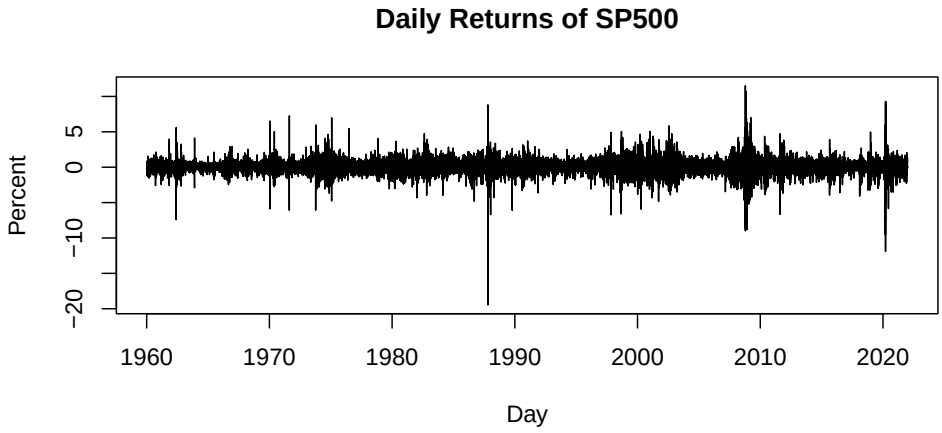
Time-Series Analysis and Forecasting of the S&P 500 Index

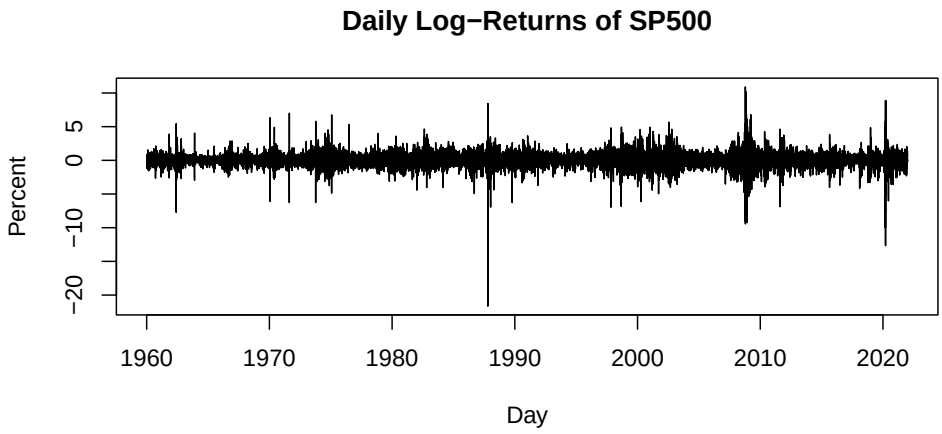
Please download the file "`sp500_full.csv`". It provides the data on daily prices and returns (among other things) of the S&P 500 index from 1957-01-02 to 2021-12-31.

Data Preprocessing and Exploratory Data Analysis

1. 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.

```
sp = read.csv("sp500_full.csv", header = T)
t.sp = as.Date(x=as.character(sp$caldt),format="%Y%m%d")
sp = sp[sp$caldt >= "19600101",]    # start from 1960-01
t.sp = t.sp[t.sp >= "1960-01-01"]
# Getting the times

n = length(sp$usdval)
R.sp = sp$usdval[-1]/sp$usdval[-n] - 1
r.sp = diff(log(sp$usdval))
# Calculate daily returns and log-returns
par(mfrow=c(2,1))
plot(t.sp[-1], R.sp*100, type="l",xlab="Day",ylab="Percent",
     main="Daily Returns of SP500")
plot(t.sp[-1], r.sp*100, type="l",xlab="Day",ylab="Percent",
     main="Daily Log-Returns of SP500")
```



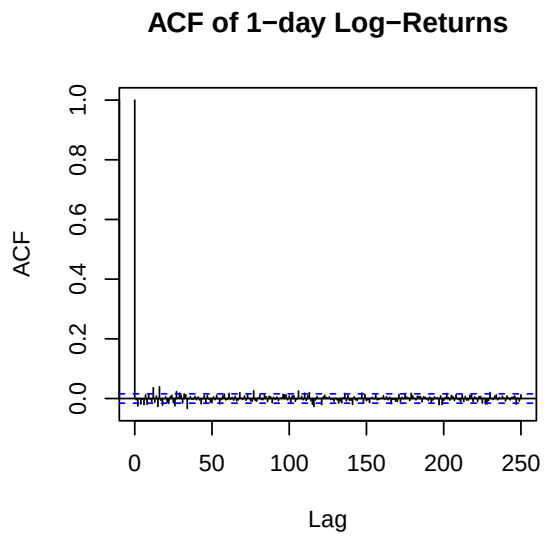


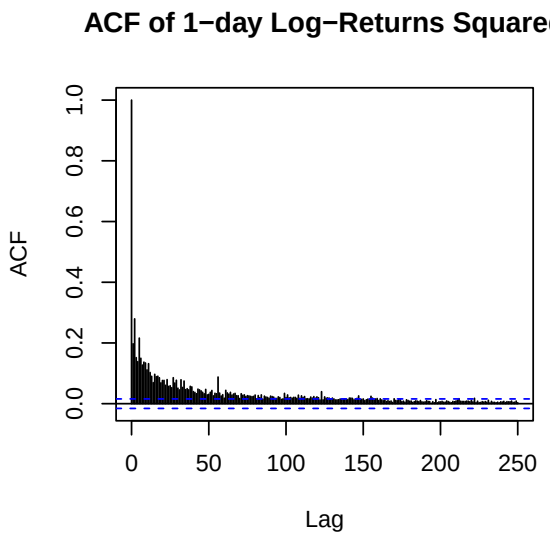
- In the code above, the returns and log-returns are computed from the daily closing values stored in `usdval`.
- From the generated plots above, it can be noticed that there are several large drops in the daily returns. At the end of 1980s is the stock market crash known as Black Monday on October 19th, 1987. In the late 1990s and early 2000s, Dot-com bubble causes several significant losses of the daily returns. From December 2007 to June 2009, Sub-prime mortgage meltdown contributes to a severe US recession.

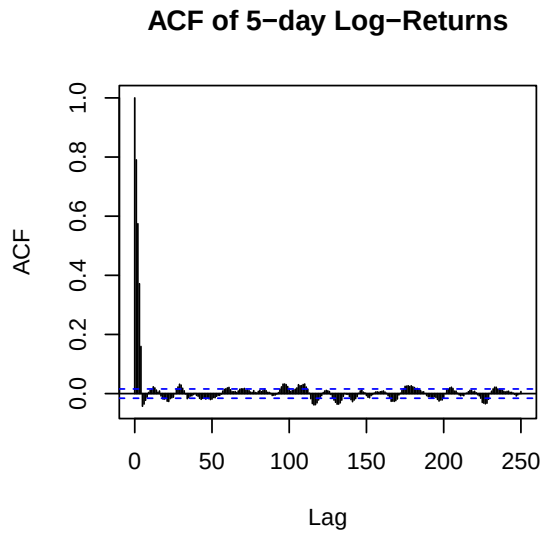
2. Plot the auto-correlation functions of the 1-day (i.e., daily), 5-day, 10-day and 20-day log-returns and their squares.

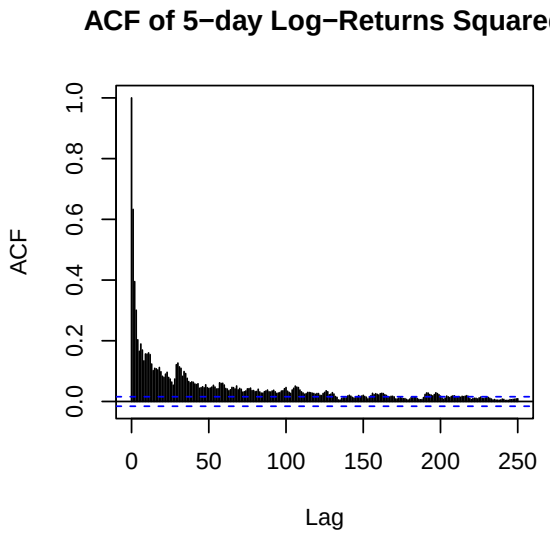
```
# calculate autocorrelation function --> measure the degree of correlation
# between the log-returns at different time lags.
r1.sp = diff(log(sp$usdval))
r5.sp = diff(log(sp$usdval), lag=5)
r10.sp = diff(log(sp$usdval), lag=10)
r20.sp = diff(log(sp$usdval), lag=20)

# 1. plot acf of the 1-day log-return and log-return squares
```

```r
par(mfrow=c(2, 2))
acf(r1.sp, main = "ACF of 1-day Log-Returns", lag.max=250)
acf(r1.sp^2, main = "ACF of 1-day Log-Returns Squared", lag.max=250)

# 2. plot acf of the 5-day log-return and log-return squares
acf(r5.sp, main = "ACF of 5-day Log-Returns", lag.max=250) # lag: specify the time lag
acf(r5.sp^2, main = "ACF of 5-day Log-Returns Squared", lag.max=250)
```



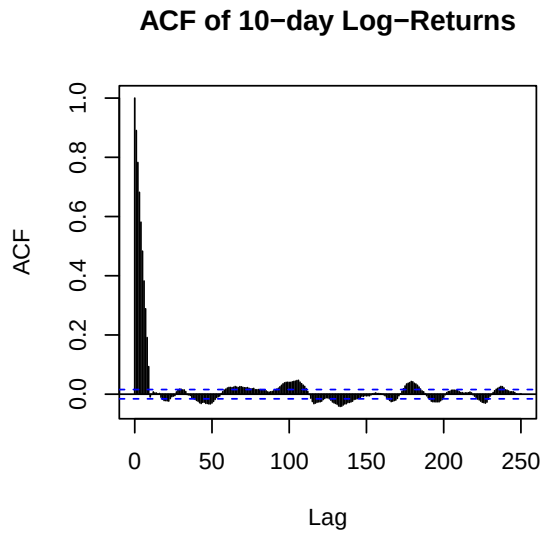


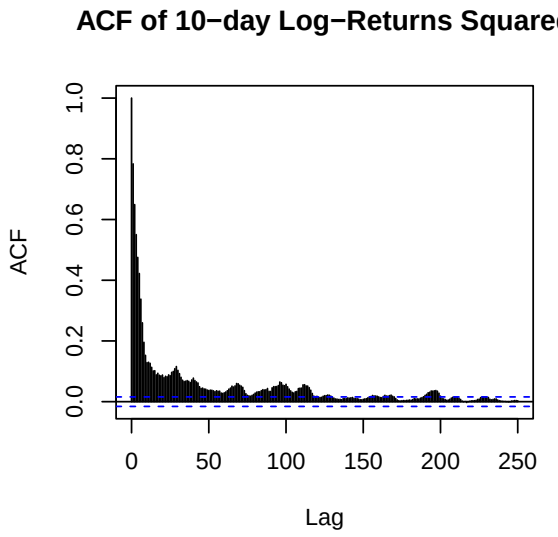


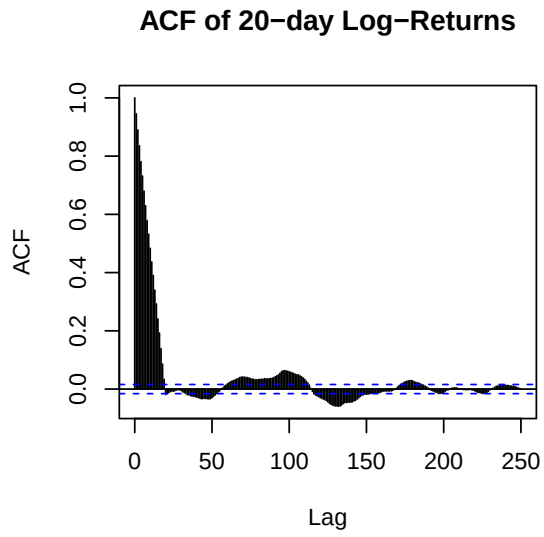


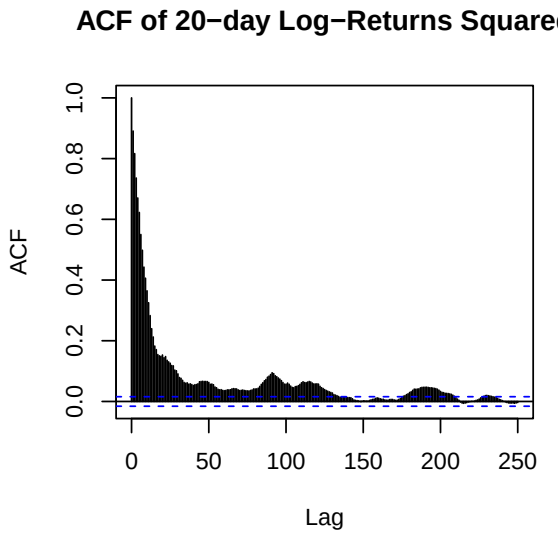

```r
# 3. plot acf of the 10-day log-return and log-return squares
par(mfrow=c(2, 2))
acf(r10.sp, main = "ACF of 10-day Log-Returns", lag.max=250)
acf(r10.sp^2, main = "ACF of 10-day Log-Returns Squared", lag.max=250)
```

```r
# 4. plot acf of the 20-day log-return and log-return squares
# par(mfrow=c(2, 1))
acf(r20.sp, main = "ACF of 20-day Log-Returns", lag.max=250)
acf(r20.sp^2, main = "ACF of 20-day Log-Returns Squared", lag.max=250)
```









3. Provide a table of the basic summary statistics for the 4 time series from part 2. 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.

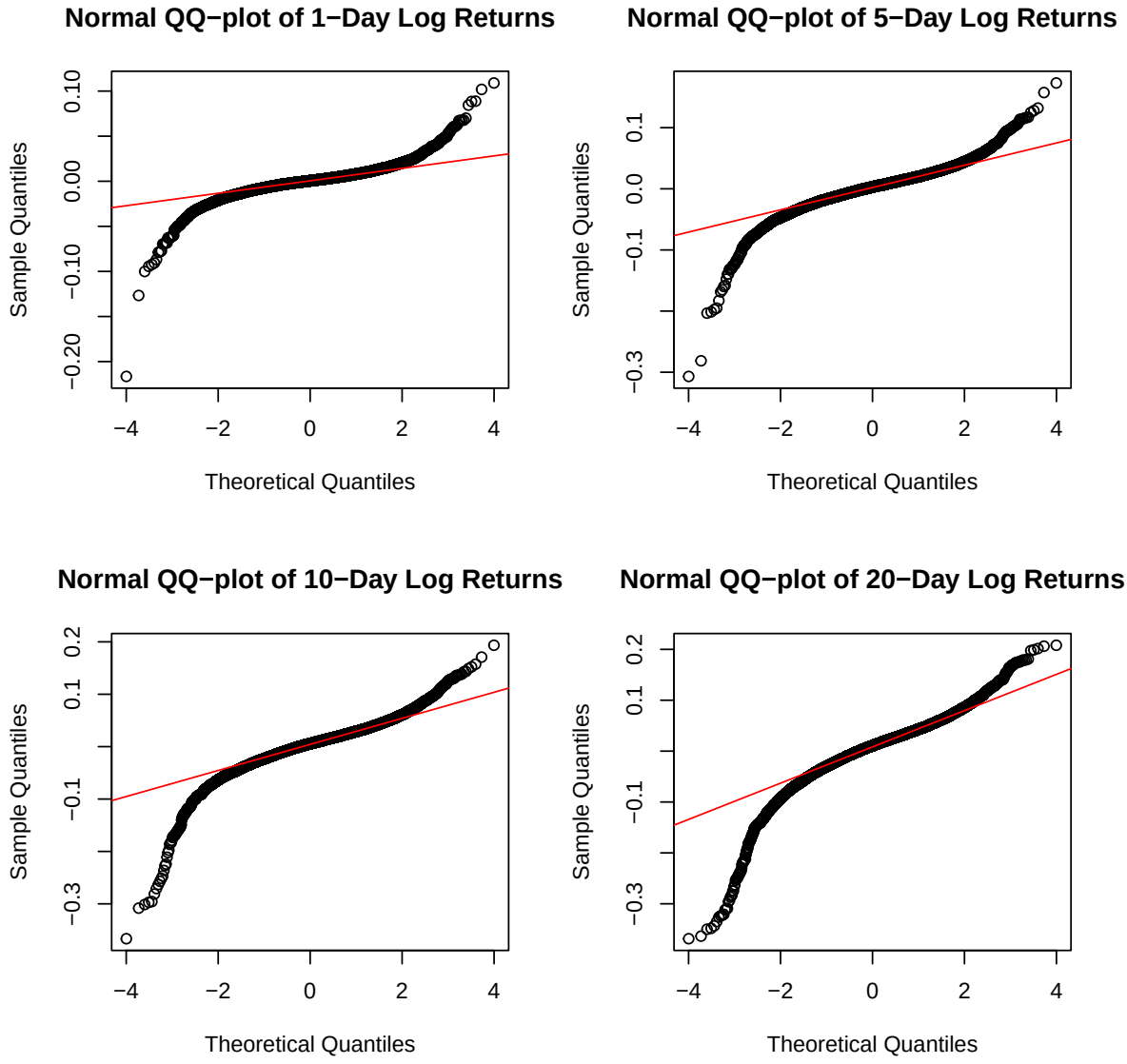
```r
returns = list("r1"=r1.sp, "r5"=r5.sp, "r10"=r10.sp, "r20"=r20.sp)
# The list is populated with random numbers, which you'd have to replace by the
# corresponding returns.
```

```
# apply summary function to each element of the list
# summary function provides a summary of data (the minimum, maximum, mean, median, 1st and 3rd quar
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	-0.2164902	-0.0041712	0.0005321	0.0003271	0.0051463	0.1089740
5-day	-0.3069056	-0.0101965	0.0030848	0.0016349	0.0143868	0.1730913
10-day	-0.3665458	-0.0125039	0.0057800	0.0032693	0.0210550	0.1934883
20-day	-0.3686597	-0.0157435	0.0108074	0.0065576	0.0323305	0.2080365

4. Using the same data set as in the previous problem, make a 2×2 array of normal quantile-quantile plots.

```
par(mfrow=c(2,2))
qqnorm(returns$r1, main="Normal QQ-plot of 1-Day Log Returns")
qqline(returns$r1, col="red")
qqnorm(returns$r5, main="Normal QQ-plot of 5-Day Log Returns")
qqline(returns$r5,col="red")
qqnorm(returns$r10, main="Normal QQ-plot of 10-Day Log Returns")
qqline(returns$r10,col="red")
qqnorm(returns$r20, main="Normal QQ-plot of 20-Day Log Returns")
qqline(returns$r20,col="red")
```

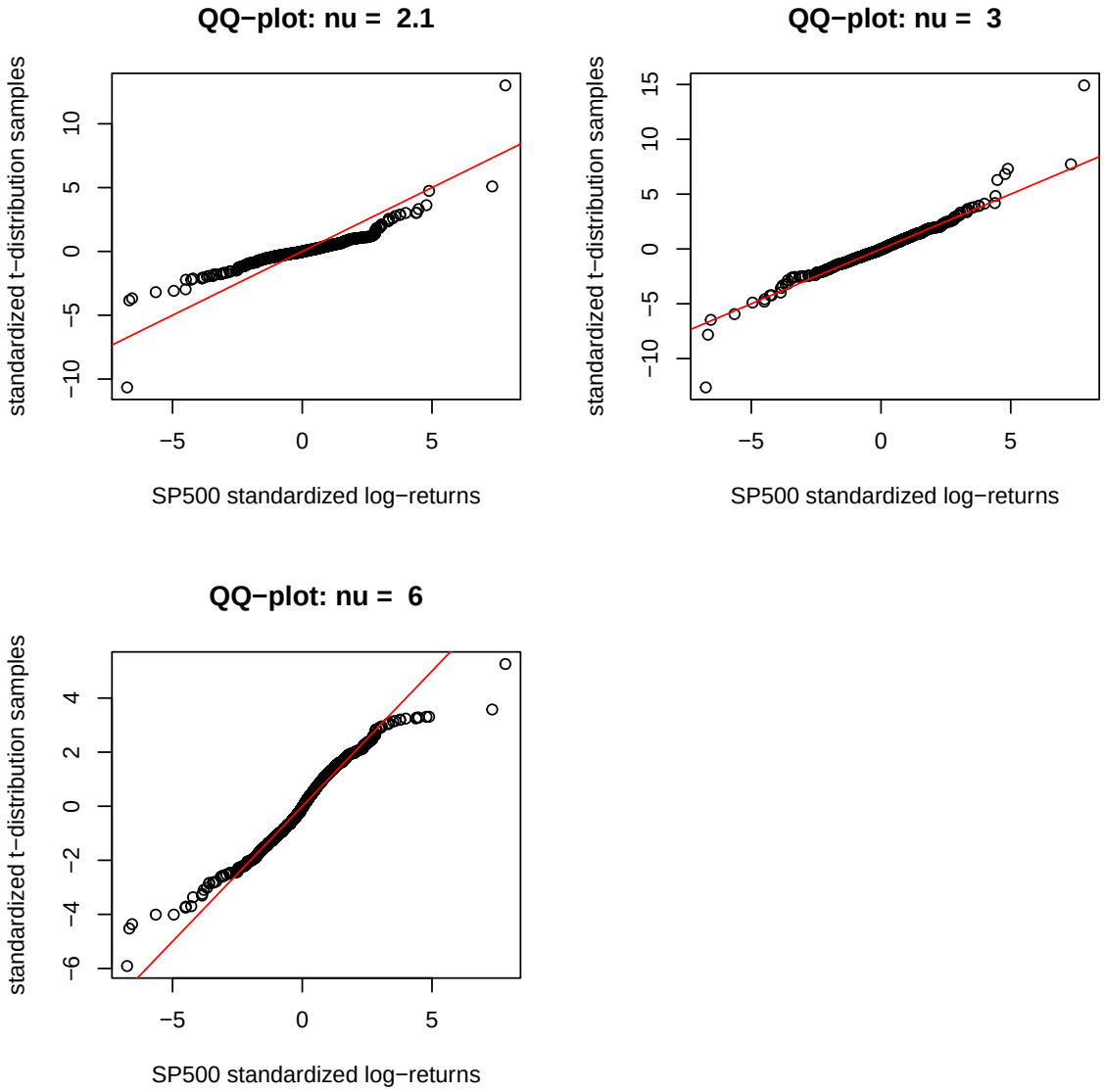


From the generated QQ-plots, it can be seen that the right tails of the returns curve up away from the normal distribution and the left tails curve down. Therefore the data distributions have both heavier right and left tails, meaning there are more extreme values.

5. 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.

```
sp500 = read.csv("sp500_full.csv",header=TRUE)
t.sp = as.Date(x=as.character(sp500$caldt),format="%Y%m%d")
idx = which(t.sp>as.Date("2000-01-01") & t.sp < as.Date("2010-01-01"))
```

```
rt = diff(log(sp500$spindx[idx]));
mu = mean(rt);
sig = sd(rt);
rt_standardized = (rt-mu)/sig;  # the standardized version of the log-returns


# The following function produces the desired QQ-plots.

qqplot_against_std <- function(standardized_log_returns,nu = 2.1){
 require(fGarch)  # load fGarch library
 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")
}

set.seed(100)
par(mfrow=c(2, 2))
qqplot_against_std(rt_standardized, 2.1)

## Loading required package:  fGarch
## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##         require("timeSeries")

qqplot_against_std(rt_standardized, 3)
qqplot_against_std(rt_standardized, 6)
```



The value $\nu = 3$ is reasonable because most of the points in the QQ-plot are on or very close to the red line, even the tail points in general follow the trend of the $y = x$ line. This means that the empirical distribution of the standardized returns are very close to the theoretical distribution t_3.