Midterm

Akshay Pradeep Patade

2023-03-26

Clearing the environment and Setting the directory in R

```
rm(list=ls())
getwd()
```

```
## [1] "/Users/akshaypatade/Desktop/FE515"
```

```
setwd("/Users/akshaypatade/Desktop/FE515/")
```

#1.1Download daily equity data of JPM and WFC (2012-01-01 to 2023-01- 01).

```
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
```

```
##
## ###################################### WARNING ######################################
## # We noticed you have dplyr installed. The dplyr lag() function breaks how    #
## # base R's lag() function is supposed to work, which breaks lag(my_xts).      #
## #                                                                              #
## # If you call library(dplyr) later in this session, then calls to lag(my_xts) #
## # that you enter or source() into this session won't work correctly.           #
## #                                                                              #
## # All package code is unaffected because it is protected by the R namespace   #
## # mechanism.                                                                   #
## #                                                                              #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
```

```
## # You can use stats::lag() to make sure you're not using dplyr::lag(), or you #
## # can add conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop   #
## # dplyr from breaking base R's lag() function.                                 #
## ################################## WARNING ##################################
```

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

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

```
getSymbols("JPM",
           from = "2012/01/01",
           to = "2023/01/01",
           periodicity = "daily")
```

```
## [1] "JPM"
```

```
head(JPM)
```

```
##            JPM.Open JPM.High JPM.Low JPM.Close JPM.Volume JPM.Adjusted
## 2012-01-03    34.06    35.19   34.01     34.98   44102800     25.54859
## 2012-01-04    34.44    35.15   34.33     34.95   36571200     25.71043
## 2012-01-05    34.71    35.92   34.40     35.68   38381400     26.24745
## 2012-01-06    35.69    35.77   35.14     35.36   33160600     26.01204
## 2012-01-09    35.44    35.68   34.99     35.30   23001800     25.96790
## 2012-01-10    36.07    36.35   35.76     36.05   35972800     26.51964
```

```
getSymbols("WFC",
           from = "2012/01/01",
           to = "2023/01/01",
           periodicity = "daily")
```

```
## [1] "WFC"
```

```
head(WFC)
```

```
##            WFC.Open WFC.High WFC.Low WFC.Close WFC.Volume WFC.Adjusted
## 2012-01-03    27.94    28.52   27.94     28.43   40071200     20.65028
## 2012-01-04    28.34    28.69   28.04     28.56   27519200     20.74472
## 2012-01-05    28.50    29.58   28.25     29.02   48435100     21.07884
## 2012-01-06    28.84    29.08   28.46     28.94   32303500     21.02073
## 2012-01-09    29.15    29.38   29.00     29.30   25720100     21.28221
## 2012-01-10    29.74    29.80   29.18     29.41   29860100     21.36211
```

#1.2 Calculate both the daily log return and weekly log return for each stock.

```
JPM_daily_log_returns <- dailyReturn(JPM[, 4], subset = NULL, type = 'log', leading = TRUE)
head(JPM_daily_log_returns)
```

```
##            daily.returns
## 2012-01-03  0.0000000000
## 2012-01-04 -0.0008579723
## 2012-01-05  0.0206718104
## 2012-01-06 -0.0090090417
## 2012-01-09 -0.0016983304
## 2012-01-10  0.0210239004
```

```
WFC_daily_log_returns <- dailyReturn(WFC[, 4], subset = NULL, type = 'log', leading = TRUE)
head(WFC_daily_log_returns)
```

```
##            daily.returns
## 2012-01-03   0.000000000
## 2012-01-04   0.004562177
## 2012-01-05   0.015978145
## 2012-01-06  -0.002760492
## 2012-01-09   0.012362726
## 2012-01-10   0.003747271
```

```
JPM_weekly_log_returns <- weeklyReturn(JPM[, 4], subset = NULL, type = 'log', leading = TRUE)
head(JPM_weekly_log_returns)
```

```
##            weekly.returns
## 2012-01-06    0.010804796
## 2012-01-13    0.015712922
## 2012-01-20    0.039306452
## 2012-01-27   -0.004023125
## 2012-02-03    0.028350025
## 2012-02-10   -0.017657541
```

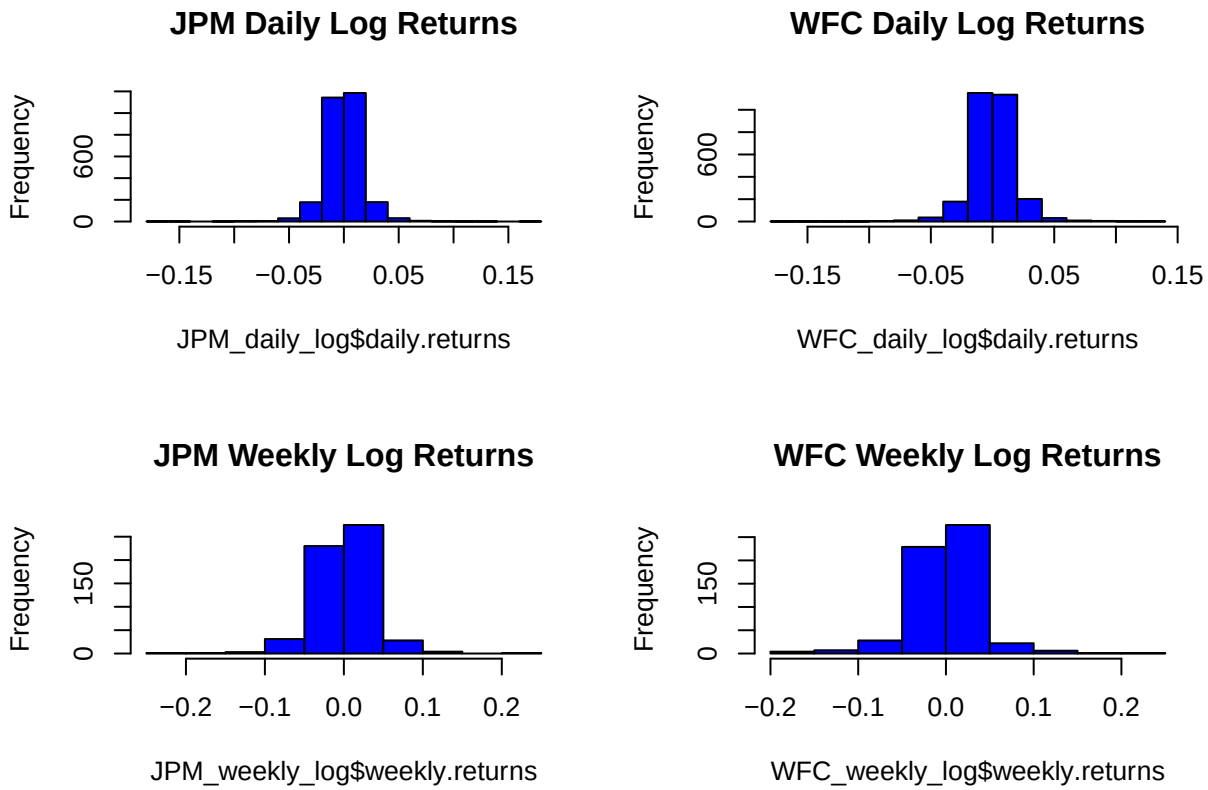
```
WFC_weekly_log_returns <- weeklyReturn(WFC[, 4], subset = NULL, type = 'log', leading = TRUE)
head(WFC_weekly_log_returns)
```

```
##            weekly.returns
## 2012-01-06     0.01777983
## 2012-01-13     0.02288742
## 2012-01-20     0.03092516
## 2012-01-27    -0.03126297
## 2012-02-03     0.03420553
## 2012-02-10    -0.01215318
```

#1.3 Visualize the distribution of these log returns using hist() function. Use par() function to put the four histogram together into one single graph, where each histogram is an individual subplot.

```
JPM_daily_log_returns <- dailyReturn(JPM[, 4], subset = NULL, type = 'log', leading = TRUE)
JPM_daily_log <- fortify.zoo( JPM_daily_log_returns )
names( JPM_daily_log )[1] <- "Dates"

WFC_daily_log_returns <- dailyReturn(WFC[, 4], subset = NULL, type = 'log', leading = TRUE)
WFC_daily_log <- fortify.zoo( WFC_daily_log_returns )
names( WFC_daily_log )[1] <- "Dates"
```

```
JPM_weekly_log_returns <- weeklyReturn(JPM[, 4], subset = NULL, type = 'log', leading = TRUE)
JPM_weekly_log <- fortify.zoo( JPM_weekly_log_returns )
names( JPM_weekly_log )[1] <- "Dates"

WFC_weekly_log_returns <- weeklyReturn(WFC[, 4], subset = NULL, type = 'log', leading = TRUE)
WFC_weekly_log <- fortify.zoo( WFC_weekly_log_returns )
names( WFC_weekly_log)[1] <- "Dates"

par(mfrow =c(2,2))
hist(JPM_daily_log$daily.returns,col = "blue",main = "JPM Daily Log Returns")
hist(WFC_daily_log$daily.returns,col = "blue", main = "WFC Daily Log Returns")
hist(JPM_weekly_log$weekly.returns,col = "blue", main = "JPM Weekly Log Returns")
hist(WFC_weekly_log$weekly.returns,col = "blue", main = "WFC Weekly Log Returns")
```



#1.4 Calculate the first four moments, i.e. mean, variance, skewness and kur- tosis, for each stock. Store the calculate result in a data frame and report the result in a table.

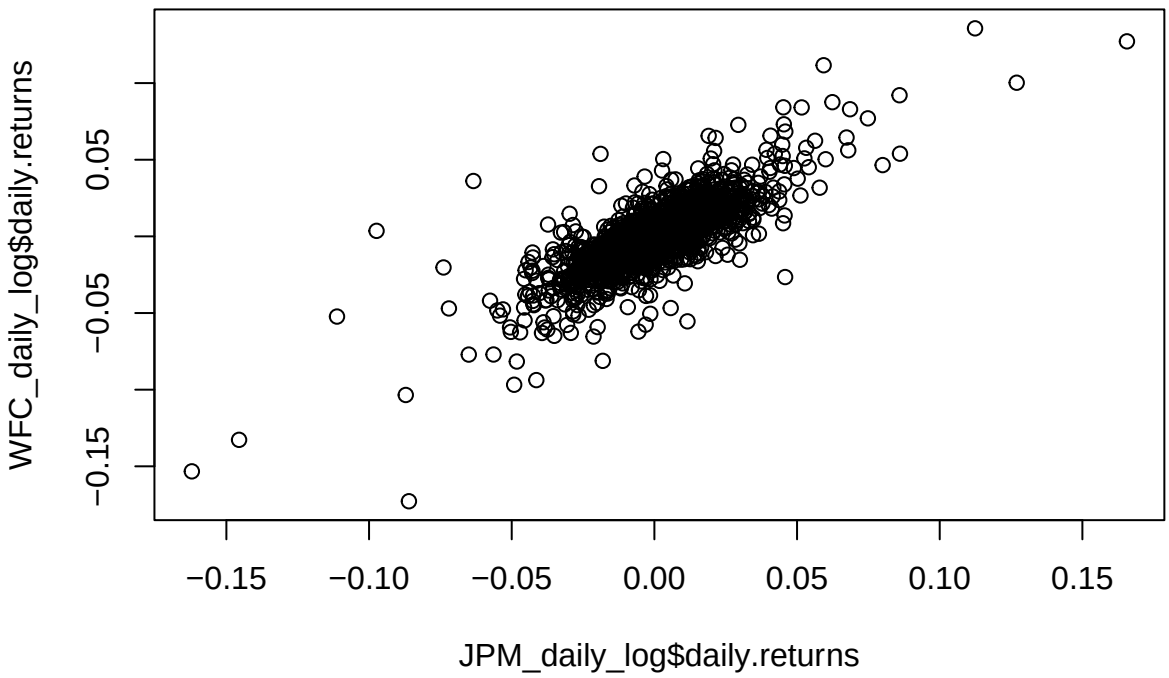
```
library(moments)
JPM.daily.mean = mean(JPM_daily_log$daily.returns)
JPM.daily.var = var(JPM_daily_log$daily.returns)
JPM.daily.skewness = skewness(JPM_daily_log$daily.returns)
JPM.daily.kurtosis = kurtosis(JPM_daily_log$daily.returns)

JPM.weekly.mean = mean(JPM_weekly_log_returns)
JPM.weekly.var = var(JPM_weekly_log_returns)
```

```
JPM.weekly.skewness = skewness(JPM_weekly_log_returns)
JPM.weekly.kurtosis = kurtosis(JPM_weekly_log_returns)

WFC.daily.mean = mean(WFC_daily_log_returns)
WFC.daily.var = var(WFC_daily_log_returns)
WFC.daily.skewness = skewness(WFC_daily_log_returns)
WFC.daily.kurtosis = kurtosis(WFC_daily_log_returns)

WFC.weekly.mean = mean(WFC_weekly_log_returns)
WFC.weekly.var = var(WFC_weekly_log_returns)
WFC.weekly.skewness = skewness(WFC_weekly_log_returns)
WFC.weekly.kurtosis = kurtosis(WFC_weekly_log_returns)

Data<-c("JPM.daily.ret","JPM.weekly.ret","WFC.daily.ret","WFC.weekly.ret")

Mean<-c(JPM.daily.mean,JPM.weekly.mean,WFC.daily.mean,WFC.weekly.mean)
Variance<-c(JPM.daily.var,JPM.weekly.var,WFC.daily.var,WFC.weekly.var)
Skewness<- c(JPM.daily.skewness,JPM.weekly.skewness,WFC.daily.skewness,WFC.weekly.skewness)
Kurtosis<- c(JPM.daily.kurtosis,JPM.weekly.kurtosis,WFC.daily.kurtosis,WFC.weekly.kurtosis)
results <- data.frame(Data,Mean,Variance,Skewness,Kurtosis)
print(results)
```

```
##             Data         Mean     Variance   Skewness  Kurtosis
## 1  JPM.daily.ret 0.0004854803 0.0002882608 -0.1024082 15.671083
## 2 JPM.weekly.ret 0.0023411313 0.0013087228 -0.3389608  7.853083
## 3  WFC.daily.ret 0.0001348177 0.0003333788 -0.3384058 14.118965
## 4 WFC.weekly.ret 0.0006501314 0.0015942354 -0.0983638  8.276138
```

#1.5 Draw a scatter plot of JPM daily return against WFC daily return. (i.e. WFC return on x-axis and JPM return on y-axis)

```
plot(JPM_daily_log$daily.returns, WFC_daily_log$daily.returns)
```



#1.6 Build a simple linear regression model using the WFC daily return as ex- planatory variable and the JPM daily return as response variable. Report the fitted model using summary( ) function.

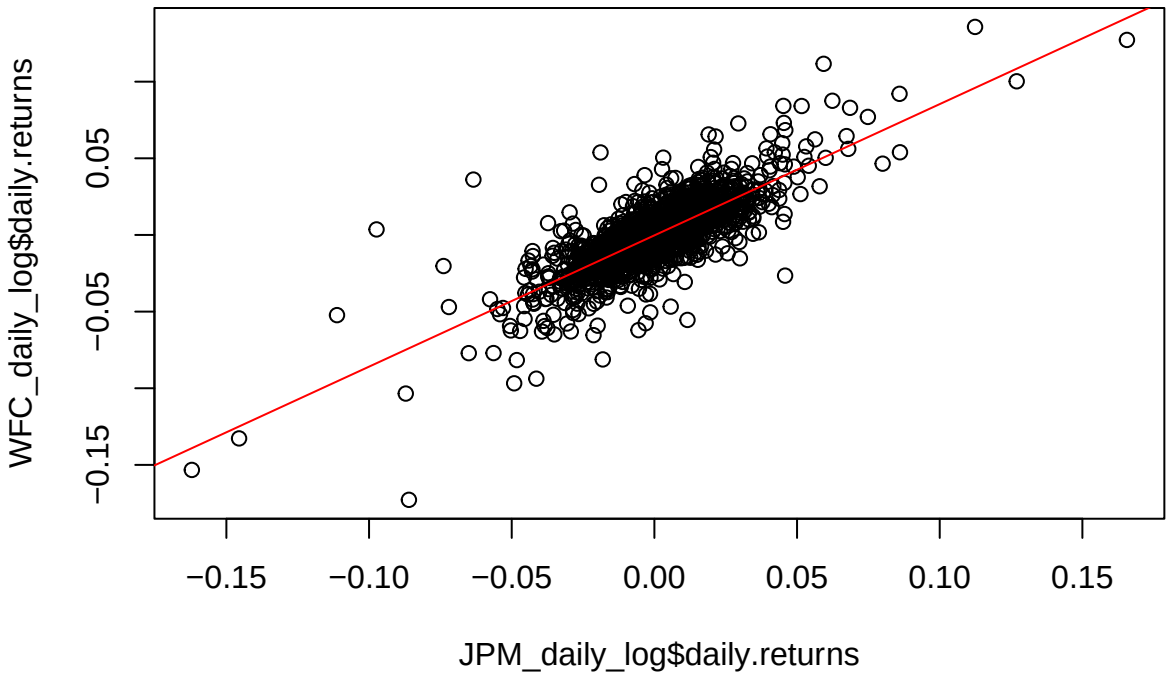
```
lm.model <- lm(WFC_daily_log$daily.returns ~ JPM_daily_log$daily.returns)
summary(lm.model)
```

```
##
## Call:
## lm(formula = WFC_daily_log$daily.returns ~ JPM_daily_log$daily.returns)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.098829 -0.005278 -0.000072  0.005226  0.090727
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -0.0002809  0.0002101  -1.337    0.181
## JPM_daily_log$daily.returns  0.8563482  0.0123691  69.233   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01105 on 2766 degrees of freedom
## Multiple R-squared:  0.6341, Adjusted R-squared:  0.634
## F-statistic:  4793 on 1 and 2766 DF,  p-value: < 2.2e-16
```

#1.7 Draw a regression line on the scatter plot using the fitted model above. Make sure use a different color

to draw the regression line.

```
plot(JPM_daily_log$daily.returns, WFC_daily_log$daily.returns)
abline(lm.model, col = "red")
```



#Generate a random data of x

```
x <- rnorm(100000, mean = 0, sd = 5)
```

#2.1.Without using packages, create a function of 2 variables “x” and “ad- justed” in r that calculates the sample skewness of “x” using the formula of skewness When ”adjusted” = TRUE, it returns the adjusted skewness of “x”, and FALSE returns the unadjusted one.

```
skewness_func <- function(x, adjusted = FALSE) {
  n <- length(x)
  mean_x <- mean(x)
  sd_x <- sd(x)
  skewness_raw <- sum((x - mean_x)^3) / n

  if (adjusted) {
    skewness_adj <- sqrt(n * (n - 1)) / (n - 2) * skewness_raw / sd_x^3
    return(skewness_adj)
  } else {
    skewness_unadj <- skewness_raw / sd_x^3
    return(skewness_unadj)
```

```
  }
}
```

#2.2 Without using packages, create a function of 2 variables "x" and "adjusted" that calculates the sample kurtosis of "x" using the formulas on Lecture 6 page 20 and page 23. When "adjusted" = TRUE, it returns the adjusted kurtosis of "x", and FALSE returns the unadjusted one.

```
kurtosis_func <- function(x,adjusted) { n <- length(x)
  kurtosis_data <- mean((x-mean(x))^4) /  (mean((x - mean(x))^2)) ^ (4 / 2)
  if(adjusted=="TRUE"){
    n <- length(x)
    temp = kurtosis_data * (n + 1)
    temp1 = temp - 3 * (n - 1)
    temp2 =  (n - 1) / ( (n - 2) * (n - 3))
    temp3 = temp2 * temp1
    kurtosis_data <- temp3 + 3
    kurtosis_data
    }

  return(kurtosis_data)
}
```

#2.3 Download historical price for ticker "SPY" for the whole 2012 and 2013 years with quantmod package, use its adjusted close price to calculate daily log return (Note the adjusted close price is different from the "adjusted" for sample moments).

```
getSymbols("SPY", from = "2012/01/01",to = "2013/12/31", periodicity = "daily")
```

```
## [1] "SPY"
```

```
head(SPY)
```

```
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted
## 2012-01-03   127.76   128.38  127.43    127.50  193697900     103.2023
## 2012-01-04   127.20   127.81  126.71    127.70  127186500     103.3642
## 2012-01-05   127.01   128.23  126.43    128.04  173895000     103.6394
## 2012-01-06   128.20   128.22  127.29    127.71  148050000     103.3723
## 2012-01-09   128.00   128.18  127.41    128.02   99530200     103.6232
## 2012-01-10   129.39   129.65  128.95    129.13  115282000     104.5217
```

```
SPY_daily_log <- dailyReturn(SPY[, 6], subset = NULL, type = 'log', leading = TRUE)
SPY_daily_log <- fortify.zoo( SPY_daily_log )
names( SPY_daily_log )[1] <- "Dates"
head(SPY_daily_log)
```

```
##        Dates daily.returns
## 1 2012-01-03   0.000000000
## 2 2012-01-04   0.001567050
## 3 2012-01-05   0.002658970
## 4 2012-01-06  -0.002580580
## 5 2012-01-09   0.002424575
## 6 2012-01-10   0.008633230
```

#2.4 Calculate the adjusted and unadjusted skewness for the daily log return in 2.3 using the function you defined. (both numbers should be close to -0.15)

```
skewness_func(SPY_daily_log$daily.returns, TRUE)
```

```
## [1] -0.1563434
```

```
skewness_func(SPY_daily_log$daily.returns, FALSE)
```

```
## [1] -0.1558749
```

#2.5 Calculate the adjusted and unadjusted kurtosis for the daily log return in 2.3 using the function you defined. (both numbers should be close to 4.1)

```
kurtosis_func(SPY_daily_log$daily.returns,TRUE)
```

```
## [1] 4.138473
```

```
kurtosis_func(SPY_daily_log$daily.returns,FALSE)
```

```
## [1] 4.11519
```