Statistical data analysis of financial time series and option pricing in R

S.M. Iacus (University of Milan)

Chicago, R/Finance 2011, April 29th
Explorative Data Analysis

NYSE data

Estimation of Financial Models

Likelihood approach

Two stage least squares estimation

Model selection

Numerical Evidence

Application to real data

The change point problem

Overview of the yuima package

Option Pricing with R

Explorative Data Analysis
Cluster analysis is concerned with discovering group structure among $n$ observations, for example time series.

Such methods are based on the notion of *dissimilarity*. A dissimilarity measure $d$ is a symmetric: $d(A, B) = d(B, A)$, non-negative, and such that $d(A, A) = 0$.

Dissimilarities can, but not necessarily be, a *metric*, i.e. $d(A, C) \leq d(A, B) + d(B, C)$. 
A clustering method is an algorithm that works on a $n \times n$ dissimilarity matrix by aggregating (or, viceversa, splitting) individuals into groups.

**Agglomerative** algorithms start from $n$ individuals and, at each step, aggregate two of them or move an individual in a group obtained at some previous step. Such an algorithm stops when only one group with all the $n$ individuals is formed.

**Divisive** methods start from a unique group of all individuals by splitting it, at each step, in subgroups till the case when $n$ singletons are formed.

These methods are mainly intended for exploratory data analysis and each method is influenced by the dissimilarity measure $d$ used.

R offers a variety to clustering algorithm and distances to play with but, up to date, not so much towards clustering of time series.
Next: time series of daily closing quotes, from 2006-01-03 to 2007-12-31, for the following 20 financial assets:

Microsoft Corporation (MSOFT in the plots)  Advanced Micro Devices Inc. (AMD)
Dell Inc. (DELL)  Intel Corporation (INTEL)
Hewlett-Packard Co. (HP)  Sony Corp. (SONY)
Motorola Inc. (MOTO)  Nokia Corp. (NOKIA)
Electronic Arts Inc. (EA)  LG Display Co., Ltd. (LG)
Borland Software Corp. (BORL)  Koninklijke Philips Electronics NV (PHILIPS)
Symantec Corporation (SYMATEC)  JPMorgan Chase & Co (JMP)
Merrill Lynch & Co., Inc. (MLINCH)  Deutsche Bank AG (DB)
Citigroup Inc. (CITI)  Bank of America Corporation (BAC)
Goldman Sachs Group Inc. (GSACHS)  Exxon Mobil Corp. (EXXON)

Quotes come from NYSE/NASDAQ. Source Yahoo.com.
Real Data from NYSE. How to cluster them?

quotes

Index

2006 2007 2008

Index

2006 2007 2008
How to measure the distance between two time series?

The are several ways to measure the distance between two time series:

- the Euclidean distance $d_{EUC}$
- the Short-Time-Series distance $d_{STS}$
- the Dynamic Time Warping distance $d_{DTW}$
- the Markov Operator distance $d_{MO}$
Let $X$ and $Y$ be two time series, then the usual Euclidean distance

$$d_{EUC}(X, Y) = \sqrt{\sum_{i=1}^{N} (X_i - Y_i)^2}$$

is one of the most used in the applied literature. We use it only for comparison purposes.
Let $X$ and $Y$ be two time series, then the usual Euclidean distance

$$d_{STS}(X, Y) = \sqrt{\sum_{i=1}^{N} \left( \frac{X_i - X_{i-1}}{\Delta} - \frac{Y_i - Y_{i-1}}{\Delta} \right)^2}.$$
Dynamic Time Warping distance $d_{DTW}$

Let $X$ and $Y$ be two time series, then

$$d_{DTW}(X, Y) = \sqrt{\sum_{i=1}^{N} \left( \frac{X_i - X_{i-1}}{\Delta} - \frac{Y_i - Y_{i-1}}{\Delta} \right)^2}.$$

DTW allows for non-linear alignments between time series not necessarily of the same length.

Essentially, all shiftings between two time series are attempted and each time a cost function is applied (e.g. a weighted Euclidean distance between the shifted series). The minimum of the cost function over all possible shiftings is the dynamic time warping distance $d_{DTW}$. 
Consider
\[ \text{d}X_t = b(X_t)\text{d}t + \sigma(X_t)\text{d}W_t \]

therefore, the discretized observations \( X_i \) form a Markov process and all the mathematical properties are embodied in the so-called \textit{transition operator}

\[ P_\Delta f(x) = E\{f(X_i)|X_{i-1} = x\} \]

with \( f \) is a generic function, e.g. \( f(x) = x^k \).

Notice that \( P_\Delta \) depends on the transition density between \( X_i \) and \( X_{i-1} \), so we put explicitly the dependence on \( \Delta \) in the notation.

Luckily, there is no need to deal with the transition density, we can estimate \( P_\Delta \) directly and easily.
Let $X$ and $Y$ be two time series, then

$$d_{MO}(X, Y) = \sum_{j,k \in J} \left| (\hat{P}_\Delta)_{j,k}(X) - (\hat{P}_\Delta)_{j,k}(Y) \right|, \quad (1)$$

where $(\hat{P}_\Delta)_{j,k}(\cdot)$ is a matrix and it is calculated as in (2) separately for $X$ and $Y$.

$$(\hat{P}_\Delta)_{j,k}(X) = \frac{1}{2N} \sum_{i=1}^{N} \{ \phi_j(X_{i-1})\phi_k(X_i) + \phi_k(X_{i-1})\phi_j(X_i) \} \quad (2)$$

The terms $(\hat{P}_\Delta)_{j,k}$ are approximations of the scalar product $< P_\Delta \phi_j, \phi_k >_{b,\sigma}$ where $b$ and $\sigma$ are the unknown drift and diffusion coefficient of the model. Thus, $(\hat{P}_\Delta)_{j,k}$ contains all the information of the Markovian structure of $X$. 
We simulate 10 paths $X_i$, $i = 1, \ldots, 10$, according to the combinations of drift $b_i$ and diffusion coefficients $\sigma_i$, $i = 1, \ldots, 4$ presented in the following table

<table>
<thead>
<tr>
<th>$b_i(x)$</th>
<th>$\sigma_1(x)$</th>
<th>$\sigma_2(x)$</th>
<th>$\sigma_3(x)$</th>
<th>$\sigma_4(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1(x)$</td>
<td>X10, X1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_2(x)$</td>
<td></td>
<td>X2, X3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_3(x)$</td>
<td></td>
<td></td>
<td>X6, X7</td>
<td></td>
</tr>
<tr>
<td>$b_4(x)$</td>
<td></td>
<td></td>
<td></td>
<td>X8</td>
</tr>
</tbody>
</table>

where

\[
b_1(x) = 1 - 2x, \quad b_2(x) = 1.5(0.9 - x), \quad b_3(x) = 1.5(0.5 - x), \quad b_4(x) = 5(0.05 - x)
\]

\[
\sigma_1(x) = 0.5 + 2x(1 - x), \quad \sigma_2(x) = \sqrt{0.55x(1 - x)}
\]

\[
\sigma_3(x) = \sqrt{0.1x(1 - x)}, \quad \sigma_4(x) = \sqrt{0.8x(1 - x)}
\]

The process $X_9 = 1 - X_1$, hence it has drift $-b_1(x)$ and the same quadratic variation of $X_1$ and $X_{10}$. 
Simulated diffusions
Dendrograms

Markov Operator Distance

Euclidean Distance

STS Distance

DTW Distance
Multidimensional scaling

![Multidimensional scaling graph]

S.M. Iacus 2011
The package `sde` for the R statistical environment is freely available at http://cran.R-Project.org.

It contains the function `MOdist` which calculates the Markov Operator distance and returns a `dist` object.

```r
data(quotes)

d <- MOdist(quotes)
cl <- hclust( d )
groups <- cutree(cl, k=4)
plot(quotes, col=groups)

cmd <- cmdscale(d)
plot( cmd, col=groups)
text( cmd, labels(d), col=groups)
```
Multidimensional scaling. Clustering of NYSE data.
Estimation of Financial Models

Examples

Likelihood approach

Two stage least squares estimation

Model selection

Numerical Evidence

Application to real data

The change point problem

Overview of the yuima package

Option Pricing with R

S.M. Iacus 2011
We focus on the general approach to statistical inference for the parameter $\theta$ of models in the wide class of diffusion processes solutions to stochastic differential equations

$$dX_t = b(\theta, X_t)dt + \sigma(\theta, X_t)dW_t$$

with initial condition $X_0$ and $\theta$ the $p$-dimensional parameter of interest.
- **geometric Brownian motion (gBm)**

  \[
  dX_t = \mu X_t dt + \sigma X_t dW_t
  \]

- **Cox-Ingersoll-Ross (CIR)**

  \[
  dX_t = (\theta_1 + \theta_2 X_t) dt + \theta_3 \sqrt{X_t} dW_t
  \]

- **Chan-Karolyi-Longstaff-Sanders (CKLS)**

  \[
  dX_t = (\theta_1 + \theta_2 X_t) dt + \theta_3 X_t^{\theta_4} dW_t
  \]
- **nonlinear mean reversion (Aït-Sahalia)**
  \[
  dX_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)dt + \beta_1 X_t^\rho dW_t
  \]

- **double Well potential (bimodal behaviour, highly nonlinear)**
  \[
  dX_t = (X_t - X_t^3)dt + dW_t
  \]

- **Jacobi diffusion (political polarization)**
  \[
  dX_t = -\theta \left( X_t - \frac{1}{2} \right) dt + \sqrt{\theta X_t(1 - X_t)}dW_t
  \]
- **Ornstein-Uhlenbeck (OU)**

  \[ dX_t = \theta X_t dt + dW_t \]

- **radial Ornstein-Uhlenbeck**

  \[ dX_t = (\theta X_t^{-1} - X_t)dt + dW_t \]

- **hyperbolic diffusion (dynamics of sand)**

  \[ dX_t = \frac{\sigma^2}{2} \left[ \beta - \gamma \frac{X_t}{\sqrt{\delta^2 + (X_t - \mu)^2}} \right] dt + \sigma dW_t \]

...and so on.
Likelihood approach
Consider the model
\[ dX_t = b(\theta, X_t)dt + \sigma(\theta, X_t)dW_t \]
with initial condition \( X_0 \) and \( \theta \) the \( p \)-dimensional parameter of interest. By Markov property of diffusion processes, the likelihood has this form
\[ L_n(\theta) = \prod_{i=1}^{n} p_\theta (\Delta, X_i|X_{i-1}) p_\theta (X_0) \]
and the log-likelihood is
\[ \ell_n(\theta) = \log L_n(\theta) = \sum_{i=1}^{n} \log p_\theta (\Delta, X_i|X_{i-1}) + \log (p_\theta (X_0)). \]

Assumption \( p_\theta (X_0) \) irrelevant, hence \( p_\theta (X_0) = 1 \)

As usual, in most cases we don’t know the conditional distribution \( p_\theta (\Delta, X_i|X_{i-1}) \)
Maximum Likelihood Estimators (MLE)

If we study the likelihood $L_n(\theta)$ as a function of $\theta$ given the $n$ numbers $(X_1 = x_1, \ldots, X_n = x_n)$ and we find that this function has a maximum, we can use this maximum value as an estimate of $\theta$.

In general we define *maximum likelihood estimator* of $\theta$, and we abbreviate this with $MLE$, the following estimator

\[ \hat{\theta}_n = \arg \max_{\theta \in \Theta} L_n(\theta) \]

\[ = \arg \max_{\theta \in \Theta} L_n(\theta | X_1, X_2, \ldots, X_n) \]

provided that the maximum exists.
It is not always the case, that maximum likelihood estimators can be obtained in explicit form.

For what concerns applications to real data, it is important to know that MLE are optimal in many respect from the statistical point of view, compared to, e.g. estimators obtained via the method of the moments (GMM) which can produce fairly biased estimates of the models.

R offers a prebuilt generic function called `mle` in the package `stats4` which can be used to maximize a likelihood.

The `mle` function actually minimizes the negative log-likelihood \(-\ell(\theta)\) as a function of the parameter \(\theta\) where \(\ell(\theta) = \log L(\theta)\).
Consider the multidimensional diffusion process

\[ dX_t = b(\theta_2, X_t)dt + \sigma(\theta_1, X_t)dW_t \]

where \( W_t \) is an \( r \)-dimensional standard Wiener process independent of the initial value \( X_0 = x_0 \). Quasi-MLE assumes the following approximation of the true log-likelihood for multidimensional diffusions

\[
el_n(X_n, \theta) = -\frac{1}{2} \sum_{i=1}^{n} \left\{ \log \det(\Sigma_{i-1}(\theta_1)) + \frac{1}{\Delta_n}(\Delta X_i - \Delta_n b_{i-1}(\theta_2))^T \Sigma_{i-1}^{-1}(\theta_1)(\Delta X_i - \Delta_n b_{i-1}(\theta_2)) \right\}
\]

where \( \theta = (\theta_1, \theta_2) \), \( \Delta X_i = X_{t_i} - X_{t_{i-1}} \), \( \Sigma_i(\theta_1) = \Sigma(\theta_1, X_{t_i}) \), \( b_i(\theta_2) = b(\theta_2, X_{t_i}) \), \( \Sigma = \sigma \otimes^2 \), \( A \otimes^2 = A^T A \) and \( A^{-1} \) the inverse of \( A \). Then the QML estimator of \( \theta \) is

\[
\tilde{\theta}_n = \arg \min_{\theta} \ell_n(X_n, \theta)
\]
Consider the matrix

$$
\varphi(n) = \begin{pmatrix}
\frac{1}{n\Delta_n} I_p & 0 \\
0 & \frac{1}{n} I_q
\end{pmatrix}
$$

where $I_p$ and $I_q$ are respectively the identity matrix of order $p$ and $q$. Then, is possible to show that

$$
\varphi(n)^{-1/2}(\hat{\theta}_n - \theta) \xrightarrow{d} N(0, \mathcal{I}(\theta)^{-1}).
$$

where $\mathcal{I}(\theta)$ is the Fisher information matrix.

The above result means that the estimator of the parameter of the drift converges slowly to the true value because $n\Delta_n = T \to \infty$ slower than $n$, as $n \to \infty$. 
QUASI - MLE using yuima package

To estimate a model we make use of the qmle function in the yuima package, whose interface is similar to mle. Consider the model

\[ dX_t = -\theta_2 X_t dt + \theta_1 dW_t \]

with \( \theta_1 = 0.3 \) and \( \theta_2 = 0.1 \)

R> library(yuima)

R> ymodel <- setModel(drift = c("(-1)*theta2*x"),
+    diffusion = matrix(c("theta1"), 1, 1) )
R> n <- 100
R> ysamp <- setSampling(Terminal = (n)^(1/3), n = n)
R> yuima <- setYuima(model = ymodel, sampling = ysamp)
R> set.seed(123)
R> yuima <- simulate(yuima, xinit = 1, true.parameter = list(theta1 = 0.3,
+    theta2 = 0.1))
We can now try to estimate the parameters

\[
\begin{align*}
R & \texttt{> mle1 <- qmle(yuima, start = list(theta1 = 0.8, theta2 = 0.7),} \\
& \quad + \texttt{lower = list(theta1=0.05, theta2=0.05),} \\
& \quad + \texttt{upper = list(theta1=0.5, theta2=0.5), method = "L-BFGS-B")} \\
R & \texttt{> coef(mle1)} \\
& \quad \texttt{theta1 theta2} \\
& \quad 0.30766981 0.05007788 \\
R & \texttt{> summary(mle1)} \\
& \texttt{Maximum likelihood estimation} \\
& \texttt{Coefficients:} \\
& \quad \texttt{Estimate Std. Error} \\
& \quad \texttt{theta1 0.30766981 0.02629925} \\
& \quad \texttt{theta2 0.05007788 0.15144393} \\
& \texttt{-2 log L: -280.0784} \\

not very good estimate of the drift so now we consider a longer time series.
QUASI - MLE. True values: $\theta_1 = 0.3, \theta_2 = 0.1$

R> n <- 1000
R> ysamp <- setSampling(Terminal = (n)^(1/3), n = n)
R> yuima <- setYuima(model = ymodel, sampling = ysamp)
R> set.seed(123)
R> yuima <- simulate(yuima, xinit = 1, true.parameter = list(theta1 = 0.3, + theta2 = 0.1))
R> mle1 <- qmle(yuima, start = list(theta1 = 0.8, theta2 = 0.7), + lower = list(theta1=0.05, theta2=0.05),
+ upper = list(theta1=0.5, theta2=0.5), method = "L-BFGS-B")
R> coef(mle1)
  theta1  theta2
0.3015202 0.1029822
R> summary(mle1)
Maximum likelihood estimation

Coefficients:
   Estimate Std. Error
theta1 0.3015202 0.006879348
theta2 0.1029822 0.114539931

-2 log L: -4192.279
Two stage least squares estimation
Aït-Sahalia in 1996 proposed a model much more sophisticated than the CKLS to include other polynomial terms.

\[
\text{d}X_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1X_t + \alpha_2X_t^2)\text{d}t + \beta_1X_t^\rho\text{d}B_t
\]

He proposed to use the two stage least squares approach in the following manner.
Aït-Sahalia’s interest rates model

Aït-Sahalia in 1996 proposed a model much more sophisticated than the CKLS to include other polynomial terms.

\[ dX_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)dt + \beta_1 X_t^\rho dB_t \]

He proposed to use the two stage least squares approach in the following manner. First estimate the drift coefficients with a simple linear regression like

\[ \mathbb{E}(X_{t+1} - X_t | X_t) = \alpha_{-1}X_t^{-1} + \alpha_0 + (\alpha_1 - 1)X_t + \alpha_2 X_t^2 \]
Aït-Sahalia in 1996 proposed a model much more sophisticated than the CKLS to include other polynomial terms.

\[ dX_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1X_t + \alpha_2X_t^2)dt + \beta_1X_t^\rho dB_t \]

He proposed to use the two stage least squares approach in the following manner. First estimate the drift coefficients with a simple linear regression like

\[ \mathbb{E}(X_{t+1} - X_t | X_t) = \alpha_{-1}X_t^{-1} + \alpha_0 + (\alpha_1 - 1)X_t + \alpha_2X_t^2 \]

then, regress the residuals \( \epsilon_{t+1}^2 \) from the first regression to obtain the estimates of the coefficients in the diffusion term with

\[ \mathbb{E}(\epsilon_{t+1}^2 | X_t) = \beta_0 + \beta_1X_t + \beta_2X_t^{\beta_3} \]
Aït-Sahalia in 1996 proposed a model much more sophisticated than the CKLS to include other polynomial terms.

\[
dX_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)dt + \beta_1 X_t^\rho dB_t
\]

He proposed to use the two stage least squares approach in the following manner. First estimate the drift coefficients with a simple linear regression like

\[
E(X_{t+1} - X_t | X_t) = \alpha_{-1}X_t^{-1} + \alpha_0 + (\alpha_1 - 1)X_t + \alpha_2 X_t^2
\]

then, regress the residuals \( \epsilon_{t+1}^2 \) from the first regression to obtain the estimates of the coefficients in the diffusion term with

\[
E(\epsilon_{t+1}^2 | X_t) = \beta_0 + \beta_1 X_t + \beta_2 X_t^{\beta_3}
\]

and finally, use the fitted values from the last regression to set the weights in the second stage regression for the drift.
Stage one: regression

\[ E(\Delta X_t | X_t) = \alpha_{-1} X_t^{-1} + \alpha_0 + (\alpha_1 - 1) X_t + \alpha_2 X_t^2 \]

R> library(Ecdat)
R> data(Irates)
R> X <- Irates[, "r1"]
R> Y <- X[-1]
R> stage1 <- lm(diff(X) ~ I(1/Y) + Y + I(Y^2))
R> coef(stage1)
   (Intercept)   I(1/Y)        Y    I(Y^2)
  0.253478884 -0.135520883 -0.094028341  0.007892979
Example of 2SLS estimation

Stage regression now we extract the squared residuals and run a regression on those

\[
E(\epsilon_{t+1}^2|X_t) = \beta_0 + \beta_1 X_t + \beta_2 X_t^{\beta_3}
\]

R> eps2 <- residuals(stage1)^2
R> mod <- nls(eps2 ~ b0 + b1 * Y + b2 * Y^b3, start = list(b0 = 1,
+ b1 = 1, b2 = 1, b3 = 0.5), lower = rep(1e-05, 4), upper = rep(2,
+ 4), algorithm = "port")
R> w <- predict(mod)

and we run a second stage regression

R> stage2 <- lm(diff(X) ~ I(1/Y) + Y + I(Y^2), weights = 1/w)
R> coef(stage2)
   (Intercept) I(1/Y) Y I(Y^2)
-0.189414678 -0.104711262 -0.073227254 0.006442481
R> coef(mod)
   b0 b1 b2 b3
0.00001000 0.09347354 0.00001000 0.00001000
Model selection
The aim is to try to identify the underlying continuous model on the basis of discrete observations using AIC (Akaike Information Criterion) statistics defined as (Akaike 1973, 1974)

\[
AIC = -2\ell_n \left( \hat{\theta}_n^{(ML)} \right) + 2 \dim(\Theta),
\]

where \( \hat{\theta}_n^{(ML)} \) is the true maximum likelihood estimator and \( \ell_n(\theta) \) is the log-likelihood.
Model selection via AIC

Akaike’s index idea $\text{AIC} = -2\ell_n\left(\hat{\theta}_n^{(ML)}\right) + 2 \dim(\Theta)$ is to penalize this value

$$-2\ell_n\left(\hat{\theta}_n^{(ML)}\right)$$

with the dimension of the parameter space

$$2 \dim(\Theta)$$

Thus, as the number of parameter increases, the fit may be better, i.e. $-2\ell_n\left(\hat{\theta}_n^{(ML)}\right)$ decreases, at the cost of overspecification and $\dim(\Theta)$ compensate for this effect.

When comparing several models for a given data set, the models such that the AIC is lower is preferred.
In order to calculate

$$\text{AIC} = -2\ell_n\left(\hat{\theta}_n^{(ML)}\right) + 2 \dim(\Theta),$$

we need to evaluate the exact value of the log-likelihood $\ell_n(\cdot)$ at point $\hat{\theta}_n^{(ML)}$.

**Problem:** for discretely observed diffusion processes the true likelihood function is not known in most cases.

Uchida and Yoshida (2005) develop the AIC statistics defined as

$$\text{AIC} = -2\tilde{\ell}_n\left(\hat{\theta}_n^{(QML)}\right) + 2 \dim(\Theta),$$

where $\hat{\theta}_n^{(QML)}$ is the quasi maximum likelihood estimator and $\tilde{\ell}_n$ the local Gaussian approximation of the true log-likelihood.
We compare three models

\[ dX_t = -\alpha_1 (X_t - \alpha_2) dt + \beta_1 \sqrt{X_t} dW_t \]  
\[ dX_t = -\alpha_1 (X_t - \alpha_2) dt + \sqrt{\beta_1 + \beta_2 X_t} dW_t \]  
\[ dX_t = -\alpha_1 (X_t - \alpha_2) dt + (\beta_1 + \beta_2 X_t)^{\beta_3} dW_t \]

(true model),  
(competing model 1),  
(competing model 2),

We call the above models Mod1, Mod2 and Mod3.

We generate data from Mod1 with parameters

\[ dX_t = -(X_t - 10) dt + 2 \sqrt{X_t} dW_t , \]

and initial value \( X_0 = 8 \). We use \( n = 1000 \) and \( \Delta = 0.1 \).

We test the performance of the AIC statistics for the three competing models
Simulation results. 1000 Monte Carlo replications

\[ dX_t = -(X_t - 10)dt + 2\sqrt{X_t}dW_t \]  
(true model),

\[ dX_t = -\alpha_1 (X_t - \alpha_2)dt + \beta_1 \sqrt{X_t}dW_t \]  
(Model 1)

\[ dX_t = -\alpha_1 (X_t - \alpha_2)dt + \sqrt{\beta_1 + \beta_2 X_t}dW_t \]  
(Model 2)

\[ dX_t = -\alpha_1 (X_t - \alpha_2)dt + (\beta_1 + \beta_2 X_t)^{\beta_3}dW_t \]  
(Model 3)

Model selection via AIC

<table>
<thead>
<tr>
<th>Model</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(true)</td>
<td>99.2 %</td>
<td>0.6 %</td>
<td>0.2 %</td>
</tr>
</tbody>
</table>

QMLE estimates under the different models

<table>
<thead>
<tr>
<th></th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
<th>$\beta_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>1.10</td>
<td>8.05</td>
<td>0.90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model 2</td>
<td>1.12</td>
<td>8.07</td>
<td>2.02</td>
<td>0.54</td>
<td></td>
</tr>
<tr>
<td>Model 3</td>
<td>1.12</td>
<td>9.03</td>
<td>7.06</td>
<td>7.26</td>
<td>0.61</td>
</tr>
</tbody>
</table>
The **Least Absolute Shrinkage and Selection Operator (LASSO)** is a useful and well studied approach to the problem of model selection and its major advantage is the simultaneous execution of both parameter estimation and variable selection (see Tibshirani, 1996; Knight and Fu, 2000, Efron et al., 2004).

To simplify the idea: take a full specified regression model

\[
Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_k X_k
\]

perform least squares estimation under $L_1$ constraints, i.e.

\[
\hat{\theta} = \arg \min_{\theta} \left\{ (Y - \theta X)^T (Y - \theta X) + \sum_{i=1}^{k} |\theta_i| \right\}
\]

model selection occurs when some of the $\theta_i$ are estimated as zeros.
Let $X_t$ be a diffusion process solution to

$$dX_t = b(\alpha, X_t)dt + \sigma(\beta, X_t)dW_t$$

$$\alpha = (\alpha_1, ..., \alpha_p)' \in \Theta_p \subset \mathbb{R}^p, \quad p \geq 1$$

$$\beta = (\beta_1, ..., \beta_q)' \in \Theta_q \subset \mathbb{R}^q, \quad q \geq 1$$

$b : \Theta_p \times \mathbb{R}^d \rightarrow \mathbb{R}^d$, $\sigma : \Theta_q \times \mathbb{R}^d \rightarrow \mathbb{R}^d \times \mathbb{R}^m$ and $W_t, t \in [0, T]$, is a standard Brownian motion in $\mathbb{R}^m$.

We assume that the functions $b$ and $\sigma$ are known up to $\alpha$ and $\beta$.

We denote by $\theta = (\alpha, \beta) \in \Theta_p \times \Theta_q = \Theta$ the parametric vector and with $\theta_0 = (\alpha_0, \beta_0)$ its unknown true value.

Let $H_n(X_n, \theta) = -\ell_n(\theta)$ from the local Gaussian approximation.
The classical adaptive LASSO objective function for the present model is then

$$\min_{\alpha, \beta} \left\{ \mathcal{H}_n(\alpha, \beta) + \sum_{j=1}^{p} \lambda_{n,j} |\alpha_j| + \sum_{k=1}^{q} \gamma_{n,k} |\beta_k| \right\}$$

$\lambda_{n,j}$ and $\gamma_{n,k}$ are appropriate sequences representing an adaptive amount of shrinkage for each element of $\alpha$ and $\beta$. 
Idea of Quadratic Approximation

By Taylor expansion of the original LASSO objective function, for $\theta$ around $\tilde{\theta}_n$ (the QMLE estimator)

$$
\mathbb{H}_n(X_n, \theta) = \mathbb{H}_n(X_n, \tilde{\theta}_n) + (\theta - \tilde{\theta}_n)'\mathbb{H}_n(X_n, \tilde{\theta}_n) + \frac{1}{2}(\theta - \tilde{\theta}_n)'\mathbb{H}_n(X_n, \tilde{\theta}_n)(\theta - \tilde{\theta}_n) + o_p(1)
$$

with $\mathbb{H}_n$ and $\mathbb{H}_n$ the gradient and Hessian of $\mathbb{H}_n$ with respect to $\theta$. 
Finally, the adaptive LASSO estimator is defined as the solution to the quadratic problem under $L_1$ constraints

$$\hat{\theta}_n = (\hat{\alpha}_n, \hat{\beta}_n) = \arg \min_{\theta} \mathcal{F}(\theta).$$

with

$$\mathcal{F}(\theta) = (\theta - \tilde{\theta}_n)^{\prime} \mathbb{H}_n(\mathbf{X}_n, \tilde{\theta}_n)(\theta - \tilde{\theta}_n) + \sum_{j=1}^{p} \lambda_{n,j} |\alpha_j| + \sum_{k=1}^{q} \gamma_{n,k} |\beta_k|$$

The lasso method is implemented in the \texttt{yuima} package.
How to choose the adaptive sequences

Clearly, the theoretical and practical implications of our method rely to the specification of the tuning parameter $\lambda_{n,j}$ and $\gamma_{n,k}$.

The tuning parameters should be chosen as is Zou (2006) in the following way

$$
\lambda_{n,j} = \lambda_0 |\tilde{\alpha}_{n,j}|^{-\delta_1}, \quad \gamma_{n,k} = \gamma_0 |\tilde{\beta}_{n,k}|^{-\delta_2}
$$

(4)

where $\tilde{\alpha}_{n,j}$ and $\tilde{\beta}_{n,k}$ are the unpenalized QML estimator of $\alpha_j$ and $\beta_k$ respectively, $\delta_1, \delta_2 > 0$ and usually taken unitary.
Numerical Evidence

Application to real data

The change point problem

Overview of the \texttt{yuima} package

Option Pricing with R
We consider this two dimensional geometric Brownian motion process solution to the stochastic differential equation

\[
\begin{pmatrix}
\frac{dX_t}{dY_t}
\end{pmatrix} = \begin{pmatrix}
1 - \mu_{11}X_t + \mu_{12}Y_t \\
2 + \mu_{21}X_t - \mu_{22}Y_t
\end{pmatrix} dt + \begin{pmatrix}
\sigma_{11}X_t & -\sigma_{12}Y_t \\
\sigma_{21}X_t & \sigma_{22}Y_t
\end{pmatrix} \begin{pmatrix}
dW_t \\
 dB_t
\end{pmatrix}
\]

with initial condition \((X_0 = 1, Y_0 = 1)\) and \(W_t, t \in [0, T]\), and \(B_t, t \in [0, T]\), are two independent Brownian motions.

This model is a classical model for pricing of basket options in mathematical finance.

We assume that \(\alpha = (\mu_{11} = 0.9, \mu_{12} = 0, \mu_{21} = 0, \mu_{22} = 0.7)'\) and \(\beta = (\sigma_{11} = 0.3, \sigma_{12} = 0, \sigma_{21} = 0, \sigma_{22} = 0.2)'\), \(\theta = (\alpha, \beta)\).
### Results

<table>
<thead>
<tr>
<th></th>
<th>$\mu_{11}$</th>
<th>$\mu_{12}$</th>
<th>$\mu_{21}$</th>
<th>$\mu_{22}$</th>
<th>$\sigma_{11}$</th>
<th>$\sigma_{12}$</th>
<th>$\sigma_{21}$</th>
<th>$\sigma_{22}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True</strong></td>
<td>0.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td><strong>Qmle: $n = 100$</strong></td>
<td>0.96</td>
<td>0.05</td>
<td>0.25</td>
<td>0.81</td>
<td>0.30</td>
<td>0.04</td>
<td>0.01</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>(0.08)</td>
<td>(0.06)</td>
<td>(0.27)</td>
<td>(0.15)</td>
<td>(0.03)</td>
<td>(0.05)</td>
<td>(0.02)</td>
<td>(0.02)</td>
</tr>
<tr>
<td><strong>Lasso: $\lambda_0 = \gamma_0 = 1, n = 100$</strong></td>
<td>0.86</td>
<td>0.00</td>
<td>0.05</td>
<td>0.71</td>
<td>0.30</td>
<td>0.02</td>
<td>0.01</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>(0.12)</td>
<td>(0.00)</td>
<td>(0.13)</td>
<td>(0.09)</td>
<td>(0.03)</td>
<td>(0.05)</td>
<td>(0.02)</td>
<td>(0.02)</td>
</tr>
<tr>
<td><strong>% of times $\theta_i = 0$</strong></td>
<td>0.0</td>
<td>99.9</td>
<td>80.2</td>
<td>0.0</td>
<td>0.3</td>
<td>67.2</td>
<td>66.7</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>Lasso: $\lambda_0 = \gamma_0 = 5, n = 100$</strong></td>
<td>0.82</td>
<td>0.00</td>
<td>0.00</td>
<td>0.66</td>
<td>0.29</td>
<td>0.01</td>
<td>0.00</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>(0.12)</td>
<td>(0.00)</td>
<td>(0.00)</td>
<td>(0.09)</td>
<td>(0.03)</td>
<td>(0.03)</td>
<td>(0.01)</td>
<td>(0.02)</td>
</tr>
<tr>
<td><strong>% of times $\theta_i = 0$</strong></td>
<td>0.0</td>
<td>100.0</td>
<td>99.9</td>
<td>0.0</td>
<td>0.4</td>
<td>86.9</td>
<td>89.7</td>
<td>0.2</td>
</tr>
<tr>
<td><strong>Qmle: $n = 1000$</strong></td>
<td>0.95</td>
<td>0.03</td>
<td>0.21</td>
<td>0.79</td>
<td>0.30</td>
<td>0.04</td>
<td>0.01</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>(0.07)</td>
<td>(0.04)</td>
<td>(0.25)</td>
<td>(0.13)</td>
<td>(0.03)</td>
<td>(0.06)</td>
<td>(0.02)</td>
<td>(0.02)</td>
</tr>
<tr>
<td><strong>Lasso: $\lambda_0 = \gamma_0 = 1, n = 1000$</strong></td>
<td>0.88</td>
<td>0.00</td>
<td>0.08</td>
<td>0.73</td>
<td>0.30</td>
<td>0.02</td>
<td>0.01</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>(0.08)</td>
<td>(0.00)</td>
<td>(0.16)</td>
<td>(0.09)</td>
<td>(0.03)</td>
<td>(0.05)</td>
<td>(0.01)</td>
<td>(0.02)</td>
</tr>
<tr>
<td><strong>% of times $\theta_i = 0$</strong></td>
<td>0.0</td>
<td>99.7</td>
<td>72.1</td>
<td>0.0</td>
<td>0.1</td>
<td>67.5</td>
<td>66.6</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>Lasso: $\lambda_0 = \gamma_0 = 5, n = 1000$</strong></td>
<td>0.86</td>
<td>0.00</td>
<td>0.00</td>
<td>0.68</td>
<td>0.29</td>
<td>0.01</td>
<td>0.00</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>(0.09)</td>
<td>(0.00)</td>
<td>(0.01)</td>
<td>(0.06)</td>
<td>(0.03)</td>
<td>(0.04)</td>
<td>(0.01)</td>
<td>(0.02)</td>
</tr>
<tr>
<td><strong>% of times $\theta_i = 0$</strong></td>
<td>0.0</td>
<td>100.0</td>
<td>99.4</td>
<td>0.0</td>
<td>0.2</td>
<td>87.8</td>
<td>89.9</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Application to real data
Interest rates LASSO estimation examples
LASSO estimation of the U.S. Interest Rates monthly data from 06/1964 to 12/1989. These data have been analyzed by many authors including Newman (1997), Aït-Sahalia (1996), Yu and Phillips (2001) and it is a nice application of LASSO.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Model</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merton (1973)</td>
<td>( dX_t = \alpha dt + \sigma dW_t )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vasicek (1977)</td>
<td>( dX_t = (\alpha + \beta X_t)dt + \sigma dW_t )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cox, Ingersoll and Ross (1985)</td>
<td>( dX_t = (\alpha + \beta X_t)dt + \sigma \sqrt{X_t}dW_t )</td>
<td>0</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>Dothan (1978)</td>
<td>( dX_t = \sigma X_t dW_t )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Geometric Brownian Motion</td>
<td>( dX_t = \beta X_t dt + \sigma X_t dW_t )</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Brennan and Schwartz (1980)</td>
<td>( dX_t = (\alpha + \beta X_t)dt + \sigma X_t dW_t )</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cox, Ingersoll and Ross (1980)</td>
<td>( dX_t = \sigma X_t^{3/2} dW_t )</td>
<td>0</td>
<td>0</td>
<td>3/2</td>
</tr>
<tr>
<td>Constant Elasticity Variance</td>
<td>( dX_t = \beta X_t dt + \sigma X_t^\gamma dW_t )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CKLS (1992)</td>
<td>( dX_t = (\alpha + \beta X_t)dt + \sigma X_t^\gamma dW_t )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Interest rates LASSO estimation examples

<table>
<thead>
<tr>
<th>Model</th>
<th>Estimation Method</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\sigma$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vasicek</td>
<td>MLE</td>
<td>4.1889</td>
<td>-0.6072</td>
<td>0.8096</td>
<td>-</td>
</tr>
<tr>
<td>CKLS</td>
<td>Nowman</td>
<td>2.4272</td>
<td>-0.3277</td>
<td>0.1741</td>
<td>1.3610</td>
</tr>
<tr>
<td>CKLS</td>
<td>Exact Gaussian (Yu &amp; Phillips)</td>
<td>2.0069</td>
<td>-0.3330</td>
<td>0.1741</td>
<td>1.3610</td>
</tr>
<tr>
<td>CKLS</td>
<td>QMLE</td>
<td>2.0822</td>
<td>-0.2756</td>
<td>0.1322</td>
<td>1.4392</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.9635)</td>
<td>(0.1895)</td>
<td>(0.0253)</td>
<td>(0.1018)</td>
</tr>
<tr>
<td>CKLS</td>
<td>QMLE + LASSO</td>
<td>1.5435</td>
<td>-0.1687</td>
<td>0.1306</td>
<td>1.4452</td>
</tr>
<tr>
<td></td>
<td>with mild penalization</td>
<td>(0.6813)</td>
<td>(0.1340)</td>
<td>(0.0179)</td>
<td>(0.0720)</td>
</tr>
<tr>
<td>CKLS</td>
<td>QMLE + LASSO</td>
<td>0.5412</td>
<td>0.0001</td>
<td>0.1178</td>
<td>1.4944</td>
</tr>
<tr>
<td></td>
<td>with strong penalization</td>
<td>(0.2076)</td>
<td>(0.0054)</td>
<td>(0.0179)</td>
<td>(0.0720)</td>
</tr>
</tbody>
</table>

LASSO selected: Cox, Ingersoll and Ross (1980) model

$$dX_t = \frac{1}{2} dt + 0.12 \cdot X_t^{3/2} dW_t$$
An example of Lasso estimation using `yuima` package. We make use of real data with CKLS model

\[ dX_t = (\alpha + \beta X_t)dt + \sigma X_t^\gamma dW_t \]

```r
R> library(Ecdat)
R> data(Irates)
R> rates <- Irates[, "r1"]
R> plot(rates)
R> require(yuima)
R> X <- window(rates, start=1964.471, end=1989.333)
R> mod <- setModel(drift="alpha+beta*x", diffusion=matrix("sigma*x^gamma",1,1))
R> yuima <- setYuima(data=setData(X), model=mod)
```
Example of Lasso estimation

```R
R> lambda10 <- list(alpha=10, beta =10, sigma =10, gamma =10)
R> start <- list(alpha=1, beta =-.1, sigma =.1, gamma =1)
R> low <- list(alpha=-5, beta =-5, sigma =-5, gamma =-5)
R> upp <- list(alpha=8, beta =8, sigma =8, gamma =8)
R> lasso10 <- lasso(yuima, lambda10, start=start, lower=low, upper=upp, method="L-BFGS-B")
Looking for MLE estimates...
Performing LASSO estimation...

R> round(lasso10$mle, 3) # QMLE
  sigma gamma alpha beta
  0.133  1.443  2.076 -0.263

R> round(lasso10$lasso, 3) # LASSO
  sigma gamma alpha beta
  0.117  1.503  0.591  0.000
```

\[ dX_t = (\alpha + \beta X_t)dt + \sigma X_t^\gamma dW_t \]

\[ dX_t = 0.6dt + 0.12X_t^{\frac{3}{2}}dW_t \]
The change point problem
Discover from the data a structural change in the generating model.
Discover from the data a structural change in the generating model. Next famous example shows historical change points for the Dow-Jones index.
Discover from the data a structural change in the generating model. Next famous example shows historical change points for the Dow-Jones index.

break of gold-US$ linkage (left)  

Watergate scandal (right)
De Gregorio and I. (2008) considered the change point problem for the ergodic model

\[ dX_t = b(X_t)dt + \sqrt{\theta} \sigma(X_t)dW_t, \quad 0 \leq t \leq T, X_0 = x_0, \]

observed at discrete time instants. The change point problem is formulated as follows

\[ X_t = \begin{cases} 
X_0 + \int_0^t b(X_s)ds + \sqrt{\theta_1} \int_0^t \sigma(X_s)dW_s, & 0 \leq t \leq \tau^* \\
X_{\tau^*} + \int_{\tau^*}^t b(X_s)ds + \sqrt{\theta_2} \int_{\tau^*}^t \sigma(X_s)dW_s, & \tau^* < t \leq T 
\end{cases} \]

where \( \tau^* \in (0, T) \) is the change point and \( \theta_1, \theta_2 \) two parameters to be estimated.
Let \( Z_i = \frac{X_{i+1} - X_i - b(X_i) \Delta_n}{\sqrt{\Delta_n} \sigma(X_i)} \) be the standardized residuals. Then the LS estimator of \( \tau \) is \( \hat{\tau}_n = \frac{\hat{k}_0}{n} \) where \( \hat{k}_0 \) is solution to

\[
\hat{k}_0 = \arg \max_k \left| \frac{k}{n} - \frac{S_k}{S_n} \right|,
\]

\( S_k = \sum_{i=1}^k Z_i^2. \)
Let \( Z_i = \frac{X_{i+1} - X_i - b(X_i)\Delta_n}{\sqrt{\Delta_n} \sigma(X_i)} \) be the standardized residuals. Then the LS estimator of \( \tau \) is \( \hat{\tau}_n = \hat{k}_0 / n \) where \( \hat{k}_0 \) is solution to

\[
\hat{k}_0 = \arg \max_k \left| \frac{k}{n} - \frac{S_k}{S_n} \right|, \quad S_k = \sum_{i=1}^k Z_i^2.
\]

If we assume to observe the following SDE with unknown drift \( b(\cdot) \)

\[
dX_t = b(X_t)dt + \sqrt{\theta}dW_t,
\]

we can still estimate \( b(\cdot) \) non parametrically and obtain a good change point estimator.

This approach has been used in Smaldone (2009) to re-analyze the recent global financial crisis using data from different markets.
cpoint function in the sde package

```r
require(sde)
cpoint(S)
$k0
[1] 123
$tau0
$theta1
2005-01-12
0.102001
$theta2
[1] 0.377011
attr("index")
[1] "2005-01-12"
```
The theory works for SDEs of the form

\[ dY_t = b_t dt + \sigma(X_t, \theta)dW_t, \quad t \in [0, T], \]

where \( W_t \) a \( r \)-dimensional Wiener process and \( b_t \) and \( X_t \) are multidimensional processes and \( \sigma \) is the diffusion coefficient (volatility) matrix.

When \( Y = X \) the problem is a diffusion model.

The process \( b_t \) may have jumps but should not explode and it is treated as a nuisance in this model and it is completely unspecified.
The change-point problem for the volatility is formalized as follows

\[ Y_t = \begin{cases} 
Y_0 + \int_0^t b_s ds + \int_0^t \sigma(X_s, \theta_1^*) dW_s & \text{for } t \in [0, \tau^*) \\
Y_{\tau^*} + \int_{\tau^*}^t b_s ds + \int_{\tau^*}^t \sigma(X_s, \theta_2^*) dW_s & \text{for } t \in [\tau^*, T].
\end{cases} \]

The change point \( \tau^* \) instant is unknown and is to be estimated, along with \( \theta_1^* \) and \( \theta_2^* \), from the observations sampled from the path of \((X, Y)\).
Consider the 2-dimensional stochastic differential equation

\[
\begin{pmatrix}
\frac{dX^1_t}{dt} \\
\frac{dX^2_t}{dt}
\end{pmatrix} =
\begin{pmatrix}
1 - X^1_t \\
3 - X^2_t
\end{pmatrix} dt +
\begin{bmatrix}
\theta_{1.1} \cdot X^1_t & 0 \cdot X^1_t \\
0 \cdot X^2_t & \theta_{2.2} \cdot X^2_t
\end{bmatrix}'
\begin{pmatrix}
dW^1_t \\
dW^2_t
\end{pmatrix}
\]

with change point instant at time \( \tau = 4 \)
the CPoint function in the yuima package

the object yuima contains the model and the data from the previous slide so we can call CPoint on this yuima object

```r
> t.est <- CPoint(yuima,param1=t1,param2=t2, plot=TRUE)
> t.est$tau
[1] 3.99
```
Overview of the \texttt{yuima} package

Yuima object functionalities how to use it

Explorative Data Analysis
Estimation of Financial Models
Likelihood approach
Two stage least squares estimation
Model selection
Numerical Evidence
Application to real data
The change point problem
Overview of the \texttt{yuima} package
Option Pricing with R

S.M. Iacus 2011
The main object is the `yuima` object which allows to describe the model in a mathematically sound way.

Then the data and the sampling structure can be included as well or, just the sampling scheme from which data can be generated according to the model.

The package exposes very few generic functions like `simulate`, `qmle`, `plot`, etc. and some other specific functions for special tasks.

Before looking at the details, let us see an overview of the main object.
Yuima

Sampling random
deterministic tick times
space disc.

Data
univariate, multivariate, t, xts, ...

Model
diffusion, Lévy, fractional BM
Markov, Switching HMM
Yuima

Sampling
random
deterministic
tick times
space disc.

... regular irregular multigrid asynch.

Data
univariate
multivariate
ts, xts, ...

Model
diffusion Lévy
fractional BM
Markov Switching
HMM
Yuima

Data

- univariate
- multivariate

- ts, xts, ...

Model

- diffusion
- Lévy

- fractional
- BM

- Markov
- Switching

- HMM
Yuima

Sampling
- multigrid
- asynchronous
- regular
- irregular
- tick times
- space discretization

Data
- univariate
- multivariate
- time series
- xts

Model
- diffusion
- Lévy
- fractional Brownian motion
- Markov switching
- HMM
Yuima
Yuima

Euler-Maruyama
Exact Simulation
Space discr.

Nonparametrics
Covariation
p-variation

Parametric
Inference
High freq.
Low freq.

Quasi
MLE
Diff., Jumps,
fBM

Adaptive
Bayes
MCMC
Change point

Model selection
Akaike's
LASSO-type
Hypotheses Testing

Option pricing
Asymptotic expansion
Monte Carlo
Yuima

- Euler-Maruyama
- Simulation
- Exact
- Covariation
- Nonparametric
- p-variation
- Space discr.

Yuima

- High freq.
- Low freq.
- Quasi
- MLE
- Diff,
- Jumps,
- fBM

- Adaptive
- Bayes
- MCMC
- Change point
- Model selection
- Akaike's
- LASSO-type
- Hypotheses Testing
- Option pricing
- Asymptotic expansion
- Monte Carlo
Yuima Simulations

- Exact Euler-Maruyama
- Covariation
- Nonparametric
- p-variation

Simulation

- Space discr.

Covariation

Nonparametric

p-variation

Parametric Inference

- High freq. Low freq.
- Jumps, fBM
- Quasi MLE Diff.

Option Pricing

- Asymptotic expansion
- Monte Carlo

Hypotheses Testing

- Change point
- Adaptive Bayes MCMC

Model Selection

- Akaike's LASSO-type
We consider here the three main classes of SDE’s which can be easily specified. All multidimensional and eventually parametric models.
We consider here the three main classes of SDE’s which can be easily specified. All multidimensional and eventually parametric models.

- **Diffusions**
  \[ dX_t = a(t, X_t) dt + b(t, X_t) dW_t \]
We consider here the three main classes of SDE’s which can be easily specified. All multidimensional and eventually parametric models.

- Diffusions \( dX_t = a(t, X_t)dt + b(t, X_t)dW_t \)

- Fractional Gaussian Noise, with \( H \) the Hurst parameter

\[
dX_t = a(t, X_t)dt + b(t, X_t)dW_t^H
\]
We consider here the three main classes of SDE’s which can be easily specified. All multidimensional and eventually parametric models.

- **Diffusions**
  \[dX_t = a(t, X_t)dt + b(t, X_t)dW_t\]

- **Fractional Gaussian Noise**, with \(H\) the Hurst parameter
  \[dX_t = a(t, X_t)dt + b(t, X_t)dW_t^H\]

- **Diffusions with jumps, Lévy**
  \[dX_t = a(X_t)dt + b(X_t)dW_t + \int_{|z|>1} c(X_t-, z)\mu(dt, dz) \]
  \[+ \int_{0<|z|\leq1} c(X_t-, z)\{\mu(dt, dz) - \nu(dz)dt}\]
\[ dX_t = -3X_t dt + \frac{1}{1+X_t^2} dW_t \]

\[
> \text{mod1} <- \text{setModel(drift = "-3*x", diffusion = "1/(1+x^2)")}
\]
\[ dX_t = -3X_t dt + \frac{1}{1+X_t^2} dW_t \]

> mod1 <- setModel(drift = "-3*x", diffusion = "1/(1+x^2)")
\[ dX_t = -3X_t \, dt + \frac{1}{1+X_t^2} \, dW_t \]

\>
> mod1 <- setModel(drift = "-3*x", diffusion = "1/(1+x^2)")

\>
> str(mod1)
Formal class 'yuima.model' [package "yuima"] with 16 slots
..@ drift : expression((-3 * x))
..@ diffusion : List of 1
...$ : expression(1/(1 + x^2))
..@ hurst : num 0.5
..@ jump.coef : expression()
..@ measure : list()
..@ measure.type : chr(0)
..@ parameter : Formal class 'model.parameter' [package "yuima"] with 6 slots
... ...@ all : chr(0)
... ...@ common : chr(0)
... ...@ diffusion : chr(0)
... ...@ drift : chr(0)
... ...@ jump : chr(0)
... ...@ measure : chr(0)
... @ state.variable : chr "x"
... @ jump.variable : chr(0)
... @ time.variable : chr "t"
... @ noise.number : num 1
... @ equation.number : int 1
... @ dimension : int [1:6] 0 0 0 0 0 0
... @ solve.variable : chr "x"
... @ xinit : num 0
... @ J.flag : logi FALSE
And we can easily simulate and plot the model like

```r
> set.seed(123)
> X <- simulate(mod1)
> plot(X)
```
$$dX_t = -3X_t dt + \frac{1}{1+X_t^2} dW_t$$

The `simulate` function fills the slots `data` and `sampling`

```r
> str(X)
```
\[ dX_t = -3X_t \, dt + \frac{1}{1+X_t^2} \, dW_t \]

The \texttt{simulate} function fills the slots \texttt{data} and \texttt{sampling}

> \texttt{str(X)}
\[ dX_t = -3X_t dt + \frac{1}{1+X_t^2} dW_t \]

The `simulate` function fills the slots `data` and `sampling`

> str(X)

```
Formal class ‘yuima’ [package “yuima”] with 5 slots
 ..@ data : Formal class ‘yuima.data’ [package “yuima”] with 2 slots
    .. ..@ original.data: ts [1:101, 1] 0 -0.217 -0.186 -0.308 -0.27 ...
    .. .. ..- attr(*, “dimnames”)=List of 2
    .. .. .. ..$ : NULL
    .. .. .. ..$: chr “Series 1”
    .. .. ..- attr(*, “tsp”)= num [1:3] 0 1 100
    .. .. ..@ zoo.data :List of 1
    .. .. ..$: Series 1:’zooreg’ series from 0 to 1
 ..@ model : Formal class ‘yuima.model’ [package “yuima”] with 16 slots
(... output dropped)
 ..@ sampling : Formal class ‘yuima.sampling’ [package “yuima”] with 11 slots
    .. ..@ Initial : num 0
    .. ..@ Terminal : num 1
    .. ..@ n : num 100
    .. ..@ delta : num 0.1
    .. ..@ grid : num(0)
    .. ..@ random : logi FALSE
    .. ..@ regular : logi TRUE
    .. ..@ sdelta : num(0)
    .. ..@ sgrid : num(0)
    .. ..@ oindex : num(0)
    .. ..@ interpolation: chr “none”
```
Parametric model: \[ dX_t = -\theta X_t \, dt + \frac{1}{1+X_t^\gamma} \, dW_t \]

> mod2 <- setModel(drift = "-theta*x", diffusion = "1/(1+x^gamma)")

Automatic extraction of the parameters for further inference

> str(mod2)
Formal class ‘yuima.model’ [package "yuima"] with 16 slots
  ..@ drift : expression((-theta * x))
  ..@ diffusion : List of 1
    .. ..$ : expression(1/(1 + x^gamma))
  ..@ hurst : num 0.5
  ..@ jump.coeff : expression()
  ..@ measure : list()
  ..@ measure.type : chr(0)
  ..@ parameter : Formal class ‘model.parameter’ [package "yuima"] with 6 slots
    .. ..@ all : chr [1:2] "theta" "gamma"
    .. ..@ common : chr(0)
    .. ..@ diffusion : chr "gamma"
    .. ..@ drift : chr "theta"
    .. ..@ jump : chr(0)
    .. ..@ measure : chr(0)
    .. ..@ state.variable : chr "x"
    .. ..@ jump.variable : chr(0)
    .. ..@ time.variable : chr "t"
    .. ..@ noise.number : num 1
    .. ..@ equation.number : int 1
    .. ..@ dimension : int [1:6] 2 0 1 1 0 0
    .. ..@ solve.variable : chr "x"
    .. ..@ xinit : num 0
    .. ..@ J.flag : logi FALSE
Parametric model: \( dX_t = -\theta X_t dt + \frac{1}{1+X_t^\gamma} dW_t \)

And this can be simulated specifying the parameters

```r
> simulate(mod2, true.param=list(theta=1, gamma=3))
```
2-dimensional diffusions with 3 noises

\[
\begin{align*}
\frac{dX_1}{dt} &= -3X_1 \, dt + dW_1^1 + X_2 \, dW_3^1 \\
\frac{dX_2}{dt} &= -(X_1^1 + 2X_2^2) \, dt + X_1^1 \, dW_1^1 + 3 \, dW_2^1
\end{align*}
\]

has to be organized into matrix form

\[
\begin{pmatrix}
\frac{dX_1}{dt} \\
\frac{dX_2}{dt}
\end{pmatrix}
= \begin{pmatrix}
-3X_1^1 \\
-X_1^1 - 2X_2^2
\end{pmatrix} \, dt +
\begin{pmatrix}
1 & 0 & X_2^1 \\
X_1^1 & 3 & 0
\end{pmatrix}
\begin{pmatrix}
dW_1^1 \\
dW_2^1 \\
dW_3^1
\end{pmatrix}
\]

\[
\text{sol} <- c("x1","x2") \quad \text{# variable for numerical solution}
\]
\[
\text{a} <- c("-3\times x1","-x1-2\times x2") \quad \text{# drift vector}
\]
\[
\text{b} <- \text{matrix}(c("1","x1","0","3","x2","0"),2,3) \quad \text{# diffusion matrix}
\]
\[
\text{mod3} <- \text{setModel}(\text{drift} = \text{a}, \text{diffusion} = \text{b}, \text{solve.variable} = \text{sol})
\]
2-dimensional diffusions with 3 noises

\[
\begin{align*}
dX_t^1 &= -3X_t^1 dt + dW_t^1 + X_t^2 dW_t^3 \\
dX_t^2 &= -(X_t^1 + 2X_t^2) dt + X_t^1 dW_t^1 + 3dW_t^2
\end{align*}
\]

> str(mod3)
Formal class 'yuima.model' [package "yuima"] with 16 slots
  ..@ drift : expression((-3 * x1), (-x1 - 2 * x2))
  ..@ diffusion : List of 2
    ..$ : expression(1, 0, x2)
    ..$ : expression(x1, 3, 0)
  ..@ hurst : num 0.5
  ..@ jump.coeff : expression()
  ..@ measure : list()
  ..@ measure.type : chr(0)
  ..@ parameter : Formal class 'model.parameter' [package "yuima"] with 6 slots
    .. .. ..@ all : chr(0)
    .. .. ..@ common : chr(0)
    .. .. ..@ diffusion: chr(0)
    .. .. ..@ drift : chr(0)
    .. .. ..@ jump : chr(0)
    .. .. ..@ measure : chr(0)
    .. ..@ state.variable : chr "x"
    .. ..@ jump.variable : chr(0)
    .. ..@ time.variable : chr "t"
    .. ..@ noise.number : int 3
    .. ..@ equation.number: int 2
    .. ..@ dimension : int [1:6] 0 0 0 0 0 0
    .. ..@ solve.variable : chr [1:2] "x1" "x2"
    .. ..@ xinit : num [1:2] 0 0
    .. ..@ J.flag : logi FALSE
Plot methods inherited by *zoo*

```r
set.seed(123)
X <- simulate(mod3)
plot(X, plot.type="single", col=c("red", "blue")
```

**Explorer Data Analysis**

- Estimation of Financial Models
- Likelihood approach
- Two stage least squares estimation
- Model selection
- Numerical Evidence
- Application to real data

**The change point problem**

**Overview of the *yuima* package**

- *yuima* object functionalities
- How to use it

**Option Pricing with *R***

S.M. Iacus 2011

R in Finance – 91 / 113
Also models likes this can be specified

\[
\begin{align*}
\text{d}X^1_t &= X^2_t |X^1_t|^{2/3} \text{d}W^1_t, \\
\text{d}X^2_t &= g(t) \text{d}X^3_t, \\
\text{d}X^3_t &= X^3_t (\mu \text{d}t + \sigma (\rho \text{d}W^1_t + \sqrt{1 - \rho^2} \text{d}W^2_t))
\end{align*}
\]

where \( g(t) = 0.4 + (0.1 + 0.2t)e^{-2t} \)

The above is an example of parametric SDE with more equations than noises.
Fractional Gaussian Noise \( dY_t = 3Y_t dt + dW^H_t \)

> mod4 <- setModel(drift=3\*y, diffusion=1, \text{hurst=0.3}, solve.var=y)
Fractional Gaussian Noise \( dY_t = 3Y_t dt + dW^H_t \)

> mod4 <- setModel(drift=3*y, diffusion=1, hurst=0.3, solve.var=y)

The hurst slot is filled

> str(mod4)

Formal class ‘yuima.model’ [package "yuima"] with 16 slots

..@ drift : expression((3 * y))
..@ diffusion :List of 1
...$ : expression(1)
..@ hurst : num 0.3
..@ jump.coeff : expression()
..@ measure : list()
..@ measure.type : chr(0)
..@ parameter :Formal class ‘model.parameter’ [package "yuima"] with 6 slots
... ...@ all : chr(0)
... ...@ common : chr(0)
... ...@ diffusion : chr(0)
... ...@ drift : chr(0)
... ...@ jump : chr(0)
... ...@ measure : chr(0)
...@ state.variable : chr "x"
...@ jump.variable : chr(0)
...@ time.variable : chr "t"
...@ noise.number : num 1
...@ equation.number: int 1
...@ dimension : int [1:6] 0 0 0 0 0 0
...@ solve.variable : chr "y"
...@ xinit : num 0
...@ J.flag : logi FALSE
Fractional Gaussian Noise \(dY_t = 3Y_t dt + dW_t^H\)

```r
> set.seed(123)
> X <- simulate(mod4, n=1000)
> plot(X, main="I'm fractional!"
```

I'm fractional!
Jump processes can be specified in different ways in mathematics (and hence in yuima package).

Let $Z_t$ be a Compound Poisson Process (i.e. jumps follow some distribution, e.g. Gaussian)

Then it is possible to consider the following SDE which involves jumps

$$dX_t = a(X_t)dt + b(X_t)dW_t + dZ_t$$

Next is an example of Poisson process with intensity $\lambda = 10$ and Gaussian jumps.

In this case we specify `measure.type` as “CP” (Compound Poisson)
Jump process: \[ dX_t = -\theta X_t dt + \sigma dW_t + Z_t \]

```r
> mod5 <- setModel(drift=c("-theta*x"), diffusion="sigma", 
jump.coeff="1", measure=list(intensity="10", df=list("dnorm(z, 0, 1)")
measure.type="CP", solve.variable="x")
> set.seed(123)
> X <- simulate(mod5, true.p=list(theta=1,sigma=3),n=1000)
> plot(X, main="I’m jumping!")
```

```
I'm jumping!
```

![Graph showing a jump process with a plot of simulated data](image)

- Explorative Data Analysis
- Estimation of Financial Models
- Likelihood approach
- Two stage least squares estimation
- Model selection
- Numerical Evidence
- Application to real data
- The change point problem
- Overview of the yuima package
  - yuima object
  - functionalities
  - how to use it
- Option Pricing with R
Another way is to specify the Lévy measure. Without going into too much details, here is an example of a simple OU process with IG Lévy measure
\[ dX_t = -X_t dt + dZ_t \]

```r
> mod6 <- setModel(drift="-x", xinit=1, jump.coeff="1", measure.type="code", measure=list(df="rIG(z, 1, 0.1)"))
> set.seed(123)
> plot( simulate(mod6, Terminal=10, n=10000), main="I’m also jumping!"")
```

I’m also jumping!
The `setModel` method

Models are specified via

```r
setModel(drift, diffusion, hurst = 0.5, jump.coeff, measure,
measure.type, state.variable = x, jump.variable = z, time.variable
= t, solve.variable, xinit) in
```

\[
dX_t = a(X_t)dt + b(X_t)dW_t + c(X_t)dZ_t
\]

The package implements many multivariate RNG to simulate Lévy paths including `rIG`, `rNIG`, `rbgamma`, `rngamma`, `rstable`.

Other user-defined or packages-defined RNG can be used freely.
The `setModel` method

Models are specified via

```r
setModel(drift, diffusion, hurst = 0.5, jump.coeff, measure,
measure.type, state.variable = x, jump.variable = z, time.variable
= t, solve.variable, xinit) in

\[dX_t = a(X_t)dt + b(X_t)dW_t + c(X_t)dZ_t\]
```

The package implements many multivariate RNG to simulate Lévy paths including `rIG`, `rNIG`, `rbgamma`, `rngamma`, `rstable`.

Other user-defined or packages-defined RNG can be used freely.
Models are specified via

\[
\text{setModel(drift, diffusion, } \text{hurst} = 0.5, \text{ jump.coeff, measure, measure.type, state.variable = x, jump.variable = z, time.variable = t, solve.variable, xinit)} \text{ in}
\]

\[
dX_t = a(X_t)dt + b(X_t)dW_t + c(X_t)dZ_t
\]

The package implements many multivariate RNG to simulate Lévy paths including \text{rIG, rNIG, rgbgamma, rngamma, rstable}.

Other user-defined or packages-defined RNG can be used freely.
Models are specified via

```
setModel(drift, diffusion, hurst = 0.5, jump.coeff, measure,
measure.type, state.variable = x, jump.variable = z, time.variable
= t, solve.variable, xinit)  in

\[ dX_t = a(X_t)dt + b(X_t)dW_t + c(X_t)dZ_t \]
```

The package implements many multivariate RNG to simulate Lévy paths
including `rIG`, `rNIG`, `rbgamma`, `rngamma`, `rstable`.

Other user-defined or packages-defined RNG can be used freely.
Models are specified via

```
setModel(drift, diffusion, hurst = 0.5, jump.coeff, measure, measure.type, state.variable = x, jump.variable = z, time.variable = t, solve.variable, xinit)
```

\[
dX_t = a(X_t)dt + b(X_t)dW_t + c(X_t)dZ_t
\]

The package implements many multivariate RNG to simulate Lévy paths including `rIG`, `rNIG`, `rbgamma`, `rngamma`, `rstable`. Other user-defined or packages-defined RNG can be used freely.
Option Pricing with R
There are several good packages on CRAN which implement the basic option pricing formulas. Just to mention a couple (with apologizes to the authors pf the other packages)

- Rmetrics suite which includes: fOptions, fAsianOptions, fExoticOptions

- RQuanLib for European, American and Asian option pricing

But if you need to go outside the standard Black & Scholes geometric Brownian Motion model, the only way remains Monte Carlo analysis. And for jump processes, maybe, yuima package is one of the current frameworks you an think to use.

In addition, yuima also offers asymptotic expansion formulas for general diffusion processes which can be used in Asian option pricing.
The \textit{yuima} package can handle asymptotic expansion of functionals of $d$-dimensional diffusion process

$$dX_t^\varepsilon = a(X_t^\varepsilon, \varepsilon)dt + b(X_t^\varepsilon, \varepsilon)dW_t, \quad \varepsilon \in (0, 1]$$

with $W_t$ and $r$-dimensional Wiener process, i.e. $W_t = (W_t^1, \ldots, W_t^r)$.

The functional is expressed in the following abstract form

$$F^\varepsilon(X_t) = \sum_{\alpha=0}^r \int_0^T f_{\alpha}(X_t^\varepsilon, d)dW_t^\alpha + F(X_t^\varepsilon, \varepsilon), \quad W_t^0 = t$$
Example: B&S asian call option

\[ dX^\varepsilon_t = \mu X^\varepsilon_t \, dt + \varepsilon X^\varepsilon_t \, dW_t \]

and the B&S price is related to \( E \left\{ \max \left( \frac{1}{T} \int_0^T X^\varepsilon_t \, dt - K, 0 \right) \right\} \). Thus the functional of interest is

\[ F^\varepsilon(X^\varepsilon_T) = \frac{1}{T} \int_0^T X^\varepsilon_t \, dt, \quad r = 1 \]
Example: B&S asian call option

\[
dX_t^\varepsilon = \mu X_t^\varepsilon dt + \varepsilon X_t^\varepsilon dW_t
\]

and the B&S price is related to \( \mathbb{E} \left\{ \max \left( \frac{1}{T} \int_0^T X_t^\varepsilon dt - K, 0 \right) \right\} \). Thus the functional of interest is

\[
F^\varepsilon(X_t^\varepsilon) = \frac{1}{T} \int_0^T X_t^\varepsilon dt, \quad r = 1
\]

with

\[
f_0(x, \varepsilon) = \frac{x}{T}, \quad f_1(x, \varepsilon) = 0, \quad F(x, \varepsilon) = 0
\]

in

\[
F^\varepsilon(X_t^\varepsilon) = \sum_{\alpha=0}^{r} \int_0^T f_{\alpha}(X_t^\varepsilon, d) dW_t^\alpha + F(X_t^\varepsilon, \varepsilon)
\]
Estimation of functionals. Example.

So, the call option price requires the composition of a smooth functional

\[ F^\varepsilon(X_t^\varepsilon) = \frac{1}{T} \int_0^T X_t^\varepsilon \, dt, \quad r = 1 \]

with the irregular function

\[ \max(x - K, 0) \]

Monte Carlo methods require a HUGE number of simulations to get the desired accuracy of the calculation of the price, while asymptotic expansion of \( F^\varepsilon \) provides unexpectedly accurate approximations.

The \texttt{yuima} package provides functions to construct the functional \( F^\varepsilon \), and automatic asymptotic expansion based on Malliavin calculus starting from a \texttt{yuima} object.
> diff.matrix <- matrix(c("x\*e"), 1, 1)
> model <- setModel(drift = c("x"), diffusion = diff.matrix)
> T <- 1
> xinit <- 1
> f <- list(expression(x/T), expression(0))
> F <- 0
> e <- .3
> yuima <- setYuima(model = model, sampling = setSampling(Terminal=T, n=1000))
> yuima <- setFunctional( yuima, f=f, F=F, xinit=xinit, e=e)
```r
> diff.matrix <- matrix(c("x*e"), 1,1)
> model <- setModel(drift = c("x"), diffusion = diff.matrix)
> T <- 1
> xinit <- 1
> f <- list(expression(x/T), expression(0))
> F <- 0
> e <- .3
> yuima <- setYuima(model = model, sampling = setSampling(Terminal=T, n=1000))
> yuima <- setFunctional( yuima, f=f, F=F, xinit=xinit, e=e)
```
> diff.matrix <- matrix( c("x*e"), 1,1)
> model <- setModel(drift = c("x"), diffusion = diff.matrix)
> T <- 1
> xinit <- 1
> f <- list( expression(x/T), expression(0))
> F <- 0
> e <- .3
> yuima <- setYuima(model = model, sampling = setSampling(Terminal=T, n=1000))
> yuima <- setFunctional( yuima, f=f,F=F, xinit=xinit, e=e)

the definition of the functional is now included in the yuima object
(some output dropped)

> str(yuima)
Formal class ’yuima’ [package "yuima"] with 5 slots
 .. @ data :Formal class ’yuima.data’ [package "yuima"] with 2 slots
 .. @ model :Formal class ’yuima.model’ [package "yuima"] with 16 slots
 .. @ sampling :Formal class ’yuima.sampling’ [package "yuima"] with 11 slots
 .. @ functional :Formal class ’yuima.functional’ [package "yuima"] with 4 slots
 ... ...@ F : num 0
 ... ...@ f :List of 2
 ... ... ...$ : expression(x/T)
 ... ... ...$ : expression(0)
 ... ...@ xinit: num 1
 ... ...@ e : num 0.3
Then, it is as easy as

\begin{verbatim}
> F0 <- F0(yuima)
> F0
[1] 1.716424
> max(F0-K,0) # asian call option price
[1] 0.7164237
\end{verbatim}
Then, it is as easy as

```r
> F0 <- F0(yuima)
> F0
[1] 1.716424
> max(F0-K,0)  # asian call option price
[1] 0.7164237
```

and back to asymptotic expansion, the following script may work

```r
> rho <- expression(0)
> get_ge <- function(x,epsilon,K,F0){
+   tmp <- (F0 - K) + (epsilon * x)
+   tmp[(epsilon * x) < (K-F0)] <- 0
+   return( tmp )
+ }
> K <- 1  # strike
> epsilon <- e  # noise level
> g <- function(x) {
+   tmp <- (F0 - K) + (epsilon * x)
+   tmp[(epsilon * x) < (K-F0)] <- 0
+   tmp
+ }
```
The expansion of previous functional gives

```r
> asymp <- asymptotic_term(yuima, block=10, rho, g)
calculating d0 ...done
calculating d1 term ...done
> asymp$d0 + e * asymp$d1  # asymp. exp. of asian call price
[1] 0.7148786
```

```r
> library(fExoticOptions)  # From RMetrics suite
> TurnbullWakemanAsianApproxOption("c", S = 1, SA = 1, X = 1,
+   Time = 1, time = 1, tau = 0.0, r = 0, b = 1, sigma = e)
Option Price:
[1] 0.7184944
```

```r
> LevyAsianApproxOption("c", S = 1, SA = 1, X = 1,
+   Time = 1, time = 1, r = 0, b = 1, sigma = e)
Option Price:
[1] 0.7184944
```

```r
> X <- sde.sim(drift=expression(x), sigma=expression(e*x), N=1000,M=1000)
> mean(colMeans((X-K) * (X-K>0)))  # MC asian call price based on M=1000 repl.
[1] 0.707046
```
Asymptotic expansion is now also available for multidimensional diffusion processes like the Heston model

\[ \begin{align*}
\, dX^1_t,\varepsilon & = aX^1_t,\varepsilon \, dt + \varepsilon X^1_t,\varepsilon \sqrt{X^2_t,\varepsilon} \, dW^1_t \\
\, dX^2_t,\varepsilon & = c(d - X^2_t,\varepsilon) \, dt + \varepsilon \sqrt{X^2_t,\varepsilon} \left( \rho dW^1_t + \sqrt{1 - \rho^2} dW^2_t \right)
\end{align*} \]

i.e. functionals of the form \( F(X^1,\varepsilon, X^2,\varepsilon) \).