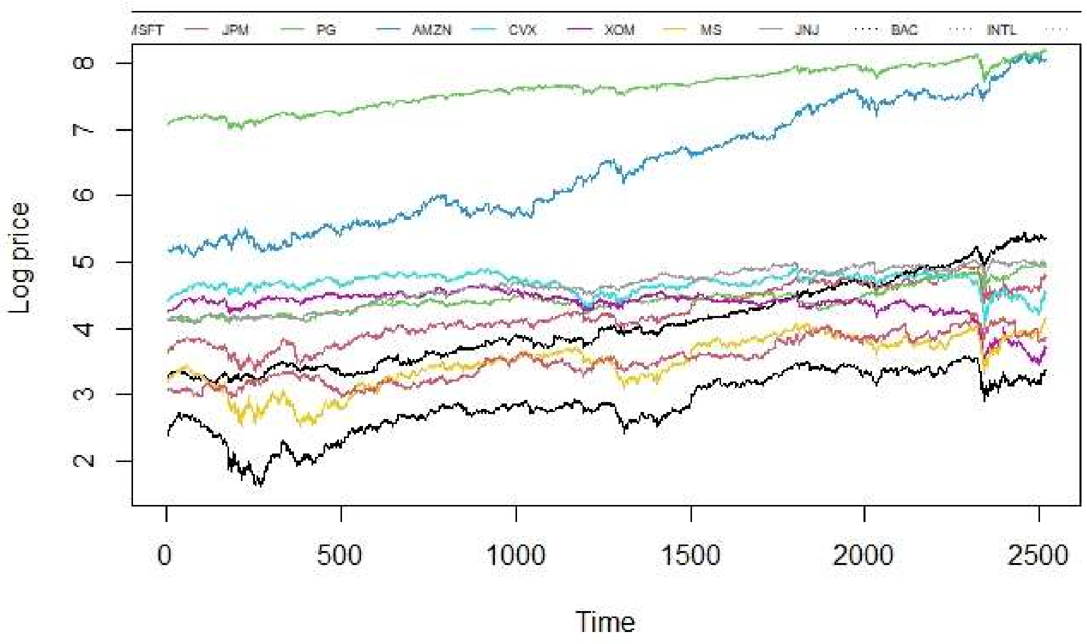
Question 1

To begin with, I have chosen Microsoft, JP Morgan, Procter & Gamble, Amazon, Chevron Corporation, Exxon Mobil, Morgan Stanley, Johnson & Johnson, Bank of America and Intel as my ten stocks from the top 50 S&P500 constituents. The data used range from the 27th of November 2010 up to the 27th of November 2020.

After downloading the data and checking that they do not contain any missing values, we proceed and plot the log-prices of the data.



From this plot we can observe that the trends for all stocks are very similar. The log prices for all stocks except Amazon seem to be slightly increasing over the ten-year period without any major changes. However, something to notice is that Amazon's log price (dark blue line) is growing at a much faster rate compared to all other stocks and in fact reaches the S&P500 log price towards the end of the time series. Also, notice that all stocks' log prices fall very sharply in the March 2020 Covid-19 pandemic.

Question 2

In this question, I present the exponential smoothing model for each time series using the individual values of lambda estimated by MLE.

The model is $\sigma_t^2 = (1-\lambda)x_{t-1}^2 + \lambda\sigma_{t-1}^2$, where x_{t-1}^2 and σ_{t-1}^2 are the previous day squared returns and conditional variance respectively, and $0 < \lambda < 1$ is the parameter to be estimated for each time series.

Microsoft: $\sigma_t^2 = 0.0308338x_{t-1}^2 + 0.9691662\sigma_{t-1}^2$

JP Morgan: $\sigma_t^2 = 0.04867x_{t-1}^2 + 0.95133\sigma_{t-1}^2$

Procter & Gamble: $\sigma_t^2 = 0.0456115x_{t-1}^2 + 0.9543885\sigma_{t-1}^2$

Amazon: $\sigma_t^2 = 0.0021582x_{t-1}^2 + 0.9978418\sigma_{t-1}^2$

Chevron: $\sigma_t^2 = 0.0644927x_{t-1}^2 + 0.9355073\sigma_{t-1}^2$

Exxon Mobil: $\sigma_t^2 = 0.0632945x_{t-1}^2 + 0.9367055\sigma_{t-1}^2$

Morgan Stanley: $\sigma_t^2 = 0.0545868x_{t-1}^2 + 0.9454132\sigma_{t-1}^2$

Johnson & Johnson: $\sigma_t^2 = 0.0791976x_{t-1}^2 + 0.9208024\sigma_{t-1}^2$

Bank of America: $\sigma_t^2 = 0.0697323x_{t-1}^2 + 0.9302677\sigma_{t-1}^2$

Intel: $\sigma_t^2 = 0.018318x_{t-1}^2 + 0.981682\sigma_{t-1}^2$

S&P500: $\sigma_t^2 = 0.0874234x_{t-1}^2 + 0.9125766\sigma_{t-1}^2$

Question 3

Prediction Algorithm & Strategy

I. Fix D, the window size to be used. Start from a certain t = t_0 (with $t_0 \geq D$).

We can write the linear model we will be using as

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$$

where $\mathbf{Y} = (Y_{t-D+1} , \dots , Y_t)^T$ is the vector of S&P500 normalized returns, $\mathbf{X} = (\mathbf{1}_\mathrm{D} , \boldsymbol{X}^1, \dots , \mathbf{X}^{q+10})^T$ with $\mathbf{X}^i = \left(X^i{}_{t-D} , \dots , X^i{}_{t-1} \right)^T$ being the vector of the i-th stock's normalized returns, $\boldsymbol{\beta} = \left(\beta_0 , \beta_1 , \dots, \beta_{q+10}\right)^T$ being the regression coefficients and $\boldsymbol{\varepsilon} = (\varepsilon_{t-D+1} , \dots , \varepsilon_t)^T$ being the vector of error terms. Note that q represents the number of S&P500 lags used for the next day prediction.

Estimate **β** by the least squares estimator:

$$\widehat{\boldsymbol{\beta}_{OLS}} = \widehat{\boldsymbol{\beta}}_\mathrm{D}(\mathrm{t}) = (\boldsymbol{X^TX})^{-1}\boldsymbol{X^TY}$$

II. The predicted return at time t+1 is:

$$\widehat{Y_{t+1}} = \widehat{\boldsymbol{\beta}}^T\boldsymbol{X_t}$$

where $\boldsymbol{X}_t = \left(1 , X^1{}_t , \dots , X^{q+10}{}_t \right)^T$.

III. Given our strategy to invest 1 unit of money into S&P500 depending on the sign of our next day return prediction, we have that the actual return at time t+1 is:

$$R_{t+1} = \mathrm{sign}(\widehat{Y_{t+1}}) \times Y_{t+1}$$

IV. Return to step I, but with t replaced by t +1. Final iteration occurs when t=n-1 (where n is the number of days in the data set).

V. Finally, after we finish with all of the predictions, we calculate the annualized Sharpe ratio (ignoring all transaction costs) in the following way:

$$\text{Sharpe ratio} = \sqrt{250}\ \text{x}\ \frac{Mean\ daily\ actual\ return}{Standard\ deviation\ of\ actual\ daily\ returns}$$

Question 4

In this question we analyse the results produced using the functions get.sharpe and tune.win found in the appendix.

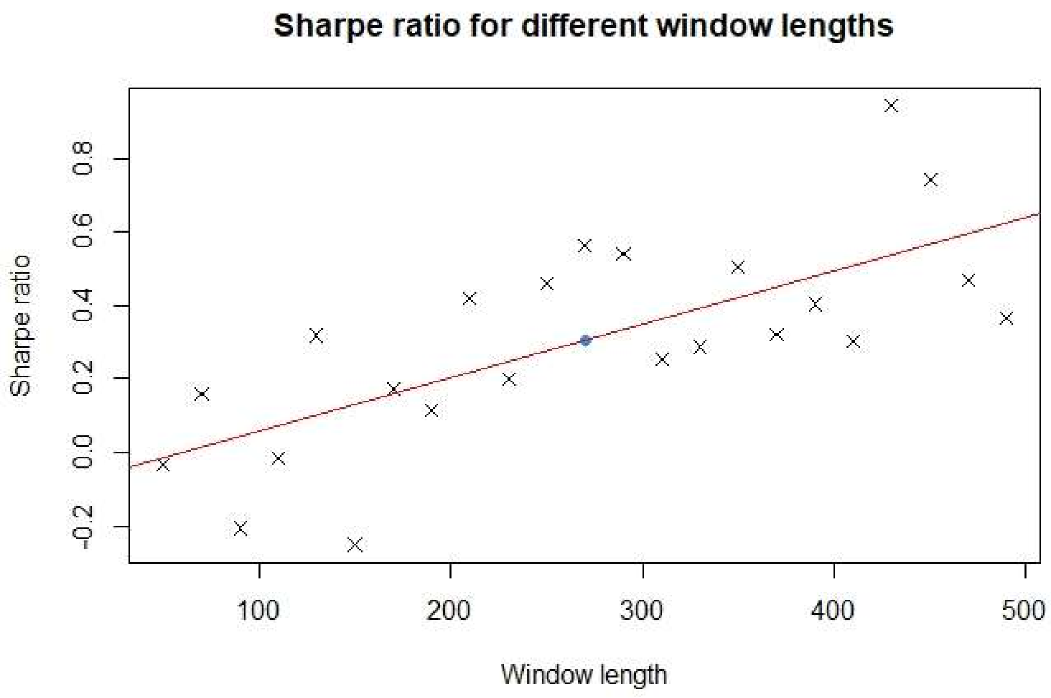
- Outputs of the **training set** with _0 S&P500 lags_ *included*.

```
[1] -0.03300624  0.15778080 -0.20621583 -0.01394391  0.31860589
-0.25065578  0.17351703  0.11593261  0.41981140  0.19935531
0.45933189
[12]  0.56315056  0.53935910  0.25263215  0.28675376
0.50422075  0.32039676  0.40374732  0.30282312  0.94246752
0.74167804  0.46781277 0.36435164
```

$mean.sharpe

```
[1] 0.3056481
```

The results are also presented graphically below:

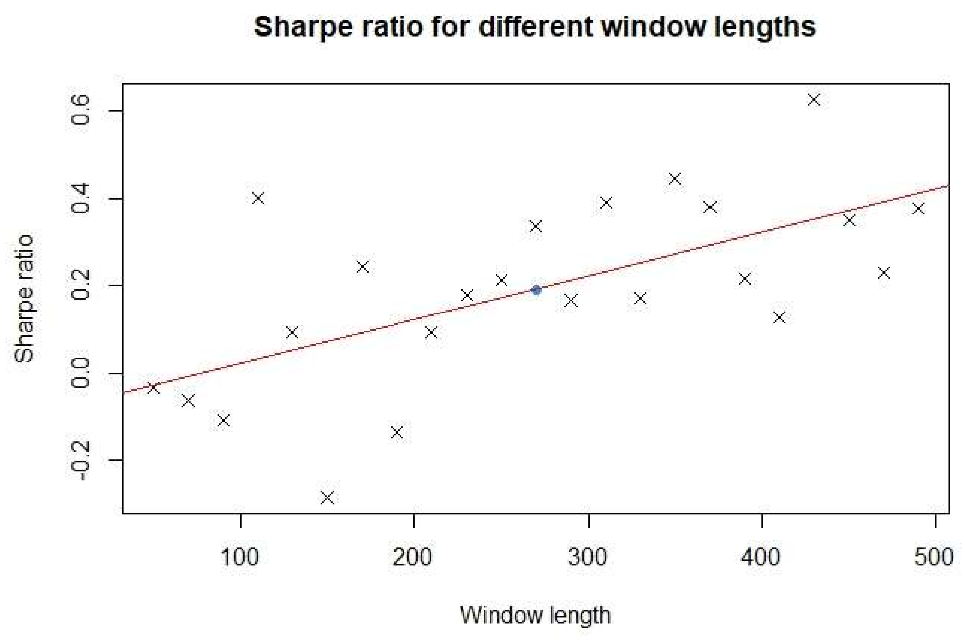


❖ Outputs of the **training set** with *<u>1 S&P500 lag</u> included*.

```
 [1] -0.03246758 -0.06308321 -0.11026132  0.39897854
0.09257883 -0.28477360  0.24427108 -0.13691534  0.09324701
0.17753445  0.21364938
[12]  0.33577612  0.16596239  0.39120250  0.17064092
0.44352307  0.38083348  0.21445345  0.12665792  0.62466878
0.34769333  0.22907683 0.37513658
```

$mean.sharpe

```
[1] 0.1912341
```



In both cases we can see that there seems to be an overall positive relationship between sharpe ratio and window length used which is something we should be expecting. However, including an S&P500 lag seems to reduce the mean sharpe ratio by a bit.

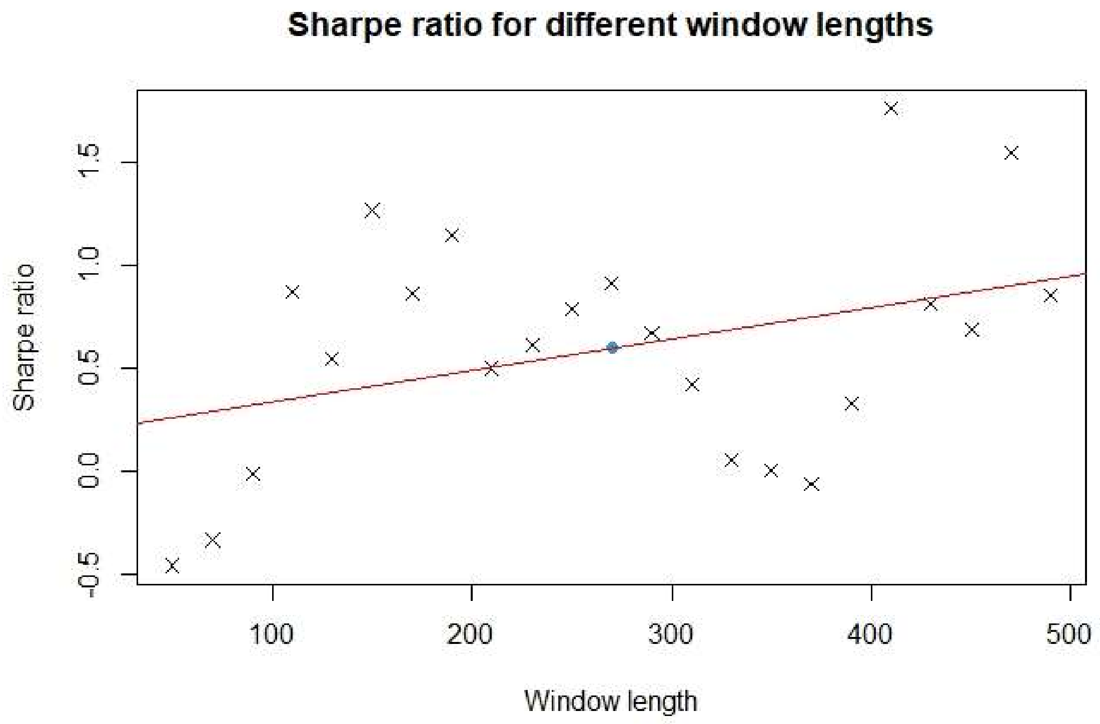
- Outputs of the **validation set** with *0 S&P500 lags included*.

```
[1] -0.460028499 -0.335661616 -0.011620147  0.872997508
0.549436018  1.268837921  0.863855547  1.150612428
0.500593982  0.615704526

[11]  0.790637391  0.912383200  0.674702283  0.422997797
0.050814776  0.006185205 -0.065591799  0.330948981
1.763633607  0.812653374 0.687853402  1.551108117
0.853536167
```

$mean.sharpe

```
[1] 0.6002865
```



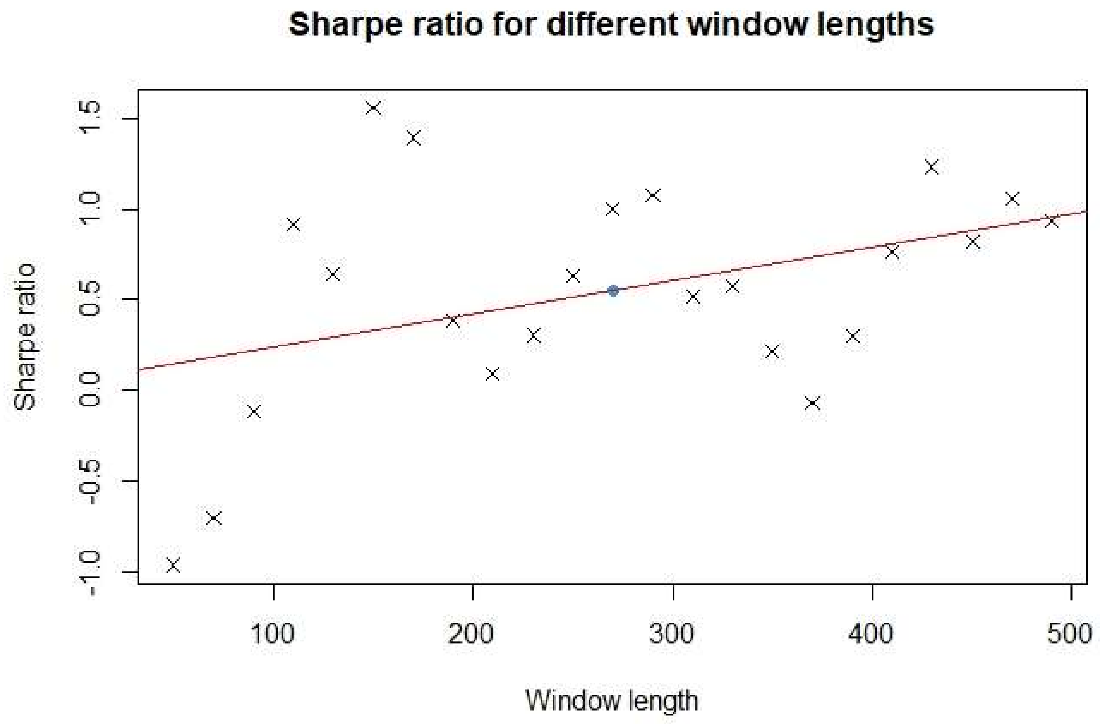
❖ Outputs of the **validation set** with *1 S&P500 lag included*.

```
[1] -0.96271811 -0.70189475 -0.11464491  0.91582233  0.64350787
1.55656541  1.39346709  0.38922461  0.09490591  0.30470364
0.62872395

[12]  0.99809791  1.07619511  0.52198241  0.57336381
0.21751992 -0.06559180  0.30423948  0.76223665  1.23423611
0.82422279  1.05982814 0.93438474
```

$mean.sharpe

```
[1] 0.5473208
```



Results in the validation set still suggest an overall positive relationship between sharpe ratio and window length, however, the results here are more variable and makes choosing the appropriate window size a difficult task. In addition, using an S&P500 lag on the validation set, slightly reduces the mean sharpe ratio just as in the training set.

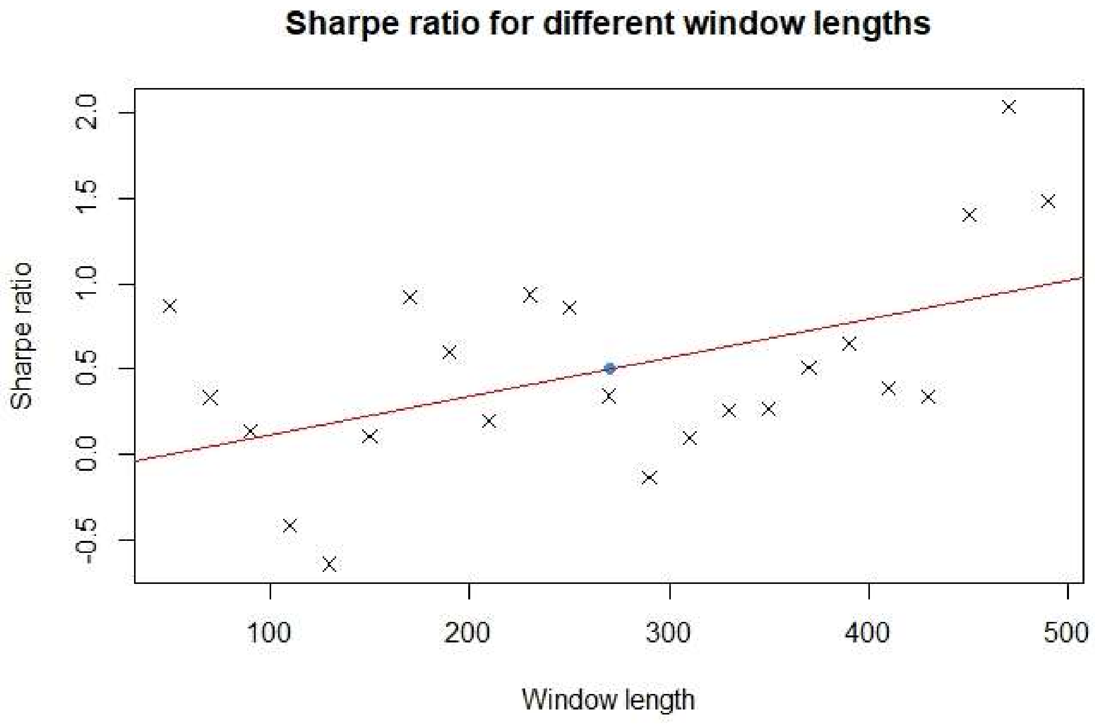
❖ Outputs of the **test set** with *<u>0 S&P500 lags</u> included*.

```
 [1]  0.86824893  0.33399139  0.14097650 -0.41461508 -
0.64051152  0.10525606  0.92059829  0.59733173  0.20143067
0.93693358  0.86368764

[12]  0.34414942 -0.13093082  0.09951611  0.26221856
0.26845719  0.51038694  0.65334249  0.39175742  0.34212467
1.40416963  2.03333458  1.48571150
```

$mean.sharpe

```
[1] 0.5033724
```



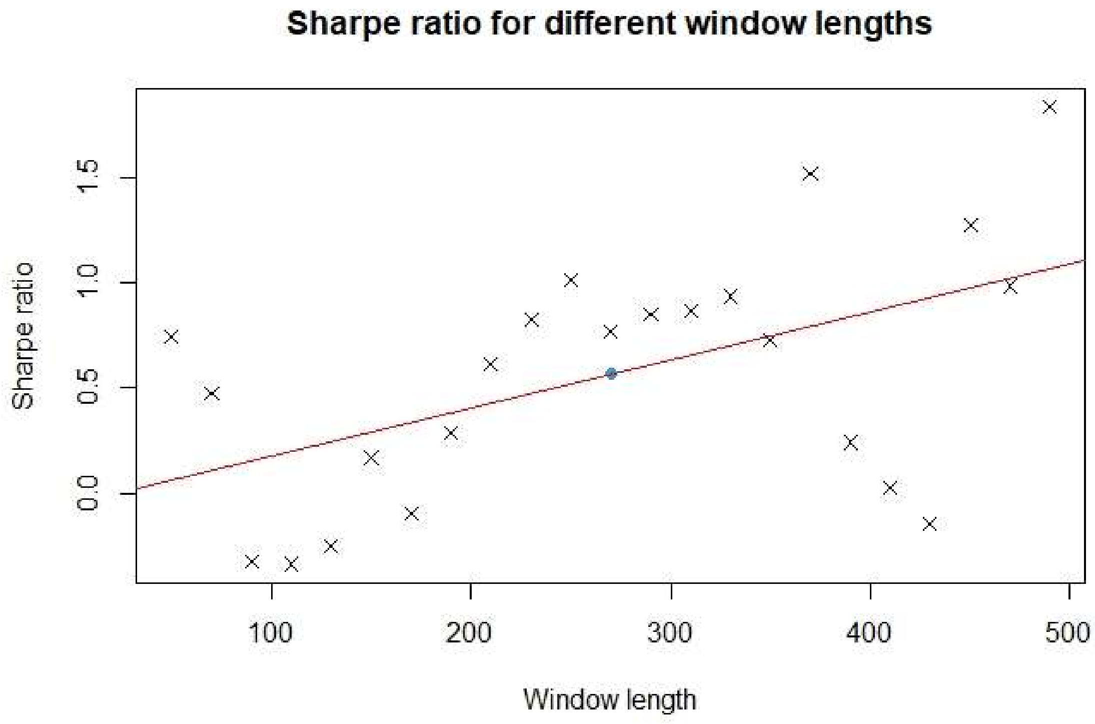
❖ Outputs of the **test set** with *1 S&P500 lag* *included*.

```
[1]  0.74265191  0.47698159 -0.32125299 -0.33466819 -0.24951357
0.17006577 -0.09620495  0.28709522  0.61251835  0.82628585
1.01289330
```

```
[12]  0.77071681  0.84811848  0.86309005  0.93508031
0.72677405  1.51856576  0.24207236  0.02954333 -0.14129035
1.27535281  0.98316307 1.83156056
```

$mean.sharpe

```
[1] 0.5656348
```



The results in the test set are relatively similar to the results obtained in the validation set. A key difference is that in this set, using an S&P500 lag improves the mean sharpe ratio. Perhaps this is because of some particular characteristics of the data in the test set, which includes data from the COVID-19 pandemic.

Concluding, using ordinary least squares for predictions over the three different sets discussed above seems to be doing a decent overall job. However, only the training set produces results which show a stable and strong positive relationship between window size and sharpe ratio, and this makes choosing the optimal window size a hard task. Following the procedure in the lecture notes, we tune the window size over the validation set, and thus a window size of around 400-450 seems to be appropriate.

A serious problem in the ordinary least squares method for predicting next day returns seems to be multicollinearity. After having checked the standard errors of the regression coefficients and the p-values of the individual significance for the coefficients, multicollinearity is indeed an issue here. The output of part of the multicollinearity analysis on the test set can be seen below:

```
Call:
lm(formula = y.test[102:251] ~ x.test[101:250, ])

Residuals:
    Min      1Q  Median      3Q     Max
-3.6236 -0.5545  0.0692  0.5831  2.8167

Coefficients:
                      Estimate  Std. Error t value  Pr(>|t|)
(Intercept)           0.009248   0.088189   0.105    0.917
x.test[101:250, ]1  -0.145938   0.155415  -0.939    0.349
x.test[101:250, ]2   0.228587   0.199012   1.149    0.253
x.test[101:250, ]3  -0.060322   0.099387  -0.607    0.545
x.test[101:250, ]4   0.036994   0.059420   0.623    0.535
x.test[101:250, ]5  -0.068220   0.122686  -0.556    0.579
x.test[101:250, ]6   0.104464   0.142220   0.735    0.464
x.test[101:250, ]7  -0.283714   0.195853  -1.449    0.150
x.test[101:250, ]8   0.002967   0.068261   0.043    0.965
x.test[101:250, ]9   0.118611   0.176230   0.673    0.502
x.test[101:250, ]10 -0.069342   0.096432  -0.719    0.473
```

The magnitude of the standard errors for each coefficient and the p-values are really large and indicate that multicollinearity must be addressed here. The motivation for using Principal Component Regression (PCR) in Q5 is that it is an effective method of addressing multicollinearity because of the orthogonality of the resulting principal components obtained through PCA.

Question 5

In this part, I analyse the results obtained using the function *pc.sharpe* and *win.pc.sharpe* which are used to obtain sharpe ratios for different window lengths and different number of principal components.

- ❖ **Training set** outputs

Using 1 principal component:

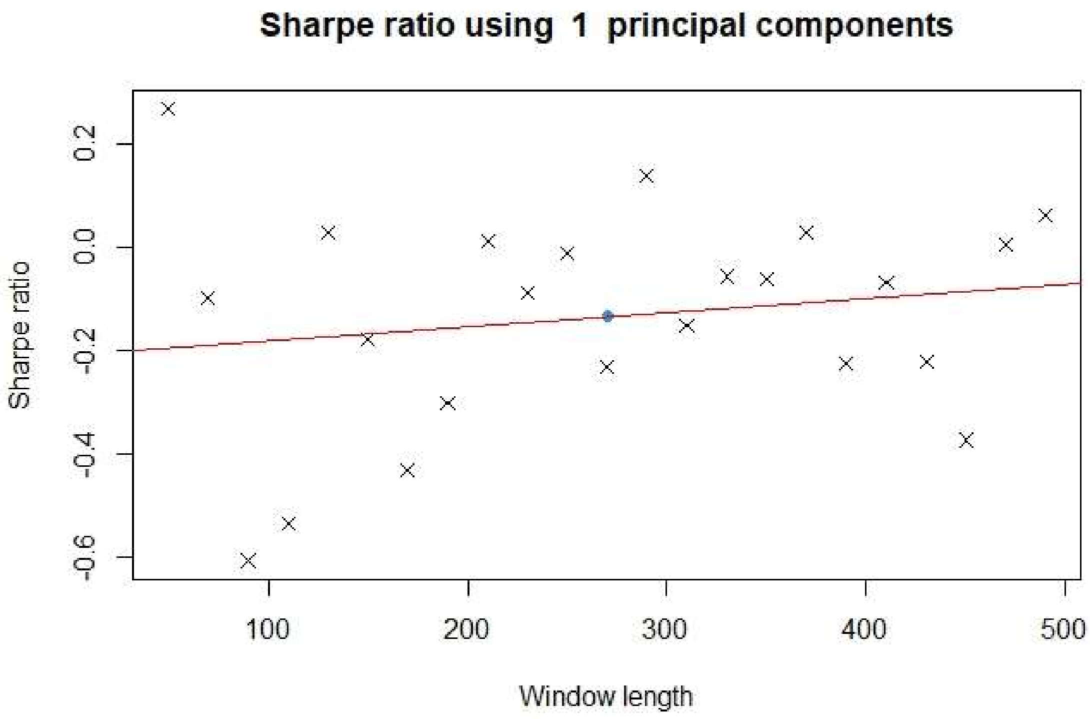
[1] 0.265780129 -0.098860286 -0.605772011 -0.534621646 0.027361120 -0.179710186 -0.431578212 -0.301480742 0.011290681 -0.088862002

[11] -0.011893262 -0.232219352 0.137726445 -0.152780584 -0.057686431 -0.062438297 0.028487909 -0.224410565 -0.070222512 -0.222325719

[21] -0.372953153 0.005036257 0.060439076

$mean

[1] -0.135291



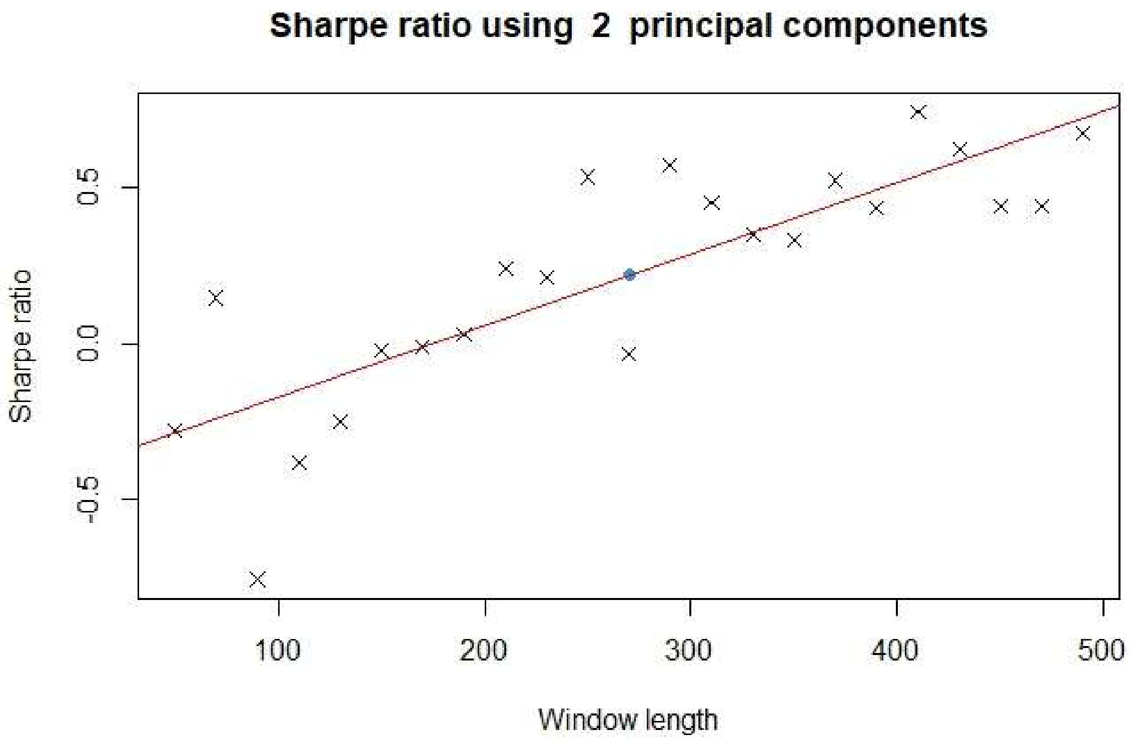
Using 2 principal components:

```
[1] -0.28099456  0.14691313 -0.75823748 -0.38245335 -0.24935728
-0.02190119 -0.00832598  0.03163579  0.24247058  0.21241575
0.53748824

[12] -0.03395739  0.57382961  0.45438541  0.35157063
0.33271284  0.52561897  0.43728860  0.74493513  0.62526788
0.44260678  0.44334507  0.67622368
```

$mean

```
[1] 0.2192818
```



In the training set we can see that using 2 principal components significantly improves the mean sharpe ratio. Although the first plot does not give a clear idea of what the window size should be, the second plot suggests that a window size of around 400 should be the best.

❖ **Validation set** outputs

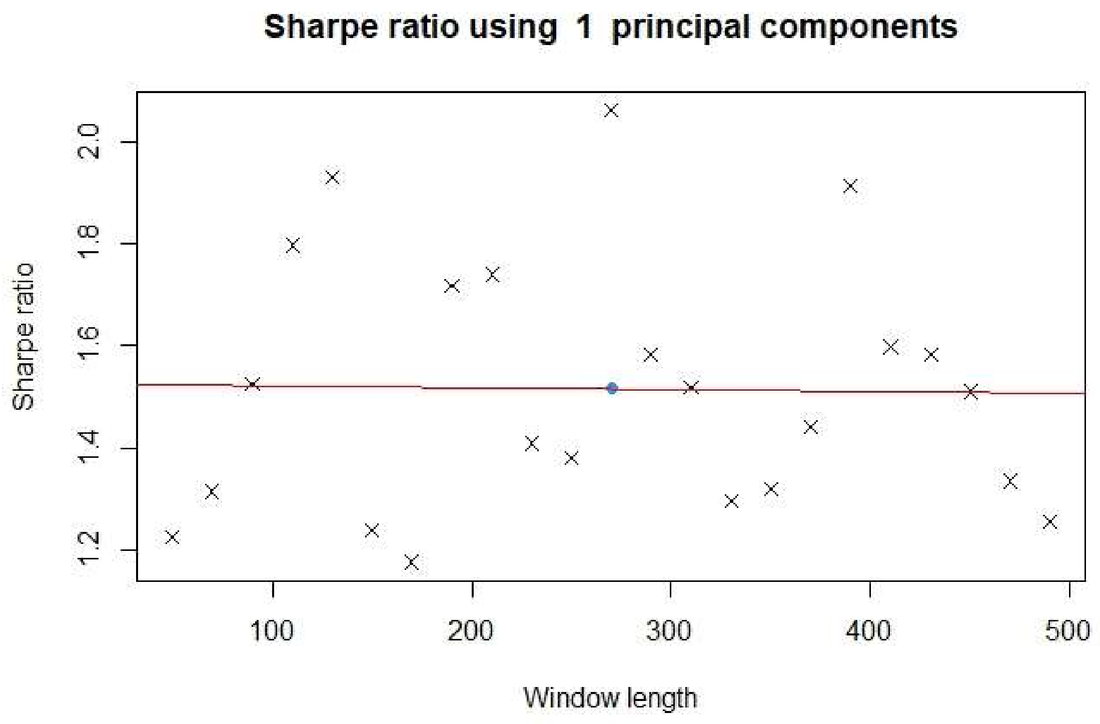
Using 1 principal component:

```
[1] 1.225419 1.313472 1.525796 1.797477 1.931787 1.238324
1.174837 1.718965 1.739959 1.408516 1.380988 2.062322
1.581881 1.517349 1.295958

[16] 1.317576 1.440871 1.914604 1.597664 1.582692 1.510090
1.334221 1.255584
```

$mean

```
[1] 1.515928
```



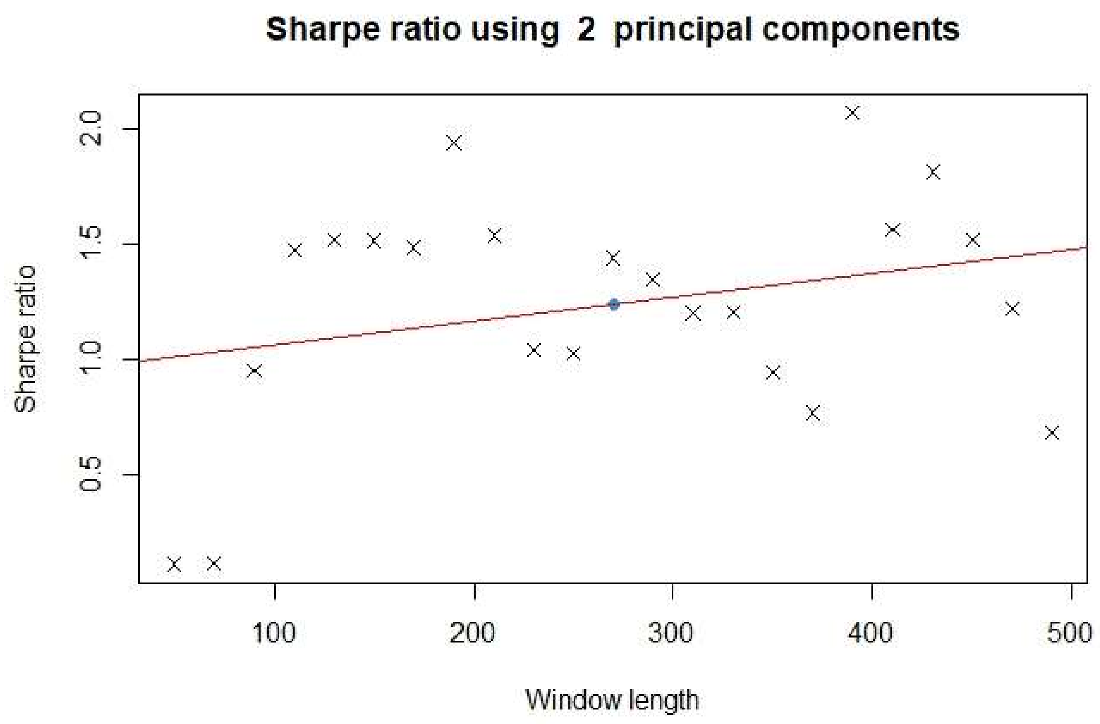
<u>Using 2 principal components:</u>

```
[1] 0.1090297 0.1106657 0.9490052 1.4730549 1.5162581
1.5147926 1.4853845 1.9410130 1.5371639 1.0377095 1.0279093
1.4407805 1.3487241 1.2009030

[15] 1.2053654 0.9435651 0.7672911 2.0700683 1.5624508
1.8142393 1.5158866 1.2175780 0.6843402
```

$mean

```
[1] 1.237964
```



Judging from the validation set results, no clear pattern exists when it comes to the window length. Also, the mean sharpe ratio using 1 PC is higher, and in both cases, the results seem to be much better compared to the validation set results in Q4.

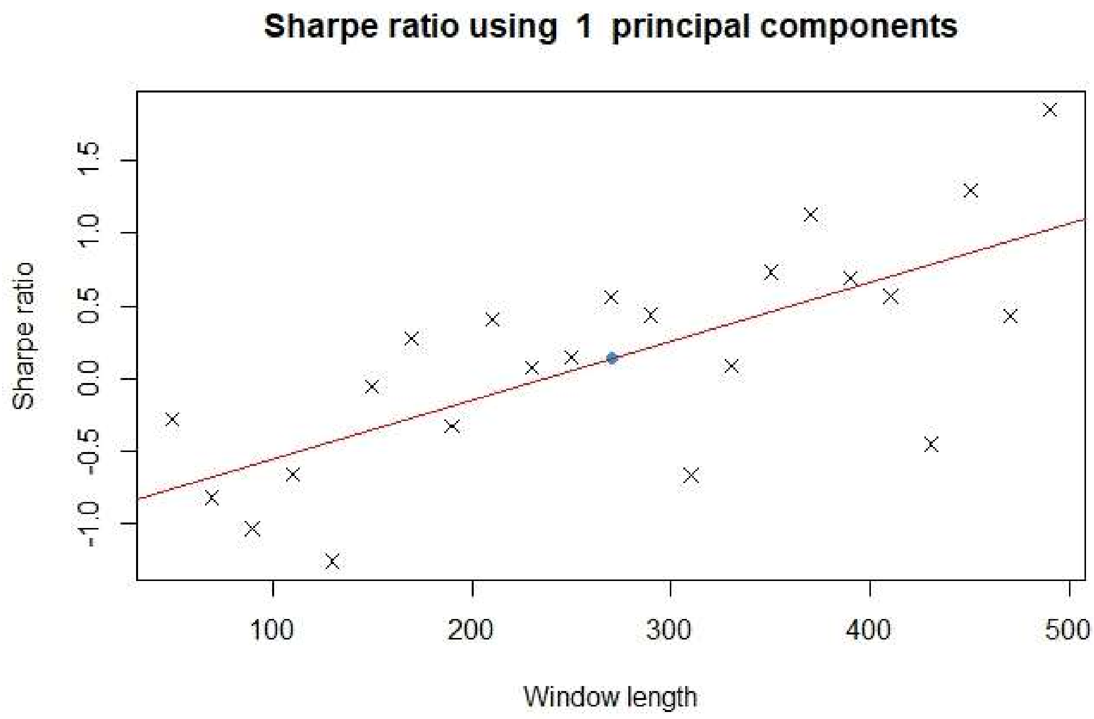
❖ **Test set** outputs

Using 1 principal component:

```
[1] -0.28372320 -0.81866067 -1.03262659 -0.65763861 -1.25802696
-0.05761120  0.27432908 -0.32292165  0.40454368  0.07283047
0.14212183
[12]  0.56425690  0.43776484 -0.66429418  0.08840808
0.73271032  1.12566077  0.68946753  0.56611388 -0.45092411
1.29940755  0.42776925  1.84989652
```

$mean

```
[1] 0.1360371
```



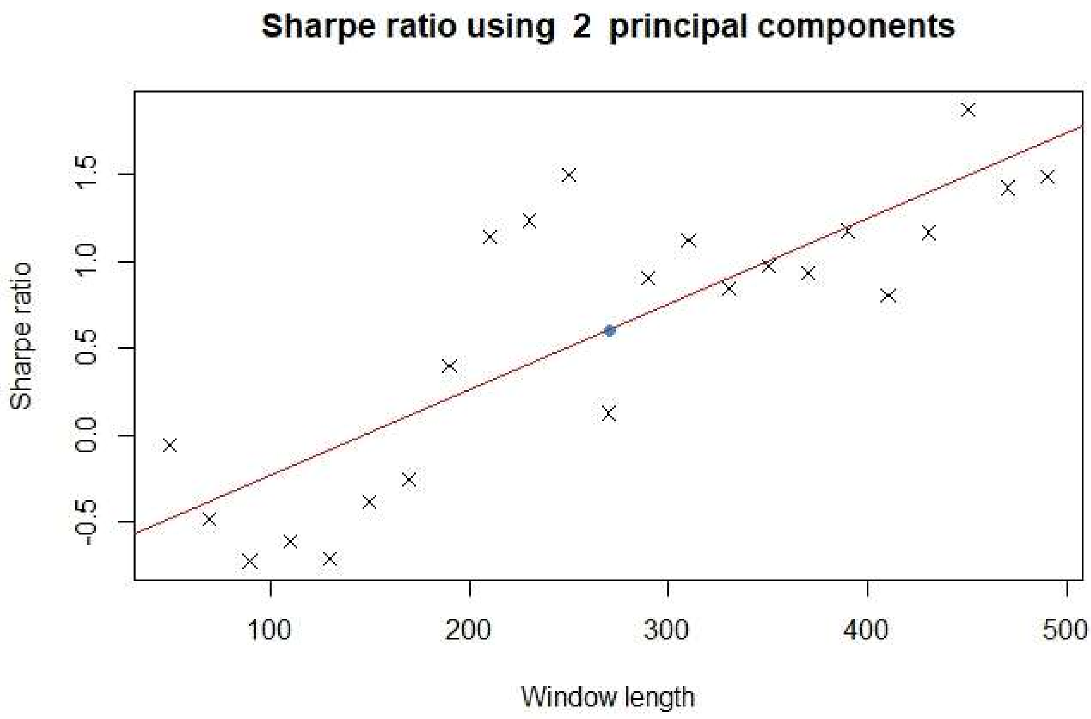
Using 2 principal components:

```
[1] -0.05538517 -0.48758905 -0.72829080 -0.61351773 -0.71649183
-0.38452531 -0.25728469  0.39975561  1.14202030  1.23326634
1.49753893

[12]  0.12524759  0.90215531  1.11919295  0.84107707
0.97205952  0.93374589  1.17542403  0.80231001  1.16459797
1.87093668  1.42175260  1.48304151
```

$mean

```
[1] 0.6017842
```



Finally, in the validation set we can clearly see that larger window sizes seem to perform significantly better in both cases. In addition, including 2 principal components improves the score by a lot.

All in all, we can conclude that in general, including 2 principal components seems to produce better results compared to using 1 principal component. However, the optimality of the window size cannot be determined using the results obtained above because there is no clear pattern.

Finally, using PCR with 2 principal components seems to outperform the method proposed in Q4 in the validation and test sets. As for the training set, the results seem to be roughly similar.

These results suggest that PCR indeed tackles the problem of multicollinearity effectively and produces a model which significantly improves ordinary least squares according to the sharpe ratio metric.