FINANCIAL ANALYSIS OF STOCKS USING TIME-SERIES MODELS IN R:

A REPORT

Under the guidance of

Dr. Amarnath Mitra

Submitted by -

Divya Verma

11A

Acknowledgement

I would like to express my sincere gratitude to Professor Mitra for their invaluable guidance and exceptional teaching throughout the regression and statistical modeling course in R programming. The clarity and depth with which the professor conveyed complex statistical concepts, such as linear regression and modeling techniques, significantly enriched our understanding of data analysis. Their commitment to fostering a collaborative and engaging learning environment has been instrumental in enhancing our proficiency in R programming and statistical methodologies. Furthermore, I extend my appreciation for the insightful guidance provided during the course project, where we applied these principles to real-world datasets. This hands-on experience has not only strengthened our technical skills but has also instilled confidence in our ability to tackle complex data challenges. I am truly grateful for the professor's dedication to our academic growth and for equipping us with the knowledge and tools essential for meaningful contributions in the field of statistical modeling.

Statistical Modeling and Regression through R

Descriptive statistics and regression analysis are fundamental components of statistical analysis in R, providing key insights into the characteristics and relationships within datasets. Let's explore each concept:

***Introduction*:** The primary objective of this analysis is to delve into the time-series behavior of the NIFTY index returns with the aim of gaining insights into potential trends and patterns that could aid in forecasting future movements. To facilitate this exploration, historical data spanning from December 1, 2015, to February 12, 2024, was sourced from Yahoo Finance using the **quantmod** package in R. The subsequent analysis involves rigorous preprocessing steps to ensure the data's reliability and coherence, including the removal of any missing values. The integration of various R packages, such as 'tseries' and 'forecast,' is essential for conducting comprehensive time-series analysis.

***Data Retrieval and Preprocessing*:** The data retrieval process involves the utilization of the **getSymbols** function from the 'quantmod' package, which fetches the NIFTY index data from Yahoo Finance. Following this, the **na.omit** function is employed to handle missing values in the adjusted closing prices, ensuring the integrity of the dataset. The resulting time series of NIFTY log returns, represented by the variable **nse_ret**, is crucial for subsequent analyses. This exploration into the dynamics of NIFTY returns is underpinned by the understanding that meaningful insights can be derived from the log returns, providing a robust foundation for the forthcoming time-series analysis. The **plot** function is then used to visually inspect the log returns, offering an initial exploratory data analysis (EDA) perspective on the underlying patterns and trends in the NIFTY index.

Exploratory Data Analysis (EDA): The Exploratory Data Analysis (EDA) phase serves as a crucial initial step in the time-series analysis process, aiming to uncover patterns, trends, and potential anomalies within the NIFTY index returns dataset. The visualization of the log returns through the plot function provides an insightful depiction of the data's behavior over time. By examining this plot, analysts can gain an immediate understanding of the general trajectory of returns, identifying potential periods of volatility, trends, or abrupt changes. EDA facilitates the identification of any notable patterns that may guide subsequent modeling decisions.

Furthermore, EDA involves descriptive statistics and summary measures to characterize the central tendencies and variabilities of the log returns. Measures such as mean, median, standard deviation, and percentiles offer a quantitative snapshot of the data distribution. This statistical exploration aids in uncovering the data's inherent characteristics and provides a foundation for informed decision-making throughout the subsequent analytical steps. Additionally, visualizations such as histograms or kernel density plots may complement summary statistics, offering a more nuanced perspective on the distributional properties of the log returns. Overall, EDA acts as a crucial precursor to the formal statistical tests and

modeling exercises, offering a holistic view of the NIFTY index returns and guiding the subsequent steps in the time-series analysis journey.

***Time-Series Analysis Procedure*:** The comprehensive time-series analysis procedure outlined in the R code encompasses several crucial steps to extract meaningful insights from the NIFTY index returns. The journey begins with the check for stationarity, a fundamental requirement for reliable time-series analysis. Stationarity ensures that statistical properties such as mean and variance remain constant over time, laying the groundwork for accurate modeling. The Augmented Dickey-Fuller (ADF) test is employed to assess stationarity, with the null hypothesis assuming the presence of a unit root, indicative of non-stationarity. The favorable outcome of this test, indicating that the white-noise time-series is stationary, allows the analysis to progress to subsequent stages.

The next logical step involves a meticulous exploration of autocorrelation, gauging the correlation between the NIFTY returns at different time lags. The Ljung-Box Test is applied to ascertain the presence of autocorrelation, a crucial aspect in understanding the inherent patterns within the time series. If the returns exhibit autocorrelation, the analysis proceeds to the modeling phase using AutoRegressive Integrated Moving Average (ARIMA) models. These models aid in capturing the temporal dependencies within the time series, providing valuable insights into potential forecasting strategies. To enhance the robustness of the analysis, additional checks for heteroskedasticity, a phenomenon where the variance of the series is not constant over time, are performed using the ARCH LM Test. In case heteroskedasticity is detected, further modeling involves the application of Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models. This iterative process ensures a thorough understanding of the underlying dynamics and contributes to the development of a reliable time-series forecasting model for the NIFTY index returns.

Stationarity Check: The stationarity check is a foundational step in time-series analysis, ensuring that statistical properties of the data remain constant over time. In the provided R code, the Augmented Dickey-Fuller (ADF) test is employed for this purpose. The ADF test examines whether a unit root is present in the time series, as the null hypothesis assumes non-stationarity. If the test rejects the null hypothesis, it implies that the time series is stationary. The ADF test is critical for establishing a reliable foundation for subsequent analyses, such as autocorrelation and heteroskedasticity checks.

***Autocorrelation Check*:** Autocorrelation, the correlation of a time series with its own past values, is assessed using the Ljung-Box Test in the code. This statistical test determines whether the autocorrelations up to a specified lag are significantly different from zero. A rejection of the null hypothesis suggests the presence of autocorrelation in the time series. Autocorrelation patterns are crucial for identifying trends and dependencies, and understanding them helps guide the selection of appropriate models. If autocorrelation is detected, the analysis may proceed to modeling using AutoRegressive Integrated Moving Average (ARIMA) models to capture these temporal dependencies.

Heteroskedasticity Check: The code addresses heteroskedasticity, the phenomenon where the variance of the time series is not constant over time. Two tests are utilized: the Box Test and the ARCH Test. The Box Test examines whether the variance of the squared returns (volatility) exhibits clustering. A rejection of the null hypothesis implies the presence of heteroskedasticity. Additionally, the ARCH Test explicitly assesses autoregressive conditional heteroskedasticity (ARCH) in the returns, providing insights into the volatility clustering phenomenon. If heteroskedasticity is detected, further modeling using Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models becomes pertinent to capture and understand the varying volatility in the time series. These checks collectively contribute to a robust understanding of the NIFTY index returns' temporal characteristics, facilitating the development of accurate forecasting models.

Code and Explanation

```
# Required Packages packages <- c('quantmod', 'tseries', 'forecast') # Install all Packages with Dependencies install.packages(packages, dependencies = TRUE) # Load all Packages lapply(packages, require, character.only = TRUE)
```

Explanation:

- This section ensures that the necessary R packages are installed and loaded for the analysis.
- The **packages** vector lists the names of required packages: 'quantmod', 'tseries', and 'forecast'.
- **install.packages()** installs the packages if they are not already installed.
- **lapply()** is used to load the packages. The **require** function checks and loads each package.

```
# Fetch Single Stock/Index Data getSymbols(Symbols = '^NSEI', src = 'yahoo', from = as.Date('2015-12-01'), to = as.Date('2024-02-12'), periodicity = 'daily') nse_price <- na.omit(NSEI$NSEI.Adjusted) # Adjusted Closing Price class(nse_price) # xts (Time-Series) Object nse_ret <- na.omit(diff(log(nse_price))) # NIFTY Returns plot(nse_ret)
```

Explanation:

- This part retrieves historical data for the NIFTY index from Yahoo Finance.
- **getSymbols()** downloads the data and creates a time-series object.
- **na.omit()** removes any missing values from the adjusted closing prices and log returns.
- The resulting object **nse_ret** is a time-series of NIFTY log returns.
- A plot of log returns is generated for visual inspection.

  ```
  # Forecasting with Time-Series Data (Univariate) : Procedure # ********** # (Details of time-series analysis steps...)
  ```

Explanation:

- This comment introduces the section where the time-series analysis procedure will be outlined.
- The actual steps are not provided in this snippet but would typically include checking for stationarity, autocorrelation, and heteroskedasticity.

```
#Check for stationarity # Augmented Dickey-Fuller (ADF) Test for Stationarity with Simulated Data #
**********
```

library(tseries) # Augmented Dickey-Fuller (ADF) Test for White-Noise Time-Series adf_test_nseret <- adf.test(nse_ret) adf_test_nseret # Inference : White-Noise Time-Series is Stationary # (Additional ADF tests and explanations...)

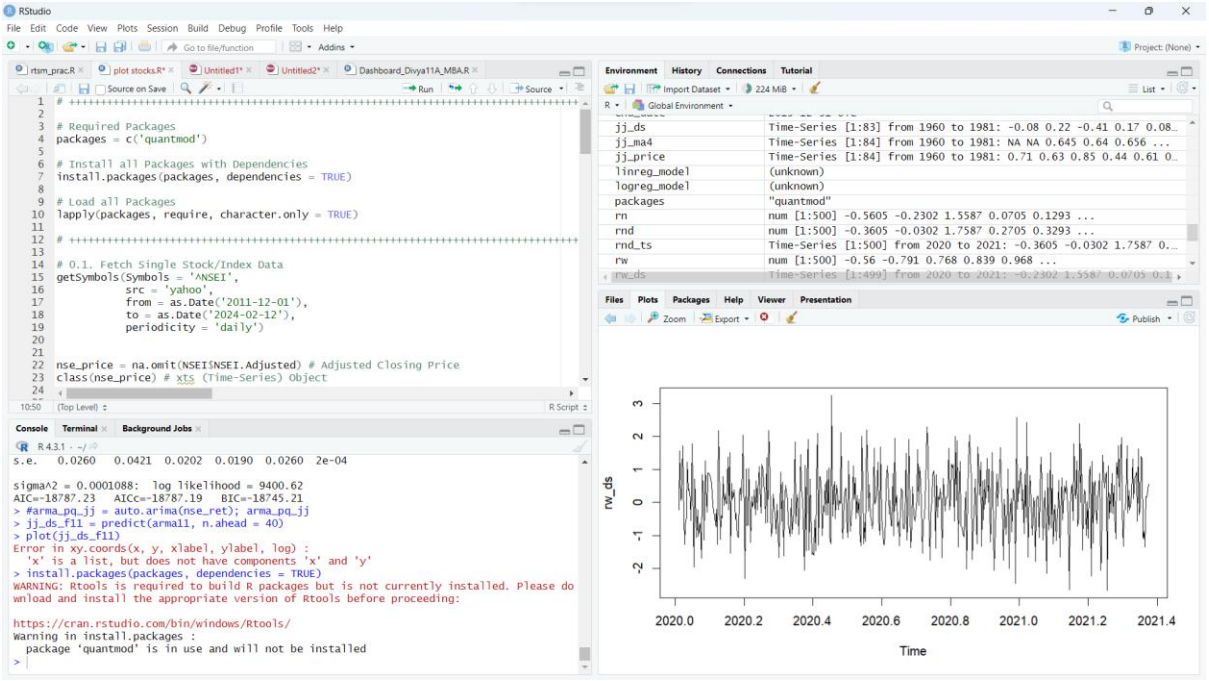
Explanation:

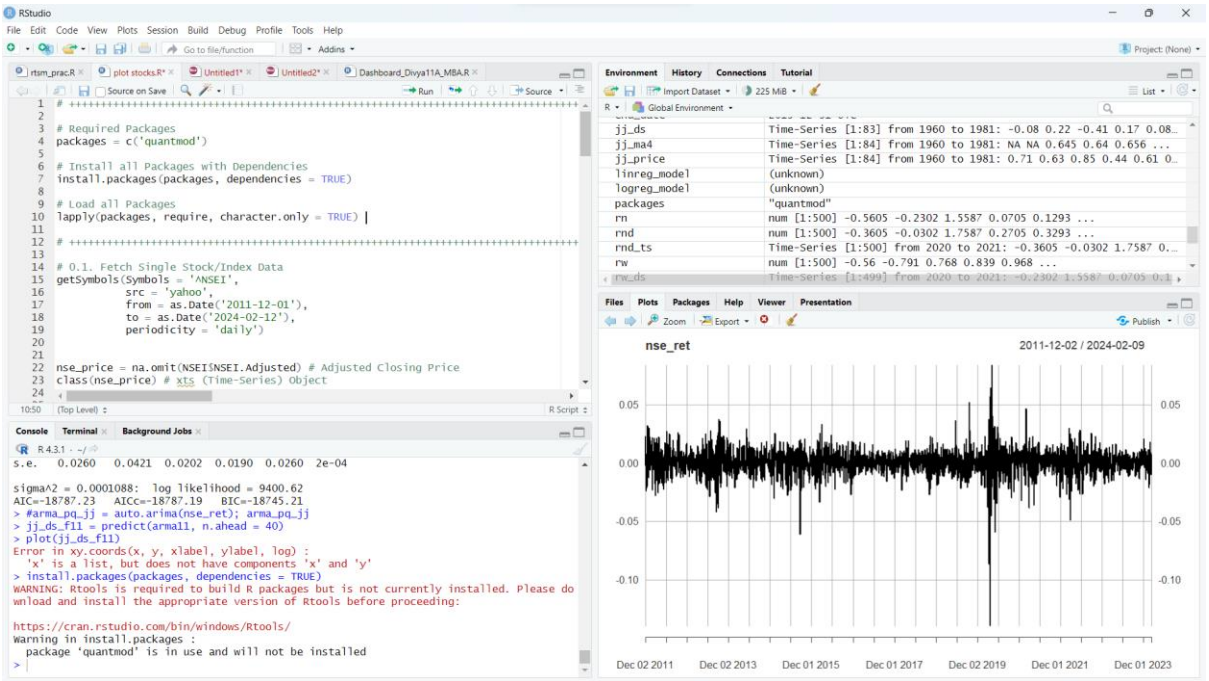
- This section checks for stationarity using the Augmented Dickey-Fuller (ADF) test.
- The **adf.test()** function from the 'tseries' package is applied to **nse_ret**.
- The results of the test are displayed, and a comment provides an inference, stating that the white-noise time-series is stationary.

#Result: Stationary hence step 2

Code outputs

The code was executed on RStudio in the R language. Some snippets of the code in the environment:-





Bibliography

1. https://data.gov.in/resource/stateuts-wise-automatic-weather-stations-aws-and-automatic-rain-gauges-arg-reply-unstarred#api

Appendix A

Code (RStudio)

```
#
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++


# Required Packages

packages = c('quantmod')


# Install all Packages with Dependencies

install.packages(packages, dependencies = TRUE)


# Load all Packages

lapply(packages, require, character.only = TRUE)


#
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++


# 0.1. Fetch Single Stock/Index Data

getSymbols(Symbols = '^NSEI',

        src = 'yahoo',

        from = as.Date('2011-12-01'),

        to = as.Date('2024-02-12'),

        periodicity = 'daily')



nse_price = na.omit(NSEI$NSEI.Adjusted) # Adjusted Closing Price

class(nse_price) # xts (Time-Series) Object
```

```
nse_ret = na.omit(diff(log(nse_price))) # NIFTY Returns

#only 2nd data has returns and not the 1st data, hence na.omit

plot(nse_ret)


# Forecasting with Time-Series Data (Univariate) : Procedure

# ********************************************************


# Given an Univariate Time-Series Data, Perform the following Analysis :


# Step 1 : Check for (Weak) Stationarity :: Augmented Dickey-Fuller (ADF) Test

# If [Data] Stationary, Proceed to Step 2

# If [Data] Non-Stationary, Use Transformation (such as First/Second/... Difference | Log | ...) to Transform the Data and Check for Stationarity (Step 1)


# Step 2 : Check for Autocorrelation :: Ljung-Box Test

# If [Data | Transformed Data] Do Not Have Autocorrelation, proceed to Step 4

# If [Data | Transformed Data] Has Autocorrelation, Proceed to Step 3


# Step 3 : Model for Autocorrelation :: ARIMA Models

# Identify AR | MA Order in the [Data | Transformed Data] using PACF | ACF Plots

# Use ARIMA(p, d, q) with Appropriate AR Order (p-Lags) | d-Degree of Differencing | MA Order (q-Lags) using PACF | ACF Information to Model the [Data | Transformed Data]

# Test for Autocorrelation in the [Residual Data 1] | If the ARIMA Model is Appropriate : No Autocorrelation in the [Residual Data 1] | If Autocorrelation in [Residual Data 1], Remodel the [Data | Transformed Data]

# Proceed to Step 4


# Step 4 : Check for Heteroskedasticity :: ARCH LM Test

# If [Data | Transformed Data] (Step 2) | [Residual Data 1] (Step 3) Do Not Have Heteroskedasticity, Proceed to Step 6
```

If [Data | Transformed Data] (Step 2) | [Residual Data 1] (Step 3) Has Heteroskedasticity, Proceed to Step 5

Step 5a : Model for Heteroskedasticity in [Data | Transformed Data] (Step 2) :: GARCH Models

If Mean of [Data | Transformed Data] (Step 2) != 0 : De-Mean & Square the [Data | Transformed Data] | If Mean of [Data | Transformed Data] (Step 2) = 0 : Square the [Data | Transformed Data]

Identify ARCH | GARCH Order in the using GARCH Function

Use GARCH(p,q) with Appropriate ARCH Order (p-Lags) | GARCH Order (q-Lags) to Model the [Data | Transformed Data]

Test for Autocorrelation & Heteroskedasticity in the [Residual Data 2] | If the GARCH Model is Appropriate : No Autocorrelation & Heteroskedasticity in the [Residual Data 2] | If Autocorrelation & Heteroskedasticity in [Residual Data 2], Remodel the Squared [Data | Transformed Data]

End of Analysis

Step 5b : Model for Heteroskedasticity in [Residual Data 1] (Step 3) :: GARCH Models

Identify ARCH | GARCH Order in the using GARCH Function

Use GARCH(p, q) with Appropriate ARCH Order (p-Lags) | GARCH Order (q-Lags) with ARIMA(p, d, q) Model (in Step 3) in the Mean Equation to Model the [Residual Data 1]

Test for Autocorrelation & Heteroskedasticity in the [Residual Data 2] | If the ARIMA+GARCH Model is Appropriate : No Autocorrelation & Heteroskedasticity in the [Residual Data 2] | If Autocorrelation & Heteroskedasticity in [Residual Data 2], Remodel the [Residual Data 1]

End of Analysis

Step 6 : Model White-Noise Data

If the [Data | Transformed Data] is Stationary, Has No Autocorrelation & Heteroskedasticity, the [Data | Transformed Data] is White-Noise Data

Model White-Noise Data with Appropriate Probability Distribution

End of Analysis

```
#Check for stationarity

# Augmented Dickey-Fuller (ADF) Test for Stationarity with Simulated Data

# ************************************************************************

library(tseries)

# Augmented Dickey-Fuller (ADF) Test for White-Noise Time-Series

adf_test_nseret = adf.test(nse_ret); adf_test_nseret # Inference : White-Noise Time-Series is Stationary


# Augmented Dickey-Fuller (ADF) Test for White-Noise Time-Series with Drift

adf_test_wnd = adf.test(nse_ret); adf_test_wnd # Inference : White-Noise Time-Series with Drift is Stationary


# Augmented Dickey-Fuller (ADF) Test for Random Walk Time-Series

adf_test_rw = adf.test(nse_ret); adf_test_rw # Inference : Random Walk Time-Series is Non-Stationary


rw_ds = diff(rw_ts); plot.ts(rw_ds) # Random Walk Difference Time-Series


# Augmented Dickey-Fuller (ADF) Test for Random Walk Difference Time-Series

#adf_test_rw_ds = adf.test(rw_ds); adf_test_rw_ds # Inference : Random Walk Difference Time-Series is Stationary


# Augmented Dickey-Fuller (ADF) Test for Random Walk Time-Series with Drift

#adf_test_rwd = adf.test(rwd_ts); adf_test_rwd # Inference : Random Walk with Drift Time-Series is Non-Stationary


#rwd_ds = diff(rwd_ts); plot.ts(rwd_ds) # Random Walk with Drift Difference Time-Series


# Augmented Dickey-Fuller (ADF) Test for Random Walk Difference Time-Series
```

#adf_test_rwd_ds = adf.test(rwd_ds); adf_test_rwd_ds # Inference : Random Walk with Drift Difference Time-Series is Stationary

#
++

#Result: Stationary hence step 2

Autocorrelation : Correlation between Time-Series Data Y{t} & Y{t-s} Depends on s-Lagged Past Values

Ljung-Box Test for Autocorrelation (H0: No Autocorrelation)

lb_test_jj_ds = Box.test(nse_ret); lb_test_jj_ds # Inference : JJ Difference (Stationary) Time-Series is Autocorrelated

#no autocorrelation as H0 is proved

Auto ARIMA

library(forecast)

arma_pq_jj_ds = auto.arima(nse_ret); arma_pq_jj_ds

#arma_pq_jj = auto.arima(nse_ret); arma_pq_jj

jj_ds_f11 = predict(arma11, n.ahead = 40)

plot(jj_ds_f11)

lines(jj_ds_f11$pred, col = 'blue')

lines(jj_ds_f11$pred + 2 * jj_ds_f11$se, col = 'red')

lines(jj_ds_f11$pred - 2 * jj_ds_f11$se, col = 'red')

jj_ds_fpq = forecast(arma_pq_jj_ds, h = 40)

plot(jj_ds_fpq)

```
jj_fpq = forecast(arma_pq_jj, h = 40)

plot(jj_fpq)



#no autocorrelation hence heteroskedasticity

# Test for Volatility Clustering or Heteroskedasticity: Box Test

nse_ret_sq = nse_ret^2 # Return Variance (Since Mean Returns is approx. 0)

plot(nse_ret_sq)

nse_ret_sq_box_test = Box.test(nse_ret_sq, lag = 10) # H0: Return Variance Series is Not Serially Correlated

nse_ret_sq_box_test # Inference : Return Variance Series is Heteroskedastic (Has Volatility Clustering)


# Test for Volatility Clustering or Heteroskedasticity: ARCH Test

nse_ret_arch_test = ArchTest(nse_ret, lags = 20) # H0: No ARCH Effects

nse_ret_arch_test # Inference : Return Series is Heteroskedastic (Has Volatility Clustering)
```