ESGtoolkit, a tool for stochastic simulation (v0.2.0)

Thierry Moudiki
13th January 2020

Contents

1 Overview

1.1 Context

Initially, I developped this package for insurance. In 2014. **If you're not working in insurance, ESGtoolkit is still relevant for stochastic simulation** in Finance, Economics, or Physics (...). In that case, you can start directly by reading 1.2. Oh, in this section, read at least this paragraph: 1.1.

An *Economic Scenario Generator* (ESG) is a tool for projection of plausible future paths of insurers' financial assets. It helps in pricing its products, and assessing its current and future solvency. Two types of ESGs are generally needed: a *real-world* ESG, and a a *market consistent* ESG.

The aim of a real-world ESG is to produce projections of risk factors, whose patterns are coherent with the past distribution of those risk factors. Real-world scenarios are mainly used for the valuation of solvency requirements.

A market consistent ESG shall produce projections of risk factors that are coherent with market prices observed at the valuation date. Market consistent scenarios are mainly used for the best estimate valuation of technical reserves.

Hence, in real-world simulations the historical probability is used and in market consistent simulations, the projection of risk factors is made in a risk-neutral probability. A risk-neutral probability measure is a measure under which the discounted prices of assets are martingales.

A simple example of transitioning from a simulation under the historical probability to a simulation under a risk-neutral probability can be made by using the Black-Scholes model, a geometric Brownian motion. In a real-world simulation, we can assume that an asset evolves according to the following Stochastic Differential Equation (SDE) (with a drift μ, a volatility σ, and $(W(t))_{t \geq 0}$ being a standard brownian motion):

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t) \tag{1}$$

Let r be a constant risk-free rate. $e^{-rt}S(t)$, the discounted price of $S(t)$, will be a martingale if

$$d(e^{-rt}S(t)) \tag{2}$$

is driftless. Applying Ito's formula to $e^{-rt}S(t)$, we have :

$$\begin{aligned} d(e^{-rt}S(t)) &= -re^{-rt}S(t)dt + e^{-rt}dS(t) - \frac{1}{2}.0 < dS(t), dS(t) > & (3) \\ &= -re^{-rt}S(t)dt + e^{-rt}\mu S(t)dt + e^{-rt}S(t)\sigma dW(t) & (4) \\ &= e^{-rt}S(t)\left[(\mu - r)dt + \sigma dW(t)\right] & (5) \end{aligned}$$

Thus, the drift vanishes iff $\mu = r$. That is, if our asset with price $S(t)$ at time t rewards the risk-free rate r. Under this martingale probability measure, asset price dynamics over time can be re-written as:

$$dS(t) = rS(t)dt + \sigma S(t)dW^*(t) \tag{6}$$

Where $(W^*(t))_{t \geq 0}$ is a standard brownian motion under the chosen risk-neutral measure.

As we'll see in section 1.2, ESGtoolkit does not directly provide multiple asset models but instead, some building blocks for constructing a variety of these. Two main functions are therefore provided for his purpose: `simshocks`, `simdiff`. Other tools for statistical testing and visualization are presented as well.

As a reminder: There are no perfect models, and the more sophisticated doesn't necessarily mean the most judicious. To avoid possible disasters, it's important to know precisely the strengths and weaknesses of a model before using it.

1.2 simdiff

Let $(W(t))_{t \geq 0}$ be a standard brownian motion. `simdiff` makes simulations of a diffusion process $(X(t))_{t \geq 0}$, which evolves through time according to the following equation:

$$dX(t) = \mu(t, X(t))dt + \sigma(t, X(t))dW(t) + \gamma(t, X(t-), J)dN(t) \tag{7}$$

Actually, (Eq. 7) is a generic formulation of all `simdiff` models. **Not all parts of this expression are required all the times**, but $\sigma(t, X(t))dW(t)$, describing our process' volatility. Let's describe the other parts.

$\gamma(t, X(t-), J)dN(t)$ is optional. It contains jumps of the process, that occur according to a homogeneous Poisson process $(N(t))_{t \geq 0}$ with intensity λ. The time elasped between two jumping times follows an exponential $\epsilon(\lambda)$ distribution; and the number of jumps of the process on $[0, t[$ follows a Poisson distribution $\mathcal{P}(\lambda t)$. The magnitude of the jumps is controlled by J.

Now, for the blue part of (Eq. 7), we could have:

- An Orsnstein-Uhlenbeck process; for `simdiff` used with parameter `model = "OU"`, and parameters `theta1`, `theta2` and `theta3` provided (if `theta1` or `theta2` are not necessary for building the model, they are to be provided and set to `0`):

$$\begin{aligned} \mu(t, X(t)) &= (\theta_1 - \theta_2 X(t)) \\ \sigma(t, X(t)) &= \theta_3 \end{aligned}$$

- A Cox-Ingersoll-Ross process; for `simdiff` used with parameter `model = "CIR"`, and parameters `theta1`, `theta2`, `theta3` provided (if `theta1` or `theta2` are not necessary for building the model, they are to be provided and set to `0`) :

$$\begin{aligned} \mu(t, X(t)) &= (\theta_1 - \theta_2 X(t)) \\ \sigma(t, X(t)) &= \theta_3 \sqrt{X(t)} \end{aligned}$$

- A Geometric Brownian motion, or *augmented* versions; for `simdiff` used with parameter `model = "GBM"`, and parameters `theta1`, `theta2`, `theta3` provided. For the sake of clarity, the argument `model` is set to `"GBM"`, but not only the Geometric Brownian motion with constant parameters is available. We could have :

  A Geometric Brownian Motion

$$\begin{aligned}\mu(t, X(t)) &= \theta_1 X(t)\\ \sigma(t, X(t)) &= \theta_2 X(t)\end{aligned}$$

  A modified Geometric Brownian Motion, with time-varying drift and constant volatility

$$\begin{aligned}\mu(t, X(t)) &= \theta_1(t) X(t)\\ \sigma(t, X(t)) &= \theta_2 X(t)\end{aligned}$$

  A modified Geometric Brownian Motion, with time-varying volatility and constant drift

$$\begin{aligned}\mu(t, X(t)) &= \theta_1 X(t)\\ \sigma(t, X(t)) &= \theta_2(t) X(t)\end{aligned}$$

It's technically possible to have both θ_1 and θ_2 varying with time (both provided as multivariate time series). But it's not advisable to do this, unless you know exactly why you're doing it.

Jumps are available only for `model = "GBM"`. The jumps arising from the Poisson process have a common magnitude $J = 1 + Z$, whose distribution ν is either lognormal or double-exponential. Between two jumps, the process behaves like a Geometric Brownian motion, and at jumping times, it increases by Z%. For lognormal jumps (Merton model), the distribution ν of J is:

$$log(J) = log(1+Z) \sim \mathcal{N}(log(1+\mu_Z) - \frac{\sigma_Z^2}{2}, \sigma_Z^2) \tag{8}$$

For double exponential jumps (Kou's model), the distribution ν of J is:

$$log(J) = log(1+Z) \sim \nu(dy) = p\frac{1}{\eta_u}e^{-\frac{1}{\eta_u}}\mathbb{1}_{y>0} + (1-p)\frac{1}{\eta_d}e^{\frac{1}{\eta_d}}\mathbb{1}_{y<0} \tag{9}$$

Hence for taking jumps into account when `model = "GBM"`, optional parameters are to be provided to `simdiff`, namely:

- lambda: intensity of the Poisson process
- mu_z: average jump magnitude (only for **lognormal** jumps)
- sigma_z: standard deviation of the jump magnitude (only for **lognormal** jumps)
- p: probability of positive jumps (only for **double exponential** jumps)
- eta_up: the mean of positive jumps (only for **double exponential** jumps)
- eta_down: the mean of negative jumps (only for **double exponential** jumps)

simdiff's core loops are written in C++ *via* Rcpp because: speed. Currently, for an Ornstein-Uhlenbeck process with model = "OU", a Cox-Ingersoll-Ross process with model = "CIR", or a geometric brownian motion with model = "GBM", an exact simulation is used, which means there's no discretization of $(X(t))_{t\geq 0}$ on a time grid. You can also choose an horizon of projection and a sampling frequency (annual, semi-annual, quarterly ...). simdiff's output is a time series object created by base R function ts(). And since this output is a ts() object, it means you can use R functions such as window.ts or frequency on it.

For a customized simulation of $\epsilon \sim \mathcal{N}(0,1)$ embedded in (Eq. 7) *via* $dW(t) = \epsilon dt$, you can fill simdiff's parameter eps with an output of function simshocks. simshocks is described in the next section, 1.3, and an example of such a procedure is described in 2.

1.3 simshocks

simshocks is the complementary function to simdiff, with which you can simulate $\epsilon \sim \mathcal{N}(0,1)$ (that we call shocks), embedded into diffusion (Eq. 7) as:

$$dW(t) = \epsilon dt \tag{10}$$

As simdiff, simshocks is written in C++ *via* Rcpp. And when it comes to the simulation of multi-factors models, or the simulation of risk factors with flexible dependence structure, simshocks calls the underlying function CDVinesim, from package CDVine. CDVineSim makes simulations of canonical (**C-vine**) and **D-vine copulas**.

Simply put, a copula is a function which gives a multidimensional distribution to given margins. If $(X_1, \ldots, X_d)^T$ is a random vector with margins of cumulative distribution functions $F_1, \ldots, F_d$, there exists a copula function C, such that the d-dimensional cumulative distribution function of $(X_1, \ldots, X_d)^T$ is :

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)) \tag{11}$$

If the marginal distributions $F_1, \ldots, F_d$ are continuous, then C is unique. On the other hand, if C is a copula, and $F_1, \ldots, F_d$ are 1-dimensional cumulative distribution functions, the previous equation defines a joint cumulative distribution function for $(X_1, \ldots, X_d)^T$, with margins $F_1, \ldots, F_d$.

Contrary to the multivariate Gaussian or Student-t copulas, vine copulas accurately model the dependence in high dimensions. They use the density functions of bivariate copulas (called pair-copula) to iteratively build a multivariate density function, which leads to a great flexibility in modeling the dependence.

`simshocks` applies inverse standard gaussian cumulative distribution function to the uniform margins of `CDVinesim` to obtains gaussian shocks, with various dependence structures between them.

`CDVineSim` can be used first, to choose the copula, and make an inference on it. Sometimes, the choice of the relevant copula is also made with expert knowledge.

2 Examples

The first section of examples, 2.1, is related to 1.3. That is, to generating correlated ϵ's from $dW(t) = \epsilon dt$ in (Eq. 7). Then, section 2.2 presents a complete example of simulation of a Stochastic Volatility Jump Diffusion (SVJD) model. This SVJD model uses 1.3.

2.1 Generating dependent shocks ϵ with `simshocks`

To use `simshocks`, you need the specify the number of simulations of ϵ that you need, `n`, the type of dependence, `family`, and additional parameters depending on the copula that you want to use. For a simulation of a Gaussian copula, `family` is `1` :

```r
library(ESGtoolkit)
```

```r
# Number of simulations
nb <- 1000

# Number of risk factors
d <- 2

# Number of possible combinations of the risk factors (here : 1)
dd <- d*(d-1)/2

# Family : Gaussian copula
fam1 <- rep(1, dd)

# Correlation coefficients between the risk factors (d*(d-1)/2)
par0_1 <- 0.1
par0_2 <- -0.9
```

A correlation coefficient is provided to simshocks through its argument par:

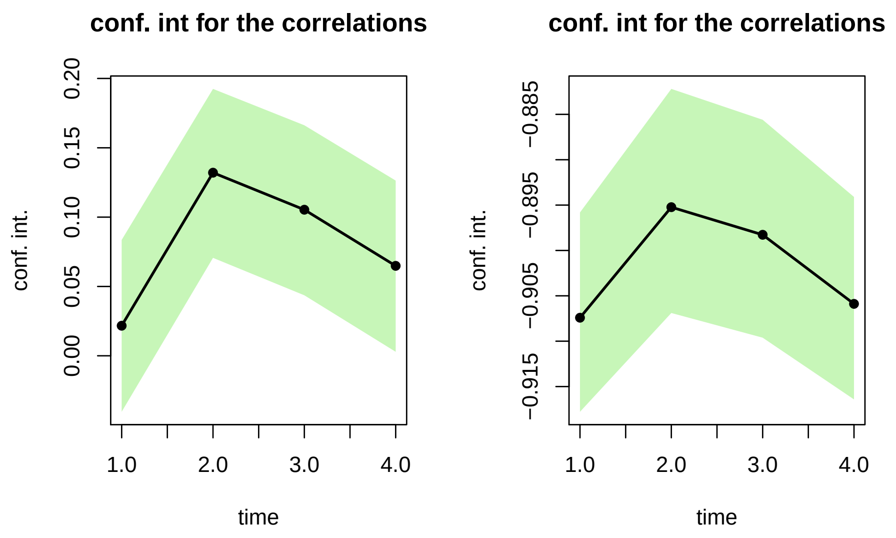
```
set.seed(2)

# Simulation of shocks for the d risk factors
s0_par1 <- simshocks(n = nb, horizon = 4,
family = fam1, par = par0_1)

s0_par2 <- simshocks(n = nb, horizon = 4,
family = fam1, par = par0_2)
```

You can make a correlation test on simshocks outputs with esgcortest, to assess whether correlation estimate between shocks is significantly close to the correlation you specified or not. If esgcortest's confidence interval contains the true value at a given confidence level, the null hypothesis is not to be rejected at this level.

```
# Correlation test
esgcortest(s0_par1)

## $cor.estimate
## Time Series:
## Start = 1
## End = 4
## Frequency = 1
## [1] 0.02168139 0.13205902 0.10534299 0.06487379
##
## $conf.int
## Time Series:
## Start = 1
## End = 4
## Frequency = 1
##        Series 1   Series 2
## 1 -0.040365943 0.08356216
## 2  0.070644283 0.19247635
## 3  0.043634863 0.16625037
## 4  0.002892336 0.12635869
```

These confidence intervals on the estimated correlations can also be visualized with esgplotbands:

```
test <- esgcortest(s0_par2)
par(mfrow=c(1, 2))
esgplotbands(esgcortest(s0_par1))
esgplotbands(test)
```

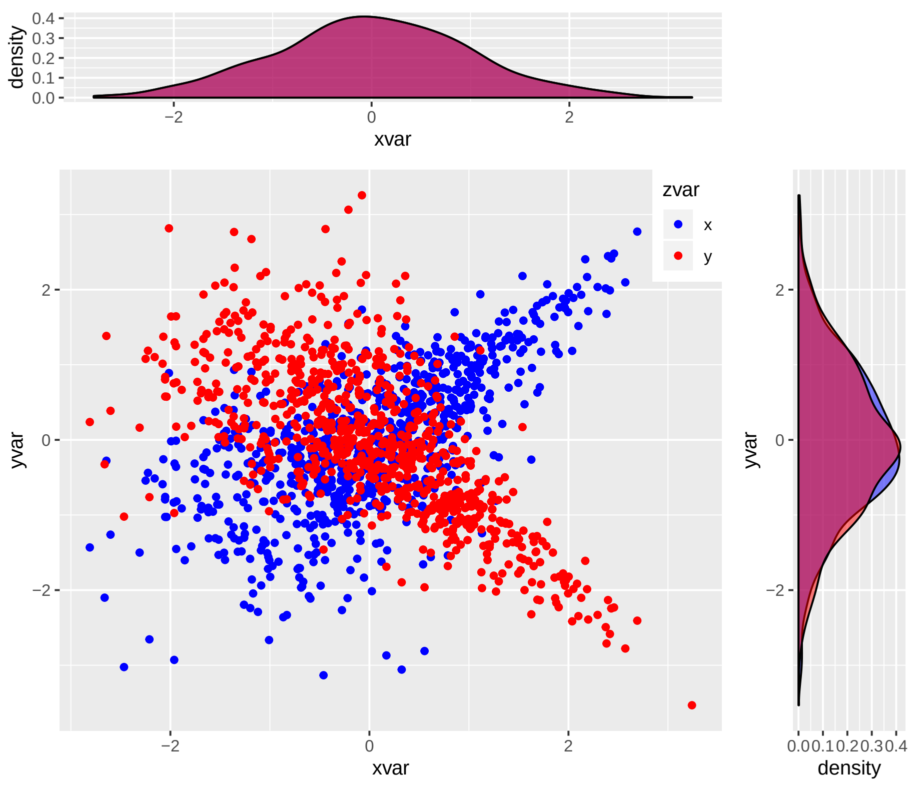


Now with other types of dependences, namely rotated versions of the Clayton copula :

```
# Family : Rotated Clayton (180 degrees)
fam2 <- 13
par0_3 <- 2

# Family : Rotated Clayton (90 degrees)
fam3 <- 23
par0_4 <- -2

# number of simulations
nb <- 200

# Simulation of shocks for the d risk factors
s0_par3 <- simshocks(n = nb, horizon = 4,
family = fam2, par = par0_3)

s0_par4 <- simshocks(n = nb, horizon = 4,
family = fam3, par = par0_4)
```

There's a nice function from the package, esgplotshocks, that helps you in visualizing dependence between shocks (inspired by this blog post) :

```
esgplotshocks(s0_par3, s0_par4)
```


density
0.4
0.3
0.2
0.1
0.0
−2
0
2
xvar
zvar
x
y
yvar
2
0
−2
−2
0
2
xvar
yvar
2
0
−2
0.00.10.20.30.4
density

2.2 Example with simdiff and simshocks : Option pricing under the Bates model (SVJD) for equity

SVJD stands for Stochastic Volatility with Jump Diffusion. In this model, the volatility of our asset's price evolves as a CIR process. The price itself is a Geometric Brownian motion between jumps, arising from a Poisson process. Here, we consider jumps with lognormal magnitude.

The model

$$\begin{aligned} dS(t) &= (r - \lambda\mu_Z)S(t)dt + \sqrt{v(t)}S(t)dW(t)^{(1)} + (J-1)dN(t) \\ dv(t) &= \kappa(\theta - v(t))dt + \sigma\sqrt{v(t)}dW(t)^{(2)} \\ dW(t)^{(1)}dW(t)^{(2)} &= \rho dt \end{aligned}$$

We use the package fOptions to compute options' prices from market implied volatility :

```
library(fOptions)
```

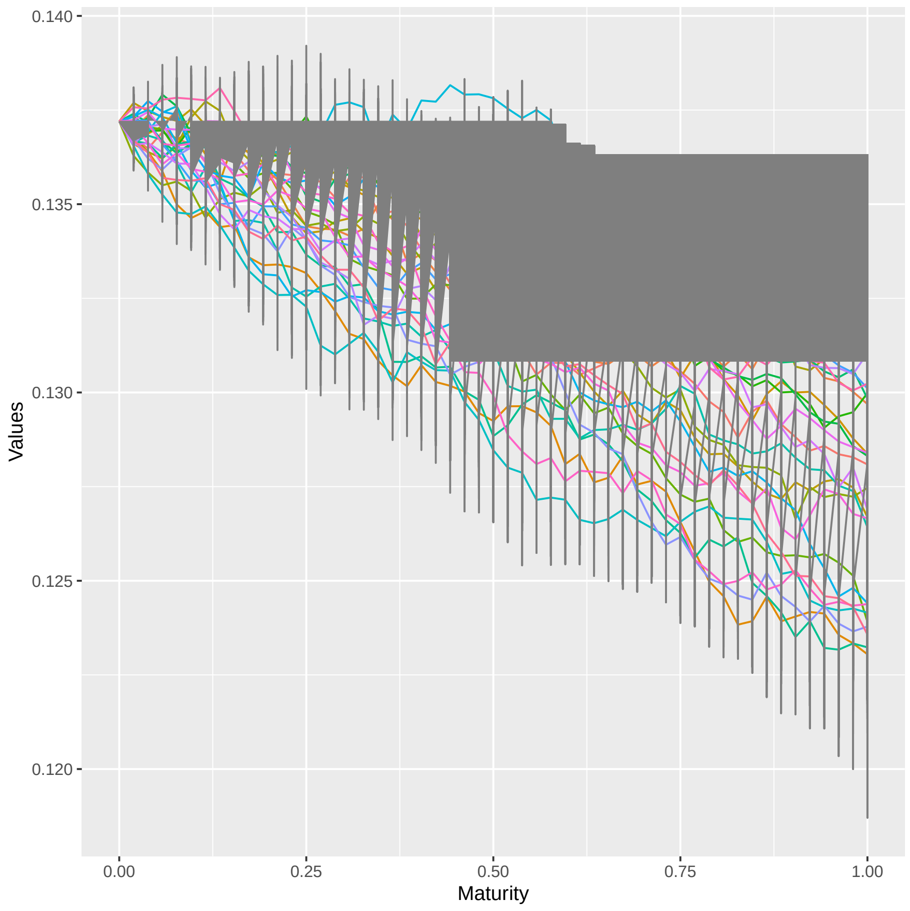
The parameters of Bates model are :

```
# Spot variance
V0 <- 0.1372
# mean-reversion speed
kappa <- 9.5110/100
# long-term variance
theta <- 0.0285
# volatility of volatility
volvol <- 0.8010/100
# Correlation between stoch. vol and prices
rho <- -0.5483
# Intensity of the Poisson process
lambda <- 0.3635
# mean and vol of the merton jumps diffusion
mu_J <- -0.2459
sigma_J <- 0.2547/100
m <- exp(mu_J + 0.5*(sigma_J^2)) - 1
# Initial stock price
S0 <- 4468.17
# Initial short rate
r0 <- 0.0357
```

Now we make 300 simulations of shocks and diffusions, on a weekly basis, from today, up to year 1. The shocks are simulated by using a variance reduction technique : antithetic variates (argument method).

```
n <- 300
horizon <- 1
```

```
freq <- "weekly"

# Simulation of shocks, with antithetic variates
shocks <- simshocks(n = n, horizon = horizon,
          frequency = freq,
          method = "anti",
          family = 1, par = rho)

# Vol simulation
sim_vol <- simdiff(n = n, horizon = horizon,
                   frequency = freq, model = "CIR", x0 = V0,
                   theta1 = kappa*theta, theta2 = kappa,
                   theta3 = volvol,
                    eps = shocks[[1]])

# Plotting the volatility (only for a low number of simulations)
esgplotts(sim_vol)
```

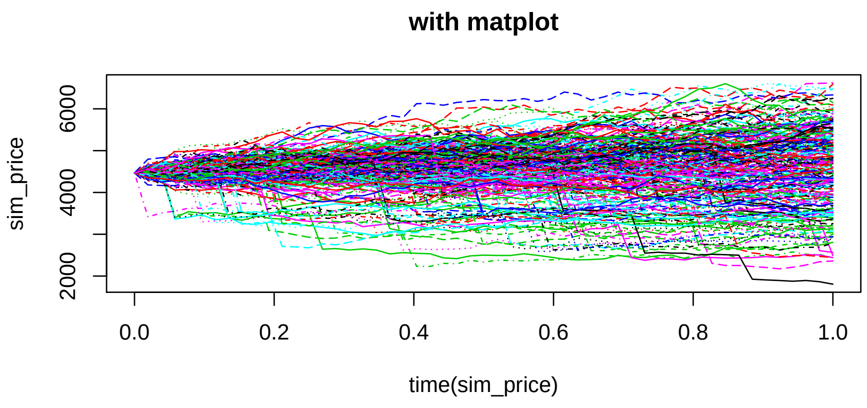


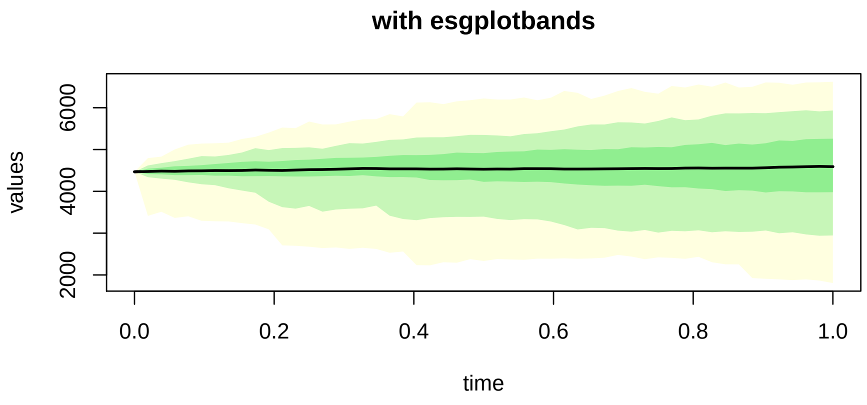
To finish, the simulation of asset price S takes **exactly the same parameters** `n, horizon, frequency` as `simshocks` and `simdiff`, and diffusion's volatility is embedded through `theta2`.

```
# prices simulation
sim_price <- simdiff(n = n, horizon = horizon,
                     frequency = freq, model = "GBM", x0 = S0,
                     theta1 = r0 - lambda*m, theta2 = sim_vol,
                     lambda = lambda, mu_z = mu_J,
                     sigma_z = sigma_J,
                     eps = shocks[[2]])
```

In the figure below, we can clearly see asset prices jumping with `matplot`. `esgplotbands`, which depicts asset paths by percentiles, will be more useful for thousands of simulations :

```
par(mfrow=c(2, 1))
matplot(time(sim_price), sim_price, type = 'l',
        main = "with matplot")
esgplotbands(sim_price, main = "with esgplotbands", xlab = "time",
             ylab = "values")
```





Now, we would like to study the convergence of estimated discounted prices to the initial asset price :

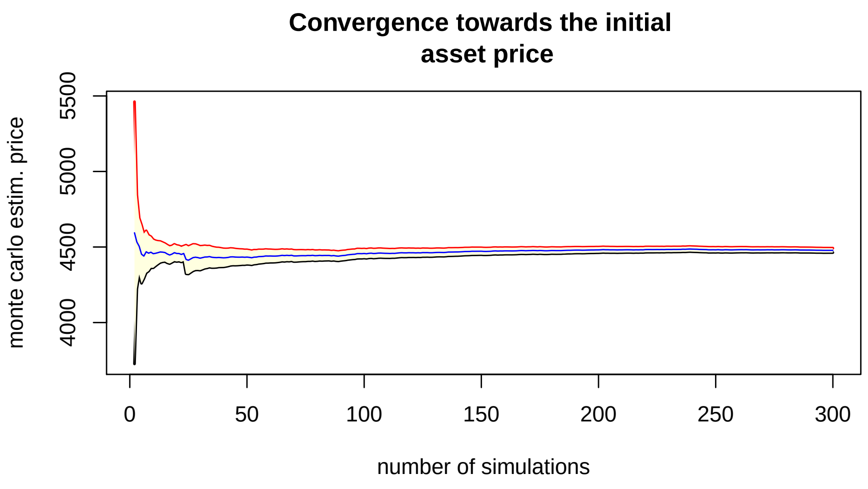
$$\frac{1}{N}\sum_{i=1}^{N} e^{-rT} S_T^{(i)} \longrightarrow \mathbb{E}[e^{-rT} S_T] = S_0 \tag{12}$$

where N is the number of simulations, r is the constant risk free rate, and T is a maturity of 2 weeks.

```
# Discounted Monte Carlo price
print(as.numeric(esgmcprices(r0, sim_price, 2/52)))

## [1] 4477.961

# Inital price
print(S0)

## [1] 4468.17

# pct. difference
print(as.numeric((esgmcprices(r0, sim_price, 2/52)/S0 - 1)*100))

## [1] 0.219129
```

One would also want to see how fast is the convergence towards $S0$:

```
# convergence of the discounted price
esgmccv(r0, sim_price, 2/52,
        main = "Convergence towards the initial \n asset price")
```



esgmcprices and esgmccv give information about the mean, but a statistical test gives more information.

```
martingaletest_sim_price <- esgmartingaletest(r = r0,
                                              X = sim_price,
                                              p0 = S0)
```

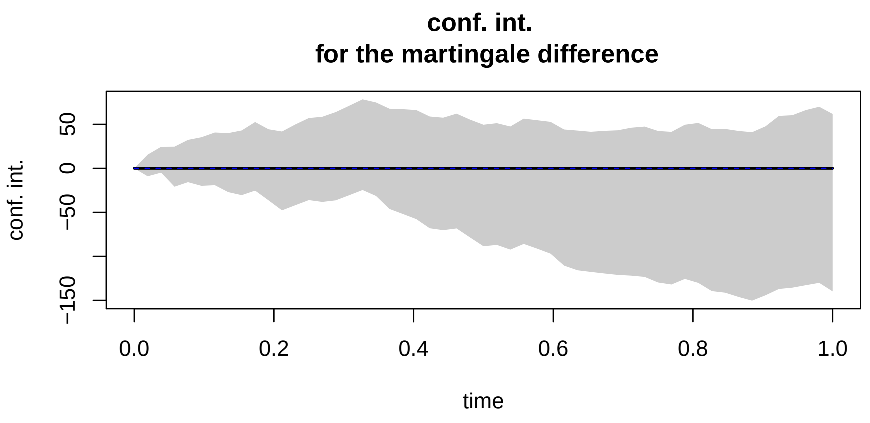
esgmartingaletest computes for each T, a Student's t-test of

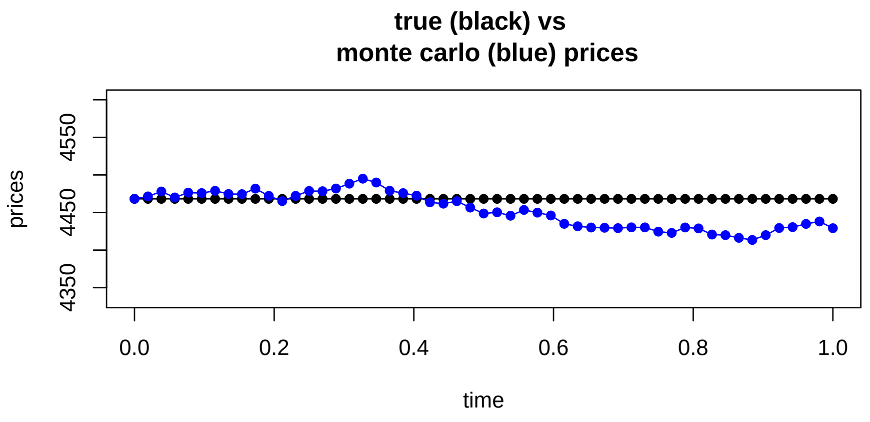
$$H_0 : \mathbb{E}[e^{-rT} S_T - S_0] = 0$$

versus the alternative hypothesis that the mean is not 0, at a given confidence level (default is 95%).

esgmartingaletest also provides p-values, and confidence intervals for the mean value (print martingaletest_sim_price!). If all the confidence intervals contain 0, then the null hypothesis is not rejected at the given level, let's say 95%. Which means that there are less than 5 chances out of 100 to be wrong by saying that the true mean of the distribution is 0. esgplotbands is a companion function to esgmartingaletest, that gives a visualization of confidence intervals and average discounted prices.

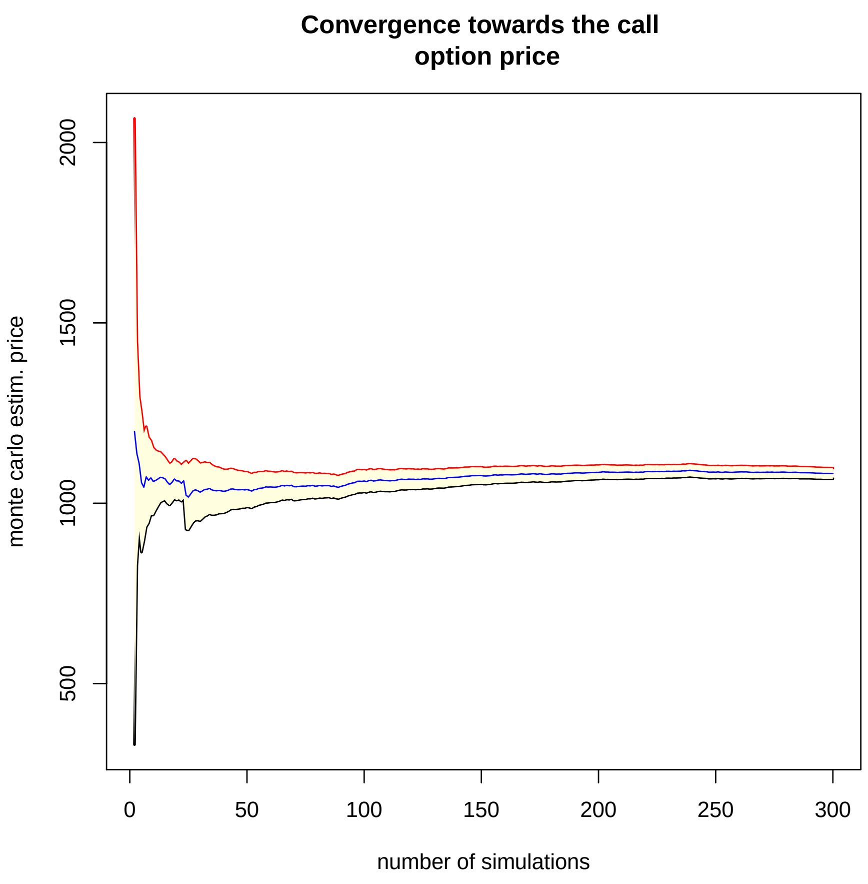
```
esgplotbands(martingaletest_sim_price)
```





Another interesting exercise is to price a Call Option under the Bates model described at the beginning of this section:

```
# Option pricing

# Strike
K <- 3400
Kts <- ts(matrix(K, nrow(sim_price), ncol(sim_price)),
               start = start(sim_price),
          deltat = deltat(sim_price),
          end = end(sim_price))

# Implied volatility
sigma_imp <- 0.6625

#Maturity
maturity <- 2/52

# payoff at maturity
payoff_ <- (sim_price - Kts)*(sim_price > Kts)
payoff <- window(payoff_,
             start = deltat(sim_price),
             deltat = deltat(sim_price),
             names = paste0("Series ", 1:n))

# True price
c0 <- GBSOption("c", S = S0, X = K, Time = maturity, r = r0,
                b = 0, sigma = sigma_imp)
print(c0@price)

## [1] 1069.945

# Monte Carlo price
print(as.numeric(esgmcprices(r = r0, X = payoff, maturity)))

## [1] 1082.626

# pct. difference
print(as.numeric((esgmcprices(r = r0, X = payoff,
             maturity = maturity)/c0@price - 1)*100))

## [1] 1.185204

# Convergence towards the option price
esgmccv(r = r0, X = payoff, maturity = maturity,
        main = "Convergence towards the call \n option price")
```


Convergence towards the call
option price
monte carlo estim. price
number of simulations
500
1000
1500
2000
0
50
100
150
200
250
300

References

Bates DS (1996). "Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options." *Review of financial studies*, **9**(1), 69–107.

Black F, Scholes M (1973). "The pricing of options and corporate liabilities." *The journal of political economy*, pp. 637–654.

Brechmann EC, Czado C (2012). "Risk management with high-dimensional vine copulas: An analysis of the Euro Stoxx 50."

Brechmann EC, Schepsmeier U (2013). "Modeling dependence with C-and D-vine copulas: The R-package CDVine." *Journal of Statistical Software*, **5**(3), 1–27.

Brigo D, Mercurio F (2006). *Interest rate models-theory and practice: with smile, inflation and credit*. Springer.

Cox JC, Ingersoll Jr JE, Ross SA (1985). "A theory of the term structure of interest rates." *Econometrica: Journal of the Econometric Society*, pp. 385–407.

Eddelbuettel D, François R (2011). "Rcpp: Seamless R and C++ integration." *Journal of Statistical Software*, **40**(8), 1–18.

Glasserman P (2004). *Monte Carlo methods in financial engineering*, volume 53. Springer.

Iacus SM (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer.

Kou SG (2002). "A jump-diffusion model for option pricing." *Management science*, **48**(8), 1086–1101.

Merton RC (1976). "Option pricing when underlying stock returns are discontinuous." *Journal of financial economics*, **3**(1), 125–144.

Uhlenbeck GE, Ornstein LS (1930). "On the theory of the Brownian motion." *Physical review*, **36**(5), 823.

Vasicek O (1977). "An equilibrium characterization of the term structure." *Journal of financial economics*, **5**(2), 177–188.

Wickham H (2009). *ggplot2: elegant graphics for data analysis*. Springer.