Summer Assessment 2020

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

COVERSHEET FOR SUBMISSION: ST436 – FINANCIAL STATISTICS

1. You must complete all sections of this coversheet for submission and insert it into the front page of your work (i.e. 'Select All' then Copy and Paste this page into your assessment). **You must ensure you upload the correct and most up to date version of your work and keep a copy of it before submitting it.**

2. Your submission should be saved (with this coversheet at the front) as a **Microsoft Word Document (.doc or .docx) or pdf file** and uploaded as **ONE individual file**.

3. Save your assessment using your five-digit **candidate number** followed by your **course number** in the filename (e.g. 78654-LL4Y9.doc).

4. Make sure your five-digit candidate number is included in your document (i.e. in the header or footer). **Your name must NOT appear anywhere in the document.**

5. You must ensure you submit your work, including the completed coversheet, within the specified deadline. Failure to submit by the deadline could result in the application of late submission penalties.

CANDIDATE NUMBER: (**Five digit number** available via LSE for You – this is not the same as the number on your LSE ID card.)	4 | 1 | 5 | 2 | 4

Fit to sit / submit

By submitting this work you have declared yourself fit enough to do so. "Fit" in this instance does not only apply just to physical or mental health, but other factors which may affect your academic performance.

If you do not feel that you are fit to submit, you should follow the School's exceptional circumstances procedure which, if applicable for the assessment type, includes seeking an extension or deferring the assessment.

Deferral

Please note, for coursework (assessments longer than 24 hours), if you submit the assessment any previous deferral request will be withdrawn and the mark for the assessment added to your record.

For 24-hour assessments, including short-timed assessments, if you wish to withdraw a previous deferral you must do so before the assessment is *first* released within the assessment window. Otherwise the deferral will stand even if you submit the assessment.

Students with a disability

Students that have been provided with a **'Letter of Notification' (LoN)** from the LSE must ensure they upload the LoN as part of their submission (in the single document submitted as part of the assignment).

Please put an X in this box if you are happy for us to use your work as an example for future students. (All identifying features will be removed).	X

Please indicate the questions you have answered before submitting your paper

Question numbers: Please indicate here the questions which have been attempted	1	2	3		

Attach this coversheet to your answer paper.

Please indicate the question number in the box below and the word count.

Question Number	1	Word count	1594

0.1 Q1

1.(a) The paper investigate whether GARCH(1,1) is useful to describe and forecasting the volatility of S&P 500 stock market index. For S&P 500 index, GARCH(1,1) is not the true data generating process and applying GARCH(1,1) to this data will be problematic, especially in long horizon. Moreover, the GARCH(1,1) model provide no better forecast than non-parametric model(i.e. Nadaraya-Whatson kernel estimate).

By fitting GARCH(1,1) and kernel estimate(uniform) to S&P 500 index and evaluating our-of-sample forecasting performance, the paper argues that the GARCH(1,1) is going to have higher MSE than kernel estimate(with uniform kernel, the benchmark model in section 2.3). The unconditional volatility by GARCH(1,1) is always higher than sample variance of return, which means that GARCH(1,1) tends to overestimate the unconditional variance and fail to provide accurate longer horizon estimate. What's more, the paper argues that the S&p 500 index is not stationary(which is the assumption of GARCH family) and GARCH model may suffer from IGARCH effect in some specific period. GARCH might have better forecast than kernel estimate for short horizon but produce worse forecast than kernel estimate for longer horizon.

1.(b) The GARCH(1,1) model can be expressed as:

$$X_t = z_t \sigma_t,$$

$$\sigma_t^2 = a_0 + a_1 X_{t-1}^2 + b_1 \sigma_{t-1}^2$$

where z_t is an independent random variable with $E(z) = 0, E(z^2) = 1$. The first '1' in GARCH representation stands for the order of X_{t-1}^2, the second '1' stands for σ_{t-1}^2. GARCH model assumes that the data is stationary(weakly and unconditionally) so that the data has constant unconditional variance over time, from which we can derive $E(\sigma_t^2) = E(\sigma_{t-\tau}^2) = E(\sigma^2)$ for any integer τ. Therefore we can calculate unconditional variance in this way:

$$\begin{aligned}
&E(\sigma_t^2) = E(a_0 + a_1 X_{t-1}^2 + b_1\sigma_{t-1}^2),\\
&E(\sigma^2) = a_0 + a_1 E(z_{t-1}^2)E(\sigma^2) + b_1 E(\sigma^2),\\
&E(\sigma^2) = \frac{a_0}{1-a_1-b_1}.
\end{aligned}$$

Also we can calculate the conditional variance. With available information up to time t, we can produce the k step variance forecast. To derive the forecast, we can first rewrite the GARCH representation as:

$$\sigma_t^2 - \sigma^2 = a_1(X_{t-1}^2 - \sigma^2) + b_1(\sigma_{t-1}^2 - \sigma^2)$$

This is done by substituting $a_0 = (1-a_1-b_1)\sigma^2$ into $\sigma_t^2 = a_0 + a_1 X_{t-1}^2 + b_1\sigma_{t-1}^2$. By taking expectation on the above equation, we ends up with:

$$\begin{aligned}
&E(\sigma_t^2 - \sigma^2|\mathscr{F}_{t-2}) = a_1 E(X_{t-1}^2 - \sigma^2|\mathscr{F}_{t-2}) + b_1 E(\sigma_{t-1}^2 - \sigma^2|\mathscr{F}_{t-2}),\\
&E(\sigma_t^2|\mathscr{F}_{t-2}) - \sigma^2 = a_1 E(\sigma_{t-1}^2 - \sigma^2|\mathscr{F}_{t-2}) + b_1 E(\sigma_{t-1}^2 - \sigma^2|\mathscr{F}_{t-2}),\\
&E(\sigma_t^2|\mathscr{F}_{t-2}) - \sigma^2 = (a_1+b_1)E(\sigma_{t-1}^2 - \sigma^2|\mathscr{F}_{t-2}) = (a_1+b_1)E(\sigma_{t-1}^2 - \sigma^2).
\end{aligned}$$

The value of σ_{t-1}^2 is already known with information up to $t-2$. The above equation holds for any value of t. Then we can substituting the above equation into $E(\sigma_{t+k}^2|\mathscr{F}_t) - \sigma^2$ iteratively. The k step ahead forecast would then be:

$$E\left(\sigma_{t+k}^2|\mathscr{F}_t\right) - \sigma^2 = (a_1+b_1)^{k-1} E(\sigma_{t+1}^2 - \sigma^2|\mathscr{F}_t),$$

$$E\left(\sigma_{t+k}^2|\mathscr{F}_t\right) = a_0\frac{1-\lambda^k}{1-\lambda} + \left(a_1 X_t^2 + b_1\sigma_t^2\right)\lambda^{k-1}$$

where $\lambda = a_1 + b_1 < 1$. When estimating GARCH(1,1) in R, a_0, a_1, b_1 is in the output. X_t is the last point in the return data and σ_t can be easily obtained in the output of GARCH, so it's easy to forecast variance, both k step conditionally and unconditionally. What's more, as k goes to infinity, the conditional variance converges to unconditional variance.

The setting of kernel estimate is different. For kernel estimate, we don't need to assume the stationarity of data. Moreover, kernel estimate is a non-parametric method and we don't assume that the data follows parametric distribution. This approach assumes that:

$$X_t = \sigma(t) z_t$$

where $\sigma(t)$ is a deterministic, smooth time varying function. Though the exact evolution of $\sigma(t)$ is unknown, we can approximate it by applying kernel function to past data. That is

$$\widehat{\sigma^2}(t;b) = \sum_{k=1}^{n} W_k(t;b) X_k^2$$

where b stands for the bandwidth of the half kernel and $W(\cdot)$ is the weighting function and can be written as:

$$W_k(t;b) = K\left(\frac{t-k}{b}\right) / \sum_{k=1}^{n} K\left(\frac{t-k}{b}\right)$$

There are various choices of kernel function $K(\cdot)$ like exponential, normal and uniform. We choose to use uniform kernel, where the function is constant for different value of input(i.e. $W(t;b) = \frac{K}{nK} = \frac{1}{n}$). Then the estimation can be simplified as $\widehat{\sigma^2}(t;b) = \sum_{k=1}^{n} \frac{1}{n} X_k^2$. By choosing a reasonable width of window, we can derive the estimated volatility from the above equation.

The k step forecast of kernel estimate is nothing but the most recent estimated variance itself. The best we can do is to measure the current level and to forecast it as the return volatility for the close future(Starica, 2003: 17).

1.(c) From my perspective, GARCH model is suitable for modeling stationary data. Kernel estimate is more flexible and can apply to almost all return data, except for those whose volatility changes dramatically. In financial time series data, unconditional volatility may change due to change in market environment, etc. Such change of volatility can not be well captured by the GARCH model, which assumes stationarity, and may cause bias in coefficients. This is not a problem for kernel estimate because it does not assume stationarity. For stationary data, kernel estimate also make sense because weighting average of past stationary data should be close to its volatility. Therefore, I think if our aim is to forecast rather than to make inference, kernel estimate should be more appropriate than GARCH, especially for non-stationary data or data with IGARCH effect.

I evaluate performance of different model by their SE(Square Error). The construction of SE is similar to that in paper(Starica, 2003). As we have mention in 1.(b), the k step forecast variance is $E\left(\sigma^2_{t+k}|\mathscr{F}_t\right) = a_0\frac{1-\lambda^k}{1-\lambda} + \left(a_1 X_t^2 + b_1\sigma_t^2\right)\lambda^{k-1}$. The next p aggregated volatility, is therefore given by

$$\bar{\sigma}^{2,GARCH}_{t,p} = E_t\left(X_{t+1} + \ldots + X_{t+p}\right)^2 = \sigma^{2,GARCH}_{t+1} + \cdots + \sigma^{2,GARCH}_{t+p}$$

The k step volatility forecast for kernel estimate is $\sigma^2_{t+k} = \widehat{\sigma^2}(t;b) = \sum_{k=1}^{n}\frac{1}{n}X_k^2$. Therefore the p aggregated volatility we derive is

$$\bar{\sigma}^{2,KS}_{t,p} = p\widehat{\sigma}^2(t) = \frac{p}{n}\sum_{i=1}^{n} r^2_{t-i+1}$$

This result is obtained by setting kernel function to uniform kernel and constraint only n data involved in volatility estimation. Similarly, we can define the p step aggregated realized volatility as

$$\bar{r}^2_{t,p} = \sum_{i=1}^{h} X^2_{t+i}$$

Then the SE function is therefore

$$SE^*(p) = \sum_{t=1}^{n}\left(\bar{X}^2_{t,p} - \bar{\sigma}^{2,\cdot}_{t,p}\right)^2$$

where * stands for KE(kernel estimate) or GARCH. Then we can calculate the SE proportion of the two models(i.e SE^{GARCH}/SE^{BM}). We then estimate GARCH(1,1) model for the first 1000 data and then re-estimate every 10 days(2 weeks) while keeping the sample size fixed(i.e. moving the estimation window). We then use kernal estimate for the 250 most recent returns in the estimation window(i.e. $n = 250$). The SE is calculated up to $p = 250$. Then for each re-estimate, we can obtain their SE proportion to 250 lags.

The GARCH models are sometimes badly performed(though most of the times all fits behave reasonable). For example, when we are estimating S&P 500 index and our estimation window is 2002/09/11-2006/08/30, the GARCH(1,1) fit will give unconditional volatility $-1.90 * 10^{-13}$ because $a_1 + b_1 = 1.08$, which is impossible for time series data to have negative volatility. Such badly performed model will have large leverage effect on our estimated MSE proportion if we simply average all SE to get MSE. To overcome this, I use the median of SE proportion as our performance indicator(i.e. $MEDIAN(SE^{GARCH}/SE^{BM})$). For every estimation window we will calculate the SE proportion up to lag 250. For every lag, we then calculate the median of SE proportion among all fits. This method is more stable than MSE(median is not sensitive to extreme value like mean does). What's more, the median provides a 50-50 division and we can clearly see that which model performs better for most circumstances.

Data we use are FTSE 100, Hang Seng, NASDAQ, S&P 500, stock price of BOEING and exchange rate of EUR/USD. The four index comes from bossa data(http://bossa.pl/pub/indzagr/mstock/mstzgr.zip). They are loaded with all data points up to 2020/05/06 and are preprocessed by 'read.bossa.data.single', which is a modified version of 'read.bossa.data' in course codes but allows for reading in single index and permit return to have whole length(rather than several times of 2). Stock price of BOEING is downloaded in Wharton/CRSP/CRSP Daily Stock with PERMNO=19561 and Date range from 2004-01-01 to 2019-12-31. The return is obtained by taking difference of log transformation and then divided by its standard error(to be consistent with 'read.bossa.data.single'). The EUR/USD exchange rate comes from https://www.ofx.com/en-gb/forex-news/historical-exchange-rates/ where reporting period is 'Last 10 Years' and data up to 6th decimal number. The return is obtained by simply differentiate the exchange rate and divided by its standard error.

The result in shown in Fig.1. For most series, kernel estimate outperform GARCH model except for FT-SE 100 index. For $p < 50$, GARCH model tends to have smaller SE than kernel estimate in FT-SE 100. This relationship reverse when p>50. What's also notable in the figure is that GARCH model performs much worse than kernel estimate in EUR/USD series. This might because exchange rate is not stationarity even in a short horizon.

Then we turn to measure the model performance model in this extraordinary period. First we divide data after Jan,2nd as the test set and the remaining as the training set. We then

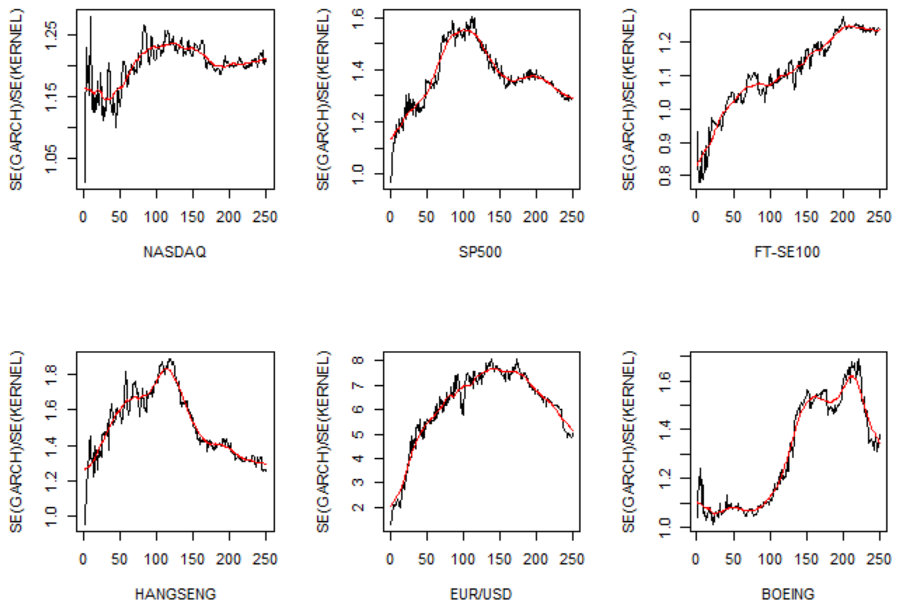


Figure 1: Median SE Proportion of Different Series

fit GARCH(1,1) with window size 1000 and fit kernel estimate with 250 most recent data. Then we compute the estimated aggregated volatility of each models and compare them with realized volatility. The result is shown in Fig.2. Black line is realized aggregated volatility, blue dotted line is from GARCH model, red dotted line is from kernel estimate. Both models fails to capture the high volatility in following periods. Then we change starting date to Mar,6th and set window size to 400 and bandwidth of kernel to 100. The result is shown in Fig.3. Similarly, both models fail. GARCH model seems to have higher volatility forecast than kernel estimate for NASDAQ and S&P 500.

What if we have observed 2 circuit breaker? We then set starting date to Mar,13th with window size 400 and kernel width 100. As we can see in Fig.4, GARCH model provides good estimate for NASDAQ and SP500, while significant overestimate for FT-SE 100 and underestimate for HANGSENG(window size 1000 and kernel width 250 provide good fit for FT-SE 100 but bad fit for the other three, we don't present the result here). Kernel estimate still provides bad volatility forecast.

To summarize, kernel estimate provide better forecast than GARCH(1,1) model for most of

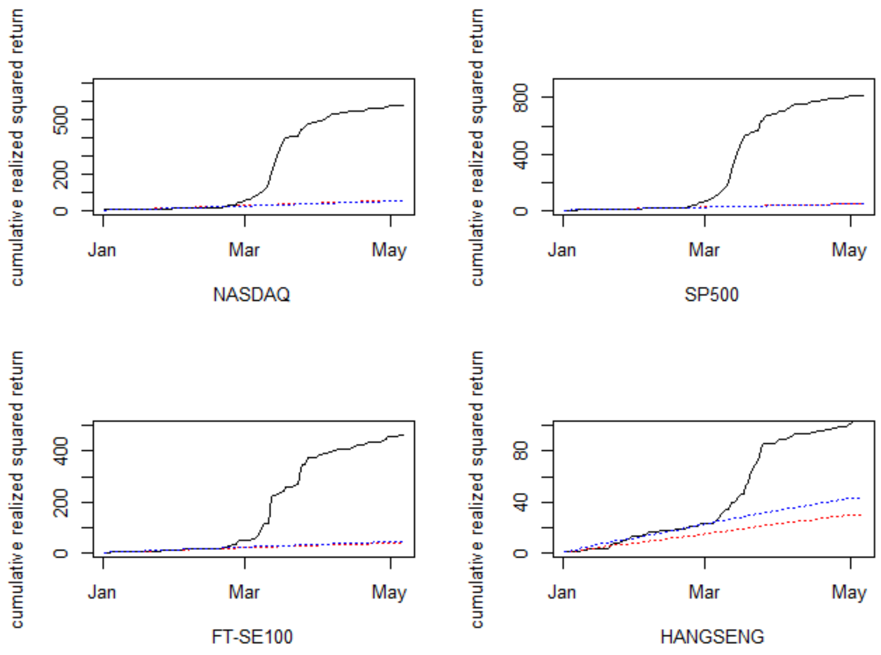


Figure 2: Estimated Volatility and Realized Volatility from Jan,2nd

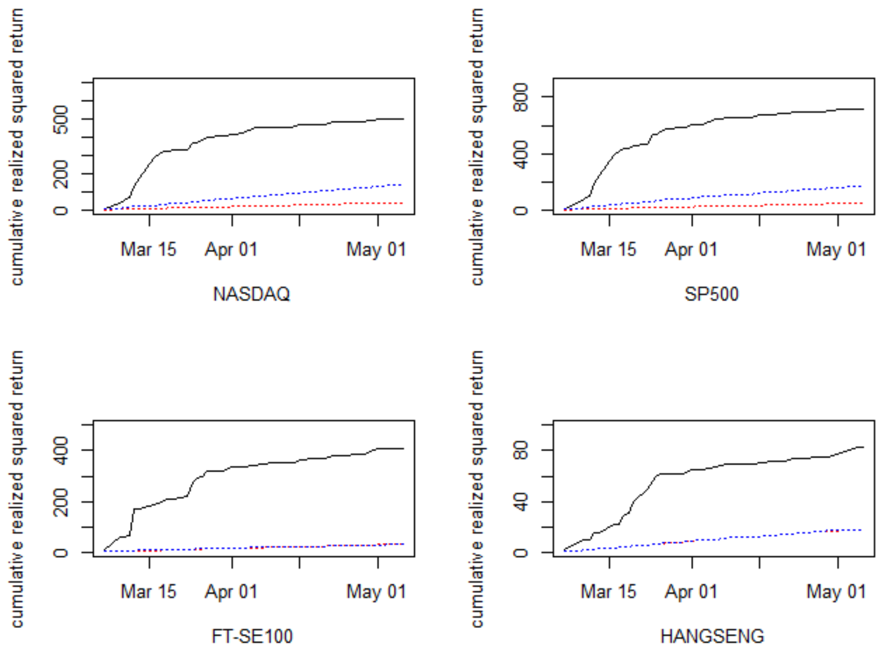


Figure 3: Estimated Volatility and Realized Volatility from Mar,6th

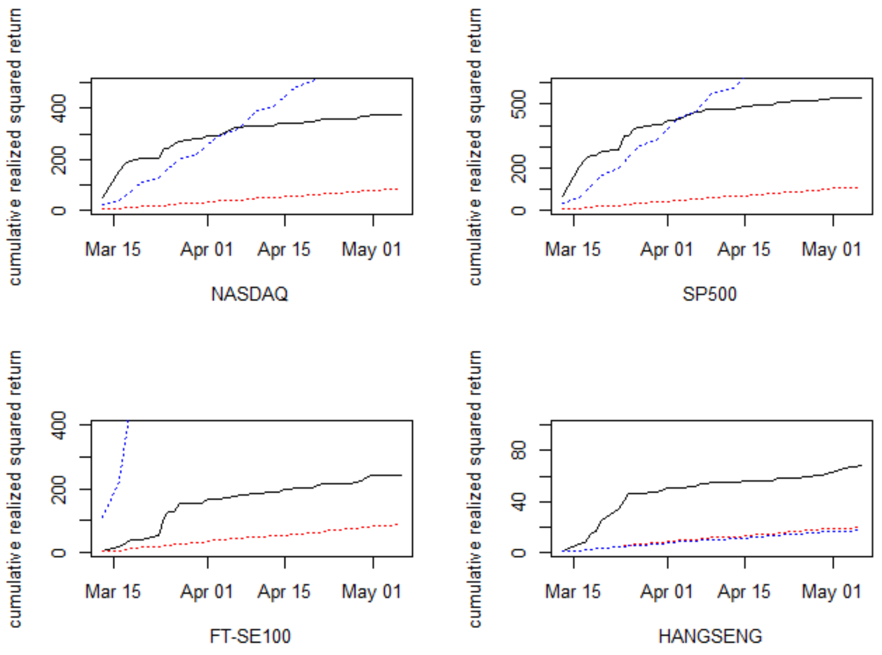


Figure 4: Estimated Volatility and Realized Volatility from Mar,13th

the time. However, both models fail if there is a significant change to the market. Though they fail, GARCH(1,1) model with short window can pick up the changing dynamics in the market more quickly than kernel estimate with short width. Therefore, in times of great uncertainty, GARCH(1,1) model may be more conservative and safer to apply.

Question Number	2	Word count	1062

0.2 Q2

2.(a) Due to availability of data, I only use Dow Jones Industrial Average Index(DJIA) to analyse the magnitude of the market crash in 1929. For the crash in 1987, data of Hang Seng Index, NASDAQ Index, S&P 500 Index, Dow Jones Industrial Average Index and Nikkel 225 Index are used for analysis. Hang Seng Index, NASDAQ Index, S&P 500 Index, FTSE 100 Index, Nikkel 225 Index and Russel 2000 Index are used for crash in 2008 and in 2020. DJIA, NASDAQ, S&P 500 are representatives of US stock market, while FTSE 100 stands for Europe market and Nikkel, Hang Seng stands for east Asia market. Performance of small-cap companies can be measured by Russel 2000 Index.

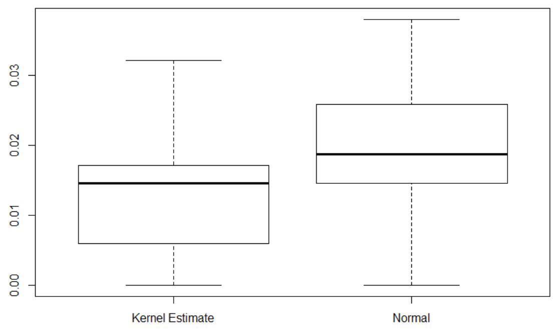
Daily close of DJIA between 1920 and 1935 is downloaded from Dow Jones Daily of WRDS database. Data for the other 3 market crashes are all from bossa data(`http://bossa.pl/pub/indzagr/mstock/mstzgr.zip`) and loaded with 'read.bossa.single' function.

Returns of DJIA are calculated by differentiating log daily close and then divided by the standard deviation. Standardization is need to be consistent with data preprocessing of 'read.bossa.single' function.

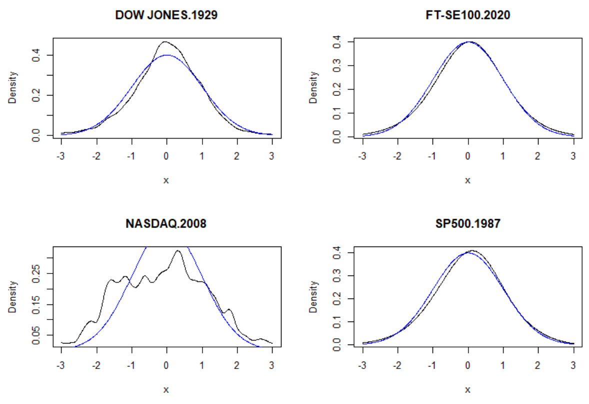
As we are modeling the magnitude of market swings, we have to specific a date where events occur. We assign oct. 28th, 1929, Oct. 15th, 1987, Sep. 29th, 2008 and Mar. 9th, 2020(when the first circuit breaker occurred in US market) as the starting date of crashes. Such specifications of starting date are made by reading the relevant Wiki page.

2.(b) In this section we mainly use Value at Risk(VaR) to model the magnitude of market movements. A simple way to visualize such movements is to compare the realized return with the 1% VaR and 5% VaR, which are calculated with data before crashes, and check if realized returns exceeded VaR levels. Another common way is to calculate the max drawdown of market index. Before we move to measure such magnitude, it's better to investigate whether normality assumption is appropriate for calculating VaR.

Here we compare the performance of normal distribution with that of kernel density estimation. A set of data, which begins from around 8 years to last day before event(except for 'HANGSENG' in 1987, in which we only have 80 data before event), is used to train, tune parameter and test the performance of estimate. We then divide this set of data into three parts. The first part is used to fit kernel density estimation with Gaussian kernel and ranges of different bandwidths. Based on the density function we estimate, we then calculate likelihood on second part of data. The bandwidth with highest likelihood is then chose as our best parameter. Then we compare the performance of kernel density estimate with normal distribution by calculating proportion of data below specific percentile of each distributions. The two distributions both perform well when we use 5-percent percentile for different indexes(we don't present plot here, data can be found in 'out.plot.temp' in code output). However, if we change the percentile to 1-percent, as we can see in Fig.5a, normal distribution tends to overestimate proportion of data



(a) Percentage of Data Below 1% Percentile of Each Distribution



(b) Comparison of Kernel Estimate(Black) and Normal Distribution(Blue)

Figure 5: Comparison of Each Distribution

below 1-percent percentile. Kernel density estimate provides a better estimation of extreme value. As VaR is exactly measuring the tail risk, kernel density estimate is then more preferred to model VaR. In Fig.5b, we compare the probability density function of kernel estimate with normal distribution for 4 random selected series.

Though we have standardized data with their standard deviation of all time, but the volatility forecast at the date just before event is not necessarily 1. Here we use 2 models for volatility forecasting. The first one is GARCH(1,1) model with window length equals 500 except Hang Seng in 1987(window size 80, only 80 points before event), FTSE 100 in 2008(window size 1000, when window size is 500 the GARCH fitting fails). The VaR for GARCH is calculated with percentile of normal distribution because normality is the default assumption of 'garch' function. For simplicity, we use unconditional volatility to approximate volatility forecast, as the forecasts will quickly converge to unconditional volatility. The second model we use is kernel estimate. Half exponential kernel with decaying parameter 0.93 is used for kernel estimate. We then use the last volatility estimate as our forecast. Percentile of kernel density estimate is used together to construct VaR. N-step VaR is simply $VaR_n = \sqrt{n}VaR_1 = \sqrt{n}\sigma_t\varepsilon_{(a)}$.

As we can see in Fig.6, Fig.7, Fig.8, Fig.9(black line stands for cumulative realized return, blue line stands for VaR of GARCH model, blue dotted line stands for VaR of kernel estimate), the VaR of GARCH model will underestimate the risk, except for Hang Seng and FTSE 100 in 1987. The crash in 1929 is the longest and largest one among the four. From the date of crash to max drawdown, it takes over 2 year and fall over 100 standard deviations, with log-close fall by 1.96. The max drawdown lie around on the 1% VaR of kernel estimate. The crash in 1987 has smaller drop than the 1929 one. What notable is that this time index dropped very quickly that the percentile is extremely small, which means that it's almost impossible to predict such a crash based on data before crash. The crash last for only few days and Hang Seng suffered the most from this crash. The magnitude of 2008 is larger than the 1987 one but the percentile of max drawdown is not that extreme than 1987. The effects on the 6 markets are similar, with a drop around 30 to 40 standard deviation. The crash last for around 1 to 2 months overall. As for the 2020, the most recent one behaved like 1987, where the index drop very quickly and last for short time. The percentile of max drawdown is therefore extreme, which means that such drops are almost impossible to estimate before the crash.

Therefore, we cannot conclude that the 2020 one is the largest one in history. For the magnitude(measured by max drawdown), the 1929 one is the largest. For the velocity of drop, this one as well as the 1987 one have the quickest drop in short time, which is very difficult to estimate before the event. The percentiles of max drawdown are 0(up to 4th decimal number) for markets in these two market crashes. However, as we only have limited data for the market swing this year, we have no idea whether the recent movement means that the swing has ended or it is the peace between aftershocks. It still can be the largest but by far it isn't. All these still remain unknown, only time will uncover the veil.

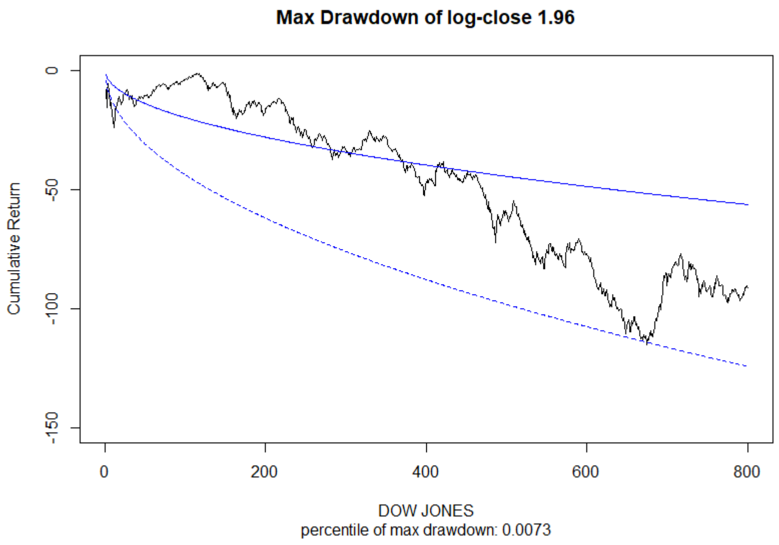


Figure 6: VaR vs. Cumulative Realized Returns-1929(Blue Filled GARCH, Blue Dotted Kernel Estimate

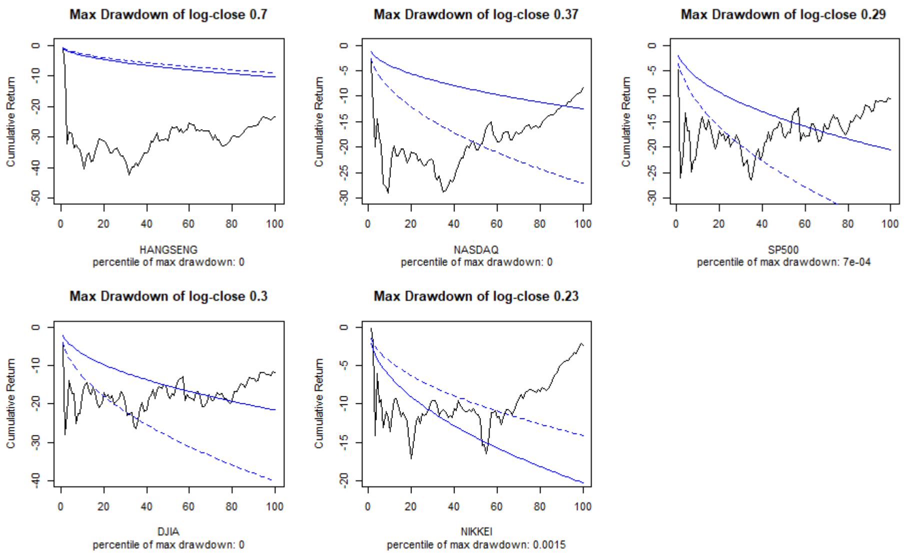


Figure 7: VaR vs. Cumulative Realized Returns-1987

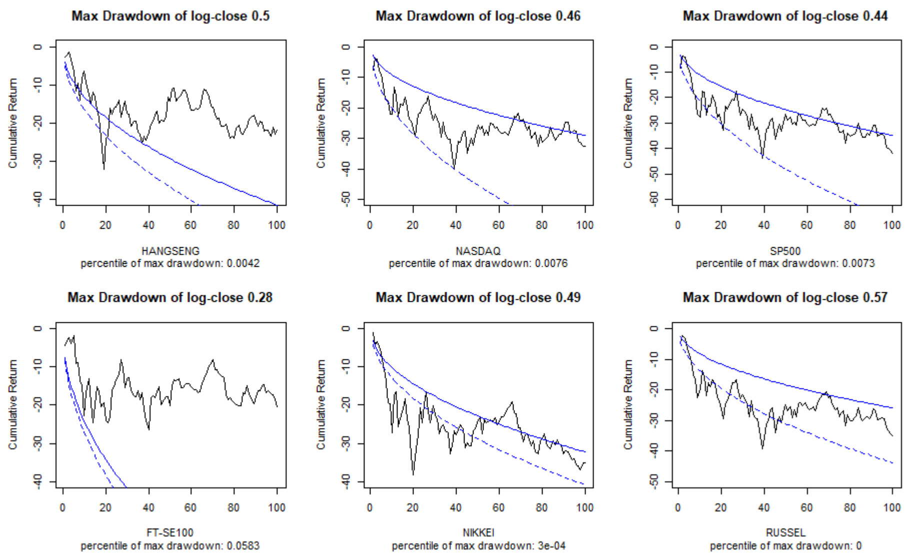


Figure 8: VaR vs. Cumulative Realized Returns-2008

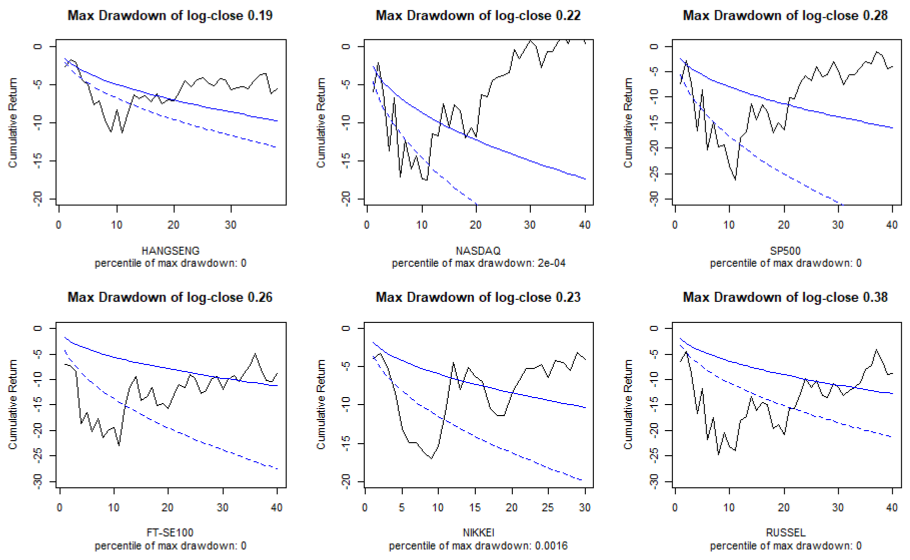


Figure 9: VaR vs. Cumulative Realized Returns-2020

Question Number	3	Word count	765

0.3 Q3

3. (a)-(d) See codes in Appendix.

3. (e)-(f) Data is available on https://github.com/wangmy22/Financial-Statistics-Project and is obtained by running 'pred.footsie.prepare' and passed through 'first.acf.square.train' in codes of seminar 7(with decaying parameter equals to 0.93, the same value as in Q2).

Data is divided into training set, validation set and test set with '50-25-25' separation. Here, we simply use these three data set as non-overlapping subsets. This is because for k-NN and CART, very early time series data(i.e. training set) will not have much predictive power on long future(i.e. validation and test set). Our alternative is to use prediction window so that prediction is based on very recent past. In general, a good model will have positive Sharpe ratio as well as be consistent over this three sunsets.

First we fit k nearest neighbours regression(k-NN regression) to make one day forward prediction. The selection of indexes is the same as 'Case Study' in lecture notes, which includes lags of FTSE 100 and other 10 indexes. The Sharpe ratio is averaged with k = 1,3,5,6,9,11. If Euclidean distance is used, the output by our defined k-NN regression function will be exactly the same as by 'knn.reg' in 'FNN' package(see Fig.10). Our own implementation of k-NN regression also enables us to use Manhattan distance as well as maximum distance. Window size of estimation is set to range from 40 to 250, with step length equals 10. The 'warmup' parameter is set equal to maximum of window size, that is, 250. This ensure that for each window size, the rolling regression function will run normally.

In Fig.10, the forecast only works well on training set. For validation set and test set, the Sharpe ratio never exceed 0 for any window size. As we can see in 11, this result is of no surprise and such magnitude is not significant at all.

To obtain the confidence interval, we simulate random standard normal distribution of size 1024*15 as our test set. Vector of length 1024(standard normal) is simulated as the response

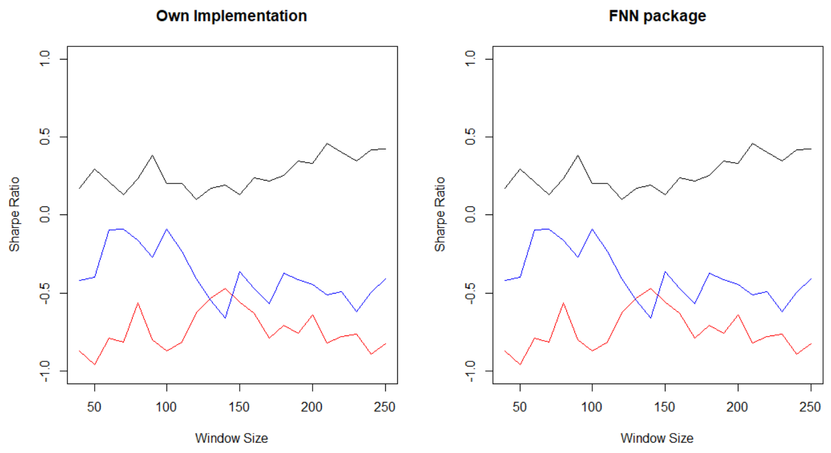


Figure 10: Sharpe Ratio of k-NN Regression, Black Training Set, Red Validation Set, Blue Test Set

variable. This is repeated for 1000 times. Then we can calculate the 5% and 95% confidence interval of Sharpe ratio. Different selections of indexes are also used for forecasting. Though the dimensions of simulation have changed with different selections, we simply use the one with all indexes because simulation is too computationally expensive. The result is shown in 11. Exclusion of non-american index will improve the performance of validation set. However, none of the selection have significant Sharpe ratio in all three subsets. K-NN regression shows no consistency with time for each of data selection. Therefore, we can not make easy money in financial market with k-NN regression, which is not a good model to capture the financial movements. The same conclusion holds for Manhattan distance(Fig.12) and Maximum Distance(Fig.13.

We can produce similar plots with CART rather than k-NN regression. The computation of nodes and leaves is based on the simplified version in lecture notes, where returns are transformed to binary variables(1 stands for positive returns and -1 stands for negative returns) and the node where tree are derived is the one has highest absolute correlation with response variable. The simplified algorithm is a bit different from algorithm in R packages because in R packages the nodes is specified with impurity function(e.g. Gini index or cross entropy), rather than absolute correlation. Similar to k-NN regression, we simulate data(500 times) to calculate confidence interval as well as the proportion that our algorithm provide same result as 'rpart' function in 'rpart' packages. As we can see in Fig.14, the proportion never fall below 80% for each window size.

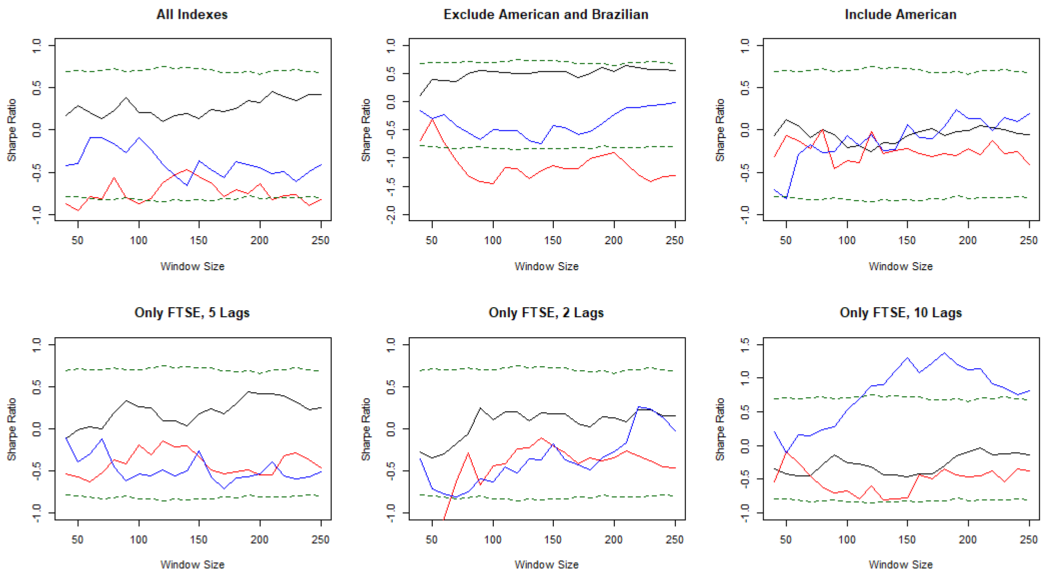


Figure 11: Sharpe Ratio of k-NN Regression, Black Training Set, Red Validation Set, Blue Test Set, Darkgreen Dotted Confidence Interval

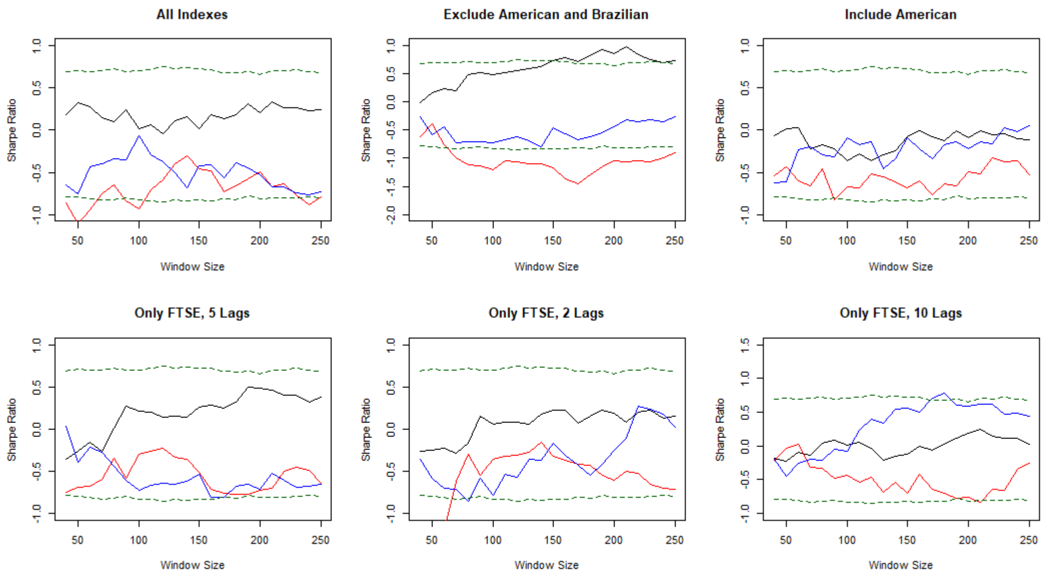


Figure 12: Sharpe Ratio of k-NN Regression with Manhattan Distance, Black Training Set, Red Validation Set, Blue Test Set, Darkgreen Dotted Confidence Interval

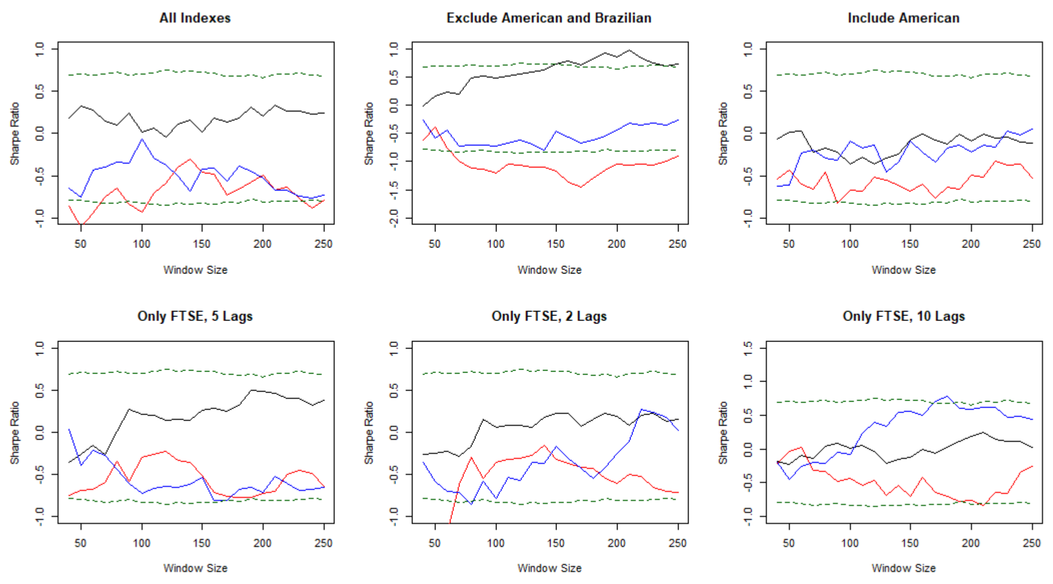


Figure 13: Sharpe Ratio of k-NN Regression with Maximum Distance, Black Training Set, Red Validation Set, Blue Test Set, Darkgreen Dotted Confidence Interval

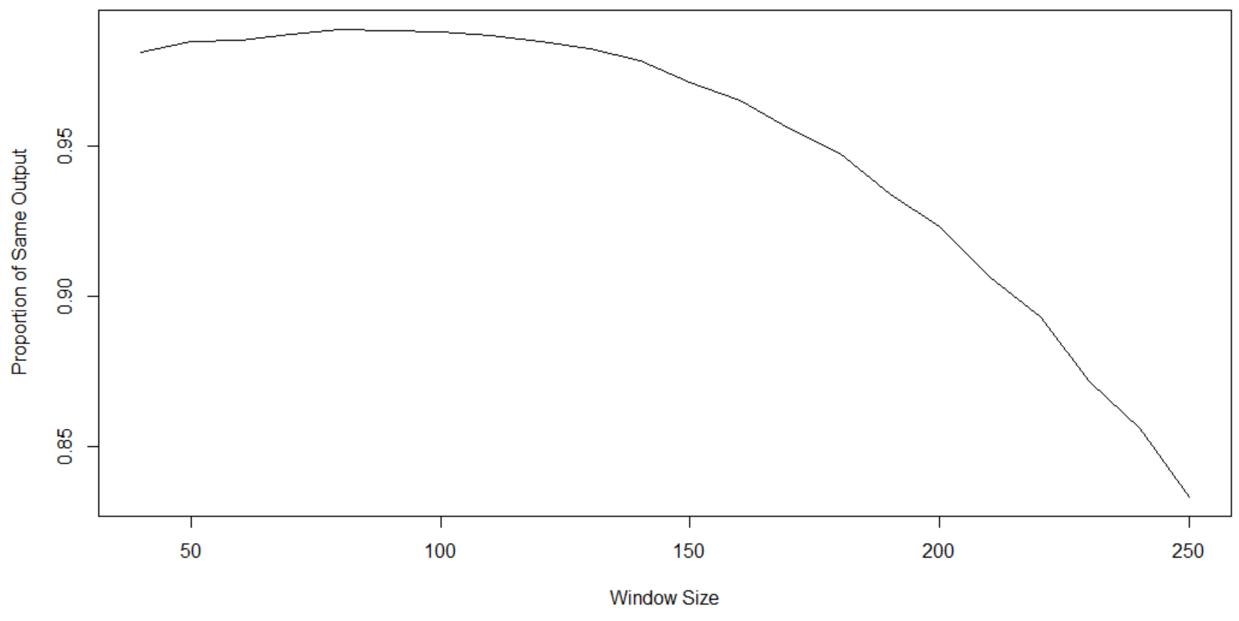


Figure 14: Proportion of Same Output between Simplified CART and R Package

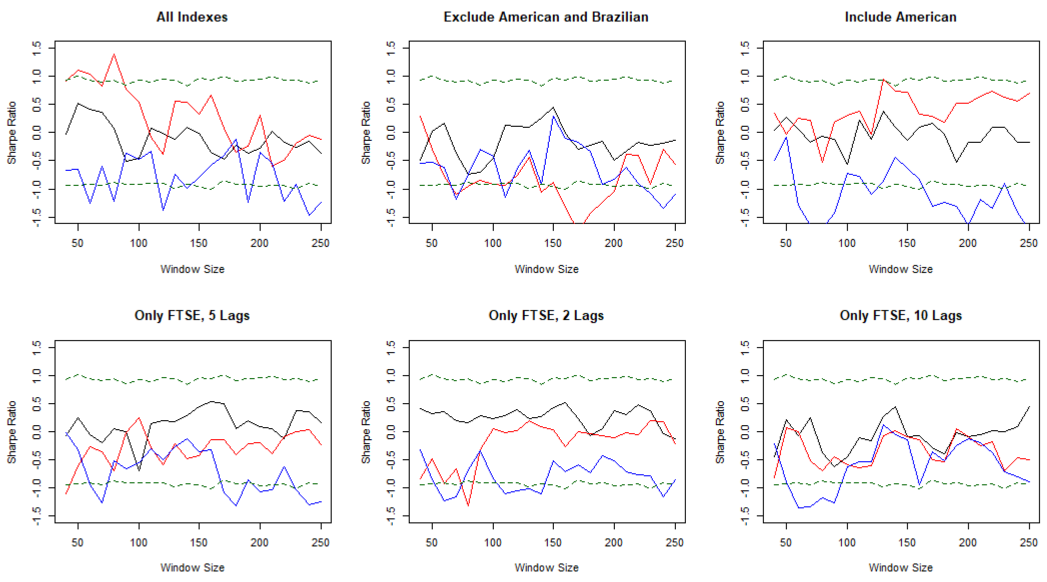


Figure 15: Sharpe Ratio of CART, Black Training Set, Red Validation Set, Blue Test Set, Darkgreen Dotted Confidence Interval

We then estimate Sharpe ratio for different window size as well as different selections of indexes. The result is shown in Fig.15. The result is similar to k-NN regression, with no selection of data successfully has positive Sharpe ratio over training set, validation set and test set. The Sharpe ratio is more sensitive to window size than k-NN regression does.

It's hard to tell what exactly cause negative Sharpe ratio because for most of the time negative Sharpe ratio falls above the 5% confidence interval. A possible cause is some indexes may dominate the prediction. If we calculate the correlation FTSE 100 index with its lags as well as other 10 indexes, we found that the correlation with its lags is not always the largest among all. For example, in training set, the correlation with its first lags is -0.03, but with Amex Index is 0.05. Similarly in validation set, it's 0.01 v.s. 0.05 with All Ordinaries Index. As we can see that after removing the 10 other indexes, the significant negative Sharpe ratios are controlled.

Reference

[1] Starica C. Is GARCH (1, 1) as good a model as the Nobel prize accolades would imply[J]. Preprint, 2003.

```
##############################################################################
# Code for Question 1
setwd('D:\\LSE\\ST436 Financial Stats\\pj\\data')
# define function to extract single bossa data, similar to course code, this is similar to course code but can extract data
      of full length
read.bossa.data.single <- function(vec.names,xlabel) {
  p <- length(vec.names);n1 <- 20000;dates <- matrix(99999999, p, n1);closes <- matrix(0, p, n1);max.n2 <- 0
  for (i in 1:p) {
    filename <- paste("bossa/mstzgr/",vec.names[i], ".mst", sep="")
    tmp <- scan(filename, list(NULL, date=numeric(), NULL, NULL, NULL, close=numeric(), NULL), skip=1, sep=",")
    n2 <- length(tmp$date); max.n2 <- max(n2, max.n2)
    dates[i,1:n2] <- tmp$date; closes[i,1:n2] <- tmp$close}
  dates <- dates[,1:max.n2]; closes <- closes[,1:max.n2];days <- rep(0, n1);arranged.closes <- matrix(0, p, n1)
  date.indices <- starting.indices <- rep(1, p);already.started <- rep(0, p);day <- 1
  while(max(date.indices) <= max.n2) {current.dates <- current.closes <- rep(0, p)
    for (i in 1:p) {current.dates[i] <- dates[date.indices[i]];current.closes[i] <- closes[date.indices[i]]}
    min.indices <- which(current.dates == min(current.dates));days[day] <- current.dates[min.indices[1]]
    arranged.closes[min.indices,day] <- log(current.closes[min.indices])
    arranged.closes[-min.indices,day] <- arranged.closes[-min.indices, max(day-1, 1)];already.started[min.indices] <- 1
    starting.indices[-which(already.started == 1)] <- starting.indices[-which(already.started == 1)] + 1
    day <- day + 1;date.indices[min.indices] <- date.indices[min.indices] + 1}
  days <- days[1:(day-1)];arranged.closes <- arranged.closes[,1:(day-1)]
  max.st.ind <- max(starting.indices); max.length <- day - max.st.ind - 1;r <- matrix(0, p, max.length)
  for (i in 1:p) {
    r[i,] <- diff(arranged.closes[(day-max.length-1):(day-1)]);r[i,] <- r[i,] / sqrt(var(r[i,]));r[i,r[i,]==0] <- rnorm(sum(r
          [i,]==0))}
  return(list(dates=dates, closes=closes, days=days, arranged.closes=arranged.closes, starting.indices=starting.indices, r=r)
       )}

# load needed packages
library(tseries);library(tidyverse);library(lubridate)
plot.mse.prop <- function(dta,xlabel){
# this function is to produce median SE proportion plot, with window size 1000 and bandwidth 250
  len <- length(dta);mse.prop.raw <- matrix(NA,floor((len-1250)/10),250) # 10 point switch of window
  for (i in (1:floor((len-1250)/10))){ # 1000 points for training data and 250 for estimate
    g <- garch(dta[(len-1249-10*(i-1)):(len-250-10*(i-1))]) # garch fit
    ke <- sum(dta[(len-499-10*(i-1)):(len-250-10*(i-1))]^2)/250 # kernel estimate
    pred.garch <- vector(); pred.garch.sum <- vector();r2.sum <- vector(); ke.sum <- vector()
    for (j in 1:250){
      # garch - k step forecast
      pred.garch[j] <- g$coef[1]*(1-(g$coef[2]+g$coef[3])^j)/(1-(g$coef[2]+g$coef[3]))+
        (g$coef[2]*dta[len-250-10*(i-1)]^2+g$coef[3]*g$fitted.values[1000])*(g$coef[2]+g$coef[3])^(j-1)
      pred.garch.sum[j] <- sum(pred.garch[1:j]) # aggregated from GARCH
      r2.sum[j] <- sum(dta[(len-249-10*(i-1)):(len-249-10*(i-1)+j)]^2) # aggregated realized volatility
      ke.sum[j] <- j*ke}# aggregated from kernel estimate
    mse.prop.raw[i,] <- ((r2.sum-pred.garch.sum)^2)/((r2.sum-ke.sum)^2)}# calculate SE proportion
  # plot the median proportion vs steps and apply kernel smoothing
  mse.prop.median <- apply(mse.prop.raw,2,median,na.rm=T)
  plot(mse.prop.median,type='l',xlab=xlabel,ylab='SE(GARCH)/SE(KERNEL)')
  points(ksmooth(1:250,mse.prop.median,bandwidth = 30)$y,type='l',col='red')}
# produce Figure 1
par(mfrow=c(2,3))
plot.mse.prop(t(read.bossa.data.single('NASDAQ')$r),xlabel='NASDAQ');plot.mse.prop(t(read.bossa.data.single('SP500')$r),
      xlabel='SP500')
plot.mse.prop(t(read.bossa.data.single('FT-SE100')$r),xlabel='FT-SE100');plot.mse.prop(t(read.bossa.data.single('HANGSENG')$r
      ),xlabel='HANGSENG')
# read in EUR/USD exchange rate data and plot
eu <- read.csv('EURUSD.csv');r <- diff(eu$Rate)/sd(diff(eu$Rate));plot.mse.prop(r,xlabel='EUR/USD')
# read in boeing exchange rate data and plot
boeing <- read.csv('boeing.csv');r <- diff(log(boeing$PRC))/sd(diff(log(boeing$PRC)));plot.mse.prop(r,xlabel='BOEING')

# compare performance of models in 2020 by VaR
compare.plot <- function(vec.names,window=1000,startdate='2020-01-02',band=250,ylim){
  r <- t(read.bossa.data.single(vec.names)$r)
  date <- as.Date(as.character(read.bossa.data.single(vec.names)$date),format='%Y%m%d')
  index <- which(date==startdate)
  # test set and training set seperation
  test <- r[(index-1):length(r)]; train <- r[(index-window-1):(index-2)]
  g <- garch(train) # garch fit
  ke <- sum(train[(window-band+1):window]^2)/band # kernel estimate
  pred.garch <- vector();pred.garch.sum <- vector();r2.sum <- vector();ke.sum <- vector()
  for (i in 1:(length(r)-index+2)){
    pred.garch[i] <- g$coef[1]*(1-(g$coef[2]+g$coef[3])^i)/(1-(g$coef[2]+g$coef[3]))+
      (g$coef[2]*train[length(train)]^2+g$coef[3]*g$fitted.values[length(train),1])*(g$coef[2]+g$coef[3])^(i-1)
    pred.garch.sum[i] <- sum(pred.garch[1:i])
    r2.sum[i] <- sum(test[1:i]^2); ke.sum[i] <- i*ke}
  plot(date[index:length(date)],r2.sum,type='l',ylab='cumulative realized squared return',xlab=vec.names,ylim=ylim)
  points(date[index:length(date)],ke.sum,type='l',col='red',lty=3)
  points(date[index:length(date)],pred.garch.sum,type='l',col='blue',lty=3)}

# compare VaR of different index
par(mfrow=c(2,2))
compare.plot('NASDAQ',ylim=c(0,700));compare.plot('SP500',ylim=c(0,900));compare.plot('FT-SE100',ylim=c(0,500));compare.plot(
      'HANGSENG',ylim=c(0,100))
# with different startdate, window size and bandwidth
compare.plot('NASDAQ',startdate = '2020-03-06',window=400,band=100,ylim=c(0,700));compare.plot('SP500',startdate = '
      2020-03-06',window=400,band=100,ylim=c(0,900))
compare.plot('FT-SE100',startdate = '2020-03-06',window=400,band=100,ylim=c(0,500));compare.plot('HANGSENG',startdate = '
      2020-03-06',window=400,band=100,ylim=c(0,100))
# with different startdate, window size and bandwidth
compare.plot('NASDAQ',startdate = '2020-03-13',window=1000,band=100,ylim=c(0,500));compare.plot('SP500',startdate = '
      2020-03-13',window=1000,band=100,ylim=c(0,600))
compare.plot('FT-SE100',startdate = '2020-03-13',window=1000,band=100,ylim=c(0,400));compare.plot('HANGSENG',startdate = '
      2020-03-13',window=1000,band=100,ylim=c(0,100))

########################################
## code for Q2
library(tidyverse);library(lubridate)
# calculate loglikelihood for given distribution, sigma denotes the bandwidth
minimise_function <- function(sigm, train,test){
  temp <- density(train,bw=sigm,from=-30,to=30,n=4096)
  yest <- approx(temp$x, temp$y, xout = test)$y;loglike <- sum(log(yest))
  return(-loglike)}
# calculate cumulative distribution function
cdf <- function(x,f){integrate(f, -30, x)$value}
# calculate percentile from cdf
invcdf <- function(q,r.before.stand,f){
  uniroot(function(x){cdf(x,f) - q}, range(r.before.stand))$root}
# Exponential smoothing of x^2 with parameter lambda, this is similar to course code
vol.exp.sm <- function(x, lambda) {
  sigma2 <- x^2; n <- length(x)
  for (i in 2:n) sigma2[i] <- sigma2[i-1] * lambda + x[i]^2 * (1-lambda)
  list(sigma2=sigma2)}
# plot VaR against realized return
plot.var <- function(vec.name,start.date,mid.date,end.date,qtile,plotlen,ylim=c(-100,0),garchlen=500,bossa=T,compare.pdf=F){
  if (bossa==T){ # read in bossa data and perform data seperation
    temp.index <- read.bossa.data.single(vec.name)$arranged.closes
    temp <- t(read.bossa.data.single(vec.name)$r)
    temp.date <- as.Date(as.character(read.bossa.data.single(vec.name)$date),format='%Y%m%d')[-1]
    r.before <- temp[which(temp.date>start.date&temp.date<=mid.date)]
    r.after <- temp[which(temp.date>mid.date&temp.date<=end.date)]
    index.after <- temp.index[which(temp.date>=mid.date&temp.date<=end.date)][-1]}
  else { # read in dow jones industrial 1929 and perform data seperation
    dow <- read.csv('dowjones.csv')
    dow$date <- as.Date(dow$date,format='%d%b%Y');dow$r <- c(NA,diff(log(dow$dji)))
    dow <- dow[-which(is.na(dow$r)),]
    dow.1929 <- dow[which(dow$date>='1920-01-03'&dow$date<='1935-12-31'),]
    dow.1929$r <- dow.1929$r/(sd(dow.1929$r)) ;index.temp <- dow.1929$dji
    index <- which(dow.1929$date>='1920-01-03'&dow.1929$date<='1929-10-25')
    r.before <- dow.1929$r[index];r.after <- dow.1929$r[-index]
    index.after <- log(index.temp[-index])}
  # kernel estimate for most recent volatility
  ke <- sqrt(vol.exp.sm(r.before, 0.93)$sigma2[length(r.before)])
  # garch model fit and extract unconditional volatility
  g <- garch(r.before[(length(r.before)-garchlen+1):length(r.before)])
  ge <- sqrt(g$coef[1]/(1-g$coef[2]-g$coef[3]))
  # standarize the return data, so that it's exactly measured as standard deviation
  n <- length(r.before)
  r.before.stand <- r.before/sd(r.before)
  # scan through the best bandwidth and refit with best parameter
  w <- optimise(minimise_function,interval=c(0,1),tol=0.001,r.before.stand[1:(n/3)],r.before.stand[(n/3+1):(2*n/3)])$minimum
  d <- density(r.before.stand[1:(n/3)],bw=w,from=-30,to=30,n=4096)
  f <- approxfun(d$x,d$y)
  # extract percentile and calculate proportion of data under percentile
  per <- invcdf(qtile,r.before.stand,f)
  prop.kernel <- vector();prop.normal <- vector()
  prop.kernel[1] <- mean(r.before.stand[(2*n/3+1):n] < invcdf(.05,r.before.stand,f))
  prop.normal[1] <- mean(r.before.stand[(2*n/3+1):n] < qnorm(.05))
  prop.normal[2] <- mean(r.before.stand[(2*n/3+1):n] < qnorm(.01))
  if (qtile==.01) prop.kernel[2] <- mean(r.before.stand[(2*n/3+1):n] < invcdf(.01,r.before.stand,f))
  # calculate max drawdown, cumulative return and percentile of max drawdown
  maxd <- maxdrawdown(index.after[1:plotlen])
  # calculate cumulative loss after event
  r.after.sum <- cumsum(r.after)
  # calculate percentile of max drawdown
  mintile <- cdf((r.after.sum[maxd$to-1]+r.after.sum[1])/(sqrt(maxd$to)*ke),f)
  # produce var vs. realized plot
  if (compare.pdf==F){
    plot(r.after.sum[1:plotlen],type='l',ylim=ylim,main=paste('Max Drawdown of log-close',round(maxd$maxdrawdown,2))
         ,xlab=vec.name,ylab = 'Cumulative Return',sub=paste('percentile of max drawdown:',round(mintile,4)))
    lines(qnorm(qtile)*ge*sqrt(1:plotlen),col='blue')
    lines(per*ke*sqrt(1:plotlen),col='blue',lty=2)}
  # produce pdf comparison plot
  else if (compare.pdf==T){
    plot(approx(d$x,d$y,xout=(-300:300)/100),type='l',xlab = 'x',ylab='Density',main=paste(vec.name,'.',year(mid.date),sep=''
         ))
    points(((-300:300)/100),dnorm(-300:300/100),type = 'l',col='blue')}
  return(list(mintile,prop.kernel,prop.normal))}
# produce VaR vs. realized, if error occur, please have more trys
# Figure 6
out18 <- plot.var('DOW JONES','1920-01-03','1929-10-25','1935-12-31', .01,800,ylim=c(-150,0),bossa=F)
par(mfrow=c(2,3))
# Figure 7
out1 <- plot.var('HANGSENG','1978-01-01','1987-10-15','1993-01-01', .05,100,ylim=c(-50,0),garchlen = 80)
out2 <- plot.var('NASDAQ','1978-01-01','1987-10-15','1993-01-01', .01,100,ylim=c(-30,0))
out3 <- plot.var('SP500','1978-01-01','1987-10-15','1993-01-01', .01,100,ylim=c(-30,0))
out4 <- plot.var('DJIA','1978-01-01','1987-10-15','1993-01-01', .01,100,ylim=c(-40,0))
out5 <- plot.var('NIKKEI','1978-01-01','1987-10-15','1993-01-01', .01,100,ylim=c(-20,0))
plot.new()
# Figure 8
out6 <- plot.var('HANGSENG','2000-01-01','2008-09-28','2013-01-01', .01,100,ylim=c(-40,0))
out7 <- plot.var('NASDAQ','2000-01-01','2008-09-28','2013-01-01', .01,100,ylim=c(-50,0))
out8 <- plot.var('SP500','2000-01-01','2008-09-28','2013-01-01', .01,100,ylim=c(-60,0))
out9 <- plot.var('FT-SE100','2000-01-01','2008-09-28','2013-01-01', .01,100,ylim=c(-40,0),garchlen = 1000)
out10 <- plot.var('NIKKEI','2000-01-01','2008-09-28','2013-01-01', .01,100,ylim=c(-40,0))
out11 <- plot.var('RUSSEL','2000-01-01','2008-09-28','2013-01-01', .01,100,ylim=c(-50,0))
# Figure 9
out12 <- plot.var('HANGSENG','2012-01-01','2020-03-06','2020-05-08', .01,38,ylim=c(-20,0))
out13 <- plot.var('NASDAQ','2012-01-01','2020-03-06','2020-05-08', .01,40,ylim=c(-20,0))
out14 <- plot.var('SP500','2012-01-01','2020-03-06','2020-05-08', .01,40,ylim=c(-30,0))
out15 <- plot.var('FT-SE100','2012-01-01','2020-03-06','2020-05-08', .01,40,ylim=c(-30,0))
out16 <- plot.var('NIKKEI','2012-01-01','2020-03-06','2020-05-08', .01,30,ylim=c(-20,0))
out17 <- plot.var('RUSSEL','2012-01-01','2020-03-06','2020-05-08', .01,40,ylim=c(-30,0))

# extract percentile in each fit
out.plot <- matrix(NA,18,4);qtile.vec <- vector()
for (i in 1:18){
  out.plot[i,1:2] <- get(paste('out',i,sep=''))[[2]];out.plot[i,3:4] <- get(paste('out',i,sep=''))[[3]]
  qtile.vec[i] <- get(paste('out',i,sep=''))[[1]]}
out.plot[1,2] <- NA;out.plot.temp = as.data.frame(out.plot[,c(2,4)])
colnames(out.plot.temp)=c('Kernel Estimate','Normal');boxplot(out.plot.temp)
par(mfrow=c(2,2)) # produce pdf comparison, figure 5
plot.var('DOW JONES','1920-01-03','1929-10-25','1935-12-31', .01,800,ylim=c(-150,0),bossa=F,compare.pdf = T)
plot.var('FT-SE100','2012-01-01','2020-03-06','2020-05-08', .01,40,ylim=c(-30,0),compare.pdf = T)
plot.var('NASDAQ','2000-01-01','2008-09-28','2013-01-01', .01,200,ylim=c(-50,0),compare.pdf = T)
plot.var('SP500','1978-01-01','1987-10-15','1993-01-01', .01,100,ylim=c(-30,0),compare.pdf = T)

############################################
## code for Q3
library(FNN) # package for k-NN
library(rpart) # package for CART

# this function performs kNN regression, k is the number of neighbours
# q=2,Euclidean distance, q=1, Manhattan distance; q large, maximum distance
knn.reg.own <- function(x.train,y.train,pred.x,q=2,k=5){
  distmat <- abs(sweep(x.train,2,pred.x)) # calculate distance
  distvec <- rowSums(distmat^q)^(1/q) # sum up distance
  index.sort <- sort(distvec,index.return=T)$ix # get index with smallest distance
  ypred <- mean(y.train[head(index.sort,k)]) # make prediction
  return(ypred)}
```

```
# moving the window and average the prediction error
rolling.knn <- function(dtax,dtay,win,warmup,k,q,own=T){
  nk <- length(k);n <- length(dtay);predi <- truth <- rep(0, n-warmup);err <- vector()
  for (j in 1:nk){# go through all k and average
    for (i in (warmup+1):n) {# extract data by window
      y <- dtay[(i-win):(i-1)];x <- dtax[(i-win):(i-1),];xpred <- dtax[i,]
      if (own==T) predi[i-warmup] <- knn.reg.own(x, y,xpred,k=k[j],q=q) # use own k-NN function
      else if (own==F) predi[i-warmup] <- knn.reg(x,xpred,y,k=k[j])$pred # use function from 'FNN' package
      truth[i-warmup] <- dtay[i]}
    ret <- as.numeric(predi) * truth;err[j] <- sqrt(250) * mean(ret) / sqrt(var(ret))}
  return(list(predi,mean(err)))}
# this function performs simplified CART
cart.own <- function(x.train,y.train,pred.x){
  x.train <- ifelse(x.train<0,-1,1) # convert numeric to binary
  y.train <- ifelse(y.train<0,-1,1) # convert numeric to binary
  xpred <- ifelse(pred.x<0,-1,1) # convert numeric to binary
  maxindex <- which.max(abs(apply(x.train,2,cor,y.train))) # extract variable with highest correlation
  ypred <- median(y.train[x.train[,maxindex]==xpred[maxindex]]) # make prediction
  # make prediction with 'rpart' package
  tr <- rpart(y.train~x.train,control=rpart.control(minsplit = 2,maxdepth = 1))
  # extract position of node, if the position equal the position estimated by our own CART function
  # these two methods produce the same output
  nodename <- as.character(tr$frame$var)[1]
  equal <- (nodename==paste('x.train',maxindex,sep = '')) # compare with simplified CART function
  return(list(ypred=ypred,equal=equal))}
# moving the window and average the prediction error
rolling.cart <- function(dtax,dtay,win,warmup){
  n <- length(dtay);predi <- truth <- equal <- rep(0, n-warmup);err <- vector()
  for (i in (warmup+1):n) {
    y <- dtay[(i-win):(i-1)];x <- dtax[(i-win):(i-1),]# subtract data by window
    xpred <- dtax[i,];temp <- cart.own(x, y,xpred) # CART for prediction
    predi[i-warmup] <- temp$ypred;equal[i-warmup] <- temp$equal;truth[i-warmup] <- dtay[i]}
  ret <- as.numeric(predi) * truth;err <- sqrt(250) * mean(ret) / sqrt(var(ret))
  return(list(predi,err,(sum(equal)/length(equal))))}
# calculate sharpe ratio
sharpe <- function(data,win,warmup,own=T,k=seq(1,11,by=2),q=2,tree=F){
  w <- length(win)
  train.curve <- valid.curve <- test.curve <- rep(0, w)
  if (tree == F){ # use k-NN regression
    for (i in 1:w) {
      train.curve[i] <- rolling.knn(data$x.train.dev,data$y.train.dev,win[i],warmup,k=k,q=q,own=own)[[2]]
      valid.curve[i] <- rolling.knn(data$x.valid.dev,data$y.valid.dev,win[i],warmup,k=k,q=q,own=own)[[2]]
      test.curve[i] <- rolling.knn(data$x.test.dev,data$y.test.dev,win[i],warmup,k=k,q=q,own=own)[[2]]}
  } else if (tree == T){ # use CART
    for (i in 1:w) {
      train.curve[i] <- rolling.cart(data$x.train.dev,data$y.train.dev,win[i],warmup)[[2]]
      valid.curve[i] <- rolling.cart(data$x.valid.dev,data$y.valid.dev,win[i],warmup)[[2]]
      test.curve[i] <- rolling.cart(data$x.test.dev,data$y.test.dev,win[i],warmup)[[2]]}}
  return(list(train.curve=train.curve,valid.curve=valid.curve,test.curve=test.curve))}

# plot three curve as well as CI
plot.sharpe <- function(x,quantile1,quantile2,ylim=c(-1,1),main){
  plot((4:25)*10,x$train.curve,type='l',ylim=ylim,ylab = 'Sharpe Ratio',xlab='Window Size',main=main)
  points((4:25)*10,x$valid.curve,type='l',col='red');points((4:25)*10,x$test.curve,type='l',col='blue')
  points((4:25)*10,quantile1,type='l',col='darkgreen',lty=2);points((4:25)*10,quantile2,type='l',col='darkgreen',lty=2)}
```

```
# load in data we need for estimation, uncomment the following lines and run with course codes
# dta.part <- pred.footsie.prepare(mask = rep(0, 10))
# dta.part <- first.acf.squares.train(dta.part,0.93)
# dta.part.exclude.america <- pred.footsie.prepare(mask = c(1,0,0,1,1,0,1,0,1,0))
# dta.part.exclude.america <- first.acf.squares.train(dta.part.exclude.america,0.93)
# dta.part.include.america <- pred.footsie.prepare(mask = c(0,0,0,0,0,1,0,1,0,1))
# dta.part.include.america <- first.acf.squares.train(dta.part.include.america,0.93)
# dta.part.lag2 <- pred.footsie.prepare(max.lag=2,mask = rep(0, 10))
# dta.part.lag2 <- first.acf.squares.train(dta.part.lag2,0.93)
# dta.part.lag10 <- pred.footsie.prepare(max.lag=10,mask = rep(0, 10))
# dta.part.lag10 <- first.acf.squares.train(dta.part.lag10,0.93)
# or we can load data from github, they are exactly the same
load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/dtafull.RData?raw=true'))
load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/dtapart.RData?raw=true'))
load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/dtapartlag10.RData?raw=true'))
load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/dtapartlag2.RData?raw=true'))
load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/dtaincludeamerica.RData?raw=true'))
load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/dtaexcludeamerica.RData?raw=true'))

# calculate confidence interval of k-NN, this is extremely time costing and extract quantile
win <- seq(40,1000,by=10);w <- length(win)
curve.random <- matrix(NA,1000,w)
for (j in 1:1000){
  x1 <- matrix(rnorm(1024*15),1024,15);y1 <- rnorm(1024)
  for (i in 1:w) {curve.random[j,i] <- rolling.knn(x1,y1,win[i],250,k=seq(1,11,by=2),q=2,own=T)[[2]]}}
for (i in 1:w){quantile1[i] <- quantile(curve.random[,i],.05);quantile2[i] <- quantile(curve.random[,i],.95)}
# or load the simulation data from github, uncomment the following line and run
# load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/curve1.RData?raw=true'))
quantile1 <- vector();quantile2 <- vector()
for (i in 1:w){quantile1[i] <- quantile(curve.random[,i],.05);quantile2[i] <- quantile(curve.random[,i],.95)}
# compare own written function with package
par(mfrow=c(1,2))
plot((4:25)*10,x1$train.curve,type='l',ylim=c(-1,1),ylab = 'Sharpe Ratio',xlab='Window Size',main='Own Implementation')
points((4:25)*10,x1$valid.curve,type='l',col='red');points((4:25)*10,x1$test.curve,type='l',col='blue')
plot((4:25)*10,x1.1$train.curve,type='l',ylim=c(-1,1),ylab = 'Sharpe Ratio',xlab='Window Size',main='FNN package')
points((4:25)*10,x1.1$valid.curve,type='l',col='red');points((4:25)*10,x1.1$test.curve,type='l',col='blue')

# plot sharpe rate with k-NN, euclidean distance
x1 <- sharpe(dta.full,win=seq(40,250,by=10),warmup = 250)
x2 <- sharpe(dta.part.exclude.america,win=seq(40,250,by=10),warmup = 250)
x3 <- sharpe(dta.part.include.america,win=seq(40,250,by=10),warmup = 250)
x4 <- sharpe(dta.part,win=seq(40,250,by=10),warmup = 250)
x5 <- sharpe(dta.part.lag2,win=seq(40,250,by=10),warmup = 250)
x6 <- sharpe(dta.part.lag10,win=seq(40,250,by=10),warmup = 250)
par(mfrow=c(2,3))
plot.sharpe(x1,quantile1,quantile2,main='All Indexes')
plot.sharpe(x2,quantile1,quantile2,main='Exclude American and Brazilian',ylim=c(-2,1))
plot.sharpe(x3,quantile1,quantile2,main='Include American',ylim=c(-1,1))
plot.sharpe(x4,quantile1,quantile2,main='Only FTSE, 5 Lags',ylim=c(-1,1))
plot.sharpe(x5,quantile1,quantile2,main='Only FTSE, 2 Lags',ylim=c(-1,1))
plot.sharpe(x6,quantile1,quantile2,main='Only FTSE, 10 Lags',ylim=c(-1,1.5))
# plot sharpe rate with k-NN, manhattan distance
x1 <- sharpe(dta.full,win=seq(40,250,by=10),warmup = 250,q=1)
x2 <- sharpe(dta.part.exclude.america,win=seq(40,250,by=10),warmup = 250,q=1)
```

```r
x3 <- sharpe(dta.part.include.america,win=seq(40,250,by=10),warmup = 250,q=1)
x4 <- sharpe(dta.part,win=seq(40,250,by=10),warmup = 250,q=1)
x5 <- sharpe(dta.part.lag2,win=seq(40,250,by=10),warmup = 250,q=1)
x6 <- sharpe(dta.part.lag10,win=seq(40,250,by=10),warmup = 250,q=1)
plot.sharpe(x1,quantile1,quantile2,main='All Indexes')
plot.sharpe(x2,quantile1,quantile2,main='Exclude American and Brazilian',ylim=c(-2,1))
plot.sharpe(x3,quantile1,quantile2,main='Include American',ylim=c(-1,1))
plot.sharpe(x4,quantile1,quantile2,main='Only FTSE, 5 Lags',ylim=c(-1,1))
plot.sharpe(x5,quantile1,quantile2,main='Only FTSE, 2 Lags',ylim=c(-1,1))
plot.sharpe(x6,quantile1,quantile2,main='Only FTSE, 10 Lags',ylim=c(-1,1.5))
# plot sharpe rate with k-NN, maximum distance
x1 <- sharpe(dta.full,win=seq(40,250,by=10),warmup = 250,q=100)
x2 <- sharpe(dta.part.exclude.america,win=seq(40,250,by=10),warmup = 250,q=100)
x3 <- sharpe(dta.part.include.america,win=seq(40,250,by=10),warmup = 250,q=100)
x4 <- sharpe(dta.part,win=seq(40,250,by=10),warmup = 250,q=100)
x5 <- sharpe(dta.part.lag2,win=seq(40,250,by=10),warmup = 250,q=100)
x6 <- sharpe(dta.part.lag10,win=seq(40,250,by=10),warmup = 250,q=100)
plot.sharpe(x1,quantile1,quantile2,main='All Indexes')
plot.sharpe(x2,quantile1,quantile2,main='Exclude American and Brazilian',ylim=c(-2,1))
plot.sharpe(x3,quantile1,quantile2,main='Include American',ylim=c(-1,1))
plot.sharpe(x4,quantile1,quantile2,main='Only FTSE, 5 Lags',ylim=c(-1,1))
plot.sharpe(x5,quantile1,quantile2,main='Only FTSE, 2 Lags',ylim=c(-1,1))
plot.sharpe(x6,quantile1,quantile2,main='Only FTSE, 10 Lags',ylim=c(-1,1.5))

# calculate confidence interval of CART and extract quantile
win <- seq(40,250,by=10);w <- length(win);curve.random <- matrix(NA,500,w);equal.random <- matrix(NA,500,w)
for (j in 1:500){
  x1 <- matrix(rnorm(1024*15),1024,15);y1 <- rnorm(1024)
  for (i in 1:w) {temp <- rolling.cart(x1,y1,win[i],250);curve.random[j,i] <- temp[[2]];equal.random[j,i] <- temp[[3]]}}
# or load from github, uncomment and run
# load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/curve2.RData?raw=true'))
# load(url('https://github.com/wangmy22/Financial-Statistics-Project/blob/master/equal.RData?raw=true'))
quantile1 <- vector();quantile2 <- vector() # extract quantile
for (i in 1:w){quantile1[i] <- quantile(curve.random[,i],.05);quantile2[i] <- quantile(curve.random[,i],.95)}
# plot proportion of same output of two function
par(mfrow=c(1,1));plot((4:25)*10,apply(equal.random,2,mean),xlab='Window Size',ylab='Proportion of Same Output',type='l')
# plot sharpe rate with CART
x1 <- sharpe(dta.full,win=seq(40,250,by=10),warmup = 250,tree = T)
x2 <- sharpe(dta.part.exclude.america,win=seq(40,250,by=10),warmup = 250,tree = T)
x3 <- sharpe(dta.part.include.america,win=seq(40,250,by=10),warmup = 250,tree = T)
x4 <- sharpe(dta.part,win=seq(40,250,by=10),warmup = 250,tree = T)
x5 <- sharpe(dta.part.lag2,win=seq(40,250,by=10),warmup = 250,tree = T)
x6 <- sharpe(dta.part.lag10,win=seq(40,250,by=10),warmup = 250,tree = T)
par(mfrow=c(2,3));plot.sharpe(x1,quantile1,quantile2,main='All Indexes',ylim=c(-1.5,1.5))
plot.sharpe(x2,quantile1,quantile2,main='Exclude American and Brazilian',ylim=c(-1.5,1.5))
plot.sharpe(x3,quantile1,quantile2,main='Include American',ylim=c(-1.5,1.5))
plot.sharpe(x4,quantile1,quantile2,main='Only FTSE, 5 Lags',ylim=c(-1.5,1.5))
plot.sharpe(x5,quantile1,quantile2,main='Only FTSE, 2 Lags',ylim=c(-1.5,1.5))
plot.sharpe(x6,quantile1,quantile2,main='Only FTSE, 10 Lags',ylim=c(-1.5,1.5))

# extract data and calculate correlation
ftse.train <- dta.full$y.train.dev;ftse.valid <- dta.full$y.valid.dev
ftselag1.train <- dta.full$x.train.dev[,1];ftselag1.valid <- dta.full$x.valid.dev[,1]
amex.train <- dta.full$x.train.dev[,7];allord.valid <- dta.full$x.valid.dev[,6]
cor(ftse.train,ftselag1.train);cor(ftse.train,amex.train);cor(ftse.valid,ftselag1.valid);cor(ftse.valid,allord.valid)
```