvnorm, 19

scree_type_ar, 19
sim varma, 19, 20
summary.gibbs_psd, 21