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

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.

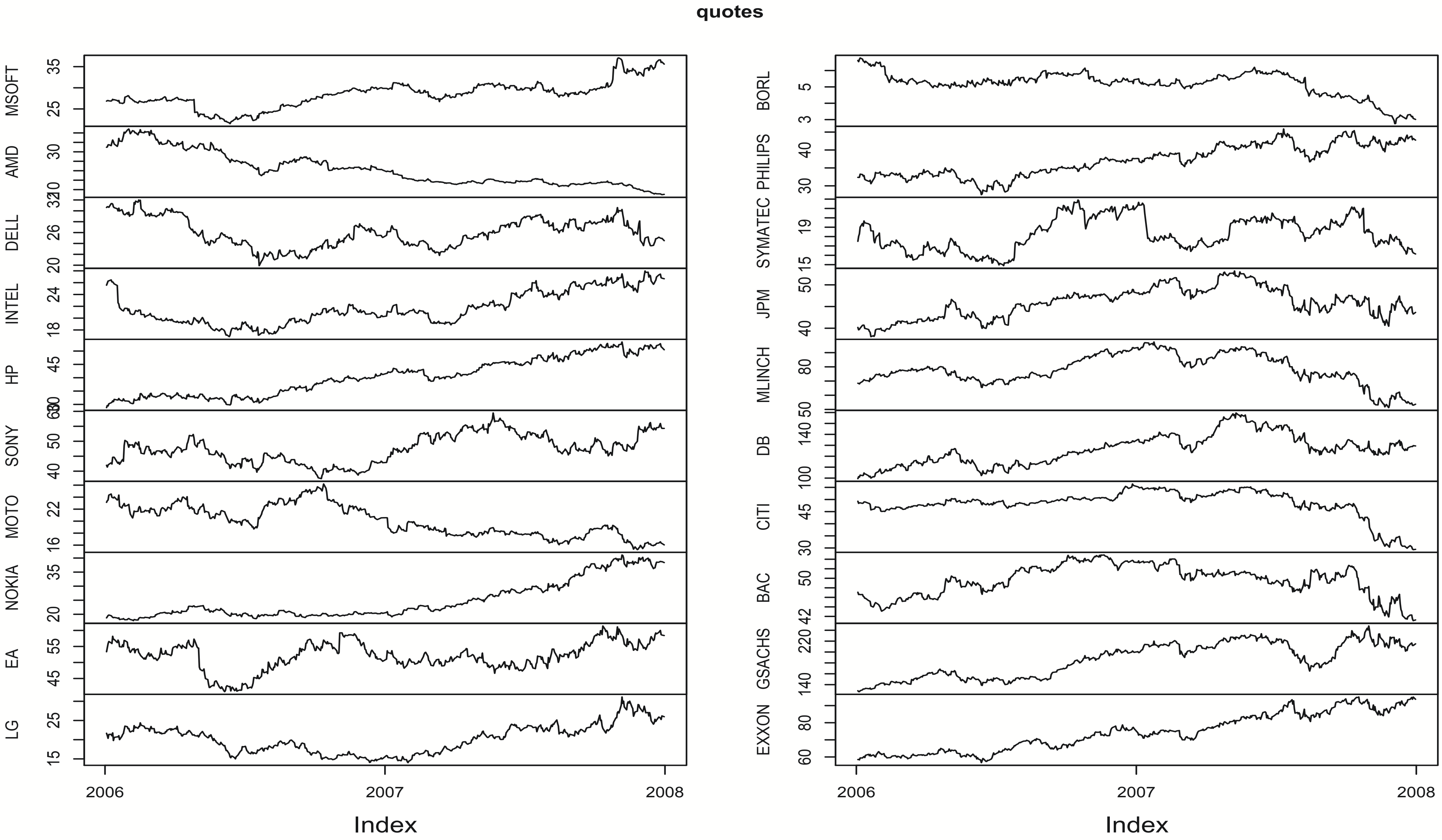
Real Data from NYSE

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?



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

$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \sigma(X_t)\mathrm{d}W_t$$

therefore, the discretized observations X_i form a Markov process and all the mathematical properties are embodied in the so-called *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_Δ depends on the transition density between X_i and X_{i-1}, so we put explicitly the dependence on Δ in the notation.

Luckily, there is no need to deal with the transition density, we can estimate P_Δ directly and easily.

Let X and Y be two time series, then

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

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

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

The terms $(\hat{P}_\Delta)_{j,k}$ are approximations of the scalar product $< P_\Delta \phi_j, \phi_k >_{b,\sigma}$ where b and σ 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 σ_i, $i = 1, \ldots, 4$ presented in the following table

	$\sigma_1(x)$	$\sigma_2(x)$	$\sigma_3(x)$	$\sigma_4(x)$
$b_1(x)$	X10, X1		X5	
$b_2(x)$		X2,X3	X4	
$b_3(x)$		X6, X7		
$b_4(x)$				X8

where

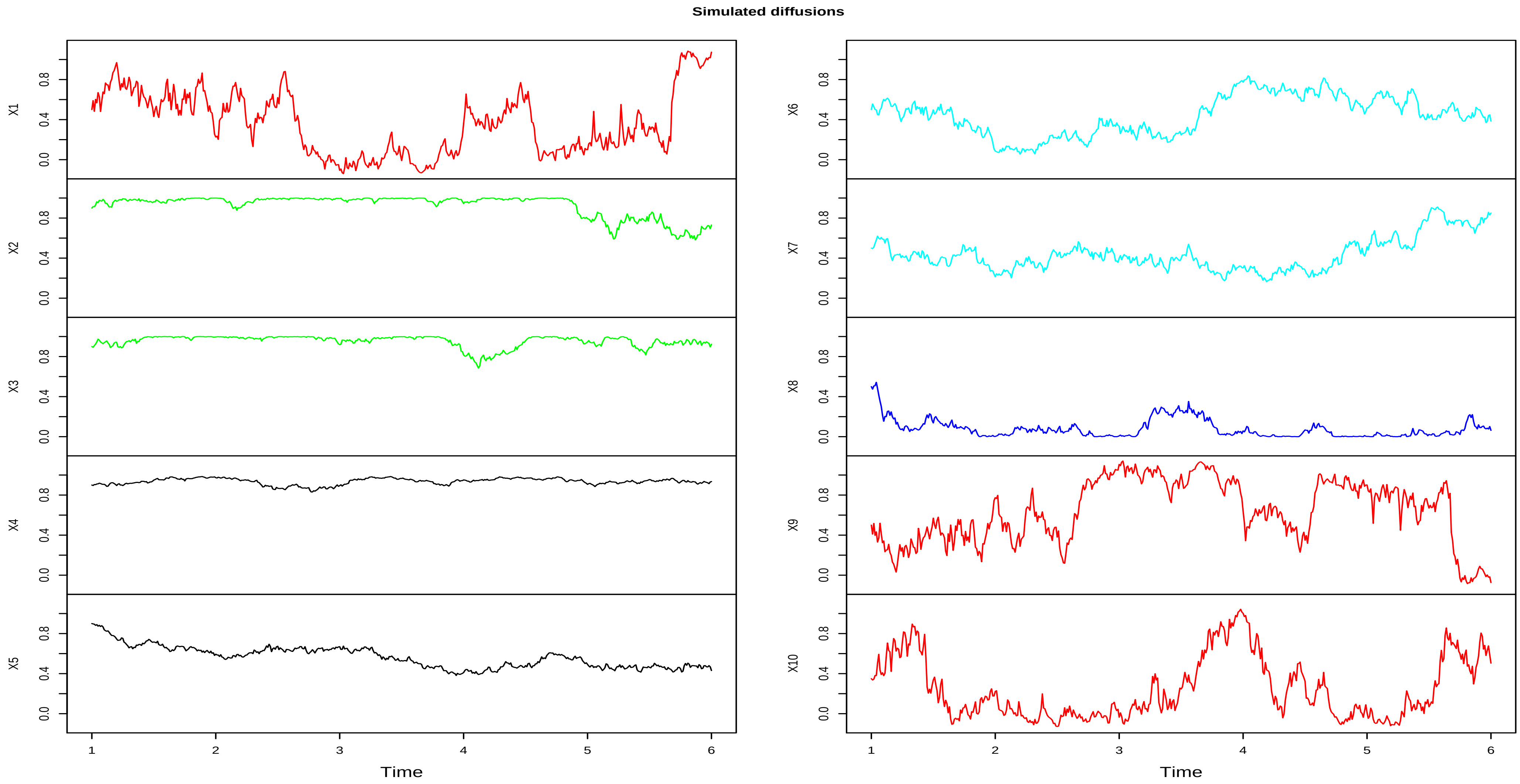
$$b_1(x) = 1 - 2x, \quad b_2(x) = 1.5(0.9 - x), \quad b_3(x) = 1.5(0.5 - x), \quad b_4(x) = 5(0.05 - x)$$

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

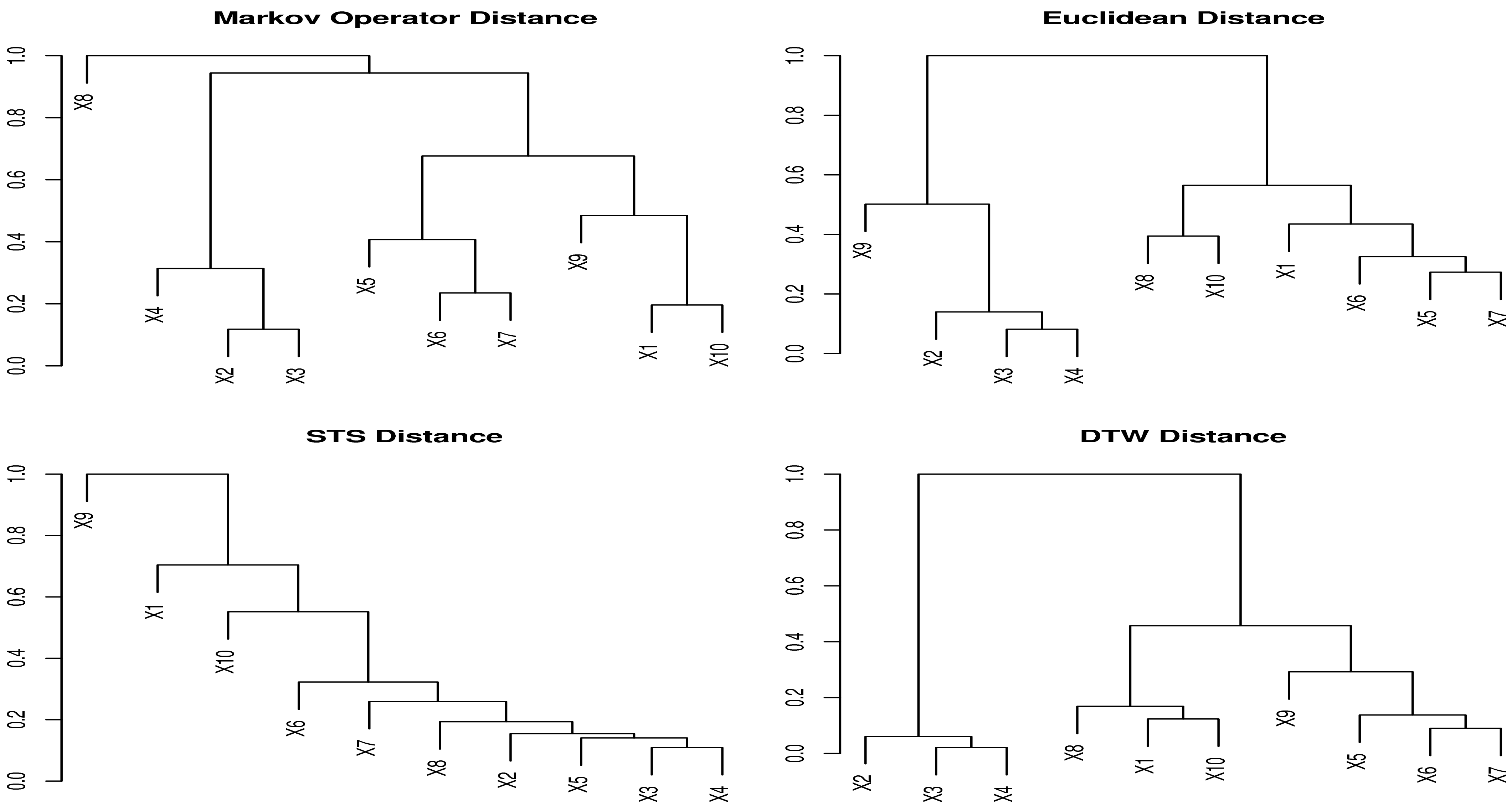
$$\sigma_3(x) = \sqrt{0.1x(1 - x)}, \quad \sigma_4(x) = \sqrt{0.8x(1 - x)}$$

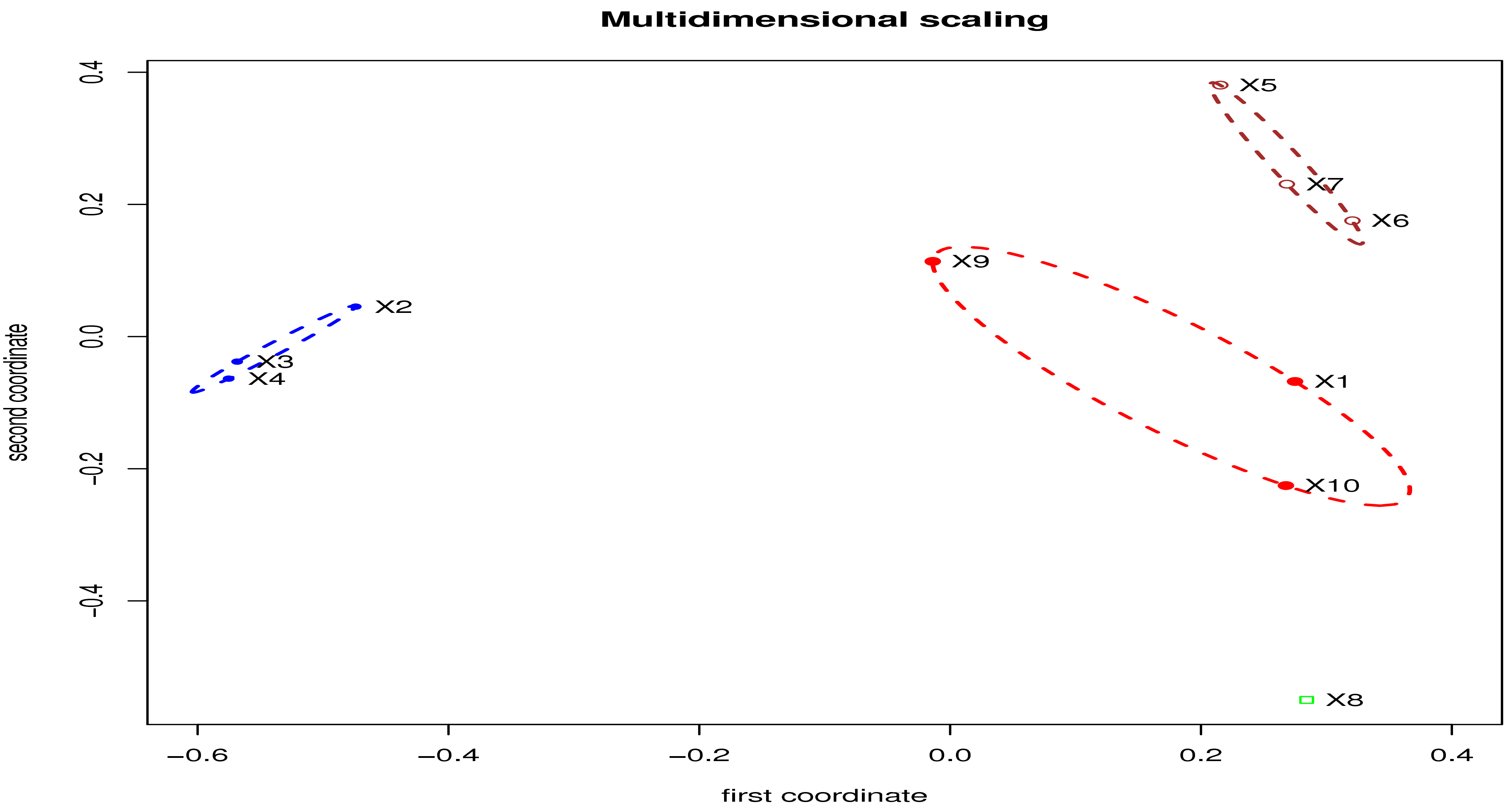
The process X9=1-X1, hence it has drift $-b_1(x)$ and the same quadratic variation of X1 and X10.

Trajectories



Dendrograms




Multidimensional scaling
X5
X7
X6
X9
X2
X3
X4
X1
X10
X8
0.4
0.2
0.0
−0.2
−0.4
second coordinate
−0.6
−0.4
−0.2
0.0
0.2
0.4
first coordinate

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

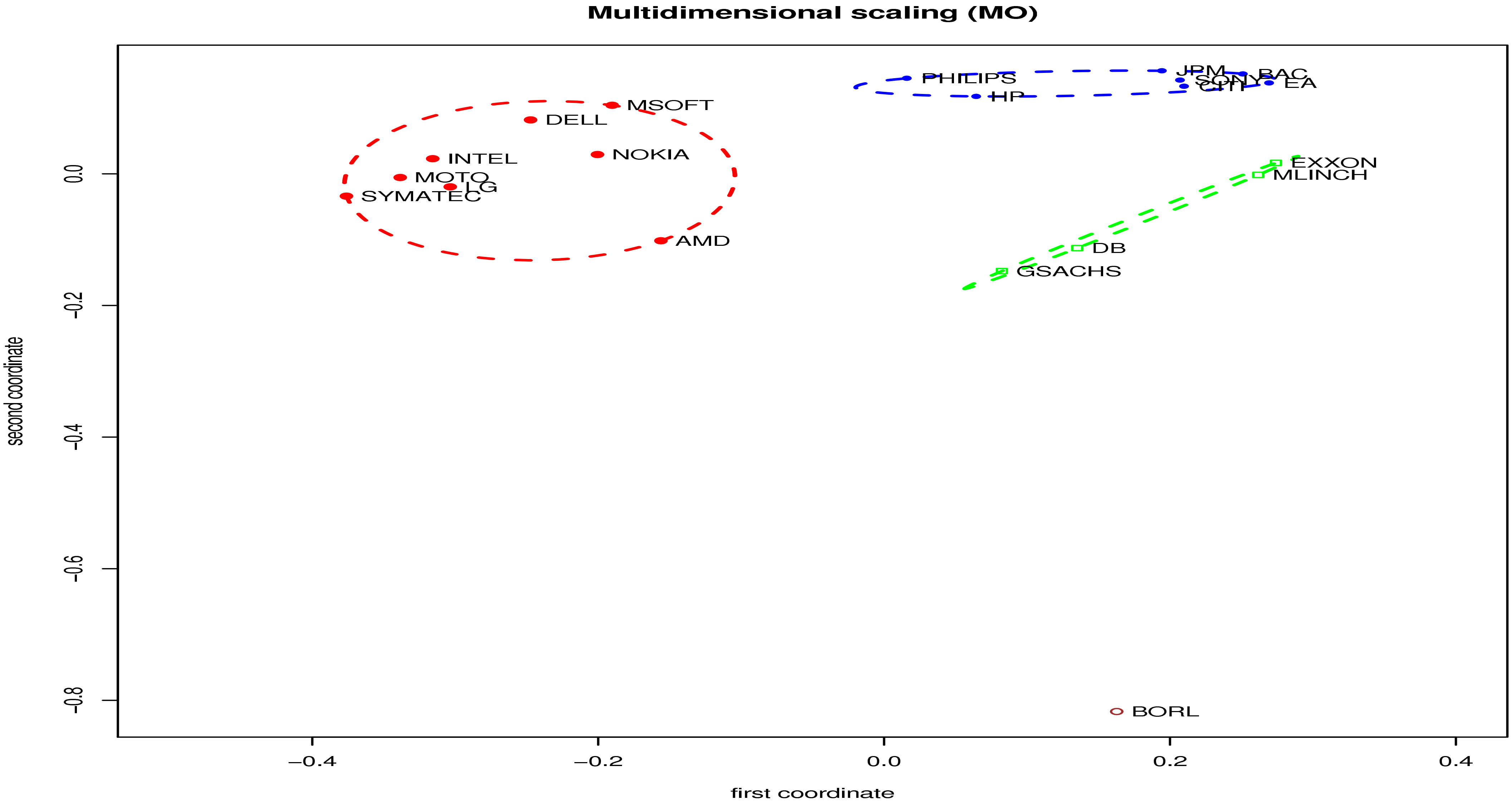
It contains the function `MOdist` which calculates the Markov Operator distance and returns a `dist` object.

```
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

We focus on the general approach to statistical inference for the parameter θ of models in the wide class of diffusion processes solutions to stochastic differential equations

$$\mathrm{d}X_t = b(\theta, X_t)\mathrm{d}t + \sigma(\theta, X_t)\mathrm{d}W_t$$

with initial condition X_0 and θ the p-dimensional parameter of interest.

- geometric Brownian motion (gBm)

$$\mathrm{d}X_t = \mu X_t \mathrm{d}t + \sigma X_t \mathrm{d}W_t$$

- Cox-Ingersoll-Ross (CIR)

$$\mathrm{d}X_t = (\theta_1 + \theta_2 X_t)\mathrm{d}t + \theta_3 \sqrt{X_t}\mathrm{d}W_t$$

- Chan-Karolyi-Longstaff-Sanders (CKLS)

$$\mathrm{d}X_t = (\theta_1 + \theta_2 X_t)\mathrm{d}t + \theta_3 X_t^{\theta_4}\mathrm{d}W_t$$

- nonlinear mean reversion (Aït-Sahalia)

$$\mathrm{d}X_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)\mathrm{d}t + \beta_1 X_t^\rho \mathrm{d}W_t$$

- double Well potential (bimodal behaviour, highly nonlinear)

$$\mathrm{d}X_t = (X_t - X_t^3)\mathrm{d}t + \mathrm{d}W_t$$

- Jacobi diffusion (political polarization)

$$\mathrm{d}X_t = -\theta\left(X_t - \frac{1}{2}\right)\mathrm{d}t + \sqrt{\theta X_t(1-X_t)}\mathrm{d}W_t$$

- 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

Likelihood in discrete time

Consider the model

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

with initial condition X_0 and θ the p-dimensional parameter of interest. By Markov property of diffusion processes, the likelihood has this form

$$L_n(\theta) = \prod_{i=1}^{n} p_\theta\left(\Delta, X_i | X_{i-1}\right) p_\theta(X_0)$$

and the log-likelihood is

$$\ell_n(\theta) = \log L_n(\theta) = \sum_{i=1}^{n} \log p_\theta\left(\Delta, X_i | X_{i-1}\right) + \log(p_\theta(X_0)) \, .$$

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

As usual, in most cases we don't know the conditional distribution $p_\theta\left(\Delta, X_i | X_{i-1}\right)$

Maximum Likelihood Estimators (MLE)

If we study the likelihood $L_n(\theta)$ as a function of θ 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 θ.

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

$$\begin{aligned}\hat{\theta}_n &= \arg\max_{\theta\in\Theta} L_n(\theta) \\ &= \arg\max_{\theta\in\Theta} L_n(\theta|X_1, X_2, \ldots, X_n)\end{aligned}$$

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 θ where $\ell(\theta) = \log L(\theta)$.

QUASI - MLE

Consider the multidimensional diffusion process

$$\mathrm{d}X_t = b(\theta_2, X_t)\mathrm{d}t + \sigma(\theta_1, X_t)\mathrm{d}W_t$$

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

$$\ell_n(\mathbf{X}_n, \theta) = -\frac{1}{2}\sum_{i=1}^{n}\left\{\log \det(\Sigma_{i-1}(\theta_1)) + \frac{1}{\Delta_n}(\Delta X_i - \Delta_n b_{i-1}(\theta_2))^T \Sigma_{i-1}^{-1}(\theta_1)(\Delta X_i - \Delta_n b_{i-1}(\theta_2))\right\} \tag{3}$$

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 θ is

$$\tilde{\theta}_n = \arg\min_{\theta} \ell_n(\mathbf{X}_n, \theta)$$

Consider the matrix

$$\varphi(n) = \begin{pmatrix} \frac{1}{n\Delta_n}\mathbf{I}_p & 0 \\ 0 & \frac{1}{n}\mathbf{I}_q \end{pmatrix}$$

where $\mathbf{I}_p$ and $\mathbf{I}_q$ 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

$$\mathrm{d}X_t = -\theta_2 X_t \mathrm{d}t + \theta_1 \mathrm{d}W_t$$

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))
```

QUASI - MLE. True values: $\theta_1 = 0.3$, $\theta_2 = 0.1$

We can now try to estimate the parameters

```
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
0.30766981 0.05007788
R> summary(mle1)
Maximum likelihood estimation

Coefficients:
         Estimate Std. Error
theta1 0.30766981 0.02629925
theta2 0.05007788 0.15144393

-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.

$$\mathrm{d}X_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)\mathrm{d}t + \beta_1 X_t^{\rho}\mathrm{d}B_t$$

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.

$$\mathrm{d}X_t = (\alpha_{-1} X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)\mathrm{d}t + \beta_1 X_t^{\rho} \mathrm{d}B_t$$

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

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

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

$$\mathrm{d}X_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)\mathrm{d}t + \beta_1 X_t^\rho \mathrm{d}B_t$$

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

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

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

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

Aït-Sahalia's interest rates model

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

$$\mathrm{d}X_t = (\alpha_{-1}X_t^{-1} + \alpha_0 + \alpha_1 X_t + \alpha_2 X_t^2)\mathrm{d}t + \beta_1 X_t^{\rho}\mathrm{d}B_t$$

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

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

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

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

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

Example of 2SLS estimation

Stage one: regression

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

```
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

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

```
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

Idea

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)

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

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

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

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

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

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

$$\mathrm{d}X_t = -\alpha_1(X_t - \alpha_2)\mathrm{d}t + \beta_1\sqrt{X_t}\mathrm{d}W_t \qquad \text{(true model)},$$

$$\mathrm{d}X_t = -\alpha_1(X_t - \alpha_2)\mathrm{d}t + \sqrt{\beta_1 + \beta_2 X_t}\mathrm{d}W_t \qquad \text{(competing model 1)},$$

$$\mathrm{d}X_t = -\alpha_1(X_t - \alpha_2)\mathrm{d}t + (\beta_1 + \beta_2 X_t)^{\beta_3}\mathrm{d}W_t \qquad \text{(competing model 2)},$$

We call the above models Mod1, Mod2 and Mod3.

We generate data from Mod1 with parameters

$$\mathrm{d}X_t = -(X_t - 10)\mathrm{d}t + 2\sqrt{X_t}\mathrm{d}W_t\,,$$

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

$$\mathrm{d}X_t = -(X_t - 10)\mathrm{d}t + 2\sqrt{X_t}\mathrm{d}W_t \qquad \text{(true model)},$$

$$\mathrm{d}X_t = -\alpha_1(X_t - \alpha_2)\mathrm{d}t + \beta_1\sqrt{X_t}\mathrm{d}W_t \qquad \text{(Model 1)}$$

$$\mathrm{d}X_t = -\alpha_1(X_t - \alpha_2)\mathrm{d}t + \sqrt{\beta_1 + \beta_2 X_t}\mathrm{d}W_t \qquad \text{(Model 2)}$$

$$\mathrm{d}X_t = -\alpha_1(X_t - \alpha_2)\mathrm{d}t + (\beta_1 + \beta_2 X_t)^{\beta_3}\mathrm{d}W_t \qquad \text{(Model 3)}$$

Model selection via AIC

Model 1 (true)	Model 2	Model 3
99.2 %	0.6 %	0.2 %

QMLE estimates under the different models

	α_1	α_2	β_1	β_2	β_3
Model 1	1.10	8.05	0.90		
Model 2	1.12	8.07	2.02	0.54	
Model 3	1.12	9.03	7.06	7.26	0.61

LASSO estimation

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 θ_i are estimated as zeros.

The SDE model

Let X_t be a diffusion process solution to

$$\mathrm{d}X_t = b(\alpha, X_t)\mathrm{d}t + \sigma(\beta, X_t)\mathrm{d}W_t$$

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

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

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

We assume that the functions b and σ are known up to α and β.

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 $\mathbb{H}_n(\mathbf{X}_n, \theta) = -\ell_n(\theta)$ from the local Gaussian approximation.

Quasi-likelihood function

The classical adaptive LASSO objective function for the present model is then

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

$\lambda_{n,j}$ and $\gamma_{n,k}$ are appropriate sequences representing an adaptive amount of shrinkage for each element of α and β.

Idea of Quadratic Approximation

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

$$
\begin{aligned}
\mathbb{H}_n(\mathbf{X}_n, \theta) &= \mathbb{H}_n(\mathbf{X}_n, \tilde{\theta}_n) + (\theta - \tilde{\theta}_n)'\dot{\mathbb{H}}_n(\mathbf{X}_n, \tilde{\theta}_n) + \frac{1}{2}(\theta - \tilde{\theta}_n)'\ddot{\mathbb{H}}_n(\mathbf{X}_n, \tilde{\theta}_n)(\theta - \tilde{\theta}_n) \\
&\quad + o_p(1) \\
&= \mathbb{H}_n(\mathbf{X}_n, \tilde{\theta}_n) + \frac{1}{2}(\theta - \tilde{\theta}_n)'\ddot{\mathbb{H}}_n(\mathbf{X}_n, \tilde{\theta}_n)(\theta - \tilde{\theta}_n) + o_p(1)
\end{aligned}
$$

with $\dot{\mathbb{H}}_n$ and $\ddot{\mathbb{H}}_n$ the gradient and Hessian of $\mathbb{H}_n$ with respect to θ.

The Adaptive LASSO estimator

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

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

The lasso method is implemented in the `yuima` package.

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

$$\lambda_{n,j} = \lambda_0 |\tilde{\alpha}_{n,j}|^{-\delta_1}, \qquad \gamma_{n,k} = \gamma_0 |\tilde{\beta}_{n,j}|^{-\delta_2} \tag{4}$$

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

Numerical Evidence

We consider this two dimensional geometric Brownian motion process solution to the stochastic differential equation

$$\begin{pmatrix} \mathrm{d}X_t \\ \mathrm{d}Y_t \end{pmatrix} = \begin{pmatrix} 1 - \mu_{11}X_t + \mu_{12}Y_t \\ 2 + \mu_{21}X_t - \mu_{22}Y_t \end{pmatrix} \mathrm{d}t + \begin{pmatrix} \sigma_{11}X_t & -\sigma_{12}Y_t \\ \sigma_{21}X_t & \sigma_{22}Y_t \end{pmatrix} \begin{pmatrix} \mathrm{d}W_t \\ \mathrm{d}B_t \end{pmatrix}$$

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.

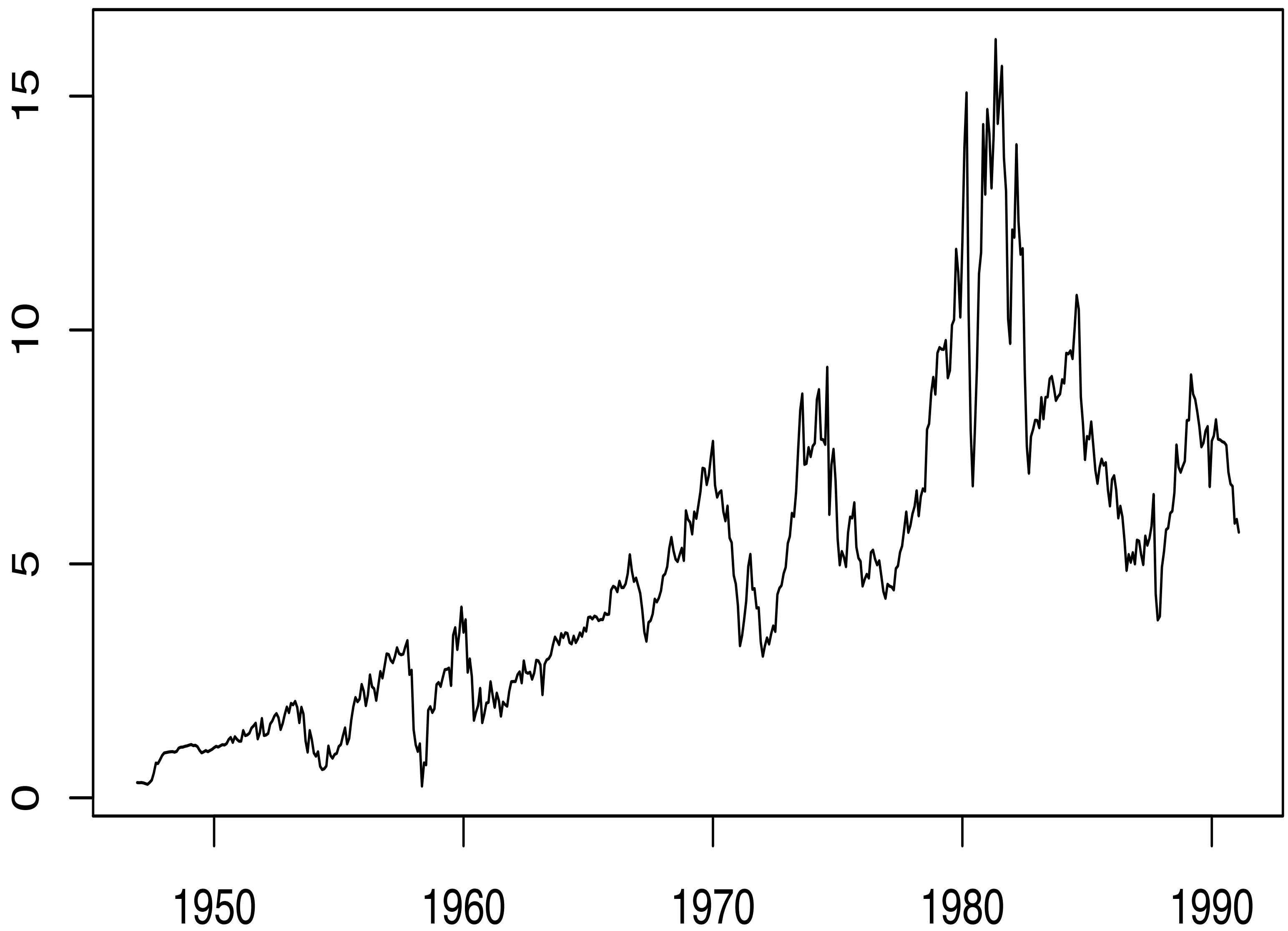
This model is a classical model for pricing of basket options in mathematical finance.

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

	μ_{11}	μ_{12}	μ_{21}	μ_{22}	σ_{11}	σ_{12}	σ_{21}	σ_{22}
True	0.9	0.0	0.0	0.7	0.3	0.0	0.0	0.2
Qmle: $n = 100$	0.96	0.05	0.25	0.81	0.30	0.04	0.01	0.20
	(0.08)	(0.06)	(0.27)	(0.15)	(0.03)	(0.05)	(0.02)	(0.02)
Lasso: $\lambda_0 = \gamma_0 = 1, n = 100$	0.86	0.00	0.05	0.71	0.30	0.02	0.01	0.20
	(0.12)	(0.00)	(0.13)	(0.09)	(0.03)	(0.05)	(0.02)	(0.02)
% of times $\theta_i = 0$	0.0	99.9	80.2	0.0	0.3	67.2	66.7	0.1
Lasso: $\lambda_0 = \gamma_0 = 5, n = 100$	0.82	0.00	0.00	0.66	0.29	0.01	0.00	0.20
	(0.12)	(0.00)	(0.00)	(0.09)	(0.03)	(0.03)	(0.01)	(0.02)
% of times $\theta_i = 0$	0.0	100.0	99.9	0.0	0.4	86.9	89.7	0.2
Qmle: $n = 1000$	0.95	0.03	0.21	0.79	0.30	0.04	0.01	0.20
	(0.07)	(0.04)	(0.25)	(0.13)	(0.03)	(0.06)	(0.02)	(0.02)
Lasso: $\lambda_0 = \gamma_0 = 1, n = 1000$	0.88	0.00	0.08	0.73	0.30	0.02	0.01	0.20
	(0.08)	(0.00)	(0.16)	(0.09)	(0.03)	(0.05)	(0.01)	(0.02)
% of times $\theta_i = 0$	0.0	99.7	72.1	0.0	0.1	67.5	66.6	0.1
Lasso: $\lambda_0 = \gamma_0 = 5, n = 1000$	0.86	0.00	0.00	0.68	0.29	0.01	0.00	0.20
	(0.09)	(0.00)	(0.01)	(0.06)	(0.03)	(0.04)	(0.01)	(0.02)
% of times $\theta_i = 0$	0.0	100.0	99.4	0.0	0.2	87.8	89.9	0.2

Application to real data

Interest rates LASSO estimation examples



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 author including Nowman (1997), Aït-Sahalia (1996), Yu and Phillips (2001) and it is a nice application of LASSO.

Reference	Model	α	β	γ
Merton (1973)	$\mathrm{d}X_t = \alpha \mathrm{d}t + \sigma \mathrm{d}W_t$		0	0
Vasicek (1977)	$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma \mathrm{d}W_t$			0
Cox, Ingersoll and Ross (1985)	$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma \sqrt{X_t}\mathrm{d}W_t$			$1/2$
Dothan (1978)	$\mathrm{d}X_t = \sigma X_t \mathrm{d}W_t$	0	0	1
Geometric Brownian Motion	$\mathrm{d}X_t = \beta X_t \mathrm{d}t + \sigma X_t \mathrm{d}W_t$	0		1
Brennan and Schwartz (1980)	$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma X_t \mathrm{d}W_t$			1
Cox, Ingersoll and Ross (1980)	$\mathrm{d}X_t = \sigma X_t^{3/2} \mathrm{d}W_t$	0	0	$3/2$
Constant Elasticity Variance	$\mathrm{d}X_t = \beta X_t \mathrm{d}t + \sigma X_t^{\gamma} \mathrm{d}W_t$	0		
CKLS (1992)	$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma X_t^{\gamma} \mathrm{d}W_t$			

Interest rates LASSO estimation examples

Model	Estimation Method	α	β	σ	γ
Vasicek	MLE	4.1889	-0.6072	0.8096	–
CKLS	Nowman	2.4272	-0.3277	0.1741	1.3610
CKLS	Exact Gaussian (Yu & Phillips)	2.0069 (0.5216)	-0.3330 (0.0677)	0.1741	1.3610
CKLS	QMLE	2.0822 (0.9635)	-0.2756 (0.1895)	0.1322 (0.0253)	1.4392 (0.1018)
CKLS	QMLE + LASSO with mild penalization	1.5435 (0.6813)	-0.1687 (0.1340)	0.1306 (0.0179)	1.4452 (0.0720)
CKLS	QMLE + LASSO with strong penalization	0.5412 (0.2076)	0.0001 (0.0054)	0.1178 (0.0179)	1.4944 (0.0720)

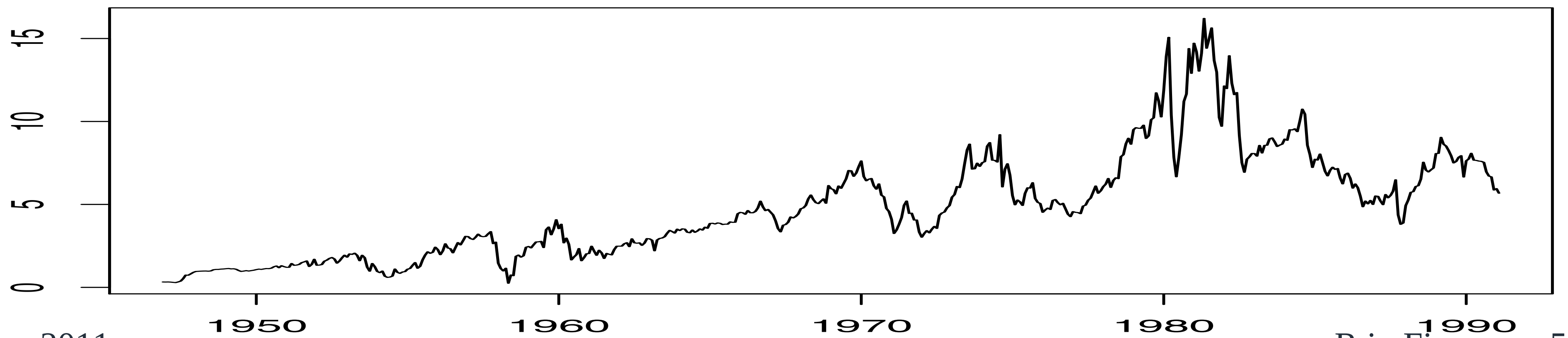
LASSO selected: Cox, Ingersoll and Ross (1980) model

$$\mathrm{d}X_t = \frac{1}{2}\mathrm{d}t + 0.12 \cdot X_t^{3/2}\mathrm{d}W_t$$

An example of Lasso estimation using `yuima` package. We make use of real data with CKLS model

$$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma X_t^{\gamma}\mathrm{d}W_t$$

```
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> 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
```

$$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma X_t^{\gamma}\mathrm{d}W_t$$

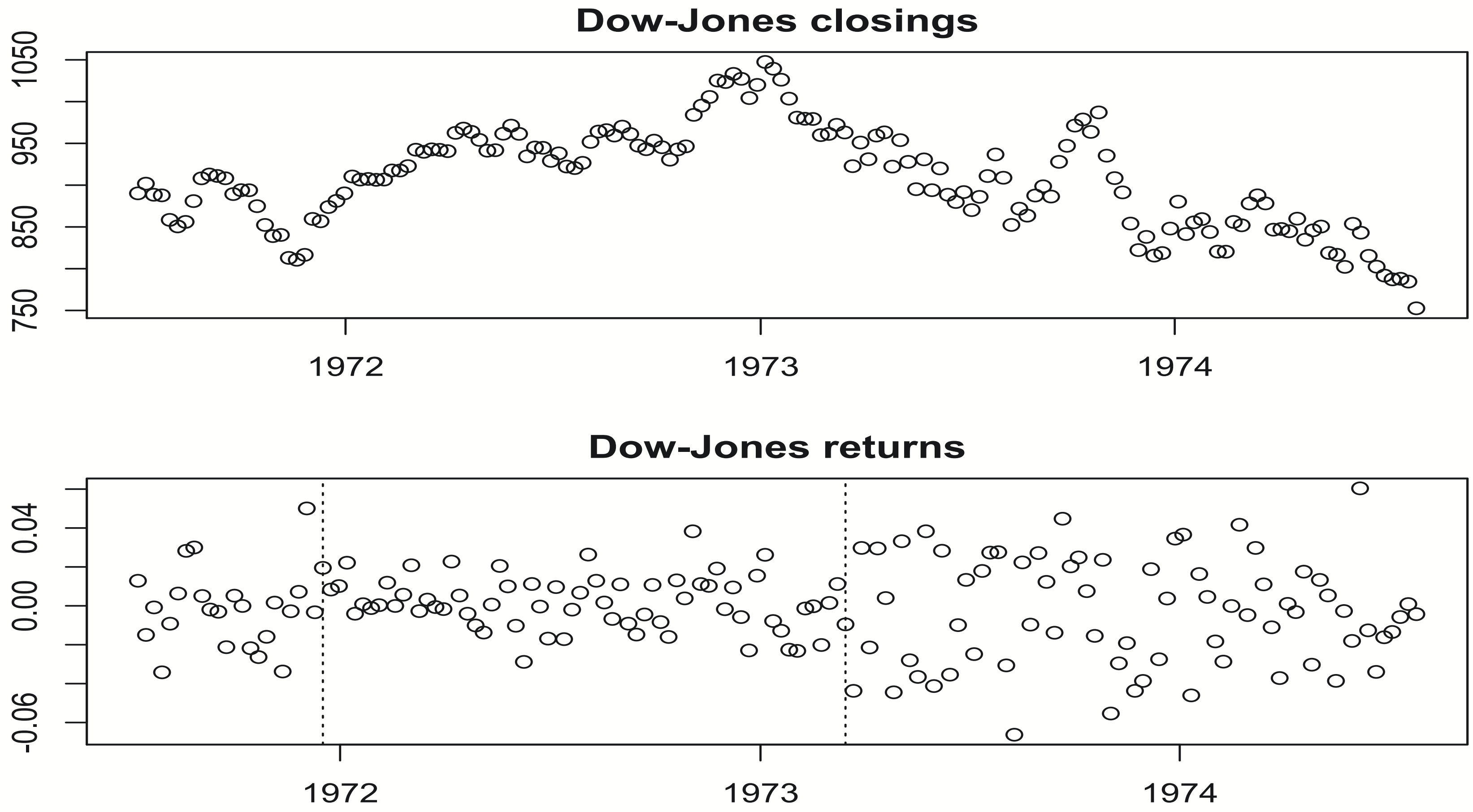
$$\mathrm{d}X_t = 0.6\mathrm{d}t + 0.12 X_t^{\frac{3}{2}}\mathrm{d}W_t$$

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

$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \sqrt{\theta}\sigma(X_t)\mathrm{d}W_t, \quad 0 \le t \le T, X_0 = x_0,$$

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

$$X_t = \begin{cases} X_0 + \int_0^t b(X_s)\mathrm{d}s + \sqrt{\theta_1}\int_0^t \sigma(X_s)\mathrm{d}W_s, & 0 \le t \le \tau^* \\ X_{\tau^*} + \int_{\tau^*}^t b(X_s)\mathrm{d}s + \sqrt{\theta_2}\int_{\tau^*}^t \sigma(X_s)\mathrm{d}W_s, & \tau^* < t \le T \end{cases}$$

where $\tau^* \in (0, T)$ is the change point and θ_1, θ_2 two parameters to be estimated.

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

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

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

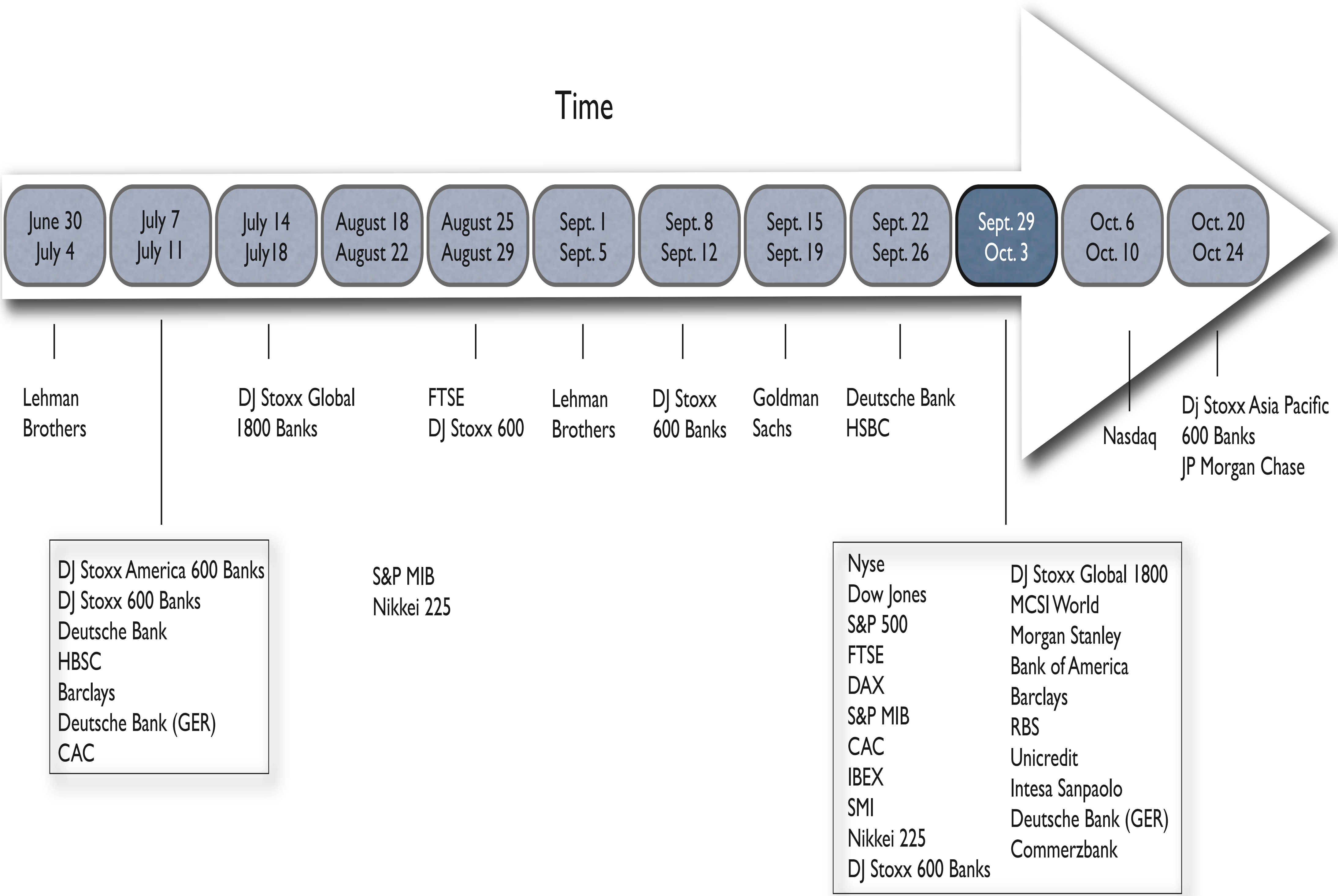
$$\hat{k}_0 = \arg\max_k \left| \frac{k}{n} - \frac{S_k}{S_n} \right|, \qquad S_k = \sum_{i=1}^k Z_i^2.$$

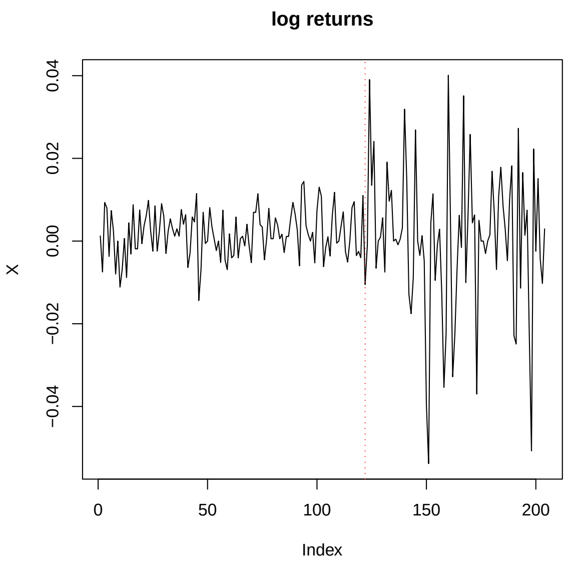
If we assume to observe the following SDE with unknown drift $b(\cdot)$

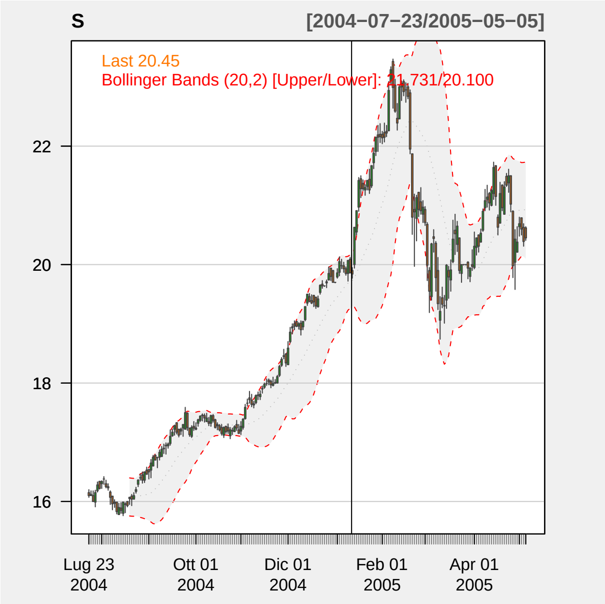
$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \sqrt{\theta}\mathrm{d}W_t,$$

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.


Time
June 30 July 4
July 7 July 11
July 14 July18
August 18 August 22
August 25 August 29
Sept. 1 Sept. 5
Sept. 8 Sept. 12
Sept. 15 Sept. 19
Sept. 22 Sept. 26
Sept. 29 Oct. 3
Oct. 6 Oct. 10
Oct. 20 Oct 24
Lehman Brothers
DJ Stoxx America 600 Banks
DJ Stoxx 600 Banks
Deutsche Bank
HBSC
Barclays
Deutsche Bank (GER)
CAC
DJ Stoxx Global 1800 Banks
FTSE
DJ Stoxx 600
S&P MIB
Nikkei 225
Lehman Brothers
DJ Stoxx 600 Banks
Goldman Sachs
Deutsche Bank
HSBC
Nyse
Dow Jones
S&P 500
FTSE
DAX
S&P MIB
CAC
IBEX
SMI
Nikkei 225
DJ Stoxx 600 Banks
DJ Stoxx Global 1800
MCSI World
Morgan Stanley
Bank of America
Barclays
RBS
Unicredit
Intesa Sanpaolo
Deutsche Bank (GER)
Commerzbank
Nasdaq
Dj Stoxx Asia Pacific
600 Banks
JP Morgan Chase






```
> S <- get.hist.quote("ATL.MI", start = "2004-07-23",  end = "2005-05-05")$Close
> require(sde)
> cpoint(S)
$k0
[1] 123
$tau0
[1] "2005-01-11"
$theta1
2005-01-12
 0.1020001
$theta2
[1] 0.3770111
attr(,"index")
[1] "2005-01-12"
```

The theory works for SDEs of the form

$$\mathrm{d}Y_t = b_t \mathrm{d}t + \sigma(X_t, \theta)\mathrm{d}W_t, \quad t \in [0, T],$$

where W_t a r-dimensional Wiener process and b_t and X_t are multidimensional processes and σ 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 τ^* instant is unknown and is to be estimated, along with θ_1^* and θ_2^*, from the observations sampled from the path of (X, Y).

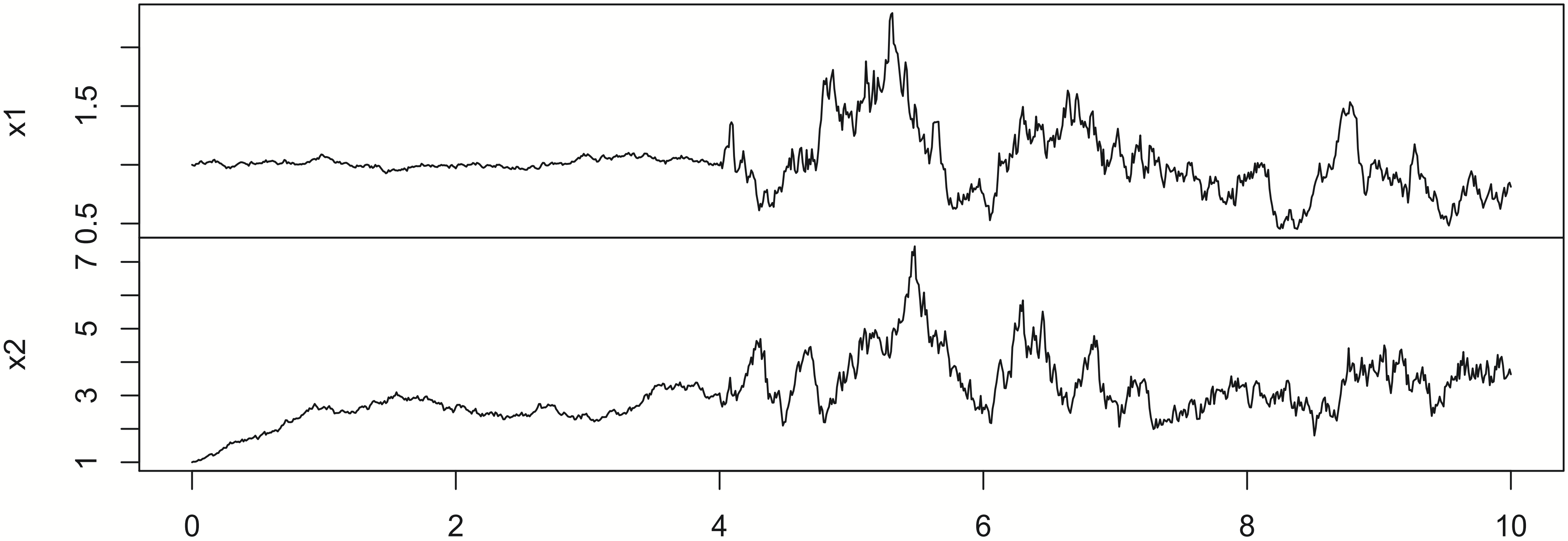
Example of Volatility Change-Point Estimation

Consider the 2-dimensional stochastic differential equation

$$\begin{pmatrix} dX_t^1 \\ dX_t^2 \end{pmatrix} = \begin{pmatrix} 1 - X_t^1 \\ 3 - X_t^2 \end{pmatrix} dt + \begin{bmatrix} \theta_{1.1} \cdot X_t^1 & 0 \cdot X_t^1 \\ 0 \cdot X_t^2 & \theta_{2.2} \cdot X_t^2 \end{bmatrix}' \begin{pmatrix} dW_t^1 \\ dW_t^2 \end{pmatrix}$$

$$X_0^1 = 1.0, \quad X_0^2 = 1.0,$$

with change point instant at time $\tau = 4$



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

```
> t.est <- CPoint(yuima,param1=t1,param2=t2, plot=TRUE)
>
> t.est$tau
[1] 3.99
```

Overview of the `yuima` package

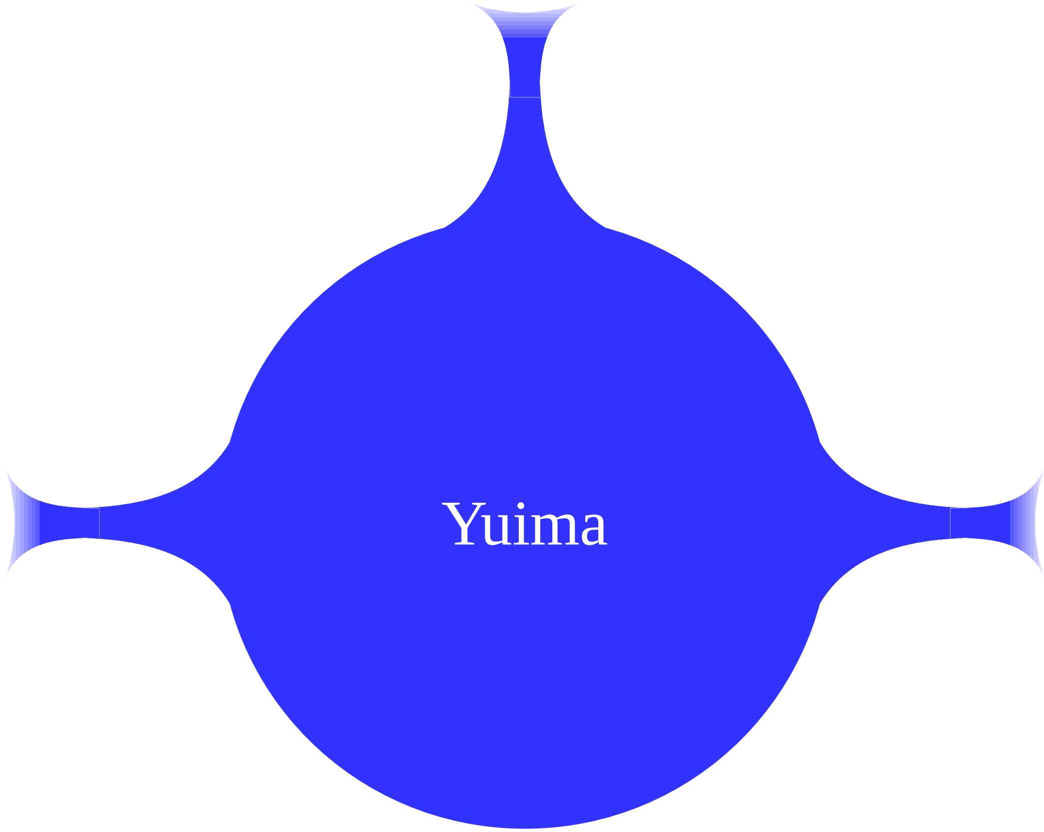
The `yuima` object

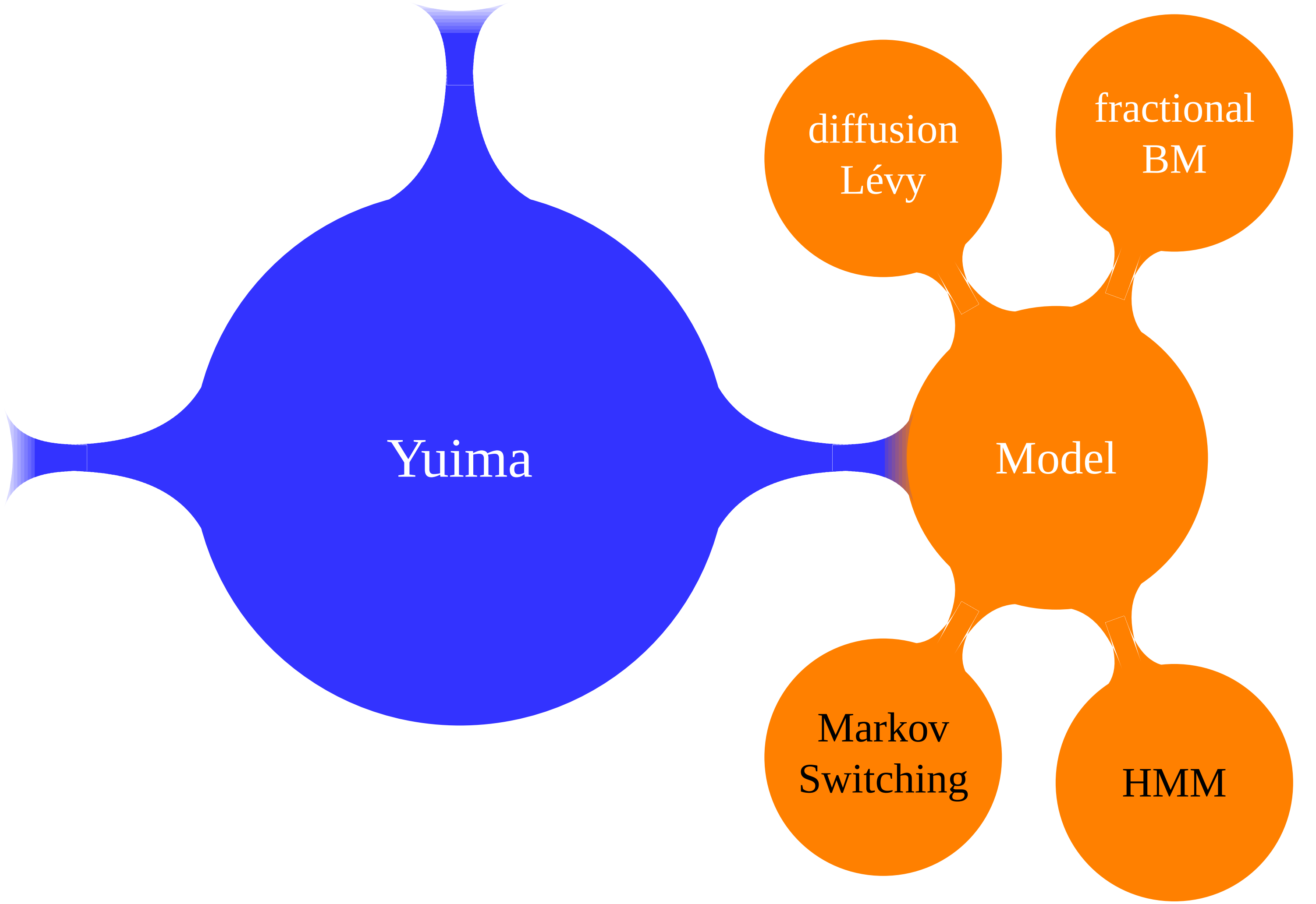
The main object is the `yuima` object which allows to describe the model in a mathematically sound way.

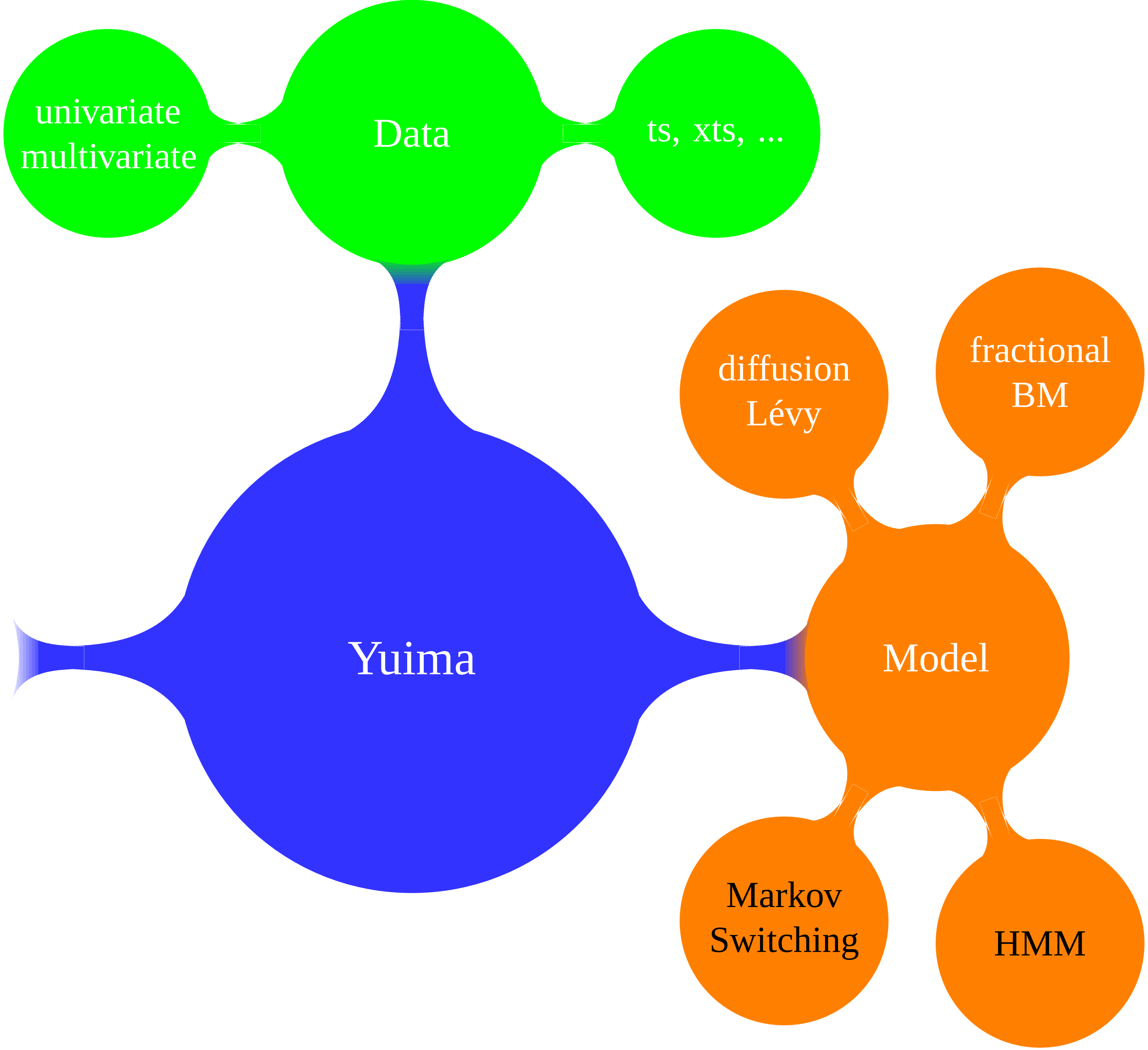
Then the data and the sampling structure can be included as well or, just the sampling scheme from which data can be generated according to the model.

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

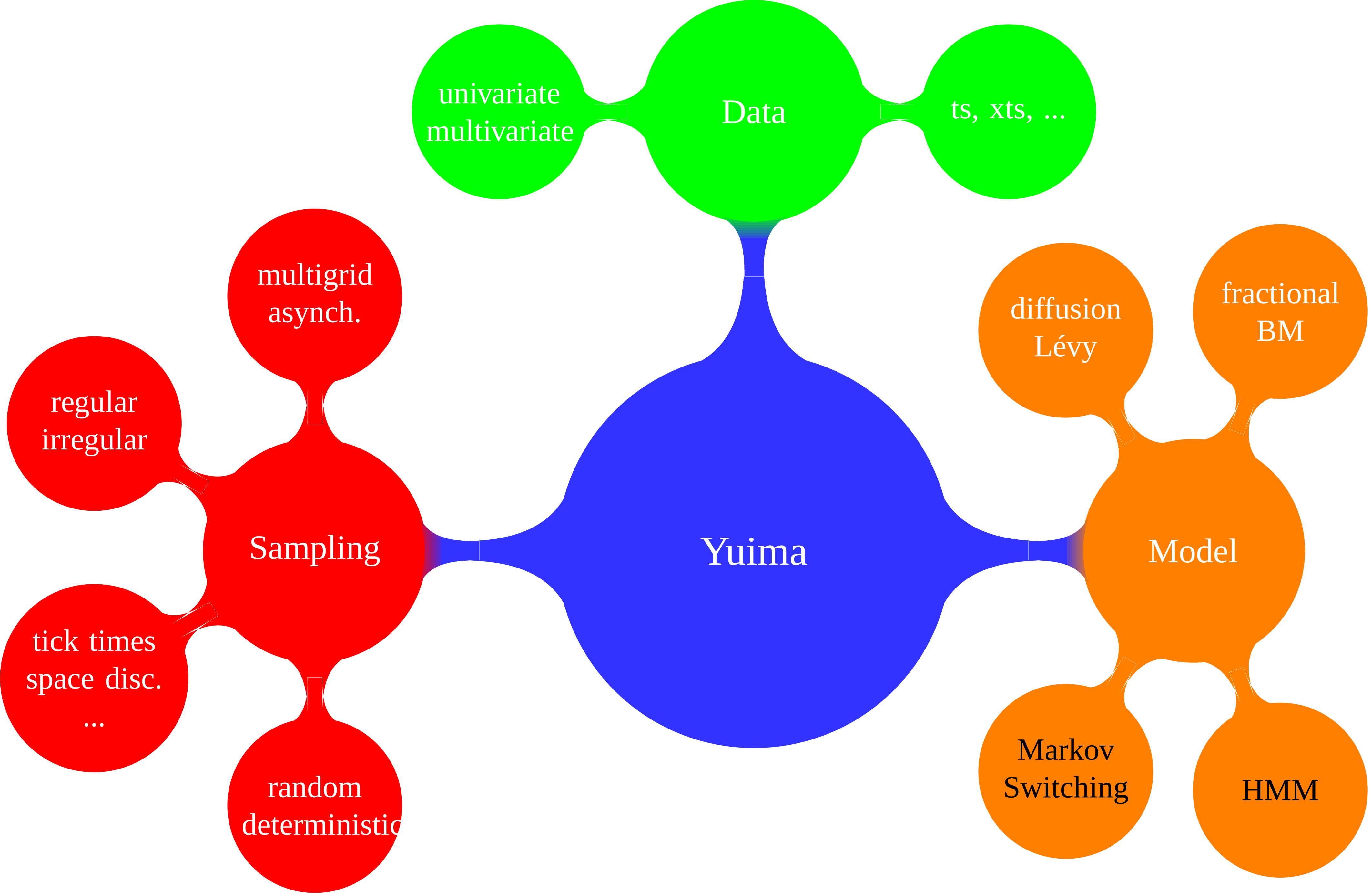
Before looking at the details, let us see an overview of the main object.


Yuima

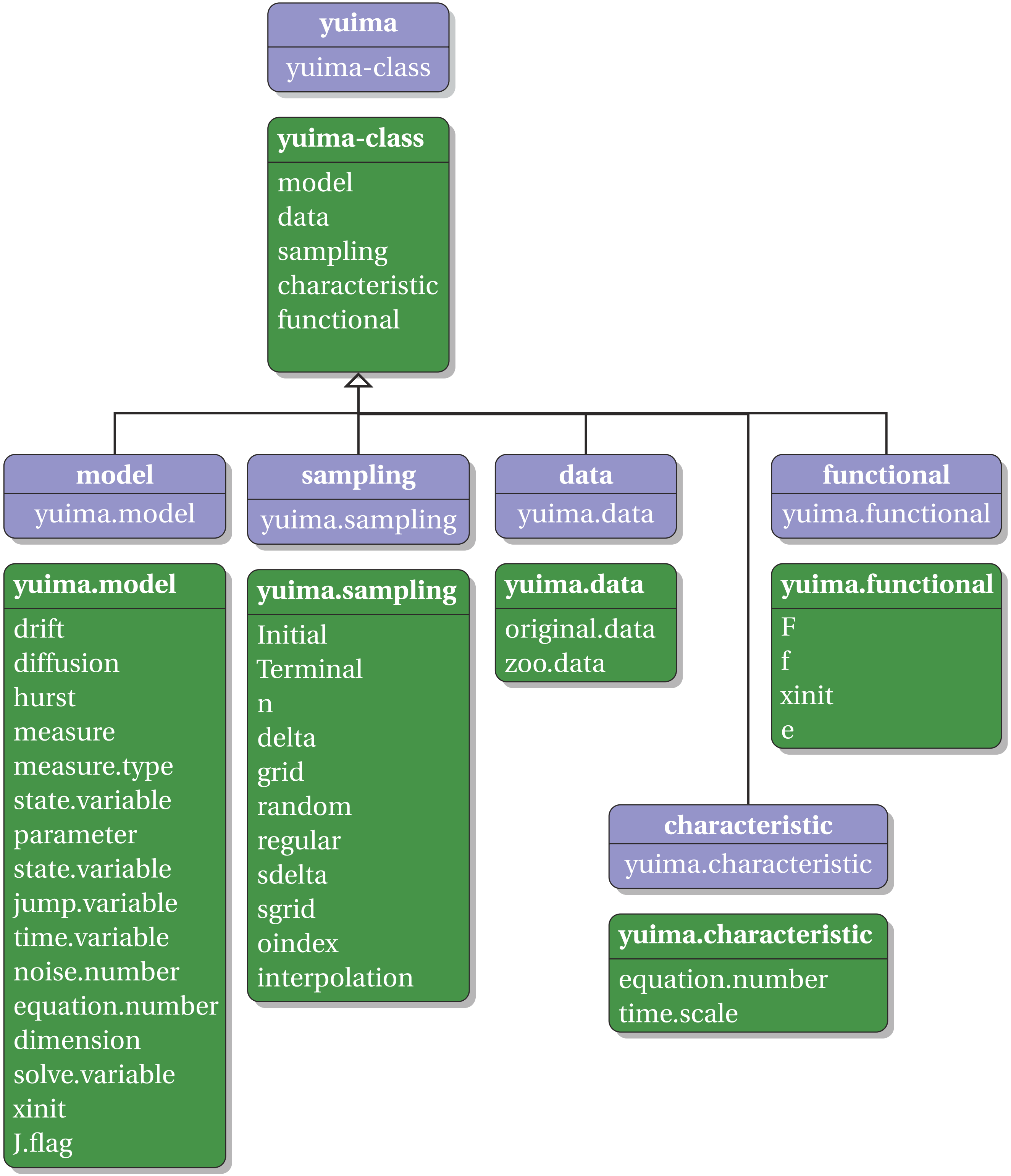

Yuima
Model
diffusion
Lévy
fractional
BM
Markov
Switching
HMM

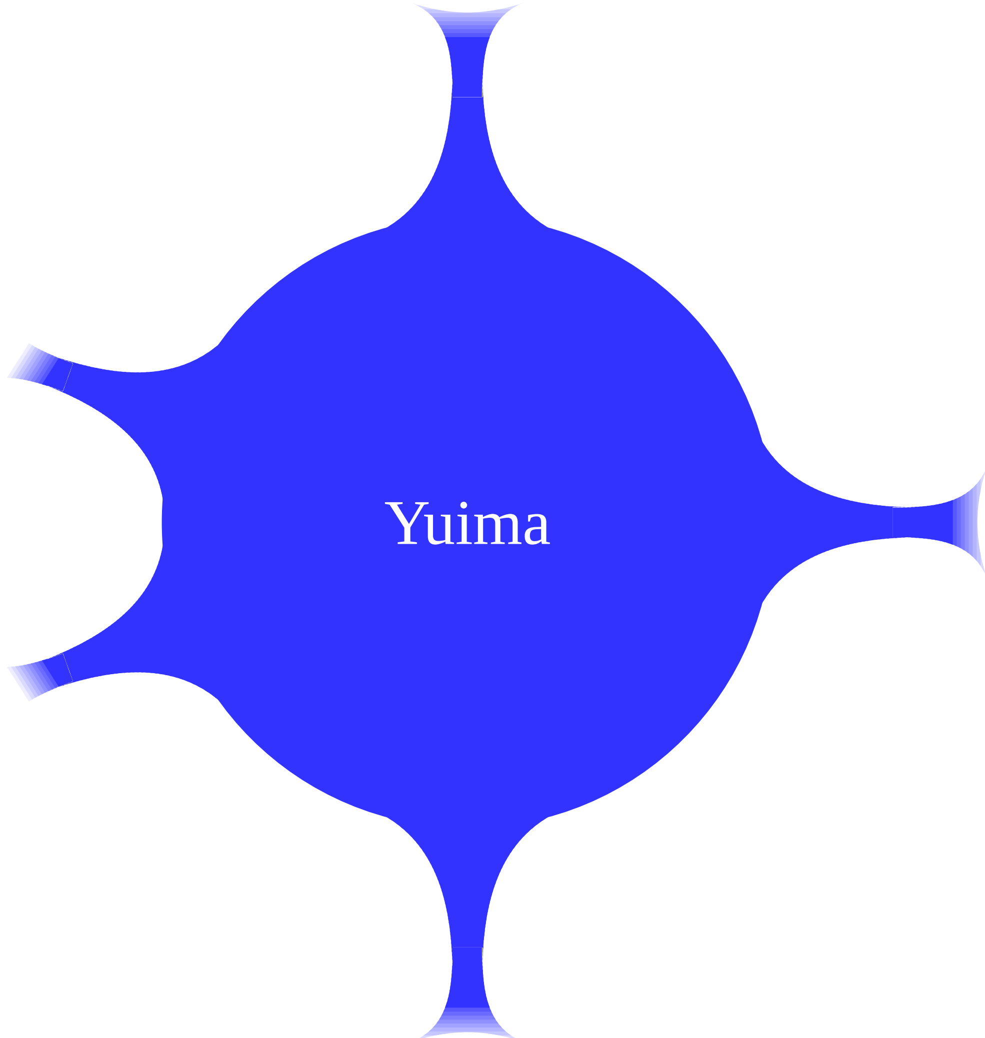


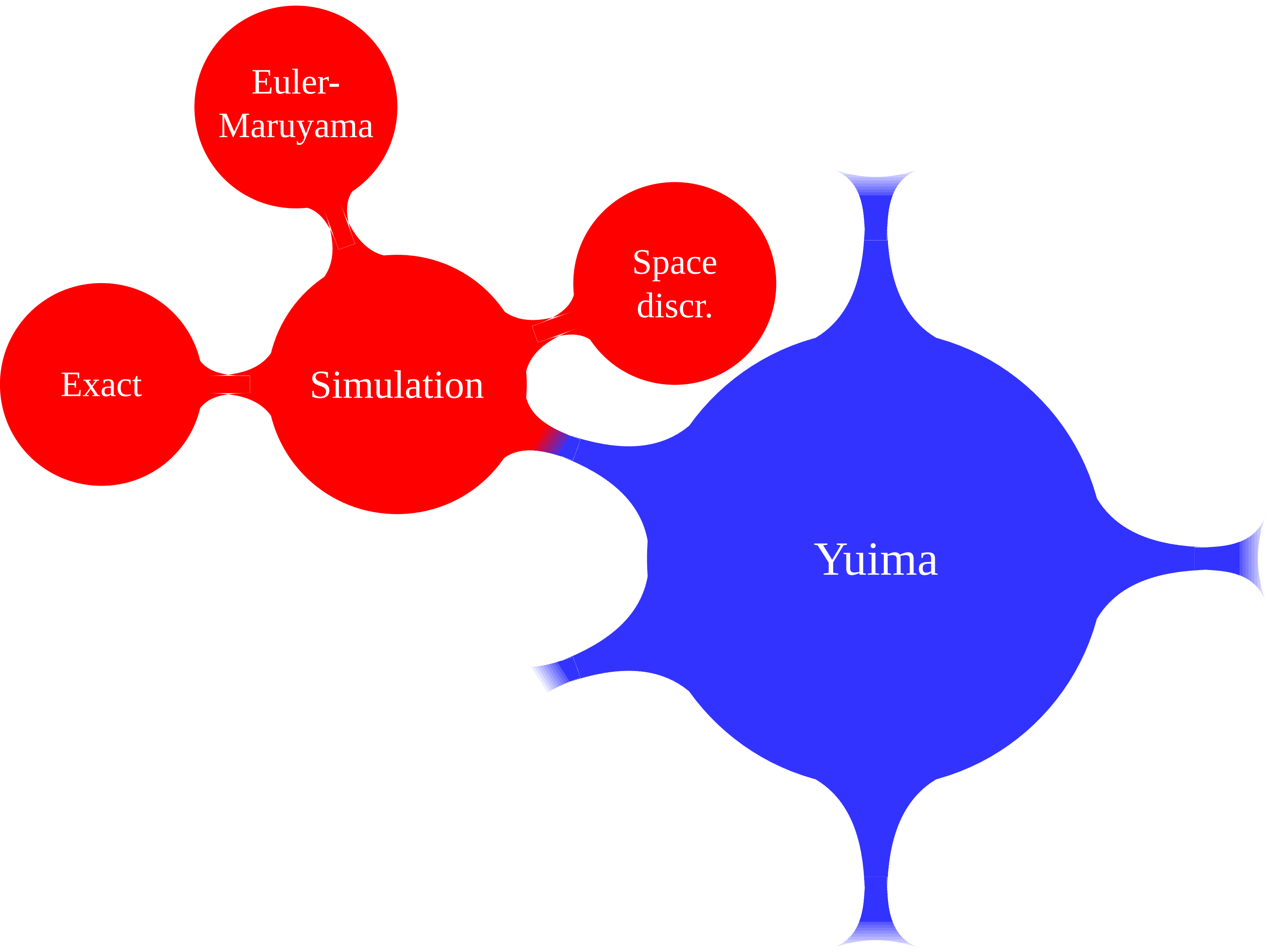
univariate
multivariate
Data
ts, xts, ...
diffusion
Lévy
fractional
BM
Yuima
Model
Markov
Switching
HMM

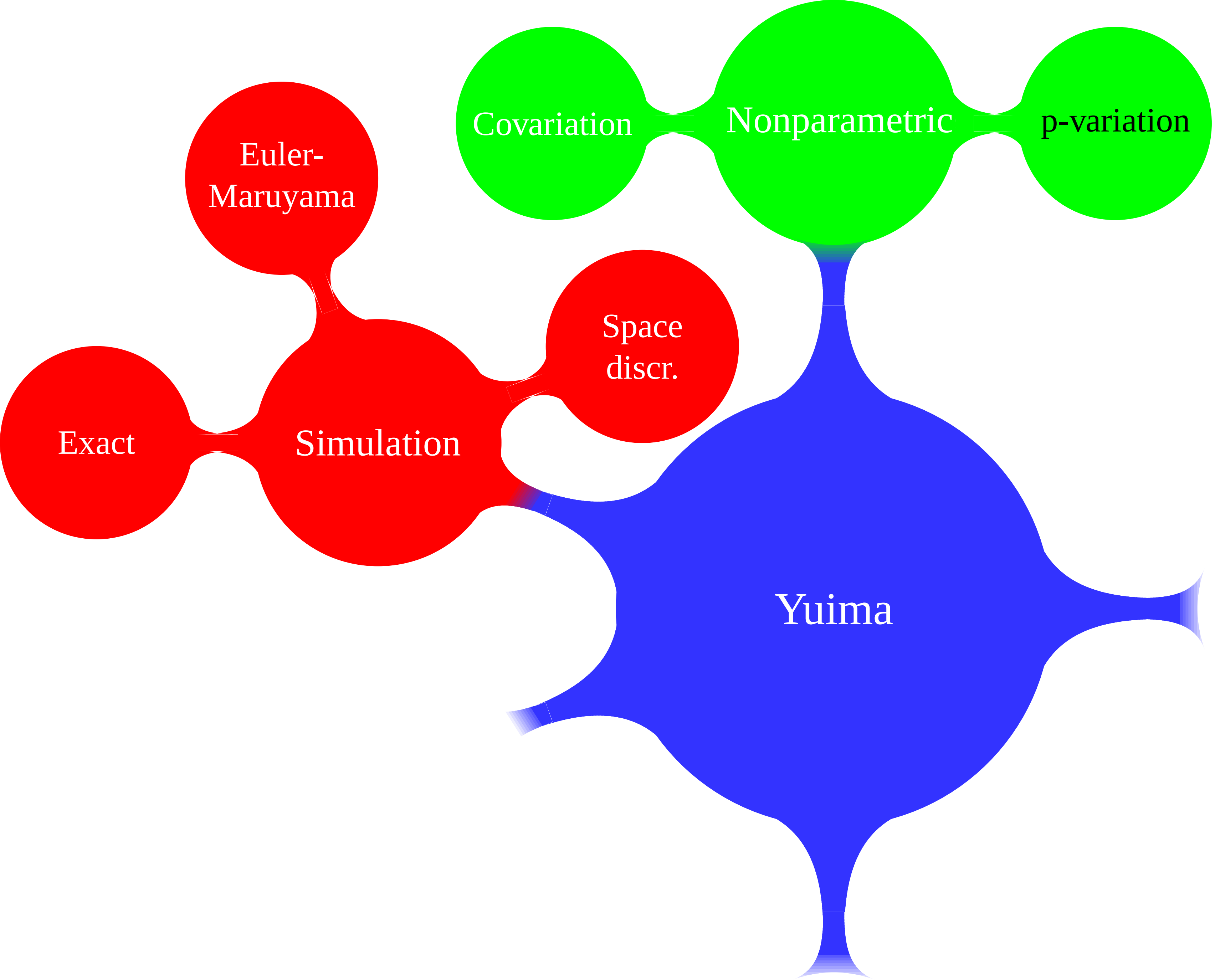


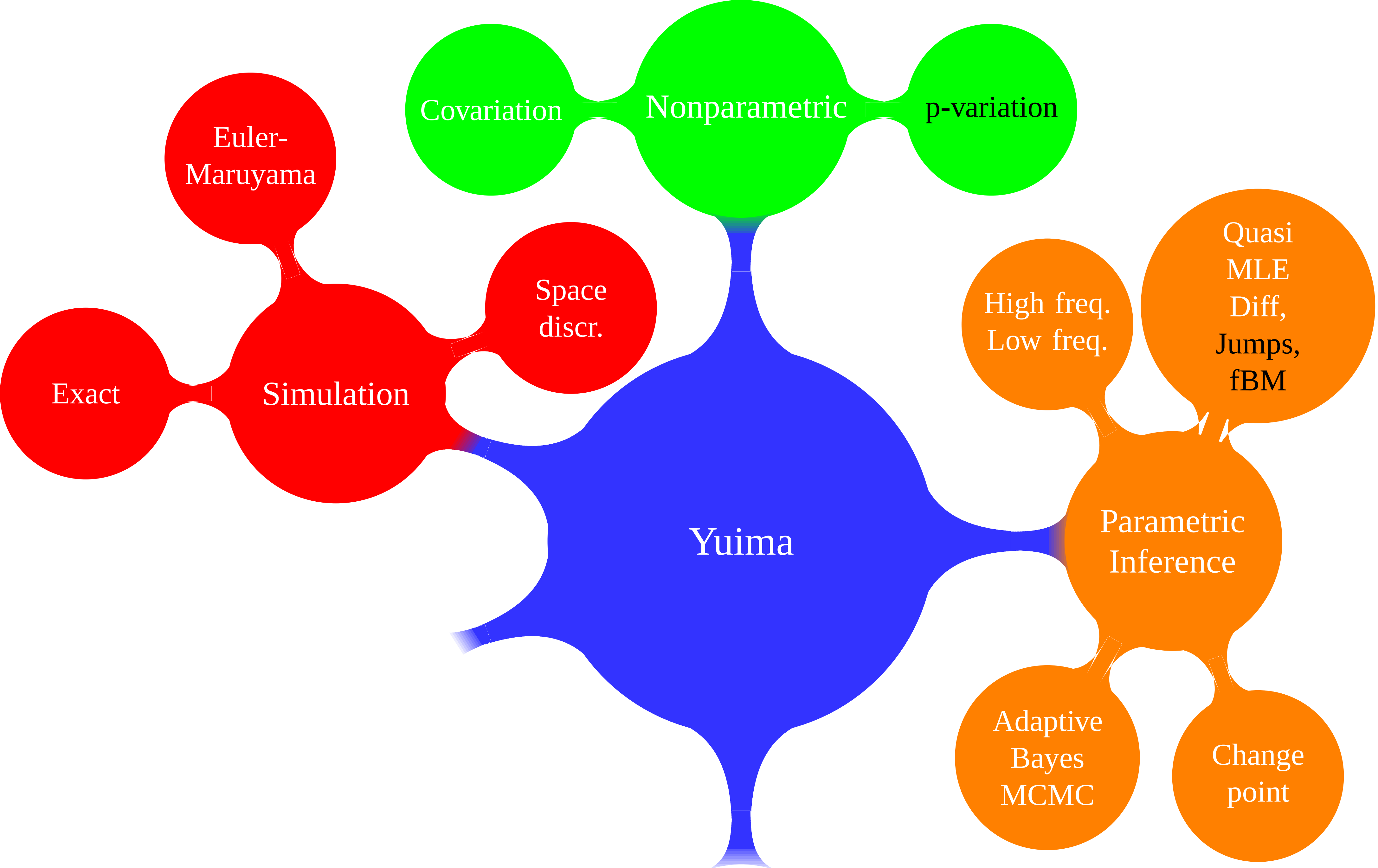
Yuima
Data
univariate
multivariate
ts, xts, ...
Sampling
multigrid
asynch.
regular
irregular
tick times
space disc.
...
random
deterministic
Model
diffusion
Lévy
fractional
BM
Markov
Switching
HMM

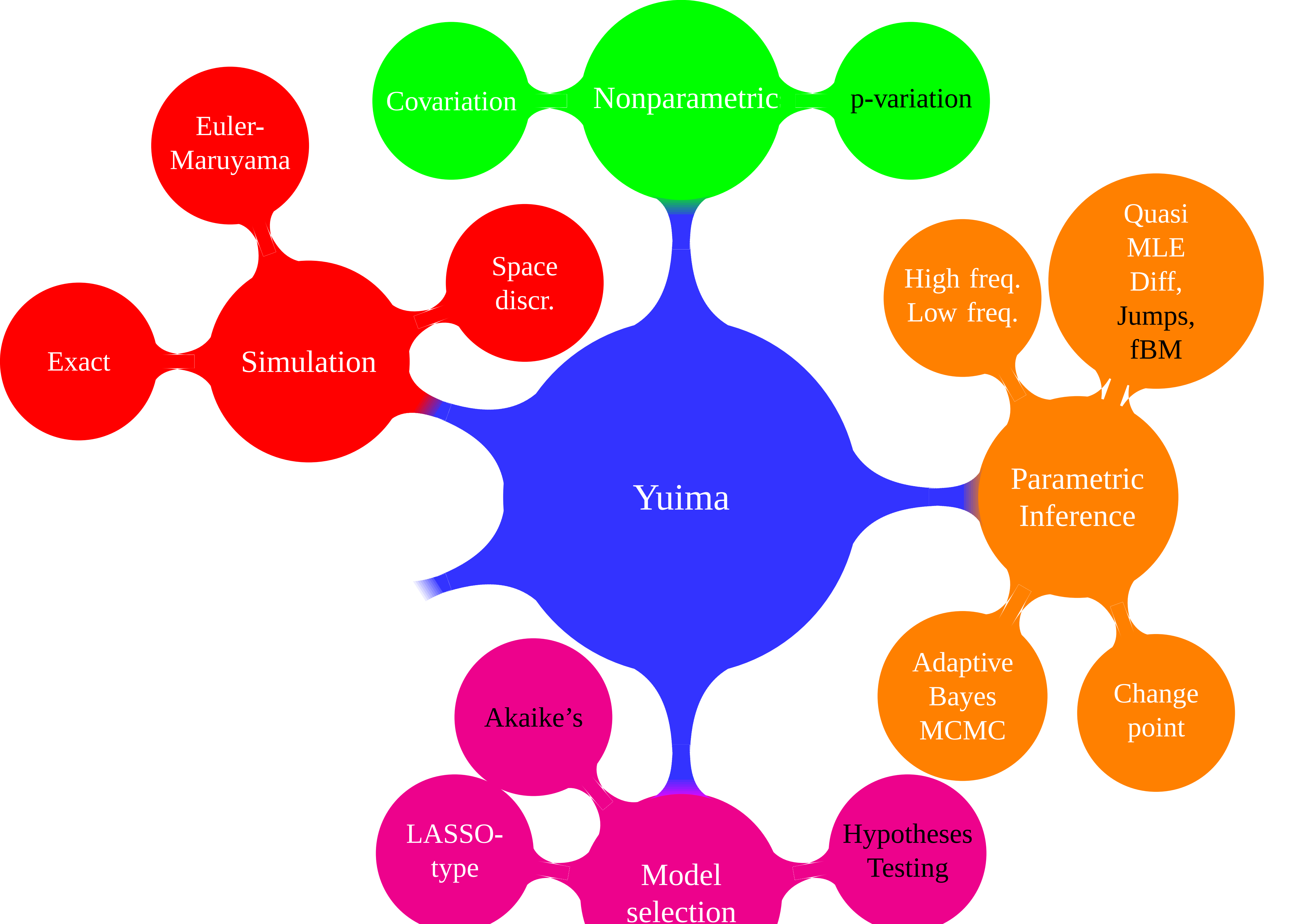

yuima
yuima-class
yuima-class
model
data
sampling
characteristic
functional
model
yuima.model
sampling
yuima.sampling
data
yuima.data
functional
yuima.functional
yuima.model
drift
diffusion
hurst
measure
measure.type
state.variable
parameter
state.variable
jump.variable
time.variable
noise.number
equation.number
dimension
solve.variable
xinit
J.flag
yuima.sampling
Initial
Terminal
n
delta
grid
random
regular
sdelta
sgrid
oindex
interpolation
yuima.data
original.data
zoo.data
yuima.functional
F
f
xinit
e
characteristic
yuima.characteristic
yuima.characteristic
equation.number
time.scale

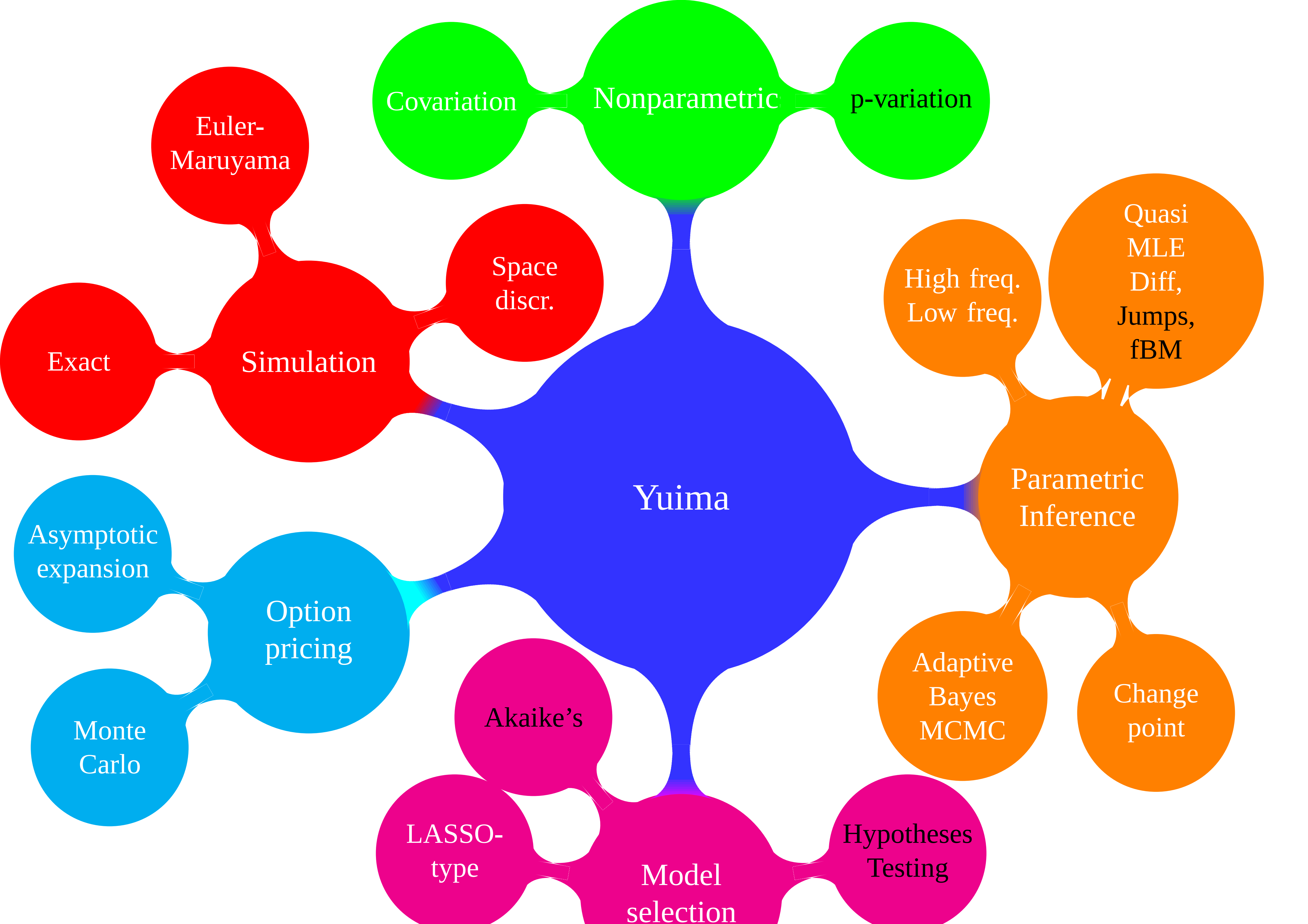

Yuima


Euler-
Maruyama
Space
discr.
Exact
Simulation
Yuima


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


Nonparametric
Covariation
p-variation
Euler-
Maruyama
Space
discr.
Exact
Simulation
Yuima
High freq.
Low freq.
Quasi
MLE
Diff,
Jumps,
fBM
Parametric
Inference
Adaptive
Bayes
MCMC
Change
point


Yuima
Nonparametric
Covariation
p-variation
Simulation
Euler-Maruyama
Space discr.
Exact
Parametric Inference
High freq. Low freq.
Quasi MLE Diff, Jumps, fBM
Adaptive Bayes MCMC
Change point
Model selection
Akaike's
LASSO-type
Hypotheses Testing


Covariation
Nonparametric
p-variation
Euler-Maruyama
Space discr.
Exact
Simulation
High freq. Low freq.
Quasi MLE Diff, Jumps, fBM
Yuima
Parametric Inference
Asymptotic expansion
Option pricing
Monte Carlo
Akaike's
Adaptive Bayes MCMC
Change point
LASSO-type
Model selection
Hypotheses Testing

The model specification

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

The model specification

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

- Diffusions $\mathrm{d}X_t = a(t, X_t)dt + b(t, X_t)\mathrm{d}W_t$

The model specification

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

- Diffusions $\mathrm{d}X_t = a(t, X_t)dt + b(t, X_t)\mathrm{d}W_t$
- Fractional Gaussian Noise, with H the Hurst parameter

$$\mathrm{d}X_t = a(t, X_t)dt + b(t, X_t)\mathrm{d}W_t^H$$

The model specification

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

- Diffusions $\mathrm{d}X_t = a(t, X_t)dt + b(t, X_t)\mathrm{d}W_t$
- Fractional Gaussian Noise, with H the Hurst parameter

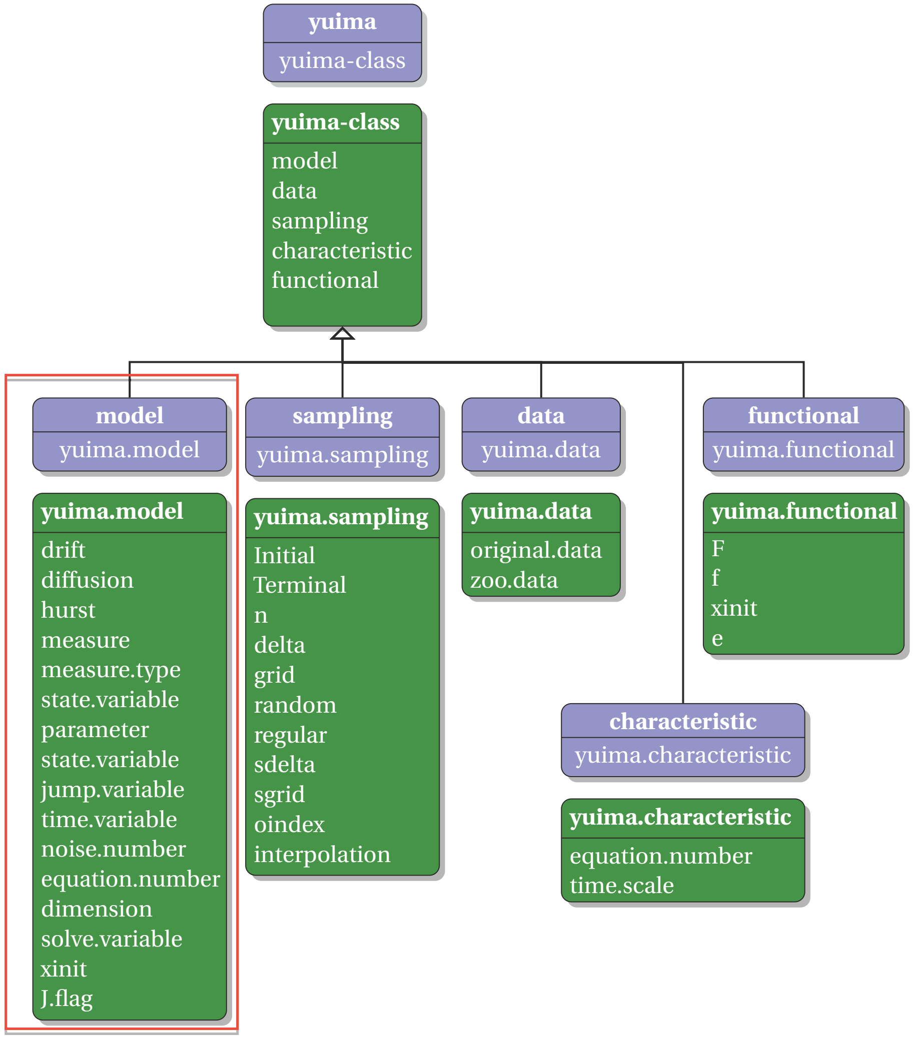
$$\mathrm{d}X_t = a(t, X_t)dt + b(t, X_t)\mathrm{d}W_t^H$$

- Diffusions with jumps, Lévy

$$\mathrm{d}X_t = a(X_t)dt + b(X_t)\mathrm{d}W_t + \int_{|z|>1} c(X_{t-}, z)\mu(\mathrm{d}t, \mathrm{d}z) + \int_{0<|z|\le 1} c(X_{t-}, z)\{\mu(\mathrm{d}t, \mathrm{d}z) - \nu(\mathrm{d}z)\mathrm{d}t\}$$

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

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

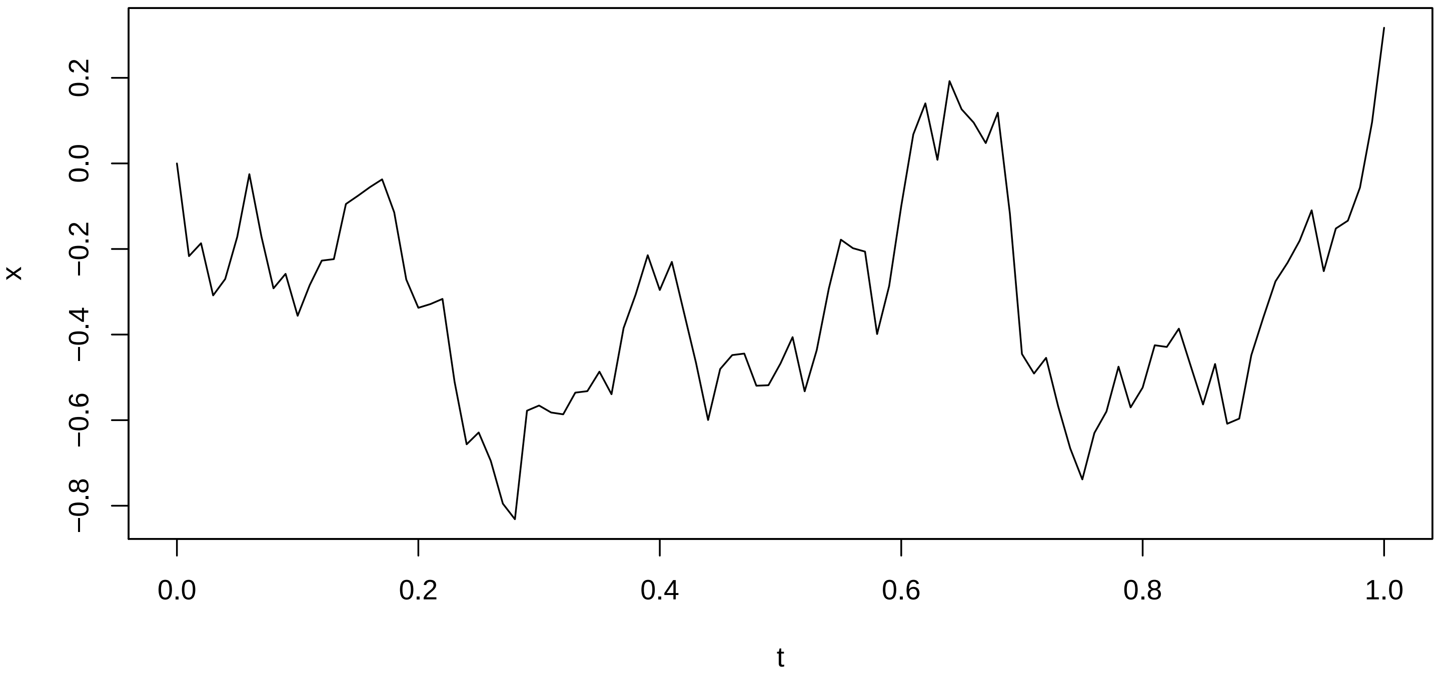


```
> mod1 <- setModel(drift = "-3*x", diffusion = "1/(1+x^2)")
```

```
> 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.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 "x"
  ..@ xinit          : num 0
  ..@ J.flag         : logi FALSE
```

And we can easily simulate and plot the model like

```
> set.seed(123)
> X <- simulate(mod1)
> plot(X)
```

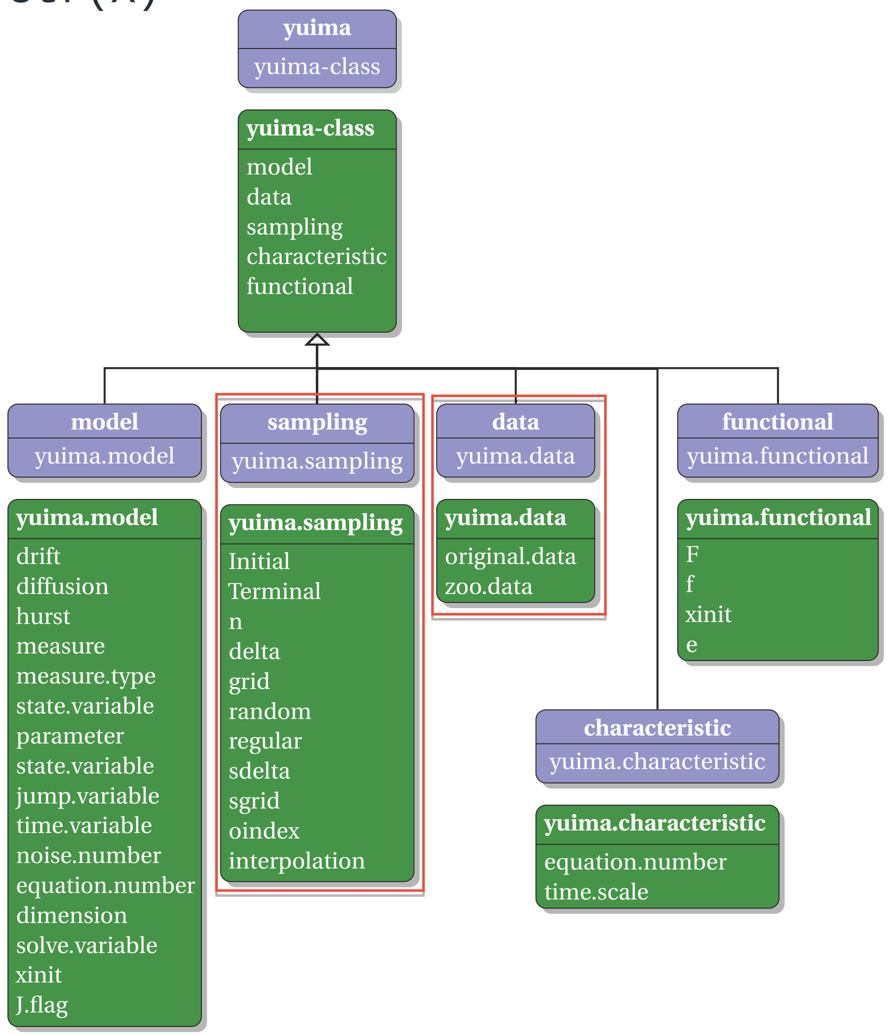


The `simulate` function fills the slots `data` and `sampling`

```
> str(X)
```

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

```
> str(X)
```



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"
```

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

Automatic extraction of the `parameter`s 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
```

And this can be simulated specifying the parameters

```
> simulate(mod2,true.param=list(theta=1,gamma=3))
```

2-dimensional diffusions with 3 noises

$$\begin{aligned}
dX_t^1 &= -3X_t^1 dt + dW_t^1 + X_t^2 dW_t^3 \\
dX_t^2 &= -(X_t^1 + 2X_t^2) dt + X_t^1 dW_t^1 + 3 dW_t^2
\end{aligned}$$

has to be organized into matrix form

$$\begin{pmatrix} dX_t^1 \\ dX_t^2 \end{pmatrix} = \begin{pmatrix} -3X_t^1 \\ -X_t^1 - 2X_t^2 \end{pmatrix} dt + \begin{pmatrix} 1 & 0 & X_t^2 \\ X_t^1 & 3 & 0 \end{pmatrix} \begin{pmatrix} dW_t^1 \\ dW_t^2 \\ dW_t^3 \end{pmatrix}$$

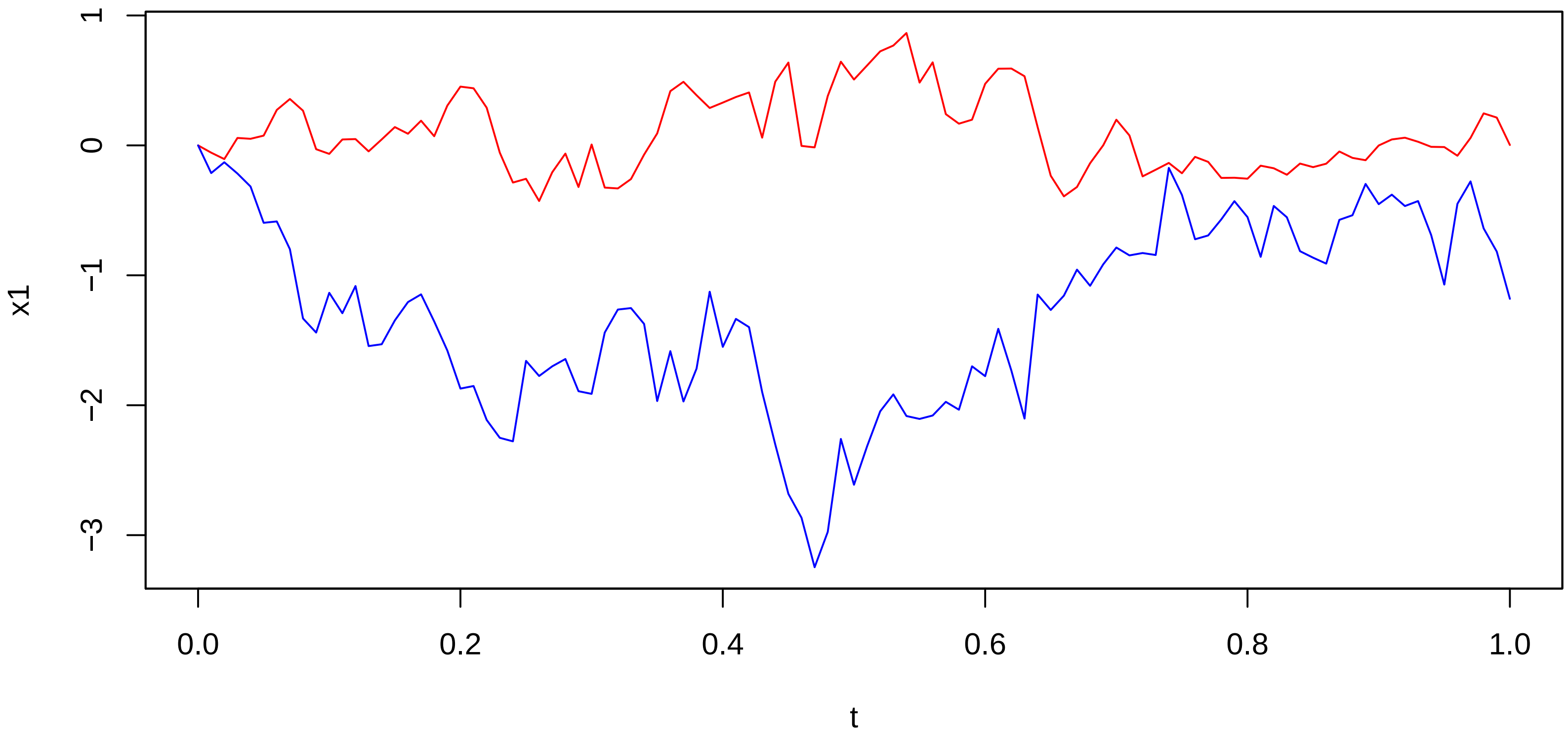
```
> sol <- c("x1","x2") # variable for numerical solution
> a <- c("-3*x1","-x1-2*x2")   # drift vector
> b <- matrix(c("1","x1","0","3","x2","0"),2,3)  #  diffusion matrix
> mod3 <- setModel(drift = a, diffusion = b, solve.variable = sol)
```

$$
\begin{aligned}
\mathrm{d}X_t^1 &= -3X_t^1\mathrm{d}t + \mathrm{d}W_t^1 + X_t^2\mathrm{d}W_t^3\\
\mathrm{d}X_t^2 &= -(X_t^1 + 2X_t^2)\mathrm{d}t + X_t^1\mathrm{d}W_t^1 + 3\mathrm{d}W_t^2
\end{aligned}
$$

```
> 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

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



Also models likes this can be specified

$$\begin{cases} \mathrm{d}X_t^1 = X_t^2 \left|X_t^1\right|^{2/3} \mathrm{d}W_t^1, \\ \mathrm{d}X_t^2 = g(t)\mathrm{d}X_t^3, \\ \mathrm{d}X_t^3 = X_t^3(\mu \mathrm{d}t + \sigma(\rho \mathrm{d}W_t^1 + \sqrt{1-\rho^2}\mathrm{d}W_t^2)) \end{cases},$$

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

The above is an example of parametric SDE with more equations than noises.

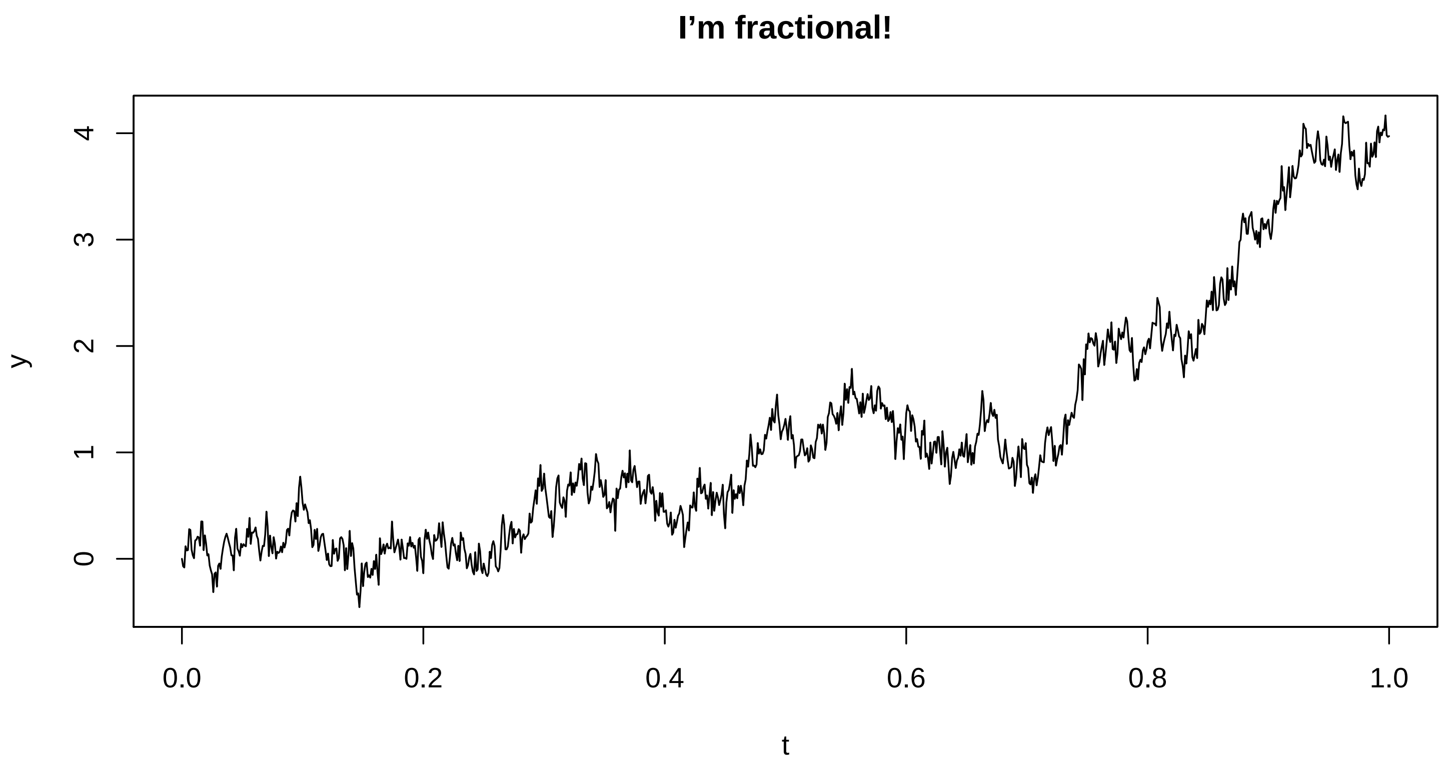
```
> mod4 <- setModel(drift=3*y, diffusion=1, hurst=0.3, solve.var=y)
```

```
> 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
```

```
> set.seed(123)
> X <- simulate(mod4, n=1000)
> plot(X, main="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 is is possible to consider the following SDE which involves jumps

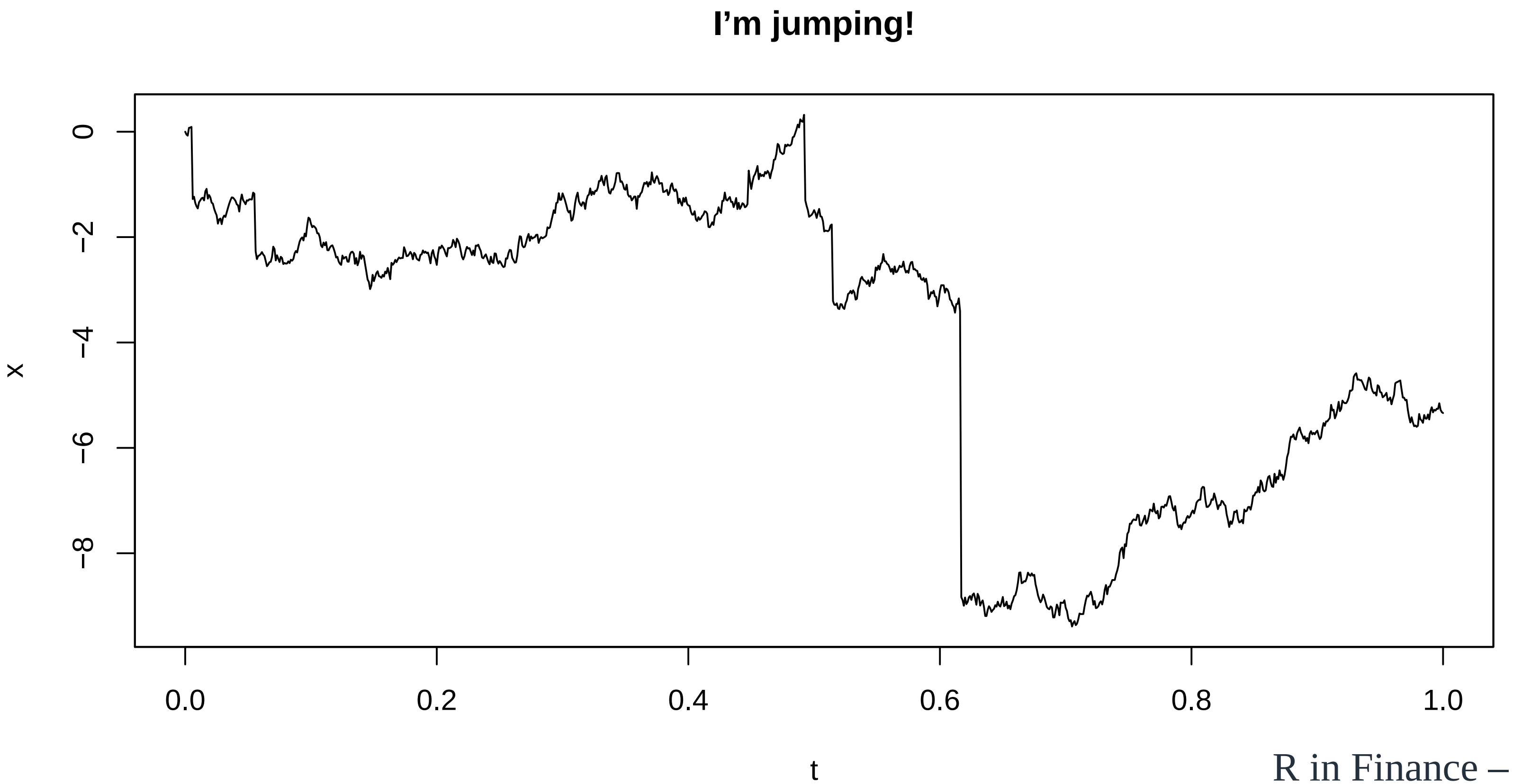
$$\mathrm{d}X_t = a(X_t)dt + b(X_t)\mathrm{d}W_t + \mathrm{d}Z_t$$

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)

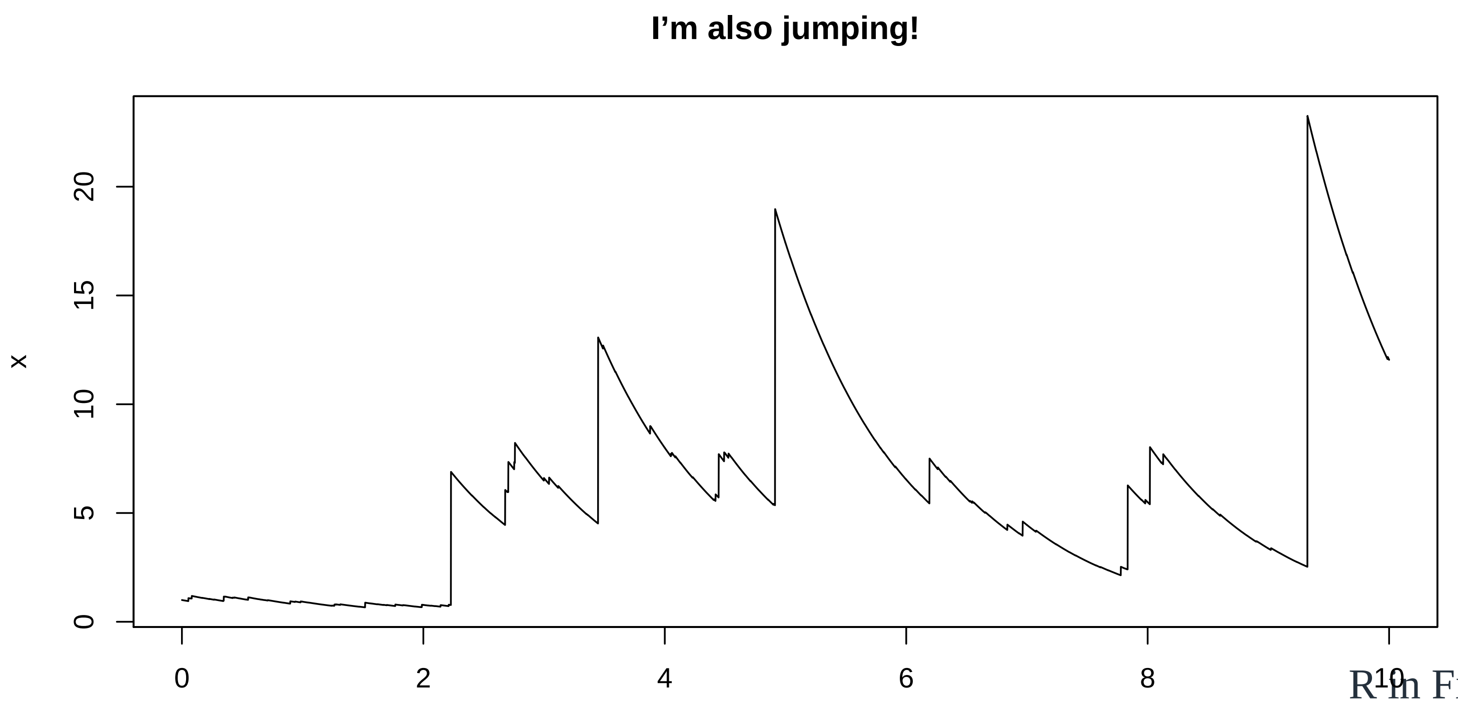
Jump process: $\mathrm{d}X_t = -\theta X_t \mathrm{d}t + \sigma \mathrm{d}W_t + Z_t$

```
> 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!")
```



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 $\mathrm{d}X_t = -X_t dt + dZ_t$

```
> 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!")
```



The `setModel` method

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)` in

$$\mathrm{d}X_t = a(X_t)\mathrm{d}t + b(X_t)\mathrm{d}W_t + c(X_t)\mathrm{d}Z_t$$

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

`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)` in

$$\mathrm{d}X_t = a(X_t)\mathrm{d}t + b(X_t)\mathrm{d}W_t + c(X_t)\mathrm{d}Z_t$$

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)` in

$$\mathrm{d}X_t = a(X_t)\mathrm{d}t + b(X_t)\mathrm{d}W_t + c(X_t)\mathrm{d}Z_t$$

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 reamins 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 `yuima` package can handle asymptotic expansion of functionals of d-dimensional diffusion process

$$\mathrm{d}X_t^\varepsilon = a(X_t^\varepsilon, \varepsilon)\mathrm{d}t + b(X_t^\varepsilon, \varepsilon)\mathrm{d}W_t, \qquad \varepsilon \in (0,1]$$

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

$$F^\varepsilon(X_t^\varepsilon) = \sum_{\alpha=0}^{r} \int_0^T f_\alpha(X_t^\varepsilon, \mathrm{d})\mathrm{d}W_t^\alpha + F(X_t^\varepsilon, \varepsilon), \qquad W_t^0 = t$$

Estimation of functionals. Example.

Example: B&S asian call option

$$\mathrm{d}X_t^\varepsilon = \mu X_t^\varepsilon \mathrm{d}t + \varepsilon X_t^\varepsilon \mathrm{d}W_t$$

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

$$F^\varepsilon(X_t^\varepsilon) = \frac{1}{T}\int_0^T X_t^\varepsilon \mathrm{d}t, \qquad r = 1$$

Example: B&S asian call option

$$\mathrm{d}X_t^\varepsilon = \mu X_t^\varepsilon \mathrm{d}t + \varepsilon X_t^\varepsilon \mathrm{d}W_t$$

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

$$F^\varepsilon(X_t^\varepsilon) = \frac{1}{T}\int_0^T X_t^\varepsilon \mathrm{d}t, \qquad r = 1$$

with

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

in

$$F^\varepsilon(X_t^\varepsilon) = \sum_{\alpha=0}^{r}\int_0^T f_\alpha(X_t^\varepsilon, \mathrm{d})\mathrm{d}W_t^\alpha + F(X_t^\varepsilon, \varepsilon)$$

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

$$F^\varepsilon(X_t^\varepsilon) = \frac{1}{T}\int_0^T X_t^\varepsilon \mathrm{d}t, \qquad r = 1$$

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^ε provides unexpectedly accurate approximations.

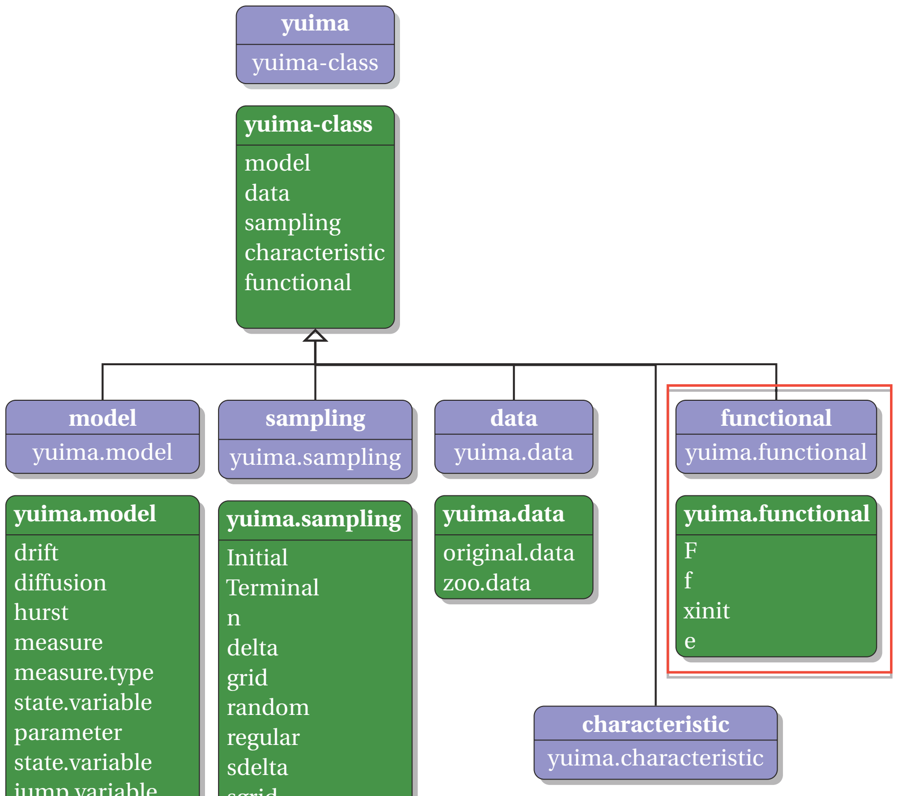
The `yuima` package provides functions to construct the functional F^ε, and automatic asymptotic expansion based on Malliavin calculus starting from a `yuima` object.

setFunctional method


```
> 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
> yuima <- setFunctional( yuima, f=f,F=F, xinit=xinit,e=e)
```

`setFunctional` method


```
> 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, r
> yuima <- setFunctional( yuima, f=f,F=F, xinit=xinit,e=e)
```



setFunctional method

```
> 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
> 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
```

Estimation of functionals. Example.

Then, it is as easy as

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

Then, it is as easy as

```
> 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

```
> 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
+ }
```

Add more terms to the expansion

The expansion of previous functional gives

```
> 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

> 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

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

[1] 0.7184944
> 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

Multivariate Asymptotic Expansion

Asymptotic expansion is now also available for multidimensional diffusion processes like the Heston model

$$
\begin{aligned}
\mathrm{d}X_t^{1,\varepsilon} &= aX_t^{1,\varepsilon}\mathrm{d}t + \varepsilon X_t^{1,\varepsilon}\sqrt{X_t^{2,\varepsilon}}\mathrm{d}W_t^1 \\
\mathrm{d}X_t^{2,\varepsilon} &= c(d - X_t^{2,\varepsilon})\mathrm{d}t + \varepsilon\sqrt{X_t^{2,\varepsilon}}\left(\rho\mathrm{d}W_t^1 + \sqrt{1-\rho^2}\mathrm{d}W_t^2\right)
\end{aligned}
$$

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

Advertizing

- Iacus (2008) *Simulation and Inference for Stochastic Differential Equations with R Examples*, Springer, New York.
- Iacus (2011) *Option Pricing and Estimation of Financial Models with R*, Wiley & Sons, Chichester.