Stock Forecasting Project

Contents

Libraries **2**

Data Gathering **2**
User Inputs (Stock and Date) 2
Stock Data Collection 2

Data Exploration & Feature Engineering **2**
EDA 2
Feature Engineering 4

Model 1: ARIMA with Exogenous Regressors (ARIMAX) **10**
Preperation for model 1 10
Build model 1 20
Performance 21

Libraries

```
library(tidyverse)
library(lubridate)
library(quantmod)
library(tseries)
library(forecast)
```

Data Gathering

User Inputs (Stock and Date)

```
symbol <- "AAPL"
start_date <- as.Date("2022-01-01")
end_date <- Sys.Date()
```

Stock Data Collection

```
getSymbols(symbol,
           src  = "yahoo",
           from = start_date,
           to   = end_date,
           auto.assign = TRUE)
```

```
## [1] "AAPL"
```

```
stock_data <- get(symbol)

# head(stock_data)
```

Data Exploration & Feature Engineering

EDA

```
# Convert time-series data (xts object) to a regular tibble
df_stock <- tibble(
  date     = index(stock_data),
  open     = as.numeric(stock_data[, paste0(symbol, ".Open")]),
  high     = as.numeric(stock_data[, paste0(symbol, ".High")]),
  low      = as.numeric(stock_data[, paste0(symbol, ".Low")]),
  close    = as.numeric(stock_data[, paste0(symbol, ".Close")]),
  volume   = as.numeric(stock_data[, paste0(symbol, ".Volume")]),
  adjusted = as.numeric(stock_data[, paste0(symbol, ".Adjusted")])
)

glimpse(df_stock)
```

```
## Rows: 803
## Columns: 7
## $ date     <date> 2022-01-03, 2022-01-04, 2022-01-05, 2022-01-06, 2022-01-07, ~
## $ open     <dbl> 177.83, 182.63, 179.61, 172.70, 172.89, 169.08, 172.32, 176.1~
## $ high     <dbl> 182.88, 182.94, 180.17, 175.30, 174.14, 172.50, 175.18, 177.1~
```

```
## $ low      <dbl> 177.71, 179.12, 174.64, 171.64, 171.03, 168.17, 170.82, 174.8~
## $ close    <dbl> 182.01, 179.70, 174.92, 172.00, 172.17, 172.19, 175.08, 175.5~
## $ volume   <dbl> 104487900, 99310400, 94537600, 96904000, 86709100, 106765600,~
## $ adjusted <dbl> 178.8799, 176.6097, 171.9118, 169.0421, 169.2091, 169.2288, 1~
```

```r
summary(df_stock)
```

```
##       date                 open            high             low
##  Min.   :2022-01-03   Min.   :126.0   Min.   :127.8   Min.   :124.2
##  1st Qu.:2022-10-19   1st Qu.:158.7   1st Qu.:160.4   1st Qu.:155.2
##  Median :2023-08-09   Median :174.8   Median :176.6   Median :173.5
##  Mean   :2023-08-08   Mean   :181.6   Mean   :183.5   Mean   :179.8
##  3rd Qu.:2024-05-26   3rd Qu.:195.6   3rd Qu.:196.9   3rd Qu.:194.2
##  Max.   :2025-03-17   Max.   :258.2   Max.   :260.1   Max.   :257.6
##      close           volume             adjusted
##  Min.   :125.0   Min.   : 23234700   Min.   :123.6
##  1st Qu.:157.6   1st Qu.: 48102550   1st Qu.:155.6
##  Median :175.1   Median : 60882300   Median :173.5
##  Mean   :181.7   Mean   : 67183503   Mean   :180.4
##  3rd Qu.:195.7   3rd Qu.: 79567300   3rd Qu.:194.1
##  Max.   :259.0   Max.   :318679900   Max.   :258.7
```

```r
# Plot Adjusted Closing Price over time with a dynamic title
ggplot(df_stock, aes(x = date, y = adjusted)) +
  geom_line() +
  labs(title = paste(symbol, "Adjusted Closing Price"),
       x = "Date",
       y = "Adjusted Price") +
  theme_minimal()
```


```
# Check if there is any NA (rare to have NA)
df_stock %>%
  summarize(across(everything(), ~ sum(is.na(.))))
```

```
## # A tibble: 1 x 7
##    date  open  high   low close volume adjusted
##   <int> <int> <int> <int> <int>  <int>    <int>
## 1     0     0     0     0     0      0        0
```

Feature Engineering

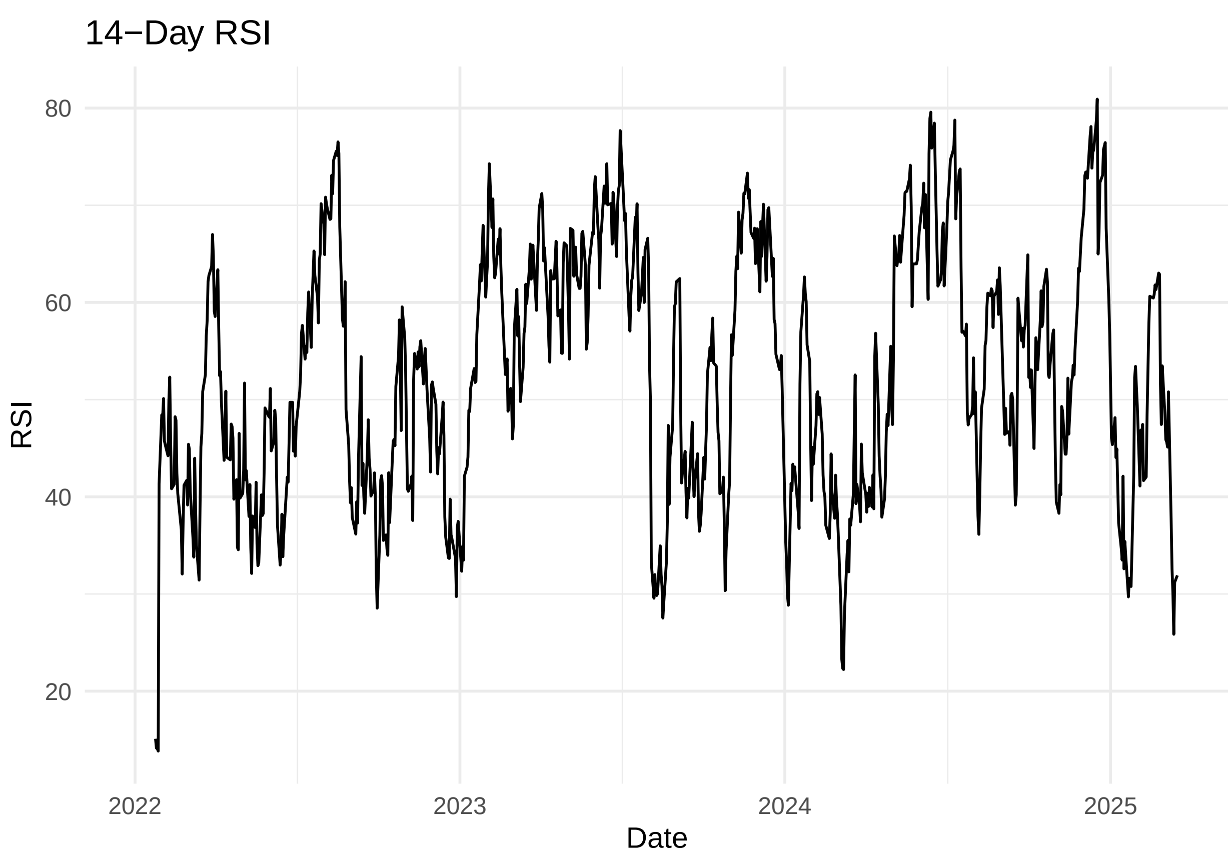
Add new variables

```
# Calculate Bollinger Bands using high, low, and close
bb <- BBands(HLC = df_stock %>% select(high, low, close),
             n = 20, maType = "SMA", sd = 2)

# Calculate MACD based on the adjusted closing price
macd_values <- MACD(df_stock$adjusted, nFast = 12, nSlow = 26, nSig = 9)

# Add all new variables
df_stock <- df_stock %>%
  arrange(date) %>%
  mutate(
    daily_return = (adjusted - lag(adjusted)) / lag(adjusted) * 100,
    lag1_close = lag(adjusted, 1),
```

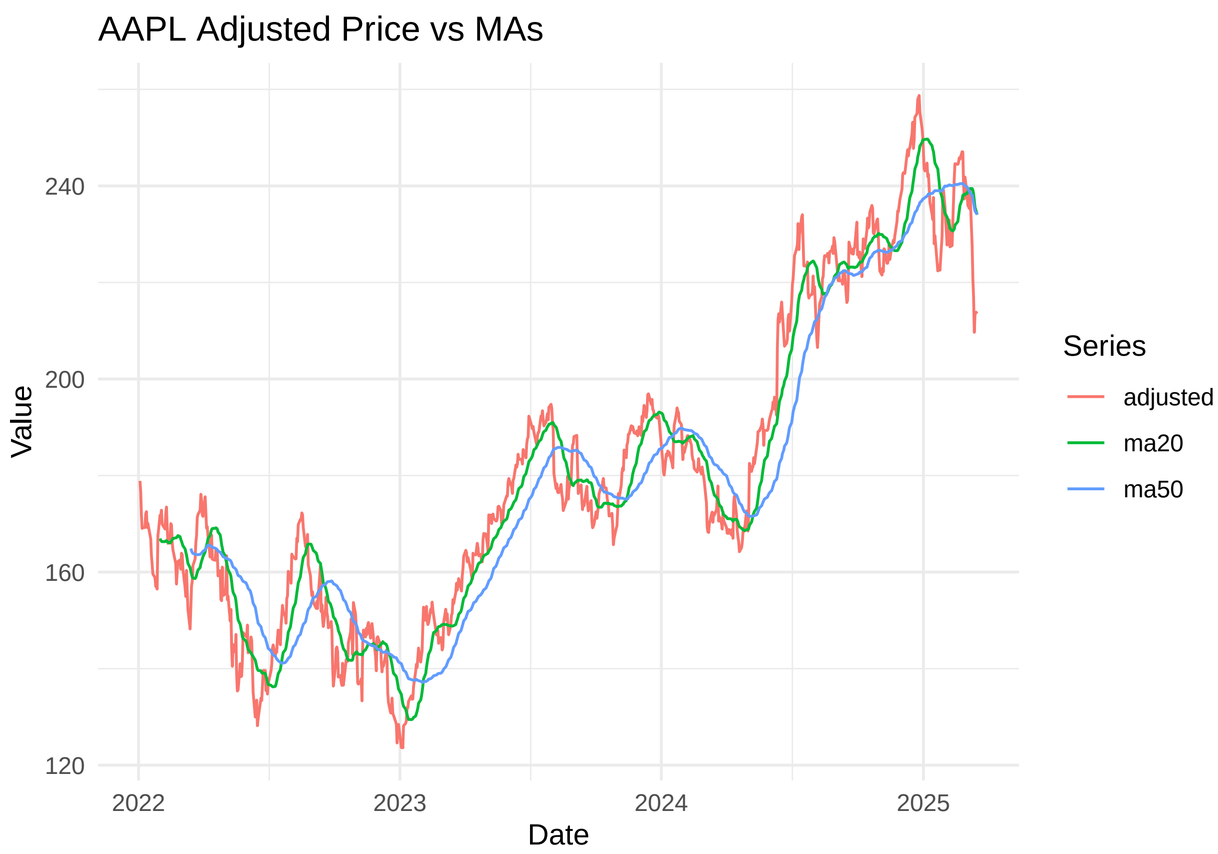
```r
    lag2_close = lag(adjusted, 2),
    ma20 = rollmean(adjusted, k = 20, fill = NA, align = "right"),
    ma50 = rollmean(adjusted, k = 50, fill = NA, align = "right"),
    rsi14 = RSI(adjusted, n = 14),
    bb_dn   = bb[, "dn"],
    bb_mavg = bb[, "mavg"],
    bb_up   = bb[, "up"],
    bb_pctB = bb[, "pctB"],
    macd    = macd_values[, "macd"],
    macdSig = macd_values[, "signal"],
    rolling_sd_20 = rollapply(daily_return, width = 20,
                              FUN = sd, fill = NA, align = "right"),
    wday = wday(date, label = TRUE),
    sin_wday = sin(2 * pi * wday(date) / 7),
    cos_wday = cos(2 * pi * wday(date) / 7)
  )
```

Visualize new variables

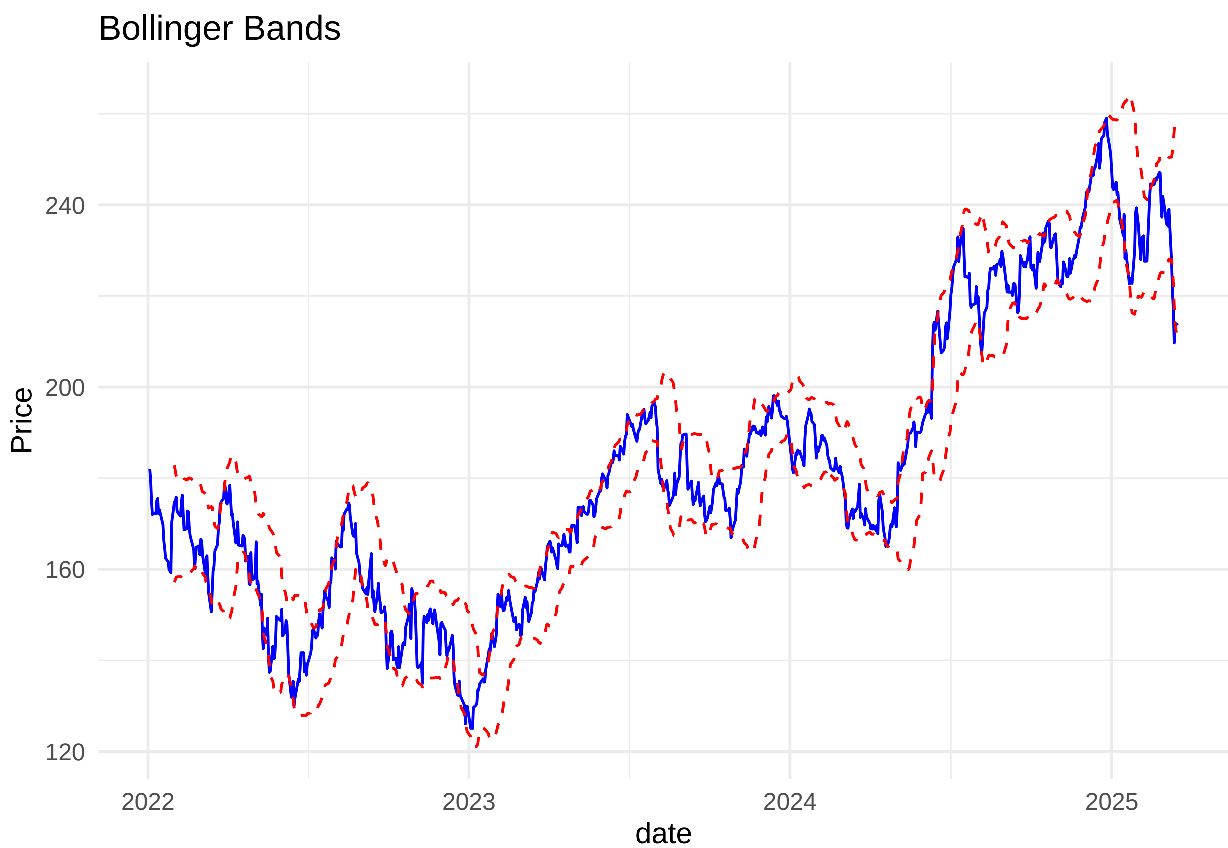
```r
# 14-Day RSI
ggplot(df_stock, aes(x = date, y = rsi14)) +
  geom_line() +
  labs(title = "14-Day RSI", x = "Date", y = "RSI") +
  theme_minimal()
```



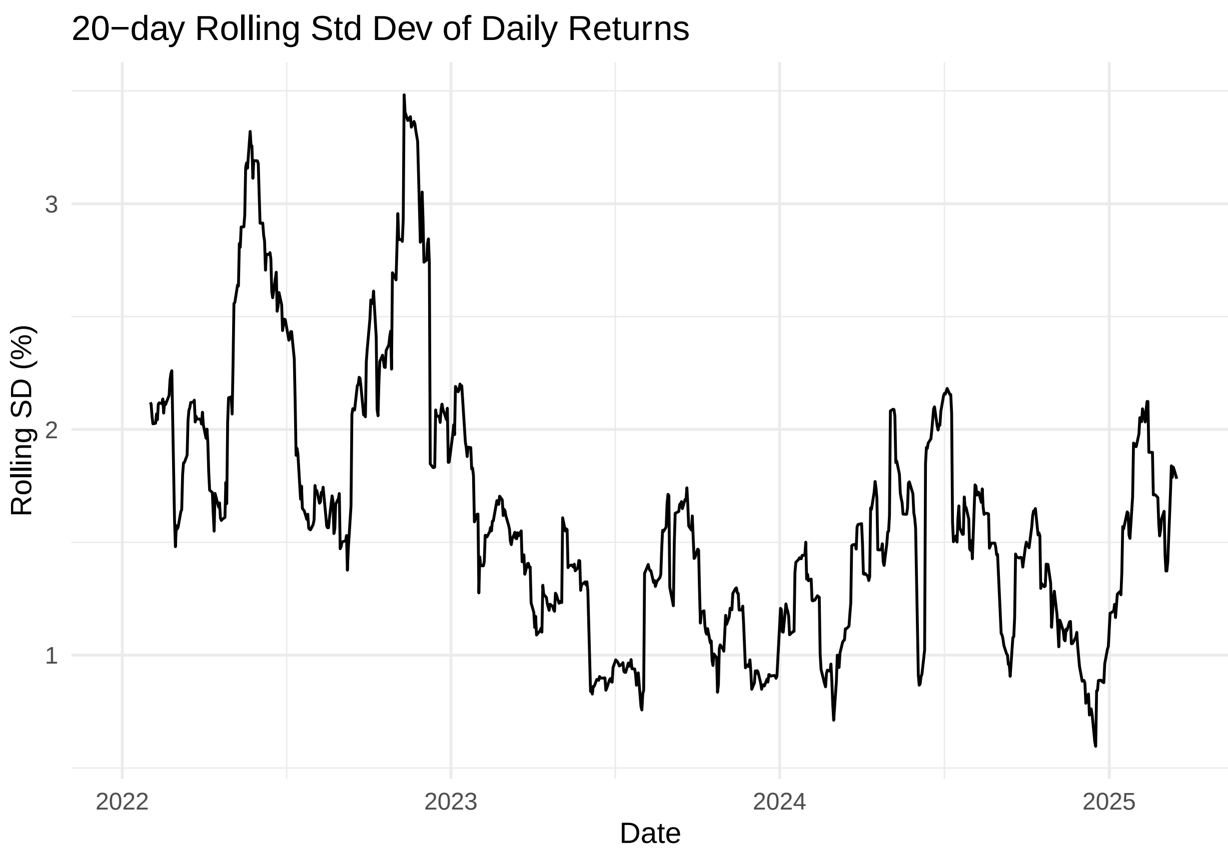
```r
# MAs
df_stock_long <- df_stock %>%
  select(date, adjusted, ma20, ma50) %>%
  pivot_longer(cols = c("adjusted", "ma20", "ma50"),
               names_to = "variable", values_to = "value")

ggplot(df_stock_long, aes(x = date, y = value, color = variable)) +
  geom_line() +
  labs(title = paste(symbol, "Adjusted Price vs MAs"),
       x = "Date", y = "Value", color = "Series") +
  theme_minimal()
```




```r
# Bollinger Bands
ggplot(df_stock, aes(x = date)) +
  geom_line(aes(y = close), color = "blue") +
  geom_line(aes(y = bb_dn), color = "red", linetype = "dashed") +
  geom_line(aes(y = bb_up), color = "red", linetype = "dashed") +
  labs(title = "Bollinger Bands", y = "Price") +
  theme_minimal()
```




```
# 20-day Rolling Std Dev of Daily Returns
ggplot(df_stock, aes(x = date, y = rolling_sd_20)) +
  geom_line() +
  labs(title = "20-day Rolling Std Dev of Daily Returns",
       x = "Date", y = "Rolling SD (%)") +
  theme_minimal()
```

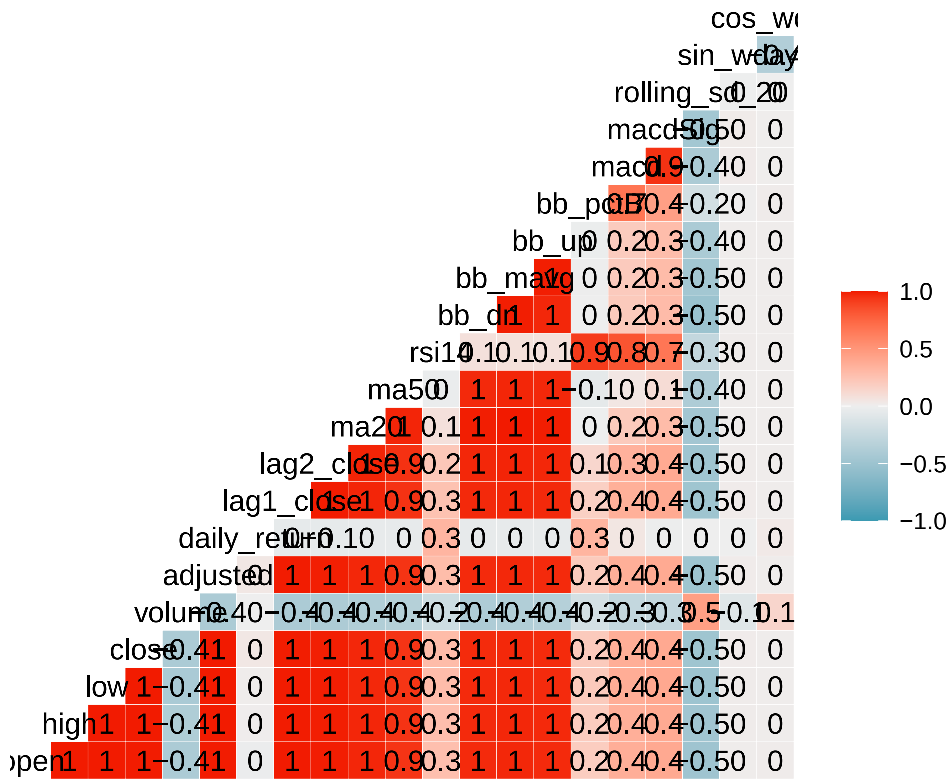


Other tests and analysis

```
# Correlation Matrix of Numeric Features
df_numerics <- df_stock %>%
  select(where(is.numeric)) %>%
  drop_na()  # Remove rows with NA for accurate correlation

GGally::ggcorr(df_numerics,
               method = c("pairwise.complete.obs", "pearson"),
               label  = TRUE) +
  ggtitle("Correlation Matrix of Numeric Features")
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

Correlation Matrix of Numeric Features



```r
# Stationarity Tests: Adjusted Price and Daily Returns
adf_result <- adf.test(df_stock$adjusted, alternative = "stationary")
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  df_stock$adjusted
## Dickey-Fuller = -3.088, Lag order = 9, p-value = 0.1178
## alternative hypothesis: stationary
```

```r
adf_returns <- adf.test(na.omit(df_stock$daily_return), alternative = "stationary")
```

```
## Warning in adf.test(na.omit(df_stock$daily_return), alternative =
## "stationary"): p-value smaller than printed p-value
```

```r
print(adf_returns)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(df_stock$daily_return)
## Dickey-Fuller = -9.0013, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

Findings on Stationarity: - The adjusted price is non-stationary, which is expected because stock prices tend to follow a random walk. - The daily returns are stationary, which is typical for financial return series since they fluctuate around a constant mean.

Model 1: ARIMA with Exogenous Regressors (ARIMAX)

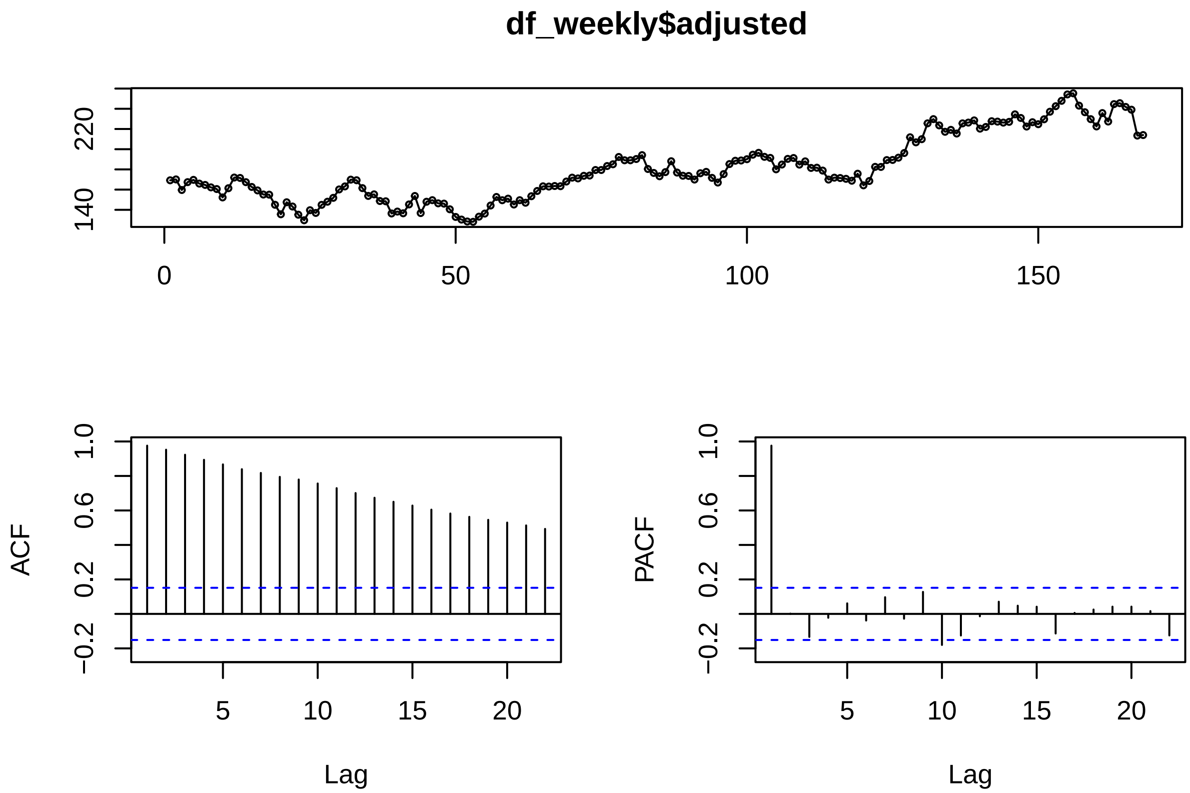
Preperation for model 1

Convert to weekly data

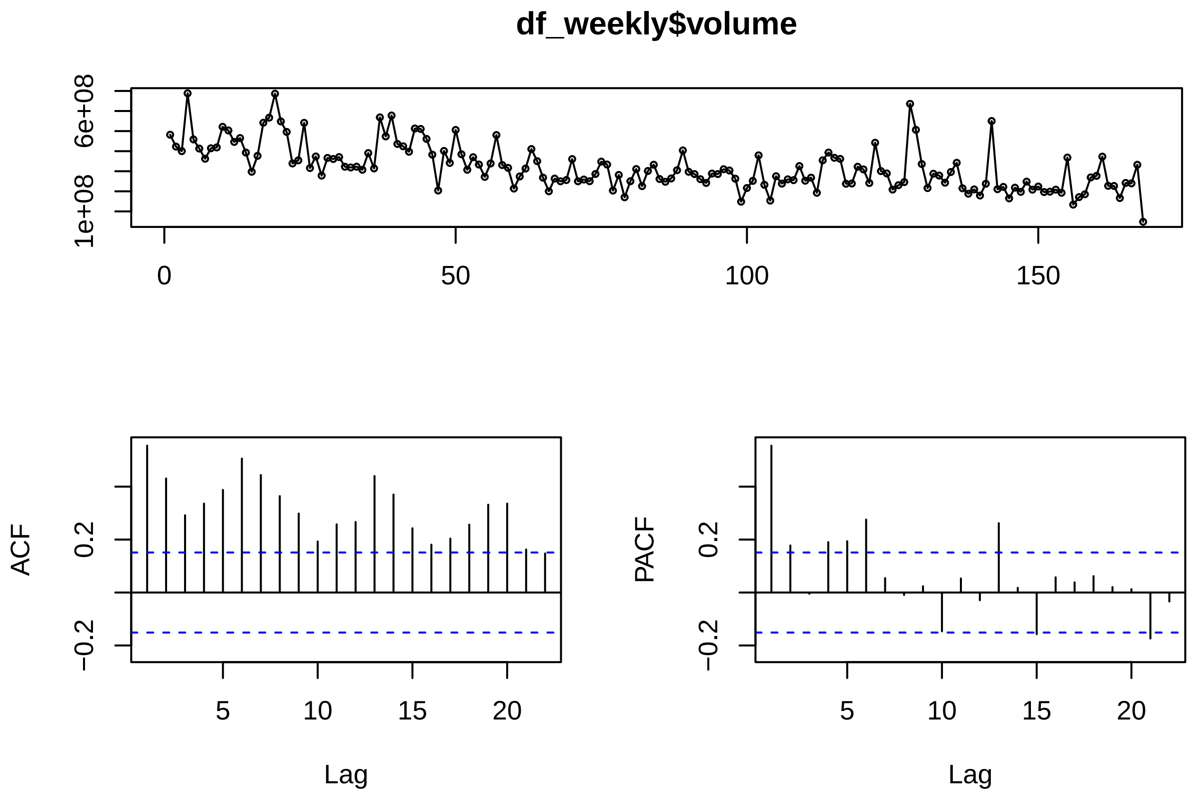
```r
# Change to weekly data for less computation
df_weekly <- df_stock %>%
  mutate(week = floor_date(date, unit = "week", week_start = 1)) %>%
  group_by(week) %>%
  summarise(
    open = first(open),
    high = max(high, na.rm = TRUE),
    low = min(low, na.rm = TRUE),
    close = last(close),
    volume = sum(volume, na.rm = TRUE),
    first_adj = first(adjusted),
    adjusted = last(adjusted),
    weekly_return = (last(adjusted) / first_adj - 1) * 100,
    ma20 = last(ma20),
    ma50 = last(ma50),
    rsi14 = last(rsi14),
    bb_dn = last(bb_dn),
    bb_mavg = last(bb_mavg),
    bb_up = last(bb_up),
    bb_pctB = last(bb_pctB),
    macd = last(macd),
    macdSig = last(macdSig),
    rolling_sd_20 = last(rolling_sd_20),
    wday = last(wday),
    sin_wday = last(sin_wday),
    cos_wday = last(cos_wday)
  ) %>%
  ungroup() %>%
  select(-c(first_adj, wday))

head(df_weekly, 2)
```

```
## # A tibble: 2 x 20
##   week        open  high   low close   volume adjusted weekly_return  ma20  ma50
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>         <dbl> <dbl> <dbl>
## 1 2022-01-03  178.  183.  171.  172.   4.82e8     169.        -5.41     NA    NA
## 2 2022-01-10  169.  177.  168.  173.   4.23e8     170.         0.511    NA    NA
## # i 10 more variables: rsi14 <dbl>, bb_dn <dbl>, bb_mavg <dbl>, bb_up <dbl>,
## #   bb_pctB <dbl>, macd <dbl>, macdSig <dbl>, rolling_sd_20 <dbl>,
## #   sin_wday <dbl>, cos_wday <dbl>
```

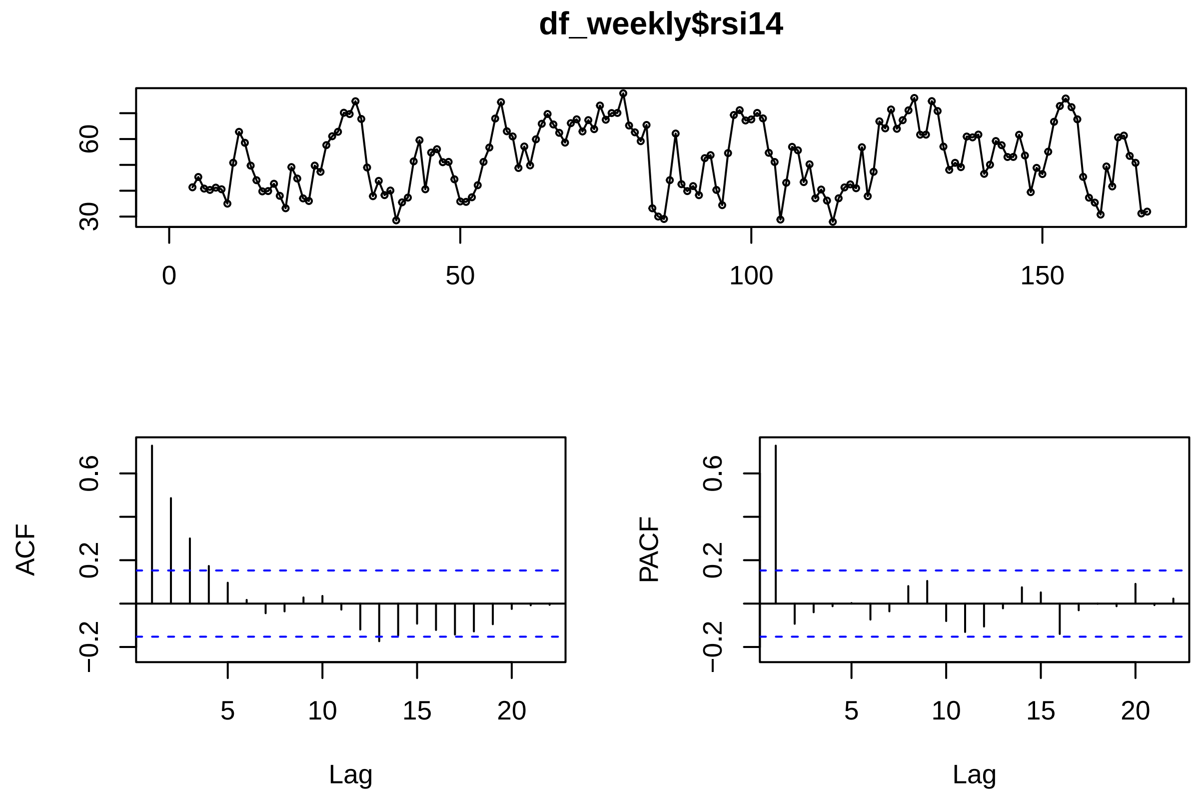
```r
tsdisplay(df_weekly$adjusted)
```



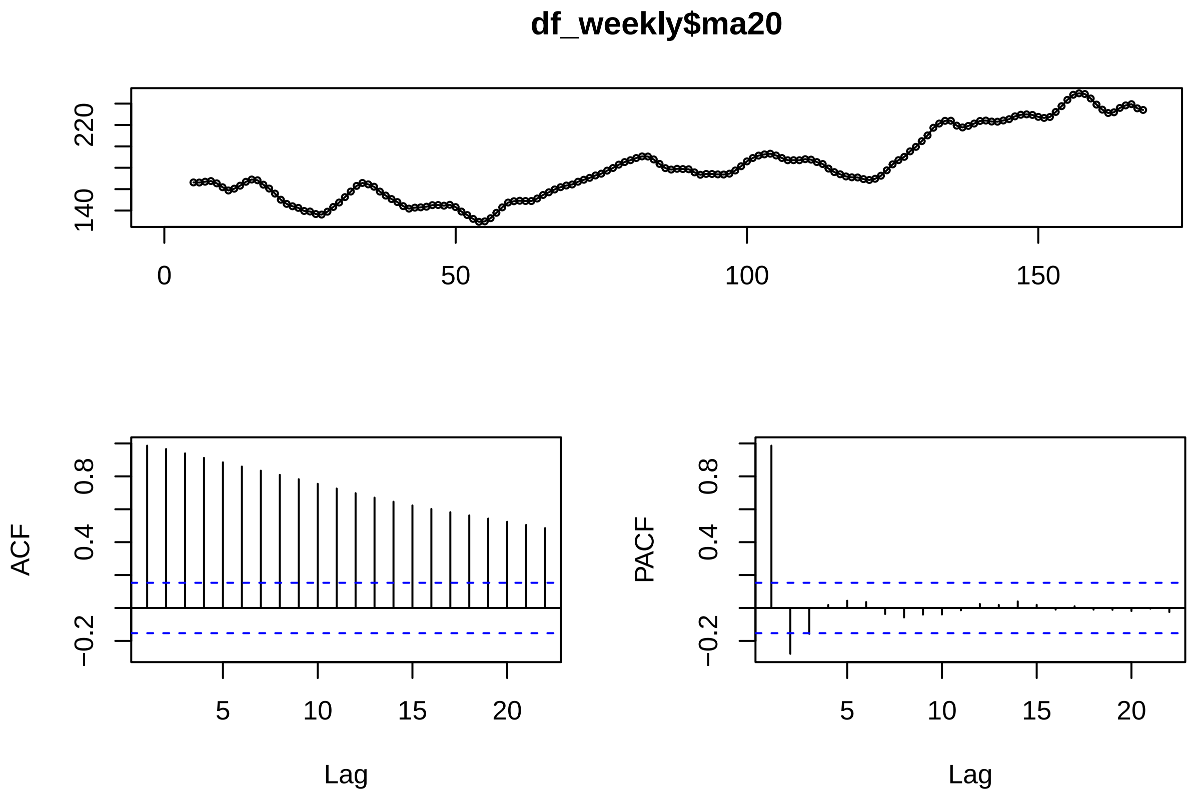

```
tsdisplay(df_weekly$volume)
```



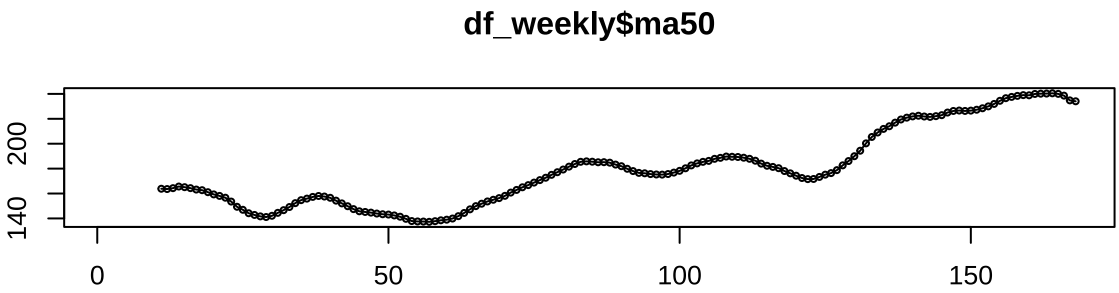

```
tsdisplay(df_weekly$rsi14)
```

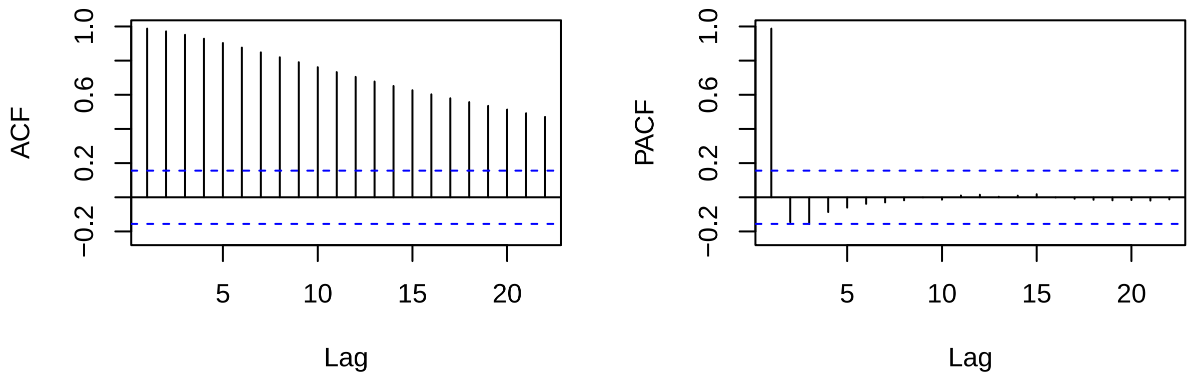



```r
tsdisplay(df_weekly$ma20)
```




```r
tsdisplay(df_weekly$ma50)
```





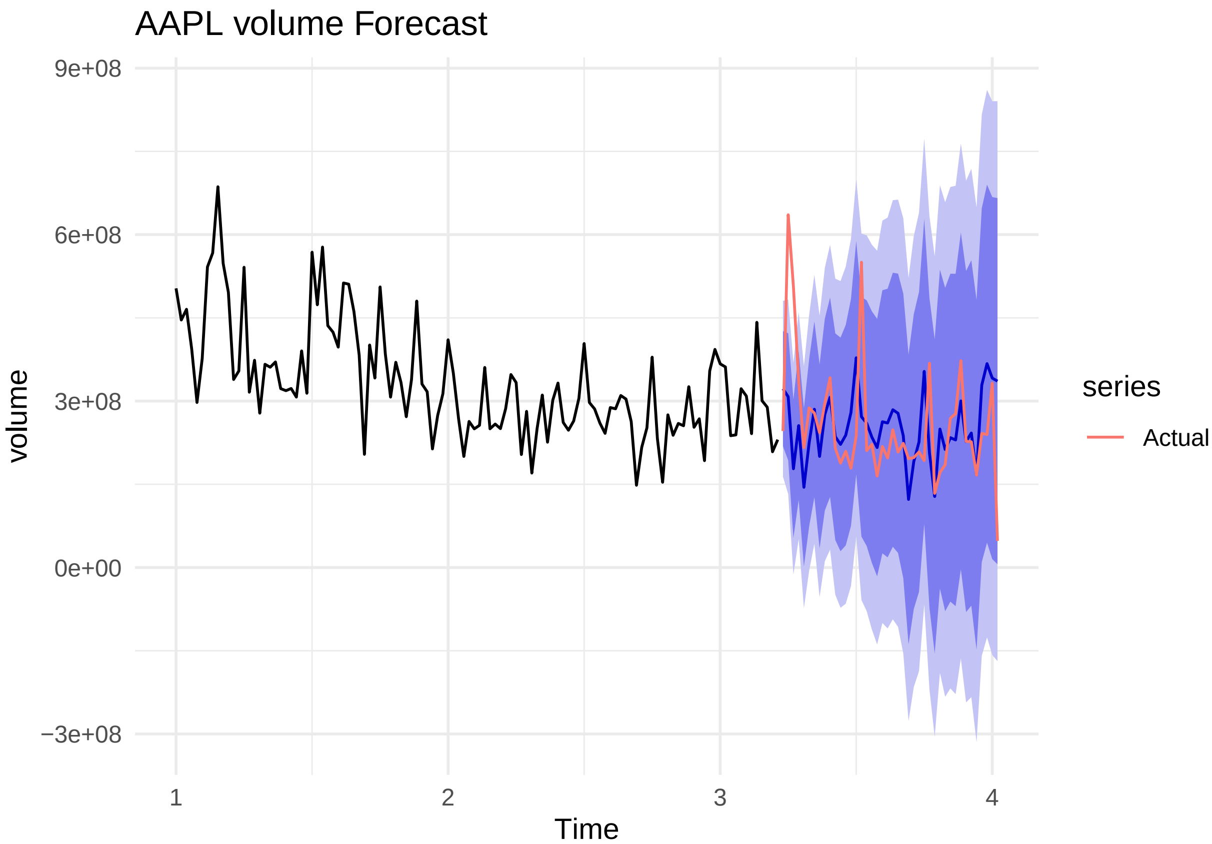
Train test split

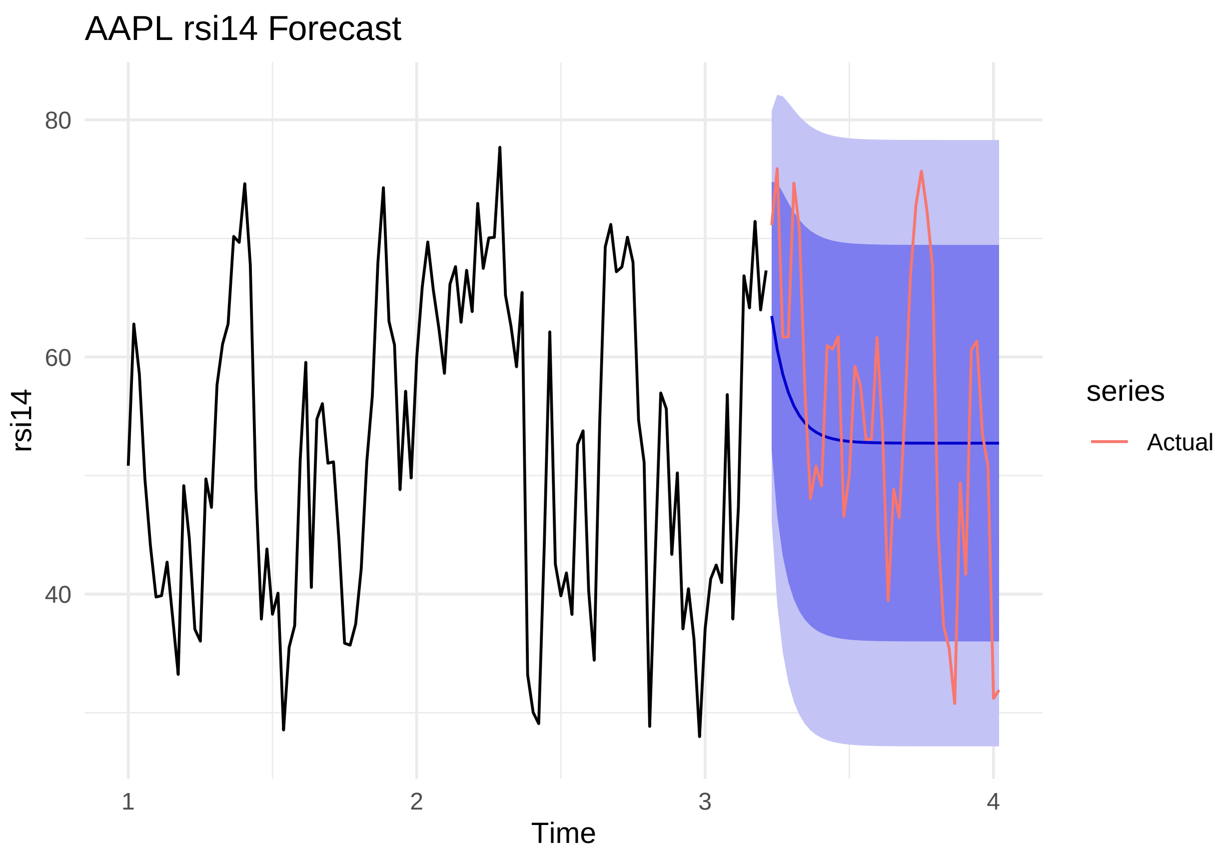
```
cutoff_date <- as.Date("2024-05-31")

train_data <- df_weekly %>%
  filter(week <= cutoff_date) %>%
  drop_na(adjusted, volume, rsi14, ma20, ma50)

test_data <- df_weekly %>%
  filter(week > cutoff_date) %>%
  drop_na(adjusted, volume, rsi14, ma20, ma50)
```

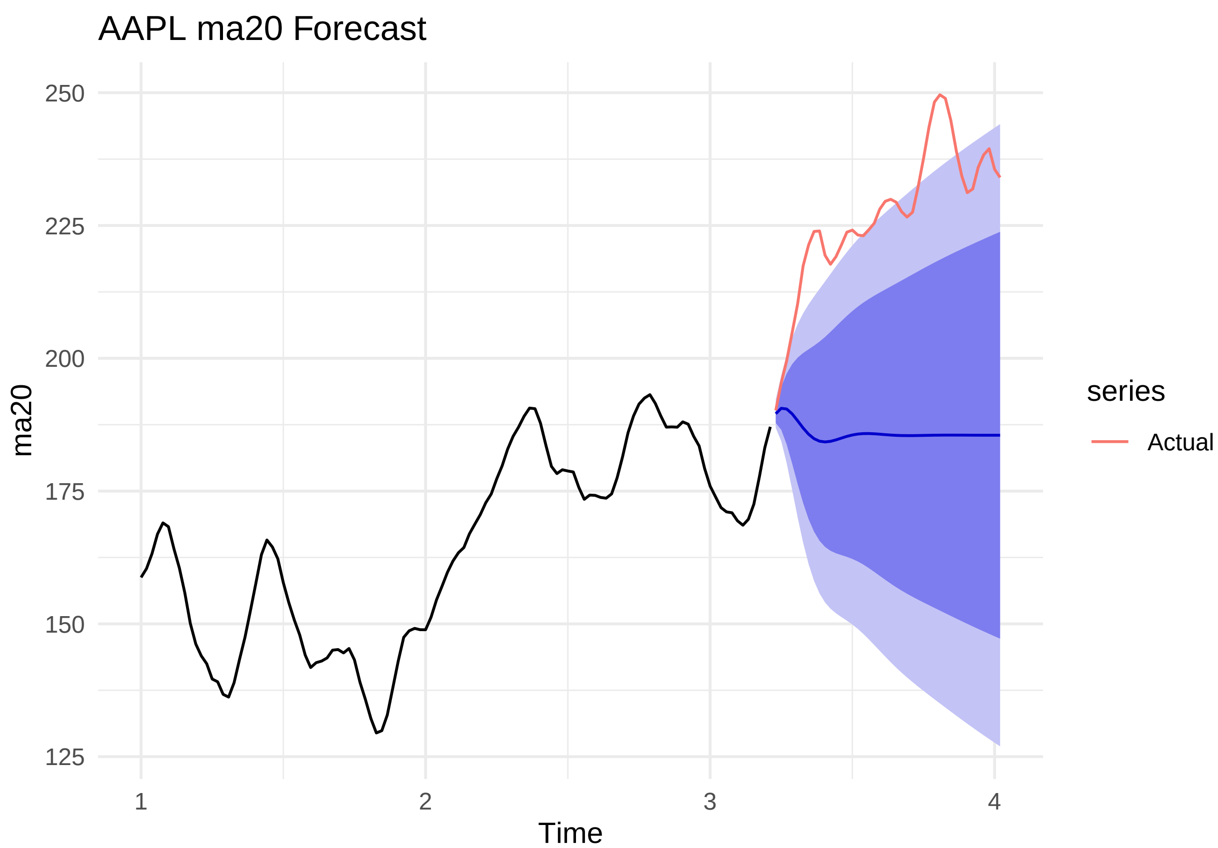
Forecast exogenous variables

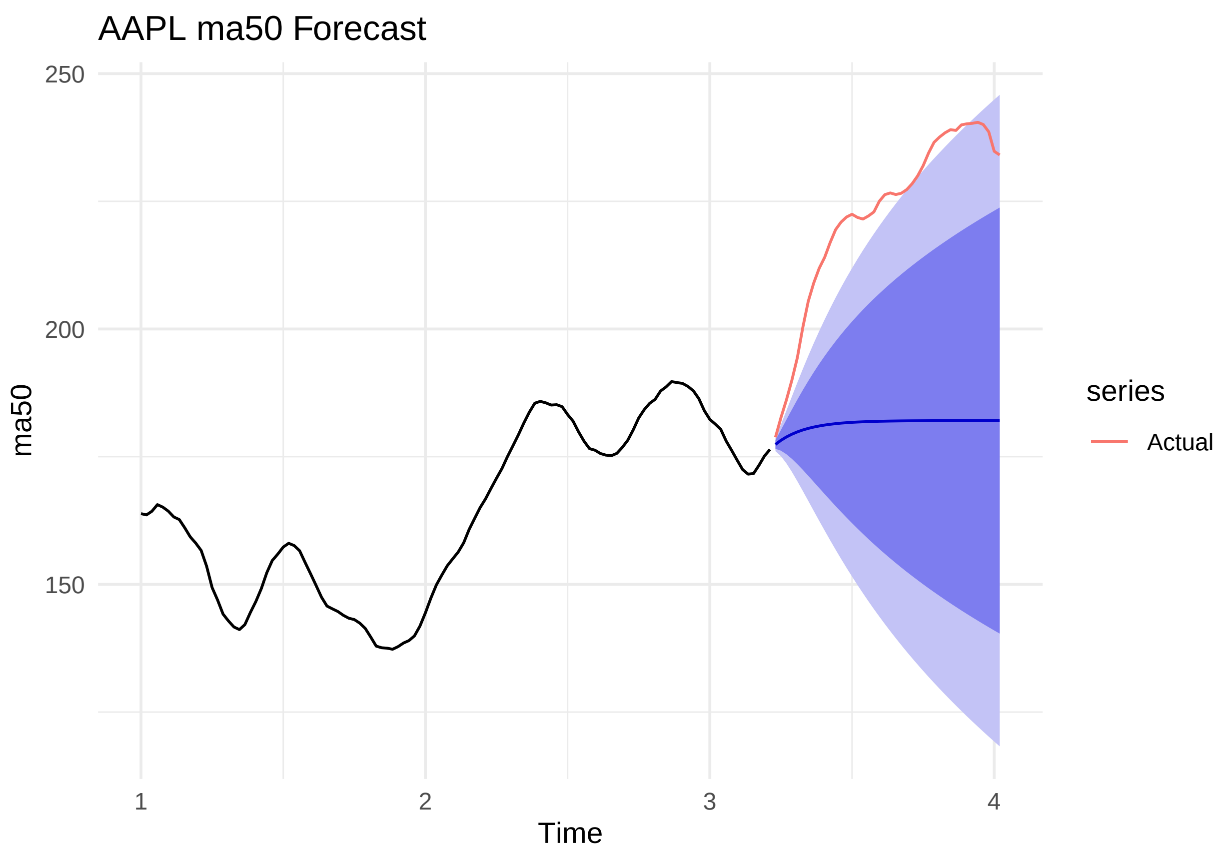
```
# Define forecast function
forecast_exog <- function(train_df, test_df, var_name, freq = 52) {

  train_ts <- ts(train_df[[var_name]], frequency = freq)
  fit <- auto.arima(train_ts, stepwise = TRUE, approximation = TRUE)
  h <- nrow(test_df)
  fc <- forecast(fit, h = h)

  list(forecast_obj = fc,
       forecast = as.numeric(fc$mean),
       AIC = AIC(fit),
```

```r
      actual = ts(test_df[[var_name]], frequency = freq,
                  start = c(1, length(train_df[[var_name]]) + 1)))
}


# Start forecast

exog_vars <- c("volume", "rsi14", "ma20", "ma50")
exog_perf <- tibble(variable = character(),
                    MAPE = numeric(),
                    MSE = numeric(),
                    AIC = numeric())


exog_forecasts <- list()

for (var in exog_vars) {
  fc_result <- forecast_exog(train_data, test_data, var)
  exog_forecasts[[var]] <- fc_result$forecast

  # Compute performance metrics
  actual_ts <- fc_result$actual
  mape_exog <- mean(abs(fc_result$forecast - actual_ts) / abs(actual_ts)) * 100
  mse_exog <- mean((fc_result$forecast - actual_ts)^2)

  exog_perf <- exog_perf %>%
    add_row(variable = var, MAPE = mape_exog,
            MSE = mse_exog, AIC = fc_result$AIC)

  # Plot
  print(
    autoplot(fc_result$forecast_obj) +
      autolayer(actual_ts, series = "Actual") +
      labs(title = paste(symbol, var, "Forecast"),
           x = "Time", y = var) +
      theme_minimal()
  )
}
```


AAPL volume Forecast
9e+08
6e+08
3e+08
0e+00
−3e+08
volume
1
2
3
4
Time
series
Actual


AAPL rsi14 Forecast
80
60
40
rsi14
1
2
3
4
Time
series
Actual


AAPL ma20 Forecast
250
225
200
175
150
125
ma20
1
2
3
4
Time
series
Actual




```
exog_perf
```

```
## # A tibble: 4 x 4
##   variable  MAPE     MSE   AIC
##   <chr>    <dbl>   <dbl> <dbl>
## 1 volume    38.8 1.31e16 2476.
## 2 rsi14     18.6 1.33e 2  839.
## 3 ma20      17.7 1.88e 3  404.
## 4 ma50      18.0 1.93e 3  241.
```

Build model 1

```
train_ts <- ts(train_data$adjusted, frequency = 52)
test_ts <- ts(test_data$adjusted, frequency = 52,
              start = c(1, length(train_data$adjusted) + 1))

xreg_train <- as.matrix(train_data %>% select(volume, rsi14, ma20, ma50))
xreg_test <- cbind(
  volume = exog_forecasts[["volume"]],
  rsi14 = exog_forecasts[["rsi14"]],
  ma20 = exog_forecasts[["ma20"]],
  ma50 = exog_forecasts[["ma50"]]
)

# Fit the model / forecast
model_arima <- auto.arima(train_ts, xreg = xreg_train)
```

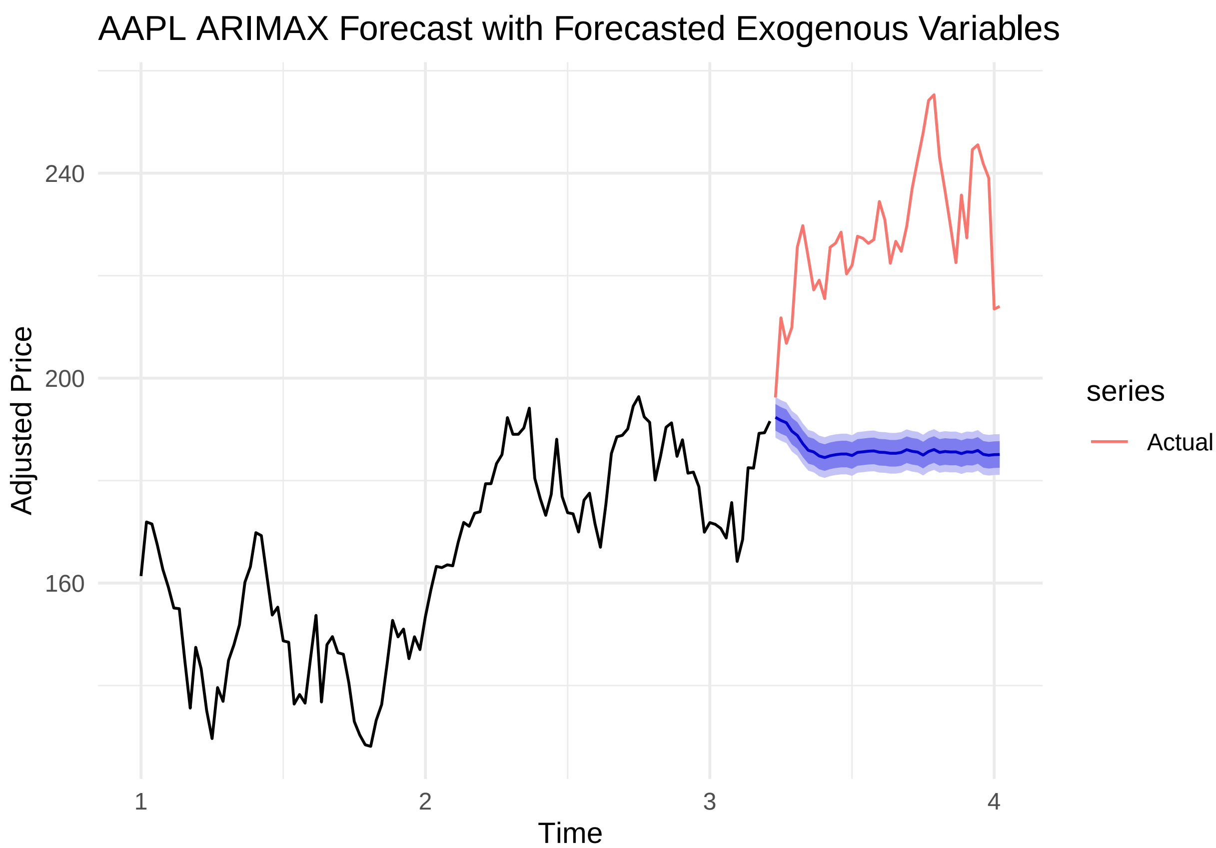
```r
summary(model_arima)
```

```
## Series: train_ts
## Regression with ARIMA(0,0,0) errors
##
## Coefficients:
##       intercept  volume   rsi14    ma20    ma50
##        -26.2523   0e+00  0.5710  0.6539  0.3378
## s.e.     2.8685   1e-04  0.0175  0.0360  0.0378
##
## sigma^2 = 4.096:  log likelihood = -243.83
## AIC=499.66   AICc=500.43   BIC=516.18
##
## Training set error measures:
##                        ME     RMSE     MAE         MPE      MAPE       MASE
## Training set 2.174506e-15 1.979877 1.56389 -0.02217531 0.9646576 0.05515573
##                   ACF1
## Training set 0.637196
```

```r
h <- nrow(test_data)
final_forecast_1 <- forecast(model_arima, xreg = xreg_test, h = h)
```

Performance

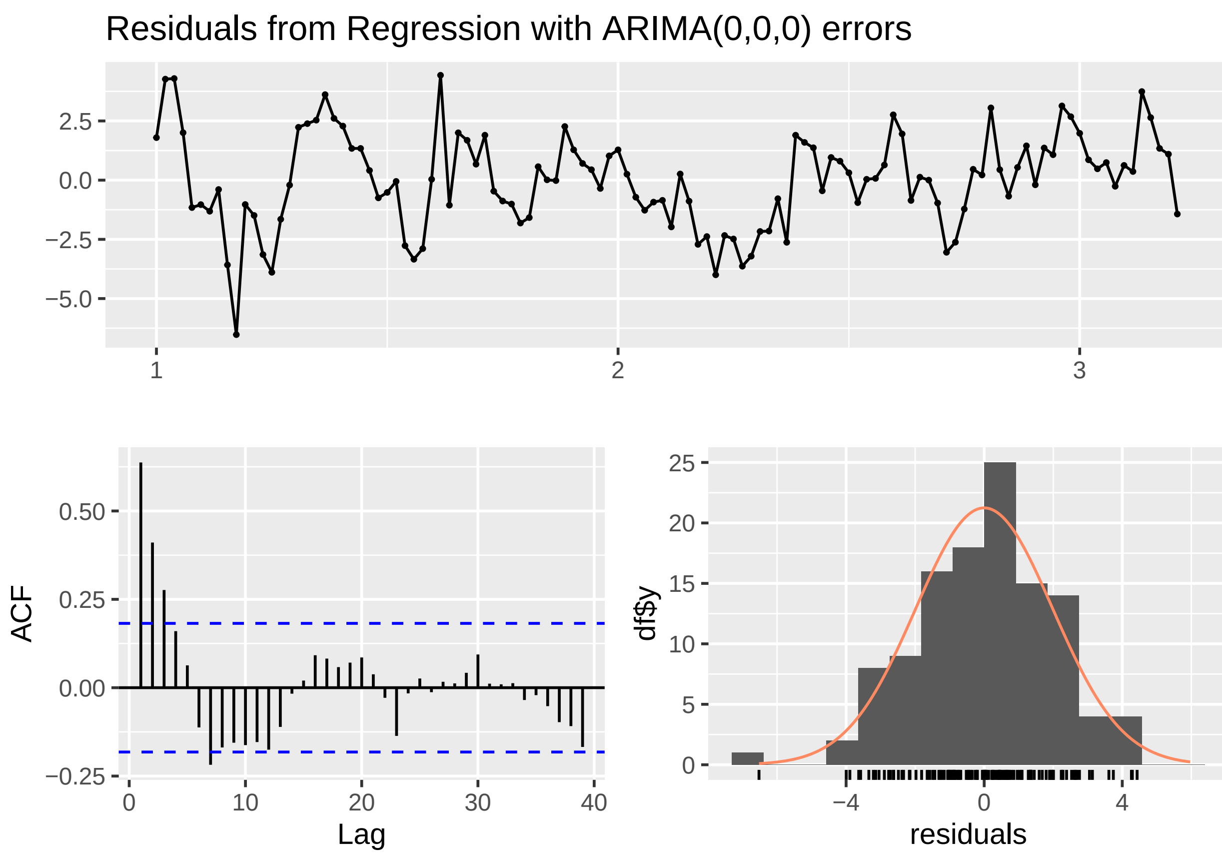
```r
autoplot(final_forecast_1) +
  autolayer(test_ts, series = "Actual") +
  labs(title = paste(symbol, "ARIMAX Forecast with Forecasted Exogenous Variables"),
       x = "Time", y = "Adjusted Price") +
  theme_minimal()
```




```r
# Calculate error metrics: MAPE and MSE for the final forecast
mape_final <- mean(abs(final_forecast_1$mean - test_ts) / abs(test_ts)) * 100
mse_final <- mean((final_forecast_1$mean - test_ts)^2)
cat("Final ARIMAX Forecast -> MAPE:", mape_final, "\nMSE:", mse_final)
```

```
## Final ARIMAX Forecast -> MAPE: 18.19925
## MSE: 1956.217
```

```r
checkresiduals(final_forecast_1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,0,0) errors
## Q* = 115.34, df = 23, p-value = 2.842e-14
##
## Model df: 0.   Total lags used: 23
```