STATS 509 HOMEWORK 2 YUAN YIN

Roxanne

2018/01/22

Question 1

(a)

First we read in the data in R, and save the closing price into a list:

```
dat = read.csv("Nasdaq_daily_Jan1_2012_Dec31_2017.csv", header = T)
t = as.Date(dat$Date, format = "%m/%d/%Y")
idx1 = which(is.na(dat$Adj.Close) == FALSE)
times = t[idx1]
acp = dat$Adj.Close[idx1]
```

Then we compute the log-return of the closing prices, also we compute the mean and standard deviation of log-return. To estimate the relative VaR, we choose double exponential distribution as the distribution of log-return. We choose significance $\alpha = 0.005$. Here is what we get:

```
acp_lreturn = diff(log(acp))
source('startup.R')
lambda = sqrt(2)/sd(acp_lreturn)
mu = mean(acp_lreturn)
sig = sd(acp_lreturn)
lret = qdexp(0.005, mu, lambda)
ret = exp(lret) - 1
VaR_tilde = - ret
```

```
## [1] "the mean is  0.000635236154944401"

## [1] "the standard deviation is  0.00892248113921368"

## [1] "VaR_tilde =  0.0280194248484432"
```

Thus, we know that $V\tilde{a}R = 0.0284$. Compare the result with VaR of simply 0.005-quantile of log-returns:

```
-(exp(quantile(acp_lreturn, 0.005))-1)
```

```
##       0.5%
## 0.02979774
```

We can see that our estimate is a little less than simply quantiles, which means we may think the real distribution of log-returns may have heavier tails than our estimate.

(b)

We generate 100000 samples to estimate the expection of shorfall:

```
set.seed(2015)
VaR = VaR_tilde * 10 * 10^6
r = rdexp(1000000, mu, lambda) # generate random numbers for 1000 times
port = 10^7 * exp(r)
s = 0
```

```r
n = 0
port = -(port - 10^7)
for (i in port)
{
  if (i > VaR)
  {
    s = s + i
    n = n + 1
  }
}
expection = s/n
```

```
## [1] "expection is 340809.097324058"
```

As the result above, $ES_q = 342294.709271728$.

(c)

First we extract positive losses from original data:

```r
idx11 = which(acp_lreturn <= 0)
acp_lreturn1 = acp_lreturn[idx11]
```

We assume that the log-returns we extract obeys one-sided exponential distribution. We use Monte Carlo simulation to estimate the relative VaR corresponding to $\alpha = 0.005$:

```r
set.seed(2015)
lambda1 = 1/sd(acp_lreturn1)
VaR_tilde1 = -(exp(-qexp(1 - 0.005, rate = lambda1))-1)
VaR1 = VaR_tilde1 * 10 * 10^6
r1 = -rexp(1000000, lambda1) # generate random numbers for 1000 times
port1 = 10^7 * exp(r1)
s1 = 0
n1 = 0
port1 = -(port1 - 10^7)
for (i in port1)
{
  if (i > VaR1)
  {
    s1 = s1 + i
    n1 = n1 + 1
  }
}
expection1 = s1/n1
```

```
## [1] "VaR_tilde is  0.0348551433271114"
```

```
## [1] "expection is  412596.766634534"
```

We can see that the result of estimated $\tilde{VaR} = 0.0348551433271114$. Compare this with simply using 0.005 quantile:

```r
-(exp(quantile(acp_lreturn1, 0.005))-1)
```

```
##       0.5%
## 0.03357202
```

We can see that our estimate of relative VaR is a little larger than simply quantile. Also, from above code we can see the expected shortfall is 412596.766634534.

Question 2

(a)

The result is as follows:

```
dat1 = read.csv("ford.csv", header = T)
t1 = as.Date(dat1$X.m..d..y, format = "%m/%d/%Y")
idx2 = which(is.na(dat1$FORD) == FALSE)
times1 = t1[idx2]
ford_return = dat1$FORD[idx2]
```

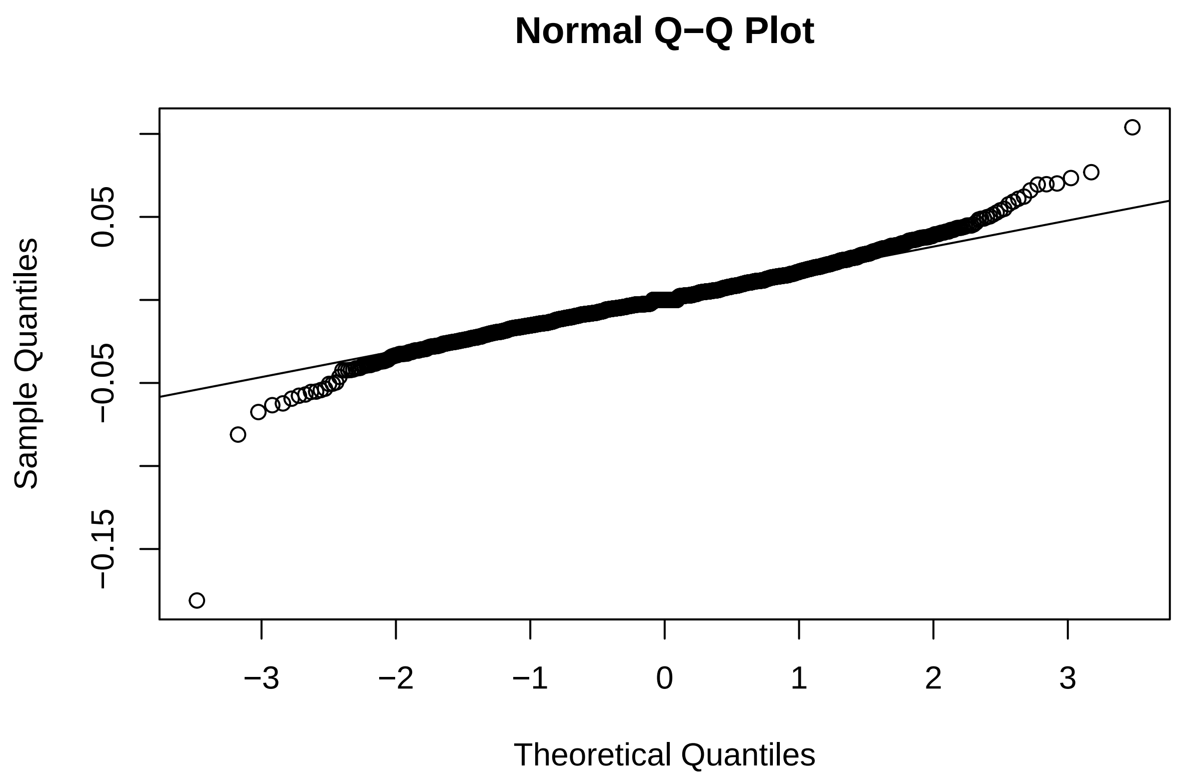
```
## [1] "mean is  0.0007600788763275"
```

```
## [1] "median is  0"
```

```
## [1] "standard deviation is  0.0183155689072455"
```

(b)&(c)

```
qqnorm(ford_return)
qqline(ford_return)
```



Obviously, the plot of returns have huge differences with normal distribution, the tails are heavier. To check this, we use W-test to confirm our assumption:

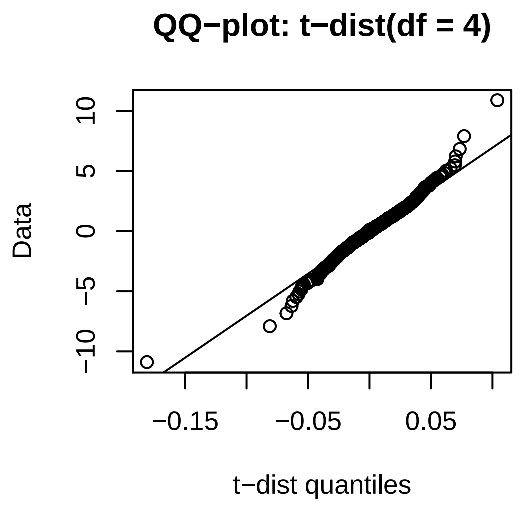
```r
shapiro.test(ford_return)
```

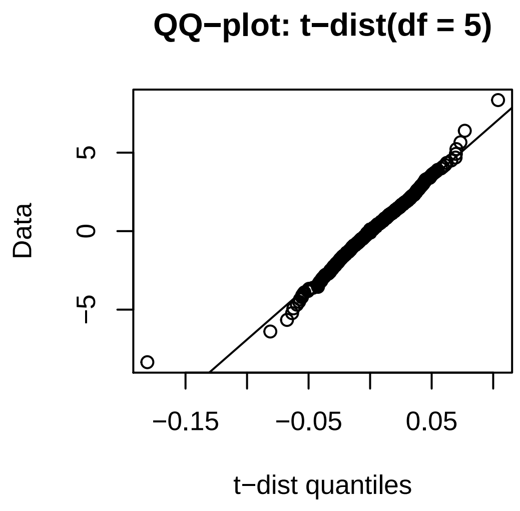
```
##
##  Shapiro-Wilk normality test
##
## data:  ford_return
## W = 0.96388, p-value < 2.2e-16
```

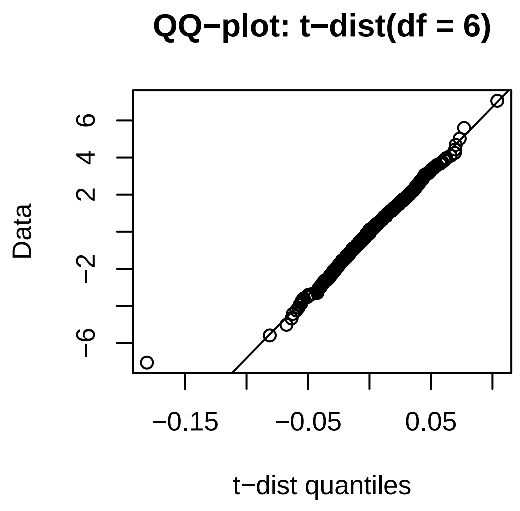
the p-value is smaller than 0.01 significantly, which means we should reject the null hyphosis that the distribution is a normal distribution.

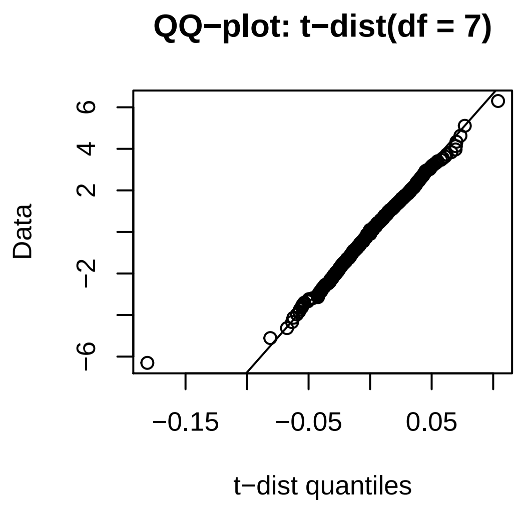
(d)

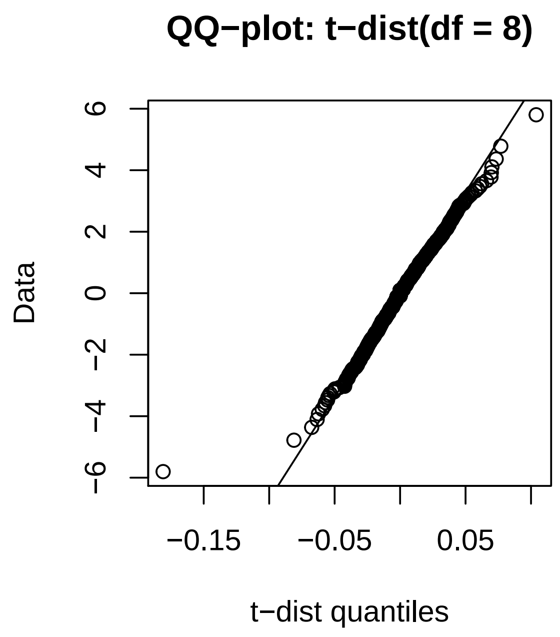
We choose 6 different degrees of freedom from 5-8, and we can see that when df = 6, the plot has the most linear look

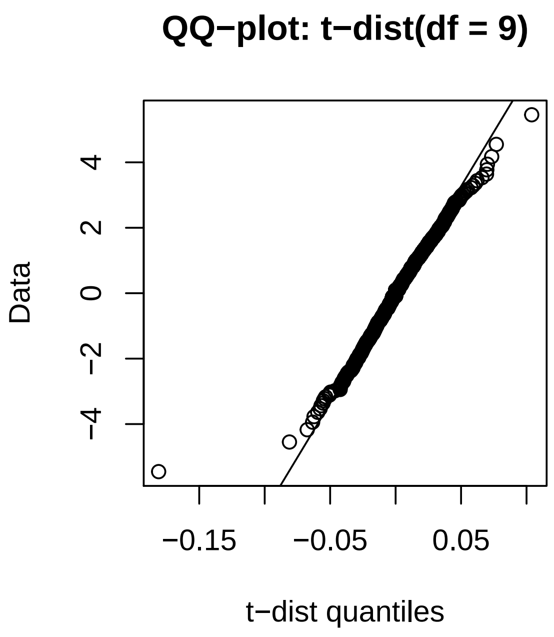












We can easily find that the minimum return happens on:

```r
t1[which.min(ford_return)]
```

```
## [1] "1987-10-19"
```

It is exactly the Black Monday in our question. We know that there is a small possibility that things like this days happens, so it can be regarded as an outlier to our data, we should delete it when we set up our model. It will make easier to find the best model fit our data. However, We don't need to ignore this data when we are looking for the best choice of df. As we know, for every distribution, there is small possibility that extreme value happens, it shouldn't affect the degree of freedom of our distribution. Although after we delete it, the QQ-plot will look much more linear, but as long as our data set has large enough samples, it won't affect the choice of degree of freedom.

(e)

we use equation below to estimate the variance of median:

$$\frac{1(1-q)}{n[f\{F^{-1}(q)\}]^2}$$

where f is the pdf function estimated by kernel density estimation, and $F^{-1}(q)$ can be estimated by q-quantile (q=0.5)

```r
F_inverse = quantile(ecdf(ford_return),0.5)
dens = density(ford_return, kernel = c("gaussian"))
f = dens$y[which.min(abs(dens$x - F_inverse))]
variance = 0.5^2/(length(ford_return)*(f)^2)
```

Thus the standard error of sample median is:

```
## [1] 0.0004274856
```

It's easy to find that the standard error of the sample mean is:

```
sd(ford_return)/sqrt(length(ford_return))
```

```
## [1] 0.0004095486
```

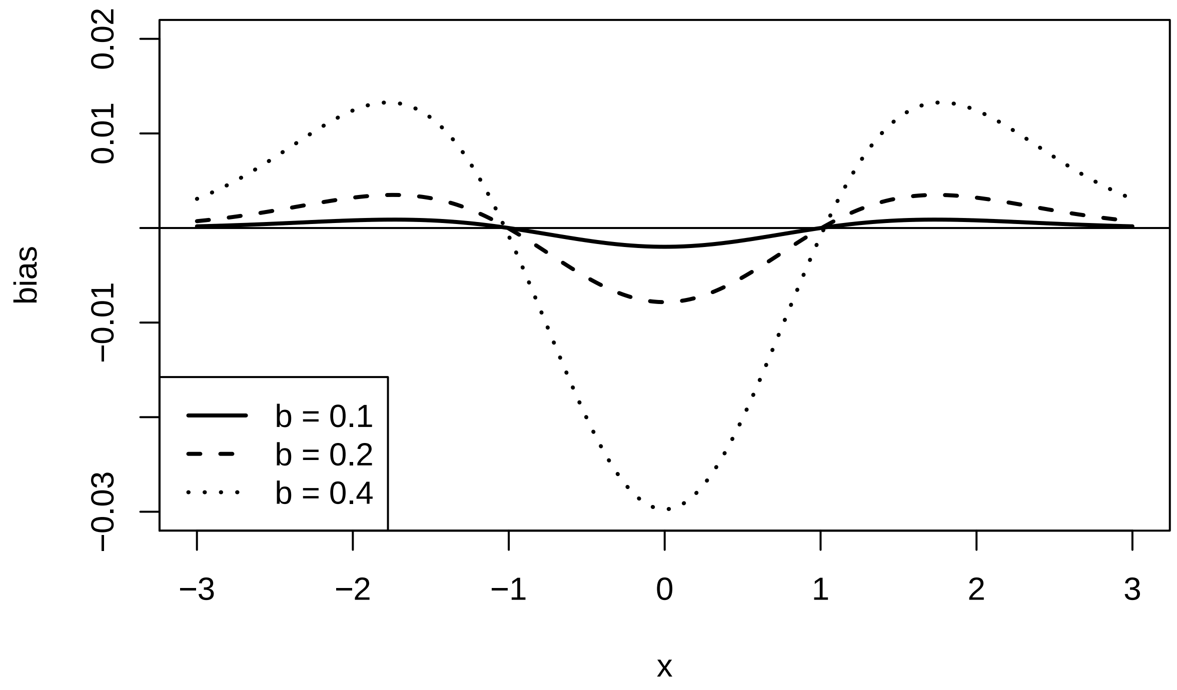
Finding that they are very close to each other, and standard error of sample median is a little larger than that of mean.

Question 3

(a)

(b)

```
x = seq(-3,3,0.01)
b1 = 0.1
b2 = 0.2
b3 = 0.4
omiga1 = 3.464 * b1
omiga2 = 3.464 * b2
omiga3 = 3.464 * b3
bias1 = (pnorm(x+omiga1/2) - pnorm(x - omiga1/2))/omiga1 - dnorm(x)
bias2 = (pnorm(x+omiga2/2) - pnorm(x - omiga2/2))/omiga2 - dnorm(x)
bias3 = (pnorm(x+omiga3/2) - pnorm(x - omiga3/2))/omiga3 - dnorm(x)
plot(x, bias1, type = "l", ylim = c(-0.03, 0.02), lwd = 2, ylab = "bias")
lines(x, bias2, lty = 2, lwd = 2)
lines(x, bias3, lty = 3, lwd = 2)
legend("bottomleft", c("b = 0.1", "b = 0.2", "b = 0.4"), lty = c(1,2,3), lwd = 2)
abline(h=0)
```



From the plot above we can find that as bandwidth increases, the limit of bias will increase. However, nomatter how bandwidth changes, there are two points stay unbiased, and that's where our estimate KDE intersect with the real pdf of normal distribution. We can find this two points:

```r
x[which.min(abs(bias1))]
```

```
## [1] -1
```

It shows that all of three estimate gets the least bias(≈ 0) around points "-1" and "1". And when x is between interval [-1,1], the bias is negative, otherwise bias is positive