The purpose of this document is to introduce the very basic elements you need to complete your project. These elements are: programming a function in R, optimizing this function with respect to one or several arguments, estimate a time series model and compute option prices from a time series model. The last questions are an extra mile for those of you that manage to finish the computer class earlier.

Part 1

1 Basics of R

1.1 Syntax for functions

```r
f <- function(input) {
  ... commands
  ... return(output)
}
```

Example: BS option pricer

```r
bs <- function(sigma, T, K, S, r) {
  d1 = log(S / (K * exp(-r * T))) / (sigma * sqrt(T)) + 1 / 2 * sigma * sqrt(T)
  d2 = d1 - sigma * sqrt(T)
  P = S * pnorm(d1) - K * exp(-r * T) * pnorm(d2)
  return(P)
}
```

Basically, functions are to be jointly used with a script. For the BS example:

```r
# Initial settings
rm(list=ls())
options(warn=-1)
try(dev.off())

# Initial parameter settings
```
mu=0
r=0.05
sigma=.2
para=c(mu,sigma)
dt=1/250
T=5
n=100
S=100
K=30

# "Sourcing" of the function
source("bs.R")

# Display of the result
print(bs(sigma,T,K,S,r))

1.2 How to request help for an existing R function
- Get the arguments used as inputs
  
  \texttt{args(...)}
  
- Get the html help for the function "rnorm()"
  
  \texttt{?rnorm}
  
- More details in the help task bar.

1.3 Using the R optimizer

Know-hows around the R optimizer are essential for numerical finance. An easy way to step in: \texttt{optim} for several variable optimization and \texttt{optimize} for single variable optimization.

\textbf{Example:} inverting the BS formula numerically (building on the previous script and functions)

invert.bs<-function(sigma,P,T,K,S,r){
  # require bs.R
  spread=P-bs(sigma,T,K,S,r)
  return(spread^2)
}

P=bs(sigma,T,K,S,r)
optimize(f = invert.bs, interval =c(0,1) , lower = 0,
    upper = 1, maximum = FALSE,
    tol = .Machine$double.eps^0.25, P,T,K,S,r)
1.4 Sampling a GARCH process

Important point: think of the recursive way to build the process. The model:

\[ r_t = \log \frac{S_t}{S_{t-1}} = \mu + \sqrt{h_t} \epsilon_t \] (1)

\[ h_t = \omega_0 + \alpha (r_{t-1} - \mu)^2 + \beta h_{t-1} \] (2)

Example from the class:

```r
set.seed(1) # Initialize the random number generation
w0=0.05 # Set of parameters
alpha=.08
beta=.87
mu=0
n=1000 # Sample size
h=w0/(1-alpha-beta) # Initializations
eps=rnorm(n,0,1)
x=eps[1]*sqrt(h)
for (i in 2:n){
x=rbind(x,eps[i]*sqrt(w0+alpha*x[i-1]^2+beta*h[i-1]))
h=rbind(h,w0+alpha*x[i-1]^2+beta*h[i-1])
}
```

The following piece of code allows you to graph the results of the simulation:

```r
layout(matrix(1:2,2,1))
plot(x,type="l")
grid(col=1)
plot(h,type="l")
```

Run the code, and guess the how each parameter impacts the distribution of returns and the shape of the time series of volatility. Use the "density()" function to estimate the distribution of the sampled returns and compare it to a Gaussian one.

1.5 Estimating via ML a GARCH process

ML estimates of GARCH models are straightforward to obtain. Log likelihood of the model is obtained by computing

\[ L(\theta|r) = f(r_1) \sum_{t=2}^{n} f^\theta(r_t|r_{t-1}) \] (3)
ML estimates are obtained by maximizing the latter expression with respect to $\theta$ the vector of the parameters. 

The following function computes the log-likelihood of a GARCH(1,1) model:

```r
garch_loglik<-function(para,x,mu){
  # Parameters
  omega0=para[1]
  alpha=para[2]
  beta=para[3]

  # Volatility and loglik initialisation
  loglik=0
  h=var(x)

  # Start of the loop
  for (i in 2:length(x)){
    h=omega0+alpha*(x[i-1]-mu)^2+beta*h
    loglik=loglik+dnorm(x[i],mu,sqrt(h),log=TRUE)
  }

  print(para)
  return(-loglik)
}
```

The estimation is obtained by numerically maximizing this function, as follows:

```r
para=c(0.2,0.2,0.8)
optim(para, garch_loglik, gr = NULL,method = c("BFGS"),x,mu)
```

Answer the following questions:

1. Use this code to estimate the parameters of the GARCH(1,1) model that you previously sampled.
2. Investigate the values of the parameters reached by the optimization (find a way to graph this evolution).
3. Can you use a different estimation procedure in R?
4. Think of a strategy to estimate the distribution of the maximum likelihood parameters, given the different pieces of code you have at hand.
5. Now, the time series model becomes

\[ r_t = \log \frac{S_t}{S_{t-1}} = \mu + \sqrt{h_t} \left( \epsilon_t + \sum_{i=0}^{N_t} x_{i,t} \right) \]  \hspace{1cm} (4)

\[ h_t = \omega_0 + \alpha (r_{t-1} - \mu)^2 + \beta h_{t-1}, \]  \hspace{1cm} (5)

with \( N_t \sim P(0.1) \) and \( x_{i,t} \sim N(-0.1, 0.3) \). Sample this new model: compare the pure gaussian to this new model’s returns distribution. Amend the log likelihood to estimate the model’s parameters by quasi maximum likelihood. Estimate the distribution of quasi maximum likelihood.