ST429

GROUP PROJECT

Candidate Number	Contribution Rate
11361	25%
15504	25%
16996	25%
19408	25%

January 2019

Contents

Executive Summary 2

Part a. Stock Analysis 3

Systematic event 5

Industry event 5

Firm-specific event 5

Part b & c. Portfolio Construction 5

Methodology 5

Step I: Estimate the Volatility 5

Step II: Estimate the Return 6

Step III: Calculate Weights of Capital Invested in Each Stock 7

Step IV: Calculate Total Wealth at the End of Each Node 8

Findings 9

Performance Comparisons Between Three Strategies with Different Rebalance Period 9

Performance Comparisons Between Three Strategies and S&P 500 9

Risk of Portfolios 10

Part d & e. Mini Index Return & Risk Analysis 13

Mini Index Construction 13

Fitting copulas to data 14

Return Level and Return Period 15

Fitting GEV Distribution 16

Calculating Return Level and Return Period 17

Bibliography 19

Appendix 19

I. Generalized Autoregressive Conditional Heteroskedasticity (GARCH) 19

II. Coefficients of AR(1) Model for Daily and Monthly Rebalanced Portfolio 20

III. Weights for Different Strategies of Daily and Monthly Rebalanced Portfolios 21

IV. Total Capital of Momentum/ Value/ Dynamic Strategy 22

V. 99% VaR and 97.5% ES of Momentum/ Value/ Dynamic Strategy 23

R Code 23

Executive Summary

The report provides a detailed analysis on the performance and risk of five technology stocks – Apple, Amazon, Google, Microsoft, and Netflix. Statistics shows that total market capital of the five stocks account for over 40% of Nasdaq and their performance dominated S&P500 over the past decade. However, they also subject to high risk exposure. The main purpose of the report is to construct portfolios and indices based on different strategies to balance their growth prospect and volatility.

Part a of the report displays some statistical and graphical summary of the five technology stocks as well as S&P500. Key statistics such as correlation, range, mean, volatility, skewness, kurtosis and growth rate could be found in this section. The latter part also discusses some special events that led to significant stock drawdown during the past decade.

Part b and c start to construct portfolios based on different trading strategies. Part b assumes all stocks follow either momentum or value strategy. Part c (so-called "dynamic strategy") loose this assumption and treat different stocks with different strategies in different periods. The report also tests three different rebalance period – daily, weekly and monthly, to test whether frequent rebalance will have effects on the portfolio performance. The latter part of this section explores the risk profile such as Value at Risk (VaR) and Expected Shortfall (ES) of these portfolios. The result shows that while the performance of all these portfolios beats S&P500 by a large extent, they are also exposed to very high potential loss compared with S&P500.

Part d and e create a value-weighted mini index based on the log return of the five technology stocks. Part d analyses the dependence between the mini index and S&P500 using copula. It is concluded that the performance of mini index is highly correlated with that of S&P 500 and t copula provides the best fitting. Part e builds a General Extreme Value (GEV) model based on some significant drawdowns of the mini index during the past 12 years. The final part of this section provides a deep study on the return level and return period based on both quarterly and monthly block maxima method.

Part a. Stock Analysis

This part provides some basic statistical and graphical summary of the performance for five technology stocks – Apple (AAPL), Amazon (AMZN), Google (GOOG), Microsoft (MSFT) and Netflix (NFLX), as well as the S&P 500 Index between 2006 Jan 1st and 2018 Nov 2nd.

It is shown in Table 1 that the range of the daily log return is quite different between each stock. Among the five technology stocks, Netflix has the largest range (78.14%), which is consistent with the highest volatility, while Microsoft has the most stable log returns. All five technology stocks are much more volatile than S&P 500.

The moment statistics describe the shape of the log return. Apple, Netflix and S&P500 Index are left-skewed, while Amazon, Google and Microsoft are right-skewed. All five technology stocks have fatter tails compared with normal distribution, however, the S&P 500 Index has slightly thinner tail.

The performance of different stocks also varies a lot. Within technology stocks, Netflix has the best performance, increasing by 85 times throughout the whole period. Apple and Amazon both increased by over 30 times. Although Google and Netflix only grew up by around five times, they still outperformed S&P500, who only went up by two times.

Summary Statistics of Technology Stocks							
		AAPL	**AMZN**	**GOOG**	**MSFT**	**NFLX**	**S&P500**
Range	Minimum	-19.75%	-24.62%	-12.34%	-12.46%	-42.92%	-9.47%
	Maximum	13.02%	23.86%	18.23%	17.06%	35.22%	10.96%
	Median	0.09%	0.07%	0.03%	0.04%	0.04%	0.07%
Moment	Mean	0.11%	0.11%	0.05%	0.05%	0.14%	0.02%
	Volatility	2.02%	2.46%	1.83%	1.69%	3.35%	1.21%
	Skewness	-0.28	0.58	0.45	0.06	-0.41	-0.38
	Kurtosis(Excess)	6.63	14.80	10.62	10.25	20.63	11.49
Price Pattern	Increase by	30.97	35.00	4.95	5.27	85.28	2.16

Table 1

Table 2 shows the correlation between different stocks. All stocks are highly correlated with each other except Netflix. This is because Netflix experienced abnormal high growth during the past 12 years, which are also shown in Table 1 that Netflix has the highest mean. The correlation between all technology stocks (still, except Netflix) and S&P500 are over 0.6. This shows that the market indicator – S&P 500 is still a good estimator for technology stocks although the log return pattern varies a lot.

Correlation Table of Technology Stocks						
	AAPL	**AMZN**	**GOOG**	**MSFT**	**NFLX**	**S&P500**
AAPL	1.00	0.42	0.51	0.45	0.26	0.59
AMZN	0.42	1.00	0.52	0.47	0.35	0.56
GOOG	0.51	0.52	1.00	0.52	0.28	0.64
MSFT	0.45	0.47	0.52	1.00	0.27	0.69
NFLX	0.26	0.35	0.28	0.27	1.00	0.35
S&P500	0.59	0.56	0.64	0.69	0.35	1.00

Table 2

Figure 1 shows the time series pattern of the daily log return. The red dashed line marks the two standard deviation boundaries. All log returns exceeding these boundaries are considered as extreme events. It is easy to tell that Netflix has the most extreme return including both negative and positive events. Amazon and Google also have many positive abnormal log returns, while Apple and Microsoft are relatively stable among all five technology stocks.

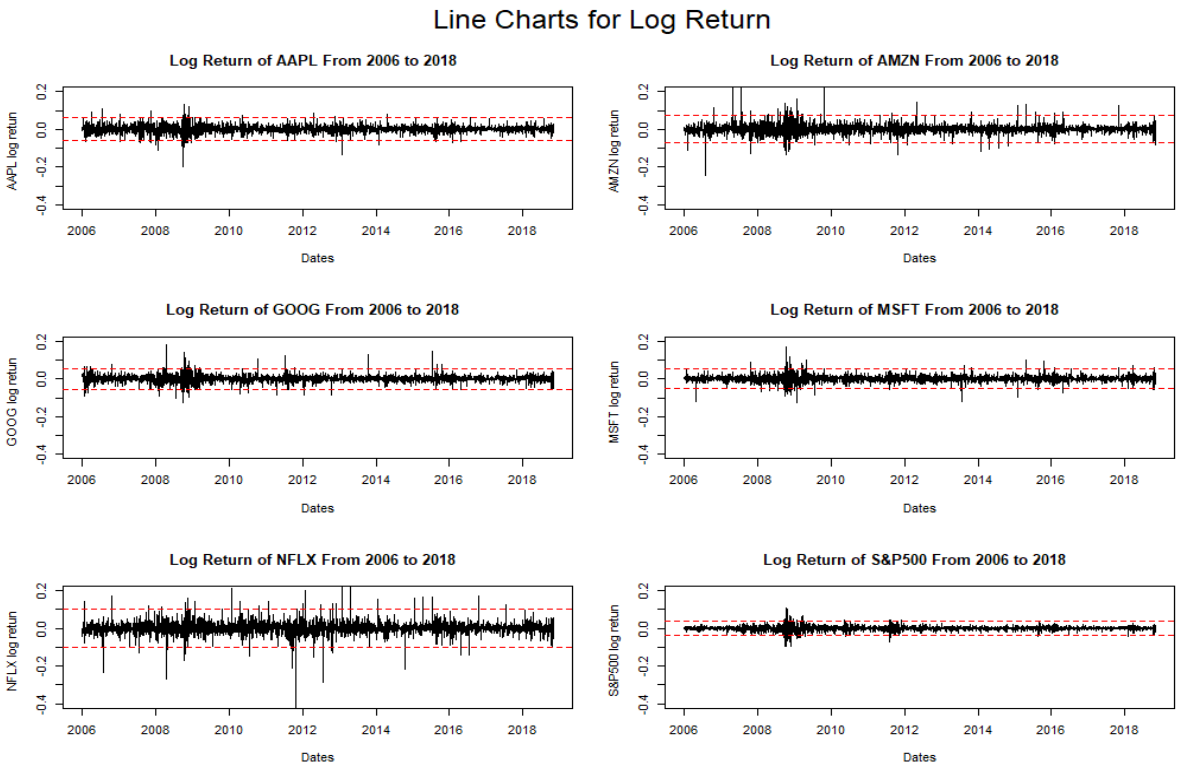


Figure 1

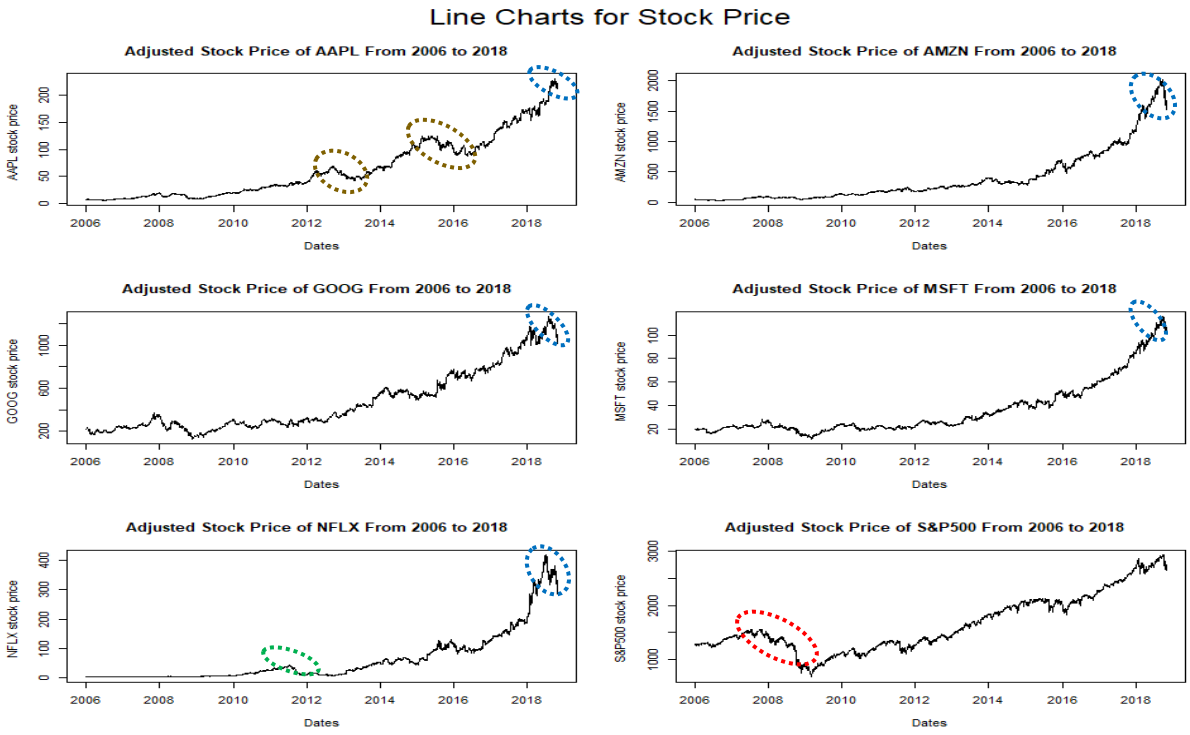


Figure 2

Figure 2 highlights some significant stock downturns from 2006 Jan 1st to 2018 Nov 2nd. Events leading to these price collapse can be classified into the following three categories: 1. Systematic event, an

event that will affect all stocks in the market; 2. Industry event, the event that only has impact on technology stocks; 3. Firm-specific event, the stock price drop event due to firm-specific factors.

Systematic event

September 2007 - January 2009 Global Financial Crisis (GFC) (marked as the red line): In 2007, there were signs about possible implosion of housing market, significant number of defaults and foreclosures were triggered shortly after. Also, the value of the multitrillion-dollar pool of mortgages was undermined by the housing price collapse (Altman, 2009). The financial market began to respond to the subprime crisis in January 2008, when some major ETFs declined by over 10% in only one day. The bankruptcy of Lehman Brothers, one of the largest investment banks at that moment, triggered another collapse in the stock market in September 2008. The giant drop continued until January 2009 (Wikipedia, 2018).

Industry event

October 2018 Technology Stocks Collapse (marked as the blue line): The technology stocks suffered their worst performance since August 2011. The major drop was caused by the huge losses in bond, which drove the interest rate higher. The rising interest rate put downside pressure on stock price, especially on the best performers in the past few years such as Alphabet, Google and Amazon (FORTUNE, 2018). Moreover, the policy uncertainty on technology companies forces more investors to sell technology stocks temporarily to eliminate risk as it is approaching the US Midterm Election in November 2018 (CNBC, 2018).

Firm-specific event

As there are many firm-specific events during the past 12 years. The report will only pick up two typical events to demonstrate.

Netflix Stock Price Drop in 2011 Sep (marked as the green line): Netflix announced in July 2011 that it will separate its DVD-by-mail service from Internet streaming service, which will push the price of each DVD from $2 up to $8. The price change led to many customers cancelled their subscriptions in Netflix and the stock has dropped by 15% by the end of September 2011 (The New York Times, 2011).

Apple Stock Decline between 2012 and 2016 (marked as the brown line): The stock price of Apple.Inc has plunged for 43% from September 2012 to April 2013 as the new-released iPhone5 underperformed the market's expectation (AppleNewsroom, Apple Reports Third Quarter Results, 2012). Due to the similar reason, the stock price dropped by 25% from June 2015 to June 2016 after the release of iPhone6s (AppleNewsroom, Apple Reports Record Second Quarter Results, 2015).

Part b & c. Portfolio Construction

Methodology

The stock data used for the six stocks are the adjusted close price data from 2006 Jan 1st to 2018 Nov 2nd, where 2006 Jan 1st – 2007 Dec 31st is training period and the remaining time is testing period. The report will use the data in training period to estimate key coefficients of some statistical models, and then use these models to make predictions in testing period. Note that all portfolios are only constructed under testing period.

Step I: Estimate the Volatility

We use GARCH (1,1) model (details about GARCH model could be found in Appendix I) to estimate the volatility of each stock at each node. The volatility at the t^{th} node for stock (i) can be calculated as the following:

$$\sigma_{t,i}^2 = \omega_i + \alpha_i \times r_{t-1,i}^2 + \beta_i \times \sigma_{t-1,i}^2, \text{ (Bollerslev, 1986)}$$

Therefore, the t^{th} node volatility can be estimated from the volatility and return at the last node, which are all available information. We expect that the volatility of each stock will deviate from its long-term mean during the whole period. Therefore, we use the unconditional long-run mean to be the start point of the volatility (the proof could be found in Appendix I), i.e.

$$\sigma_{0,i}^2 = \frac{\omega_i}{1-\alpha_i-\beta_i}$$

For example, if the rebalance period is one week, the coefficients of GARCH (1,1) model for each stock estimated from the training dataset are shown in Table 3:

	ω	a	b	σ_0
AAPL	0.0025416519	7.796639e-02	1.080557e-13	0.0027565718
AMZN	0.0038952266	4.589284e-16	4.882407e-02	0.0040951695
GOOG	0.0016289979	5.000037e-02	5.000019e-02	0.0018099988
MSFT	0.0005787634	4.039812e-01	3.681705e-01	0.0009785808
NFLX	0.0035579417	9.378127e-16	4.970812e-02	0.0037440515
S&P500	0.0002591043	2.350701e-08	3.478590e-02	0.0002684424

Table 3

Step II: Estimate the Return

Momentum Strategy:

The momentum strategy assumes that the trend of all stock price movements will continue in the future. For example, if the daily realised return on yesterday is 0.2%, then today's stock price is also very likely to increase by 0.2%. Similarly, if the stock declined by a large extent at last node, then it is highly possible to also drop a lot at the coming rebalance node. Therefore, the expected return of stock (i) in the next node will be the same as latest node's realised return, i.e. $E(r_{t,i}) = r_{t-1,i}$.

Value Strategy:

Value strategy assumes that all stock prices have a mean-reversion moving pattern. When the stock has a return higher than its long-term mean, its price is expected to fall back to its long-term average in the future. For example, if the stock's long-term monthly log-return is estimated to be 2% and the stock's realised log-return in last month was 5%, then the log return of next month will be (2%-5%) = -3% to meet its mean-reversion pattern. To simplify, the expected return of stock (i) at the t^{th} node will be the estimated long-run mean at t^{th} node minus the realised return at $(t-1)^{th}$ node, i.e. $E(r_{t,i}) = \mu_{t,i} - r_{t-1,i}$. In this report, we calculate the long-run return at t^{th} node as the mean of previous one-year-long rebalance periods, i.e. $\mu_{t,i} = \frac{\sum_{t-n}^{t} r_i}{n}$, where n is the number of rebalance periods in one year. For example, n=52 if rebalance period is one week and n=12 if rebalance period is one month.

Dynamic Strategy:

Instead of assuming all stocks follow only the value or momentum pattern, we estimate each stock's return more precisely by applying autocorrelation model – AR(1) in this strategy. AR(1) model assumes that the return at t^{th} node for stock (i) will be the return at $(t-1)^{th}$ node multiplies a coefficient plus an error term, i.e. $r_{t,i} = \rho_{t,i} * r_{t-1,i} + \epsilon_{t,i}$, where ε has an expected mean of zero. Positive $\rho_{t,i}$ means that for stock (i) at time t, the estimated return at this node will be positive if the return at last node is also positive, which is consistent with momentum strategy. Negative $\rho_{t,i}$ then corresponds to value strategy vice versa. We estimate the coefficients for each stock on a rolling basis where the moving window is one year. In this way $\rho_{t,i}$ could be updated at the end of each rebalance period using the data available in the past one year. R output shows that coefficient's sign of a single stock may change throughout the testing period. For example, Apple was a momentum stock in 2008-2010 and 2013. However, it became a value stock in 2011-2012 and 2014-2017. The coefficients for all stocks rebalance each week is shown

in Figure 3. Coefficient plots of daily and monthly rebalanced portfolios could be found in Appendix II.

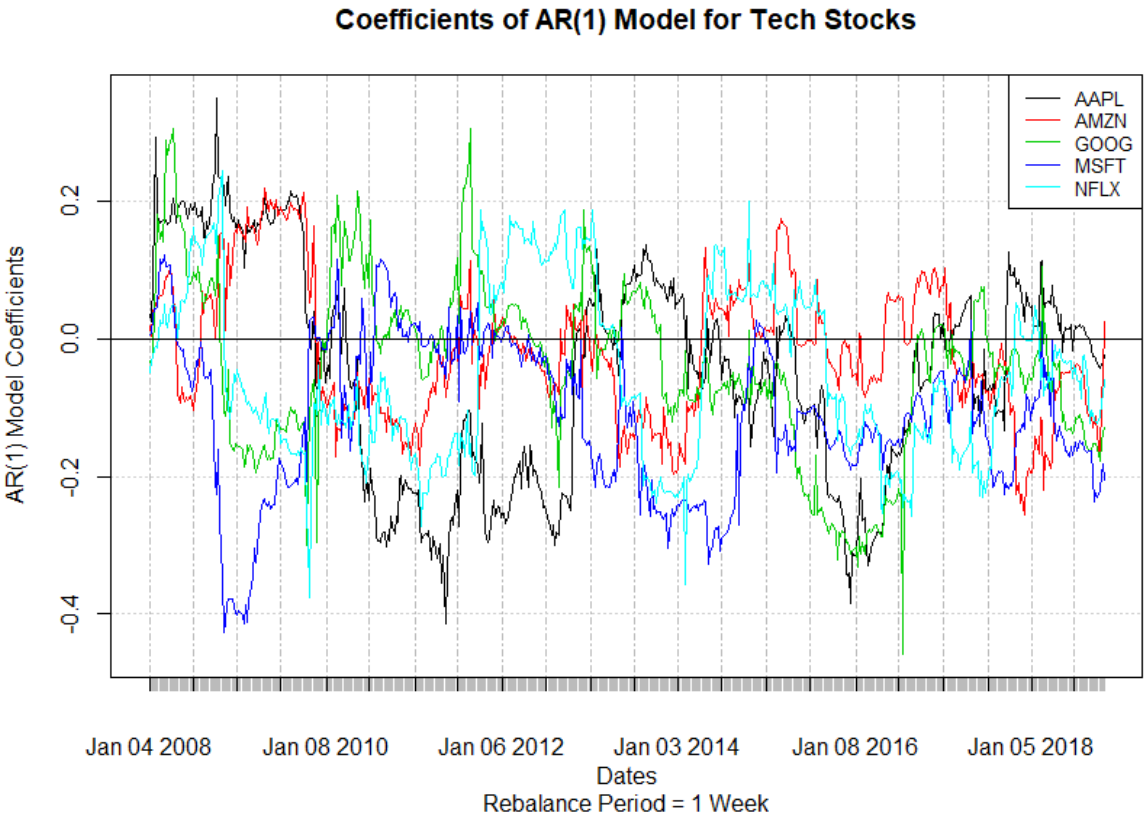


Figure 3

Step III: Calculate Weights of Capital Invested in Each Stock

In all of our strategies, the weights are determined in order to achieve the most efficient portfolio, i.e. the maximum Sharpe Ratio. We assume no borrowing in this strategy, i.e. the weights of all five stocks will sum up to one. Some constraints are added on the weights in order to hold reasonable positions of all the stocks – the weights should be in the range of negative one and positive one. To conclude, we need to find a set of weights such that:

$$\begin{cases} arg\ max_{w_t} = \{\frac{{w_t}' r_t - r_f}{w_t' \Sigma w_t}\} \\ \Sigma_{i=1}^{5} w = 1 \\ w_i \in [-1,1] \end{cases}$$, where Σ is the covariance matrix of r_t

The weights of each stock for different strategies are shown as the following line chart. It is easy to see that the weights in value strategy are nearly the inverse of that in momentum strategy. This is because the estimated long run return in value strategy is always close to zero. Therefore, the estimated log return at each node is approximately to be negative r_{t-1}, which is the opposite number to the expected return in momentum strategy. For example, Microsoft has a relatively high return and a relatively low volatility, whose trend is expected to be continued in momentum strategy, however, will be reversed in value strategy. Therefore, momentum strategy tends to invest a large proportion of the capital (around 50%) in MSFT while value strategy tends to short MSFT a lot.

The weights of autocorrelation strategy are similar to that of the value strategy. This is because many stocks are defined as value stocks (have negative autocorrelation coefficients) most of the time during the past ten years as is shown in Figure 4. Weight plots of daily and monthly rebalanced portfolios are available in Appendix III.

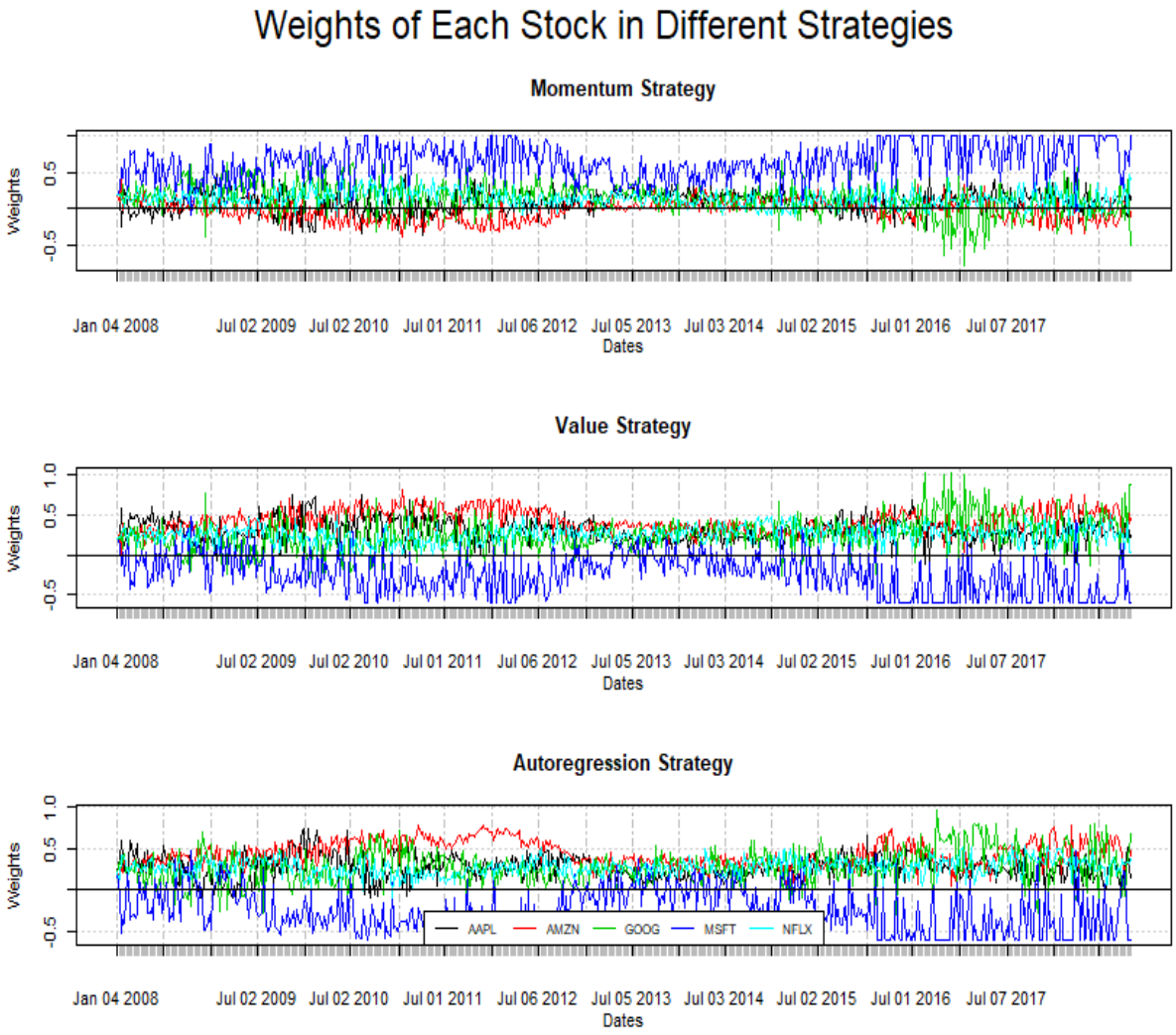


Figure 4

Step IV: Calculate Total Wealth at the End of Each Node

The initial capital is $5000, with $1000 invested in each stock at the first rebalance period (i.e. initial weight is 0.2 for all stocks). If the beginning wealth at the end of node (t-1) is: V_{t-1}, and the weight of stock (i) at node t is calculated as $w_{t,i}$, then the ending wealth of stock (i) at node t will be:

$$V_t = Number \text{ of stock(i) at node t} * \text{Close price of node t}$$

$$= \frac{\text{Total capital invested in stock (i) at node t}}{\text{Open price at node t}} * \text{ Close price at node t}$$

$$= \frac{\text{Total capital at the end of node t} - 1 \ * \text{ weigh of stock (i) at node t}}{\text{Open price at node t}} * \text{ Close price at node t}$$

$$= V_{t-1} * w_{t,i} * \frac{S_{t-1,i}}{S_{t,i}}$$

Findings

Performance Comparisons Between Three Strategies with Different Rebalance Period

We designed three strategies as above mentioned, namely momentum strategy, value strategy and dynamic strategy. For these three strategies, portfolios are rebalanced daily, weekly and monthly. The performance of the three strategies under different rebalance period are shown in Figure 5 and Table 4. While there is no obvious advantage between weekly and monthly rebalanced portfolios, it is easy to see that the daily-rebalanced portfolios performed the worst in all three strategies. This indicates that rebalancing portfolios too frequently may not be a good practice since it may lead to overfitting of time series pattern.

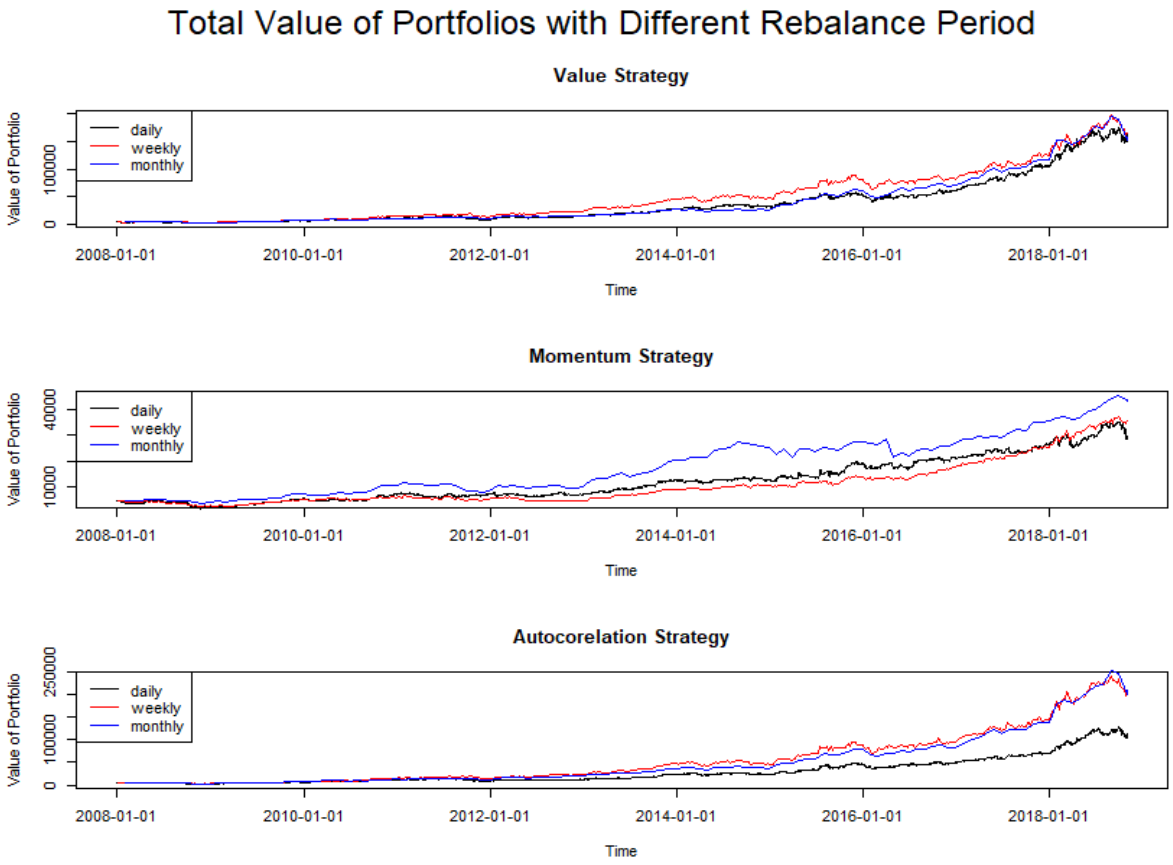


Figure 5

Total Value of Portfolios at the End								
Momentum Strategy			Value Strategy			Dynamic Strategy		
Daily	Weekly	Monthly	Daily	Weekly	Monthly	Daily	Weekly	Monthly
$28,994.00	$35,567.03	$43,316.57	$166,269.50	$165,092.60	$157,769.80	$113,698.50	$205,629.50	$208,283.90

Table 4

In the following sections, we will use weekly rebalanced portfolio as an example to further discuss the performance and risk measurement. Information about daily and monthly rebalanced portfolios can be found in the Appendix IV & V.

Performance Comparisons Between Three Strategies and S&P 500

As can be seen from Figure 6, the dynamic strategy performs the best – total value of portfolio is $205,629.5 at the end of period, which is about 19 times of the S&P500 value. The portfolio capital movement pattern of value strategy is very similar to that of dynamic strategy. But total capital of value portfolio ($165,092.6) is lower than that of dynamic portfolio, which is about 15 times of the S&P500

index stock price. There's no significant capital fluctuation in each week for the momentum portfolio, whose total capital is $35,567 at the end, which is much less than the capital of dynamic portfolio. The capital of momentum strategy is about 3.3 times of the S&P500 index value. Overall, all these three strategies perform better than S&P500 index.

Note: The S&P500 index value is calculated assuming we invest $5,000 in S&P500.

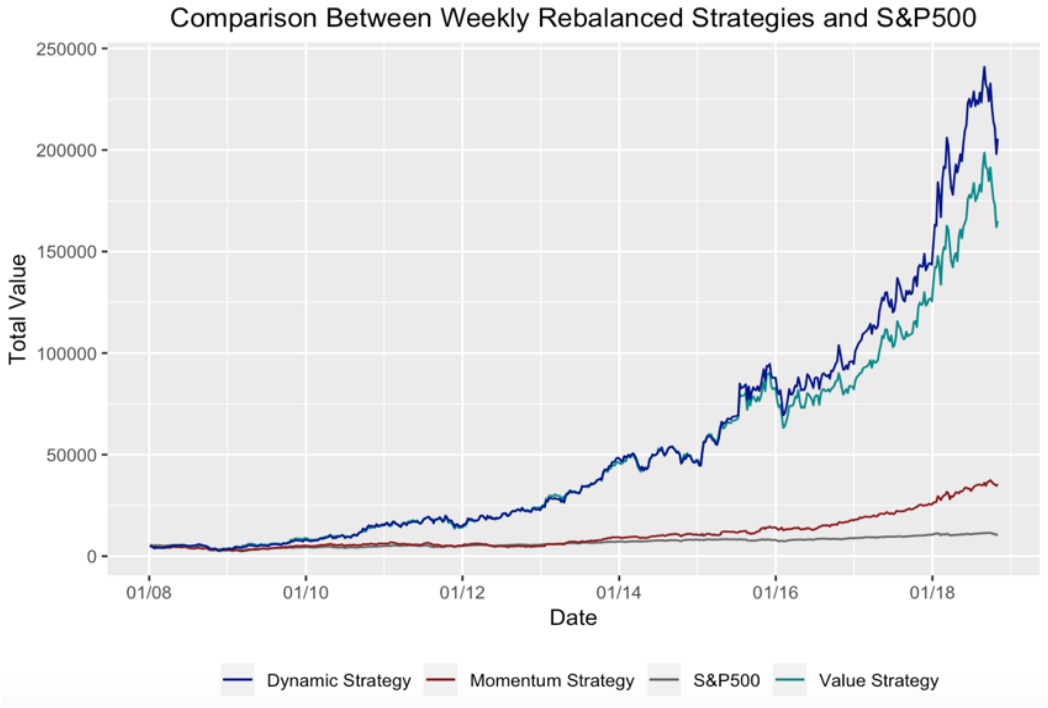


Figure 6

Total Value of Weekly Rebalanced Strategy at End of Period	
Momentum Strategy	$35,567.00
Value Strategy	$165,092.60
Dynamic Strategy	$205,629.50
S&P500	$10,805.60

Table 5

Risk of Portfolios

99%Value-at-Risk (VaR) and 97.5% Expected Shortfall (ES) are used to measure the severity of risk arisen from holding these portfolios over one-week period. We compare two different models to estimate the loss distribution, both normal distribution and historical simulation.

For the normal distribution method, we assume the log returns of portfolios follow the normal distribution. Then, VaR of log returns can be calculated as $\text{VaR}^*_{0.99} = \mu + \sigma \times \Phi^{-1}(0.99)$, where μ and σ are the mean and variance of log returns, $\Phi^{-1}(0.99)$ denotes the 99th quantile of standard normal distribution. $\text{VaR}^*_{0.99}$ can be converted to $\text{VaR}_{0.99}$ of non-linearised loss distribution by taking $V_t * (\exp(VaR^*_{0.99}) - 1)$, where V_t corresponds to portfolio value at the end. When working with a short time horizon and the market is not too volatile, $(\exp(VaR^*_{0.99}) - 1)$ is approximate to $VaR_{0.99}$ then the $VaR_{0.99}$ of linearised loss will be $V_t * \text{VaR}^*_{0.99}$. The 97.5% expected shortfall of log return is $ES_{0.975} = \mu + \sigma \times \frac{\phi(\Phi^{-1}(0.975))}{1-0.975}$, where ϕ denotes the cumulative density function of standard normal distribution. The $ES_{0.97}$ of linearised and non-linearised loss distribution can be found in the same way as VaR.

Instead of using pre-defined model with explicit parameters for loss distribution, historical simulation is the method for estimating the loss distribution according to the empirical distribution of log return. If there are n historical observations of log returns, Then the $\mathrm{Va}R_{0.99}$ is estimated by taking the $(1\%n)^{th}$ largest value and $ES_{0.975}$ is estimated by taking the average of the 2.5%n largest losses. The log return is converted to change of value by taking $\exp(\log \text{return}) - 1$ for non-linearised loss distribution, whereas log return is approximate to change of value for linearised loss distribution.

Momentum Strategy:

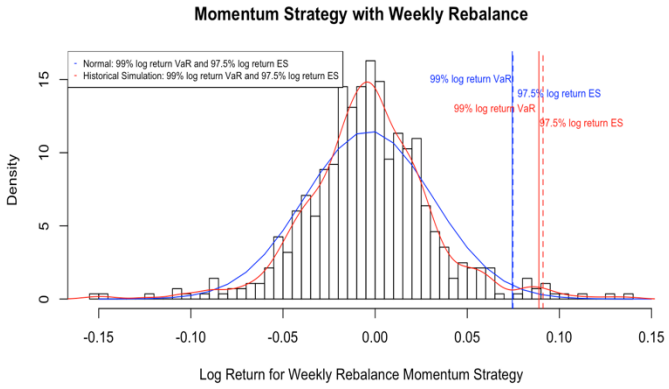


Figure 7

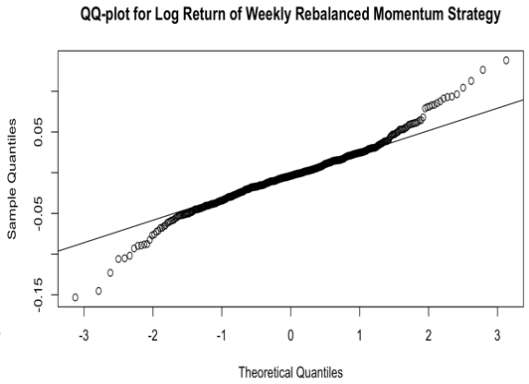


Figure 8

99%VaR and 97.5%Expected Shortfall for Weekly Rebalanced Momentum Strategy							
Non-linear Loss Distribution Function				Linear Loss Distribution Function			
Normal Model		Historical Simulation		Normal Model		Historical Simulation	
99%VaR	97.5%ES	99%VaR	97.5%ES	99%VaR	97.5%ES	99%VaR	97.5%ES
$2,647.24	$2,660.34	$3,160.88	$3,240.57	$2,750.93	$2,765.09	$3,310.26	$3,402.93

Table 6

Figure 7 summarises the distributions and non-linearised risk measures of log returns for the weekly rebalanced momentum strategy. Blue curve represents the normal log return distribution, 99% normal log return VaR ($0.074) is marked as the blue solid line, which is slightly smaller than 97.5% normal ES ($0.075) that is shown with a blue dashed line. By multiplying the total value of this portfolio ($35,567), The 99% VaR and 97.5% ES of weekly momentum portfolio can be obtained. As shown in Table 6, 99% VaR is $2,647.24, which means the probability of loss of momentum portfolio in next period larger than $2,647.24 is smaller than 1%. The 97.5% ES is $2,660.34, which is the average of all losses which larger than 97.5% VaR ($2,225.55).

On the other hand, red curve represents the historical simulation distribution of log return, which is narrower in the central and have heavier tails than blue line, indicating that historical simulation loss distribution has a higher kurtosis, i.e. fatter tails. Figure 8 is a QQ plot of historical simulation log return against a normal reference distribution, it resembles an inverted "S-shaped" curves. That indicates the empirical quantiles of log return tend to be larger than the normal distribution's corresponding quantiles. Furthermore, by applying the Mardia's tests of multivariate normality, the result in Table 7 shows that the p values of both skewness and kurtosis coefficient are 0. Therefore, normal distribution may not be an appropriate model for log return distribution of weekly momentum strategy – the tails are too thin to assign enough weight to extreme returns. Whereas, historical simulation distribution has heavier tails, so that large returns are more likely. Table 6 shows that the 99% VaR is $ 3,160.88 which is slightly smaller than 97.5% ES ($3,240.57), both are greater than that of normal distribution. In addition, risk measures based on linearized loss distribution are close but all larger than non-linearised loss distribution's corresponding risk measures.

Mardia's Test Result			
b	p-value	k	p-value
6.624	0.000	156.944	0.000

Table 7

Value Strategy:

Figure 9 shows that the kurtosis and skewness of historical simulation distribution of log return (red curve) is just slightly higher than that of normal log return distribution (blue curve). Even though the QQ plot for log return of value strategy still indicates heavy tails, most of points fall along the solid straight line and less observations severe deviate from theoretical normal distribution compared with momentum strategy. Therefore, the gap between blue and red lines in Figure 9 is smaller than that of momentum strategy. Information of risk measures of value portfolio are summarised in Table 8. The 99% non-linearised normal VaR is $15,147.79, which is slightly lower than the 97.5% non-linearised normal ES ($15,223.40). In contrast, the 99% non-linearised VaR of historical simulation distribution ($17,089.58) is larger than 97.5% non-linearised ES ($16,655.29). This is also different from the case of momentum portfolio. Because the expected shortfall is related to both size and the number of extreme losses that larger than 97.5% VaR ($12,709.29), just few observations larger than $17,089.58 (99% VaR), most of outliers concentrated between 97.5% VaR and 99% VaR for value portfolio. Besides, Figure 8 has more heavier tails than Figure 10. Similarly, risk measures based on non-linearised loss distribution are smaller than the corresponding values of linearised loss distribution.

In addition, these four values are much greater than those of momentum strategy, which means value strategy is riskier than momentum strategy. The main reason is that the size of value portfolio is more than four times of the size of momentum portfolio at the end of period, which results in requiring more capital to buffer against the losses of value portfolio for next week.

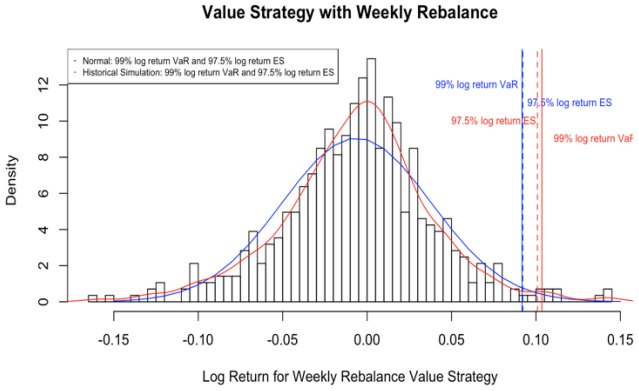


Figure 9

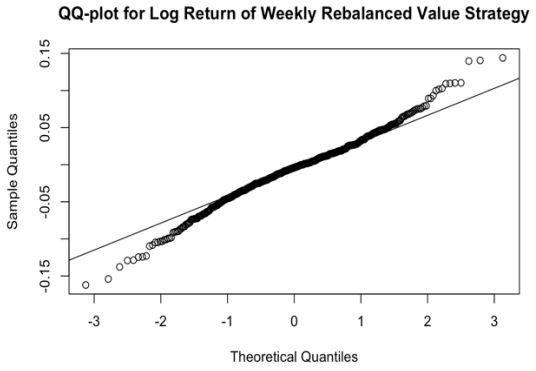


Figure 10

99%VaR and 97.5%Expected Shortfall for Weekly Rebalanced Momentum Strategy							
Non-linear Loss Distribution Function				Linear Loss Distribution Function			
Normal Model		Historical Simulation		Normal Model		Historical Simulation	
99%VaR	97.5%ES	99%VaR	97.5%ES	99%VaR	97.5%ES	99%VaR	97.5%ES
$15,147.79	$15,223.40	$17,089.58	$16,655.29	$15,888.39	$15,971.65	$18,040.31	$17,589.81

Table 8

Dynamic Strategy:

The outcome of dynamic strategy is very similar to value strategy. This is because most of the stocks in the dynamic portfolio are defined as value stocks most of the time. As can be seen from Figure 11, The kurtosis and skewness of historical simulation loss return distribution is higher than those of normal loss distribution. Table 9 summarises the risk measures of dynamic portfolios. For the normal non-linearised loss distribution, the 99% VaR is $19,391.97 and the 97.5% ES is $19,488.81. For the

historical simulation model, the 99% VaR is \$20,811.87 and the 97.5% ES is \$21,329.38. In both loss distribution model, the 97.5% ES is larger than 99% VaR. But these four values are very close to each other, all around \$20,000. Similarly, the risk measures of non-linearised loss distribution is smaller than those of linearised loss distribution respectively. These four measures of dynamic strategy are much greater than those of value strategy, this is mainly because the size of dynamic portfolio (\$205,629.5) is greater than the size of value portfolio (\$165,092.6) at the end of period. The larger the portfolio, the more capital needed as a buffer to against unexpected losses in next week.

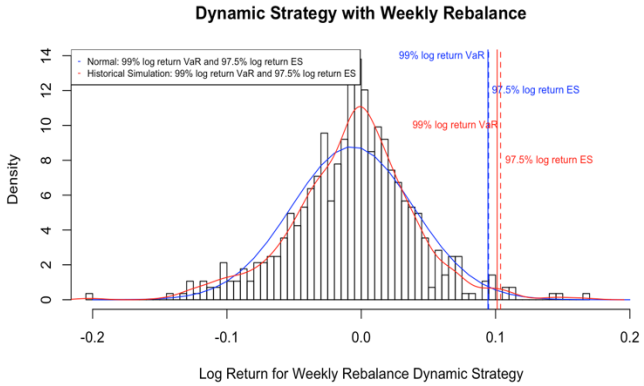


Figure 11

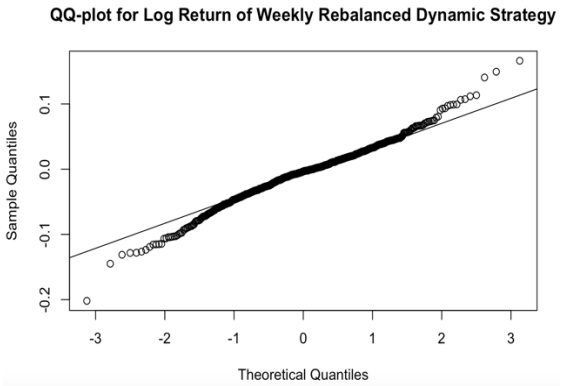


Figure 12

99%VaR and 97.5%Expected Shortfall for Weekly Rebalanced Momentum Strategy							
Non-linear Loss Distribution Function				Linear Loss Distribution Function			
Normal Model		Historical Simulation		Normal Model		Historical Simulation	
99%VaR	97.5%ES	99%VaR	97.5%ES	99%VaR	97.5%ES	99%VaR	97.5%ES
\$19,391.97	\$19,488.81	\$20,811.87	\$21,329.38	\$20,368.24	\$20,475.20	\$21,942.02	\$22,573.19

Table 9

Part d & e. Mini Index Return & Risk Analysis

Mini Index Construction

We use the market capital of the five technology stocks in Nasdaq to calculate their weights in the mini index. Weight of $stock_i$ in the mini index is the proportion of its market cap out of the total market value of all five technology stocks, i.e.,

$$\omega_i = \frac{\text{market capital}_i}{\sum_{i=1}^{5} market\ capital_i}$$

For example, weight of AAPL is $718.7/(718.17 + 685.83 + 704.22 + 749.44 + 106.26)$, which equals 24.23%. Market capital and weights' result is available in Table 10.

Market Cap (\$bn)				
AAPL	AMZN	GOOG	MSFT	NFLX
\$718.17	\$685.83	\$704.22	\$749.44	\$106.26
24.23%	23.14%	23.76%	25.29%	3.59%

Table 10

The report assumes fixed weights over the whole trading period, and the log return of the mini index is the weighted average of the log return for all five stocks:

$r_{t,mini\_index} = \sum_{i=1}^{5} \omega_{t,i} * r_{t,i}$, where $r_{t,i}$ is the log-return of $stock_i$ at time t.

From Figure 13, we can see that log-return of five stocks and mini-index fluctuated between -10% and 10% in most of the time during the past 12 years, and large drawdown happened in 2008 due to global financial crisis.

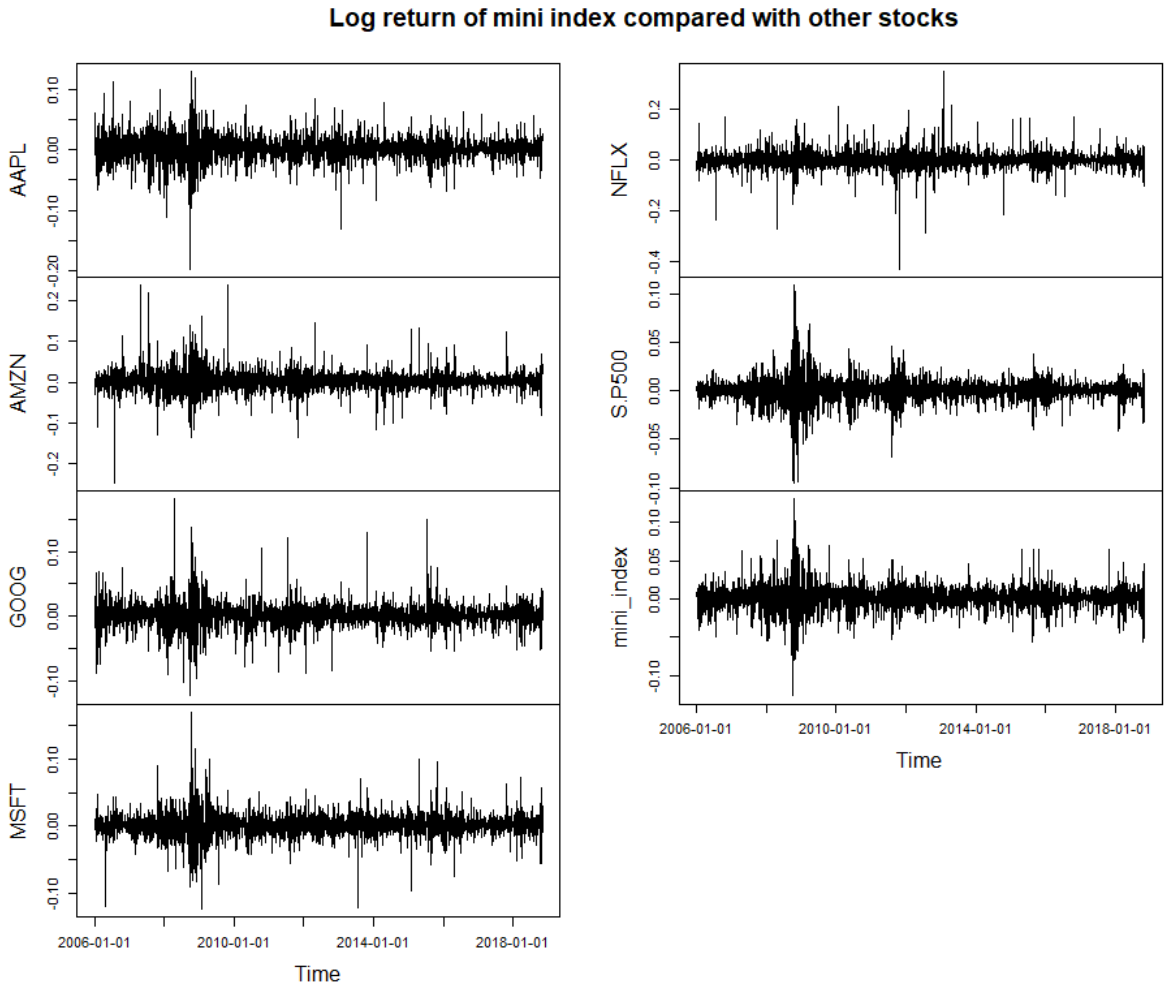


Figure 13

Fitting copulas to data

Copula measures the dependence between different stocks based on their marginal distributions. We denote $\widehat{F_1}, \dots, \widehat{F_5}$ as the estimated marginal density functions. By Sklar's Theorem, the cumulative density function follows the standard uniform distribution, i.e., $F_X(x) \sim U(0,1)$, or equivalently, $F_{X_i}(x_i) = U_i$. Therefore, the pseudo-sample from the copula consists vectors $(\widehat{U_1}, \dots, \widehat{U_n})'$, where $\widehat{U_t} = (\widehat{U_{t,1}}, \dots, \widehat{U_{t,5}})' = \left(F_1(\widehat{X_{t,1}}), \dots, F_5(\widehat{X_{t,5}})\right)'$

We use non-parametric estimation $\widehat{F_i}$ to fit copulas to data, where

$$\widehat{F_i(x)} = \sum_{t=1}^{n} \frac{I(X_{t,i} < x)}{n+1}$$

The pattern of the fitting result is symmetric along the red line as is shown in Figure 14. Also, most dots concentrate on the red diagonal line and few dots are in the (0,1) and (1,0) corners, which indicates

dependence between S&P500 and mini index and it is less likely to find small value of mini-index when S&P500 is large. Spearman rank correlation (0.7287) verifies the dependence statement.

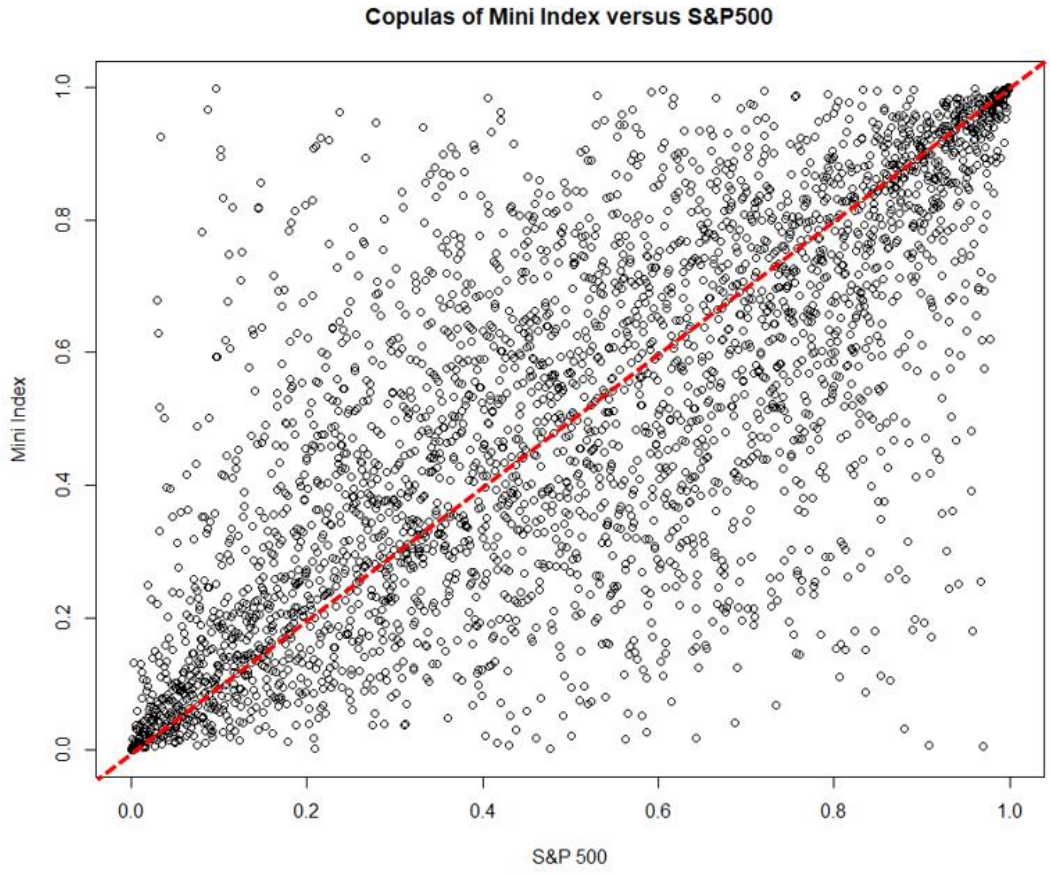


Figure 14

We then fit Gaussian copula, t copula, Gumbel copula and Clayton copula to see which copula fits the observations best. The corresponding maximum likelihood indicators are 1404.427, 5773.118, 1432.371 and 1241.503 respectively and therefore t copula is the best fitting.

Return Level and Return Period

Based on the mini index constructed using data from 2006 to 2018, it is obvious that there exist large magnitude fluctuations during the latter half of 2008 because of the global financial crisis (Figure 15). This global financial crisis is considered by many economists to have been the worst financial crisis since the Great Depression in 1930s.

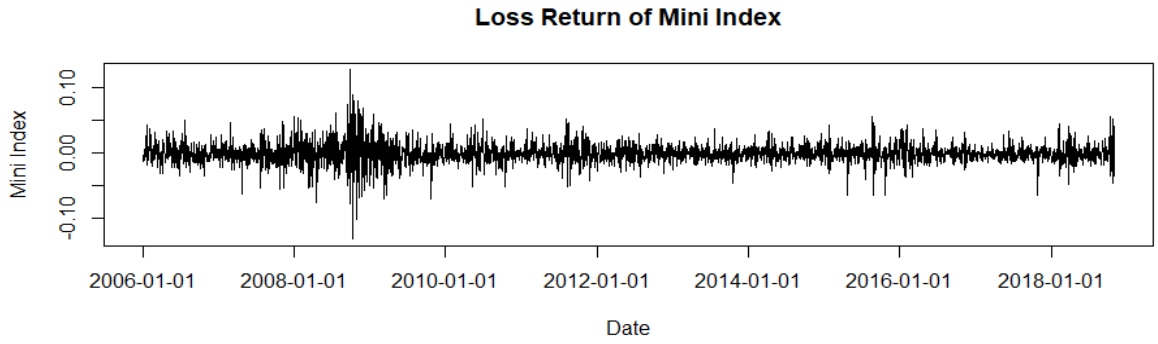


Figure 15

Fitting GEV Distribution

We first calculate the monthly and quarterly maxima from the loss return dataset and fit into General Extreme Value (GEV) distribution function. Since there does not exist a "perfect" way on determining the optimal block size, we need to try different block size and fit the GEV distribution to test the parameters' sensitivity.

Figure 16 displays the density distribution function of each monthly and quarterly maxima selection. The solid lines in both plots are the theoretical density function and the dashed lines are the true density function. The rough information from Figure 16 is that both density distributions are right-skewed. Shape of the tail still needs to be verified through numerical calculations.

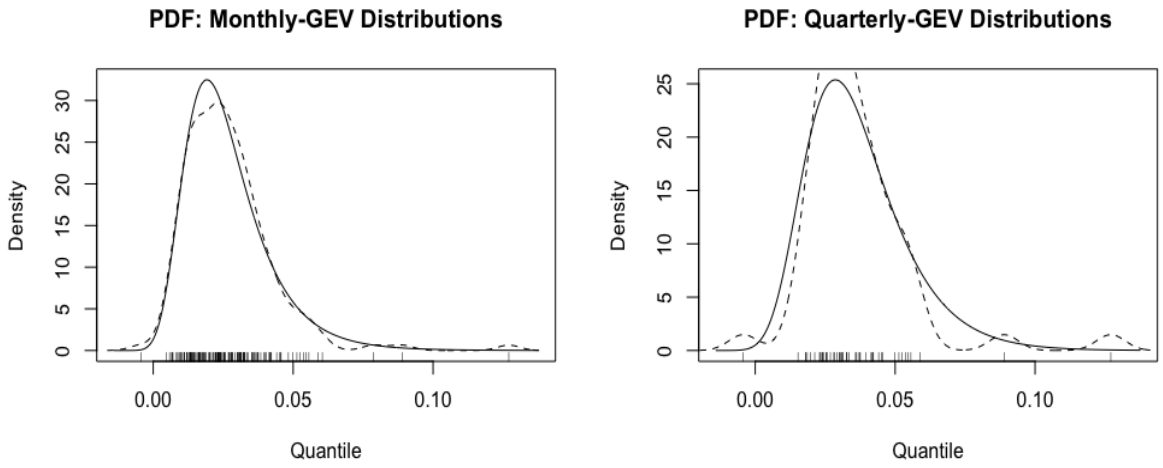


Figure 16

The coefficients of fitted GEV model for monthly and quarterly maxima by Maximum Likelihood Estimation (MLE) are shown as the following:

Note: μ is location parameter, showing the "center" or mean of the distribution;

σ is the scale parameter, showing the extent of the observations deviating from the "center";

ξ is the shape parameter, when ξ<0, the distribution has a thinner tail; when ξ>0, the distribution has a heavier tail.

- Fit the GEV Distribution to Monthly Maxima Dataset (one block corresponds to one month):
  - $\hat{\mu}_{monthly} = 0.0197$ with standard error 0.0010;
  - $\hat{\xi}_{monthly} = 0.0424$ with standard error 0.0451,
    Frechet distribution as $\hat{\xi}_{monthly} > 0$
  - $\hat{\sigma}_{monthly} = 0.0113$ with standard error 0.0007
- Fit the GEV Distribution to Quarterly Maxima Dataset (one block corresponds to a quarter)
  - $\hat{\mu}_{quarterly} = 0.0284$ with standard error 0.0021;
  - $\hat{\xi}_{quarterly} = -0.0200$ with standard error 0.0529
    Weibull distribution as $\hat{\xi}_{quarterly} < 0$
  - $\hat{\sigma}_{quarterly} = 0.0144$ with standard error 0.0014

In risk management practice, Frechet distributions are more desirable due to its heavy-tailed characteristics.

Calculating Return Level and Return Period

Return level shows the loss return threshold that the mini index exceeds once every k blocks, which is equivalent to the (1-1/k)th quantile of the GEV distribution. For example, if the return level for the quarter maxima over the past 10 years is 7.965%, then the loss of the mini index is expected to exceed 7.965% once every 40 quarters. And 7.965% is also the (1-1/40)=97.5%th quantile of the fitted GEV distribution of the quarter maxima dataset. The detailed formulae to calculate the return level is: $\hat{r}_{n,k} = H^{-1}_{\hat{\xi},\hat{\mu},\hat{\sigma}}\left(1-\frac{1}{k}\right) = \hat{\mu} + \frac{\hat{\sigma}}{\hat{\xi}}\left(\left(-\ln\left(1-\frac{1}{k}\right)\right)^{\hat{\xi}} - 1\right)$, where H denote the distribution function of the true distribution of the n-block maximum.

The return period $\hat{k}_{n,u}$ measures how many blocks on average do we need to see a single block exceeding the return level u, where u stands for the maximum loss of the past n trading days. For example, the maximum quarterly loss over the past 10 years is 12.07% and the return period is 63.27, then it indicates loss is expected to be over 12.71% every 63.27 quarters (15.8 years). Detailed formulae to calculate the return period is: $\hat{k}_{n,u} = \frac{1}{\bar{H}_{\hat{\xi},\hat{\mu},\hat{\sigma}}(u)}$, where $\bar{H}_{\hat{\xi},\hat{\mu},\hat{\sigma}}(u)$ is the probability that the loss return of the mini index is larger than u.

- Monthly Maxima Dataset:
  - 120-month return level-10 years: $\hat{r}_{n,120} \approx 7.965\%$
    only expect to see once every 10 year that in one year in a day the loss is larger than 7.965%.
  - Maximum loss over the past 120 months: 12.71%; Return period: $\hat{k}_{n,u} = 189.64 \text{ months} = 15.8 \text{ years}$
- Quarterly Maxima Dataset:
  - 40-quarter return level-10 years: $\hat{r}_{n,40} \approx 7.951\%$
    only expect to see once every 10 year that in one year in a day the loss is larger than 7.951%.
  - Maximum loss over the past 40 quarters: 12.71%; Return period: $\hat{k}_{n,u} = 63.27 \text{ quarters} = 15.8 \text{ years}$

Figure 17 shows the monthly and quarterly block maxima respectively and the red line marks the return level. The information that can be extracted from both return levels and return periods is that the estimated return level and return period is not very sensitive to the change of the block size. In addition, the risk level of the portfolio is "moderate" as the return level is not very large, and the return period is relative long.

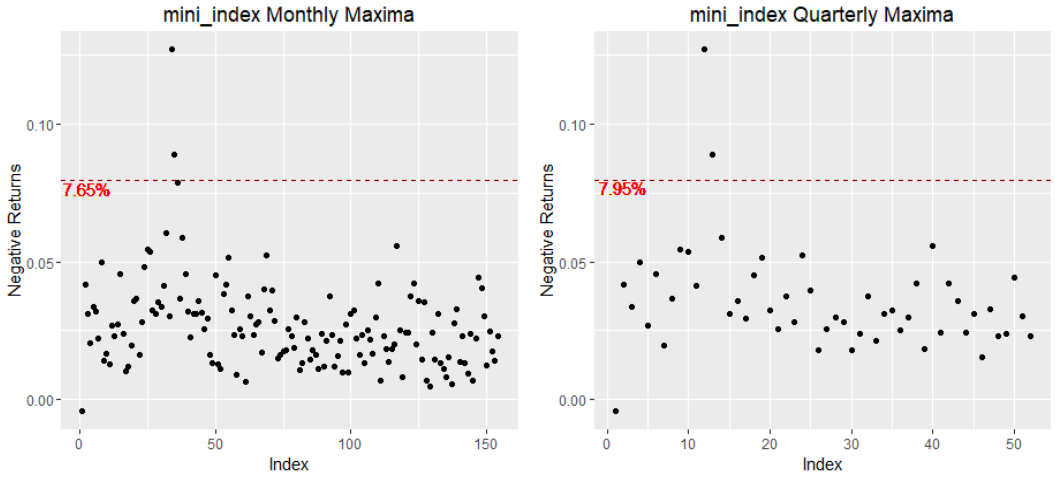


Figure 17

One potential issue of this estimation is the so-called “variance-bias” trade-off. Quarterly and monthly maxima only contain 60 and 12 observations in each block respectively, which are relatively small compared with that of semi-annual and annual maxima. This may lead to high variance of estimation as small number of observations does not necessarily force convergence of the cumulative density function to GEV distribution. While large number of observations in each block provides better convergence, it ends up with less datapoints to estimate the coefficients of GEV distribution, which may also result in higher bias when calculating μ, σ and ξ parameters.

Bibliography

AppleNewsroom. (2012, Jul). *Apple Reports Third Quarter Results*. Retrieved from Apple: https://www.apple.com/newsroom/2012/07/24Apple-Reports-Third-Quarter-Results/

AppleNewsroom. (2015, Apr). *Apple Reports Record Second Quarter Results*. Retrieved from Apple: https://www.apple.com/uk/newsroom/2015/04/27Apple-Reports-Record-Second-Quarter-Results/

Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 311.

CNBC. (2018, Oct). *Tech stocks have their worst day since August 2011*. Retrieved from CNBC: https://www.cnbc.com/2018/10/10/tech-stocks-have-their-worst-day-since-august-2011.html

FORTUNE. (2018, Oct). *Tech Stocks From Apple to Amazon Got Crushed Today. Here's Why*. Retrieved from Forture: http://fortune.com/2018/10/10/stock-market-news-today-apple-amazon-facebook-google/

The New York Times. (2011, Sep). *Netflix Stock Falls After Change in Pricing*. Retrieved from The New York Times: https://mediadecoder.blogs.nytimes.com/2011/09/15/price-hike-sends-netflixs-stock-downward/

Wikipedia. (2018, 12). *Financial crisis of 2007–2008*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Financial_crisis_of_2007%E2%80%932008

Appendix

I. Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

GARCH model is a further extension of Autoregressive Conditional Heteroskedasticity (ARCH), which is first published in Journal of Economics by Tim Bollerslev in 1986. Detailed article could be found through the following link:
https://pdfs.semanticscholar.org/7da8/bfa5295375c1141d797e80065a599153c19d.pdf

General $GARCH(p,\ q)$ specification:

$$\sigma_t^2 = \omega + \sum_{n=1}^{q} \alpha_n r_{t-n}^2 + \sum_{n=1}^{p} \beta_n \sigma_{t-n}^2$$

where $\omega, \alpha_n, \beta_n > 0$

The process is weakly stationary if $\sum_{n=1}^{q} \alpha_n + \sum_{n=1}^{p} \beta_n < 1$

The unconditional variance of a GARCH process can be computed the same way as for an ARCH process: use the fact that $\mathrm{E}[r_t^2] = \mathrm{E}[\sigma_t^2]$ to conclude

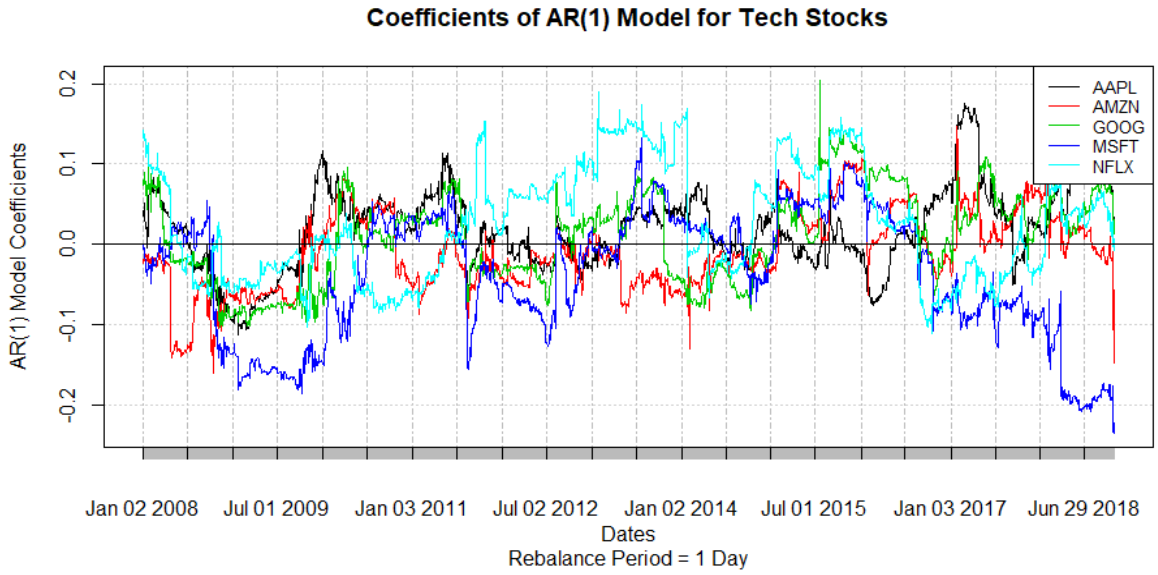
$$\sigma_t^2 = \omega + \sum_{n=1}^{q} \alpha_n r_{t-n}^2 + \sum_{n=1}^{p} \beta_n \sigma_{t-n}^2 \Rightarrow$$

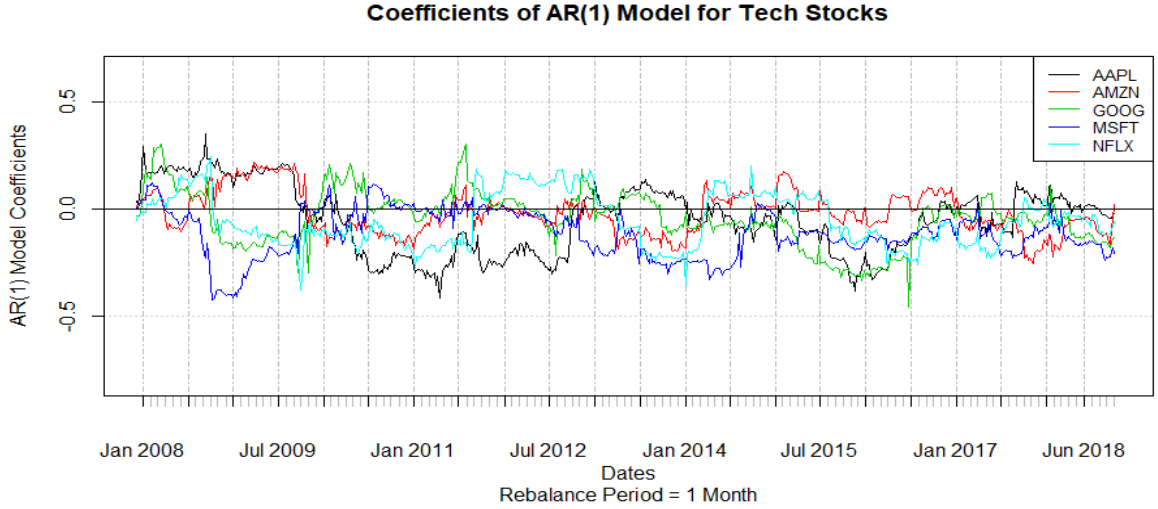
$$\mathrm{E}[\sigma_t^2] = \omega + \sum_{n=1}^{q} \alpha_n \mathrm{E}[r_{t-n}^2] + \sum_{n=1}^{p} \beta_n \mathrm{E}[\sigma_{t-n}^2] \Rightarrow$$

$$\bar{\sigma}^2 = \omega + \left[\sum_{n=1}^{q} \alpha_n + \sum_{n=1}^{p} \beta_n \right] \bar{\sigma}^2 \Rightarrow$$

$$\bar{\sigma}^2 = \frac{\omega}{1 - \sum_{n=1}^{q} \alpha_n - \sum_{n=1}^{p} \beta_n}.$$

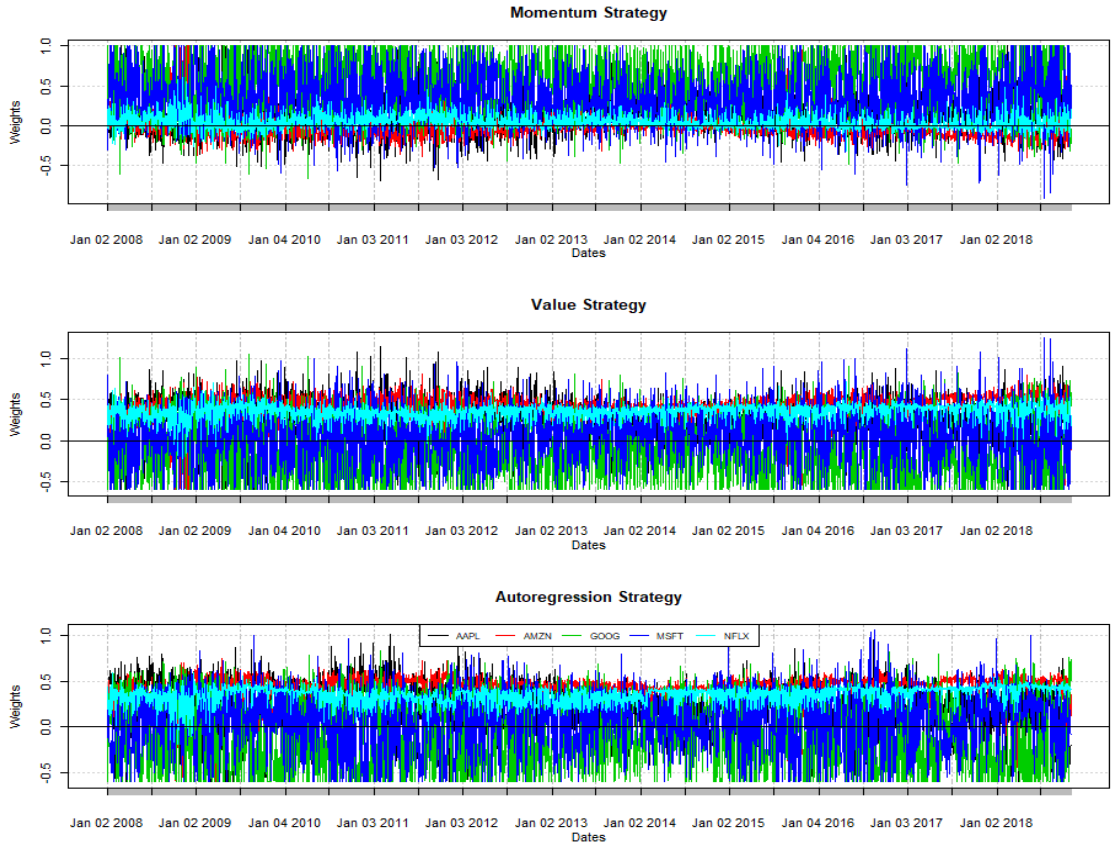
II. Coefficients of AR(1) Model for Daily and Monthly Rebalanced Portfolio



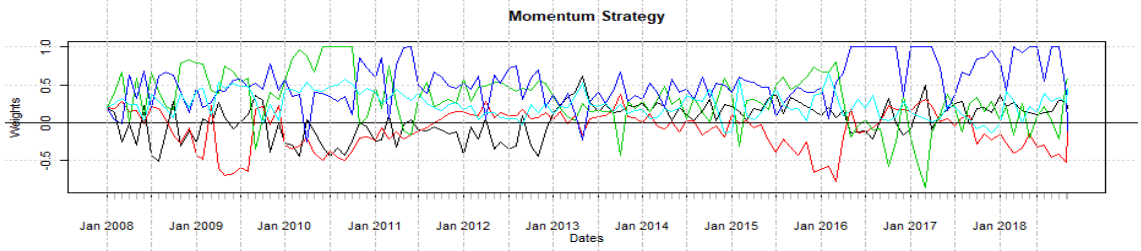


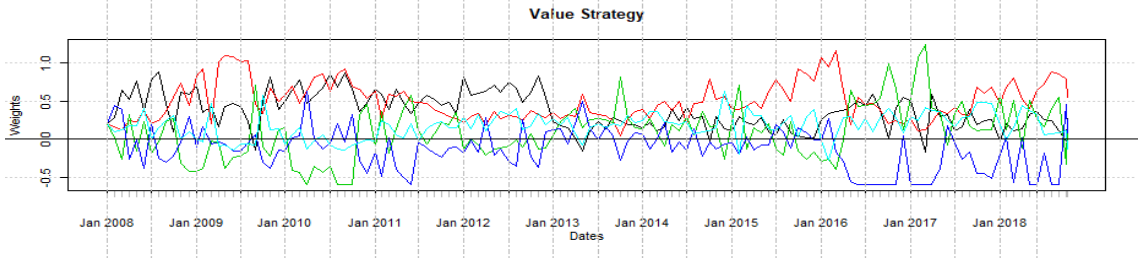
III. Weights for Different Strategies of Daily and Monthly Rebalanced Portfolios

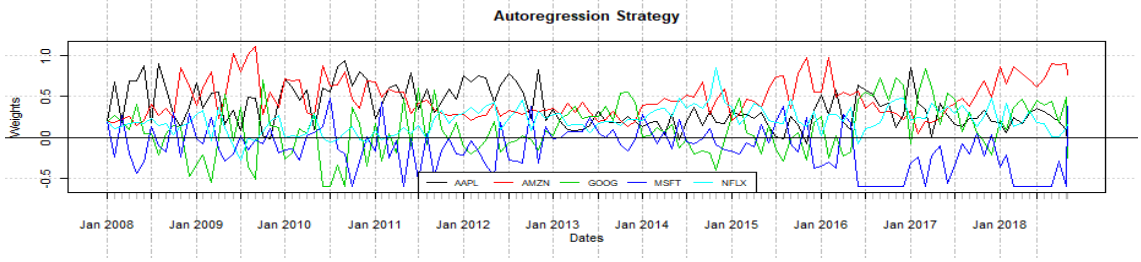
Weights of Each Stock in Different Strategies,Rebalance Period: 1 Day



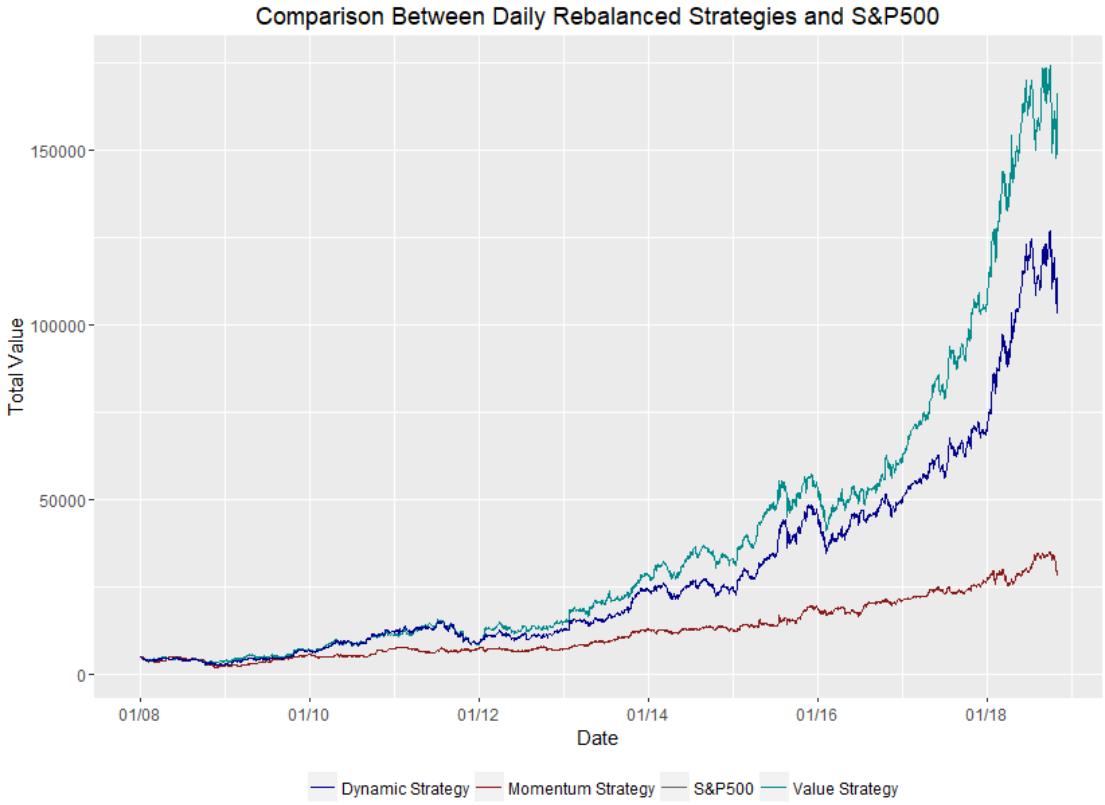
Weights of Each Stock in Different Strategies, Rebance Period: 1 Month

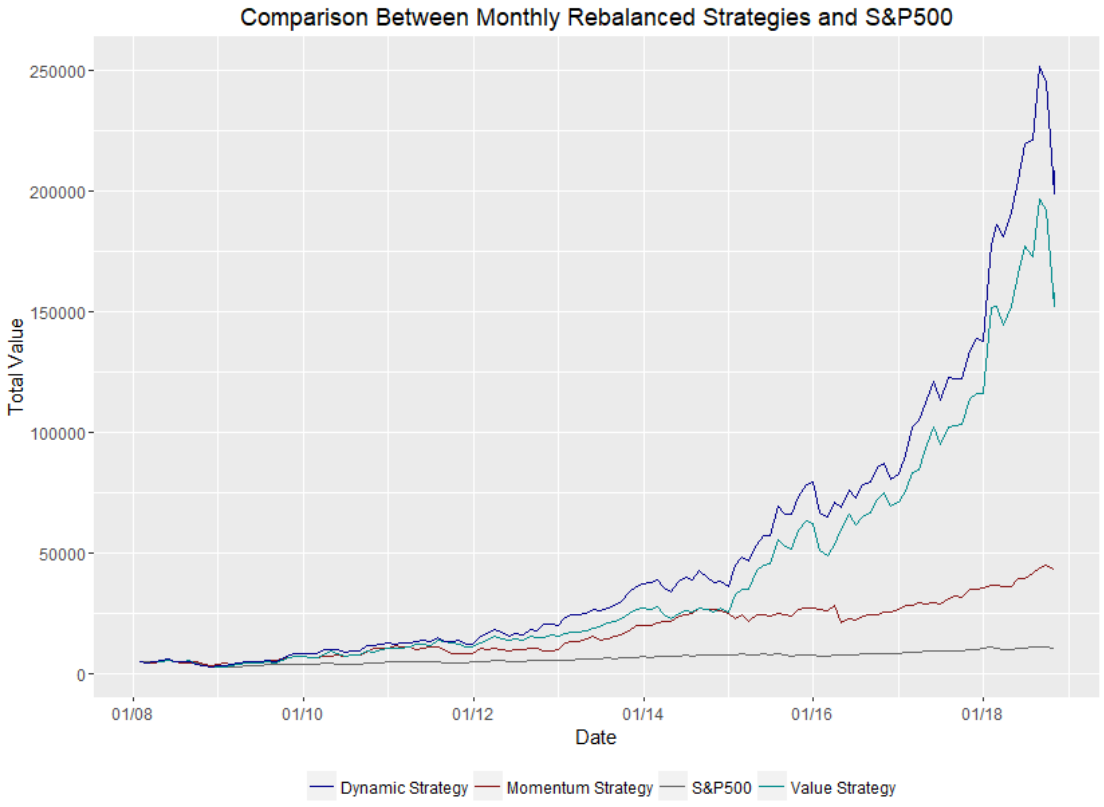






IV. Total Capital of Momentum/ Value/ Dynamic Strategy





V. 99% VaR and 97.5% ES of Momentum/ Value/ Dynamic Strategy

99% VaR and 97.5% Expected Shortfall for Daily and Monthly Rebalanced Momentum Strategy								
	Non-linear Loss Distribution Function				Linear Loss Distribution Function			
Daily	Normal Model		Historical Simulation		Normal Model		Historical Simulation	
	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES
	$1,159.46	$1,165.14	$1,442.13	$1,553.43	$1,183.33	$1,189.24	$1,479.30	$1,602.12
Monthly	Normal Model		Historical Simulation		Normal Model		Historical Simulation	
	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES
	$6,144.17	$6,175.20	$7,704.20	$7,714.21	$6,626.08	$6,662.25	$8,483.59	$8,570.11

99% VaR and 97.5% Expected Shortfall for Daily and Monthly Rebalanced Value Strategy								
	Non-linear Loss Distribution Function				Linear Loss Distribution Function			
Daily	Normal Model		Historical Simulation		Normal Model		Historical Simulation	
	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES
	$8,033.17	$8,072.75	$10,162.55	$10,392.24	$8,233.71	$8,275.30	$10,486.42	$10,775.24
Monthly	Normal Model		Historical Simulation		Normal Model		Historical Simulation	
	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES
	$26,878.51	$27,015.92	$32,606.29	$35,222.51	$29,466.64	$29,632.36	$36,526.26	$40,183.47

99% VaR and 97.5% Expected Shortfall for Daily and Monthly Rebalanced Dynamic Strategy								
	Non-linear Loss Distribution Function				Linear Loss Distribution Function			
Daily	Normal Model		Historical Simulation		Normal Model		Historical Simulation	
	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES
	$5,462.59	$5,489.44	$6,375.57	$6,705.50	$5,598.18	$5,626.38	$6,561.36	$6,935.27
Monthly	Normal Model		Historical Simulation		Normal Model		Historical Simulation	
	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES	99% VaR	97.5% ES
	$34,210.46	$34,388.75	$37,460.15	$41,534.63	$37,371.29	$37,584.73	$41,318.91	$46,853.94

R Code

```
rm(list=ls())
#=============
#part a#
#=============
#import some library:
setwd("D:/Desktop/LSE/course/ST429-MT/project")
library(tseries)
library(timeSeries)
library(ggplot2)
library(xts)
library(QRM)
library(gridExtra)
library(evd)

#import and normalise the dataset
portfolio=read.csv("combined file.csv",header=T)
names(portfolio)=c("Date","AAPL_price","AMZN_price","GOOG_price","MSFT_price","NFLX_price","S&P500_price")
#transform portfolio into a timeseries
portfolio.ts <- xts(portfolio[,-1], as.Date(portfolio$Date, format="%d/%m/%Y"))
#calculate the log return
log_return=diff(log(portfolio.ts))

#plot the log return
stock_list=c("AAPL","AMZN","GOOG","MSFT","NFLX","S&P500")
stock_number=length(stock_list)
```

```
26. 
27. par(mfrow=c(3,2),oma=c(0,0,4,0))
28. 
29. for (i in 1:stock_number){
30.   plot.zoo(log_return[,i],xlab="Dates",ylab=paste(stock_list[i], "log retun",seq=" "),main=paste("Log Return of",stock_list[i],"From 2006 to 2018",seq= " "),type="l",ylim=c(-0.4,0.2))
31.   abline(h=(mean(as.numeric(log_return[-1,i]))-3*sqrt(var(log_return[-1,i]))),col="red",lty=2)
32.   abline(h=(mean(as.numeric(log_return[-1,i]))+3*sqrt(var(log_return[-1,i]))),col="red",lty=2)
33. }
34. mtext(side=3,line=0,cex=1.5,outer=T, "Line Charts for Log Return")
35. 
36. #plot the adjusted price to view special event:
37. for (i in 1:stock_number){
38.   plot.zoo(portfolio.ts[,i],xlab="Dates",ylab=paste(stock_list[i], "stock price",seq=" "),main=paste("Adjusted Stock Price of",stock_list[i],"From 2006 to 2018",seq= " "),type="l")
39. }
40. mtext(side=3,line=0,cex=1.5,outer=T, "Line Charts for Stock Price")
41. par(mfrow=c(1,1))
42. 
43. #==============
44. #part b#
45. #==============
46. #use 2006/01/01 - 2007/12/31 as trainning data:
47. start_date="2006-01-03"
48. end_date="2018-11-01"
49. cut_date_1="2007-12-31"
50. cut_date_2="2008-01-01"
51. portfolio_training <- window(portfolio.ts, start=start_date, end=cut_date_1)
52. 
53. #~~~~~~~~~~~~~~~~~~~~~~~~~
54. #some common functions:
55. #~~~~~~~~~~~~~~~~~~~~~~~~~
56. ##########common function for calculating coefficients of garch(1,1) model###########
57. garch_coeff=function(data){
58.   stock_list=c("AAPL","AMZN","GOOG","MSFT","NFLX","S&P500")
59.   garch_model=matrix(0,nrow=6,ncol=3)
60.   for (i in 1:6){
61.     garch_model[i,]=garch(data[-1,i],order=c(1,1))$coef
62.   }
63.   colnames(garch_model)=c("w","a","b")
64.   rownames(garch_model)=stock_list
65.   return(garch_model)
66. }
67. 
68. ##########common function for calculating volotility from garch(1,1) model###########
69. volatility=function(garch_model,log_return){
70.   volatility_result=log_return
71.   stock_list=c("AAPL","AMZN","GOOG","MSFT","NFLX","S&P500")
72.   for (i in 1:6){
73.     volatility_result[2,i]=garch_model[i,1]/(1-garch_model[i,2]-garch_model[i,3])
74.     for (j in 3:nrow(volatility_result)){
75.       volatility_result[j,i]=garch_model[i,1]+garch_model[i,2]*volatility_result[j-1,i]+garch_model[i,3]*log_return[j-1,i]^2
76.     }
77.     names(volatility_result)[i]=paste0(stock_list[i],"_volatility")
78.   }
79.   return(volatility_result)
80. }
81. 
82. ##########common function to calculate long-term mean###########
83. long_term_mean=function(data,moving_window){
84.   stock_list=c("AAPL","AMZN","GOOG","MSFT","NFLX","S&P500")
```

```
  long_mean=data
  long_mean[,]=NA
  for (i in moving_window:nrow(long_mean)){
    for (j in 1:6) {
      long_mean[i,j]=mean(data[(i+1-moving_window):i,j])
    }
  }
  colnames(long_mean)=paste("mean",stock_list)
  return(long_mean)
}

##########common function to calculate weights in value strategy###########
weight_value=function(data_traininig,data_testing,data_full,long_mean,moving_window,volatility_result){
  stock_list=c("AAPL","AMZN","GOOG","MSFT","NFLX","S&P500")
  number_training=nrow(data_traininig)
    weight_result=data_testing[,1:5]
    weight_result[,]=NA
    colnames(weight_result)=paste0(stock_list[1:5],"_weight")

    for (i in 2:nrow(weight_result)){
      #calculate the covariance matrix of the 5 stocks at each node using the correlation and the estimated volatility from garch(1,1)
      covariance_matrix_temp=matrix(nrow=5,ncol=5)
      for (j in 1:5){
        for (k in 1:5){
          covariance_matrix_temp[j,k]=sqrt(volatility_result[i+number_training,j])*cor(data_full[(i+number_training-moving_window):(i+number_training),1:5])[j,k]*sqrt(volatility_result[i+number_training,k])
        }
      }
      #(long-run mean of ith node - realised log return of (i-1)th node will be a good estimate of return for the ith node
      #we use portfolio.optim function to estimate the optimal weights for each node using the previous 250/52/12 datapoints. shorts=True means short seeling is allowed in this strategy. reslow and reshigh defined the lower bound and upper bound of the weights.
      weight_result[i,]=0.4-unlist(portfolio.optim(x=matrix((as.numeric(long_mean[i+number_training,1:5])-as.numeric(data_testing[i-1,1:5])),ncol=5),shorts=TRUE,reslow=rep(-1,5),reshigh = rep(1,5),covmat=covariance_matrix_temp)[1])
    }
    weight_result[1,]=rep(0.2,5) #for the first day of testing,the weight of each stock should be same
    return(weight_result)
  }


  ##########common function to calculate weights in momentum strategy###########
  weight_momentum=function(data_traininig,data_testing,data_full,moving_window,volatility_result){
    stock_list=c("AAPL","AMZN","GOOG","MSFT","NFLX","S&P500")
    number_training=nrow(data_traininig)
    weight_result=data_testing[,1:5]
    weight_result[,]=NA
    colnames(weight_result)=paste0(stock_list[1:5],"_weight")

    for (i in 2:nrow(weight_result)){
      covariance_matrix_temp=matrix(nrow=5,ncol=5)
      for (j in 1:5){
        for (k in 1:5){
          covariance_matrix_temp[j,k]=sqrt(volatility_result[i+number_training,j])*cor(data_full[(i+number_training-moving_window):(i+number_training),1:5])[j,k]*sqrt(volatility_result[i+number_training,k])
```

```
        }
      }
      weight_result[i,]=unlist(portfolio.optim(x=data_testing[i-
  1,1:5],shorts=TRUE,reslow=rep(-
  1,5),reshigh = rep(1,5),covmat=covariance_matrix_temp)[1])
    }
    weight_result[1,]=rep(0.2,5) #for the first day of testing,the weight of each sto
  ck should be same
    return(weight_result)
  }


  ##########common function for wealth update###########
  capital_update=function(stock_price,weight,initial_capital){
    capital=stock_price[,1]
    capital[,]=NA
    capital[1,1]=initial_capital
    individual_wealth=stock_price
    individual_wealth[,]=NA
    individual_wealth[1,]=initial_capital*weight[1,]

    for (i in 2:nrow(capital)){
      for (j in 1:ncol(stock_price)){
        individual_wealth[i,j]=as.numeric(capital[i-1,1])*as.numeric(weight[i-
  1,j])*as.numeric(stock_price[i,j])/as.numeric(stock_price[i-1,j])
      }
      capital[i,]=sum(individual_wealth[i,])
    }
    return(capital)
  }

  ##########common function to calculate linearised loss###########
  loss.fn <- function(x,weights,value, linear=FALSE){
    if ((!(is.vector(x))) & (!(is.matrix(x))))
      stop("x must be vector or matrix with rows corresponding to risk
           factor return observations")
    N <- length(weights)
    T <- 1
    if (is.matrix(x)) T <- dim(x)[1]
    weight.mat <- matrix(weights,nrow=T,ncol=N,byrow=TRUE)
    tmp.mat <- value * weight.mat
    if (linear) { summand <- x*tmp.mat } else { summand <- (exp(x)-1)*tmp.mat }
    loss <- -rowSums(summand)
    return(loss)
  }
  #-----------------------------------------
  #rebalance period = daily
  #-----------------------------------------
  log_return_daily_training=window(log_return,start=start_date,end=cut_date_1)
  log_return_daily_testing=window(log_return,start=cut_date_2,end=end_date)
  moving_window_daily=250

  ##########estiate the volatility of each day##########
  garch_model_daily=garch_coeff(log_return_daily_training)

  #use garch(1,1) coefficients to estimate volitility at each node:
  volatility_daily=volatility(garch_model_daily,log_return)

  #######value strategy#######
  ##long-term mean##  update daily
  long_mean_daily=long_term_mean(log_return,250)

  #####calculate daily weights######
  weights_value_daily=weight_value(log_return_daily_training,log_return_daily_testing
  ,log_return,long_mean_daily,250,volatility_daily)

```

```
195.    ####calculate total wealth at the end of each day for Value Trading Strategy
196.    stock_price_daily=portfolio.ts[index(log_return),]
197.    stock_price_daily_testing=window(stock_price_daily,start=cut_date_2,end=end_date)
198.    capital_value_daily=capital_update(stock_price_daily_testing[,1:5],weights_value_da
    ily,5000)
199.    colnames(capital_value_daily)="capital_value_daily"
200.
201.
202.
203.    ######### momentum strategy:daily #################
204.    #calculate weights for each stocks:
205.    weights_momentum_daily=weight_momentum(log_return_daily_training,log_return_daily_t
    esting,log_return,250,volatility_daily)
206.
207.    ####calculate total wealth at the end of each day
208.    capital_momentum_daily=capital_update(stock_price_daily_testing[,1:5],weights_momen
    tum_daily,5000)
209.    colnames(capital_momentum_daily)="capital_momentum_daily"
210.
211.    #-----------------------------------------
212.    #rebalance period = 1 week
213.    #-----------------------------------------
214.    log_return_weekly=apply.weekly(log_return, function(x) apply(x, 2, sum))
215.    log_return_weekly_training=window(log_return_weekly,start=start_date,end=cut_date_1
    )
216.    log_return_weekly_testing=window(log_return_weekly,start=cut_date_2,end=end_date)
217.    moving_window_weekly<-52
218.
219.    ##########estiate the volatility of each stock at each weekly node##########
220.    garch_model_weekly=garch_coeff(log_return_weekly_training)
221.
222.    #use garch(1,1) coefficients to estimate volitility at each node:
223.    #estimate v0=w/(1-a-b), i.e long-run mean
224.    volatility_weekly=volatility(garch_model_weekly,log_return_weekly)
225.
226.    ######### momentum strategy #################
227.    #calculate weights for each stocks:
228.    weights_momentum_weekly=weight_momentum(log_return_weekly_training,log_return_weekl
    y_testing,log_return_weekly,52,volatility_weekly)
229.
230.
231.    ####calculate total wealth at the end of each day
232.    stock_price_weekly=portfolio.ts[index(log_return_weekly),]
233.    stock_price_weekly_testing=window(stock_price_weekly,start=cut_date_2,end=end_date)

234.    capital_momentum_weekly=capital_update(stock_price_weekly_testing[,1:5],weights_mom
    entum_weekly,5000)
235.    colnames(capital_momentum_weekly)="capital_momentum_weekly"
236.
237.
238.    #######value strategy#######
239.    ##long-term mean##
240.    long_mean_weekly=long_term_mean(log_return_weekly,52)
241.
242.    #####calculate weights######
243.    weights_value_weekly=weight_value(log_return_weekly_training,log_return_weekly_test
    ing,log_return_weekly,long_mean_weekly,52,volatility_weekly)
244.
245.    ####calculate total wealth at the end of each day
246.    capital_value_weekly=capital_update(stock_price_weekly_testing[,1:5],weights_value_
    weekly,5000)
247.    colnames(capital_value_weekly)="capital_value_weekly"
248.
249.
250.    #-----------------------------------------
251.    #rebalance period = 1 month
```

```
252.    #-----------------------------------------
253.    log_return_monthly=apply.monthly(log_return, function(x) apply(x, 2, sum))
254.    log_return_monthly_training=window(log_return_monthly,start=start_date,end=cut_date
    _1)
255.    log_return_monthly_testing=window(log_return_monthly,start=cut_date_2,end=end_date)

256.    moving_window_monthly<-12
257.
258.
259.    ##########estiate the volatility of each stock at each monthly node##########
260.    #use the monthly log return of training dataset to estimate garch (1,1) for each st
    ock and store the coefficients in the garch_model_monthly matrix
261.    garch_model_monthly=garch_coeff(log_return_monthly_training)
262.
263.    #use garch(1,1) coefficients to estimate volitility at each node:
264.    #estimate v0=w/(1-a-b), i.e long-run mean
265.    volatility_monthly=volatility(garch_model_monthly,log_return_monthly)
266.
267.    #######value strategy#######
268.    ##long-term mean##
269.    long_mean_monthly=long_term_mean(log_return_monthly,12)
270.
271.    #####calculate weights######
272.    weights_value_monthly=weight_value(log_return_monthly_training,log_return_monthly_t
    esting,log_return_monthly,long_mean_monthly,12,volatility_monthly)
273.
274.
275.    ####calculate total wealth at the end of each month
276.    stock_price_monthly=portfolio.ts[index(log_return_monthly),]
277.    stock_price_monthly_testing=window(stock_price_monthly,start=cut_date_2,end=end_dat
    e)
278.    capital_value_monthly=capital_update(stock_price_monthly_testing[,1:5],weights_valu
    e_monthly,5000)
279.    colnames(capital_value_monthly)="capital_value_monthly"
280.
281.
282.    ######### momentum strategy ##################
283.    #calculate weights for each stocks:
284.    weights_momentum_monthly=weight_momentum(log_return_monthly_training,log_return_mon
    thly_testing,log_return_monthly,12,volatility_monthly)
285.
286.    ####calculate total wealth at the end of each month
287.    capital_momentum_monthly=capital_update(stock_price_monthly_testing[,1:5],weights_m
    omentum_monthly,5000)
288.    colnames(capital_momentum_monthly)="capital_momentum_monthly"
289.
290.
291.    ######## compare the performance of these two portfolio and the SP500 ########
292.    # daily
293.    portfolio_daily_testing<-window(portfolio.ts, start=cut_date_2, end=end_date)
294.    SP500_daily<-portfolio_daily_testing$`S&P500_price`*(5000/1268)
295.    daily_comparison<-
    xts(cbind(SP500_daily,capital_momentum_daily,capital_value_daily))
296.    colnames(daily_comparison)<-c("S&P500","Momentum Strategy","Value Strategy")
297.
298.    daily.plot<-ggplot(daily_comparison,aes(x = index(daily_comparison)))+
299.      geom_line(aes(y = daily_comparison[,1], color = "S&P500"))+
300.      ggtitle("Comparison Between Daily Rebalanced Strategies and S&P500") +
301.      geom_line(aes(y = daily_comparison[,2], color = "Momentum Strategy")) +
302.      geom_line(aes(y = daily_comparison[,3], color = "Value Strategy")) +
303.      xlab("Date") + ylab("Total Value") +
304.      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
305.      scale_x_date(date_labels = "%m/%y") +
306.      scale_colour_manual("", values=c("S&P500"="gray40", "Momentum Strategy"="firebric
    k4", "Value Strategy"="darkcyan"))+
307.      theme(legend.position="bottom")
```

```
308.
309.
310.    # weekly
311.    SP500_weekly<-to.weekly(portfolio_daily_testing$`S&P500_price`)[,4]*(5000/1268)
312.    weekly_comparison<-
    xts(cbind(SP500_weekly,capital_momentum_weekly,capital_value_weekly))
313.    colnames(weekly_comparison)<-c("S&P500","Momentum Strategy","Value Strategy")
314.    weekly.plot<-ggplot(weekly_comparison,aes(x = index(weekly_comparison)))+
315.      geom_line(aes(y = weekly_comparison[,1], color = "S&P500"))+
316.      ggtitle("Comparison Between Weekly Rebalanced Strategies and S&P500") +
317.      geom_line(aes(y = weekly_comparison[,2], color = "Momentum Strategy")) +
318.      geom_line(aes(y = weekly_comparison[,3], color = "Value Strategy")) +
319.      xlab("Date") + ylab("Total Value") +
320.      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
321.      scale_x_date(date_labels = "%m/%y") +
322.      scale_colour_manual("", values=c("S&P500"="gray40", "Momentum Strategy"="firebric
    k4", "Value Strategy"="darkcyan"))+
323.      theme(legend.position="bottom")
324.
325.    # monthly
326.    SP500_monthly<-to.monthly(portfolio_daily_testing$`S&P500_price`)[,4]*(5000/1268)
327.    colnames(SP500_monthly)<-"S&P500"
328.    colnames(capital_momentum_monthly[,1])<-"Momentum Strategy"
329.    colnames(capital_value_monthly[,1])<-"Value Strategy"
330.
331.    monthly.plot<-ggplot(SP500_monthly,aes(x = index(capital_momentum_monthly)))+
332.      geom_line(aes(y =SP500_monthly, color = "S&P500"))+
333.      ggtitle("Comparison Between Monthly Rebalanced Strategies and S&P500") +
334.      geom_line(aes(y =capital_momentum_monthly[,1], color = "Momentum Strategy")) +
335.      geom_line(aes(y = capital_value_monthly[,1], color = "Value Strategy")) +
336.      xlab("Date") + ylab("Total Value") +
337.      scale_x_date(date_labels = "%m/%y") +
338.      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
339.      scale_colour_manual("", values=c("S&P500"="gray40", "Momentum Strategy"="firebric
    k4", "Value Strategy"="darkcyan"))+
340.      theme(legend.position="bottom")
341.
342.
343.
344.    #===============
345.    #part c
346.    #===============
347.
348.    #---------------------------
349.    # Daily Rebalance
350.    #---------------------------
351.    ##use AR(1) model to predict the return.
352.    number_training_daily<-nrow(log_return_daily_training)
353.    ar_model_daily=log_return_daily_testing[,1:5]
354.    ar_model_daily[,]=NA
355.    for (i in 1:nrow(ar_model_daily)){
356.      for (j in 1:(stock_number-1)){
357.        ar_model_daily[i,j]=arima(log_return[(number_training_daily+i-1-
    moving_window_daily):(number_training_daily+i-1),j],order=c(1,0,0))$coef[1]
358.      }
359.    }
360.
361.    #use the coefficient to predict the log_return at the next datapoint
362.    log_return_daily_ac=log_return_daily_testing[,1:5]
363.    log_return_daily_ac[,]=NA
364.    for (i in 1:nrow(log_return_daily_ac)){
365.      for (j in 1:(stock_number-1)){
366.        log_return_daily_ac[i,j]=as.numeric(ar_model_daily[i,j])*as.numeric(log_return[
    (i+number_training_daily-1),j])
367.      }
368.    }
```

```
369.
370.    ##calcualte the weights
371.    weights_ac_daily=log_return_daily_testing[,1:5]
372.    weights_ac_daily[,]=NA
373.    colnames(weights_ac_daily)=paste0(stock_list[1:5],"_weight")
374.
375.    for (i in 2:nrow(weights_ac_daily)){
376.      covariance_matrix_temp=matrix(nrow=5,ncol=5)
377.      for (j in 1:5){
378.        for (k in 1:5){
379.          covariance_matrix_temp[j,k]=sqrt(volatility_daily[i+number_training_daily,j])
    *cor(log_return[(i+number_training_daily-
    moving_window_daily):(i+number_training_daily),1:5])[j,k]*sqrt(volatility_daily[i+numbe
    r_training_daily,k])
380.        }
381.      }
382.      weights_ac_daily[i,]=0.4-unlist(portfolio.optim(log_return_daily_ac[i-
    1,1:5],shorts=TRUE,reslow=rep(-
    1,5),reshigh = rep(1,5),covmat=covariance_matrix_temp)[1])
383.      rm(covariance_matrix_temp)
384.    }
385.    weights_ac_daily[1,]=rep(0.2,5)
386.
387.    ####calculate total wealth at the end of each day
388.    capital_ac_daily=capital_update(stock_price_daily_testing[,1:5],weights_ac_daily,50
    00)
389.    colnames(capital_ac_daily)="capital_autocorrelation_daily"
390.
391.
392.    #========================
393.    #Rebalance: Weekly
394.    #========================
395.    ##use AR(1) model to predict the return. store the coefficents in the "ar_model_wee
    kly" matrix. The coefficients are updated in a rolling window=52.
396.    number_training_weekly<-nrow(log_return_weekly_training)
397.    ar_model_weekly=log_return_weekly_testing[,1:5]
398.    ar_model_weekly[,]=NA
399.    for (i in 1:nrow(ar_model_weekly)){
400.      for (j in 1:(stock_number-1)){
401.        ar_model_weekly[i,j]=arima(log_return_weekly[(number_training_weekly+i-1-
    moving_window_weekly):(number_training_weekly+i-1),j],order=c(1,0,0))$coef[1]
402.      }
403.    }
404.
405.    #use the coefficient to predict the log_return at the next datapoint
406.    log_return_weekly_ac=log_return_weekly_testing[,1:5]
407.    log_return_weekly_ac[,]=NA
408.    for (i in 1:nrow(log_return_weekly_ac)){
409.      for (j in 1:(stock_number-1)){
410.        log_return_weekly_ac[i,j]=as.numeric(ar_model_weekly[i,j])*as.numeric(log_retur
    n_weekly[(i+number_training_weekly-1),j])
411.      }
412.    }
413.    ##calcualte the weights
414.    weights_ac_weekly=log_return_weekly_testing[,1:5]
415.    weights_ac_weekly[,]=NA
416.    colnames(weights_ac_weekly)=paste0(stock_list[1:5],"_weight")
417.
418.    for (i in 2:nrow(weights_ac_weekly)){
419.      covariance_matrix_temp=matrix(nrow=5,ncol=5)
420.      for (j in 1:5){
421.        for (k in 1:5){
422.          covariance_matrix_temp[j,k]=sqrt(volatility_weekly[i+number_training_weekly,j
    ])*cor(log_return_weekly[(i+number_training_weekly-
    moving_window_weekly):(i+number_training_weekly),1:5])[j,k]*sqrt(volatility_weekly[i+nu
    mber_training_weekly,k])
```

```
423.       }
424.     }
425.     weights_ac_weekly[i,]=0.4-unlist(portfolio.optim(log_return_weekly_ac[i-
    1,1:5],shorts=TRUE,reslow=rep(-
    1,5),reshigh = rep(1,5),covmat=covariance_matrix_temp)[1])
426.     rm(covariance_matrix_temp)
427.   }
428.   weights_ac_weekly[1,]=rep(0.2,5)
429.
430.   ####calculate total wealth at the end of each week
431.   capital_ac_weekly=capital_update(stock_price_weekly_testing[,1:5],weights_ac_weekly
    ,5000)
432.   colnames(capital_ac_weekly)="capital_autocorrelation_weekly"
433.
434.
435.   #-----------------------
436.   #Rebalance: Monthly
437.   #-----------------------
438.   ##use AR(1) model to predict the return. store the coefficents in the "ar_model_mon
    thly" matrix. The coefficients are updated in a rolling window=12.
439.   number_training_monthly<-nrow(log_return_monthly_training)
440.   ar_model_monthly=log_return_monthly_testing[,1:5]
441.   ar_model_monthly[,]=NA
442.   for (i in 1:nrow(ar_model_monthly)){
443.     for (j in 1:(stock_number-1)){
444.       ar_model_monthly[i,j]=arima(log_return_monthly[(number_training_monthly+i-1-
    moving_window_monthly):(number_training_monthly+i-1),j],order=c(1,0,0))$coef[1]
445.     }
446.   }
447.
448.   #use the coefficient to predict the log_return at the next datapoint
449.   log_return_monthly_ac=log_return_monthly_testing[,1:5]
450.   log_return_monthly_ac[,]=NA
451.   for (i in 1:nrow(log_return_monthly_ac)){
452.     for (j in 1:(stock_number-1)){
453.       log_return_monthly_ac[i,j]=as.numeric(ar_model_monthly[i,j])*as.numeric(log_ret
    urn_monthly[(i+number_training_monthly-1),j])
454.     }
455.   }
456.   ##calcualte the weights
457.   weights_ac_monthly=log_return_monthly_testing[,1:5]
458.   weights_ac_monthly[,]=NA
459.   colnames(weights_ac_monthly)=paste0(stock_list[1:5],"_weight")
460.
461.   for (i in 2:nrow(weights_ac_monthly)){
462.     covariance_matrix_temp=matrix(nrow=5,ncol=5)
463.     for (j in 1:5){
464.       for (k in 1:5){
465.         covariance_matrix_temp[j,k]=sqrt(volatility_monthly[i+number_training_monthly
    ,j])*cor(log_return_monthly[(i+number_training_monthly-
    moving_window_monthly):(i+number_training_monthly),1:5])[j,k]*sqrt(volatility_monthly[i
    +number_training_monthly,k])
466.       }
467.     }
468.     weights_ac_monthly[i,]=0.4-unlist(portfolio.optim(log_return_monthly_ac[i-
    1,1:5],shorts=TRUE,reslow=rep(-
    1,5),reshigh = rep(1,5),covmat=covariance_matrix_temp)[1])
469.     rm(covariance_matrix_temp)
470.   }
471.   weights_ac_monthly[1,]=rep(0.2,5)
472.
473.   ####calculate total wealth at the end of each day
474.   capital_ac_monthly=capital_update(stock_price_monthly_testing[,1:5],weights_ac_mont
    hly,5000)
475.   colnames(capital_ac_monthly)="capital_autocorrelation_monthly"
476.
```

```
477.
478.    ######## compare the performance of dynamic portfolio and the SP500 ########
479.
480.    # daily
481.    SP500_daily<-to.weekly(portfolio_daily_testing$`S&P500_price`)[,4]*(5000/1268)
482.    daily_ac.comparison<-xts(cbind(SP500_daily,capital_ac_daily[,1]))
483.    colnames(daily_ac.comparison)<-c("S&P500","Autocorrelation Trading Strategy")
484.    daily.ac.plot<-ggplot(daily_ac.comparison,aes(x = index(daily_comparison)))+
485.      geom_line(aes(y = daily_ac.comparison[,1], color = "S&P500"))+
486.      ggtitle("Comparison Between Daily Rebalanced Autocorrelation Strategy and S&P500"
    ) +
487.      geom_line(aes(y = daily_ac.comparison[,2], color = "Autocorrelation Trading Strat
    egy")) +
488.      xlab("Date") + ylab("Total Value") +
489.      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
490.      scale_x_date(date_labels = "%m/%y") +
491.      scale_colour_manual("", values=c("S&P500"="gray40", "Autocorrelation Trading Stra
    tegy"="firebrick4"))+
492.      theme(legend.position="bottom")
493.
494.
495.    # weekly
496.    SP500_weekly<-to.weekly(portfolio_daily_testing$`S&P500_price`)[,4]*(5000/1268)
497.    weekly_ac.comparison<-xts(cbind(SP500_weekly,capital_ac_weekly[,1]))
498.    colnames(weekly_ac.comparison)<-c("S&P500","Autocorrelation Trading Strategy")
499.    weekly.ac.plot<-
    ggplot(weekly_ac.comparison,aes(x = index(weekly_ac.comparison)))+
500.      geom_line(aes(y = weekly_ac.comparison[,1], color = "S&P500"))+
501.      ggtitle("Comparison Between Weekly Rebalanced Autocorrelation Strategy and S&P500
    ") +
502.      geom_line(aes(y = weekly_ac.comparison[,2], color = "Autocorrelation Trading Stra
    tegy")) +
503.      xlab("Date") + ylab("Total Value") +
504.      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
505.      scale_x_date(date_labels = "%m/%y") +
506.      scale_colour_manual("", values=c("S&P500"="gray40", "Autocorrelation Trading Stra
    tegy"="firebrick4"))+
507.      theme(legend.position="bottom")
508.
509.    # monthly
510.    SP500_monthly<-to.monthly(portfolio_daily_testing$`S&P500_price`)[,4]*(5000/1268)
511.    colnames(SP500_monthly)<-"S&P500"
512.    colnames(capital_ac_monthly[,1])<-"Autocorrelation Trading Strategy"
513.
514.    monthly.ac.plot<-ggplot(SP500_monthly,aes(x = index(capital_ac_monthly)))+
515.      geom_line(aes(y =SP500_monthly, color = "S&P500"))+
516.      ggtitle("Comparison Between Monthly Rebalanced Autocorrelation Strategy and S&P50
    0") +
517.      geom_line(aes(y =capital_ac_monthly[,1], color = "Autocorrelation Trading Strateg
    y")) +
518.      xlab("Date") + ylab("Total Value") +
519.      scale_x_date(date_labels = "%m/%y") +
520.      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
521.      scale_colour_manual("", values=c("S&P500"="gray40", "Autocorrelation Trading Stra
    tegy"="firebrick4"))+
522.      theme(legend.position="bottom")
523.
524.    # Compare the performance of 3 strategies with S&P500
525.
526.    # Daily
527.    daily_comparison<-
    xts(cbind(SP500_daily,capital_momentum_daily,capital_value_daily,capital_ac_daily[,1]))

528.    colnames(daily_comparison)<-
    c("S&P500","Momentum Strategy","Value Strategy","Dynamic Strategy")
529.    daily.plot.comparison<-ggplot(daily_comparison,aes(x = index(daily_comparison)))+
```

```
      geom_line(aes(y = daily_comparison[,1], color = "S&P500"))+
      ggtitle("Comparison Between Daily Rebalanced Strategies and S&P500") +
      geom_line(aes(y = daily_comparison[,2], color = "Momentum Strategy")) +
      geom_line(aes(y = daily_comparison[,3], color = "Value Strategy")) +
      geom_line(aes(y = daily_comparison[,4], color = "Dynamic Strategy"))+
      xlab("Date") + ylab("Total Value") +
      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
      scale_x_date(date_labels = "%m/%y") +
      scale_colour_manual("", values=c("S&P500"="gray40", "Momentum Strategy"="firebric
    k4", "Value Strategy"="darkcyan","Dynamic Strategy"="darkblue"))+
      theme(legend.position="bottom")

    # weekly
    SP500_weekly<-to.weekly(portfolio_daily_testing$`S&P500_price`)[,4]*(5000/1268)
    weekly_comparison<-
    xts(cbind(SP500_weekly,capital_momentum_weekly,capital_value_weekly,capital_ac_weekly))

    colnames(weekly_comparison)<-
    c("S&P500","Momentum Strategy","Value Strategy","Dynamic Strategy")
    weekly.plot.comparison<-
    ggplot(weekly_comparison,aes(x = index(weekly_comparison)))+
      geom_line(aes(y = weekly_comparison[,1], color = "S&P500"))+
      ggtitle("Comparison Between Weekly Rebalanced Strategies and S&P500") +
      geom_line(aes(y = weekly_comparison[,2], color = "Momentum Strategy")) +
      geom_line(aes(y = weekly_comparison[,3], color = "Value Strategy")) +
      geom_line(aes(y = weekly_comparison[,4], color = "Dynamic Strategy")) +
      xlab("Date") + ylab("Total Value") +
      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
      scale_x_date(date_labels = "%m/%y") +
      scale_colour_manual("", values=c("S&P500"="gray40", "Momentum Strategy"="firebric
    k4", "Value Strategy"="darkcyan","Dynamic Strategy"="darkblue"))+
      theme(legend.position="bottom")

    # monthly
    SP500_monthly<-to.monthly(portfolio_daily_testing$`S&P500_price`)[,4]*(5000/1268)
    colnames(SP500_monthly)<-"S&P500"
    colnames(capital_momentum_monthly[,1])<-"Momentum Strategy"
    colnames(capital_value_monthly[,1])<-"Value Strategy"
    colnames(capital_ac_monthly[,1])<-"Dynamic Strategy"
    monthly.plot.comparison<-
    ggplot(SP500_monthly,aes(x = index(capital_momentum_monthly)))+
      geom_line(aes(y =SP500_monthly, color = "S&P500"))+
      ggtitle("Comparison Between Monthly Rebalanced Strategies and S&P500") +
      geom_line(aes(y =capital_momentum_monthly[,1], color = "Momentum Strategy")) +
      geom_line(aes(y = capital_value_monthly[,1], color = "Value Strategy")) +
      geom_line(aes(y = capital_ac_monthly[,1], color = "Dynamic Strategy")) +
      xlab("Date") + ylab("Total Value") +
      scale_x_date(date_labels = "%m/%y") +
      theme(plot.title = element_text(hjust = 0.5), panel.border = element_blank()) +
      scale_colour_manual("", values=c("S&P500"="gray40", "Momentum Strategy"="firebric
    k4", "Value Strategy"="darkcyan","Dynamic Strategy"="darkblue"))+
      theme(legend.position="bottom")

    #-----------------------------------------
    #99% VaR and 97.5% ES for part (b) and (c)
    #-----------------------------------------

    ############### Daily:ES and VaR of Value strategy for normal and historical model #
    ###############

    #  Normal model

    capital.Value_lastday_daily<-capital_value_daily[nrow(capital_value_daily),]
    loss_return_daily.value.strategy<--diff(log(capital_value_daily))[-1,]
    mu.return.daily<-mean(as.numeric(loss_return_daily.value.strategy))
    variance.return_daily<-var(as.numeric(loss_return_daily.value.strategy))
```

```
VaR.return_daily.normal<- mu.return.daily + sqrt(variance.return_daily) * qnorm(0.99)
VaR.return_value.normal.nonlinear.daily=capital.Value_lastday_daily*(1-exp(-VaR.return_daily.normal))
VaR.return_value.normal.linear.daily=capital.Value_lastday_daily*VaR.return_daily.normal
ES.return_daily.normal<- mu.return.daily + sqrt(variance.return_daily) * dnorm(qnorm(0.975))/(1-0.975)
ES.return_daily.normal.nonlinear=capital.Value_lastday_daily*(1-exp(-ES.return_daily.normal))
ES.return_daily.normal.linear=capital.Value_lastday_daily*ES.return_daily.normal

#  Historical simulation

# non-linear
his_return_daily.value<-1-exp(-loss_return_daily.value.strategy)
VaR.hs.daily_value.nonlinear <- quantile(his_return_daily.value,0.99)*capital.Value_lastday_daily
VaR.hs.daily_value_0.975 <- quantile(his_return_daily.value,0.975)
ES.hs.daily_value.nonlinear <- mean(his_return_daily.value[his_return_daily.value > VaR.hs.daily_value_0.975])*capital.Value_lastday_daily
# linear
VaR.hs.daily_value.linear <- quantile(loss_return_daily.value.strategy,0.99)*capital.Value_lastday_daily
VaR.hs.daily_value_0.975 <- quantile(loss_return_daily.value.strategy,0.975)
ES.hs.daily_value.linear <- mean(loss_return_daily.value.strategy[loss_return_daily.value.strategy > VaR.hs.daily_value_0.975])*capital.Value_lastday_daily


############### Weekly:ES and VaR of Value strategy for normal and historical model #################
#  Normal model
capital.Value_lastday_weekly<-capital_value_weekly[nrow(capital_value_weekly),]
loss_return_weekly.value.strategy<--diff(log(capital_value_weekly))[-1,]
mu.return.weekly<-mean(as.numeric(loss_return_weekly.value.strategy))
variance.return_weekly<-var(as.numeric(loss_return_weekly.value.strategy))
VaR.return_weekly.normal<- mu.return.weekly + sqrt(variance.return_weekly) * qnorm(0.99)
VaR.return_value.normal.nonlinear=capital.Value_lastday_weekly*(1-exp(-VaR.return_weekly.normal))
VaR.return_value.normal.linear=capital.Value_lastday_weekly*VaR.return_weekly.normal
ES.return_weekly.normal<- mu.return.weekly + sqrt(variance.return_weekly) * dnorm(qnorm(0.975))/(1-0.975)
ES.return_weekly.normal.nonlinear=capital.Value_lastday_weekly*(1-exp(-ES.return_weekly.normal))
ES.return_weekly.normal.linear=capital.Value_lastday_weekly*ES.return_weekly.normal


#  Historical simulation

# non-linear
his_return_weekly.value<-1-exp(-loss_return_weekly.value.strategy)
VaR.hs.weekly_value.nonlinear <- quantile(his_return_weekly.value,0.99)*capital.Value_lastday_weekly
VaR.hs.weekly_value_0.975 <- quantile(his_return_weekly.value,0.975)
ES.hs.weekly_value.nonlinear <- mean(his_return_weekly.value[his_return_weekly.value > VaR.hs.weekly_value_0.975])*capital.Value_lastday_weekly
qqnorm(loss_return_weekly.value.strategy,main = "QQ-plot for Log Return of Weekly Rebalanced Value Strategy")
qqline(loss_return_weekly.value.strategy)
# linear
VaR.hs.weekly_value.linear <- quantile(loss_return_weekly.value.strategy,0.99)*capital.Value_lastday_weekly
VaR.hs.weekly_value_0.975 <- quantile(loss_return_weekly.value.strategy,0.975)
```

```
632.    ES.hs.weekly_value.linear <- mean(loss_return_weekly.value.strategy[loss_return_wee
    kly.value.strategy > VaR.hs.weekly_value_0.975])*capital.Value_lastday_weekly
633.
634.    # plot value strategy results for weekly rebalance (non-linear)
635.    hist(loss_return_weekly.value.strategy,nclass=100, prob=TRUE, xlab="Log Return for
    Weekly Rebalance Value Strategy",
636.         main="Value Strategy with Weekly Rebalance")
637.    lines(seq(-0.15,0.15,by=1/100), dnorm(seq(-
    0.15,0.15, by=1/100),mu.return.weekly, sqrt(variance.return_weekly)), col="blue")
638.    abline(v=c(VaR.return_value.normal.nonlinear/capital.Value_lastday_weekly,ES.return
    _weekly.normal.nonlinear/capital.Value_lastday_weekly),col=c("blue"),lty=c(1,2))
639.    text(x=0.065,y=12,label="99% log return VaR",col="blue",cex=0.7)
640.    text(x=0.12,y=11,label="97.5% log return ES",col="blue",cex=0.7)
641.    lines(density(loss_return_weekly.value.strategy), col="red")
642.    abline(v=c(VaR.hs.weekly_value.nonlinear/capital.Value_lastday_weekly,ES.hs.weekly_
    value.nonlinear/capital.Value_lastday_weekly),col=c("red"),lty=c(1,2))
643.    text(x=0.075,y=10,label="97.5% log return ES",col="red",cex=0.7)
644.    text(x=0.135,y=9,label="99% log return VaR",col="red",cex=0.7)
645.
646.    legendnames <- c("Normal: 99% log return VaR and 97.5% log return ES","Historical S
    imulation: 99% log return VaR and 97.5% log return ES")
647.    legend("topleft", legend = legendnames, col=c("blue","red"), pch="-",cex = 0.63)
648.
649.    ############### Monthly:ES and VaR of Value strategy for normal and historical model
     #################
650.
651.    #  Normal model
652.
653.    capital.Value_lastday_monthly<-
    capital_value_monthly[nrow(capital_value_monthly),]
654.    loss_return_monthly.value.strategy<--diff(log(capital_value_monthly))[-1,]
655.    mu.return.monthly<-mean(as.numeric(loss_return_monthly.value.strategy))
656.    variance.return_monthly<-var(as.numeric(loss_return_monthly.value.strategy))
657.    VaR.return_monthly.normal<- mu.return.monthly + sqrt(variance.return_monthly) * qno
    rm(0.99)
658.    VaR.return_value.normal.nonlinear.monthly=capital.Value_lastday_monthly*(1-exp(-
    VaR.return_monthly.normal))
659.    VaR.return_value.normal.linear.monthly=capital.Value_lastday_monthly*VaR.return_mon
    thly.normal
660.    ES.return_monthly.normal<- mu.return.monthly + sqrt(variance.return_monthly) * dnor
    m(qnorm(0.975))/(1-0.975)
661.    ES.return_monthly.normal.nonlinear=capital.Value_lastday_monthly*(1-exp(-
    ES.return_monthly.normal))
662.    ES.return_monthly.normal.linear=capital.Value_lastday_monthly*ES.return_monthly.nor
    mal
663.
664.    #  Historical simulation
665.
666.    # non-linear
667.    his_return_monthly.value<-1-exp(-loss_return_monthly.value.strategy)
668.    VaR.hs.monthly_value.nonlinear <- quantile(his_return_monthly.value,0.99)*capital.V
    alue_lastday_monthly
669.    VaR.hs.monthly_value_0.975 <- quantile(his_return_monthly.value,0.975)
670.    ES.hs.monthly_value.nonlinear <- mean(his_return_monthly.value[his_return_monthly.v
    alue > VaR.hs.monthly_value_0.975])*capital.Value_lastday_monthly
671.    # linear
672.    VaR.hs.monthly_value.linear <- quantile(loss_return_monthly.value.strategy,0.99)*ca
    pital.Value_lastday_monthly
673.    VaR.hs.monthly_value_0.975 <- quantile(loss_return_monthly.value.strategy,0.975)
674.    ES.hs.monthly_value.linear <- mean(loss_return_monthly.value.strategy[loss_return_m
    onthly.value.strategy > VaR.hs.monthly_value_0.975])*capital.Value_lastday_monthly
675.
676.
677.    ############### Daily:ES and VaR of momentum strategy for normal and historical mode
    l #################
678.    #  Normal model
```

```
679.    capital.momentum_lastday_daily<-
    capital_momentum_daily[nrow(capital_momentum_daily),]
680.    loss_return_daily.momentum.strategy<--diff(log(capital_momentum_daily))[-1,]
681.    mu.return.daily.momentum<-mean(as.numeric(loss_return_daily.momentum.strategy))
682.    variance.return_daily.momentum<-
    var(as.numeric(loss_return_daily.momentum.strategy))
683.    VaR.return_daily.normal.momentum<- mu.return.daily.momentum + sqrt(variance.return_
    daily.momentum) * qnorm(0.99)
684.    VaR.return_daily.normal.momentum.nonlinear=capital.momentum_lastday_daily*(1-exp(-
    VaR.return_daily.normal.momentum))
685.    VaR.return_momentum.daily.normal.linear=capital.momentum_lastday_daily*VaR.return_d
    aily.normal.momentum
686.    ES.return_daily.momentum.normal<- mu.return.daily.momentum + sqrt(variance.return_d
    aily.momentum) * dnorm(qnorm(0.975))/(1-0.975)
687.    ES.return_daily.momentum.normal.nonlinear=capital.momentum_lastday_daily*(1-exp(-
    ES.return_daily.momentum.normal))
688.    ES.return_daily.momentum.normal.linear=capital.momentum_lastday_daily*ES.return_dai
    ly.momentum.normal
689.
690.    #  Historical simulation
691.
692.    # non-linear
693.    his_return_daily.momentum<-1-exp(-loss_return_daily.momentum.strategy)
694.    VaR.hs.daily_momentum.nonlinear <- quantile(his_return_daily.momentum,0.99)*capital
    .momentum_lastday_daily
695.    VaR.hs.daily_momentum_0.975 <- quantile(his_return_daily.momentum,0.975)
696.    ES.hs.daily_momentum.nonlinear <- mean(his_return_daily.momentum[his_return_daily.m
    omentum > VaR.hs.daily_momentum_0.975])*capital.momentum_lastday_daily
697.
698.    # linear
699.    VaR.hs.daily_momentum.linear <- quantile(loss_return_daily.momentum.strategy,0.99)*
    capital.momentum_lastday_daily
700.    VaR.hs.daily_momentum_0.975 <- quantile(loss_return_daily.momentum.strategy,0.975)

701.    ES.hs.daily_momentum.linear <- mean(loss_return_daily.momentum.strategy[loss_return
    _daily.momentum.strategy > VaR.hs.daily_momentum_0.975])*capital.momentum_lastday_daily

702.
703.    ############### Weekly:ES and VaR of momentum strategy for normal and historical mod
    el #################
704.    #  Normal model
705.    capital.momentum_lastday_weekly<-
    capital_momentum_weekly[nrow(capital_momentum_weekly),]
706.    loss_return_weekly.momentum.strategy<--diff(log(capital_momentum_weekly))[-1,]
707.    mu.return.weekly.momentum<-mean(as.numeric(loss_return_weekly.momentum.strategy))
708.    variance.return_weekly.momentum<-
    var(as.numeric(loss_return_weekly.momentum.strategy))
709.    VaR.return_weekly.normal.momentum<- mu.return.weekly.momentum + sqrt(variance.retur
    n_weekly.momentum) * qnorm(0.99)
710.    VaR.return_weekly.normal.momentum.nonlinear=capital.momentum_lastday_weekly*(1-
    exp(-VaR.return_weekly.normal.momentum))
711.    VaR.return_momentum.weekly.normal.linear=capital.momentum_lastday_weekly*VaR.return
    _weekly.normal.momentum
712.    ES.return_weekly.momentum.normal<- mu.return.weekly.momentum + sqrt(variance.return
    _weekly.momentum) * dnorm(qnorm(0.975))/(1-0.975)
713.    ES.return_weekly.momentum.normal.nonlinear=capital.momentum_lastday_weekly*(1-exp(-
    ES.return_weekly.momentum.normal))
714.    ES.return_weekly.momentum.normal.linear=capital.momentum_lastday_weekly*ES.return_w
    eekly.momentum.normal
715.
716.    #  Historical simulation
717.
718.    # non-linear
719.    his_return_weekly.momentum<-1-exp(-loss_return_weekly.momentum.strategy)
720.    VaR.hs.weekly_momentum.nonlinear <- quantile(his_return_weekly.momentum,0.99)*capit
    al.momentum_lastday_weekly
```

```
721.    VaR.hs.weekly_momentum_0.975 <- quantile(his_return_weekly.momentum,0.975)
722.    ES.hs.weekly_momentum.nonlinear <- mean(his_return_weekly.momentum[his_return_weekl
    y.momentum > VaR.hs.weekly_momentum_0.975])*capital.momentum_lastday_weekly
723.
724.    qqnorm(loss_return_weekly.momentum.strategy,main = "QQ-
    plot for Log Return of Weekly Rebalanced Momentum Strategy")
725.    qqline(loss_return_weekly.momentum.strategy)
726.    # linear
727.    VaR.hs.weekly_momentum.linear <- quantile(loss_return_weekly.momentum.strategy,0.99
    )*capital.momentum_lastday_weekly
728.    VaR.hs.weekly_momentum_0.975 <- quantile(loss_return_weekly.momentum.strategy,0.975
    )
729.    ES.hs.weekly_momentum.linear <- mean(loss_return_weekly.momentum.strategy[loss_retu
    rn_weekly.momentum.strategy > VaR.hs.weekly_momentum_0.975])*capital.momentum_lastday_w
    eekly
730.
731.    # test normality
732.
733.    MardiaTest(as.timeSeries(log_return_weekly_testing[,1:5]))
734.
735.    # plot value strategy results for weekly rebalance (non-linear)
736.    hist(loss_return_weekly.momentum.strategy,nclass=100, prob=TRUE, xlab="Log Return f
    or Weekly Rebalance Momentum Strategy",
737.         main="Momentum Strategy with Weekly Rebalance")
738.    lines(seq(-0.15,0.15,by=1/100), dnorm(seq(-
    0.15,0.15, by=1/100),mu.return.weekly.momentum, sqrt(variance.return_weekly.momentum)),
     col="blue")
739.    abline(v=c(VaR.return_weekly.normal.momentum.nonlinear/capital.momentum_lastday_wee
    kly,ES.return_weekly.momentum.normal.nonlinear/capital.momentum_lastday_weekly),col=c("
    blue"),lty=c(1,2))
740.    text(x=0.052,y=15,label="99% log return VaR",col="blue",cex=0.7)
741.    text(x=0.1,y=14,label="97.5% log return ES",col="blue",cex=0.7)
742.    lines(density(loss_return_weekly.momentum.strategy), col="red")
743.    abline(v=c(VaR.hs.weekly_momentum.nonlinear/capital.momentum_lastday_weekly,ES.hs.w
    eekly_momentum.nonlinear/capital.momentum_lastday_weekly),col=c("red"),lty=c(1,2))
744.    text(x=0.065,y=13,label="99% log return VaR",col="red",cex=0.7)
745.    text(x=0.112,y=12,label="97.5% log return ES",col="red",cex=0.7)
746.    legendnames <- c("Normal: 99% log return VaR and 97.5% log return ES","Historical S
    imulation: 99% log return VaR and 97.5% log return ES")
747.    legend("topleft", legend = legendnames, col=c("blue","red"), pch="-",cex = 0.65)
748.
749.
750.    ############### Monthly:ES and VaR of momentum strategy for normal and historical mo
    del #################
751.    #  Normal model
752.    capital.momentum_lastday_monthly<-
    capital_momentum_monthly[nrow(capital_momentum_monthly),]
753.    loss_return_monthly.momentum.strategy<--diff(log(capital_momentum_monthly))[-1,]
754.    mu.return.monthly.momentum<-
    mean(as.numeric(loss_return_monthly.momentum.strategy))
755.    variance.return_monthly.momentum<-
    var(as.numeric(loss_return_monthly.momentum.strategy))
756.    VaR.return_monthly.normal.momentum<- mu.return.monthly.momentum + sqrt(variance.ret
    urn_monthly.momentum) * qnorm(0.99)
757.    VaR.return_monthly.normal.momentum.nonlinear=capital.momentum_lastday_monthly*(1-
    exp(-VaR.return_monthly.normal.momentum))
758.    VaR.return_momentum.monthly.normal.linear=capital.momentum_lastday_monthly*VaR.retu
    rn_monthly.normal.momentum
759.    ES.return_monthly.momentum.normal<- mu.return.monthly.momentum + sqrt(variance.retu
    rn_monthly.momentum) * dnorm(qnorm(0.975))/(1-0.975)
760.    ES.return_monthly.momentum.normal.nonlinear=capital.momentum_lastday_monthly*(1-
    exp(-ES.return_monthly.momentum.normal))
761.    ES.return_monthly.momentum.normal.linear=capital.momentum_lastday_monthly*ES.return
    _monthly.momentum.normal
762.
763.    #  Historical simulation
```

```
764.
765.    # non-linear
766.    his_return_monthly.momentum<-1-exp(-loss_return_monthly.momentum.strategy)
767.    VaR.hs.monthly_momentum.nonlinear <- quantile(his_return_monthly.momentum,0.99)*cap
    ital.momentum_lastday_monthly
768.    VaR.hs.monthly_momentum_0.975 <- quantile(his_return_monthly.momentum,0.975)
769.    ES.hs.monthly_momentum.nonlinear <- mean(his_return_monthly.momentum[his_return_mon
    thly.momentum > VaR.hs.monthly_momentum_0.975])*capital.momentum_lastday_monthly
770.
771.    # linear
772.    VaR.hs.monthly_momentum.linear <- quantile(loss_return_monthly.momentum.strategy,0.
    99)*capital.momentum_lastday_monthly
773.    VaR.hs.monthly_momentum_0.975 <- quantile(loss_return_monthly.momentum.strategy,0.9
    75)
774.    ES.hs.monthly_momentum.linear <- mean(loss_return_monthly.momentum.strategy[loss_re
    turn_monthly.momentum.strategy > VaR.hs.monthly_momentum_0.975])*capital.momentum_lastd
    ay_monthly
775.
776.
777.    ############### Daily:ES and VaR of dynamic strategy for normal and historical model
     #################
778.    #  Normal model
779.    capital.auto_lastday_daily<-capital_ac_daily[nrow(capital_ac_daily),]
780.    loss_return_daily.auto.strategy<--diff(log(capital_ac_daily))[-1,]
781.    mu.return.daily.auto<-mean(as.numeric(loss_return_daily.auto.strategy))
782.    variance.return_daily.auto<-var(as.numeric(loss_return_daily.auto.strategy))
783.    VaR.return_daily.normal.auto<- mu.return.daily.auto + sqrt(variance.return_daily.au
    to) * qnorm(0.99)
784.    VaR.return_daily.normal.auto.nonlinear=capital.auto_lastday_daily*(1-exp(-
    VaR.return_daily.normal.auto))
785.    VaR.return_auto.daily.normal.linear=capital.auto_lastday_daily*VaR.return_daily.nor
    mal.auto
786.    ES.return_daily.auto.normal<- mu.return.daily.auto + sqrt(variance.return_daily.aut
    o) * dnorm(qnorm(0.975))/(1-0.975)
787.    ES.return_daily.auto.normal.nonlinear=capital.auto_lastday_daily*(1-exp(-
    ES.return_daily.auto.normal))
788.    ES.return_daily.auto.normal.linear=capital.auto_lastday_daily*ES.return_daily.auto.
    normal
789.
790.    #  Historical simulation
791.
792.    # non-linear
793.    his_return_daily.auto<-1-exp(-loss_return_daily.auto.strategy)
794.    VaR.hs.daily_auto.nonlinear <- quantile(his_return_daily.auto,0.99)*capital.auto_la
    stday_daily
795.    VaR.hs.daily_auto_0.975 <- quantile(his_return_daily.auto,0.975)
796.    ES.hs.daily_auto.nonlinear <- mean(his_return_daily.auto[his_return_daily.auto > Va
    R.hs.daily_auto_0.975])*capital.auto_lastday_daily
797.
798.    # linear
799.    VaR.hs.daily_auto.linear <- quantile(loss_return_daily.auto.strategy,0.99)*capital.
    auto_lastday_daily
800.    VaR.hs.daily_auto_0.975 <- quantile(loss_return_daily.auto.strategy,0.975)
801.    ES.hs.daily_auto.linear <- mean(loss_return_daily.auto.strategy[loss_return_daily.a
    uto.strategy > VaR.hs.daily_auto_0.975])*capital.auto_lastday_daily
802.
803.    ############### Weekly:ES and VaR of dynamic strategy for normal and historical mode
    l #################
804.    #  Normal model
805.    capital.auto_lastday_weekly<-capital_ac_weekly[nrow(capital_ac_weekly),]
806.    loss_return_weekly.auto.strategy<--diff(log(capital_ac_weekly))[-1,]
807.    mu.return.weekly.auto<-mean(as.numeric(loss_return_weekly.auto.strategy))
808.    variance.return_weekly.auto<-var(as.numeric(loss_return_weekly.auto.strategy))
809.    VaR.return_weekly.normal.auto<- mu.return.weekly.auto + sqrt(variance.return_weekly
    .auto) * qnorm(0.99)
```

```
VaR.return_weekly.normal.auto.nonlinear=capital.auto_lastday_weekly*(1-exp(-VaR.return_weekly.normal.auto))
VaR.return_auto.weekly.normal.linear=capital.auto_lastday_weekly*VaR.return_weekly.normal.auto
ES.return_weekly.auto.normal<- mu.return.weekly.auto + sqrt(variance.return_weekly.auto) * dnorm(qnorm(0.975))/(1-0.975)
ES.return_weekly.auto.normal.nonlinear=capital.auto_lastday_weekly*(1-exp(-ES.return_weekly.auto.normal))
ES.return_weekly.auto.normal.linear=capital.auto_lastday_weekly*ES.return_weekly.auto.normal

#  Historical simulation

# non-linear
his_return_weekly.auto<-1-exp(-loss_return_weekly.auto.strategy)
VaR.hs.weekly_auto.nonlinear <- quantile(his_return_weekly.auto,0.99)*capital.auto_lastday_weekly
VaR.hs.weekly_auto_0.975 <- quantile(his_return_weekly.auto,0.975)
ES.hs.weekly_auto.nonlinear <- mean(his_return_weekly.auto[his_return_weekly.auto > VaR.hs.weekly_auto_0.975])*capital.auto_lastday_weekly
qqnorm(loss_return_weekly.auto.strategy,main = "QQ-plot for Log Return of Weekly Rebalanced Dynamic Strategy")
qqline(loss_return_weekly.auto.strategy)

# linear
VaR.hs.weekly_auto.linear <- quantile(loss_return_weekly.auto.strategy,0.99)*capital.auto_lastday_weekly
VaR.hs.weekly_auto_0.975 <- quantile(loss_return_weekly.auto.strategy,0.975)
ES.hs.weekly_auto.linear <- mean(loss_return_weekly.auto.strategy[loss_return_weekly.auto.strategy > VaR.hs.weekly_auto_0.975])*capital.auto_lastday_weekly

# plot value strategy results for weekly rebalance (non-linear)
hist(loss_return_weekly.auto.strategy,nclass=100, prob=TRUE, xlab="Log Return for Weekly Rebalance Dynamic Strategy",
        main="Dynamic Strategy with Weekly Rebalance",xlim = c(-0.2,0.2))
lines(seq(-0.2,0.2,by=1/100), dnorm(seq(-0.2,0.2, by=1/100),mu.return.weekly.auto, sqrt(variance.return_weekly.auto)), col="blue")
abline(v=c(VaR.return_weekly.normal.auto.nonlinear/capital.auto_lastday_weekly,ES.return_weekly.auto.normal.nonlinear/capital.auto_lastday_weekly),col=c("blue"),lty=c(1,2))
text(x=0.06,y=14,label="99% log return VaR",col="blue",cex=0.7)
text(x=0.13,y=12,label="97.5% log return ES",col="blue",cex=0.7)
lines(density(loss_return_weekly.auto.strategy), col="red")
abline(v=c(VaR.hs.weekly_auto.nonlinear/capital.auto_lastday_weekly,ES.hs.weekly_auto.nonlinear/capital.auto_lastday_weekly),col=c("red"),lty=c(1,2))
text(x=0.07,y=10,label="99% log return VaR",col="red",cex=0.7)
text(x=0.14,y=8,label="97.5% log return ES",col="red",cex=0.7)
legendnames <- c("Normal: 99% log return VaR and 97.5% log return ES","Historical Simulation: 99% log return VaR and 97.5% log return ES")
legend("topleft", legend = legendnames, col=c("blue","red"), pch="-",cex = 0.65)

############### Monthly:ES and VaR of dynamic strategy for normal and historical model #################
#  Normal model
capital.auto_lastday_monthly<-capital_ac_monthly[nrow(capital_ac_monthly),]
loss_return_monthly.auto.strategy<--diff(log(capital_ac_monthly))[-1,]
mu.return.monthly.auto<-mean(as.numeric(loss_return_monthly.auto.strategy))
variance.return_monthly.auto<-var(as.numeric(loss_return_monthly.auto.strategy))
VaR.return_monthly.normal.auto<- mu.return.monthly.auto + sqrt(variance.return_monthly.auto) * qnorm(0.99)
VaR.return_monthly.normal.auto.nonlinear=capital.auto_lastday_monthly*(1-exp(-VaR.return_monthly.normal.auto))
VaR.return_auto.monthly.normal.linear=capital.auto_lastday_monthly*VaR.return_monthly.normal.auto
```

```
854.    ES.return_monthly.auto.normal<- mu.return.monthly.auto + sqrt(variance.return_month
    ly.auto) * dnorm(qnorm(0.975))/(1-0.975)
855.    ES.return_monthly.auto.normal.nonlinear=capital.auto_lastday_monthly*(1-exp(-
    ES.return_monthly.auto.normal))
856.    ES.return_monthly.auto.normal.linear=capital.auto_lastday_monthly*ES.return_monthly
    .auto.normal
857.
858.    #  Historical simulation
859.
860.    # non-linear
861.    his_return_monthly.auto<-1-exp(-loss_return_monthly.auto.strategy)
862.    VaR.hs.monthly_auto.nonlinear <- quantile(his_return_monthly.auto,0.99)*capital.aut
    o_lastday_monthly
863.    VaR.hs.monthly_auto_0.975 <- quantile(his_return_monthly.auto,0.975)
864.    ES.hs.monthly_auto.nonlinear <- mean(his_return_monthly.auto[his_return_monthly.aut
    o > VaR.hs.monthly_auto_0.975])*capital.auto_lastday_monthly
865.
866.    # linear
867.    VaR.hs.monthly_auto.linear <- quantile(loss_return_monthly.auto.strategy,0.99)*capi
    tal.auto_lastday_monthly
868.    VaR.hs.monthly_auto_0.975 <- quantile(loss_return_monthly.auto.strategy,0.975)
869.    ES.hs.monthly_auto.linear <- mean(loss_return_monthly.auto.strategy[loss_return_mon
    thly.auto.strategy > VaR.hs.monthly_auto_0.975])*capital.auto_lastday_monthly
870.
871.    #--------------------
872.    #some other plots
873.    #--------------------
874.    #plot the ar coefficients for each stocks
875.
876.    #daily
877.    par(mfrow=c(1,1),oma=rep(0,4))
878.    plot(ar_model_daily[,1],xlab="Dates",ylab="AR(1) Model Coefficients", ylim=c(min(ar
    _model_daily),max(ar_model_daily)),type="n",main="Coefficients of AR(1) Model for Tech
    Stocks",sub="Rebalance Period = 1 Day")
879.    for (i in 1:(stock_number-1)){
880.      lines(ar_model_daily[,i],col=i)
881.    }
882.    legend("topright",legend=stock_list[1:5],col=1:5, lty=1,cex=0.8)
883.    abline(h=0)
884.
885.    #weekly
886.    par(mfrow=c(1,1),oma=rep(0,4))
887.    plot(ar_model_weekly[,1],xlab="Dates",ylab="AR(1) Model Coefficients", ylim=c(min(a
    r_model_weekly),max(ar_model_weekly)),type="n",main="Coefficients of AR(1) Model for Te
    ch Stocks",sub="Rebalance Period = 1 Week")
888.    for (i in 1:(stock_number-1)){
889.      lines(ar_model_weekly[,i],col=i)
890.    }
891.    legend("topright",legend=stock_list[1:5],col=1:5, lty=1,cex=0.8)
892.    abline(h=0)
893.
894.    #monthly
895.    par(mfrow=c(1,1),oma=rep(0,4))
896.    plot(ar_model_monthly[,1],xlab="Dates",ylab="AR(1) Model Coefficients", ylim=c(min(
    ar_model_monthly),max(ar_model_monthly)),type="n",main="Coefficients of AR(1) Model for
     Tech Stocks",sub="Rebalance Period = 1 Month")
897.    for (i in 1:(stock_number-1)){
898.      lines(ar_model_weekly[,i],col=i)
899.    }
900.    legend("topright",legend=stock_list[1:5],col=1:5, lty=1,cex=0.8)
901.    abline(h=0)
902.
903.
904.    #plot the weight
905.    ####daily####
906.    #momentum
```

```
907.    par(mfrow=c(3,1),oma=c(0,0,2,0))
908.
909.    plot(weights_momentum_daily[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_mome
    ntum_daily),max(weights_momentum_daily)),type="n",main="Momentum Strategy")
910.    for ( i in 1:(stock_number-1)){
911.      lines (weights_momentum_daily[,i],col=i)
912.    }
913.    abline(h=0)
914.    #value
915.    plot(weights_value_daily[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_value_d
    aily),max(weights_value_daily)),type="n",main="Value Strategy")
916.    for ( i in 1:(stock_number-1)){
917.      lines (weights_value_daily[,i],col=i)
918.    }
919.    abline(h=0)
920.
921.    #autoregression
922.    plot(weights_ac_daily[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_ac_daily),
    max(weights_ac_daily)),type="n",main="Autoregression Strategy")
923.    for ( i in 1:(stock_number-1)){
924.      lines (weights_ac_daily[,i],col=i)
925.    }
926.    abline(h=0)
927.    par(xpd=NA)
928.
929.    legend("bottom",legend=stock_list[1:5],col=1:5,lty=1,cex=0.8,horiz = TRUE)
930.    mtext(side=3,line=0,cex=1.5,outer=T, "Weights of Each Stock in Different Strategies
    ,Rebalance Period: 1 Day")
931.
932.    ####weekly####
933.    #momentum
934.    par(mfrow=c(3,1),oma=c(0,0,2,0))
935.
936.    plot(weights_momentum_weekly[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_mom
    entum_weekly),max(weights_momentum_weekly)),type="n",main="Momentum Strategy")
937.    for ( i in 1:(stock_number-1)){
938.      lines (weights_momentum_weekly[,i],col=i)
939.    }
940.    abline(h=0)
941.    #value
942.    plot(weights_value_weekly[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_value_
    weekly),max(weights_value_weekly)),type="n",main="Value Strategy")
943.    for ( i in 1:(stock_number-1)){
944.      lines (weights_value_weekly[,i],col=i)
945.    }
946.    abline(h=0)
947.
948.    #autoregression
949.    plot(weights_ac_weekly[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_ac_weekly
    ),max(weights_ac_weekly)),type="n",main="Autoregression Strategy")
950.    for ( i in 1:(stock_number-1)){
951.      lines (weights_ac_weekly[,i],col=i)
952.    }
953.    abline(h=0)
954.    par(xpd=NA)
955.
956.    legend("bottom",legend=stock_list[1:5],col=1:5,lty=1,cex=0.8,horiz = TRUE)
957.    mtext(side=3,line=0,cex=1.5,outer=T, "Weights of Each Stock in Different Strategies
    , Rebalance Period: 1 Week")
958.
959.
960.    ####monthly####
961.    #momentum
962.    par(mfrow=c(3,1),oma=c(0,0,2,0))
963.
```

```
964.    plot(weights_momentum_monthly[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_mo
    mentum_monthly),max(weights_momentum_monthly)),type="n",main="Momentum Strategy")
965.    for ( i in 1:(stock_number-1)){
966.      lines (weights_momentum_monthly[,i],col=i)
967.    }
968.    abline(h=0)
969.    #value
970.    plot(weights_value_monthly[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_value
    _monthly),max(weights_value_monthly)),type="n",main="Value Strategy")
971.    for ( i in 1:(stock_number-1)){
972.      lines (weights_value_monthly[,i],col=i)
973.    }
974.    abline(h=0)
975.
976.    #autoregression
977.    plot(weights_ac_monthly[,1],xlab="Dates",ylab="Weights",ylim=c(min(weights_ac_month
    ly),max(weights_ac_monthly)),type="n",main="Autoregression Strategy")
978.    for ( i in 1:(stock_number-1)){
979.      lines (weights_ac_monthly[,i],col=i)
980.    }
981.    abline(h=0)
982.    par(xpd=NA)
983.
984.    legend("bottom",legend=stock_list[1:5],col=1:5,lty=1,cex=0.8,horiz = TRUE)
985.    mtext(side=3,line=0,cex=1.5,outer=T, "Weights of Each Stock in Different Strategies
    , Rebance Period: 1 Month")
986.
987.
988.    ##### Compare the value of portfolio (daily,weekly,monthly)
989.    par(mfrow=c(3,1),oma=c(0,0,2,0))
990.    # Value strategy:
991.    plot(as.timeSeries(capital_value_daily),type="l",ylab="Value of Portfolio",col=1,yl
    im=c(min(capital_value_weekly),max(capital_value_weekly)))
992.    title("Value Strategy")
993.    lines(as.timeSeries(capital_value_weekly),type="l",col=2)
994.    lines(as.timeSeries(capital_value_monthly),type="l",col=4)
995.    legend("topleft", legend = c("daily","weekly","monthly"), col=c(1,2,4), lty=1)
996.
997.    # Momentum strategy:
998.    plot(as.timeSeries(capital_momentum_daily),type="l",ylab="Value of Portfolio",col=1
    ,ylim=c(min(capital_momentum_monthly),max(capital_momentum_monthly)))
999.    title("Momentum Strategy")
1000.   lines(as.timeSeries(capital_momentum_weekly),type="l",col=2)
1001.   lines(as.timeSeries(capital_momentum_monthly),type="l",col=4)
1002.   legend("topleft", legend = c("daily","weekly","monthly"), col=c(1,2,4), lty=1)
1003.
1004.   # Autocorrelation strategy:
1005.   plot(as.timeSeries(capital_ac_daily),type="l",ylab="Value of Portfolio",col=1,ylim=
    c(min(capital_ac_weekly),max(capital_ac_weekly)))
1006.   title("Autocorelation Strategy")
1007.   lines(as.timeSeries(capital_ac_weekly),type="l",col=2)
1008.   lines(as.timeSeries(capital_ac_monthly),type="l",col=4)
1009.   legend("topleft", legend = c("daily","weekly","monthly"), col=c(1,2,4), lty=1)
1010.
1011.   mtext(side=3,line=0,cex=1.5,outer=T, "Total Value of Portfolios with Different Reba
    lance Period")
1012.
1013.
1014.
1015.   #==================
1016.   #part d
1017.   #==================
1018.   stockIndexLogReturn.ts <-timeSeries(log_return[-1,])
1019.   names(stockIndexLogReturn.ts)=paste0(stock_list)
1020.   #create index
```

```
mini_index=stockIndexLogReturn.ts$AAPL*0.2423+stockIndexLogReturn.ts$AMZN*0.2314+stockIndexLogReturn.ts$GOOG*0.2376+stockIndexLogReturn.ts$MSFT*0.2529+stockIndexLogReturn.ts$NFLX*0.0359
mini_index=timeSeries(mini_index,as.Date(portfolio$Date[-1], format="%d/%m/%Y"))
names(mini_index)="mini_index"
stockIndexLogReturn.ts=merge(stockIndexLogReturn.ts,mini_index)
plot(stockIndexLogReturn.ts,col="black",main="Log return of mini index compared with other stocks")#show the independence
par(mfrow=c(1,1))
X <- as.matrix(stockIndexLogReturn.ts)
X <- X[X[,6]!=0 & X[,7] !=0,c(6,7)]
copulaX <- apply(X,2,edf,adjust=1)
plot(copulaX, xlab="S&P 500", ylab="Mini Index",main="Copulas of Mini Index versus S&P500")

# Gaussian copula fit
copulaXGauss <- fit.gausscopula(copulaX)
copulaXGauss
# t copula
copulaXt <- fit.tcopula(copulaX)
copulaXt
# 2-dimensional Archimedian copulas (Gumbel and Clayton)
copulaXGumb <- fit.AC(copulaX,"gumbel")
copulaXGumb
copulaXClay <- fit.AC(copulaX,"clayton")
copulaXClay

c(copulaXGauss$ll.max, copulaXt$ll.max, copulaXGumb$ll.max, copulaXClay$ll.max)
##t is the best

# Estimate Spearman rank correlations
copulacorr <- Spearman(copulaX)
copulacorr #show the independence

#================
#part e
#================
Y=-mini_index#negative return
#find large drowdown
plot(Y,type="l", ylab="Mini Index", xlab="Date",main="Loss Return of Mini Index")

#large drowdown in 2006,2018
#large up and down value about +/-10% on several days

# Find monthly and quarterly maxima
by <- timeSequence(from = start(Y),  to = end(Y), by = "month")
Y.monthMax <- aggregate(Y,by,max)#max negative log-return in a month
by <- timeSequence(from = start(Y),  to = end(Y), by = "quarter")
Y.quarterMax <- aggregate(Y,by,max)

#~ Transform to matrix and fit generalized extreme value (GEV) distribution
Y.monthMax <- as.matrix(Y.monthMax)
GEVmonth <- fit.GEV(Y.monthMax)#fit GEM distribution
Y.quarterMax <- as.matrix(Y.quarterMax)
GEVquart <- fit.GEV(Y.quarterMax)

#40-quarter return level-10 years
quarterReturn <- qGEV(1-1/40,xi=GEVquart$par.ests[1],mu=GEVquart$par.ests[2], sigma=GEVquart$par.ests[3])

#return period
1/(1-pGEV(max(Y.quarterMax),
          xi=GEVquart$par.ests[1],mu=GEVquart$par.ests[3],sigma=GEVquart$par.ests[2]))
```

```
#120-month return level-10 years
monthReturn <- qGEV(1-
  1/120,xi=GEVmonth$par.ests[1],mu=GEVmonth$par.ests[2], sigma=GEVmonth$par.ests[3])

#return period
1/(1-pGEV(max(Y.monthMax),
          xi=GEVmonth$par.ests[1],mu=GEVmonth$par.ests[3],sigma=GEVmonth$par.ests[2
  ]))

# convert timeSeries into data.frame
Y.monthMax_df = data.frame(mini_index=c(Y.monthMax), Date=rownames(Y.monthMax))
Y.quarterMax_df = data.frame(mini_index=c(Y.quarterMax), Date=rownames(Y.quarterMax
  ))


# PDF of GEV Distribution
m_GEV = fgev(Y.monthMax)
q_GEV = fgev(Y.quarterMax)
par(mfrow=c(1,2))
plot(m_GEV, which=3, main = c("PDF: Monthly-GEV Distributions"))
plot(q_GEV, which=3, main = c("PDF: Quarterly-GEV Distributions"))

# Monthly Maxima
MMaxi <- ggplot(Y.monthMax_df, aes(x=1:nrow(Y.monthMax_df), y=mini_index)) +
  geom_point() +
  labs(title = "mini_index Monthly Maxima", y="Negative Returns",x="Index") +
  geom_hline(aes(yintercept=monthReturn), colour="#990000", linetype="dashed")+
  geom_text(aes(3, 0.0765), label="7.65%",col="red")

# Quarterly Maxima
QMaxi <- ggplot(Y.quarterMax_df, aes(x=1:nrow(Y.quarterMax_df), y=mini_index)) +
  geom_point() +
  labs(title = "mini_index Quarterly Maxima", y="Negative Returns",x="Index") +
  geom_hline(aes(yintercept=quarterReturn), colour="#990000", linetype="dashed")+
  geom_text(aes(2 , 0.077), label="7.95%",col="red")
grid.arrange(MMaxi, QMaxi,ncol=2,nrow=1)
```