Financial Data & Returns

This is sourced from STAD70 course practice questions and sample R code taught by professor Sotos. If you have any questions/concerns/comments feel free to email me: cristal.wang111@gmail.com (mailto:cristal.wang111@gmail.com).

1.Closing and adjusted closing daily prices

use the function 'get.hist.quote()' from the tseries package.

```
## Requires libraries quantmod and tseries
## Download & install the most recent versions
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

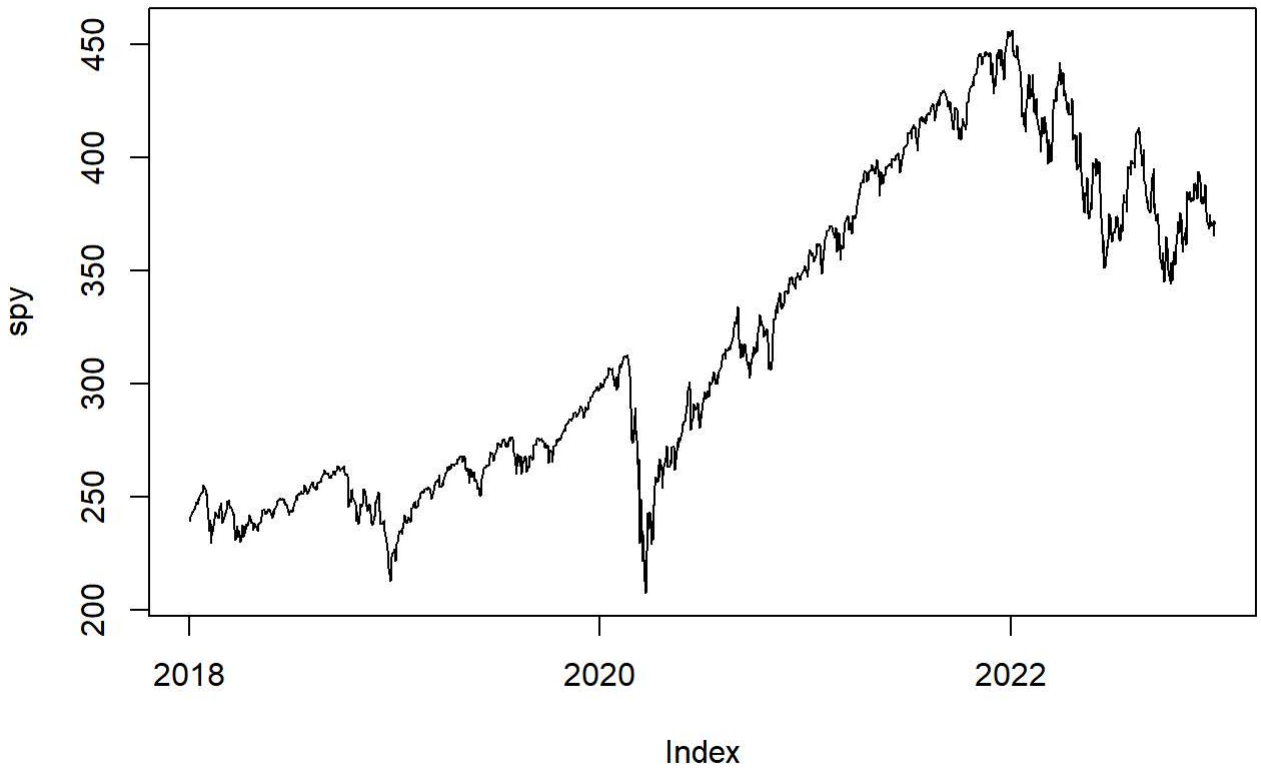
```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

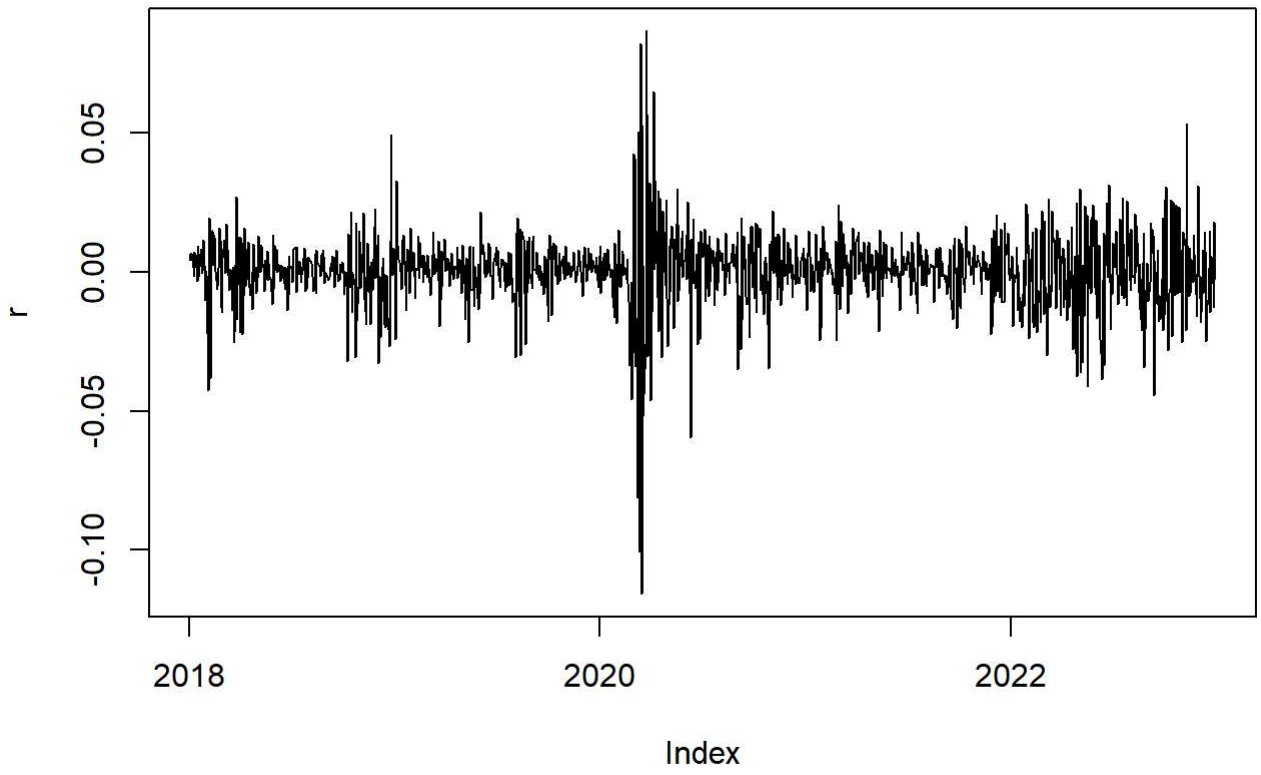
```
library(tseries)

# Download the SPDR S&P 500 ETF adjusted close prices
spy = get.hist.quote(instrument= 'SPY',
                     start = "2018-01-01",
                     end = "2022-12-31",
                     quote="AdjClose", ## quote="Close"
                     provider = "yahoo",
                     compression = "d",
                     retclass="zoo")
```

```
## time series starts 2018-01-02
## time series ends   2022-12-30
```

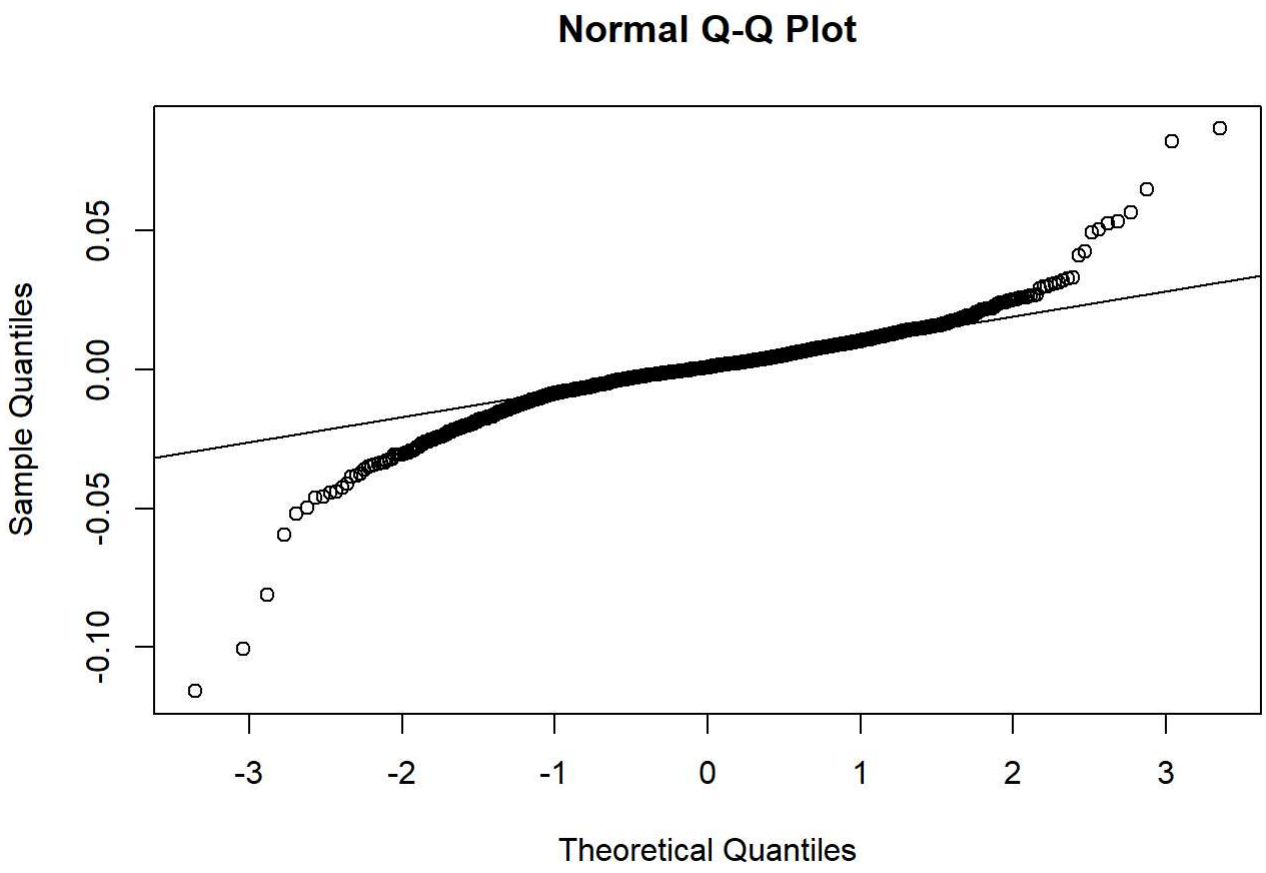
```
# Plot the adjusted prices
plot(spy)
```



```
# calculate & plot the log returns
r = diff(log(spy))
  # diff: computes the first-order difference (current value - previous value).
plot(r)
```




```
# qq plot of the log returns
qqnorm(r); qqline(r)
```



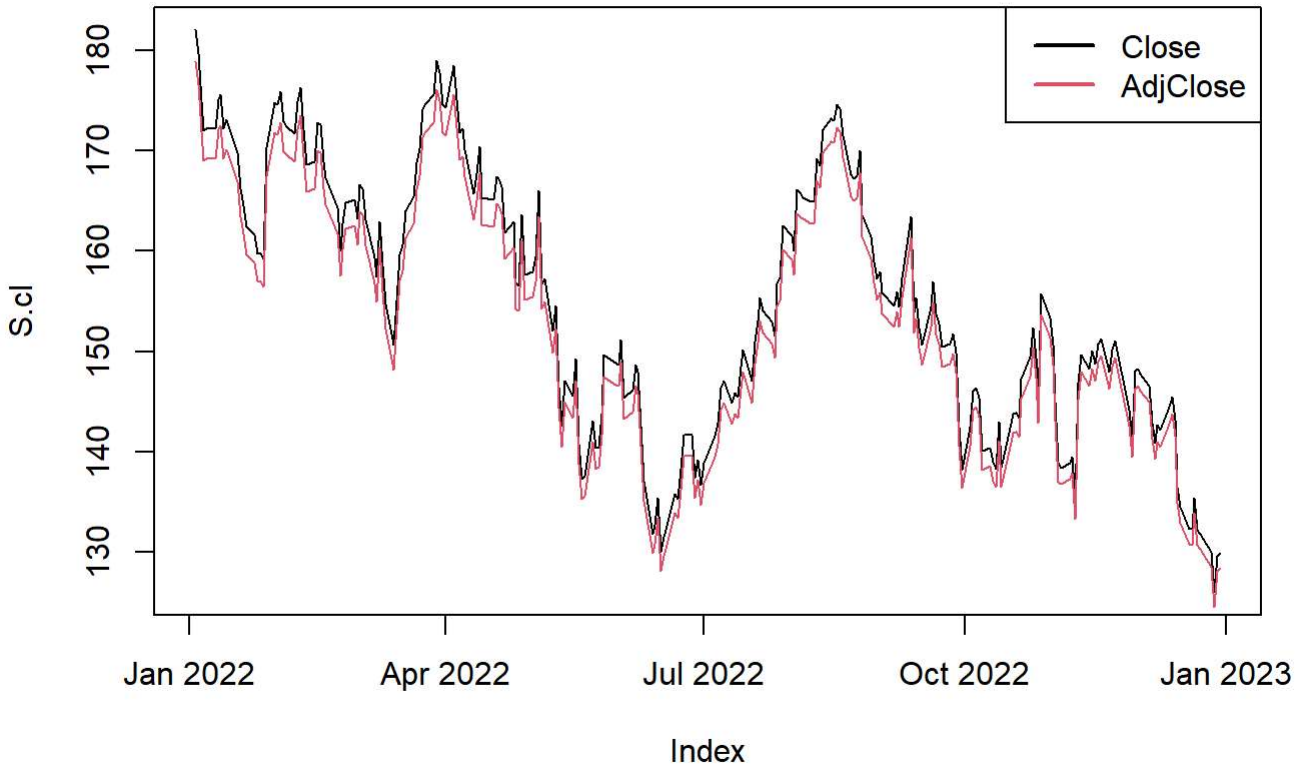
2.Return and Finance Data

Q1 (Daily Return Data)

Download the closing and adjusted closing daily prices for Apple Inc. (symbol AAPL) for the previous year (Jan 1, 2022 to Dec 31, 2022). Use the function `get.hist.quote()` from the `tseries` package

(a) Plot the closing and adjusted-closing price series on the same plot. Do you see any differences?

```
plot( S.cl )
lines( S.adj, col = 2 )
legend("topright", lwd=2, col = 1:2, c("Close", "AdjClose"))
```



The adjusted closing price is different than the closing price for days prior to Nov 4 (last event).

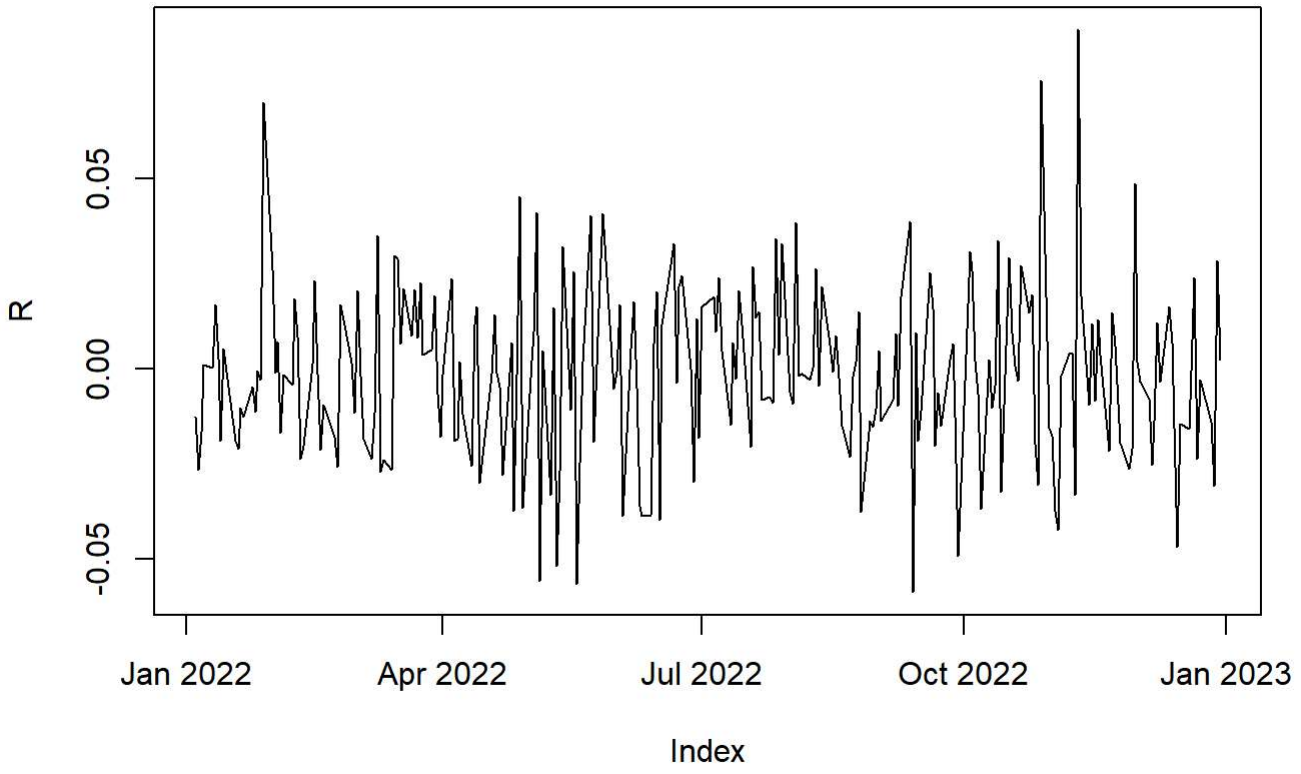
(b) net returns calculate & plot (adjusted closing price)

Calculate and plot the net returns based on the adjusted closing price.

The net return R_t is calculated as:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

```
R = diff(S.adj) / lag(S.adj,-1) #lag(x, k): shifts values of a time series by k periods.
plot(R)
```




```
## Alternatively, you can use the function quantmod::Delt()
## R = Delt( S.adj )
### Delt: calculates percentage changes (returns) of a given time series. But 1th is NA.

## Note: for log-returns you can use either of:
## r = Delt( S.adj, type = "log" )
## r = diff( log( S.adj ) )
## Also note that: R = exp( r ) - 1 <=> r = log( R + 1 )
```

(c) Dividend on days, adjust net returns

Apple paid a dividend of $0.23 on Nov 4, 2022 (such events are documented here). Calculate the net return of Nov 4, using both the adjusted closing prices (designated way), and the actual closing prices, where you manually adjust for the dividend (i.e., you add the dividend amount to the Nov 4 close price before calculating the log-return). Are the two returns equal?

The return calculated based on the AdjClose prices is:

```
R[as.Date("2022-11-04")]
```

```
##   2022-11-04
## -0.001947311
```

Note: `get.hist.quote()` creates time series that are irregularly-space (because of weekday trading days, holidays, etc) and cannot be represented by the base-R class `ts`. Therefore, the function outputs an object of class `zoo`, which is a precursor and special case of class `xts`. For more details on these classes and on working with irregularly spaced time series in R, see Manipulating Time Series Data in R with xts & zoo (https://rstudio-pubs-static.s3.amazonaws.com/288218_117e183e74964557a5da4fc5902fc671.html)

The return calculated based on the dividend correction is:

```
Div = .23
S.cl=get.hist.quote("AAPL", start='2022-01-01', end='2022-12-31', quote='Close')
```

```
## time series starts 2022-01-03
## time series ends   2022-12-30
```

```
Pt_1 = as.numeric( S.cl[as.Date("2022-11-03")] ) # convert to numeric so that time indexes
don't clash
Pt = S.cl[as.Date("2022-11-04")]
R_DivCorrection = ( ( Pt + Div) - Pt_1 ) / ( Pt_1 )

print(paste("Return Dividend Corrected:", R_DivCorrection))
```

```
## [1] "Return Dividend Corrected: -0.00194412435561071"
```

This is slightly different than the return based on the adjusted closing price. The reason for that is how Yahoo! Finance implements its dividend adjustment, which is multiplicative rather than additive, in order to make it apply cumulatively to all previous prices (hence the difference in Close and AdjClose values before Nov 4 2022).

(d) strategy - net returns (daily and cumulative)

Consider a simple momentum strategy whereby:

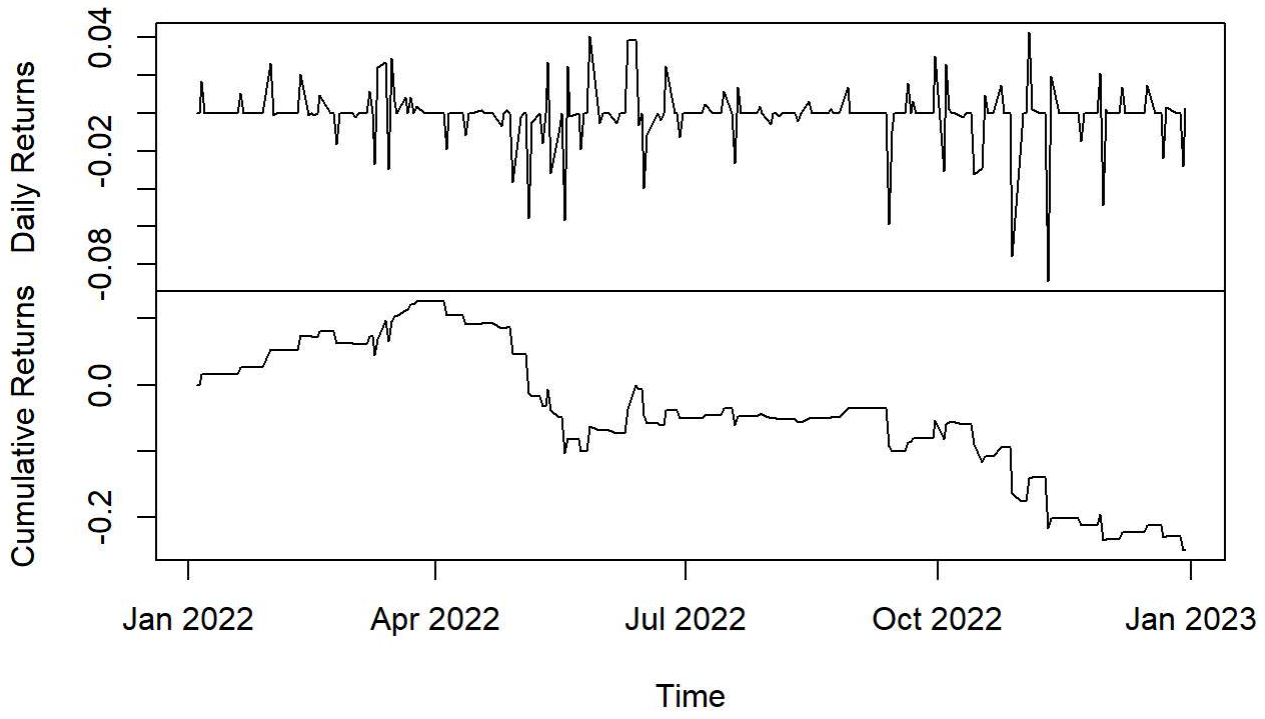
- you buy APPLE stock for 1 day, if its price increased by more than 2% on the previous day, and
- you (short) sell APPLE stock for 1 day, if its price decreased by more than 2% on the previous day.

Calculate and plot the daily and cumulative net returns of this strategy.

```
library(zoo)
S.adj=get.hist.quote("AAPL", start='2022-01-01', end='2022-12-31', quote='AdjClose')
```

```
## time series starts 2022-01-03
## time series ends   2022-12-30
```

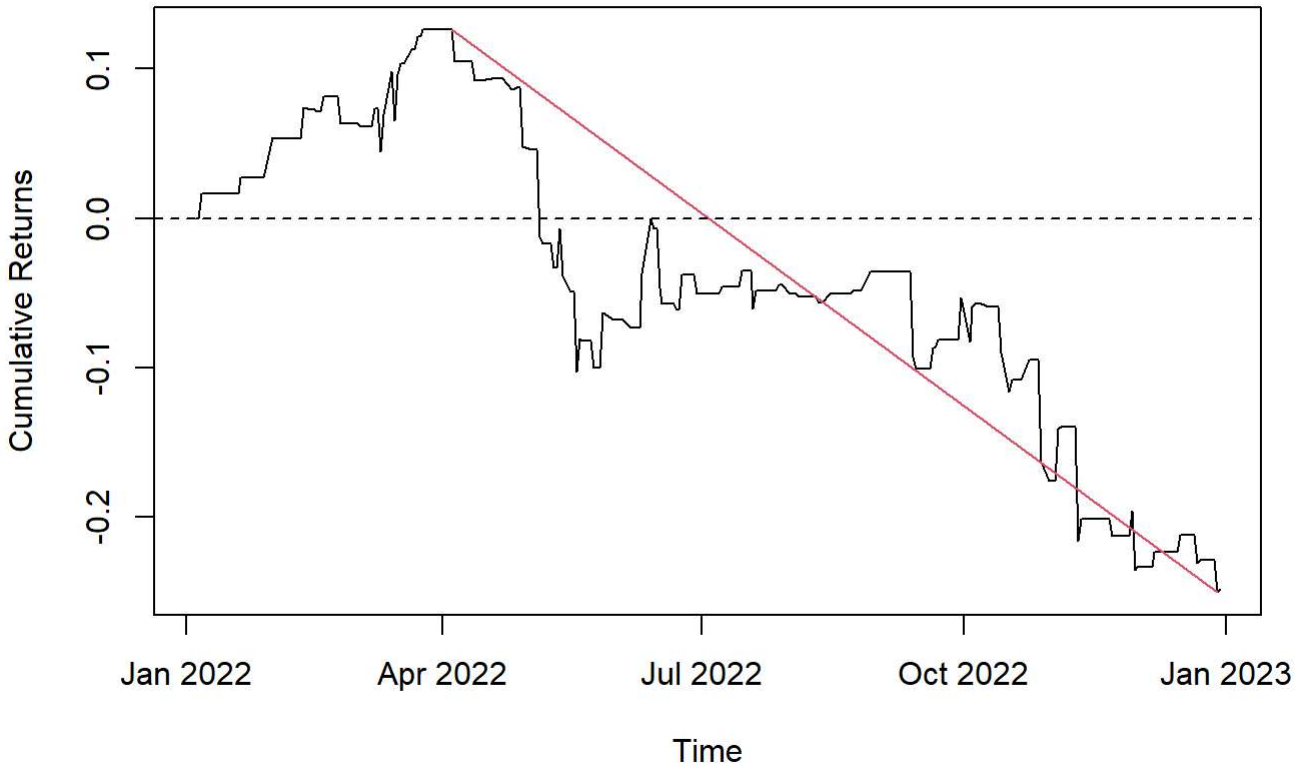
```
R = diff(S.adj) / lag(S.adj,-1)

mom.ret = zoo( rep(0,length(R) ), order.by = index(R) )
for( i in 2:length(R) ){
  if( R[i-1] < -.02 ){ mom.ret[i] = -R[i] }
  if( R[i-1] > +.02 ){ mom.ret[i] = +R[i] }
}
cum.mom.ret = cumprod( 1 + mom.ret ) - 1 # cumulative returns ( have to multiply/compound
net returns )

plot(cbind(mom.ret,cum.mom.ret), main="", ylab=c("Daily Returns","Cumulative Returns") , x
lab='Time')
```



(e) maximum drawdown - simple momentum strategy

Find the maximum drawdown of the strategy, i.e. the biggest drop in cumulative net returns from their highest point. You can use the **`maxdrawdown()`** function from the **`tseries`** package.

```
MDD = maxdrawdown(cum.mom.ret)

plot(cum.mom.ret,ylab="Cumulative Returns" , xlab='Time')
lines( cum.mom.ret[c(MDD$from[1],MDD$to[1])], col=2 ) # col=2 → red
abline(h=0,lty=2) # h=0: horizontal; lty=2: dashed line
```



Q2 theory - Model Daily Log-Return - Normal Random Walk

A stock has an expected annual log-return of 8% and an annual volatility (i.e. st. deviation of annual log-return) of 20%. You want to model the daily log-return using a Normal Random Walk model. Assume that a calendar year has 252 trading days of daily returns. What should the mean and variance of the Normal distribution of the daily log-returns be, so that your model matches the annual parameters above?

SOLUTION:

The random walk model is additive in log-returns (r), which means that over 1year = 252days we have:

$$r_{1yr} = \sum_{t=1}^{252} r_t$$

Assuming i.i.d. daily log-returns with *daily* moments (mean return and daily volatility) $\mathbb{E}[r_t] = \mu$ and $\mathbb{V}[r_t] = \sigma^2$, we have:

$$\mathbb{E}[r_{1yr}] = \sum_{t=1}^{252} \mathbb{E}[r_t]$$

$$\Rightarrow 0.08 = 252\mu \Rightarrow \mu = \frac{0.08}{252} = 0.0003174603 = .03174603\%$$

$$\mathbb{V}[r_{1yr}] = \sum_{t=1}^{252} \mathbb{V}[r_t] \quad \text{(by indep.)}$$

$$\Rightarrow 0.2^2 = 252\sigma^2 \Rightarrow \sigma = \sqrt{\frac{0.2^2}{252}} = \frac{0.2^2}{\sqrt{252}} = 0.01259882 = 1.259882\%$$

Q3 Probability - Price

Assume that the daily log returns on a stock with current price S0 = 100 are i.i.d. with a Normal distribution with mean 0.0003 and standard deviation 0.01.

We have $r_t \sim^{iid} \mathrm{N}(\mu = 0.0003, \sigma^2 = 0.01^2)$, therefore
$r_{1-n} = (\sum_{t=1}^{n} r_t) \sim \mathrm{N}(0.0003 \times 50, 0.01^2 \times 50)$, and $S_n = S_0 e^{r_{1-n}} = 100 e^{r_{1-n}}$

Note: You will have to use (**`pnorm()`** , **`pbinom()`**) to find the numerical values of these probabilities.

(a) pnorm, Calculate the probability that the price will be above $120 after 50 days.

We have

$$P(S_{50} > 120) = 1 - P(S_0 e^{r_{1-50}} \leq 120) = 1 - P(r_{1-50} \leq \ln(120/100)) =$$
$$= 1 - P(Z \leq \frac{ln(1.2) - .0003 \times 50}{0.01 \times \sqrt{50}})$$

which gives

```
1-pnorm( (log(1.2) - .0003*50)/(.01*sqrt(50))  )
```

```
## [1] 0.008983824
```

(b) pbinom,pnorm. Calculate the probability that the price will increase (i.e., have +ve return) in 30 or more of the next 50 days.

The probability the price will increase on any given day is

$$P(S_t > S_{t-1}) = P(S_t/S_{t-1} > 1) = P(\ln(S_t/S_{t-1}) > \ln(1)) =$$
$$= P(r_t > 0) = P(Z > \frac{0 - 0.0003}{0.01}) = \Phi(0.03)$$

which is

```
(p = pnorm(0.03))
```

```
## [1] 0.5119665
```

The probability that the price will increase in at least 30 out of 50 days is equal to the probability of a Binomial($n = 50, p = P(S_t > S_{t-1})$) RV X taking a value greater or equal to 30, i.e. $(P(X \geq 30) = 1 - P(X \leq 29))$.

```
1 - pbinom( 29, 50, p)
```

```
## [1] 0.1346917
```

or, equivalently,

```
pbinom( 29, 50, p, lower.tail = F)
```

```
## [1] 0.1346917
```

Q4 Limit Order Book data - 5 minute intervals

Go to https://lobsterdata.com/info/DataSamples.php (https://lobsterdata.com/info/DataSamples.php) and download the level 10 data for Amazon stock (AMZN). Use the provided sample R code to process the data; you can find a description of the resulting R data-frames (i.e. message & order book file) generated by the sample R code here.

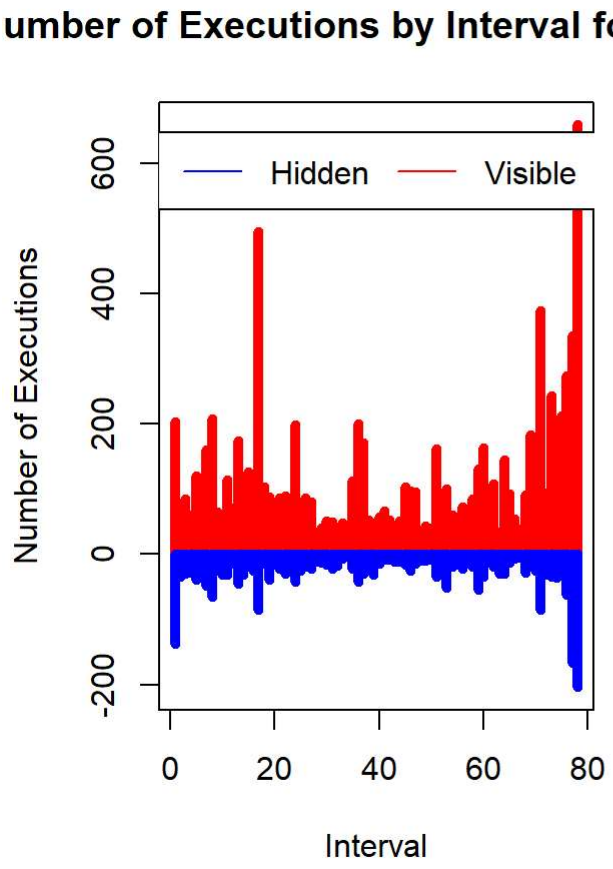
Use the data to create a daily plot of the evolution of the limit order book at 5-minute intervals (The data is too large to plot for all times, so sample the time (data row) that is closest to 5 minute increments).

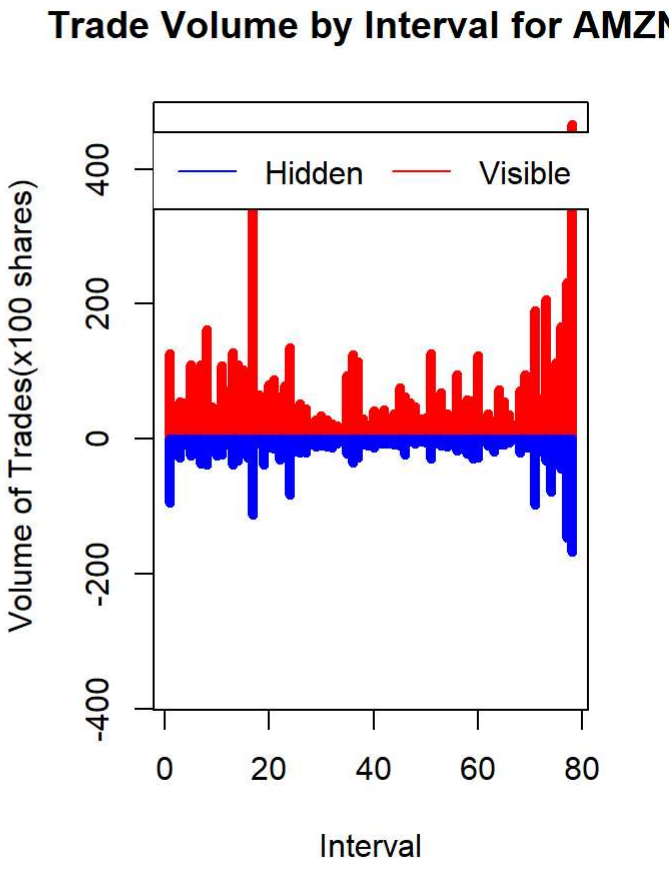
```
# load lobster demo data in R
source("LOBSTER_demo.R", chdir = TRUE)
```

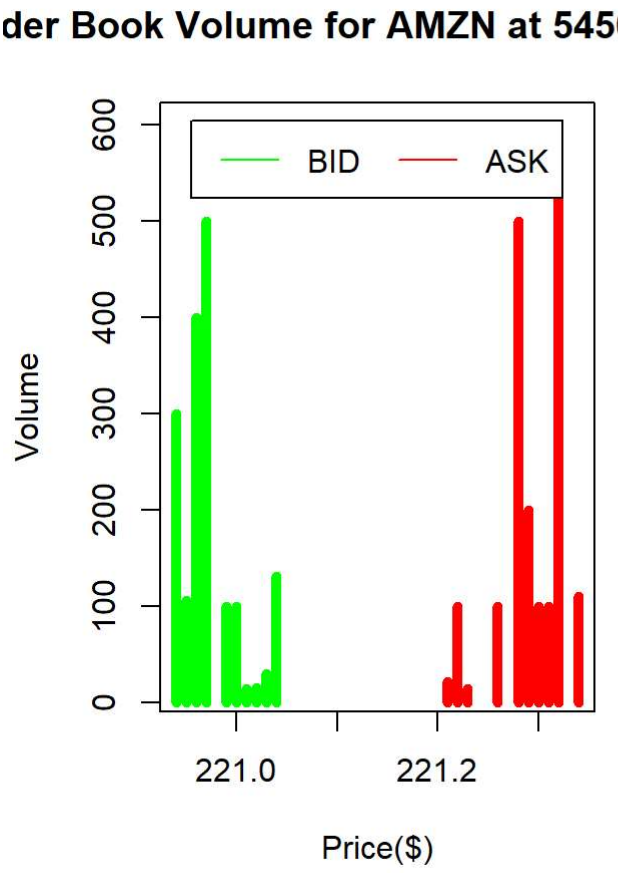
```
## [1] "No trading halts detected."
```

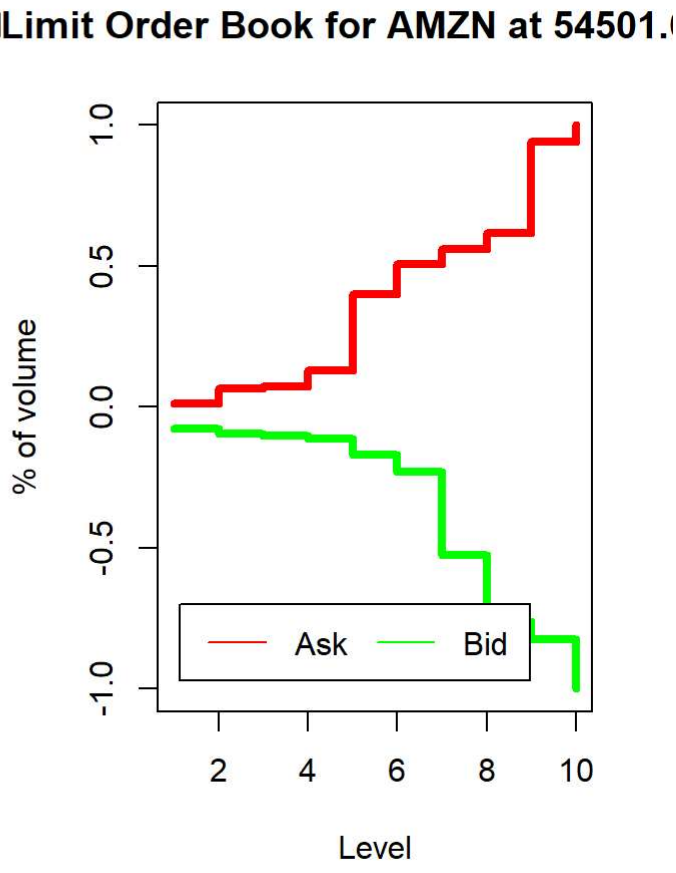
```
## Warning in sprintf("Number of Executions by Interval for %s ", ticker, cex =
## 0.8): one argument not used by format 'Number of Executions by Interval for %s
## '
```

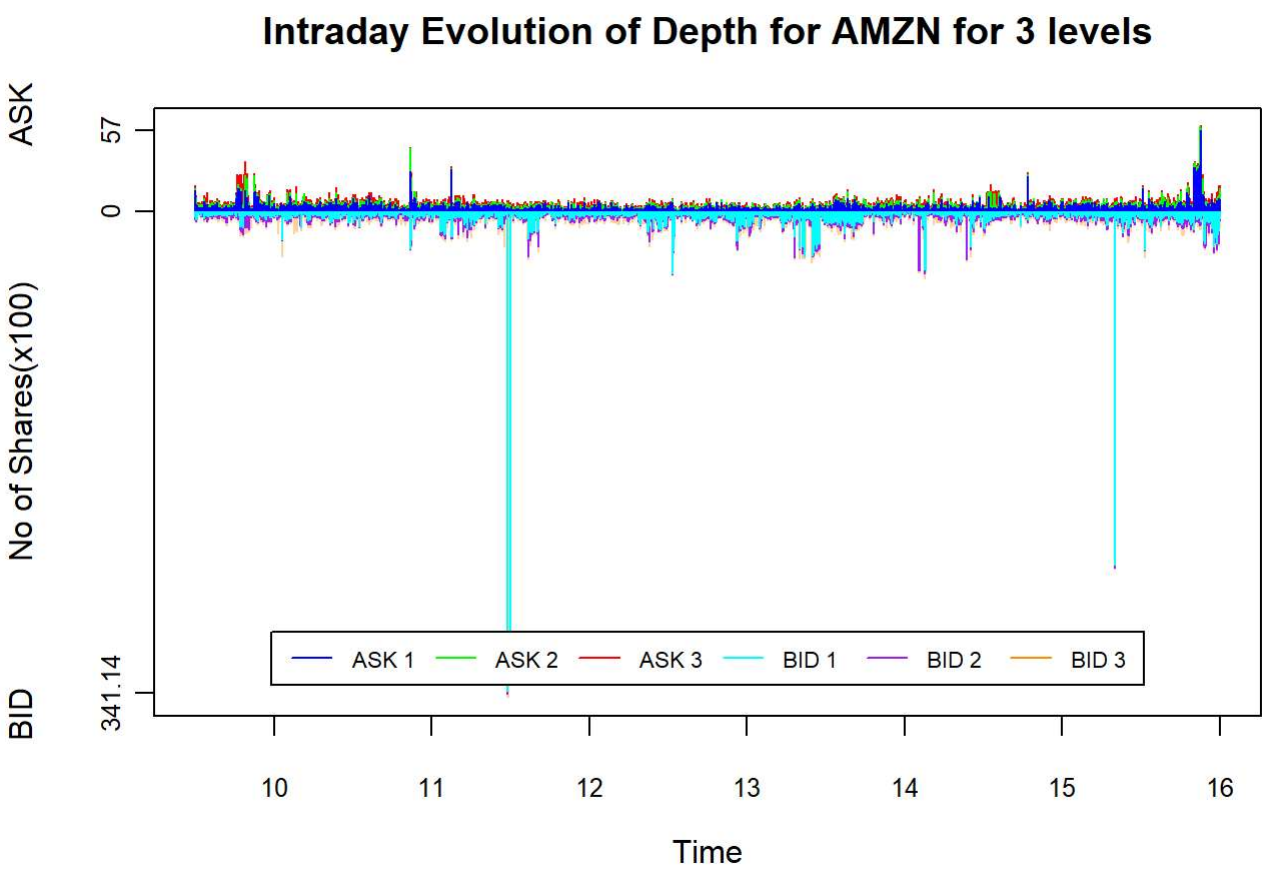
```
## Warning in sprintf("Trade Volume by Interval for %s ", ticker, cex = 0.8): one
## argument not used by format 'Trade Volume by Interval for %s '
```


umber of Executions by Interval for A
Hidden
Visible
Number of Executions
600
400
200
0
-200
0
20
40
60
80
Interval


Trade Volume by Interval for AMZN
Hidden
Visible
Volume of Trades(x100 shares)
400
200
0
-200
-400
0
20
40
60
80
Interval


der Book Volume for AMZN at 54501.(
BID
ASK
Volume
600
500
400
300
200
100
0
221.0
221.2
Price($)


Limit Order Book for AMZN at 54501.(
% of volume
1.0
0.5
0.0
-0.5
-1.0
Ask
Bid
2
4
6
8
10
Level




```
# trading happens from 9:30am to 4:00pm
# in term of seconds, this is from 34,200(=9.5*60*60=9:30am) to 57,600(=16*60*60=4pm)
# (note that the time variable in the first column of data is the time of day in seconds)
times = seq( from=34200, to=57600, by = 5*60 ) # 5 minute intervals !!!!!

timeindex=rep(0,length(times)) # index to hold row numbers of '5min' intervals
ASKP=ASKV=BIDP=BIDV=matrix(0,length(times),10) # Ask & Bid price and volume at '5min' inte
rvals

for(i in 1: length(times)){
  timeindex[i] = which.min( abs(dataM$Time-times[i]) ) #  finds index of which is the clos
est times.
  ASKP[i,] = as.numeric(dataOB[timeindex[i],(0:9)*4+1]) # top 10 levels of the order book.
  ASKV[i,] = as.numeric(dataOB[timeindex[i],(0:9)*4+2])
  BIDP[i,] = as.numeric(dataOB[timeindex[i],(0:9)*4+3])
  BIDV[i,] = as.numeric(dataOB[timeindex[i],(0:9)*4+4])
}
```

```
ylim=range( cbind(ASKP,BIDP) ) + c(-1,+1)*10000
xlim=range( times )

MIDP= (ASKP[,1] + BIDP[,1])/2 # midpoint of best bid & ask

plot(-1,-1, xlim = xlim, ylim = ylim, xlab="time of day (sec)", ylab='Price ($1/10,000)')
abline( h=seq(2200000, 2300000, by=10000), lty=3, col='lightgrey') #lty=3: Dashline
# plots of 10 best bid/ask prices at 5min intervals
# volume at each price level is represented by size of points
for(i in 1:length(times)){
  xloc = times[i] + 10*(1:10)
  points( xloc, ASKP[i,], col=2, pch='.', cex=(ASKV[i,])^(1/3) )
  # col=2: red; cex:point size prop to cube root of ask volume
  points( xloc, BIDP[i,], col=3, pch='.', cex=(BIDV[i,])^(1/3) )
  # col=3: green; cex:point size prop to cube root of bid volume
}
lines(times, MIDP)
abline(v=(10:16)*60*60, lty=3, col="lightgrey") # Adds vertical dashed lines between 10AM-4PM (in seconds).
legend('topright', title="Volume", c("100","250",'1000'), pch='.', pt.cex=c(100,250,1000)^(1/3))
```