Package ‘pmhtutorial’

October 14, 2022

Type Package

Title Minimal Working Examples for Particle Metropolis-Hastings

Version 1.5

Author Johan Dahlin

Maintainer Johan Dahlin <uni@johandahlin.com>

URL https://github.com/compops/pmh-tutorial-rpkg

Description Routines for state estimate in a linear
Gaussian state space model and a simple stochastic volatility model using particle filtering. Parameter inference is also carried out in these models using the particle Metropolis-Hastings algorithm that includes the particle filter to provided an unbiased estimator of the likelihood. This package is a collection of minimal working examples of these algorithms and is only meant for educational use and as a start for learning to them on your own.

Depends R (>= 3.2.3)

License GPL-2

Imports mvtnorm, Quandl, grDevices, graphics, stats

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-03-22 18:10:03 UTC

R topics documented:

`example1_lgss` *State estimation in a linear Gaussian state space model*

Description

Minimal working example of state estimation in a linear Gaussian state space model using Kalman filtering and a fully-adapted particle filter. The code estimates the bias and mean squared error (compared with the Kalman estimate) while varying the number of particles in the particle filter.

Usage

```
example1_lgss()
```

Details

The Kalman filter is a standard implementation without an input. The particle filter is fully adapted (i.e. takes the current observation into account when proposing new particles and computing the weights).

Value

Returns a plot with the generated observations y and the difference in the state estimates obtained by the Kalman filter (the optimal solution) and the particle filter (with 20 particles). Furthermore, the function returns plots of the estimated bias and mean squared error of the state estimate obtained using the particle filter (while varying the number of particles) and the Kalman estimates.

The function returns a list with the elements:

- y: The observations generated from the model.
- x: The states generated from the model.
- kfEstimate: The estimate of the state from the Kalman filter.
- pfEstimate: The estimate of the state from the particle filter with 20 particles.

Note

See Section 3.2 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
example1_lgss()
```

---

example2_lgss — *Parameter estimation in a linear Gaussian state space model*

Description

Minimal working example of parameter estimation in a linear Gaussian state space model using the particle Metropolis-Hastings algorithm with a fully-adapted particle filter providing an unbiased estimator of the likelihood. The code estimates the parameter posterior for one parameter using simulated data.

Usage

```
example2_lgss(noBurnInIterations = 1000, noIterations = 5000,
  noParticles = 100, initialPhi = 0.5)
```

Arguments

`noBurnInIterations`	The number of burn-in iterations in the PMH algorithm. This parameter must be smaller than `noIterations`.
`noIterations`	The number of iterations in the PMH algorithm. 100 iterations takes about ten seconds on a laptop to execute. 5000 iterations are used in the reference below.
`noParticles`	The number of particles to use when estimating the likelihood.
`initialPhi`	The initial guess of the parameter phi.

Details

The Particle Metropolis-Hastings (PMH) algorithm makes use of a Gaussian random walk as the proposal for the parameter. The PMH algorithm is run using different step lengths in the proposal. This is done to illustrate the difficulty when tuning the proposal and the impact of a too small/large step length.

Value

Returns the estimate of the posterior mean.

Note

See Section 4.2 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
example2_lgss(noBurnInIterations=200, noIterations=1000)
```

---

example3_sv *Parameter estimation in a simple stochastic volatility model*

Description

Minimal working example of parameter estimation in a stochastic volatility model using the particle Metropolis-Hastings algorithm with a bootstrap particle filter providing an unbiased estimator of the likelihood. The code estimates the parameter posterior for three parameters using real-world data.

Usage

```
example3_sv(noBurnInIterations = 2500, noIterations = 7500,
  noParticles = 500, initialTheta = c(0, 0.9, 0.2),
  stepSize = diag(c(0.1, 0.01, 0.05)^2), syntheticData = FALSE)
```

Arguments

`noBurnInIterations`	The number of burn-in iterations in the PMH algorithm. Must be smaller than `noIterations`.
`noIterations`	The number of iterations in the PMH algorithm. 100 iterations takes about a minute on a laptop to execute.
`noParticles`	The number of particles to use when estimating the likelihood.
`initialTheta`	The initial guess of the parameters theta.
`stepSize`	The step sizes of the random walk proposal. Given as a covariance matrix.
`syntheticData`	If TRUE, data is not downloaded from the Internet. This is only used when running tests of the package.

Details

The Particle Metropolis-Hastings (PMH) algorithm makes use of a Gaussian random walk as the proposal for the parameters. The data are scaled log-returns from the OMXS30 index during the period from January 2, 2012 to January 2, 2014.

This version of the code makes use of a somewhat well-tuned proposal as a pilot run to estimate the posterior covariance and therefore increase the mixing of the Markov chain.

Value

The function returns the estimated marginal parameter posteriors for each parameter, the trace of the Markov chain and the resulting autocorrelation function. The data is also presented with an estimate of the log-volatility.

The function returns a list with the elements:

- thhat: The estimate of the mean of the parameter posterior.
- xhat: The estimate of the mean of the log-volatility posterior.
- thhatSD: The estimate of the standard deviation of the parameter posterior.
- xhatSD: The estimate of the standard deviation of the log-volatility posterior.
- iact: The estimate of the integrated autocorrelation time for each parameter.
- estCov: The estimate of the covariance of the parameter posterior.
- theta: The trace of the chain exploring the parameter posterior.

Note

See Section 5 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
## Not run:
    example3_sv(noBurnInIterations=200, noIterations=1000)

## End(Not run)
```

example4_sv *Parameter estimation in a simple stochastic volatility model*

Description

Minimal working example of parameter estimation in a stochastic volatility model using the particle Metropolis-Hastings algorithm with a bootstrap particle filter providing an unbiased estimator of the likelihood. The code estimates the parameter posterior for three parameters using real-world data.

Usage

```
example4_sv(noBurnInIterations = 2500, noIterations = 7500,
  noParticles = 500, initialTheta = c(0, 0.9, 0.2),
  syntheticData = FALSE)
```

Arguments

`noBurnInIterations`	The number of burn-in iterations in the PMH algorithm. Must be smaller than `noIterations`.
`noIterations`	The number of iterations in the PMH algorithm. 100 iterations takes about a minute on a laptop to execute.
`noParticles`	The number of particles to use when estimating the likelihood.
`initialTheta`	The initial guess of the parameters theta.
`syntheticData`	If TRUE, data is not downloaded from the Internet. This is only used when running tests of the package.

Details

The Particle Metropolis-Hastings (PMH) algorithm makes use of a Gaussian random walk as the proposal for the parameters. The data are scaled log-returns from the OMXS30 index during the period from January 2, 2012 to January 2, 2014.

This version of the code makes use of a proposal that is tuned using a run of `example3_sv` and therefore have better mixing properties.

Value

The function returns the estimated marginal parameter posteriors for each parameter, the trace of the Markov chain and the resulting autocorrelation function. The data is also presented with an estimate of the log-volatility.

The function returns a list with the elements:

- thhat: The estimate of the mean of the parameter posterior.
- xhat: The estimate of the mean of the log-volatility posterior.
- thhatSD: The estimate of the standard deviation of the parameter posterior.

- xhatSD: The estimate of the standard deviation of the log-volatility posterior.
- iact: The estimate of the integrated autocorrelation time for each parameter.
- estCov: The estimate of the covariance of the parameter posterior.

Note

See Section 6.3.1 in the reference for more details.

Author(s)

Johan Dahlin `<uni@johandahlin.com>`

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
## Not run:
    example4_sv(noBurnInIterations=200, noIterations=1000)

## End(Not run)
```

`example5_sv` *Parameter estimation in a simple stochastic volatility model*

Description

Minimal working example of parameter estimation in a stochastic volatility model using the particle Metropolis-Hastings algorithm with a bootstrap particle filter providing an unbiased estimator of the likelihood. The code estimates the parameter posterior for three parameters using real-world data.

Usage

```
example5_sv(noBurnInIterations = 2500, noIterations = 7500,
  noParticles = 500, initialTheta = c(0, 0.9, 0.2),
  syntheticData = FALSE)
```

Arguments

`noBurnInIterations`
: The number of burn-in iterations in the PMH algorithm. Must be smaller than `noIterations`.

`noIterations`
: The number of iterations in the PMH algorithm. 100 iterations takes about a minute on a laptop to execute.

`noParticles`	The number of particles to use when estimating the likelihood.
`initialTheta`	The initial guess of the parameters theta.
`syntheticData`	If TRUE, data is not downloaded from the Internet. This is only used when running tests of the package.

Details

The Particle Metropolis-Hastings (PMH) algorithm makes use of a Gaussian random walk as the proposal for the parameters. The data are scaled log-returns from the OMXS30 index during the period from January 2, 2012 to January 2, 2014.

This version of the code makes use of a proposal that is tuned using a pilot run. Furthermore the model is reparameterised to enjoy better mixing properties by making the parameters unrestricted to a certain part of the real-line.

Value

The function returns the estimated marginal parameter posteriors for each parameter, the trace of the Markov chain and the resulting autocorrelation function. The data is also presented with an estimate of the log-volatility.

The function returns a list with the elements:

- thhat: The estimate of the mean of the parameter posterior.
- xhat: The estimate of the mean of the log-volatility posterior.
- thhatSD: The estimate of the standard deviation of the parameter posterior.
- xhatSD: The estimate of the standard deviation of the log-volatility posterior.
- iact: The estimate of the integrated autocorrelation time for each parameter.
- estCov: The estimate of the covariance of the parameter posterior.

Note

See Section 6.3.2 in the reference for more details.

Author(s)

Johan Dahlin `<uni@johandahlin.com>`

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
## Not run:
    example5_sv(noBurnInIterations=200, noIterations=1000)

## End(Not run)
```

`generateData` *Generates data from a linear Gaussian state space model*

Description

Generates data from a specific linear Gaussian state space model of the form $x_t = \phi x_{t-1} + \sigma_v v_t$ and $y_t = x_t + \sigma_e e_t$, where v_t and e_t denote independent standard Gaussian random variables, i.e. $N(0, 1)$.

Usage

```
generateData(theta, noObservations, initialState)
```

Arguments

`theta`	The parameters $\theta = \{\phi, \sigma_v, \sigma_e\}$ of the LGSS model. The parameter ϕ that scales the current state in the state dynamics is restricted to [-1,1] to obtain a stable model. The standard deviations of the state process noise σ_v and the observation process noise σ_e must be positive.
`noObservations`	The number of time points to simulate.
`initialState`	The initial state.

Value

The function returns a list with the elements:

- x: The latent state for $t = 0, ..., T$.
- y: The observation for $t = 0, ..., T$.

Author(s)

Johan Dahlin `<uni@johandahlin.com>`

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

`kalmanFilter` *Kalman filter for state estimate in a linear Gaussian state space model*

Description

Estimates the filtered state and the log-likelihood for a linear Gaussian state space model of the form $x_t = \phi x_{t-1} + \sigma_v v_t$ and $y_t = x_t + \sigma_e e_t$, where v_t and e_t denote independent standard Gaussian random variables, i.e.$N(0, 1)$.

Usage

```
kalmanFilter(y, theta, initialState, initialStateCovariance)
```

Arguments

`y`	Observations from the model for $t = 1, ..., T$.
`theta`	The parameters $\theta = \{\phi, \sigma_v, \sigma_e\}$ of the LGSS model. The parameter ϕ scales the current state in the state dynamics. The standard deviations of the state process noise and the observation process noise are denoted σ_v and σ_e, respectively.
`initialState`	The initial state.
`initialStateCovariance`	The initial covariance of the state.

Value

The function returns a list with the elements:

- xHatFiltered: The estimate of the filtered state at time $t = 1, ..., T$.
- logLikelihood: The estimate of the log-likelihood.

Note

See Section 3 in the reference for more details.

Author(s)

Johan Dahlin <`uni@johandahlin.com`>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
# Generates 500 observations from a linear state space model with
# (phi, sigma_e, sigma_v) = (0.5, 1.0, 0.1) and zero initial state.
theta <- c(0.5, 1.0, 0.1)
d <- generateData(theta, noObservations=500, initialState=0.0)

# Estimate the filtered state using Kalman filter
kfOutput <- kalmanFilter(d$y, theta,
                         initialState=0.0, initialStateCovariance=0.01)

# Plot the estimate and the true state
par(mfrow=c(3, 1))
plot(d$x, type="l", xlab="time", ylab="true state", bty="n",
  col="#1B9E77")
plot(kfOutput$xHatFiltered, type="l", xlab="time",
  ylab="Kalman filter estimate", bty="n", col="#D95F02")
plot(d$x-kfOutput$xHatFiltered, type="l", xlab="time",
  ylab="difference", bty="n", col="#7570B3")
```

`makePlotsParticleMetropolisHastingsSVModel`
Make plots for tutorial

Description

Creates diagnoistic plots from runs of the particle Metropolis-Hastings algorithm.

Usage

```
makePlotsParticleMetropolisHastingsSVModel(y, res, noBurnInIterations,
  noIterations, nPlot)
```

Arguments

`y`	Observations from the model for $t = 1, ..., T$.
`res`	The output from a run of particleMetropolisHastings, particleMetropolisHastingsSVmodel or particleMetropolisHastingsSVmodelReparameterised.
`noBurnInIterations`	The number of burn-in iterations in the PMH algorithm. Must be smaller than noIterations.
`noIterations`	The number of iterations in the PMH algorithm.
`nPlot`	Number of steps in the Markov chain to plot.

Value

The function returns plots similar to the ones in the reference as well as the estimate of the integrated autocorrelation time for each parameter.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

`particleFilter` *Fully-adapted particle filter for state estimate in a linear Gaussian state space model*

Description

Estimates the filtered state and the log-likelihood for a linear Gaussian state space model of the form $x_t = \phi x_{t-1} + \sigma_v v_t$ and $y_t = x_t + \sigma_e e_t$, where v_t and e_t denote independent standard Gaussian random variables, i.e.$N(0, 1)$.

Usage

```
particleFilter(y, theta, noParticles, initialState)
```

Arguments

`y`	Observations from the model for $t = 1, ..., T$.
`theta`	The parameters $\theta = \{\phi, \sigma_v, \sigma_e\}$ of the LGSS model. The parameter ϕ scales the current state in the state dynamics. The standard deviations of the state process noise and the observation process noise are denoted σ_v and σ_e, respectively.
`noParticles`	The number of particles to use in the filter.
`initialState`	The initial state.

Value

The function returns a list with the elements:

- xHatFiltered: The estimate of the filtered state at time $t = 1, ..., T$.
- logLikelihood: The estimate of the log-likelihood.
- particles: The particle system at each time point.
- weights: The particle weights at each time point.

Note

See Section 3 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Non-linear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
# Generates 500 observations from a linear state space model with
# (phi, sigma_e, sigma_v) = (0.5, 1.0, 0.1) and zero initial state.
theta <- c(0.5, 1.0, 0.1)
d <- generateData(theta, noObservations=500, initialState=0.0)

# Estimate the filtered state using a Particle filter
pfOutput <- particleFilter(d$y, theta, noParticles = 50,
  initialState=0.0)

# Plot the estimate and the true state
par(mfrow=c(3, 1))
plot(d$x[1:500], type="l", xlab="time", ylab="true state", bty="n",
  col="#1B9E77")
plot(pfOutput$xHatFiltered, type="l", xlab="time",
  ylab="paticle filter estimate", bty="n", col="#D95F02")
plot(d$x[1:500]-pfOutput$xHatFiltered, type="l", xlab="time",
  ylab="difference", bty="n", col="#7570B3")
```

`particleFilterSVmodel` *Bootstrap particle filter for state estimate in a simple stochastic volatility model*

Description

Estimates the filtered state and the log-likelihood for a stochastic volatility model of the form $x_t = \mu + \phi(x_{t-1} - \mu) + \sigma_v v_t$ and $y_t = \exp(x_t/2)e_t$, where v_t and e_t denote independent standard Gaussian random variables, i.e. $N(0, 1)$.

Usage

```
particleFilterSVmodel(y, theta, noParticles)
```

Arguments

`y`	Observations from the model for $t = 1, ..., T$.
`theta`	The parameters $\theta = \{\mu, \phi, \sigma_v\}$. The mean of the log-volatility process is denoted μ. The persistence of the log-volatility process is denoted ϕ. The standard deviation of the log-volatility process is denoted σ_v.
`noParticles`	The number of particles to use in the filter.

Value

The function returns a list with the elements:

- xHatFiltered: The estimate of the filtered state at time $t = 1, ..., T$.
- logLikelihood: The estimate of the log-likelihood.

Note

See Section 5 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
## Not run:
  # Get the data from Quandl
  library("Quandl")
  d <- Quandl("NASDAQOMX/OMXS30", start_date="2012-01-02",
              end_date="2014-01-02", type="zoo")
  y <- as.numeric(100 * diff(log(d$"Index Value")))

  # Estimate the filtered state using a particle filter
  theta <- c(-0.10, 0.97, 0.15)
  pfOutput <- particleFilterSVmodel(y, theta, noParticles=100)

  # Plot the estimate and the true state
  par(mfrow=c(2, 1))
  plot(y, type="l", xlab="time", ylab="log-returns", bty="n",
    col="#1B9E77")
  plot(pfOutput$xHatFiltered, type="l", xlab="time",
    ylab="estimate of log-volatility", bty="n", col="#D95F02")

## End(Not run)
```

---

particleMetropolisHastings

Particle Metropolis-Hastings algorithm for a linear Gaussian state space model

Description

Estimates the parameter posterior for phi a linear Gaussian state space model of the form $x_t = \phi x_{t-1} + \sigma_v v_t$ and $y_t = x_t + \sigma_e e_t$, where v_t and e_t denote independent standard Gaussian random variables, i.e.$N(0, 1)$.

Usage

```
particleMetropolisHastings(y, initialPhi, sigmav, sigmae, noParticles,
  initialState, noIterations, stepSize)
```

Arguments

`y`	Observations from the model for $t = 1, ..., T$.
`initialPhi`	The mean of the log-volatility process μ.
`sigmav`	The standard deviation of the state process σ_v.
`sigmae`	The standard deviation of the observation process σ_e.
`noParticles`	The number of particles to use in the filter.
`initialState`	The inital state.
`noIterations`	The number of iterations in the PMH algorithm.
`stepSize`	The standard deviation of the Gaussian random walk proposal for ϕ.

Value

The trace of the Markov chain exploring the marginal posterior for ϕ.

Note

See Section 4 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
# Generates 100 observations from a linear state space model with
# (phi, sigma_e, sigma_v) = (0.5, 1.0, 0.1) and zero initial state.
theta <- c(0.5, 1.0, 0.1)
d <- generateData(theta, noObservations=100, initialState=0.0)

# Estimate the marginal posterior for phi
```

```
pmhOutput <- particleMetropolisHastings(d$y,
  initialPhi=0.1, sigmav=1.0, sigmae=0.1, noParticles=50,
  initialState=0.0, noIterations=1000, stepSize=0.10)

# Plot the estimate
nbins <- floor(sqrt(1000))
par(mfrow=c(1, 1))
hist(pmhOutput, breaks=nbins, main="", xlab=expression(phi),
  ylab="marginal posterior", freq=FALSE, col="#7570B3")
```

`particleMetropolisHastingsSVmodel`
Particle Metropolis-Hastings algorithm for a stochastic volatility model model

---

Description

Estimates the parameter posterior for $\theta = \{\mu, \phi, \sigma_v\}$ in a stochastic volatility model of the form $x_t = \mu + \phi(x_{t-1} - \mu) + \sigma_v v_t$ and $y_t = \exp(x_t/2)e_t$, where v_t and e_t denote independent standard Gaussian random variables, i.e. $N(0, 1)$.

Usage

```
particleMetropolisHastingsSVmodel(y, initialTheta, noParticles,
  noIterations, stepSize)
```

Arguments

`y`	Observations from the model for $t = 1, ..., T$.
`initialTheta`	An inital value for the parameters $\theta = \{\mu, \phi, \sigma_v\}$. The mean of the log-volatility process is denoted μ. The persistence of the log-volatility process is denoted ϕ. The standard deviation of the log-volatility process is denoted σ_v.
`noParticles`	The number of particles to use in the filter.
`noIterations`	The number of iterations in the PMH algorithm.
`stepSize`	The standard deviation of the Gaussian random walk proposal for θ.

Value

The trace of the Markov chain exploring the posterior of θ.

Note

See Section 5 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
## Not run:
  # Get the data from Quandl
  library("Quandl")
  d <- Quandl("NASDAQOMX/OMXS30", start_date="2012-01-02",
              end_date="2014-01-02", type="zoo")
  y <- as.numeric(100 * diff(log(d$"Index Value")))

  # Estimate the marginal posterior for phi
  pmhOutput <- particleMetropolisHastingsSVmodel(y,
    initialTheta = c(0, 0.9, 0.2),
    noParticles=500,
    noIterations=1000,
    stepSize=diag(c(0.05, 0.0002, 0.002)))

  # Plot the estimate
  nbins <- floor(sqrt(1000))
  par(mfrow=c(3, 1))
  hist(pmhOutput$theta[,1], breaks=nbins, main="", xlab=expression(mu),
    ylab="marginal posterior", freq=FALSE, col="#7570B3")
  hist(pmhOutput$theta[,2], breaks=nbins, main="", xlab=expression(phi),
    ylab="marginal posterior", freq=FALSE, col="#E7298A")
  hist(pmhOutput$theta[,3], breaks=nbins, main="",
    xlab=expression(sigma[v]), ylab="marginal posterior",
    freq=FALSE, col="#66A61E")

## End(Not run)
```

`particleMetropolisHastingsSVmodelReparameterised`

Particle Metropolis-Hastings algorithm for a stochastic volatility model model

---

Description

Estimates the parameter posterior for $\theta = \{\mu, \phi, \sigma_v\}$ in a stochastic volatility model of the form $x_t = \mu + \phi(x_{t-1} - \mu) + \sigma_v v_t$ and $y_t = \exp(x_t/2)e_t$, where v_t and e_t denote independent standard Gaussian random variables, i.e. $N(0, 1)$. In this version of the PMH, we reparameterise the model and run the Markov chain on the parameters $\vartheta = \{\mu, \psi, \varsigma\}$, where $\phi = \tanh(\psi)$ and $sigma_v = \exp(\varsigma)$.

Usage

```
particleMetropolisHastingsSVmodelReparameterised(y, initialTheta,
  noParticles, noIterations, stepSize)
```

Arguments

`y`	Observations from the model for $t = 1, ..., T$.
`initialTheta`	An inital value for the parameters $\theta = \{\mu, \phi, \sigma_v\}$. The mean of the log-volatility process is denoted μ. The persistence of the log-volatility process is denoted ϕ. The standard deviation of the log-volatility process is denoted σ_v.
`noParticles`	The number of particles to use in the filter.
`noIterations`	The number of iterations in the PMH algorithm.
`stepSize`	The standard deviation of the Gaussian random walk proposal for θ.

Value

The trace of the Markov chain exploring the posterior of θ.

Note

See Section 5 in the reference for more details.

Author(s)

Johan Dahlin <uni@johandahlin.com>

References

Dahlin, J. & Schon, T. B. "Getting Started with Particle Metropolis-Hastings for Inference in Nonlinear Dynamical Models." Journal of Statistical Software, Code Snippets, 88(2): 1–41, 2019.

Examples

```
## Not run:
  # Get the data from Quandl
  library("Quandl")
  d <- Quandl("NASDAQOMX/OMXS30", start_date="2012-01-02",
              end_date="2014-01-02", type="zoo")
  y <- as.numeric(100 * diff(log(d$"Index Value")))

  # Estimate the marginal posterior for phi
  pmhOutput <- particleMetropolisHastingsSVmodelReparameterised(
    y, initialTheta = c(0, 0.9, 0.2), noParticles=500,
    noIterations=1000, stepSize=diag(c(0.05, 0.0002, 0.002)))

  # Plot the estimate
  nbins <- floor(sqrt(1000))
  par(mfrow=c(3, 1))
```

```
  hist(pmhOutput$theta[,1], breaks=nbins, main="", xlab=expression(mu),
    ylab="marginal posterior", freq=FALSE, col="#7570B3")
  hist(pmhOutput$theta[,2], breaks=nbins, main="", xlab=expression(phi),
    ylab="marginal posterior", freq=FALSE, col="#E7298A")
  hist(pmhOutput$theta[,3], breaks=nbins, main="",
    xlab=expression(sigma[v]), ylab="marginal posterior",
    freq=FALSE, col="#66A61E")

## End(Not run)
```

Index