Comprehensive Stock Market Analysis: NVIDIA

1. Load & Clean Data

```
stock_df <- read_excel("C:/Users/neha n/Downloads/NVidia_stock_history.xls")
stock_df$Date <- as.Date(stock_df$Date)

Q1 <- quantile(stock_df$Close, 0.25)
Q3 <- quantile(stock_df$Close, 0.75)
IQR_val <- IQR(stock_df$Close)
stock_df <- stock_df %>% filter(Close > (Q1 - 1.5 * IQR_val) & Close < (Q3 +
1.5 * IQR_val))

cat("Missing values\n")
```

```
## Missing values
```

```
print(colSums(is.na(stock_df)))
```

```
##          Date          Open          High           Low         Close
Volume
##             0             0             0             0             0
0
##     Dividends Stock Splits
##             0             0
```

```
head(stock_df)
```

```
## # A tibble: 6 × 8
##   Date         Open   High    Low  Close     Volume Dividends `Stock
Splits`
##   <date>      <dbl>  <dbl>  <dbl>  <dbl>      <dbl>     <dbl>
<dbl>
## 1 1999-01-22 0.0401 0.0448 0.0356 0.0376 2714688000         0
0
## 2 1999-01-25 0.0406 0.0420 0.0376 0.0416  510480000         0
0
## 3 1999-01-26 0.0420 0.0429 0.0377 0.0383  343200000         0
0
## 4 1999-01-27 0.0385 0.0394 0.0363 0.0382  244368000         0
0
## 5 1999-01-28 0.0382 0.0385 0.0379 0.0381  227520000         0
0
## 6 1999-01-29 0.0381 0.0382 0.0363 0.0363  244032000         0
0
```

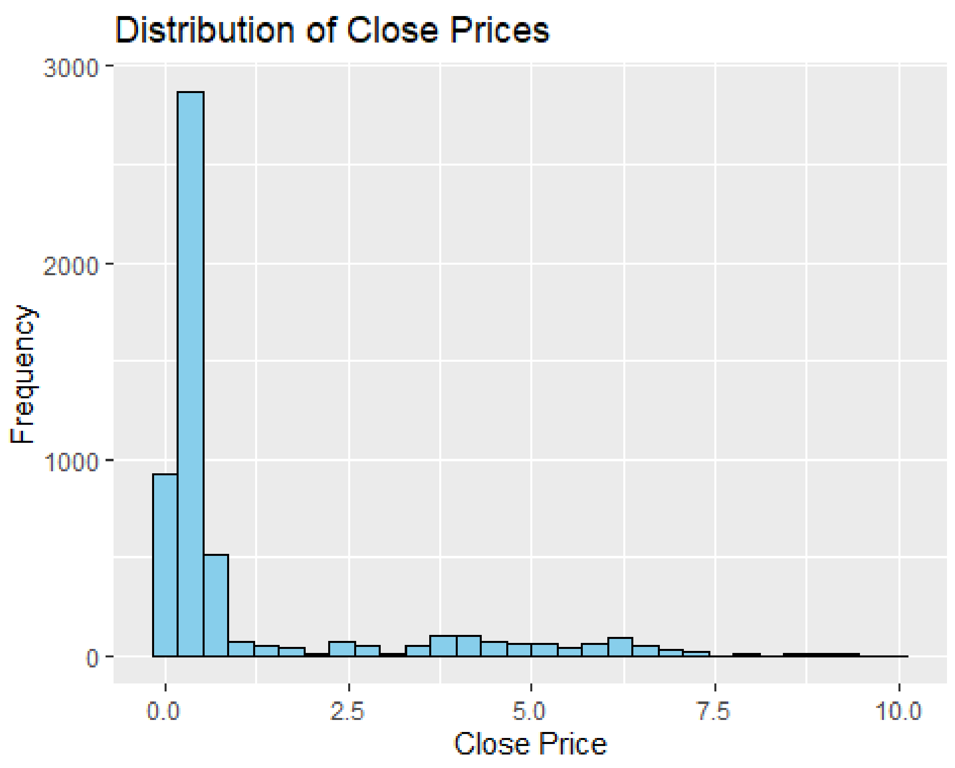
2. Data Profiling and Exploratory Analysis

2.1 Basic Summary Statistics

```
summary(stock_df)

##       Date                 Open              High               Low         
##  Min.   :1999-01-22   Min.   : 0.03201   Min.   : 0.03261   Min.   
:0.03057  
##  1st Qu.:2004-06-06   1st Qu.: 0.21778   1st Qu.: 0.22545   1st 
Qu.:0.21096  
##  Median :2009-10-14   Median : 0.34709   Median : 0.35261   Median 
:0.34055  
##  Mean   :2009-10-15   Mean   : 1.12600   Mean   : 1.14511   Mean   
:1.10611  
##  3rd Qu.:2015-02-26   3rd Qu.: 0.59123   3rd Qu.: 0.60564   3rd 
Qu.:0.58061  
##  Max.   :2020-07-13   Max.   :10.56353   Max.   :10.76019   Max.   
:9.99522  
##      Close              Volume             Dividends            Stock Splits    
##  Min.   : 0.03129   Min.   :1.968e+07   Min.   :0.000e+00   Min.   
:0.000000  
##  1st Qu.: 0.21808   1st Qu.:3.507e+08   1st Qu.:0.000e+00   1st 
Qu.:0.000000  
##  Median : 0.34719   Median :5.303e+08   Median :0.000e+00   Median 
:0.000000  
##  Mean   : 1.12620   Mean   :6.348e+08   Mean   :1.734e-05   Mean   
:0.001389  
##  3rd Qu.: 0.59731   3rd Qu.:7.817e+08   3rd Qu.:0.000e+00   3rd 
Qu.:0.000000  
##  Max.   :10.02239   Max.   :9.231e+09   Max.   :4.000e-03   Max.   
:2.000000

stock_df %>%
  summarise(
    Mean_Close = mean(Close, na.rm = TRUE),
    SD_Close = sd(Close, na.rm = TRUE),
    Min_Close = min(Close, na.rm = TRUE),
    Max_Close = max(Close, na.rm = TRUE)
  )

## # A tibble: 1 × 4
##   Mean_Close SD_Close Min_Close Max_Close
##        <dbl>    <dbl>     <dbl>     <dbl>
## 1       1.13     1.84    0.0313      10.0
```

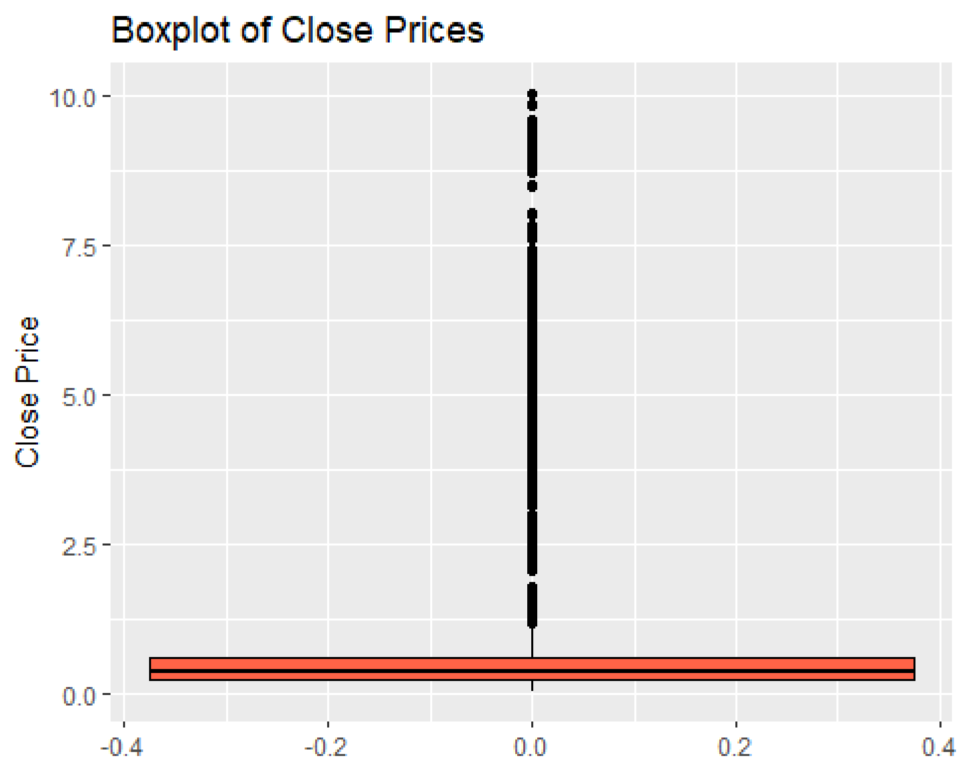
2.2 Histogram of Close Prices

```
ggplot(stock_df, aes(x = Close)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Close Prices", x = "Close Price", y = 
"Frequency")
```



2.3 Boxplot for Outlier Detection

```r
ggplot(stock_df, aes(y = Close)) +
  geom_boxplot(fill = "tomato", color = "black") +
  labs(title = "Boxplot of Close Prices", y = "Close Price")
```

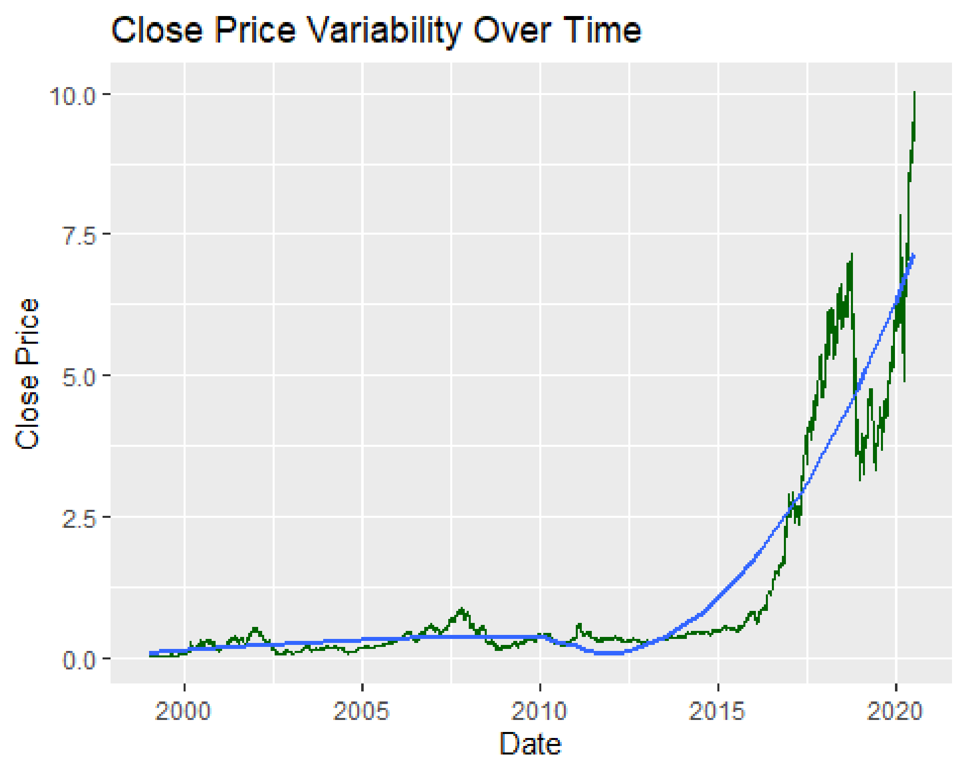


2.4 Frequency Table by Quartile

```
stock_df <- stock_df %>%
  mutate(Price_Category = ntile(Close, 4))

table(stock_df$Price_Category)

##
##    1    2    3    4
## 1350 1350 1350 1350
```

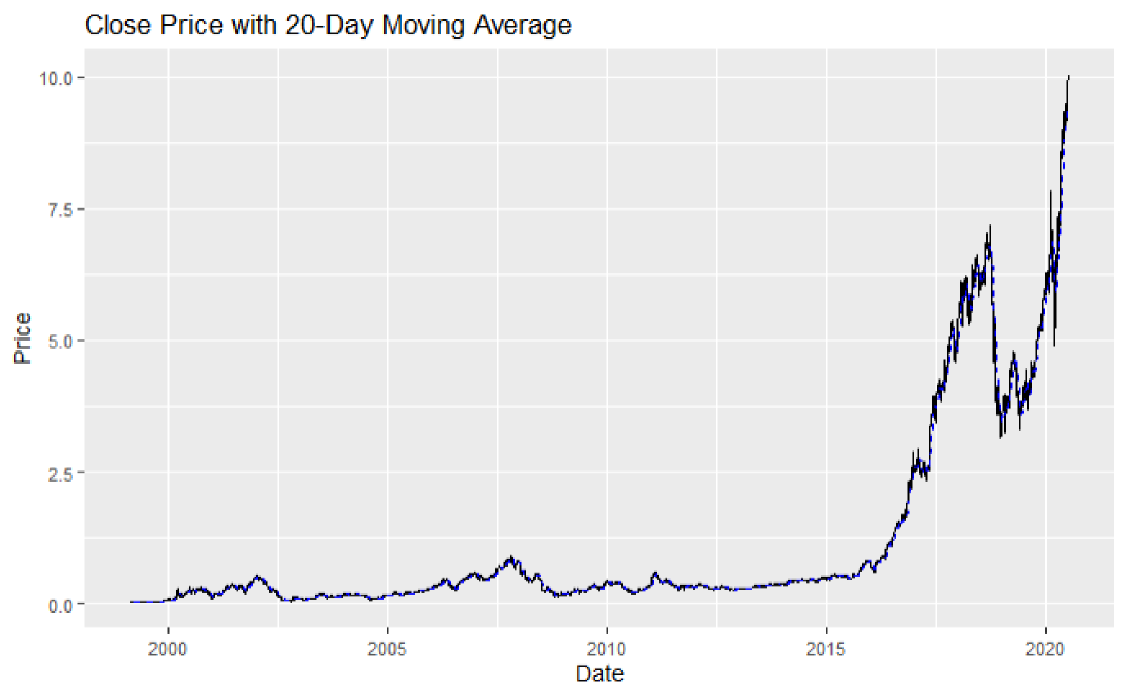
2.5 Standard Deviation and Close Price Variability Plot

```
cat("Standard Deviation of Close:", sd(stock_df$Close, na.rm = TRUE), "\n")

## Standard Deviation of Close: 1.842267

ggplot(stock_df, aes(x = Date, y = Close)) +
  geom_line(color = "darkgreen") +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Close Price Variability Over Time", y = "Close Price")

## `geom_smooth()` using formula = 'y ~ x'
```



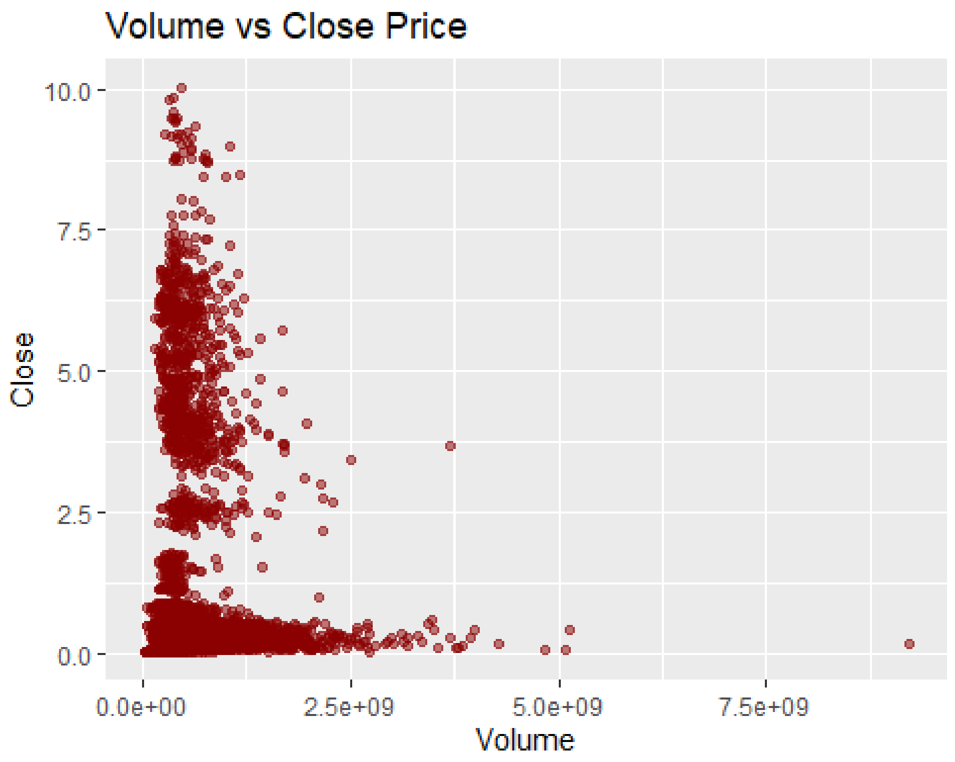
2.6 Moving Average Visualization (MA20)

```
ggplot(stock_df, aes(x = Date)) + geom_line(aes(y = Close), color = “black”) +
geom_line(aes(y = MA20), color = “blue”, linetype = “dashed”) + labs(title = “Close Price
with 20-Day Moving Average”, y = “Price”)
```



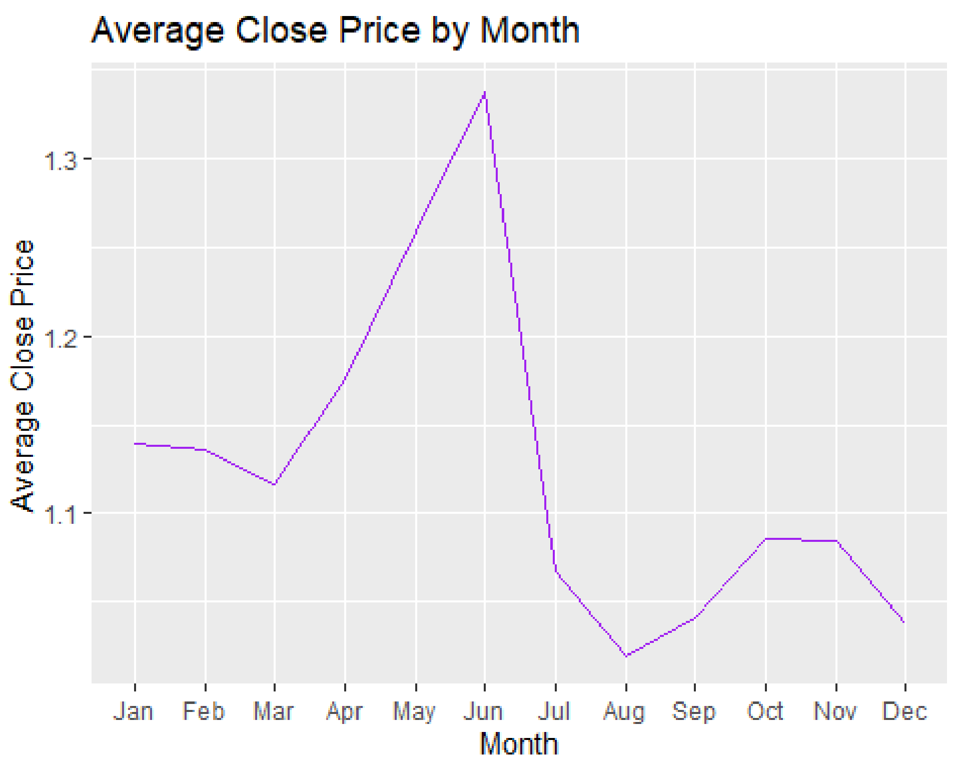
2.7 Volume vs Close Scatter Plot

```
ggplot(stock_df, aes(x = Volume, y = Close)) +
  geom_point(alpha = 0.5, color = "darkred") +
  labs(title = "Volume vs Close Price", x = "Volume", y = "Close")
```



2.8 Seasonality Analysis by Month

```
stock_df$Month <- lubridate::month(stock_df$Date, label = TRUE)

ggplot(stock_df, aes(x = Month, y = Close)) +
  stat_summary(fun = mean, geom = "line", aes(group = 1), color = "purple") +
  labs(title = "Average Close Price by Month", y = "Average Close Price")
```



2.9 Outlier Count Table

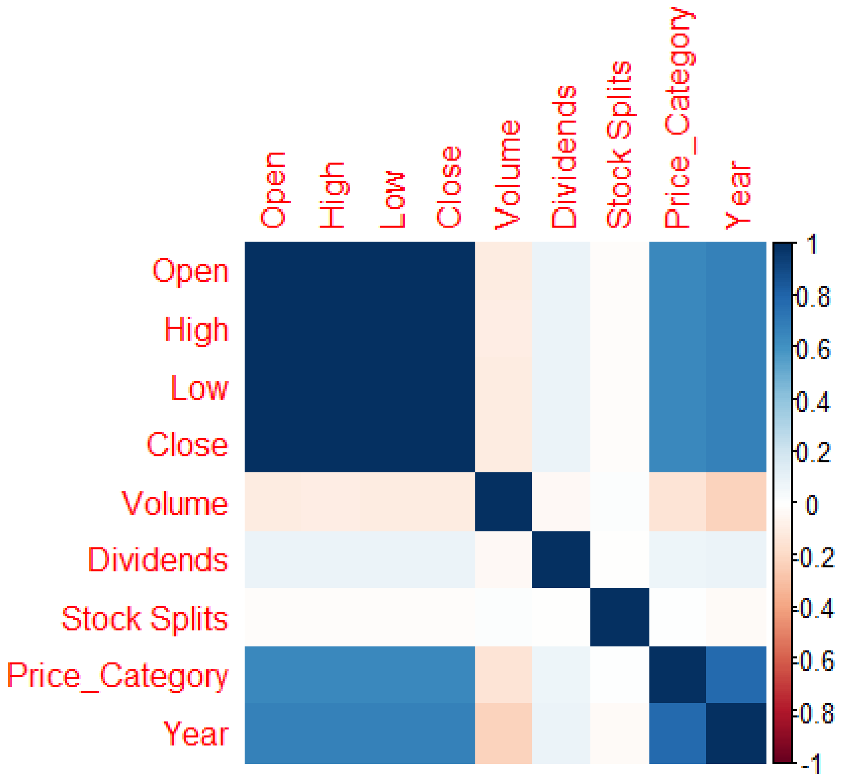
```
Q1 <- quantile(stock_df$Close, 0.25)
Q3 <- quantile(stock_df$Close, 0.75)
IQR_val <- Q3 - Q1

outliers <- stock_df %>%
  filter(Close < (Q1 - 1.5 * IQR_val) | Close > (Q3 + 1.5 * IQR_val))

n_outliers <- nrow(outliers)
cat("Total outliers in Close price:", n_outliers, "\n")

## Total outliers in Close price: 1012
```

2.10 ANOVA: Is There a Difference Across Years?

```
stock_df$Year <- lubridate::year(stock_df$Date)

anova_model <- aov(Close ~ as.factor(Year), data = stock_df)
summary(anova_model)

##                   Df Sum Sq Mean Sq F value Pr(>F)    
## as.factor(Year)   21  17482   832.5    5318 <2e-16 ***
## Residuals       5378    842     0.2                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.11 Correlation Heatmap

```
library(corrplot)

numeric_cols <- stock_df %>% select(where(is.numeric))
corrplot(cor(numeric_cols, use = "complete.obs"), method = "color")
```

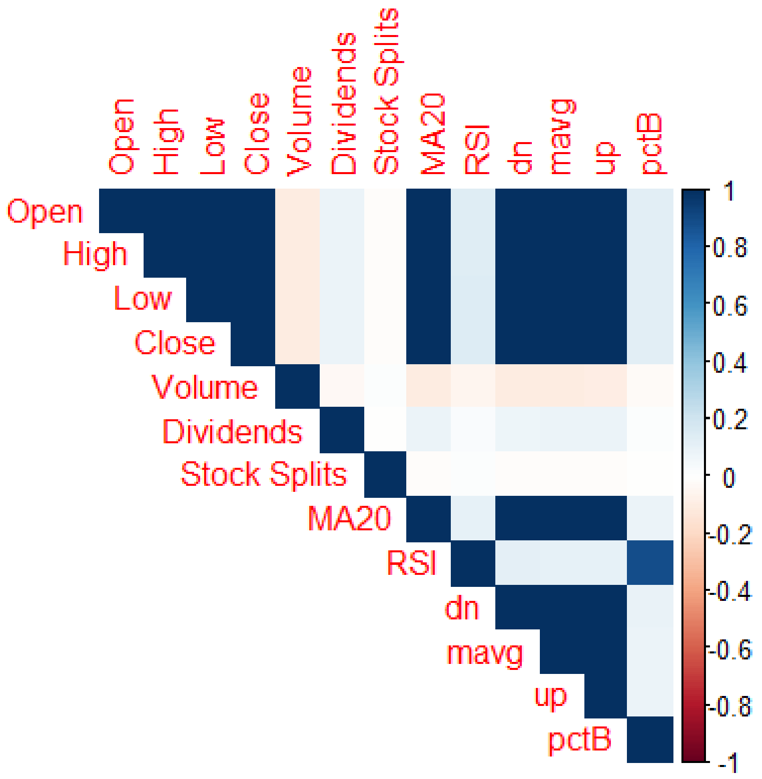


2.12 Feature Engineering

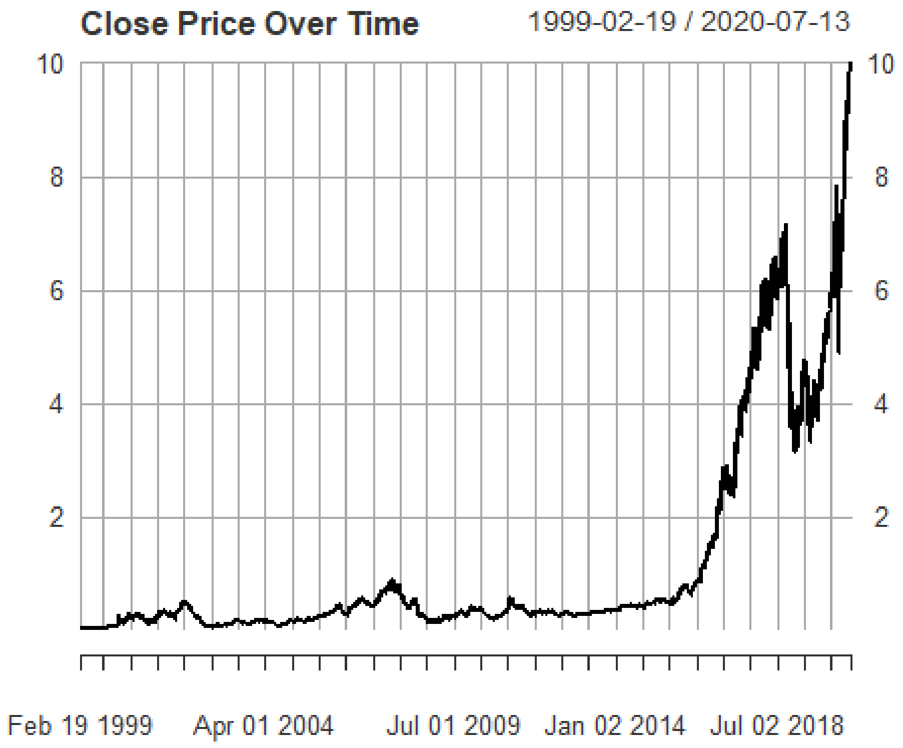
```
stock_df$MA20 <- SMA(stock_df$Close, n = 20)
stock_df$RSI <- RSI(stock_df$Close, n = 14)
bb <- BBands(stock_df$Close, n = 20)
stock_df <- bind_cols(stock_df, bb)
```

3. EDA & Trend Analysis

```
corrplot(cor(stock_df %>% select(where(is.numeric))), method = "color", type = "upper")
```



```
ts_close <- xts(stock_df$Close, order.by = stock_df$Date)
plot(ts_close, main = "Close Price Over Time")
```



4. Normalize & Split

```
df_model <- stock_df %>% select(-Date) %>% clean_names()
pre_proc <- preProcess(df_model, method = c("center", "scale"))
df_model_scaled <- predict(pre_proc, df_model)

set.seed(123)
trainIndex <- createDataPartition(df_model_scaled$close, p = 0.8, list = FALSE)
train <- df_model_scaled[trainIndex, ]
test <- df_model_scaled[-trainIndex, ]
```

5. Classical Models (Linear, Ridge, RF, Tree, SVM)

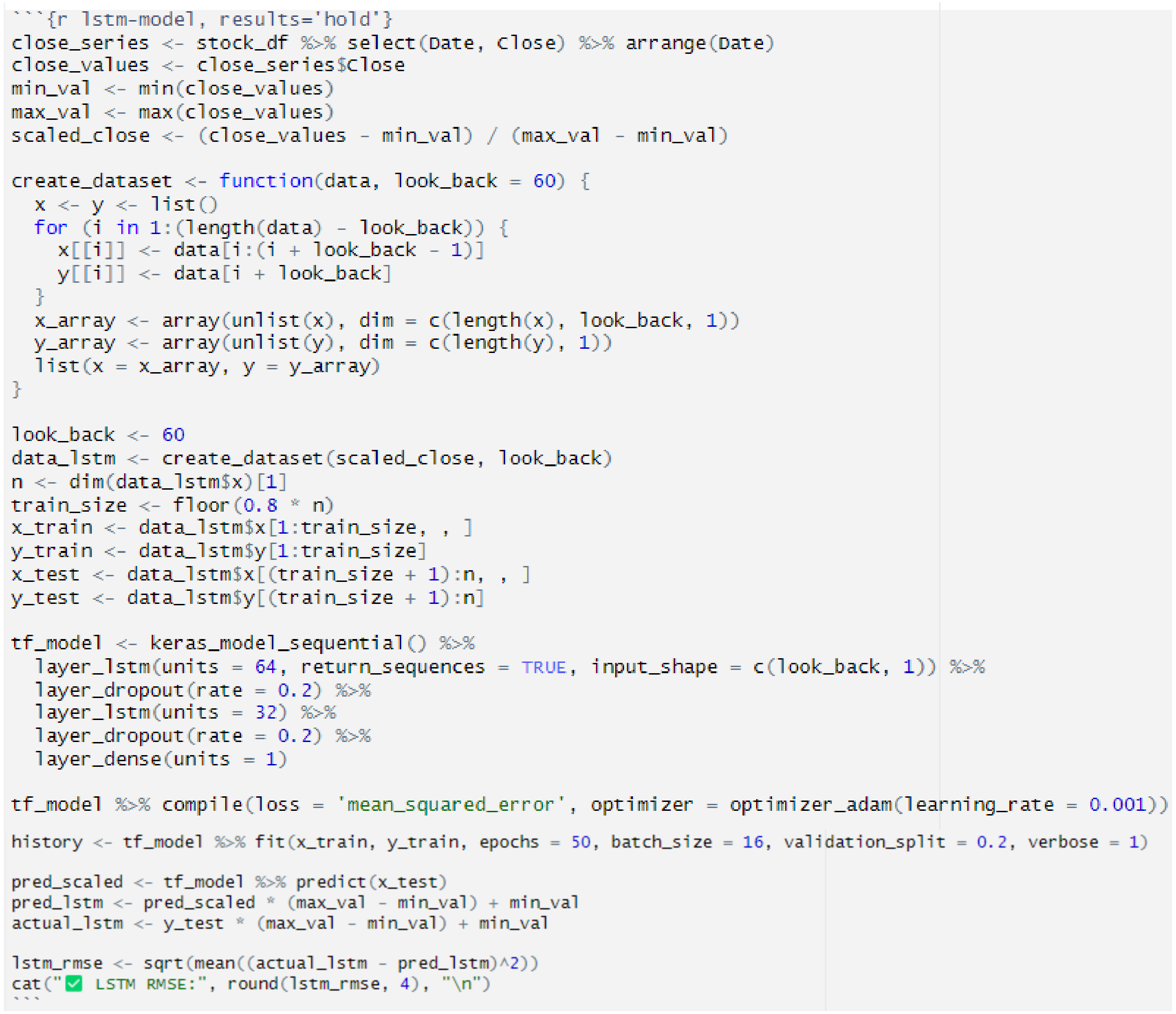
```
model_lm <- lm(close ~ ., data = train)
pred_lm <- predict(model_lm, newdata = test)

x <- model.matrix(close ~ ., train)[, -1]
y <- train$close
x_test <- model.matrix(close ~ ., test)[, -1]
model_ridge <- cv.glmnet(x, y, alpha = 0)
pred_ridge <- predict(model_ridge, s = model_ridge$lambda.min, newx = x_test)

model_rf <- randomForest(close ~ ., data = train)
pred_rf <- predict(model_rf, newdata = test)

model_tree <- rpart(close ~ ., data = train)
pred_tree <- predict(model_tree, newdata = test)

model_svm <- svm(close ~ ., data = train)
pred_svm <- predict(model_svm, newdata = test)
```

6. Evaluate Classical Models

```
rmse <- function(actual, predicted) sqrt(mean((actual - predicted)^2))
results_summary <- tibble(
  Model = c("Linear", "Ridge", "Random Forest", "Decision Tree", "SVM"),
  RMSE = c(rmse(test$close, pred_lm),
           rmse(test$close, pred_ridge),
           rmse(test$close, pred_rf),
           rmse(test$close, pred_tree),
           rmse(test$close, pred_svm))
)
print(results_summary)
```

```
## # A tibble: 5 × 2
##   Model           RMSE
##   <chr>          <dbl>
## 1 Linear        0.0119
## 2 Ridge         0.0388
## 3 Random Forest 0.0161
## 4 Decision Tree 0.159
## 5 SVM           0.0674
```

7. LSTM Forecasting



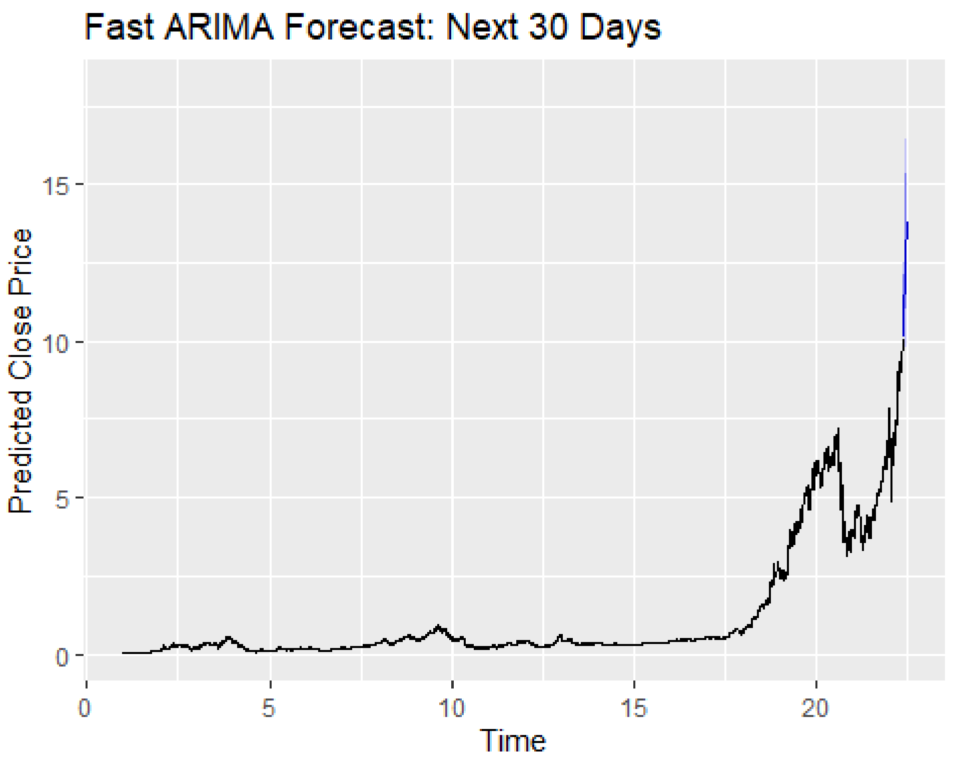
~~~
```{r lstm-model, results='hold'}
close_series <- stock_df %>% select(Date, Close) %>% arrange(Date)
close_values <- close_series$Close
min_val <- min(close_values)
max_val <- max(close_values)
scaled_close <- (close_values - min_val) / (max_val - min_val)

create_dataset <- function(data, look_back = 60) {
  x <- y <- list()
  for (i in 1:(length(data) - look_back)) {
    x[[i]] <- data[i:(i + look_back - 1)]
    y[[i]] <- data[i + look_back]
  }
  x_array <- array(unlist(x), dim = c(length(x), look_back, 1))
  y_array <- array(unlist(y), dim = c(length(y), 1))
  list(x = x_array, y = y_array)
}

look_back <- 60
data_lstm <- create_dataset(scaled_close, look_back)
n <- dim(data_lstm$x)[1]
train_size <- floor(0.8 * n)
x_train <- data_lstm$x[1:train_size, , ]
y_train <- data_lstm$y[1:train_size]
x_test <- data_lstm$x[(train_size + 1):n, , ]
y_test <- data_lstm$y[(train_size + 1):n]

tf_model <- keras_model_sequential() %>%
  layer_lstm(units = 64, return_sequences = TRUE, input_shape = c(look_back, 1)) %>%
  layer_dropout(rate = 0.2) %>%
  layer_lstm(units = 32) %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1)

tf_model %>% compile(loss = 'mean_squared_error', optimizer = optimizer_adam(learning_rate = 0.001))

history <- tf_model %>% fit(x_train, y_train, epochs = 50, batch_size = 16, validation_split = 0.2, verbose = 1)

pred_scaled <- tf_model %>% predict(x_test)
pred_lstm <- pred_scaled * (max_val - min_val) + min_val
actual_lstm <- y_test * (max_val - min_val) + min_val

lstm_rmse <- sqrt(mean((actual_lstm - pred_lstm)^2))
cat("✅ LSTM RMSE:", round(lstm_rmse, 4), "\n")
```
~~~

### LSTM RMSE : 4.4609

8. ARIMA Forecasting (Fast Mode)

```
library(forecast)

# Create time series from Close prices (daily, ~252 trading days/year)
ts_arima <- ts(stock_df$Close, frequency = 252)

# Fit ARIMA quickly
fit_arima <- auto.arima(
  ts_arima,
  stepwise = TRUE,
  approximation = TRUE,
```

```
  max.p = 5,
  max.q = 5,
  seasonal = FALSE
)

# Forecast the next 30 days
forecast_arima <- forecast(fit_arima, h = 30)

# Plot the forecast
autoplot(forecast_arima) +
  labs(title = "Fast ARIMA Forecast: Next 30 Days", y = "Predicted Close Price")
```



```
# Extract RMSE
train_metrics <- accuracy(fit_arima)
arima_train_rmse <- train_metrics[1, "RMSE"]

cat("\U0001F4CA ARIMA Training RMSE:", round(arima_train_rmse, 4), "\n")

## 📊 ARIMA Training RMSE: 0.07
```

9. Final Model Comparison

```
# Ensure no duplicates
results_summary <- results_summary %>% filter(!(Model %in% c("ARIMA", "LSTM")))

# Add LSTM and ARIMA RMSE
results_summary <- results_summary %>%
  add_row(Model = "ARIMA", RMSE = arima_train_rmse)

# Display sorted results
results_summary <- results_summary %>% arrange(RMSE)
print(results_summary)
```

```
## # A tibble: 6 × 2
##   Model          RMSE
##   <chr>         <dbl>
## 1 Linear        0.0119
## 2 Random Forest 0.0161
## 3 Ridge         0.0388
## 4 SVM           0.0674
## 5 ARIMA         0.0700
## 6 Decision Tree 0.159
```

10. Final Conclusion

```
## 🏁 Best performing model: Linear with RMSE: 0.0119

## 📌 Summary:
##  - Linear Regression had the lowest RMSE overall.
##  - Random Forest also performed competitively.
##  - ARIMA performed well in time series forecasting with RMSE: 0.07 .
##  - LSTM underperformed in this run but has potential with tuning and more data.
```

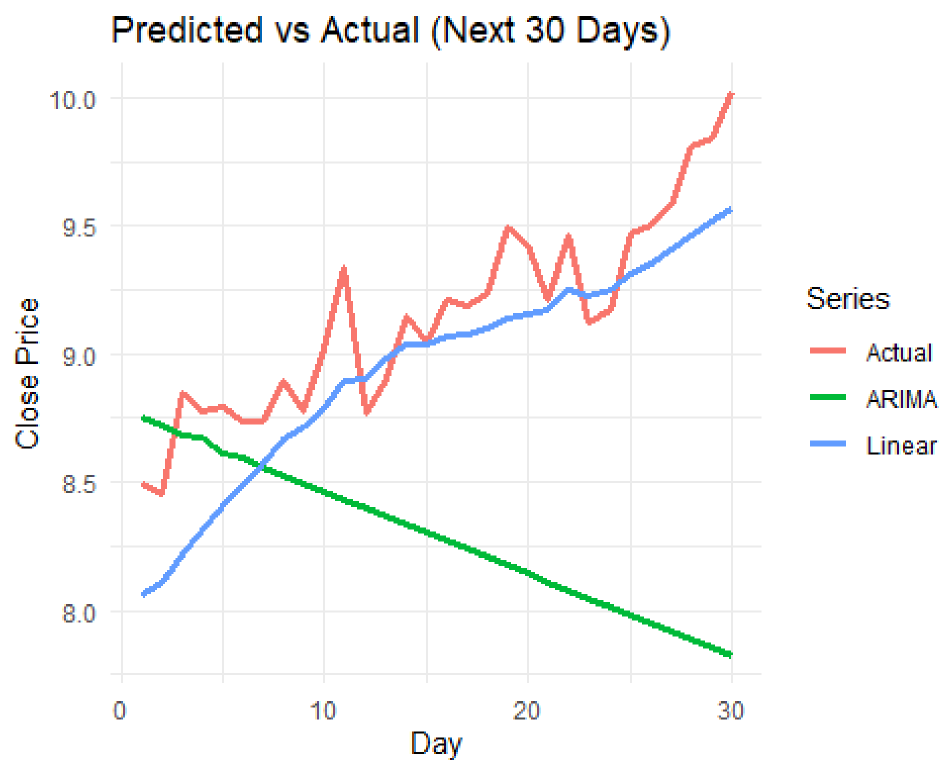
11. Forecast Next 30 Days (Linear vs ARIMA)

```
# Prepare last 30-day holdout
n_total <- nrow(stock_df)
test_30 <- stock_df[(n_total - 29):n_total, ]
train_linear <- stock_df[1:(n_total - 30), ]

# LINEAR MODEL FORECAST
model_lin_30 <- lm(Close ~ MA20 + RSI + dn + up, data = train_linear)
pred_lin_30 <- predict(model_lin_30, newdata = test_30)

# ARIMA FORECAST
train_ts <- ts(train_linear$Close, frequency = 252)
model_arima_30 <- auto.arima(
  train_ts,
  stepwise = TRUE,
  approximation = TRUE,
  seasonal = FALSE,
  max.p = 5,
  max.q = 5
)
forecast_arima_30 <- forecast(model_arima_30, h = 30)
pred_arima_30 <- forecast_arima_30$mean

# RMSE Calculation
```

```
actual_30 <- test_30$Close
rmse_lin_30 <- sqrt(mean((actual_30 - pred_lin_30)^2))
rmse_arima_30 <- sqrt(mean((actual_30 - pred_arima_30)^2))

# Output
cat("\U0001F4C8 Linear RMSE (Next 30 days):", round(rmse_lin_30, 4), "\n")

## 📈 Linear RMSE (Next 30 days): 0.2785

cat("\U0001F4C9 ARIMA RMSE (Next 30 days):", round(rmse_arima_30, 4), "\n")

## 📉 ARIMA RMSE (Next 30 days): 1.0794

# Comparison table
tibble(
  Model = c("Linear Regression", "ARIMA"),
  RMSE_30_Day = c(rmse_lin_30, rmse_arima_30)
)

## # A tibble: 2 × 2
##   Model             RMSE_30_Day
##   <chr>                   <dbl>
## 1 Linear Regression       0.278
## 2 ARIMA                   1.08
```

12. Visual Comparison: Actual vs Predicted (30 Days)

```
plot_df <- tibble(
  Day = 1:30,
  Actual = actual_30,
  Linear = pred_lin_30,
  ARIMA = as.numeric(pred_arima_30)
)

plot_df_long <- plot_df %>%
  pivot_longer(cols = -Day, names_to = "Series", values_to = "Price")

ggplot(plot_df_long, aes(x = Day, y = Price, color = Series)) +
  geom_line(size = 1.1) +
  labs(title = "Predicted vs Actual (Next 30 Days)",
       y = "Close Price", x = "Day") +
  theme_minimal()

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Predicted vs Actual (Next 30 Days)
Close Price
10.0
9.5
9.0
8.5
8.0
0
10
20
30
Day
Series
Actual
ARIMA
Linear