IDS 575 Business Analytic Statistics

Price Forecasting and Analysis of Exchange Traded Fund

Submitted By:

Nandini Poddar	-	652404994
Vatsal Shah	-	653763850
Anurag Banerjee	-	677398706

<u>INDEX</u>:

1. Introduction

1.1. Dataset Description

An ETF, or exchange-traded fund, is a marketable security that tracks an index, a commodity, bonds, or a basket of assets like an index fund. Unlike mutual funds, an ETF trades like a common stock on a stock exchange. ETFs experience price changes throughout the day as they are bought and sold. ETFs typically have higher daily liquidity and lower fees than mutual fund shares, making them an attractive alternative for individual investors.

Variable	Dataset Description
Date	This captures all the Opening, Closing, High and Low ETF that are captured in a day.
Open	**Open** is the price of the ETF at the beginning of the trading day (it need not be the closing price of the previous trading day).
High	**High** is the highest price of the ETF on that trading day.
Low	**Low** the lowest price of the ETF on that trading day.
Close	**Close** the price of the ETF at closing time.
Adj_Close	**Adjusted close** is the closing price of the ETF that adjusts the price of the ETF for corporate actions.
Volume	**Volume** is the number of shares or contracts traded in a security or an entire market during a given period. For every buyer, there is a seller, and each transaction contributes to the count of total volume. That is, when buyers and sellers agree to make a transaction at a certain price, it is considered one transaction. If only five transactions occur in a day, the volume for the day is five.

1.1.1. Source Of Data

We were divided on the source of our data and the after a lot of deliberation we came to the conclusion that Kaggle is one of the best and trusted places from where we can get our data. It is a platform for predictive modeling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and other users.

1.2. Purpose Of The Time Series Analysis Of These Data

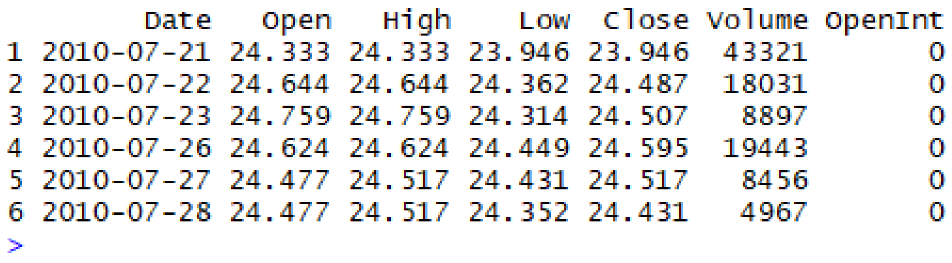
ETFs are baskets of securities designed to track the performance of an index. They are designed to provide exposure to broad-based indexes at a lower cost. We first analyzed why ETF should be the choice for an investment. We provide a brief history of this segment, key attributes of ETFs, and investments strategies and implementations with ETFs. This report presents data analysis and a series of forecasting methods with data analysis techniques to evaluate the performance of each method. The data analysis and the forecast evaluation is to determine the best forecasting model for a single ETF (WCM/BNY Mellon Focused Growth ADR ETF). The different techniques considered include single exponential smoothing, Holt's exponential smoothing, simple linear regression, multiple regression and various versions of Box-Jenkins (ARIMA) models. Based on the evaluation of a decade of past historical data, we provide a guidance for the price of our ETF (WCM/BNY Mellon Focused Growth ADR ETF) using the ARIMA model, which produced promising results (with low forecast errors of 1% across several forecast metrics), among the different techniques evaluated.

2. Statistical Analysis

2.1. Pre-Processing Of Data

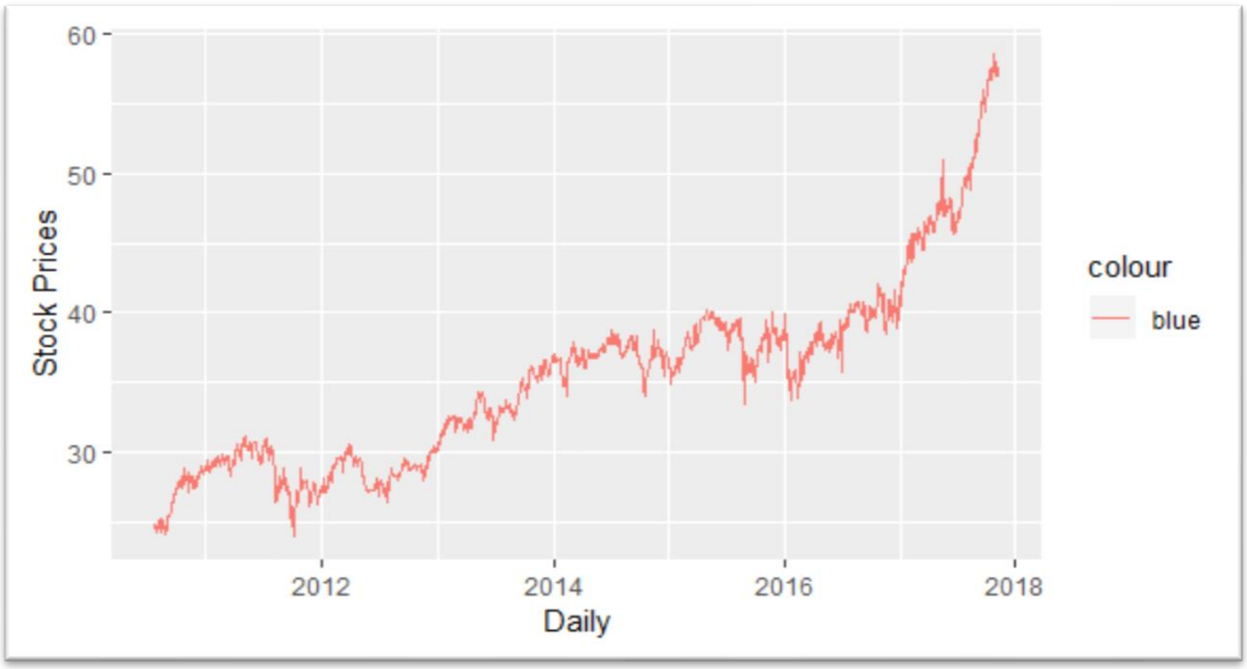
- **Reading Time Series Data**

  We read the file using read.csv.
  The data set has these variables and looks like this in general:



```
        Date   Open   High    Low  Close Volume OpenInt
1 2010-07-21 24.333 24.333 23.946 23.946  43321       0
2 2010-07-22 24.644 24.644 24.362 24.487  18031       0
3 2010-07-23 24.759 24.759 24.314 24.507   8897       0
4 2010-07-26 24.624 24.624 24.449 24.595  19443       0
5 2010-07-27 24.477 24.517 24.431 24.517   8456       0
6 2010-07-28 24.477 24.517 24.352 24.431   4967       0
>
```

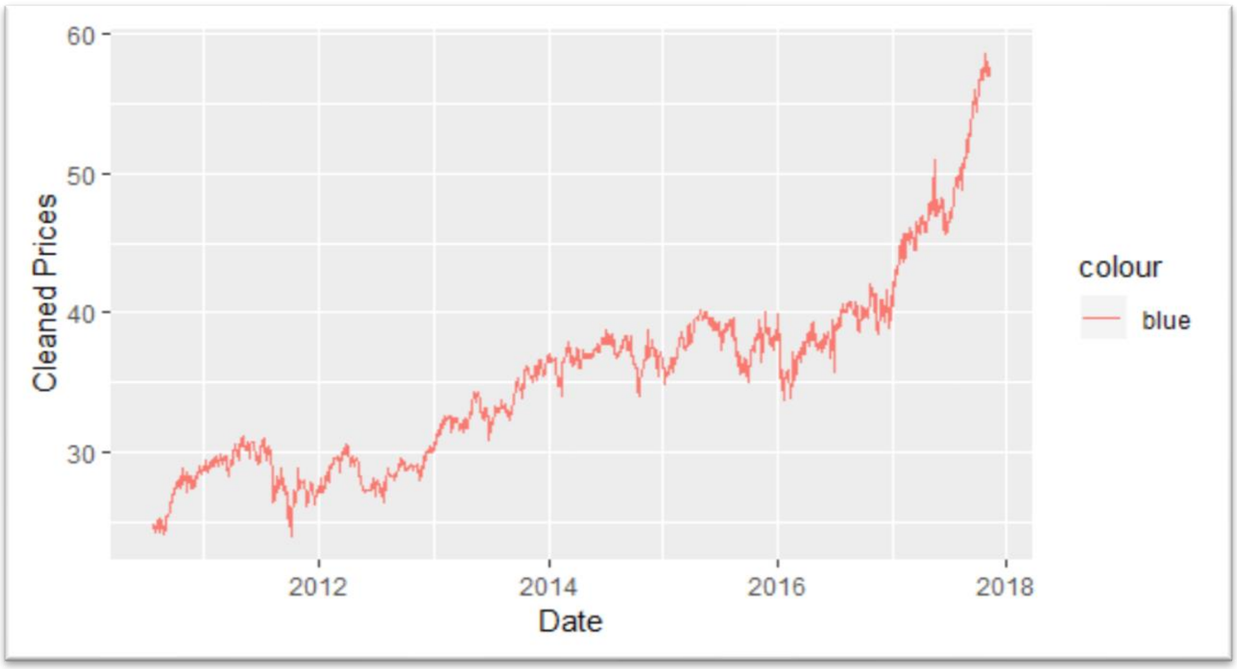
- **Plotting Time Series**
  After loading time series data into R, we plot the data using ggplot2.



We observed that our dataset has a linear increase in price along with time.

```
#Removing outliers from the data

count_ts = ts(daily[,c('Open')])

daily$clean_open = tsclean(count_ts)
```

We counted all the Open Price values and used tsclean() to identify and replace outliers and missing values in our time series data. Here is the plot of Clean Data.



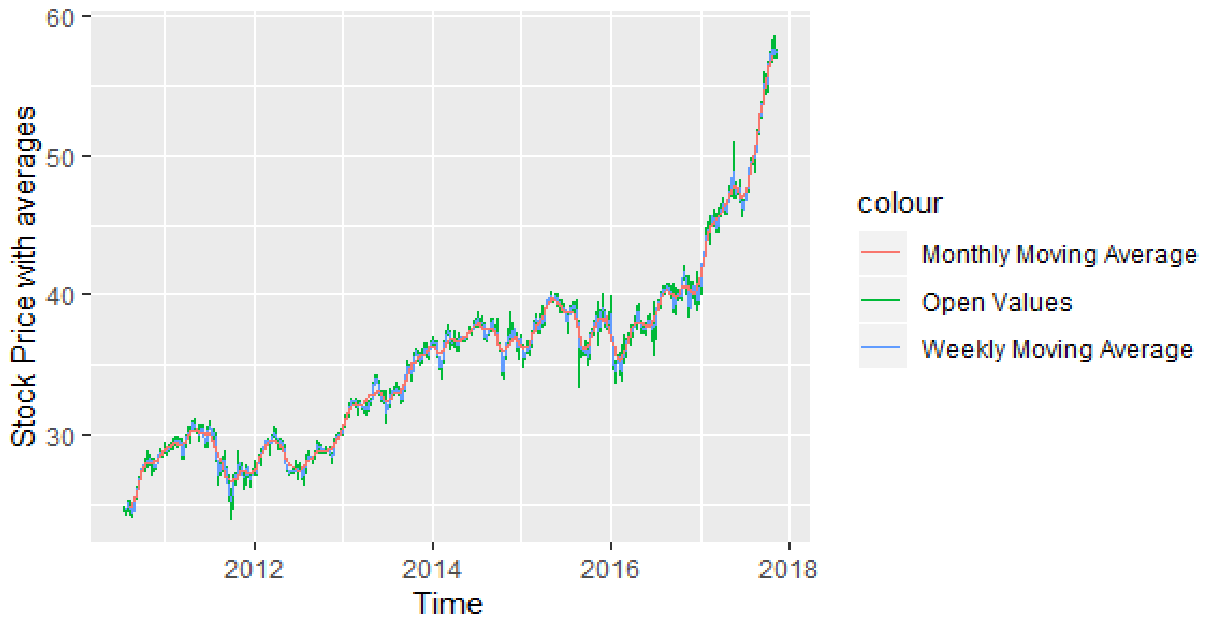
After removing the outliers, we observe that our daily data is pretty stationery. Visually we could draw a line through it i.e. the line would contain average points across several time periods, called the Moving Averages thereby smoothing the observed data into a more stable predictable series. The moving average is extremely useful for forecasting long-term trends. We calculated the Weekly and Monthly Moving Averages.

```
daily$Open_ma7 = ma(daily$Open,order=7) #Weekly Moving Average

daily$Open_ma30 = ma(daily$Open,order=30) #Monthly Moving Average
```

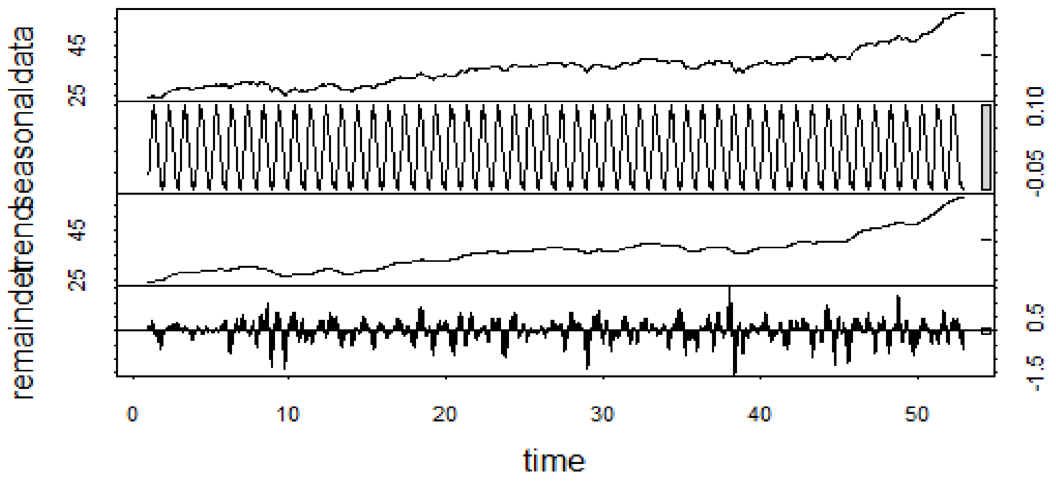
- **Adding Moving Averages**

Here, we calculate the weekly and monthly moving averages. The plot of these moving averages along with the daily open value is plotted as shown in the below chart.

Plot of Open Values along with Weekly and Monthly Moving averages:



- **Decomposing Time Series**
  Decomposing a time series means separating it into its constituent components, which are usually a trend component, and an irregular component i.e. residual or error, and if it is a seasonal time series, a seasonal component.
  We decomposed our data using **stl()** tried to predict the market for Open Stock prices.



- **Stationarity Check:**

  ARIMA model needs the series to be stationary i.e. the mean, Variance and Auto-covariance should be independent on time. Hence, we will be using Augmented Dickey-Fuller (ADF) test to formally test the stationarity.

2.2. Software Used

We used R-Studio for our analysis and forecasting modeling, and for cleaning data. Some of the characteristics of R that we thought we would use to our advantage were:

- "**ts**" is the basic class for regularly spaced time series using numeric time stamps.
- The zoo package provides infrastructure for regularly and irregularly spaced time series using arbitrary classes for the time stamps (i.e., allowing all classes from the previous section). It is designed to be as consistent as possible with "ts".
- The package **xts** is based on zoo and provides uniform handling of R's different time-based data classes.
- Various packages implement irregular time series based on "POSIXct" time stamps, intended especially for financial applications. These include "irts" from tseries, and "fts" from fts.
- The class "timeSeries" in timeSeries implements time series with "timeDate" time stamps.
- The class "tis" in tis implements time series with "ti" time stamps.
- The package tframe contains infrastructure for setting time frames in different formats.

Forecasting and Univariate Modeling

- The **forecast** package provides a class and methods for univariate time series forecasts, and provides many functions implementing different forecasting models.
- *Exponential smoothing:* HoltWinters() in stats provides some basic models with partial optimization, ets() from the forecast package provides a larger set of models and facilities with full optimization. robets provides a robust alternative to the ets() function. smooth implements some generalizations of exponential smoothing. The MAPA package combines exponential smoothing models at different levels of temporal aggregation to improve forecast accuracy.
- *Autoregressive models:* ar() in stats (with model selection) and FitAR for subset AR models.
- *ARIMA models:* arima() in stats is the basic function for ARIMA, SARIMA, ARIMAX, and subset ARIMA models. It is enhanced in the forecast package via the function arima() along with auto.arima() for automatic order selection. arima() in the **tseries** package provides different algorithms for ARMA and subset ARMA models. Other estimation methods including the innovations algorithm are provided by itsmr. FitARMA implements a fast MLE algorithm for ARMA models.

2.3. Procedures Used

We used **autoregressive integrated moving average (ARIMA)** model that is a generalization of an autoregressive moving average (ARMA) model. As ARIMA models are applied in cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity as we had to with our dataset.

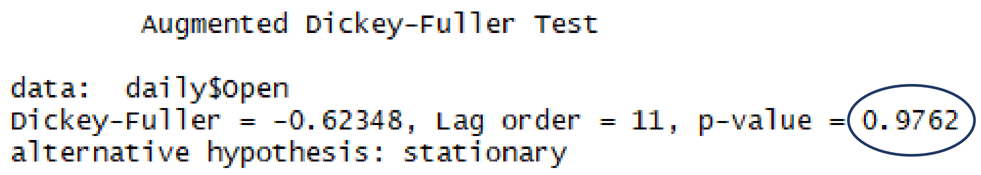
The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

Non-seasonal ARIMA models are generally denoted ARIMA(p,d,q) where parameters p, d, and q are non-negative integers, p is the order (number of time lags) of the autoregressive model, d is the degree of differencing (the number of times the data have had past values subtracted), and q is the order of the moving-average model. Seasonal ARIMA models are usually denoted ARIMA(p,d,q)(P,D,Q)$_m$, where m refers to the number of periods in each season, and the uppercase P,D,Q refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model.

When two out of the three terms are zeros, the model may be referred to be based on the non-zero parameter, dropping "AR", "I" or "MA" from the acronym describing the model. For example, ARIMA (1,0,0) is AR(1), ARIMA(0,1,0) is I(1), and ARIMA(0,0,1) is MA(1).

Test Statistics of Open Values:

adf.test(daily$Open, alternative = "stationary")



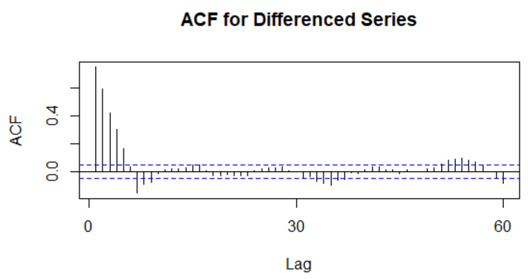
```
        Augmented Dickey-Fuller Test

data:  daily$Open
Dickey-Fuller = -0.62348, Lag order = 11, p-value = 0.9762
alternative hypothesis: stationary
```

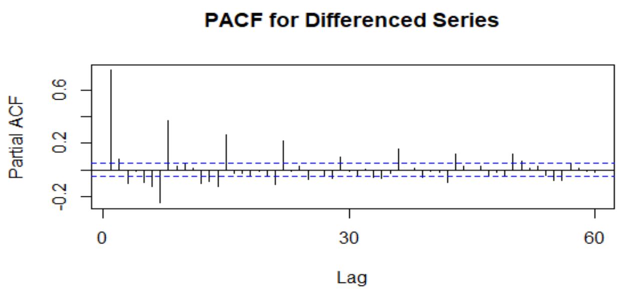
Since, the P-value is huge, we cannot reject the null hypothesis i.e. weekly moving average is not stationary. Hence, we will check for the stationarity for the difference of weekly moving average. We removed the seasonal component by using **seasadj()** of the decomposed data and performed the ADF test on this decomposed data.

Test Statistics of 'Differenced' Weekly Moving Averages:

```
        Augmented Dickey-Fuller Test

data:  count_d1
Dickey-Fuller = -9.8365, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

Now we observe that the P-value is less than the alpha values, so we reject the null hypothesis, i.e. the difference of weekly moving average is stationary. Hence, we will go ahead with the differenced data for building the ARIMA model. Below are the plots for ACF and PACF for difference of weekly moving averages which supports the ADF test results.





Here we can observe that most of the significant values are within the confidence intervals.

ARIMA Model:

ARIMA is combination of Auto Regression and Moving averages. We try to fit an ARIMA model with c (1,1,1) for which below are the statistics

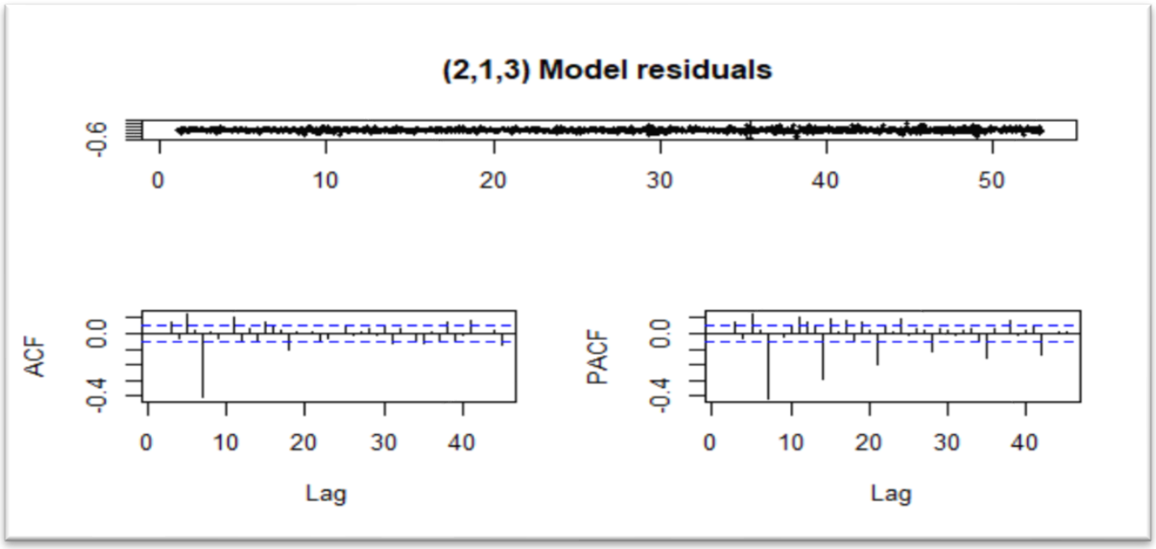
T statistics for ARIMA Model order 1,1,1:

```
> summary(try)

Call:
arima(x = deseasonal_cnt, order = c(1, 1, 1))

Coefficients:
         ar1      ma1
      0.7889  -0.0852
s.e.  0.0196   0.0295

sigma^2 estimated as 0.008279:  log likelihood = 1523.38,  aic = -3040.76
```

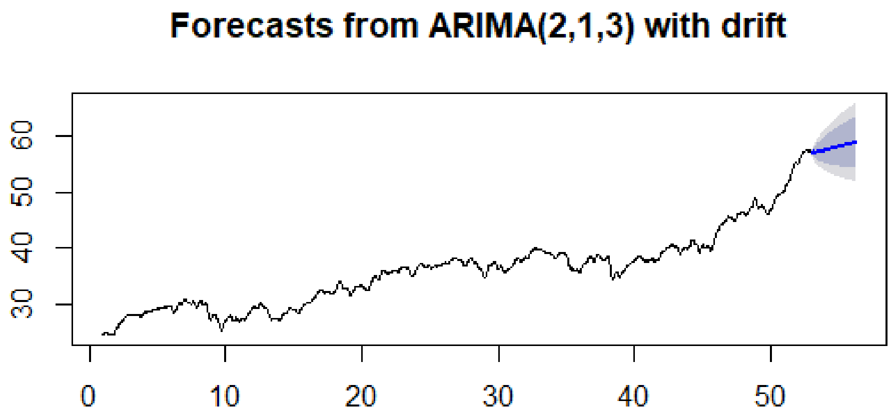
We used **auto.arima()**that gave us the best model according to sigma^2 value as shown. Hence, ARIMA(2,1,3) with drift is our best model.

```
Series: deseasonal_cnt
ARIMA(2,1,3) with drift

Coefficients:
          ar1     ar2     ma1     ma2     ma3   drift
      -0.1744  0.705  0.8936  0.0367  0.0037  0.0209
s.e.   0.0379  0.029  0.0428  0.0516  0.0372  0.0093

sigma^2 estimated as 0.008031:  log likelihood=1550.05
AIC=-3086.11   AICc=-3086.04   BIC=-3048.65
```

We went ahead and plotted the residuals plot for this.



Forecasting Values: We used the Forecast package to predict the next 100 values for this and this is shown in the graph in the blue line within confidence bounds.



Conclusions:

- If the sign of the forecasted return equals the sign of the actual returns we have assigned it a positive accuracy score.
- The accuracy percentage of the model comes to around 68%.
- We can try running the model for other possible combinations of (p,d,q) or instead use the auto.arima function which selects the best optimal parameters to run the model.
- Due to time constraints we kept our analysis limited to Open Price values. For better forecasting experience we could also repeat the entire procedure for close, low and high values, observe the trends and results and make wise investment decisions.