Package 'DiffusionRimp'

August 29, 2016

Version 0.1.2

Title Inference and Analysis for Diffusion Processes via Data Imputation and Method of Lines

Description Tools for performing inference and analysis using a data-imputation scheme and the method of lines.

URL https://github.com/eta21

BugReports https://github.com/eta21/DiffusionRimp/issues

MailingList Please send questions and comments to etiennead@gmail.com.

Depends R (>= 3.2.1)

Imports Rcpp, RcppArmadillo, rgl, colorspace

LinkingTo Rcpp, RcppArmadillo

Suggests DiffusionRgqd, DiffusionRjgqd, knitr, coda, Quandl, R.rsp

License GPL (>= 2)

VignetteBuilder knitr, R.rsp

NeedsCompilation yes

Author Etienne A.D. Pienaar [aut, cre], Melvin M. Varughese [ctb]

Maintainer Etienne A.D. Pienaar <etiennead@gmail.com>

Repository CRAN

Date/Publication 2016-08-24 18:38:24

R topics documented:

`DiffusionRimp-package` *Data-imputation and density approximations for diffusion processes.*

Description

A package for performing data imputation on discretely observed diffusion processes as well as calculating numerical approximations to transition and first passage time densities.

Details

Package:	DiffusionRimp
Type:	Package
Version:	0.1.0
Date:	2015-12-01
License:	GPL (>= 2)

Functions included in the package:

`RS.impute`	:	Perform inference on a diffusion model using the random walk Metropolis-Hastings algorithm using th
`BiRS.impute`	:	Perform inference on a bivariate diffusion model using the random walk Metropolis-Hastings algorithm
`MOL.density`	:	Calculate the transitional density of a diffusion model using the method of lines.
`BiMOL.density`	:	Calculate the transitional density of a bivariate diffusion model using the method of lines.
`MOL.passage`	:	Calculate the first passage time density of a time-homogeneous diffusion model with fixed barriers (i.e.
`BiMOL.passage`	:	Calculate the first passage time density of a time-homogeneous bivariate diffusion model with fixed bar
`MOL.aic`*	:	Calculate a pseudo-AIC value for a diffusion model using the method of lines.
`BiMOL.aic`*	:	Calculate pseudo-AIC value for a bivariate diffusion model using the method of lines.

* Functions use C++.

Author(s)

Etienne A.D. Pienaar <etiennead@gmail.com>

Examples

```
# example(RS.impute)
# example(MOL.density)
# example(MOL.passage)
```

`BiMOL.aic` *Calculate a Pseudo AIC Value for a Bivariate Diffusion Model via the Method of Lines*

Description

`MOL.aic()` approximates the likelihood function for a bivariate diffusion model under a given dataset and parameter vector.

Usage

```
BiMOL.aic(X, time, delt, xlims, ylims, N, theta, diff.type, plt = TRUE,
          wrt = FALSE, border = NA)
```

Arguments

`X`	N x 2 matrix of coordinates at which the diffusion process was observed, where N is the number of observations.
`time`	A vector of time nodes at which the process was observed.
`delt`	Step size for the time domain of the lattice (see note **[4]**).
`xlims`	X-limits for the spatial nodes of the lattice. These limits should be wide enough for the transition density to be negligibly small at the endpoints (see note **[2]**).
`ylims`	Y-limits for the spatial nodes of the lattice. These limits should be wide enough for the transition density to be negligibly small at the endpoints (see note **[2]**).
`N`	The number of nodes in each spatial domain at which to evaluate the transitional density. Increase `N` for more accurate approximations (see note **[3]** and warning **[2]**).
`theta`	Parameter vector at which the AIC should be evaluated. Typically the parameter vector is calculated using `RS.impute()`
`diff.type`	2-Component vector of indicators, each assuming values 1, 2 or 3, for which of the predefined volatility structures to impose.
`plt`	Draw a plot of the calculation as it takes place.
`wrt`	Write a .cpp file. Useful for inspection purposes.
`border`	Border colour for the mesh of the perspective plot.

Value

`AIC`	An approximate AIC value.
`likelihood`	The approximate likelihood value used in the calculation.
`p`	The dimesnion of the parameter vector.

Interface

`BiMOL.aic` uses a function-input interface whereby the drift and diffusion components of the stochastic differential equation (SDE)

$$dX_t = mu_1(X_t, Y_t, t)dt + sigma_1(X_t, Y_t, t)dW_t^1$$

$$dY_t = mu_2(X_t, Y_t, t)dt + sigma_2(X_t, Y_t, t)dW_t^2,$$

are defined as R-functions in the current workspace. That is by defining the drift and diffusion components

```
mu1=function(X,Y,t){some expression i.t.o. X, Y and t}
mu2=function(X,Y,t){some expression i.t.o. X, Y and t}
sig11=function(X,Y,t){some expression i.t.o. X, Y and t}
sig22=function(X,Y,t){some expression i.t.o. X, Y and t}
```

further analysis may be conducted by calling the function `BiMOL.aic()`.

Warning

Warning [1]: Note that if the lattice is shifted, degeneracies may occur for certain drift/volatility specifications if the shifted lattice moves into non-nonsensical values of the drift/volatility functions' domains.

Warning [2]: Although increasing the spatial resolution of the lattice by increasing `N` improves approximations, instabilities will occur if `delt` is not sufficiently small. This tends to manifest as jagged/spiked solutions that oscillate between positive and negative values.

Note

Note [1]: Although the spatial limits of the lattice is defined by the user using `xlims`, if the initial value `Xs` does not fall on one of the lattice nodes, then the lattice is shifted accordingly.

Note [2]: The approximation assumes that the entire density of the process falls on a finite interval. Thus the algorithm may breakdown for certain problems. Depending on the parameters of the process, the limits may be very far apart or near. Some experimentation may be required. Otherwise, set `autofind = TRUE` to find useful limits. Note also that larger `N` may be required for wider limits.

Note [3]: Increasing `N` will likely require smaller `delt`, thus increasing computation time. For some problems, slight increases in `N` will require much smaller `delt`.

Note [4]: `delt` is used as the step size for a 10(8)-th order Runge-Kutta method for solving the resulting system of ODEs. Note again the inter-play between `N` and `delt` (see note **[3]**).

Author(s)

Etienne A.D. Pienaar <etiennead@gmail.com>

References

Hamdi, S., Schiesser, W. E. and Griffiths, G. W. 2007 Method of lines. *Scholarpedia*, **2(7)**:2859. Revision #124335.

Feagin, T. 2007 A tenth-order Runge-Kutta method with error estimate. *In Proceedings of the IAENG Conf. on Scientifc Computing*.

See Also

MOL.aic, BiRS.impute.

Examples

```
#============================================================================
```

`BiMOL.density` *Approximate the Transition Density of a Bivariate Diffusion with Arbitrary Drift and Volatility Specification*

Description

`BiMOL.density` approximates the transition density of a bivariate diffusion on a lattice `[xlims[1],xlims[2]] x [ylims[1]` with `N x N` spatial nodes and time discretization `delt`, via the method of lines. The method of lines approximates the solution of the Fokker-Planck equation by an `N x N`-dimensional system of ordinary differential equations (ODEs) evaluated on [s,t].

Usage

```
BiMOL.density(Xs, Ys, s , t , xlims, ylims, N, delt, mu1, mu2,
             sig11, sig12, sig21, sig22, final.only = FALSE,
             show.lattice = FALSE)
```

Arguments

`Xs`	Starting/Initial X-coordinate for the diffusion process (see note **[1]** and warning **[1]**).
`Ys`	Starting/Initial Y-coordinate for the diffusion process (see note **[1]** and warning **[1]**).
`s`	Starting time for the diffusion process.
`t`	Value (>s) giving the terminal point for the transition horizon (the final time at which to evaluate the transition density).
`xlims`	X-limits for the spatial nodes of the lattice. These limits should be wide enough for the transition density to be negligibly small at the endpoints (see note **[2]**).
`ylims`	Y-limits for the spatial nodes of the lattice. These limits should be wide enough for the transition density to be negligibly small at the endpoints (see note **[2]**).

N	The number of nodes in each spatial domain at which to evaluate the transitional density. Increase N for more accurate approximations (see note **[3]** and warning **[2]**).
mu1	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
mu2	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
sig11	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
sig12	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
sig21	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
sig22	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
delt	Step size for the time domain of the lattice (see note **[4]**).
final.only	Should the transition density on the entire lattice be returned (FALSE) or only the at the terminal point of the transition horizon t (TRUE). Default = FALSE.
show.lattice	If =TRUE (default) then the X-Y lattice is drawn and the initial value is indicated in red.

Value

density	3D array containing the density approximation (matrix if final.only = TRUE).
Xt	Vector of X-coordinates at which approximation was carried out.
Yt	Vector of Y-coordinates at which approximation was carried out.
time	Vector of time nodes at which the approximation was evaluated.

Interface

BiMOL.density uses a function-input interface whereby the drift and diffusion components of the stochastic differential equation (SDE)

$$dX_t = mu_1(X_t, Y_t, t)dt + sigma_1(X_t, Y_t, t)dW_t^1$$

$$dY_t = mu_2(X_t, Y_t, t)dt + sigma_2(X_t, Y_t, t)dW_t^2,$$

are defined as R-functions in the current workspace. That is by defining the drift and diffusion components

```
mu1=function(X,Y,t){some expression i.t.o. X, Y and t}
mu2=function(X,Y,t){some expression i.t.o. X, Y and t}
sig11=function(X,Y,t){some expression i.t.o. X, Y and t}
sig22=function(X,Y,t){some expression i.t.o. X, Y and t}
```

further analysis may be conducted by calling the function BiMOL.density().

Warning

Warning [1]: Note that if the lattice is shifted, degeneracies may occur for certain drift/volatility specifications if the shifted lattice moves into non-nonsensical values of the drift/volatility functions' domains.

Warning [2]: Although increasing the spatial resolution of the lattice by increasing `N` improves approximations, instabilities will occur if `delt` is not sufficiently small. This tends to manifest as jagged/spiked solutions that oscillate between positive and negative values.

Note

Note [1]: Although the spatial limits of the lattice is defined by the user using `xlims`, if the initial value `Xs` does not fall on one of the lattice nodes, then the lattice is shifted accordingly.

Note [2]: The approximation assumes that the entire density of the process falls on a finite interval. Thus the algorithm may breakdown for certain problems. Depending on the parameters of the process, the limits may be very far apart or near. Some experimentation may be required. Otherwise, set `autofind = TRUE` to find useful limits. Note also that larger `N` may be required for wider limits.

Note [3]: Increasing `N` will likely require smaller `delt`, thus increasing computation time. For some problems, slight increases in `N` will require much smaller `delt`.

Note [4]: `delt` is used as the step size for a 10(8)-th order Runge-Kutta method for solving the resulting system of ODEs. Note again the inter-play between `N` and `delt` (see note **[3]**).

Author(s)

Etienne A.D. Pienaar <etiennead@gmail.com>

References

Hamdi, S., Schiesser, W. E. and Griffiths, G. W. 2007 Method of lines. *Scholarpedia*, **2(7)**:2859. Revision #124335.

Feagin, T. 2007 A tenth-order Runge-Kutta method with error estimate. *In Proceedings of the IAENG Conf. on Scientifc Computing*.

See Also

`MOL.passage`, `MOL.density`.

Examples

```
#============================================================================

 # For detailed notes and examples on how to use the BiMOL.density() function, see
 # the following vignette:

 RShowDoc('Part_2_Transition_Densities',type='html','DiffusionRimp')

#============================================================================
```

`BiMOL.passage` *Approximate the First Passage Time Density of a Four-Barrier Problem for Time-Homogeneous Bivariate Diffusions.*

Description

`BiMOL.passage()` approximates a solution to partial differential equation (PDE) that governs the evolution of the survaval distribution of the first passage time density of a bivariate diffusion passing through fixed thresholds `limits[1]` or `limits[2]` in the X-dimension and `limits[3]` or `limits[4]` in the Y-dimension.

Usage

```
BiMOL.passage(Xs, Ys, t, limits, N, delt, mu1, mu2, sig11, sig12,
              sig21, sig22, desc = 1, Phi, plt = FALSE)
```

Arguments

`Xs`	Starting/Initial X-coordinate for the diffusion process (see note **[1]**).
`Ys`	Starting/Initial Y-coordinate for the diffusion process (see note **[1]**).
`t`	Value (>0) giving the terminal point for the transition horizon (the final time at which to evaluate the transition density).
`limits`	Limits for the spatial nodes of the lattice. These limits now represent the limits in the spatial domain (see note **[2]**).
`N`	The number of nodes in the spatial domain at which to evaluate the transitional density. Increase `N` for more accurate approximations (see note **[3]** and warning **[2]**).
`delt`	Step size for the time domain of the lattice (see note **[4]**).
`mu1`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`mu2`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`sig11`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`sig12`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`sig21`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`sig22`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`desc`	The type of discretization used (see note **[5]**).
`Phi`	An optional indicator function for defining non-trivial boundary shapes (See vignettes).
`plt`	Should a plot be made (for dev purposes).

Value

surface	An array giving the approximate survival probability volume for all starting values contained in the discretization of the polygon enclosed by the limits.
density	A vector containing the approximate first passage time density for trajectories starting at (Xs,Ys) (see note **[i]**).
time	A vector of time nodes at which the approximation was evaluated.

Interface

BiMOL.passage uses a function-input interface whereby the drift and diffusion components of the time-homogeneous bivariate stochastic differential equation (SDE):

$$dX_t = mu_1(X_t, Y_t)dt + sigma_1(X_t, Y_t)dW_t^1$$

$$dY_t = mu_2(X_t, Y_t)dt + sigma_2(X_t, Y_t)dW_t^2,$$

are defined as R-functions in the current workspace. That is by defining the drift and diffusion components

```
mu1=function(X,Y){some expression i.t.o. X and Y}
mu2=function(X,Y){some expression i.t.o. X and Y}
sig11=function(X,Y){some expression i.t.o. X and Y}
sig22=function(X,Y){some expression i.t.o. X and Y}
```

further analysis may be conducted by calling the function BiMOL.passage().

Warnings

Warning [1]:

Note

Note [1]: If the initial value Xs does not fall on one of the lattice nodes, then the first passage time density is calculated by linearly interpolating between approximations at the two nearest lattice nodes.

Note [2]: Note that that enough nodes, N, are needed in order to generate a sufficiently accurate approximation, especially when limits[1] and limits[2] are far apart.

Note [3]: Increasing N will likely require smaller delt, thus increasing computation time. For some problems, slight increases in N will require much smaller delt.

Note [4]: delt is used as the step size for a 10(8)-th order Runge-Kutta method for solving the resulting system of ODEs. Note again the inter-play between N and delt (see note **[3]**).

Note [5]: When one of the limits is sufficiently far away to not be accessible within the provided time-horizon, instabilities may occur and an alternative discretization may be required in order to ensure smooth operation. Possible values are desc = 1 (close limits), desc = 2 (limits[1] is inaccessible) and desc = 3 (limits[2] is inaccessible).

Author(s)

Etienne A.D. Pienaar <etiennead@gmail.com>

References

Hamdi, S., Schiesser, W. E. and Griffiths, G. W. 2007 Method of lines. *Scholarpedia*, **2(7)**:2859. Revision #124335.

Feagin, T. 2007 A tenth-order Runge-Kutta method with error estimate. *In Proceedings of the IAENG Conf. on Scientifc Computing*.

See Also

MOL.passage, BiMOL.density.

Examples

```
#============================================================================

 # For detailed notes and examples on how to use the BiMOL.passage() function, see
 # the following vignette:

 RShowDoc('Part_3_First_Passage_Times',type='html','DiffusionRimp')

#============================================================================
```

BiRS.impute *Brownian Bridge Data Imputation for Bivariate Diffusion Processes.*

Description

BiRS.impute performs inference on bivariate diffusion processes with quite arbitrary drift functionals by imputing missing sample paths with Brownian bridges. The procedure was developed by Roberts and Stramer (2001) and subsequently extended in later papers (). Currently, the diffusion is assumed to take on the form

$$dX_t = mux(X_t, Y_t, t, theta)dt + sigma[1]f(X_t)dW_t^1$$

$$dY_t = muy(X_t, Y_t, t, theta)dt + sigma[2]g(Y_t)dW_t^2,$$

where $mux(X_t, Y_t, t, theta)$ and $muy(X_t, Y_t, t, theta)$ may be defined with near-complete freedom as R-functions in the current workspace. $f(.)$ and $g(.)$ may take on predefined forms (see details).

Usage

```
BiRS.impute(X, time, M, theta, sds, diff.type = c(1, 1),
            burns = min(floor(updates/2),25000), updates, plot.chain = TRUE,
            imputation.plot = FALSE, palette = 'mono')
```

Arguments

X	N x 2 matrix of coordinates at which the diffusion process was observed, where N is the number of observations.
time	A vector of time nodes at which the process was observed.
M	The number of points to impute between successive observations. Note that as the transition horizon increases, more points may be required in order to get desirable acceptance rates for Brownian bridges. Note that some transition horizons may be removed from likelihood calculations - see exclude.
theta	Starting parameters for the model process.
sds	Proposal standard deviations for the drift parameter chain.
diff.type	2-Component vector of indicators, each assuming values 1, 2 or 3, for which of the predefined volatility structures to impose - see note **[3]**.
burns	The number of updates to burn (only affects traceplots).
updates	The number of updates to perform.
imputation.plot	Display imputed paths for successive updates - see note **[4]**. Default = FALSE.
plot.chain	Display the resulting MCMC chain - see notes **[5]**, **[6]** and **[7]**.
palette	Colour palette for drawing trace plots. Default palette = 'mono', otherwise a qualitative palette will be used.

Details

DETAILS ABOUT THE METHODOLOGY

Value

per.matrix	Matrix containing the MCMC chain updates.
acceptence.rate	Acceptance rates for the two parameter chains.
bridge.rate	Average Acceptance rate for Brownian bridge updates.
run.time	The total run time of the algorithm.

Note

Note [1]: The functional form of the drift components may be defined by the user as R-functions. Although the function body may take on arbitrary forms both the name of the drift functions and the input structure must assume the form

mux = function(variable1, variable2, time, theta){...} and

`muy = function(variable1, variable2, time, theta){...}`.

Note that `theta` is a reserved variable for the parameters. It is left to the user to ensure that the functional forms are not nonsensical. See the examples below.

Note [2]: Both drift functions have to be defined in order for the algorithm to execute.

Note [3]: The functional form for the volatility of each dimension may take on one of the following: $f(X_t) = 1$, corresponding to constant volatility , $f(X_t) = \sqrt{X_t}$ corresponding to CIR type models and $f(X_t) = X_t$ such as for geometric Brownian motion. Corresponding indicators are simply given buy 1, 2, and 3.

Note [4]: Paths are imputed on a unit -volatility process. For display purposes the back-transformed imputed trajectories may plotted along with vertical lines indicating which bridges remained unchanged over successive updates.

Note [5]: By default the MCMC chain is displayed in a panel plot. Standard MCMC diagnostics may be performed on this chain which is returned in the value list as `per.matrix`.

Note [6]: In addition to the MCMC chain, acceptance rates are given for both the drift vector parameter updates as well as the volatility parameter updates. A, target region for the acceptance rate trajectories is displayed in blue.

Note [7]: Average acceptance rates are indicated for bridges per the transition number by blue bars. A target region for the bridge acceptance rates is given in light blue. Average acceptance rates that are lower than 60% are indicated along with their respective transition number. Vertical grey lines indicate `exclude`-d transitions.

Author(s)

Etienne A.D. Pienaar `<etiennead@gmail.com>`

References

Dellaportas, P. 2006 Bayesian model selection for partially observed diffusion models. *Biometrika*, **93(4)**: 809.

Kalogeropoulos, K., Dellaportas, P., and Roberts, G. O. 2011 Likelihood based inference for correlated diffusions. *Canadian Journal of Statistics*, **39(1)**:52–72.

Roberts, G. and Stramer, O. 2001 On inference for partially observed nonlinear diffusion models using the Metropolis-Hastings algorithm. *Biometrika*, **88**:603–621.

See Also

`RS.impute`, `BiMOL.density`.

Examples

```
#============================================================================
```

DoubleWell *Simulted Double Well with Shifting Modality*

Description

A simulated diffusion of the form

$$dX_t = X_t(1 + sin(2 * pi * t) - X^2)dt + dW_t,$$

with $X_0 = 1$.

Usage

```
data(DoubleWell)
```

Format

A data frame with 201 observations on the following 2 variables.

`X_t` Vector containing the simulated trajectory.

`t` Time points at which the diffusion was observed.

Examples

```
data(DoubleWell)
plot(rev(DoubleWell),type='l')
```

MOL.aic *Calculate a Pseudo AIC Value for a Diffusion Model via the Method of Lines*

Description

`MOL.aic()` approximates the likelihood function for a diffusion model under a given dataset and parameter vector.

Usage

```
MOL.aic(X, time, delt, xlims, N, theta, diff.type,plt = TRUE, wrt = FALSE)
```

Arguments

X	Vector of coordinates at which the diffusion process was observed, where N is the number of observations.
time	A vector of time nodes at which the process was observed.
xlims	Limits for the spatial nodes of the lattice. These limits should be wide enough for the transition density to be negligibly small at the endpoints (see note **[2]**).
N	The number of nodes in the spatial domain at which to evaluate the transitional density. Increase N for more accurate approximations (see note **[3]** and warning **[2]**).
delt	Step size for the time domain of the lattice (see note **[4]**).
theta	Parameter vector at which the AIC should be evaluated. Typically the parameter vector is calculated using `RS.impute()`
diff.type	An indicator assuming values 1, 2 or 3, for which of the predefined volatility structures to impose.
plt	Draw a plot of the calculation as it takes place.
wrt	Write a .cpp file. Useful for inspection purposes.

Value

AIC	An approximate AIC value.
likelihood	The approximate likelihood value used in the calculation.
p	The dimesnion of the parameter vector.

Interface

`MOL.aic` uses a function-input interface whereby the drift and diffusion components of the stochastic differential equation (SDE)

$$dX_t = mu(X_t, t)dt + sigma(X_t, t)dW_t,$$

are defined as R-functions in the current workspace. That is by defining the drift and diffusion components

```
mu=function(X,t){some expression i.t.o. X and t}
```

```
sig=function(X,t){some expression i.t.o. X and t}
```

further analysis may be conducted by calling the function `MOL.aic()`.

Warning

Warning [1]: Note that if the lattice is shifted, degeneracies may occur for certain drift/volatility specifications if the shifted lattice moves into non-nonsensical values of the drift/volatility functions' domains.

Warning [2]: Although increasing the spatial resolution of the lattice by increasing N improves approximations, instabilities will occur if `delt` is not sufficiently small. This tends to manifest as jagged/spiked solutions that oscillate between positive and negative values.

Note

Note [1]: Although the spatial limits of the lattice is defined by the user using `xlims`, if the initial value `Xs` does not fall on one of the lattice nodes, then the lattice is shifted accordingly.

Note [2]: The approximation assumes that the entire density of the process falls on a finite interval. Thus the algorithm may breakdown for certain problems. Depending on the parameters of the process, the limits may be very far apart or near. Some experimentation may be required. Otherwise, set `autofind = TRUE` to find useful limits. Note also that larger `N` may be required for wider limits.

Note [3]: Increasing `N` will likely require smaller `delt`, thus increasing computation time. For some problems, slight increases in `N` will require much smaller `delt`.

Note [4]: `delt` is used as the step size for a 10(8)-th order Runge-Kutta method for solving the resulting system of ODEs. Note again the inter-play between `N` and `delt` (see note **[3]**).

Author(s)

Etienne A.D. Pienaar <etiennead@gmail.com>

See Also

`RS.impute`, `BiMOL.aic`.

Examples

```
#============================================================================
```

`MOL.density` *Approximate the Transition Density of a Scalar Diffusion with Arbitrary Drift and Volatility Specification*

Description

For scalar diffusions with drift `mu=function(X,t){}` and diffusion `sig=function(X,t){}`, `MOL.density` approximates the transition density of a scalar diffusion on a lattice `[xlims[1],xlims[2]] x [s,t]` with `N` spatial nodes and time discretization `delt`, via the method of lines. The method of lines approximates the solution of the Fokker-Planck equation by an `N`-dimensional system of ordinary differential equations (ODEs) evaluated on [s,t].

Usage

```
MOL.density(Xs, s, t, xlims, N = 31, delt, mu, sig, final.only = FALSE)
```

Arguments

Xs	Starting/Initial value for the diffusion process (see note **[1]** and warning **[1]**).
s	Starting time for the diffusion process.
t	Value (>s) giving the terminal point for the transition horizon (the final time at which to evaluate the transition density).
xlims	Limits for the spatial nodes of the lattice. These limits should be wide enough for the transition density to be negligibly small at the endpoints (see note **[2]**).
N	The number of nodes in the spatial domain at which to evaluate the transitional density. Increase N for more accurate approximations (see note **[3]** and warning **[2]**).
delt	Step size for the time domain of the lattice (see note **[4]**).
mu	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
sig	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
final.only	Should the transition density on the entire lattice be returned (FALSE) or only the at the terminal point of the transition horizon t (TRUE). Default = FALSE.

Value

density	Matrix containing the density approximation (vector if final.only = TRUE).
Xt	Vector of spatial values at which approximation was carried out.
time	Vector of time nodes at which the approximation was evaluated.

Interface

MOL.density uses a function-input interface whereby the drift and diffusion components of the stochastic differential equation (SDE)

$$dX_t = mu(X_t, t)dt + sigma(X_t, t)dW_t,$$

are defined as R-functions in the current workspace. That is by defining the drift and diffusion components

```
mu=function(X,t){some expression i.t.o. X and t}
```

```
sig=function(X,t){some expression i.t.o. X and t}
```

further analysis may be conducted by calling the function MOL.density().

Warning

Warning [1]: Note that if the lattice is shifted, degeneracies may occur for certain drift/volatility specifications if the shifted lattice moves into non-nonsensical values of the drift/volatility functions' domains.

Warning [2]: Although increasing the spatial resolution of the lattice by increasing N improves approximations, instabilities will occur if delt is not sufficiently small. This tends to manifest as jagged/spiked solutions that oscillate between positive and negative values.

Note

Note [1]: Although the spatial limits of the lattice is defined by the user using `xlims`, if the initial value `Xs` does not fall on one of the lattice nodes, then the lattice is shifted accordingly.

Note [2]: The approximation assumes that the entire density of the process falls on a finite interval. Thus the algorithm may breakdown for certain problems. Depending on the parameters of the process, the limits may be very far apart or near. Some experimentation may be required. Otherwise, set `autofind = TRUE` to find useful limits. Note also that larger `N` may be required for wider limits.

Note [3]: Increasing `N` will likely require smaller `delt`, thus increasing computation time. For some problems, slight increases in `N` will require much smaller `delt`.

Note [4]: `delt` is used as the step size for a 10(8)-th order Runge-Kutta method for solving the resulting system of ODEs. Note again the inter-play between `N` and `delt` (see note **[3]**).

Author(s)

Etienne A.D. Pienaar <etiennead@gmail.com>

References

Hamdi, S., Schiesser, W. E. and Griffiths, G. W. 2007 Method of lines. *Scholarpedia*, **2(7)**:2859. Revision #124335.

Feagin, T. 2007 A tenth-order Runge-Kutta method with error estimate. *In Proceedings of the IAENG Conf. on Scientifc Computing*.

See Also

`MOL.passage`, `BiMOL.density`.

Examples

```
#=============================================================================

 # For detailed notes and examples on how to use the MOL.density() function, see
 # the following vignette:

 RShowDoc('Part_2_Transition_Densities',type='html','DiffusionRimp')

#=============================================================================
```

`MOL.passage` *Approximate the First Passage Time Density of a Two-Barrier Problem for Time-Homogeneous Scalar Diffusions.*

Description

For scalar diffusions with drift `mu=function(X){}` and diffusion `sig=function(X){}`, moving in relation to lower and upper bounds `limits[1]` and `limits[2]` respectively, `MOL.passage()` approximates a solution to the partial differential equation (PDE) that governs the evolution of the survaval distribution of the first passage time density via the method of lines (MOL).

Usage

```
MOL.passage(Xs, t, limits, N, delt, mu, sig, desc = 1)
```

Arguments

`Xs`	Starting/Initial value for the diffusion process (see note **[1]**).
`t`	Value (>0) giving the terminal point for the transition horizon (the final time at which to evaluate the transition density).
`limits`	Limits for the spatial nodes of the lattice. These limits now represent the limits in the spatial domain (see note **[2]**).
`N`	The number of nodes in the spatial domain at which to evaluate the transitional density. Increase `N` for more accurate approximations (see note **[3]** and warning **[2]**).
`delt`	Step size for the time domain of the lattice (see note **[4]**).
`mu`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`sig`	Optional (if drift and diffusion coefficients are already defined) arguments giving the drift and diffusion coefficients as text expressions (See **Interface** below).
`desc`	The type of discretization used (see note **[5]**).

Value

`surface`	A matrix giving the approximate survival probability over time for all starting values contained in the discretization of the interval enclosed by `limits`.
`density`	A vector containing the approximate first passage time density for trajectories starting at `Xs` (see note **[i]**).
`time`	A vector of time nodes at which the approximation was evaluated.

Interface

MOL.passage uses a function-input interface whereby the drift and diffusion components of the time-homogeneous stochastic differential equation (SDE):

$$dX_t = mu(X_t)dt + sigma(X_t)dW_t,$$

are defined as R-functions in the current workspace. That is by defining the drift and diffusion components

```
mu=function(X){some expression i.t.o. X}
```

```
sig=function(X){some expression i.t.o. X}
```

further analysis may be conducted by calling the function MOL.passage().

Warnings

Warning [1]:

Note

Note [1]: If the initial value Xs does not fall on one of the lattice nodes, then the first passage time density is calculated by linearly interpolating between approximations at the two nearest lattice nodes.

Note [2]: Note that that enough nodes, N, are needed in order to generate a sufficiently accurate approximation, especially when limits[1] and limits[2] are far apart.

Note [3]: Increasing N will likely require smaller delt, thus increasing computation time. For some problems, slight increases in N will require much smaller delt.

Note [4]: delt is used as the step size for a 10(8)-th order Runge-Kutta method for solving the resulting system of ODEs. Note again the inter-play between N and delt (see note **[3]**).

Note [5]: When one of the limits is sufficiently far away to not be accessible within the provided time-horizon, instabilities may occur and an alternative discretization may be required in order to ensure smooth operation. Possible values are desc = 1 (close limits), desc = 2 (limits[1] is inaccessible) and desc = 3 (limits[2] is inaccessible).

Author(s)

Etienne A.D. Pienaar <etiennead@gmail.com>

References

Hamdi, S., Schiesser, W. E. and Griffiths, G. W. 2007 Method of lines. *Scholarpedia*, **2(7)**:2859. Revision #124335.

Feagin, T. 2007 A tenth-order Runge-Kutta method with error estimate. *In Proceedings of the IAENG Conf. on Scientifc Computing*.

See Also

MOL.density, BiMOL.density.

Examples

```
#============================================================================

 # For detailed notes and examples on how to use the MOL.passage() function, see
 # the following vignette:

 RShowDoc('Part_3_First_Passage_Times',type='html','DiffusionRimp')

#============================================================================
```

`MOL.plot` *Quick Plots for DiffusionRimp Objects*

Description

`MOL.plot()` recognizes output objects calculated using routines from the **DiffusionRimp** package and subsequently constructs an appropriate plot, for example a perspective plot of a transition density.

Usage

```
MOL.plot(x)
```

Arguments

`x`	Generic MOL-objects, i.e. `res = MOL.density()`.

Value

Varies in accordance with input type.

Author(s)

Etienne A.D. Pienaar: <etiannead@gmail.com>

References

Updates available on GitHub at https://github.com/eta21.

See Also

`MOL.density`, `BiMOL.density`, `MOL.passage` etc.

Examples

```
#============================================================================
```

RcppArmadillo-Functions *A Junk Funktion For Build Purposes*

Description

This function was created as a filler in order for the package to build correctly.

Usage

```
junkfunction3()
```

Details

This function was created as a filler in order for the package to build correctly.

Value

`junkfunction3()` does nothing useful.

Author(s)

Etienne A.D. Pienaar

References

See the documentation for Armadillo, and RcppArmadillo, for more details.

RS.estimates *Extract Parmaeter Estimates from* `.impute()` *Objects.*

Description

`RS.estimates()` calculates parameter estimates from `.impute()` model objects.

Usage

```
RS.estimates(x, thin = 100, burns, CI = c(0.05, 0.95), corrmat = FALSE,
             acf.plot = TRUE, palette = 'mono')
```

Arguments

x	List object of type 'RS.impute' or 'BiRS.impute'. That is, when `model = RS.impute()` then `model` constitutes an appropriate object for `x`.
thin	Thinnging level for parameter chain.
burns	Number of MCMC updates to discard before calculating estimates.
CI	Credibility interval quantiles (for MCMC chains).
corrmat	If TRUE, an estimated correlation matrix is returned in addition to the estimate vector.
acf.plot	If TRUE, an acf plot is drawn for each element of the parameter chain.
palette	Colour palette for drawing trace plots. Default `palette = 'mono'`, otherwise a qualitative palette will be used.

Value

Data frame with parameter estimates and appropriate interval statistics.

Author(s)

Etienne A.D. Pienaar: <etiannead@gmail.com>

References

Updates available on GitHub at https://github.com/eta21.

See Also

RS.impute, BiRS.impute.

Examples

```
example(RS.impute)
```

RS.impute *Brownian Bridge Data Imputation for Scalar Diffusion Processes.*

Description

RS.impute performs inference on bivariate diffusion processes with quite arbitrary drift functionals by imputing missing sample paths with Brownian bridges. The procedure was developed by Roberts and Stramer (2001) and subsequently extended in later papers (Dellaportas et al, 2006; Kalogeropoulos et al., 2011). Currently, the diffusion is assumed to take on the form

$$dX_t = mu(X_t, t, theta)dt + sigma[1]f(X_t)dW_t,$$

where $mu(X_t, theta)$ may be defined with near-complete freedom as R-functions in the current workspace. $f(.)$ may take on predefined forms (see details).

Usage

```
RS.impute(X, time, M, theta, sds, diff.type = 1, burns = min(floor(updates/2),25000),
          updates, plot.chain = TRUE, imputation.plot = FALSE, palette = 'mono')
```

Arguments

X	Vector of coordinates at which the diffusion process was observed, where N is the number of observations.
time	A vector of time nodes at which the process was observed.
M	The number of points to impute between successive observations. Note that as the transition horizon increases, more points may be required in order to get desirable acceptance rates for Brownian bridges. Note that some transition horizons may be removed from likelihood calculations - see exclude.
theta	Starting parameters for the model process.
sds	Proposal standard deviations for the drift parameter chain.
diff.type	An indicator assuming values 1, 2 or 3, for which of the predefined volatility structures to impose - see note **[3]**.
burns	The number of updates to burn (only affects traceplots).
updates	The number of updates to perform.
imputation.plot	Display imputed paths for successive updates - see note **[4]**. Default = FALSE.
plot.chain	Display the resulting MCMC chain - see notes **[5]**, **[6]** and **[7]**.
palette	Colour palette for drawing trace plots. Default palette = 'mono', otherwise a qualitative palette will be used.

Value

per.matrix	Matrix containing the MCMC chain updates.
acceptence.rate	Vector of acceptance rates for the two parameter chains.
bridge.rate	Vector of average acceptance rates for Brownian bridge updates.
run.time	The total run time of the algorithm.

Note

Note [1]: The functional form of the drift components may be defined by the user as R-functions. Although the function body may take on arbitrary forms both the name of the drift functions and the input structure must assume the form

`mu = function(variable, time, theta){...}`.

Note that `theta` is a reserved variable for the parameters. It is left to the user to ensure that the functional forms do not degenerate. See the examples below.

Note [2]: The drift function has to be defined in order for the algorithm to execute.

Note [3]: The functional form for the volatility may take on one of the following: $f(X_t) = 1$, corresponding to constant volatility , $f(X_t) = \sqrt{X_t}$ corresponding to CIR type models and $f(X_t) = X_t$ such as for geometric Brownian motion. Corresponding indicators are simply given buy 1, 2, and 3.

Note [4]: Paths are imputed on a unit -volatility process. For display purposes the back-transformed imputed trajectories may plotted along with vertical lines indicating which bridges remained unchanged over successive updates.

Note [5]: By default the MCMC chain is displayed in a panel plot. Standard MCMC diagnostics may be performed on this chain which is returned in the value list as `per.matrix`.

Note [6]: In addition to the MCMC chain, acceptance rates are given for both the drift vector parameter updates as well as the volatility parameter updates. A, target region for the acceptance rate trajectories is displayed in blue.

Note [7]: Average acceptance rates are indicated for bridges per the transition number by blue bars. A target region for the bridge acceptance rates is given in light blue. Average acceptance rates that are lower than 60% are indicated along with their respective transition number. Vertical grey lines indicate `exclude`-d transitions.

Author(s)

Etienne A.D. Pienaar `<etiennead@gmail.com>`

References

Dellaportas, P. 2006 Bayesian model selection for partially observed diffusion models. *Biometrika*, **93(4)**: 809.

Kalogeropoulos, K., Dellaportas, P., and Roberts, G. O. 2011 Likelihood based inference for correlated diffusions. *Canadian Journal of Statistics*, **39(1)**:52–72.

Roberts, G. and Stramer, O. 2001 On inference for partially observed nonlinear diffusion models using the Metropolis-Hastings algorithm. *Biometrika*, **88**:603–621.

See Also

`BiRS.impute`, `MOL.density`.

Examples

```
#============================================================================
```

TiCIR *Simulated Bivariate Time In-Homogeneous CIR process.*

Description

A simulated diffusion of the form

$$dX_t = X_t(1 + sin(2 * pi * t) - X_t^2)dt + dW_t^1,$$

$$dY_t = Y_t(1 + sin(2 * pi * t) - Y_t^2)dt + dW_t^2,$$

with $X_0 = 1$ and $Y_0 = 1$.

Usage

```
data(TiCIR)
```

Format

A data frame with 101 observations on the following 3 variables.

`X_t` A numeric vector of X-coordinates for the simulated data.

`Y_t` A numeric vector of X-coordinates for the simulated data.

`t` A numeric vector of time nodes at which the simulated data were observed.

Examples

```
data(TiCIR)
```

Index