Seminar 8

In this class we will:

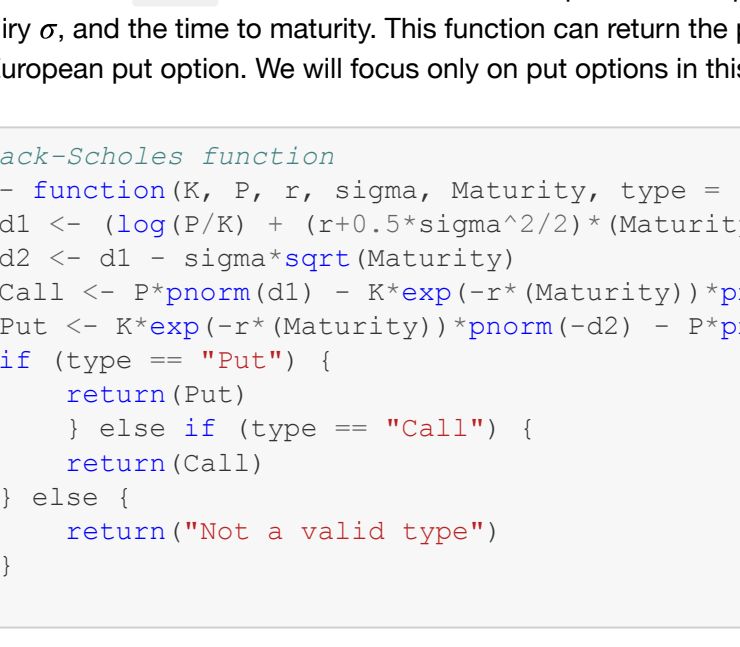
Using Black-Scholes to price an option

We will define a `bs()` function that takes a strike price, K, the price of the underlying asset P, the annual risk-free interest rate, r, annual volatiliry σ, and the time to maturity. This function can return the price of a European put or call option (put by default) will return the price of a European put option. We will focus only on put options in this section, but the process is analogous for call options:

In [86]:
```
# Black-Scholes function
bs <- function(K, P, r, sigma, Maturity, type = "Put"){
    d1 <- (log(P/K) + (r+0.5*sigma^2/2)*(Maturity))/(sigma*sqrt(Maturity))
    d2 <- d1 - sigma*sqrt(Maturity)
    Call <- P*pnorm(d1) - K*exp(-r*(Maturity))*pnorm(d2)
    Put <- K*exp(-r*(Maturity))*pnorm(-d2) - P*pnorm(-d1)
    if (type == "Put") {
        return(Put)
        } else if (type == "Call") {
        return(Call)
    } else {
        return("Not a valid type")
    }
}
```

We will work with the JP Morgan stock. Let's first load it into our environment and estimate its volatility:

In [87]:
```
# Load JPM stock
load("Y.RData")
y <- Y$JPM
plot(y, type = "l", main = "JPM stock")
```



We need to scale the standard deviation of the stock by $\sqrt{250}$ to get the estimated yearly volatility. We use 250 because it is the number of *trading days* in a year (Monday - Friday). Note that this is different from the *calendar days*, which are 365. The latter are used when dealing with the yearly risk-free interest rate.

In [88]:
```
# Estimating volatility with returns
n <- length(y)
sigma <- sd(y)*sqrt(250)
sigma
```

0.370395546800258

In order to use the Black-Scholes formula, we need the underlying price of the asset. We will load the `PRC.RData` file and get the last observation available for JP Morgan:

In [89]:
```
# Getting the last price available for JPM
load("PRC.RData")
P <- tail(PRC$JPM,1)
P
```

139.39999

We need to make assumptions on the risk-free rate and necessary parameters for an option. We will assume the risk-free rate is $r = 3\%$, the maturity is 1 year, and the strike price is $K = 100$:

In [90]:
```
# Assuming the necessary parameters
r <- 0.03
Maturity <- 1
K <- 100
```

Now we can use the `bs()` function we defined to price the option:

In [91]:
```
# Calculating the option price
put <- bs(K = K, P = P, r = r, sigma = sigma, Maturity = Maturity)
put
```

3.65571347437609

Option price as a function of the Underlying stock price

We have used the last price observed to calculate the price of the option.

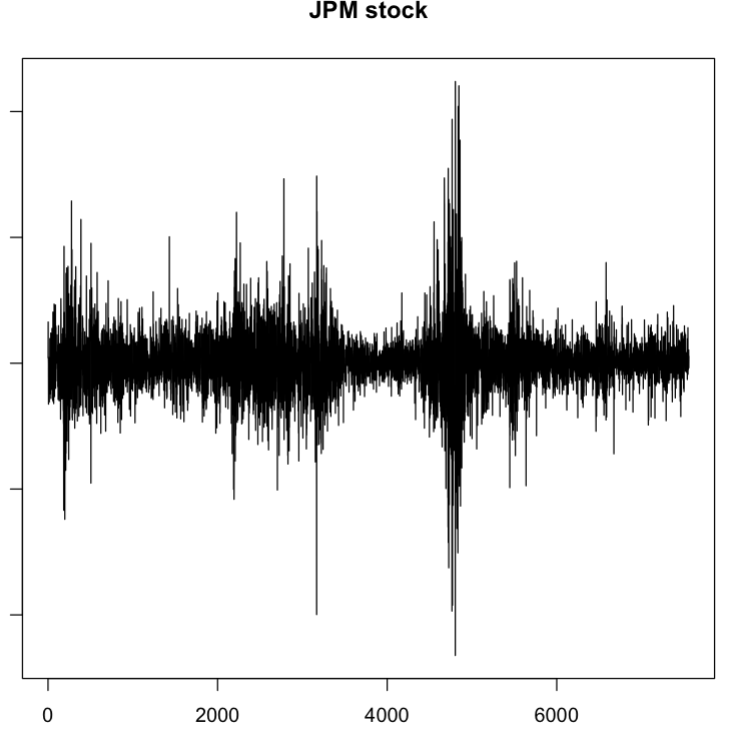
However, we can do a plot of the option price as a function of the underlying stock price. To do so, we will create a variable called `x` that will contain a sequence of values between 50 and 150. You can think of this variable as the x-axis of the plot. These are all the possible underlying prices we want will consider. We will use the function `seq()` with a length of 1000. Then, we will call the option `bs()` with the variable `x` as the argument `P`, for underlying price, and the result will be a vector of length 1000, with the option price for each value in the sequence `x`. Finally, we plot them together:

In [92]:
```
# Option price as a function of underlying stock price

# Sequence of underlying prices
x <- seq(50, 150, length = 1000)

# Plot option price
plot(x, bs(K = K, P = x, r = r, sigma = sigma, Maturity = Maturity),
    type = "l", ylab = "Option", xlab = "Underlying price", lwd = 3, col = "pink", las = 1,
    bty = "l", main = "Price of put option depending on underlying stock price")

# Add a point for the observed price
points(P, bs(K=K, P=P, r=r, sigma=sigma, Maturity = Maturity),
       pch = 16, col="blue")
```



We can see the put price is decreasing on the underlying price, and convex. We repeat the same procedure using a sequence for the Strike price, risk-free rate, volatility, and time to maturity:

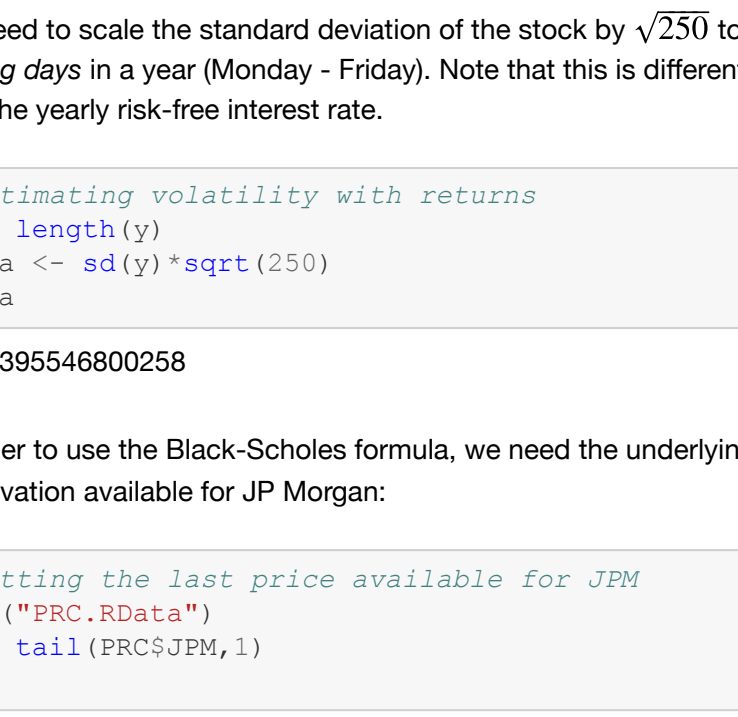
Option price as a function of the Strike price

In [93]:
```
# Option price as a function of Strike price

# Sequence of Strike
x <- seq(50, 150, length = 1000)

# Plot option price
plot(x, bs(K = x, P = P, r = r, sigma = sigma, Maturity = Maturity),
    type = "l", ylab = "Option", xlab = "Strike", lwd = 3, col = "pink", las = 1,
    bty = "l", main = "Price of put option depending on Strike price")

# Add a point for the observed price
points(K, bs(K=K, P=P, r=r, sigma=sigma, Maturity = Maturity),
       pch = 16, col="blue")
```



The price of the put option is increasing and convex in the Strike price.

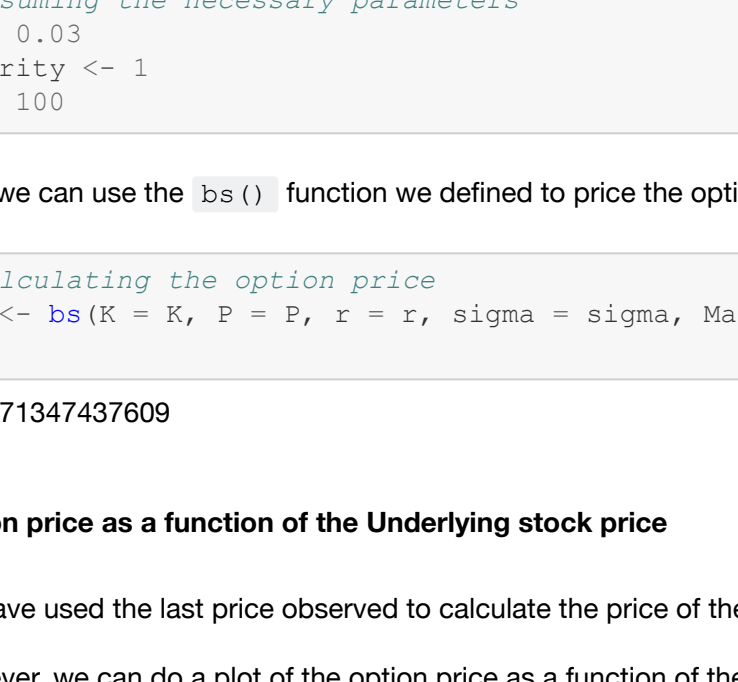
Option price as a function of the Risk-free rate

In [94]:
```
# Option price as a function of risk-free rate

# Sequence of risk-free
x <- seq(0.01, 0.2, length = 1000)

# Plot option price
plot(x, bs(K = K, P = P, r = x, sigma = sigma, Maturity = Maturity),
    type = "l", ylab = "Option", xlab = "Risk-free", lwd = 3, col = "pink", las = 1,
    bty = "l", main = "Price of put option depending on risk-free rate")

# Add a point for the observed price
points(r, bs(K=K, P=P, r=r, sigma=sigma, Maturity = Maturity),
       pch = 16, col="blue")
```



The price of the put option is decreasing and convex in the risk-free rate.

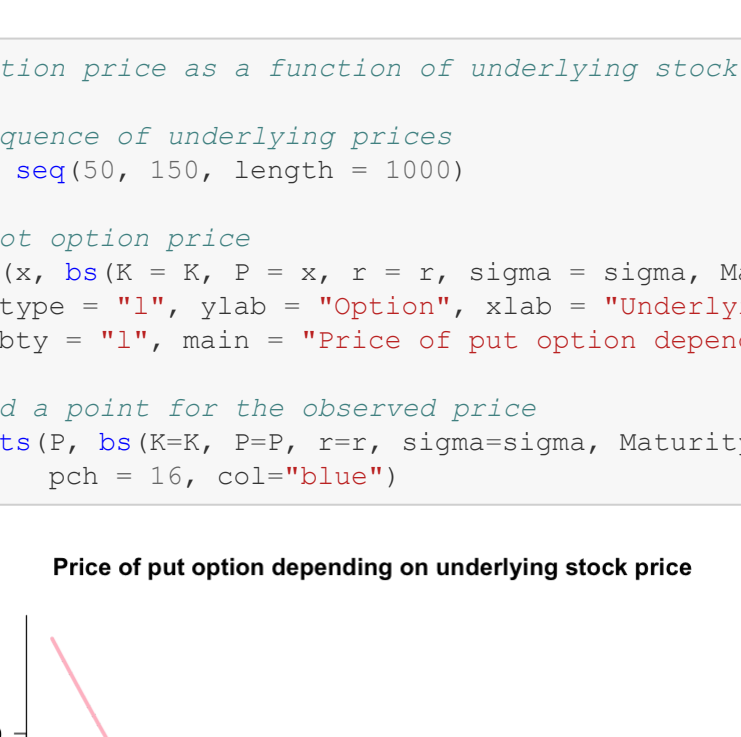
Option price as a function of volatility

In [95]:
```
# Option price as a function of sigma

# Sequence of sigma
x <- seq(0.01, 0.4, length = 1000)

# Plot option price
plot(x, bs(K = K, P = P, r = r, sigma = x, Maturity = Maturity),
    type = "l", ylab = "Option", xlab = "Sigma", lwd = 3, col = "pink", las = 1,
    bty = "l", main = "Price of put option depending on sigma")

# Add a point for the observed price
points(sigma, bs(K=K, P=P, r=r, sigma=sigma, Maturity = Maturity),
       pch = 16, col="blue")
```



The price of the option is increasing and convex in sigma, with the presence of exponential growth.

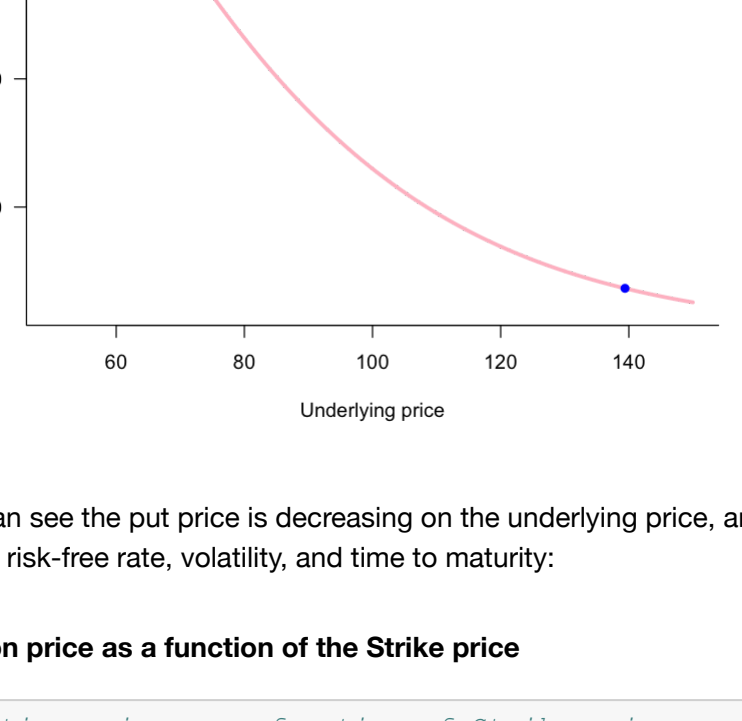
Option price as a function of Maturity

In [96]:
```
# Option price as a function of Maturity

# Sequence of maturity
x <- seq(0.1, 10, length = 1000)

# Plot option price
plot(x, bs(K = K, P = P, r = r, sigma = sigma, Maturity = x),
    type = "l", ylab = "Option", xlab = "Maturity", lwd = 3, col = "pink", las = 1,
    bty = "l", main = "Price of put option depending on maturity")

# Add a point for the observed price
points(Maturity, bs(K=K, P=P, r=r, sigma=sigma, Maturity = Maturity),
       pch = 16, col="blue")
```



The option price is increasing and concave in maturity.

Using Monte-Carlo simulation to price an option

We can use a random number generator to simulate the price of an option:

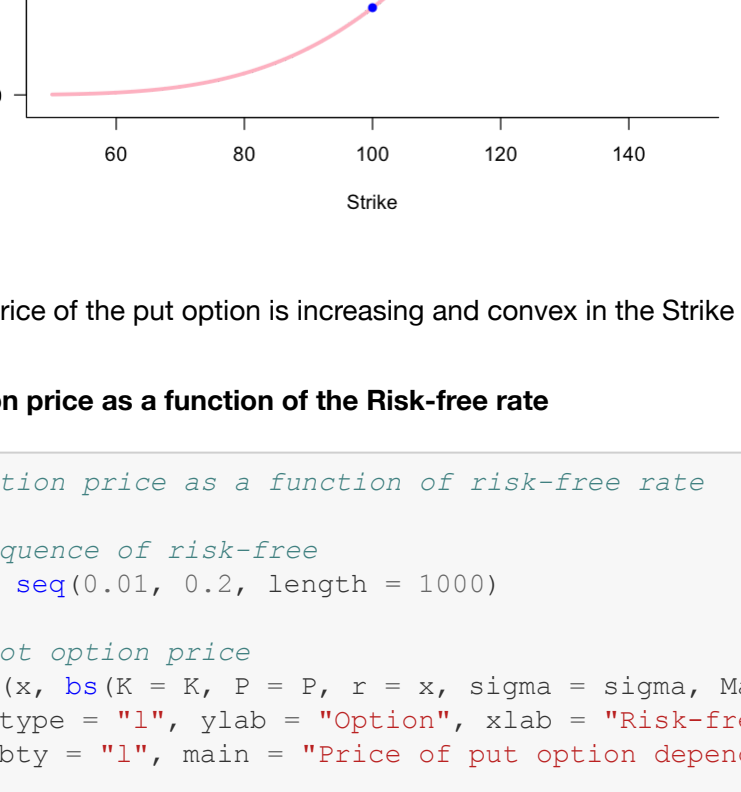
In [97]:
```
# Using Monte Carlo

# Simulations
S <- 1e6

# Set seed for replicability
set.seed(420)

# Get 1e6 simulations from a standard normal
ysim <- rnorm(S)

# Plot the random numbers
plot(ysim)
hist(ysim)
```



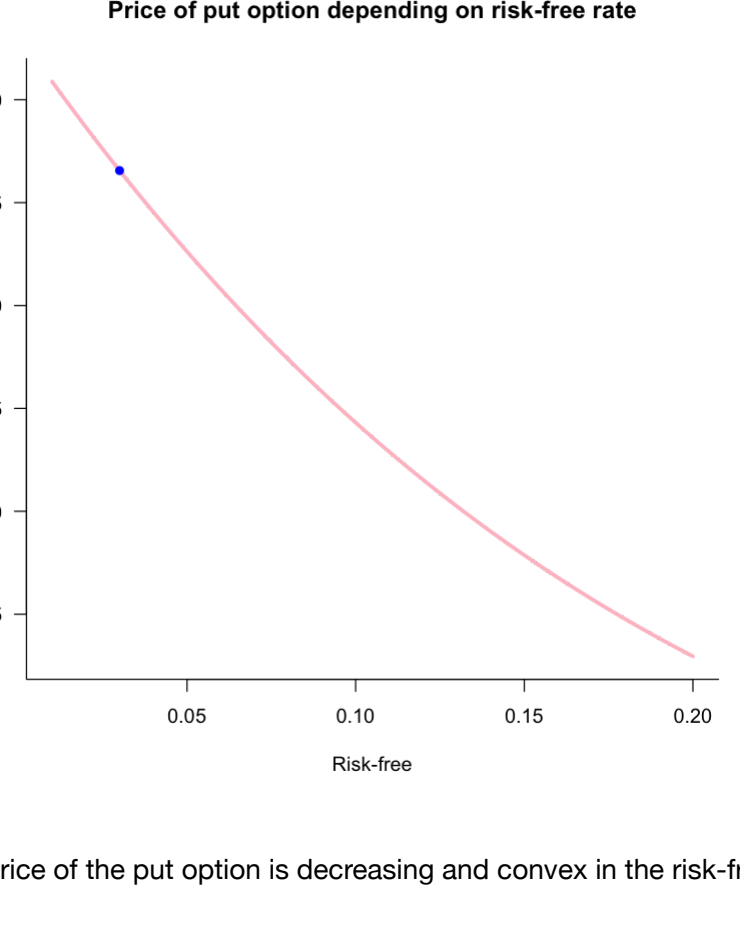
Remember we need to apply a **log-normal correction** in the simulation. To do so, we subtract $\frac{1}{2}\sigma^2$ from the simulated stock returns. By default, `rnorm()` uses a standard normal, we would also need to scale for the desired variance:

In [98]:
```
# Centering in the desired mean and changing the variance
ysim <- ysim - 0.5*sigma^2*Maturity
ysim <- ysim*sigma*sqrt(Maturity)
```

We can directly specify this in `rnorm()` to account for the variance and log-normal correction:

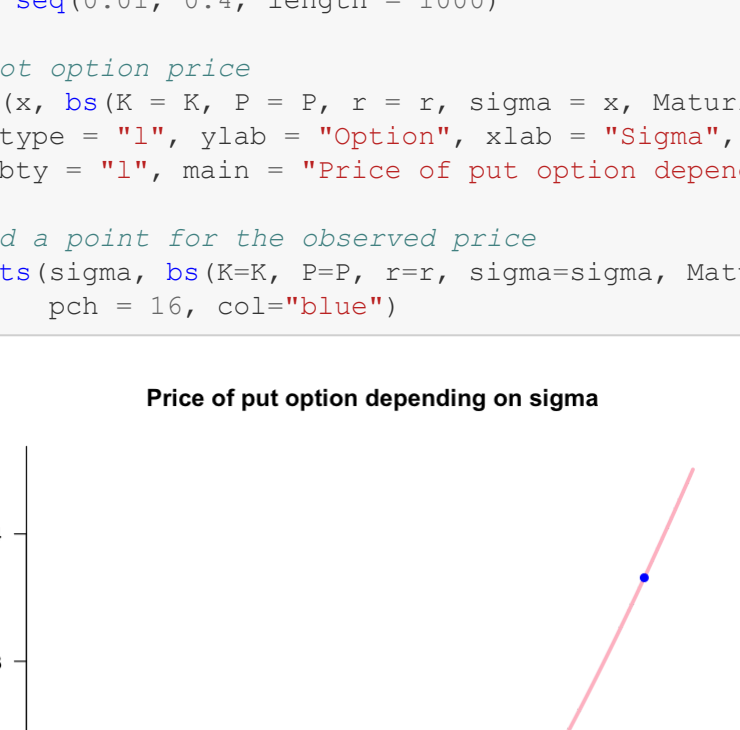
In [99]:
```
# Specifying mean/sd in rnorm directly
set.seed(420)
ysim <- rnorm(S, mean=-0.5*sigma^2*Maturity, sd = sigma*sqrt(Maturity))

# Histogram
hist(ysim, probability = TRUE)
```



Now we can simulate the no-arbitrage future prices of the stock:

In [100]:
```
# Simulating future prices of stock
Fsim <- P*exp(r*Maturity)*exp(ysim)
hist(Fsim, probability = TRUE)
```

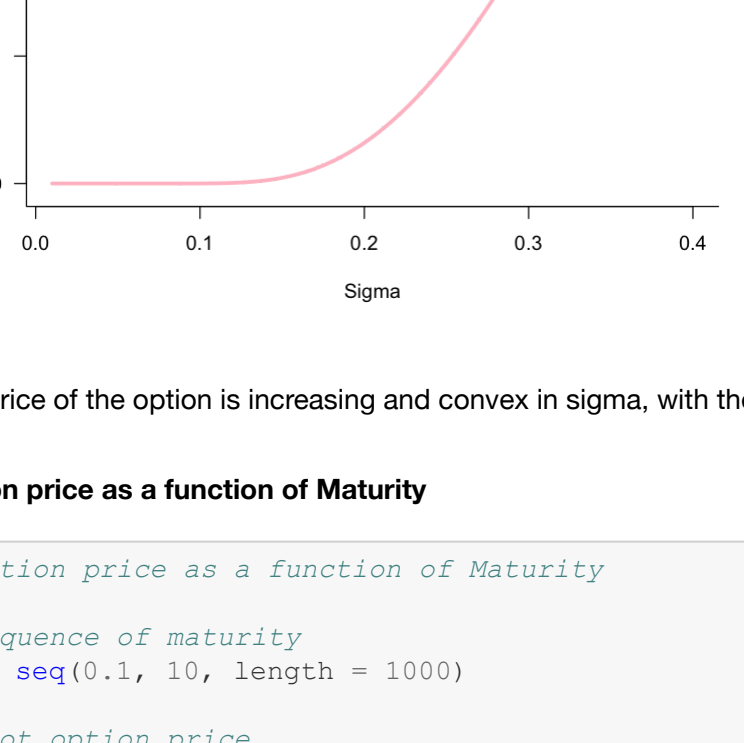


To simulate the price of the put option, we subtract the simulated prices of the underlying stock from the strike price, and replace it with 0 if negative, since the option would not be executed:

In [101]:
```
# Simulated price of stock
Psim <- K - Fsim

# Replacing negatives with zero
Psim[Psim<0] <- 0

# Histogram
hist(Psim, probability = TRUE)
```



The histogram has a peak in 0 due to all the cases when the put option is not executed. Now let's compare the price we obtained analytically using the Black-Scholes formula and the price obtained through Simulation:

In [102]:
```
# Comparing analytical and simulation prices
Psim <- Psim*exp(-r*Maturity)

cat("Analytic price:", put, "\n",
    "Simulation price:", mean(Psim))
```

```
Analytic price: 3.655713
 Simulation price: 3.68599
```

Experimenting using Simulation sizes

We can repeat the experiment as a function of the simulation size:

In [105]:
```
# Experimenting using Simulation sizes
# For a given simulation size S, say S = 100
S <- 100

# Set the seed, e.g. 100
set.seed(100)

# Do the simulation
ysim <- rnorm(S,
             mean = -0.5*sigma^2*Maturity,
             sd = sigma*sqrt(Maturity))

# Get the simulated paths
Fsim <- P*exp(r*Maturity)*exp(ysim)

# Subtract from strike price and only exercise the option if its positive
Psim <- K - Fsim
Psim[Psim<0] <- 0
Psim <- Psim * exp(-r*Maturity)

round(mean(Psim),4)
```

3.702

Now let's create a function `Sim()` that does these steps for us:

In [106]:
```
# Function on simulation size and seed
Sim <- function(S, seed = 999, message = TRUE){
    set.seed(seed)
    ysim <- rnorm(S,
                 mean = -0.5*sigma^2*Maturity,
                 sd = sigma*sqrt(Maturity))
    Fsim <- P*exp(r*Maturity)*exp(ysim)
    Psim <- K-Fsim
    Psim[Psim<0] <- 0
    Psim <- Psim*exp(-r*Maturity)
    if (message) {
        cat("**************************", "\n",
            "Number of simulations:", S, "\n",
           "Using seed:", seed, "\n",
           "Simulated price:", mean(Psim),"\n",
           "\n")
    }
    return(mean(Psim))
}
```

Let's see how the simulated price evolves with simulation size for our stock:

In [107]:
```
# Different simulation sizes
x <- c(10, 100, 1000, 10000, 100000, 1000000)
for(size in x){
    Sim(size, seed = 420)
}
```

```
**************************
 Number of simulations: 10
 Using seed: 420
 Simulated price: 5.842383

**************************
 Number of simulations: 100
 Using seed: 420
 Simulated price: 5.005495

**************************
 Number of simulations: 1000
 Using seed: 420
 Simulated price: 3.667088

**************************
 Number of simulations: 10000
 Using seed: 420
 Simulated price: 3.500996

**************************
 Number of simulations: 1e+05
 Using seed: 420
 Simulated price: 3.659979

**************************
 Number of simulations: 1e+06
 Using seed: 420
 Simulated price: 3.68599
```

We can make a plot to visually see how we converge to the analytic price as simulation size increases:

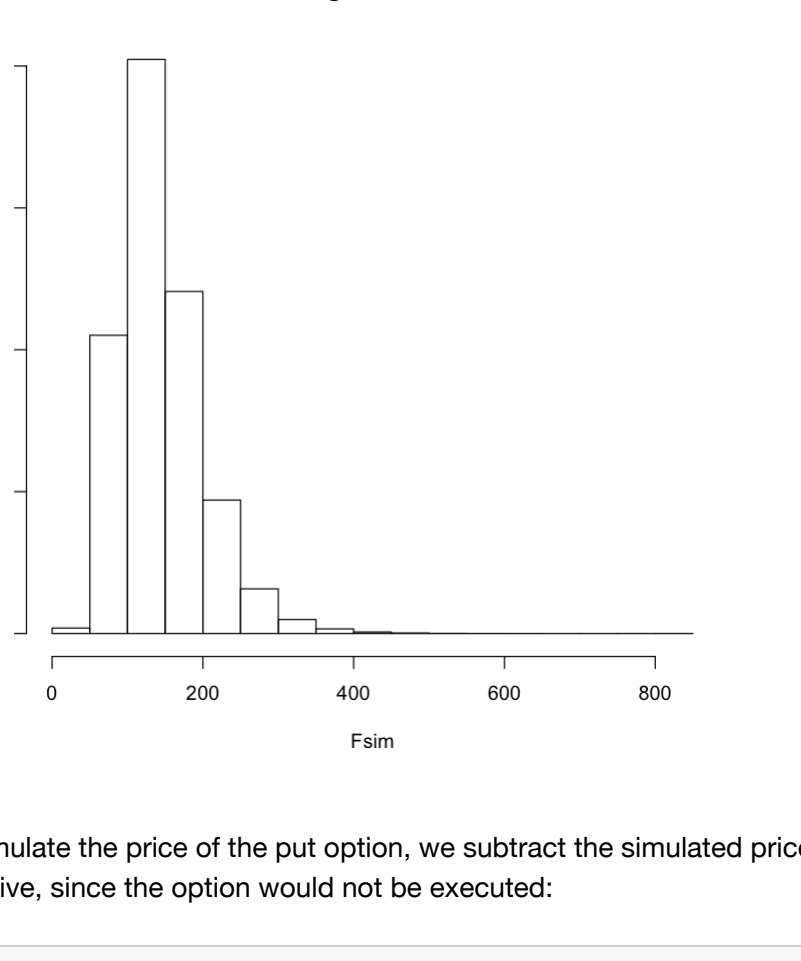
In [108]:
```
# Plot converge of price
seed <- 420

# Sequence of simulations
x <- seq(100, 100000, length = 100)

# Simulation price of each
P_sim <- sapply(x, Sim, seed = seed, message = FALSE)

# Plot
plot(x, P_sim, type = "l", main = "Simulated price for option", sub = paste0("Seed: ", seed),
    xlab = "Simulation Size", ylab = "Price", las = 1)

# Add horizontal line on Analytic price
abline(h = put, col = "red")
axis(2, at = put, label = round(put,2), las = 1)
```



We can look at other seeds:

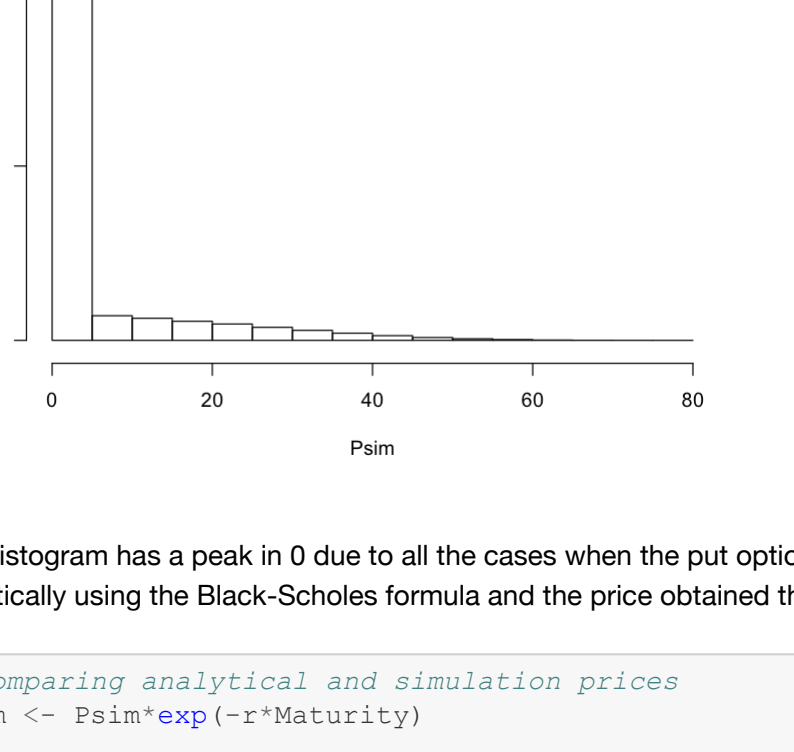
In [109]:
```
# Different seeds

seed <- 666

# Simulation price of each
P_sim <- sapply(x, Sim, seed = seed, message = FALSE)

# Plot
plot(x, P_sim, type = "l", main = "Simulated price for option", sub = paste0("Seed: ", seed),
    xlab = "Simulation Size", ylab = "Price", las = 1)

# Add horizontal line on Analytic price
abline(h = put, col = "red")
axis(2, at = put, label = round(put,2), las = 1)
```

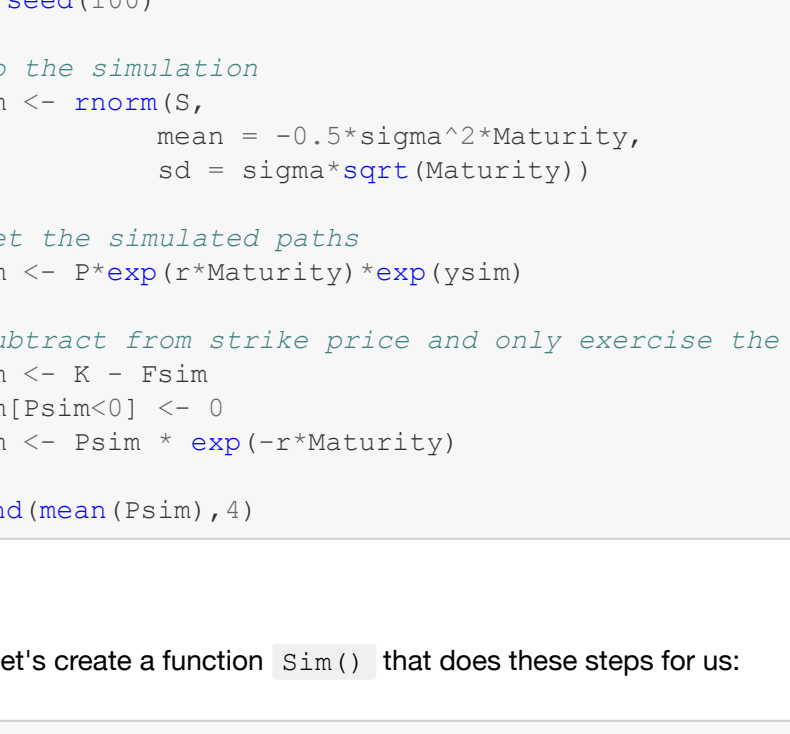


In [110]:
```
seed <- sample(100:999,1)

# Simulation price of each
P_sim <- sapply(x, Sim, seed = seed, message = FALSE)

# Plot
plot(x, P_sim, type = "l", main = "Simulated price for option", sub = paste0("Seed: ", seed),
    xlab = "Simulation Size", ylab = "Price", las = 1)

# Add horizontal line on Analytic price
abline(h = put, col = "red")
axis(2, at = put, label = round(put,2), las = 1)
```



Analytical and simulations method for VaR

Let's first find the VaR of the stock using a simulation:

In [111]:
```
# Simulating arithmetic returns

# Specifying VaR probability
probability <- 0.01

# Sample standard deviation
sigma <- sd(y)

# Number of simulations
S <- 1e3

# Set a seed
seed <- 456
set.seed(seed)

# Draw from a normal distribution, using daily rate and standard deviation
ysim <- rnorm(S, mean = r/365, sd = sigma)

# Simulate prices
Psim <- P*(1+ysim)

# Sort the simulated profit/loss
q <- sort(Psim - P)

# VaR from quantile
VaR1s <- -q[ceiling(probability*S)]

# Round
round(VaR1s,2)
```

7.47

We can compare it with the analytical VaR:

In [112]:
```
# Analytical VaR
VaR1t <- abs(sigma*qnorm(probability)*P)
round(VaR1t,2)
```

7.6

Larger simulation size

Instead of using 1000 simulations, we can see if a larger number gets us closer to the analytical price:

In [113]:

```r
# Function that computes simulation VaR for a number of simulations and seed
simVaR <- function(S, seed = 999, message = TRUE) {
    # Include elapsed time
    old <- Sys.time()

    set.seed(seed)
    ysim <- rnorm(S, mean = r/365, sd = sigma)
    Psim <- P*(1+ysim)
    q <- sort(Psim - P)
    VaR1s <- -q[ceiling(probability*S)]
    if (message) {
        cat("Number of simulations", S, "\n")
        cat("Elapsed time:", difftime(Sys.time(), old, units = "secs"), "seconds")
    }

    return(VaR1s)
}

round(simVaR(S = 1e4, seed = seed),2)
round(simVaR(S = 1e6, seed = seed),2)
```

```
Number of simulations 10000
Elapsed time: 0.002403021 seconds
```

7.37

```
Number of simulations 1e+06
Elapsed time: 0.131144 seconds
```

7.6

Creating a plot:

In [114]:

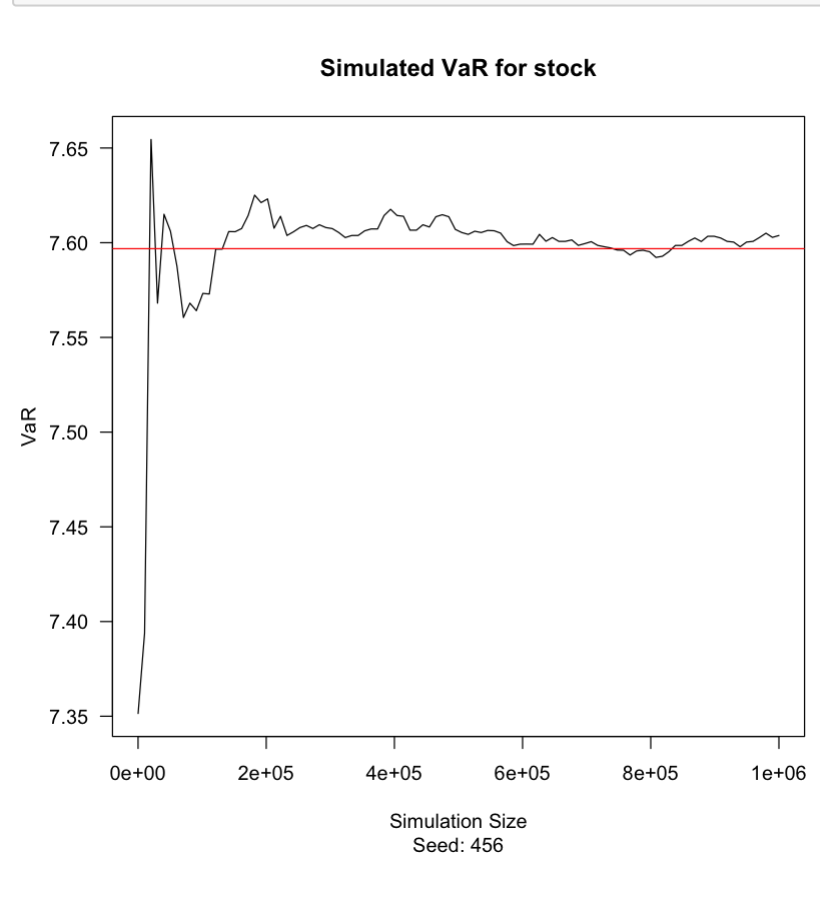
```r
# Plotting

# Sequence of simulations
x <- seq(100, 1000000, length = 100)

# Simulation VaR of each
VaR_sim <- sapply(x, simVaR, seed = seed, message = FALSE)

# Plot
plot(x, VaR_sim, type = "l", main = "Simulated VaR for stock", sub = paste0("Seed: ", seed),
    xlab = "Simulation Size", ylab = "VaR", las = 1)

# Add horizontal line on Analytic VaR
abline(h = VaR1t, col = "red")
```



Calculating VaR for the option

To calculate the VaR for the option we can apply the Black-Scholes formula using the observed price and the simulated prices, and a sigma scaled for the yearly trading days:

In [115]:

```r
# VaR for the option

# Calculate put
put <- bs(K = K, P = P, r = r, sigma = sqrt(sigma^2 * 250), Maturity = Maturity)

# Simulate prices
S <- 1e3
set.seed(444)
ysim <- rnorm(S, mean=r/365, sd=sigma)
Psim <- P*(1+ysim)

# Use simulated prices in Black-Scholes formula
fsim <- bs(K=K, P=Psim, r=r, sigma=sqrt(sigma^2 *250), Maturity = Maturity-(1/365))

# Sort the vector
q <- sort(fsim - put)

# Get the value that meets the VaR probability
VaR2 <- -q[ceiling(probability*S)]
round(VaR2,3)
```

0.824

Calculating VaR for a portfolio of stock and option

If we have a portfolio that has both an option and the underlying stock, we can calculate VaR as follows:

In [116]:

```r
# VaR for a portfolio

# Assets
xb <- 3
xo <- 2

# Create portfolio
portfolio_today <- xb*P + xo*put
round(portfolio_today)
```

426

Use the vector of simulated prices, `Psim`, to calculate tomorrow's value of the portfolio:

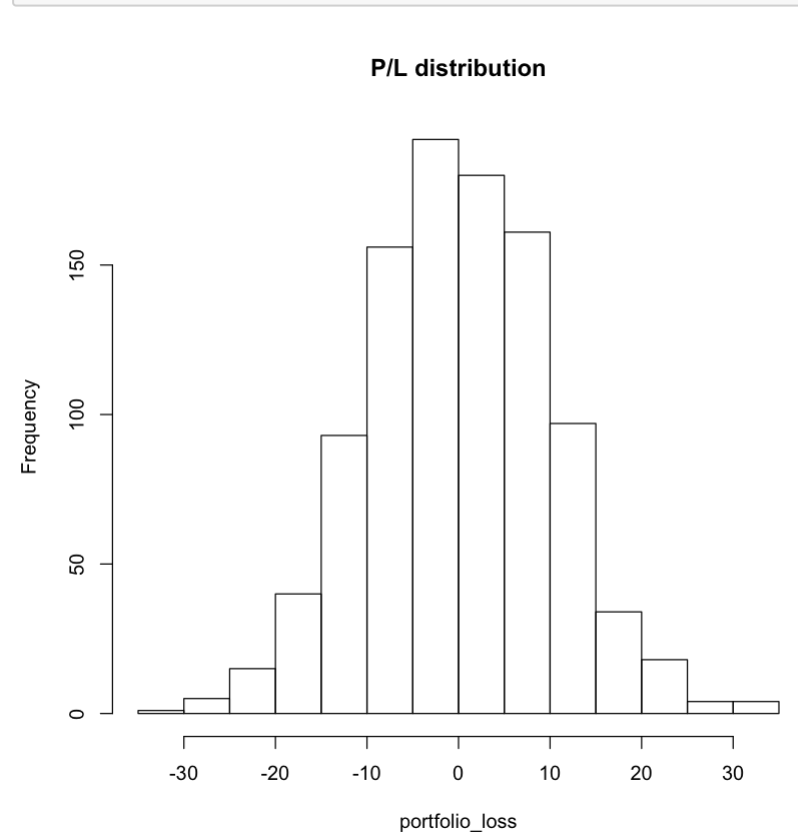
In [117]:

```r
# Calculating tomorrow's value of the portfolio
option_tomorrow <- bs(K = K, P = P, r = r, sigma = sqrt(sigma^2 * 250), Maturity = Maturity - (1/365))
portfolio_tomorrow <- xb*Psim + xo*option_tomorrow
```

In [118]:

```r
# Find the profit-loss
portfolio_loss <- portfolio_tomorrow - portfolio_today

# We can see its distribution
hist(portfolio_loss, main = "P/L distribution")
```



In [119]:

```r
# Find VaR
portfolio_loss <- sort(portfolio_loss)
portfolio_VaR <- -portfolio_loss[ceiling(S*probability)]
round(portfolio_VaR,2)
```

23.09

We can include a new option, with a different strike price, and see how it changes our VaR:

In [120]:

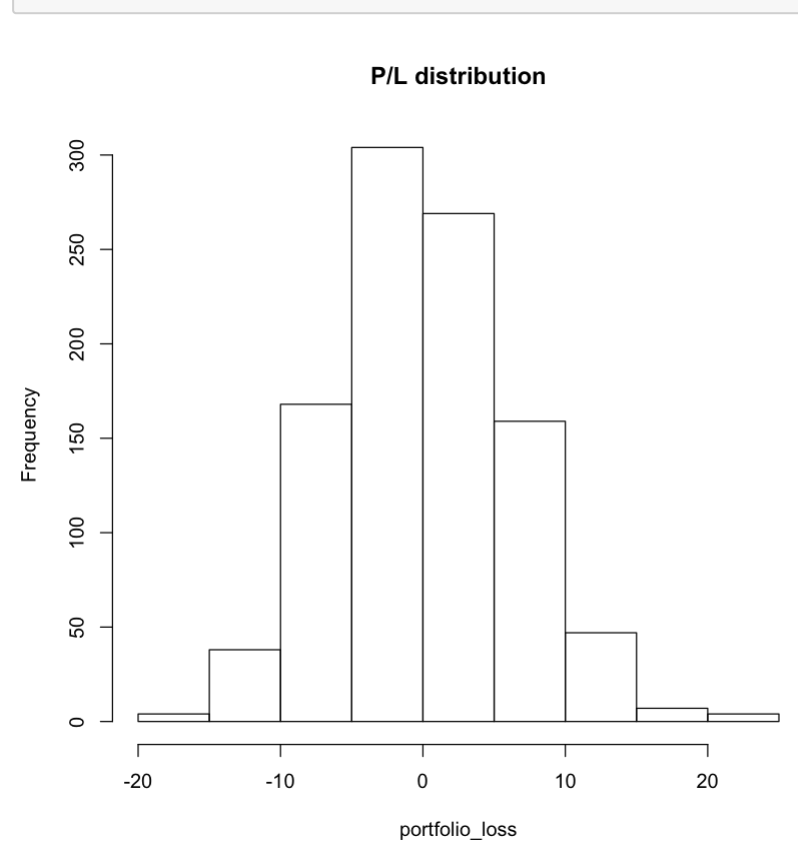
```r
# Introducing a second option
xo2 <- 15
K2 <- 90

# Get the prices using Black-Scholes
option2_today <- bs(K=K2, P=P, r=r, sigma=sqrt(sigma^2 *250), Maturity=Maturity)
option2_tomorrow <- bs(K=K2, P=Psim, r=r, sigma=sqrt(sigma^2 * 250), Maturity=Maturity-(1/365))

# Portfolio values
portfolio_today <- xb*P + xo*put + xo2*option2_today
portfolio_tomorrow <- xb*Psim + xo*option_tomorrow + xo2*option2_tomorrow

# Find the profit-loss
portfolio_loss <- portfolio_tomorrow - portfolio_today

# We can see its distribution
hist(portfolio_loss, main = "P/L distribution")
```



In [121]:

```r
# Find VaR
portfolio_loss <- sort(portfolio_loss)
portfolio_VaR <- -portfolio_loss[ceiling(S*probability)]
round(portfolio_VaR,2)
```

13.48

Recap

In this seminar we have covered:

- Pricing a put option analytically with Black-Scholes
- Simulating option prices with Monte Carlo
- Evaluating convergence to analytical solutions for different simulation sizes
- Stock, option and portfolio VaR calculations

Some new functions used:

- `bs()`
- `Sim()`
- `SimVaR()`

For more discussion on the material covered in this seminar, refer to *Chapter 6: Analytical value-at-risk for options and bonds* and *Chapter 7: Simulation methods for VaR for options and bonds* on *Financial Risk Forecasting* by Jon Danielsson.

Acknowledgements: Thanks to Alvaro Aguirre for creating these notebooks
© Jon Danielsson, 2020