Financial Model Implementations based on real-world market data

by

Yishu Yang

A thesis
presented to Pro. Liu Zhi from
University of Macau in fulfillment of the
thesis requirement for the Summer
Research Program at June, 2024 for the
major of Statistics and Data Science in
Mathematics

Taipa, Macau SAR, China, 2024

Author's Declaration

I am here to declare that this thesis is solely written by myself and the only reference is the book: <Analyzing Financial Data and Implementing Financial Models Using R> by Clifford S. Ang, which is the requirement for the summer research program for me as required by Pro. Liu Zhi.

I also declare that the only program software is R language as required by Pro. Liu Zhi to complete this thesis, where the package of codes and outputs are attached to this thesis and code is provided in the Appendix.

Abstract

This thesis mainly does research on the relevant financial modeling techniques studied by Clifford S. Ang in his book: < Analyzing Financial Data and Implementing Financial Models Using R>, which implemented several common financial quantities in term of real-world data and provides me with basic financial knowledge for preparation of possible further project on study of price staleness's effect realize covariance.

Hence the majority of this thesis is my understanding of relevant chapters of book attached by programming outcomes and ideology summarization after successful programming of financial models.

This thesis won't follow every step of programming algorithm in the book, but all outcome screenshots will be provided and important step-by-step programming will be provided in terms of vital programming examples of real-world data.

This thesis contains analysis of basic terminologies like prices, individual security returns, portfolio returns and risks, where after the introduction of basic terminologies it also provides detailed analysis of specific financial models like Factor Models, Risk-Adjusted Portfolio Performance Measures, Markowitz Mean-Variance Optimization, Fixed Income and various Options.

We will refer to the book: < Analyzing Financial Data and Implementing Financial Models Using R> as "This Book" in the later chapters.

Acknowledgements

I would firstly thank all my mathematics professors who give me solid and comprehensive mathematical knowledge which helps me to review this financial mathematics book to improve my application of my mathematics knowledge in financial areas.

Secondly, I would thank Clifford S. Ang, who is the author of this book: <Analyzing Financial Data and Implementing Financial Models Using R> to provide resources and materials with my summer research program about specific R programming in financial areas.

Finaly, I would thank Pro. Liu Zhi to give this opportunity to review a financial classic book to help me improve my mathematical ability in programming and doing research on realistic mathematics problems.

I would also thank my family to support my study as always by financial assistance to my study in University of Macau.

Dedication

This is dedicated to my grandma, who has raised me up since I was born and will be my love forever until my death.

Chapter 1 Prices

In Chapter I Price the author of This Book mainly discusses the distinctive difference of price and value of certain securities, whose distinction majorly serves as the inner driven characteristic of market trades of buying and selling of certain securities. And the author firstly introduces how to import business data in Yahoo Finance to manipulate and analyses later on, which is mostly stock data and exchange traded funds (ETF).

1.1 Import Data

The way of importing data from Yahoo Finance here is very direct, just downloading data into a CSV file and upload CSV file into R software. However, we still need to perform some data manipulation to transform the raw data.

1.2 Import Price Data from Yahoo Finance

Step 1: Import CSV File from Yahoo Finance

I will download data from Amazon.com with ticker symbol AMZN in Yahoo Finance, which means I retrieve a CSV file of the historical data for AMZN from December 31, 2010 to December 31, 2013.

The below screenshots are from the file 'AMZN Yahooo.csv' opened in visual studio code for view.

```
1   Date,Open,High,Low,Close,Adj Close,Volume
2   2010-12-31,9.098000,9.115000,8.975500,9.000000,9.000000,69038000
3   2011-01-03,9.068500,9.300000,9.060500,9.211000,9.211000,106628000
4   2011-01-04,9.307500,9.385000,9.189000,9.250500,9.250500,100636000
5   2011-01-05,9.205000,9.372500,9.203500,9.371000,9.371000,68376000
752 2013-12-24,20.125999,20.186001,19.818501,19.959999,19.959999,27608000
753 2013-12-26,20.089500,20.226000,19.840500,20.219500,20.219500,37370000
754 2013-12-27,20.232500,20.281500,19.812500,19.903999,19.903999,39738000
755 2013-12-30,19.970501,19.996000,19.622499,19.668501,19.668501,49742000
756 2013-12-31,19.729000,19.941500,19.690001,19.939501,19.939501,39930000
```

```
> getwd()
[1] "C:/Users/yishu/Documents"
```

Use this command to get path of the directory of R files and data files.

Step 2: Import Data into R

Read the data by “read.csv()” and sort it from the latest to the oldest:

```
> data.AMZN<-read.csv("AMZN Yahooo.csv",header=TRUE)
> View(data.AMZN)
> data.AMZN <-data.AMZN[nrow(data.AMZN):1,]
> View(data.AMZN)
```

data.AMZN — 755 obs. of 7 variables

	Date	Open	High	Low	Close	Adj.Close	Volume
755	2013-12-31	19.7290	19.9415	19.6900	19.9395	19.9395	39930000
754	2013-12-30	19.9705	19.9960	19.6225	19.6685	19.6685	49742000
753	2013-12-27	20.2325	20.2815	19.8125	19.9040	19.9040	39738000
752	2013-12-26	20.0895	20.2260	19.8405	20.2195	20.2195	37370000
751	2013-12-24	20.1260	20.1860	19.8185	19.9600	19.9600	27608000

By using “head()” and “tail()” in R language:

```
> head(data.AMZN)
          Date    Open    High     Low   Close Adj.Close   Volume
755 2013-12-31 19.7290 19.9415 19.6900 19.9395   19.9395 39930000
754 2013-12-30 19.9705 19.9960 19.6225 19.6685   19.6685 49742000
753 2013-12-27 20.2325 20.2815 19.8125 19.9040   19.9040 39738000
752 2013-12-26 20.0895 20.2260 19.8405 20.2195   20.2195 37370000
751 2013-12-24 20.1260 20.1860 19.8185 19.9600   19.9600 27608000
750 2013-12-23 20.1845 20.2500 19.9600 20.1460   20.1460 53190000
> tail(data.AMZN)
        Date   Open   High    Low  Close Adj.Close    Volume
6 2011-01-07 9.3940 9.4225 9.1870 9.2745    9.2745 104434000
5 2011-01-06 9.3250 9.3705 9.2625 9.2930    9.2930  63594000
4 2011-01-05 9.2050 9.3725 9.2035 9.3710    9.3710  68376000
3 2011-01-04 9.3075 9.3850 9.1890 9.2505    9.2505 100636000
2 2011-01-03 9.0685 9.3000 9.0605 9.2110    9.2110 106628000
1 2010-12-31 9.0980 9.1150 8.9755 9.0000    9.0000  69038000
```

Step 3: Convert the date Variable from a Factor to a Date

```
> class(data.AMZN$Date)
[1] "character"
> date<-as.Date(data.AMZN$Date,format="%Y-%m-%d")
> head(date)
[1] "2013-12-31" "2013-12-30" "2013-12-27" "2013-12-26" "2013-12-24" "2013-12-23"
> tail(date)
[1] "2011-01-07" "2011-01-06" "2011-01-05" "2011-01-04" "2011-01-03" "2010-12-31"
> class(date)
[1] "Date"
```

Step 4: Combining *date* and *data.AMZN*

Combine the "Date" class "date" with all columns of data.AMZN except the first column of "character" class "Date":

```
> data.AMZN<-cbind(date, data.AMZN[,-1])
> head(data.AMZN)
          date    Open    High     Low   Close Adj.Close   Volume
755 2013-12-31 19.7290 19.9415 19.6900 19.9395   19.9395 39930000
754 2013-12-30 19.9705 19.9960 19.6225 19.6685   19.6685 49742000
753 2013-12-27 20.2325 20.2815 19.8125 19.9040   19.9040 39738000
752 2013-12-26 20.0895 20.2260 19.8405 20.2195   20.2195 37370000
751 2013-12-24 20.1260 20.1860 19.8185 19.9600   19.9600 27608000
750 2013-12-23 20.1845 20.2500 19.9600 20.1460   20.1460 53190000
> tail(data.AMZN)
        date   Open   High    Low  Close Adj.Close    Volume
6 2011-01-07 9.3940 9.4225 9.1870 9.2745    9.2745 104434000
5 2011-01-06 9.3250 9.3705 9.2625 9.2930    9.2930  63594000
4 2011-01-05 9.2050 9.3725 9.2035 9.3710    9.3710  68376000
3 2011-01-04 9.3075 9.3850 9.1890 9.2505    9.2505 100636000
2 2011-01-03 9.0685 9.3000 9.0605 9.2110    9.2110 106628000
1 2010-12-31 9.0980 9.1150 8.9755 9.0000    9.0000  69038000
```

Step 5: Sort Data in Chronological Order

Now I sort the rows by the order of date time again to make the top row with date of 2010-12-31.

Also from the order it could see that the oldest date has smallest number: “1”, which means it has the correct order as time goes on, which is better than the output in the book:

```
> data.AMZN<-data.AMZN[order(data.AMZN$date),]
> head(data.AMZN)
        date   Open   High    Low  Close Adj.Close    Volume
1 2010-12-31 9.0980 9.1150 8.9755 9.0000    9.0000  69038000
2 2011-01-03 9.0685 9.3000 9.0605 9.2110    9.2110 106628000
3 2011-01-04 9.3075 9.3850 9.1890 9.2505    9.2505 100636000
4 2011-01-05 9.2050 9.3725 9.2035 9.3710    9.3710  68376000
5 2011-01-06 9.3250 9.3705 9.2625 9.2930    9.2930  63594000
6 2011-01-07 9.3940 9.4225 9.1870 9.2745    9.2745 104434000
> tail(data.AMZN)
          date    Open    High     Low   Close Adj.Close   Volume
750 2013-12-23 20.1845 20.2500 19.9600 20.1460   20.1460 53190000
751 2013-12-24 20.1260 20.1860 19.8185 19.9600   19.9600 27608000
752 2013-12-26 20.0895 20.2260 19.8405 20.2195   20.2195 37370000
753 2013-12-27 20.2325 20.2815 19.8125 19.9040   19.9040 39738000
754 2013-12-30 19.9705 19.9960 19.6225 19.6685   19.6685 49742000
755 2013-12-31 19.7290 19.9415 19.6900 19.9395   19.9395 39930000
```

Step 6: Convert from data.frame Object to xts Object data.AMZN is structured as a data.frame class.

```
> class(data.AMZN)
[1] "data.frame"
> library(xts)
载入需要的程序包：zoo

载入程序包：‘zoo’

The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric

> data.AMZN<-xts(data.AMZN[,2:7],order.by=data.AMZN[,1])
> head(data.AMZN)
            Open   High    Low  Close Adj.Close    Volume
2010-12-31 9.0980 9.1150 8.9755 9.0000    9.0000  69038000
2011-01-03 9.0685 9.3000 9.0605 9.2110    9.2110 106628000
2011-01-04 9.3075 9.3850 9.1890 9.2505    9.2505 100636000
2011-01-05 9.2050 9.3725 9.2035 9.3710    9.3710  68376000
2011-01-06 9.3250 9.3705 9.2625 9.2930    9.2930  63594000
2011-01-07 9.3940 9.4225 9.1870 9.2745    9.2745 104434000
> tail(data.AMZN)
            Open    High     Low   Close Adj.Close   Volume
2013-12-23 20.1845 20.2500 19.9600 20.1460   20.1460 53190000
2013-12-24 20.1260 20.1860 19.8185 19.9600   19.9600 27608000
2013-12-26 20.0895 20.2260 19.8405 20.2195   20.2195 37370000
2013-12-27 20.2325 20.2815 19.8125 19.9040   19.9040 39738000
2013-12-30 19.9705 19.9960 19.6225 19.6685   19.6685 49742000
2013-12-31 19.7290 19.9415 19.6900 19.9395   19.9395 39930000
> class(data.AMZN)
[1] "xts" "zoo"
```

Step 7: Rename Variables

```
> names(data.AMZN)
[1] "Open"      "High"      "Low"       "Close"     "Adj.Close" "Volume"
> names(data.AMZN)<-paste(c("AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Adjusted","AMZN.Volume"))
> head(data.AMZN)
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2010-12-31    9.0980    9.1150   8.9755     9.0000        9.0000    69038000
2011-01-03    9.0685    9.3000   9.0605     9.2110        9.2110   106628000
2011-01-04    9.3075    9.3850   9.1890     9.2505        9.2505   100636000
2011-01-05    9.2050    9.3725   9.2035     9.3710        9.3710    68376000
2011-01-06    9.3250    9.3705   9.2625     9.2930        9.2930    63594000
2011-01-07    9.3940    9.4225   9.1870     9.2745        9.2745   104434000
> tail(data.AMZN)
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2013-12-23   20.1845   20.2500  19.9600    20.1460       20.1460    53190000
2013-12-24   20.1260   20.1860  19.8185    19.9600       19.9600    27608000
2013-12-26   20.0895   20.2260  19.8405    20.2195       20.2195    37370000
2013-12-27   20.2325   20.2815  19.8125    19.9040       19.9040    39738000
2013-12-30   19.9705   19.9960  19.6225    19.6685       19.6685    49742000
2013-12-31   19.7290   19.9415  19.6900    19.9395       19.9395    39930000
```

Now I could just use "AMZN. …" to refer to these quantities which match the names in **getSymbols** command.

1.3 Checking the data

1.3.1 Plotting the data

I plot the Amazon.com closing price (AMZN.Close) against the time:

Now I manipulate the data to make an effect that there are some data losses in some dates

and compare these to the above original one to see the missing data effect:

Clearly from the light blue circle part we see that there is a flat group of spots in the missing data plot, which is just due to the missing of important data from no.400 to no.500.

1.3.2 Checking the Dimension

I use a simple code here to output the dimension of the Amazon.com data:

```
> dim(data.AMZN)
[1] 755   6
```

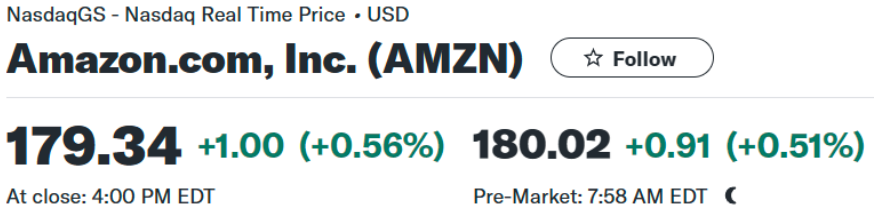
1.3.3 Outputting Summary Statistics

I also use a simple line of code here to output the summary table of statistics of the Amazon.com data:

```
> summary(data.AMZN)
     Index              AMZN.Open        AMZN.High         AMZN.Low        AMZN.Close      AMZN.Adjusted
 Min.   :2010-12-31   Min.   : 8.059   Min.   : 8.177   Min.   : 8.030   Min.   : 8.049   Min.   : 8.049
 1st Qu.:2011-09-29   1st Qu.: 9.642   1st Qu.: 9.770   1st Qu.: 9.511   1st Qu.: 9.665   1st Qu.: 9.665
 Median :2012-06-29   Median :11.329   Median :11.534   Median :11.230   Median :11.367   Median :11.367
 Mean   :2012-06-30   Mean   :11.908   Mean   :12.050   Mean   :11.762   Mean   :11.915   Mean   :11.915
 3rd Qu.:2013-04-03   3rd Qu.:13.336   3rd Qu.:13.472   3rd Qu.:13.189   3rd Qu.:13.322   3rd Qu.:13.322
 Max.   :2013-12-31   Max.   :20.233   Max.   :20.282   Max.   :19.960   Max.   :20.220   Max.   :20.220
  AMZN.Volume
 Min.   : 19688000
 1st Qu.: 53215000
 Median : 74124000
 Mean   : 86390472
 3rd Qu.:103219000
 Max.   :482684000
```

1.3.4 Checking the Ticker Symbol

We double check the ticker symbol here for the Amazon.com again:



This screenshot from Yahoo Finance clearly shows that the ticker symbol for Amazon.com, Inc. is "AMZN" which is on market of Nasdaq Real Time Price by using US Dollars.

The reason that Ticker Symbol should be double checked is that it changes occasionally and may be the same with other companies having similar initial alphabet letters.

1.4 Basic Data Manipulation Techniques

1.4.1 Keeping and Deleting One Row

1: Only keep the first row of data at 2010-12-31:

```
> AMZN.onlyFirst<-data.AMZN[1,]
> AMZN.onlyFirst
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2010-12-31     9.098     9.115   8.9755          9             9    69038000
```

2: Only leave the first row of data and keep all the remaining data:

```
> AMZN.delFirst<-data.AMZN[-1,]
> head(AMZN.delFirst)
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2011-01-03    9.0685    9.3000   9.0605     9.2110        9.2110   106628000
2011-01-04    9.3075    9.3850   9.1890     9.2505        9.2505   100636000
2011-01-05    9.2050    9.3725   9.2035     9.3710        9.3710    68376000
2011-01-06    9.3250    9.3705   9.2625     9.2930        9.2930    63594000
2011-01-07    9.3940    9.4225   9.1870     9.2745        9.2745   104434000
2011-01-10    9.2520    9.2645   9.1255     9.2340        9.2340    67518000
```

We could see that only the first row of data (2010-12-31) here is deleted and all the remaining is kept.

1.4.2 Keeping First and Last Rows

Here we compare the first row data (2010-12-31) and the last row data (2013-12-31) to see

the overall change in this range of time:

```
> data.AMZN[c(1,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2010-12-31     9.098    9.1150   8.9755     9.0000        9.0000    69038000
2013-12-31    19.729   19.9415  19.6900    19.9395       19.9395    39930000
```

1.4.3 Keeping Contiguous Rows

Suppose we need to output the row data of the first week of year 2011, then we need to output row data within a sequence of row numbers.

Here is the solution of manipulating row index in R language:

```
> AMZN.first.week<-data.AMZN[2:6,]
> AMZN.first.week
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2011-01-03    9.0685    9.3000   9.0605     9.2110        9.2110   106628000
2011-01-04    9.3075    9.3850   9.1890     9.2505        9.2505   100636000
2011-01-05    9.2050    9.3725   9.2035     9.3710        9.3710    68376000
2011-01-06    9.3250    9.3705   9.2625     9.2930        9.2930    63594000
2011-01-07    9.3940    9.4225   9.1870     9.2745        9.2745   104434000
```

Similarly, we could use this technique to output the last 30 row data of the total observation:

```
> AMZN.last30<-data.AMZN[(nrow(data.AMZN)-29):nrow(data.AMZN),]
> AMZN.last30
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2013-11-18   18.5140   18.6745  18.2335    18.3090       18.3090    54748000
2013-11-19   18.2910   18.4390  18.1250    18.2470       18.2470    38096000
2013-11-20   18.3780   18.3780  18.0225    18.1285       18.1285    35430000
2013-11-21   18.2025   18.4625  18.1650    18.4460       18.4460    39292000
2013-11-22   18.5000   18.7250  18.3155    18.6155       18.6155    59310000
2013-11-25   18.6910   18.8895  18.6590    18.8320       18.8320    59446000
2013-11-26   18.8805   19.1250  18.7410    19.0685       19.0685    54488000
2013-11-27   19.1750   19.3500  19.1305    19.3355       19.3355    45394000
2013-11-29   19.4550   19.7050  19.4310    19.6810       19.6810    48120000
2013-12-02   19.9500   19.9500  19.4550    19.6150       19.6150    94280000
2013-12-03   19.5055   19.5475  19.1550    19.2330       19.2330    74058000
2013-12-04   19.1750   19.4845  19.0745    19.2980       19.2980    47106000
2013-12-05   19.3325   19.3325  19.0685    19.2245       19.2245    38130000
2013-12-06   19.4175   19.4175  19.1915    19.3475       19.3475    39694000
2013-12-09   19.4055   19.4105  19.1285    19.2445       19.2445    55236000
2013-12-10   19.1870   19.4530  19.1510    19.3890       19.3890    54736000
2013-12-11   19.3670   19.4490  19.1000    19.1095       19.1095    49026000
2013-12-12   19.0630   19.2500  18.9750    19.0625       19.0625    42474000
2013-12-13   19.2660   19.4710  19.1900    19.2120       19.2120    60500000
2013-12-16   19.2515   19.5850  19.2500    19.4485       19.4485    45034000
2013-12-17   19.5325   19.5680  19.3250    19.3825       19.3825    46878000
2013-12-18   19.4615   19.8150  19.1550    19.7980       19.7980    69782000
2013-12-19   19.7135   19.8645  19.6300    19.7595       19.7595    48544000
2013-12-20   19.8275   20.2360  19.7890    20.1100       20.1100   100678000
2013-12-23   20.1845   20.2500  19.9600    20.1460       20.1460    53190000
2013-12-24   20.1260   20.1860  19.8185    19.9600       19.9600    27608000
2013-12-26   20.0895   20.2260  19.8405    20.2195       20.2195    37370000
2013-12-27   20.2325   20.2815  19.8125    19.9040       19.9040    39738000
2013-12-30   19.9705   19.9960  19.6225    19.6685       19.6685    49742000
2013-12-31   19.7290   19.9415  19.6900    19.9395       19.9395    39930000
> nrow(AMZN.last30)
[1] 30
```

1.4.4 Keeping First Three Rows and Last Row

I use colon or ":" and comma or "," together in c() to make the selection of keeping both continuous and discrete rows:

```
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2010-12-31    9.0980    9.1150   8.9755     9.0000        9.0000    69038000
2011-01-03    9.0685    9.3000   9.0605     9.2110        9.2110   106628000
2011-01-04    9.3075    9.3850   9.1890     9.2505        9.2505   100636000
2013-12-31   19.7290   19.9415  19.6900    19.9395       19.9395    39930000
```

1.4.5 Keeping and Deleting One Column

I will provide two different ways of extracting one specific column in AMZN data.

I output the names of columns first:

```
> names(data.AMZN)
[1] "AMZN.Open"     "AMZN.High"     "AMZN.Low"      "AMZN.Close"    "AMZN.Adjusted" "AMZN.Volume"
```

The first way is to use index after comma to represent the number of column and use group of indices before comma to select row data in terms of the number of rows:

```
> AMZN.onlyPrice<-data.AMZN[,4]
> AMZN.onlyPrice[c(1:3,nrow(AMZN.onlyPrice)),]
           AMZN.Close
2010-12-31     9.0000
2011-01-03     9.2110
2011-01-04     9.2505
2013-12-31    19.9395
```

The second way is to just nominate the column name we want to keep, like "AMZN.Close":

```
> AMZN.onlyPrice2<-data.AMZN$AMZN.Close
> AMZN.onlyPrice2[c(1:3,nrow(AMZN.onlyPrice2)),]
           AMZN.Close
2010-12-31     9.0000
2011-01-03     9.2110
2011-01-04     9.2505
2013-12-31    19.9395
```

Since we need to delete the column "AMZN.Adjusted" here in the model of OHLC financially, hence we only need to type [, -5] to eliminate Adjusted:

```
> AMZN.delAdjPrice<-data.AMZN[,-5]
> AMZN.delAdjPrice[c(1:3,nrow(AMZN.delAdjPrice)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000
```

1.4.6 Keeping Non-Contiguous Columns

Here we use "c()" in R language to combine non-contiguous columns to analyze different features data:

```
> AMZN.OpenClose<-data.AMZN[,c(1,4)]
> AMZN.OpenClose[c(1:3,nrow(AMZN.OpenClose)),]
           AMZN.Open AMZN.Close
2010-12-31    9.0980     9.0000
2011-01-03    9.0685     9.2110
2011-01-04    9.3075     9.2505
2013-12-31   19.7290    19.9395
```

1.4.7 Keeping Contiguous Columns

Here we use "c()" in R language to combine contiguous columns to analyze different features data in a sequence:

```
> AMZN.PriceVol<-data.AMZN[,4:5]
> AMZN.PriceVol[c(1:3,nrow(AMZN.PriceVol)),]
           AMZN.Close AMZN.Adjusted
2010-12-31     9.0000        9.0000
2011-01-03     9.2110        9.2110
2011-01-04     9.2505        9.2505
2013-12-31    19.9395       19.9395
```

1.4.8 Keeping Contiguous and Non-Contiguous Columns

Here we use ":" and "," in "c()" to combine both continuous and discrete features together to do analysis on AMZN data:

```
> AMZN.OpenCloseVol<-data.AMZN[,c(1,4:5)]
> AMZN.OpenCloseVol[c(1:3,nrow(AMZN.OpenCloseVol)),]
           AMZN.Open AMZN.Close AMZN.Adjusted
2010-12-31    9.0980     9.0000        9.0000
2011-01-03    9.0685     9.2110        9.2110
2011-01-04    9.3075     9.2505        9.2505
2013-12-31   19.7290    19.9395       19.9395
```

Also, we could do this by using "-" in "c()" to delete intermediate columns:

```
> AMZN.OpenCloseVol<-data.AMZN[,c(-2:-3,-6)]
> AMZN.OpenCloseVol[c(1:3,nrow(AMZN.OpenCloseVol)),]
           AMZN.Open AMZN.Close AMZN.Adjusted
2010-12-31    9.0980     9.0000        9.0000
2011-01-03    9.0685     9.2110        9.2110
2011-01-04    9.3075     9.2505        9.2505
2013-12-31   19.7290    19.9395       19.9395
```

1.4.9 Subsetting Rows and Columns

We combine row information before comma and column information after comma together to use the subset of both rows and columns, here we only output the first three rows and last rows:

```
> data.vwap<-data.AMZN[((nrow(data.AMZN)-29)):nrow(data.AMZN),c(4,5)]
> data.vwap[c(1:3,nrow(data.vwap)),]
           AMZN.Close AMZN.Adjusted
2013-11-18    18.3090       18.3090
2013-11-19    18.2470       18.2470
2013-11-20    18.1285       18.1285
2013-12-31    19.9395       19.9395
```

This is the information of AMZN data of the last 30 days with 4th and 5th columns.

1.4.10 Sub-setting Using Dates

Since dates are more obvious and tangible in time series analysis, so using dates in R language is more used apparently. Here It should be noted that the "Date" here is an "xts" object.

Here we define all information of the year 2012 and output the first three row information and the last day information, which is accompanied by the output of the class of "data.AMZN" to show the trivial "xts" class:

```
> class(data.AMZN)
[1] "xts" "zoo"
> xts.data.AMZN.2012<-subset(data.AMZN[,4],index(data.AMZN) >= "2012-01-01" &index(data.AMZN) <= "2012-12-31")
> xts.data.AMZN.2012[c(1:3,nrow(xts.data.AMZN.2012))]
           AMZN.Close
2012-01-03     8.9515
2012-01-04     8.8755
2012-01-05     8.8805
2012-12-31    12.5435
```

As in some cases we need to transform the data class to be "data.frame" object to perform some data analysis, hence we also transfer the data class from "xts" here to be "data.frame":

```
> class(index(data.AMZN))
[1] "Date"
> data.frame.AMZN.2012<-cbind(index(data.AMZN),data.frame(data.AMZN[,4]))
> data.frame.AMZN.2012[c(1:3,nrow(data.frame.AMZN.2012)),]
           index(data.AMZN) AMZN.Close
2010-12-31       2010-12-31     9.0000
2011-01-03       2011-01-03     9.2110
2011-01-04       2011-01-04     9.2505
2013-12-31       2013-12-31    19.9395
> class(data.frame.AMZN.2012)
[1] "data.frame"
```

There are two main problems here in the above table. The first problem is that the name of

the first column is “index(data.AMZN)”, which is not a meaningful name. The second problem is that the index is also in the form of date, which is redundant compared to first column, so we need to change the name of first column and change the form of the index:

```
> names(data.frame.AMZN.2012)[1]<-paste("date")
> rownames(data.frame.AMZN.2012)<-seq(1,nrow(data.frame.AMZN.2012),1)
> data.frame.AMZN.2012[c(1:3,nrow(data.frame.AMZN.2012)),]
          date AMZN.Close
1   2010-12-31     9.0000
2   2011-01-03     9.2110
3   2011-01-04     9.2505
755 2013-12-31    19.9395
```

Now we apply the above subnet technique in this data.frame form again to make the year 2012 data demonstration:

```
> data.frame.AMZN.2012<-subset(data.frame.AMZN.2012,data.frame.AMZN.2012$date >= "2012-01-01"&data.frame.AMZN.
2012$date <= "2012-12-31")
> data.frame.AMZN.2012[c(1:3,nrow(data.frame.AMZN.2012)),]
          date AMZN.Close
254 2012-01-03     8.9515
255 2012-01-04     8.8755
256 2012-01-05     8.8805
503 2012-12-31    12.5435
```

1.4.11 Converting Daily Prices to Weekly and Monthly Prices

We need to convert daily data into weekly or monthly data to get summarized data analysis, so here we transfer the AMZN data into weekly and monthly form, with the help from “xts” class object:

1. Weekly Price:

```
> wk<-data.AMZN
> wk<-wk[,c(1,2,3,4,6,5)]
> data.weekly<-to.weekly(wk)
> data.weekly[c(1:3,nrow(data.weekly)),]
           wk.Open wk.High  wk.Low wk.Close wk.Volume wk.Adjusted
2010-12-31  9.0980  9.1150  8.9755   9.0000  69038000      9.0000
2011-01-07  9.0685  9.4225  9.0605   9.2745 443668000      9.2745
2011-01-14  9.2520  9.4470  9.1255   9.4375 317980000      9.4375
2013-12-31 19.9705 19.9960 19.6225  19.9395  89672000     19.9395
```

Also, we use the original data to verify the results weekly, whose data is ranged from 2011-01-03 to 2011-01-07:

```
> data.AMZN[2:6,]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Adjusted AMZN.Volume
2011-01-03    9.0685    9.3000   9.0605     9.2110        9.2110   106628000
2011-01-04    9.3075    9.3850   9.1890     9.2505        9.2505   100636000
2011-01-05    9.2050    9.3725   9.2035     9.3710        9.3710    68376000
2011-01-06    9.3250    9.3705   9.2625     9.2930        9.2930    63594000
2011-01-07    9.3940    9.4225   9.1870     9.2745        9.2745   104434000
> sum(data.AMZN[2:6,6])
[1] 443668000
```

We could see that the weekly Volume is matched with the Volume of week 2011-01-07, with the number of 443668000.

2. Monthly Price:

```
> mo<-data.AMZN
> mo<-mo[,c(1,2,3,4,6,5)]
> data.monthly<-to.monthly(mo)
> data.monthly[c(1:3,nrow(data.monthly)),]
           mo.Open mo.High  mo.Low mo.Close  mo.Volume mo.Adjusted
12月 2010  9.0980  9.1150  8.9755   9.0000   69038000       9.0000
1月 2011   9.0685  9.5800  8.3450   8.4820 2272226000       8.4820
2月 2011   8.5260  9.5700  8.4755   8.6645 1915528000       8.6645
12月 2013 19.9500 20.2815 18.9750  19.9395 1113734000      19.9395
```

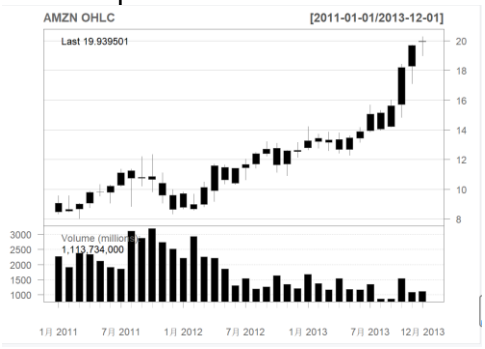
We could see the data is correct especially for December 2010 since only one day from that month is counted when summarizing volume.

3. Plotting a Candlestick Chart Using Monthly Data

We would need to first convert the data into an open-high-low-close (OHLC) object, which means we will present a OHLC table of monthly data:

```
> library(quantmod)
载入需要的程序包：TTR
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
> OHLC<-data.monthly[-1,-6]
> AMZN.ohlc<-as.quantmod.OHLC(OHLC,col.names=c("Open","High","Low","Close","Volume"))
> class(AMZN.ohlc)
[1] "quantmod.OHLC" "zoo"
> AMZN.ohlc[c(1:3,nrow(AMZN.ohlc)),]
          OHLC.Open OHLC.High OHLC.Low OHLC.Close OHLC.Volume
1月 2011     9.0685    9.5800   8.3450     8.4820  2272226000
2月 2011     8.5260    9.5700   8.4755     8.6645  1915528000
3月 2011     8.6765    9.0785   8.0295     9.0065  2379582000
12月 2013   19.9500   20.2815  18.9750    19.9395  1113734000
>
```

Now we plot the OHLC Chart with above line bars and below bar charts:



Here the only one day data December 2010 is dropped and Adjusted column is also dropped for redundant.

1.5 Comparing Capital Gains of Multiple Securities Over Time

At first, we clear out all the old data objects and show again to verify that all old objects have been cleared out:

```
> ###show data objects in memory of R
> ls()
 [1] "AMZN.delAdjPrice"     "AMZN.delFirst"        "AMZN.first.week"      "AMZN.last30"
 [5] "AMZN.ohlc"            "AMZN.onlyFirst"       "AMZN.onlyPrice"       "AMZN.onlyPrice2"
 [9] "AMZN.OpenClose"       "AMZN.OpenCloseVol"    "AMZN.PriceVol"        "data.AMZN"
[13] "data.frame.AMZN.2012" "data.missing"         "data.monthly"         "data.vwap"
[17] "data.weekly"          "date"                 "mo"                   "OHLC"
[21] "wk"                   "xts.data.AMZN.2012"
> ###delete all the objects
> rm(list=ls())
> ###show again
> ls()
character(0)
```

Here we combine 4 securities with the assistance of data from Yahoo Finance within the time range from December 31, 2010 to December 31, 2013 to find out which one is the best security for investment.

The four chosen security is AMZN, IBM, YHOO and ^GSPC.

We download these four csv data files and begin to plot the combined graph.

I am really sorry that Yahoo has officially exited the stock market many years ago, so I have to change YHOO to AAPL.

Hope Apple will not fail and exit the market in the future (lol).

Also ^GSPC cannot be downloaded from Yahoo Finance since it is a market index, we change it into NVIDIA (NVDA).

Now we import the data from the csv files.

Step 1: Import Data for Each of the Four Securities:

```
> data.AMZN<-read.csv("AMZN Yahooo.csv",header=TRUE)
> data.AMZN<-data.AMZN[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.AMZN$Date,format="%Y-%m-%d")
> data.AMZN<-cbind(date, data.AMZN[,-1])
> data.AMZN<-data.AMZN[order(data.AMZN$date),]
> data.AMZN<-xts(data.AMZN[,2:7],order.by=data.AMZN[,1])
> names(data.AMZN)<-paste(c("AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"))
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
```

This is the example from AMZN, other securities follow this style similarly.

This is AAPL:

```
> ###AAPL
> data.AAPL<-read.csv("AAPL Yahooo.csv",header=TRUE)
> data.AAPL<-data.AAPL[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.AAPL$Date,format="%Y-%m-%d")
> data.AAPL<-cbind(date, data.AAPL[,-1])
> data.AAPL<-data.AAPL[order(data.AAPL$date),]
> data.AAPL<-xts(data.AAPL[,2:7],order.by=data.AAPL[,1])
> names(data.AAPL)<-paste(c("AAPL.Open","AAPL.High","AAPL.Low","AAPL.Close","AAPL.Volume","AAPL.Adjusted"))
> data.AAPL[c(1:3,nrow(data.AAPL)),]
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2010-12-31  11.53393  11.55286 11.47536   11.52000   193508000      9.739614
2011-01-03  11.63000  11.79500 11.60143   11.77036   445138400      9.951281
2011-01-04  11.87286  11.87500 11.71964   11.83179   309080800     10.003214
2013-12-31  19.79179  20.04571 19.78571   20.03643   223084400     17.519623
```

This is IBM:

```
> ###IBM
> data.IBM<-read.csv("IBM Yahooo.csv",header=TRUE)
> data.IBM<-data.IBM[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.IBM$Date,format="%Y-%m-%d")
> data.IBM<-cbind(date, data.IBM[,-1])
> data.IBM<-data.IBM[order(data.IBM$date),]
> data.IBM<-xts(data.IBM[,2:7],order.by=data.IBM[,1])
> names(data.IBM)<-paste(c("IBM.Open","IBM.High","IBM.Low","IBM.Close","IBM.Volume","IBM.Adjusted"))
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
```

This is NVDA:

```
> ###NVDA
> data.NVDA<-read.csv("NVDA Yahooo.csv",header=TRUE)
> data.NVDA<-data.NVDA[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.NVDA$Date,format="%Y-%m-%d")
> data.NVDA<-cbind(date, data.NVDA[,-1])
> data.NVDA<-data.NVDA[order(data.NVDA$date),]
> data.NVDA<-xts(data.NVDA[,2:7],order.by=data.NVDA[,1])
> names(data.NVDA)<-paste(c("NVDA.Open","NVDA.High","NVDA.Low","NVDA.Close","NVDA.Volume","NVDA.Adjusted"))
> data.NVDA[c(1:3,nrow(data.NVDA)),]
           NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume NVDA.Adjusted
2010-12-31    3.7500    3.8550    3.745     3.8500    39125200      3.531613
2011-01-03    3.8800    3.9925    3.875     3.9550    81744800      3.627930
2011-01-04    3.9625    3.9800    3.855     3.9425    65138400      3.616464
2013-12-31    4.0000    4.0250    3.975     4.0050    23577600      3.778282
>
```

And here are my comments for the entire codes for NVDA, which is the same for other securities.

```
> ###NVDA
> ###read csv file
> data.NVDA<-read.csv("NVDA Yahooo.csv",header=TRUE)
>
> ###Change order of the columns in raw data
> data.NVDA<-data.NVDA[,c(1,2,3,4,5,7,6)]
>
> ###Change Date column to be "Date" class object with form of date
> date<-as.Date(data.NVDA$Date,format="%Y-%m-%d")
>
> ###Replace the first column by the "Date" class object
> data.NVDA<-cbind(date, data.NVDA[,-1])
>
> ###Order the rows by the "Date" class order
> data.NVDA<-data.NVDA[order(data.NVDA$date),]
>
> ###Make the data to be "xts" object,
> ###where the first column "Date" object transferred to be order and left columns to be variables
> data.NVDA<-xts(data.NVDA[,2:7],order.by=data.NVDA[,1])
>
> ###Rename the variable columns
> names(data.NVDA)<-paste(c("NVDA.Open","NVDA.High","NVDA.Low","NVDA.Close","NVDA.Volume","NVDA.Adjusted"))
>
> ###output first three rows and last rows
> data.NVDA[c(1:3,nrow(data.NVDA)),]
```

Step 2: Combine Data into One Data Object:

Here we only need the "Close" data columns of the four securities. Hence, we use the only column required to be combined together to build a new "Close.Prices" data.

```
> ###Combine the Close columns of the four securities
> Close.Prices<-data.AMZN$AMZN.Close
> Close.Prices<-cbind(Close.Prices,data.NVDA$NVDA.Close,
+                     + data.AAPL$AAPL.Close,data.IBM$IBM.Close)
> Close.Prices[c(1:3,nrow(Close.Prices)),]
           AMZN.Close NVDA.Close AAPL.Close IBM.Close
2010-12-31     9.0000      3.8500   11.52000  140.3059
2011-01-03     9.2110      3.9550   11.77036  140.9943
2011-01-04     9.2505      3.9425   11.83179  141.1472
2013-12-31    19.9395      4.0050   20.03643  179.3212
```

Step 3: Convert Data into a "data.frame" object.

```
> ###combine the index (date above) and the whole table together and transform it into "data.frame" class
> multi.df<-cbind(index(Close.Prices),
+                 + data.frame(Close.Prices))
>
> ###Give the names in a simple way
> names(multi.df)<-paste(c("date","AMZN","NVDA","AAPL","IBM"))
>
> ###Give the index from 1 to 755
> rownames(multi.df)<-seq(1,nrow(multi.df),1)
>
> ###Output the first three rows and last row
> multi.df[c(1:3,nrow(multi.df)),]
          date    AMZN   NVDA     AAPL      IBM
1   2010-12-31  9.0000 3.8500 11.52000 140.3059
2   2011-01-03  9.2110 3.9550 11.77036 140.9943
3   2011-01-04  9.2505 3.9425 11.83179 141.1472
755 2013-12-31 19.9395 4.0050 20.03643 179.3212
```

Step 4: Calculate Normalized Values for Each Security.

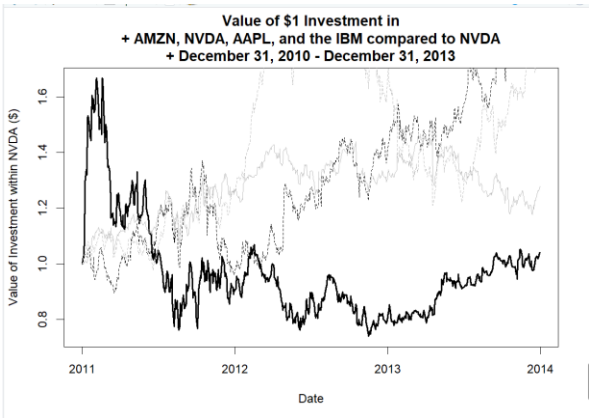
Here normalized means we divide every close price by the price of December 31, 2010 for each security in the data, which will not affect the increasing rate for each security. The advantage is that the numeric value of close prices will not outclass too much to compose too many outliers.

```
> ###Normalized by dividing the first price for each security
> multi.df$AMZN.idx<-multi.df$AMZN/multi.df$AMZN[1]
> multi.df$NVDA.idx<-multi.df$NVDA/multi.df$NVDA[1]
> multi.df$AAPL.idx<-multi.df$AAPL/multi.df$AAPL[1]
> multi.df$IBM.idx<-multi.df$IBM/multi.df$IBM[1]
> options(digits=5)
> multi.df[c(1:3,nrow(multi.df)),]
          date    AMZN   NVDA   AAPL    IBM AMZN.idx NVDA.idx AAPL.idx IBM.idx
1   2010-12-31  9.0000 3.8500 11.520 140.31   1.0000   1.0000   1.0000  1.0000
2   2011-01-03  9.2110 3.9550 11.770 140.99   1.0234   1.0273   1.0217  1.0049
3   2011-01-04  9.2505 3.9425 11.832 141.15   1.0278   1.0240   1.0271  1.0060
755 2013-12-31 19.9395 4.0050 20.036 179.32   2.2155   1.0403   1.7393  1.2781
```

Step 5: Plot the Capital Appreciation of Each Security

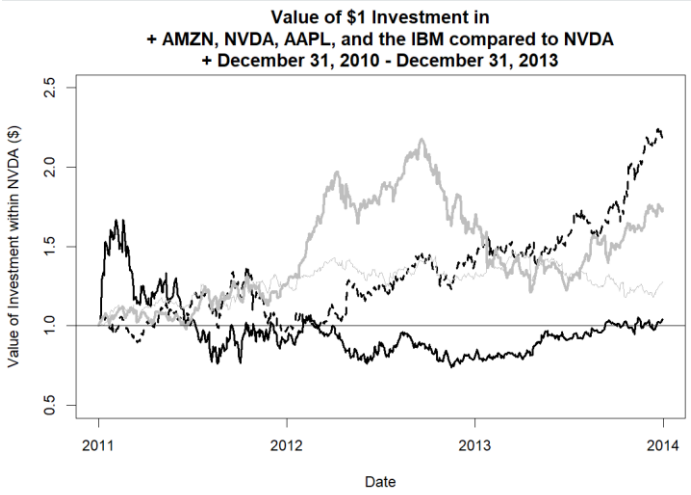
To plot the comparison of four securities, we need to select the first basic security to chart the data, which should be the smallest data in relevant, namely NVDA. This is the first section:

The next step is including all other three securities by plotting the lines with similar arguments. This is the second section:

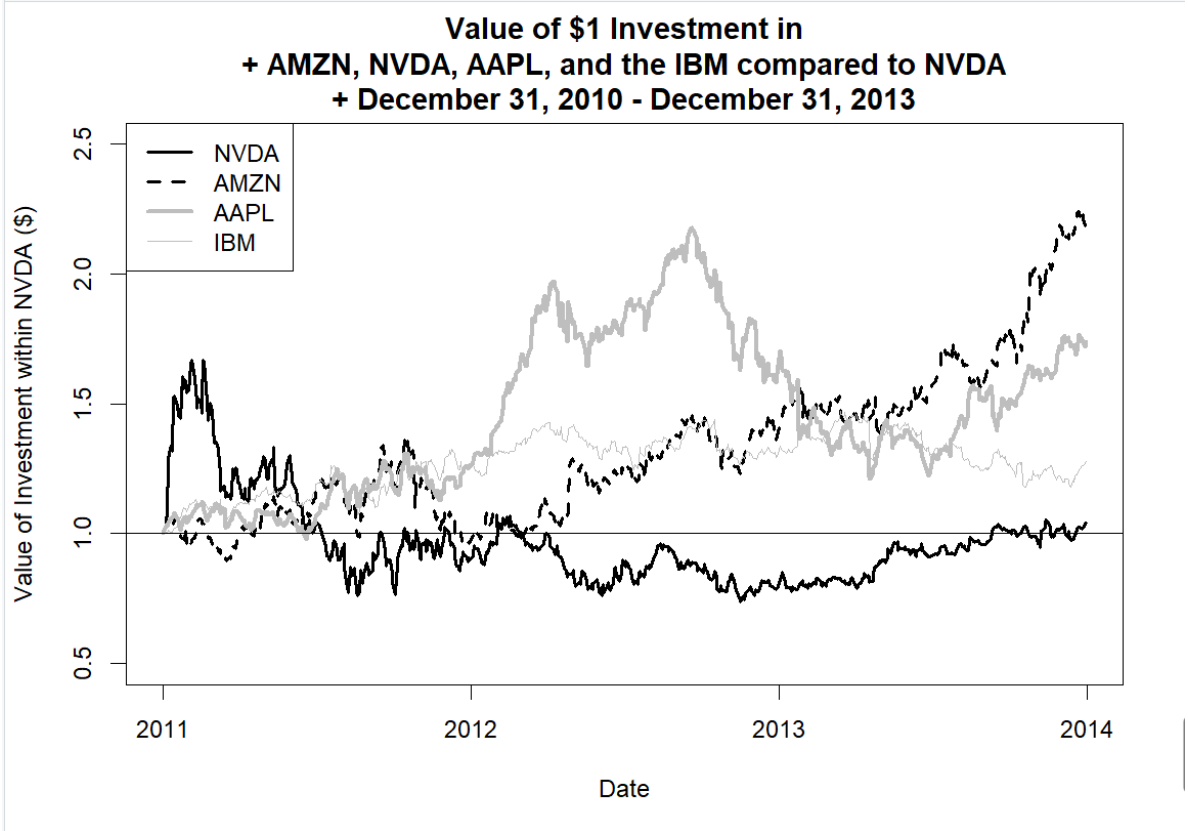


Now we can see that some lines are out of range, hence we need to add “ylim” parameter in plot and change the range.

I also add a horizonal line with value 1 to mark the beginning line. This is the third section.



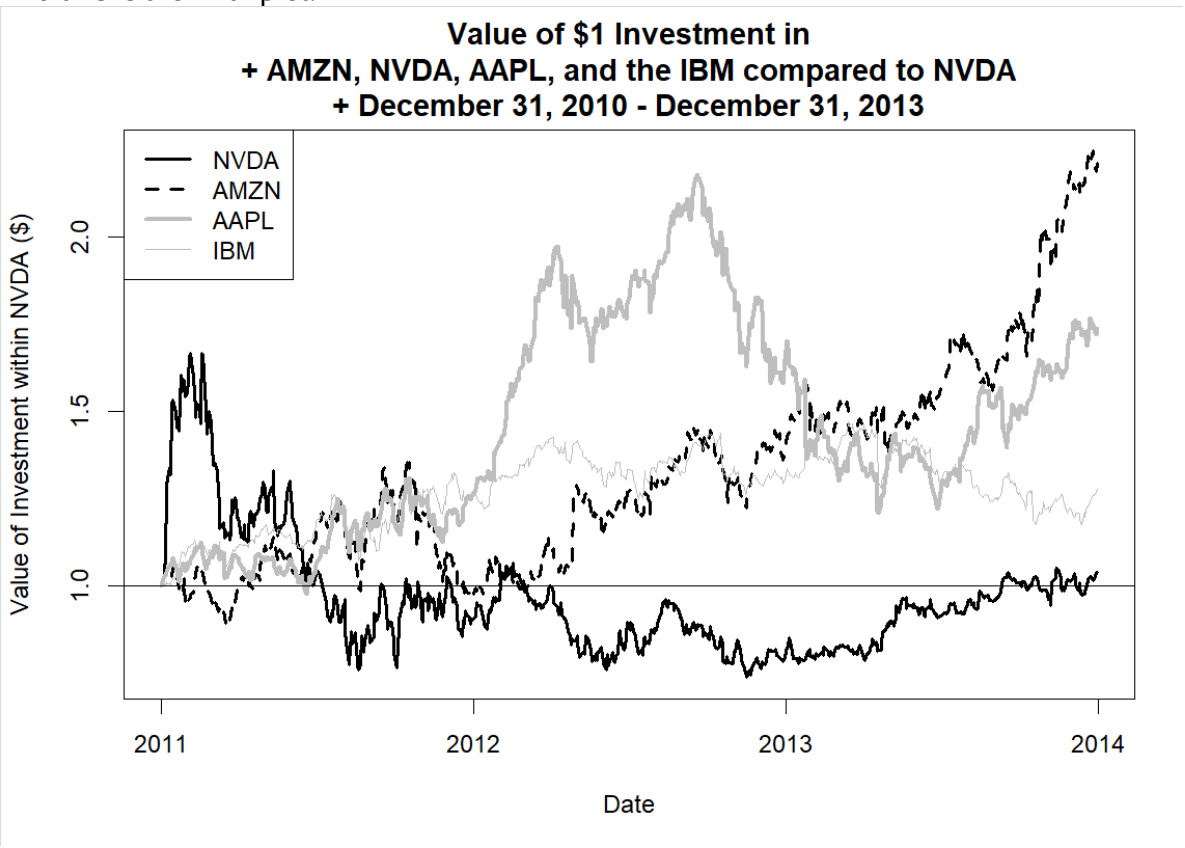
Now we add the legend for each line of each security to distinguish each other. This is the final section.



Step 6: Fix the y-axis to Encompass Range of Normalized Values for All Securities:

Here we calculate the y range for all 4 securities to calculate the minimum and maximum to fit the graph in the suitable range, and insert this range in the "ylim" parameter in the first plot arguments.

And this is the final plot:



1.5.1 Alternative Presentation of Normalized Price Chart

Another approach is that we plot four similar mini-graphs, where each security is highlighted by a dark bond line with all other three grey and thin lines, which could also be an effective plot.

Step 1: Setup Chart Layout in R

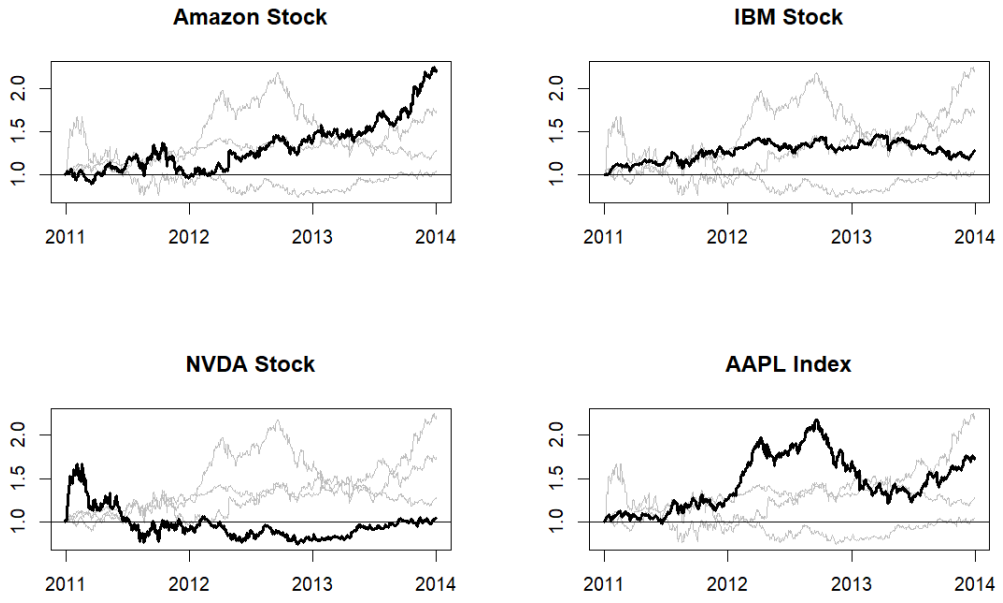
```
> ###set the top margin
> par(oma=c(0,0,3,0))
```

Step 2: Let R Know We Will Be Plotting 4 Charts with 2 Charts in Each Column and 2 Charts in Each Row

```
> ###Set the outlet of mini-plot
> par(mfrow=c(2,2))
```

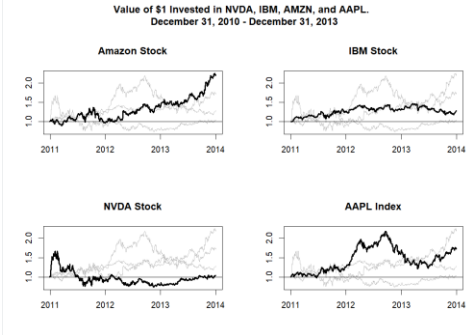
Step 3: Create the 4 Plots

Now we plot the four individual mini-graphs.



Step 4: Create a Global Title for the Charts

Use “outer = T” here to set a global title in the upper margin settled before.



1.6 Technical Analysis Examples

The major understanding of stock prices up and down should be related to demand and supply. More obviously, if demand surpasses the supply then the price is going to increase, if not then the price is likely to go down. The effectiveness of technical analysis is just to predict and follow these possible trends at early stage.

Here are three major types of technical indicators, trend indicator, volatility indicator and momentum indicator. Normally the analysis is made in composition of various indicators at the same time.

1.6.1 Trend: Simple Moving Average Crossover

Here we use the moving average to estimate the coming trend indicator. Typically it is called SMA (Simple Moving Average) such that it treats all future days the same, regardless of near or far away.

We use the crossover of two SMA lines to do the prediction.

Step 1: Obtain Closing Prices for Amazon.com Stock

```
> AMZN.sma<-data.AMZN[,4]
> AMZN.sma[c(1:3,nrow(AMZN.sma)),]
           AMZN.Close
2010-12-31     9.0000
2011-01-03     9.2110
2011-01-04     9.2505
2013-12-31    19.9395
>
```

Step 2: Calculate the Rolling 50-Day and 200-DayAverage Price

```
> AMZN.sma$sma50<-rollmeanr(AMZN.sma$AMZN.Close,k=50)
> AMZN.sma$sma200<-rollmeanr(AMZN.sma$AMZN.Close,k=200)
> AMZN.sma[c(1:3,nrow(AMZN.sma)),]
           AMZN.Close  sma50 sma200
2010-12-31     9.0000     NA     NA
2011-01-03     9.2110     NA     NA
2011-01-04     9.2505     NA     NA
2013-12-31    19.9395 18.645 15.304
```

To show that sma50 begins at the 50th date, we output row data from 48th to 52nd date:

```
> AMZN.sma[48:52,]
           AMZN.Close  sma50 sma200
2011-03-10     8.3070     NA     NA
2011-03-11     8.4035     NA     NA
2011-03-14     8.3365 8.9602     NA
2011-03-15     8.2540 8.9453     NA
2011-03-16     8.2350 8.9257     NA
```

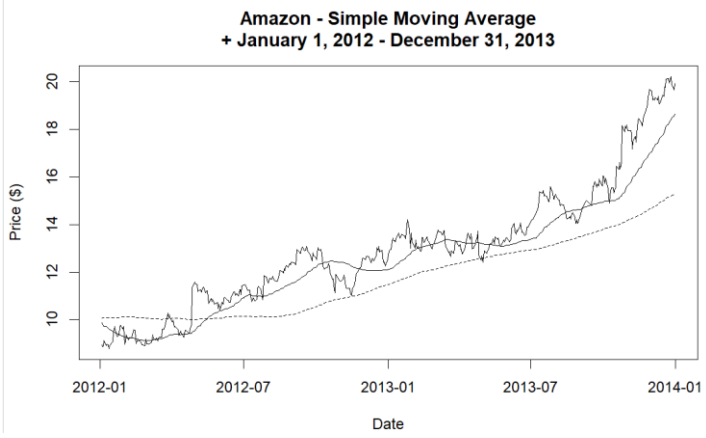
Similarly show again the 198th to 202nd date to show the sma200 beginning at 200th date:

```
> AMZN.sma[198:202,]
           AMZN.Close  sma50 sma200
2011-10-12     11.841 10.636     NA
2011-10-13     11.807 10.663     NA
2011-10-14     12.335 10.708 9.7692
2011-10-17     12.117 10.748 9.7848
2011-10-18     12.194 10.798 9.7997
>
```

Step 3: Subset to Only Show 2012 and 2013 Data

```
> AMZN.sma2012<-subset(AMZN.sma,index(AMZN.sma)>="2012-01-01")
> AMZN.sma2012[c(1:3,nrow(AMZN.sma2012)),]
           AMZN.Close   sma50 sma200
2012-01-03     8.9515  9.8923 10.077
2012-01-04     8.8755  9.8350 10.080
2012-01-05     8.8805  9.7750 10.084
2013-12-31    19.9395 18.6454 15.304
>
```

Step 4: Plot the SMA



1.6.2 Volatility: Bollinger Bands

The Bollinger Bands have three components. The first component is a 20-day simple moving average (SMA). The second component is an upper band, which is two standard deviations above the 20-day SMA. The third component is a lower band, which is two standard deviations below the 20-day SMA.

Bollinger Bands are considered as volatility indicators because it widens or narrows volatility in the stock market.

Step 1: Obtain Closing Prices for Amazon.com Stock

```
> AMZN.bb<-data.AMZN[,4]
> AMZN.bb[c(1:3,nrow(AMZN.bb)),]
           AMZN.Close
2010-12-31     9.0000
2011-01-03     9.2110
2011-01-04     9.2505
2013-12-31    19.9395
```

Step 2: Calculate Rolling 20-Day Mean and Standard Deviation.

```
> AMZN.bb$avg<-rollmeanr(AMZN.bb$AMZN.Close,k=20)
> AMZN.bb$sd<-rollapply(AMZN.bb$AMZN.Close,width=20,FUN=sd,fill=NA)
> AMZN.bb[c(1:3,nrow(AMZN.bb)),]
           AMZN.Close    avg      sd
2010-12-31     9.0000     NA      NA
2011-01-03     9.2110     NA      NA
2011-01-04     9.2505     NA      NA
2013-12-31    19.9395 19.573 0.37597
```

Now we output the 18th to 22nd row data since the calculation is from 20-day interval.

```
> AMZN.bb[18:22,]
            AMZN.Close    avg      sd
2011-01-26     8.7695     NA      NA
2011-01-27     9.2225     NA      NA
2011-01-28     8.5570 9.1435 0.25309
2011-01-31     8.4820 9.1176 0.29205
2011-02-01     8.6055 9.0873 0.31253
```

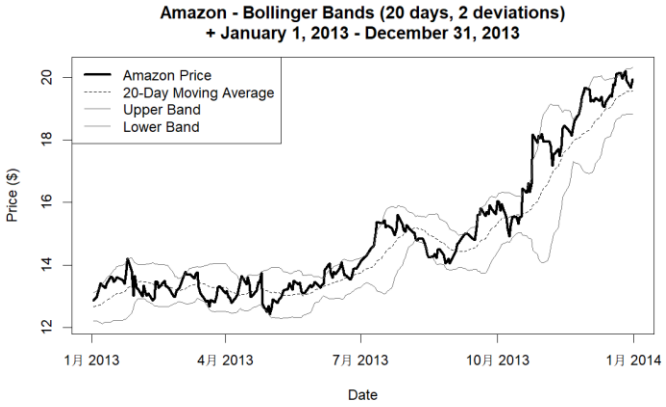
Step 3: Subset to Only Show 2013 Data.

```
> AMZN.bb2013<-subset(AMZN.bb,index(AMZN.bb)>="2013-01-01")
> AMZN.bb2013[c(1:3,nrow(AMZN.bb2013)),]
           AMZN.Close    avg      sd
2013-01-02     12.866 12.658 0.22290
2013-01-03     12.924 12.673 0.23044
2013-01-04     12.957 12.686 0.23905
2013-12-31     19.940 19.573 0.37597
```

Step 4: Calculate the Bollinger Bands.

```
> AMZN.bb2013$sd2up<-AMZN.bb2013$avg+2*AMZN.bb2013$sd
> AMZN.bb2013$sd2down<-AMZN.bb2013$avg-2*AMZN.bb2013$sd
> AMZN.bb2013[c(1:3,nrow(AMZN.bb2013)),]
           AMZN.Close    avg      sd  sd2up sd2down
2013-01-02     12.866 12.658 0.22290 13.104  12.213
2013-01-03     12.924 12.673 0.23044 13.134  12.212
2013-01-04     12.957 12.686 0.23905 13.164  12.208
2013-12-31     19.940 19.573 0.37597 20.325  18.821
```

Step 5: Plot the Bollinger Bands.



1.6.3 Momentum: Relative Strength Index

A common technical analysis momentum indicator is the Relative Strength Index (RSI). The typical calculation is to use a 14-day period. The RSI is calculated as:

$$RSI = 100 - 100/(1 + RS).$$

where RS is equal to the up average divided by the down average with the averages calculated using the Wilder Exponential Moving Average described below.

The RSI is used in conjunction with an overbought line and an oversold line. The overbought line is typically set at a level of 70 and the oversold line is typically set at a level of 30. A buy signal is created when the RSI rises from below the oversold line and crosses the oversold line. Conversely, a sell signal is created when the RSI falls from above the overbought line and crosses the overbought line.

Step 1: Obtain Closing Prices for Amazon.com Stock

```
> AMZN.RSI<-data.AMZN[,4]
> AMZN.RSI$delta<-diff(AMZN.RSI$AMZN.Close)
> AMZN.RSI[c(1:3,nrow(AMZN.RSI)),]
           AMZN.Close  delta
2010-12-31     9.0000     NA
2011-01-03     9.2110 0.2110
2011-01-04     9.2505 0.0395
2013-12-31    19.9395 0.2710
```

Step 2: Create Dummy Variables to Indicate Whether Price Went Up or Price Went Down

```
> AMZN.RSI$up<-ifelse(AMZN.RSI$delta>0,1,0)
> AMZN.RSI$down<-ifelse(AMZN.RSI$delta<0,1,0)
> AMZN.RSI[c(1:3,nrow(AMZN.RSI)),]
           AMZN.Close  delta up down
2010-12-31     9.0000     NA NA   NA
2011-01-03     9.2110 0.2110  1    0
2011-01-04     9.2505 0.0395  1    0
2013-12-31    19.9395 0.2710  1    0
```

Step 3: Calculate Prices for Up Days and Prices for Down Days

```
> AMZN.RSI$up.val<-AMZN.RSI$delta*AMZN.RSI$up
> AMZN.RSI$down.val<--AMZN.RSI$delta*AMZN.RSI$down
> AMZN.RSI<-AMZN.RSI[-1,]
> AMZN.RSI[c(1:3,nrow(AMZN.RSI)),]
           AMZN.Close  delta up down up.val down.val
2011-01-03     9.2110 0.2110  1    0 0.2110        0
2011-01-04     9.2505 0.0395  1    0 0.0395        0
2011-01-05     9.3710 0.1205  1    0 0.1205        0
2013-12-31    19.9395 0.2710  1    0 0.2710        0
```

Step 4: Calculate Initial Up and Down 14-Day Averages

```
> AMZN.RSI$up.first.avg<-rollapply(AMZN.RSI$up.val,width=14,FUN=mean,fill=NA,na.rm=TRUE)
> AMZN.RSI$down.first.avg<-rollapply(AMZN.RSI$down.val,width=14,FUN=mean,fill=NA,na.rm=TRUE)
> AMZN.RSI[c(1:15,nrow(AMZN.RSI)),]
           AMZN.Close   delta up down up.val down.val up.first.avg down.first.avg
2011-01-03     9.2110  0.2110  1    0 0.2110   0.0000           NA             NA
2011-01-04     9.2505  0.0395  1    0 0.0395   0.0000           NA             NA
2011-01-05     9.3710  0.1205  1    0 0.1205   0.0000           NA             NA
2011-01-06     9.2930 -0.0780  0    1 0.0000   0.0780           NA             NA
2011-01-07     9.2745 -0.0185  0    1 0.0000   0.0185           NA             NA
2011-01-10     9.2340 -0.0405  0    1 0.0000   0.0405           NA             NA
2011-01-11     9.2170 -0.0170  0    1 0.0000   0.0170           NA             NA
2011-01-12     9.2040 -0.0130  0    1 0.0000   0.0130           NA             NA
2011-01-13     9.2765  0.0725  1    0 0.0725   0.0000           NA             NA
2011-01-14     9.4375  0.1610  1    0 0.1610   0.0000           NA             NA
2011-01-18     9.5625  0.1250  1    0 0.1250   0.0000           NA             NA
2011-01-19     9.3435 -0.2190  0    1 0.0000   0.2190           NA             NA
2011-01-20     9.0980 -0.2455  0    1 0.0000   0.2455           NA             NA
2011-01-21     8.8710 -0.2270  0    1 0.0000   0.2270     0.052107       0.061321
2011-01-24     8.8425 -0.0285  0    1 0.0000   0.0285     0.037036       0.063357
2013-12-31    19.9395  0.2710  1    0 0.2710   0.0000     0.122750       0.083428
```

Step 5: Calculate the Wilder Exponential Moving Average to Calculate Final Up and Down 14-DayAverages

```
> up.val<-as.numeric(AMZN.RSI$up.val)
> down.val<-as.numeric(AMZN.RSI$down.val)
>
> AMZN.RSI$up.avg<-AMZN.RSI$up.first.avg
> for (i in 15:nrow(AMZN.RSI)){AMZN.RSI$up.avg[i] <-((AMZN.RSI$up.avg[i-1]*13+up.val[i])/14)}
> AMZN.RSI$down.avg<-AMZN.RSI$down.first.avg
> for (i in 15:nrow(AMZN.RSI)){AMZN.RSI$down.avg[i]<-((AMZN.RSI$down.avg[i-1]*13+down.val[i])/14)}
> AMZN.RSI[c(1:20,nrow(AMZN.RSI)),]
           AMZN.Close   delta up down up.val down.val up.first.avg down.first.avg   up.avg down.avg
2011-01-03     9.2110  0.2110  1    0 0.2110   0.0000           NA             NA       NA       NA
2011-01-04     9.2505  0.0395  1    0 0.0395   0.0000           NA             NA       NA       NA
2011-01-05     9.3710  0.1205  1    0 0.1205   0.0000           NA             NA       NA       NA
2011-01-06     9.2930 -0.0780  0    1 0.0000   0.0780           NA             NA       NA       NA
2011-01-07     9.2745 -0.0185  0    1 0.0000   0.0185           NA             NA       NA       NA
2011-01-10     9.2340 -0.0405  0    1 0.0000   0.0405           NA             NA       NA       NA
2011-01-11     9.2170 -0.0170  0    1 0.0000   0.0170           NA             NA       NA       NA
2011-01-12     9.2040 -0.0130  0    1 0.0000   0.0130           NA             NA       NA       NA
2011-01-13     9.2765  0.0725  1    0 0.0725   0.0000           NA             NA       NA       NA
2011-01-14     9.4375  0.1610  1    0 0.1610   0.0000           NA             NA       NA       NA
2011-01-18     9.5625  0.1250  1    0 0.1250   0.0000           NA             NA       NA       NA
2011-01-19     9.3435 -0.2190  0    1 0.0000   0.2190           NA             NA       NA       NA
2011-01-20     9.0980 -0.2455  0    1 0.0000   0.2455           NA             NA       NA       NA
2011-01-21     8.8710 -0.2270  0    1 0.0000   0.2270     0.052107       0.061321 0.052107 0.061321
2011-01-24     8.8425 -0.0285  0    1 0.0000   0.0285     0.037036       0.063357 0.048385 0.058977
2011-01-25     8.8350 -0.0075  0    1 0.0000   0.0075     0.034214       0.063893 0.044929 0.055300
2011-01-26     8.7695 -0.0655  0    1 0.0000   0.0655     0.025607       0.068571 0.041720 0.056029
2011-01-27     9.2225  0.4530  1    0 0.4530   0.0000     0.057964       0.063000 0.071097 0.052027
2011-01-28     8.5570 -0.6655  0    1 0.0000   0.6655     0.057964       0.109214 0.066019 0.095846
2011-01-31     8.4820 -0.0750  0    1 0.0000   0.0750     0.057964       0.111679 0.061303 0.094357
2013-12-31    19.9395  0.2710  1    0 0.2710   0.0000     0.122750       0.083428 0.128878 0.082741
>
```

Step 6: Calculate the RSI

```
> AMZN.RSI$RS<-AMZN.RSI$up.avg/AMZN.RSI$down.avg
> AMZN.RSI$RSI<-100-(100/(1+AMZN.RSI$RS))
> AMZN.RSI[c(14:20,nrow(AMZN.RSI)),]
           AMZN.Close   delta up down up.val down.val up.first.avg down.first.avg   up.avg down.avg
2011-01-21     8.8710 -0.2270  0    1  0.000   0.2270     0.052107       0.061321 0.052107 0.061321
2011-01-24     8.8425 -0.0285  0    1  0.000   0.0285     0.037036       0.063357 0.048385 0.058977
2011-01-25     8.8350 -0.0075  0    1  0.000   0.0075     0.034214       0.063893 0.044929 0.055300
2011-01-26     8.7695 -0.0655  0    1  0.000   0.0655     0.025607       0.068571 0.041720 0.056029
2011-01-27     9.2225  0.4530  1    0  0.453   0.0000     0.057964       0.063000 0.071097 0.052027
2011-01-28     8.5570 -0.6655  0    1  0.000   0.6655     0.057964       0.109214 0.066019 0.095846
2011-01-31     8.4820 -0.0750  0    1  0.000   0.0750     0.057964       0.111679 0.061303 0.094357
2013-12-31    19.9395  0.2710  1    0  0.271   0.0000     0.122750       0.083428 0.128878 0.082741
                  RS    RSI
2011-01-21 0.84974 45.938
2011-01-24 0.82041 45.067
2011-01-25 0.81246 44.826
2011-01-26 0.74462 42.681
2011-01-27 1.36655 57.744
2011-01-28 0.68880 40.786
2011-01-31 0.64969 39.383
2013-12-31 1.55760 60.901
>
```

Step 7: Subset to Show Only 2012 and 2013 Data

```
> AMZN.RSI2012<-subset(AMZN.RSI[,ncol(AMZN.RSI)],index(AMZN.RSI)>="2012-01-01")
> AMZN.RSI2012[c(1:3,nrow(AMZN.RSI2012)),]
               RSI
2012-01-03 42.618
2012-01-04 41.178
2012-01-05 41.319
2013-12-31 60.901
>
```

Step 8: Plot the RSI

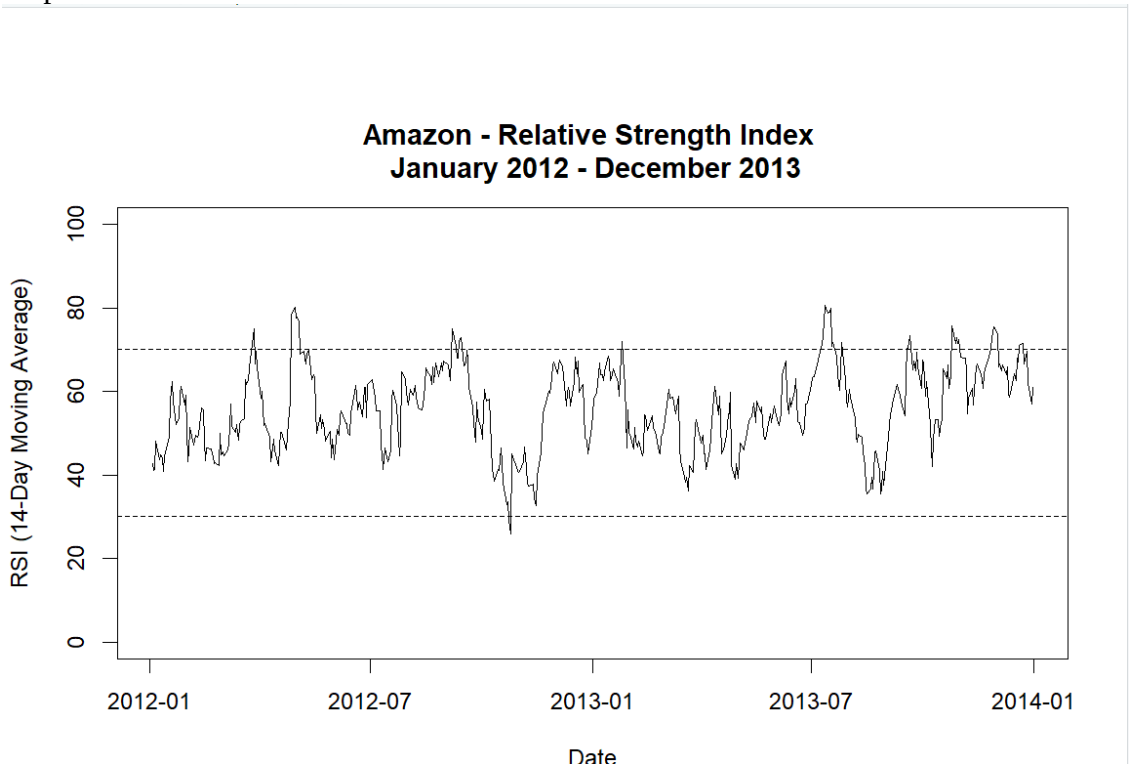


Figure shows that around April, May, and September 2012 and July 2013, a buy indication for Amazon.com may have been made as the RSI was above 70 and crossed below 70 during those periods. In contrast, around November 2012, the RSI was below 30 and crossed above 30 during that period, which could be taken as a sell signal.

1.7 Further Reading

The technical analysis charts above can be recreated using chartSeries. For example, the 50-day and 200-day moving average example above can be replicated by:


data.AMZN
[2012-01-03/2013-12-31]
4
Last 19.939501
20
18
16
14
12
10
1月 03 2012
6月 01 2012
11月 01 2012
4月 01 2013
8月 01 2013
12月 31 2013

Chapter 2 Individual Security Returns

It is reasonable to assume that percentage returns are more vital than absolute incomes because absolute incomes don't show the relevant arising improvement of returns but only shows an absolute value.

For example, 500$ means different compared to 5,000$ or 50,000$, but absolute return only depicts the value of 500$.

Total return is a measure of the return we receive over the entire time we hold the security, this type of return is also known as the holding period return. When necessary to make the distinction, we will call the return without the cash flow yield price return.

We begin this chapter by showing how to calculate price returns and total returns. Then, for statistical and programming considerations, we will show how to calculate logarithmic total returns.

We then show how to cumulate daily returns into multi-day returns using both the arithmetic (i.e., non-logarithmic) returns and logarithmic returns.

We then show that for dividend paying stocks, the total return over a long period of time can be much higher than the price return over the same period.

2.1 Price Returns

The daily price return is the percentage change in the price of a security today relative to its price yesterday. That is,

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

where PRett is the price return on day t, Pt is the price of the security on day t, and Pt−1 is the price of the security the trading day before.

Step 1: Import IBM Data from Yahoo Finance

```
> library(xts)
> data.IBM<-read.csv("IBM Yahooo.csv",header=TRUE)
> data.IBM<-data.IBM[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.IBM$Date,format="%Y-%m-%d")
> data.IBM<-cbind(date, data.IBM[,-1])
> data.IBM<-data.IBM[order(data.IBM$date),]
> data.IBM<-xts(data.IBM[,2:7],order.by=data.IBM[,1])
> names(data.IBM)<-paste(c("IBM.Open","IBM.High","IBM.Low","IBM.Close","IBM.Volume","IBM.Adjusted"))
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
```

Step 2: Subset the Data to Only Include the Closing Price

```
> IBM.prc.ret<-data.IBM[,4]
> IBM.prc.ret[c(1:3,nrow(IBM.prc.ret)),]
           IBM.Close
2010-12-31  140.3059
2011-01-03  140.9943
2011-01-04  141.1472
2013-12-31  179.3212
```

Step 3: Calculate IBM's Price Return

```
> IBM.prc.ret$IBM.prc.ret<-Delt(IBM.prc.ret$IBM.Close)
> IBM.prc.ret[c(1:3,nrow(IBM.prc.ret)),]
           IBM.Close IBM.prc.ret
2010-12-31  140.3059          NA
2011-01-03  140.9943 0.004905994
2011-01-04  141.1472 0.001084931
2013-12-31  179.3212 0.006222884
```

Step 4: Clean up Data Object

```
> options(digits=3)
> IBM.prc.ret<-IBM.prc.ret[-1,c(2)]
> IBM.prc.ret[c(1:3,nrow(IBM.prc.ret)),]
           IBM.prc.ret
2011-01-03     0.00491
2011-01-04     0.00108
2011-01-05    -0.00400
2013-12-31     0.00622
```

2.2 Total Returns

The total return a shareholder can receive from a stock that pays dividends includes both the change in the price of the shares she owns as well as any income generated from the dividends and the reinvestment of those dividends on the ex-date. This is known as the holding period return or total return.

The total return from holding a security is calculated as follows:

$$R_t = \frac{P_t + CF_t + P_{t-1}}{P_{t-1}} = \underbrace{\left[\frac{P_t}{P_{t-1}} - 1\right]}_{\text{Capital Appreciation}} + \underbrace{\frac{CF_t}{P_{t-1}}}_{\text{CF Yield}}$$

Where CFt is the cash flow (e.g., dividend) payment on day t.

Continuing from our IBM example from the previous section, we now calculate IBM's total returns from January 2011 to December 2013. As of the writing of this book, the last time IBM declared dividends was on October 29, 2013, which was payable on December 10, 2013. However, the important dividend date for calculating returns is the ex-dividend date or the ex-date, which was on November 6, 2013. The ex-date means that IBM shares will be trading without the dividend beginning on November 6, 2013, such that those who purchase IBM shares on November 6, 2013 and after would not be entitled to the dividend declared on October 29, 2013. The output below shows that the closing price and adjusted closing price for IBM on or after November 6, 2013 are the same, but those two variables have different values prior to November 6, 2013.

```
> data.IBM[715:720,]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2013-11-01 171.9025 172.4092 171.0134  171.3480    3812147     109.9084
2013-11-04 171.9885 172.8490 171.4532  172.3423    3643532     110.5462
2013-11-05 171.6444 171.8929 169.8948  170.0287    6377253     109.0622
2013-11-06 170.0860 171.8451 169.9618  171.3098    4770492     110.4740
2013-11-07 171.7017 173.4130 171.7017  172.0841    5459597     110.9734
2013-11-08 170.9656 172.1606 169.5507  172.0746    6563650     110.9672
>
```

Step 1: Import Adjusted Closing Price Data

```
> IBM.ret<-data.IBM[,6]
> IBM.ret[c(1:3,nrow(IBM.ret)),]
            IBM.Adjusted
2010-12-31     85.83682
2011-01-03     86.25792
2011-01-04     86.35150
2013-12-31    115.64046
```

Step 2: Calculate Total Return

```
> IBM.ret$IBM.tot.ret=Delt(IBM.ret$IBM.Adjusted)
> IBM.ret[c(1:3,nrow(IBM.ret)),]
            IBM.Adjusted IBM.tot.ret
2010-12-31     85.83682          NA
2011-01-03     86.25792 0.004905867
2011-01-04     86.35150 0.001084909
2013-12-31    115.64046 0.006223193
```

Step 3: Clean up the Data

```
> options(digits=3)
> IBM.ret<-IBM.ret[,2]
> IBM.ret[c(1:3,nrow(IBM.ret)),]
            IBM.tot.ret
2010-12-31          NA
2011-01-03     0.00491
2011-01-04     0.00108
2013-12-31     0.00622
> options(digits=7)
```

2.3 Logarithmic Total Returns

The logarithmic return, r_t , is calculated as

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln(1 + R_t) = \ln P_t - \ln P_{t-1}$$

Where ln is the natural logarithm operator and the rest of the variables are defined the same.

Therefore, we can take the difference of the log prices to calculate log returns.

Step 1: Import Adjusted Closing Price Data

```
> IBM.ret<-data.IBM[,6]
> IBM.ret[c(1:3,nrow(IBM.ret)),]
           IBM.Adjusted
2010-12-31     85.83682
2011-01-03     86.25792
2011-01-04     86.35150
2013-12-31    115.64046
```

Step 2: Calculate Log Returns

```
> IBM.ret$IBM.log.ret<-diff(log(IBM.ret$IBM.Adjusted))
> IBM.ret[c(1:3,nrow(IBM.ret)),]
           IBM.Adjusted IBM.log.ret
2010-12-31     85.83682          NA
2011-01-03     86.25792 0.004893872
2011-01-04     86.35150 0.001084321
2013-12-31    115.64046 0.006203909
```

Step 3: Clean up the Data

```
> options(digits=3)
> IBM.ret<-IBM.ret[,2]
> IBM.ret[c(1:3,nrow(IBM.ret)),]
           IBM.log.ret
2010-12-31          NA
2011-01-03     0.00489
2011-01-04     0.00108
2013-12-31     0.00620
```

Compare Log Returns with Arithmetic Returns

```
> options(digits=3,scipen=100)
> tot.rets<-cbind(IBM.total.ret,IBM.log.ret)
> tot.rets[c(1:3,nrow(tot.rets)),]
           IBM.tot.ret IBM.log.ret
2010-12-31          NA          NA
2011-01-03     0.00491     0.00489
2011-01-04     0.00108     0.00108
2013-12-31     0.00622     0.00620
> max(abs(tot.rets$IBM.tot.ret-tot.rets$IBM.log.ret),na.rm=TRUE)
[1] 0.00363
> min(abs(tot.rets$IBM.tot.ret-tot.rets$IBM.log.ret),na.rm=TRUE)
[1] 0.00000000119
> options(digits=7,scipen=0)
>
```

2.4 Cumulating Multi-Day Returns

To fully capture the effects of being able to reinvest dividends, we should calculate daily returns and string those returns together for longer periods. Otherwise, if we simply apply Eq. (2.2) using prices at the beginning and end of the investment horizon and add the dividend, we are assuming that the dividends are received at the end of the period and no additional returns on those dividends are earned.

In other words, when we receive the dividend, we would reinvest that amount back into the stock. The returns of that stock going forward determines whether the reinvested dividend earned a positive or negative return.

Suppose we are interested in knowing how much would an investment in Amazon have made through the end of 2013 if we purchased AMZN shares at the closing price on December 31, 2010.

We show how to implement this calculation using arithmetic returns and logarithmic returns, and show that, when consistently implemented, both types of returns yield the same result.

However, some may find the programming for logarithmic returns slightly easier.

2.4.1 Cumulating Arithmetic Returns

To string together multiple days of arithmetic returns, we have to take the product of the daily gross returns.

We can generalize this calculation over a T -day investment horizon as:

$$R_{1 \text{ to } T} = (1 + R_1) \times (1 + R_2) \times \cdots \times (1 + R_T).$$

Step 1: Import Data and Calculate Arithmetic Returns

```
> IBM.acum<-IBM.total.ret
> IBM.acum[c(1:3,nrow(IBM.acum)),]
            IBM.tot.ret
2010-12-31          NA
2011-01-03 0.004905867
2011-01-04 0.001084909
2013-12-31 0.006223193
>
```

Step 2: Set First Day Total Return Value to Zero

```
> IBM.acum[1,1]<-0
> IBM.acum[c(1:3,nrow(IBM.acum)),]
            IBM.tot.ret
2010-12-31 0.000000000
2011-01-03 0.004905867
2011-01-04 0.001084909
2013-12-31 0.006223193
>
```

Step 3: Calculate Gross Daily Returns

```
> IBM.acum$GrossRet<-1+IBM.acum$IBM.tot.ret
> IBM.acum[c(1:3,nrow(IBM.acum)),]
            IBM.tot.ret GrossRet
2010-12-31 0.000000000 1.000000
2011-01-03 0.004905867 1.004906
2011-01-04 0.001084909 1.001085
2013-12-31 0.006223193 1.006223
>
```

Step 4: Calculate Cumulative Gross Returns

```
> IBM.acum$GrossCum<-cumprod(IBM.acum$GrossRet)
> IBM.acum[c(1:3,nrow(IBM.acum)),]
            IBM.tot.ret GrossRet GrossCum
2010-12-31 0.000000000 1.000000 1.000000
2011-01-03 0.004905867 1.004906 1.004906
2011-01-04 0.001084909 1.001085 1.005996
2013-12-31 0.006223193 1.006223 1.347213
>
```

Step 5: Convert Cumulative Gross Returns to Cumulative Net Returns

```
> IBM.acum$NetCum<-IBM.acum$GrossCum-1
> IBM.acum[c(1:3,nrow(IBM.acum)),]
            IBM.tot.ret GrossRet GrossCum      NetCum
2010-12-31 0.000000000 1.000000 1.000000 0.000000000
2011-01-03 0.004905867 1.004906 1.004906 0.004905867
2011-01-04 0.001084909 1.001085 1.005996 0.005996099
2013-12-31 0.006223193 1.006223 1.347213 0.347212813
>
```

2.4.2 Cumulating Logarithmic Returns

An alternative way to calculate multi-period returns is to take the sum of the daily logarithmic returns. That is,

$$\begin{aligned} r_{1 \text{ to } T} &= \ln((1+R_1)\times(1+R_2)\times\cdots\times(1+R_T)) \\ &= r_1 + r_2 + \cdots + r_T \\ &= \sum_{t=1}^{T} r_t \end{aligned}$$

Step 1: Import Data and Calculate Logarithmic Returns

```
> IBM.logcum<-IBM.log.ret
> IBM.logcum[c(1:3,nrow(IBM.logcum)),]
            IBM.log.ret
2010-12-31          NA
2011-01-03 0.004893872
2011-01-04 0.001084321
2013-12-31 0.006203909
>
```

Step 2: Set the First Log Return to Zero

```
> IBM.logcum[1,1]<-0
> IBM.logcum[c(1:3,nrow(IBM.logcum)),]
           IBM.log.ret
2010-12-31 0.000000000
2011-01-03 0.004893872
2011-01-04 0.001084321
2013-12-31 0.006203909
>
```

Step 3: Take the Sum of all Logarithmic Returns During the Investment Period

```
> logcumret=sum(IBM.logcum$IBM.log.ret)
> logcumret
[1] 0.2980379
>
```

Step 4: Convert Log Return Back to Arithmetic Return

```
> cumret=exp(logcumret)-1
> cumret
[1] 0.3472128
>
```

We can see that it takes fewer and simpler steps to calculate multi-period returns using logarithmic returns than it is using arithmetic returns.

2.4.3 Comparing Price Return and Total Return

Step 1: Import Data and Calculate Price and Total Returns

```
> IBM.Ret<-cbind(IBM.prc.ret,IBM.acum[,1])
> names(IBM.Ret)<-c("prc.ret","tot.ret")
> IBM.Ret[c(1:3,nrow(IBM.Ret)),]
               prc.ret     tot.ret
2010-12-31          NA 0.000000000
2011-01-03 0.004905994 0.004905867
2011-01-04 0.001084931 0.001084909
2013-12-31 0.006222884 0.006223193
```

Step 2: Set First Returns to Zero

```
> IBM.Ret$prc.ret[1]<-0
> IBM.Ret$tot.ret[1]<-0
> IBM.Ret[c(1:3,nrow(IBM.Ret)),]
               prc.ret     tot.ret
2010-12-31 0.000000000 0.000000000
2011-01-03 0.004905994 0.004905867
2011-01-04 0.001084931 0.001084909
2013-12-31 0.006222884 0.006223193
>
```

Step 3: Calculate Gross Returns

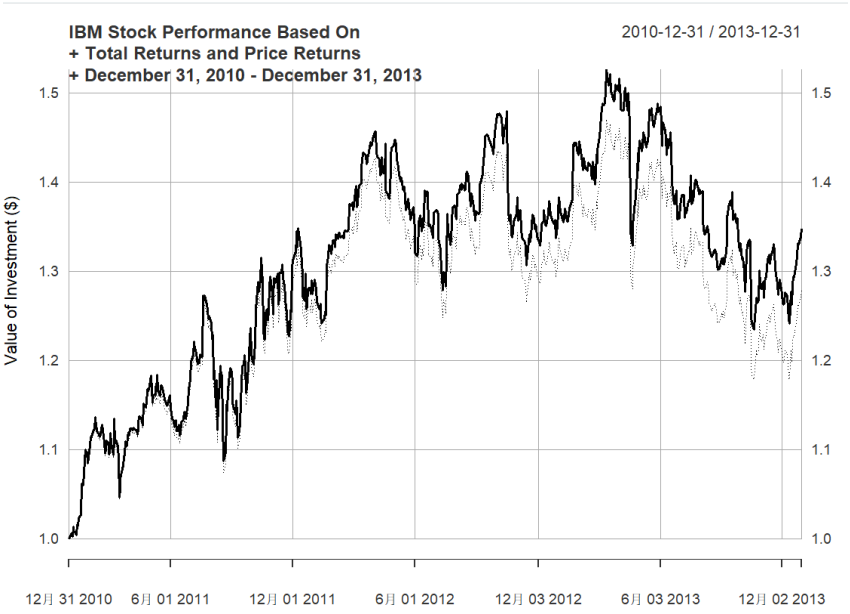
```
> IBM.Ret$gross.prc<-1+IBM.Ret$prc.ret
> IBM.Ret$gross.tot<-1+IBM.Ret$tot.ret
> IBM.Ret[c(1:3,nrow(IBM.Ret)),]
              prc.ret     tot.ret gross.prc gross.tot
2010-12-31 0.000000000 0.000000000  1.000000  1.000000
2011-01-03 0.004905994 0.004905867  1.004906  1.004906
2011-01-04 0.001084931 0.001084909  1.001085  1.001085
2013-12-31 0.006222884 0.006223193  1.006223  1.006223
>
```

Step 4: Cumulate the Gross Returns

```
> IBM.Ret$cum.prc<-cumprod(IBM.Ret$gross.prc)
> IBM.Ret$cum.tot<-cumprod(IBM.Ret$gross.tot)
> IBM.Ret[c(1:3,nrow(IBM.Ret)),]
              prc.ret     tot.ret gross.prc gross.tot  cum.prc  cum.tot
2010-12-31 0.000000000 0.000000000  1.000000  1.000000 1.000000 1.000000
2011-01-03 0.004905994 0.004905867  1.004906  1.004906 1.004906 1.004906
2011-01-04 0.001084931 0.001084909  1.001085  1.001085 1.005996 1.005996
2013-12-31 0.006222884 0.006223193  1.006223  1.006223 1.278073 1.347213
>
```

Step 5: Plot the Two Return Series

We then plot the cum.prc and cum.tot variables. We first plot the cum.tot variable, and then plot the cum.prc variable. Using the abline command, we add a horizontal line at $ 1 to make it easy for us to interpret whether the investment is making money or losing money.



2.5 Weekly Returns

In the previous sections, we used daily returns. However, there are applications in which we may have to use returns of lower frequency, such as weekly returns. In this section, we will revert back to using AMZN stock data.

```
> data.AMZN<-read.csv("AMZN Yahooo.csv",header=TRUE)
> data.AMZN<-data.AMZN[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.AMZN$Date,format="%Y-%m-%d")
> data.AMZN<-cbind(date, data.AMZN[,-1])
> data.AMZN<-data.AMZN[order(data.AMZN$date),]
> data.AMZN<-xts(data.AMZN[,2:7],order.by=data.AMZN[,1])
> names(data.AMZN)<-paste(c("AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"))
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
>
```

Show the class of "data.AMZN":

```
> class(data.AMZN)
[1] "xts" "zoo"
>
```

Step 1: Import Data into R

```
> wk<-data.AMZN
> wk[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
>
```

Step 2: Convert to Daily Data to Weekly Data

```
> AMZN.weekly<-to.weekly(wk)
> AMZN.weekly[c(1:3,nrow(AMZN.weekly)),]
           wk.Open wk.High  wk.Low wk.Close wk.Volume wk.Adjusted
2010-12-31  9.0980  9.1150  8.9755   9.0000  69038000      9.0000
2011-01-07  9.0685  9.4225  9.0605   9.2745 443668000      9.2745
2011-01-14  9.2520  9.4470  9.1255   9.4375 317980000      9.4375
2013-12-31 19.9705 19.9960 19.6225  19.9395  89672000     19.9395
>
```

Step 3: Clean up Weekly Data to Keep Only the Adjusted Closing Prices at the End of Each Week

```
> AMZN.weekly<-AMZN.weekly[,6]
> AMZN.weekly[c(1:3,nrow(AMZN.weekly)),]
           wk.Adjusted
2010-12-31      9.0000
2011-01-07      9.2745
2011-01-14      9.4375
2013-12-31     19.9395
>
```

Step 4: Calculate Weekly Returns

```
> AMZN.weekly$Ret<-Delt(AMZN.weekly$wk.Adjusted)
> AMZN.weekly[c(1:3,nrow(AMZN.weekly)),]
           wk.Adjusted         Ret
2010-12-31      9.0000          NA
2011-01-07      9.2745 0.030500000
2011-01-14      9.4375 0.017575071
2013-12-31     19.9395 0.001783662
>
```

Step 5: Cleanup Weekly Returns Data by Deleting the First Observation

```
> AMZN.weekly<-AMZN.weekly[-1,2]
> AMZN.weekly[c(1:3,nrow(AMZN.weekly)),]
                   Ret
2011-01-07  0.030500000
2011-01-14  0.017575071
2011-01-21 -0.060026490
2013-12-31  0.001783662
>
```

2.6 Monthly Returns

Step 1: Import Data into R

```
> mo<-data.AMZN
> mo[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
>
```

Step 2: Convert Daily Data to Monthly Data.

```
> AMZN.monthly<-to.monthly(mo)
> AMZN.monthly[c(1:3,nrow(AMZN.monthly)),]
          mo.Open mo.High  mo.Low mo.Close  mo.Volume mo.Adjusted
12月 2010  9.0980  9.1150  8.9755   9.0000   69038000      9.0000
1月 2011   9.0685  9.5800  8.3450   8.4820 2272226000      8.4820
2月 2011   8.5260  9.5700  8.4755   8.6645 1915528000      8.6645
12月 2013 19.9500 20.2815 18.9750  19.9395 1113734000     19.9395
>
```

Step 3: Clean up Data to Include Only Adjusted Closing Prices for the End of Each Month

```
> AMZN.monthly<-AMZN.monthly[,6]
> AMZN.monthly[c(1:3,nrow(AMZN.monthly)),]
          mo.Adjusted
12月 2010      9.0000
1月 2011       8.4820
2月 2011       8.6645
12月 2013     19.9395
>
```

Step 4: Calculate Monthly Returns

```
> AMZN.monthly$Ret<-Delt(AMZN.monthly$mo.Adjusted)
> AMZN.monthly[c(1:3,nrow(AMZN.monthly)),]
          mo.Adjusted         Ret
12月 2010      9.0000          NA
1月 2011       8.4820 -0.05755556
2月 2011       8.6645  0.02151615
12月 2013     19.9395  0.01313455
>
```

Step 5: Cleanup Monthly Data Object

```
> AMZN.monthly<-AMZN.monthly[-1,2]
> AMZN.monthly[c(1:3,nrow(AMZN.monthly)),]
                   Ret
1月 2011  -0.05755556
2月 2011   0.02151615
3月 2011   0.03947141
12月 2013  0.01313455
>
```

2.7 Comparing Performance of Multiple Securities: Total Returns

AMZN and YHOO have not paid dividends, but IBM has. The S&P 500 Index data we use does not include dividends. Therefore, the index value for AMZN, YHOO, and GSPC would be the same regardless of whether we use price returns or total returns. However, for IBM, the index value using total returns will be higher than the index value using price returns.

Step 1: Importing Price Data

```
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
>
```

Now import another two securities.

```
> # AAPL Data
> data.AAPL<-read.csv("AAPL Yahooo.csv",header=TRUE)
> data.AAPL<-data.AAPL[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.AAPL$Date,format="%Y-%m-%d")
> data.AAPL<-cbind(date, data.AAPL[,-1])
> data.AAPL<-data.AAPL[order(data.AAPL$date),]
> data.AAPL<-xts(data.AAPL[,2:7],order.by=data.AAPL[,1])
> names(data.AAPL)[1:6]<-paste(c("AAPL.Open","AAPL.High","AAPL.Low","AAPL.Close","AAPL.Volume","AAPL.Adjuste
d"))
> data.AAPL[c(1:3,nrow(data.AAPL)),]
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2010-12-31  11.53393  11.55286 11.47536   11.52000   193508000      9.739614
2011-01-03  11.63000  11.79500 11.60143   11.77036   445138400      9.951281
2011-01-04  11.87286  11.87500 11.71964   11.83179   309080800     10.003214
2013-12-31  19.79179  20.04571 19.78571   20.03643   223084400     17.519623
>


> # NVDA Data
> data.NVDA<-read.csv("NVDA Yahooo.csv",header=TRUE)
> data.NVDA<-data.NVDA[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.NVDA$Date,format="%Y-%m-%d")
> data.NVDA<-cbind(date, data.NVDA[,-1])
> data.NVDA<-data.NVDA[order(data.NVDA$date),]
> data.NVDA<-xts(data.NVDA[,2:7],order.by=data.NVDA[,1])
> names(data.NVDA)[1:6]<-paste(c("NVDA.Open","NVDA.High","NVDA.Low","NVDA.Close","NVDA.Volume","NVDA.Adjuste
d"))
> data.NVDA[c(1:3,nrow(data.NVDA)),]
           NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume NVDA.Adjusted
2010-12-31    3.7500    3.8550    3.745     3.8500    39125200      3.531613
2011-01-03    3.8800    3.9925    3.875     3.9550    81744800      3.627930
2011-01-04    3.9625    3.9800    3.855     3.9425    65138400      3.616464
2013-12-31    4.0000    4.0250    3.975     4.0050    23577600      3.778282
>
```

Step 2: Combine Data

```
> multi<-data.AMZN[,6]
> multi<-merge(multi,data.AAPL[,6])
> multi<-merge(multi,data.NVDA[,6])
> multi<-merge(multi,data.IBM[,6])
> multi[c(1:3,nrow(multi)),]
           AMZN.Adjusted AAPL.Adjusted NVDA.Adjusted IBM.Adjusted
2010-12-31        9.0000      9.739614      3.531613     85.83682
2011-01-03        9.2110      9.951281      3.627930     86.25792
2011-01-04        9.2505     10.003214      3.616464     86.35150
2013-12-31       19.9395     17.519623      3.778282    115.64046
>
```

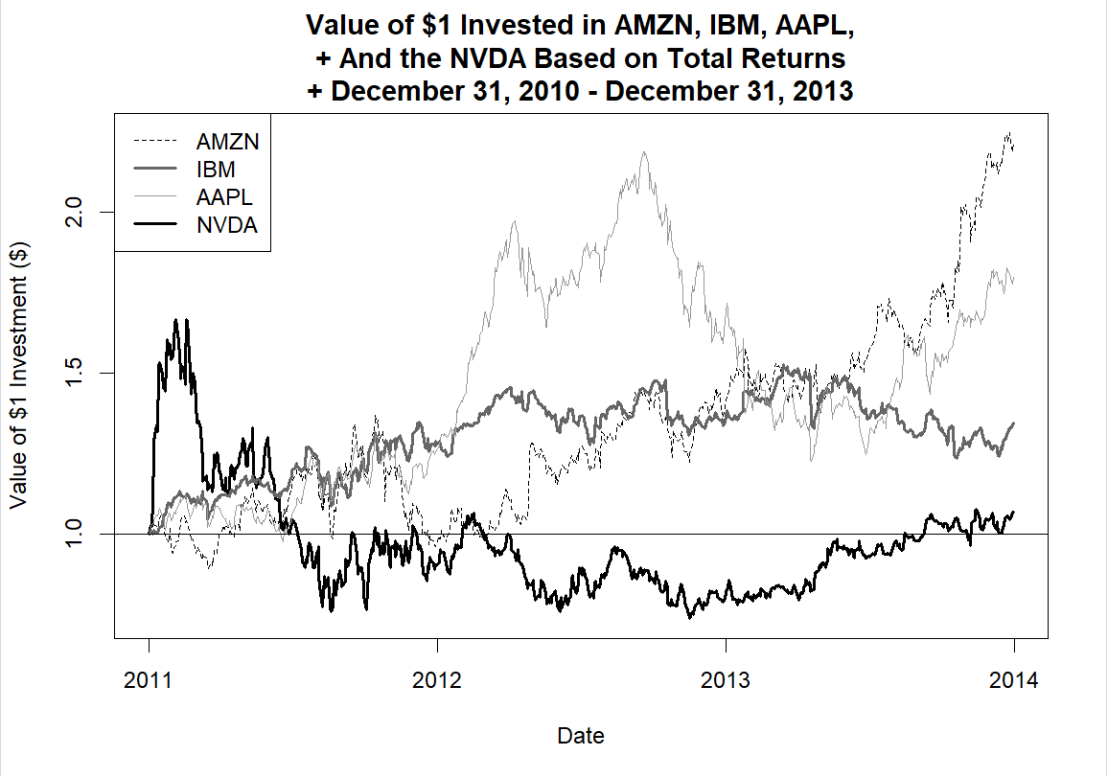
Step 3: Converting Data into a data.frame Object

```
> multi.df<-cbind(data.frame(index(multi)),data.frame(multi))
> names(multi.df)<-paste(c("date","AMZN","AAPL","NVDA","IBM"))
> multi.df[c(1:3,nrow(multi.df)),]
                date    AMZN      AAPL     NVDA      IBM
2010-12-31 2010-12-31  9.0000  9.739614 3.531613  85.83682
2011-01-03 2011-01-03  9.2110  9.951281 3.627930  86.25792
2011-01-04 2011-01-04  9.2505 10.003214 3.616464  86.35150
2013-12-31 2013-12-31 19.9395 17.519623 3.778282 115.64046
> 
```

Step 4: Constructing Normalized Values for Each Security

```
> multi.df$AMZN.idx<-multi.df$AMZN/multi.df$AMZN[1]
> multi.df$AAPL.idx<-multi.df$AAPL/multi.df$AAPL[1]
> multi.df$NVDA.idx<-multi.df$NVDA/multi.df$NVDA[1]
> multi.df$IBM.idx<-multi.df$IBM/multi.df$IBM[1]
> multi.df[c(1:3,nrow(multi.df)),6:9]
            AMZN.idx AAPL.idx NVDA.idx  IBM.idx
2010-12-31 1.000000 1.000000 1.000000 1.000000
2011-01-03 1.023444 1.021733 1.027273 1.004906
2011-01-04 1.027833 1.027065 1.024026 1.005996
2013-12-31 2.215500 1.798801 1.069846 1.347213
> 
```

Step 5: Plotting the Index Values of Each Security



We can see from this graph that AMZN outperformed other three securities with a total return of approximately 120%, whereas the total return of NVDA nearly approaches 0. Generally, Internet Company like Apple and Amazon perform better from 2011-2013.

Chapter 3 Portfolio Returns

Portfolio returns represent a combination of different securities as the investment of a specific stock market investor calculated as the weighted average, where the weight of each security is referred to the amount of money allocated to each security regarding to the overall investment.

Two approaches are presented here for calculation of Portfolio Returns. The first one is called long way where every detailed step is demonstrated which in later would be used when assessing mean-variance optimization across multiple securities.

The second one is called Benchmark Portfolio Returns, which are returns of a hypothetical portfolio that we can exploit to compare the performance of our portfolio, where we either use an equal-weighted portfolio or a value-weighted portfolio with quarterly rebalancing.

As the literal meaning, an equal-weighted portfolio redistributes the overall value to all the basketed securities within the equal weight on each rebalancing date.

Whereas the total value of all the basketed securities is redistributed based on the capitalization of each security on the market, namely larger firms get bigger share of the pie.

But whatever the weighting method we are using, the weight of each security in the basket is allowed to freely change based on their performance.

3.1 Constructing Portfolio Returns (Long Way)

Namely, the portfolio return is the weighted-average of individual returns for each selected security in the investor's basket, where the weights are calculated as the percentage occupations of different securities compared to the total volume in the basket.

That is, for a two-asset portfolio, we have

$$r_p = \frac{I_1}{I_1 + I_2} * r_1 + \frac{I_2}{I_1 + I_2} * r_2 = w_1 * r_1 + w_2 * r_2,$$

where r_p is the portfolio return, I_1 is the dollar amount invested in security 1, I_2 is the dollar amount invested in security 2, r_1 is the return of security 1, r_2 is the return of security 2.

we can then simplify the percentage invested in security 1 as weight 1 and denote it by w_1 and we can simplify the percentage invested in security 2 as weight 2 and denote it by w_2.

We can then extend this 2-asset portfolio return calculation into multiple assets.

That is,

$$r_p = w_1 * r_1 + w_2 * r_2 + w_3 * r_3 + w_4 * r_4 + \cdots = \sum_{i=1}^{N} w_i * r_i$$

Where the summation is done over the weighted-returns of each of the N assets in the portfolio.

In the case that we have cashes in the portfolio, we could still assign a weight as well as an individual return for the cashes, where the return could be short-term interest rate of money fund.

Note that the summarized percentage should still be 100% even cashes are allowed in the portfolio.

Now our assumption is that $50,000 in AMZN, $10,000 in AAPL, $30,000 in NVDA and $10,000 in IBM.

How much would our portfolio return be as of December 31, 2013 if we made the investment in those securities on December 31, 2010?

Step 1: Find First and Last Adjusted Closing Price for Each Security Over the Investment Period

```
> ###True Chapter 3
> ###Periodic Return between first day and last day of the time duration
> period.ret<-multi[c(1,nrow(multi)),]
> period.ret
           AMZN.Adjusted AAPL.Adjusted NVDA.Adjusted IBM.Adjusted
2010-12-31        9.0000      9.739614      3.531613     85.83682
2013-12-31       19.9395     17.519623      3.778282    115.64046
>
```

Step 2: Calculate Returns for Each Security Over the Investment Period

To apply the Delt command to all four securities, we can use the lapply command. The lapply command applies a function to each of the variables in the data object. In this case, we are applying Delt to each variable. The output below shows that we get the net total return for each security in a list object.

```
> ###lapply for applying Delt for all four variables
> rets<-lapply(period.ret,Delt)
> rets
$AMZN.Adjusted
           Delt.1.arithmetic
2010-12-31                NA
2013-12-31            1.2155

$AAPL.Adjusted
           Delt.1.arithmetic
2010-12-31                NA
2013-12-31         0.7988005

$NVDA.Adjusted
           Delt.1.arithmetic
2010-12-31                NA
2013-12-31        0.06984599

$IBM.Adjusted
           Delt.1.arithmetic
2010-12-31                NA
2013-12-31         0.3472128

>
```

Step 3: Convert to a data.frame and Cleanup Data

Since the returns data is in a list object, we have to convert it to a data.frame to continue with our calculations. We convert rets into a data frame using the data.frame command.

```
> ###convert from list to data.frame
> rets<-data.frame(rets)
> rets
           Delt.1.arithmetic Delt.1.arithmetic.1 Delt.1.arithmetic.2 Delt.1.arithmetic.3
2010-12-31                NA                  NA                  NA                  NA
2013-12-31            1.2155           0.7988005          0.06984599           0.3472128
>

> ###Only keep the second row since first row "NA"
> rets<-rets[2,]*100
> ###Rename with efficiency
> names(rets)<-paste(c("AMZN","GSPC","YHOO","IBM"))
> ###output
> rets
              AMZN     GSPC     YHOO      IBM
2013-12-31 121.55 79.88005 6.984599 34.72128
>
```

Step 4: Calculate Weight of Each Security in the Portfolio

```
> ###Dollars in AMZN
> i.AMZN<-50000
> ###Dollars in AAPL
> i.AAPL<-10000
> ###Dollars in NVDA
> i.NVDA<-30000
> ###Dollars in IBM
> i.IBM<-10000
> ###AMZN weight
> w.AMZN<-i.AMZN/(i.AMZN+i.AAPL+i.NVDA+i.IBM)
> w.AMZN
[1] 0.5
> ###AAPL weight
> w.AAPL<-i.AAPL/(i.AMZN+i.AAPL+i.NVDA+i.IBM)
> w.AAPL
[1] 0.1
> ###NVDA weight
> w.NVDA<-i.NVDA/(i.AMZN+i.AAPL+i.NVDA+i.IBM)
> w.NVDA
[1] 0.3
> ###IBM weight
> w.IBM<-i.IBM/(i.AMZN+i.AAPL+i.NVDA+i.IBM)
> w.IBM
[1] 0.1
>
```

Step 5: Calculate Portfolio Return

```
> ###TOTAL PERCENTAGE RATE RETURN BY WEIGHTED SUMMATION
> port.ret.4asset<-w.AMZN*rets$AMZN+w.AAPL*rets$AAPL+ w.NVDA*rets$NVDA+w.IBM*rets$IBM
> ###OUTPUT
> port.ret.4asset
[1] 74.33052
>
```

3.2 Constructing Portfolio Returns (Matrix Algebra)

A convenient way to calculate portfolio returns when there are many securities in the portfolio is to use matrix algebra.

To implement this, we have to calculate two vectors, where a vector can be thought of as a series of numbers organized as either a row or a column.

The first is a vector of weights, which in this case is 50 % for AMZN, 10 % for AAPL, 30 % for NVDA, and 10 % for IBM.

The second is a column vector of returns.

We can now calculate the portfolio return by multiplying the row vector of weights by the column vector of returns using the matrix multiplication operator %*%.

```
> ###Weight row vector with a row of 4 numbers
> wgt<-c(0.5,0.1,0.3,0.1)
> mat.wgt<-matrix(wgt,1)
> mat.wgt
     [,1] [,2] [,3] [,4]
[1,]  0.5  0.1  0.3  0.1
>
> ###Return, a four-row vector with one return each in a row
> ret<-c(rets$AMZN,rets$AAPL,rets$NVDA,rets$IBM)
> mat.ret<-matrix(ret,4)
> mat.ret
          [,1]
[1,] 121.550011
[2,]  79.880055
[3,]   6.984599
[4,]  34.721281
>
> ###Two vectors multiplied together
> port.ret<-mat.wgt %*% mat.ret
> port.ret
          [,1]
[1,] 74.33052
>
```

3.3 Constructing Benchmark Portfolio Returns

To determine whether our investment is performing well, we compare the returns of our investment to that of a benchmark.

Typically, some indices of comparable securities are used as the benchmark. Some benchmarks are readily-available (e.g., S&P 500 Index for large capitalization stocks). Alternatively, benchmarks can be constructed.

In this section, we demonstrate how to construct an equal-weighted (EW) index and value-weighted (VW) index.

Step 1: Importing the Price Data

```
> ###Make sure that AMZN's data still in R Memory
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000         9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000         9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000         9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000        19.9395
>
> ###Make sure that NVDA's data still in R Memory
> data.NVDA[c(1:3,nrow(data.NVDA)),]
           NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume NVDA.Adjusted
2010-12-31    3.7500    3.8550    3.745     3.8500    39125200       3.531613
2011-01-03    3.8800    3.9925    3.875     3.9550    81744800       3.627930
2011-01-04    3.9625    3.9800    3.855     3.9425    65138400       3.616464
2013-12-31    4.0000    4.0250    3.975     4.0050    23577600       3.778282
>
> ###Make sure that IBM's data still in R Memory
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
>
```

Step 2: Create Object with Only the Relevant Data

```
> ###Merge Close and Adjusted Price for each three security
> port<-data.AMZN[,c(4,6)]
> port<-merge(port,data.NVDA[,c(4,6)])
> port<-merge(port,data.IBM[,c(4,6)])
> port[c(1:3,nrow(port)),]
           AMZN.Close AMZN.Adjusted NVDA.Close NVDA.Adjusted IBM.Close IBM.Adjusted
2010-12-31     9.0000        9.0000     3.8500      3.531613  140.3059     85.83682
2011-01-03     9.2110        9.2110     3.9550      3.627930  140.9943     86.25792
2011-01-04     9.2505        9.2505     3.9425      3.616464  141.1472     86.35150
2013-12-31    19.9395       19.9395     4.0050      3.778282  179.3212    115.64046
>
```

Step 3: Calculate Returns of Each Security

```
> ###For each security,we use delta of adjusted close price to calculate return
> port$AMZN.ret<-Delt(port$AMZN.Adjusted)
> port$NVDA.ret<-Delt(port$NVDA.Adjusted)
> port$IBM.ret<-Delt(port$IBM.Adjusted)
> port[c(1:3,nrow(port)),]
           AMZN.Close AMZN.Adjusted NVDA.Close NVDA.Adjusted IBM.Close IBM.Adjusted    AMZN.ret
2010-12-31     9.0000        9.0000     3.8500      3.531613  140.3059     85.83682          NA
2011-01-03     9.2110        9.2110     3.9550      3.627930  140.9943     86.25792 0.023444444
2011-01-04     9.2505        9.2505     3.9425      3.616464  141.1472     86.35150 0.004288351
2013-12-31    19.9395       19.9395     4.0050      3.778282  179.3212    115.64046 0.013778376
               NVDA.ret     IBM.ret
2010-12-31           NA          NA
2011-01-03  0.02727281 0.004905867
2011-01-04 -0.00316048 0.001084909
2013-12-31  0.00313050 0.006223193
> 
```

Step 4: Convert to data.frame Object and Subset Data

```
> ###To only show data from 2012-12-31 to 2013-12-31 by using subset
> port<-subset(port,
+                + index(port) >= "2012-12-31" &
+                  + index(port) <= "2013-12-31")
> port[c(1:3,nrow(port)),]
           AMZN.Close AMZN.Adjusted NVDA.Close NVDA.Adjusted IBM.Close IBM.Adjusted    AMZN.ret
2012-12-31    12.5435       12.5435     3.0650      2.829670  183.1262     115.8462 0.023207439
2013-01-02    12.8655       12.8655     3.1800      2.935841  187.7151     118.7492 0.025670666
2013-01-03    12.9240       12.9240     3.1825      2.938150  186.6826     118.0960 0.004547044
2013-12-31    19.9395       19.9395     4.0050      3.778282  179.3212     115.6405 0.013778376
               NVDA.ret      IBM.ret
2012-12-31 0.0132228469  0.009060829
2013-01-02 0.0375206296  0.025058963
2013-01-03 0.0007864867 -0.005500654
2013-12-31 0.0031305000  0.006223193
> 
```

Here I must point out that I think the code and output is not efficient considering what it did regarding to the use of the data.frame object. If the port data is converted into the data.frame object, then "port$date" will also be a data.frame object, which function well when you use the line of the code:

port$date >= "2012-12-31"

But it is inefficient since such output could also be achieved using xts but not converted to data.frame object, only use the following code to substitute above,

index (port) >= "2012-12-31"

I have to say the following codes are too cumbersome.

And you could see the initial code and output generated by myself.

```
> ###The provement of not efficient codes and output in the book with codes pasted here
> port<-cbind(data.frame(index(port)),
+               + data.frame(port))
> names(port)[1]<-paste("date")
> port[c(1:3,nrow(port)),]
                date AMZN.Close AMZN.Adjusted NVDA.Close NVDA.Adjusted IBM.Close IBM.Adjusted
2012-12-31 2012-12-31    12.5435       12.5435     3.0650      2.829670  183.1262     115.8462
2013-01-02 2013-01-02    12.8655       12.8655     3.1800      2.935841  187.7151     118.7492
2013-01-03 2013-01-03    12.9240       12.9240     3.1825      2.938150  186.6826     118.0960
2013-12-31 2013-12-31    19.9395       19.9395     4.0050      3.778282  179.3212     115.6405
              AMZN.ret     NVDA.ret      IBM.ret
2012-12-31 0.023207439 0.0132228469  0.009060829
2013-01-02 0.025670666 0.0375206296  0.025058963
2013-01-03 0.004547044 0.0007864867 -0.005500654
2013-12-31 0.013778376 0.0031305000  0.006223193
> port<-subset(port,
+                + port$date >= "2012-12-31" &
+                  + port$date <= "2013-12-31")
> port[c(1:3,nrow(port)),]
                date AMZN.Close AMZN.Adjusted NVDA.Close NVDA.Adjusted IBM.Close IBM.Adjusted
2012-12-31 2012-12-31    12.5435       12.5435     3.0650      2.829670  183.1262     115.8462
2013-01-02 2013-01-02    12.8655       12.8655     3.1800      2.935841  187.7151     118.7492
2013-01-03 2013-01-03    12.9240       12.9240     3.1825      2.938150  186.6826     118.0960
2013-12-31 2013-12-31    19.9395       19.9395     4.0050      3.778282  179.3212     115.6405
              AMZN.ret     NVDA.ret      IBM.ret
2012-12-31 0.023207439 0.0132228469  0.009060829
2013-01-02 0.025670666 0.0375206296  0.025058963
2013-01-03 0.004547044 0.0007864867 -0.005500654
2013-12-31 0.013778376 0.0031305000  0.006223193
```

We now turn to discussing the specific steps used to construct an EW portfolio and a VW portfolio.

3.3.1 Equal-Weighted Portfolio

An equal-weighted (EW) index gives equal weight to small firms and large firms. As such, an EW index is sometimes considered as an approach to capture the small capitalization stock premium (i.e., the belief that small capitalization stocks yield higher returns over large capitalization stocks).

These indexes are rebalanced quarterly, which means that in between quarters the constituent weights are allowed to fluctuate based on their performance. We follow this quarterly rebalancing approach.

Quarterly Rebalancing Explained

The quarterly rebalancing approach means that on each rebalancing date, the weight of each firm is set or reset to 1/N where N is the number of securities in the portfolio. For example, if we

have 3 firms in our portfolio and $ 1000 to invest, each firm will be allocated $ 333.33 [= $ 1000/3] or 33.33 % of the portfolio value. Subsequent to this date, each $ 333.33 investment would be allowed to grow at the returns of that security. Then, on each rebalancing date, if we have the same number of firms, we will allocate 33.33 % of the portfolio's then-existing value to each firm. For example, if the aggregate portfolio value as of March 31, 2013 was $ 1500, then each of the three firms would then be allocated $ 500 [=$ 1500 / 3]. Suppose that Stock 1 had a value of $ 200, Stock 2 had a value of $ 500, and Stock 3 had a value of $ 800 at quarter end. Then, this means we would have to sell $ 300 of Stock 3 and purchase $ 300 of Stock 1. That way, each of the three stocks in the portfolio would have an investment of $ 500 after rebalancing.

We now turn to the mechanics of constructing an EW portfolio.

Step 1: Keep Only Variables We Need to Construct EW Portfolio

```
> ### Keep 1st,8th,9th,10th column of above port data
> ewport<-port[c(1,8:10)]
> ### Output
> ewport[c(1:3,nrow(ewport)),]
                date    AMZN.ret     NVDA.ret      IBM.ret
2012-12-31 2012-12-31 0.023207439 0.0132228469  0.009060829
2013-01-02 2013-01-02 0.025670666 0.0375206296  0.025058963
2013-01-03 2013-01-03 0.004547044 0.0007864867 -0.005500654
2013-12-31 2013-12-31 0.013778376 0.0031305000  0.006223193
>
> ### Rename in a way of simplicity
> names(ewport)<-paste(c("date","AMZN","NVDA","IBM"))
> ### Reorder the index from 1 to 2 to 3 to 4 to ...
> rownames(ewport)<-seq(1:nrow(ewport))
> ### Ouput
> ewport[c(1:3,nrow(ewport)),]
          date         AMZN          NVDA           IBM
1   2012-12-31 0.023207439 0.0132228469  0.009060829
2   2013-01-02 0.025670666 0.0375206296  0.025058963
3   2013-01-03 0.004547044 0.0007864867 -0.005500654
253 2013-12-31 0.013778376 0.0031305000  0.006223193
>
```

Step 2: Converting Net Returns to Gross Returns

```
> ### Replace the net return by gross return for AMZN by adding one to net return
> ewport$AMZN<-1+ewport$AMZN
> ### Replace the net return by gross return for NVDA by adding one to net return
> ewport$NVDA<-1+ewport$NVDA
> ### Replace the net return by gross return for IBM by adding one to net return
> ewport$IBM<-1+ewport$IBM
> ### Output
> ewport[c(1:3,nrow(ewport)),]
          date     AMZN     NVDA      IBM
1   2012-12-31 1.023207 1.013223 1.0090608
2   2013-01-02 1.025671 1.037521 1.0250590
3   2013-01-03 1.004547 1.000786 0.9944993
253 2013-12-31 1.013778 1.003130 1.0062232
>
```

Step 3: Calculate EW Portfolio Values for 1Q 2013

The reason why we would want to include December 31, 2012 is because, when we want to

determine how much we made based on the returns in 2013, we are implicitly making the assumption that the investment was made at the closing price on the last trading day of 2012 (i.e., December 31, 2012).

```
> ### We limit date from 2012-12-31 to 2013-03-31 to initiate the Q1 return, note that 2012-12-31
> ### is the date for investment, functioning as the placeholder
> ew.q1<-subset(ewport,
+                 + ewport$date >= as.Date("2012-12-31") &
+                   + ewport$date <= as.Date("2013-03-31"))
> ### Output
> ew.q1[c(1:3,nrow(ew.q1)),]
         date     AMZN     NVDA       IBM
1  2012-12-31 1.023207 1.013223 1.0090608
2  2013-01-02 1.025671 1.037521 1.0250590
3  2013-01-03 1.004547 1.000786 0.9944993
61 2013-03-28 1.004485 1.014230 1.0114279
>
```

Although we need December 31, 2012, we are not interested in the return on that date. Our main interest for December 31, 2012 is for a placeholder to begin our indexing of each security's investment performance to $ 1.

```
> ### Set the first row data except "date" as 1,since 2012-12-31 is a placeholder for indexing,
> ### whose return on that date is meaningless, so set as "1".
> ew.q1[1,2:4]<-1
> ### use cumprod command to calculate cumulative return in Q1 for AMZN
> ew.q1$AMZN<-cumprod(ew.q1$AMZN)
> ### use cumprod command to calculate cumulative return in Q1 for NVDA
> ew.q1$NVDA<-cumprod(ew.q1$NVDA)
> ### use cumprod command to calculate cumulative return in Q1 for IBM
> ew.q1$IBM<-cumprod(ew.q1$IBM)
> ew.q1[c(1:3,nrow(ew.q1)),]
         date     AMZN     NVDA      IBM
1  2012-12-31 1.000000 1.000000 1.000000
2  2013-01-02 1.025671 1.037521 1.025059
3  2013-01-03 1.030334 1.038337 1.019420
61 2013-03-28 1.062263 1.052913 1.118235
>
```

Construct a variable to denote the number of securities in our portfolio.

```
> ### Set up a variable that means the number of securities = 3
> num.sec<-3
> num.sec
[1] 3
>
```

Since in our example each stock has 33 % of the portfolio value, we multiply the cumulative gross returns of each security by 33 % or 1/num.sec to get the index value for each security during the quarter. The variables that hold the index values have a suffix of idx.

```
> ### Calculate the index value of AMZN by multiplying Gross Return with percentage proportion
> ew.q1$AMZN.idx<-(1/num.sec)*ew.q1$AMZN
> ### Calculate the index value of NVDA by multiplying Gross Return with percentage proportion
> ew.q1$NVDA.idx<-(1/num.sec)*ew.q1$NVDA
> ### Calculate the index value of IBM by multiplying Gross Return with percentage proportion
> ew.q1$IBM.idx<-(1/num.sec)*ew.q1$IBM
> ### Output
> ew.q1[c(1:3,nrow(ew.q1)),]
         date     AMZN     NVDA      IBM  AMZN.idx  NVDA.idx   IBM.idx
1  2012-12-31 1.000000 1.000000 1.000000 0.3333333 0.3333333 0.3333333
2  2013-01-02 1.025671 1.037521 1.025059 0.3418902 0.3458402 0.3416863
3  2013-01-03 1.030334 1.038337 1.019420 0.3434448 0.3461122 0.3398068
61 2013-03-28 1.062263 1.052913 1.118235 0.3540878 0.3509710 0.3727449
>
```

To calculate the value of the EW portfolio on each day, we sum the values of the three index variables.

```
> ### Use rowSums command to calculate the sum of 5th to 7th column as EW portfolio value
> ### Then transform it to be data.frame object
> q1.val<-data.frame(rowSums(ew.q1[,5:7]))
> ### Output
> q1.val[c(1:3,nrow(q1.val)),]
[1] 1.000000 1.029417 1.029364 1.077804
> ### Give a name to the value
> names(q1.val)<-paste("port.val")
> ### Import "date" column into the q1.val dataset
> q1.val$date<-ew.q1$date
> ### Output again
> q1.val[c(1:3,nrow(q1.val)),]
   port.val       date
1  1.000000 2012-12-31
2  1.029417 2013-01-02
3  1.029364 2013-01-03
61 1.077804 2013-03-28
> ### In the last row, choose the value in the first column as q2.inv,
> ### which is the final EW return for Q1
> q2.inv<-q1.val[nrow(q1.val),1]
> ### Output this value
> q2.inv
[1] 1.077804
>
```

"q2.inv" will be redistributed equally among the three stocks at the beginning of the second quarter. As such, we will use q2.inv value of $ 1.121 in the second quarter calculations as the starting investment value instead of $ 1.

Step 4: Calculate EW Portfolio Values for 2Q 2013 (which is quite similar to above Q1 situation)

Note that the following two screenshots are very similar to the codes above, because we implement the same technical analysis method here.

Well, it should note that I still write comment for each step of codes one by one to let you understand what the R is doing here for computing and calculating.

```
> ### Limit Q2 duration from 2013-04-01 to 2013-06-30
> ew.q2<-subset(ewport,
+                 + ewport$date >= as.Date("2013-04-01") &
+                   + ewport$date <= as.Date("2013-06-30"))
> ### Output
> ew.q2[c(1:3,nrow(ew.q2)),]
          date      AMZN      NVDA       IBM
62   2013-04-01 0.9816879 0.9672642 0.9956870
63   2013-04-02 1.0065364 0.9895248 1.0093229
64   2013-04-03 0.9837080 0.9877847 0.9920693
125 2013-06-28 1.0005044 1.0021413 0.9767952
>
> ### Again,Calculate cumulative return for each security for Q2
> ew.q2$AMZN<-cumprod(ew.q2$AMZN)
> ew.q2$NVDA<-cumprod(ew.q2$NVDA)
> ew.q2$IBM<-cumprod(ew.q2$IBM)
> ### Output
> ew.q2[c(1:3,nrow(ew.q2)),]
          date      AMZN      NVDA       IBM
62   2013-04-01 0.9816879 0.9672642 0.9956870
63   2013-04-02 0.9881046 0.9571319 1.0049698
64   2013-04-03 0.9720065 0.9454403 0.9969997
125 2013-06-28 1.0420278 1.0998689 0.9001676
>
> ### Again,calculate the index value of each security by multiplying Gross Return with percentage proportio
n
> ew.q2$AMZN.idx<-(q2.inv/num.sec)*ew.q2$AMZN
> ew.q2$NVDA.idx<-(q2.inv/num.sec)*ew.q2$NVDA
> ew.q2$IBM.idx<-(q2.inv/num.sec)*ew.q2$IBM
> ew.q2[c(1:3,nrow(ew.q2)),]
          date      AMZN      NVDA       IBM  AMZN.idx  NVDA.idx   IBM.idx
62   2013-04-01 0.9816879 0.9672642 0.9956870 0.3526889 0.3475069 0.3577184
63   2013-04-02 0.9881046 0.9571319 1.0049698 0.3549942 0.3438667 0.3610533
64   2013-04-03 0.9720065 0.9454403 0.9969997 0.3492107 0.3396663 0.3581899
125 2013-06-28 1.0420278 1.0998689 0.9001676 0.3743671 0.3951475 0.3234013
>
> ### Use rowSums command to calculate the sum of 5th to 7th column as EW portfolio value
> ### Then transform it to be data.frame object
> q2.val<-data.frame(rowSums(ew.q2[,5:7]))
> ### Output
> q2.val[c(1:3,nrow(q2.val)),]
[1] 1.057914 1.059914 1.047067 1.092916
>
> ### Give a name to the value
> names(q2.val)<-paste("port.val")
> ### Import "date" column into the q2.val dataset
> q2.val$date<-ew.q2$date
> ### Output again
> q2.val[c(1:3,nrow(q2.val)),]
     port.val       date
62   1.057914 2013-04-01
63   1.059914 2013-04-02
64   1.047067 2013-04-03
125 1.092916 2013-06-28
>
> ### In the last row, choose the value in the first column as q3.inv,
> ### which is the final EW return for Q2
> q3.inv<-q2.val[nrow(q2.val),1]
> ### Output this value
> q3.inv
[1] 1.092916
>
```

Step 5: Calculate EW Portfolio Values for 3Q 2013

Again, we repeat what we did in step 4.

```
> ### Limit Q3 duration from 2013-07-01 to 2013-09-30
> ew.q3<-subset(ewport,
+               + ewport$date >= as.Date("2013-07-01") &
+                 + ewport$date <= as.Date("2013-09-30"))
> ### Output
> ew.q3[c(1:3,nrow(ew.q3)),]
          date      AMZN      NVDA       IBM
126 2013-07-01 1.0158810 1.0042735 1.0008897
127 2013-07-02 1.0057781 0.9992908 1.0011500
128 2013-07-03 1.0010573 1.0028389 1.0091383
189 2013-09-30 0.9893358 0.9987162 0.9906912
>
> ### Again,Calculate cumulative return for each security for Q3
> ew.q3$AMZN<-cumprod(ew.q3$AMZN)
> ew.q3$NVDA<-cumprod(ew.q3$NVDA)
> ew.q3$IBM<-cumprod(ew.q3$IBM)
> ### Output
> ew.q3[c(1:3,nrow(ew.q3)),]
          date     AMZN     NVDA       IBM
126 2013-07-01 1.015881 1.004273 1.0008897
127 2013-07-02 1.021751 1.003561 1.0020408
128 2013-07-03 1.022831 1.006410 1.0111978
189 2013-09-30 1.125860 1.113850 0.9738148
>
> ### Again,calculate the index value of each security by multiplying Gross Return with percentage proportion
> ew.q3$AMZN.idx<-(q3.inv/num.sec)*ew.q3$AMZN
> ew.q3$NVDA.idx<-(q3.inv/num.sec)*ew.q3$NVDA
> ew.q3$IBM.idx<-(q3.inv/num.sec)*ew.q3$IBM
> ew.q3[c(1:3,nrow(ew.q3)),]
          date     AMZN     NVDA       IBM  AMZN.idx  NVDA.idx   IBM.idx
126 2013-07-01 1.015881 1.004273 1.0008897 0.3700909 0.3658622 0.3646295
127 2013-07-02 1.021751 1.003561 1.0020408 0.3722293 0.3656027 0.3650488
128 2013-07-03 1.022831 1.006410 1.0111978 0.3726229 0.3666406 0.3683847
189 2013-09-30 1.125860 1.113850 0.9738148 0.4101567 0.4057815 0.3547659
>
> ### Use rowSums command to calculate the sum of 5th to 7th column as EW portfolio value
> ### Then transform it to be data.frame object
> q3.val<-data.frame(rowSums(ew.q3[,5:7]))
> ### Output
> q3.val[c(1:3,nrow(q3.val)),]
[1] 1.100582 1.102881 1.107648 1.170704
>
> ### Give a name to the value
> names(q3.val)<-paste("port.val")
> ### Import "date" column into the q3.val dataset
> q3.val$date<-ew.q3$date
> ### Output again
> q3.val[c(1:3,nrow(q3.val)),]
    port.val       date
126 1.100582 2013-07-01
127 1.102881 2013-07-02
128 1.107648 2013-07-03
189 1.170704 2013-09-30
>
> ### In the last row, choose the value in the first column as q3.inv,
> ### which is the final EW return for Q3
> q4.inv<-q3.val[nrow(q3.val),1]
> ### Output this value
> q4.inv
[1] 1.170704
>
```

Step 6: Calculate EW Portfolio Values for 4Q 2013

Again, we repeat what we did in step 5.

```
> ### Limit Q4 duration from 2013-10-01 to 2013-12-31
> ew.q4<-subset(ewport,
+               + ewport$date >= as.Date("2013-10-01") &
+                 + ewport$date <= as.Date("2013-12-31"))
> ### Output
> ew.q4[c(1:3,nrow(ew.q4)),]
          date      AMZN      NVDA       IBM
190 2013-10-01 1.0265802 0.9993575 1.0064803
191 2013-10-02 0.9986289 0.9993569 0.9923809
192 2013-10-03 0.9820599 0.9897035 0.9940529
253 2013-12-31 1.0137784 1.0031305 1.0062232
>
> ### Again,Calculate cumulative return for each security for Q4
> ew.q4$AMZN<-cumprod(ew.q4$AMZN)
> ew.q4$NVDA<-cumprod(ew.q4$NVDA)
> ew.q4$IBM<-cumprod(ew.q4$IBM)
> ### Output
> ew.q4[c(1:3,nrow(ew.q4)),]
          date     AMZN      NVDA       IBM
190 2013-10-01 1.026580 0.9993575 1.0064803
191 2013-10-02 1.025173 0.9987148 0.9988118
192 2013-10-03 1.006781 0.9884316 0.9928718
253 2013-12-31 1.275557 1.0351387 1.0183462
>
> ### Again,calculate the index value of each security by multiplying Gross Return with percentage proportion
> ew.q4$AMZN.idx<-(q4.inv/num.sec)*ew.q4$AMZN
> ew.q4$NVDA.idx<-(q4.inv/num.sec)*ew.q4$NVDA
> ew.q4$IBM.idx<-(q4.inv/num.sec)*ew.q4$IBM
> ew.q4[c(1:3,nrow(ew.q4)),]
          date     AMZN      NVDA       IBM  AMZN.idx  NVDA.idx   IBM.idx
190 2013-10-01 1.026580 0.9993575 1.0064803 0.4006072 0.3899840 0.3927635
191 2013-10-02 1.025173 0.9987148 0.9988118 0.4000579 0.3897332 0.3897710
192 2013-10-03 1.006781 0.9884316 0.9928718 0.3928809 0.3857203 0.3874530
253 2013-12-31 1.275557 1.0351387 1.0183462 0.4977664 0.4039470 0.3973940
>
> ### Use rowSums command to calculate the sum of 5th to 7th column as EW portfolio value
> ### Then transform it to be data.frame object
> q4.val<-data.frame(rowSums(ew.q4[,5:7]))
> ### Output
> q4.val[c(1:3,nrow(q4.val)),]
[1] 1.183355 1.179562 1.166054 1.299107
>
> ### Give a name to the value
> names(q4.val)<-paste("port.val")
> ### Import "date" column into the q3.val dataset
> q4.val$date<-ew.q4$date
> ### Output again
> q4.val[c(1:3,nrow(q4.val)),]
    port.val       date
190 1.183355 2013-10-01
191 1.179562 2013-10-02
192 1.166054 2013-10-03
253 1.299107 2013-12-31
>
```

Step 7: Combine Quarterly EW Portfolio Values into One Data Object

```
> ### Combine four data-sets one after another by stacking on top of one by one
> ew.portval<-rbind(q1.val,q2.val,q3.val,q4.val)
> ### Output
> ew.portval[c(1:3,nrow(ew.portval)),]
    port.val       date
1   1.000000 2012-12-31
2   1.029417 2013-01-02
3   1.029364 2013-01-03
253 1.299107 2013-12-31
>
```

Quarterly rebalancing makes more practical sense as we incur transactions costs only during

the rebalancing dates. This makes the use of this EW portfolio more suitable as a benchmark because it is realistically implementable

3.3.2 Value-Weighted Portfolio

Value-weighted returns are also called capitalization-weighted returns.

In a VW portfolio, the returns of larger firms are given more weight.

In the context of say a VW sector index, one can think of capitalization-weighting as an approach to tracking the changes in the size of that sector.

A value-weighted return r_{vw} is constructed as follows:

$$r_{t,VW} = \sum_{i=1}^{N} w_{i,t} * r_{i,t},$$

where $w_{i,t}$ is the weight of security i at time t and $r_{i,t}$ is the return of asset i and time t.

For our purposes, the weight of security i is equal to the market capitalization of security i divided by the market capitalization of all securities in the portfolio.

If we use the market capitalization on day t as the weights for day t return, we are assuming perfect foresight as to what the end-of day return would be.

Step 1: Keep Only Variables We Need to Construct VW Portfolio

```
> ### Value-weighted portfolio
> ### Select the columns of adjusted closing prices and net returns for these three securities
> vwport<-port[,c(1,2,4,6,8:10)]
> ### Output
> vwport[c(1:3,nrow(vwport)),]
                date AMZN.Close NVDA.Close IBM.Close    AMZN.ret     NVDA.ret      IBM.ret
2012-12-31 2012-12-31    12.5435     3.0650  183.1262 0.023207439 0.0132228469  0.009060829
2013-01-02 2013-01-02    12.8655     3.1800  187.7151 0.025670666 0.0375206296  0.025058963
2013-01-03 2013-01-03    12.9240     3.1825  186.6826 0.004547044 0.0007864867 -0.005500654
2013-12-31 2013-12-31    19.9395     4.0050  179.3212 0.013778376 0.0031305000  0.006223193
>
```

```
> ### change the index series from date object to sequence from 1 to the end
> rownames(vwport)<-seq(1:nrow(vwport))
> ### Output
> vwport[c(1:3,nrow(vwport)),]
          date AMZN.Close NVDA.Close IBM.Close    AMZN.ret     NVDA.ret      IBM.ret
1   2012-12-31    12.5435     3.0650  183.1262 0.023207439 0.0132228469  0.009060829
2   2013-01-02    12.8655     3.1800  187.7151 0.025670666 0.0375206296  0.025058963
3   2013-01-03    12.9240     3.1825  186.6826 0.004547044 0.0007864867 -0.005500654
253 2013-12-31    19.9395     4.0050  179.3212 0.013778376 0.0031305000  0.006223193
>
```

Step 2: Converting Net Returns to Gross Returns

```
> ### Again,net return is overwritten by gross return for each security of these three.
> ### Method is just plus one from the basic net return
> vwport$AMZN.ret<-1+vwport$AMZN.ret
> vwport$NVDA.ret<-1+vwport$NVDA.ret
> vwport$IBM.ret<-1+vwport$IBM.ret
> ### Output
> vwport[c(1:3,nrow(vwport)),]
          date AMZN.Close NVDA.Close IBM.Close AMZN.ret NVDA.ret   IBM.ret
1   2012-12-31    12.5435     3.0650  183.1262 1.023207 1.013223 1.0090608
2   2013-01-02    12.8655     3.1800  187.7151 1.025671 1.037521 1.0250590
3   2013-01-03    12.9240     3.1825  186.6826 1.004547 1.000786 0.9944993
253 2013-12-31    19.9395     4.0050  179.3212 1.013778 1.003130 1.0062232
> |
```

Step 3: Calculate the Market Capitalization of Each Security in the Portfolio

Now, we have to construct the market capitalization weights for each security at the end of each quarter.

A security's market cap is equal to its price multiplied by its shares outstanding.

As such, we need to get the data for those two components.

Let us first discuss how to find the price that is applicable at the end of each quarter.

Construct Series of Calendar Days

```
> ### Create a sequence of "date" object from end of 2012 to end of 2013
> date<-seq(as.Date("2012-12-31"),as.Date("2013-12-31"),by=1)
> ### Convert data: "date" to data.frame object
> date<-data.frame(date)
> ### Output
> date[c(1:3,nrow(date)),]
[1] "2012-12-31" "2013-01-01" "2013-01-02" "2013-12-31"
> |
```

Create Data Object with Daily Prices, Filling in Last Available Price on Non-trading Days

We extract the price data first

```
> ### Extract the first four columns of price data
> PRICE.qtr<-vwport[,c(1,2,3,4)]
> ### Output
> PRICE.qtr[c(1:3,nrow(PRICE.qtr)),]
          date AMZN.Close NVDA.Close IBM.Close
1   2012-12-31    12.5435     3.0650  183.1262
2   2013-01-02    12.8655     3.1800  187.7151
3   2013-01-03    12.9240     3.1825  186.6826
253 2013-12-31    19.9395     4.0050  179.3212
> |
```

We then merge date and PRICE.qtr using a combination of the na.locf and merge commands.

The merge command combines the two data objects. Using the all.x=TRUE option tells R that when merging we should keep all the data in the "by" variable that is available in x=date data object. The "by" variable in this case is the date.

What the na.locf command does is that it copies over the last available data in y=PRICE.qtr when there is a date in x that is not available in y.

For example, recall that there is no trading on New Year's Day, January 1, 2013, so the output below shows data on January 1, 2013 which is equal to the closing price on December 31, 2012.

Going forward, any time there is a weekend or trading holiday, the last available value is used.

```
> ### Here the left hand side is still PRICE.qtr, which means the overall logic here is overwrite.
> ### Here we use merge to combine two datasets, PRICE.qtr and date,
> ### by="date" means that merge is according to date object
> ### all.x = TRUE means that all variables in x = date are kept
> ### na.locf means that in y if some variables are not available which only available in x,
> ### we use last available data in y to fill it again
> PRICE.qtr<-na.locf(merge(x=date,y=PRICE.qtr,by="date",all.x=TRUE))
> ### Output
> PRICE.qtr[c(1:3,nrow(PRICE.qtr)),]
          date AMZN.Close NVDA.Close IBM.Close
1   2012-12-31    12.5435      3.065  183.1262
2   2013-01-01    12.5435      3.065  183.1262
3   2013-01-02    12.8655      3.180  187.7151
366 2013-12-31    19.9395      4.005  179.3212
>
```

Keep Only Prices at the End of Each Calendar Quarter

We do not need data for December 31, 2013 because we are not calculating any returns in the first quarter of 2014.

```
> ### Use subset to limit PRICE.qtr data range of only four dates,
> ### which are the beginning date of the four quarters in 2013
> PRICE.qtr<-subset(PRICE.qtr,
+                     + PRICE.qtr$date==as.Date("2012-12-31") |
+                       + PRICE.qtr$date==as.Date("2013-03-31") |
+                       + PRICE.qtr$date==as.Date("2013-06-30") |
+                       + PRICE.qtr$date==as.Date("2013-09-30"))
> ### Output
> PRICE.qtr
          date AMZN.Close NVDA.Close IBM.Close
1   2012-12-31    12.5435     3.0650  183.1262
91  2013-03-31    13.3245     3.2075  203.9197
182 2013-06-30    13.8845     3.5100  182.7056
274 2013-09-30    15.6320     3.8900  177.0363
>
```

Obtain Shares Outstanding Data from SEC Filings

```
> ### For each security, manually set up the quarter share outstandings
> PRICE.qtr$AMZN.shout<-c(4193066000,4227876000,3890850000,3095592000)
> PRICE.qtr$NVDA.shout<-c(26158724000,25697000000,26973324000,17906988000)
> PRICE.qtr$IBM.shout<-c(269206612,238283505,292425198,243346561)
> ### Output
> PRICE.qtr
          date AMZN.Close NVDA.Close IBM.Close AMZN.shout  NVDA.shout IBM.shout
1   2012-12-31    12.5435     3.0650  183.1262 4193066000 26158724000 269206612
91  2013-03-31    13.3245     3.2075  203.9197 4227876000 25697000000 238283505
182 2013-06-30    13.8845     3.5100  182.7056 3890850000 26973324000 292425198
274 2013-09-30    15.6320     3.8900  177.0363 3095592000 17906988000 243346561
>
```

Calculate Market Capitalization of Each Security

We first check the structure of the data in PRICE.qtr. We see that the date and closing price variables are read-in as character variables.

```
> ### Output the detail of the structure of the PRICE.qtr dataset
> str(PRICE.qtr)
'data.frame':   4 obs. of  7 variables:
 $ date      : Date, format: "2012-12-31" "2013-03-31" "2013-06-30" "2013-09-30"
 $ AMZN.Close: num  12.5 13.3 13.9 15.6
 $ NVDA.Close: num  3.06 3.21 3.51 3.89
 $ IBM.Close : num  183 204 183 177
 $ AMZN.shout: num  4.19e+09 4.23e+09 3.89e+09 3.10e+09
 $ NVDA.shout: num  2.62e+10 2.57e+10 2.70e+10 1.79e+10
 $ IBM.shout : num  2.69e+08 2.38e+08 2.92e+08 2.43e+08
>
```

As this is different from the book output, where date and three security closing price is “str” type of data. But here my PRICE.qtr has already been Date format and “num” type. Hence no need to convert according to the book output.

```
> ### rename dataset with more relation to the weights
> weights<-PRICE.qtr
> ### calculate the market cap for each security by multiplying close price by share outstandings at the end of each quarter
> weights$AMZN.mcap<-weights$AMZN.Close*weights$AMZN.shout
> weights$NVDA.mcap<-weights$NVDA.Close*weights$NVDA.shout
> weights$IBM.mcap<-weights$IBM.Close*weights$IBM.shout
> ### Output
> weights
          date AMZN.Close NVDA.Close IBM.Close AMZN.shout  NVDA.shout IBM.shout   AMZN.mcap   NVDA.mcap    IBM.mcap
1   2012-12-31    12.5435     3.0650  183.1262 4193066000 26158724000 269206612 52595723371 80176489060 49298781178
91  2013-03-31    13.3245     3.2075  203.9197 4227876000 25697000000 238283505 56334333762 82423127500 48590699187
182 2013-06-30    13.8845     3.5100  182.7056 3890850000 26973324000 292425198 54022506825 94676367240 53427706927
274 2013-09-30    15.6320     3.8900  177.0363 3095592000 17906988000 243346561 48390294144 69658183320 43081182321
>
```

Calculate Quarter-end Aggregate Market Capitalization

```
> ### Calculate the total market cap by summarize in row of column 8th to 10th
> weights$tot.mcap<-rowSums(weights[8:10])
> ### Output
> weights
          date AMZN.Close NVDA.Close IBM.Close AMZN.shout  NVDA.shout IBM.shout   AMZN.mcap   NVDA.mcap    IBM.mcap     tot.mcap
1   2012-12-31    12.5435     3.0650  183.1262 4193066000 26158724000 269206612 52595723371 80176489060 49298781178 182070993609
91  2013-03-31    13.3245     3.2075  203.9197 4227876000 25697000000 238283505 56334333762 82423127500 48590699187 187348160449
182 2013-06-30    13.8845     3.5100  182.7056 3890850000 26973324000 292425198 54022506825 94676367240 53427706927 202126580992
274 2013-09-30    15.6320     3.8900  177.0363 3095592000 17906988000 243346561 48390294144 69658183320 43081182321 161129659785
>
```

Step 4: Calculate Quarter-end Weights of Each Security in the Portfolio

```
> ### For each security, dividing each security's market cap by total market cap makes weight.
> weights$AMZN.wgt<-weights$AMZN.mcap/weights$tot.mcap
> weights$NVDA.wgt<-weights$NVDA.mcap/weights$tot.mcap
> weights$IBM.wgt<-weights$IBM.mcap/weights$tot.mcap
> ### Output
> weights
         date AMZN.Close NVDA.Close IBM.Close AMZN.shout  NVDA.shout IBM.shout   AMZN.mcap   NVDA.mcap    IBM.mcap     tot.mcap  AMZN.wgt
1   2012-12-31    12.5435     3.0650  183.1262 4193066000 26158724000 269206612 52595723371 80176489060 49298781178 182070993609 0.2888748
91  2013-03-31    13.3245     3.2075  203.9197 4227876000 25697000000 238283505 56334333762 82423127500 48590699187 187348160449 0.3006933
182 2013-06-30    13.8845     3.5100  182.7056 3890850000 26973324000 292425198 54022506825 94676367240 53427706927 202126580992 0.2672707
274 2013-09-30    15.6320     3.8900  177.0363 3095592000 17906988000 243346561 48390294144 69658183320 43081182321 161129659785 0.3003190
     NVDA.wgt   IBM.wgt
1   0.4403584 0.2707668
91  0.4399463 0.2593604
182 0.4684014 0.2643280
274 0.4323114 0.2673697
>
```

Now we only keep weight data.

```
> ### Keep the only weights data
> WEIGHT<-weights[,c(1,12:14)]
> ### Output
> WEIGHT
           date  AMZN.wgt  NVDA.wgt   IBM.wgt
1   2012-12-31 0.2888748 0.4403584 0.2707668
91  2013-03-31 0.3006933 0.4399463 0.2593604
182 2013-06-30 0.2672707 0.4684014 0.2643280
274 2013-09-30 0.3003190 0.4323114 0.2673697
>

> ### The last day's weight are applicable to the next first date of each quarter
> ### So we can add one to the date
> WEIGHT$date<-WEIGHT$date+1
> ### Output
> WEIGHT
           date  AMZN.wgt  NVDA.wgt   IBM.wgt
1   2013-01-01 0.2888748 0.4403584 0.2707668
91  2013-04-01 0.3006933 0.4399463 0.2593604
182 2013-07-01 0.2672707 0.4684014 0.2643280
274 2013-10-01 0.3003190 0.4323114 0.2673697
>
```

Step 5: Calculating the Quarterly VW Portfolio Values

In the VW portfolio, as shown above, the weights used on the quarterly rebalancing dates are different than that of 33%.

We start with creating a series of dates from December 31, 2012 to December 31, 2013 so that we can find the applicable weights at the beginning of each quarter.

```
> ### Create a sequence of date from the end of 2012 to the end of 2013
> date<-seq(as.Date("2012-12-31"),as.Date("2013-12-31"),by=1)
> ### Convert data type to be data.frame
> date<-data.frame(date)
> ### Output
> date[c(1:3,nrow(date)),]
[1] "2012-12-31" "2013-01-01" "2013-01-02" "2013-12-31"
>
```

We then merge in the WEIGHT into date using a combination of the na.locf command and merge command. The merge command is implemented with the all.x=TRUE option, which means all dates in date are kept in the new data object, such that there will be missing values for the weights on all days except for the quarter end dates. However, recall that the na.locf command can be used to retain the last available value of weights, such that instead of NAs, we end up using the last available value.

```
> ### Merge date and WEIGHT according to date, all variables in x=date are kept, missing values will be filled by last available variables
> vwret<-na.locf(merge(x=date,y=WEIGHT,by="date",all.x=TRUE))
> ### Output
> vwret[c(1:3,nrow(vwret)),]
          date  AMZN.wgt  NVDA.wgt   IBM.wgt
2   2013-01-01 0.2888748 0.4403584 0.2707668
3   2013-01-02 0.2888748 0.4403584 0.2707668
4   2013-01-03 0.2888748 0.4403584 0.2707668
366 2013-12-31 0.3003190 0.4323114 0.2673697
>
```

Again, view details of structure of dataset: vwret.

```
> ### Again, view details of the dataset vwret
> str(vwret)
'data.frame':   365 obs. of  4 variables:
 $ date    : Date, format: "2013-01-01" "2013-01-02" "2013-01-03" "2013-01-04" ...
 $ AMZN.wgt: num  0.289 0.289 0.289 0.289 0.289 ...
 $ NVDA.wgt: num  0.44 0.44 0.44 0.44 0.44 ...
 $ IBM.wgt : num  0.271 0.271 0.271 0.271 0.271 ...
 - attr(*, "na.action")= 'omit' Named int 1
  ..- attr(*, "names")= chr "1"
>
```

Since already in data type “num”, no need to convert data type like in book.

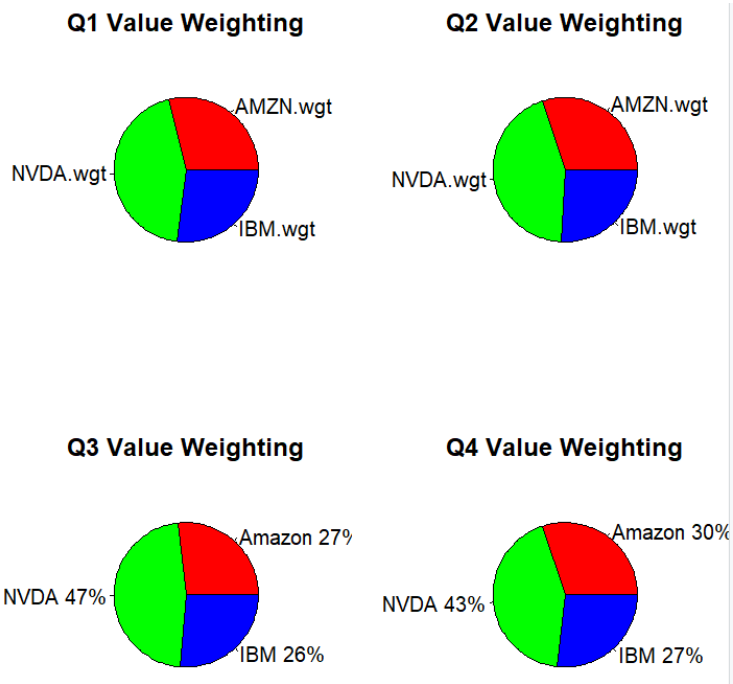
Now extract the row data of first date of each quarter to find the single line of weight data.

```
> ### Extract first quarter weight by subset
> q1.vw.wgt<-subset(vwret,vwret$date==as.Date("2013-01-01"))
> ### Output
> q1.vw.wgt
        date  AMZN.wgt  NVDA.wgt   IBM.wgt
2 2013-01-01 0.2888748 0.4403584 0.2707668
>
> ### Extract second quarter weight by subset
> q2.vw.wgt<-subset(vwret,vwret$date==as.Date("2013-04-01"))
> ### Output
> q2.vw.wgt
         date  AMZN.wgt  NVDA.wgt   IBM.wgt
92 2013-04-01 0.3006933 0.4399463 0.2593604
>
> ### Extract third quarter weight by subset
> q3.vw.wgt<-subset(vwret,vwret$date==as.Date("2013-07-01"))
> ### Output
> q3.vw.wgt
          date  AMZN.wgt  NVDA.wgt  IBM.wgt
183 2013-07-01 0.2672707 0.4684014 0.264328
>
> ### Extract fourth quarter weight by subset
> q4.vw.wgt<-subset(vwret,vwret$date==as.Date("2013-10-01"))
> ### Output
> q4.vw.wgt
          date AMZN.wgt  NVDA.wgt   IBM.wgt
275 2013-10-01 0.300319 0.4323114 0.2673697
>
```

Step 6: Create Pie Charts of the Weights

Note that Q1 and Q2 are use the variable names as labels, while Q3 and Q4 uses re-labeled values and adds the actual percentage weight to the label.

```
> ### Generate a 2x2 plot
> par(mfrow=c(2,2))
> ### Construct the numeric data except first column of date
> Q1.pie.values<-as.numeric(q1.vw.wgt[,-1])
> ### Use the labels of the selected securities in the weight dataset above
> Q1.pie.labels<-names(q1.vw.wgt[,-1])
> ### pie command to show pie chart with 3 type rainbow as color
> pie(Q1.pie.values,labels=Q1.pie.labels,col=rainbow(length(Q1.pie.labels)),main="Q1 Value Weighting")
> Q2.pie.values<-as.numeric(q2.vw.wgt[,-1])
> Q2.pie.labels<-names(q1.vw.wgt[,-1])
> pie(Q2.pie.values,labels=Q2.pie.labels,col=rainbow(length(Q2.pie.labels)),main="Q2 Value Weighting")
> Q3.pie.values<-as.numeric(q3.vw.wgt[,-1])
> Q3.pie.labels<-c("Amazon","NVDA","IBM")
> pct<-round(Q3.pie.values*100)
> Q3.pie.labels<-paste(Q3.pie.labels,pct) # Add Pct to Labels
> Q3.pie.labels<-paste(Q3.pie.labels,"%",sep="") # Add % Sign
> pie(Q3.pie.values,labels=Q3.pie.labels,col=rainbow(length(Q3.pie.labels)),main="Q3 Value Weighting")
> ### Construct the numeric data except first column of date
> Q4.pie.values<-as.numeric(q4.vw.wgt[,-1])
> ### Input the three security names manually
> Q4.pie.labels<-c("Amazon","NVDA","IBM")
> ### For each share, we use percentage data type, so multiplying 100 first
> pct<-round(Q4.pie.values*100)
> ### paste percentage after the name of security
> Q4.pie.labels<-paste(Q4.pie.labels,pct) # Add Pct to Labels
> ### paste the % sign after above label
> Q4.pie.labels<-paste(Q4.pie.labels,"%",sep="") # Add % Sign
> ### pie command to show pie chart with 3 type rainbow as color
> pie(Q4.pie.values,labels=Q4.pie.labels,col=rainbow(length(Q4.pie.labels)),main="Q4 Value Weighting")
>
```



Step 7: Calculating VW Portfolio Values for 1Q 2013

```
> ### Use sub-net command to limit date range in Q1, including only three columns of return variables for three securities
> vw.q1<-subset(vwport[,c(1,5:7)],
+                 + vwport$date >= as.Date("2012-12-31") &
+                   + vwport$date <= as.Date("2013-03-31"))
> ### Output
> vw.q1[c(1:3,nrow(vw.q1)),]
        date AMZN.ret NVDA.ret   IBM.ret
1  2012-12-31 1.023207 1.013223 1.0090608
2  2013-01-02 1.025671 1.037521 1.0250590
3  2013-01-03 1.004547 1.000786 0.9944993
61 2013-03-28 1.004485 1.014230 1.0114279
>
```

```
> ### Rename the names of columns
> names(vw.q1)<-paste(c("date","AMZN","NVDA","IBM"))
> ### Output
> vw.q1[c(1:3,nrow(vw.q1)),]
        date     AMZN     NVDA       IBM
1  2012-12-31 1.023207 1.013223 1.0090608
2  2013-01-02 1.025671 1.037521 1.0250590
3  2013-01-03 1.004547 1.000786 0.9944993
61 2013-03-28 1.004485 1.014230 1.0114279
>
> ### Set the 2th to 4th columns data of first row = 1,
> ### since 2012-12-31 data is the placeholder
> vw.q1[1,2:4]<-1
> ### Use cumprod command to calculate gross return for each security
> vw.q1$AMZN<-cumprod(vw.q1$AMZN)
> vw.q1$NVDA<-cumprod(vw.q1$NVDA)
> vw.q1$IBM<-cumprod(vw.q1$IBM)
> ### Output
> vw.q1[c(1:3,nrow(vw.q1)),]
        date     AMZN     NVDA      IBM
1  2012-12-31 1.000000 1.000000 1.000000
2  2013-01-02 1.025671 1.037521 1.025059
3  2013-01-03 1.030334 1.038337 1.019420
61 2013-03-28 1.062263 1.052913 1.118235
>
```

```
> ### For each security, multiplying Q1 gross return by value-weighted weight makes the index value
> vw.q1$AMZN.idx<-(1*q1.vw.wgt$AMZN.wgt)*vw.q1$AMZN
> vw.q1$NVDA.idx<-(1*q1.vw.wgt$NVDA.wgt)*vw.q1$NVDA
> vw.q1$IBM.idx<-(1*q1.vw.wgt$IBM.wgt)*vw.q1$IBM
> vw.q1[c(1:3,nrow(vw.q1)),]
        date     AMZN     NVDA      IBM  AMZN.idx  NVDA.idx   IBM.idx
1  2012-12-31 1.000000 1.000000 1.000000 0.2888748 0.4403584 0.2707668
2  2013-01-02 1.025671 1.037521 1.025059 0.2962904 0.4568809 0.2775519
3  2013-01-03 1.030334 1.038337 1.019420 0.2976377 0.4572402 0.2760252
61 2013-03-28 1.062263 1.052913 1.118235 0.3068611 0.4636590 0.3027808
>
```

```
> ### Summarize the 5th to 7th column index values to get total return for every day of the portfolio
> q1.vw.val<-data.frame(rowSums(vw.q1[,5:7]))
> ### Output
> q1.vw.val[c(1:3,nrow(q1.vw.val)),]
[1] 1.000000 1.030723 1.030903 1.073301
> ### Name the data-set with "port.val" = "portfolio.value"
> names(q1.vw.val)<-paste("port.val")
> ### Make a new column of date variable in previous data-set
> q1.vw.val$date<-vw.q1$date
> ### Output
> q1.vw.val[c(1:3,nrow(q1.vw.val)),]
   port.val       date
1  1.000000 2012-12-31
2  1.030723 2013-01-02
3  1.030903 2013-01-03
61 1.073301 2013-03-28
> ### Take the index value of last day in Q1 as beginning value for Q2
> q2.vw.inv<-q1.vw.val[nrow(q1.vw.val),1]
> q2.vw.inv
[1] 1.073301
>
```

Step 8: Calculating VW Portfolio Values for 2Q 2013

```
> ### Q2
> vw.q2<-subset(vwport[,c(1,5:7)],
+                 + vwport$date >= as.Date("2013-04-01") &
+                   + vwport$date <= as.Date("2013-06-30"))
> vw.q2[c(1:3,nrow(vw.q2)),]
          date  AMZN.ret  NVDA.ret   IBM.ret
62  2013-04-01 0.9816879 0.9672642 0.9956870
63  2013-04-02 1.0065364 0.9895248 1.0093229
64  2013-04-03 0.9837080 0.9877847 0.9920693
125 2013-06-28 1.0005044 1.0021413 0.9767952
> names(vw.q2)<-paste(c("date","AMZN","NVDA","IBM"))
> vw.q2[c(1:3,nrow(vw.q2)),]
          date      AMZN      NVDA       IBM
62  2013-04-01 0.9816879 0.9672642 0.9956870
63  2013-04-02 1.0065364 0.9895248 1.0093229
64  2013-04-03 0.9837080 0.9877847 0.9920693
125 2013-06-28 1.0005044 1.0021413 0.9767952
> vw.q2$AMZN<-cumprod(vw.q2$AMZN)
> vw.q2$NVDA<-cumprod(vw.q2$NVDA)
> vw.q2$IBM<-cumprod(vw.q2$IBM)
> vw.q2[c(1:3,nrow(vw.q2)),]
          date      AMZN      NVDA       IBM
62  2013-04-01 0.9816879 0.9672642 0.9956870
63  2013-04-02 0.9881046 0.9571319 1.0049698
64  2013-04-03 0.9720065 0.9454403 0.9969997
125 2013-06-28 1.0420278 1.0998689 0.9001676
> vw.q2$AMZN.idx<-(q2.vw.inv*q2.vw.wgt$AMZN.wgt)*vw.q2$AMZN
> vw.q2$NVDA.idx<-(q2.vw.inv*q2.vw.wgt$NVDA.wgt)*vw.q2$NVDA
> vw.q2$IBM.idx<-(q2.vw.inv*q2.vw.wgt$IBM.wgt)*vw.q2$IBM
> vw.q2[c(1:3,nrow(vw.q2)),]
          date      AMZN      NVDA       IBM  AMZN.idx  NVDA.idx   IBM.idx
62  2013-04-01 0.9816879 0.9672642 0.9956870 0.3168244 0.4567371 0.2771712
63  2013-04-02 0.9881046 0.9571319 1.0049698 0.3188953 0.4519527 0.2797552
64  2013-04-03 0.9720065 0.9454403 0.9969997 0.3136999 0.4464319 0.2775366
125 2013-06-28 1.0420278 1.0998689 0.9001676 0.3362982 0.5193523 0.2505813
> q2.vw.val<-data.frame(rowSums(vw.q2[,5:7]))
> q2.vw.val[c(1:3,nrow(q2.vw.val)),]
[1] 1.050733 1.050603 1.037668 1.106232
> names(q2.vw.val)<-paste("port.val")
> q2.vw.val$date<-vw.q2$date
> q2.vw.val[c(1:3,nrow(q2.vw.val)),]
    port.val       date
62  1.050733 2013-04-01
63  1.050603 2013-04-02
64  1.037668 2013-04-03
125 1.106232 2013-06-28
> q3.vw.inv<-q2.vw.val[nrow(q2.vw.val),1]
> q3.vw.inv
[1] 1.106232
>
```

Step 9: Calculating VW Portfolio Values for 3Q 2013

```
> ### Q3
> vw.q3<-subset(vwport[,c(1,5:7)],
+                 + vwport$date >= as.Date("2013-07-01") &
+                   + vwport$date <= as.Date("2013-09-30"))
> vw.q3[c(1:3,nrow(vw.q3)),]
          date  AMZN.ret  NVDA.ret   IBM.ret
126 2013-07-01 1.0158810 1.0042735 1.0008897
127 2013-07-02 1.0057781 0.9992908 1.0011500
128 2013-07-03 1.0010573 1.0028389 1.0091383
189 2013-09-30 0.9893358 0.9987162 0.9906912
> names(vw.q3)<-paste(c("date","AMZN","NVDA","IBM"))
> vw.q3[c(1:3,nrow(vw.q3)),]
          date      AMZN      NVDA       IBM
126 2013-07-01 1.0158810 1.0042735 1.0008897
127 2013-07-02 1.0057781 0.9992908 1.0011500
128 2013-07-03 1.0010573 1.0028389 1.0091383
189 2013-09-30 0.9893358 0.9987162 0.9906912
> vw.q3$AMZN<-cumprod(vw.q3$AMZN)
> vw.q3$NVDA<-cumprod(vw.q3$NVDA)
> vw.q3$IBM<-cumprod(vw.q3$IBM)
> vw.q3[c(1:3,nrow(vw.q3)),]
          date     AMZN     NVDA       IBM
126 2013-07-01 1.015881 1.004273 1.0008897
127 2013-07-02 1.021751 1.003561 1.0020408
128 2013-07-03 1.022831 1.006410 1.0111978
189 2013-09-30 1.125860 1.113850 0.9738148
> vw.q3$AMZN.idx<-(q3.vw.inv*q3.vw.wgt$AMZN.wgt)*vw.q3$AMZN
> vw.q3$NVDA.idx<-(q3.vw.inv*q3.vw.wgt$NVDA.wgt)*vw.q3$NVDA
> vw.q3$IBM.idx<-(q3.vw.inv*q3.vw.wgt$IBM.wgt)*vw.q3$IBM
> vw.q3[c(1:3,nrow(vw.q3)),]
          date     AMZN     NVDA       IBM  AMZN.idx  NVDA.idx   IBM.idx
126 2013-07-01 1.015881 1.004273 1.0008897 0.3003588 0.5203748 0.2926682
127 2013-07-02 1.021751 1.003561 1.0020408 0.3020943 0.5200058 0.2930047
128 2013-07-03 1.022831 1.006410 1.0111978 0.3024137 0.5214820 0.2956823
189 2013-09-30 1.125860 1.113850 0.9738148 0.3328754 0.5771530 0.2847512
> q3.vw.val<-data.frame(rowSums(vw.q3[,5:7]))
> q3.vw.val[c(1:3,nrow(q3.vw.val)),]
[1] 1.113402 1.115105 1.119578 1.194780
> names(q3.vw.val)<-paste("port.val")
> q3.vw.val$date<-vw.q3$date
> q3.vw.val[c(1:3,nrow(q3.vw.val)),]
    port.val       date
126 1.113402 2013-07-01
127 1.115105 2013-07-02
128 1.119578 2013-07-03
189 1.194780 2013-09-30
> q4.vw.inv<-q3.vw.val[nrow(q3.vw.val),1]
> q4.vw.inv
[1] 1.19478
>
```

Step 10: Calculating VW Portfolio Values for 4Q 2013

```
> ### Q4
> vw.q4<-subset(vwport[,c(1,5:7)],
+                 + vwport$date >= as.Date("2013-10-01") &
+                   + vwport$date <= as.Date("2013-12-31"))
> vw.q4[c(1:3,nrow(vw.q4)),]
          date  AMZN.ret  NVDA.ret   IBM.ret
190 2013-10-01 1.0265802 0.9993575 1.0064803
191 2013-10-02 0.9986289 0.9993569 0.9923809
192 2013-10-03 0.9820599 0.9897035 0.9940529
253 2013-12-31 1.0137784 1.0031305 1.0062232
> names(vw.q4)<-paste(c("date","AMZN","NVDA","IBM"))
> vw.q4[c(1:3,nrow(vw.q4)),]
          date      AMZN      NVDA       IBM
190 2013-10-01 1.0265802 0.9993575 1.0064803
191 2013-10-02 0.9986289 0.9993569 0.9923809
192 2013-10-03 0.9820599 0.9897035 0.9940529
253 2013-12-31 1.0137784 1.0031305 1.0062232
> vw.q4$AMZN<-cumprod(vw.q4$AMZN)
> vw.q4$NVDA<-cumprod(vw.q4$NVDA)
> vw.q4$IBM<-cumprod(vw.q4$IBM)
> vw.q4[c(1:3,nrow(vw.q4)),]
          date     AMZN      NVDA       IBM
190 2013-10-01 1.026580 0.9993575 1.0064803
191 2013-10-02 1.025173 0.9987148 0.9988118
192 2013-10-03 1.006781 0.9884316 0.9928718
253 2013-12-31 1.275557 1.0351387 1.0183462
> vw.q4$AMZN.idx<-(q4.vw.inv*q4.vw.wgt$AMZN.wgt)*vw.q4$AMZN
> vw.q4$NVDA.idx<-(q4.vw.inv*q4.vw.wgt$NVDA.wgt)*vw.q4$NVDA
> vw.q4$IBM.idx<-(q4.vw.inv*q4.vw.wgt$IBM.wgt)*vw.q4$IBM
> vw.q4[c(1:3,nrow(vw.q4)),]
          date     AMZN      NVDA       IBM  AMZN.idx  NVDA.idx   IBM.idx
190 2013-10-01 1.026580 0.9993575 1.0064803 0.3683524 0.5161850 0.3215180
191 2013-10-02 1.025173 0.9987148 0.9988118 0.3678473 0.5158530 0.3190683
192 2013-10-03 1.006781 0.9884316 0.9928718 0.3612481 0.5105416 0.3171708
253 2013-12-31 1.275557 1.0351387 1.0183462 0.4576889 0.5346666 0.3253085
> q4.vw.val<-data.frame(rowSums(vw.q4[,5:7]))
> q4.vw.val[c(1:3,nrow(q4.vw.val)),]
[1] 1.206055 1.202769 1.188960 1.317664
> names(q4.vw.val)<-paste("port.val")
> q4.vw.val$date<-vw.q4$date
> q4.vw.val[c(1:3,nrow(q4.vw.val)),]
    port.val       date
190 1.206055 2013-10-01
191 1.202769 2013-10-02
192 1.188960 2013-10-03
253 1.317664 2013-12-31
>
```

Step 11: Combining Quarterly VW Portfolio Values into One Data Object

```
> ### Combine the four quarterly v-w portfolio gross return
> vw.portval<-rbind(q1.vw.val,q2.vw.val,q3.vw.val,q4.vw.val)
> ### Output
> vw.portval[c(1:3,nrow(vw.portval)),]
    port.val       date
1   1.000000 2012-12-31
2   1.030723 2013-01-02
3   1.030903 2013-01-03
253 1.317664 2013-12-31
>
```

3.3.2 Normalized EW and VW Portfolio Price Chart

We now compare the performance of an investment in the EW portfolio and an investment in the VW portfolio over the 1-year period from December 31, 2012 to December 31, 2013.

Step 1: Combine the Data

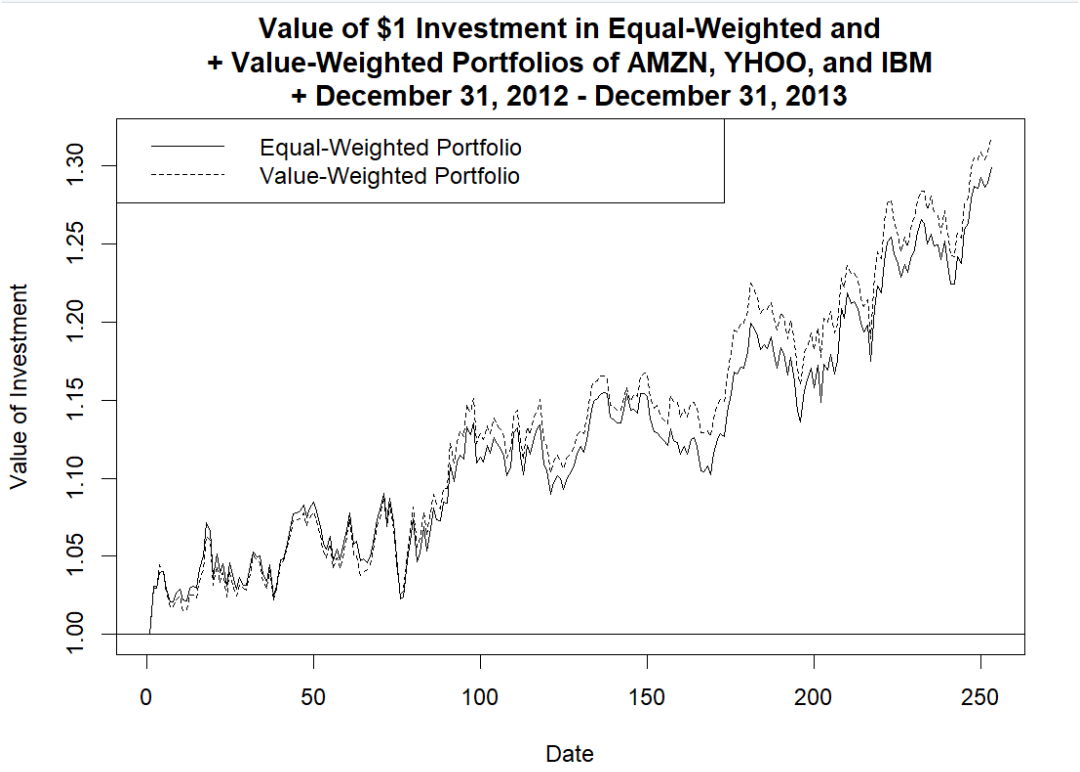
```
> ### merge two method portfolio by the sharing date object
> port.val<-merge(vw.portval,ew.portval,by="date")
> ### Output
> port.val[c(1:3,nrow(port.val)),]
          date port.val.x port.val.y
1   2012-12-31   1.000000   1.000000
2   2013-01-02   1.030723   1.029417
3   2013-01-03   1.030903   1.029364
253 2013-12-31   1.317664   1.299107
>
```

Step 2: Rename the Variables

```
> ### Rename with simple column names
> names(port.val)<-paste(c("date","VW.cum","EW.cum"))
> ### Output
> port.val[c(1:3,nrow(port.val)),]
          date   VW.cum   EW.cum
1   2012-12-31 1.000000 1.000000
2   2013-01-02 1.030723 1.029417
3   2013-01-03 1.030903 1.029364
253 2013-12-31 1.317664 1.299107
>
```

Step 3: Plot the Data

```
> ### Create a 1x1 plot
> par(mfrow=c(1,1))
> ### Calculate y range of VW and EW index returns
> y.range<-range(port.val[,2:3])
> ### Output
> y.range
[1] 1.000000 1.317664
> plot(port.val$EW.cum,type="l",xlab="Date",ylab="Value of Investment",ylim=y.range,lty=1,main="Value of $1 Investment in Equal-Weighted and
+ + Value-Weighted Portfolios of AMZN, YHOO, and IBM
+ + December 31, 2012 - December 31, 2013")
> lines(port.val$VW.cum,lty=2)
> abline(h=1,lty=1)
> legend("topleft",c("Equal-Weighted Portfolio","Value-Weighted Portfolio"),lty=c(1,2))
>
```



3.3.4 Saving Benchmark Portfolio Returns into a CSV File

In this section, we show how to then save to a CSV file the EW and VW portfolio returns into a CSV file that can be used later.

Because the way the portfolio is constructed, we can calculate the portfolio returns by basing them off the changes in the daily portfolio values.

Step 1: Convert the Data into an xts Object

```
> ### Use xts(·) to convert to xts data type
> ### Only select VW and EW columns, ordered by date
> port.xts<-xts(port.val[,2:3],order.by=port.val[,1])
> ### Output
> port.xts[c(1:3,nrow(port.xts)),]
             VW.cum   EW.cum
2012-12-31 1.000000 1.000000
2013-01-02 1.030723 1.029417
2013-01-03 1.030903 1.029364
2013-12-31 1.317664 1.299107
>
```

```
> ### Calculate Lagged datasets, which means one day earlier data variables
> port.xts$Lag.VW<-Lag(port.xts$VW.cum,k=1)
> port.xts$Lag.EW<-Lag(port.xts$EW.cum,k=1)
> ### Output
> port.xts[c(1:3,nrow(port.xts)),]
             VW.cum   EW.cum   Lag.VW   Lag.EW
2012-12-31 1.000000 1.000000       NA       NA
2013-01-02 1.030723 1.029417 1.000000 1.000000
2013-01-03 1.030903 1.029364 1.030723 1.029417
2013-12-31 1.317664 1.299107 1.307763 1.288624
>
```

Step 2: Create EW and VW Returns

```
> ### For each type of portfolio, daily return is calculate by gross index values divided by gross lagged index values
> port.xts$VW.ret<-port.xts$VW.cum/port.xts$Lag.VW-1
> port.xts$EW.ret<-port.xts$EW.cum/port.xts$Lag.EW-1
> ### Output
> port.xts[c(1:3,nrow(port.xts)),]
             VW.cum   EW.cum   Lag.VW   Lag.EW       VW.ret        EW.ret
2012-12-31 1.000000 1.000000       NA       NA           NA            NA
2013-01-02 1.030723 1.029417 1.000000 1.000000 0.0307232679  2.941675e-02
2013-01-03 1.030903 1.029364 1.030723 1.029417 0.0001744982 -5.139738e-05
2013-12-31 1.317664 1.299107 1.307763 1.288624 0.0075709348  8.135480e-03
>
```

Step 3: Clean up the Data

```
> ### Only keep cumulative index value and daily return values
> Port.Ret<-port.xts[,c(1,2,5,6)]
> ### Output
> Port.Ret[c(1:3,nrow(Port.Ret)),]
             VW.cum   EW.cum       VW.ret        EW.ret
2012-12-31 1.000000 1.000000           NA            NA
2013-01-02 1.030723 1.029417 0.0307232679  2.941675e-02
2013-01-03 1.030903 1.029364 0.0001744982 -5.139738e-05
2013-12-31 1.317664 1.299107 0.0075709348  8.135480e-03
>
```

Step 4: Create a Date Variable in the Data Object

If we directly save Port.Ret into a CSV file, we will not have a date variable. So, we first create a DATE object that holds the dates in Port.Ret. Then, we combine DATE with Port.Ret

```
> ### Since the index of Port.Ret is time format, data.frame(index(Port.Ret)) just gives the date columns
> ### in data.frame format, combined with original Port.Ret
> csv.port<-cbind(data.frame(index(Port.Ret)),data.frame(Port.Ret))
> ### Give name of date for date object
> names(csv.port)[1]<-paste("date")
> ### Output
> csv.port[c(1:3,nrow(csv.port)),]
                date    VW.cum    EW.cum        VW.ret        EW.ret
2012-12-31 2012-12-31 1.000000 1.000000           NA            NA
2013-01-02 2013-01-02 1.030723 1.029417 0.0307232679  2.941675e-02
2013-01-03 2013-01-03 1.030903 1.029364 0.0001744982 -5.139738e-05
2013-12-31 2013-12-31 1.317664 1.299107 0.0075709348  8.135480e-03
>
```

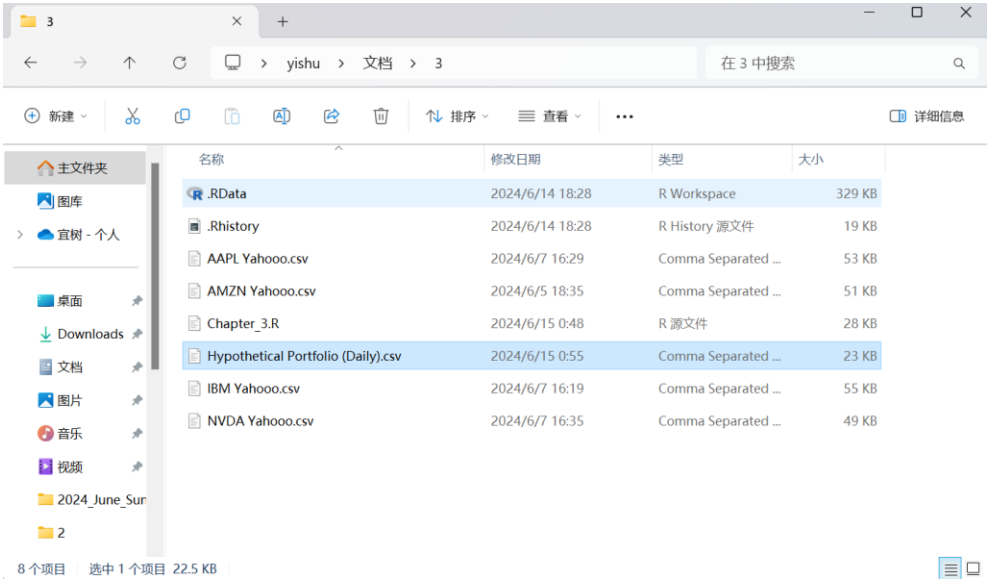
Step 5: Replace Index with Observation Numbers

```
> ### Re-Assign the sequence series of numerical index of 1,2,3,...
> ### This is achieved by rownames(·) command and sequence(·) command
> ### Note we use double ":" in sequence(·) to pass the date sequence
> rownames(csv.port)<-seq(1:nrow(csv.port))
> ### Output
> csv.port[c(1:3,nrow(csv.port)),]
          date    VW.cum    EW.cum        VW.ret        EW.ret
1   2012-12-31 1.000000 1.000000           NA            NA
2   2013-01-02 1.030723 1.029417 0.0307232679  2.941675e-02
3   2013-01-03 1.030903 1.029364 0.0001744982 -5.139738e-05
253 2013-12-31 1.317664 1.299107 0.0075709348  8.135480e-03
>
```

Step 6: Save the Data to a CSV File

```
> ### Use write.csv command to save the csv file of above dataset
> write.csv(csv.port,"Hypothetical Portfolio (Daily).csv")
>
```

We see that the file is automatically saved in the folder.



3.4 Further Reading

Calculating returns is at the heart of most investment applications. We have only touched upon the basic return concepts in the previous chapter and in this chapter. There are more advanced return concepts that are used in practice, such as Time-Weighted Returns and Money-Weighted Returns.

Chapter 4 Risk

For most individual investors, it is unlikely to figure out any good chances of profiting from arbitrage opportunities because these opportunities are mostly captured quickly by large corporations to make profit in a sudden of time and eliminated once these corporations finish the process of making profit.

Hence it is very important that the minor investors should fully understand potential risks in the market investment.

As a concept, risk is fairly easy to grasp.

We can summarize some basic principles related to risks from our daily knowledge, here are some examples below:

1. Larger range of changes in price is likely to cause more risks compared to the portfolio of stable securities.
2. More frequent price changes mean that there are more potential risks inside the portfolio of investment.
3. Generally speaking, savings in bank account is less risky than the investment in stock market.

More importantly, we haven't come up with an exquisite measurement of risks to quantify its influence on the stock market.

Markowitz gave us the most common measure of risk we use today, which is the variance of a security's price. Variance does not care whether the deviation from the average is a positive deviation or a negative deviation—both are treated as risk.

Apart from variance calculation, some other techniques for measuring risks are measuring loss or downside risk, such as Value-at-Risk and Expected Shortfall. Other measures are modifications of the close-to-close calculation that is typically applied when calculating variance.

First, we begin with a discussion of the risk-return trade-off. At its core, this means that we should expect higher returns only if we are willing to take on more risk.

Next, we discuss individual security risk, as measured typically by the variance (or standard deviation) of returns.

Third, we discuss portfolio risk and emphasize that in the context of a portfolio we are less concerned with individual security variance but are more concerned with the

covariance of the securities in the portfolio (i.e., in a portfolio context, the way the assets in the portfolio move together is more important).

Then, we discuss two measures of portfolio loss: Value-at-Risk (VaR) and Expected Shortfall (ES).

Finally, we end with a discussion of several other measures of risk that are used in practice.

4.1 Risk-Return Trade-Off

Here is the screenshot of the explanation of the factors in the .txt file.

Rm-Rf, the excess return on the market, is the value-weighted return on all NYSE, AMEX, and NASDAQ stocks (from CRSP) minus the one-month Treasury bill rate (from Ibbotson Associates).

SMB (Small Minus Big) is the average return on three small portfolios minus the average return on three big portfolios,

SMB = 1/3 (Small Value + Small Neutral + Small Growth)
- 1/3 (Big Value + Big Neutral + Big Growth).

HML (High Minus Low) is the average return on two value portfolios minus the average return on two growth portfolios,

HML = 1/2 (Small Value + Big Value)
- 1/2 (Small Growth + Big Growth).

See Fama/French, 1993, "Common Risk Factors in the Returns on Stocks and Bonds," *Journal of Financial Economics*, for a complete description of the factor returns.

The only way you can get a higher expected return from a security is if you are willing to take on more risk.

To see why, let us first take a look at how stocks and bonds have performed over the last 50 years.

Step 1: Import Fama-French Data from Professor Kenneth French's Website

```
> ### Let's begin!
> ### Read the .txt file and save as FF.raw data
> ### widths mean the width of each column which is fixed for better visualization
> ### skip = 4 means the first four lines are skipped
> FF.raw<-read.fwf(file="F-F_Research_Data_Factors.txt",widths=c(6,8,8,8,8),skip=4)
> ### See the head of the raw data
> head(FF.raw)
      V1       V2       V3       V4       V5
1 192607     2.62    -2.16    -2.92     0.22
2 192608     2.56    -1.50     4.88     0.25
3 192609     0.36    -1.38    -0.01     0.23
4 192610    -3.43     0.04     0.71     0.32
5 192611     2.39    -0.20    -0.31     0.31
6 192612     2.77    -0.01    -0.10     0.28
> 
> ### See the tail of the raw data
> tail(FF.raw)
         V1       V2       V3       V4       V5
1056   2003    32.12    27.63     4.57     1.02
1057   2004    11.82     5.02     9.41     1.19
1058   2005     4.35    -2.09     8.80     2.98
1059   2006    11.41     0.30    14.25     4.81
1060   <NA>     <NA>     <NA>     <NA>     <NA>
1061 Copyri ght 2007  Kenneth  R. Fren       ch
> 
```

Step 2: Clean up Data

The bottom part of the file contains summary annual data, which we do not need.

As such, we delete the data from Row 981 to Row 1066.

We also rename the variable names from V1 to V5 to something more meaningful.

```
> ### Remove the data rows from 977th row to 1066th row which we don't need
> FF.raw<-FF.raw[-977:-1066,]
> ### Rename meaningfully
> names(FF.raw)<-paste(c("text.date","RmxRf","SMB","HML","Rf"))
> ### See the head of the changed data
> head(FF.raw)
  text.date    RmxRf      SMB      HML       Rf
1    192607     2.62    -2.16    -2.92     0.22
2    192608     2.56    -1.50     4.88     0.25
3    192609     0.36    -1.38    -0.01     0.23
4    192610    -3.43     0.04     0.71     0.32
5    192611     2.39    -0.20    -0.31     0.31
6    192612     2.77    -0.01    -0.10     0.28
> ### See the tail of the changed data
> tail(FF.raw)
     text.date    RmxRf      SMB      HML       Rf
971     200705     3.48    -0.08    -0.11     0.41
972     200706    -1.88     0.70    -1.03     0.40
973     200707    -3.57    -2.64    -3.03     0.40
974     200708     0.74    -0.08    -2.42     0.42
975     200709     3.77    -2.52    -2.14     0.32
976     200710     2.26     0.15    -1.97     0.32
> View(FF.raw)
> 
```

As such, we need to convert the data to numeric variables. For the date variable, it is easier for us to create a sequence of monthly dates rather than converting the text.date variable, so we take that simpler approach.

```
> ### Remove first, third and fourth columns
> FF.raw<-FF.raw[,c(-1,-3,-4)]
> ### change RmxRf column to be character then to be numeric and then divided by 100 to make percentage
> FF.raw$RmxRf<-as.numeric(FF.raw$RmxRf)/100
> ### change Rf column to be character then to be numeric and then divided by 100 to make percentage
> FF.raw$Rf<-as.numeric(FF.raw$Rf)/100
> ### Create a sequence of date object of monthly data
> FF.raw$date<-seq(as.Date("1926-07-01"),as.Date("2007-10-31"),by="months")
> ### "yearmon" is a class for representing monthly data
> FF.raw$date<-as.yearmon(FF.raw$date,"%Y-%m-%d")
> ### Output
> FF.raw[c(1:3,nrow(FF.raw)),]
     RmxRf     Rf      date
1   0.0262 0.0022  7月 1926
2   0.0256 0.0025  8月 1926
3   0.0036 0.0023  9月 1926
976 0.0226 0.0032 10月 2007
> ### See details of this data object after changes
> str(FF.raw)
'data.frame':   976 obs. of  3 variables:
 $ RmxRf: num  0.0262 0.0256 0.0036 -0.0343 0.0239 0.0277 -0.001 0.0432 0.0032 0.0041 ...
 $ Rf   : num  0.0022 0.0025 0.0023 0.0032 0.0031 0.0028 0.0025 0.0026 0.003 0.0025 ...
 $ date : 'yearmon' num  7月 1926 8月 1926 9月 1926 10月 1926 ...
>
```

Step 3: Calculate Raw Market Return Variable

```
> ### Raw market return = (Raw market return - Raw risk-free rate return) + Raw risk-free rate return
> FF.raw$Rm<-FF.raw$RmxRf+FF.raw$Rf
> ### Output
> FF.raw[c(1:3,nrow(FF.raw)),]
     RmxRf     Rf      date     Rm
1   0.0262 0.0022  7月 1926 0.0284
2   0.0256 0.0025  8月 1926 0.0281
3   0.0036 0.0023  9月 1926 0.0059
976 0.0226 0.0032 10月 2007 0.0258
>
```

Step 4: Subset Data from December 1963 to December 2013

```
> ### Use subset to limit time duration of 50 years beginning from 1957 to 2006
> ### Note that December, 1956 is imported as a placeholder
> FF<-subset(FF.raw,FF.raw$date>="1956-12-01" & FF.raw$date<="2006-12-31")
> ### Output
> FF[c(1:3,nrow(FF)),]
      RmxRf     Rf      date      Rm
366  0.0325 0.0024 12月 1956  0.0349
367 -0.0347 0.0027  1月 1957 -0.0320
368 -0.0211 0.0024  2月 1957 -0.0187
966  0.0068 0.0040 12月 2006  0.0108
>
```

Step 5: Calculate Gross Returns for the Market and Risk-free Rate

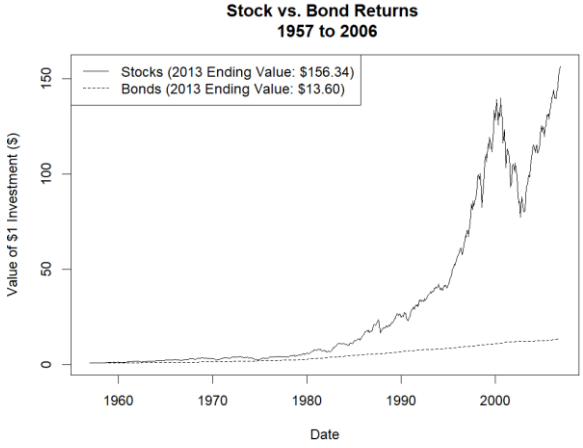
```
> ### Gross return of Rm should add one
> FF$Gross.Rm<-1+FF$Rm
> ### Set first row of Gross Rm = 1 since 1956.12 is not calculated for return
> FF$Gross.Rm[1]<-1
> ### Gross return of Rf should add one
> FF$Gross.Rf<-1+FF$Rf
> ### Set first row of Gross Rf = 1 since 1956.12 is not calculated for return
> FF$Gross.Rf[1]<-1
> ### Output
> FF[c(1:3,nrow(FF)),]
      RmxRf     Rf      date      Rm Gross.Rm Gross.Rf
366  0.0325 0.0024 12月 1956  0.0349   1.0000   1.0000
367 -0.0347 0.0027  1月 1957 -0.0320   0.9680   1.0027
368 -0.0211 0.0024  2月 1957 -0.0187   0.9813   1.0024
966  0.0068 0.0040 12月 2006  0.0108   1.0108   1.0040
>
```

Step 6: Calculate Cumulative Returns for the Market and Risk-free Rate

```
> ### use cumprod command to calculate cumulative return for both Rm and Rf
> FF$cum.Rm<-cumprod(FF$Gross.Rm)
> FF$cum.Rf<-cumprod(FF$Gross.Rf)
> ### Output
> FF[c(1:3,nrow(FF)),]
      RmxRf     Rf      date      Rm Gross.Rm Gross.Rf      cum.Rm    cum.Rf
366  0.0325 0.0024 12月 1956  0.0349   1.0000   1.0000   1.0000000  1.000000
367 -0.0347 0.0027  1月 1957 -0.0320   0.9680   1.0027   0.9680000  1.002700
368 -0.0211 0.0024  2月 1957 -0.0187   0.9813   1.0024   0.9498984  1.005106
966  0.0068 0.0040 12月 2006  0.0108   1.0108   1.0040 156.3380535 13.597227
>
```

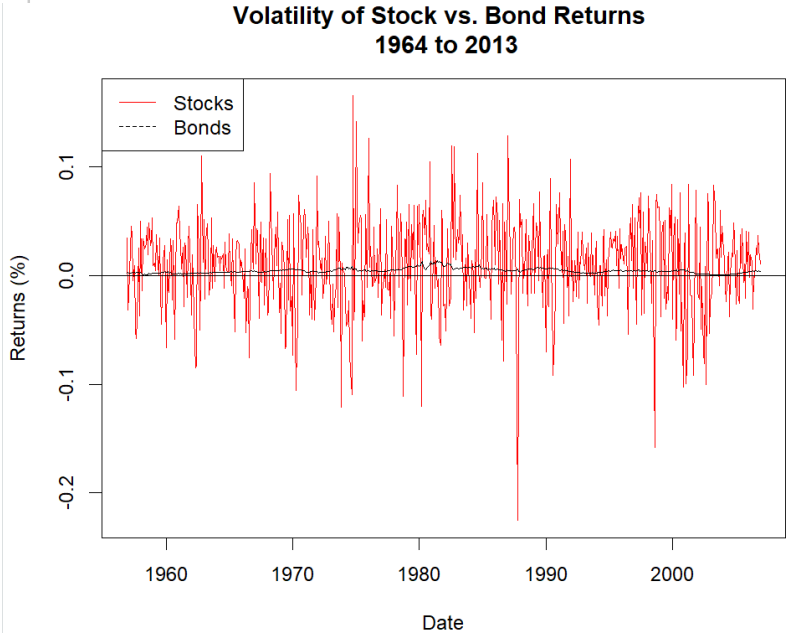
Step 7: Plot the Data

```
> ### Create two titles
> title1<-"Stock vs. Bond Returns"
> title2<-"1957 to 2006"
> ### Plot the data
> plot(x=FF$date,y=FF$cum.Rm,type="l",xlab="Date",ylab="Value of $1 Investment ($)",ylim=y.range,main=paste(title
1,"\n",title2))
> lines(x=FF$date,y=FF$cum.Rf,lty=2)
> legend("topleft",c("Stocks (2013 Ending Value: $156.34)","Bonds (2013 Ending Value: $13.60)"),lty=c(1,2))
>
```



Step 8: Plot Stock and Bond Returns

```
> ### Calculate y range of two returns
> y.range<-range(FF$Rm,FF$Rf)
> ### Output y range
> y.range
[1] -0.2253  0.1656
> ### Create two titles
> title1<-"Volatility of Stock vs. Bond Returns"
> title2<-"1964 to 2013"
> ### Plot the data
> plot(x=FF$date,FF$Rm,type="l",xlab="Date",ylab="Returns (%)",ylim=y.range,col="red",main=paste(title1,"\n",title2))
> lines(x=FF$date,y=FF$Rf)
> abline(h=0)
> legend("topleft",c("Stocks","Bonds"),lty=c(1,2),col=c("red","black"))
>
```



4.2 Individual Security Risk

It is common for investors to use variance or, its positive square root, standard deviation as the measure of risk.

The variance of an asset's return, σ2, is

$$\sigma^2 = \frac{1}{T-1}\sum_{t=1}^{T}(R_t - \bar{R})^2,$$

where R_t is the return of the asset on day t for $t = 1, ..., T$ and $\bar{R}$ is the average return of the asset over days *1* to *T*. The standard deviation is equal to the square root of the variance and is

denoted by σ.

We now calculate the variance and standard deviation of AMZN's returns.

Note that the variance and standard deviation calculations change depending on the time period we choose.

Step 1: Import AMZN Data from Yahoo Finance

```
> ### Read the csv file, which contains the header
> data.AMZN<-read.csv("AMZN Yahooo.csv",header=TRUE)
> ### Change the order of the columns
> data.AMZN<-data.AMZN[,c(1,2,3,4,5,7,6)]
> ### Change the Date column to be a "Date" object with a specific format by creating a new variable
> date<-as.Date(data.AMZN$Date,format="%Y-%m-%d")
> ### combine the new variable with the remaining columns
> data.AMZN<-cbind(date, data.AMZN[,-1])
> ### order the data-set by the date column
> data.AMZN<-data.AMZN[order(data.AMZN$date),]
> ### Use the library of "xts"
> library(xts)
> ### keep the first column as order and remaining column as content
> data.AMZN<-xts(data.AMZN[,2:7],order.by=data.AMZN[,1])
> ### Rename meaningfully
> names(data.AMZN)<-paste(c("AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"))
> ### Output
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
>
```

Step 2: Calculate Returns

```
> ### Extract the sixth column as raw return
> AMZN.ret<-data.AMZN[,6]
> ### Open the library quantmod
> library(quantmod)
> ### make a variable Return by taking delta of adjusted price in the AMZN.ret data-set
> AMZN.ret$Return=Delt(AMZN.ret$AMZN.Adjusted)
> ### Eliminate first row and take only second column Return
> AMZN.ret<-AMZN.ret[-1,2]
> ### Output
> AMZN.ret[c(1:3,nrow(AMZN.ret)),]
                Return
2011-01-03 0.023444444
2011-01-04 0.004288351
2011-01-05 0.013026323
2013-12-31 0.013778376
>
```

Step 3: Calculate Full Period (2011–2013) Variance and Standard Deviation

```
> ### Use var command to output the total variance of the data-set in all time
> AMZN.var.full<-var(AMZN.ret$Return)
> AMZN.var.full
            Return
Return 0.000425095
> ### Use sd command to output the total stand diviation of the data-set in all time
> AMZN.sd.full<-sd(AMZN.ret$Return)
> AMZN.sd.full
[1] 0.02061783
>
```

Step 4: Calculate Variance and Standard Deviation for 2011

```
> ### Use subset command to limit the year to be 2011, with index to set the range of dates
> AMZN.2011<-subset(AMZN.ret,index(AMZN.ret) >= "2011-01-01" &index(AMZN.ret) <= "2011-12-31")
> ### Output
> AMZN.2011[c(1:3,nrow(AMZN.2011)),]
                Return
2011-01-03  0.023444444
2011-01-04  0.004288351
2011-01-05  0.013026323
2011-12-30 -0.004371333
>
> ### Again use var and sd command to calculate variance and stand diviation, but for 2011.
> AMZN.var.2011<-var(AMZN.2011)
> ### Output
> AMZN.var.2011
            Return
Return 0.0005852692
> AMZN.sd.2011<-sd(AMZN.2011)
> ### Output
> AMZN.sd.2011
[1] 0.02419234
>
```

Step 5: Calculate Variance and Standard Deviation for 2012 and 2013

2012

```
> ### Use subset command to limit the year to be 2012, with index to set the range of dates
> AMZN.2012<-subset(AMZN.ret,index(AMZN.ret) >= "2012-01-01" & index(AMZN.ret) <= "2012-12-31")
> ### Output
> AMZN.2012[c(1:3,nrow(AMZN.2012)),]
                 Return
2012-01-03  0.0342576545
2012-01-04 -0.0084901972
2012-01-05  0.0005633485
2012-12-31  0.0232074394
>
> ### Again use var and sd command to calculate variance and stand diviation, but for 2012.
> AMZN.var.2012<-var(AMZN.2012)
> ### Output
> AMZN.var.2012
             Return
Return 0.0004015231
> AMZN.sd.2012<-sd(AMZN.2012)
> ### Output
> AMZN.sd.2012
[1] 0.02003804
>
```

2013

```
> ### Use subset command to limit the year to be 2013, with index to set the range of dates
> AMZN.2013<-subset(AMZN.ret,index(AMZN.ret) >= "2013-01-01" &index(AMZN.ret) <= "2013-12-31")
> ### Output
> AMZN.2013[c(1:3,nrow(AMZN.2013)),]
                Return
2013-01-02 0.025670666
2013-01-03 0.004547044
2013-01-04 0.002592077
2013-12-31 0.013778376
>
> ### Again use var and sd command to calculate variance and standard deviation, but for 2013.
> AMZN.var.2013<-var(AMZN.2013)
> ### Output
> AMZN.var.2013
              Return
Return 0.0002897267
> AMZN.sd.2013<-sd(AMZN.2013)
> ### Output
> AMZN.sd.2013
[1] 0.01702136
>
```

Step 6: Calculate Average Return for the Full Period and Each of the Subperiods

```
> ### Use mean command to calculate average return for four periods: full, 2011, 2012 and 2013
> ### And then output the numeric value
> mean.ret.full<-mean(AMZN.ret)
> mean.ret.full
[1] 0.001266679
> mean.ret.2011<-mean(AMZN.2011)
> mean.ret.2011
[1] 0.0001385631
> mean.ret.2012<-mean(AMZN.2012)
> mean.ret.2012
[1] 0.001680247
> mean.ret.2013<-mean(AMZN.2013)
> mean.ret.2013
[1] 0.00198451
>
```

Step 7: Combine All Data

```
> ### Use rbind command to stack rows together by their sequence of order
> AMZN.risk<-rbind(
+   ### Use cbind to combine four vars
+   + cbind(AMZN.var.full,AMZN.var.2011,
+           + AMZN.var.2012,AMZN.var.2013),
+   ### Use cbind to combine four sds
+   + cbind(AMZN.sd.full,AMZN.sd.2011,
+           + AMZN.sd.2012,AMZN.sd.2013),
+   ### Use cbind to combine four means
+   + cbind(mean.ret.full,mean.ret.2011,
+           + mean.ret.2012,mean.ret.2013))
> ### Output
> AMZN.risk
            Return       Return       Return       Return
Return 0.000425095 0.0005852692 0.0004015231 0.0002897267
       0.020617832 0.0241923377 0.0200380423 0.0170213587
       0.001266679 0.0001385631 0.0016802471 0.0019845099
>
```

Step 8: Cleanup Data

```
> ### At least 3 digits after first numeric "0" digit
> options(digits=3)
> ### Rename rows
> rownames(AMZN.risk)<-c("Variance","Std Dev","Mean")
> ### Rename columns
> colnames(AMZN.risk)<-c("2011-2013","2011","2012","2013")
> ### Output
> AMZN.risk
         2011-2013     2011     2012    2013
Variance  0.000425 0.000585 0.000402 0.00029
Std Dev   0.020618 0.024192 0.020038 0.01702
Mean      0.001267 0.000139 0.001680 0.00198
> ### Change to be default of digits
> options(digits=7)
>
```

Based on daily returns, we can see that for the 2011–2013 period, AMZN's average return is 0.13 % but the standard deviation around this mean is 2.07 %. Assuming a normal distribution, 68 % of AMZN returns lie within one standard deviation from the mean and 95 % of the AMZN returns lie within two standard deviations from the mean. Using 2011 to 2013 data, this means that 68 % of AMZN's daily returns are between −1.9 and 2.2 % and 95 % of AMZN's daily returns are between −4.0 and 4.3 %.

Standard deviations of returns based on different frequencies are not comparable. We have to convert such volatility measures to an annualized volatility number to make such a comparison.

```
> ### At least 3 digits after first numeric "0" digit
> options(digits=3)
> ### Re-assignment
> annual.vol<-AMZN.risk
> ### annual variance = 252 * daily variance
> annual.vol[1,]<-annual.vol[1,]*252
> ### annual standard deviation = sqrt(252) * daily deviation
> annual.vol[2,]<-annual.vol[2,]*sqrt(252)
> ### annual mean = 252 * daily mean
> annual.vol[3,]<-annual.vol[3,]*252
> ### Output
> annual.vol
         2011-2013   2011  2012  2013
Variance     0.107 0.1475 0.101 0.073
Std Dev      0.327 0.3840 0.318 0.270
Mean         0.319 0.0349 0.423 0.500
> ### Change to be default of digits
> options(digits=7)
>
```

As the output above shows, AMZN's annualized standard deviation for the full period was 32.7 %. Breaking down the data into three different years, we see that 2011 had the highest standard deviation at 38.4 % while 2013 had the lowest standard deviation at 27.0 %.

4.3 Portfolio Risk

In particular, when we add assets that are not perfectly correlated with the securities in our portfolio, the overall risk of the portfolio decreases.

In fact, Reilly and Brown [7] state that using 12–18 well-selected stocks can yield 90 % of the maximum benefits of diversification. Therefore, as a general rule, a portfolio's risk is not simply the weighted average of the standard deviations of each of the securities in the portfolio but likely something lower.

4.3.1 Two Assets (Manual Approach)

In the case of a two-asset portfolio, portfolio risk is calculated as

$$\sigma_p^2 = w_1^2 r_1^2 + w_2^2 r_2^2 + 2\sigma_{1,2} w_1 w_2$$

Where w_i is the weight of security i in the portfolio, r_i is the return of security i, and $\sigma_{1,2}$ is the covariance between the returns of securities 1 and 2.

Let us work through an example of calculating the portfolio risk for a $ 10,000 portfolio that invested $ 2500 in AMZN and $ 7500 in IBM.

Step 1: Calculate Weights of Securities in the Portfolio

```
> ### Set the original weight of two securities
> wgt.AMZN=.25
> wgt.IBM=.75
>
```

Step 2: Import AMZN and IBM Data from Yahoo Finance and Calculate Total Returns

```
> ### Import AMZN data
> ### Read the csv file, which contains the header
> data.AMZN<-read.csv("AMZN Yahooo.csv",header=TRUE)
> ### Change the order of the columns
> data.AMZN<-data.AMZN[,c(1,2,3,4,5,7,6)]
> ### Change the Date column to be a "Date" object with a specific format by creating a new variable
> date<-as.Date(data.AMZN$Date,format="%Y-%m-%d")
> ### combine the new variable with the remaining columns
> data.AMZN<-cbind(date, data.AMZN[,-1])
> ### order the data-set by the date column
> data.AMZN<-data.AMZN[order(data.AMZN$date),]
> ### Use the library of "xts"
> library(xts)
> ### keep the first column as order and remaining column as content
> data.AMZN<-xts(data.AMZN[,2:7],order.by=data.AMZN[,1])
> ### Rename meaningfully
> names(data.AMZN)<-paste(c("AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"))
> ### Output
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
>
> ### Import IBM data
> ### Read the csv file, which contains the header
> data.IBM<-read.csv("IBM Yahooo.csv",header=TRUE)
> ### Change the order of the columns
> data.IBM<-data.IBM[,c(1,2,3,4,5,7,6)]
> ### Change the Date column to be a "Date" object with a specific format by creating a new variable
> date<-as.Date(data.IBM$Date,format="%Y-%m-%d")
> ### combine the new variable with the remaining columns
> data.IBM<-cbind(date, data.IBM[,-1])
> ### order the data-set by the date column
> data.IBM<-data.IBM[order(data.IBM$date),]
> ### Use the library of "xts"
> library(xts)
> ### keep the first column as order and remaining column as content
> data.IBM<-xts(data.IBM[,2:7],order.by=data.IBM[,1])
> ### Rename meaningfully
> names(data.IBM)<-paste(c("IBM.Open","IBM.High","IBM.Low","IBM.Close","IBM.Volume","IBM.Adjusted"))
> ### Output
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
>
> ### Calculate return by delta of adjusted prices for IBM
> IBM.Ret<-Delt(data.IBM$IBM.Adjusted)
> ### Output
> IBM.Ret[c(1:3,nrow(IBM.Ret)),]
           Delt.1.arithmetic
2010-12-31                NA
2011-01-03       0.004905867
2011-01-04       0.001084909
2013-12-31       0.006223193
>
> ### Calculate return by delta of adjusted prices for AMZN
> AMZN.Ret<-Delt(data.AMZN$AMZN.Adjusted)
> ### Output
> AMZN.Ret[c(1:3,nrow(AMZN.Ret)),]
           Delt.1.arithmetic
2010-12-31                NA
2011-01-03       0.023444444
2011-01-04       0.004288351
2013-12-31       0.013778376
>
```

Step 3: Combine the Two Return Series

```
> ### Use cbind command to combine two return variables together
> returns<-cbind(AMZN.Ret,IBM.Ret)
> ### Output
> returns[c(1:3,nrow(returns)),]
           Delt.1.arithmetic Delt.1.arithmetic.1
2010-12-31                NA                  NA
2011-01-03       0.023444444         0.004905867
2011-01-04       0.004288351         0.001084909
2013-12-31       0.013778376         0.006223193
>
> ### Rename meaningfully
> names(returns)<-paste(c("AMZN.Ret","IBM.Ret"))
> ### Output
> returns[c(1:3,nrow(returns)),]
             AMZN.Ret     IBM.Ret
2010-12-31          NA          NA
2011-01-03 0.023444444 0.004905867
2011-01-04 0.004288351 0.001084909
2013-12-31 0.013778376 0.006223193
>
> ### Eliminate first row for placeholder
> returns<-returns[-1,]
> ### Output
> returns[c(1:3,nrow(returns)),]
              AMZN.Ret      IBM.Ret
2011-01-03 0.023444444  0.004905867
2011-01-04 0.004288351  0.001084909
2011-01-05 0.013026323 -0.003995657
2013-12-31 0.013778376  0.006223193
>
```

Step 4: Calculate Standard Deviation and Covariance of the Securities

```
> ### Calculate standard deviation and covariance for AMZN and IBM annually
> ### And then output
> sd.AMZN<-sd(returns$AMZN.Ret)*sqrt(252)
> sd.AMZN
[1] 0.3272979
> sd.IBM<-sd(returns$IBM.Ret)*sqrt(252)
> sd.IBM
[1] 0.1923476
> ret.cov<-cov(returns$AMZN.Ret,returns$IBM.Ret)*252
> ret.cov
            IBM.Ret
AMZN.Ret 0.02326951
>
```

Step 5: Calculate Portfolio Risk

```
> ### Calculate the variance of the portfolio by using the formula
> port.var<-wgt.AMZN**2*sd.AMZN**2 + wgt.IBM**2*sd.IBM**2 + 2*ret.cov*wgt.AMZN*wgt.IBM
> ### Output
> port.var
            IBM.Ret
AMZN.Ret 0.03623245
> ### Use sqrt() command to calculate standard deviation of the portfolio
> port.sd<-sqrt(port.var)
> ### Output
> port.sd
           IBM.Ret
AMZN.Ret 0.1903482
>
```

Benefit of Diversification

Note that this portfolio standard deviation is lower than the weighted-average standard deviation of AMZN and IBM returns of 22.6 %. Since AMZN and IBM returns only have a 0.37 correlation, portfolio risk is reduced by combining the two securities in a portfolio. In fact, the smaller the correlation, the more gains from diversification we can achieve.

4.3.2 Two Assets (Matrix Algebra)

The formula for portfolio variance σ_P^2 in matrix form is

$$\sigma_P^2 = w\Sigma w^T, \tag{4.3}$$

where w is a vector of weights and Σ is the Variance–Covariance matrix of the securities in the portfolio. Although different texts may use different symbols, w and Σ are symbols typically used to refer to these variables in the finance literature. One can think of a *vector* as a column or row or numbers. A *matrix* is a set of vectors stacked either on top of each other or side-by-side. The T superscript denotes the transposition of a matrix, which in our cases flips a row vector of weights into a column vector of weights.

Step 1: Create Vector of Weights

```
> ### Make a series of weights
> WGT.2asset<-c(0.25,0.75)
> ### Output
> WGT.2asset
[1] 0.25 0.75
> ### Make a row matrix of the series by 1x2
> WGT.2asset<-matrix(WGT.2asset,1)
> ### Output
> WGT.2asset
     [,1] [,2]
[1,] 0.25 0.75
>
```

Step 2: Create Transposed Vector of Weights

```
> ### Make the transpose of the row matrix above
> tWGT.2asset<-t(WGT.2asset)
> ### Output
> tWGT.2asset
     [,1]
[1,] 0.25
[2,] 0.75
>
```

Step 3: Construct Variance–Covariance Matrix

```
> ### Convert returns data-set to be matrix object
> mat.Ret<-as.matrix(returns)
> ### Output
> head(mat.Ret)
               AMZN.Ret      IBM.Ret
2011-01-03  0.023444444  0.004905867
2011-01-04  0.004288351  0.001084909
2011-01-05  0.013026323 -0.003995657
2011-01-06 -0.008323551  0.010948246
2011-01-07 -0.001990746 -0.004910654
2011-01-10 -0.004366812 -0.001960412
> 
> ### Prevent scientific notations
> options(scipen="100")
> ### Output the covariance matrix
> cov(mat.Ret)
              AMZN.Ret       IBM.Ret
AMZN.Ret 0.00042509501 0.00009233932
IBM.Ret  0.00009233932 0.00014681580
> 
> ### multiply 252 to the VCOV matrix to annualize the matrix
> VCOV.2asset<-cov(mat.Ret)*252
> ### Output
> VCOV.2asset
           AMZN.Ret    IBM.Ret
AMZN.Ret 0.10712394 0.02326951
IBM.Ret  0.02326951 0.03699758
> 
```

Step 4: Calculate Portfolio Risk

```
> ### Make a sequence of multiplications together by the formula to calculate variances of the portfolio
> mat.var2asset<-WGT.2asset %*% VCOV.2asset %*% tWGT.2asset
> ### Output
> mat.var2asset
          [,1]
[1,] 0.03623245
> 
> ### Use sqrt command to convert variance to be standard deviation
> mat.sd2asset<-sqrt(mat.var2asset)
> ### Output
> mat.sd2asset
          [,1]
[1,] 0.1903482
> 
```

4.3.3 Multiple Assets

Step 1: Import Data for AMZN, IBM, AAPL, and NVDA

```
> ### Check if AMZN imported
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
> ### Check if IBM imported
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
>
> # Import AAPL
> data.AAPL<-read.csv("AAPL Yahooo.csv",header=TRUE)
> data.AAPL<-data.AAPL[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.AAPL$Date,format="%Y-%m-%d")
> data.AAPL<-cbind(date, data.AAPL[,-1])
> data.AAPL<-data.AAPL[order(data.AAPL$date),]
> data.AAPL<-xts(data.AAPL[,2:7],order.by=data.AAPL[,1])
> names(data.AAPL)<-paste(c("AAPL.Open","AAPL.High","AAPL.Low","AAPL.Close","AAPL.Volume","AAPL.Adjusted"))
> data.AAPL[c(1:3,nrow(data.AAPL)),]
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2010-12-31  11.53393  11.55286 11.47536   11.52000   193508000      9.739614
2011-01-03  11.63000  11.79500 11.60143   11.77036   445138400      9.951281
2011-01-04  11.87286  11.87500 11.71964   11.83179   309080800     10.003214
2013-12-31  19.79179  20.04571 19.78571   20.03643   223084400     17.519623
>
> # Import NVDA
> data.NVDA<-read.csv("NVDA Yahooo.csv",header=TRUE)
> data.NVDA<-data.NVDA[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.NVDA$Date,format="%Y-%m-%d")
> data.NVDA<-cbind(date, data.NVDA[,-1])
> data.NVDA<-data.NVDA[order(data.NVDA$date),]
> data.NVDA<-xts(data.NVDA[,2:7],order.by=data.NVDA[,1])
> names(data.NVDA)<-paste(c("NVDA.Open","NVDA.High","NVDA.Low","NVDA.Close","NVDA.Volume","NVDA.Adjusted"))
> data.NVDA[c(1:3,nrow(data.NVDA)),]
           NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume NVDA.Adjusted
2010-12-31    3.7500    3.8550    3.745     3.8500    39125200      3.531613
2011-01-03    3.8800    3.9925    3.875     3.9550    81744800      3.627930
2011-01-04    3.9625    3.9800    3.855     3.9425    65138400      3.616464
2013-12-31    4.0000    4.0250    3.975     4.0050    23577600      3.778282
>
```

Step 2: Extract Adjusted Prices of Each Security

```
> ### Extract the last column of each security by the command of merge
> multi<-data.AMZN[,6]
> multi<-merge(multi,data.NVDA[,6])
> multi<-merge(multi,data.IBM[,6])
> multi<-merge(multi,data.AAPL[,6])
> ### Output
> multi[c(1:3,nrow(multi)),]
           AMZN.Adjusted NVDA.Adjusted IBM.Adjusted AAPL.Adjusted
2010-12-31        9.0000      3.531613     85.83682      9.739614
2011-01-03        9.2110      3.627930     86.25792      9.951281
2011-01-04        9.2505      3.616464     86.35150     10.003214
2013-12-31       19.9395      3.778282    115.64046     17.519623
>
```

Step 3: Calculate Returns for Each Security

```
> ### Convert multi to be a matrix with nrow(multi) rows
> mat.price<-matrix(multi,nrow(multi))
> ### Define the function of Delt
> prc2ret<-function(x) Delt(x)
> ### Use apply command to apply delta
> mat.ret<-apply(mat.price,2,function(x){prc2ret(c(x))})
> ### Output
> mat.ret[1:4,]
            [,1]        [,2]         [,3]        [,4]
[1,]          NA          NA           NA          NA
[2,] 0.023444444  0.02727281  0.004905867 0.021732586
[3,] 0.004288351 -0.00316048  0.001084909 0.005218725
[4,] 0.013026323  0.07672826 -0.003995657 0.008180371
>
```

Step 4: Clean up Returns Data

```
> ### Delete first row for NAs
> mat.ret<-mat.ret[-1,]
> ### Output
> mat.ret[1:4,]
             [,1]        [,2]         [,3]          [,4]
[1,]  0.023444444  0.02727281  0.004905867  0.0217325861
[2,]  0.004288351 -0.00316048  0.001084909  0.0052187251
[3,]  0.013026323  0.07672826 -0.003995657  0.0081803708
[4,] -0.008323551  0.13839729  0.010948246 -0.0008086231
> ### Rename the columns meaningfully
> colnames(mat.ret)<-c("AMZN","NVDA","IBM","AAPL")
> ### Output
> mat.ret[1:4,]
             AMZN        NVDA          IBM          AAPL
[1,]  0.023444444  0.02727281  0.004905867  0.0217325861
[2,]  0.004288351 -0.00316048  0.001084909  0.0052187251
[3,]  0.013026323  0.07672826 -0.003995657  0.0081803708
[4,] -0.008323551  0.13839729  0.010948246 -0.0008086231
>
```

Step 5: Calculate Annualized Variance–Covariance Matrix

```
> ### Calculate the covariance of the matrix of 4 returns
> VCOV<-cov(mat.ret)
> ### Output
> VCOV
               AMZN          NVDA           IBM          AAPL
AMZN 0.00042509501 0.0001728474 0.00009233932 0.00010234050
NVDA 0.00017284735 0.0006381585 0.00012508491 0.00014804654
IBM  0.00009233932 0.0001250849 0.00014681580 0.00007456329
AAPL 0.00010234050 0.0001480465 0.00007456329 0.00031312957
>
> ### Annualization the covariance matrix
> VCOV.annual<-252 * VCOV
> ### Output
> VCOV.annual
          AMZN       NVDA        IBM       AAPL
AMZN 0.10712394 0.04355753 0.02326951 0.02578981
NVDA 0.04355753 0.16081595 0.03152140 0.03730773
IBM  0.02326951 0.03152140 0.03699758 0.01878995
AAPL 0.02578981 0.03730773 0.01878995 0.07890865
>
```

Step 6: Create a Row vector of Weights

```
> ### Create a series of data of the weight
> wgt=c(.2,.2,.3,.3)
> ### Convert it into a row matrix
> mat.wgt<-matrix(wgt,1)
> ### Output
> mat.wgt
     [,1] [,2] [,3] [,4]
[1,]  0.2  0.2  0.3  0.3
>
```

Step 7: Create a Column Vector of Weights by Transposing the Row Vector of Weights

```
> ### Use t command to calculate transpose of matrix
> tmat.wgt<-t(mat.wgt)
> ### Output
> tmat.wgt
     [,1]
[1,]  0.2
[2,]  0.2
[3,]  0.3
[4,]  0.3
>
```

Step 8: Calculate the Portfolio Variance

```
> ### Use formula to calculate variance of portfolio
> port.var<-mat.wgt %*% VCOV.annual %*% tmat.wgt
> ### Output
> port.var
             [,1]
[1,] 0.04216256
> port.var[1,1]
[1] 0.04216256
>
```

Step 9: Calculate the Portfolio Standard Deviation

```
> ### Use sd command to calculate standard deviation
> port.sd<-sqrt(port.var)
> ### [1,1] in output because value is outputed in 1x1 matrix
> port.sd[1,1]
[1] 0.2053352
>
```

4.4 Value-at-Risk

A popular measure of the risk of loss in a portfolio is Value-at-Risk or VaR.

VaR measures the loss in our portfolio over a pre-specified time horizon, assuming some level of probability.、

For example, we will calculate as an example the 1 and 5 % 1- Day VaR on December 31, 2013 of the Value-Weighted portfolio we constructed in Chapter 3, which was based on a portfolio of AMZN, IBM, and NVDA

In this section, we will use the market convention of representing VaR as a positive number and using the significance level (e.g., 1 or 5 %) instead of confidence level (e.g., 99 or 95 %).

We also implement two types of VaR calculations: Gaussian VaR and Historical VaR.

The former assumes that the data follow a normal or Gaussian distribution, while the latter uses the distributional properties of the actual data. As such, Historical VaR requires more data to implement and Gaussian VaR requires less data to implement.

VaR has been criticized for not being able to fully capture how large the loss we should expect if the loss exceeds VaR or what is known as tail risk.

If the tail of the distribution is large, then there is a risk of a very large loss.

ES tells us how much we should expect to lose if our losses exceed VaR.

In this section, we calculate the VaR using both the Gaussian VaR/ES and Historical VaR/ES based on the Value-Weighted Portfolio we constructed in Chap. 3, which consists of AMZN, IBM, and NVDA. We will calculate the 1-Day VaR based on a 1 and 5 % significance level.

4.4.1 Gaussian VaR

One of the simplest approaches to estimate VaR is to assume that the portfolio returns follow a normal distribution. Hence, this approach is called Gaussian VaR.

Step 1: Import Daily Portfolio Returns for the Last Year

```
> ### Import the csv file of hypothetical portfolio data of daily returns
> port.ret<-read.csv("Hypothetical Portfolio (Daily).csv")
> ### Output raw data
> port.ret[c(1:3,nrow(port.ret)),]
      X       date   VW.cum   EW.cum       VW.ret         EW.ret
1     1 2012-12-31 1.000000 1.000000           NA             NA
2     2 2013-01-02 1.030723 1.029417 0.0307232679  0.02941675279
3     3 2013-01-03 1.030903 1.029364 0.0001744982 -0.00005139738
253 253 2013-12-31 1.317664 1.299107 0.0075709348  0.00813547989
> ### We only need value-weighted return here and then delete first row NA
> port.ret<-port.ret$VW.ret[-1]
> ### See first five values
> port.ret[1:5]
[1]  0.0307232679  0.0001744982  0.0136264507 -0.0039524123 -0.0123305334
>
```

Step 2: Calculate Mean and Standard Deviation of Historical Daily Portfolio Returns

In practice, some assume that because we are calculating VaR over a short horizon, the portfolio mean return is zero. In our example, we do not make this assumption but we can see from the output below that our calculated portfolio mean is small.

```
> ### Calculate average return of the portfolio
> port.mean<-mean(port.ret)
> ### Output
> port.mean
[1] 0.001147063
> ### Calculate standard deviation of the portfolio
> port.risk<-sd(port.ret)
> ### Output
> port.risk
[1] 0.01020148
>
```

Step 3: Calculate 1 and 5 % VaR

Step 3: Calculate 1 and 5 % VaR The VaR is calculated as

$$VaR_\alpha = -(\mu - \sigma * Z_\alpha) * I, \tag{4.4}$$

where α is the significance level of the VaR, μ is the portfolio average return, σ is the standard deviation of the portfolio returns, Z_α is the z-score based on the VaR significance level, and I is the current portfolio value. The z-score is calculated using the `qnorm` command, which returns the inverse cumulative density function. The

And we know that $z_{0.01}$ = 2.33

```
> ### See what qnorm look like
> qnorm(0.01)
[1] -2.326348
> ### Significant level = 0.01
> ### calculate by formula
> VaR01.Gaussian<- -(port.mean+port.risk*qnorm(0.01))*1317664
> ### Convert to the form containing ","
> VaR01.Gaussian<-format(VaR01.Gaussian,big.mark=",")
> ### Output
> VaR01.Gaussian
[1] "29,759.62"
>
> ### Significant level = 0.05
> ### calculate by formula
> VaR05.Gaussian<- -(port.mean+port.risk*qnorm(0.05))*1317664
> ### Convert to the form containing ","
> VaR05.Gaussian<-format(VaR05.Gaussian,big.mark=",")
> ### Output
> VaR05.Gaussian
[1] "20,598.89"
>
```

This means that there is a 1 % (5 %) chance that our portfolio loses more than $ 29,760 ($ 20,599) over the next day. Note that to make the VaR result easier to read, we used the format command to format the VaR output to include a comma.

4.4.2 Historical VaR

Historical VaR uses a mix of current weights in the portfolio and a simulation of historical returns of the securities in the portfolio to construct a simulated series of portfolio Profits & Losses (P&L).

Given its use of historical returns data, Historical VaR works better when we have lots of data. Typically, three to five years of data is recommended for this approach.

Step 1: Import Three Years of Daily Returns Data for Each Security in the Portfolio

```
> ### Output the AMZN data
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000         9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000         9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000         9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000        19.9395
> ### Use delta function to calculate the return
> AMZN.Ret<-Delt(data.AMZN$AMZN.Adjusted)
> ### Output the return data-set
> AMZN.Ret[c(1:3,nrow(AMZN.Ret)),]
           Delt.1.arithmetic
2010-12-31                NA
2011-01-03       0.023444444
2011-01-04       0.004288351
2013-12-31       0.013778376
> ### Output the NVDA data
> data.NVDA[c(1:3,nrow(data.NVDA)),]
           NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume NVDA.Adjusted
2010-12-31    3.7500    3.8550    3.745     3.8500    39125200      3.531613
2011-01-03    3.8800    3.9925    3.875     3.9550    81744800      3.627930
2011-01-04    3.9625    3.9800    3.855     3.9425    65138400      3.616464
2013-12-31    4.0000    4.0250    3.975     4.0050    23577600      3.778282
> ### Use delta function to calculate the return
> NVDA.Ret<-Delt(data.NVDA$NVDA.Adjusted)
> ### Output the return data-set
> NVDA.Ret[c(1:3,nrow(NVDA.Ret)),]
           Delt.1.arithmetic
2010-12-31                NA
2011-01-03        0.02727281
2011-01-04       -0.00316048
2013-12-31        0.00313050
> ### Output the IBM data
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
> ### Use delta function to calculate the return
> IBM.Ret<-Delt(data.IBM$IBM.Adjusted)
> ### Output the return data-set
> IBM.Ret[c(1:3,nrow(IBM.Ret)),]
           Delt.1.arithmetic
2010-12-31                NA
2011-01-03       0.004905867
2011-01-04       0.001084909
2013-12-31       0.006223193
>
```

Step 2: Combine Returns Data into One Data Object

```
> ### combine the three returns and delete first rows of them
> ret.data<-cbind(AMZN.Ret[-1,],NVDA.Ret[-1,],IBM.Ret[-1,])
> ### Output
> ret.data[c(1:3,nrow(ret.data)),]
           Delt.1.arithmetic Delt.1.arithmetic.1 Delt.1.arithmetic.2
2011-01-03       0.023444444          0.02727281         0.004905867
2011-01-04       0.004288351         -0.00316048         0.001084909
2011-01-05       0.013026323          0.07672826        -0.003995657
2013-12-31       0.013778376          0.00313050         0.006223193
> ### Rename them meaningfully
> names(ret.data)<-paste(c("AMZN.Ret","NVDA.Ret","IBM.Ret"))
> ### Output
> ret.data[c(1:3,nrow(ret.data)),]
              AMZN.Ret    NVDA.Ret      IBM.Ret
2011-01-03 0.023444444  0.02727281  0.004905867
2011-01-04 0.004288351 -0.00316048  0.001084909
2011-01-05 0.013026323  0.07672826 -0.003995657
2013-12-31 0.013778376  0.00313050  0.006223193
>
```

Step 3: Identify the Value of Each Security in the Portfolio as of December 31, 2013

```
> ### Pre-set the index of ending value of three securities
> last.idx<-c(0.4576889,0.5346666,0.3253085)*1000000
> ### Output
> last.idx
[1] 457688.9 534666.6 325308.5
> ### Portfolio ending value
> port.val<-sum(last.idx)
> ### Output
> port.val
[1] 1317664
>
```

Step 4: Calculate Simulated Portfolio Returns Applying Current Portfolio Weights to Historical Security Returns

three years of historical returns data of each security in the portfolio. We assume that the current value of each security in the portfolio remains frozen over our VaR time horizon (e.g., one day). Therefore, each return observation is applied to the current value of the security. That is,

$$P\&L = V_{AMZN}R^t_{AMZN} + V_{YHOO} * R^t_{YHOO} + V_{IBM} * R^t_{IBM}, \qquad (4.5)$$

where V denotes the value of the security in the portfolio and R is the return of the security on day t. The variable name generated by this step is `AMZN.Ret`, which is misleading, so we rename this variable to `Port.PnL`, which is more fitting.

Note that in the formula YHOO is replaced by NVDA since I have mentioned that YHOO has already existed the market and no longer able to show the historical data.

```
> ### Use volume multiply return for all three securities and take sum for a specific date
> sim.portPnL<-last.idx[1]*ret.data$AMZN.Ret + last.idx[2]*ret.data$NVDA.Ret + last.idx[3]*ret.data$IBM.Ret
> ### Output
> sim.portPnL[c(1:3,nrow(sim.portPnL)),]
             AMZN.Ret
2011-01-03 26908.0413
2011-01-04   625.8577
2011-01-05 45686.2215
2013-12-31 10004.4409
> ### Rename with Profit and Loss
> names(sim.portPnL)<-paste("Port.PnL")
> ### Output
> sim.portPnL[c(1:3,nrow(sim.portPnL)),]
             Port.PnL
2011-01-03 26908.0413
2011-01-04   625.8577
2011-01-05 45686.2215
2013-12-31 10004.4409
>
```

Step 5: Calculate Appropriate Quantile for the 1 and 5 % VaR

```
> ### Use quantile to calculate VaR
> ### The first argument is the negative of the simulated portfolio P&L and the second argument is the 1 - α
> ### confidence level.
> VaR01.Historical=quantile(-sim.portPnL$Port.PnL,0.99)
> ### Format to read easier
> VaR01.Historical<-format(VaR01.Historical,big.mark=",")
> ### Output
> VaR01.Historical
       99%
"51,007.56"
> ### Use quantile to calculate VaR
> ### The first argument is the negative of the simulated portfolio P&L and the second argument is the 1 - α
> ### confidence level.
> VaR05.Historical=quantile(-sim.portPnL$Port.PnL,0.95)
> ### Format to read easier
> VaR05.Historical<-format(VaR05.Historical,big.mark=",")
> ### Output
> VaR05.Historical
       95%
"31,611.49"
>
```

Note that the higher the significance level (i.e., 5 vs. 1 %) or the lower the confidence level (i.e., 95 vs. 99 %), the lower the VaR amount. To see why, suppose we were asked to give our best estimate of a number that we are sure we would not exceed. The higher the estimate of this number we come up with, the higher the likelihood that we will not exceed it. As such, we should expect to have a higher number at the 99 % confidence level than at the 95 % confidence level.

In my own words, when the estimate is as much high as the maximized value we can exceed, then the probability is 100%, which is just the bound of the estimation.

Step 6: Plot the VaR in Relation to P&L Density

In this step, we plot the following three contents:

i: the density of the simulated portfolio P&L.

ii: the normal distribution of P&L's based on the mean and standard deviation of the

simulated portfolio P&Ls.

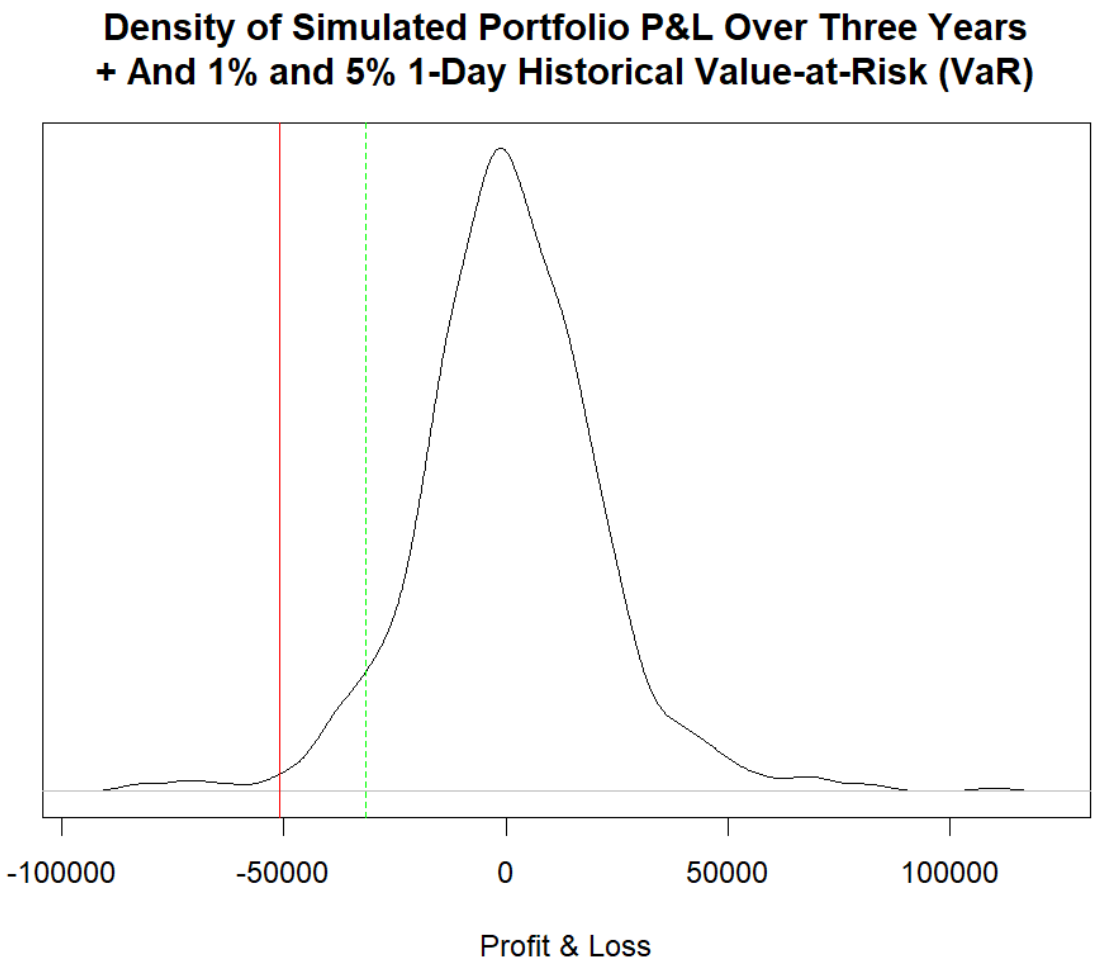
iii: our estimates of the 1 and 5 % 1-Day Historical VaR.

We first plot (i) and (iii), then we add (ii).

```
> ### Use density function to get the details of density of PnL
> ret.d=density(sim.portPnL$Port.PnL)
> ### Output
> ret.d

Call:
        density.default(x = sim.portPnL$Port.PnL)

Data: sim.portPnL$Port.PnL (754 obs.);  Bandwidth 'bw' = 4276

       x                 y
 Min.   :-95587   Min.   :0.00000000141
 1st Qu.:-40992   1st Qu.:0.00000024885
 Median : 13603   Median :0.00000058112
 Mean   : 13603   Mean   :0.00000457023
 3rd Qu.: 68198   3rd Qu.:0.00000578339
 Max.   :122793   Max.   :0.00002295483
>
```

We can now plot (i) and (iii). Note that we add two vertical lines using the abline command. The two lines denote our estimates of VaR, so we can visually see from the density of the simulated portfolio P&L, where the Historical VaR estimates are.

```
> ### Plot the density
> plot(ret.d,xlab="Profit & Loss",ylab="",yaxt="n",main="Density of Simulated Portfolio P&L Over Three Years
+ + And 1% and 5% 1-Day Historical Value-at-Risk (VaR)")
> ### Line of estimate of 1%
> abline(v=-quantile(-sim.portPnL$Port.PnL,0.99),col="red",lty=1)
> ### line of estimate of 5%
> abline(v=-quantile(-sim.portPnL$Port.PnL,0.95),col="green",lty=2)
>
```



Next, we now turn to number (ii) on the list, which is the plot of the normal distribution based on the mean and standard deviation of the simulated portfolio P&L. First, we create a sequence of 1000 numbers between the smallest and largest simulated P&L. This creates bounds for the normal density plot that follows.

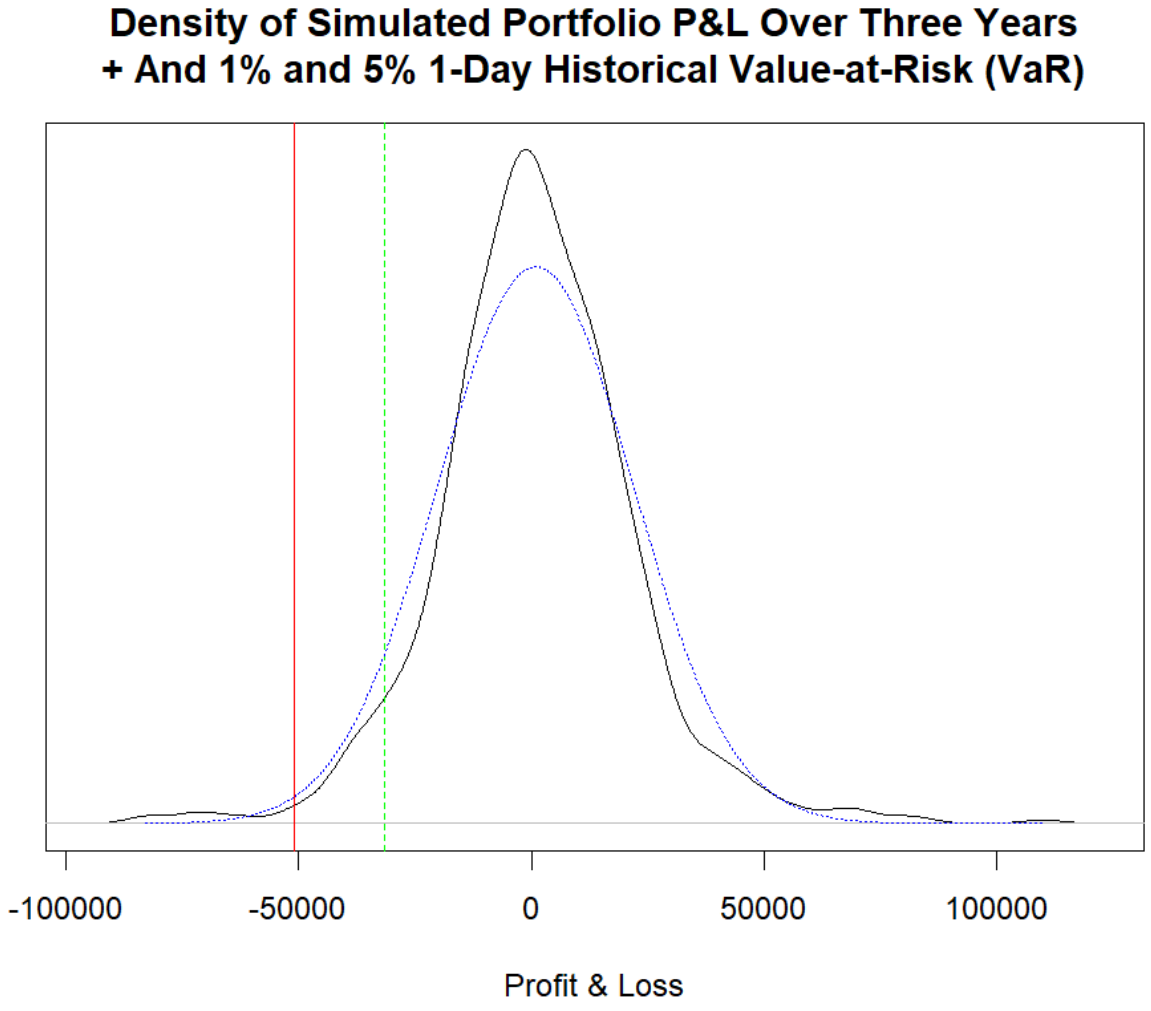
```
> ### Create a sequence x from minimum to maximum of density with 1000 elements
> x<-seq(min(sim.portPnL$Port.PnL),max(sim.portPnL$Port.PnL),length=1000)
> ### See head of x
> head(x)
[1] -82759.26 -82566.34 -82373.43 -82180.51 -81987.59 -81794.67
> ### See tail of x
> tail(x)
[1] 109000.6 109193.5 109386.5 109579.4 109772.3 109965.2
>
```

Then, we use the dnorm command to provide the values of the probability density function for the normal distribution.

```
> ### Use dnorm command to assign values to the normal distribution
> ### three arguments: 1. x series of point intervals 2. mean. 3. standard deviation.
> y<-dnorm(x,mean=mean(sim.portPnL$Port.PnL),sd=sd(sim.portPnL$Port.PnL))
> ### See head of y density
> head(y)
[1] 0.000000006922707 0.000000007179640 0.000000007445483 0.000000007720521 0.000000008005045 0.000000008299357
> ### See tail of y density
> tail(y)
[1] 0.00000000003548808 0.00000000003385410 0.00000000003229263 0.00000000003080060 0.00000000002937503
[6] 0.00000000002801308
>
```

We then use the lines command to add the normal density plot as a dashed bell curve onto the plot.

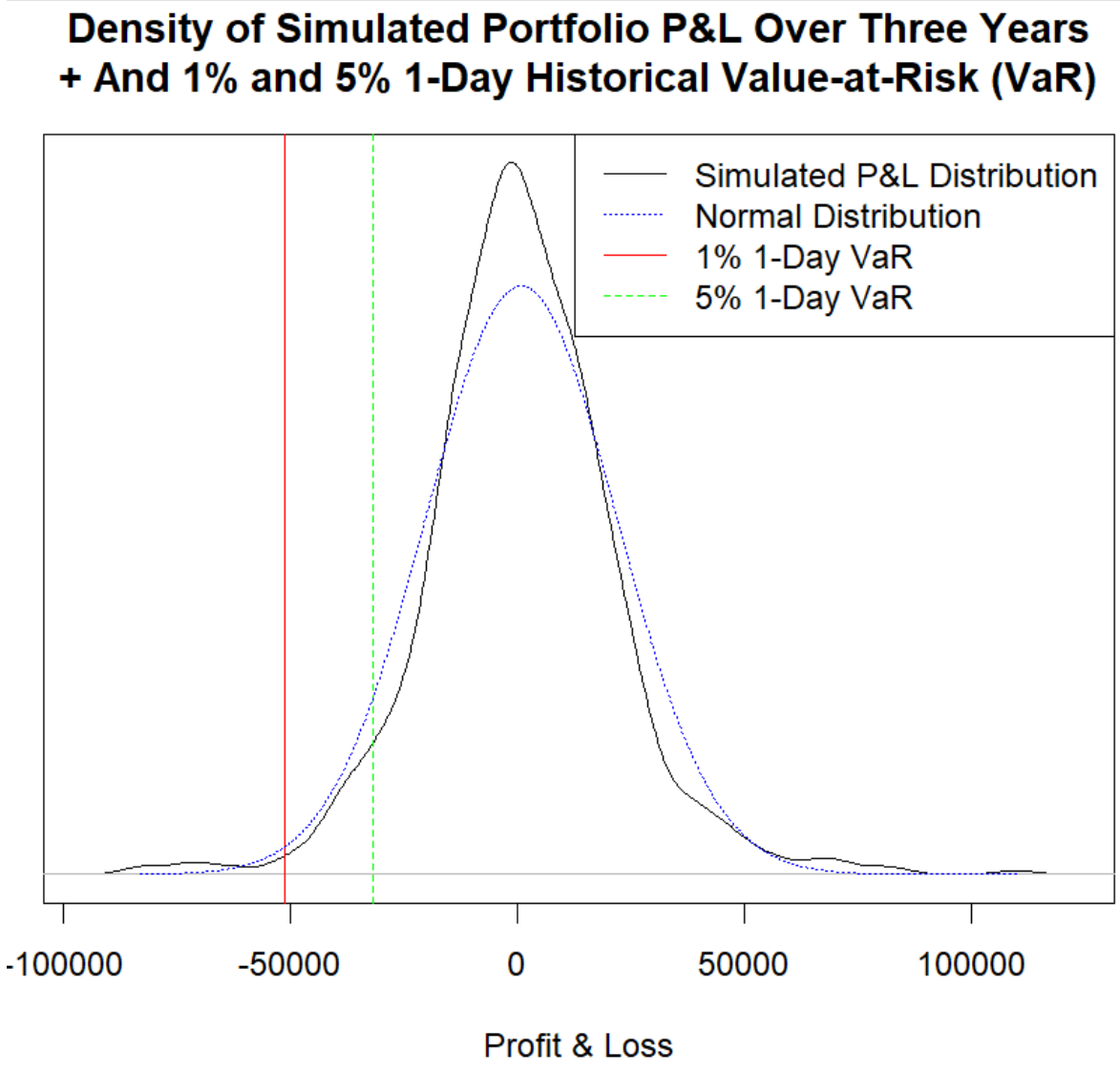
```
> ### Add line of normal distribution by mean and standard deviation
> lines(x,y,type="l",col="blue",lwd=1,lty=3)
>
```



Finally, we add a legend on the top-right portion of the chart, so the legend does not obstruct

the other elements of the chart.

```
> ### Add the legend
> legend("topright",c("Simulated P&L Distribution","Normal Distribution","1% 1-Day VaR","5% 1-Day VaR"),col=c("black","blue","red","green"),lty=c(1,3,1,2))
>
```



The chart shows that the simulated P&L distribution is more peaked than what a normal distribution with the same mean and standard deviation would show. This means that there are more observations that are around the mean or average P&L of $ 1127 in the simulated P&L.

Using the Square Root of T rule to Scale 1-Day VaR to a T -Day VaR

```
> ### If transfer from 1-day to 10-day, multiply sqrt(10) by 1-day quantile
> VaR01.10day<-quantile(-sim.portPnL$Port.PnL,0.99)*sqrt(10)
> ### Output
> VaR01.10day
     99%
161300.1
>
```

4.5 Expected Shortfall

One of the issues with VaR is that it does not capture the shape of the tail of the distribution. That is, VaR does not allow us to answer the question, "if losses exceed VaR, how bad should we expect that loss to be?"

The answer to this is called expected shortfall (ES). ES is known by many other names, such as tail VaR and tail loss.

We show two methods of calculating ES: Gaussian ES and Historical ES.

Similar to the difference between Historical VaR and Gaussian VaR, the difference between Historical ES and Gaussian ES is the assumption of a specific distribution or a lack thereof.

4.5.1 Gaussian ES

Similar to VaR, we can also calculate ES by assuming a normal distribution. This is known as a Gaussian ES. The calculation of Gaussian ES is

$$\mu + \sigma * \mathrm{dnorm}(\mathrm{qnorm}(\alpha))/\alpha,$$

Where φ is the dnorm function (i.e., height of the normal probability density function) in R and Φ is the qnorm function (i.e., inverse normal cumulative density function) in R. μ is the mean portfolio return, σ is the standard deviation of the portfolio returns, and α is the significance level.

```
> ###   Gaussian Expected Shortfall
> ### alpha = .01
> ### Calculate by formula
> ES01.Gaussian<-1317664*(port.mean+port.risk*(dnorm(qnorm(.01))/.01))
> ### Format easier to read
> ES01.Gaussian<-format(ES01.Gaussian,big.mark=",")
> ### Output
> ES01.Gaussian
[1] "37,337.59"
> ### alpha = .05
> ### Calculate by formula
> ES05.Gaussian<-1317664*(port.mean+port.risk*(dnorm(qnorm(.05))/.05))
> ### Format easier to read
> ES05.Gaussian<-format(ES05.Gaussian,big.mark=",")
> ### Output
> ES05.Gaussian
[1] "29,238.69"
>
```

The result means that there is a 1 % (5 %) chance our losses exceed VaR, but when it does, we expect that, on average, we will lose $ 37,338 ($ 29,239).

4.5.2 Historical ES

The Historical ES is calculated by taking the average portfolio loss that falls short of the Historical VaR estimate.

Step 1: Identify Historical VaR Limit for Portfolio

```
> ### Calculate limit of VaR historical lost of alpha = 0.01
> VaR01.hist=-quantile(-sim.portPnL$Port.PnL,0.99)
> ### Output
> VaR01.hist
      99%
-51007.56
> ### Calculate limit of VaR historical lost of alpha = 0.05
> VaR05.hist=-quantile(-sim.portPnL$Port.PnL,0.95)
> ### Output
> VaR05.hist
      95%
-31611.49
>
```

Step 2: Identify Simulated Portfolio Losses in Excess of VaR

To calculate ES, we have to find the average of the losses exceeding a loss of $ 51,008 for

the 1 % 1-Day Historical ES and $ 31,611 for the 5 % 1-Day Historical ES.

As such, the next step would be for us to use the P&L of our simulated portfolio to identify those losses. To do this, we use dummy variables to indicate what returns fall short of the threshold for 1 and 5 % Historical ES.

```
### Extract the PnL from previous portfolio
ES.PnL<-sim.portPnL$Port.PnL
### Output
ES.PnL[c(1:3,nrow(ES.PnL)),]
### If PnL < VaR01, 1 ; else, 0
ES.PnL$dummy01<-ifelse(ES.PnL$Port.PnL<VaR01.hist,1,0)
### If PnL < VaR05, 1 ; else, 0
ES.PnL$dummy05<-ifelse(ES.PnL$Port.PnL<VaR05.hist,1,0)
### Output
ES.PnL[c(1:3,nrow(ES.PnL)),]
```

```
             Port.PnL
2011-01-03 26908.0413
2011-01-04   625.8577
2011-01-05 45686.2215
2013-12-31 10004.4409
             Port.PnL dummy01 dummy05
2011-01-03 26908.0413       0       0
2011-01-04   625.8577       0       0
2011-01-05 45686.2215       0       0
2013-12-31 10004.4409       0       0
>
```

Step 3: Extract Portfolio Losses in Excess of VaR

Now that we have dummy variables that identify the days on which the P&L falls short of the Historical VaR estimate, we then extract those observations into a data object for the 1 % Historical ES shortfall01 and 5 % Historical ES shortfall05.

```
> ### Create ES of .01 by subset of PnL < VaR01
> shortfall01<-subset(ES.PnL,ES.PnL$dummy01==1)
> ### See head
> head(shortfall01)
            Port.PnL dummy01 dummy05
2011-01-28 -52296.94       1       1
2011-02-22 -71107.95       1       1
2011-05-13 -70599.94       1       1
2011-08-04 -82759.26       1       1
2011-08-08 -73785.79       1       1
2011-08-10 -64243.38       1       1
> ### Create ES of .05 by subset of PnL < VaR05
> shortfall05<-subset(ES.PnL,ES.PnL$dummy05==1)
> ### See head
> head(shortfall05)
            Port.PnL dummy01 dummy05
2011-01-28 -52296.94       1       1
2011-02-22 -71107.95       1       1
2011-02-23 -38382.54       0       1
2011-03-01 -37837.86       0       1
2011-03-10 -49489.74       0       1
2011-05-13 -70599.94       1       1
>
```

Step 4: Compute Average of Losses in Excess of VaR

To get the ES, we take the average of the returns is each of these data objects. We find that the 1 % 1-Day Historical ES is equal to $ 69,718. This result means that we have a 1 % probability that the loss in our portfolio would exceed the VaR, but, when it does, we expect that, on average, we would lose $ 69,718. Similarly, the 5 % 1-Day Historical ES is equal to $ 45,258. This result means that we have a 5 % probability that the loss in our portfolio would exceed the VaR, but, when it does, we expect that, on average, we would lose $ 45,258.

```
> ### Mean of .01 ES
> avg.ES01<--mean(shortfall01$Port.PnL)
> ### Output
> avg.ES01
[1] 69717.87
> ### Format to read
> ES01.Historical<-format(avg.ES01,big.mark=",")
> ### Output
> ES01.Historical
[1] "69,717.87"
> ### Mean of .05 ES
> avg.ES05<--mean(shortfall05$Port.PnL)
> ### Output
> avg.ES05
[1] 45257.57
> ### Format to read
> ES05.Historical<-format(avg.ES05,big.mark=",")
> ### Output
> ES05.Historical
[1] "45,257.57"
>
```

4.5.3 Comparing VaR and ES

We expect that the ES values to be bigger losses than the VaR because, by definition, the ES is the mean or average loss when losses exceed VaR. This can easily be seen when we combine the results into one table.

We use a combination of the rbind and cbind commands to bring together all the resulting calculations of Historical and Gaussian VaR and ES. The intermediate results look quite messy with variable names and row labels that are not too meaningful.

```
> ### rbind to stack together, cbind to combine all 4 risk by 01 and 05
> VaR.ES.Combined<-data.frame(rbind(cbind(VaR01.Historical,ES01.Historical[1],VaR01.Gaussian,ES01.Gaussian[1]),cbind(VaR05.Historical,ES05.Historical[1],VaR05.Gaussian,ES05.Gaussian[1])
> ### Output
> VaR.ES.Combined
    VaR01.Historical       V2 VaR01.Gaussian       V4
99%        51,007.56 69,717.87      29,759.62 37,337.59
95%        31,611.49 45,257.57      20,598.89 29,238.69
>
```

As such, we rename the variable names to make them easy to understand and consistent. We also rename the row names to conform to the convention of using significance levels and, to avoid confusion, include the time horizon in our VaR and ES calculations.

```
> ### Rename the columns meaningfully
> names(VaR.ES.Combined)<-paste(c("VaR Historical","ES Historical","VaR Gaussian","ES Gaussian"))
> ### Rename row names meaningfully
> rownames(VaR.ES.Combined)<-paste(c("1% 1-Day","5% 1-Day"))
> VaR.ES.Combined
          VaR Historical ES Historical VaR Gaussian ES Gaussian
1% 1-Day       51,007.56     69,717.87    29,759.62   37,337.59
5% 1-Day       31,611.49     45,257.57    20,598.89   29,238.69
>
```

Looking at the output above, we can easily confirm that the ES is larger than the VaR, which is almost always the case.

4.6 Alternative Risk Measures

The advantage of using close-to-close volatility is that it only requires you to look at closing prices, but you have to use many observations to get a good estimate of volatility.

Using a large number of observations entails getting a long historical series. The earlier part of such long historical data may be less relevant to measure volatility today. The measures we will discuss are the Parkinson, Garmann-Klass, Rogers-Satchell-Yoon, and YangZhou. Parkinson uses high and low prices, while the remaining alternative volatility measures all use open, high, low, and close data.

Therefore, we may want to consider alternative measures of volatility that are more efficient

than close-to-close volatility by utilizing the open, high, and low prices of the day in addition to the closing price.

4.6.1 Parkinson

The Parkinson volatility measure uses the stock's **high** and **low** price of the day. Specifically,

$$\sigma_{\text{Parkinson}} = \sqrt{\frac{1}{4T \ln 2} \sum_{t=1}^{T} \ln\left(\frac{h_t}{l_t}\right)^2},$$

where T is the number of days in the sample period, h_t is the high price on day t, and l_T is the low price on day t.

We now demonstrate how to implement this in R. Let's apply this to the **Amazon.com** data we have been using as our example in this book.

Step 1: Import Amazon Data from Yahoo Finance

```
> ### Read the AMZN data
> data.AMZN<-read.csv("AMZN Yahooo.csv",header=TRUE)
> ### Change the order of columns
> data.AMZN<-data.AMZN[,c(1,2,3,4,5,7,6)]
> ### Create a "Date" object from the AMZN$Date
> date<-as.Date(data.AMZN$Date,format="%Y-%m-%d")
> ### Replace the data-set with date variable and data-set without first column
> data.AMZN<-cbind(date, data.AMZN[,-1])
> ### rows are ordered by created date
> data.AMZN<-data.AMZN[order(data.AMZN$date),]
> ### Open xts library
> library(xts)
> ### convert data-set to be xts object, order from 1st column and content with remaining
> data.AMZN<-xts(data.AMZN[,2:7],order.by=data.AMZN[,1])
> ### Rename meaningfully
> names(data.AMZN)<-paste(c("AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"))
> ### Output
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
>
```

Step 2: Keep the High and Low Price

```
> ### Keep only high and low price for Parkinson and delete first row
> parkinson<-data.AMZN[-1,2:3]
> ### Output
> parkinson[c(1:3,nrow(parkinson)),]
           AMZN.High AMZN.Low
2011-01-03    9.3000   9.0605
2011-01-04    9.3850   9.1890
2011-01-05    9.3725   9.2035
2013-12-31   19.9415  19.6900
>
```

Step 3: Calculate the Terms in the Parkinson Formula

We first calculate the term ln (h_t/l_t) and then take the square of that value.

```
> ### Take log of high/low
> parkinson$log.hi.low<-log(parkinson$AMZN.High/parkinson$AMZN.Low)
> ### Square
> parkinson$log.square<-(parkinson$log.hi.low)**2
> ### Output
> parkinson[c(1:3,nrow(parkinson)),]
           AMZN.High AMZN.Low log.hi.low   log.square
2011-01-03    9.3000   9.0605 0.02609009 0.0006806930
2011-01-04    9.3850   9.1890 0.02110555 0.0004454444
2011-01-05    9.3725   9.2035 0.01819602 0.0003310953
2013-12-31   19.9415  19.6900 0.01269204 0.0001610880
>
```

Step 4: Calculate the Sum of the Values Under the log.square Column

```
> ### Take the sum of log.square
> parkinson.sum<-sum(parkinson$log.square)
> ### Output
> parkinson.sum
[1] 0.5612609
>
```

Step 5: Calculate the Daily Parkinson Volatility Measure

```
> ### Use formula to calculate Parkinson daily risk
> parkinson.vol<-sqrt(1/(4*nrow(parkinson)*log(2))*parkinson.sum)
> ### Output
> parkinson.vol
[1] 0.01638528
>
```

Step 6: Calculate the Annualized Parkinson Volatility

Using the square root of time rule, we convert daily volatility into annual volatility by multiplying the daily volatility by the square root of 252. The output below shows that the annualized Parkinson volatility is 26.0 %.

```
> ### output by annualization of multiplying sqrt(252)
> annual.parkinson.vol<-parkinson.vol*sqrt(252)
> ### Output
> annual.parkinson.vol
[1] 0.2601083
>
```

4.6.2 Garman-Klass

The Garman-Klass volatility measure can be viewed as an extension of the Parkinson volatility measure that includes opening and closing prices. The Garman-Klass volatility is calculated as follows:

$$\sigma_{\text{Garman-Klass}} = \sqrt{\frac{1}{2T}\sum_{t=1}^{T}\ln\left(\frac{h_t}{l_t}\right)^2 - \frac{2\ln 2 - 1}{T}\ln\left(\frac{c_t}{o_t}\right)^2}, \qquad (4.8)$$

where o_t is the open price on day t and all other terms are defined in the same manner as in Eq. (4.7). We now demonstrate how to calculate the Garman-Klass volatility measure using our Amazon data.

Step 1: Import Amazon Open, High, Low, and Close Price Data

```
> ### Garman and Klass
> ### Delete first row and keep only open,hign,low,close
> garman.klass<-data.AMZN[-1,1:4]
> ### Output
> garman.klass[c(1:3,nrow(garman.klass)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close
2011-01-03    9.0685    9.3000   9.0605     9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505
2011-01-05    9.2050    9.3725   9.2035     9.3710
2013-12-31   19.7290   19.9415  19.6900    19.9395
>
```

Step 2: Calculate the First Term

We calculate the first term inside the square root in Eq. (4.8). Notice also that the summation term is equal to the parkinson.sum we calculated above. Using that same variable can help us save some time coding.

```
> ### First term in garman.klass in square root by importing parkinson.sum
> garman.klass.one<-(1/(2*nrow(garman.klass)))*parkinson.sum
> ### Output
> garman.klass.one
[1] 0.0003721889
>
```

Step 3: Calculate the Second Term

```
> ### Use the formula to calculate second term
> ### Note that Close and Open Price is a series of numbers, so sum function must be applied to turn to be a value
> garman.klass.two<-((2*log(2)-1)/nrow(garman.klass))*sum(log(garman.klass$AMZN.Close/garman.klass$AMZN.Open)**2)
> ### Output
> garman.klass.two
[1] 0.0001009639
>
```

Step 4: Calculate the Daily Garman-Klass Volatility

```
> ### Square the minus of two terms by 1st - 2nd
> garman.klass.vol<-sqrt(garman.klass.one-garman.klass.two)
> ### Output final daily value
> garman.klass.vol
[1] 0.01646891
>
```

Step 5: Annualize the Volatility

```
> ### Annualize the risk by multiply daily risk by square root of 252 (marketing dates)
> annual.garman.klass.vol<-garman.klass.vol*sqrt(252)
> ### Output annual risk value
> annual.garman.klass.vol
[1] 0.2614358
>
```

4.6.3 Rogers, Satchell, and Yoon

The prior risk measures we have discussed all assume that the mean return is zero. In contrast, the Rogers, Satchell, and Yoon (RSY) volatility properly measures the volatility of securities with an average return that is not zero. The equation for the RSY volatility is:

$$\sigma_{\text{RSY}} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\ln\left(\frac{h_t}{c_t}\right)\ln\left(\frac{h_t}{o_t}\right)+\ln\left(\frac{l_t}{c_t}\right)\ln\left(\frac{l_t}{o_t}\right)\right)}, \tag{4.9}$$

where all the variables are defined in the same way as they were in the Parkinson and Garman-Klass formulas. We now demonstrate how to implement the RSY volatility measure on our Amazon.com data.

Step 1: Obtain Open, High, Low, and Close Data

```
> ### Rogers, Satchell and Yoon
> ### Delete first row and contains only open,high,low,close price
> rsy.vol<-data.AMZN[-1,1:4]
> ### Output
> rsy.vol[c(1:3,nrow(rsy.vol)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close
2011-01-03    9.0685    9.3000   9.0605     9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505
2011-01-05    9.2050    9.3725   9.2035     9.3710
2013-12-31   19.7290   19.9415  19.6900    19.9395
>
```

Step 2: Calculate the Product of First Two Log Terms

We first calculate each of the log terms (rsy.one and rsy.two) and then take their product (rsy.one.two).

```
> ### 1st = high/close
> rsy.one<-log(rsy.vol$AMZN.High/rsy.vol$AMZN.Close)
> ### Output
> rsy.one[c(1:3,nrow(rsy.one)),]
              AMZN.High
2011-01-03 0.0096159782
2011-01-04 0.0144350659
2011-01-05 0.0001600555
2013-12-31 0.0001002482
> ### 2nd = high/open
> rsy.two<-log(rsy.vol$AMZN.High/rsy.vol$AMZN.Open)
> ### Output
> rsy.two[c(1:3,nrow(rsy.two)),]
             AMZN.High
2011-01-03 0.025207530
2011-01-04 0.008292143
2011-01-05 0.018033055
2013-12-31 0.010713353
> ### 1st * 2nd = 1,2 term
> rsy.one.two<-rsy.one*rsy.two
> ### Output
> rsy.one.two[c(1:3,nrow(rsy.one.two)),]
                AMZN.High
2011-01-03 0.000242395059
2011-01-04 0.000119697635
2011-01-05 0.000002886289
2013-12-31 0.000001073995
>
```

Step 3: Calculate the Product of Last Two Log Terms

Similar to Step 2, we first calculate each of the log terms (rsy.three and rsy.four) and then take their product (rsy.three.four).

```
> ### 3rd = low/close
> rsy.three<-log(rsy.vol$AMZN.Low/rsy.vol$AMZN.Close)
> ### Output
> rsy.three[c(1:3,nrow(rsy.three)),]
                AMZN.Low
2011-01-03 -0.016474116
2011-01-04 -0.006670488
2011-01-05 -0.018035968
2013-12-31 -0.012591796
> ### 4th = low/open
> rsy.four<-log(rsy.vol$AMZN.Low/rsy.vol$AMZN.Open)
> ### Output
> rsy.four[c(1:3,nrow(rsy.four)),]
                AMZN.Low
2011-01-03 -0.0008825639
2011-01-04 -0.0128134102
2011-01-05 -0.0001629682
2013-12-31 -0.0019786911
> ### 3rd * 4th = 3,4 term
> rsy.three.four<-rsy.three*rsy.four
> ### Output
> rsy.three.four[c(1:3,nrow(rsy.three.four)),]
                 AMZN.Low
2011-01-03 0.000014539460
2011-01-04 0.000085471694
2011-01-05 0.000002939289
2013-12-31 0.000024915274
>
```

Step 4: Calculate the RSY Volatility Measure

```
> ### Use formula
> rsy.vol<-sqrt((1/nrow(rsy.vol))*sum(rsy.one.two+rsy.three.four))
> ### Output
> rsy.vol
[1] 0.01642272
>
```

Step 5: Annualize the RSY Volatility Measure

```
> ### Annualize the risk by multiplying square root of 252 which is marketing dates
> annual.rsy.vol<-rsy.vol*sqrt(252)
> ### Output
> annual.rsy.vol
[1] 0.2607026
>
```

4.6.4 Yang and Zhang

Yang and Zhang developed a volatility measure that handles both opening jumps and drift, which is sum of the overnight (i.e., volatility from the prior day's close to today's open) and a weighted average of the RSY Volatility and the day's open-to-close volatility.

$$\sigma_{\text{Yang-Zhang}} = \sqrt{\sigma^2_{\text{overnightvol}} + k\sigma^2_{\text{open-to-closevol}} + (1-k)\sigma^2_{\text{RSY}}},$$

where

$$\sigma^2_{\text{overnight vol}} = \frac{1}{T-1}\sum_{t=1}^{T}\left(\ln\left(\frac{o_t}{c_{t-1}}\right) - \text{Avg}\ln\left(\frac{o_t}{c_{t-1}}\right)\right)^2,$$

$$\sigma^2_{\text{open-to-close vol}} = \frac{1}{T-1}\sum_{t=1}^{T}\left(\ln\left(\frac{c_t}{o_t}\right) - \text{Avg}\ln\left(\frac{c_t}{o_t}\right)\right)^2, \text{and}$$

$$k = \frac{\alpha - 1}{\alpha + \frac{T+1}{T-1}}.$$

Step 1: Import Amazon Open, High, Low, and Close Data and Create Variable for Yesterday's Closing Price

```
> ### Yang and Zhang
> ### Only keep the 4 prices
> yz.vol<-data.AMZN[,1:4]
> ### Use lag command to calculate the close price yesterday
> yz.vol$Lag.Close<-Lag(yz.vol$AMZN.Close,k=1)
> ### Output
> yz.vol[c(1:3,nrow(yz.vol)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close Lag.Close
2010-12-31    9.0980    9.1150   8.9755     9.0000        NA
2011-01-03    9.0685    9.3000   9.0605     9.2110    9.0000
2011-01-04    9.3075    9.3850   9.1890     9.2505    9.2110
2013-12-31   19.7290   19.9415  19.6900    19.9395   19.6685
>
```

Step 2: Delete December 31, 2010 Data

```
> ### Delete the first row data as placeholder
> yz.vol<-yz.vol[-1,]
> ### Output
> yz.vol[c(1:3,nrow(yz.vol)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close Lag.Close
2011-01-03    9.0685    9.3000   9.0605     9.2110    9.0000
2011-01-04    9.3075    9.3850   9.1890     9.2505    9.2110
2011-01-05    9.2050    9.3725   9.2035     9.3710    9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395   19.6685
>
```

Step 3: Calculate the First Term in the Yang-Zhang Equation

```
> ### calculate mean log open/close
> yz.one.mean<-mean(log(yz.vol$AMZN.Open/yz.vol$Lag.Close))
> ### Output
> yz.one.mean
[1] 0.0003968058
> ### Insert mean into the formula
> yz.one<-1/(nrow(yz.vol)-1)*sum((log(yz.vol$AMZN.Open/yz.vol$Lag.Close)-yz.one.mean)**2)
> ### Outut 1st term
> yz.one
[1] 0.0001483809
>
```

Step 4: Calculate the Second Term in the Yang-Zhang Equation

```
> ### Calculate the mean in the 2nd term
> yz.two.mean<-mean(log(yz.vol$AMZN.Close/yz.vol$AMZN.Open))
> ### Output
> yz.two.mean
[1] 0.000658205
> ### Insert mean into the formula
> yz.two<-1/(nrow(yz.vol)-1)*sum((
+     + log(yz.vol$AMZN.Close/yz.vol$AMZN.Open)-yz.two.mean)**2)
> ### Output
> yz.two
[1] 0.0002612785
>
```

Step 5: Calculate k

```
> ### alpha = 1.34 is suggested by Yang and Zhang
> k=0.34/(1.34+(nrow(yz.vol)+1)/(nrow(yz.vol)-1))
> ### Output the value of k
> k
[1] 0.1451344
>
```

Step 6: Calculate the Annualized Yang-Zhang Volatility

```
> ### Use the formula above to calculate the volatility and convert to annual data by multiplication of sqrt(252)
> annual.yz.vol<-sqrt(yz.one+k*yz.two+(1-k)*rsy.vol**2)*sqrt(252)
> ### Output
> annual.yz.vol
[1] 0.3241136
>
```

4.6.5 Comparing the Risk Measures

Before we can compare these numbers, we have to first calculate the close-to-close volatility for this same time period.

Step 1: Calculate Amazon Returns Data

```
> ### Only keep Adjusted Price
> AMZN.ret<-data.AMZN[,6]
> ### Open quantmod library
> library(quantmod)
> ### Use Delt command to calculate daily return
> AMZN.ret$Return=Delt(AMZN.ret$AMZN.Adjusted)
> ### Delete first row and keep only Return column
> AMZN.ret<-AMZN.ret[-1,2]
> ### Output
> AMZN.ret[c(1:3,nrow(AMZN.ret)),]
                Return
2011-01-03 0.023444444
2011-01-04 0.004288351
2011-01-05 0.013026323
2013-12-31 0.013778376
>
```

Step 2: Calculate Log Returns

```
> ### Construct a new data-set
> cl2cl.ret<-AMZN.ret
> ### log return = log(gross return)
> cl2cl.ret$logret<-log(1+cl2cl.ret$Return)
> ### Output
> cl2cl.ret[c(1:3,nrow(cl2cl.ret)),]
                Return      logret
2011-01-03 0.023444444 0.023173845
2011-01-04 0.004288351 0.004279182
2011-01-05 0.013026323 0.012942210
2013-12-31 0.013778376 0.013684317
>
```

Step 3: Calculate Standard Deviation of the Returns

```
> ### Calculate standard deviation of log return
> cl2cl.vol<-sd(cl2cl.ret$logret)
> ### Output
> cl2cl.vol
[1] 0.02053515
>
```

Step 4: Annualize the Close-to-Close Volatility

```
> ### Annualize sd by multiplying square root of 252 of number of marketing dates
> annual.cl2cl.vol<-cl2cl.vol*sqrt(252)
> ### Output
> annual.cl2cl.vol
[1] 0.3259855
>
```

Step 5: Create Table of the Different Volatility Measures

```
> ### Combine all volatility measurement indices above with close-to-close volatility
> vol.measures<-rbind(annual.cl2cl.vol,annual.parkinson.vol,
+                     + annual.garman.klass.vol,annual.rsy.vol,annual.yz.vol)
> ### Give names to all kinds of volatility
> rownames(vol.measures)<-c("Close-to-Close","Parkinson","Garman-Klass","Rogers et al","Yang-Zhang")
> ### Point out volatility for comparison
> colnames(vol.measures)<-c("Volatility")
> ### Output
> vol.measures
              Volatility
Close-to-Close  0.3259855
Parkinson       0.2601083
Garman-Klass    0.2614358
Rogers et al    0.2607026
Yang-Zhang      0.3241136
>
```

4.7 Further Reading

Many investment textbooks (e.g., Bodie et al. [3] and Reilly and Brown [7]) have more detailed discussions of standard deviation/variance as well as portfolio risk. Jorion [5] provides an excellent and readable treatment of value-at-risk. A comprehensive discussion of value-at-risk and expected shortfall, as well as the use of a benchmark VaR, can be found in Alexander [1]. Bennett and Gil [2] and Sinclair [8] have a very good discussion of various risk measures

Chapter 5 Factor Models

Factor models are used in many financial applications, such as identifying the determinants of a security's return as well as in cost of capital calculations. Factors help explain the variation in a security's return.

The simplest factor model is one based on a single factor. The most popular of these models is the Capital Asset Pricing Model (CAPM). In the CAPM, only the sensitivity of the security's return to the market portfolio's return matters. The CAPM is based on a series of assumptions, which are detailed in any decent corporate finance or investments textbook. **However, the CAPM does not perform well in empirical testing, but, to its defense, it is unclear whether such poor performance is really a failure of the CAPM or the failure to use the "true" market portfolio of all available assets in these tests (i.e., this is known as Roll's Critique).**

In many empirical tests, especially for practical applications, a simpler version of the CAPM is often used. This is known as the market model. Like the CAPM, the market model is also a single factor model but, this time, there is no imposition as to what kind of market proxy should be used.

Hence, any broad-based stock market index is often used, such as the S&P 500 Index, MSCI World Index, etc.

An alternative to the CAPM, which is a single factor model, are multi-factor models. These models including additional factors that help explain more of the variation in expected stock returns.

The most common multi-factor model in finance is the Three Factor Model developed by Eugene Fama and Kenneth French (FF Model).

In addition to the market's return, the FF Model also includes as a factor the difference in returns between small and large capitalization stocks and the difference in returns of high and low book-to-market (i.e., value and growth stocks) stocks.

The FF Model is based on statistical results, but is now becoming the prevalent factor model used in academia.

5.1 CAPM

Due to its ease of implementation, the most commonly-applied factor model is the Capital Asset Pricing Model (CAPM) by Sharpe.

At least the starting point of many costs of capital calculations is the CAPM, although adjustments may be made to account for the size premium, country risk premium, and other premiums some analysts believe may not be captured by the CAPM.

The value of a company's stock can be characterized as the **present value of the stream of dividend payments** shareholders expect to receive from holding the stock.

Since these dividend payments are expected to **arrive at different points in time in the future**, we have to **discount those dividend payments** by an appropriate risk-adjusted **discount rate**.

The CAPM gives us the appropriate risk-adjusted discount rate to use. From a mathematical perspective, the formula for the CAPM is:

$$r_i = r_f + \beta_i(r_m - r_f),$$

where r_i is the return on asset i, r_f is the return on the risk-free asset, r_m is the return on the market proxy, and β_i is the sensitivity of asset i to the overall market.

CAPM Regression

Empirical tests of the CAPM typically convert the formula into its **excess return** form.

The **excess** pertains to the return over the risk-free rate of both the subject security's return and the market return. That is, the CAPM in **excess** return form is as follows:

$$r_i - r_f = \alpha + \beta_i(r_m - r_f), \tag{5.2}$$

where all the variables are defined in the same way as in Eq. (5.1). The α and β_i in Eq. (5.2) are estimated using an OLS regression.

To perform the CAPM regression, we need to choose:

(i) the length of the estimation period,

(ii) the frequency of the returns data,

(iii) the risk-free rate used in the calculation.

Technically, this gives us a large number of possibilities and could potentially lead to a large number of variations in the results.

In our implementation, for (i) and (ii), we follow the methodology described on the Morningstar and Yahoo Finance websites, which is to use 3 years of monthly returns or 36 monthly return observations for both the subject security's return and market return.

For (iii), we use the 3-Month Treasury Bill rate following Morningstar's description.

Step 1: Import Portfolio Returns and Convert to a data.frame Object

In this Appendix, we construct a hypothetical portfolio assuming portfolio returns are equal to that of an equal-weighted portfolio of Amazon.com (AMZN), Nvidia (NVDA), and IBM stocks from January 2011 to December 2013 with monthly rebalancing. To implement this, we need to make sure we have data from December 2010 in order to calculate a monthly return for the monthly of January 2011. Note that this is merely a hypothetical portfolio with constructed returns. This is meant as a proxy for our actual portfolio returns used in this book. To construct the portfolio returns, we calculate the monthly returns for AMZN, NVDA, and IBM using the techniques we discussed in Chap. 2.

Here is the code and the output below for simplicity.

```
> ### Amazon Data
> data.AMZN<-read.csv("AMZN Yahooo.csv",header=TRUE)
> data.AMZN<-data.AMZN[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.AMZN$Date,format="%Y-%m-%d")
> data.AMZN<-cbind(date, data.AMZN[,-1])
> data.AMZN<-data.AMZN[order(data.AMZN$date),]
> data.AMZN<-xts(data.AMZN[,2:7],order.by=data.AMZN[,1])
> names(data.AMZN)<-paste(c("AMZN.Open","AMZN.High","AMZN.Low","AMZN.Close","AMZN.Volume","AMZN.Adjusted"))
> data.AMZN[c(1:3,nrow(data.AMZN)),]
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
> AMZN.monthly<-to.monthly(data.AMZN)
> AMZN.monthly[c(1:3,nrow(AMZN.monthly)),]
         data.AMZN.Open data.AMZN.High data.AMZN.Low data.AMZN.Close data.AMZN.Volume data.AMZN.Adjusted
12月 2010         9.0980         9.1150        8.9755          9.0000         69038000             9.0000
1月 2011          9.0685         9.5800        8.3450          8.4820       2272226000             8.4820
2月 2011          8.5260         9.5700        8.4755          8.6645       1915528000             8.6645
12月 2013        19.9500        20.2815       18.9750         19.9395       1113734000            19.9395
> AMZN.monthly<-AMZN.monthly[,6]
> AMZN.ret<-Delt(AMZN.monthly$data.AMZN.Adjusted)
> names(AMZN.ret)<-paste("AMZN.ret")
> AMZN.ret[c(1:3,nrow(AMZN.ret)),]
            AMZN.ret
12月 2010          NA
1月 2011  -0.05755556
2月 2011   0.02151615
12月 2013  0.01313455
> ### Nvidia Data
> data.NVDA<-read.csv("NVDA Yahooo.csv",header=TRUE)
> data.NVDA<-data.NVDA[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.NVDA$Date,format="%Y-%m-%d")
> data.NVDA<-cbind(date, data.NVDA[,-1])
> data.NVDA<-data.NVDA[order(data.NVDA$date),]
> data.NVDA<-xts(data.NVDA[,2:7],order.by=data.NVDA[,1])
> names(data.NVDA)<-paste(c("NVDA.Open","NVDA.High","NVDA.Low","NVDA.Close","NVDA.Volume","NVDA.Adjusted"))
> data.NVDA[c(1:3,nrow(data.NVDA)),]
           NVDA.Open NVDA.High NVDA.Low NVDA.Close NVDA.Volume NVDA.Adjusted
2010-12-31    3.7500    3.8550   3.745     3.8500    39125200       3.531613
2011-01-03    3.8800    3.9925   3.875     3.9550    81744800       3.627930
2011-01-04    3.9625    3.9800   3.855     3.9425    65138400       3.616464
2013-12-31    4.0000    4.0250   3.975     4.0050    23577600       3.778282
> NVDA.monthly<-to.monthly(data.NVDA)
> NVDA.monthly[c(1:3,nrow(NVDA.monthly)),]
         data.NVDA.Open data.NVDA.High data.NVDA.Low data.NVDA.Close data.NVDA.Volume data.NVDA.Adjusted
12月 2010         3.7500         3.8550        3.745          3.850         39125200            3.531613
1月 2011          3.8800         6.2625        3.855          5.980       3273688400            5.485468
2月 2011          6.0325         6.5425        5.445          5.665       2402576400            5.196517
12月 2013         3.8850         4.0250        3.725          4.005        616797600            3.778282
> NVDA.monthly<-NVDA.monthly[,6]
> NVDA.ret<-Delt(NVDA.monthly$data.NVDA.Adjusted)
> names(NVDA.ret)<-paste("NVDA.ret")
> NVDA.ret[c(1:3,nrow(NVDA.ret)),]
```

```
             NVDA.ret
12月 2010          NA
1月 2011   0.55324720
2月 2011  -0.05267572
12月 2013  0.02692277
> ### IBM Data
> data.IBM<-read.csv("IBM Yahooo.csv",header=TRUE)
> data.IBM<-data.IBM[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.IBM$Date,format="%Y-%m-%d")
> data.IBM<-cbind(date, data.IBM[,-1])
> data.IBM<-data.IBM[order(data.IBM$date),]
> data.IBM<-xts(data.IBM[,2:7],order.by=data.IBM[,1])
> names(data.IBM)<-paste(c("IBM.Open","IBM.High","IBM.Low","IBM.Close","IBM.Volume","IBM.Adjusted"))
> data.IBM[c(1:3,nrow(data.IBM)),]
           IBM.Open IBM.High  IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2010-12-31 140.2773 140.6023 139.5411  140.3059    3106411     85.83682
2011-01-03 140.7361 141.6826 140.6692  140.9943    4815575     86.25792
2011-01-04 141.0707 141.7017 140.1912  141.1472    5292865     86.35150
2013-12-31 178.2887 179.5316 178.1071  179.3212    3786206    115.64046
> IBM.monthly<-to.monthly(data.IBM)
> IBM.monthly[c(1:3,nrow(IBM.monthly)),]
          data.IBM.Open data.IBM.High data.IBM.Low data.IBM.Close data.IBM.Volume data.IBM.Adjusted
12月 2010      140.2773      140.6023     139.5411       140.3059         3106411          85.83682
1月 2011       140.7361      157.1224     140.1912       154.8757       124297854          94.75039
2月 2011       154.9809      158.9388     152.0363       154.7610        93164606          95.05507
12月 2013      171.5679      179.5316     165.1338       179.3212       102500782         115.64046
> IBM.monthly<-IBM.monthly[,6]
> IBM.ret<-Delt(IBM.monthly$data.IBM.Adjusted)
> names(IBM.ret)<-paste("IBM.ret")
> IBM.ret[c(1:3,nrow(IBM.ret)),]
             IBM.ret
12月 2010          NA
1月 2011  0.103843252
2月 2011  0.003215607
12月 2013 0.043911677
> port<-cbind(AMZN.ret,NVDA.ret,IBM.ret)
> port[c(1:3,nrow(port)),]
             AMZN.ret    NVDA.ret     IBM.ret
12月 2010          NA          NA          NA
1月 2011  -0.05755556  0.55324720 0.103843252
2月 2011   0.02151615 -0.05267572 0.003215607
12月 2013  0.01313455  0.02692277 0.043911677
> port$port.ret<-rowMeans(port)
> port[c(1:3,nrow(port)),]
             AMZN.ret    NVDA.ret     IBM.ret     port.ret
12月 2010          NA          NA          NA           NA
1月 2011  -0.05755556  0.55324720 0.103843252  0.199844965
2月 2011   0.02151615 -0.05267572 0.003215607 -0.009314655
12月 2013  0.01313455  0.02692277 0.043911677  0.027989666
> port<-port[-1,4]
> port[c(1:3,nrow(port)),]
              port.ret
1月 2011   0.199844965
2月 2011  -0.009314655
3月 2011  -0.046175365
12月 2013  0.027989666
> csv.port<-cbind(data.frame(index(port)),data.frame(port))
> names(csv.port)[1]<-paste("date")
> csv.port[c(1:3,nrow(csv.port)),]
              date     port.ret
1月 2011   1月 2011  0.199844965
2月 2011   2月 2011 -0.009314655
3月 2011   3月 2011 -0.046175365
12月 2013 12月 2013  0.027989666
> rownames(csv.port)<-seq(1,nrow(csv.port),by=1)
> csv.port[c(1:3,nrow(csv.port)),]
        date     port.ret
1   1月 2011  0.199844965
2   2月 2011 -0.009314655
3   3月 2011 -0.046175365
36 12月 2013  0.027989666
> write.csv(csv.port,"Hypothetical Portfolio (Monthly).csv")
>
```

IBM Yahooo.csv	2024/6/7 16:19	Comma Separated ...	55 KB
NVDA Yahooo.csv	2024/6/7 16:35	Comma Separated ...	49 KB
Hypothetical Portfolio (Monthly).csv	2024/6/21 16:22	Comma Separated ...	2 KB

Obviously, the monthly hypothetical portfolio data has been saved in the hierarchy.

Now we go back to implement CAPM Factor Model.

```
> ### Read the monthly saved file of hypothesis
> port<-read.csv("Hypothetical Portfolio (Monthly).csv")
> ### Output
> port[c(1:3,nrow(port)),]
    X      date     port.ret
1   1  1月 2011  0.199844965
2   2  2月 2011 -0.009314655
3   3  3月 2011 -0.046175365
36 36 12月 2013  0.027989666
>
```

Since we are only using month and year, we use the yearmon class, which we can apply by using the as.yearmon command. Note the %b %Y format. %b tells R the format being read has a three-letter month and the %Y tells R the year is a four-digit year. Details on the different date formats are found in Appendix A.

```
> ### Show current class of port$date
> class(port$date)
[1] "character"
>
> ### Change to year and month by as.yearmon command
> ### %b as the three-letter month and %Y as the four-digit year
> port$date<-as.yearmon(as.character(port$date),"%b %Y")
> ### Output
> port[c(1:3,nrow(port)),]
    X      date     port.ret
1   1  1月 2011  0.199844965
2   2  2月 2011 -0.009314655
3   3  3月 2011 -0.046175365
36 36 12月 2013  0.027989666
>
> ### Show the class again to see the convertion
> class(port$date)
[1] "yearmon"
>
> ### Convert to data.frame type
> port.df<-data.frame(port)
> ### Output
> port.df[c(1:3,nrow(port.df)),]
    X      date     port.ret
1   1  1月 2011  0.199844965
2   2  2月 2011 -0.009314655
3   3  3月 2011 -0.046175365
36 36 12月 2013  0.027989666
>
```

Step 2: Import S&P 500 Index Data from Yahoo Finance and Calculate Monthly Market Returns

As I mentioned above, I have replaced S&P 500 Index by Apple (AAPL) to kind of

modelling the trends of the market.

```
> ### Use AAPL to model the market instead of using GSPC
> data.AAPL<-read.csv("AAPL Yahooo.csv",header=TRUE)
> data.AAPL<-data.AAPL[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.AAPL$Date,format="%Y-%m-%d")
> data.AAPL<-cbind(date, data.AAPL[,-1])
> data.AAPL<-data.AAPL[order(data.AAPL$date),]
> data.mkt<-xts(data.AAPL[,2:7],order.by=data.AAPL[,1])
> names(data.mkt)[1:6]<-paste(c("AAPL.Open","AAPL.High","AAPL.Low","AAPL.Close","AAPL.Volume","AAPL.Adjusted"))
> data.mkt[c(1:3,nrow(data.mkt)),]
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2010-12-31  11.53393  11.55286 11.47536   11.52000   193508000      9.739614
2011-01-03  11.63000  11.79500 11.60143   11.77036   445138400      9.951281
2011-01-04  11.87286  11.87500 11.71964   11.83179   309080800     10.003214
2013-12-31  19.79179  20.04571 19.78571   20.03643   223084400     17.519623
> 
> ### Convert to monthly data
> mkt.monthly<-to.monthly(data.mkt)
> ### Output
> mkt.monthly[c(1:3,nrow(mkt.monthly)),]
          data.mkt.Open data.mkt.High data.mkt.Low data.mkt.Close data.mkt.Volume data.mkt.Adjusted
12月 2010      11.53393      11.55286     11.47536       11.52000       193508000          9.739614
1月 2011       11.63000      12.45000     11.60143       12.11857     10841535600         10.245680
2月 2011       12.18929      13.03214     12.06143       12.61464      9295949600         10.665081
12月 2013      19.92857      20.54071     19.24286       20.03643      7057397200         17.519623
> 
> ### Only Adjusted price is needed
> mkt.monthly<-mkt.monthly[,6]
> ### Calculate return by delta function
> mkt.ret<-Delt(mkt.monthly$data.mkt.Adjusted)
> ### Rename meaningfully
> names(mkt.ret)<-paste("mkt.ret")
> ###  Output
> mkt.ret[c(1:3,nrow(mkt.ret)),]
             mkt.ret
12月 2010          NA
1月 2011  0.051959554
2月 2011  0.040934423
12月 2013 0.008901618
> 
> ### Delete first row
> mkt.ret<-mkt.ret[-1,]
> ### Output
> mkt.ret[c(1:3,nrow(mkt.ret)),]
                mkt.ret
1月 2011   0.051959554
2月 2011   0.040934423
3月 2011  -0.013306416
12月 2013  0.008901618
> 
> ### Also convert to data.frame type
> market.df<-data.frame(mkt.ret)
> ### See the head
> head(market.df)
             mkt.ret
1月 2011  0.051959554
2月 2011  0.040934423
3月 2011 -0.013306416
4月 2011  0.004648601
5月 2011 -0.006568903
6月 2011 -0.034959503
> 
```

Step 3: Import Risk-Free Rate Data from FRED and Setup Data to Contain Monthly Risk-Free Returns

```
> ### Read the file of risk-free rate
> rf<-read.csv("DGS3MO.csv",header=TRUE)
> ### See first three rows
> rf[1:3,]
        DATE DGS3MO
1 1982-01-04  11.87
2 1982-01-05  12.20
3 1982-01-06  12.16
> 
> ### create a Date object date to replace first column by converting
> rf$date<-as.Date(rf$DATE,"%Y-%m-%d")
> ### Convert to character and then numeric values of 2nd column
> rf$DGS3MO<-as.numeric(as.character(rf$DGS3MO))
警告信息:
强制改变过程中产生了NA
> 
> ### output
> rf[c(1:3,nrow(rf)),]
           DATE DGS3MO       date
1    1982-01-04  11.87 1982-01-04
2    1982-01-05  12.20 1982-01-05
3    1982-01-06  12.16 1982-01-06
8350 2014-01-03   0.07 2014-01-03
> 
> ### See details of data-set
> str(rf)
'data.frame':   8350 obs. of  3 variables:
 $ DATE  : chr  "1982-01-04" "1982-01-05" "1982-01-06" "1982-01-07" ...
 $ DGS3MO: num  11.9 12.2 12.2 12.2 12 ...
 $ date  : Date, format: "1982-01-04" "1982-01-05" "1982-01-06" "1982-01-07" ...
> 
```

Step 3a: Convert to xts Object

```
> ### Change to xts and select data column and order by date object
> rf<-xts(rf$DGS3MO,order.by=rf$date)
> ### Output
> rf[1:3,]
           [,1]
1982-01-04 11.87
1982-01-05 12.20
1982-01-06 12.16
> ### Rename
> names(rf)<-paste("DGS3MO")
> ### Output
> rf[1:3,]
           DGS3MO
1982-01-04  11.87
1982-01-05  12.20
1982-01-06  12.16
> 
```

Step 3b: Apply to.monthly Command to Identify First Yield for Each Month

```
> ### Convert to monthly data
> rf.monthly<-to.monthly(rf)
警告信息:
In to.period(x, "months", indexAt = indexAt, name = name, ...) :
  missing values removed from data
> ### Show first three rows
> rf.monthly[1:3,]
         rf.Open rf.High rf.Low rf.Close
1月 1982   11.87   14.06  11.87    13.08
2月 1982   14.77   15.49  12.79    13.00
3月 1982   12.81   14.16  12.68    13.99
>
```

Step 3c: Convert Opening Annualized Yield for Each Month Into a Monthly Yield

```
> ### Not to convert into scientific notation
> options(scipen="100")
> ### convert annualized monthly data into monthly data
> rf.monthly<-(1+rf.monthly[,1]/100)**(1/12)-1
> ### Output
> rf.monthly[c(1:3,nrow(rf.monthly)),]
               rf.Open
1月 1982 0.00939109694
2月 1982 0.01154614300
3月 1982 0.01009518279
1月 2014 0.00005831463
>
```

Step 3d: Subset Data to January 2011 Through December 2013

```
> ### Use subset to limit into the needed time duration
> rf.sub<-subset(rf.monthly,index(rf.monthly) >= as.yearmon("1月 2011") &index(rf.monthly) <= as.yearmon("12月 2013"))
> ### Output
> rf.sub[c(1:3,nrow(rf.sub)),]
               rf.Open
1月 2011  0.00012491414
2月 2011  0.00012491414
3月 2011  0.00011659187
12月 2013 0.00004165712
>
```

Step 4: Combine Firm, Market, and Risk-Free Data into One Data Object

```
> ### Combine market, risk-free data and portfolio return
> combo<-cbind(market.df,data.frame(rf.sub),port.df$port.ret)
> ### Output
> combo[c(1:3,nrow(combo)),]
                 mkt.ret       rf.Open port.df$port.ret
1月 2011    0.051959554 0.00012491414      0.199844965
2月 2011    0.040934423 0.00012491414     -0.009314655
3月 2011   -0.013306416 0.00011659187     -0.046175365
12月 2013   0.008901618 0.00004165712      0.027989666
>
```

```
> ### Rename meaningfully
> names(combo)<-paste(c("mkt.ret","rf","port.ret"))
> ### Output
> combo[c(1:3,nrow(combo)),]
               mkt.ret            rf     port.ret
1月 2011    0.051959554 0.00012491414  0.199844965
2月 2011    0.040934423 0.00012491414 -0.009314655
3月 2011   -0.013306416 0.00011659187 -0.046175365
12月 2013   0.008901618 0.00004165712  0.027989666
>
```

Step 5: Calculate Excess Firm Return and Excess Market Return

```
> ### excess portfolio return
> combo$exret<-combo$port.ret-combo$rf
> ### excess market return
> combo$exmkt<-combo$mkt.ret-combo$rf
> ### output
> combo[c(1:3,nrow(combo)),]
              mkt.ret            rf     port.ret        exret        exmkt
1月 2011    0.051959554 0.00012491414  0.199844965  0.199720051  0.051834640
2月 2011    0.040934423 0.00012491414 -0.009314655 -0.009439569  0.040809509
3月 2011   -0.013306416 0.00011659187 -0.046175365 -0.046291957 -0.013423007
12月 2013   0.008901618 0.00004165712  0.027989666  0.027948009  0.008859961
>
```

Step 6: Run Regression of Excess Firm Return on Excess Market Return

```
> ### at least 3 digits after first non-zero digit
> options(digits=3)
> ### Use lm to run the regression
> CAPM<-lm(combo$exret~combo$exmkt)
> summary(CAPM)

Call:
lm(formula = combo$exret ~ combo$exmkt)

Residuals:
    Min      1Q  Median      3Q     Max
-0.1005 -0.0302  0.0031  0.0215  0.1791

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01030    0.00898    1.15    0.260
combo$exmkt  0.19857    0.11613    1.71    0.096 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0522 on 34 degrees of freedom
Multiple R-squared:  0.0792,    Adjusted R-squared:  0.0521
F-statistic: 2.92 on 1 and 34 DF,  p-value: 0.0964

>
```

CAPM Alpha

When making investments into a fund, investors often consider the contribution of the fund manager to the performance of the fund. As such, we need a way to measure whether a fund manager provided value that goes beyond simply investing in the index (i.e., is there a benefit to active management by the manager instead of simply using a passive investment strategy). An alternative way to look at this problem is we want to know how well did the **fund manager** do compared to her ***benchmark***?

To answer this question, we look at the ***alpha*** of the manager as a measure this outperformance.

The regression controls for the sensitivity of the portfolio's return to its benchmark.

If this alpha is **positive and statistically significant**, the manager is taken to have provided positive value. Conversely, a **negative and statistically significant** alpha is taken to mean that the manager has provided negative value (i.e., we would have been better off investing in the benchmark).

The alpha of the portfolio is the intercept term from the regression output above. The intercept is equal to 0.01030, which translates to a monthly return of 1.03 %. This alpha is statistically insignificant at the 5 % level, which is a conventional level of significance used in practice. As such, assuming our hypothetical portfolio returns, we can't make the decision since it is statistically insignificant.

CAPM Beta

Another important measure we can derive from the results of the CAPM regression is the portfolio's ***beta***. The ***beta*** of a portfolio measures how ***sensitive*** the portfolio's return is to the movement of the overall market. Therefore, the ***beta*** of the portfolio measures what is called ***systematic risk or market risk***. **Systematic risk** is the portion of a security's risk that **cannot be diversified away** and, as such, it is commonly thought of as the level of risk that investors are ***compensated*** from taking on.

From taking on. From the regression results above, we can read-off what the beta for the portfolio is. This is the parameter estimate of the combo$exmkt variable of 0.199 with a p-value of 0.096. The p-value tells us the minimum significance level for which the beta is statistically significant. A p-value of 0.096 means that only at significant level of 10% it is statistically significant, neither that we still cannot make the decision.

An alternative way to view the beta and the p-value is to call them directly from the regression summary results. Instead of simply stopping at **summary (CAPM)**, we can include **$coefficient[...]** where inside the square brackets we could place **one or a vector of numbers**

using the **c(...) operator** to output. Looking at the regression summary output above, we see the coefficients have eight values. The numbering works like this. Number 1 is the Estimate of the intercept, number 2 is the estimate of the excess market return (or the beta), number 3 goes back up as the Std. Err. of the intercept, then number 4 goes down as the Std. Err. of the beta, and so on. So, if we need to know what the **beta and p-value of the beta** are, we choose numbers **2 and 8**.

```
> ### Left to Right, Up and Down
> ### 2nd and 8th numeric number of coefficients data-set
> beta<-summary(CAPM)$coefficients[2]
> beta
[1] 0.199
> beta.pval<-summary(CAPM)$coefficients[8]
> beta.pval
[1] 0.0964
>
> beta_pval = summary(CAPM)$coefficients[c(2,8)]
> beta_pval
[1] 0.1986 0.0964
>
```

The results of the CAPM regression show that the CAPM beta is 0.199. This beta of 0.199 can then be used in the CAPM to calculate, say, the **cost of equity for the company**. This means that if the market goes up by 1 %, we expect our portfolio to go up by only 0.2 %. However, if the market goes down by 1 %, we expect our portfolio to only go down by 0.2 %. A beta less than one is consistent with **betas of defensive stocks** as these stocks are less affected by adverse market movements.

Using Calculated Beta to Estimate Cost of Equity Using CAPM

To calculate the cost of equity using the CAPM, we need to find an **Equity Risk Premium (ERP)** that is compatible.

For our example, we should use an **ERP** that is calculated using a **3-Month Treasury security**. One such source for the historical **ERP** is **Professor Damodaran**'s website.

Professor Damodaran's calculations show an ERP of 5.38 % based on an arithmetic average of returns from 1928 to 2012 and 7.19 % based on an arithmetic average of returns from 1962 to 2012.

This range of ERPs is consistent with the general range of ERPs of around 5–8 %. Given that the 3-Month Treasury as of December 31, 2013 was approximately 0.07 %, we estimate the cost of equity of our portfolio is 1.15 % [= 0.07 % + 0.2 * 5.38 %] based on the ERP from 1926 to 2012 and 1.51 % [= 0.07 % + 0.2 * 7.19 %] based on the ERP from 1962 to 2012.

The low cost of equity is due to the low beta of 0.199 we estimated for our portfolio.

Calculate Adjusted Beta

There have been studies that show betas that are above the market beta of one tend to go down in the long-term, while betas that are below the market beta of one tend to go up in the long-term.

Since valuations take a long-term view,

some believe betas used in calculating the cost of equity need to be adjusted to reflect this reversion to the market beta.

One common adjustment is to apply 2/3 weight to the raw beta, which is the beta we calculated above, and 1/3 weight to the market beta of one.

```
> ### Adjusted Beta reverse to market Beta by weighted-average with 1
> adj.beta<-(2/3)*beta+(1/3)*1
> ### Output
> adj.beta
[1] 0.466
> ### More effective digits
> options(digits=7)
>
```

The weighting we used for the adjusted beta calculation is the standard assumption used by Bloomberg when reporting adjusted beta.

5.2 Market Model

A more common way to calculate beta in practice is to use the market model, because the market model uses a market proxy without the requirement that this market proxy be the "true" market portfolio.

In addition, the market model does not require the use of a risk-free rate and, therefore, there is no need to calculate the excess returns of the firm and the market. That is,

$$r_i = \alpha + \beta r_m,$$

where all variables are defined the same way as in the CAPM regression.

```
Call:
lm(formula = combo$port.ret ~ combo$mkt.ret)

Residuals:
     Min       1Q   Median       3Q      Max
-0.10050 -0.03021  0.00311  0.02149  0.17919

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.01034    0.00898    1.15    0.258
combo$mkt.ret  0.19854    0.11615    1.71    0.097 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0522 on 34 degrees of freedom
Multiple R-squared:  0.0791,    Adjusted R-squared:  0.0521
F-statistic: 2.92 on 1 and 34 DF,  p-value: 0.0965

>
```

The beta calculated using the market model is 0.19854, which is very close to the beta calculated using the CAPM of 0.19857. The small difference is primarily driven by the low, stable risk-free rate during the estimation period.

```
> ### Adjusted Beta reversing to 1
> beta.mktmod<-summary(reg)$coefficients[2]
> beta.mktmod
[1] 0.199
> adj.beta.mktmod<-(2/3)*beta.mktmod+(1/3)*1
> adj.beta.mktmod
[1] 0.466
> options(digits=7)
>
```

5.3 Rolling Window Regressions

In this section, we demonstrate how to run regressions over a rolling window, so we can calculate the alphas and betas for Amazon.com over multiple periods to analyze the variation of the alpha and beta through time.

In our example, we will calculate alphas and betas using regressions on 252 trading day periods from 2012 to 2013. Since our first return starts in 2011, the 252 trading day window, which is approximately one calendar year in length, will use virtually all of the returns in 2011 to generate the first estimate of alpha and beta on December 31, 2011.

Step 1: ImportAmazon.com and Apple.com Data

```
> data.AMZN
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2010-12-31    9.0980    9.1150   8.9755     9.0000    69038000        9.0000
2011-01-03    9.0685    9.3000   9.0605     9.2110   106628000        9.2110
2011-01-04    9.3075    9.3850   9.1890     9.2505   100636000        9.2505
2011-01-05    9.2050    9.3725   9.2035     9.3710    68376000        9.3710
2011-01-06    9.3250    9.3705   9.2625     9.2930    63594000        9.2930
2011-01-07    9.3940    9.4225   9.1870     9.2745   104434000        9.2745
2011-01-10    9.2520    9.2645   9.1255     9.2340    67518000        9.2340
2011-01-11    9.2710    9.3000   9.1605     9.2170    56284000        9.2170
2011-01-12    9.2680    9.2690   9.1650     9.2040    53582000        9.2040
2011-01-13    9.1800    9.3225   9.1755     9.2765    67340000        9.2765
      ...
2013-12-17   19.5325   19.5680  19.3250    19.3825    46878000       19.3825
2013-12-18   19.4615   19.8150  19.1550    19.7980    69782000       19.7980
2013-12-19   19.7135   19.8645  19.6300    19.7595    48544000       19.7595
2013-12-20   19.8275   20.2360  19.7890    20.1100   100678000       20.1100
2013-12-23   20.1845   20.2500  19.9600    20.1460    53190000       20.1460
2013-12-24   20.1260   20.1860  19.8185    19.9600    27608000       19.9600
2013-12-26   20.0895   20.2260  19.8405    20.2195    37370000       20.2195
2013-12-27   20.2325   20.2815  19.8125    19.9040    39738000       19.9040
2013-12-30   19.9705   19.9960  19.6225    19.6685    49742000       19.6685
2013-12-31   19.7290   19.9415  19.6900    19.9395    39930000       19.9395
>
> data.AAPL<-xts(data.AAPL[,2:7],order.by=data.AAPL[,1])
> names(data.AAPL)[1:6]<-paste(c("AAPL.Open","AAPL.High","AAPL.Low","AAPL.Close","AAPL.Volume","AAPL.Adjusted"))
> data.AAPL
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2010-12-31  11.53393  11.55286 11.47536   11.52000   193508000      9.739614
2011-01-03  11.63000  11.79500 11.60143   11.77036   445138400      9.951281
2011-01-04  11.87286  11.87500 11.71964   11.83179   309080800     10.003214
2011-01-05  11.76964  11.94071 11.76786   11.92857   255519600     10.085044
2011-01-06  11.95429  11.97321 11.88929   11.91893   300428800     10.076889
2011-01-07  11.92821  12.01250 11.85357   12.00429   311931200     10.149058
2011-01-10  12.10107  12.25821 12.04179   12.23036   448560000     10.340188
2011-01-11  12.31714  12.32000 12.12393   12.20143   444108000     10.315734
2011-01-12  12.25893  12.30107 12.21429   12.30071   302590400     10.399675
2011-01-13  12.32714  12.38000 12.28036   12.34571   296780400     10.437716
      ...
2013-12-17  19.85036  19.98000 19.76357   19.82107   229902400     17.331314
2013-12-18  19.63214  19.69464 19.24286   19.67036   565863200     17.199532
2013-12-19  19.62500  19.64286 19.41893   19.44500   320308800     17.002474
2013-12-20  19.47964  19.70036 19.45786   19.60786   436413600     17.144880
2013-12-23  20.28571  20.38286 20.09857   20.36036   501306400     17.802856
2013-12-24  20.35321  20.42429 20.21536   20.27393   167554800     17.727283
2013-12-26  20.28929  20.33929 20.12071   20.13929   204008000     17.609562
2013-12-27  20.13643  20.15750 19.98214   20.00321   225884400     17.490583
2013-12-30  19.90929  20.00321 19.72571   19.80429   253629600     17.316637
2013-12-31  19.79179  20.04571 19.78571   20.03643   223084400     17.519623
>
```

Both have been imported by R memory.

Step 2: Calculate the Amazon.com and Apple.com Returns

```
> ### difference of log return of two securities
> rets<-diff(log(data.AMZN$AMZN.Adjusted))
> rets$GSPC<-diff(log(data.mkt$AAPL.Adjusted))
> ### Rename
> names(rets)[1]<-"AMZN"
> ### Output
> rets[c(1:3,nrow(rets)),]
                   AMZN         GSPC
2010-12-31           NA           NA
2011-01-03 0.023173845 0.021499800
2011-01-04 0.004279182 0.005205155
2013-12-31 0.013684317 0.011653851
> ### Delete first row
> rets<-rets[-1,]
> ### Output
> rets[c(1:3,nrow(rets)),]
                   AMZN         GSPC
2011-01-03 0.023173845 0.021499800
2011-01-04 0.004279182 0.005205155
2011-01-05 0.012942210 0.008147093
2013-12-31 0.013684317 0.011653851
>
```

Step 3: Create the Rolling Window Regression Function

```
> ### Require zoo library
> require(zoo)
> ### Use rollapply command to regression the coefficients
> coeffs<-rollapply(rets,width=252,FUN=function(X){roll.reg=lm(AMZN~AAPL,data=as.data.frame(X))
+                       + return(roll.reg$coef)},by.column=FALSE)
>
> coeffs[c(1,251:253,nrow(coeffs)),]
            X.Intercept.          AAPL
2011-01-03            NA            NA
2011-12-29            NA            NA
2011-12-30 -0.0008611635  0.781754223
2012-01-03 -0.0008023798  0.784330726
2013-12-31  0.0018417205 -0.007904591
>
```

Note that the term return(roll.reg$coeff) is used to output the parameter estimates (i.e., the intercept term and the coefficient for the market return) of the regression.

Step 4: Remove NAs From the Data

As the output above shows, the first 251 observations of coeffs are NAs. This is because we need 252 observations to generate the first regression, which would be on December 30, 2011. As such, we would want to delete all those NAs.

```
> ### Delete all NA coeffs
> coeffs<-na.omit(coeffs)
> coeffs[c(1:3,nrow(coeffs)),]
            X.Intercept.          AAPL
2011-12-30 -0.0008611635  0.781754223
2012-01-03 -0.0008023798  0.784330726
2012-01-04 -0.0008529346  0.783486375
2013-12-31  0.0018417205 -0.007904591
>
```

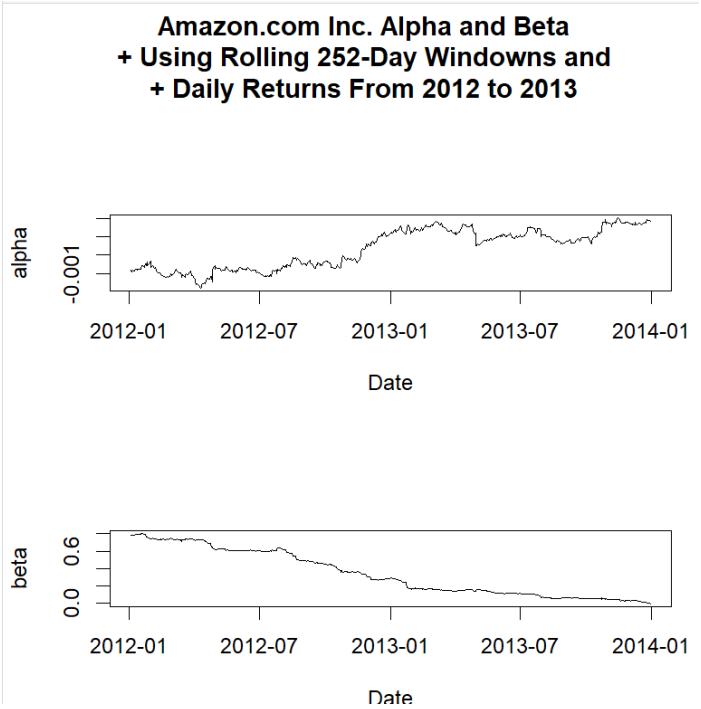
Step 5: Clean-Up Data

For presentation purposes, the coeffs data has a single observation in 2011, contains labels that may not make sense to other people, and we may feel that there are too many decimals that are being reported.

```
> ### Delete only one data in 2011
> coeffs<-coeffs[-1,]
> ### Rename meaningfully
> names(coeffs)<-c("Alpha","Beta")
> ### Fewer digits in output
> options(digits=3)
> ### Output
> coeffs[c(1:3,nrow(coeffs)),]
              Alpha    Beta
2012-01-03 -0.000802  0.7843
2012-01-04 -0.000853  0.7835
2012-01-05 -0.000909  0.7816
2013-12-31  0.001842 -0.0079
>
```

Step 6: Plot the Data

```
> ### Plot the data
> par(oma=c(0,0,4,0))
> par(mfrow=c(2,1))
> plot(x=index(coeffs),xlab="Date",y=coeffs$Alpha,ylab="alpha",type="l")
> plot(x=index(coeffs),xlab="Date",y=coeffs$Beta,ylab="beta",type="l")
> title(main="Amazon.com Inc. Alpha and Beta
+ + Using Rolling 252-Day Windowns and
+ + Daily Returns From 2012 to 2013",outer=TRUE)
> par(mfrow=c(1,1))
>
```



From the graph it is known that risk-free assets are increasing in values over the time, and compared to Apple.com, Amazon.com becomes more and more of a defensive stock since beta decreases steady to 0, where AMZN is not affected by AAPL finally.

5.4 Fama-French Three Factor Model

The popularity of the CAPM is due to its simplicity and it is grounded in finance theory. However, the CAPM has not performed well in empirical testing. This suggests other factors may need to be added to help explain the remaining variation in asset returns unexplained by the market.

One such model that has gained popularity, especially in the academic community, is the Fama-French Three Factor (FF) Model (see Fama and French).

For example, Cochrane points out that the FF Model has taken the place of the CAPM for routine risk adjustment in empirical work.

In the FF Model, we have

$$r_i = r_f + \beta_i(r_m - r_F) + hHML + sSMB, \tag{5.4}$$

where HML is the difference in the returns of portfolios with high B/M ratios and low B/M ratios and SMB is the difference in returns of portfolios of small company stocks and big company stocks. The rest of the variables are defined the same way as in Eq. (5.1).

The data needed to implement the FF Model can be downloaded from Professor Kenneth French's Data Library (U.S. Research Returns Data). Note that the files on Professor French's data library are regularly updated, so downloading data at a later date results in retrieving more data albeit the start of the data should remain the same as what we report below.

Step 1: Import Portfolio Returns Data

```
> ### Check R memory
> port[c(1:3,nrow(port)),]
    X      date port.ret
1   1  1月 2011  0.19984
2   2  2月 2011 -0.00931
3   3  3月 2011 -0.04618
36 36 12月 2013  0.02799
> 
```

Step 2: Import Fama-French Data Retrieved From Ken French's Website

```
> ### Import FF Data
> FF.raw<-read.fwf(file="F-F_Research_Data_Factors.txt",widths=c(6,8,8,8,8),skip=4)
> head(FF.raw)
      V1       V2       V3       V4       V5
1 192607     2.96    -2.56    -2.43     0.22
2 192608     2.64    -1.17     3.82     0.25
3 192609     0.36    -1.40     0.13     0.23
4 192610    -3.24    -0.09     0.70     0.32
5 192611     2.53    -0.10    -0.51     0.31
6 192612     2.62    -0.03    -0.05     0.28
> tail(FF.raw)
         V1       V2       V3       V4       V5
1271   2020    23.66    13.18   -46.67     0.45
1272   2021    23.56    -3.89    25.49     0.04
1273   2022   -21.60    -6.95    25.81     1.43
1274   2023    21.70    -3.24   -13.60     4.95
1275   <NA>     <NA>     <NA>     <NA>     <NA>
1276 Copyri ght 2024  Kenneth  R. Fren       ch
> ### Delete not useful info
> FF.raw<-FF.raw[-1175:-1277,]
> ### Rename
> names(FF.raw)<-paste(c("text.date","RmxRf","SMB","HML","Rf"))
> head(FF.raw)
  text.date    RmxRf      SMB      HML       Rf
1    192607     2.96    -2.56    -2.43     0.22
2    192608     2.64    -1.17     3.82     0.25
3    192609     0.36    -1.40     0.13     0.23
4    192610    -3.24    -0.09     0.70     0.32
5    192611     2.53    -0.10    -0.51     0.31
6    192612     2.62    -0.03    -0.05     0.28
> tail(FF.raw)
     text.date    RmxRf      SMB      HML       Rf
1169    202311     8.84    -0.02     1.64     0.44
1170    202312     4.87     6.34     4.93     0.43
1171    202401     0.70    -5.09    -2.38     0.47
1172    202402     5.06    -0.24    -3.49     0.42
1173    202403     2.83    -2.49     4.19     0.43
1174    202404    -4.67    -2.39    -0.51     0.47
>
>
> ### Delete first column
> FF.raw<-FF.raw[,-1]
> ### Change type and percentage
> FF.raw$RmxRf<-as.numeric(as.character(FF.raw$RmxRf))/100
> FF.raw$Rf<-as.numeric(as.character(FF.raw$Rf))/100
> FF.raw$SMB<-as.numeric(as.character(FF.raw$SMB))/100
> FF.raw$HML<-as.numeric(as.character(FF.raw$HML))/100
> ### Create sequence of date object type Date
> FF.raw$FF.date<-seq(as.Date("1926-07-01"),as.Date("2024-04-30"),by="months")
> ### Change to yearmon type
> FF.raw$FF.date<-as.yearmon(FF.raw$FF.date,"%Y-%m-%d")
> ### Output
> FF.raw[c(1:3,nrow(FF.raw)),]
       RmxRf     SMB     HML     Rf  FF.date
1     0.0296 -0.0256 -0.0243 0.0022 7月 1926
2     0.0264 -0.0117  0.0382 0.0025 8月 1926
3     0.0036 -0.0140  0.0013 0.0023 9月 1926
1174 -0.0467 -0.0239 -0.0051 0.0047 4月 2024
>
```

Step 3: Subset Fama-French Data to Relevant Time Period

```
> ### Subset date period
> FF.data<-subset(FF.raw,FF.raw$FF.date>="2011-01-01" &FF.raw$FF.date<="2013-12-31")
> ### Output
> FF.data[c(1:3,nrow(FF.data)),]
       RmxRf     SMB     HML     Rf   FF.date
1015 0.0199 -0.0250  0.0083 0.0001  1月 2011
1016 0.0349  0.0153  0.0127 0.0001  2月 2011
1017 0.0046  0.0254 -0.0183 0.0001  3月 2011
1050 0.0281 -0.0046 -0.0002 0.0000 12月 2013
>
```

Step 4: Combine Portfolio Returns Data and Fama-French Data

```
> ### Not too many digits
> options(digits=3)
> ### Combine portfolio return and FF data
> FF.data<-cbind(FF.data,data.frame(port))
> ### Output
> FF.data[c(1:3,nrow(FF.data)),]
      RmxRf     SMB     HML     Rf   FF.date  X      date port.ret
1015 0.0199 -0.0250  0.0083 0.0001  1月 2011  1  1月 2011  0.19984
1016 0.0349  0.0153  0.0127 0.0001  2月 2011  2  2月 2011 -0.00931
1017 0.0046  0.0254 -0.0183 0.0001  3月 2011  3  3月 2011 -0.04618
1050 0.0281 -0.0046 -0.0002 0.0000 12月 2013 36 12月 2013  0.02799
> 

> ### define rownames
> rownames(FF.data)<-seq(1,nrow(FF.data))
> ### Change date format
> FF.data$date<-format(FF.data$date,"%Y-%m")
> ### excess return = portfolio return - risk-free asset value
> FF.data$exret<-FF.data$port.ret-FF.data$Rf
> ### output
> FF.data[c(1:3,nrow(FF.data)),]
    RmxRf     SMB     HML     Rf   FF.date  X    date port.ret    exret
1  0.0199 -0.0250  0.0083 0.0001  1月 2011  1 2011-01  0.19984  0.19974
2  0.0349  0.0153  0.0127 0.0001  2月 2011  2 2011-02 -0.00931 -0.00941
3  0.0046  0.0254 -0.0183 0.0001  3月 2011  3 2011-03 -0.04618 -0.04628
36 0.0281 -0.0046 -0.0002 0.0000 12月 2013 36 2013-12  0.02799  0.02799
> 
```

Step 5: Run Regression Using Fama-French Factors

```
> ### Run the regression of the equation
> FF.reg<-lm(FF.data$exret~RmxRf+SMB+HML,data=FF.data)
> ### Get summary of regression
> summary(FF.reg)

Call:
lm(formula = FF.data$exret ~ RmxRf + SMB + HML, data = FF.data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.08007 -0.01969 -0.00351  0.01866  0.16255

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.000854   0.007853   -0.11     0.91
RmxRf        1.153844   0.250189    4.61 0.000061 ***
SMB         -0.823385   0.519533   -1.58     0.12
HML         -0.662112   0.511702   -1.29     0.20
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.043 on 32 degrees of freedom
Multiple R-squared:  0.414,     Adjusted R-squared:  0.359
F-statistic: 7.52 on 3 and 32 DF,  p-value: 0.000605

> 
```

Now, we can run a regression based on Eq to find the parameter estimates. The FF model shows that only the excess market return (RmxRf) is the only coefficient that is statistically significant. However, the model has an F-statistic that is highly significant.

Compare Fama-French Results with CAPM Results

```
> ### CAPM regression
> CAPM.reg<-lm(exret~RmxRf,data=FF.data)
> ### Summary
> summary(CAPM.reg)

Call:
lm(formula = exret ~ RmxRf, data = FF.data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.08729 -0.01458 -0.00414  0.01595  0.18005

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.00262    0.00783    0.33  0.73985
RmxRf        0.85794    0.20303    4.23  0.00017 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0441 on 34 degrees of freedom
Multiple R-squared:  0.344,     Adjusted R-squared:  0.325
F-statistic: 17.9 on 1 and 34 DF,  p-value: 0.000169

>
```

We can create a small summary table comparing the FF and CAPM beta, p-value of the beta, and adjusted R-squared. We can do this by combining the rbind and cbind commands.

```
> ### Use rbind and cbind command to see beta, p and R-squared data for two factor models
> betas<-rbind(cbind(summary(FF.reg)$coefficient[2],summary(FF.reg)$coefficient[14],summary(FF.reg)$adj.r.square
d),cbind(summary(CAPM.reg)$coefficient[2],summary(CAPM.reg)$coefficient[8],summary(CAPM.reg)$adj.r.squared))
> betas
      [,1]      [,2]  [,3]
[1,] 1.154 0.0000612 0.359
[2,] 0.858 0.0001688 0.325
> colnames(betas)<-paste(c("Beta","p-Value","Adj. R-Squared"))
> rownames(betas)<-paste(c("Fama-French","CAPM"))
> betas
             Beta   p-Value Adj. R-Squared
Fama-French 1.154 0.0000612         0.359
CAPM        0.858 0.0001688         0.325
> options(digits=7)
>
```

Fama-French’s Beta here is more than 1, which is not a relatively good defensive factor model, however, the calculation cannot just be ERF multiplying beta here since it is a more complicated model. But its R squared is bigger.

5.5 Event Studies

An event study tests the impact of new information on a firm’s stock price. As such, event studies presume that markets are semi-strong form efficient.

The foundation of the event study is based on the Capital Asset Pricing Model (CAPM). An event study estimates the impact of a specific event on a company or a group of companies by looking at the firm's stock price performance.

The event study uses the historical relationship between the firm's stock return and one or more independent variables.

The independent variables are often chosen to be a proxy for the market and/or the firm's industry.

Based on this historical relationship, an event study predicts a return for the company on the event date. An abnormal return is then calculated as the difference between the firm's actual return on the event date and the predicted return on the event date.

As the name suggests, an event study requires an "event" to be identified. This "event" could be something that affects a particular firm (e.g., corporate earnings announcements) or something that affects an entire industry (e.g., passing of new legislation). Moreover, some events are not reported in isolation (i.e., events are announced together with other information that could also affect the firm's stock price).

Therefore, the hypothesis we test in an event study depends on the specifics of the event and a more involved analysis may be necessary to determine the effect of the particular "event" we intend to study. Put differently, we cannot simply apply a cookie-cutter approach to performing event studies, as the underlying economics may not match a purely mechanical application of event studies. This is truer for single-firm event studies, because we cannot benefit from averaging our results across multiple firms.

To conduct an event study, we need to identify two time periods. The first time period is the day or days around which the effects of an event has occurred. We call this the event window. The length of the event window depends on the type of event being studied. In the US, strong empirical evidence exists to show that stock prices react very quickly, usually within minutes, of material news announcements. Therefore, a one-day event window is likely sufficient when the timing of the event can be clearly identified. In some instances, news may be leaked days or weeks prior to the official announcement. This can be the case for some mergers. In this case, the event window can be extended to longer periods to fully capture the effect of the event.

Once the event window is defined, we can then select the appropriate *estimation period* from which we estimate the parameters under "normal" conditions unaffected by the event. The estimation procedure uses a market model of the form

$$r_{i,t} = \alpha + \beta r_{m,t} + \epsilon_{i,t}, \tag{5.5}$$

where $r_{i,t}$ is the return on day t of the subject firm i and $r_{m,t}$ is the return on day t of the market proxy m. It is typical to calculate the returns in an event study using log returns, so $r_{i,t} = \ln(P_{i,t}/P_{1,t-1})$ and $r_{m,t} = \ln(P_{m,t}/P_{m,t-1})$. However, using arithmetic returns is also used in practice. Although the capital asset pricing model (CAPM) can also be used instead of Eq. (5.5), the market model has the advantage of not needing to use the "true" market portfolio and allows us to use historical returns rather than expected returns. A common choice for the market proxy is the S&P 500 Index or another broad-based index.

Note that what constitutes "normal" depends on the event being studied. For example, if we are studying earnings announcements, then placing the estimation period prior to the event window may be appropriate. However, if we are studying an event immediately after a large merger that changes the underlying economics of the firm, then placing the estimation period after the event window may be better. Further, one must also be cognizant of whether the volatility of the abnormal return is significantly different due to non-event factors. For example, we likely will find many significant days if we analyze an event during the 2008/2009 financial crisis when we choose an estimation period before or after the crisis.

Typically, when daily data is used, a 6-month or 1-year period is sufficient. Note that the choice of the length of the estimation period is a trade-off between including more data with the risk of including less relevant data to predict the performance of the stock during the event window.

Since we are using statistical methods to estimate the abnormal return, not all non-zero abnormal returns should be interpreted as a reaction to the disclosed information. The reason is that we have to account for the normal volatility of the firm's stock return. For example, a highly volatile stock may move up or down 5–10 % in any given day, which means that a 10 abnormal return for that stock cannot be discerned from the normal volatility of the stock. Hence, we have to test whether the abnormal return is sufficiently large that it can be considered statistically significant.

$$t_{i,t} = \frac{\varepsilon_{i,t}}{\sigma},$$

An approach to determining statistical significance of the abnormal return is to find the corresponding p-value of the t-statistic, which is based on the number of degrees of freedom. The p-value will tell us the smallest possible significance level for which the t-statistic is considered statistically significant. In other words, for a sufficiently large number of observations, a t-statistic of 1.65 will have a 2-tail p-value of 0.10. This means that the abnormal return is statistically significant at the 10 % level. Other conventional levels of significance are equal to 1 and 5 %.

5.5.1 Example: Netflix July 2013 Earnings Announcement

After markets closed on July 22, 2013, Netflix announced earnings that more than quadrupled from a year ago but its new subscriber numbers for the second quarter of 630,000 failed to meet investors' expectations of 800,000 new subscribers.5 Netflix stock dropped 4.5 % on July 23, 2013, the first full trading day on which investors could have reacted to Netflix's information. News reports attributed the drop to the miss in new subscriber numbers. A 4.5 % drop seems huge, but we must put this in the context of NFLX stock volatility, which may be high as its stock tripled from its price 1 year before the announcement. Therefore, we can use an event study to determine whether the price reaction on July 23, 2013 can be distinguished from normal volatility in Netflix stock.

Step 1: Identify the Event and the Event Window

The event is the July 22, 2013 Netflix earnings announcement that was made after markets closed.

As such, the first full trading day investors could react to this news would be on July 23, 2013. We therefore use a one-day Event Window of July 23, 2013.

Step 2: Identify the Estimation Period

A one calendar year period ending the day before the Event Window as our Estimation Period.

Our Estimation Period uses returns from July 23, 2012 to July 22, 2013.

Step 3: Import Netflix Data From Yahoo Finance From July 20, 2012 to July 23, 2013

Since July 21 and 22, 2012 are non-trading days, we need the price from July 20, 2012.

```
> ### NFLX Data from 2012-07-20 to 2013-07-23
> data.NFLX<-read.csv("NFLX Yahoo (event study).csv",header=TRUE)
> data.NFLX<-data.NFLX[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.NFLX$Date,format="%Y-%m-%d")
> data.NFLX<-cbind(date, data.NFLX[,-1])
> data.NFLX<-data.NFLX[order(data.NFLX$date),]
> library(xts)
> firm<-xts(data.NFLX[,2:7],order.by=data.NFLX[,1])
> firm[c(1:3,nrow(firm)),]
               Open     High      Low    Close   Volume Adj.Close
2012-07-20 11.80857 11.93143 11.64286 11.68857 24369800  11.68857
2012-07-23 11.53286 11.60286 11.21429 11.42000 30965200  11.42000
2012-07-24 11.58143 11.78143 11.11571 11.48429 64700300  11.48429
2013-07-23 35.91429 37.46143 35.17143 35.75143 76829200  35.75143
> names(firm)<-paste(c("Firm.Open","Firm.High","Firm.Low","Firm.Close","Firm.Volume","Firm.Adjusted"))
> firm[c(1:3,nrow(firm)),]
           Firm.Open Firm.High Firm.Low Firm.Close Firm.Volume Firm.Adjusted
2012-07-20  11.80857  11.93143 11.64286   11.68857    24369800      11.68857
2012-07-23  11.53286  11.60286 11.21429   11.42000    30965200      11.42000
2012-07-24  11.58143  11.78143 11.11571   11.48429    64700300      11.48429
2013-07-23  35.91429  37.46143 35.17143   35.75143    76829200      35.75143
>
```

Step 4: Import SPDR S&P 500 Index ETF Data From Yahoo Finance From July 22, 2012 to July 23, 2013

```
> ### SPY Market Index
> data.SPY<-read.csv("SPY Yahoo (event study).csv",header=TRUE)
> data.SPY<-data.SPY[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.SPY$Date,format="%Y-%m-%d")
> data.SPY<-cbind(date, data.SPY[,-1])
> data.SPY<-data.SPY[order(data.SPY$date),]
> market<-xts(data.SPY[,2:7],order.by=data.SPY[,1])
> market[c(1:3,nrow(market)),]
              Open   High    Low  Close    Volume Adj.Close
2012-07-20 136.95 137.16 136.32 136.47 142904500  109.5765
2012-07-23 134.47 136.38 133.84 135.09 145210900  108.4684
2012-07-24 135.19 135.25 133.03 133.93 173301200  107.5370
2013-07-23 169.80 169.83 169.05 169.14  80829700  138.8440
> names(market)<-paste(c("Mkt.Open","Mkt.High","Mkt.Low","Mkt.Close","Mkt.Volume","Mkt.Adjusted"))
> market[c(1:3,nrow(market)),]
           Mkt.Open Mkt.High Mkt.Low Mkt.Close Mkt.Volume Mkt.Adjusted
2012-07-20   136.95   137.16  136.32    136.47  142904500     109.5765
2012-07-23   134.47   136.38  133.84    135.09  145210900     108.4684
2012-07-24   135.19   135.25  133.03    133.93  173301200     107.5370
2013-07-23   169.80   169.83  169.05    169.14   80829700     138.8440
>
```

Step 5: Combine the Two Data Objects

```
> ### combine firm and market adj price
> data.all<-merge(firm[,6],market[,6])
> data.all[c(1:3,nrow(data.all)),]
           Firm.Adjusted Mkt.Adjusted
2012-07-20      11.68857     109.5765
2012-07-23      11.42000     108.4684
2012-07-24      11.48429     107.5370
2013-07-23      35.75143     138.8440
>
```

Step 6: Calculate NFLX and SPY Returns

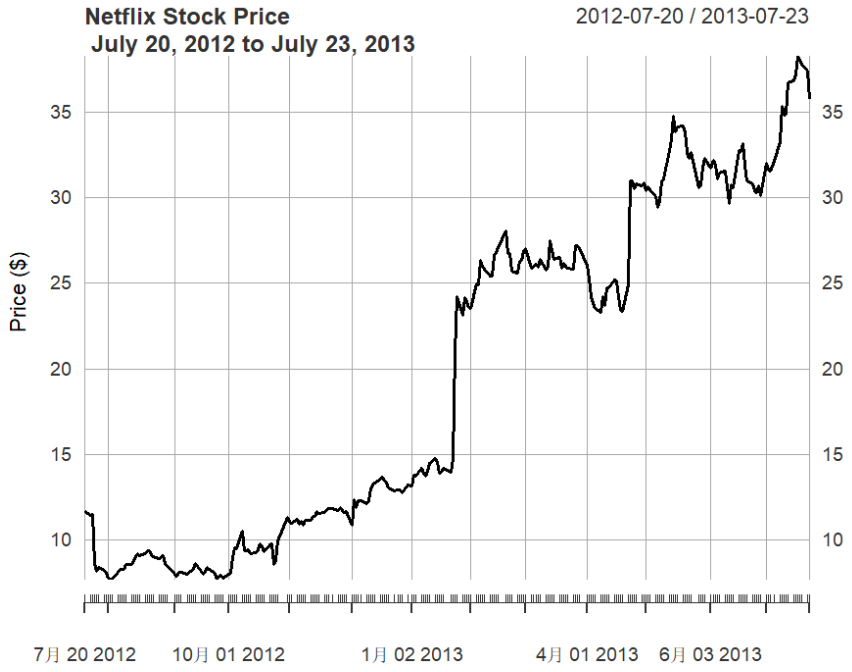
```
> ### log return = difference of log of adj price and then percentage it
> library(quantmod)
> data.all$Firm.Ret<-diff(log(data.all$Firm.Adjusted))*100
> data.all$Mkt.Ret<-diff(log(data.all$Mkt.Adjusted))*100
> data.all[c(1:3,nrow(data.all)),]
           Firm.Adjusted Mkt.Adjusted   Firm.Ret    Mkt.Ret
2012-07-20      11.68857     109.5765         NA         NA
2012-07-23      11.42000     108.4684 -2.3245323 -1.0163915
2012-07-24      11.48429     107.5370  0.5613462 -0.8623598
2013-07-23      35.75143     138.8440 -4.5691318 -0.2126256
>
```

Step 7: Perform Market Model Regression During the Estimation Period

```
> ### Only remain firm and market return
> est.per<-data.all[c(-1,-nrow(data.all)),3:4]
> est.per[c(1:3,nrow(est.per)),]
              Firm.Ret     Mkt.Ret
2012-07-23  -2.3245323 -1.01639151
2012-07-24   0.5613462 -0.86235982
2012-07-25 -28.7889392  0.02240931
2013-07-22  -0.9951877  0.19490012
> ### Run the least square regression
> mkt.model<-lm(est.per$Firm.Ret~est.per$Mkt.Ret)
> summary(mkt.model)

Call:
lm(formula = est.per$Firm.Ret ~ est.per$Mkt.Ret)

Residuals:
    Min      1Q  Median      3Q     Max
-29.182  -1.858  -0.500   1.480  34.826

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       0.3706     0.2914   1.272   0.2046
est.per$Mkt.Ret   0.9928     0.3831   2.591   0.0101 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.57 on 248 degrees of freedom
Multiple R-squared:  0.02636,   Adjusted R-squared:  0.02244
F-statistic: 6.715 on 1 and 248 DF,  p-value: 0.01013

>
```

Step 8: Calculate Abnormal Return, t-Statistic, and p-Value for Dates in Event Window

```
> ### Summary of all statistics we need
> ### True firm and market return
> event.window<-data.all[nrow(data.all),3:4]
> ### Output
> event.window
            Firm.Ret    Mkt.Ret
2013-07-23 -4.569132 -0.2126256
> ### prediction = intercept + regression beta * true market return
> event.window$Pred.Ret<-summary(mkt.model)$coefficients[1]+summary(mkt.model)$coefficients[2]*event.window$Mkt.Ret
> ### Abnormal return = True firm return - Predicted firm return
> event.window$Ab.Ret<-event.window$Firm.Ret-event.window$Pred.Ret
> ### t statistic = abnormal return / sigma
> event.window$tStat<-event.window$Ab.Ret/summary(mkt.model)$sigma
> ### P value calculated by above t statistic and degree of freedom = n - 2
> event.window$pval<-2*(1-pt(abs(event.window$tStat),df=nrow(est.per)-2))
> ### Not too many digits in Output
> options(digits=3)
> ### Output
> event.window
           Firm.Ret Mkt.Ret Pred.Ret Ab.Ret tStat  pval
2013-07-23    -4.57  -0.213     0.16  -4.73 -1.03 0.302
> ### Return back
> options(digits=7)
>
```



Netflix's stock price tripled during the one calendar year period prior to the July 23, 2013 announcement, which caused the significant volatility in the stock relative to the market.

Although an estimation period of one calendar year for daily returns data is typical, we may want to estimate the market model over a shorter period given that the period closer to the event window but after the jump may be more relevant in measuring the price impact of the event. We now continue from our previous calculations and create a second set of calculations.

Step 9: Identify the Date of the Jump in January 2013

```
> ### Identify jumping date
> subset(est.per,index(est.per)>="2013-01-01" & index(est.per)<="2013-01-31")
             Firm.Ret     Mkt.Ret
2013-01-02 -0.6283867  2.53070996
2013-01-03  4.8577897 -0.22622889
2013-01-04 -0.6335317  0.43823946
2013-01-07  3.2998177 -0.27366820
2013-01-08 -2.0778940 -0.28814579
2013-01-09 -1.2948821  0.25388486
2013-01-10  2.1557196  0.79180551
2013-01-11  3.3020211 -0.00678967
2013-01-14  2.1100685 -0.06802167
2013-01-15 -1.7159396  0.06802167
2013-01-16 -4.2281771 -0.01360582
2013-01-17  0.2254361  0.64391903
2013-01-18  1.4933990  0.22278202
2013-01-22 -1.3808748  0.53787257
2013-01-23  5.4223297  0.16079092
2013-01-24 35.2229644  0.02681193
2013-01-25 14.3727089  0.56062009
2013-01-28 -4.4931742 -0.11986049
2013-01-29  4.2333423  0.39241579
2013-01-30 -0.8431806 -0.39241579
2013-01-31 -1.4777724 -0.24686185
>
```

The output above shows that Netflix's stock price increased by 42 and 15 % on January 24, 2013 and January 25, 2013, respectively. For purpose of this analysis, we will exclude all returns on or before January 25, 2013. Therefore, we start our alternative estimation period on January 28, 2013.

Step 10: Construct Series for Estimation Period Beginning January 28, 2013

```
> ### Subset again to the new estimation period
> est.per2<-subset(est.per,index(est.per)>="2013-01-28")
> est.per2[c(1,3,nrow(est.per2)),]
             Firm.Ret    Mkt.Ret
2013-01-28 -4.4931742 -0.1198605
2013-01-30 -0.8431806 -0.3924158
2013-07-22 -0.9951877  0.1949001
> nrow(est.per2)
[1] 122
>
```

As the output shows, there are 122 observations in our alternative estimation period. Although there are no strict rules as to what the minimum number of observations that should be used, 120 observations for daily data is reasonably sufficient.

Step 11: Calculate the Market Model Parameters of this Alternative Estimation Period

```
> ### Run the regression again
> mkt.model2<-lm(est.per2$Firm.Ret~est.per2$Mkt.Ret)
> ### Summary
> summary(mkt.model2)

Call:
lm(formula = est.per2$Firm.Ret ~ est.per2$Mkt.Ret)

Residuals:
    Min      1Q  Median      3Q     Max
-7.4156 -1.7377 -0.4334  1.0279 20.5984

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.2505     0.2962   0.846   0.3993
est.per2$Mkt.Ret    0.9928     0.3837   2.587   0.0109 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.24 on 120 degrees of freedom
Multiple R-squared:  0.05283,   Adjusted R-squared:  0.04493
F-statistic: 6.693 on 1 and 120 DF,  p-value: 0.01087

>
```

Step 12: Calculate Abnormal Return and Statistical Significance of the Event Date

```
> ### Again, like before summary of statistics
> ### Summary of all statistics we need
> ### True firm and market return
> event.window2<-data.all[nrow(data.all),3:4]
> ### Output
> event.window2
           Firm.Ret    Mkt.Ret
2013-07-23 -4.569132 -0.2126256
> ### prediction = intercept + regression beta * true market return
> event.window2$Pred.Ret<-summary(mkt.model2)$coefficients[1]+summary(mkt.model2)$coefficients[2]*event.window2$Mkt.Ret
> ### Abnormal return = True firm return - Predicted firm return
> event.window2$Ab.Ret<-event.window2$Firm.Ret-event.window2$Pred.Ret
> ### t statistic = abnormal return / sigma
> event.window2$tStat<-event.window2$Ab.Ret/summary(mkt.model2)$sigma
> ### P value calculated by above t statistic and degree of freedom = n - 2
> event.window2$pval<-2*(1-pt(abs(event.window2$tStat),df=nrow(est.per2)-2))
> ### Not too many digits in Output
> options(digits=3)
> ### Output
> event.window2
           Firm.Ret Mkt.Ret Pred.Ret Ab.Ret tStat  pval
2013-07-23    -4.57  -0.213   0.0395  -4.61 -1.42 0.157
> ### Return back
> options(digits=7)
>
```

In many academic studies, the final event study result is often an average across a large sample. This averaging has the benefit of reducing the individual firm-specific effects on the abnormal returns and it may be less important to have very precise inputs and assumptions. However, when dealing with individual firm event studies, we have to be more careful in the

inputs and assumptions that we make.

5.6 Further Reading

The original paper on the CAPM is Sharpe [12]. In the above example, we calculated the beta using 36 months of returns data. Another common method is to use 2 weeks of weekly returns data

The first trade-off is that higher frequency (i.e., daily returns data) is more susceptible to asynchronous trading issues than lower frequency (i.e., monthly data).

The second trade-off is that shorter estimation periods may be more relevant to the current beta but may be more affected by extreme events than longer estimation periods, but longer estimation periods may include older and less relevant data.

The original paper on the FF Model is Fama and French [6]. Much more data compiled for the FF Model can be found in Ken French's website. An excellent treatment of factor models can be found in Damodaran [5] and Koller et al. [9].

The seminal paper on event studies is Fama et al. [7]. A good background on event studies can be found in Kothari and Warner [10]. The authors report over 575 published articles in five top finance journals made use of event studies from 1974 to 2000. A good discussion of the approach we use is in MacKinlay [11]. An alternative approach is to implement the event study in one-pass, by using a dummy variable to represent the abnormal return on event dates. This is discussed in Binder [2] and Karafiath [8]. Another variation of the event study is to augment the market model to include an industry proxy, such that the abnormal return can be interpreted as a return after accounting for market and industry factors. This model is commonly used in the context of securities litigation and is discussed in Crew et al. [4].

Chapter 6 Risk-Adjusted Portfolio Performance Measures

We should only expect higher returns if we take on more risk. Therefore, the performance of an investment should be taken in the context of the level of risk taken.

A more appropriate comparison would be to somehow normalize the returns from the different investments by their respectively risk levels. The complicating issue is that there is no single definition of returns as well as no single definition of risk.

In this chapter, we implement several risk-adjusted performance measures that differ in their definition of returns and/or risks.

In particular, we will look at the Sharpe Ratio, Roy's Safety-First Ratio, Treynor Ratio, Sortino Ratio, and Information Ratio. However, because of the differences in the definition of return and risk, these metrics may rank investments differently.

6.1 Portfolio and Benchmark Data

We first need to have portfolios whose performance we wish to analyze. However, for our simplified analysis, we will compare the performance of two portfolios: an Actual Portfolio, which is a proxy for our investment portfolio, and an Alternative Portfolio, which is a proxy for an investment alternative.

In addition, we need to set a benchmark portfolio for use with some metrics. The portfolios could either be our actual portfolios or existing securities we view as potential investment candidates. For some metrics, we need a benchmark portfolio.

Step 1: Import Actual Portfolio Data

However, for illustrative purposes, we will import monthly returns data from a hypothetical portfolio constructed using equal-weighted returns of AMZN, IBM, and NVDA with monthly rebalancing from January 2011 to December 2013.

```
> ### Import data from hypothetical Portfolio (Monthly)
> port<-read.csv("Hypothetical Portfolio (Monthly).csv",header=TRUE)
> port[c(1:3,nrow(port)),]
    X      date     port.ret
1   1  1月 2011  0.199844965
2   2  2月 2011 -0.009314655
3   3  3月 2011 -0.046175365
36 36 12月 2013  0.027989666
>
```

Step 2: Import Alternative Portfolio Data

We then construct monthly returns for the SPY from January 2011 to December 2013 using techniques we discuss in earlier chapters.

```
> ### Import SPY index data as alternative data
> SPY<-read.csv("SPY Yahoo.csv",header=TRUE)
> SPY = SPY[,c(1,2,3,4,5,7,6)]
> date<-as.Date(SPY$Date,format="%Y-%m-%d")
> SPY<-cbind(date, SPY[,-1])
> SPY<-SPY[order(SPY$date),]
> library(xts)
> SPY<-xts(SPY[,2:7],order.by=SPY[,1])
> names(SPY)<-paste(c("SPY.Open","SPY.High","SPY.Low","SPY.Close","SPY.Volume","SPY.Adjusted"))
> SPY[c(1:3,nrow(SPY)),]
           SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted
2010-12-31   125.53   125.87  125.33    125.75   91218900     97.95515
2011-01-03   126.71   127.60  125.70    127.05  138725200     98.96784
2011-01-04   127.33   127.37  126.19    126.98  137409700     98.91330
2013-12-31   184.07   184.69  183.93    184.69   86119900    153.17487
> SPY<-to.monthly(SPY)
> SPY[c(1:3,nrow(SPY)),]
          SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adjusted
12月 2010   125.53   125.87  125.33    125.75   91218900     97.95515
1月 2011    126.71   130.35  125.70    128.68 2860314700    100.23754
2月 2011    129.46   134.69  129.38    133.15 2820073500    103.71952
12月 2013   181.09   184.69  177.32    184.69 2232775900    153.17487
> SPY<-SPY[,6]
> library(quantmod)
> altport.ret<-Delt(SPY$SPY.Adjusted)
> names(altport.ret)<-paste("altport.ret")
> altport.ret[c(1:3,nrow(altport.ret)),]
          altport.ret
12月 2010          NA
1月 2011   0.02330040
2月 2011   0.03473728
12月 2013  0.02592700
>
```

Step 3: Import Benchmark Portfolio Data

We use the S&P 500 Index as our benchmark portfolio. The S&P 500 Index data is obtained from Yahoo Finance and is saved in a file labeled GSPC Yahoo.csv.

Again, keep in mind that GSPC index has been replaced by AAPL apple.com.

```
> ### Again, AAPL to substitute ^GSPC
> AAPL<-read.csv("AAPL Yahooo.csv",header=TRUE)
> AAPL = AAPL[,c(1,2,3,4,5,7,6)]
> date<-as.Date(AAPL$Date,format="%Y-%m-%d")
> AAPL<-cbind(date, AAPL[,-1])
> AAPL<-AAPL[order(AAPL$date),]
> AAPL<-xts(AAPL[,2:7],order.by=AAPL[,1])
> names(AAPL)<-paste(c("AAPL.Open","AAPL.High","AAPL.Low","AAPL.Close","AAPL.Volume","AAPL.Adjusted"))
> AAPL[c(1:3,nrow(AAPL)),]
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2010-12-31  11.53393  11.55286 11.47536   11.52000   193508000      9.739614
2011-01-03  11.63000  11.79500 11.60143   11.77036   445138400      9.951281
2011-01-04  11.87286  11.87500 11.71964   11.83179   309080800     10.003214
2013-12-31  19.79179  20.04571 19.78571   20.03643   223084400     17.519623
> AAPL<-to.monthly(AAPL)
> AAPL[c(1:3,nrow(AAPL)),]
         AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
12月 2010  11.53393  11.55286 11.47536   11.52000   193508000      9.739614
1月 2011   11.63000  12.45000 11.60143   12.11857 10841535600     10.245680
2月 2011   12.18929  13.03214 12.06143   12.61464  9295949600     10.665081
12月 2013  19.92857  20.54071 19.24286   20.03643  7057397200     17.519623
> benchmark.ret<-Delt(AAPL$AAPL.Adjusted)
> names(benchmark.ret)<-paste("bench.ret")
> benchmark.ret[c(1:3,nrow(benchmark.ret)),]
           bench.ret
12月 2010          NA
1月 2011  0.051959554
2月 2011  0.040934423
12月 2013 0.008901618
> 
```

Step 4: Combine Portfolios and Benchmark Data

```
> ### Fewer useful digits
> options(digits=3)
> ### cbind all three returns of portfolio
> port<-cbind(port[,-1],data.frame(altport.ret[-1,],data.frame(benchmark.ret[-1,])))
> ### Output
> port[c(1:3,nrow(port)),]
               date port.ret altport.ret bench.ret
1月 2011   1月 2011  0.19984     0.02330    0.0520
2月 2011   2月 2011 -0.00931     0.03474    0.0409
3月 2011   3月 2011 -0.04618     0.00012   -0.0133
12月 2013 12月 2013  0.02799     0.02593    0.0089
> 
```

Step 5: Rename Index Values to Observation Number

```
> ### Re-index numerically
> rownames(port)<-seq(1,nrow(port),1)
> port[c(1:3,nrow(port)),]
        date port.ret altport.ret bench.ret
1   1月 2011  0.19984     0.02330    0.0520
2   2月 2011 -0.00931     0.03474    0.0409
3   3月 2011 -0.04618     0.00012   -0.0133
36 12月 2013  0.02799     0.02593    0.0089
> 
```

We will use the data object port in our subsequent analysis.

6.2 Sharpe Ratio

The Sharpe Ratio compares the excess return of a portfolio relative to the risk-free rate with the portfolio's standard deviation. That is,

$$Sharpe_p = \frac{E(r_p) - r_f}{\sigma_p}, \tag{6.1}$$

where $E(r_p)$ is the annualized average return of portfolio p, r_f is the annual risk-free rate, and σ_p is the annualized standard deviation of portfolio p. The higher the Sharpe Ratio, the higher the risk-adjusted performance of the portfolio. Below, we show the steps in calculating the Sharpe Ratio for the Actual Portfolio and Alternative Portfolio.

Step 1: Identify the Frequency of the Returns and the Length of the Estimation Period

According to the Morningstar Investment Glossary, we can calculate the Sharpe Ratio using the past 36 months of returns. We follow this definition for our example.

Step 2: Identify the Risk-Free Rate

Recall we used the 3-Month Treasury in our CAPM calculation in Chap. 5. For consistency, we will also use the 3-Month Constant Maturity Treasury to calculate the Sharpe Ratio. On December 31, 2013, the 3-Month T-Bill had a yield of 0.07 %.

```
> ### risk-free rate on 2013-12-31
> Rf=0.0007
> Rf
[1] 7e-04
>
```

Step 3: Annualize Monthly Portfolio Returns and Standard Deviation

```
> ### Annual return = monthly return's mean * 12
> annual.port.ret<-mean(port$port.ret)*12
> annual.port.ret
[1] 0.17
>
> ### Annual sd = monthly sd * 12
> annual.port.sd<-sd(port$port.ret)*sqrt(12)
> annual.port.sd
[1] 0.186
>
```

Step 4: Calculate Sharpe Ratio

```
> ### Sharpe ratio = (annual return - risk-free) / annual sd
> Sharpe.port<-(annual.port.ret-Rf)/annual.port.sd
> Sharpe.port
[1] 0.91
>
```

Step 5: Repeat Steps 1–4 for the Alternative Portfolio

```
> ### Repeat above steps for Alternative Portfolio
> annual.altport.ret<-mean(port$altport.ret)*12
> annual.altport.ret
[1] 0.157
> annual.altport.sd<-sd(port$altport.ret)*sqrt(12)
> annual.altport.sd
[1] 0.121
> Sharpe.altport<-(annual.altport.ret-Rf)/annual.altport.sd
> Sharpe.altport
[1] 1.29
>
```

The Actual Portfolio has a Sharpe Ratio of 0.91, which is higher than the Sharpe Ratio of the Alternative Portfolio of 1.29. As such, based on the Sharpe Ratio, the Actual Portfolio yields a lower risk-adjusted return.

6.3 Roy's Safety-First Ratio

Roy's Safety First (SF) Ratio makes a slight modification to the Sharpe Ratio. Specifically, instead of using the risk-free rate, Roy's SF Ratio instead uses a target or minimum acceptable return to calculate the excess return. That is,

$$RoySF_p = \frac{E(r_p) - MAR}{\sigma_p}, \tag{6.2}$$

where *MAR* is the minimum acceptable return and all other variables are defined similarly to Eq. (6.1). Again, the higher the Roy's Safety First Ratio, the better the risk-adjusted performance of the portfolio.

Step 1: Determine the MAR

Based on the December 2013 Livingston Survey, the 2012–2013 inflation rate is 1.5 %. We use this as our proxy for MAR and calculate Roy's SF Ratio for our portfolio and alternative portfolio.

```
> ### Minimum Acceptable Return
> mar<-0.015
> mar
[1] 0.015
>
```

Step 2: Calculate the Annual Portfolio Return and Standard Deviation

```
> ### Recall in R memory
> annual.port.ret
[1] 0.17
> annual.port.sd
[1] 0.186
>
```

Step 3: Calculate the Roy's Safety-First Ratio for the Actual Portfolio

```
> ### Roy SF ratio = (annual return - minimum acceptable return) / annual sd
> Roy.SF.port<-(annual.port.ret-mar)/annual.port.sd
> Roy.SF.port
[1] 0.833
>
```

Step 4: Repeat Steps 1–3 for the Alternative Portfolio

```
> ### Repeat above steps for Alternative Portfolio
> annual.altport.ret
[1] 0.157
> annual.altport.sd
[1] 0.121
> Roy.SF.altport<-(annual.altport.ret-mar)/annual.altport.sd
> Roy.SF.altport
[1] 1.17
>
```

The Actual Portfolio has a Roy's SF Ratio of 0.833, which is higher than the Roy's SF Ratio of the Alternative Portfolio of 1.17. As such, based on the Roy's SF Ratio, the Actual Portfolio yields a lower risk-adjusted return.

6.4 Treynor Ratio

The Treynor Ratio modifies the denominator in the Sharpe Ratio to use beta instead of standard deviation in the denominator. That is,

$$Treynor_p = \frac{E(r_p) - r_f}{\beta_p},$$

where βp is the beta of the portfolio and the rest of the terms are defined similarly to above equation. The higher the Treynor Ratio, the better the risk-adjusted performance of the portfolio.

Step 1: Identify the Frequency of Returns and Length of the Estimation Period

Using beta instead of standard deviation implies that the Treynor Ratio only considers systematic risk while the Sharpe Ratio considers total risk (i.e., systematic and firm-specific risk). Therefore, the ratios could differ if the ratio of systematic risk to total risk is different in the two funds we are comparing.

Step 2: Import Actual Portfolio Returns

```
> ### recall annual port ret
> annual.port.ret
[1] 0.17
>
```

Step 3: Obtain the Relevant Risk-Free Rate

```
> ### Not in scientific notation
> options(scipen=100)
> ### Risk-free
> Rf
[1] 0.0007
>
```

Step 4: Estimate the Beta of the Actual Portfolio

```
> ### lm regression to find beta
> port[c(1:3,nrow(port)),]
        date port.ret altport.ret bench.ret
1   1月 2011  0.19984     0.02330    0.0520
2   2月 2011 -0.00931     0.03474    0.0409
3   3月 2011 -0.04618     0.00012   -0.0133
36 12月 2013  0.02799     0.02593    0.0089
> reg<-lm(port$port.ret~port$bench.ret)
> port.beta<-summary(reg)$coefficients[2]
> port.beta
[1] 0.199
>
```

Step 5: Calculate the Treynor Ratio for the Actual Portfolio

```
> ### Tr ratio = (annual port ret - risk-free) / beta
> Treynor.port<-(annual.port.ret-Rf)/port.beta
> Treynor.port
[1] 0.852
>
```

Step 6: Repeat Steps 1–5 for the Alternative Portfolio

```
> ### Again, repeat above steps for alternative portfolio
> annual.altport.ret
[1] 0.157
> Rf
[1] 0.0007
> reg.altport<-lm(port$altport.ret~port$bench.ret)
> beta.altport<-summary(reg.altport)$coefficients[2]
> beta.altport
[1] 0.14
> Treynor.altport<-(annual.altport.ret-Rf)/beta.altport
> Treynor.altport
[1] 1.12
>
```

The Actual Portfolio has a Treynor Ratio of 0.852, which is higher than the Treynor Ratio of the Alternative Portfolio of 1.12. As such, based on the Treynor Ratio, the Actual Portfolio yields a lower risk-adjusted return.

6.5 Sortino Ratio

Given our implementation of Roy's SF Ratio using a minimum acceptable return (MAR), the Sortino Ratio has the same numerator Roy's SF Ratio in above Equation. However, the denominator measures only the deviation of values lower than the MAR, which is referred to as Downside Deviation (DD). That is,

$$Sortino_p = \frac{E(r_p) - MAR}{DD_p},$$

where E(rp) is the average return of portfolio p, MAR is the minimum acceptable return, and DDp is the downside deviation. The higher the Sortino Ratio, the better the risk-adjusted performance of the portfolio.

Step 1: Determine the Period MAR

```
> ### MAR
> mar
[1] 0.015
> ### Period MAR
> period.mar<-mar/12
> period.mar
[1] 0.00125
>
```

Step 2: Identify Monthly Returns in Actual Portfolio That are Less Than the MAR

```
> ### if port ret < period MAR, dummy = 1, else 0.
> port[c(1:3,nrow(port)),]
         date port.ret altport.ret bench.ret
1    1月 2011  0.19984     0.02330    0.0520
2    2月 2011 -0.00931     0.03474    0.0409
3    3月 2011 -0.04618     0.00012   -0.0133
36 12月 2013  0.02799     0.02593    0.0089
> downside.port<-port
> downside.port$dummy<-ifelse(port$port.ret<period.mar,1,0)
> downside.port[c(1:3,nrow(downside.port)),]
         date port.ret altport.ret bench.ret dummy
1    1月 2011  0.19984     0.02330    0.0520     0
2    2月 2011 -0.00931     0.03474    0.0409     1
3    3月 2011 -0.04618     0.00012   -0.0133     1
36 12月 2013  0.02799     0.02593    0.0089     0
>
```

Step 3: Extract Monthly Returns That are Less Than the MAR

```
> ### Select dummy ==1
> downside.port<-subset(downside.port,downside.port$dummy==1)
> downside.port[c(1:3,nrow(downside.port)),]
       date  port.ret altport.ret bench.ret dummy
2   2月 2011 -0.009315     0.03474   0.04093     1
3   3月 2011 -0.046175     0.00012  -0.01331     1
5   5月 2011  0.000417    -0.01121  -0.00657     1
32 8月 2013 -0.033792    -0.02999   0.08377     1
>
```

Step 4: Calculate Downside Deviation

downside.port only contains returns that are less than the period MAR. The standard deviation of these returns is called the downside deviation. We then annualize the downside deviation to make it compatible with the other inputs to the Sortino Ratio.

```
> ### downside deviation annually
> dd.port<-sd(downside.port$port.ret)*sqrt(12)
> dd.port
[1] 0.0967
>
```

Step 5: Calculate Sortino Ratio

```
> ### Sortino ratio = ( annual ret - MAR ) / dd
> Sortino.port=(annual.port.ret-mar)/dd.port
> Sortino.port
[1] 1.6
>
```

Step 6: Repeat Steps 1–5 for the Alternative Portfolio

```
> ### Again for alternative portfolio
> downside.altport<-port
> downside.altport$dummy<-ifelse(port$altport.ret<period.mar,1,0)
> downside.altport[c(1:3,nrow(downside.altport)),]
        date port.ret altport.ret bench.ret dummy
1   1月 2011  0.19984     0.02330    0.0520     0
2   2月 2011 -0.00931     0.03474    0.0409     0
3   3月 2011 -0.04618     0.00012   -0.0133     1
36 12月 2013  0.02799     0.02593    0.0089     0
> downside.altport<-subset(downside.altport,downside.altport$dummy==1)
> downside.altport[c(1:3,nrow(downside.altport)),]
       date  port.ret altport.ret bench.ret dummy
3   3月 2011 -0.046175     0.00012  -0.01331     1
5   5月 2011  0.000417    -0.01121  -0.00657     1
6   6月 2011 -0.049808    -0.01687  -0.03496     1
32 8月 2013 -0.033792    -0.02999   0.08377     1
> dd.altport<-sd(downside.altport$altport.ret)*sqrt(12)
> dd.altport
[1] 0.0809
> Sortino.altport<-(annual.altport.ret-mar)/dd.altport
> Sortino.altport
[1] 1.76
>
```

The Actual Portfolio has a Sortino Ratio of 1.6, which is higher than the Sortino Ratio of the Alternative Portfolio of 1.76. As such, based on the Sortino Ratio, the Actual Portfolio yields a lower risk-adjusted return.

6.6 Information Ratio

The IR is the ratio of the portfolio alpha over the tracking error. The alpha of the portfolio is the annualized active return, which is the difference between the portfolio's return and the benchmark return (e.g., return of the S&P 500 Index). The tracking error is the annualized standard deviation of the active return.

Step 1: Calculate the Active Return of the Actual Portfolio

```
> ### Act ret port = port ret - bench ret
> port[c(1:3,nrow(port)),]
        date port.ret altport.ret bench.ret
1   1月 2011  0.19984     0.02330    0.0520
2   2月 2011 -0.00931     0.03474    0.0409
3   3月 2011 -0.04618     0.00012   -0.0133
36 12月 2013  0.02799     0.02593    0.0089
> Act.Ret.port<-port$port.ret-port$bench.ret
> head(Act.Ret.port)
[1]  0.14789 -0.05025 -0.03287  0.06753  0.00699 -0.01485
>
```

Step 2: Calculate Portfolio Alpha

```
> ### alpha port = mean of act ret port * 12
> alpha.port<-mean(Act.Ret.port)*12
> alpha.port
[1] -0.0606
>
```

Step 3: Calculate Tracking Error

```
> ### tract error = sd of active return of portfolio
> tracking.error.port<-sd(Act.Ret.port)*sqrt(12)
> tracking.error.port
[1] 0.276
>
```

Step 4: Calculate the Information Ratio

```
> ### info ratio = alpha / tract error
> IR.port<-alpha.port/tracking.error.port
> IR.port
[1] -0.219
>
```

Step 5: Repeat Steps 1–4 for the Alternative Portfolio

```
> ### Again, repeat for alternative portfolio against benchmark portfolio
> Act.Ret.altport<-port$altport.ret-port$bench.ret
> head(Act.Ret.altport)
[1] -0.02866 -0.00620  0.01343  0.02431 -0.00465  0.01809
> alpha.altport<-mean(Act.Ret.altport)*12
> alpha.altport
[1] -0.0733
> tracking.error.altport<-sd(Act.Ret.altport)*sqrt(12)
> tracking.error.altport
[1] 0.254
> IR.altport<-alpha.altport/tracking.error.altport
> IR.altport
[1] -0.288
>
```

6.7 Combining Results

```
> RARet<-rbind(cbind(Sharpe.port,Roy.SF.port,Treynor.port,Sortino.port,IR.port),
+ +             cbind(Sharpe.altport,Roy.SF.altport,Treynor.altport,Sortino.altport,IR.altport))
> RARet
     Sharpe.port Roy.SF.port Treynor.port Sortino.port IR.port
[1,]        0.91       0.833        0.852         1.60  -0.219
[2,]        1.29       1.171        1.118         1.76  -0.288
>
> ### Rename columns and rows meaningfully
> colnames(RARet)<-paste(c("Sharpe","Roy SF","Treynor","Sortino","Info Ratio"))
> rownames(RARet)<-paste(c("Portfolio","Alt Portfolio"))
> ### Output
> RARet
              Sharpe Roy SF Treynor Sortino Info Ratio
Portfolio       0.91  0.833   0.852    1.60     -0.219
Alt Portfolio   1.29  1.171   1.118    1.76     -0.288
>
> ### Return back setting of digits
> options(digits=7)
>
```

Recall that the rule for all these ratios is that the higher the ratio the better.

As we can see from the output above, our Actual Portfolio has a lower Sharpe Ratio, Roy SF Ratio, Treynor Ratio, and Sortino Ratio than the Alternative Portfolio. Looking specifically at the Sharpe and Treynor Ratios, we can see that our portfolio provides worse risk-adjusted returns regardless of whether we measure risk as total risk (i.e., standard deviation) or only systematic risk (i.e., beta).

Looking at the Sharpe Ratio and Roy SF Ratio, even considering a minimum acceptable return slightly higher than the risk-free rate does not change our results. Comparing the Roy SF Ratio and Sortino Ratio, we can see the difference in the two ratios is that the Roy SF Ratio uses standard deviation in the denominator while the Sortino Ratio uses the downside deviation (i.e., standard deviation of returns that fail to meet the MAR).

The results are consistent with the Actual Portfolio having many more returns that exceed the MAR that caused a lot more of the deviation, as we have quite a small downside deviation.

The results above for the IR suggests a different story. The IR implies that the manager of the Actual Portfolio performed worse relative to the risk assumed compared to the manager of the Alternative Portfolio. The active choice by the manager of our Actual Portfolio to select securities resulted in a higher alpha but also a correspondingly higher tracking error. The Alternative Portfolio had a lower alpha but also tracked our benchmark much closer.

6.8 Further Reading

Maginn et al. [2] contains a detailed discussion of the portfolio performance measures we discussed in this chapter. Applications of risk-adjusted performance measures for hedge funds can be found in Tran [6] and Gregoriou [1].

References

1. Gregoriou, G., Hubner, G., Papageorgiou, N., & Rouah, F. (2005). Hedge funds: Insights in performance measurement, risk analysis, and portfolio allocation. New Jersey: Wiley.

2. Maginn, J., Tuttle, D., Pinto, J., & McLeavey, D. (2007). Managing investment portfolios: A dynamic process (3rd ed). New Jersey: Wiley.

3. Markowitz, H. (1952). Portfolio selection. The Journal of Finance, 7, 77–91.

4. Sharpe, W. (1975). Adjusting for risk in portfolio performance. Journal of Portfolio Management, 1, 29–34.

5. Sharpe, W. (1994). The Sharpe ratio. Journal of Portfolio Management, 21, 49–58.

6. Tran, V. (2006). Evaluating hedge fund performance. New Jersey: Wiley.

Chapter 7 Markowitz Mean-Variance Optimization

In Markowitz [3], Harry Markowitz put forth a theory of how to use mean and variance to determine portfolios that offer the highest return for a given level of risk.

In this chapter, we show how to construct the Mean-Variance (MV) Efficient Frontier, which is the set of all portfolio generated by various combinations of the securities in the portfolio that yield the highest return for a given level of risk.

We begin by manually constructing the MV Efficient Frontier using two assets, so we can gain the intuition of what we are trying to accomplish. Then, we convert this manual approach for two assets to an approach that uses *quadratic programming*.

Since many investors (e.g., pension funds) do not or cannot short sell securities for various reasons, we implement the two-asset case not allowing short selling.

Then we extend the quadratic programming approach to show how we find the MV efficient portfolios when we have multiple securities. In the multiple asset scenario, we first continue with the no short sale case and then demonstrate the effect on the MV Efficient Frontier of allowing short selling.

We will see that allowing short-selling extends outward the MV Efficient Frontier such that we are able to achieve higher returns for a given level of risk.

7.1 Two Assets the "Long Way"

We first demonstrate the intuition of how to construct the MV Efficient Frontier using the case of a 2-asset portfolio and manually laying out each step of the calculation.

This approach is termed the "Long Way" because we lay out each step without the benefit of any mathematical optimization techniques.

In determining how much money to invest in each of the two securities, we may want to know what combination of these two securities yields the small est portfolio risk (i.e., the minimum variance portfolio) and we may want to know what combination of these two securities yields the highest Sharpe Ratio (i.e., the tangency portfolio).

We identify the set of portfolios that meet that criteria and these portfolios are called mean-variance ***efficient portfolios***.

Step 1: Import SPY and LAG Data from Yahoo Finance

```
> library(quantmod)
> library(xts)
> # Large-Cap Equity ETF Data
> data.SPY<-read.csv("SPY Yahoo.csv",header=TRUE)
> data.SPY<-data.SPY[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.SPY$Date,format="%Y-%m-%d")
> data.SPY<-cbind(date, data.SPY[,-1])
> data.SPY<-data.SPY[order(data.SPY$date),]
> library(xts)
> data.SPY<-xts(data.SPY[,2:7],order.by=data.SPY[,1])
> names(data.SPY)<-paste(c("SPY.Open","SPY.High","SPY.Low","SPY.Close","SPY.Volume","SPY.Adjusted"))
> library(quantmod)
> SPY.monthly<-to.monthly(data.SPY)
> SPY.monthly<-SPY.monthly[,6]
> SPY.ret<-Delt(SPY.monthly$data.SPY.Adjusted)
> names(SPY.ret)<-paste("SPY.Ret")
> SPY.ret[c(1:3,nrow(SPY.ret)),]
             SPY.Ret
12月 2010          NA
1月 2011   0.02330040
2月 2011   0.03473728
12月 2013 0.02592700
>
> # Aggregate Bond Market ETF Data
> data.LAG<-read.csv("SPAB Yahoo.csv",header=TRUE)
> data.LAG<-data.LAG[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.LAG$Date,format="%Y-%m-%d")
> data.LAG<-cbind(date, data.LAG[,-1])
> data.LAG<-data.LAG[order(data.LAG$date),]
> data.LAG<-xts(data.LAG[,2:7],order.by=data.LAG[,1])
> names(data.LAG)<-paste(c("LAG.Open","LAG.High","LAG.Low","LAG.Close","LAG.Volume","LAG.Adjusted"))
> LAG.monthly<-to.monthly(data.LAG)
> LAG.monthly<-LAG.monthly[,6]
> LAG.ret<-Delt(LAG.monthly$data.LAG.Adjusted)
> names(LAG.ret)<-paste("LAG.Ret")
> LAG.ret[c(1:3,nrow(LAG.ret)),]
              LAG.Ret
12月 2010           NA
1月 2011   0.000000000
2月 2011   0.002670325
12月 2013 -0.004367986
>
```

Step 2: Combine Monthly Return Data for SPY and LAG

```
> ### Fewer digits on display
> options(digits=3)
> ### Combined return monthly data for two securities
> Ret.monthly<-cbind(SPY.ret[-1,],LAG.ret[-1,])
> Ret.monthly[c(1:3,nrow(Ret.monthly)),]
           SPY.Ret   LAG.Ret
1月 2011   0.02330  0.000000
2月 2011   0.03474  0.002670
3月 2011   0.00012 -0.000812
12月 2013 0.02593 -0.004368
>
```

Step 3: Calculate the Mean and Standard Deviation of SPY and LAG Returns

```
> ### Mean and Standard Deviation for two securities
> SPY.Avg<-mean(Ret.monthly$SPY.Ret)
> SPY.Avg
[1] 0.0131
> SPY.sd<-sd(Ret.monthly$SPY.Ret)
> SPY.sd
[1] 0.035
> LAG.Avg<-mean(Ret.monthly$LAG.Ret)
> LAG.Avg
[1] 0.00259
> LAG.sd<-sd(Ret.monthly$LAG.Ret)
> LAG.sd
[1] 0.00845
>
```

Step 4: Find the Covariance of SPY and LAG Returns

```
> ### Covariance of two securities' return
> covar<-cov(Ret.monthly$SPY.Ret,Ret.monthly$LAG.Ret)
> covar
            LAG.Ret
SPY.Ret -5.25e-05
>
```

31°C 多云

```
> ### No labels and display of scientific notation
> options(scipen=100)
> covar<-covar[1,1]
> covar
[1] -0.0000525
>
```

Step 5: Create a Series of Weights for SPY Stock

To find out all the risk/return combination of SPY and LAG stock, we need to create a series that all possible weight combinations.

Since individual investors and some institutional investors do not or are not allowed to short sell stock, we can restrict our weights from 100 % SPY/0 % LAG to 0 % SPY/100 % LAG.

Since the weights must sum to one, we can create a series of weights for SPY between 0 and 100 % by 1 % increments.

Note that the number of increments is a trade-off between speed and accuracy. For our purpose, using 1 % increments would be sufficiently accurate and would not affect the speed of running the program substantially.

```
> ### Series of sequence of weight of SPY
> Portfolio<-data.frame(seq(0,1,by=.01))
> names(Portfolio)<-paste("SPY.wgt")
> Portfolio[1:5,]
[1] 0.00 0.01 0.02 0.03 0.04
> Portfolio[(nrow(Portfolio)-5):nrow(Portfolio),]
[1] 0.95 0.96 0.97 0.98 0.99 1.00
>
```

Step 6: Create a Series of Weights for LAG Stock

No short sell, so weight of LAG = 1 – weight of SPY

```
> ### Weight of LAG = 1 - Weight of SPY
> Portfolio$LAG.wgt<-1-Portfolio$SPY.wgt
> Portfolio[c(1:3,nrow(Portfolio)),]
    SPY.wgt LAG.wgt
1      0.00    1.00
2      0.01    0.99
3      0.02    0.98
101    1.00    0.00
>
```

Step 7: Calculate Portfolio Return for Each Weight Combination

```
> ### Use weight to compute port ret
> Portfolio$PortRet<-Portfolio$SPY.wgt*SPY.Avg+Portfolio$LAG.wgt*LAG.Avg
> Portfolio[c(1:3,nrow(Portfolio)),]
    SPY.wgt LAG.wgt PortRet
1      0.00    1.00 0.00259
2      0.01    0.99 0.00269
3      0.02    0.98 0.00280
101    1.00    0.00 0.01309
>
```

Step 8: Calculate Portfolio Risk for Each Weight Combination

```
> ### sqrt of w1^2*sd1^2 + w2^2*sd2^2 + 2*w1*w2*covariance1,2
> Portfolio$PortRisk<-sqrt((Portfolio$SPY.wgt**2*SPY.sd**2)+(Portfolio$LAG.wgt**2*LAG.sd**2)+(2*covar*Portfolio$SPY.wgt*Portfolio$LAG.wgt))
> Portfolio[c(1:3,nrow(Portfolio)),]
    SPY.wgt LAG.wgt PortRet PortRisk
1      0.00    1.00 0.00259  0.00845
2      0.01    0.99 0.00269  0.00831
3      0.02    0.98 0.00280  0.00819
101    1.00    0.00 0.01309  0.03502
>
```

Step 9: Identify the Minimum-Variance Portfolio

```
> ### min var achieved when risk is minimized
> minvar.port<-subset(Portfolio,Portfolio$PortRisk==min(Portfolio$PortRisk))
> minvar.port
   SPY.wgt LAG.wgt PortRet PortRisk
10    0.09    0.91 0.00353  0.00778
>
```

Step 10: Identify the Tangency Portfolio

The *tangency portfolio* is the portfolio that yields the highest Sharpe Ratio.

```
> ### tangent portfolio: import idea of sharpe ratio and find maximum of it
> riskfree=.0007/12
> Portfolio$Sharpe<-(Portfolio$PortRet-riskfree)/Portfolio$PortRisk
> Portfolio[c(1:3,nrow(Portfolio)),]
    SPY.wgt LAG.wgt PortRet PortRisk Sharpe
1      0.00    1.00 0.00259  0.00845  0.299
2      0.01    0.99 0.00269  0.00831  0.317
3      0.02    0.98 0.00280  0.00819  0.334
101    1.00    0.00 0.01309  0.03502  0.372
> tangency.port<-subset(Portfolio,Portfolio$Sharpe==max(Portfolio$Sharpe))
> tangency.port
   SPY.wgt LAG.wgt PortRet PortRisk Sharpe
23    0.22    0.78  0.0049  0.00921  0.525
>
```

Step 11: Identify Efficient Portfolios

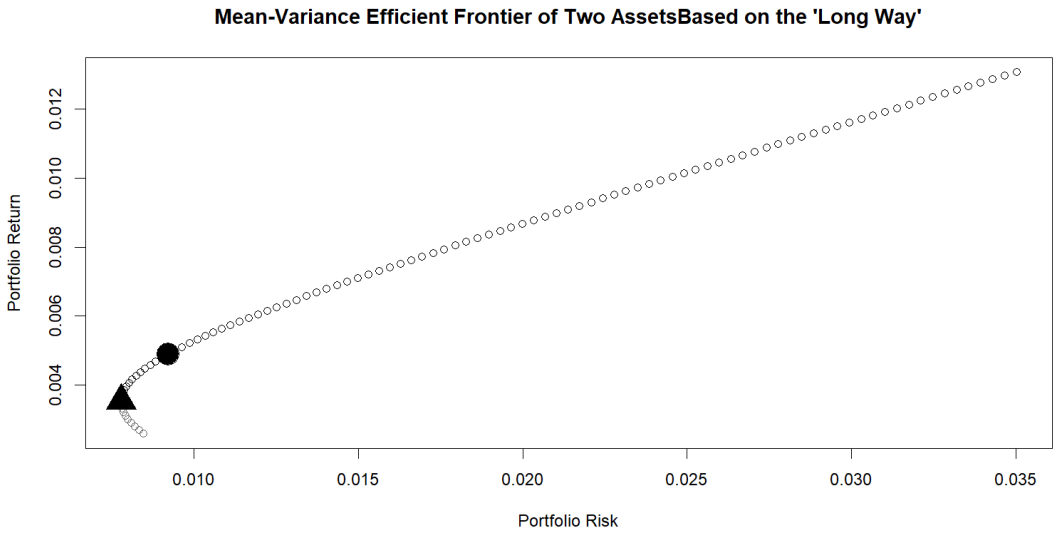
Efficient portfolios are only those portfolios that yield the highest return for a given level of risk. We extract these efficient portfolios by using the subset command and only keep those portfolios with returns greater than the return of the minimum variance portfolio.

```
> ### Efficient portfolio = return higher than minvar portfolio's return
> eff.frontier<-subset(Portfolio,Portfolio$PortRet>=minvar.port$PortRet)
> eff.frontier[c(1:3,nrow(eff.frontier)),]
    SPY.wgt LAG.wgt PortRet PortRisk Sharpe
10     0.09    0.91 0.00353  0.00778  0.447
11     0.10    0.90 0.00364  0.00779  0.459
12     0.11    0.89 0.00374  0.00782  0.471
101    1.00    0.00 0.01309  0.03502  0.372
>
```

Step 12: Plot the MV Efficient Frontier

```
> plot(x=Portfolio$PortRisk,xlab="Portfolio Risk",y=Portfolio$PortRet,ylab="Portfolio Return",col="gray40",main=" Mean-Variance Efficient Front
ier of Two AssetsBased on the 'Long Way' ")
> abline(h=0,lty=1)
> points(x=minvar.port$PortRisk,y=minvar.port$PortRet,pch=17,cex=3)
> points(x=tangency.port$PortRisk,y=tangency.port$PortRet,pch=19,cex=3)
> points(x=eff.frontier$PortRisk,y=eff.frontier$PortRet)
>
```

From the below graph, **triangle point** is the minimum variance point and the **circle point** is the tangent portfolio judged by Sharpe Ratio.



7.2 Two-Assets Using Quadratic Programming

The "Long Way" is useful to gain an intuition of what is going on when we apply quadratic programming. The technique is called quadratic programming because we are performing an optimization on a quadratic function. A quadratic function is a function that has a variable (in this case weight) that is squared. In the two-asset case, we are minimizing Eq. (4.2). The weights in that formula are squared. For our purpose, quadratic programming finds the combination of securities that yield the minimum risk for a specific level of return.

Step 1: Import Returns Data and Convert it Into a Matrix

```
> ### Convert monthly return data into a matrix with renamed columns
> Ret.monthly[c(1:3,nrow(Ret.monthly)),]
           SPY.Ret   LAG.Ret
1月 2011  0.02330  0.000000
2月 2011  0.03474  0.002670
3月 2011  0.00012 -0.000812
12月 2013 0.02593 -0.004368
> mat.ret<-matrix(Ret.monthly,nrow(Ret.monthly))
> mat.ret[1:3,]
        [,1]      [,2]
[1,] 0.02330  0.000000
[2,] 0.03474  0.002670
[3,] 0.00012 -0.000812
> colnames(mat.ret)<-c("SPY","LAG")
> head(mat.ret)
           SPY       LAG
[1,]  0.02330  0.000000
[2,]  0.03474  0.002670
[3,]  0.00012 -0.000812
[4,]  0.02896  0.018448
[5,] -0.01121  0.012140
[6,] -0.01687 -0.005976
> tail(mat.ret)
           SPY       LAG
[31,]  0.0517 -0.000369
[32,] -0.0300 -0.007871
[33,]  0.0316  0.015182
[34,]  0.0463  0.005557
[35,]  0.0296 -0.003487
[36,]  0.0259 -0.004368
>
```

Step 2: Calculate Variance-Covariance (VCOV) Matrix of Returns

```
> ### cov command to compute covariance of a matrix in R
> VCOV<-cov(mat.ret)
> VCOV
           SPY        LAG
SPY  0.0012265 -0.0000525
LAG -0.0000525  0.0000714
> 
```

Step 3: Construct the Target Portfolio Return Vector

```
> ### Average return of two securities in matrix
> avg.ret<-matrix(apply(mat.ret,2,mean))
> colnames(avg.ret)<-paste("Avg.Ret")
> rownames(avg.ret)<-paste(c("SPY","LAG"))
> avg.ret
    Avg.Ret
SPY 0.01309
LAG 0.00259
> 
> ### min of average as min, max of average as max
> min.ret<-min(avg.ret)
> min.ret
[1] 0.00259
> max.ret<-max(avg.ret)
> max.ret
[1] 0.0131
> ### Sequence of return with 100-step increment
> increments=100
> tgt.ret<-seq(min.ret,max.ret,length=increments)
> head(tgt.ret)
[1] 0.00259 0.00269 0.00280 0.00290 0.00301 0.00312
> tail(tgt.ret)
[1] 0.0126 0.0127 0.0128 0.0129 0.0130 0.0131
> 
```

Step 4: Construct Dummy Portfolio Standard Deviation Vector

```
> ### pre-set sd sequence of 100-step increment
> tgt.sd<-rep(0,length=increments)
> tgt.sd
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [58] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> 
```

Step 5: Construct Dummy Portfolio Weights Vector

```
> ### pre-set weight sequence of 100 * 2 format
> wgt<-matrix(0,nrow=increments,ncol=length(avg.ret))
> head(wgt)
     [,1] [,2]
[1,]    0    0
[2,]    0    0
[3,]    0    0
[4,]    0    0
[5,]    0    0
[6,]    0    0
> tail(wgt)
       [,1] [,2]
 [95,]    0    0
 [96,]    0    0
 [97,]    0    0
 [98,]    0    0
 [99,]    0    0
[100,]    0    0
>
```

Step 6: Run the quadprog Optimizer

```
> library(quadprog)
> for (i in 1:increments){
+   Dmat<-2*VCOV
+   dvec<-c(rep(0,length(avg.ret)))
+   Amat<-cbind(rep(1,length(avg.ret)),avg.ret,diag(1,nrow=ncol(Ret.monthly)))
+   bvec<-c(1,tgt.ret[i],rep(0,ncol(Ret.monthly)))
+   soln<-solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=2)
+   tgt.sd[i]<-sqrt(soln$value)
+   wgt[i,]<-soln$solution}
> head(tgt.sd)
[1] 0.00845 0.00831 0.00818 0.00807 0.00798 0.00790
> tail(tgt.sd)
[1] 0.0332 0.0335 0.0339 0.0343 0.0347 0.0350
> options(scipen=100)
> head(wgt)
      [,1]  [,2]
[1,] 0.0000 1.000
[2,] 0.0101 0.990
[3,] 0.0202 0.980
[4,] 0.0303 0.970
[5,] 0.0404 0.960
[6,] 0.0505 0.949
> tail(wgt)
       [,1]                  [,2]
 [95,] 0.949  0.050505050505050164
 [96,] 0.960  0.040404040404039776
 [97,] 0.970  0.030303030303029832
 [98,] 0.980  0.020202020202019888
 [99,] 0.990  0.010101010101009500
[100,] 1.000 -0.000000000000000444
> colnames(wgt)<-paste(c("wgt.SPY","wgt.LAG"))
> wgt[1,1]<-0
> wgt[nrow(wgt),2]<-0
> head(wgt)
     wgt.SPY wgt.LAG
[1,]  0.0000   1.000
[2,]  0.0101   0.990
[3,]  0.0202   0.980
[4,]  0.0303   0.970
[5,]  0.0404   0.960
[6,]  0.0505   0.949
> tail(wgt)
       wgt.SPY wgt.LAG
 [95,]   0.949  0.0505
 [96,]   0.960  0.0404
 [97,]   0.970  0.0303
 [98,]   0.980  0.0202
 [99,]   0.990  0.0101
[100,]   1.000  0.0000
>
```

Step 7: Combine Portfolio Returns, Portfolio Standard Deviations, and Portfolio Weights

```
> tgt.port<-data.frame(cbind(tgt.ret,tgt.sd,wgt))
> head(tgt.port)
  tgt.ret  tgt.sd wgt.SPY wgt.LAG
1 0.00259 0.00845  0.0000   1.000
2 0.00269 0.00831  0.0101   0.990
3 0.00280 0.00818  0.0202   0.980
4 0.00290 0.00807  0.0303   0.970
5 0.00301 0.00798  0.0404   0.960
6 0.00312 0.00790  0.0505   0.949
> tail(tgt.port)
    tgt.ret tgt.sd wgt.SPY wgt.LAG
95   0.0126 0.0332   0.949  0.0505
96   0.0127 0.0335   0.960  0.0404
97   0.0128 0.0339   0.970  0.0303
98   0.0129 0.0343   0.980  0.0202
99   0.0130 0.0347   0.990  0.0101
100  0.0131 0.0350   1.000  0.0000
>
```

Step 8: Identify the Minimum Variance Portfolio

```
> minvar.port<-subset(tgt.port,tgt.port$tgt.sd==min(tgt.port$tgt.sd))
> minvar.port
   tgt.ret  tgt.sd wgt.SPY wgt.LAG
10 0.00354 0.00778  0.0909   0.909
>
```

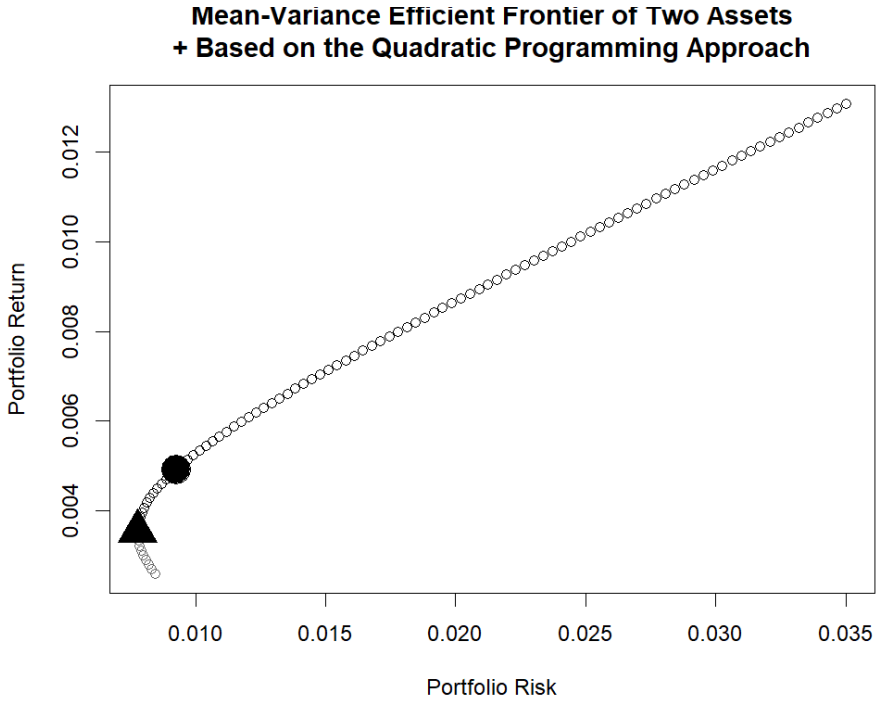
Step 9: Identify the Tangency Portfolio

```
> riskfree
[1] 0.0000583
> tgt.port$Sharpe<-(tgt.port$tgt.ret-riskfree)/tgt.port$tgt.sd
> head(tgt.port)
  tgt.ret  tgt.sd wgt.SPY wgt.LAG Sharpe
1 0.00259 0.00845  0.0000   1.000  0.299
2 0.00269 0.00831  0.0101   0.990  0.317
3 0.00280 0.00818  0.0202   0.980  0.335
4 0.00290 0.00807  0.0303   0.970  0.352
5 0.00301 0.00798  0.0404   0.960  0.370
6 0.00312 0.00790  0.0505   0.949  0.387
> tail(tgt.port)
    tgt.ret tgt.sd wgt.SPY wgt.LAG Sharpe
95   0.0126 0.0332   0.949  0.0505  0.377
96   0.0127 0.0335   0.960  0.0404  0.376
97   0.0128 0.0339   0.970  0.0303  0.375
98   0.0129 0.0343   0.980  0.0202  0.374
99   0.0130 0.0347   0.990  0.0101  0.373
100  0.0131 0.0350   1.000  0.0000  0.372
> tangency.port<-subset(tgt.port,tgt.port$Sharpe==max(tgt.port$Sharpe))
> tangency.port
   tgt.ret  tgt.sd wgt.SPY wgt.LAG Sharpe
23 0.00492 0.00925   0.222   0.778  0.525
>
```

Step 10: Identify Efficient Portfolios

```
> eff.frontier<-subset(tgt.port,tgt.port$tgt.ret>=minvar.port$tgt.ret)
> eff.frontier[c(1:3,nrow(eff.frontier)),]
    tgt.ret  tgt.sd wgt.SPY wgt.LAG Sharpe
10  0.00354 0.00778  0.0909   0.909  0.448
11  0.00365 0.00779  0.1010   0.899  0.461
12  0.00375 0.00782  0.1111   0.889  0.472
100 0.01309 0.03502  1.0000   0.000  0.372
>
```

Step 11: Plot the MV Efficient Frontier



7.3 Multiple Assets Using Quadratic Programming

Step 1: Import SPY, LAG, SLY, and CWI Data From Yahoo Finance and Calculate Monthly Returns for Each Security

```
> # Large-Cap ETF Data
> SPY.ret[c(1:3,nrow(SPY.ret)),]
          SPY.Ret
12月 2010      NA
1月 2011   0.0233
2月 2011   0.0347
12月 2013  0.0259
> # Aggregate Bond Market Data
> LAG.ret[c(1:3,nrow(LAG.ret)),]
           LAG.Ret
12月 2010       NA
1月 2011   0.00000
2月 2011   0.00267
12月 2013 -0.00437
> # Small Cap Data Data
> data.SLY<-read.csv("SLY Yahoo.csv",header=TRUE)
> data.SLY = data.SLY[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.SLY$Date,format="%Y-%m-%d")
> data.SLY<-cbind(date, data.SLY[,-1])
> data.SLY<-data.SLY[order(data.SLY$date),]
> data.SLY<-xts(data.SLY[,2:7],order.by=data.SLY[,1])
> names(data.SLY)<-paste(c("SLY.Open","SLY.High","SLY.Low","SLY.Close","SLY.Volume","SLY.Adjusted"))
> SLY.monthly<-to.monthly(data.SLY)
> SLY.monthly<-SLY.monthly[,6]
> SLY.ret<-Delt(SLY.monthly$data.SLY.Adjusted)
> names(SLY.ret)<-paste("SLY.Ret")
> SLY.ret[c(1:3,nrow(SLY.ret)),]
           SLY.Ret
12月 2010       NA
1月 2011  -0.00319
2月 2011   0.03998
12月 2013  0.01435
>
>
> # Global Equities Data
> data.CWI<-read.csv("CWI Yahoo.csv",header=TRUE)
> data.CWI = data.CWI[,c(1,2,3,4,5,7,6)]
> date<-as.Date(data.CWI$Date,format="%Y-%m-%d")
> data.CWI<-cbind(date, data.CWI[,-1])
> data.CWI<-data.CWI[order(data.CWI$date),]
> data.CWI<-xts(data.CWI[,2:7],order.by=data.CWI[,1])
> names(data.CWI)<-paste(c("CWI.Open","CWI.High","CWI.Low","CWI.Close","CWI.Volume","CWI.Adjusted"))
> CWI.monthly<-to.monthly(data.CWI)
> CWI.monthly<-CWI.monthly[,6]
> CWI.ret<-Delt(CWI.monthly$data.CWI.Adjusted)
> names(CWI.ret)<-paste("CWI.Ret")
> CWI.ret[c(1:3,nrow(CWI.ret)),]
          CWI.Ret
12月 2010      NA
1月 2011   0.0112
2月 2011   0.0292
12月 2013  0.0151
>
```

Step 2: Combine Monthly Returns Data for the Four Securities and Convert Combined Data Into a Returns Matrix

```
> Ret.monthly<-cbind(SPY.ret[-1,],LAG.ret[-1,],SLY.ret[-1,],CWI.ret[-1,])
> Ret.monthly[c(1:3,nrow(Ret.monthly)),]
          SPY.Ret   LAG.Ret  SLY.Ret CWI.Ret
1月 2011  0.02330  0.000000 -0.00319  0.0112
2月 2011  0.03474  0.002670  0.03998  0.0292
3月 2011  0.00012 -0.000812  0.02355 -0.0051
12月 2013 0.02593 -0.004368  0.01435  0.0151
> mat.ret<-matrix(Ret.monthly,nrow(Ret.monthly))
> mat.ret[1:3,]
        [,1]      [,2]     [,3]    [,4]
[1,] 0.02330  0.000000 -0.00319  0.0112
[2,] 0.03474  0.002670  0.03998  0.0292
[3,] 0.00012 -0.000812  0.02355 -0.0051
> colnames(mat.ret)<-c("SPY","LAG","SLY","CWI")
> head(mat.ret)
          SPY       LAG      SLY     CWI
[1,]  0.02330  0.000000 -0.00319  0.0112
[2,]  0.03474  0.002670  0.03998  0.0292
[3,]  0.00012 -0.000812  0.02355 -0.0051
[4,]  0.02896  0.018448  0.01477  0.0516
[5,] -0.01121  0.012140 -0.02305 -0.0333
[6,] -0.01687 -0.005976 -0.01386 -0.0130
> tail(mat.ret)
          SPY       LAG     SLY      CWI
[31,]  0.0517 -0.000369  0.0743  0.04321
[32,] -0.0300 -0.007871 -0.0338 -0.01473
[33,]  0.0316  0.015182  0.0668  0.06665
[34,]  0.0463  0.005557  0.0363  0.03416
[35,]  0.0296 -0.003487  0.0414  0.00311
[36,]  0.0259 -0.004368  0.0144  0.01507
>
```

Step 3: Calculate Variance-Covariance (VCOV) Matrix of Returns

```
> VCOV<-cov(mat.ret)
> VCOV
            SPY         LAG       SLY         CWI
SPY  0.0012265 -0.0000525  0.001599  0.0014866
LAG -0.0000525  0.0000714 -0.000101 -0.0000125
SLY  0.0015986 -0.0001009  0.002322  0.0020040
CWI  0.0014866 -0.0000125  0.002004  0.0023178
>
```

Step 4: Construct the Target Portfolio Return Vector

```
> avg.ret<-matrix(apply(mat.ret,2,mean))
> rownames(avg.ret)<-c("SPY","LAG","SLY","CWI")
> colnames(avg.ret)<-c("Avg.Ret")
> avg.ret
    Avg.Ret
SPY 0.01309
LAG 0.00259
SLY 0.01465
CWI 0.00507
>
```

```
> min.ret<-min(avg.ret)
> min.ret
[1] 0.00259
> max.ret<-max(avg.ret)
> max.ret
[1] 0.0147
>
```

```
> increments=100
> tgt.ret<-seq(min.ret,max.ret,length=increments)
> head(tgt.ret)
[1] 0.00259 0.00271 0.00283 0.00295 0.00307 0.00320
> tail(tgt.ret)
[1] 0.0140 0.0142 0.0143 0.0144 0.0145 0.0147
>
```

Step 5: Construct Dummy Portfolio Standard Deviation Vector

```
> tgt.sd<-rep(0,length=increments)
> tgt.sd
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [58] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>
```

Step 6: Construct Dummy Portfolio Weights Vector

```
> wgt<-matrix(0,nrow=increments,ncol=length(avg.ret))
> head(wgt)
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
[5,]    0    0    0    0
[6,]    0    0    0    0
> tail(wgt)
       [,1] [,2] [,3] [,4]
 [95,]    0    0    0    0
 [96,]    0    0    0    0
 [97,]    0    0    0    0
 [98,]    0    0    0    0
 [99,]    0    0    0    0
[100,]    0    0    0    0
>
```

Step 7: Run the quadprog Optimizer

```
> library(quadprog)
> for (i in 1:increments){
+   Dmat<-2*VCOV
+   dvec<-c(rep(0,length(avg.ret)))
+   Amat<-cbind(rep(1,length(avg.ret)),avg.ret,diag(1,nrow=ncol(Ret.monthly)))
+   bvec<-c(1,tgt.ret[i],rep(0,ncol(Ret.monthly)))
+   soln<-solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=2)
+   tgt.sd[i]<-sqrt(soln$value)
+   wgt[i,]<-soln$solution}
>
```

```
> head(tgt.sd)
[1] 0.00845 0.00822 0.00809 0.00796 0.00786 0.00779
> tail(tgt.sd)
[1] 0.0426 0.0436 0.0448 0.0459 0.0470 0.0482
> head(wgt)
                         [,1]  [,2]    [,3]                   [,4]
[1,] -0.0000000000000001388 1.000 0.00000 -0.00000000000000011796
[2,]  0.0000000000000000000 0.974 0.00608  0.01958011437911514468
[3,]  0.0000000000000000000 0.972 0.01829  0.00928350718344366985
[4,]  0.0000000000000000000 0.970 0.03030 -0.00000000000000000918
[5,]  0.0000000000000000000 0.960 0.04040 -0.00000000000000000672
[6,]  0.0000000000000000347 0.949 0.05051 -0.00000000000000000427
> tail(wgt)
                      [,1]                  [,2]  [,3]                   [,4]
 [95,] 0.38953665830624629 0.0000000000000000139 0.610 -0.00000000000000000540
 [96,] 0.31162932664499710 0.0000000000000000000 0.688 -0.00000000000000000609
 [97,] 0.23372199498375079 0.0000000000000000139 0.766 -0.00000000000000000678
 [98,] 0.15581466332250116 0.0000000000000000000 0.844 -0.00000000000000000747
 [99,] 0.07790733166125574 0.0000000000000000000 0.922 -0.00000000000000000816
[100,] 0.0000000000000977 0.0000000000000000000 1.000 -0.00000000000000000885
> colnames(wgt)<-c("wgt.SPY","wgt.LAG","wgt.SLY","wgt.CWI")
> wgt[,1]<-ifelse(abs(wgt[,1])<=0.0000001,0,wgt[,1])
> wgt[,2]<-ifelse(abs(wgt[,2])<=0.0000001,0,wgt[,2])
> wgt[,3]<-ifelse(abs(wgt[,3])<=0.0000001,0,wgt[,3])
> wgt[,4]<-ifelse(abs(wgt[,4])<=0.0000001,0,wgt[,4])
> head(wgt)
     wgt.SPY wgt.LAG wgt.SLY wgt.CWI
[1,]       0   1.000 0.00000 0.00000
[2,]       0   0.974 0.00608 0.01958
[3,]       0   0.972 0.01829 0.00928
[4,]       0   0.970 0.03030 0.00000
[5,]       0   0.960 0.04040 0.00000
[6,]       0   0.949 0.05051 0.00000
> tail(wgt)
       wgt.SPY wgt.LAG wgt.SLY wgt.CWI
 [95,]  0.3895       0   0.610       0
 [96,]  0.3116       0   0.688       0
 [97,]  0.2337       0   0.766       0
 [98,]  0.1558       0   0.844       0
 [99,]  0.0779       0   0.922       0
[100,]  0.0000       0   1.000       0
>
```

```
> CHECK<-rowSums(wgt)
> CHECK
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [58] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
>
```

Step 8: Combine Portfolio Returns, Portfolio Standard Deviations, and Portfolio Weights

```
> tgt.port<-data.frame(cbind(tgt.ret,tgt.sd,wgt))
> head(tgt.port)
  tgt.ret  tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI
1 0.00259 0.00845       0   1.000 0.00000 0.00000
2 0.00271 0.00822       0   0.974 0.00608 0.01958
3 0.00283 0.00809       0   0.972 0.01829 0.00928
4 0.00295 0.00796       0   0.970 0.03030 0.00000
5 0.00307 0.00786       0   0.960 0.04040 0.00000
6 0.00320 0.00779       0   0.949 0.05051 0.00000
> tail(tgt.port)
    tgt.ret tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI
95   0.0140 0.0426  0.3895       0   0.610       0
96   0.0142 0.0436  0.3116       0   0.688       0
97   0.0143 0.0448  0.2337       0   0.766       0
98   0.0144 0.0459  0.1558       0   0.844       0
99   0.0145 0.0470  0.0779       0   0.922       0
100  0.0147 0.0482  0.0000       0   1.000       0
> no.short.tgt.port<-tgt.port
>
```

Step 9: Identify the Minimum Variance Portfolio

```
> minvar.port<-subset(tgt.port,tgt.port$tgt.sd==min(tgt.port$tgt.sd))
> minvar.port
  tgt.ret  tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI
8 0.00344 0.00774  0.0241   0.926  0.0497       0
>
```

Step 10: Identify the Tangency Portfolio

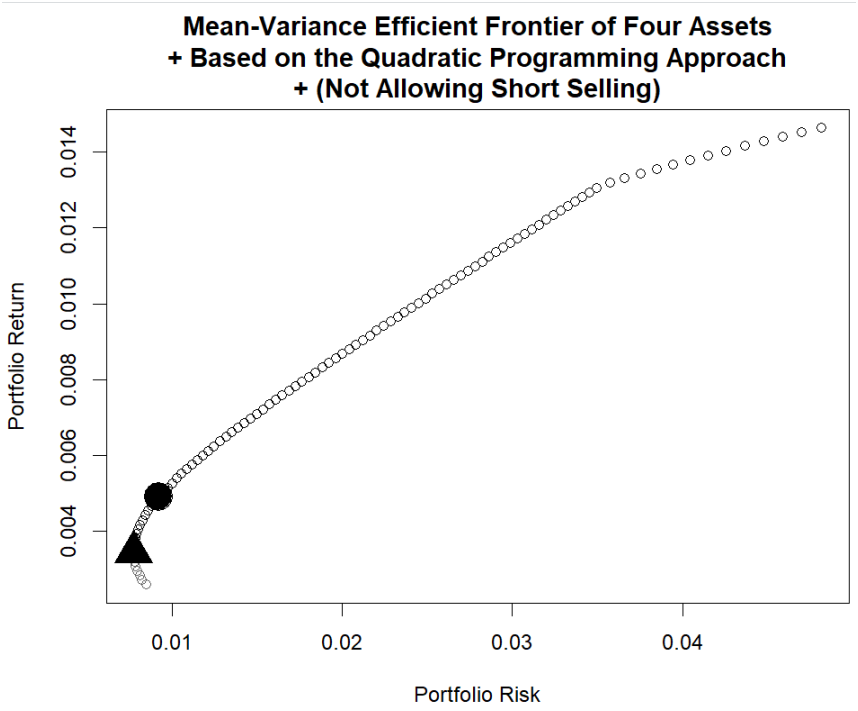
```
> riskfree
[1] 0.0000583
> tgt.port$Sharpe<-(tgt.port$tgt.ret-riskfree)/tgt.port$tgt.sd
> head(tgt.port)
  tgt.ret  tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI Sharpe
1 0.00259 0.00845       0   1.000 0.00000 0.00000  0.299
2 0.00271 0.00822       0   0.974 0.00608 0.01958  0.322
3 0.00283 0.00809       0   0.972 0.01829 0.00928  0.343
4 0.00295 0.00796       0   0.970 0.03030 0.00000  0.363
5 0.00307 0.00786       0   0.960 0.04040 0.00000  0.384
6 0.00320 0.00779       0   0.949 0.05051 0.00000  0.403
> tail(tgt.port)
    tgt.ret tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI Sharpe
95   0.0140 0.0426  0.3895       0   0.610       0  0.329
96   0.0142 0.0436  0.3116       0   0.688       0  0.323
97   0.0143 0.0448  0.2337       0   0.766       0  0.318
98   0.0144 0.0459  0.1558       0   0.844       0  0.313
99   0.0145 0.0470  0.0779       0   0.922       0  0.308
100  0.0147 0.0482  0.0000       0   1.000       0  0.303
>
> tangency.port<-subset(tgt.port,tgt.port$Sharpe==max(tgt.port$Sharpe))
> tangency.port
   tgt.ret  tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI Sharpe
20  0.0049 0.00922   0.221   0.779       0       0  0.525
>
```

Step 11: Identify Efficient Portfolios

```
> eff.frontier<-subset(tgt.port,tgt.port$tgt.ret>=minvar.port$tgt.ret)
> eff.frontier[c(1:3,nrow(eff.frontier)),]
    tgt.ret  tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI Sharpe
8   0.00344 0.00774  0.0241   0.926  0.0497       0  0.437
9   0.00356 0.00775  0.0467   0.913  0.0401       0  0.452
10  0.00368 0.00778  0.0693   0.900  0.0306       0  0.466
100 0.01465 0.04818  0.0000   0.000  1.0000       0  0.303
>
```

Step 12: Plot the MV Efficient Frontier

```
> plot(x=tgt.sd,y=tgt.ret,col="gray40",xlab="Portfolio Risk",ylab="Portfolio Return",
+ main="Mean-Variance Efficient Frontier of Four Assets
+ + Based on the Quadratic Programming Approach
+ + (Not Allowing Short Selling)")
> abline(h=0,lty=1)
> points(x=minvar.port$tgt.sd,y=minvar.port$tgt.ret,pch=17,cex=3)
> points(x=tangency.port$tgt.sd,y=tangency.port$tgt.ret,pch=19,cex=3)
> points(x=eff.frontier$tgt.sd,y=eff.frontier$tgt.ret)
>
```



7.4 Effect of Allowing Short Selling

Note that by not allowing short selling we are adding a constraint that restricts the optimizer from putting negative weights on securities that could enhance return if we were allowed to take on short positions.

First, we cannot be worse off when we do not impose a short selling restriction.

Second, short selling should enhance returns in situations where taking a short position in a security can benefit us.

Now, let us turn to the portions of the code that need to be modified when short selling is allowed. No modification is necessary for Steps 1–3. We will begin the modifications with Step 4 of the last section.

Step 4ss: Construct the Target Portfolio Return Vector (Modifying to Allow for Short Selling)

However, when short selling is allowed, we can in theory sell the other securities and invest the proceeds from the sale of those securities in the security with the largest return.

For the upper bound, we will construct it to be twice the maximum return of the four securities in our portfolio. This number is arbitrary but gives us a good enough range of portfolio returns for our purpose.

```
> tgt.ret<-seq(min.ret,max.ret*2,length=increments)
> head(tgt.ret)
[1] 0.00259 0.00286 0.00313 0.00340 0.00367 0.00394
> tail(tgt.ret)
[1] 0.0280 0.0282 0.0285 0.0288 0.0290 0.0293
> 
```

Step 5ss: Construct Dummy Portfolio Standard Deviation Vector

There is no change here.

Step 6ss: Construct Dummy Portfolio Weights Vector

There is no change here.

Step 7ss: Run the quadprog Optimizer (Modifying to Allow for Short Selling)

```
> library(quadprog)
> for (i in 1:length(tgt.ret)){
+   Dmat<-2*VCOV
+   dvec<-c(rep(0,length(avg.ret)))
+   Amat<-cbind(rep(1,length(avg.ret)),avg.ret)
+   bvec<-c(1,tgt.ret[i])
+   soln<-solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=2)
+   tgt.sd[i]<-sqrt(soln$value)
+   wgt[i,]<-soln$solution}
> head(tgt.sd)
[1] 0.00822 0.00798 0.00776 0.00758 0.00743 0.00732
> tail(tgt.sd)
[1] 0.0471 0.0476 0.0481 0.0486 0.0492 0.0497
> head(wgt)
      wgt.SPY wgt.LAG wgt.SLY  wgt.CWI
[1,] -0.11828   1.003  0.0997  0.01585
[2,] -0.08718   0.992  0.0989 -0.00326
[3,] -0.05609   0.980  0.0982 -0.02237
[4,] -0.02500   0.969  0.0974 -0.04149
[5,]  0.00609   0.958  0.0966 -0.06060
[6,]  0.03718   0.947  0.0959 -0.07971
> tail(wgt)
       wgt.SPY wgt.LAG wgt.SLY wgt.CWI
 [95,]    2.80 -0.0514  0.0276   -1.78
 [96,]    2.84 -0.0626  0.0269   -1.80
 [97,]    2.87 -0.0738  0.0261   -1.82
 [98,]    2.90 -0.0850  0.0253   -1.84
 [99,]    2.93 -0.0962  0.0246   -1.86
[100,]    2.96 -0.1074  0.0238   -1.88
> CHECK.wgt<-rowSums(wgt)
> CHECK.wgt
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [58] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
> 
```

Step 8ss: Combine Portfolio Returns, Portfolio Standard Deviations, and Portfolio Weights

There is no change in the code here.

```
> tgt.port<-data.frame(cbind(tgt.ret,tgt.sd,wgt))
> head(tgt.port)
  tgt.ret  tgt.sd  wgt.SPY wgt.LAG wgt.SLY  wgt.CWI
1 0.00259 0.00822 -0.11828   1.003  0.0997  0.01585
2 0.00286 0.00798 -0.08718   0.992  0.0989 -0.00326
3 0.00313 0.00776 -0.05609   0.980  0.0982 -0.02237
4 0.00340 0.00758 -0.02500   0.969  0.0974 -0.04149
5 0.00367 0.00743  0.00609   0.958  0.0966 -0.06060
6 0.00394 0.00732  0.03718   0.947  0.0959 -0.07971
> tail(tgt.port)
    tgt.ret tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI
95   0.0280 0.0471    2.80 -0.0514  0.0276   -1.78
96   0.0282 0.0476    2.84 -0.0626  0.0269   -1.80
97   0.0285 0.0481    2.87 -0.0738  0.0261   -1.82
98   0.0288 0.0486    2.90 -0.0850  0.0253   -1.84
99   0.0290 0.0492    2.93 -0.0962  0.0246   -1.86
100  0.0293 0.0497    2.96 -0.1074  0.0238   -1.88
> with.short.tgt.port<-tgt.port
>
```

Step 9ss: Identify the Minimum Variance Portfolio

```
> with.short.tgt.port<-tgt.port
> minvar.port<-subset(tgt.port,tgt.port$tgt.sd==min(tgt.port$tgt.sd))
> minvar.port
  tgt.ret  tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI
8 0.00447 0.00721  0.0994   0.924  0.0943  -0.118
>
```

Step 10ss: Identify the Tangency Portfolio

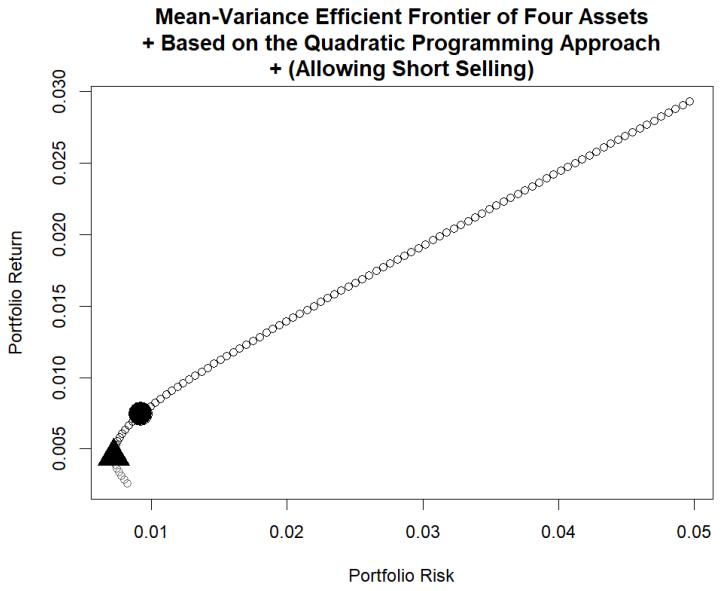
```
> riskfree
[1] 0.0000583
> tgt.port$Sharpe<-(tgt.port$tgt.ret-riskfree)/tgt.port$tgt.sd
> head(tgt.port)
  tgt.ret  tgt.sd  wgt.SPY wgt.LAG wgt.SLY  wgt.CWI Sharpe
1 0.00259 0.00822 -0.11828   1.003  0.0997  0.01585  0.307
2 0.00286 0.00798 -0.08718   0.992  0.0989 -0.00326  0.351
3 0.00313 0.00776 -0.05609   0.980  0.0982 -0.02237  0.395
4 0.00340 0.00758 -0.02500   0.969  0.0974 -0.04149  0.440
5 0.00367 0.00743  0.00609   0.958  0.0966 -0.06060  0.485
6 0.00394 0.00732  0.03718   0.947  0.0959 -0.07971  0.530
> tangency.port<-subset(tgt.port,tgt.port$Sharpe==max(tgt.port$Sharpe))
> tangency.port
   tgt.ret tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI Sharpe
19 0.00744 0.0092   0.441   0.801  0.0859  -0.328  0.803
>
```

Step 11ss: Identify Efficient Portfolios

```
> eff.frontier<-subset(tgt.port,tgt.port$tgt.ret>=minvar.port$tgt.ret)
> eff.frontier[c(1:3,nrow(eff.frontier)),]
    tgt.ret  tgt.sd wgt.SPY wgt.LAG wgt.SLY wgt.CWI Sharpe
8   0.00447 0.00721  0.0994   0.924  0.0943  -0.118  0.612
9   0.00474 0.00722  0.1305   0.913  0.0936  -0.137  0.649
10  0.00501 0.00727  0.1616   0.902  0.0928  -0.156  0.682
100 0.02931 0.04970  2.9599  -0.107  0.0238  -1.876  0.588
>
```

Step 12ss: Plot the MV Efficient Frontier

```
> plot(x=tgt.sd,y=tgt.ret,col="gray40",xlab="Portfolio Risk",ylab="Portfolio Return",
+ main="Mean-Variance Efficient Frontier of Four Assets
+ + Based on the Quadratic Programming Approach
+ + (Allowing Short Selling)")
> abline(h=0,lty=1)
> points(x=minvar.port$tgt.sd,y=minvar.port$tgt.ret,pch=17,cex=3)
> points(x=tangency.port$tgt.sd,y=tangency.port$tgt.ret,pch=19,cex=3)
> points(x=eff.frontier$tgt.sd,y=eff.frontier$tgt.ret)
>
```



Comparing the Efficient Frontier With and Without Short Sales

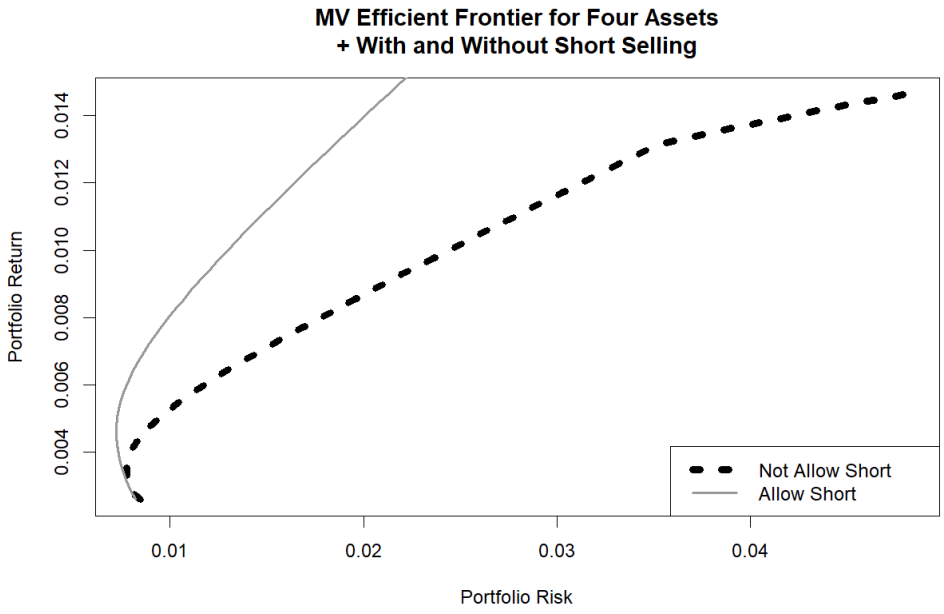
The black line represents the MV Efficient Frontier when short selling is not allowed. The gray line represents the MV Efficient Frontier when short selling is allowed.

That is, the gray line (efficient frontier when short sale restrictions are lifted) attains higher levels of portfolio returns for the same level of risk as the black line (efficient frontier when short sales are not allowed) once we start moving to higher levels of portfolio risk.

For lower levels of portfolio risk, relaxing the short sale restrictions does not enhance

returns.

```
> plot(x=no.short.tgt.port$tgt.sd,y=no.short.tgt.port$tgt.ret,xlab="Portfolio Risk",ylab="Portfolio Return",type="l",lwd=6,
lty=3,
+ main="MV Efficient Frontier for Four Assets
+ + With and Without Short Selling")
> lines(x=with.short.tgt.port$tgt.sd,y=with.short.tgt.port$tgt.ret,col="gray60",type="l",lwd=2)
> legend("bottomright",c("Not Allow Short","Allow Short"),col=c("black","gray60"),lty=c(3,1),lwd=c(6,2))
>
```



7.5 Further Reading

More details on the Sharpe Ratio are found in Sharpe [4] and [5]. Markowitz [3] is the original paper on portfolio selection. A more comprehensive and modern treatment of asset allocation and portfolio management can be found in Maginn et al. [2]. Ad ditional techniques used to calculate the efficient frontier are found in Elton et al. [1].

References

1. Elton, E., Gruber, M., Brown, S., & Goetzmann, W. (2010). Modern portfolio theory and investment analysis (8th ed.). New Jersey: Wiley.

2. Maginn, J., Tuttle, D., Pinto, J., & McLeavey, D. (2007). Managing investment portfolios: A dynamic process (3rd ed.). New Jersey: Wiley.

3. Markowitz, H. (1952). Portfolio selection. The Journal of Finance, 7, 77–91.

4. Sharpe, W. (1975). Adjusting for risk in portfolio performance. Journal of Portfolio Management, 1, 29–34.

5. Sharpe, W. (1994). The Sharpe ratio. Journal of Portfolio Management, 21, 49–58.

Chapter 8 Fixed Income

In this chapter, our analysis focuses on fixed income securities as examples. Many of the techniques used in this chapter are also equally applicable to both equity and fixed income securities.

We begin our analysis of fixed income securities by showing how to obtain and analyze economic data, which makes analyzing economic data important to understanding how our investments may perform during our holing period.

We then demonstrate an analysis of Treasury yields. Since other fixed income instruments rely on the rates of Treasury securities, understanding the shape of the yield curve and the slope of the yield curve is essential to making sound fixed income investment decisions.

Then, we demonstrate how to identify mean reverting patterns in Treasury securities.

Next, we analyze the time series of spreads between corporates of different investment grade ratings. We show how such an analysis can reveal the widening or tightening of credit spreads.

The second part of this chapter demonstrates discounted cash flow bond valuation. We also demonstrate how to calculate duration and convexity of bonds, which are tools used to manage interest rate risk.

8.1 Economic Analysis

As such, for illustrative purposes, we only analyze three economic indicators: Real GDP, unemployment, and inflation. The sources and techniques used in this section can be applied to identify and analyze other economic data.

8.1.1 Real GDP

When the economy does well, securities prices tend to increase. Conversely, when the economy does poorly, securities prices tend to decrease.

As such, it is important to know where we are in the economic cycle to determine the best investments that fit our strategy.

In particular, we obtain the data from the October 2013 World Economic Outlook.

As of early 2014, the US Real GDP growth data contains actual historical results from 1980 through 2012 and projected results from 2013 through 2018. From a presentation perspective, a key element for this analysis is to distinguish between historical results and projected results. For that purpose, we will construct a chart that uses different colored bars for the historical data and for the projected data.

Step 1: Import Historical and Projected Real GDP Growth Data from the IMF Website

```
> library(quantmod)
载入需要的程序包：xts
载入需要的程序包：zoo

载入程序包：'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric

载入需要的程序包：TTR
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
> library(xts)
> us.rgdp<-read.csv("USRGDP IMF WEO.csv",header=FALSE)
> us.rgdp[c(1:3,nrow(us.rgdp)),]
     V1     V2
1  1980 -0.245
2  1981  2.595
3  1982 -1.911
39 2018  3.066
>
```

Step 2: Rename Variables and Fix Index of Data Object

```
> colnames(us.rgdp)<-paste(c("Year","Value"))
> us.rgdp<-data.frame(us.rgdp)
> us.rgdp[c(1:3,nrow(us.rgdp)),]
   Year  Value
1  1980 -0.245
2  1981  2.595
3  1982 -1.911
39 2018  3.066
>
```

Step 3: Create Separate Variable for Historical Real GDP Growth Data

```
> us.rgdp$historical<-ifelse(us.rgdp$Year<=2012,us.rgdp$Value,0)
> us.rgdp[c(1:3,nrow(us.rgdp)),]
   Year  Value historical
1  1980 -0.245     -0.245
2  1981  2.595      2.595
3  1982 -1.911     -1.911
39 2018  3.066      0.000
> us.rgdp$projected<-ifelse(us.rgdp$Year>2012,us.rgdp$Value,0)
> us.rgdp[c(1:3,nrow(us.rgdp)),]
   Year  Value historical projected
1  1980 -0.245     -0.245     0.000
2  1981  2.595      2.595     0.000
3  1982 -1.911     -1.911     0.000
39 2018  3.066      0.000     3.066
>
```

Step 4: Setup Data for Transposition and Transpose the Data

```
> us.rgdp<-us.rgdp[,3:4]
> us.rgdp[c(1:3,nrow(us.rgdp)),]
   historical projected
1      -0.245     0.000
2       2.595     0.000
3      -1.911     0.000
39      0.000     3.066
> us.mat<-as.matrix(us.rgdp)
> t.us<-t(us.mat)
> head(t.us)
             [,1]  [,2]   [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11]  [,12] [,13] [,14] [,15] [,16]
historical -0.245 2.595 -1.911 4.633 7.259 4.239 3.512 3.462 4.204  3.68 1.919 -0.073 3.555 2.745 4.037 2.719
projected   0.000 0.000  0.000 0.000 0.000 0.000 0.000 0.000 0.000  0.00 0.000  0.000 0.000 0.000 0.000 0.000
           [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28]  [,29]  [,30] [,31] [,32]
historical 3.796 4.487  4.45 4.846 4.091 0.949 1.776 2.791 3.798 3.351 2.667  1.79 -0.291 -2.802 2.507 1.847
projected  0.000 0.000  0.00 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000  0.00  0.000  0.000 0.000 0.000
           [,33] [,34] [,35] [,36] [,37] [,38] [,39]
historical 2.779  0.00 0.000  0.00 0.000 0.000 0.000
projected  0.000  1.56 2.588  3.35 3.476 3.364 3.066
>
```

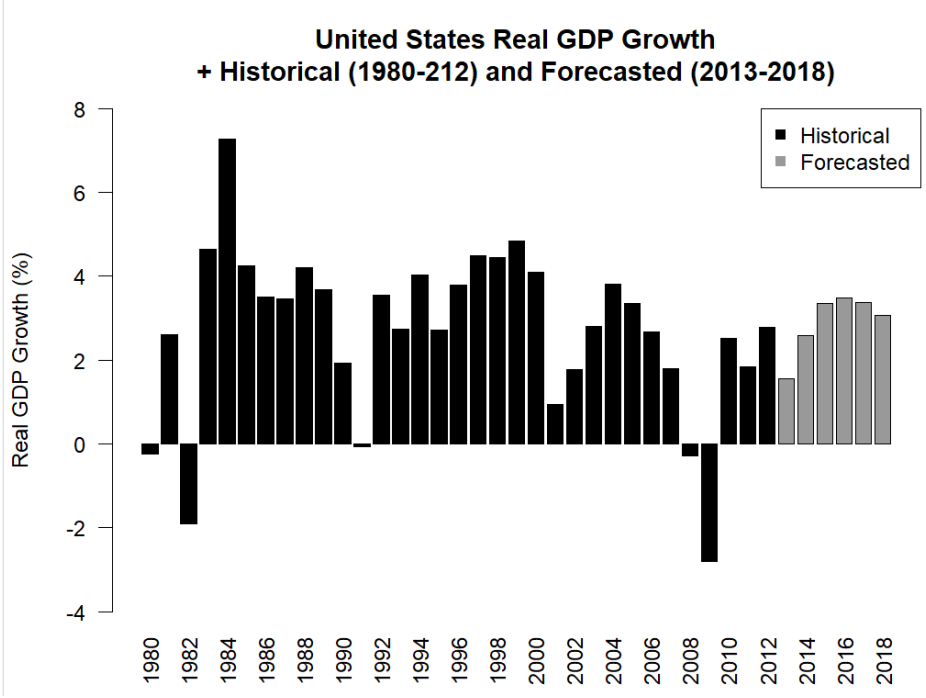
Step 6: Setup Sequence of Even Years to Customize x-axis

```
> xlabel=seq(1980,2018,by=1)
> even.years<-xlabel %% 2==0
> even.years
 [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
[19]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
[37]  TRUE FALSE  TRUE
> years.even<-cbind(data.frame(xlabel),data.frame(even.years))
> head(years.even)
  xlabel even.years
1   1980       TRUE
2   1981      FALSE
3   1982       TRUE
4   1983      FALSE
5   1984       TRUE
6   1985      FALSE
> years.even$Year<-ifelse(years.even$even.years=="TRUE",xlabel," ")
> xlabel.even<-years.even[,3]
> head(xlabel.even)
[1] "1980" " "    "1982" " "    "1984" " "
>
```

Step 7: Plot a Bar Chart of Annual Real GDP Growth Rate

```
> range(us.rgdp)
[1] -2.802  7.259
> y.range<-c(-4,8)
> y.range
[1] -4  8
> barplot(t.us,col=c("black","gray60"),ylab="Real GDP Growth (%)",ylim=y.range,names.arg=xlabel.even,las=2,
+ main="United States Real GDP Growth
+ + Historical (1980-212) and Forecasted (2013-2018)")
> legend("topright",c("Historical","Forecasted"),col=c("black","gray60"),pch=c(15,15))
>
```

The las=2 argument flips the text of the x-axis 90◦.



The names.arg argument tells R to use the xlabel.even sequence for the x-axis.

This presentation of the data makes it easy to distinguish which data are historical data and which data are projections.

8.1.2 Unemployment Rate

For our example, we analyze the unemployment rate over the last 50 years and identify its peaks and long-term average. To emphasize the peaks and long-term average, we need to annotate the chart we create. Below we show how to implement this analysis.

Step 1: Import Unemployment Rate Data from FRED

```
> US.unempl<-read.csv("UNRATE FRED.csv",header = TRUE)
> US.unempl$date<-as.Date(US.unempl$DATE,"%Y-%m-%d")
> US.unempl$UNRATE<-as.numeric(as.character(US.unempl$UNRATE))
> US.unempl<-xts(US.unempl$UNRATE,order.by=US.unempl$date)
> names(US.unempl)<-paste("UNRATE")
> US.unempl[1:3,]
           UNRATE
1948-01-01    3.4
1948-02-01    3.8
1948-03-01    4.0
>
```

Step 2: Subset Data from 1964 to 2013

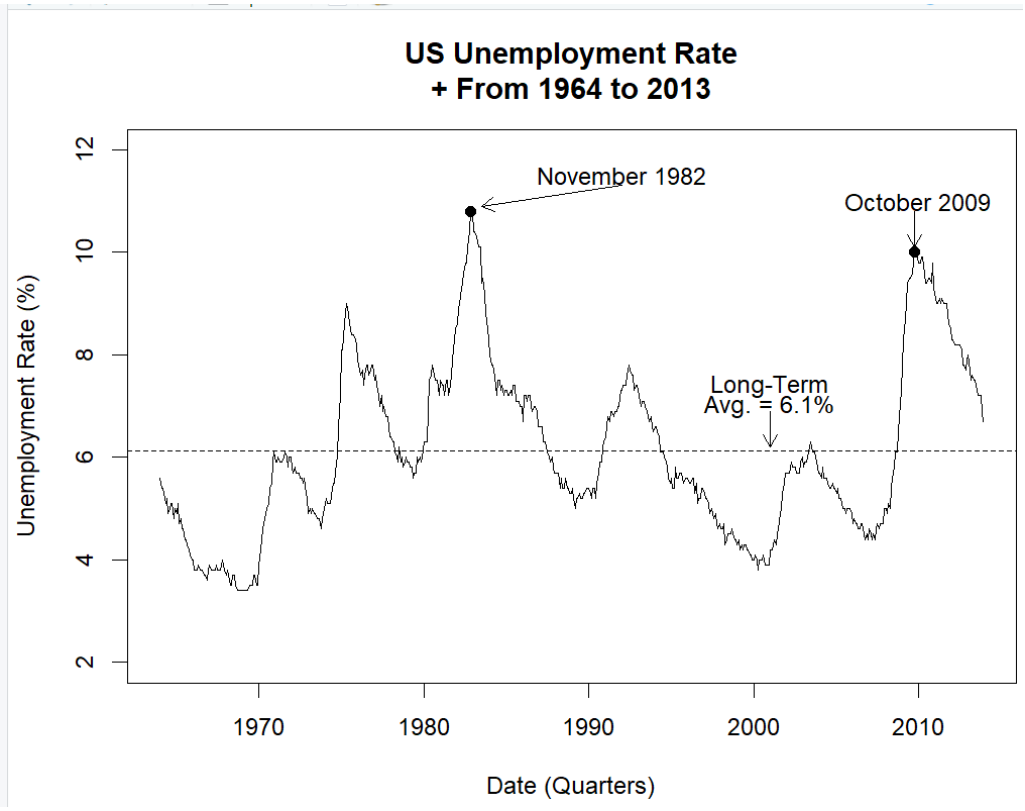
```
> US.unempl<-subset(US.unempl,index(US.unempl)>="1964-01-01" &index(US.unempl)<="2013-12-31")
> US.unempl[1:3,]
           UNRATE
1964-01-01    5.6
1964-02-01    5.4
1964-03-01    5.4
>
```

Step 3: Calculate Long-Term Average Unemployment Rate

```
> lt.avg<-mean(US.unempl$UNRATE)
> lt.avg
[1] 6.125333
>
```

Step 4: Plot the Unemployment Rate Data

```
> plot(x=index(US.unempl),xlab="Date (Quarters)",y=US.unempl$UNRATE,ylab="Unemployment Rate (%)",ylim=c(2,12),type
="l",
+ main="US Unemployment Rate
+ + From 1964 to 2013")
> abline(h=lt.avg,lty=2)
> text(as.Date("2001-01-01"),7.4,"Long-Term")
> text(as.Date("2001-01-01"),7,"Avg. = 6.1%")
> arrows(x0=as.Date("2001-01-01"),y0=6.9,x1=as.Date("2001-01-01"),y1=6.2,code=2,length=0.10)
> points(as.Date("1982-11-01"),10.8,pch=16)
> text(as.Date("1992-01-01"),11.5,"November 1982")
> arrows(x0=as.Date("1992-01-01"),y0=11.3,x1=as.Date("1983-07-01"),y1=10.9,code=2,length=0.10)
> points(as.Date("2009-10-01"),10,pch=16)
> text(as.Date("2010-01-01"),11,"October 2009")
> arrows(x0=as.Date("2009-10-01"), y0=10.8, x1=as.Date("2009-10-01"), y1=10.1, code=2, length=0.10)
>
```



8.1.3 Inflation Rate

A higher inflation rate would lead to a higher required expected rate of return, as a dollar tomorrow is worth less than a dollar is worth today.

Step 1: Import CPI Data from FRED

```
> US.CPI<-read.csv("CPIAUCNS_FRED.csv",header = TRUE)
> US.CPI$date<-as.Date(US.CPI$DATE,"%Y-%m-%d")
> US.CPI$CPIAUCNS<-as.numeric(as.character(US.CPI$CPIAUCNS))
> US.CPI<-xts(US.CPI$CPIAUCNS,order.by=US.CPI$date)
> names(US.CPI)<-paste("CPIAUCNS")
> US.CPI[1:3,]
           CPIAUCNS
1913-01-01      9.8
1913-02-01      9.8
1913-03-01      9.8
>
```

Step 2: Calculate a 12-Month Lag Variable

```
> US.Lag12<-Lag(US.CPI$CPIAUCNS,k=12)
> US.Lag12[1:20,]
           Lag.12
1913-01-01     NA
1913-02-01     NA
1913-03-01     NA
1913-04-01     NA
1913-05-01     NA
1913-06-01     NA
1913-07-01     NA
1913-08-01     NA
1913-09-01     NA
1913-10-01     NA
1913-11-01     NA
1913-12-01     NA
1914-01-01    9.8
1914-02-01    9.8
1914-03-01    9.8
1914-04-01    9.8
1914-05-01    9.7
1914-06-01    9.8
1914-07-01    9.9
1914-08-01    9.9
>
```

Step 3: Combine CPI and Lag CPI Data

```
> US.CPI<-merge(US.CPI,US.Lag12)
> names(US.CPI)<-paste(c("us.cpi","lag.cpi"))
> US.CPI[10:15,]
           us.cpi lag.cpi
1913-10-01   10.0      NA
1913-11-01   10.1      NA
1913-12-01   10.0      NA
1914-01-01   10.0     9.8
1914-02-01    9.9     9.8
1914-03-01    9.9     9.8
>
```

Step 4: Calculate Inflation Rate

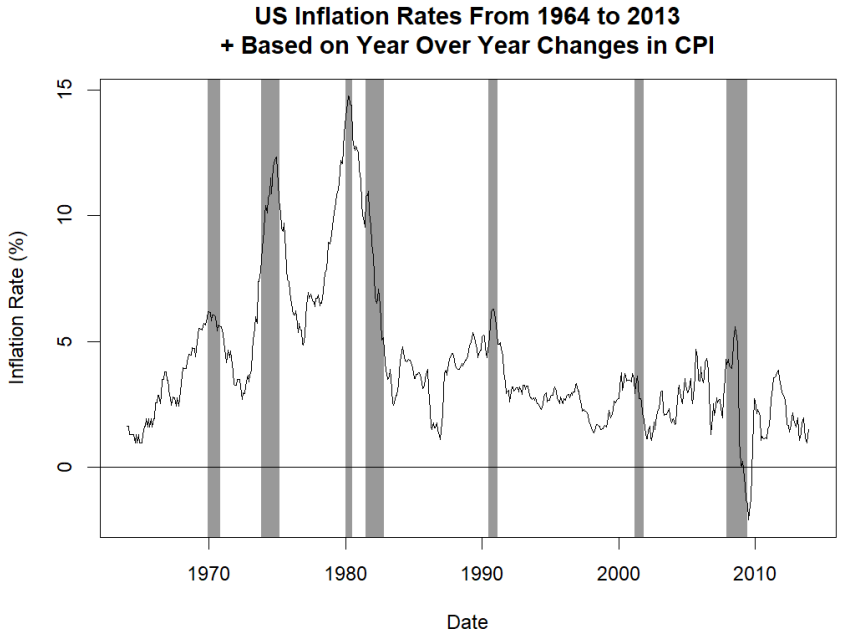
```
> US.CPI$inflation<-(US.CPI$us.cpi/US.CPI$lag.cpi-1)*100
> US.CPI[10:15,]
           us.cpi lag.cpi inflation
1913-10-01   10.0      NA        NA
1913-11-01   10.1      NA        NA
1913-12-01   10.0      NA        NA
1914-01-01   10.0     9.8  2.040816
1914-02-01    9.9     9.8  1.020408
1914-03-01    9.9     9.8  1.020408
>
```

Step 5: Subset Inflation Data Over the Last 50 Years

```
> US.CPI<-subset(US.CPI[,3],index(US.CPI)>="1964-01-01" &index(US.CPI)<="2013-12-01")
> US.CPI[c(1:3,nrow(US.CPI)),]
           inflation
1964-01-01  1.644737
1964-02-01  1.644737
1964-03-01  1.311475
2013-12-01  1.501736
>
```

Step 6: Plot the Inflation Rate Data

```
> plot(x=index(US.CPI),y=US.CPI$inflation,xlab="Date", ylab="Inflation Rate (%)", type="l", main="US Inflation Rates F
rom 1964 to 2013
+ + Based on Year Over Year Changes in CPI")
>
> shade<-par("usr")
> shade
[1] -2921.280000 16769.280000    -2.771306    15.430591
> rect(as.Date("2007-12-01"),shade[2],
+        + as.Date("2009-06-01"),shade[3],col="gray60",lty=0)
> rect(as.Date("2001-03-01"),shade[2],
+        + as.Date("2001-11-01"),shade[3],col="gray60",lty=0)
> rect(as.Date("1990-07-01"),shade[2],
+        + as.Date("1991-03-01"),shade[3],col="gray60",lty=0)
> rect(as.Date("1981-07-01"),shade[2],
+        + as.Date("1982-11-01"),shade[3],col="gray60",lty=0)
> rect(as.Date("1980-01-01"),shade[2],
+        + as.Date("1980-07-01"),shade[3],col="gray60",lty=0)
> rect(as.Date("1973-11-01"),shade[2],
+        + as.Date("1975-03-01"),shade[3],col="gray60",lty=0)
> rect(as.Date("1969-12-01"),shade[2],
+        + as.Date("1970-11-01"),shade[3],col="gray60",lty=0)
> rect(as.Date("1960-04-01"),shade[2],
+        + as.Date("1961-02-01"),shade[3],col="gray60",lty=0)
> box(which="plot",lty=1)
> lines(x=index(US.CPI),y=US.CPI$inflation)
> abline(h=0)
>
```



8.2 US Treasuries

8.2.1 Shape of the US Treasury Yield Curve

As such, knowledge of Treasury yields is required to determine whether the yields of other fixed income securities are large enough for the incremental risk we are taking on. In addition, trading strategies can be constructed around the differential between short rates and long-term rates.

If T-bill rates for 3-months is substantially lower than 6-month rates, an investor who expects to hold a T-Bill for 3 months may instead purchase a 6-month T-Bill and sell it in 3 months to take advantage of the higher yields on the longer dated instrument.

This strategy is known as *riding the yield curve*.

In this section, we demonstrate how to find examples of these different shapes of the yield curve and plot these examples.

Step 1: Obtaining US Treasury Yields for Key Maturities

```
> t20yr<-read.csv("DGS20.csv",header = TRUE)
> t20yr$date<-as.Date(t20yr$DATE,"%Y-%m-%d")
> t20yr$DGS20<-as.numeric(as.character(t20yr$DGS20))
警告信息:
强制改变过程中产生了NA
> t20yr<-xts(t20yr$DGS20,order.by=t20yr$date)
> names(t20yr)<-paste("DGS20")
> t20yr[1:3,]
           DGS20
2019-06-26  2.35
2019-06-27  2.31
2019-06-28  2.31
>
> t30yr<-read.csv("DGS30.csv",header = TRUE)
> t30yr$date<-as.Date(t30yr$DATE,"%Y-%m-%d")
> t30yr$DGS30<-as.numeric(as.character(t30yr$DGS30))
警告信息:
强制改变过程中产生了NA
> t30yr<-xts(t30yr$DGS30,order.by=t30yr$date)
> names(t30yr)<-paste("DGS30")
> t30yr[1:3,]
           DGS30
2019-06-26  2.57
2019-06-27  2.52
2019-06-28  2.52
>
```

```
> ### US Treasury
> t3mo<-read.csv("DGS3MO.csv",header = TRUE)
> t3mo$date<-as.Date(t3mo$DATE,"%Y-%m-%d")
> t3mo$DGS3MO<-as.numeric(as.character(t3mo$DGS3MO))
警告信息:
强制改变过程中产生了NA
> t3mo<-xts(t3mo$DGS3MO,order.by=t3mo$date)
> names(t3mo)<-paste("DGS3MO")
> t3mo[1:3,]
           DGS3MO
2019-06-26   2.15
2019-06-27   2.14
2019-06-28   2.12
> t6mo<-read.csv("DGS6MO.csv",header = TRUE)
> t6mo$date<-as.Date(t6mo$DATE,"%Y-%m-%d")
> t6mo$DGS6MO<-as.numeric(as.character(t6mo$DGS6MO))
警告信息:
强制改变过程中产生了NA
> t6mo<-xts(t6mo$DGS6MO,order.by=t6mo$date)
> names(t6mo)<-paste("DGS6MO")
> t6mo[1:3,]
           DGS6MO
2019-06-26   2.12
2019-06-27   2.12
2019-06-28   2.09
>
> t1yr<-read.csv("DGS1.csv",header = TRUE)
> t1yr$date<-as.Date(t1yr$DATE,"%Y-%m-%d")
> t1yr$DGS1<-as.numeric(as.character(t1yr$DGS1))
警告信息:
强制改变过程中产生了NA
> t1yr<-xts(t1yr$DGS1,order.by=t1yr$date)
> names(t1yr)<-paste("DGS1")
> t1yr[1:3,]
           DGS1
2019-06-26 1.96
2019-06-27 1.93
2019-06-28 1.92
>
> t2yr<-read.csv("DGS2.csv",header = TRUE)
> t2yr$date<-as.Date(t2yr$DATE,"%Y-%m-%d")
> t2yr$DGS2<-as.numeric(as.character(t2yr$DGS2))
警告信息:
强制改变过程中产生了NA
> t2yr<-xts(t2yr$DGS2,order.by=t2yr$date)
> names(t2yr)<-paste("DGS2")
> t2yr[1:3,]
           DGS2
2019-06-26 1.77
2019-06-27 1.74
2019-06-28 1.75
>
```

```
> t3yr<-read.csv("DGS3.csv",header = TRUE)
> t3yr$date<-as.Date(t3yr$DATE,"%Y-%m-%d")
> t3yr$DGS3<-as.numeric(as.character(t3yr$DGS3))
警告信息:
强制改变过程中产生了NA
> t3yr<-xts(t3yr$DGS3,order.by=t3yr$date)
> names(t3yr)<-paste("DGS3")
> t3yr[1:3,]
           DGS3
2019-06-26 1.74
2019-06-27 1.71
2019-06-28 1.71
>
> t5yr<-read.csv("DGS5.csv",header = TRUE)
> t5yr$date<-as.Date(t5yr$DATE,"%Y-%m-%d")
> t5yr$DGS5<-as.numeric(as.character(t5yr$DGS5))
警告信息:
强制改变过程中产生了NA
> t5yr<-xts(t5yr$DGS5,order.by=t5yr$date)
> names(t5yr)<-paste("DGS5")
> t5yr[1:3,]
           DGS5
2019-06-26 1.80
2019-06-27 1.76
2019-06-28 1.76
>
> t7yr<-read.csv("DGS7.csv",header = TRUE)
> t7yr$date<-as.Date(t7yr$DATE,"%Y-%m-%d")
> t7yr$DGS7<-as.numeric(as.character(t7yr$DGS7))
警告信息:
强制改变过程中产生了NA
> t7yr<-xts(t7yr$DGS7,order.by=t7yr$date)
> names(t7yr)<-paste("DGS7")
> t7yr[1:3,]
           DGS7
2019-06-26 1.92
2019-06-27 1.88
2019-06-28 1.87
>
> t10yr<-read.csv("DGS10.csv",header = TRUE)
> t10yr$date<-as.Date(t10yr$DATE,"%Y-%m-%d")
> t10yr$DGS10<-as.numeric(as.character(t10yr$DGS10))
警告信息:
强制改变过程中产生了NA
> t10yr<-xts(t10yr$DGS10,order.by=t10yr$date)
> names(t10yr)<-paste("DGS10")
> t10yr[1:3,]
           DGS10
2019-06-26  2.05
2019-06-27  2.01
2019-06-28  2.00
>
```

Step 2: Combine Yield Data into One Data Object

```
> treasury<-t3mo
> treasury<-merge(treasury,t6mo)
> treasury<-merge(treasury,t1yr)
> treasury<-merge(treasury,t2yr)
> treasury<-merge(treasury,t3yr)
> treasury<-merge(treasury,t5yr)
> treasury<-merge(treasury,t7yr)
> treasury<-merge(treasury,t10yr)
> treasury<-merge(treasury,t20yr)
> treasury<-merge(treasury,t30yr)
> treasury[1:3,]
           DGS3MO DGS6MO DGS1 DGS2 DGS3 DGS5 DGS7 DGS10 DGS20 DGS30
2019-06-26   2.15   2.12 1.96 1.77 1.74 1.80 1.92  2.05  2.35  2.57
2019-06-27   2.14   2.12 1.93 1.74 1.71 1.76 1.88  2.01  2.31  2.52
2019-06-28   2.12   2.09 1.92 1.75 1.71 1.76 1.87  2.00  2.31  2.52
>
```

Step 3: Subset Data to Only Include Yields from 2020 to 2024-06-26

```
> extreme<-subset(treasury,index(treasury) >= "2020-01-01" &index(treasury) <= "2024-6-26")
> extreme<-na.omit(extreme[,c(1,8,10)])
> extreme[c(1:3,nrow(extreme)),]
           DGS3MO DGS10 DGS30
2020-01-02   1.54  1.88  2.33
2020-01-03   1.52  1.80  2.26
2020-01-06   1.56  1.81  2.28
2024-06-26   5.50  4.32  4.45
>
```

Step 4: Identify Examples of Different Shapes of the Yield Curve

```
> ### extreme inverted
> extreme$sign.diff<-extreme$DGS30-extreme$DGS3MO
> extreme$inverted<-ifelse(extreme$sign.diff==min(extreme$sign.diff),1,0
> inverted<-subset(extreme,extreme$inverted==1)
> inverted
           DGS3MO DGS10 DGS30 sign.diff inverted
2023-06-26    5.5  3.72  3.83     -1.67        1
> ### extreme upward
> extreme$upward<-ifelse(extreme$sign.diff==max(extreme$sign.diff),1,0)
> upward<-subset(extreme,extreme$upward==1)
> upward
           DGS3MO DGS10 DGS30 sign.diff inverted upward
2021-03-18   0.01  1.71  2.45      2.44        0      1
2021-03-19   0.01  1.74  2.45      2.44        0      1
> extreme$abs.diff<-abs(extreme$DGS30-extreme$DGS3MO)
> extreme$flat<-ifelse(extreme$abs.diff==min(extreme$abs.diff),1,0)
> flat<-subset(extreme,extreme$flat==1)
> flat$abs.diff2<-abs(flat$DGS30-flat$DGS10)
> flat$flat2<-ifelse(flat$abs.diff2==min(flat$abs.diff2),1,0)
> flat[,c(-4:-6)]
           DGS3MO DGS10 DGS30 abs.diff flat abs.diff2 flat2
2022-10-18   4.04  4.01  4.04        0    1      0.03     1
2022-10-31   4.22  4.10  4.22        0    1      0.12     0
2022-11-08   4.28  4.14  4.28        0    1      0.14     0
>
```

Step 5: Extract the Yield Curve on Dates Selected and on January 8th, 2020

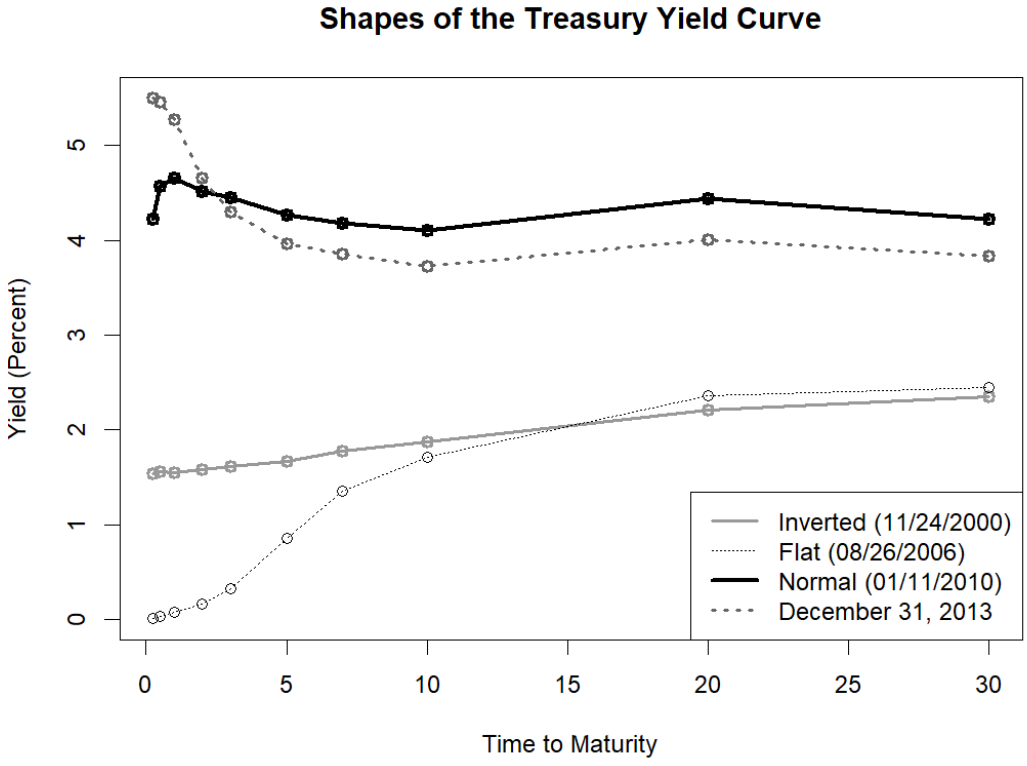
```
> invert.date<-as.Date("2023-06-26")
> normal.date<-as.Date("2021-03-18")
> flat.date<-as.Date("2022-10-31")
> current.date<-as.Date("2020-01-08")
>
> tyld.curve<-subset(treasury,
+                          + index(treasury) == invert.date |
+                            + index(treasury) == flat.date |
+                            + index(treasury) == normal.date |
+                            + index(treasury) == current.date)
> tyld.curve
           DGS3MO DGS6MO DGS1 DGS2 DGS3 DGS5 DGS7 DGS10 DGS20 DGS30
2020-01-08   1.54   1.56 1.55 1.58 1.61 1.67 1.78  1.87  2.21  2.35
2021-03-18   0.01   0.03 0.08 0.16 0.33 0.86 1.35  1.71  2.36  2.45
2022-10-31   4.22   4.57 4.66 4.51 4.45 4.27 4.18  4.10  4.44  4.22
2023-06-26   5.50   5.45 5.27 4.65 4.30 3.96 3.85  3.72  4.01  3.83
>
```

Step 6: Prepare Data for Plotting

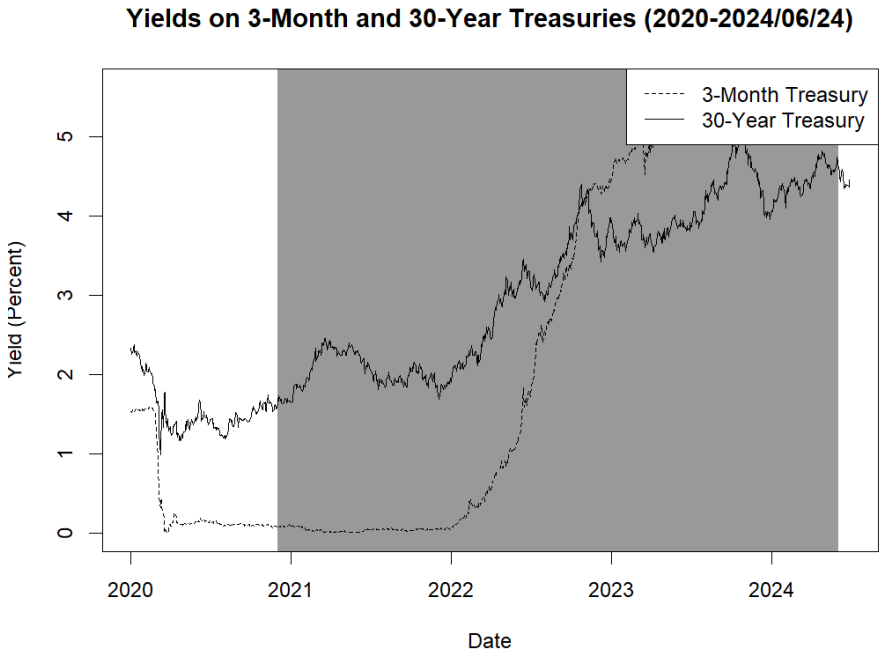
```
> class(tyld.curve)
[1] "xts" "zoo"
> tyld.curve<-t(tyld.curve)
> tyld.curve
       2020-01-08 2021-03-18 2022-10-31 2023-06-26
DGS3MO       1.54       0.01       4.22       5.50
DGS6MO       1.56       0.03       4.57       5.45
DGS1         1.55       0.08       4.66       5.27
DGS2         1.58       0.16       4.51       4.65
DGS3         1.61       0.33       4.45       4.30
DGS5         1.67       0.86       4.27       3.96
DGS7         1.78       1.35       4.18       3.85
DGS10        1.87       1.71       4.10       3.72
DGS20        2.21       2.36       4.44       4.01
DGS30        2.35       2.45       4.22       3.83
> class(tyld.curve)
[1] "matrix" "array"
> rownames(tyld.curve)<-paste(c(0.25,0.5,1,2,3,5,7,10,20,30))
> colnames(tyld.curve)<-paste(c("inverted","flat","normal","current"))
> tyld.curve
     inverted flat normal current
0.25     1.54 0.01   4.22    5.50
0.5      1.56 0.03   4.57    5.45
1        1.55 0.08   4.66    5.27
2        1.58 0.16   4.51    4.65
3        1.61 0.33   4.45    4.30
5        1.67 0.86   4.27    3.96
7        1.78 1.35   4.18    3.85
10       1.87 1.71   4.10    3.72
20       2.21 2.36   4.44    4.01
30       2.35 2.45   4.22    3.83
>
```

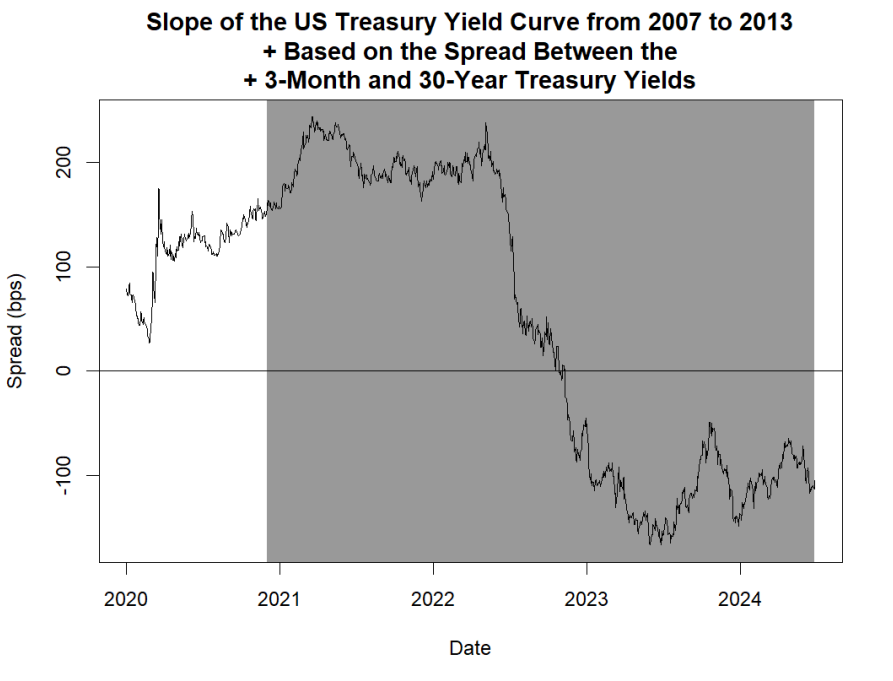
Step 7: Plot the Yield Curve on the Four Dates Selected

```
> plot(x=TTM,y=tyld.curve[,1],type="o",ylim=y.range,xlab="Time to Maturity",ylab="Yield (Percent)",col="gray60",lwd=2,
+ main="Shapes of the Treasury Yield Curve")
> lines(x=TTM,y=tyld.curve[,2],type="o",lty=3,col="black")
> lines(x=TTM,y=tyld.curve[,3],type="o",col="black",lwd=3)
> lines(x=TTM,y=tyld.curve[,4],type="o",lty=3,col="gray40",lwd=2)
> legend("bottomright",c("Inverted (11/24/2000)", "Flat (08/26/2006)","Normal (01/11/2010)", "December 31, 2013"),col=
c("gray60","black","black","gray40"), lty=c(1,3,1,3), lwd=c(2,1,3,2))
>
```



8.2.2 Slope of the US Treasury Yield Curve





8.3 Further Reading

An excellent overview of the bond markets can be found in Thau [5]. More details on bond calculations can be found in Fabozzi [3] and Tuckman and Serrat [6]. Two excellent books on bond strategies are Crescenzi [1] and Huggins and Schaller [4].

There are additional resources on the web from which we can obtain up-to-date fixed income information. US bond prices can be obtained from Yahoo Finance Bonds Center (http://bonds.yahoo.com). A wealth of information about bonds can be found on various regulatory websites. For example, the website of the Financial Industry Regulatory Authority (FINRA) (http://www.finra.org) contains, among other things, information on US corporate and agency data. Another regulator, the Municipal Securities Rulemaking Board (MSRB) provides information and market data through its Electronic Municipal Market Access (EMMA) website (http://emma.msrb.org/). Sterling (UK) corporate bond and bond index prices, as well as other information on Sterling bond markets can be obtained from Fixed Income Investor (http://www.fixed incomeinvestor.co.uk).

References

1. Crescenzi, A. (2010). The strategic bond investor: Strategies and tools to unlock the power of the bond market (2nd ed.). New York McGraw-Hill.

2. Estrella, A., & Trubin, M. (2006). The yield curve as a leading indicator: Some practical issues. Current Issues in Economics and Finance, 12, 1–7.

3. Fabozzi, F. (2009). Bond markets, analysis and strategies (7th ed.). New Jersey: Prentice-Hall.

4. Huggins, D., & Schaller, C. (2013). Fixed income relative value analysis: A practitioner's guide to the theory, tools, and trades. United Kingdom: Bloomberg Press.

5. Thau, A. (2011). The bond book (3rd ed.). New York McGraw-Hill.

6. Tuckman, B., & Serrat, A. (2012). Fixed income securities: Tools for today's markets (3rd ed.). New Jersey: Wiley.

Reference

Ang, C. S. (2015). *Analyzing financial data and implementing financial models using R.* : Springer Cham Heidelberg New York Dordrecht London.

Attachment of zips of R codes

Summer Research.zip