Package ‘mixedsde’

April 19, 2016

Version 1.0

Date 2016-01-19

Title Estimation Methods for Stochastic Differential Mixed Effects Models

Author Charlotte Dion [aut, cre],Adeline Sansom [aut], Simone Hermann [aut]

Maintainer Charlotte Dion <charlotte.dion@imag.fr>

Depends R (>= 3.0.2),sde,moments,MASS,stats,graphics,methods

Imports plot3D,grDevices

Description Inference on stochastic differential models Ornstein-Uhlenbeck or Cox-Ingersoll-Ross, with one or two random effects in the drift function.

License GPL (>= 2)

URL http://www.r-project.org

R topics documented:

`mixedsde-package` *Density estimation in mixed stochastic differential models*

Description

This package proposes 3 methods for density estimation in the special context of stochastic differential equation with linear random effects in the drift.

Details

Package:	mixedsde
Type:	Package
Version:	1.0
Date:	2016-04-19
License:	GLP-2, GLP-3

An overview of how to use the package, including the most important functions

Author(s)

Charlotte Dion, Simone Hermann, Adeline Samson

Maintainer: Charlotte Dion <charlotte.dion@imag.fr>

References

See Bidimensional random effect estimation in mixed stochastic differential model, C. Dion and V. Genon-Catalot, *Stochastic Inference for Stochastic Processes 2015, Springer Netherlands* **1–28**

Maximum likelihood estimation for stochastic differential equations with random effects, M. Delattre, V. Genon-Catalot and A. Samson, *Scandinavian Journal of Statistics 2012*, Vol 40, **322–343**

Bayesian Prediction of Crack Growth Based on a Hierarchical Diffusion Model. S. Hermann, K. Ickstadt and C. Mueller, *appearing in: Applied Stochastic Models in Business and Industry 2016*.

Examples

```
# Frequentist estimation, two random effects

model = 'CIR'; M <- 200;  T <- 10 ; delta <- 0.001; N <- floor(T/delta); sigma <- 0.01
random <- c(1,2); density.phi <- 'gammainvgamma2'
param<- c(1.8, 0.8, 8, 0.05);
simu <- mixedsde.sim(M=M,T=T,N=N,model=model,random=random,density.phi=density.phi,param=param,
              sigma=sigma, invariant = 1)
X <- simu$X ; phi <- simu$phi; times <- simu$times
estim.method<- 'nonparam'
estim <- mixedsde.fit(times=times, X=X, model=model, random=random, estim.method= 'nonparam')
outputsNP <- out(estim)

summary(estim)
print(estim)
## Not run:
plot(estim)
validation <- valid(estim, numj=floor(runif(1,1,M)))

## End(Not run)
estim.method<-'paramML'
estim_param <- mixedsde.fit(times= times, X= X, model= model, random= random,
estim.method = 'paramML')
outputsP <- out(estim_param)

summary(estim_param)

## Not run:
plot(estim_param)
test1 <- pred(estim, invariant = 1)
test2 <- pred(estim_param, invariant = 1)

## End(Not run)
cutoff <- outputsNP$cutoff
phihat <- outputsNP$estimphi
phihat_trunc <- outputsNP$estimphi_trunc
par(mfrow=c(1,2))
plot.ts(phi[1,], phihat[1,], xlim=c(0, 15), ylim=c(0,15), pch=18); abline(0,1)
points(phi[1,]*(1-cutoff), phihat[1,]*(1-cutoff), xlim=c(0, 20), ylim=c(0,20), pch=18, col='red')
abline(0,1)
plot.ts(phi[2,], phihat[2,], xlim=c(0, 15), ylim=c(0,15),pch=18); abline(0,1)
points(phi[2,]*(1-cutoff), phihat[2,]*(1-cutoff), xlim=c(0, 20), ylim=c(0,20), pch=18, col='red')
abline(0,1)



# Parametric Bayesian estimation one random effect

model <- 'OU'; random <- 1; sigma <- 0.1; fixed <- 5
M <- 50 ; T <- 1; N <- 100
density.phi <- 'normal'; param <- c(3, 0.5)
```

```
simu <- mixedsde.sim(M, T = T, N = N, model= model, random = random, fixed = fixed,
      density.phi= density.phi, param= param, sigma= sigma, X0 = 0)
X <- simu$X; phi <- simu$phi; times <- simu$times
#plot(times, X[1,], ylim = range(X), type = 'l'); for(i in 2:M) lines(times, X[i,])

estim_Bayes_withoutprior <- mixedsde.fit(times, X= X, model = model, random = random,
            estim.method = 'paramBayes', nMCMC = 100)  # nMCMC should be much larger
plot(estim_Bayes_withoutprior)

prior <- list(m = c(param[1], fixed), v = c(param[1], fixed), alpha.omega = 11,
               beta.omega = param[2]^2*10, alpha.sigma = 10, beta.sigma = sigma^2*9)
estim_Bayes <- mixedsde.fit(times, X = X, model = model, random = random,
              estim.method = 'paramBayes', prior = prior, nMCMC = 100)

plot(estim_Bayes)
outputBayes <- out(estim_Bayes)
summary(outputBayes)
(results_Bayes <- summary(estim_Bayes))
plot(estim_Bayes, style = 'cred.int', true.phi = phi)
plot(estim_Bayes_withoutprior, style = 'cred.int', true.phi = phi, reduced = TRUE)

plot2compare(estim_Bayes, estim_Bayes_withoutprior, names = c('with prior', 'without prior'))

print(estim_Bayes)

## Not run:
pred.result <- pred(estim_Bayes)
summary(out(pred.result))
plot(pred.result)

pred.result.trajectories <- pred(estim_Bayes, trajectories = TRUE)

validbayes <- valid(estim_Bayes, numj = 1)

## End(Not run)
```

`ad.propSd` *Adaptation For The Proposal Variance*

Description

Calculation of new proposal standard deviation

Usage

```
ad.propSd(chain, propSd, iteration, lower = 0.3, upper = 0.6,
  delta.n = function(n) min(0.1, 1/sqrt(n)))
```

Arguments

`chain`	vector of Markov chain samples
`propSd`	old proposal standard deviation
`iteration`	number of current iteration

`lower`	lower bound
`upper`	upper bound
`delta.n`	function for adding/subtracting from the log propSd

References

Rosenthal, J. S. (2011). Optimal proposal distributions and adaptive MCMC. Handbook of Markov Chain Monte Carlo, 93-112.

`ad.propSd_random` *Adaptation For The Proposal Variance*

Description

Calculation of new proposal standard deviation for the random effects

Usage

```
ad.propSd_random(chain, propSd, iteration, lower = 0.3, upper = 0.6,
  delta.n = function(n) min(0.1, 1/sqrt(n)))
```

Arguments

`chain`	matrix of Markov chain samples
`propSd`	old proposal standard deviation
`iteration`	number of current iteration
`lower`	lower bound
`upper`	upper bound
`delta.n`	function for adding/subtracting from the log propSd

References

Rosenthal, J. S. (2011). Optimal proposal distributions and adaptive MCMC. Handbook of Markov Chain Monte Carlo, 93-112.

`Bayes.fit-class` *S4 class for the Bayesian estimation results*

Description

S4 class for the Bayesian estimation results

Slots

`sigma2` vector of posterior samples for σ^2

`mu` matrix of posterior samples for μ

`omega` matrix of posterior samples for ω

`alpha` matrix of posterior samples for α

`beta` matrix of posterior samples for β

`random` 1, 2 or c(1,2)

`burnIn` proposal for the burn-in phase

`thinning` proposal for the thinning rate

`model` 'OU' or 'CIR'

`prior` list of prior values, input variable or calculated by the first 10% of series

`times` vector of observation times, storage of input variable

`X` matrix of observations, storage of input variable

`ind.4.prior` indices of series used for the prior parameter calculation, if prior knowledge is availabe it is set to M+1

`Bayes.pred-class` *S4 class for the Bayesian prediction results*

Description

S4 class for the Bayesian prediction results

Slots

`phi.pred` matrix of predictive samples for the random effect

`Xpred` matrix of predictive samples for observations

`coverage.rate` amount of covering prediction intervals

`qu.u` upper prediction interval bound

`qu.l` lower prediction interval bound

`estim` list of Bayes.fit object entries, storage of input variable

BayesianNormal *Bayesian Estimation In Mixed Stochastic Differential Equations*

Description

Gibbs sampler for Bayesian estimation of the random effects (α_j, β_j) in the mixed SDE $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma a(X_j(t))dW_j(t)$.

Usage

```
BayesianNormal(times, X, model = c("OU", "CIR"), prior, start, random,
  nMCMC = 1000, propSd = 0.2)
```

Arguments

`times`	vector of observation times
`X`	matrix of the M trajectories (each row is a trajectory with $N = T/\Delta$ column).
`model`	name of the SDE: 'OU' (Ornstein-Uhlenbeck) or 'CIR' (Cox-Ingersoll-Ross).
`prior`	list of prior parameters: mean and variance of the Gaussian prior on the mean mu, shape and scale of the inverse Gamma prior for the variances omega, shape and scale of the inverse Gamma prior for sigma
`start`	list of starting values: mu, sigma
`random`	random effects in the drift: 1 if one additive random effect, 2 if one multiplicative random effect or c(1,2) if 2 random effects.
`nMCMC`	number of iterations of the MCMC algorithm
`propSd`	proposal standard deviation of ϕ is $\lvert\mu\rvert *$`propSd`$/\log(N)$ at the beginning, is adjusted when acceptance rate is under 30% or over 60%

Value

`alpha`	posterior samples (Markov chain) of α
`beta`	posterior samples (Markov chain) of β
`mu`	posterior samples (Markov chain) of μ
`omega`	posterior samples (Markov chain) of Ω
`sigma2`	posterior samples (Markov chain) of σ^2

References

Hermann, S., Ickstadt, K. and C. Mueller (2016). Bayesian Prediction of Crack Growth Based on a Hierarchical Diffusion Model. *Appearing in: Applied Stochastic Models in Business and Industry*.

Rosenthal, J. S. (2011). 'Optimal proposal distributions and adaptive MCMC.' Handbook of Markov Chain Monte Carlo (2011): 93-112.

bx *Computation Of The Drift Coefficient*

Description

Computation of the drift coefficient

Usage

```
bx(x, fixed, random)
```

Arguments

`x`	vector of data
`fixed`	drift constant in front of X (when there is one additive random effect), 0 otherwise
`random`	1 if there is one additive random effect, 2 one multiplicative random effect or c(1,2) for 2 random effects

Value

`b`	The drift is $b(x, \phi) = \phi_1 b_1(x) + \phi_2 b_2(x)$, the output is b_2 except when random c(1,2) then the output is the vector $(b_1, b_2)^t$

chain2samples *Removing Of Burn-in Phase And Thinning*

Description

Transfers class object Bayes.fit from the original to the thinned chains

Usage

```
chain2samples(res, burnIn, thinning)
```

Arguments

`res`	Bayes.fit class object
`burnIn`	number of burn-in samples
`thinning`	thinning rate

dcCIR2 — *Likelihood Function For The CIR Model*

Description

Likelihood

Usage

```
dcCIR2(x, t, x0, theta, log = FALSE)
```

Arguments

`x`	current observation
`t`	time of observation
`x0`	starting point, i.e. observation in time 0
`theta`	parameter (α, β, σ)
`log`	logical(1) if TRUE, log likelihood

References

Iacus, S. M. (2008). Simulation and Inference for Stochastic Differential Equations.

diagnostic — *Calcucation Of Burn-in Phase And Thinning Rate*

Description

Proposal for burn-in and thin rate

Usage

```
diagnostic(results, random)
```

Arguments

`results`	Bayes.fit class object
`random`	one out of 1, 2, c(1,2)

`discr` *Simulation Of Random Variables*

Description

Simulation of (discrete) random variables from a vector of probability (the nonparametrically estimated values of the density renormalised to sum at 1) and a vectors of real values (the grid of estimation)

Usage

```
discr(x, p)
```

Arguments

`x`	n real numbers
`p`	vector of probability, length n

Value

y a simulated value from the discrete distribution

`eigenvaluesV` *Matrix Of Eigenvalues Of A List Of Symetric Matrices*

Description

Computation of the eigenvalues of each matrix Vj in the case of two random effects (random =c(1,2)), done via `eigen`

Usage

```
eigenvaluesV(V)
```

Arguments

`V`	list of matrices Vj

Value

`eigenvalues`	Matrix of 2 rows and as much columns as matrices V

References

See Bidimensional random effect estimation in mixed stochastic differential model, C. Dion and V. Genon-Catalot, *Stochastic Inference for Stochastic Processes 2015, Springer Netherlands*, **1–28**

EstParamNormal	*Maximization Of The Log Likelihood In Mixed Stochastic Differential Equations*

Description

Maximization of the loglikelihood of the mixed SDE with Normal distribution of the random effects $dXj(t) = (\alpha_j - \beta_j Xj(t))dt + \sigma a(Xj(t))dWj(t)$, done with `likelihoodNormal`

Usage

```
EstParamNormal(U, V, K, random, estim.fix, fixed = 0)
```

Arguments

`U`	matrix of M sufficient statistics U
`V`	list of the M sufficient statistics matrix V
`K`	number of times of observations
`random`	random effects in the drift: 1 if one additive random effect, 2 if one multiplicative random effect or c(1,2) if 2 random effects.
`estim.fix`	1 if the fixed parameter is estimated, when random 1 or 2 , 0 otherwise
`fixed`	value of the fixed parameter if known (not estimated)

Value

`mu`	estimated value of the mean
`Omega`	estimated value of the variance
`BIChere`	BIC indicator
`AIChere`	AIC indicator

Freq.fit-class	*S4 class for the frequentist estimation results*

Description

S4 class for the frequentist estimation results

Slots

`model` character 'OU' or 'CIR'

`random` numeric 1, 2, or c(1,2)

`fixed` numeric value of the fixed effect if there is one

`gridf` matrix of values on which the estimated is done

`mu` numeric MLE estimator for parametric approach

`omega` numeric MLE estimator for parametric approach `cutoff` value of the cutoff if there is one

`sigma2` numeric estimated value of σ^2

`estimf.trunc` matrix estimator of the density of ϕ for the truncated estimateur of the random effects

`estimphi.trunc` matrix truncated estimator of the random effects

`index` index of the used trajectories

`estimphi` matrix of the estimator of the random effects

`estimf` estimator of the density of ϕ

`estim.fixed` estimator of the fixed parameter if option estim.fix = 1

`estim.fix` 1 if the user asked for the estimation of fixed parameter

`bic` numeric bic

`aic` numeric aic

`times` vector of observation times, storage of input variable

`X` matrix of observations, storage of input variable

`likelihoodNormal` *Computation Of The Log Likelihood In Mixed Stochastic Differential Equations*

Description

Computation of -2 loglikelihood of the mixed SDE with Normal distribution of the random effects $dXj(t) = (\alpha_j - \beta_j Xj(t))dt + \sigma a(Xj(t))dWj(t)$.

Usage

```
likelihoodNormal(mu, omega, U, V, estimphi, random)
```

Arguments

`mu`	current value of the mean of the normal distribution
`omega`	current value of the standard deviation of the normal distribution
`U`	vector of the M sufficient statistics U (see `UV`)
`V`	vector of the M sufficient statistics V (see `UV`)
`estimphi`	vector or matrix of estimators of the random effects
`random`	random effects in the drift: 1 if one additive random effect, 2 if one multiplicative random effect or c(1,2) if 2 random effects.

Value

`L`	value of -2 x loglikelihood

References

Maximum likelihood estimation for stochastic differential equations with random effects, M. Delattre, V. Genon-Catalot and A. Samson, *Scandinavian Journal of Statistics 2012*, Vol 40, **322–343**

`likelihoodNormalestimfix` *Likelihood Function When The Fixed Effect Is Estimated*

Description

Computation of -2 loglikelihood of the mixed SDE with Normal distribution of the random effects when the fixed effect is estimated for random 1 or 2 $dXj(t) = (\alpha_j - \beta_j Xj(t))dt + \sigma a(Xj(t))dWj(t)$.

Usage

```
likelihoodNormalestimfix(mu1, mu2, omega, U, V, estimphi, random)
```

Arguments

`mu1`	current value of the mean of the first effect
`mu2`	current value of the mean of the second effect
`omega`	current value of the standard deviation of the normal distribution
`U`	vector of the M sufficient statistics U (see `UV`)
`V`	vector of the M sufficient statistics V (see `UV`)
`estimphi`	vector or matrix of estimators of the random effects
`random`	random effects in the drift: 1 if one additive random effect, 2 if one multiplicative random effect or c(1,2) if 2 random effects.

Value

`L`	value of -2 x loglikelihood

References

Maximum likelihood estimation for stochastic differential equations with random effects, M. Delattre, V. Genon-Catalot and A. Samson, *Scandinavian Journal of Statistics 2012*, Vol 40, **322–343**

`mixedsde.fit` *Estimation Of The Random Effects In Mixed Stochastic Differential Equations*

Description

Estimation of the random effects (α_j, β_j) and of their density, parametrically or nonparametrically in the mixed SDE $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma a(X_j(t))dW_j(t)$.

Usage

```
mixedsde.fit(times, X, model = c("OU", "CIR"), random, fixed = 0,
  estim.fix = 0, estim.method = c("nonparam", "paramML", "paramBayes"),
  gridf = NULL, prior, nMCMC = NULL)
```

Arguments

`times`	vector of observation times
`X`	matrix of the M trajectories (each row is a trajectory with as much columns as observations)
`model`	name of the SDE: 'OU' (Ornstein-Uhlenbeck) or 'CIR' (Cox-Ingersoll-Ross)
`random`	random effects in the drift: 1 if one additive random effect, 2 if one multiplicative random effect or c(1,2) if 2 random effects
`fixed`	fixed effect in the drift: value of the fixed effect when there is only one random effect and it is not estimated, 0 otherwise
`estim.method`	estimation method: 'paramML' for a Gaussian parametric estimation by maximum likelihood, 'paramBayes' for a Gaussian parametric Bayesian estimation or 'nonparam' for a non-parametric estimation
`gridf`	if nonparametric estimation: grid of values on which the density is estimated, a matrix with two rows if two random effects; NULL by default and then grid is chosen as a function of the estimated values of the random effects. For the plots this grid is used.
`estim.fix`	default 0, 1 if random = 1 or 2, method = 'paramML' and an estimator of the fixed parameter is needed (to lead the nonparametric estimation after for example)
`prior`	if method = 'paramBayes', list of prior parameters: mean and variance of the Gaussian prior on the mean mu, shape and scale of the inverse Gamma prior for the variances omega, shape and scale of the inverse Gamma prior for sigma
`nMCMC`	if method = 'paramBayes', number of iterations of the MCMC algorithm

Details

Estimation of the random effects density from M independent trajectories of the SDE (the Brownian motions W_j are independent), with linear drift. Two diffusions are implemented, with one or two random effects:

Ornstein-Uhlenbeck model (OU):
If random = 1, β is a fixed effect: $dX_j(t) = (\alpha_j - \beta X_j(t))dt + \sigma dW_j(t)$
If random = 2, α is a fixed effect: $dX_j(t) = (\alpha - \beta_j X_j(t))dt + \sigma dW_j(t)$
If random = c(1,2), $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma dW_j(t)$

Cox-Ingersoll-Ross model (CIR):
If random = 1, β is a fixed effect: $dX_j(t) = (\alpha_j - \beta X_j(t))dt + \sigma\sqrt{X_(t)}dWj_(t)$
If random = 2, α is a fixed effect: $dX_j(t) = (\alpha - \beta_j X_j(t))dt + \sigma\sqrt{X_j(t)}dW_j(t)$
If random = c(1,2), $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma\sqrt{X_j(t)}dW_j(t)$

The nonparametric method estimates the density of the random effects with a kernel estimator (one-dimensional or two-dimensional density). The parametric method estimates the mean and standard deviation of the Gaussian distribution of the random effects.

Value

`index`	is the vector of subscript in $1, ..., M$ where the estimation of phi has been done, most of the time $index = 1 : M$

`estimphi`	matrix of estimators of $\phi = \alpha, or \beta, or (\alpha, \beta)$ from the efficient statitics (see UV), matrix of two lines if random =c(1,2), numerical type otherwise
`estim.fixed`	if estim.fix is TRUE and random = 1 or 2, estimator of $\phi = \alpha, or \beta$ from the efficient statitics (see UV), 0 otherwise
`gridf`	grid of values on which the estimated is done for the nonparametric method, otherwise, grid used for the plots, matrix form
`estimf`	estimator of the density of ϕ from a kernel estimator from package: stats, function: density, or package: MASS, function: kde2D. Matrix form: one line if one random effect or square matrix otherwise

If there is a truncation threshold in the estimator

`cutoff`	the binary vector of cutoff, FALSE otherwise
`estimphi.trunc`	troncated estimator of ϕ, vector or matrix of 0 if we do not use truncation, matrix of two lines if random =c(1,2), numerical type otherwise
`estimf.trunc`	troncated estimator of the density of ϕ, vector or matrix of 0 if we do not use truncation, matrix if random =c(1,2), numerical type otherwise

For the parametric maximum likelihood estimation

`mu`	estimator of the mean of the random effects normal density, 0 if we do nonparametric estimation
`omega`	estimator of the standard deviation of the random effects normal density, 0 if we do nonparametric estimation
`bic`	BIC criterium, 0 if we do nonparametric estimation
`aic`	AIC criterium, 0 if we do nonparametric estimation
`model`	initial choice
`random`	initial choice
`fixed`	initial choice
`times`	initial choice
`X`	initial choice

For the parametric Bayesian estimation

`alpha`	posterior samples (Markov chain) of α
`beta`	posterior samples (Markov chain) of β
`mu`	posterior samples (Markov chain) of μ
`omega`	posterior samples (Markov chain) of Ω
`sigma2`	posterior samples (Markov chain) of σ^2
`model`	initial choice
`random`	initial choice
`burnIn`	proposal for burn-in period
`thinning`	proposal for thinning rate
`prior`	initial choice or calculated by the first 10% of series
`times`	initial choice
`X`	initial choice
`ind.4.prior`	in the case of calculation of prior parameters: the indices of used series

References

For the parametric estimation see: Maximum likelihood estimation for stochastic differential equations with random effects, M. Delattre, V. Genon-Catalot and A. Samson, *Scandinavian Journal of Statistics 2012*, Vol 40, **322–343**

Bayesian Prediction of Crack Growth Based on a Hierarchical Diffusion Model. S. Hermann, K. Ickstadt and C. Mueller, *appearing in: Applied Stochastic Models in Business and Industry 2016*.

For the nonparametric estimation see:

Nonparametric estimation for stochastic differential equations with random effects, F. Comte, V. Genon-Catalot and A. Samson, *Stochastic Processes and Their Applications 2013*, Vol 7, **2522–2551**

Estimation for stochastic differential equations with mixed effects, V. Genon-Catalot and C. Laredo 2014 *e-print: hal-00807258*

Bidimensional random effect estimation in mixed stochastic differential model, C. Dion and V. Genon-Catalot, *Stochastic Inference for Stochastic Processes 2015, Springer Netherlands*, **1–28**

Examples

```
# Frequentist estimation
# Two random effects
model = 'CIR'; M <- 200;  T <- 10 ; delta <- 0.001; N <- floor(T/delta); sigma <- 0.01 ;
random <- c(1,2); density.phi <- 'gammainvgamma2'; param<- c(1.8, 0.8, 8, 0.05);
simu <- mixedsde.sim(M=M, T=T, N=N, model=model,random=random, density.phi=density.phi,
               param=param, sigma=sigma, invariant = 1)
X <- simu$X ; phi <- simu$phi; times <- simu$times
estim.method<- 'nonparam'
estim <- mixedsde.fit(times=times, X=X, model=model, random=random, estim.method= 'nonparam')
#To stock the results of the function, use method \code{out}
#which put the outputs of the function on a list according to the frequentist or
# Bayesian approach.
outputsNP <- out(estim)
# Not run
## Not run:
plot(estim)
## End(Not run)
# It represents the bidimensional density, the histogram of the first estimated random
# effect \eqn{\alpha} with the  marginal of \eqn{\hat{f}} from the first coordonate which
# estimates  the density of \eqn{\alpha}. And the same for the second random effect
# \eqn{\beta}. More, it plots a qq-plot for the sample of estimator of the random effects
# compared with the quantiles of a Gaussian sample with the same mean and standard deviation.

summary(estim)
print(estim)
# Validation
# If numj is fixed by the user: this function simulates Mrep =100 (by default) new
# trajectories with the value of the estimated random effect. Then it plots on the
# left graph the Mrep new trajectories \eqn{(Xnumj^{k}(t1), ... Xnumj^{k}(tN)),
# k= 1, ... Mrep} with in red the true trajectory \eqn{(Xnumj(t1), ... Xnumj(tN))}.
#The right graph is a qq-plot of the quantiles of samples
# \eqn{(Xnumj^{1}(ti), ... Xnumj^{Mrep}(ti))}
# for each time \eqn{ti} compared with the uniform quantiles. The outputs of the function
# are: a matrix \code{Xnew} dimension Mrepx N+1, vector of quantiles \code{quantiles} length
# N and the number of the trajectory for the plot \code{plotnumj= numj}
# If numj is not precised by the user, then, this function simulates Mrep =100 (by default)
# new trajectories for each estimated random effect. Then left graph is a plot of the Mrep
```

```
# new trajectories \eqn{(Xj^{k}(t1), ... Xj^{k}(tN)), k= 1, ... Mrep}
#for a randomly chosen number j with in red the true trajectory \eqn{(Xj(t1), ... Xj(tN))}.
#The right graph is a qq-plot of the quantiles of samples \eqn{(Xj^{1}(ti), ... Xj^{Mrep}(ti))},
# for the same j and for each time \eqn{ti}. The outputs of the function are: a list of
# matrices \code{Xnew} length M, matrix of quantiles \code{quantiles} dimension MxN
# and the number of the trajectory for the plot \code{plotnumj}

validation <- valid(estim,  numj=floor(runif(1,1,M)))

# Parametric estimation
estim.method<-'paramML'
estim_param <- mixedsde.fit(times= times, X= X, model= model, random= random,
           estim.method = 'paramML')
outputsP <- out(estim_param)

#plot(estim_param)
summary(estim_param)

# Prediction for the frequentist approach
# This function uses the estimation of the density function to simulate a
# new sample of random effects according to this density. If \code{plot.pred =1} (default)
# is plots on the top the predictive random effects versus the estimated random effects
# from the data. On the bottom, the left graph is the true trajectories, on the right
#the predictive trajectories and the empiric prediciton intervals at level
# \code{level=0.05} (defaut). The function return on a list the prediction of phi
# \code{phipred}, the prediction of X \code{Xpred}, and the indexes of the
# corresponding true trajectories \code{indexpred}

# Not run
## Not run:
test1 <- pred(estim,  invariant  = 1)
test2 <- pred(estim_param, invariant  = 1)

## End(Not run)
# More graph
fhat <- outputsNP$estimf
fhat_trunc <- outputsNP$estimf.trunc
fhat_param <- outputsP$estimf

gridf <- outputsNP$gridf; gridf1 <- gridf[1,]; gridf2 <- gridf[2,]

marg1 <- ((max(gridf2)-min(gridf2))/length(gridf2))*apply(fhat,1,sum)
marg1_trunc <- ((max(gridf2)-min(gridf2))/length(gridf2))*apply(fhat_trunc,1,sum)
marg2 <- ((max(gridf1)-min(gridf1))/length(gridf1))*apply(fhat,2,sum)
marg2_trunc <- ((max(gridf1)-min(gridf1))/length(gridf1))*apply(fhat_trunc,2,sum)

marg1_param <- ((max(gridf2)-min(gridf2))/length(gridf2))*apply(fhat_param,1,sum)
marg2_param <- ((max(gridf1)-min(gridf1))/length(gridf1))*apply(fhat_param,2,sum)
f1 <-  (gridf1^(param[1]-1))*exp(-gridf1/param[2])/((param[2])^param[1]*gamma(param[1]))
f2 <-  (gridf2^(-param[3]-1)) * exp(-(1/param[4])*(1/gridf2)) *
 ((1/param[4])^param[3])*(1/gamma(param[3]))
par(mfrow=c(1,2))
plot(gridf1,f1,type='l', lwd=1,  xlab='', ylab='')
lines(gridf1,marg1_trunc,col='blue', lwd=2)
lines(gridf1,marg1,col='blue', lwd=2, lty=2)
lines(gridf1,marg1_param,col='red', lwd=2, lty=2)
plot(gridf2,f2,type='l', lwd=1, xlab='', ylab='')
```

```
lines(gridf2,marg2_trunc,col='blue', lwd=2)
lines(gridf2,marg2,col='blue', lwd=2, lty=2)
lines(gridf2,marg2_param,col='red', lwd=2, lty=2)

cutoff <- outputsNP$cutoff
phihat <- outputsNP$estimphi
phihat_trunc <- outputsNP$estimphi.trunc
par(mfrow=c(1,2))
plot.ts(phi[1,], phihat[1,], xlim=c(0, 15), ylim=c(0,15), pch=18); abline(0,1)
points(phi[1,]*(1-cutoff), phihat[1,]*(1-cutoff), xlim=c(0, 20), ylim=c(0,20),pch=18, col='red');
abline(0,1)
plot.ts(phi[2,], phihat[2,], xlim=c(0, 15), ylim=c(0,15),pch=18); abline(0,1)
points(phi[2,]*(1-cutoff), phihat[2,]*(1-cutoff), xlim=c(0, 20), ylim=c(0,20),pch=18, col='red');
abline(0,1)

# one random effect:
## Not run:
model <-'OU'
random <- 1
M <- 80; T <- 100  ; N <- 2000
sigma <- 0.1 ; X0 <- 0
density.phi <- 'normal'
fixed <- 2 ; param <- c(1, 0.2)
#-------------------
#- simulation
simu <- mixedsde.sim(M,T=T,N=N,model=model,random=random, fixed=fixed,density.phi=density.phi,
               param=param, sigma=sigma, X0=X0)
X <- simu$X
phi <- simu$phi
times <-simu$times
plot(times, X[10,], type='l')

#- parametric estimation
estim.method<-'paramML'
estim_param <- mixedsde.fit(times, X=X, model=model, random=random, estim.fix= 1,
               estim.method=estim.method)
outputsP <- out(estim_param)
estim.fixed <- outputsP$estim.fixed #to compare with fixed
#or
estim_param2 <- mixedsde.fit(times, X=X, model=model, random=random, fixed = fixed,
             estim.method=estim.method)
outputsP2 <- out(estim_param2)
#- nonparametric estimation
estim.method <- 'nonparam'
estim <- mixedsde.fit(times, X, model=model, random=random, fixed = fixed,
           estim.method=estim.method)
outputsNP <- out(estim)

plot(estim)
print(estim)
summary(estim)

plot(estim_param)
print(estim_param)
summary(estim_param)

valid1 <- valid(estim,  numj=floor(runif(1,1,M)))
```

```
test1 <- pred(estim )
test2 <- pred(estim_param)

## End(Not run)

# Parametric Bayesian estimation
# one random effect
random <- 1; sigma <- 0.1; fixed <- 5; param <- c(3, 0.5)
sim <- mixedsde.sim(M = 50, T = 1, N = 100, model = 'OU', random = random, fixed = fixed,
       density.phi = 'normal',param= param, sigma= sigma, X0 = 0, op.plot = 1)

# here: only 100 iterations for example - should be much more!
estim_Bayes_withoutprior <- mixedsde.fit(times = sim$times, X = sim$X, model = 'OU',
             random, estim.method = 'paramBayes',  nMCMC = 100)
prior <- list( m = c(param[1], fixed), v = c(param[1], fixed), alpha.omega = 11,
            beta.omega = param[2]^2*10, alpha.sigma = 10, beta.sigma = sigma^2*9)
estim_Bayes <- mixedsde.fit(times = sim$times, X = sim$X, model = 'OU', random,
           estim.method = 'paramBayes', prior = prior, nMCMC = 100)

validation <- valid(estim_Bayes, numj = 10)
plot(estim_Bayes)
outputBayes <- out(estim_Bayes)
summary(outputBayes)
(results_Bayes <- summary(estim_Bayes))
plot(estim_Bayes, style = 'cred.int', true.phi = sim$phi)
plot(estim_Bayes_withoutprior, style = 'cred.int', true.phi = sim$phi, reduced = TRUE)

plot2compare(estim_Bayes, estim_Bayes_withoutprior, names = c('with prior', 'without prior'))

print(estim_Bayes)
## Not run:
pred.result <- pred(estim_Bayes)
summary(out(pred.result))
plot(pred.result)

pred.result.trajectories <- pred(estim_Bayes, trajectories = TRUE)

## End(Not run)
# second example
## Not run:
random <- 2; sigma <- 0.2; fixed <- 5; param <- c(3, 0.5)
sim <- mixedsde.sim(M = 20, T = 1, N = 100, model = 'CIR', random = random,
      fixed = fixed, density.phi = 'normal',param = param, sigma = sigma, X0 = 0.1, op.plot = 1)

prior <- list(m = c(fixed, param[1]), v = c(fixed, param[1]), alpha.omega = 11,
         beta.omega = param[2]^2*10, alpha.sigma = 10, beta.sigma = sigma^2*9)

estim_Bayes <- mixedsde.fit(times = sim$times, X = sim$X, model = 'CIR', random = random,
                 estim.method = 'paramBayes', prior = prior, nMCMC = 1000)
plot(estim_Bayes)
outputBayes <- out(estim_Bayes)
summary(outputBayes)
(results_Bayes <- summary(estim_Bayes))
plot(estim_Bayes, style = 'cred.int', true.phi = sim$phi, reduced = TRUE)

print(estim_Bayes)
pred.result <- pred(estim_Bayes)
```

```
summary(out(pred.result))
plot(pred.result)

## End(Not run)

# for two random effects
random <- c(1,2); sigma <- 0.1; param <- c(3, 0.5, 5, 0.2)

sim <- mixedsde.sim(M = 20, T = 1, N = 100, model = 'OU', random = random,
       density.phi = 'normalnormal', param = param, sigma = sigma, X0 = 0, op.plot = 1)

# here: only 200 iterations for example - should be much more!
estim_Bayes_withoutprior <- mixedsde.fit(times = sim$times, X = sim$X, model = 'OU',
             random = random, estim.method = 'paramBayes', nMCMC = 100)
plot(estim_Bayes_withoutprior, style = 'cred.int', true.phi = sim$phi, reduced = TRUE)

prior <- list(m = param[c(1,3)], v = param[c(1,3)], alpha.omega = c(11,11),
           beta.omega = param[c(2,4)]^2*10, alpha.sigma = 10, beta.sigma = sigma^2*9)
estim_Bayes <- mixedsde.fit(times = sim$times, X = sim$X, model = 'OU', random = random,
                estim.method = 'paramBayes', prior = prior, nMCMC = 100)
outputBayes <- out(estim_Bayes)
summary(outputBayes)
summary(estim_Bayes)
plot(estim_Bayes)
plot(estim_Bayes, style = 'cred.int', true.phi = sim$phi)
print(estim_Bayes)

pred.result <- pred(estim_Bayes)


# invariant case
## Not run:
random <- 1; sigma <- 0.1; fixed <- 5; param <- c(3, 0.5)
sim <- mixedsde.sim(M = 50, T = 5, N = 100, model = 'OU', random = random, fixed = fixed,
          density.phi = 'normal',param = param, sigma = sigma, invariant = 1, op.plot = 1)

prior <- list(m = c(param[1], fixed), v = c(param[1], 1e-05), alpha.omega = 11,
       beta.omega = param[2]^2*10, alpha.sigma = 10, beta.sigma = sigma^2*9)
estim_Bayes <- mixedsde.fit(times = sim$times, X = sim$X, model = 'OU', random,
       estim.method = 'paramBayes', prior = prior, nMCMC = 100)
plot(estim_Bayes)

pred.result <- pred(estim_Bayes, invariant = 1)
pred.result.traj <- pred(estim_Bayes, invariant = 1, trajectories = TRUE)

## End(Not run)
```

`mixedsde.sim` *Simulation Of A Mixed Stochastic Differential Equation*

Description

Simulation of M independent trajectories of a mixed stochastic differential equation (SDE) with linear drift and two random effects (α_j, β_j) $dX_j(t) = (\alpha_j - \beta_j X_(t))dt + \sigma a(X_j(t))dW_j(t)$, for $j = 1, ..., M$.

Usage

```
mixedsde.sim(M, T, N = 100, model, random, fixed = 0, density.phi, param,
  sigma, t0 = 0, X0 = 0.01, invariant = 0, delta = T/N, op.plot = 0,
  add.plot = FALSE)
```

Arguments

`M`	number of trajectories
`T`	horizon of simulation.
`N`	number of simulation steps, default Tx100.
`model`	name of the SDE: 'OU' (Ornstein-Uhlenbeck) or 'CIR' (Cox-Ingersoll-Ross).
`random`	random effects in the drift: 1 if one additive random effect, 2 if one multiplicative random effect or c(1,2) if 2 random effects.
`fixed`	fixed effects in the drift: value of the fixed effect when there is only one random effect, 0 otherwise. If random =2, fixed can be 0 but β has to be a non negative random variable for the estimation.
`density.phi`	name of the density of the random effects.
`param`	vector of parameters of the distribution of the two random effects.
`sigma`	diffusion parameter
`t0`	time origin, default 0.
`X0`	initial value of the process, default X0=0.
`invariant`	1 if the initial value is simulated from the invariant distribution, default 0.01 and X0 is fixed.
`delta`	time step of the simulation (T/N).
`op.plot`	1 if a plot of the trajectories is required, default 0.
`add.plot`	1 for add trajectories to an existing plot

Details

Simulation of M independent trajectories of the SDE (the Brownian motions Wj are independent), with linear drift. Two diffusions are implemented, with one or two random effects:

Ornstein-Uhlenbeck model (OU): If random = 1, β is a fixed effect: $dX_j(t) = (\alpha_j - \beta X_j(t))dt + \sigma dW_j(t)$
If random = 2, α is a fixed effect: $dX_j(t) = (\alpha - \beta_j X_j(t))dt + \sigma dW_j(t)$
If random = c(1,2), $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma dW_j(t)$

Cox-Ingersoll-Ross model (CIR): If random = 1, β is a fixed effect: $dX_j(t) = (\alpha_j - \beta X_j(t))dt + \sigma \sqrt{X_j(t)} dW_j(t)$
If random = 2, α is a fixed effect: $dX_j(t) = (\alpha - \beta_j X_j(t))dt + \sigma \sqrt{X_j(t)} dW_j(t)$
If random = c(1,2), $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma \sqrt{X_j(t)} dW_j(t)$

The initial value of each trajectory can be simulated from the invariant distribution of the process: Normal distribution with mean α/β and variance $\sigma^2/(2\beta)$ for the OU, a gamma distribution $\Gamma(2\alpha/\sigma^2, \sigma^2/(2\beta))$ for the C-I-R model.

Density of the random effects: Several densities are implemented for the random effects, depending on the number of random effects.

If two random effects, choice between

'normalnormal': Normal distributions for both α β and param=c(mean_α, sd_α, mean_β, sd_β)

'gammagamma': Gamma distributions for both α β and param=c(shape_α, scale_α, shape_β, scale_β)

'gammainvgamma': Gamma for α, Inverse Gamma for β and param=c(shape_α, scale_α, shape_β, scale_β)

'normalgamma': Normal for α, Gamma for β and param=c(mean_α, sd_α, shape_β, scale_β)

'normalinvgamma': Normal for α, Inverse Gamma for β and param=c(mean_α, sd_α, shape_β, scale_β)

'gammagamma2': Gamma $+2 * \sigma^2$ for α, Gamma $+1$ for β and param=c(shape_α, scale_α, shape_β, scale_β)

'gammainvgamma2': Gamma $+2 * \sigma^2$ for α, Inverse Gamma for β and param=c(shape_α, scale_α, shape_β, scale_β)

If only α is random, choice between

'normal': Normal distribution with param=c(mean, sd)

lognormal': logNormal distribution with param=c(mean, sd)

'mixture.normal': mixture of normal distributions $pN(\mu 1, \sigma 1^2) + (1-p)N(\mu 2, \sigma 2^2)$ with param=c(p, $\mu 1, \sigma 1, \mu 2, \sigma 2$)

'gamma': Gamma distribution with param=c(shape, scale)

'mixture.gamma': mixture of Gamma distribution $p\Gamma(shape1, scale1) + (1-p)\Gamma(shape2, scale2)$ with param=c(p, shape1, scale1, shape2, scale2)

'gamma2': Gamma distribution $+2 * \sigma^2$ with param=c(shape, scale)

'mixed.gamma2': mixture of Gamma distribution $p\Gamma(shape1, scale1) + (1-p)\Gamma(shape2, scale2)$ + $+2 * \sigma^2$ with param=c(p, shape1, scale1, shape2, scale2)

If only β is random, choice between 'normal': Normal distribution with param=c(mean, sd)

'gamma': Gamma distribution with param=c(shape, scale)

'mixture.gamma': mixture of Gamma distribution $p\Gamma(shape1, scale1) + (1-p)\Gamma(shape2, scale2)$ with param=c(p, shape1, scale1, shape2, scale2)

Value

`X`	matrix (M x (N+1)) of the M trajectories.
`phi`	vector (or matrix) of the M simulated random effects.

References

This function mixedsde.sim is based on the package sde, function sde.sim. See Simulation and Inference for stochastic differential equation, S.Iacus, *Springer Series in Statistics 2008* Chapter 2

See Also

http://cran.r-project.org/package=sde

Examples

```
#Simulation of 5 trajectories of the OU SDE with random =1, and a Gamma distribution.

simuOU <- mixedsde.sim(M=5, T=10,N=1000,model='OU', random=1,fixed=0.5,
```

```
density.phi='gamma', param=c(1.8, 0.8) , sigma=0.1,op.plot=1)
X <- simuOU$X ;
phi <- simuOU$phi
hist(phi)
```

---

mixture.sim *Simulation Of A Mixture Of Two Normal Or Gamma Distributions*

Description

Simulation of M random variables from a mixture of two Gaussian or Gamma distributions

Usage

```
mixture.sim(M, density.phi, param)
```

Arguments

M	number of simulated variables
density.phi	name of the chosen density 'mixture.normal' or 'mixture.gamma'
param	vector of parameters with the proportion of mixture of the two distributions and means and standard-deviations of the two normal or shapes and scales of the two Gamma distribution

Details

If 'mixture.normal', the distribution is $pN(\mu1, \sigma1^2) + (1-p)N(\mu2, \sigma2^2)$

and param=c(p, $\mu1, \sigma1, \mu2, \sigma2$)

If 'mixture.gamma', the distribution is $pGamma(shape1, scale1)+(1-p)Gamma(shape2, scale2)$

and param=c(p, shape1, scale1, shape2, scale2)

Value

Y	vector of simulated variables

Examples

```
density.phi <- 'mixture.gamma'
param <- c(0.2,1.8,0.5,5.05,1); M <- 200
gridf <- seq(0, 8, length = 200)
f <- param[1] * 1/gamma(param[2]) * (gridf)^(param[2]-1) *
          exp(-(gridf) / param[3]) / param[3]^param[2] +
(1-param[1]) * 1/gamma(param[4]) * (gridf)^(param[4]-1) *
    exp(-(gridf) / param[5]) / param[5]^param[4]
Y <- mixture.sim(M, density.phi, param)
hist(Y)
lines(gridf, f)
```

neuronal.data *Trajectories Interspike Of A Single Neuron Of A Ginea Pig*

Description

The neuronal.data data has 240 measurements of the membrane potential in volts for one single neuron of a pig between the spikes, along time, with 2000 points for each. The step time is $delta = 0.00015$ s.

Usage

```
neuronal.data
```

Format

This data frame has a list form of length 2. The first element in the matrix named Xreal. Each row is a trajectory, that one can model by a diffusion process with random effect. The realisation can be assumed independent. The second element is a vector of times of observations times

Source

The parameters of the stochastic leaky integrate-and-fire neuronal model. Lansky, P., Sanda, P. and He, J. (2006). *Journal of Computational Neuroscience* Vol 21, **211–223**

References

The parameters of the stochastic leaky integrate-and-fire neuronal model. Lansky, P., Sanda, P. and He, J. (2006). *Journal of Computational Neuroscience* Vol 21, **211–223**

Examples

```
require(plot3D)
model <- "OU"
random <- c(1,2)
M <- 240     # number of trajectories, number of rows of the matrix of the data
T <- 0.3     # width of the interval of observation
delta <- 0.00015   # step time
N <- T/delta  # number of points in the time interval 2000
# load ("data/neuronal.data.rda")
data(neuronal.data)
X <- neuronal.data[[1]]
times <- neuronal.data[[2]]

#plot(times,X[10, ], type = 'l', xlab = 'time', ylab='', col = 'blue', ylim=c(0,0.016))

random <- c(1,2)

#- nonparametric estimation
estim.method <- 'nonparam'
estim <- mixedsde.fit(times=times, X=X, model=model, random=random,  estim.method='nonparam')

#- parametric estimation
estim.method<-'paramML'
```

```
estim_param <- mixedsde.fit(times=times, X=X, model=model, random= random, estim.method= 'paramML')

#- implemented methods
# plot(estim);
print(estim); #valid(estim)
print(estim_param); #plot(estim_param);  valid(estim_param)

#test1 <- pred(estim, X,  estim.method= 'nonparam',times = times)
#test2 <- pred(estim_param, X,estim.method= 'paramML', times = times)

#- Other possible plots
par(mfrow=c(1,2))

outputsNP <-  out(estim)
outputsP <- out(estim_param)
fhat <- outputsNP$estimf
fhat_param <- outputsP$estimf

 gridf <- outputsNP$gridf
 gridf1 <- gridf[1,]; gridf2 <- gridf[2,]
 marg1 <- ((max(gridf2)-min(gridf2))/length(gridf2))*apply(fhat,1,sum) #with cutoff
 marg2 <- ((max(gridf1)-min(gridf1))/length(gridf1))*apply(fhat,2,sum)
 marg1_param <- ((max(gridf2)-min(gridf2))/length(gridf2))*apply(fhat_param,1,sum)
 marg2_param <- ((max(gridf1)-min(gridf1))/length(gridf1))*apply(fhat_param,2,sum)

 plot(gridf1,marg1,type='l', col='red')
 lines(gridf1,marg1_param, lwd=2, col='red')
 plot(gridf2, marg2,type='l', col='red')
 lines(gridf2,marg2_param, lwd=2, col='red')


# Bayesian
ind <- seq(1, 2000, by = 10)
estim_Bayes <- mixedsde.fit(times[ind], X[,ind], model = "OU", random = 1,
              estim.method = "paramBayes", nMCMC = 1000)
plot(estim_Bayes)
pred_Bayes1 <- pred(estim_Bayes)
pred_Bayes2 <- pred(estim_Bayes, trajectories = TRUE)
```

`out` *Transfers the class object to a list*

Description

Method for the S4 classes

Usage

```
out(x)
```

Arguments

x	Freq.fit, Bayes.fit or Bayes.pred class

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

---

plot,Bayes.fit,ANY-method

Plot method for the Bayesian estimation class object

Description

Plot method for the S4 class Bayes.fit

Usage

```
## S4 method for signature 'Bayes.fit,ANY'
plot(x, plot.priorMean = FALSE, reduced = FALSE,
  style = c("chains", "acf", "density", "cred.int"), level = 0.05, true.phi,
  newwindow = FALSE, ...)
```

Arguments

`x`	Bayes.fit class
`plot.priorMean`	logical(1), if TRUE, prior means are added to the plots
`reduced`	logical(1), if TRUE, the chains are reduced with the burn-in and thin rate
`style`	one out of 'chains', 'acf', 'density' or 'cred.int'
`level`	alpha for the credibility intervals, only for style 'cred.int', default = 0.05
`true.phi`	only for style 'cred.int', for the case of known true values, e.g. for simulation
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

---

plot,Bayes.pred,ANY-method

Plot method for the Bayesian prediction class object

Description

Plot method for the S4 class Bayes.pred

Usage

```
## S4 method for signature 'Bayes.pred,ANY'
plot(x, newwindow = FALSE, plot.legend = TRUE,
  ylim, xlab = "times", ylab = "X", col = 3, lwd = 2, ...)
```

Arguments

`x`	Bayes.fit class
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`plot.legend`	logical(1)
`ylim`	optional
`xlab`	optional, default 'times'
`ylab`	optional, default 'X'
`col`	color for the prediction intervals, default 3
`lwd`	linewidth for the prediction intervals, default 2
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`plot,Freq.fit,ANY-method` *Plot method for the frequentist estimation class object*

Description

Plot method for the S4 class Freq.fit

Usage

```
## S4 method for signature 'Freq.fit,ANY'
plot(x, newwindow = FALSE, ...)
```

Arguments

`x`	Freq.fit class
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`plot2compare` *Comparing plot method*

Description

Method for classes

Usage

```
plot2compare(x, y, z, ...)
```

Arguments

`x`	Bayes.fit or Bayes.pred class
`y`	Bayes.fit or Bayes.pred class
`z`	Bayes.fit or Bayes.pred class (optional)
`...`	other parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`plot2compare,Bayes.fit-method`
Comparing plot method plot2compare for three Bayesian estimation class objects

---

Description

Comparison of the posterior densities for up to three S4 class Bayes.fit objects

Usage

```
## S4 method for signature 'Bayes.fit'
plot2compare(x, y, z, names, true.values,
  reduced = TRUE, newwindow = FALSE)
```

Arguments

`x`	Bayes.fit class
`y`	Bayes.fit class
`z`	Bayes.fit class (optional)
`names`	character vector of names for x, y and z
`true.values`	list of parameters to compare with the estimations, if available
`reduced`	logical(1), if TRUE, the chains are reduced with the burn-in and thin rate
`newwindow`	logical(1), if TRUE, a new window is opened for the plot

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`plot2compare,Bayes.pred-method`

Comparing plot method plot2compare for three Bayesian prediction class objects

---

Description

Comparison of the results for up to three S4 class Bayes.pred objects

Usage

```
## S4 method for signature 'Bayes.pred'
plot2compare(x, y, z, newwindow = FALSE,
  plot.legend = TRUE, names, ylim, xlab = "times", ylab = "X", ...)
```

Arguments

`x`	Bayes.pred class
`y`	Bayes.pred class
`z`	Bayes.pred class (optional)
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`plot.legend`	logical(1), if TRUE, a legend is added
`names`	character vector with names for the three objects appearing in the legend
`ylim`	optional
`xlab`	optional, default 'times'
`ylab`	optional, default 'X'
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

pred — *Prediction method*

Description

Prediction

Usage

```
pred(x, ...)
```

Arguments

`x`	Freq.fit or Bayes.fit class
`...`	other optional parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

pred,Bayes.fit-method — *Bayesian prediction method for a class object Bayes.fit*

Description

Bayesian prediction

Usage

```
## S4 method for signature 'Bayes.fit'
pred(x, invariant = FALSE, level = 0.05,
  newwindow = FALSE, plot.pred = TRUE, plot.legend = TRUE, burnIn,
  thinning, only.interval = TRUE, sample.length = 500, cand.length = 100,
  trajectories = FALSE, ylim, xlab = "times", ylab = "X", col = 3,
  lwd = 2, ...)
```

Arguments

`x`	Bayes.fit class
`invariant`	logical(1), if TRUE, the initial value is from the invariant distribution $X_t \, N(\alpha/\beta, \sigma^2/2\beta)$ for the OU and $X_t \, \Gamma(2\alpha/\sigma^2, \sigma^2/2\beta)$ for the CIR process, if FALSE (default) X0 is fixed from the data starting points
`level`	alpha for the predicion intervals, default 0.05
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`plot.pred`	logical(1), if TRUE, the results are depicted grafically
`plot.legend`	logical(1), if TRUE, a legend is added to the plot
`burnIn`	optional, if missing, the proposed value of the mixedsde.fit function is taken

`thinning`	optional, if missing, the proposed value of the mixedsde.fit function is taken
`only.interval`	logical(1), if TRUE, only prediction intervals are calculated, much faster than sampling from the whole predictive distribution
`sample.length`	number of samples to be drawn from the predictive distribution, if only.interval = FALSE
`cand.length`	number of candidates for which the predictive density is calculated, i.e. the candidates to be drawn from
`trajectories`	logical(1), if TRUE, only trajectories are drawn from the point estimations instead of sampling from the predictive distribution, similar to the frequentist approach
`ylim`	optional
`xlab`	optional, default 'times'
`ylab`	optional, default 'X'
`col`	color for the prediction intervals, default 3
`lwd`	linewidth for the prediction intervals, default 3
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`pred,Freq.fit-method` *Prediction method for the Freq.fit class object*

Description

Frequentist prediction

Usage

```
## S4 method for signature 'Freq.fit'
pred(x, invariant = 0, level = 0.05,
  newwindow = FALSE, plot.pred = TRUE, ...)
```

Arguments

`x`	Freq.fit class
`invariant`	1 if the initial value is from the invariant distribution, default X0 is fixed from Xtrue
`level`	alpha for the predicion intervals, default 0.05
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`plot.pred`	logical(1), if TRUE, the results are depicted grafically
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

print,Bayes.fit-method *Print of acceptance rates of the MH steps*

Description

Method for the S4 class Bayes.fit

Usage

```
## S4 method for signature 'Bayes.fit'
print(x)
```

Arguments

x	Bayes.fit class

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

print,Freq.fit-method *Description of print*

Description

Method for the S4 class Freq.fit

Usage

```
## S4 method for signature 'Freq.fit'
print(x)
```

Arguments

x	Freq.fit class

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

summary,Bayes.fit-method
Short summary of the results of class object Bayes.fit

Description

Method for the S4 class Bayes.fit

Usage

```
## S4 method for signature 'Bayes.fit'
summary(object, level = 0.05, burnIn, thinning)
```

Arguments

`object`	Bayes.fit class
`level`	default is 0.05
`burnIn`	optional
`thinning`	optional

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

summary,Freq.fit-method
Short summary of the results of class object Freq.fit

Description

Method for the S4 class Freq.fit

Usage

```
## S4 method for signature 'Freq.fit'
summary(object)
```

Arguments

`object`	Freq.fit class

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`UV` *Computation Of The Sufficient Statistics*

Description

Computation of U and V, the two sufficient statistics of the likelihood of the mixed SDE $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma a(X_j(t))dW_j(t)$.

Usage

```
UV(X, model, random, fixed, times)
```

Arguments

`X`	matrix of the M trajectories.
`model`	name of the SDE: 'OU' (Ornstein-Uhlenbeck) or 'CIR' (Cox-Ingersoll-Ross).
`random`	random effects in the drift: 1 if one additive random effect, 2 if one multiplicative random effect or c(1,2) if 2 random effects.
`fixed`	fixed effects in the drift: value of the fixed effect when there is only one random effect, 0 otherwise.
`times`	times vector of observation times.

Details

Computation of U and V, the two sufficient statistics of the likelihood of the mixed SDE $dX_j(t) = (\alpha_j - \beta_j X_j(t))dt + \sigma a(X_j(t))dW_j(t) = (\alpha_j, \beta_j)b(X_j(t))dt + \sigma a(X_j(t))dW_j(t)$ with $b(x) = (1, -x)^t$:

U : $U(Tend) = \int_0^{Tend} b(X(s))/a^2(X(s))dX(s)$

V : $V(Tend) = \int_0^{Tend} b(X(s))^2/a^2(X(s))ds$

Value

`U`	vector of the M statistics U(Tend)
`V`	list of the M matrices V(Tend)

References

See Bidimensional random effect estimation in mixed stochastic differential model, C. Dion and V. Genon-Catalot, *Stochastic Inference for Stochastic Processes 2015, Springer Netherlands* **1–28**

`valid` *Validation of the chosen model.*

Description

Validation of the chosen model. For the index numj, Mrep=100 new trajectories are simulated with the value of the estimated random effect number numj. Two plots are given: on the left the simulated trajectories and the true one (red) and one the left the corresponding qq-plot for each time.

Usage

```
valid(x, ...)
```

Arguments

`x`	Freq.fit or Bayes.fit class
`...`	other optional parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`valid,Bayes.fit-method`
Validation of the chosen model.

Description

Validation of the chosen model. For the index numj, Mrep=100 new trajectories are simulated with the value of the estimated random effect number numj. Two plots are given: on the left the simulated trajectories and the true one (red) and one the left the corresponding qq-plot for each time.

Usage

```
## S4 method for signature 'Bayes.fit'
valid(x, Mrep = 100, newwindow = FALSE,
  plot.valid = TRUE, numj, ...)
```

Arguments

`x`	Bayes.fit class
`Mrep`	number of trajectories to be drawn
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`plot.valid`	logical(1), if TRUE, the results are depicted grafically
`numj`	optional number of series to be validated
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

`valid,Freq.fit-method` *Validation of the chosen model.*

Description

Validation of the chosen model. For the index numj, Mrep=100 new trajectories are simulated with the value of the estimated random effect number numj. Two plots are given: on the left the simulated trajectories and the true one (red) and one the left the corresponding qq-plot for each time.

Usage

```
## S4 method for signature 'Freq.fit'
valid(x, Mrep = 100, newwindow = FALSE,
  plot.valid = TRUE, numj, ...)
```

Arguments

`x`	Freq.fit class
`Mrep`	number of trajectories to be drawn
`newwindow`	logical(1), if TRUE, a new window is opened for the plot
`plot.valid`	logical(1), if TRUE, the results are depicted grafically
`numj`	optional number of series to be validated
`...`	optional plot parameters

References

Dion, C., Hermann, S. and Samson, A. (2016). Mixedsde: an R package to fit mixed stochastic differential equations.

Index