Homework 4

Jordan Giebas

Due Thursday, November 30 at 3:00 PM

You can submit separate pdf files, one generated from the R Markdown, and the other from the "derivations" required in Question 2. The relevant .Rmd file should also be submitted.

Please do not submit photos of your homework. Scanners are available for your use.

Question 1

We have already seen that the log returns are not well-modelled by a normal distribution, as the normal distribution does not place enough probability in the tails to model the extreme events that can occur.

Here, we will assume that the daily log returns for an equity can be modelled as being i.i.d. with distribution given by σT, where T is a random variable with the t-distribution with ν degrees of freedom. The median of the distribution is assumed to be zero.

a. Write an R function that takes as input a vector of values (e.g., log daily returns from a single equity), along with an assumed value for ν, and then returns the MLE for σ along with the standard error for that estimator and a 95% confidence interval for σ.

b. Demonstrate the use of this function on some real log return data found using `quantmod()`. Try at least three different equities.

c. Create a second function which maximizes the likelihood over different values of ν, in addition to maximizing over σ. This function does not need to return standard errors or confidence intervals. (We will discuss how to do this soon.) Test this on some different examples.

```
options(warn = -1)

library(quantmod)

print_results = function(ticker_char, vec){

  cat(
  sprintf("RESULTS FOR %s", ticker_char), "\n",
  sprintf("The MLE is: %f", as.double(vec[1])), "\n",
  sprintf("The SE for the MLE is: %f", as.double(vec[2])), "\n",
  sprintf("The real confidence interval is [ %f , %f ]", as.double(vec[3]), as.double(v
  sprintf("The approx confidence interval is [ %f , %f ]", as.double(vec[5]), as.double
  )
}

# Input : Log returns vector (lr_vec), d.o.f. (nu)
```

```r
# Output: MLE for (sigma) with Standard Error (std_err)

t_dist = function(x){

  # Need the negative of the log-likelihood function.
  # To be minimized using optim.
  negllh = function(sigma, x, nu){

    return ( (-1)*(sum(dt(x/sigma, nu, log=TRUE)) - length(x)*log(sigma)) )

  }

  # Define the starting point for the iterative optim method
  sigma_startingEstimate = 0.01

  # Perform the optimization
  optimout = optim(sigma_startingEstimate,negllh, x=x, nu=5, hessian=TRUE, method="BFGS"

  ## Output Vars

  # Get the MLE
  mle = optimout$par

  # Get the Standard_Error
  std_err = sqrt(1/optimout$hessian)

  # Construct CI
  z = qnorm(1-(0.05/2))
  L = mle - z*std_err
  U = mle + z*std_err

  # Approx CI
  L_approx = mle - 2*std_err
  U_approx = mle + 2*std_err


  return ( list(mle, std_err, L, U, L_approx, U_approx) )

}

# Get the symbols for Goldman Sachs, Apple, and Ford for testing
getSymbols(c("GS","AAPL","F"),src="google")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
```

```
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "GS"   "AAPL" "F"
```

```
#### Test Cases ####

# Goldman Sachs trial
GS_vec = dailyReturn(GS, type='log')
res_GS = t_dist(GS_vec)
print_results("GS",res_GS)
```

```
## RESULTS FOR GS
##  The MLE is: 0.015653
##  The SE for the MLE is: 0.000276
##  The real confidence interval is [ 0.015112 , 0.016194 ]
##  The approx confidence interval is [ 0.015101 , 0.016205 ]
```

```
# Apple trial
apple_vec = dailyReturn(AAPL, type='log')
res_apple = t_dist(apple_vec)
print_results('AAPL',res_apple)
```

```
## RESULTS FOR AAPL
##  The MLE is: 0.014599
##  The SE for the MLE is: 0.000253
##  The real confidence interval is [ 0.014103 , 0.015096 ]
##  The approx confidence interval is [ 0.014092 , 0.015106 ]
```

```
# Ford trial
F_vec = dailyReturn(F, type='log')
res_F = t_dist(F_vec)
print_results('F',res_F)
```

```
## RESULTS FOR F
##  The MLE is: 0.017388
##  The SE for the MLE is: 0.000310
##  The real confidence interval is [ 0.016781 , 0.017995 ]
##  The approx confidence interval is [ 0.016768 , 0.018007 ]
```

```
options(warn = -1)

## Part (C)

print_pretty = function(ticker_char, vec) {
  cat(
    sprintf("\nRESULTS FOR %s", as.character(ticker_char)), "\n",
```

```
    sprintf("The MLE for sigma and nu respectively is: %f, %f", as.double(vec[1]), as.do
  )
}

t_dist_partc = function(x) {

  # Need the negative of the log-likelihood function.
  # To be minimized using optim
  negllh = function(pars, x) {
    return ( (-1)*(sum(dt(x/pars[1], pars[2], log=TRUE)) - length(x)*log(pars[1])) )
  }

  # Define the starting point for the iterative optim method
  sig_start = 0.01
  nu_start = 5

  # Perform the optimization
  optimout = optim(c(sig_start,nu_start), negllh, x=x, lower=c(-Inf,2))
  out = optimout$par

  # Return results
  return ( out )

}

#### Test Cases ####

# Goldman Sachs trial
GS_vec = dailyReturn(GS, type='log')
GS_res = t_dist_partc(GS_vec)
print_pretty('GS', GS_res)
```

```
##
## RESULTS FOR GS
##  The MLE for sigma and nu respectively is: 0.013255, 2.584752
```

Question 2

Suppose that X is binomial(n, p). The MLE for p is, not surprisingly, the sample proportion X/n. (You do not need to prove this.)

a. What is the MLE for $p/(1-p)$? (This is called the *odds*.)

b. Approximate the distribution for the MLE of the odds.

c. Use part (b) to construct a $100(1-\alpha)\%$ confidence interval for the odds.

d. Write a simulation procedure that tests the validity of the confidence interval found

in part (c). Is the confidence interval an adequate approximation when $n = 10$ and $p = 0.10$?

```
# Part (d)

# Define the correct definitions of n, N.
n = 10
N = 1 # Assume that the number of trials in each binomial experiment is 1

# Define prob/theta(prob) = p/(1-p)
p = 0.10
theta0 = p/(1-p)

# Define the confidence level, and the number of simulations
reps = 10000
alpha = 0.05

# Do like notes
thetahat = numeric(reps)
lower = numeric(reps)
upper = numeric(reps)
coverstruth = logical(reps)

# Perform each simulation
for(i in 1:reps)
{
  x = rbinom(N,n,p)

  thetahat = mean(x)/(n-mean(x))

  # Define fisher information analytically
  fisher = n/(thetahat*(1+thetahat)^2)

  lower = thetahat - qnorm(1-alpha/2)*sqrt(1/(N*fisher))
  upper = thetahat + qnorm(1-alpha/2)*sqrt(1/(N*fisher))
  coverstruth[i] = (lower < theta0) & (upper > theta0)

}

# Output results
print(sum(coverstruth)/reps)
```

```
## [1] 0.6611
```

With N=1, the confidence interval is at 65%, so the 95% interval is clearly not valid. However, upon testing the same code with N=1000 we get 0.9457 which shows as N grows that it will approach the 95% CI. However, the way you take the log likelihood function is different with N not equal to one, so I'm not sure if this is a valid way to think about it.