HW 3
MFE 409: Financial Risk Measurement and Management
Professor Valentin Haddad
Group 9
Students: Xiahao Wang, Haoxuan Tong, Yuhua Deng, Nupur Solanki

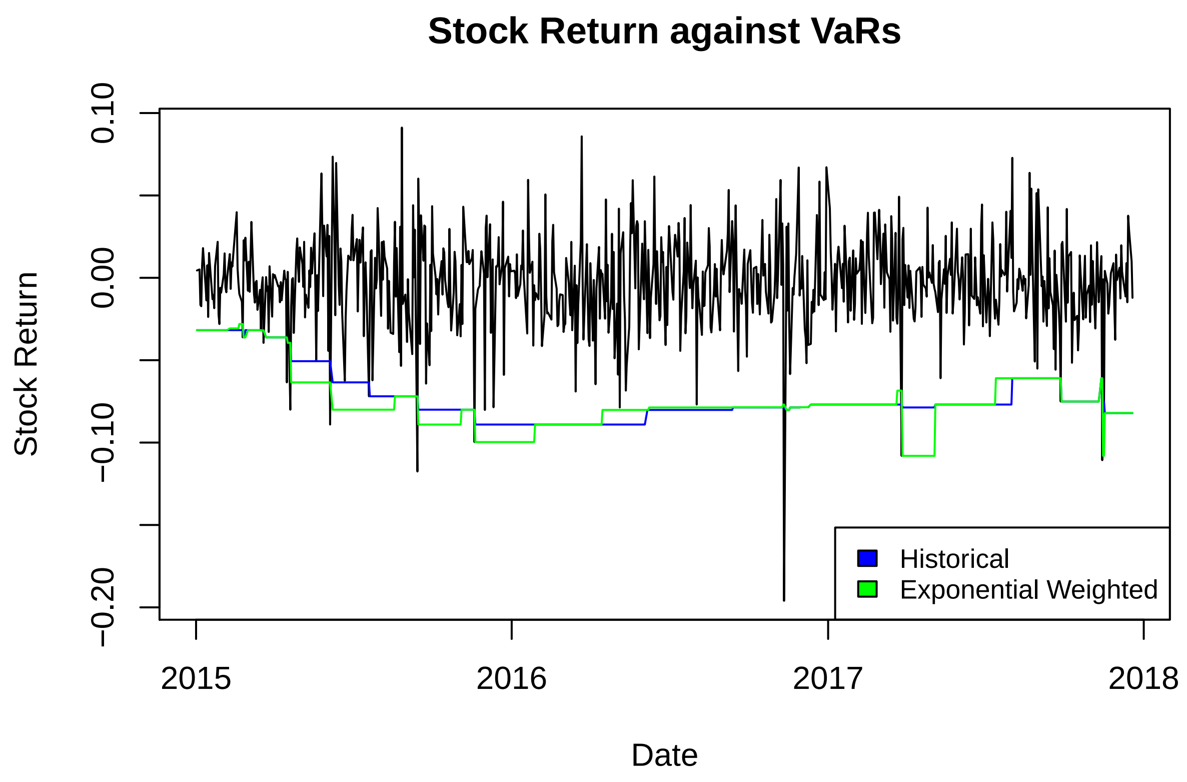
Qn1

1.

```r
suppressMessages(require(data.table))
suppressMessages(require(lubridate))
suppressMessages(require(zoo))
suppressMessages(require(ggplot2))
suppressMessages(require(dplyr))
```

```
## Warning: package 'dplyr' was built under R version 3.5.1
```

```r
suppressMessages(require(moments))
suppressMessages(library(knitr))
suppressMessages(require(quantmod))

rm(list=ls())
returns_data <- as.data.table(read.csv("hw3_returns2.csv"))
returns_data[,Date:=mdy(Date)]

# use rolling window of 251 days in general
rollingWindow <- 251
c <- 0.01
lambda  <- 0.995

cal_hist_VaR <- function(return, c){
  sorted_ret <- sort(return)
  Var_point <- ceiling(length(sorted_ret) * c)
  var <- sorted_ret[Var_point]
  return(var)
}

cal_exponential_VaR <- function(return, c){
  n <- length(return)
  weight <- lambda^seq((n-1), 0, -1) * (1-lambda)/(1-lambda^n)
  df <- cbind(weight, return)
  df_sorted <- df[order(return),]
  df_sorted <- cbind(df_sorted, cumsum=cumsum(df_sorted[,1]))
  pos <- which(df_sorted[,3]>c)[1]
  return(df_sorted[pos,2])
}

cal_parametric_StdDev_VaR <- function(return,ci_c){
  sigma <- sd(return)
  n <- length(return)
  mu <- mean(return)
  x <- mu + sigma * qnorm(ci_c)
  fx <- (1/sqrt(2*pi*sigma^2)) * exp(- (x-mu)^2/(2*sigma^2))
  stdVaR <- (1/fx) * sqrt((1 - ci_c) * ci_c/n)
  return(stdVaR)
}


# calculating historical VaR
returns_data[, historicalVaR := shift(rollapply(returns_data$Return, rollingWindow, function (return){
  return(cal_hist_VaR(return,c))
}, fill=NA, align="right"))
]

# calculating exponential Weighted VaR
returns_data[, exponentialWeightedVaR := shift(rollapply(returns_data$Return, rollingWindow, function(r
  cal_exponential_VaR(return, c)
}, fill=NA, align="right"))]

data_2005_onwards <- returns_data[Date >= "2015-01-01"]

plot(x = data_2005_onwards$Date, y= data_2005_onwards$Return, type = "l",
     main="Stock Return against VaRs", xlab = "Date",
     ylab= "Stock Return")
lines(x= data_2005_onwards$Date, y= data_2005_onwards$historicalVaR, col = "blue")
lines(x= data_2005_onwards$Date, y= data_2005_onwards$exponentialWeightedVaR, col = "green")
legend("bottomright",legend=c("Historical","Exponential Weighted"),fill=c("blue","green"), cex = 0.8)
```



```
historicalExcept <- sum(data_2005_onwards[,Return < historicalVaR ])
exExcept <-sum(data_2005_onwards[,Return < exponentialWeightedVaR ])
```

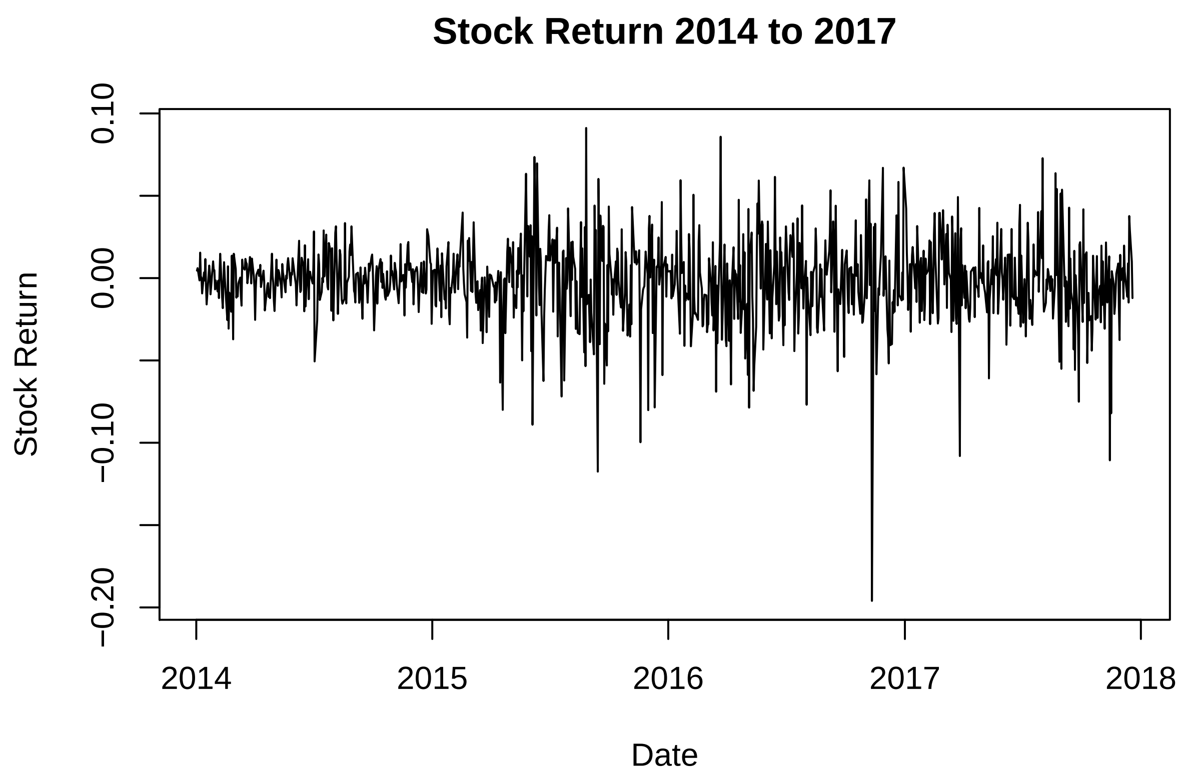
```r
cat("There are", historicalExcept, "exceptions using the historical VaR while", exExcept, "using the ex
```

```
## There are 14 exceptions using the historical VaR while 12 using the exponential Weighted VaR
```

One possbile explanantion is that the stock return has become more volatile from 2015 onwards into 2017. As exponential weighted VaR assigns more weight to recent returns, it is expected that few exceptions will be likely to occur.

By plotting the stock return from 2014 to 2017, we can see that the return is less volatile in 2014, which helps to explain less exception when using exponential weighted VaRs.

```r
plot(x = returns_data$Date, y= returns_data$Return, type = "l",
     main="Stock Return 2014 to 2017", xlab = "Date",
     ylab= "Stock Return")
```



2.1 Parametric

Use sample size 251.

For historical (Parametric) assuming normal distribution,

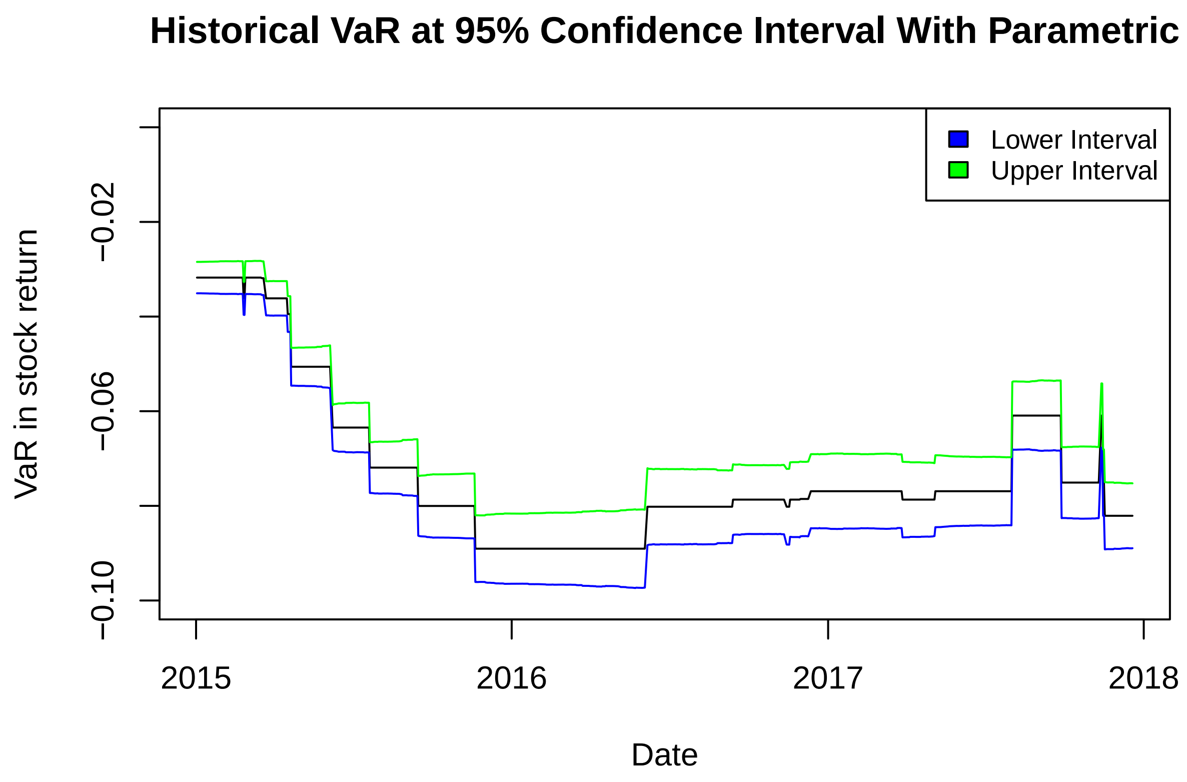
- Find the mean and standard deviation
- Find the pdf of the normal distribution
- Find the standard deviation of the VaR at each day.
- Find the confidence internal at 95 % for the VaR

Upper Interval = historical VaR + $Z_{0.975}$ * $StdDev_{VaR}$

Lower Interval = historical VaR + $Z_{0.025}$ * $StdDev_{VaR}$

calculate the standard deviation of the VaR

```r
sample_size <- 251
# 95% confidence internval
ci_c <- 0.05

returns_data[, parametric_VaR_StdDev :=  shift(rollapply(returns_data$Return, rollingWindow, function (r
   cal_parametric_StdDev_VaR(returns, ci_c)
},fill=NA, align="right"))]

returns_data[, `:=`(parametric_Hist_LI =  historicalVaR + qnorm(0.025) * parametric_VaR_StdDev, paramet

qn21plot <- returns_data[Date >= "2015-01-01"]

plot(x = qn21plot$Date, y= qn21plot$historicalVaR, type = "l",
     main="Historical VaR at 95% Confidence Interval With Parametric", xlab = "Date",
     ylab= "VaR in stock return", ylim = c(-0.1,0))
lines(x= qn21plot$Date, y= qn21plot$parametric_Hist_LI, col = "blue")
lines(x= qn21plot$Date, y= qn21plot$parametric_Hist_UI, col = "green")

legend("topright",legend=c("Lower Interval","Upper Interval"),fill=c("blue","green"), cex = 0.8)
```



2.2 Bootstrap

Still using sample size 251. Choose the number of draws for bootstrap to be 1000

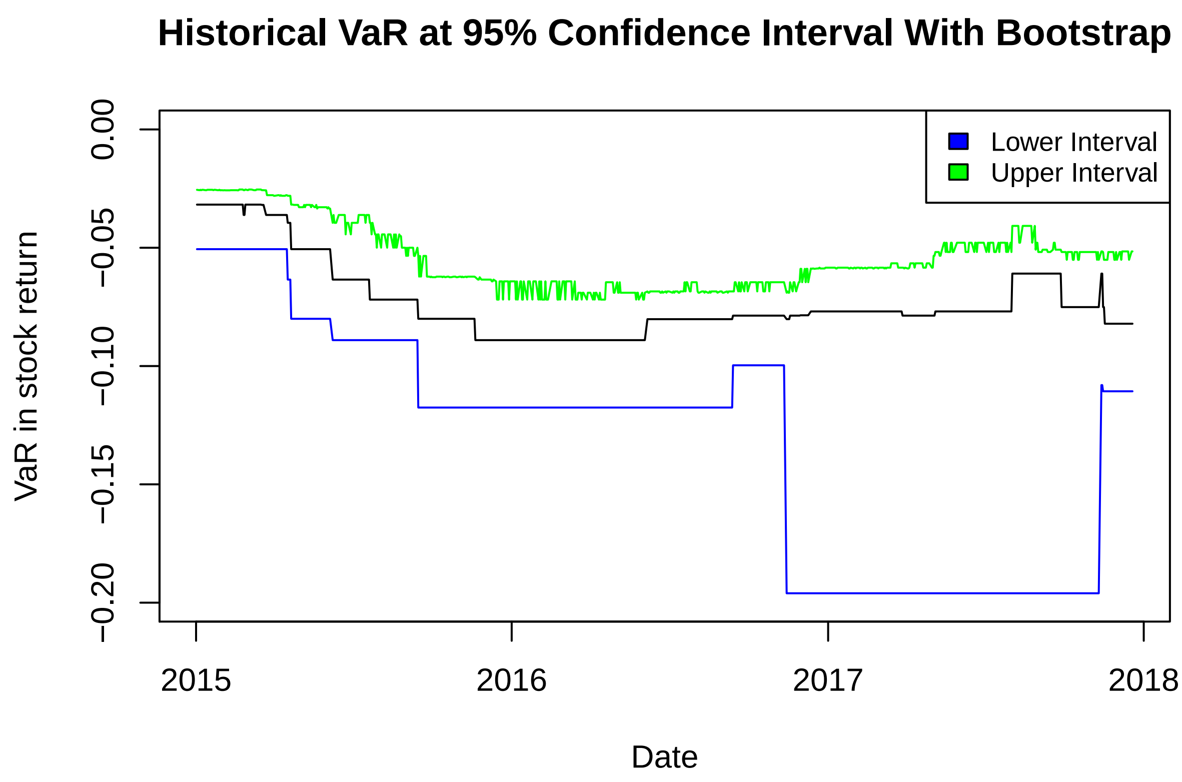
At each period (rolling 251 days):

- Take sample size of 251
- Get historical VaR

At the end, take the 2.5% and 97.5% quantile of the VaRs collected from each period.

For historical (rolling window)

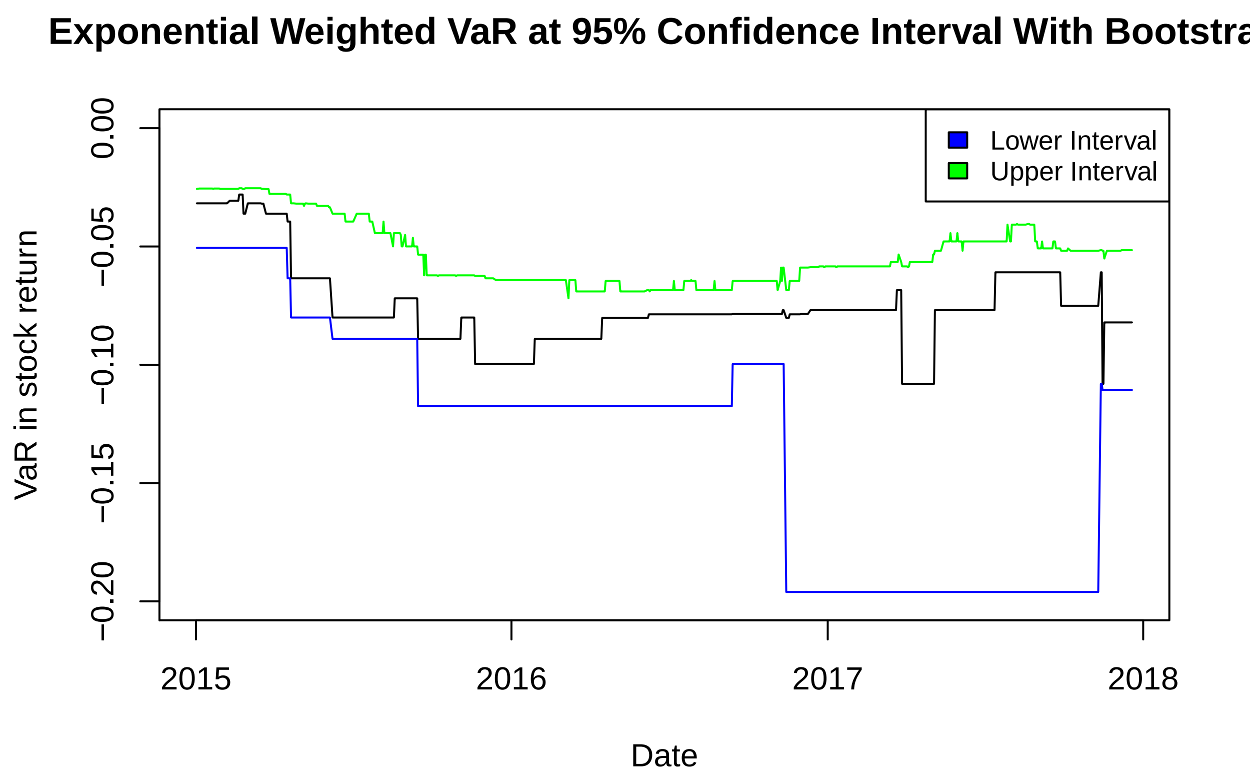
```r
sample_size <- 251
iter <- 1000
lambda <- 0.995

returns_data[, bstrap_range_hist :=  shift(rollapply(returns_data$Return, rollingWindow, function (retu
    #initialze empty list of 1000
    totalVaR <- rep(0, iter)

    # take the sample 1000 times, get totalVaR for each time
    for(i in 1:iter){
      # take 251 returns with replacement
      VaR_this_period <- sample(returns, sample_size, replace = TRUE)
      totalVaR[i] <- cal_hist_VaR(VaR_this_period, c)
    }
    # sort the totalVaRs drawn from 1000 samples
    sortedTotalVaR <- sort(totalVaR)
    return(paste0(sortedTotalVaR[25],",",sortedTotalVaR[975]))
},fill=NA, align="right"))]

returns_data[, bstrap_hist_LI:= sapply(returns_data$bstrap_range_hist, function(range){
  unlist(strsplit(range, ","))[1]
})]

returns_data[, bstrap_hist_UI:= sapply(returns_data$bstrap_range_hist, function(range){
  unlist(strsplit(range, ","))[2]
})]

# plot the data
qn22hist <- returns_data[Date >= "2015-01-01"]

plot(x = qn22hist$Date, y= qn22hist$historicalVaR, type = "l",
     main="Historical VaR at 95% Confidence Interval With Bootstrap", xlab = "Date",
     ylab= "VaR in stock return", ylim = c(-0.2,0))
lines(x= qn22hist$Date, y= qn22hist$bstrap_hist_LI, col = "blue")
lines(x= qn22hist$Date, y= qn22hist$bstrap_hist_UI, col = "green")

legend("topright",legend=c("Lower Interval","Upper Interval"),fill=c("blue","green"), cex = 0.8)
```



For exponential Weighted

```
returns_data[, bstrap_range_exp_wt :=  shift(rollapply(returns_data$Return, rollingWindow, function (re
    #initialze empty list of 1000
    totalVaR_ex <- rep(0, iter)

    # take the sample 1000 times, get totalVaR for each time
    for(i in 1:iter){
      # take 251 returns with replacement
      VaR_this_period <- sample(returns, sample_size, replace = TRUE)
      totalVaR_ex[i] <- cal_exponential_VaR(VaR_this_period, c)
    }
    # sort the totalVaRs drawn from 1000 samples
    sortedTotalVaR <- sort(totalVaR_ex)
    return(paste0(sortedTotalVaR[25],",",sortedTotalVaR[975]))
},fill=NA, align="right"))]

# plot the data
returns_data[, bstrap_exp_LI := sapply(returns_data$bstrap_range_exp_wt, function(range){
  unlist(strsplit(range, ","))[1]
})]

returns_data[, bstrap_exp_UI := sapply(returns_data$bstrap_range_exp_wt, function(range){
  unlist(strsplit(range, ","))[2]
})]


# Plot the data

qn2exp <- returns_data[Date >= "2015-01-01"]
```

```
plot(x = qn2exp$Date, y= qn2exp$exponentialWeightedVaR, type = "l",
     main="Exponential Weighted VaR at 95% Confidence Interval With Bootstrap", xlab = "Date",
     ylab= "VaR in stock return", ylim = c(-0.2,0))
lines(x= qn2exp$Date, y= qn2exp$bstrap_exp_LI, col = "blue")
lines(x= qn2exp$Date, y= qn2exp$bstrap_exp_UI, col = "green")

legend("topright",legend=c("Lower Interval","Upper Interval"),fill=c("blue","green"), cex = 0.8)
```



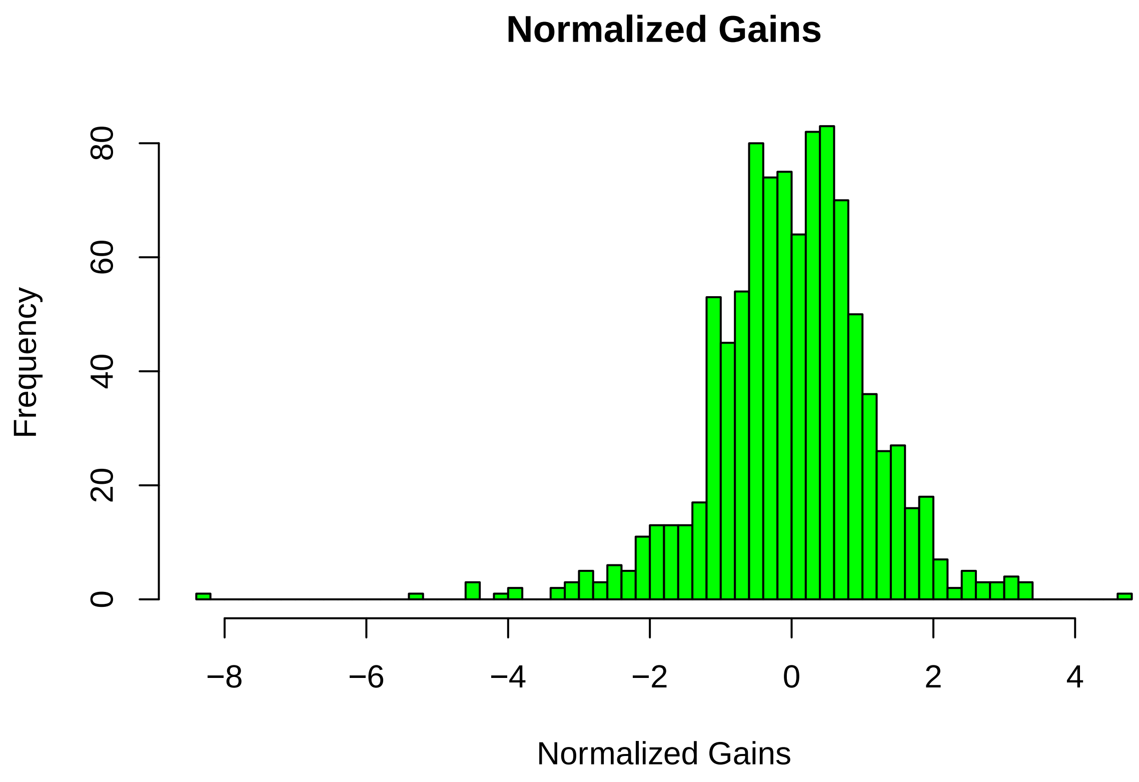
```
#write.table(returns_data, file = "returns_data.csv", row.names=FALSE, sep=",")
```

3.

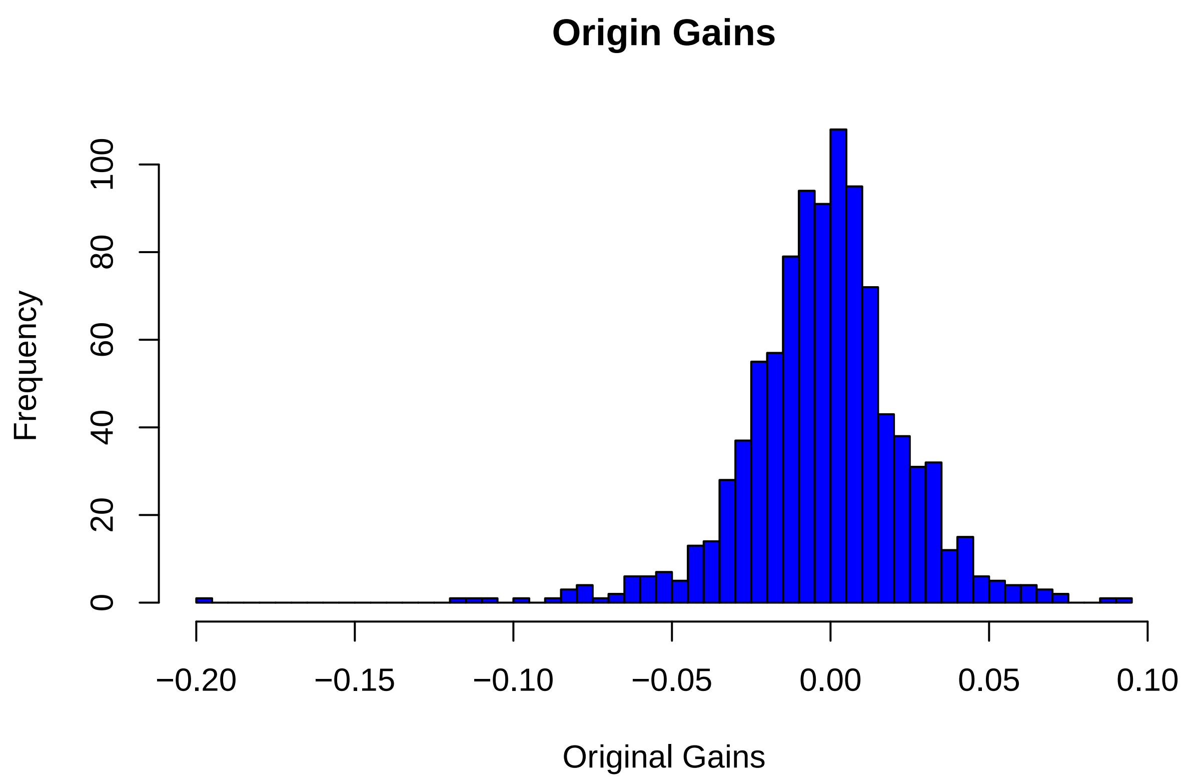
set rolling window size to be 20 days in a month, Use `shift` to obtain the the volatility of past 20 days

```
# reload data gain
Q3_returns_data <- as.data.table(read.csv("hw3_returns2.csv"))
Q3_returns_data[,Date:=mdy(Date)]

rolling_win <- 20

# find stddev, mean
Q3_returns_data[, `:=`(volatility = shift(rollapply(Q3_returns_data$Return, rolling_win, sd ,fill=NA, a
                       mean = shift(rollapply(Q3_returns_data$Return,
                       rolling_win, mean ,fill=NA, align="right")))]

# normalized return
Q3_returns_data[,normalized:= (Return - mean)/volatility]

summary <- Q3_returns_data[normalized != "NA"]
```

```r
# plot the distribution
hist(summary$normalized,xlab= "Normalized Gains", breaks = 50
     ,col = "green", main="Normalized Gains")
```



```r
hist(summary$Return,xlab= "Original Gains", breaks = 50, col = "blue", main="Origin Gains")
```



```r
normalizedmat <- c()
normalizedmat[1] <- mean(summary$normalized, na.rm = T)
normalizedmat[2] <- sd(summary$normalized,  na.rm = T)
normalizedmat[3] <- skewness(summary$normalized,  na.rm = T)
normalizedmat[4] <- kurtosis(summary$normalized, na.rm = T) - 3

result <- as.data.frame(round(normalizedmat,4), col=1, row.names =
                            c("Return", "Volatility","Skewness","Excess Kurtosis"))

originalmat <- c()
originalmat[1] <- mean(summary$Return, na.rm = T)
originalmat[2] <- sd(summary$Return,  na.rm = T)
originalmat[3] <- skewness(summary$Return,  na.rm = T)
originalmat[4] <- kurtosis(summary$Return, na.rm = T) - 3

result <- cbind(result,round(originalmat,4))

kable(result, col.names = c("Normalized Gains", "Original Return"), align = "l",caption = "Comparision 
```

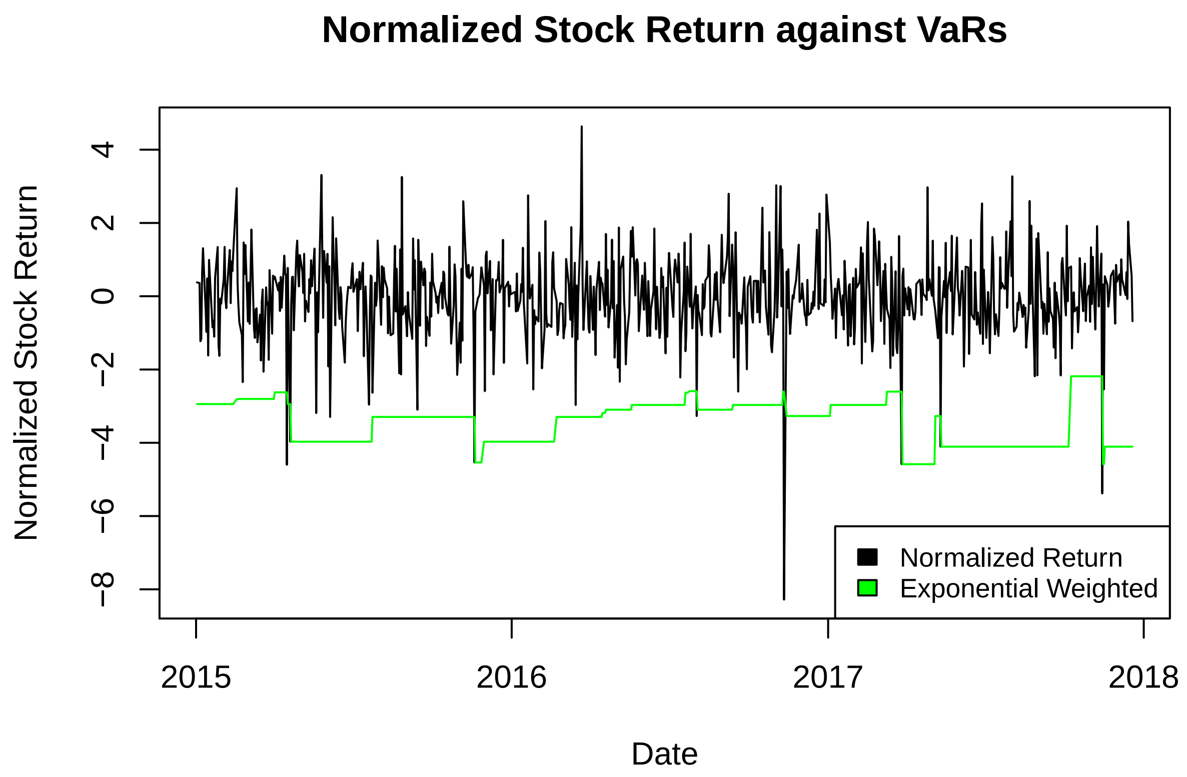
Table 1: Comparision of distribution of the normalized gain with the original ones

	Normalized Gains	Original Return
Return	-0.0034	-0.0019
Volatility	1.1546	0.0254
Skewness	-0.6990	-0.7630
Excess Kurtosis	4.0948	5.0444

4. Develop an approach to measure VaR which takes advantage of your response to the previous question.

We can try to develop a exponential weighted approach using normalized return from the previous year.

```
# calculating exponential Weighted VaR using normalized return
rollingWindow <- 251
lambda <- 0.995
Q3_returns_data[, exponentialWeightedVaR := shift(rollapply(Q3_returns_data$normalized, rollingWindow, f
  cal_exponential_VaR(return, c)
}, fill=NA, align="right"))]

Q3_returns_data_2005<- Q3_returns_data[Date >= "2015-01-01"]

plot(x = Q3_returns_data_2005$Date, y= Q3_returns_data_2005$normalized, type = "l",
     main="Normalized Stock Return against VaRs", xlab = "Date",
     ylab= "Normalized Stock Return")

lines(x= Q3_returns_data_2005$Date, y= Q3_returns_data_2005$exponentialWeightedVaR, col = "green")

legend("bottomright",legend=c("Normalized Return","Exponential Weighted"),fill=c("black","green"), cex =
```



```
cat("Number of Exceptions using normalized return: ",
    sum(Q3_returns_data_2005[,normalized < exponentialWeightedVaR ]))
```

```
## Number of Exceptions using normalized return:  8
```

Using exponential weighted on normalized return, we get 8 exceptions, which is less than both historical and exponentail weight on in qn1.

5

There are 748 data entries from 2015 to 2017. By taking 99% VaR on these observation, we should be seeing 8 exceptions. Hence I would suggest using the approach in part 4 (I called it normalized exponential weighted VaR) in this case.

Qn 2

1.

Consider this question as 8 balls and 3 drawers. In the worst case, there are 2 balls in each drawer. Hence, the whichever drawer the 7th ball is placed, the number of balls will be equal to 3.

2.

Assume it follows normal distribution

$$VaR_5 = \sigma_5 \Phi^{-1}(0.98) = 10$$

$$\sigma_5 = 10/\Phi^{-1}(0.98) = 4.869144$$

$$\sigma_{10} = \sigma_5 \sqrt{2}$$

Solve for:

$$VaR_{10} = \sigma_{10} \Phi^{-1}(0.99)$$

Hence

$$VaR_{10} = \sigma_5 \sqrt{2} \Phi^{-1}(0.99) = 16.01925$$

3.

Assume VaR = 0.99

Prob(More than one exception) = 1 - Prob(No Exception + 1 Exception)

$$= 1 - \Sigma_{k=0}^{1} \frac{20!}{k!(n-k)!}(1-c)^k c^{n-k}$$

where c = 0.99

```r
cat("Probability is:", 1- (pbinom(1,20, 0.01)))
```

```
## Probability is: 0.01685934
```

Definition of bunching:

A bunching of exceptions refers clustering of exception.

We could require a test to determine if there is bunching in the exceptions.

Set a threshold for punching, and use the binomial distribution to determine if the probability is higher than the threshold value. Say if the punching threshold is only 1 in a month and there are two exceptions. Then it is highly likely that punching exists.

4.

First Intuition is to use Lyft which IPOed recently as a comparative firm to do the analysis. However, as there are only 15 days, so this may not be a good measure due to short horizon.

As I have observed, the sector or subsector which Lyft is in outperforms the company by a huge margin. Hence, using the sector or subsector return is not going to reflect the VaR for Uber.

Instead, I would look at IPO ETFs to determine the VaR. The intuition is that Uber's return may be similar to a lot of newly IPOs firms. I chose Renaissance IPO ETF, that have the most allocation of Lyft and similar tech firms.

```r
suppressMessages(getSymbols("IPO"))
```

```
## [1] "IPO"
```

```r
IPO <- as.data.table(IPO)

IPO[, Return:= (IPO.Adjusted - shift(IPO.Adjusted)) /shift(IPO.Adjusted)]

rollingWindow <- 251
c <- 0.01

# Using the stress VaR method since we didn't use in the early exercise
# Rolling to get the 1% daily VaR first
IPO[, VaR := rollapply(Return, rollingWindow, function(Return){
        sorted_ret <- sort(Return)
        cutoff <- ceiling(rollingWindow * c)
        sorted_ret[cutoff]
}, fill=NA, align="right")]

# Rolling to get the min VaR
IPO[, MinVaR := rollapply(Return, rollingWindow, function(Return){
      min(Return,na.rm = T)
},fill=NA, align="right")]


cat("The number of the VaR: ", min(IPO$MinVaR,na.rm = T))
```

```
## The number of the VaR:  -0.04991152
```