Package 'DynamicSimulation'

May 19, 2013

Version 0.1

Title Dymamic Simulation

Author P. Henaff

Depends fInstrument,timeSeries,Hmisc,xtable,empfin

Maintainer P. Henaff <henaff.iae@univ-paris1.fr>

Description Dynamic simulation

LazyLoad yes

LazyData yes

License GPL (>=2)

Collate 'DynamicSimulator.r''PathSimulator.r'

R topics documented:

deltaHedge 2
logNormal 4
makeTable 4
pathSimulator 5
rnormInnovations 6
sobolInnovations 6
sobolSeedInnovations 7
UnknownVolLogNormal 7

Index **8**

`deltaHedge` *Dynamic delta hedging*

Description

Delta hedging simulator

Usage

```
deltaHedge(instruments, env, params, trace = F)
```

Arguments

`instruments`	list of instruments to be hedged
`env`	a `DataProvider`
`params`	list of parameters that define the hedging policy: **dtSim [vector of timeDate]** simulation dates **dtFirst [timeDate]** first simulation date **dtLast [timeDate]** last simulation date **nbSteps [numeric]** number of simulation steps **transaction.cost [numeric]** proportional transaction cost for the underlying asset
`trace`	output debuging information?

Details

This function simulates a dynamic hedging strategy of a derivative or of a portfolio of derivatives, all function of the same underlying asset.

Value

a list with the following items:

wealth [matrix] residual wealth by time step and scenario

portfolio [matrix] value of hedge portfolio by time step and scenario

bond [matrix] quantity of zero-coupon bond held in the hedge portfolio by time step and scenario

price [matrix] price of derivative portfolio, by time step and scenario

stock [matrix] quantity of underlying asset in the hedge portfolio, by time step and scenario

description [string] description of hedging strategy

Examples

```
library(fInstrument)

dtExpiry <- mytDate('01jan2011')
dtStart <- mytDate('01jan2010')
nbSteps <- 100;
nbPaths <- 500;

dtSim <- timeSequence(dtStart, dtExpiry, length.out=nbSteps+1)
horizon <- tDiff(dtExpiry, dtStart)
delta.t <- horizon/nbSteps

sigma <- .3

underlying <- 'IBM'
K<-100
# define derivative
a <- fInstrumentFactory("vanilla", quantity=1,
                  params=list(cp='c', strike=K,
                    dtExpiry=dtExpiry,
                    underlying=underlying,
                    discountRef='USD.LIBOR', trace=FALSE))

# market data in default environment - basic PV calculation

base.env <- DataProvider()
setData(base.env, underlying, 'Price', dtStart, 100)
setData(base.env, underlying, 'DivYield', dtStart, .02)
setData(base.env, underlying, 'ATMVol', dtStart, sigma)
setData(base.env, underlying, 'discountRef', dtStart, 'USD.LIBOR')
setData(base.env, 'USD.LIBOR', 'Yield', dtStart, .02)

getValue(a, 'Price', dtStart, base.env)

# price paths
tSpot <- pathSimulator(dtSim = dtSim, nbPaths=nbPaths,
    innovations.gen=sobolInnovations, path.gen=logNormal,
    path.param = list(mu=0, sigma=sigma), S0=100, antithetic = FALSE,
    standardization = TRUE, trace = FALSE)

# derived environment for scenario analysis
sce.env <- DataProvider(parent=base.env)
setData(sce.env, underlying, 'Price',
        time(tSpot), as.matrix(tSpot))

# simulate a delta-hedge strategy along each path
assets = list(a)
res <- deltaHedge(assets, sce.env,
                  params=list(dtSim=time(tSpot),
                  transaction.cost=0), trace=FALSE)

x11()
```

```
hist(tail(res$wealth,1), 50, xlab="wealth",
     main=paste("distribution of wealth at expiry ", attr(a,'desc')))
```

`logNormal` *Log-normal path generator*

Description

Log-normal dynamic with constant drift and volatility

Usage

```
logNormal(S0, eps, delta.t, params)
```

Arguments

`S0`	[numeric] initial value
`eps`	[numeric] matrix of uniform (0,1) deviates
`delta.t`	[numeric] time step
`params`	[list] definition of process: **mu [numeric]** drift (annual rate) **sigma [numeric]** standard deviation (annual rate)

Value

matrix of paths, one path by column

`makeTable` *Dynamic delta hedging*

Description

Delta hedging simulator

Usage

```
makeTable(iScenario, res)
```

Arguments

`iScenario`	number of scenario to be displayed
`res`	result from a dynamic hedging simulation function, such as `deltaHedge`

Details

This function simulates a dynamic hedging strategy of a derivative or of a portfolio of derivatives, all function of the same underlying asset.

Value

an xtable with 6 columns:

time time step

stock price the price of the underlying asset

delta the delta of the option

option option value

bond pos zero-coupon bond position

hedge port. net value of hedge portfolio

`pathSimulator` *Path simulator*

Description

Function for generating saple paths, given a generator of uniform (0,1) sequences and the definition of the dynamic process

Usage

```
pathSimulator(dtSim = NULL, horizon = NULL,
  delta.t = NULL, nbPaths = NULL,
  innovations.gen = sobolInnovations,
  path.gen = logNormal, path.param, S0 = 100,
  antithetic = TRUE, standardization = FALSE,
  trace = FALSE)
```

Arguments

`dtSim`	[timeDate] vector of sampling dates
`horizon`	[numeric] simulation horizon, in fraction of years
`delta.t`	[numeric] time step
`nbPaths`	[numeric] number of paths to be generated
`innovations.gen`	[function] generator of uniform (0,1) random numbers
`path.gen`	[function] price dynamic
`path.param`	[numeric] list of parameters specific to `path.gen`
`S0`	[numeric] initial value

antithetic	[boolean] draw antithetic variates?
standardization	[boolean] recenter and normalize the output of innovation.gen?
trace	[boolean] output debugging information?

Value

time series of paths, one path per column

`rnormInnovations` *Generator of Random Matrices (rnorm)*

Description

Random matrix generator

Usage

```
rnormInnovations(nbPaths, pathLength)
```

Arguments

nbPaths	[numeric] number of paths
pathLength	[numeric] number of steps

Value

a matrix of normal(0,1) random numbers

`sobolInnovations` *Generator of Random Matrices (Sobol)*

Description

Random matrix generator

Usage

```
sobolInnovations(nbPaths, pathLength)
```

Arguments

nbPaths	[numeric] number of paths
pathLength	[numeric] number of steps

Value

a matrix of normal(0,1) random numbers

sobolSeedInnovations *Generator of Random Matrices (Sobol, fixed seed)*

Description

Random matrix generator

Usage

```
sobolSeedInnovations(nbPaths, pathLength)
```

Arguments

nbPaths	[numeric] number of paths
pathLength	[numeric] number of steps

Value

a matrix of normal(0,1) random numbers

UnknownVolLogNormal *Stochastic Volatility path generator*

Description

Log-normal dynamic with constant drift and stochastic volatility. volatility is drawn uniformly between sigmaMin and sigmaMax

Usage

```
UnknownVolLogNormal(S0, eps, delta.t, params)
```

Arguments

S0	[numeric] initial value
eps	[numeric] matrix of uniform (0,1) deviates
delta.t	[numeric] time step
params	[list] process definition:
	mu [numeric] drift (annual rate)
	sigmaMin [numeric] min standard deviation (annual rate)
	sigmaMax [numeric] max standard deviation (annual rate)

Value

matrix of paths, one path by column

Index

DataProvider, *2*
deltaHedge, 2, *4*

logNormal, 4

makeTable, 4

pathSimulator, 5

rnormInnovations, 6

sobolInnovations, 6
sobolSeedInnovations, 7

UnknownVolLogNormal, 7