Homework 1

Due on Friday, January 24th

Instructions:

- Install `pdflatex`, `R`, and `RStudio` on your computer.
- Please edit the `HW1_First_Last.Rnw` file in `Rstudio` and compile with `knitr` instead of `Sweave`. Go to the menu `RStudio|Preferences...|Sweave` choose the `knitr` option, i.e., `Weave Rnw files using knitr?` You may have to install `knitr` and other necessary packages.
- Replace "First" and "Last" in the file-name with your first and last names, respectively. Complete your assignment by modifying and/or adding necessary R-code in the text below.
- You should submit both the **data** and the *HW1_First_Last.Rnw* file in a zip-file in Canvas. The zip-file should be named "HW1_First_Last.zip" and it should contain all the necessary data files, the `HW1_First_Last.Rnw` and `HW1_First_Last.pdf` file, which was obtained from compiling `HW1_First_Last.Rnw` with `knitr` and LaTeX.

  NOTE: "First" is your first name and "Last" your last name.
- The GSI grader will unzip your file and compile it with Rstudio and knitr. If the file fails to compile due to errors other than missing packages, there will be an automatic 10% deduction to your score. Then, the GSI will proceed with grading your submitted PDF.
- **Data.** The zip-file should include also the ".csv" file with the stock market data you downloaded frin WRDS so that when everything is unzipped in one folder, the GSI is able to compile and reproduce your PDF.

Problems:

1. **(a)** Read the tutorial on accessing and downloading data from WRDS available on this link.

   (b) Execute a query in Center for Research in Security Prices, LLC (CRSP) database to download the closing daily prices of the stocks with tickers `TSLA` and `NVDA` for the period Jan 1, 2022 through Dec 31, 2024.

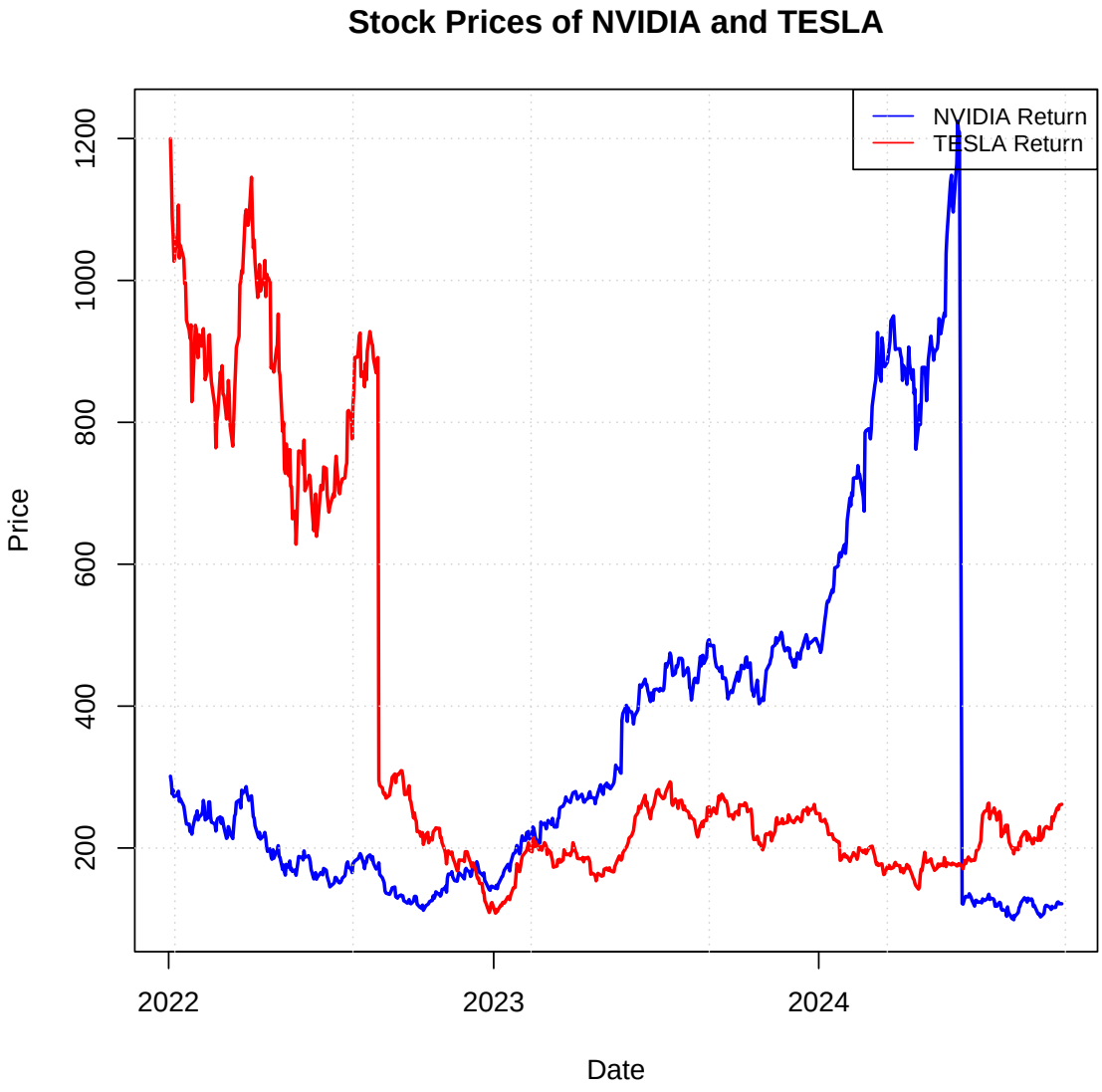
   (c) Place the resulting `csv` file in a desired folder in your computer and modify the following code to produce plots of daily `prices`, `returns`, `log-returns`, and a scatter-plot of the log-returns of `TSLA` against those of `NVDA`.

   Identify the times of splits if any for the `TSLA` and `NVDA` stock in this period, if any.

   Uncomment, i.e., remove the '%', in the following lines when ready to compile. Remove also `\begin{verbatim}` *and* `\end{verbatim}` *in the ".Rnw" file.*

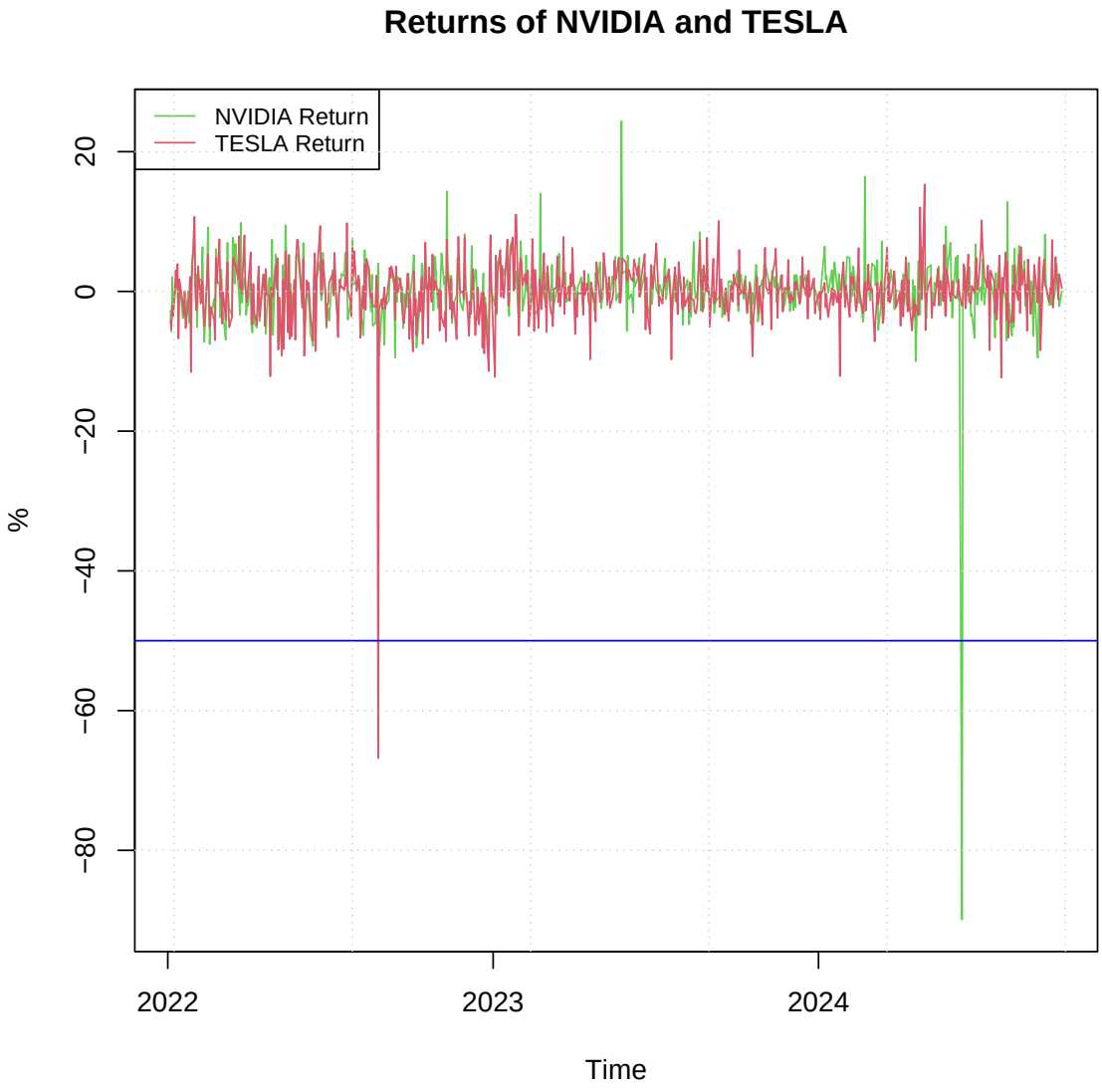
```
dat = read.csv("Daily_Stock_Price.csv",header=TRUE)

stock_companies = c("NVIDIA CORP", "TESLA INC")
stocks = list()
```

```r
times = list()

for (company in stock_companies) {
  stock_data = subset(dat, dat$COMNAM == company)
  stocks[[company]] = stock_data$PRC
  times[[company]] = as.Date(as.character(stock_data$date), "%Y-%m-%d")
}

# Price of NVDA & TSLA
par(mfrow=c(1, 1))
y_min = min(c(min(stocks$"NVIDIA CORP"), min(stocks$"TESLA INC")))
y_max = max(c(max(stocks$"NVIDIA CORP"), max(stocks$"TESLA INC")))

plot(times$"NVIDIA CORP", stocks$"NVIDIA CORP", type = "l", col = "blue", lwd = 2,
     main = "Stock Prices of NVIDIA and TESLA",
     xlab = "Date", ylab = "Price", ylim = c(y_min, y_max))
lines(times$"TESLA INC", stocks$"TESLA INC", col = "red", lwd = 2)
legend("topright", legend=c("NVIDIA Return", "TESLA Return"),
       lty=1, col=c("blue", "red"), cex=0.8)
grid()
```




```r
# Return of NVDA & TSLA
par(mfrow=c(1,1))

n_NVDA = length(stocks$"NVIDIA CORP")
R.NVDA = stocks$"NVIDIA CORP"[-1] / stocks$"NVIDIA CORP"[-n_NVDA] - 1
R.TSLA = stocks$"TESLA INC"[-1] / stocks$"TESLA INC"[-length(stocks$"TESLA INC")] - 1

y_min = min(c(min(R.NVDA)*100, min(R.TSLA)*100))
y_max = max(c(max(R.NVDA)*100, max(R.TSLA)*100))

plot(times$"NVIDIA CORP"[-1], 100*R.NVDA, type="l", col=3, xlab="Time", ylab="%",
     ylim = c(y_min, y_max), main="Returns of NVIDIA and TESLA")
```

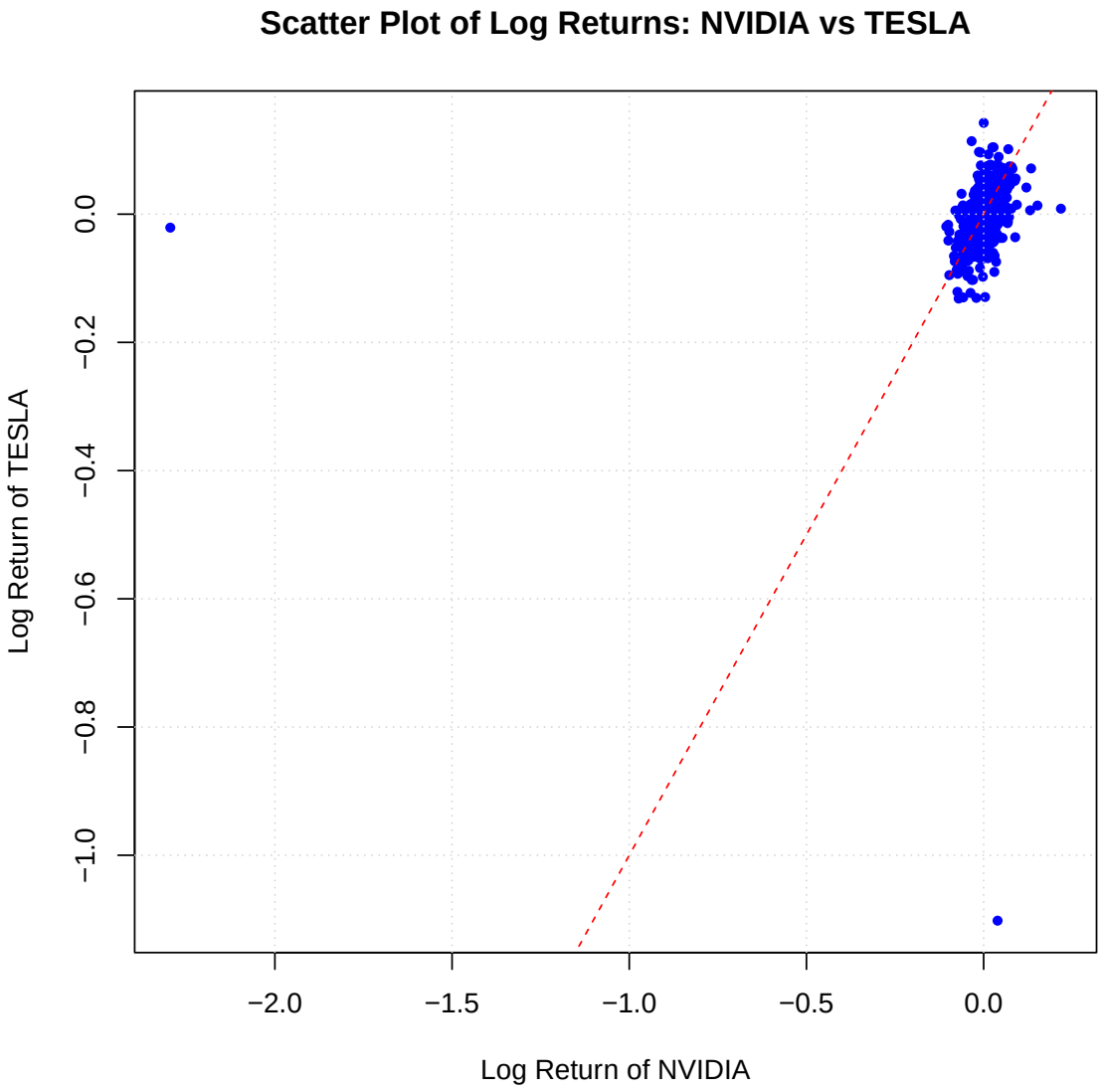
```
lines(times$"TESLA INC"[-1], 100*R.TSLA, col=2)
legend("topleft", legend=c("NVIDIA Return", "TESLA Return"),
       lty=1, col=c(3, 2), cex=0.8)
abline(h=-50, col="blue")
grid()
```




```
# Log Return of NVDA & TSLA
par(mfrow=c(1,1))

r.NVDA = diff(log(stocks$"NVIDIA CORP"))
r.TSLA = diff(log(stocks$"TESLA INC"))
```

```
y_min = min(c(min(r.NVDA)*100, min(r.TSLA)*100))
y_max = max(c(max(r.NVDA)*100, max(r.TSLA)*100))

plot(times$"NVIDIA CORP"[-1], 100*r.NVDA, type="l", col=3, xlab="Time", ylab="%",
     ylim = c(y_min, y_max), main="Log Returns of NVIDIA and TESLA")
lines(times$"TESLA INC"[-1], 100*r.TSLA, col=2)
legend("bottomright", legend=c("NVIDIA Log Return", "TESLA Log Return"),
       lty=1, col=c(3, 2), cex=0.8)
abline(h=-50, col="blue")
```

```r
# Scatter Plot of Log Return
par(mfrow=c(1,1))
plot(r.NVDA, r.TSLA, xlab="Log Return of NVIDIA", ylab="Log Return of TESLA",
     main="Scatter Plot of Log Returns: NVIDIA vs TESLA", pch=16, col="blue", cex=0.8)

abline(0, 1, col="red", lty=2)
grid()
```




```r
# Identify Extreme Drop
split_date_NVDA = as.character(times$`NVIDIA CORP`[which(R.NVDA < -0.5) + 1])
split_date_TSLA = as.character(times$`TESLA INC`[which(R.TSLA < -0.5) + 1])
```

```r
cat("Split date : ")
```

```
## Split date :
```

```r
cat("NVDA : ", ifelse(length(split_date_NVDA) == 0, 'N/A', split_date_NVDA))
```

```
## NVDA :  2024-06-10
```

```r
cat("TSLA : ", ifelse(length(split_date_TSLA) == 0, 'N/A', split_date_TSLA))
```

```
## TSLA :  2022-08-25
```

```r
# Identify Split Ratio
ratio_NVDA = round(1 / (R.NVDA[which(R.NVDA < -0.5)] + 1), 0)
ratio_TSLA = round(1 / (R.TSLA[which(R.TSLA < -0.5)] + 1), 0)
cat("At", split_date_TSLA, "TSLA declared a", ratio_TSLA, 'in 1 split')
```

```
## At 2022-08-25 TSLA declared a 3 in 1 split
```

```r
cat("At", split_date_NVDA, "NVDA declared a", ratio_NVDA, 'in 1 split')
```

```
## At 2024-06-10 NVDA declared a 10 in 1 split
```

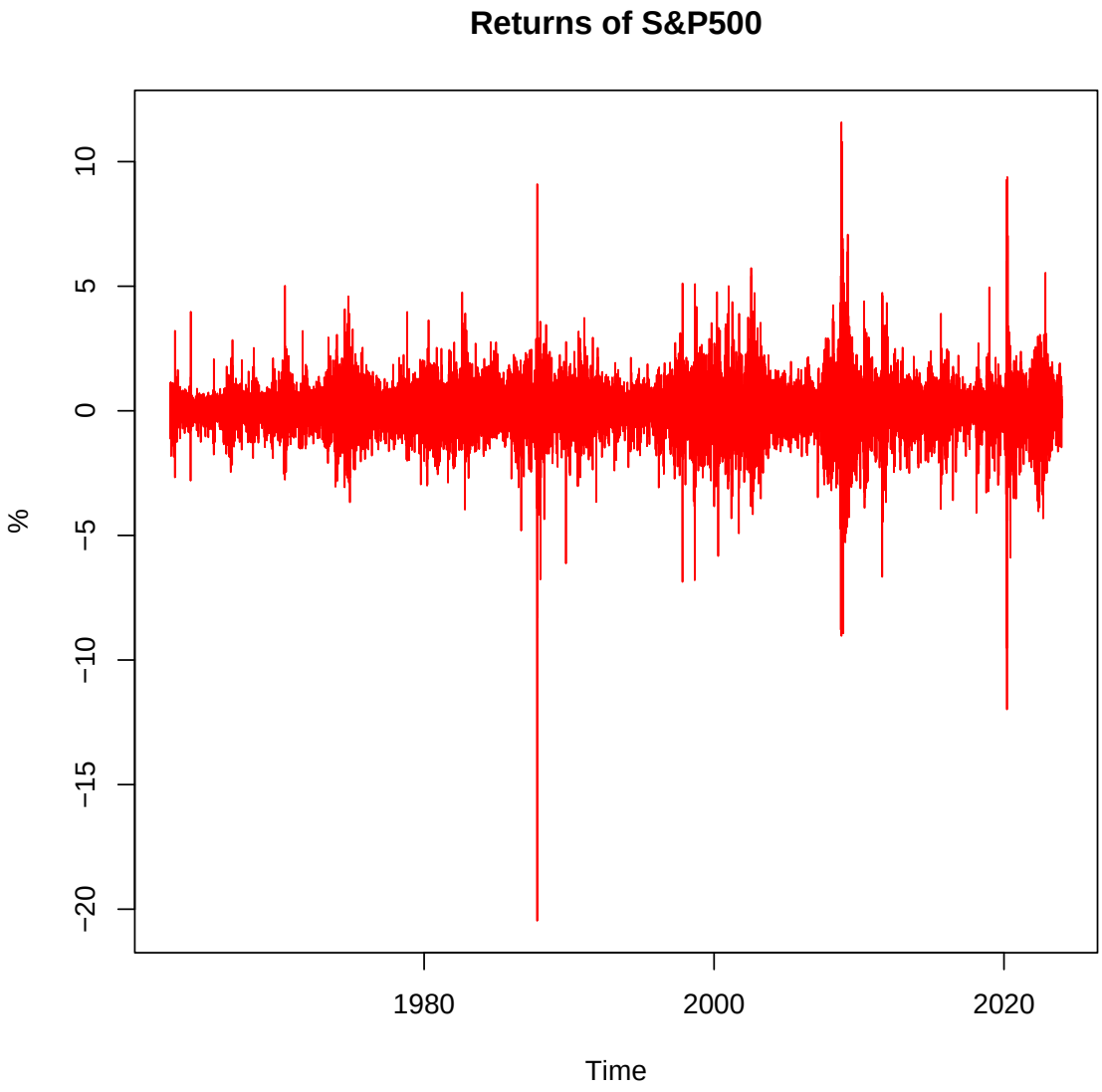
At 2022-08-25 TSLA declared a 3-in-1 split, which aligned with this link. And at 2024-06-10 NVDA declared a 10 in 1 split, which aligned with this link

2. Download the file "`sp500_full.csv`" available in the `Data` folder of the `Files` section in `Canvas`. It provides the data on daily prices and returns (among other things) of the S&P 500 index. This is the same data set used in lectures.

   (a) Plot the time-series of daily returns and daily log-returns. You need to identify which variable gives the returns and also compute the log-returns from the daily closing values of the index. Comment on the extreme drops in daily returns/log-returns. Can they be associated with known market events?

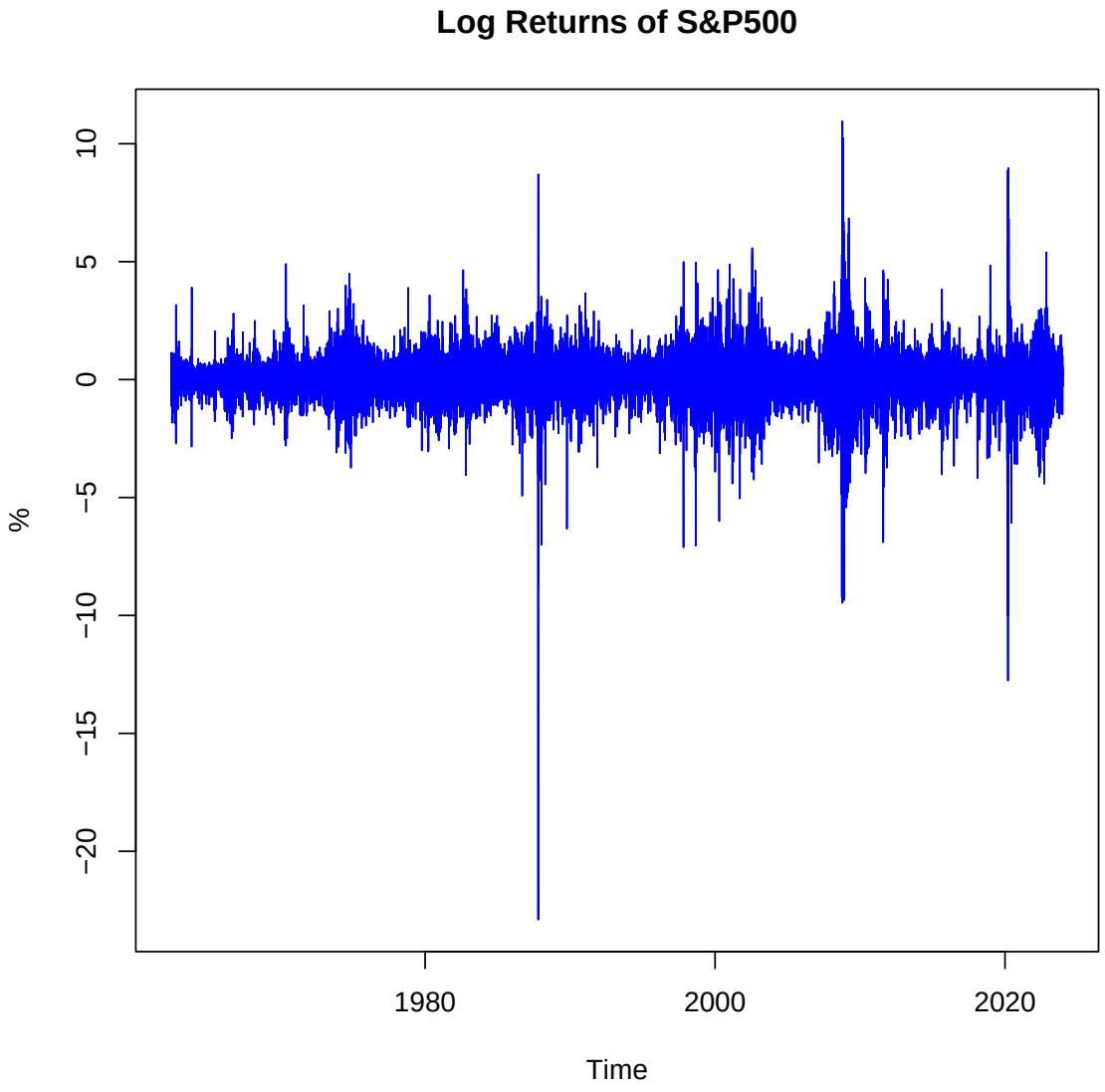
```r
dat_2 = read.csv("sp500_full.csv",header=TRUE)
dat_2 = dat_2[1386:16865, ]

# Return of S&P500
par(mfrow=c(1,1))

time = as.Date(as.character(dat_2$caldt), format = "%Y%m%d")
return = dat_2$sprtrn
```

```r
y_min = min(return*100, na.rm=TRUE)
y_max = max(return*100, na.rm=TRUE)

plot(time, 100*return, type="l", col="red", xlab="Time", ylab="%",
     ylim = c(y_min, y_max), main="Returns of S&P500")
abline(h=-50, col="blue")
```




```r
# Log Return of S&P500
par(mfrow=c(1,1))

log_return = log(1 + dat_2$sprtrn)
```

```r
y_min = min(log_return*100, na.rm=TRUE)
y_max = max(log_return*100, na.rm=TRUE)

plot(time, 100*log_return, type="l", col="blue", xlab="Time", ylab="%",
     ylim = c(y_min, y_max), main="Log Returns of S&P500")
```



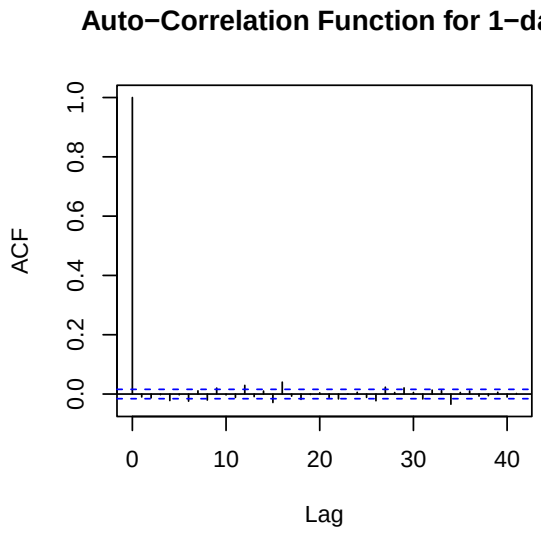

```r
# Identify huge drop in return
threshold <- -0.10

extreme_drops_indices <- which(return < threshold)
```

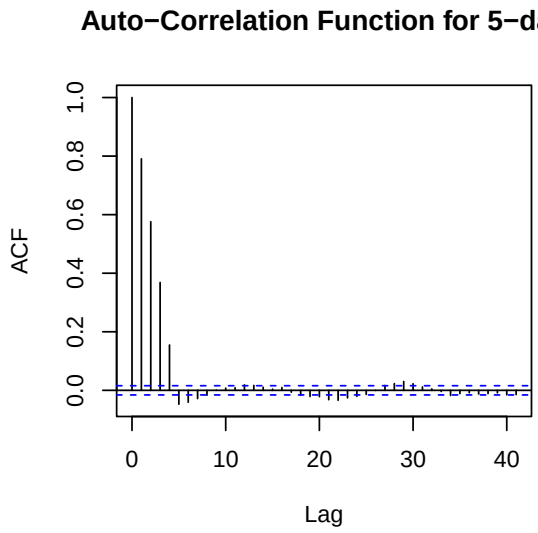
```r
extreme_dates <- as.character(time[extreme_drops_indices])
extreme_returns <- return[extreme_drops_indices]

if (length(extreme_dates) > 0) {
  cat("Extreme drops in returns:\n")
  for (i in seq_along(extreme_dates)) {
    cat("Date:", extreme_dates[i], "Return:", extreme_returns[i] * 100, "%\n")
  }
}

## Extreme drops in returns:
## Date: 1987-10-19 Return: -20.4669 %
## Date: 2020-03-16 Return: -11.9841 %
```

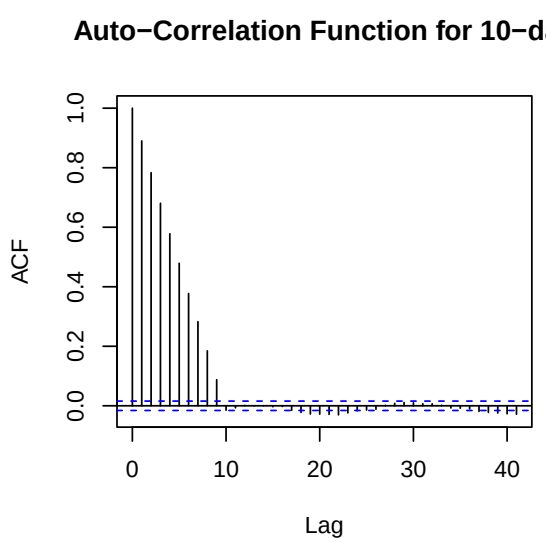
At 1987-10-19, there is a huge drop in return -20.47%, which aligned with the date of Black Monday. And the date of 2020-03-16 is the outbreak of COVID pandemic, causing the -11.98% of return

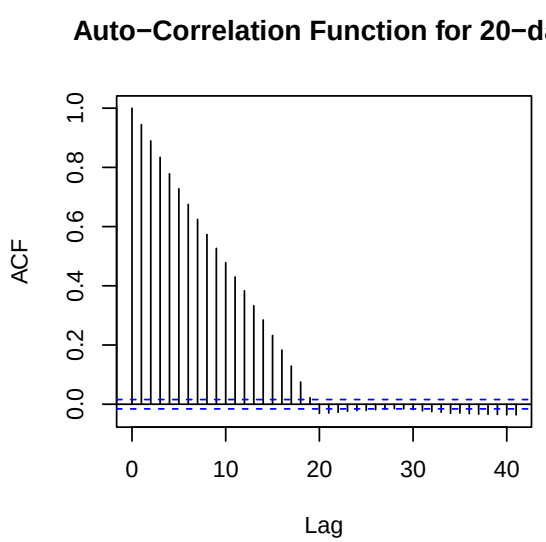
(b) Plot the auto-correlation functions of the 1-day (i.e., daily), 5-day, 10-day and 20-day log-returns and their squares.

```r
log_return_1 = na.omit(log(dat_2$spindx / lag(dat_2$spindx,1)))
log_return_5 = na.omit(log(dat_2$spindx / lag(dat_2$spindx,5)))
log_return_10 = na.omit(log(dat_2$spindx / lag(dat_2$spindx,10)))
log_return_20 = na.omit(log(dat_2$spindx / lag(dat_2$spindx,20)))

par(mfrow=c(2,2))
acf(log_return_1, main="Auto-Correlation Function for 1-day")
acf(log_return_5, main="Auto-Correlation Function for 5-day")
acf(log_return_10, main="Auto-Correlation Function for 10-day")
acf(log_return_20, main="Auto-Correlation Function for 20-day")
```

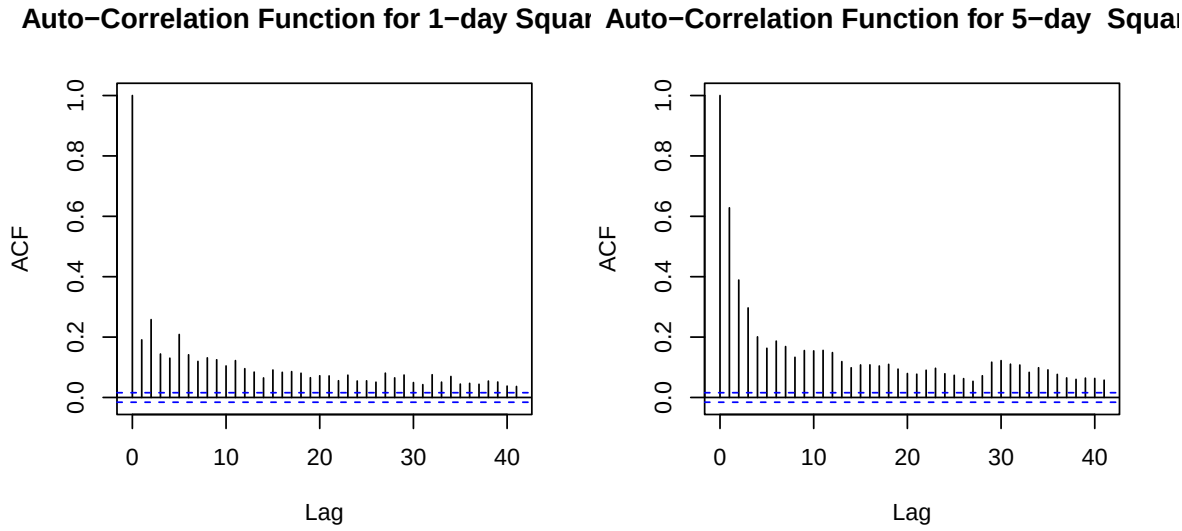


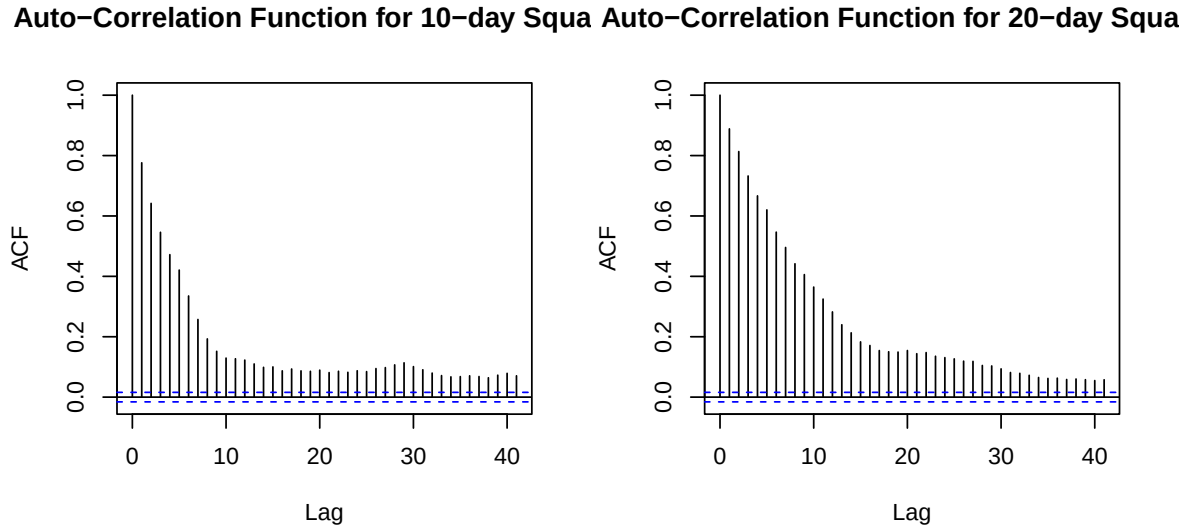







```r
par(mfrow=c(2,2))
acf(log_return_1**2, main="Auto-Correlation Function for 1-day Squared")
acf(log_return_5**2, main="Auto-Correlation Function for 5-day  Squared")
acf(log_return_10**2, main="Auto-Correlation Function for 10-day Squared")
acf(log_return_20**2, main="Auto-Correlation Function for 20-day Squared")
```





(c) Provide a table of the basic summary statistics for the 4 time series from part **(b)**. That is, compute the minimum, 1st quartile, the median, the 3rd quartile and the maximum, for the k-day log-returns for $k = 1, 5, 10$, and 20. You can modify the code in the ".Rnw" file that produces:

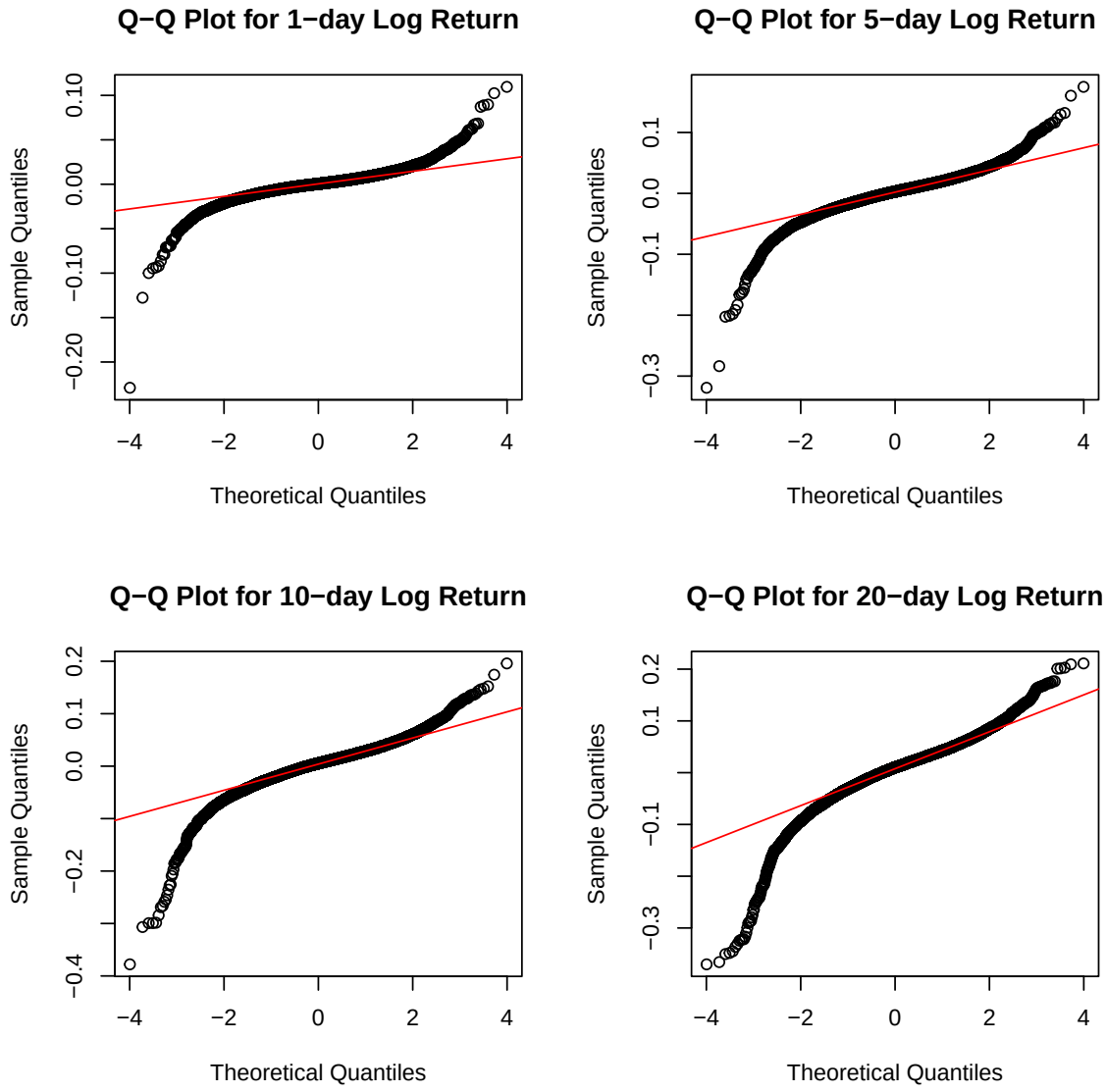
```
returns = list("r1"=log_return_1,"r5"=log_return_5, "r10"=log_return_10, "r20"=log_return_
sum.stat = lapply(returns,summary)
x = matrix(0,nrow=4,ncol=6,dimnames=list(c("1-day","5-day","10-day","20-day"),
                                         names(sum.stat$r1)))
x[1,]=sum.stat$r1
x[2,]=sum.stat$r5
```

```
x[3,]=sum.stat$r10
x[4,]=sum.stat$r20
kable(x)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1-day	-0.2289972	-0.0043132	0.0004639	0.0002873	0.0052056	0.1095720
5-day	-0.3192136	-0.0103098	0.0028846	0.0014341	0.0142628	0.1748870
10-day	-0.3778684	-0.0129092	0.0055208	0.0028590	0.0206999	0.1958822
20-day	-0.3702510	-0.0164150	0.0100448	0.0057059	0.0317122	0.2110296

3. **(a)** Using the same data set as in the previous problem, make a 2×2 array of normal quantile-quantile plots. You can uncomment and and modify the following code

```
par(mfrow=c(2,2))
qqnorm(returns$r1, main = "Q-Q Plot for 1-day Log Return")
qqline(returns$r1,col="red")
qqnorm(returns$r5, main = "Q-Q Plot for 5-day Log Return")
qqline(returns$r5,col="red")
qqnorm(returns$r10, main = "Q-Q Plot for 10-day Log Return")
qqline(returns$r10,col="red")
qqnorm(returns$r20, main = "Q-Q Plot for 20-day Log Return")
qqline(returns$r20,col="red")
```



(b) What can you say about the tails of the returns? **The QQ plots of 1, 5, 10, and 20-day log returns exhibit deviations from the normality line, particularly in the tails, indicating that returns have heavier tails than a normal distribution. And over longer periods, individual daily returns can compound, leading to larger aggregated returns. This magnifies both gains and losses, making the tails more pronounced.**

4. **(a)** Go to the US Treasury web-site and look up the annual yield to maturity of US government issued zero-coupon bonds of maturities 3 months, 5 years and 30 years, respectively. What are the names of these 3 types of securities?
Annual yield of 3 months zero-coupon bond (Treasury bills) is: 4.34% Annual yield of 5 years zero-coupon bond (Treasury notes) is: 4.42% Annual yield of 30

years zero-coupon bond (Treasury bonds) is: 4.84% The above information is based on US Treasury web-site at Jan 17, 2025.

(b) Suppose that a bond with maturity $T = 30$ years, `PAR` \$1000 and annual coupon payments C is selling precisely at its par value. Determine the value of the coupon payments C. You can assume that the yield to maturity is the one you found in part **(a)**, corresponding to maturity $T = 30$.

```
r = 0.0484
PAR = 1000
C = uniroot(function(C) (C/r + (PAR - C/r)/(1+r)^30) - 1000, interval=c(1,1000))$root

cat("Annual coupon payments : $", C)

## Annual coupon payments : $ 48.4
```

(c) Use the R-function `uniroot` to find the yield to maturity of a 30-year, par \$1000 bond with annual coupon payments of \$40, which is selling for \$1200 now. Provide the R-code as well as the output.

```
PAR = 1000
C = 40
V = 1200
T = 30
r = uniroot(function(r) (C/r + (PAR - C/r)/(1+r)^T) - V, interval=c(0.0001,1))$root

cat("Annual yield :", r*100, '%')

## Annual yield : 2.985027 %
```

Hint: Write an R-function that computes the price of a bond with yield to maturity r, annual coupon payments C, par value `PAR` and maturity T. You can borrow the function definition from the lectures. You can then use this function in your call to `uniroot` as I did in class, for example.

5. Suppose (although this is dangerously far from reality) that the daily log-returns of the S&P 500 are independent and identically distributed $N(\mu, \sigma^2)$.

   (a) Using the data in file "`sp500_full.csv`", estimate μ and σ with the sample mean and standard deviation.

```
mu = mean(log_return, na.rm = TRUE)
sigma = sd(log_return, na.rm = TRUE)
cat("Mean : ", mu)

## Mean :  0.0002873079
```

```r
cat("Standard Deviation : ", sigma)
```

```
## Standard Deviation :  0.01040871
```

(b) Using the independence and Normality assumptions, compute the value-at-risk at levels $\alpha = 0.05$ and $\alpha = 0.01$, i.e., $\text{VaR}_{0.05}$ and $\text{VaR}_{0.01}$ for the k-day log-returns, where $k = 1, 5, 10$ and 20.

```r
alpha_005 <- 0.05  # 5% VaR
alpha_001 <- 0.01  # 1% VaR

returns = list(log_return_1, log_return_5, log_return_10, log_return_20)
k = list(1, 5, 10, 20)
VaR_001 = list()
VaR_005 = list()

for (i in 1:4){
  mu_k = mu * k[[i]]
  sigma_k = sqrt(k[[i]]) * sigma
  VaR_005[[i]] = -(mu_k + qnorm(alpha_005) * sigma_k)
  VaR_001[[i]] = -(mu_k + qnorm(alpha_001) * sigma_k)

  # Print results for each k
  cat("For k =", k[[i]], "days:\n")
  cat("VaR at alpha = 0.05 (5% level):", VaR_005[[i]], "\n")
  cat("VaR at alpha = 0.01 (1% level):", VaR_001[[i]], "\n\n")
}
```

```
## For k = 1 days:
## VaR at alpha = 0.05 (5% level): 0.0168335
## VaR at alpha = 0.01 (1% level): 0.02392697
##
## For k = 5 days:
## VaR at alpha = 0.05 (5% level): 0.03684675
## VaR at alpha = 0.01 (1% level): 0.05270824
##
## For k = 10 days:
## VaR at alpha = 0.05 (5% level): 0.05126766
## VaR at alpha = 0.01 (1% level): 0.07369921
##
## For k = 20 days:
## VaR at alpha = 0.05 (5% level): 0.07082041
## VaR at alpha = 0.01 (1% level): 0.1025434
```

Hint: What is the distribution of the k-day log-returns, under the assumptions in the problem.

(c) Provide a 2-row table of the proportion of times the k-day log-returns were lower than the $-VaR_\alpha$ values computed in part **(b)** for $\alpha = 0.05$ and $\alpha = 0.01$. That is, each entry in the table is the proportion of time the k-day log-losses were worse than the corresponding value-at-risk.

```
p = matrix(0,nrow=2,ncol=4)
colnames(p) = c("1-day","5-day","10-day","20-day")
rownames(p) = c("0.05", "0.01")

for (i in 1:4) {
  # Calculate the proportion for 0.05 alpha (5% VaR)
  p[1, i] = mean(returns[[i]] < -VaR_005[[i]])
  # Calculate the proportion for 0.01 alpha (1% VaR)
  p[2, i] = mean(returns[[i]] < -VaR_001[[i]])
}
# Put your code here that populates p with the desired empirical proportions
kable(p)
```

	1-day	5-day	10-day	20-day
0.05	0.0407003	0.0443942	0.0421461	0.0439198
0.01	0.0166031	0.0155735	0.0153846	0.0177878

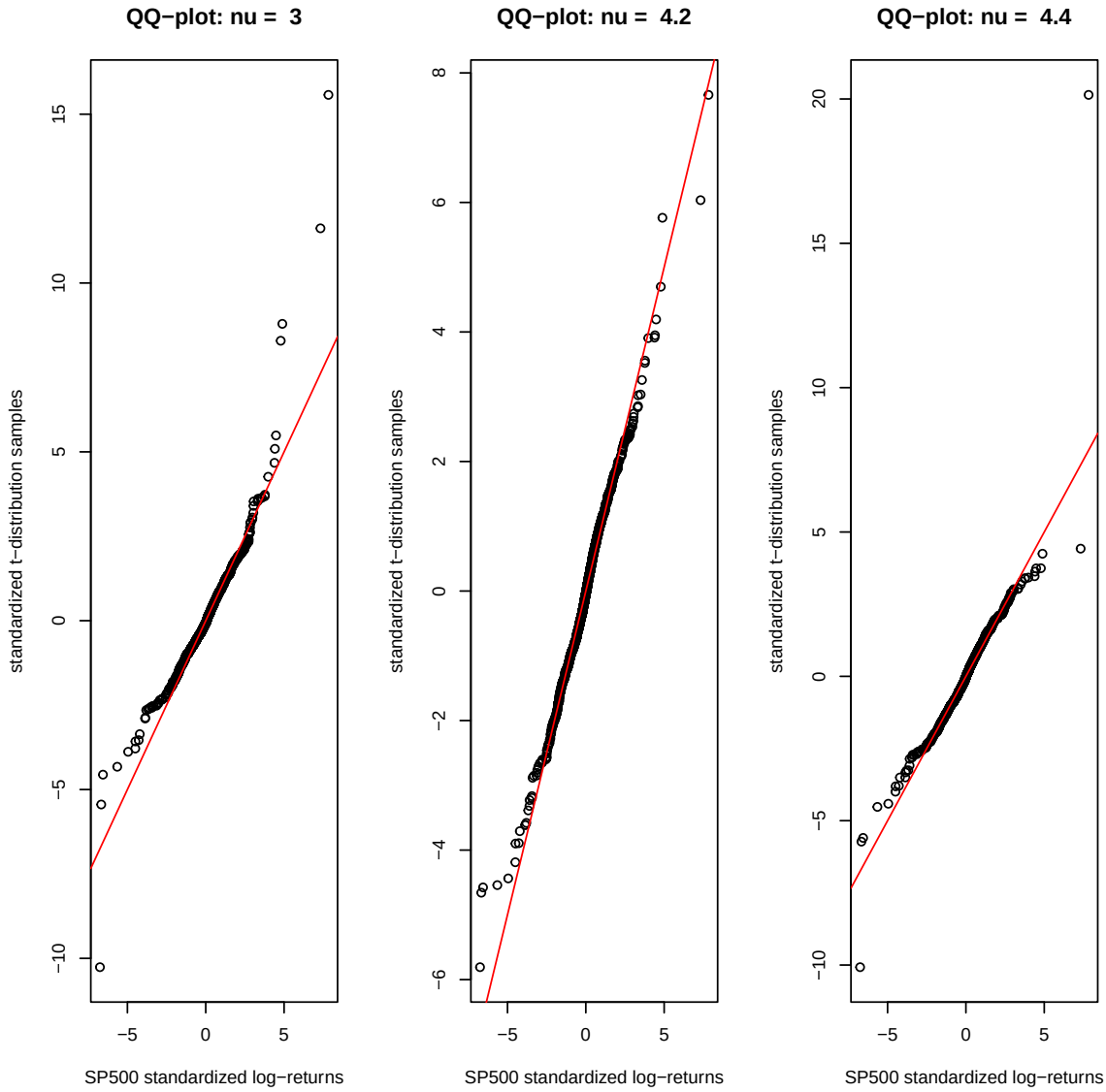
Comment on the results. What should the values of these empirical frequencies be for models that agree with the data? **These proportions are compared to the theoretical probabilities of 5% and 1%, respectively. And the empirical value should be closed to 0.01 and 0.05**

6. In this problem we will first find a slightly more appropriate model for the log-returns of the SP500 index data set. The idea is to replace the increments of the normal random walk model in the previous problem with t-distribued increments. We will do so by trial and error and eyeballing QQ-plots. More sophisticated inference methods will come shortly.

(a) As in Problem 2 above, load the SP500 data set from file `sp500_full.csv`. Select a consecutive period of 10 years. Standardize the log-returns in this period and produce a series of QQ-plots of the daily log-returns against simulated samples from the standardized t-distribution for varying degrees of freedom $\nu = 2.1, 3, 4, \ldots$. Pick a value for the degrees of freedom $\nu > 2$ that result in the QQ-plot with the best fit. Produce a QQ-plot with this "best fit" value as well as with two other values of ν – one smaller and the other larger. You can use some of the following code as a template and modify it as you see fit. **Comment on why the value of ν you chose is reasonable.**

```
t.sp = as.Date(x=as.character(dat_2$caldt),format="%Y%m%d")
idx = which(t.sp>as.Date("2000-01-01") & t.sp < as.Date("2010-01-01"))
rt = diff(log(dat_2$spindx[idx]));
mu = mean(rt);
sig = sd(rt);
```

```
rt_standardized = (rt-mu)/sig;

# The following function produces the desired QQ-plots.

qqplot_against_std <- function(standardized_log_returns,nu = 2.1){
 require(fGarch)
 n = length(standardized_log_returns);
 qqplot(standardized_log_returns,fGarch::rstd(n,nu = nu),
       xlab="SP500 standardized log-returns",
       ylab = "standardized t-distribution samples", main=paste("QQ-plot: nu = ",nu));
abline(a=0,b=1,col="red")
}

par(mfrow=c(1,3))
qqplot_against_std(rt_standardized, nu = 3)
qqplot_against_std(rt_standardized, nu = 4.2)
qqplot_against_std(rt_standardized, nu = 4.4)
```



The selected value $\nu = 4.2$ visually minimizes deviations across the full range of returns in the QQ-plot. The QQ-plot with $\nu = 4.2$ shows a significantly better alignment between the observed data and the simulated t-distribution, particularly in the tail regions where the normal distribution assumption often fails to capture extreme returns.

(b) In the previous part, we have obtained using graphical methods a rough fit of the SP500 log-returns via the follwing model:

$$r_t = \log(S_t) - \log(S_{t-1}) = \mu + \sigma\epsilon_t, \tag{1}$$

where ϵ_t follow the standardized t-distribution with ν degrees of freedom and S_t is the value of the SP500 index on day t.

Read **Problem 4** on page 13 of Ruppert & Matteson.

Modify the code therein where the returns are modeled as in (1) and write an R-function that takes as an input the mean μ and the standard deviation σ of the log-returns, the parameter ν of the degrees of freedom for the standardized t-distribution, and `niter`. The output of the function should be the simulation-based estimate of the probability of default for the hedge fund.

```
calc_default_prob = function(mu, sigma, v, niter = 1e5)
{
  below = rep(0, niter)
  set.seed(2009)

  for (i in 1:niter) {
    r = mu + sigma * rt(45, df = v)  # log-returns based on t-distribution
    logPrice = log(1e6) + cumsum(r)
    minlogP = min(logPrice)
    below[i] = as.numeric(minlogP < log(950000))
  }
  return(mean(below))
}
```

Produce a table or a graph with the default probabilities as a function of $\nu > 2$ and the volatility σ. Comment on how do the volatility and the weight of the tail (represented by the parameter ν) affect the default probability. **Note:** Initially, use $\mu = 0.05/253$ and $\sigma = 0.23/\sqrt{253}$ as in the text and vary ν. Then you can fix a value of ν and vary σ. Feel free to also produce heat-map images or 2-way tables of the default probability for a range of values of ν and σ.

```
# Set initial parameters for mu and sigma
# mu <- 0.05 / 253
# sigma <- 0.23 / sqrt(253)
#
# v_values <- seq(3, 5, by = 0.1)  # degrees of freedom from 3 to 5
# sigma_values <- seq(0.1, 0.5, by = 0.05)  # volatility from 0.1 to 0.5
#
# # Create a matrix to store the probabilities
# probabilities_v <- matrix(NA, nrow = length(v_values), ncol = length(sigma_values))
# rownames(probabilities_v) <- v_values
# colnames(probabilities_v) <- sigma_values
#
# # Loop through each combination of v and sigma, calculating the default probability
# for (v in v_values) {
#   for (s in sigma_values) {
#     probabilities_v[as.character(v), as.character(s)] <- calc_default_prob(mu, s, v)
#   }
# }
```

```
#
# # Print the probabilities table
# print(probabilities_v)
#
# probabilities_v_long <- as.data.frame(as.table(probabilities_v))
# colnames(probabilities_v_long) <- c("v", "sigma", "probability")
#
# ggplot(probabilities_v_long, aes(x = sigma, y = v, fill = probability)) +
#   geom_tile() +
#   scale_fill_gradient(low = "white", high = "red") +
#   theme_minimal() +
#   labs(title = "Default Probability Heatmap",
#        x = "Volatility (sigma)",
#        y = "Degrees of Freedom (v)",
#        fill = "Probability")
```

We can notice that when we fix the degree of freedom ν, when the volatility larger, the risk of lossing 950000 is getting larger as well. This observation aligns with our instinct that when the stock is more easily to fluctuate, the risk of lossing money is higher. And smaller ν increases default probability due to heavier tails, while larger ν reduces it by approximating normal distribution behavior. This makes ν an important parameter when modeling risk with fat-tailed return distributions.