Homework 4

Ze Yang (zey@andrew.cmu.edu)

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.

Solution:

```
# get data
getSymbols("GOOG",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] "GOOG"
```

```
colnames(GOOG) = c('o', 'h', 'l', 'c', 'volume')
GOOG$log.ret = log(lag(GOOG$c)) - log(GOOG$c)
```

```
GOOG.log.ret = as.vector(GOOG$log.ret[-1])

getSymbols("AAPL",src="google")
```

```
## [1] "AAPL"
```

```
colnames(AAPL) = c('o', 'h', 'l', 'c', 'volume')
AAPL$log.ret = log(lag(AAPL$c)) - log(AAPL$c)
AAPL.log.ret = as.vector(AAPL$log.ret[-1])

set.seed(42)
```

(a), (b)

We calculate the MLE using GOOG's log return data:

```
mle.logret = function(log.ret, nu, alpha=0.95) {

  neg.loglike.logret = function(sigma, log.ret, nu) {
    #' The negative log-likelihood of a sample of log returns.
    #' @param log.ret: log return sample, ~ i.i.d sigma * T
    #'    where T ~ student.t(nu), sigma is scaling constant.
    #' @param sigma: scaling parameter.
    #' @param nu: degrees of freedom parameter.
    #' @return the value of negative log-likelihood.
    return((-1)*sum(dt(log.ret/sigma, nu, log=T))+
             length(log.ret)*log(sigma))
  }

  sigma.init = sd(log.ret) # initial value
  opt = optim(sigma.init, neg.loglike.logret,
              log.ret=log.ret, nu=nu, hessian=T)
  se = sqrt(1/opt$hessian)
  z = -qnorm((1-alpha)/2)
  conf = c(opt$par - z*se, opt$par + z*se)
  return(list(
    l=opt$value,
    mle=opt$par,
    se=se,
    hessian=opt$hessian,
    conf=conf
  ))
}

mle.logret(GOOG.log.ret, nu=20)
```

```
## Warning in optim(sigma.init, neg.loglike.logret, log.ret = log.ret, nu = nu, : one-di
## use "Brent" or optimize() directly
```

```
## $l
## [1] -7331.228
##
## $mle
## [1] 0.01517279
##
## $se
##               [,1]
## [1,] 0.0002289526
##
## $hessian
##          [,1]
## [1,] 19076941
##
## $conf
## [1] 0.01472405 0.01562153
```

We have

$$\hat{\sigma}_{mle} = 0.015173; \qquad SE(\hat{\sigma}_{mle}) = 0.000229$$

And the 95% confidence interval:

$$CI(95\%) = [0.01472405, \quad 0.01562153]$$

(c)

We conduct bivariate MLE to estimate both $\hat{\sigma}$ and $\hat{\nu}$

```r
mle.logret.2 = function(log.ret) {

  neg.loglike.logret.2 = function(pars, y) {
    #' The negative log-likelihood of a sample of log returns.
    #' @param pars: c(sigma, nu)
    #' @param log.ret: log return sample, ~ i.i.d sigma * T
    #'    where T ~ student.t(nu), sigma is scaling constant.
    #' @return the value of negative log-likelihood.
    return((-1)*sum(dt(y/pars[1], pars[2], log=TRUE))+
             length(y)*log(pars[1]))
  }
  #

  pars.init = c(sd(log.ret), 1) # initial value
  names(pars.init) = c("sigma", "nu")
  opt = optim(pars.init, neg.loglike.logret.2,
              y=log.ret, hessian=T)
  return(list(
    l=opt$value,
    mle=opt$par,
    hessian=opt$hessian
```

```
  ))
}

mle.logret.2(GOOG.log.ret)
```

```
## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced
## $l
## [1] -7547.14
##
## $mle
##      sigma         nu
## 0.01061227 2.81710611
##
## $hessian
##              sigma           nu
## sigma 24383763.78 -23967.46050
## nu      -23967.46     56.36243
```

```
mle.logret.2(AAPL.log.ret)
```

```
## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced

## Warning in log(pars[1]): NaNs produced
## $l
## [1] -7085.869
##
## $mle
##    sigma       nu
```

```
## 0.013161 3.202538
##
## $hessian
##              sigma           nu
## sigma 16555652.23 -16672.56387
## nu      -16672.56     36.48162
```

We have

$$\hat{\sigma}_{mle} = 0.01058995; \qquad \hat{\nu}_{mle} = 2.81039509$$

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 $p = 0.10$?

Solution:

(a): $X \sim$ Binomial(n, p). The log likelihood function is:

$$\log L(X; p) = \log\left(p^X (1-p)^{n-X}\right) = X \log p + (n - X)\log(1-p) \tag{1}$$

The first order condition yields:

$$\frac{\partial \log L(X; p)}{\partial p} = \frac{X}{p} - \frac{n-X}{1-p} = \frac{X - np}{p(1-p)} = 0 \tag{2}$$

So $\hat{p}_{mle} = X/n$. We regard the odds as a function of p:

$$f(p) = \frac{p}{1-p}$$

By the invariance property of the MLE, and the fact that $\hat{p}_{mle} = \frac{X}{n}$ we know that

$$\hat{f(p)}_{mle} = f(\hat{p}_{mle}) = \frac{X/n}{1 - X/n} = \frac{X}{n - X}$$

Moreover,

$$\begin{aligned}
\frac{\partial^2 \log L(X;p)}{\partial p^2} &= -\frac{X - 2Xp + np^2}{p^2(1-p)^2} \\
\Rightarrow \qquad NI(p) &= \mathbb{E}\left[-\frac{\partial^2 \log L(X;p)}{\partial p^2}\right] \\
&= \frac{n}{p(1-p)}
\end{aligned} \tag{3}$$

(b): $f'(p) = \frac{1}{(1-p)^2}$. By the Delta Method, approximately we have:

$$f(p) \sim \mathcal{N}\left(f(p_0), (f'(p_0))^2 \frac{1}{NI(\theta_0)}\right) \tag{4}$$

Substitute p_0 with $\hat{p}$, and $N = 1$ in this problem since we only have 1 Binomial sample X. The above distribution is approximately:

$$f(p) \sim \mathcal{N}\left(\frac{\hat{p}}{1-\hat{p}}, \frac{\hat{p}}{n(1-\hat{p})^3}\right)$$

Where $\hat{p} = X/n$, i.e.

$$f(p) \sim \mathcal{N}\left(\frac{X}{n-X}, \frac{X/n}{n(1-X/n)^3}\right) \tag{5}$$

(c): The $100(1-\alpha)\%$ confidence interval of $f(p)$ is:

$$\begin{aligned}
CI(\alpha) &= \left[f(\hat{p}) - z_{\frac{\alpha}{2}} SE(f(\hat{p})), \quad f(\hat{p}) + z_{\frac{\alpha}{2}} SE(f(\hat{p}))\right] \\
&= \left[\frac{\hat{p}}{1-\hat{p}} - z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{p}}{n(1-\hat{p})^3}}, \quad \frac{\hat{p}}{1-\hat{p}} + z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{p}}{n(1-\hat{p})^3}}\right] \\
&= \left[\frac{X}{n-X} - z_{\frac{\alpha}{2}}\sqrt{\frac{X/n}{n(1-X/n)^3}}, \quad \frac{X}{n-X} + z_{\frac{\alpha}{2}}\sqrt{\frac{X/n}{n(1-X/n)^3}}\right]
\end{aligned} \tag{6}$$

Where $\hat{p} = X/n$.

(d)

```
confint.contains = function(x, n, p0, alpha=0.95) {
  # Whether the confidence interval contains true value
  #' @param x: one sample ~ Binomial(n,p)
  #' @param n: parameter n.
  #' @param p0: true parameter p.
  #' @param alpha: the confidence level.
  #' @return bool: whether the confidence interval
  #'    contains true value
```

```
  p = x/n
  if(p == 1) {
    # singular case, X = n, odds -> inf
    return(TRUE)
  }
  odds = p/(1-p)
  odds.0 = p0/(1-p0)
  z = -qnorm((1-alpha)/2)
  se = sqrt(p/(n*(1-p)^3))
  ub = odds + z*se
  lb = odds - z*se
  return((odds.0 > lb) & (odds.0 < ub))
}

odds.confidence = function(N, p, n, alpha=0.95) {
  # Check the confidence interval coverage with simulated samples.
  #' @param N: How many samples.
  #' @param n: parameter n.
  #' @param p: true parameter p.
  #' @param alpha: the confidence level.
  sample = rbinom(N, n, p)
  value.in.CI = sapply(sample, confint.contains,
                       n=n, p0=p, alpha=alpha)
  conf = sum(value.in.CI) / N
  return(conf)
}
```

```
# Simulation (1) with p=0.5, n=100
odds.confidence(N=10000, p=0.5, n=100)
```

```
## [1] 0.9543
```

```
# Simulation (2) with p=0.8, n=70
odds.confidence(N=10000, p=0.8, n=70)
```

```
## [1] 0.9406
```

```
# Simulation (3) with p = 0.1, n=10
odds.confidence(N=10000, p=0.1, n=10)
```

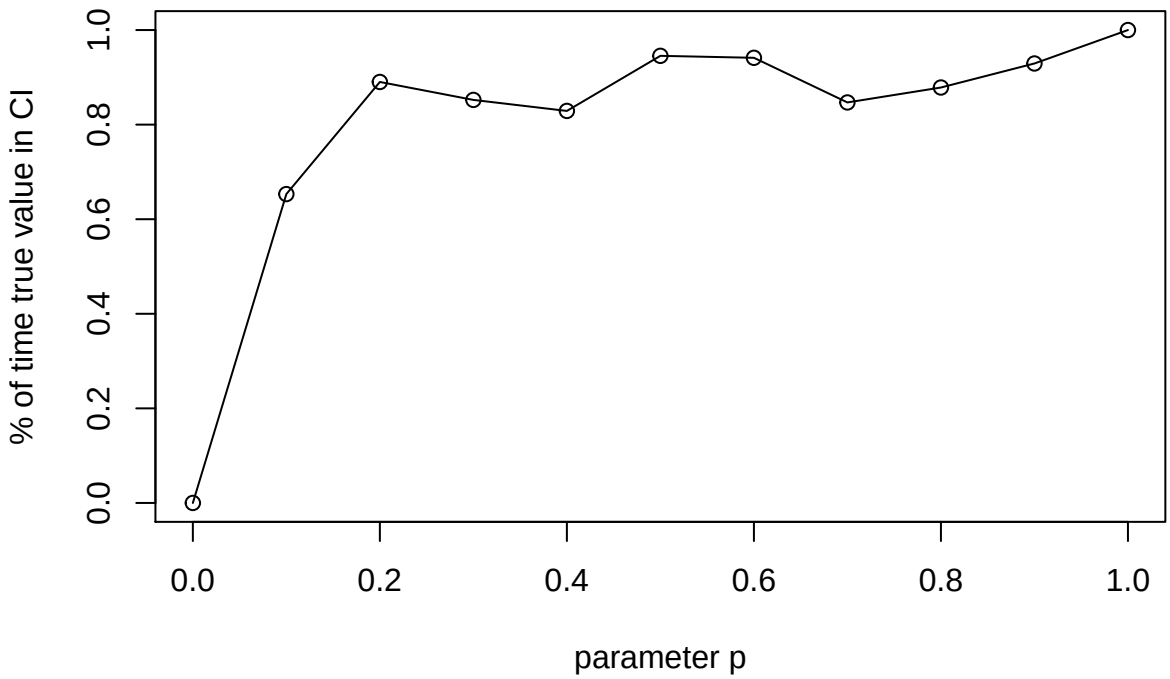
```
## [1] 0.6571
```

```
# Simulation (4) with with p = 0.5, n=10
odds.confidence(N=10000, p=0.5, n=10)
```

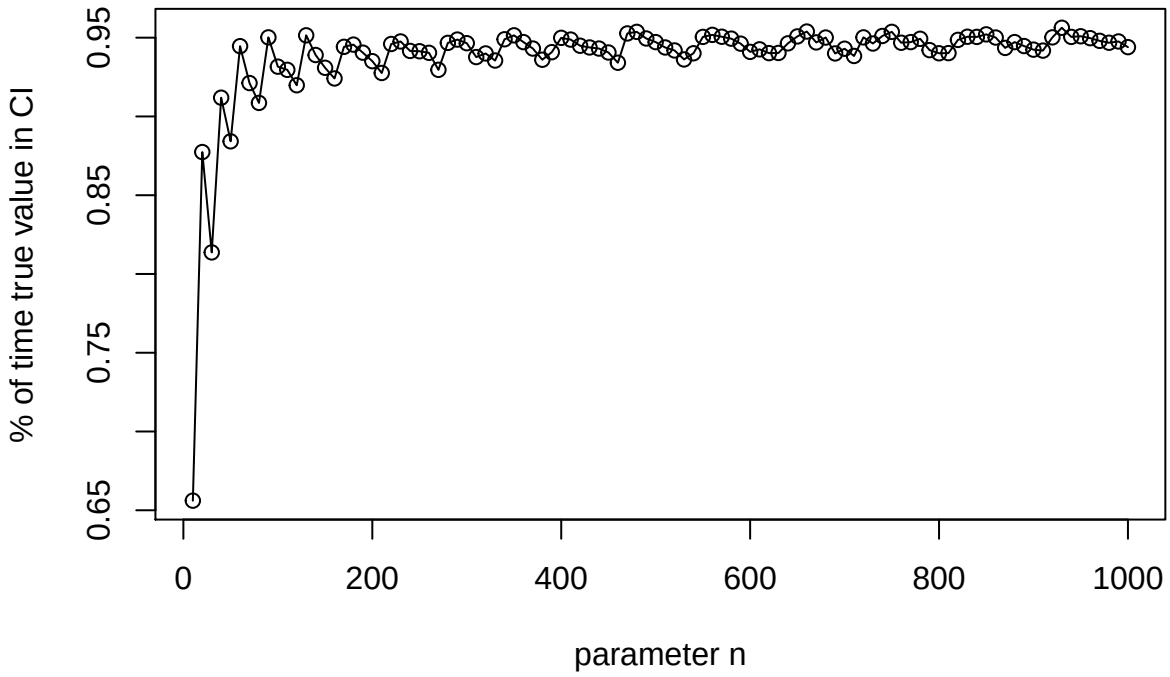
```
## [1] 0.9468
```

```
params.p = seq(from=0,to=1, by=0.1)
conf.insample = sapply(params.p, odds.confidence, N=10000, n=10)
```

```r
plot(params.p, conf.insample, type='o',
     xlab="parameter p", ylab="% of time true value in CI")
```




```r
params.n = seq(from=10,to=1000, by=10)
conf.insample = sapply(params.n, odds.confidence, N=10000, p=0.1)
plot(params.n, conf.insample, type='o',
     xlab="parameter n", ylab="% of time true value in CI")
```



Conclusion

- We can see that for larger n, the 95% confidence interval does a pretty good job. Over all experiments, about 95% of the time the true odds value is contained in the confidence interval constructed with MLE estimate. However, when $n = 10, p = 0.1$, there is only 65% of the time when the confidence interval contains the true value.
- We simulated the confidence intervals for different choices of p and n (See the plot above).
- When we keep $n = 10$ fixed, the confidence interval performs relatively better for larger p's.
- When we keep $p = 0.1$ fixed, the chances in which the confidence interval includes the true value increases with n (with some oscillations), and it converges to 0.95, which is what we expect it to be as we have set $\alpha = 0.95$.