National College of Ireland

Project Submission Sheet – 2015/2016

School of Computing

Student Name:	Anicia Lafayette-Madden
Student ID:	15006590
Programme:	M.Sc Data Analytics **Year:** 2015-2016
Module:	Configuration Manual
Lecturer:	Vikas Sahni
Submission Due Date:	15/09/2016
Project Title:	Analysing historical stock market data to determine if a correlation exists between major stock market indexes and if time series is sufficient to make predictions.
Word Count:	2400

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:

Date: 16/09/2016

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on the computer. Please do not bind projects or place in covers unless specifically requested.
3 Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable)	

Chapter 1

Configuration Manual

1.1 SYSTEM SUMMARY

1.1.1 System Configuration

All tools used for the analyses in this research are located on the researcher's laptop. Below list the basic configurations that were needed for this research.

Operating System: Windows 10

RAM: 8 GB

Hard Disc: 1Terabyte

Processor: Intel® Core (TM) i3 processor.
System type: 64-bit, x64-based processor

1.2 GETTING STARTED

Download and install R – First, R was downloaded from CRAN (Comprehensive R Archive Network). To do this, https://cran.rstudio.com/ was visited and R package 3.3.1 for windows (64bits) was downloaded and installed.

Download and install R- studio – R-studio is a user interface for R, making it easier to use R and includes a code editor, debugging and visualization tools. To download and install, visit https://www.rstudio.com/ and follow the necessary instructions.

1.2.1 TOOLS USED

Tools used for analysis and forecasting of stock market price data include:

- R-studio – R-Studio's ease of use and the fact that it is an open source software made it ideal for use for in this research. It has installable packages that are ideal for almost any analytical test, is easy to learn the language and has fast analytical performance.
- Microsoft Excel 2016 – Very easy to use and create graphs.

1.3 Software overview

This research has chosen to utilize software tools and programming language such as R-studio, MS Excel as a means conducting the necessary analyses needed to complete this research.

R (3.3.1) and R-studio are both open source data analysis applications which ensure analysis can be reproduced enabling easy collaboration. Both requires some programming knowledge or skill and has many packages that minimize this which also provides advanced graphical capabilities. R-studio allows for flexibility creativity and originality in the analysis of the data. Packages installed include:

```
install.packages("dplyr")
library(dplyr)
install.packages("xts") ##(zoo package was needed for replacing mi data)
library(xts)
install.packages("forecast")
library(forecast)
install.packages("ggplot2")
library(ggplot2)
```

1.3.1 User Access Levels

Since R-studio is an open source application, it is available for download and use by any and everyone who wish to utilize it. Ensure CRAN libraries installed into R before attempting to conduct your analysis using R-studio. Also, the user must be aware that some packages have to be constantly reinstalled from R-studio panel before conducting the certain analysis, as they may not have been included in the CRAN libraries.

.

1.3.2 Installation

R and R-Studio were updated with their newest versions for this research. Instructions on their installation are briefly listed above in getting started or the intended user can visit https:cran.r-project.org/bin/windows/base along with the CRAN installation guide.

1.3.3 System Menu

System Implementation

Gathering all the necessary data - Stock market data was downloaded from Yahoo Finance website into CSV files. This included daily closing stock prices going back 24 years (1991 to 2015).

Nikkei 225 (^N225) - Osaka

16,650.57 ↑396.12(2.44%) 07:00

Historical Prices

Get Historical Prices for: GO

Set Date Range

Start Date: 01 Jan 1991 Eg. 1 Jan 2010

End Date: 31 Dec 2015

Daily

Weekly

Monthly

Dividends Only

Get Prices

First | Previous | Next | Las

Prices

Date	Open	High	Low	close	Volume	Adj Close*
5 Feb 1991	23,367.00	23,822.00	23,367.00	23,822.00	0	23,822.00
4 Feb 1991	23,179.00	23,294.00	23,135.00	23,287.00	0	23,287.00
1 Feb 1991	23,271.00	23,271.00	22,860.00	23,157.00	0	23,157.00
31 Jan 1991	23,477.00	23,655.00	23,222.00	23,293.00	0	23,293.00
30 Jan 1991	23,444.00	23,555.00	23,405.00	23,410.00	0	23,410.00
29 Jan 1991	23,563.00	23,609.00	23,422.00	23,460.00	0	23,460.00

***Yahoo Finance site**

Data Cleaning and transformation - Each CSV file was then transformed to only include the closing price and trading date for each stock market index.

Date	Dow	Date	SP	Date	Nasdaq	Date	FTSE	Date	Nikkei	Date	SSE
07/01/1991	2522.77	07/01/1991	315.44	07/01/1991	384.7	07/01/1991	2113.3	07/01/1991	23737	07/01/1991	132.06
08/01/1991	2509.41	08/01/1991	314.9	08/01/1991	384.18	08/01/1991	2099.9	08/01/1991	22898	08/01/1991	132.68
09/01/1991	2470.3	09/01/1991	311.49	09/01/1991	383.37	09/01/1991	2128.9	09/01/1991	22969	09/01/1991	133.34
10/01/1991	2498.76	10/01/1991	314.53	10/01/1991	392.16	10/01/1991	2108.7	10/01/1991	23047	10/01/1991	133.97
11/01/1991	2501.49	11/01/1991	315.23	11/01/1991	392.39	11/01/1991	2106.1	11/01/1991	23241	11/01/1991	134.6
14/01/1991	2483.91	14/01/1991	312.49	14/01/1991	387.54	14/01/1991	2080.8	14/01/1991	23213	14/01/1991	134.67
15/01/1991	2490.59	15/01/1991	313.73	15/01/1991	390.67	15/01/1991	2070.9	16/01/1991	22443	15/01/1991	134.74
16/01/1991	2508.91	16/01/1991	316.17	16/01/1991	405.91	16/01/1991	2054.8	17/01/1991	23447	16/01/1991	134.24
17/01/1991	2623.51	17/01/1991	327.97	17/01/1991	418.52	17/01/1991	2104.6	18/01/1991	23808	17/01/1991	134.25
18/01/1991	2646.78	18/01/1991	332.23	18/01/1991	422.99	18/01/1991	2102.7	21/01/1991	23352	18/01/1991	134.24
21/01/1991	2629.21	21/01/1991	331.06	21/01/1991	428.73	21/01/1991	2084	22/01/1991	23254	21/01/1991	134.24
22/01/1991	2603.22	22/01/1991	328.31	22/01/1991	423.95	22/01/1991	2081.6	23/01/1991	23050	22/01/1991	133.72
23/01/1991	2619.06	23/01/1991	330.21	23/01/1991	431.66	23/01/1991	2080.5	24/01/1991	23269	23/01/1991	133.17

***Each index closing price along with its corresponding trading date.**

Each CSV file containing each index with only its date and the closing price was imported into R-studio. Using the date attribute from the DowJones data as the base trading date, all six indexes were merged in order to properly align the trading dates and prices. This created missing information for the indexes that did not trade on some of the dates as the DowJones.

Date	Dow	SP	Nasdaq	FTSE	NIkkel	SSE
07/01/1991	2522.77	315.44	384.7	2113.3	23737	132.06
08/01/1991	2509.41	314.9	384.18	2099.9	22898	132.68
14/01/1991	2483.91	312.49	387.54	2080.8	23213	134.67
15/01/1991	2490.59	313.73	390.67	2070.9	NA	134.74
16/01/1991	2508.91	316.17	405.91	2054.8	22443	134.24
11/02/1991	2902.23	368.58	498.61	2279	NA	130.97
12/02/1991	2874.75	365.5	496.4	2264.5	24935	131.35
13/02/1991	2909.16	369.02	500.67	2267.8	25139	131.92
14/02/1991	2877.23	364.22	491.18	2294.4	25356	132.53
15/02/1991	2934.65	369.06	496.67	2296.9	25344	NA
20/03/1991	2872.03	367.92	512.18	2441.2	26449	123.66
21/03/1991	2855.45	366.58	505.94	2474.8	NA	123.12
22/03/1991	2858.91	367.48	504.54	2440.5	26613	122.62
26/04/1991	2912.38	379.02	542.6	2471.3	26124	115.56
29/04/1991	2876.98	373.66	532.29	2498.2	NA	114.75
30/04/1991	2887.87	375.34	527.31	2486.2	26111	113.94
01/05/1991	2930.2	380.29	528.44	2508.4	26489	NA
02/05/1991	2938.61	380.52	533.03	2530.7	26478	113.16
03/05/1991	2938.86	380.8	534.22	2522.7	NA	112.41
06/05/1991	2941.64	380.08	534.66	2522.7	NA	111.61

***Merged data**

Dates for which there was data, the assumption of a linear change was taken instead of no change at all. For this, the average of the price of the day above and below was used as a replacement.

- **stock_full3$Mean <- rowMeans(stock_full3[,2:7])**

Date	Dow	SP	Nasdaq	FTSE	Nikkei	SSE	Mean
07/01/1991	2522.77	315.44	384.7	2113.3	23737	132.06	4867.545
08/01/1991	2509.41	314.9	384.18	2099.9	22898	132.68	4723.178
14/01/1991	2483.91	312.49	387.54	2080.8	23213	134.67	4768.735
15/01/1991	2490.59	313.73	390.67	2070.9	22828	134.74	4704.772
16/01/1991	2508.91	316.17	405.91	2054.8	22443	134.24	4643.838
11/02/1991	2902.23	368.58	498.61	2279	24615.5	130.97	5132.482
12/02/1991	2874.75	365.5	496.4	2264.5	24935	131.35	5177.917
13/02/1991	2909.16	369.02	500.67	2267.8	25139	131.92	5219.595
14/02/1991	2877.23	364.22	491.18	2294.4	25356	132.53	5252.593
15/02/1991	2934.65	369.06	496.67	2296.9	25344	132.83	5262.352
20/03/1991	2872.03	367.92	512.18	2441.2	26449	123.66	5460.998
21/03/1991	2855.45	366.58	505.94	2474.8	26531	123.12	5476.148
22/03/1991	2858.91	367.48	504.54	2440.5	26613	122.62	5484.508
26/04/1991	2912.38	379.02	542.6	2471.3	26124	115.56	5424.143
29/04/1991	2876.98	373.66	532.29	2498.2	26117.5	114.75	5418.897
30/04/1991	2887.87	375.34	527.31	2486.2	26111	113.94	5416.943
01/05/1991	2930.2	380.29	528.44	2508.4	26489	113.55	5491.647
02/05/1991	2938.61	380.52	533.03	2530.7	26478	113.16	5495.67
03/05/1991	2938.86	380.8	534.22	2522.7	26432.67	112.41	5486.943
06/05/1991	2941.64	380.08	534.66	2522.7	26387.33	111.61	5479.671

The average values calculated were then used as a scale to convert all the data into percentages in excel thus normalizing the data for further use against the forecasting model.

- **stock_full3$dow <- (stock_full3$Dow/stock_full3$Mean)*100**

Date	dow	sp	nas	ftse	nikkei	sse
07/01/1991	51.82839	6.480474	7.903368	43.41614	487.6586	2.713072
08/01/1991	53.12969	6.667121	8.13393	44.45947	484.8007	2.809125
09/01/1991	52.19605	6.581609	8.100393	44.98246	485.3221	2.817399
10/01/1991	52.61448	6.622818	8.257414	44.40129	485.2831	2.820904
11/01/1991	52.31271	6.592285	8.205903	44.04407	486.0302	2.814839
14/01/1991	52.0874	6.552891	8.126683	43.63421	486.7748	2.824019
15/01/1991	52.93753	6.668336	8.303697	44.01701	485.2095	2.863901
16/01/1991	54.02664	6.808377	8.74083	44.24788	483.2856	2.890712
17/01/1991	54.17518	6.772543	8.64239	43.45975	484.1779	2.772247
18/01/1991	53.92981	6.769396	8.618688	42.84384	485.103	2.735225
21/01/1991	54.47401	6.859158	8.882761	43.17793	483.8249	2.781288
22/01/1991	54.18709	6.833907	8.824693	43.32936	484.0415	2.783436
23/01/1991	54.85976	6.916696	9.041704	43.57889	482.8135	2.789426
24/01/1991	54.83864	6.946044	9.120236	43.55646	482.7872	2.751404
25/01/1991	54.56028	6.89479	9.068659	43.14501	483.6221	2.709129
28/01/1991	54.44408	6.892115	9.116469	43.44106	483.41	2.696299
29/01/1991	54.80117	6.912149	9.239936	43.50554	482.846	2.69517

***Normalized data**

A baseline/scale value was calculated by finding the average across all the indexes for each date. This baseline value was then used to scale all the data by converting each data point into percentages of that baseline value providing a more even distribution. This resulted in the data being normalized which is thought to be important as this has now brought all the indexes into a proportion with each other has helped to reduce the variability.

Correlation Implementation

First cross-correlation matrix was used to assess the level of correlation between all stock market index prices.

- **stockcor5<-cor(as.matrix(stock_full4))**

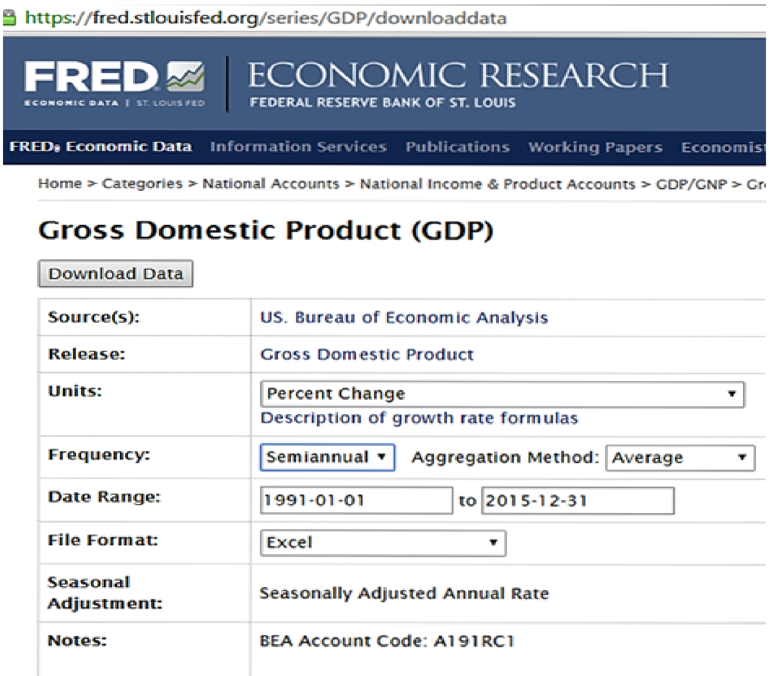
	dow	s.p	nas	ftse	nikkei	sse
dow	1	0.978681	0.766	0.828	-0.985	0.67
s.p	0.978681	1	0.826	0.841	-0.974	0.608
nas	0.766352	0.826342	1	0.568	-0.804	0.498
ftse	0.827658	0.840752	0.568	1	-0.862	0.566
nikkei	-0.98461	-0.97418	-0.804	-0.862	1	-0.749
sse	0.67015	0.60798	0.498	0.566	-0.749	1

Ranges	
above zero - 0.399	weak
0.4 - 0.699	moderate
0.7 - 0.999	strong

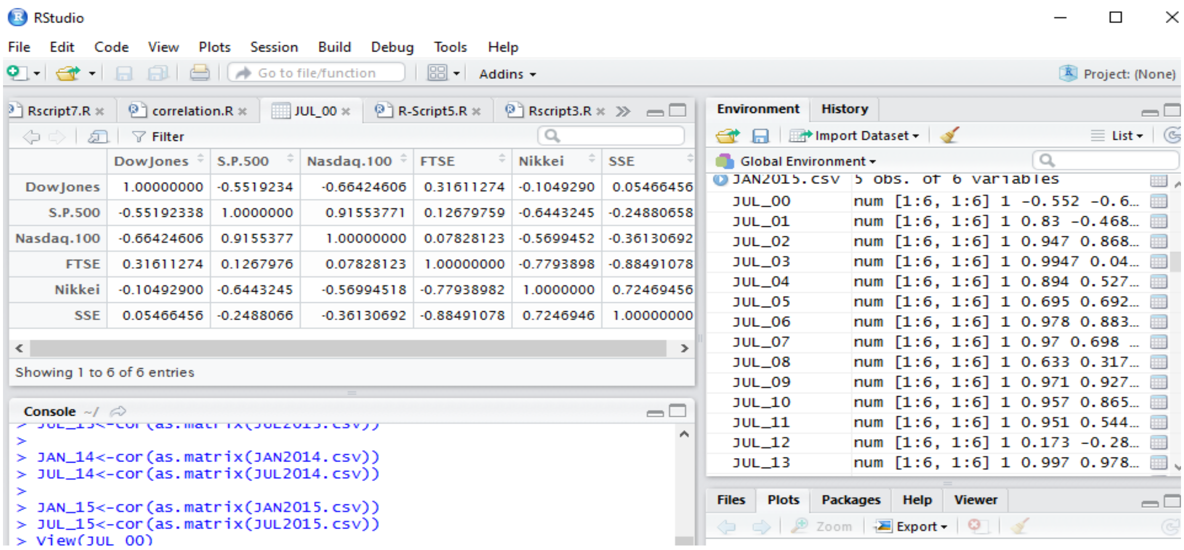
***Correlation Matrix**

Correlation based on the DowJones and US GDP growth rate

US GDP growth data was first extracted from FRED Economic Data at https://fred.stlouisfed.org/series/GDP/downloaddata into one CSV file and saved to the machine's' desktop.



Fifty excel files were then created in order to complete this analysis, with each excel file contained the six stock market indexes along with five rows of data relating to the date the GDP value was released. Each data file created was based on the date of the economic data release, where two days before and after were taken as data to be analyzed. All fifty file were then loaded into R where a cross-correlation matrix was conducted on all 50 datasets.



Generating cross-correlation matrix for all 50 files

Results from correlation matrix were merged in an excel file with the original GDP for further observation and visual analysis.

DATE	% CHANGE IN GDP GROWTH	FTSE	NIKKEI	SSE
1991-01-01	1.2	0.46861	-0.83677	0.62731
1991-07-01	2.5	0.96555	-0.97902	0.58811
1992-01-01	3.0	-0.28775	-0.68451	0.96032
1992-07-01	3.2	0.95978	-0.99082	0.91051
1993-01-01	2.2	-0.90923	-0.15766	-0.14514
1993-07-01	2.6	0.94053	-0.99587	0.9143
1994-01-01	3.4	0.47301	0.63199	0.66872
1994-07-01	2.9	0.61613	-0.99362	0.37995
1995-01-01	2.2	-0.60244	-0.96167	0.64123
1995-07-01	2.3	-0.08528	-0.81511	-0.27157
1996-01-01	2.9	0.3823	-0.75954	0.53938
1996-07-01	3.1	0.72445	-0.94707	-0.87478
1997-01-01	3.1	-0.28872	-0.84089	-0.12341
1997-07-01	3.1	0.97437	-0.97908	-0.88077
1998-01-01	2.3	0.69575	-0.86599	0.68102
1998-07-01	3.3	0.90776	-0.99546	0.96052
1999-01-01	2.9	0.89933	-0.95144	-0.8808
1999-07-01	3.3	0.24584	-0.93927	-0.35071
2000-01-01	3.4	0.23041	-0.77672	-0.28185
2000-07-01	2.6	0.31611	-0.10493	0.05466
2001-01-01	1.5	-0.96416	-0.86336	0.30063
2001-07-01	0.9	0.56887	-0.95607	0.4552
2002-01-01	2.0	0.32071	-0.88984	0.29631
2002-07-01	1.7	0.10926	-0.86962	0.16414
2003-01-01	2.1	-0.35776	-0.9734	-0.68944
2003-07-01	3.7	0.66566	-0.90948	0.72546
2004-01-01	3.1	-0.54876	-0.77692	-0.15294
2004-07-01	3.1	0.51943	-0.80785	-0.17997

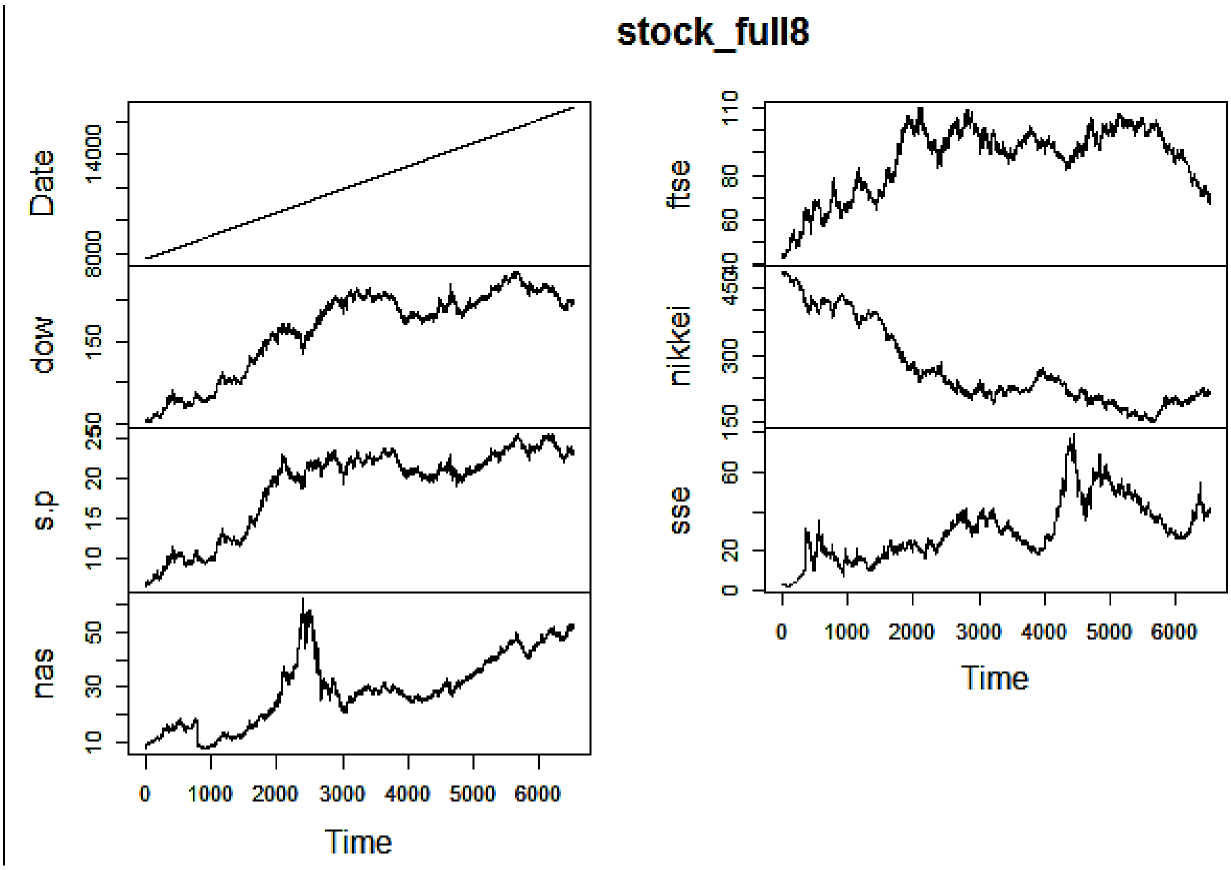
DATE	% CHANGE IN GDP GROWTH	FTSE	NIKKEI	SSE
2005-01-01	3.4	-0.97426	-0.99547	0.79965
2005-07-01	3.1	-0.3761	-0.97243	0.26075
2006-01-01	3.2	-0.3105	-0.67761	-0.50681
2006-07-01	1.9	0.24615	-0.99465	0.12502
2007-01-01	2.4	-0.59641	0.81938	-0.85995
2007-07-01	2.1	-0.36418	-0.63687	0.0745
2008-01-01	0.8	-0.62614	0.1623	-0.5701
2008-07-01	-0.3	-0.55495	-0.23605	-0.58885
2009-01-01	-2.3	-0.48297	-0.86356	-0.8344
2009-07-01	0.8	0.47304	-0.75324	-0.78903
2010-01-01	2.1	-0.69252	-0.71068	-0.31848
2010-07-01	2.4	-0.17553	-0.85814	0.02561
2011-01-01	1.4	-0.86474	-0.73884	0.15313
2011-07-01	2.2	0.37638	-0.8125	-0.81972
2012-01-01	2.3	-0.68342	-0.56094	-0.0559
2012-07-01	1.3	-0.8131	-0.33258	0.56118
2013-01-01	1.5	-0.91689	-0.84012	-0.35841
2013-07-01	2.2	0.72882	-0.98716	-0.60342
2014-01-01	1.7	0.52922	-0.82544	-0.20936
2014-07-01	2.8	0.702	-0.95038	0.20538
2015-01-01	1.5	0.61442	-0.68774	-0.70426
2015-07-01	1.6	0.647	0.44861	-0.73729

Three non-US stock markets were used to examine correlation movement with the Dowjones based on the release date of US GDP growth data. Graphs for examination were created in excel.

ARIMA Implementation

Steps:

1. **Visualize the time series –**The data is plotted to identify and understand trends (seasonality). The plots indicated a few sudden changes and indicated no real abnormal changes. Sudden drops were not considered particularly significant and may be attributed to global economic factors.



*Time scale is based on date count

2. **Check stationarity of the data and Plot ACF and PACF –** The data was tested to ensure stationarity. The augmented Dickey-Fuller test was applied in this instance. Each stock index being used in this research was taken as a separate variable and the Dickey-Fuller test applied to each. The Dickey-Fuller test is used to test the null hypothesis of no stationarity of an ARIMA process against the alternative that stationarity does exist (Cheung and Lai, 1995). ACF and PACF plots are done to determine the optimal parameters and possible candidates for the models. They also visually show the stationarity of the variables being forecast and can plot along with performing the Dickey-Fuller. The results were as follow:

Null hypothesis: There is no stationarity

Alternative hypothesis: There is stationarity

> adf.test(x, alternative="stationary")

data: stock_train2$dow

Dickey-Fuller = -1.6634, Lag order= 17, p-**value = 0.7209**

alternative hypothesis: stationary

data: stock_train2$s.p

Dickey-Fuller = -1.5345, Lag order=17, **p-value = 0.7755**

alternative hypothesis: stationary

data: stock_train2$nas

Dickey-Fuller = -1.7793, Lag order =17, **p-value = 0.6718**

alternative hypothesis: stationary

data: stock_train2$ftse

Dickey-Fuller = -2.6041, Lag order=17, **p-value = 0.3225** alternative hypothesis: stationary

data: stock_train2$nikkei

Dickey-Fuller = -1.887, Lag order = 17, **p-value = 0.6262**

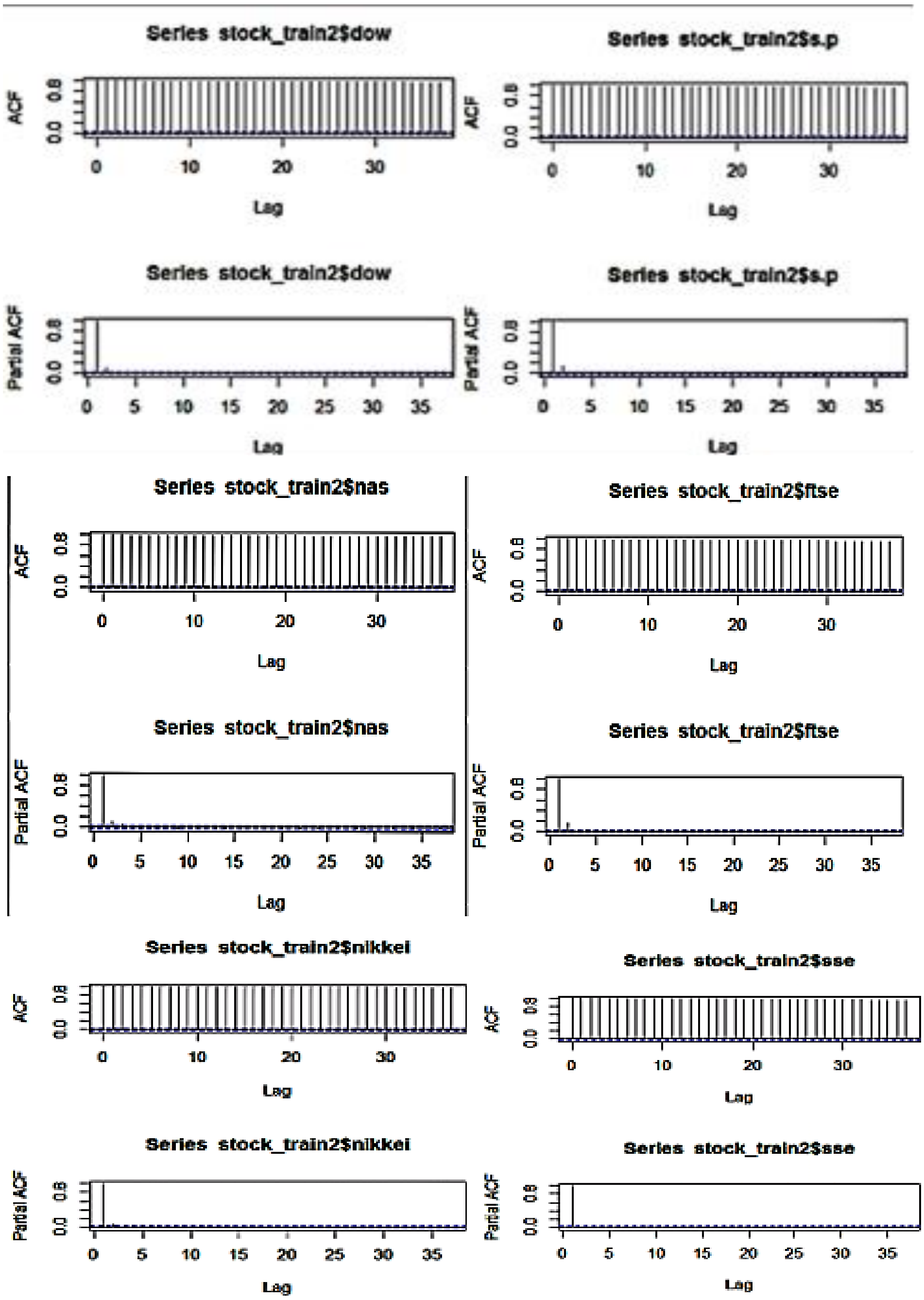
alternative hypothesis: stationary

data: stock_train2$sse

Dickey-Fuller = -2.993, Lag order = 17, **p-value = 0.1578**

alternative hypothesis: stationary

The results of the Dickey-Fuller test revealed all the variables having a large p-value, which resulted in the null hypothesis not being rejected, thus is not stationary.



The ACF plots for each variable confirms Dickey-Fuller test results, also indicating non-stationarity, showing ACF not tailing off quickly. This meant that the application of differencing was needed in order to get it stationary. Differencing function in R was used which works by taking each observation and differencing it from the one previous to it. Another Dickey-Fuller test was then reapplied to check for stationarity along with ACF and PACF plots. The results are as follow:

>adf.test(d.x, alternative="stationary")

data: d.dowf2

Dickey-Fuller = -18.755, Lag order = 17, **p-value = 0.01**

alternative hypothesis: stationary

data: d.s.pf2

Dickey-Fuller = -18.186, Lag order = 17, **p-value = 0.01**

alternative hypothesis: stationary

data: d.nasf2

Dickey-Fuller = -17.218, Lag order = 17, **p-value = 0.01**

alternative hypothesis: stationary

data: d.ftsef2

Dickey-Fuller = -18.398, Lag order = 17, **p-value = 0.01**

alternative hypothesis: stationary

data: d.nikkeif2

Dickey-Fuller = -17.674, Lag order = 17, **p-value = 0.01**

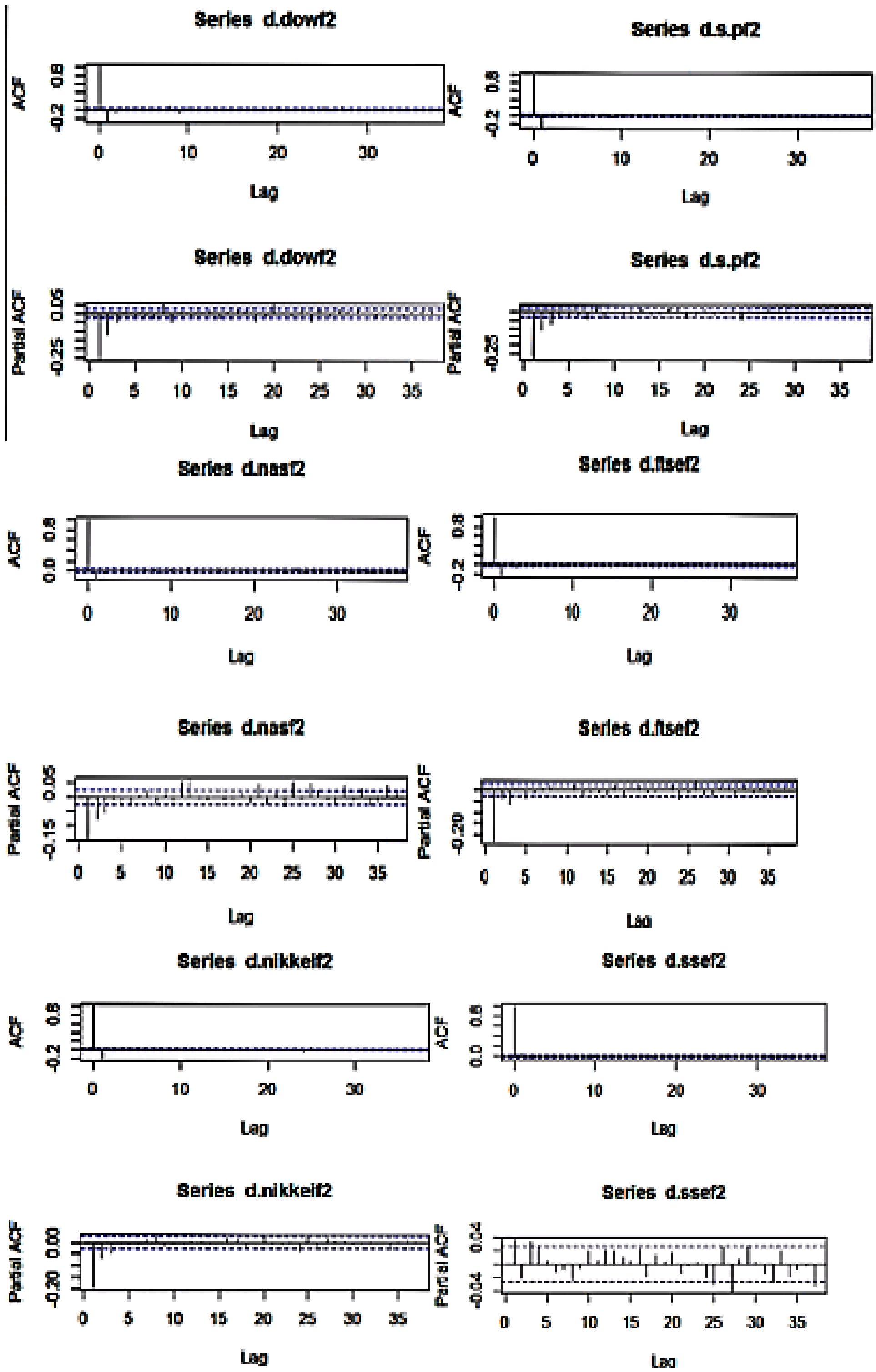
alternative hypothesis: stationary

data: d.ssef2

Dickey-Fuller = -16.397, Lag order = 17, **p-value = 0.01**

alternative hypothesis: stationary

The results of the Dickey-Fuller test now shows all variables having a very small p-value, which resulted in the null hypothesis being rejected and the conclusion that stationarity does exist.



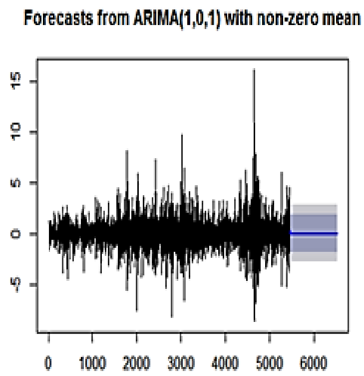
Again the ACF plots confirm the Dickey-Fuller test showing that there is stationarity with the data. The plots for each index shows significant trends within the data.

3. **Build Model**- The ACF and PACF plots were used to select the optimal models for fitting the data. Here it was determined that the ARIMA (1,0,1) and ARIMA (2,0,2) will be used to find the best model fit.

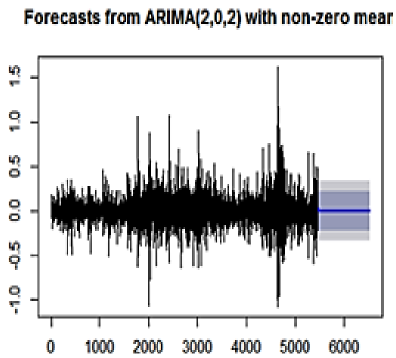
ARIMA 1,0,1	ARIMA 2,0,2
arima(x = stock_dow, order = c(1, 0, 1)) AIC=19133.22	arima(x = stock_dow, order = c(2, 0, 2)) AIC = 19135.77
arima (x = stock_sp, order = c(1, 0, 1)) AIC = -4371.56	**arima(x = stock_sp, order = c(2, 0, 2)) AIC = -4376.64**
arima(x = stock_nas, order = c(1, 0, 1)) AIC=8045.34	arima(x = stock_nas, order = c(2, 0, 2)) AIC = 8047.65
arima(x = stock_ftse, order = c(1, 0, 1)) AIC = 13357.22	**arima(x = stock_ftse, order = c(2, 0, 2)) AIC=13340.58**
arima(x = stock_nikkei, order = c(1, 0, 1)) AIC=23589.29	arima(x = stock_nikkei, order = c(2, 0, 2)) AIC = 23591.38
arima(x = stock_sse, order = c(1, 0, 1)) AIC=11211.23	arima(x = stock_sse, order = c(2, 0, 2)) AIC = 11215.37

The AIC values for all ARIMA models were compared with the model having the smallest AIC value chosen as the best model. ARIMA (1,0,1) was determined better suited for 4 of the Dow, Nasdaq, Nikkei and SSE, while ARIMA (2,0,2) worked best for the S&P and FTSE.

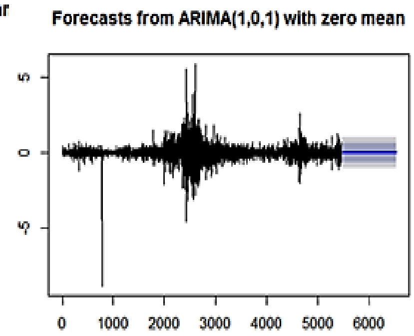
4. **Make Prediction**



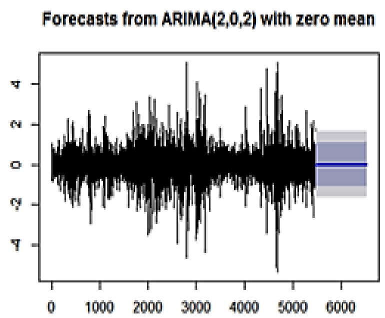
DOWJONES



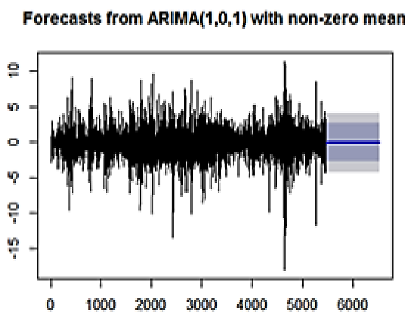
S&P



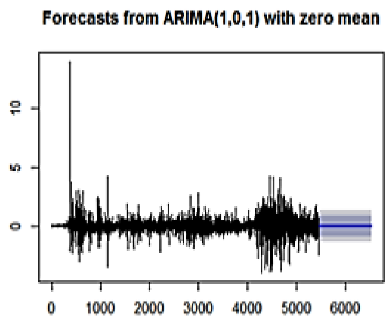
NASDAQ



FTSE



NIKKEI



SSE

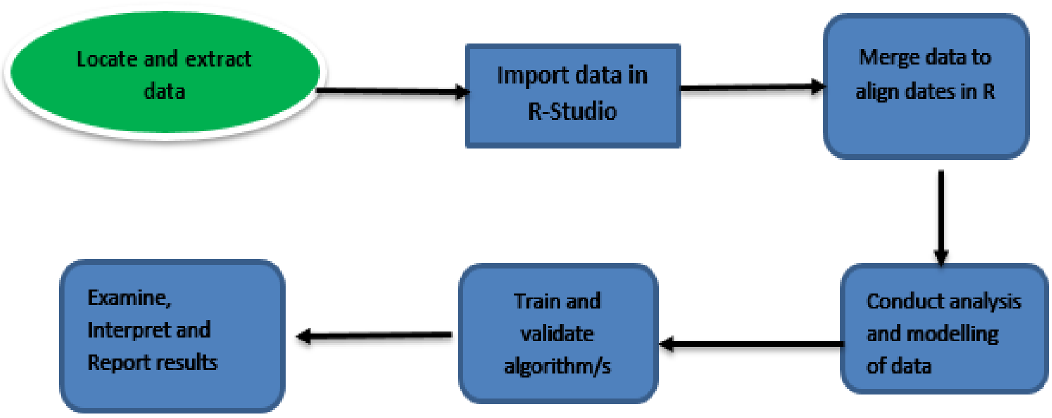
Results of the ARIMA forecast showing a straight line. This is not the results that were anticipated, however, it is a result.

Using the System

ARIMA – ARIMA's implementation required a number of steps to be followed in order to successfully analyze the data and make a forecast.

- First, the normalized data was plotted to identify trends that may exist.
- Second, since ARIMA requires stationarity of the data, Dickey-Fuller test was applied along with ACF and PACF plots to check for stationarity. The Dickey-Fuller test was performed using **adf.test(x, alternative="stationary"),** without specifying the number of additional lags **k** as its inclusion did not make a difference on the results. After the Dickey-Fuller. Differencing was also conducted where data was found to be non-stationary, with a second Dickey-Fuller test being applied afterward.
- Third, models are built based on results from the plot and the optimal model fits chosen based on the best AIC calculated by each model. The models were trained on data spanning 20 years and then 4 years of data used to validate prediction accuracy.
- Forecast time series.
- Evaluation of results will be done by accessing the results from the forecast plots and by comparing forecast figures with validation set figures.

Design Workflow



Flow Description

R-Script

Anicia Lafayette-Madden
15006590
MSc in Data Analytics

##IMPORT STOCK FILES INTO R (6 files)

```
folder <- "C:/Users/Nerine/Desktop/STOCK/"      # path to folder that holds multiple .csv files
file_list <- list.files(path=folder, pattern="*.csv") # create list of all .csv files in folder

# read in each .csv file in file_list and create a data frame with the same name as the .csv file
for (i in 1:length(file_list)){
  assign(file_list[i],
        read.csv(paste(folder, file_list[i], sep=''))
  )}
```

##Install package for use

```
install.packages("tseries")
library('tseries')
install.packages("dplyr")
install.packages("xts") ##(zoo package was needed for replacing mi data)
library(xts)
install.packages("forecast")
library(forecast)
library(dplyr)
library(ggplot2)
```

##Merge stock index prices by date

```
stock1 <- merge(Dow.csv, SP.csv, by="Date", all.x = TRUE)
stock2<- merge(stock1, Nasdaq.csv, by="Date", all.x = TRUE)
stock3 <- merge(stock2, FTSE.csv, by="Date", all.x = TRUE)
stock4<- merge(stock3, Nikkei.csv, by="Date", all.x = TRUE)
stock_full <- merge(stock4, SSE.csv, by="Date", all.x = TRUE)
```

Put in order of date and convert date variable from factor to date format:

```
stock_full2<- stock_full[order(as.Date(stock_full$Date, format="%d/%m/%Y")),]
```

##Replace missing values with mean of the row above and below

```
stock_full2$FTSE <- na.approx(stock_full2$FTSE)
stock_full2$Nikkei <- na.approx(stock_full2$Nikkei)
stock_full2$SSE <- na.approx(stock_full2$SSE)
```

##Calculate mean for each day for all the indexes

```
stock_full2$Mean <- rowMeans(stock_full2[,2:7])
```

##Convert data points to a percentage value of the mean

```
stock_full3<- stock_full2     ### making a copy
stock_full3$dow <- (stock_full3$Dow/stock_full3$Mean)*100
stock_full3$sp<- (stock_full3$SP/stock_full3$Mean)*100
stock_full3$nas <- (stock_full3$Nasdaq/stock_full3$Mean)*100
stock_full3$ftse <- (stock_full3$FTSE/stock_full3$Mean)*100
stock_full3$nikkei <- (stock_full3$Nikkei/stock_full3$Mean)*100
stock_full3$sse <- (stock_full3$SSE/stock_full3$Mean)*100
```

##Omit columns not needed create one copy with the date column

```
stock_full4 <- stock_full3[c(9,10,11,12,13,14)]
stock_full5 <- stock_full3[c(1,9,10,11,12,13,14)]
```

##Generate correlation matrix
stockcor5<-cor(as.matrix(stock_full4))

##Split data into training and validation sets
stock_train <- stock_full5[1:5291,]
stock_val <- stock_full5[5292:6296,]
stock_train2 <- stock_train ## make a copy

ARIMA IMPLEMENTATION

Steps:

1. **##Visualize the time series**

stock_plot <- stock_full5 **## make copy of table for plotting**
plot.ts(stock_plot, main = "Time Series plot")

2. Check stationarity of the data using Dickey-Full test and generate ACF/PACF plots

##Plot each index

```
par(mfrow=c(2,1))
acf(stock_train2$dow)
pacf(stock_train2$dow)

par(mfrow=c(2,1))
acf(stock_train2$sp)
pacf(stock_train2$sp)

par(mfrow=c(2,1))
acf(stock_train2$nas)
pacf(stock_train2$nas)

par(mfrow=c(2,1))
acf(stock_train2$ftse)
pacf(stock_train2$ftse)

par(mfrow=c(2,1))
acf(stock_train2$nikkei)
pacf(stock_train2$nikkei)

par(mfrow=c(2,1))
acf(stock_train2$sse)
pacf(stock_train2$sse)
```

##Apply Dickey-Fuller test to check if series if stationary

```
adf.test(stock_train2$dow, alternative="stationary")
adf.test(stock_train2$sp, alternative="stationary")
adf.test(stock_train2$nas, alternative="stationary")
adf.test(stock_train2$ftse, alternative="stationary")
adf.test(stock_train2$nikkei, alternative="stationary")
adf.test(stock_train2$sse, alternative="stationary")
```

##Apply differencing to non-stationary series to make it stationary

```
dow2<- stock_train2$dow
d.dowf2 <- diff(dow2)
sp2<- stock_train2$sp
d.spf2 <- diff(sp2)
nas2<- stock_train2$nas
d.nasf2 <- diff(nas2)
ftse2<- stock_train2$ftse
d.ftsef2 <- diff(ftse2)
nikkei2<- stock_train2$nikkei
d.nikkeif2 <- diff(nikkei2)
sse2<- stock_train2$sse
d.ssef2 <- diff(sse2)
```

#Re-apply Dickey-Fuller test to check data again if stationary

```
adf.test(d.dowf2, alternative="stationary")
adf.test(d.spf2, alternative="stationary")
```

```
adf.test(d.nasf2, alternative="stationary")
adf.test(d.ftsef2, alternative="stationary")
adf.test(d.nikkeif2, alternative="stationary")
adf.test(d.ssef2, alternative="stationary")
```

Plot ACF and PACF for difference data

```
par(mfrow=c(2,1))
acf(d.dowf2)
pacf(d.dowf2)

par(mfrow=c(2,1))
acf(d.spf2)
pacf(d.spf2)

par(mfrow=c(2,1))
acf(d.nasf2)
pacf(d.nasf2)

par(mfrow=c(2,1))
acf(d.ftsef2)
pacf(d.ftsef2)

par(mfrow=c(2,1))
acf(d.nikkeif2)
pacf(d.nikkeif2)

par(mfrow=c(2,1))
acf(d.ssef2)
pacf( d.ssef
```

3. Build Model – Build three model each using the differenced data. Then choose the best one based on its AIC value.

##ARIMA 1,0,1

```
fit_dow2 <- arima(d.dowf2, c(1, 0, 1))
print(fit_dow2)

fit_sp2 <- arima(d.spf2, c(1, 0, 1))
print(fit_sp2)

fit_nas2 <- arima(d.nas2, c(1, 0, 1))
print(fit_nas2)

fit_ftse2 <- arima(d.ftsef2, c(1, 0, 1))
print(fit_ftse2)

fit_nikkei2 <- arima(d.nikkeif2, c(1, 0, 1))
print(fit_nikkei2)

fit_sse2 <- arima(d.ssef2, c(1, 0, 1))
print(fit_sse2)
```

##ARIMA 2,0,2

```
fit_dow3 <- arima(d.dowf2, c(2, 0, 2))
print(fit_dow3)

fit_sp3 <- arima(d.spf2, c(2, 0, 2))
print(fit_sp2)

fit_nas3 <- arima(d.nasf2, c(2, 0, 2))
print(fit_nas3)

fit_ftse3 <- arima(d.ftsef2, c(2, 0, 2))
print(fit_ftse2)

fit_nikkei3 <- arima(d.nikkeif2, c(2, 0, 2))
print(fit_nikkei3)

fit_sse3 <- arima(d.ssef2, c(2, 0, 2))
print(fit_sse3)
```

4. Prediction

##Forecast

```
forecast<- forecast(fit_dow2, h=1005)  # h indicating the number of days being forecast
forecast2<- forecast(fit_sp3, h=1005)
forecast3<- forecast(fit_nas2, h=1005)
forecast4<- forecast(fit_ftse3, h=1005)
forecast5<- forecast(fit_nikkei2, h=1005)
forecast6<- forecast(fit_sse2, h=1005)
```

##Plot Forecast

```
forecast<- plot(forecast(fit_dow2, h=1005))
forecast2<- plot(forecast(fit_sp3, h=1005))
forecast3<- plot(forecast(fit_nas2, h=1005))
forecast4<- plot(forecast(fit_ftse3, h=1005))
```

```
forecast5<- plot(forecast(fit_nikkei2, h=1005))
forecast6<- plot(forecast(fit_sse2, h=1005))
```

CORRELATION IMPLEMENTATION

```
##IMPORT FILES ON GDP FIGURES INTO R (50 files)
folder <- "C:/Users/Nerine/Desktop/GDP/"      # path to folder that holds multiple .csv files
file_list <- list.files(path=folder, pattern="*.csv") # create list of all .csv files in folder

# read in each .csv file in file_list and create a data frame with the same name as the .csv file
for (i in 1:length(file_list)){
  assign(file_list[i],
        read.csv(paste(folder, file_list[i], sep=''))
  )}

##Generate cross-correlation matrix for all 50 data frames
JAN_91<-cor(as.matrix(JAN1991.csv))
JUL_91<-cor(as.matrix(JUL1991.csv))

JAN_92<-cor(as.matrix(JAN1992.csv))
JUL_92<-cor(as.matrix(JUL1992.csv))

JAN_93<-cor(as.matrix(JAN1993.csv))
JUL_93<-cor(as.matrix(JUL1993.csv))

JAN_94<-cor(as.matrix(JAN1994.csv))
JUL_94<-cor(as.matrix(JUL1994.csv))

JAN_95<-cor(as.matrix(JAN1995.csv))
JUL_95<-cor(as.matrix(JUL1995.csv))

JAN_96<-cor(as.matrix(JAN1996.csv))
JUL_96<-cor(as.matrix(JUL1996.csv))

JAN_97<-cor(as.matrix(JAN1997.csv))
JUL_97<-cor(as.matrix(JUL1997.csv))

JAN_98<-cor(as.matrix(JAN1998.csv))
JUL_98<-cor(as.matrix(JUL1998.csv))

JAN_99<-cor(as.matrix(JAN1999.csv))
JUL_99<-cor(as.matrix(JUL1999.csv))

JAN_00<-cor(as.matrix(JAN2000.csv))
JUL_00<-cor(as.matrix(JUL2000.csv))

JAN_01<-cor(as.matrix(JAN2001.csv))
JUL_01<-cor(as.matrix(JUL2001.csv))

JAN_02<-cor(as.matrix(JAN2002.csv))
JUL_02<-cor(as.matrix(JUL2002.csv))

JAN_03<-cor(as.matrix(JAN2003.csv))
JUL_03<-cor(as.matrix(JUL2003.csv))

JAN_04<-cor(as.matrix(JAN2004.csv))
JUL_04<-cor(as.matrix(JUL2004.csv))

JAN_05<-cor(as.matrix(JAN2005.csv))
JUL_05<-cor(as.matrix(JUL2005.csv))

JAN_06<-cor(as.matrix(JAN2006.csv))
JUL_06<-cor(as.matrix(JUL2006.csv))

JAN_07<-cor(as.matrix(JAN2007.csv))
JUL_07<-cor(as.matrix(JUL2007.csv))

JAN_08<-cor(as.matrix(JAN2008.csv))
JUL_08<-cor(as.matrix(JUL2008.csv))

JAN_09<-cor(as.matrix(JAN2009.csv))
JUL_09<-cor(as.matrix(JUL2009.csv))

JAN_10<-cor(as.matrix(JAN2010.csv))
JUL_10<-cor(as.matrix(JUL2010.csv))

JAN_11<-cor(as.matrix(JAN2011.csv))
JUL_11<-cor(as.matrix(JUL2011.csv))

JAN_12<-cor(as.matrix(JAN2012.csv))
JUL_12<-cor(as.matrix(JUL2012.csv))

JAN_13<-cor(as.matrix(JAN2013.csv))
JUL_13<-cor(as.matrix(JUL2013.csv))

JAN_14<-cor(as.matrix(JAN2014.csv))
JUL_14<-cor(as.matrix(JUL2014.csv))

JAN_15<-cor(as.matrix(JAN2015.csv))
JUL_15<-cor(as.matrix(JUL2015.csv))
```