Comparative TIME SERIES FORECASTING of Major Technology Stocks

Analyzing Apple, Microsoft, Amazon & Tesla Stocks

Comparative Time Series Forecasting of Major Technology Stocks

Seif H. Kungulio

January 22, 2026

Contents

Business Understanding

Problem Statement

Historical stock prices of major technology companies exhibit distinct trends, volatility patterns, and market dynamics. Understanding these behaviors is essential for forecasting price movements and assessing financial risk.

The objective of this project is to perform a comparative time series analysis of Apple, Microsoft, Tesla, and Amazon stock prices using historical market data. The project aims to identify trends, seasonality, and volatility across each stock and develop forecasting models to predict short-term price movements. Model performance will be evaluated to assess forecasting accuracy and differences in predictability across companies.

Business Objectives

- Compare trend, seasonality, and volatility across AAPL, MSFT, TSLA, and AMZN.
- Forecast short-term price movements using multiple models.
- Evaluate models using time-series appropriate validation and accuracy metrics.
- Rank stocks by forecastability (which stock is easier/harder to predict).

Success Criteria

- Clean daily dataset per stock with aligned calendars.
- Baseline model + at least one statistical forecasting model per stock.
- Residual diagnostics + accuracy metrics (RMSE/MAE/MAPE).
- Clear comparative summary and recommendation.

Data Understanding

Data Source

Historical stock price data for Apple (AAPL), Microsoft (MSFT), Tesla (TSLA), and Amazon (AMZN) will be sourced from Yahoo Finance using the `quantmod` package in R. The dataset will include daily adjusted closing prices, volume, and other relevant financial metrics from January 1, 2016, to current date.

Stock Tickers

- Apple Inc. (AAPL)
- Microsoft Corporation (MSFT)
- Tesla, Inc. (TSLA)
- Amazon.com, Inc. (AMZN)

```r
# Define stock tickers and date range
tickers <- c("AAPL", "MSFT", "TSLA", "AMZN")
start_date <- as.Date("2016-01-01")
end_date <- as.Date("2026-01-20")
```

Pull Historical Prices

```r
prices_list <- map(
  tickers,
  ~ getSymbols(
    .x,
    src = "yahoo",
    from = start_date,
    to = end_date,
    auto.assign = FALSE
  )
)
names(prices_list) <- tickers

# Preview data
map(prices_list, head)
```

```
## $AAPL
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2016-01-04   25.6525   26.3425  25.5000    26.3375   270597600      23.75315
## 2016-01-05   26.4375   26.4625  25.6025    25.6775   223164000      23.15791
## 2016-01-06   25.1400   25.5925  24.9675    25.1750   273829600      22.70472
## 2016-01-07   24.6700   25.0325  24.1075    24.1125   324377600      21.74648
## 2016-01-08   24.6375   24.7775  24.1900    24.2400   283192000      21.86147
## 2016-01-11   24.7425   24.7650  24.3350    24.6325   198957600      22.21545
##
## $MSFT
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
## 2016-01-04     54.32     54.80    53.39      54.80    53778000      47.98346
```

```
## 2016-01-05      54.93      55.39     54.54      55.05    34079700      48.20237
## 2016-01-06      54.32      54.40     53.64      54.05    39518900      47.32674
## 2016-01-07      52.70      53.49     52.07      52.17    56564900      45.68061
## 2016-01-08      52.37      53.28     52.15      52.33    48754000      45.82071
## 2016-01-11      52.51      52.85     51.46      52.30    36943800      45.79443
##
## $TSLA
##            TSLA.Open TSLA.High TSLA.Low TSLA.Close TSLA.Volume TSLA.Adjusted
## 2016-01-04  15.38133  15.42533 14.60000   14.89400   102406500      14.89400
## 2016-01-05  15.09067  15.12600 14.66667   14.89533    47802000      14.89533
## 2016-01-06  14.66667  14.67000 14.39867   14.60267    56686500      14.60267
## 2016-01-07  14.27933  14.56267 14.24467   14.37667    53314500      14.37667
## 2016-01-08  14.52400  14.69600 14.05133   14.06667    54421500      14.06667
## 2016-01-11  14.26733  14.29667 13.53333   13.85667    61371000      13.85667
##
## $AMZN
##            AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
## 2016-01-04   32.8145   32.8860  31.3755    31.8495   186290000       31.8495
## 2016-01-05   32.3430   32.3455  31.3825    31.6895   116452000       31.6895
## 2016-01-06   31.1000   31.9895  31.0155    31.6325   106584000       31.6325
## 2016-01-07   31.0900   31.5000  30.2605    30.3970   141498000       30.3970
## 2016-01-08   30.9830   31.2070  30.3000    30.3525   110258000       30.3525
## 2016-01-11   30.6240   30.9925  29.9285    30.8870    97832000       30.8870
```

```r
map(prices_list, tail)
```

```
## $AAPL
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2026-01-09    259.08    260.21   256.22     259.37    39997000        259.37
## 2026-01-12    259.16    261.30   256.80     260.25    45263800        260.25
## 2026-01-13    258.72    261.81   258.39     261.05    45730800        261.05
## 2026-01-14    259.49    261.82   256.71     259.96    40019400        259.96
## 2026-01-15    260.65    261.04   257.05     258.21    39388600        258.21
## 2026-01-16    257.90    258.90   254.93     255.53    72142800        255.53
##
## $MSFT
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
## 2026-01-09    474.06    479.82   472.20     479.28    18491000        479.28
## 2026-01-12    476.67    480.99   475.68     477.18    23519900        477.18
## 2026-01-13    474.68    475.78   465.95     470.67    28545800        470.67
## 2026-01-14    466.46    468.20   457.17     459.38    28184300        459.38
## 2026-01-15    464.12    464.25   455.90     456.66    23225800        456.66
## 2026-01-16    457.83    463.19   456.48     459.86    34246700        459.86
##
## $TSLA
##            TSLA.Open TSLA.High TSLA.Low TSLA.Close TSLA.Volume TSLA.Adjusted
## 2026-01-09    435.95    449.05   430.39     445.01    67331500        445.01
## 2026-01-12    441.23    454.30   438.00     448.96    61649600        448.96
## 2026-01-13    450.20    451.81   443.95     447.20    53719200        447.20
## 2026-01-14    442.81    443.91   434.22     439.20    57259500        439.20
## 2026-01-15    441.13    445.36   437.65     438.57    49465800        438.57
## 2026-01-16    439.50    447.25   435.26     437.50    60220600        437.50
##
## $AMZN
```

```
##            AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
## 2026-01-09    244.57    247.86   242.24     247.38    34560000        247.38
## 2026-01-12    246.73    248.94   245.96     246.47    35867800        246.47
## 2026-01-13    246.53    247.66   240.25     242.60    38371800        242.60
## 2026-01-14    241.15    241.28   236.22     236.65    41410600        236.65
## 2026-01-15    239.31    240.65   236.63     238.18    43003600        238.18
## 2026-01-16    239.09    239.57   236.41     239.12    45888300        239.12
```

Extract Adjusted Close Prices

```
close_list <- map(prices_list, ~ Cl(.x))
names(close_list) <- tickers

# Combine into a single xts with aligned dates
close_xts <- do.call(merge, close_list)
colnames(close_xts) <- tickers

head(close_xts)
```

```
##               AAPL  MSFT     TSLA    AMZN
## 2016-01-04 26.3375 54.80 14.89400 31.8495
## 2016-01-05 25.6775 55.05 14.89533 31.6895
## 2016-01-06 25.1750 54.05 14.60267 31.6325
## 2016-01-07 24.1125 52.17 14.37667 30.3970
## 2016-01-08 24.2400 52.33 14.06667 30.3525
## 2016-01-11 24.6325 52.30 13.85667 30.8870
```

```
tail(close_xts)
```

```
##              AAPL   MSFT   TSLA   AMZN
## 2026-01-09 259.37 479.28 445.01 247.38
## 2026-01-12 260.25 477.18 448.96 246.47
## 2026-01-13 261.05 470.67 447.20 242.60
## 2026-01-14 259.96 459.38 439.20 236.65
## 2026-01-15 258.21 456.66 438.57 238.18
## 2026-01-16 255.53 459.86 437.50 239.12
```

Missing Dates / Alignment

Markets close on weekends/holidays; we keep the market calendar as-is.

```
close_xts_aligned <- na.omit(close_xts)
dim(close_xts); dim(close_xts_aligned)
```

```
## [1] 2525    4
```

```
## [1] 2525    4
```

Data Preparation

Convert to Tidy Data

```r
close_df <- close_xts_aligned %>%
  fortify.zoo() %>%
  as_tibble() %>%
  rename(date = Index) %>%
  pivot_longer(-date, names_to = "ticker", values_to = "close")

glimpse(close_df)
```

```
## Rows: 10,100
## Columns: 3
## $ date   <date> 2016-01-04, 2016-01-04, 2016-01-04, 2016-01-04, 2016-01-05, 20~
## $ ticker <chr> "AAPL", "MSFT", "TSLA", "AMZN", "AAPL", "MSFT", "TSLA", "AMZN",~
## $ close  <dbl> 26.33750, 54.80000, 14.89400, 31.84950, 25.67750, 55.05000, 14.~
```

Create Returns (Risk/Volatility Lens)

Returns analysis is essential for risk assessment.

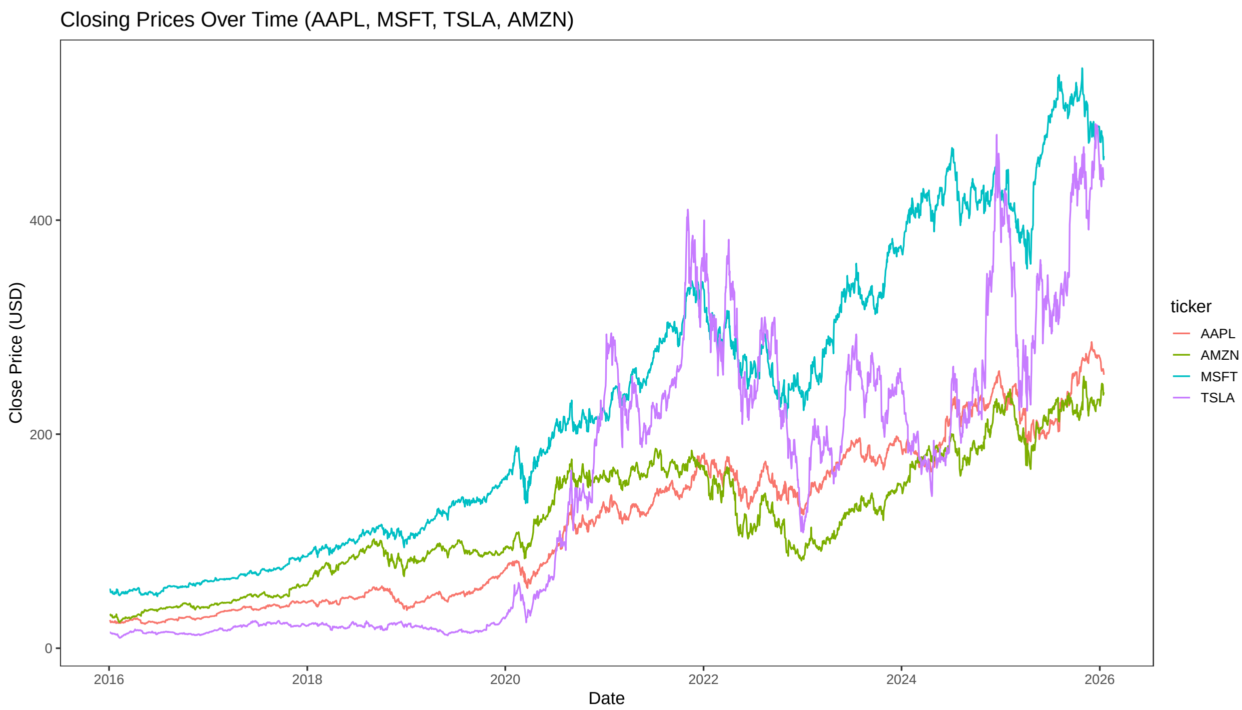
```r
returns_xts <- na.omit(Return.calculate(close_xts_aligned, method = "log"))
colnames(returns_xts) <- tickers

returns_df <- returns_xts %>%
  fortify.zoo() %>%
  as_tibble() %>%
  rename(date = Index) %>%
  pivot_longer(-date, names_to = "ticker", values_to = "log_return")

summary(returns_df$log_return)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.2365179 -0.0091692  0.0010848  0.0009702  0.0119456  0.2044906
```

Exploratory Data Analysis (EDA)

Price Trends

```
close_df %>%
  ggplot(aes(date, close, color = ticker)) +
  geom_line() +
  labs(
    title = "Closing Prices Over Time (AAPL, MSFT, TSLA, AMZN)",
    x = "Date", y = "Close Price (USD)"
  )
```

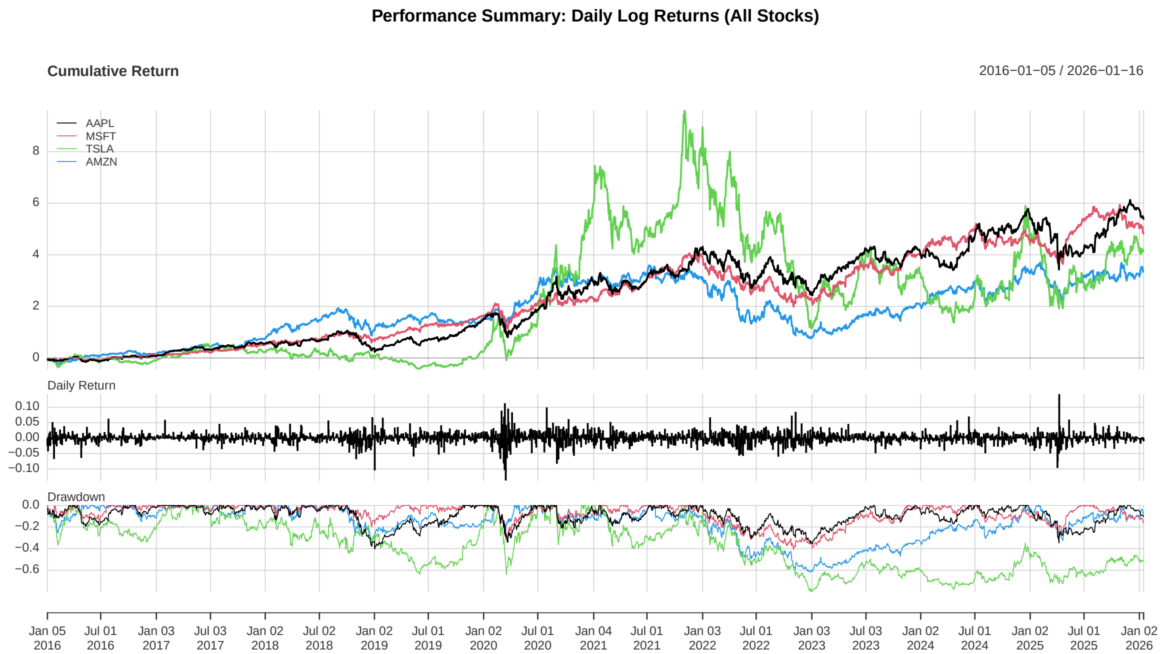


Normalize Prices (Indexed to 100)

```
close_indexed <- close_df %>%
  group_by(ticker) %>%
  arrange(date) %>%
  mutate(index_100 = 100 * close / first(close)) %>%
  ungroup()

close_indexed %>%
  ggplot(aes(date, index_100, color = ticker)) +
  geom_line() +
  labs(
    title = "Indexed Closing Prices (Start = 100)",
    x = "Date", y = "Index"
  )
```

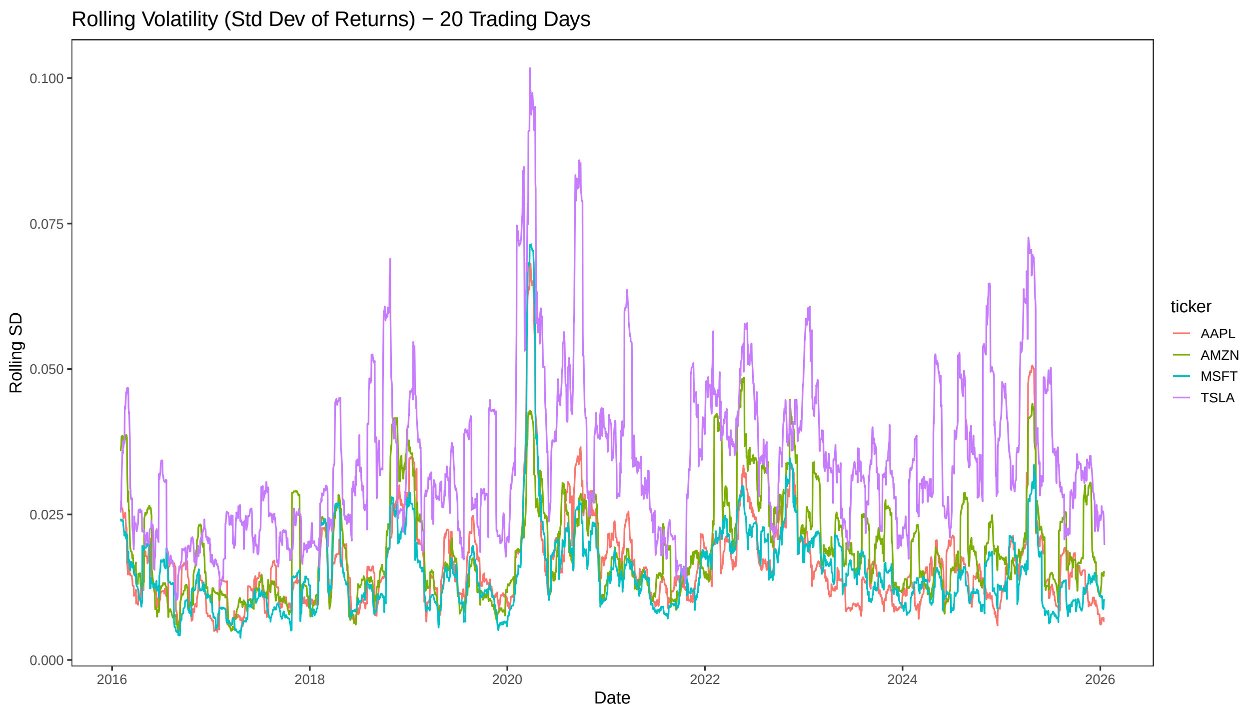
Return & Drawdown Summary

```r
charts.PerformanceSummary(
  returns_xts,
  main = "Performance Summary: Daily Log Returns (All Stocks)")
```



Volatility Comparison (Rolling Std. Dev.)

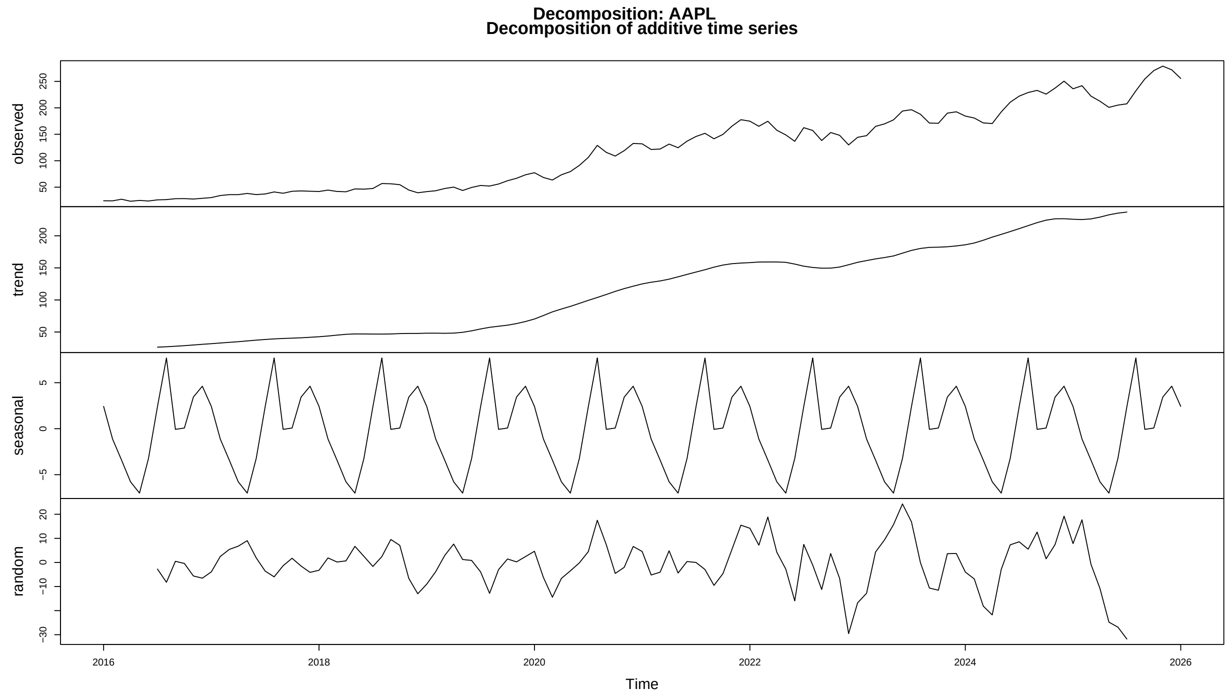
```
roll_n <- 20 # ~1 trading month
roll_vol <- rollapply(returns_xts,
                      width = roll_n,
                      FUN = sd,
                      by.column = TRUE,
                      align = "right",
                      fill = NA) %>%
  na.omit()

roll_vol_df <- roll_vol %>%
  fortify.zoo() %>%
  as_tibble() %>%
  rename(date = Index) %>%
  pivot_longer(-date,
               names_to = "ticker",
               values_to = "roll_sd")

roll_vol_df %>%
  ggplot(aes(date, roll_sd, color = ticker)) +
  geom_line() +
  labs(
    title = glue::glue("Rolling Volatility (Std Dev of Returns) - {roll_n} Trading Days"),
    x = "Date",
    y = "Rolling SD"
  )
```

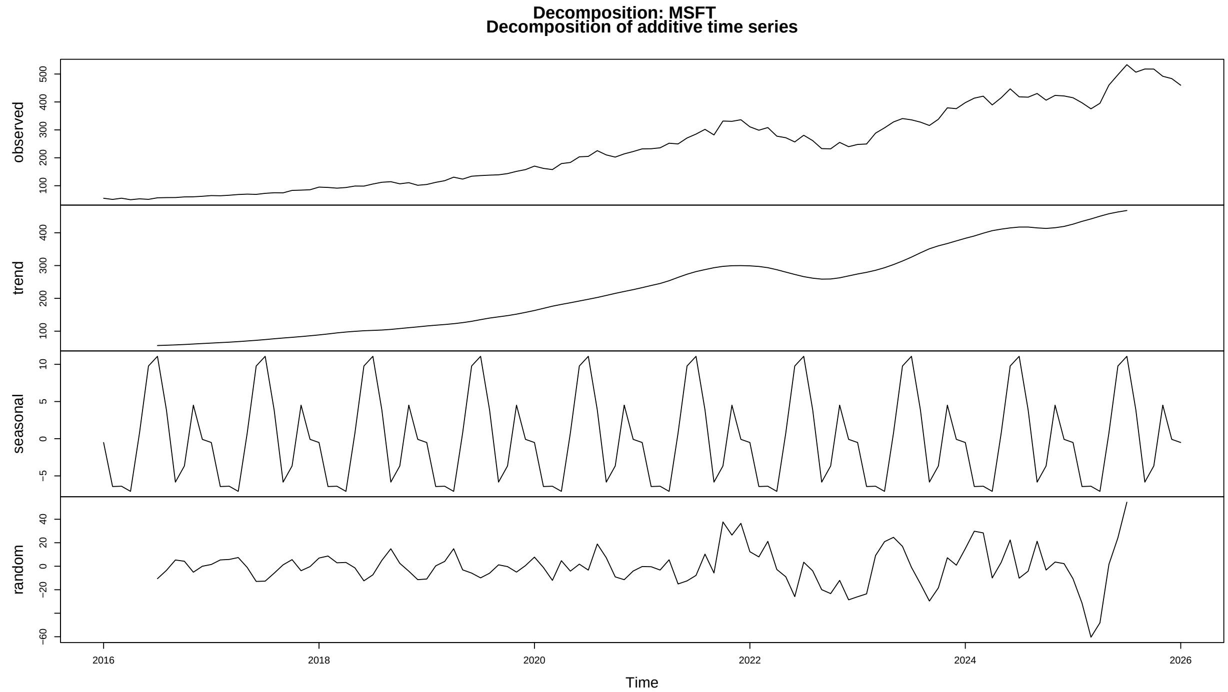


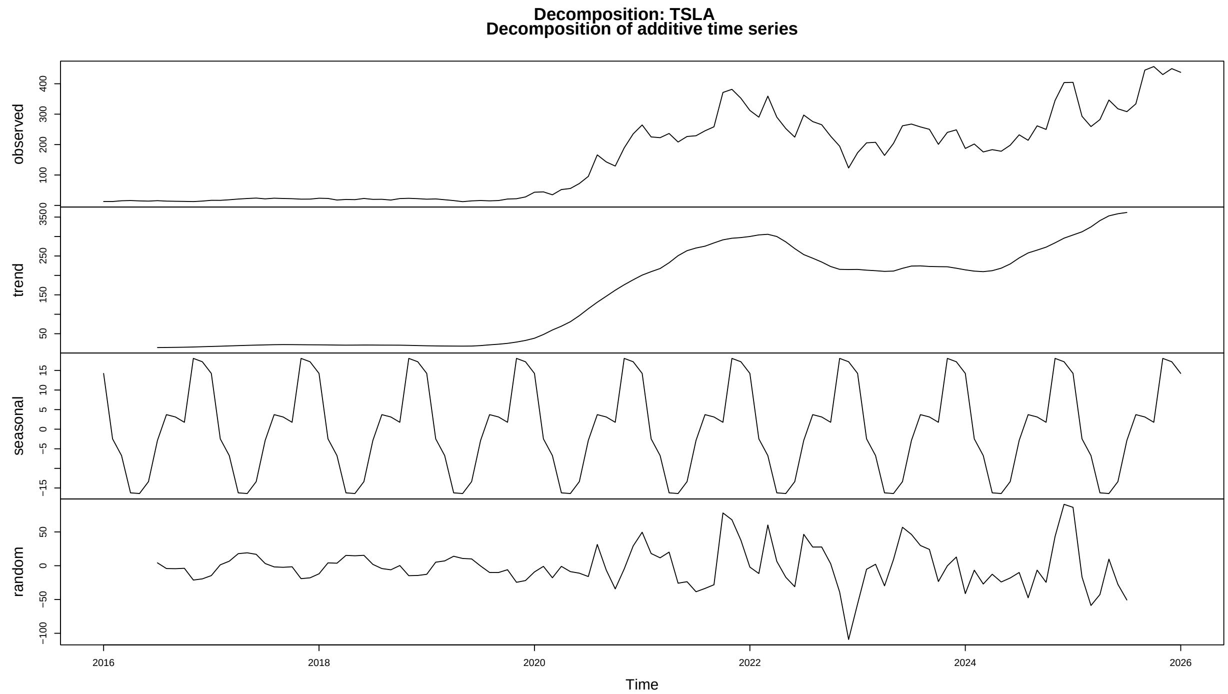
Decomposition (Monthly Aggregation)

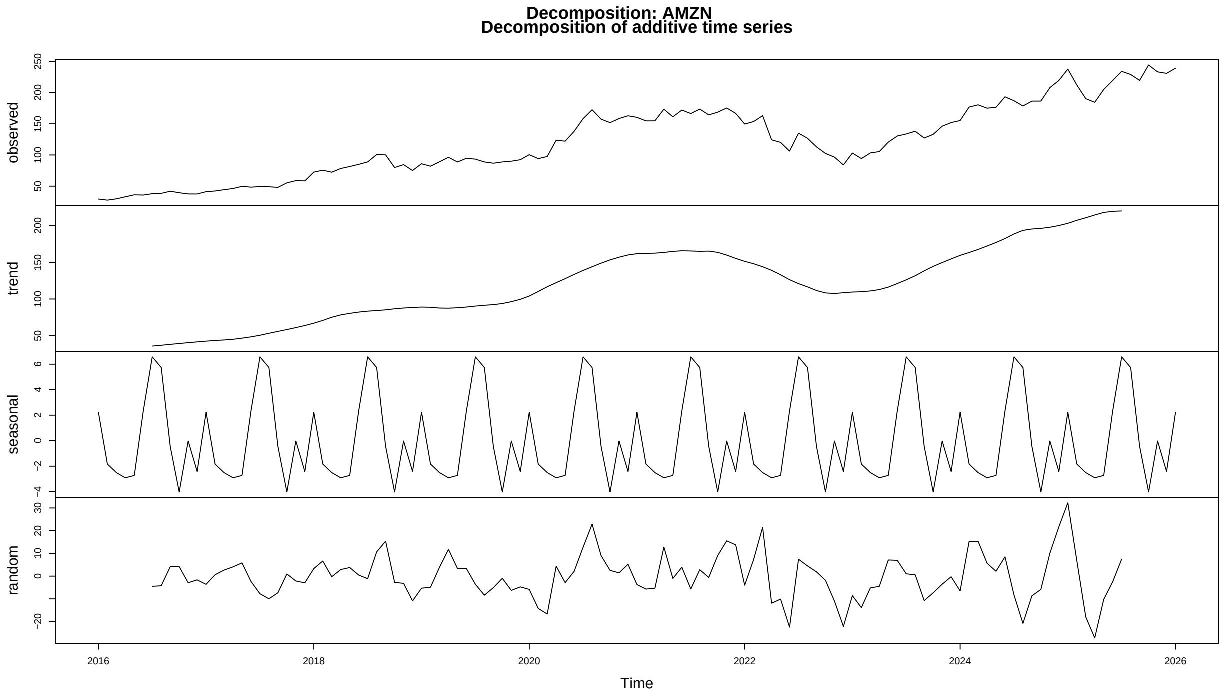
Classic decomposition is easiest on regular seasonal frequency. We convert daily close to monthly close and decompose per stock.

```
monthly_close_list <- map(close_list, ~ Cl(to.monthly(.x)) )
names(monthly_close_list) <- tickers

decomp_plots <- function(x_xts, ticker_name){
  ts_obj <- ts(
    as.numeric(x_xts),
    frequency = 12,
    start = c(year(start(x_xts)), month(start(x_xts)))
    )
  dc <- decompose(ts_obj)
  plot(dc)
  title(main = paste("Decomposition:", ticker_name),
        outer = TRUE, line = -1)
  invisible(dc)
}

# Decompose each ticker (plots)
decomps <- imap(monthly_close_list, ~ decomp_plots(.x, .y))
```




Decomposition: MSFT
Decomposition of additive time series
observed
trend
seasonal
random
Time
2016
2018
2020
2022
2024
2026


Decomposition: TSLA
Decomposition of additive time series
observed
trend
seasonal
random
Time
2016
2018
2020
2022
2024
2026



Decomposition: AMZN
Decomposition of additive time series
observed
trend
seasonal
random
Time
2016
2018
2020
2022
2024
2026

Modeling

Forecasting Design

We'll forecast short-term movement using:

- Naive baseline (last value persists) - strong baseline in finance.
- ARIMA (auto.arima) - standard statistical model for time series.

We'll do a rolling-origin evaluation (time-series CV) AND a simple holdout for interpretability.

Helper Functions

```r
make_train_test <- function(x, h = 60){
  # x is a numeric vector or ts; h is forecast horizon
  n <- length(x)
  list(train = x[1:(n-h)], test = x[(n-h+1):n], h = h)
}

rmse <- function(actual, pred){
  sqrt(mean((actual - pred)^2, na.rm = TRUE))
}

mae <- function(actual, pred){
  mean(abs(actual - pred), na.rm = TRUE)
}

mape <- function(actual, pred){
  mean(abs((actual - pred) / actual), na.rm = TRUE) * 100
}
```

Build Models per Stock

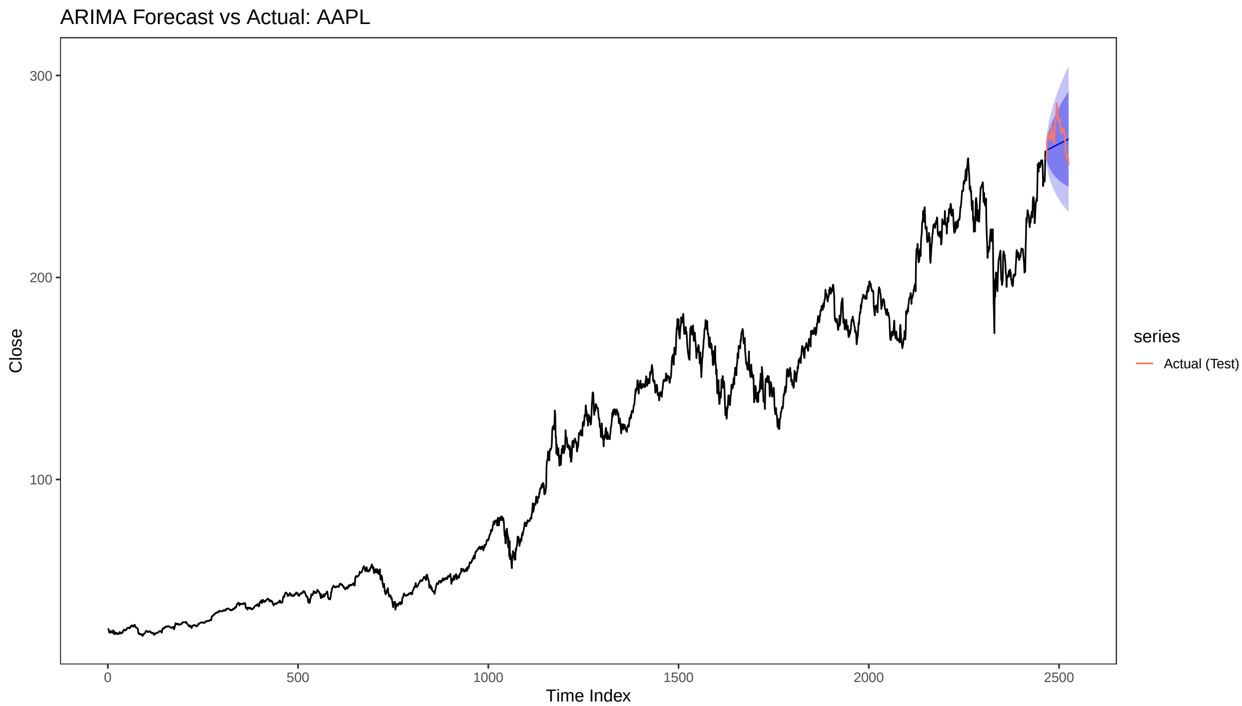
```r
h <- 60  # forecast horizon ~ 3 months of trading days

results <- map_dfr(tickers, function(tk){
  x <- as.numeric(close_xts_aligned[, tk])
  spl <- make_train_test(x, h = h)
  train <- spl$train
  test  <- spl$test

  # --- Naive baseline ---
  fc_naive <- naive(train, h = h)
  pred_naive <- as.numeric(fc_naive$mean)

  # --- ARIMA ---
  fit_arima <- auto.arima(train)
  fc_arima <- forecast(fit_arima, h = h)
```

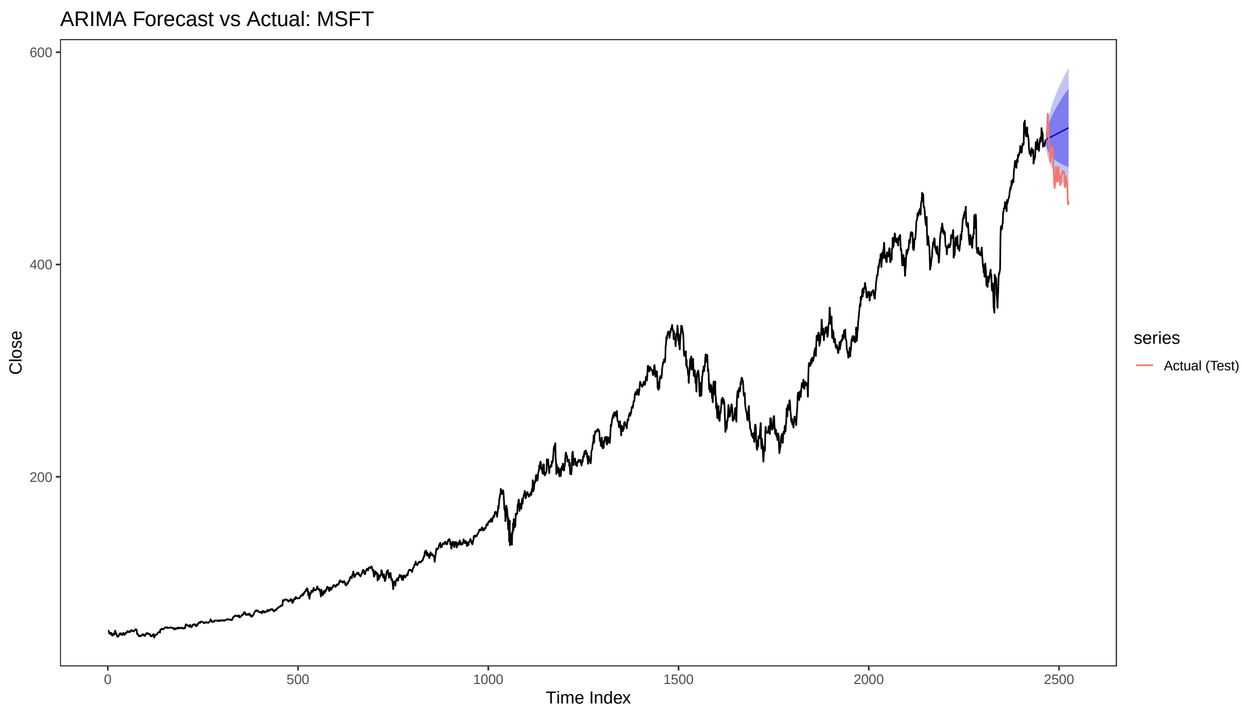
```
  pred_arima <- as.numeric(fc_arima$mean)

  tibble(
    ticker = tk,
    model  = c("Naive", "ARIMA"),
    RMSE   = c(rmse(test, pred_naive), rmse(test, pred_arima)),
    MAE    = c(mae(test, pred_naive),  mae(test, pred_arima)),
    MAPE   = c(mape(test, pred_naive), mape(test, pred_arima))
  )
})

results %>% arrange(ticker, RMSE)
```

```
## # A tibble: 8 x 5
##   ticker model  RMSE   MAE  MAPE
##   <chr>  <chr> <dbl> <dbl> <dbl>
## 1 AAPL   ARIMA  8.93  7.90  2.90
## 2 AAPL   Naive 10.5   9.19  3.35
## 3 AMZN   ARIMA 12.7   9.75  4.07
## 4 AMZN   Naive 14.3  11.5   4.81
## 5 MSFT   Naive 32.4  28.8   5.97
## 6 MSFT   ARIMA 38.7  34.3   7.11
## 7 TSLA   Naive 23.2  17.9   4.04
## 8 TSLA   ARIMA 23.2  17.9   4.04
```

Visualize Forecasts vs Actuals per Stock

```
plot_forecast <- function(tk){
  x <- as.numeric(close_xts_aligned[, tk])
  spl <- make_train_test(x, h = h)
  train <- spl$train
  test  <- spl$test

  fc_naive <- naive(train, h = h)
  fit_arima <- auto.arima(train)
  fc_arima <- forecast(fit_arima, h = h)

  # Plot ARIMA by default; overlay actual test
  autoplot(fc_arima) +
    autolayer(ts(test, start = length(train) + 1),
              series = "Actual (Test)") +
    labs(title = paste("ARIMA Forecast vs Actual:", tk),
         x = "Time Index", y = "Close")
}

walk(tickers, ~ print(plot_forecast(.x)))
```


ARIMA Forecast vs Actual: AAPL
300
200
100
Close
0
500
1000
1500
2000
2500
Time Index
series
Actual (Test)


ARIMA Forecast vs Actual: MSFT
600
400
200
Close
0
500
1000
1500
2000
2500
Time Index
series
Actual (Test)


ARIMA Forecast vs Actual: TSLA
Close
400
200
0
0
500
1000
1500
2000
2500
Time Index
series
Actual (Test)


ARIMA Forecast vs Actual: AMZN
Close
200
100
0
500
1000
1500
2000
2500
Time Index
series
Actual (Test)

Evaluation

Accuracy Leaderboard (Model Comparison)

Forecast Accuracy by Stock and Model (Lower RMSE/MAE/MAPE is Better)

```r
results %>%
  group_by(ticker) %>%
  arrange(RMSE, .by_group = TRUE) %>%
  mutate(rank = row_number()) %>%
  ungroup() %>%
  arrange(ticker, rank) %>%
  knitr::kable(digits = 3,
               caption = "Forecast Accuracy by Stock and Model")
```

Table 1: Forecast Accuracy by Stock and Model

ticker	model	RMSE	MAE	MAPE	rank
AAPL	ARIMA	8.931	7.903	2.899	1
AAPL	Naive	10.548	9.187	3.350	2
AMZN	ARIMA	12.652	9.753	4.066	1
AMZN	Naive	14.275	11.513	4.811	2
MSFT	Naive	32.446	28.789	5.969	1
MSFT	ARIMA	38.654	34.274	7.110	2
TSLA	Naive	23.215	17.863	4.040	1
TSLA	ARIMA	23.215	17.863	4.040	2

Best Model per Stock

```r
best_by_stock <- results %>%
  group_by(ticker) %>%
  slice_min(RMSE, n = 1, with_ties = FALSE) %>%
  ungroup()

best_by_stock %>% knitr::kable(digits = 3,
                               caption = "Best Model Per Stock (by RMSE)")
```

Table 2: Best Model Per Stock (by RMSE)

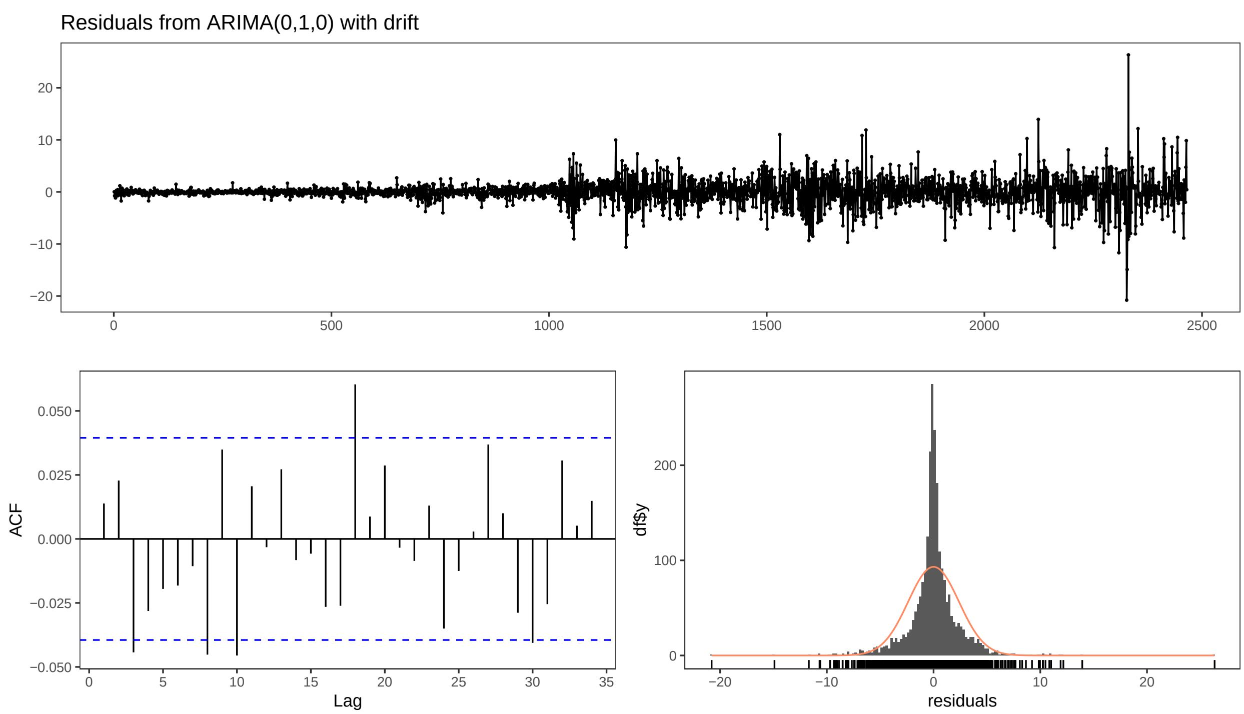
ticker	model	RMSE	MAE	MAPE
AAPL	ARIMA	8.931	7.903	2.899
AMZN	ARIMA	12.652	9.753	4.066
MSFT	Naive	32.446	28.789	5.969
TSLA	Naive	23.215	17.863	4.040

Residual Diagnostics for Best Models

We check residuals to see if assumptions look reasonable (ACF, histogram, Ljung-Box)

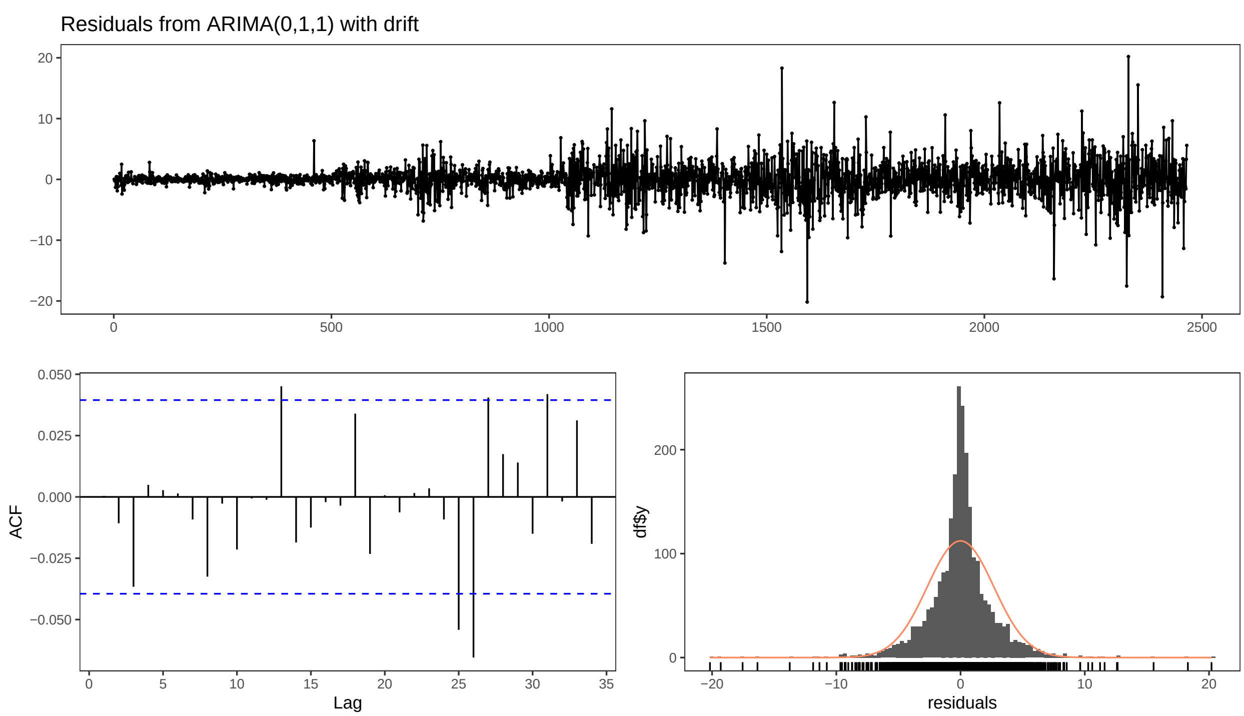
```
check_best_residuals <- function(tk){
  x <- as.numeric(close_xts_aligned[, tk])
  spl <- make_train_test(x, h = h)
  train <- spl$train

  fit_arima <- auto.arima(train)
  fc_arima <- forecast(fit_arima, h = h)

  cat("\n\n### Residual checks for", tk, "\n")
  print(fit_arima)
  checkresiduals(fc_arima)
}

walk(best_by_stock$ticker, check_best_residuals)
```

```
##
##
## ### Residual checks for AAPL
## Series: train
## ARIMA(0,1,0) with drift
##
## Coefficients:
##        drift
##        0.096
## s.e.   0.048
##
## sigma^2 = 5.669:  log likelihood = -5633.38
## AIC=11270.77   AICc=11270.77   BIC=11282.39
```



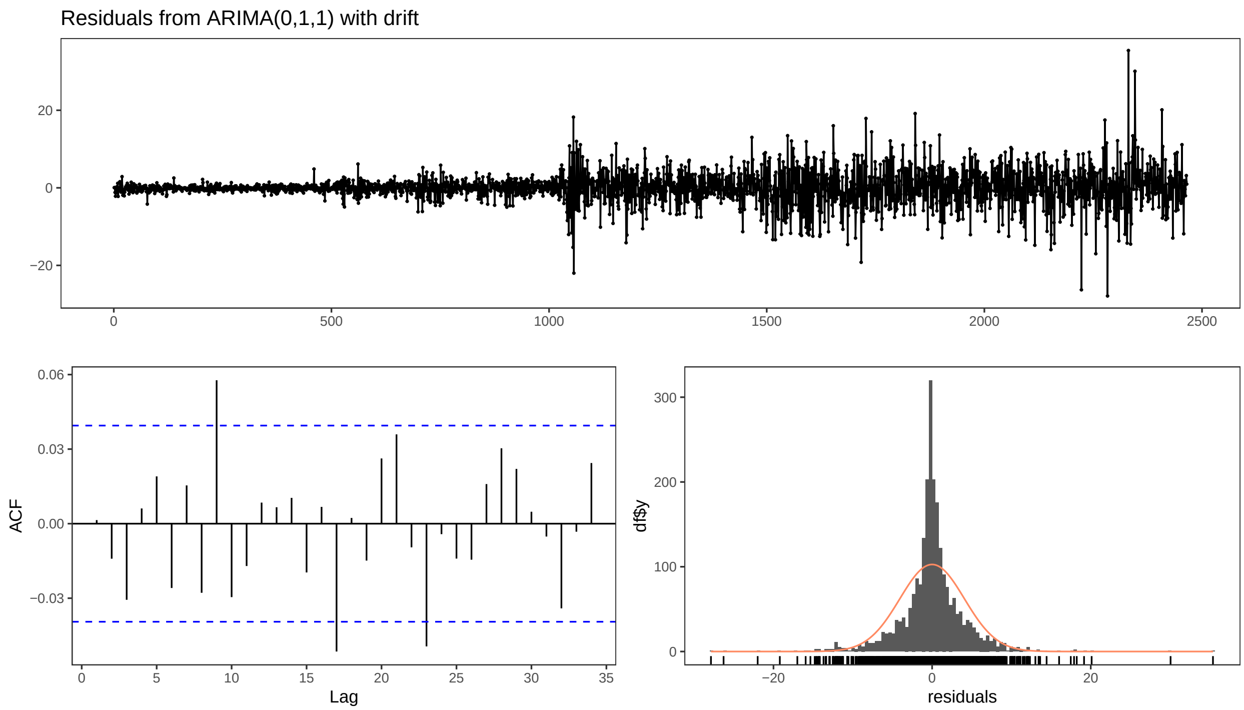

```
##
```

```
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0) with drift
## Q* = 23.804, df = 10, p-value = 0.008138
##
## Model df: 0.   Total lags used: 10
##
##
##
## ### Residual checks for AMZN
## Series: train
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.0352  0.0772
## s.e.   0.0204  0.0521
##
## sigma^2 = 7.188:  log likelihood = -5925.34
## AIC=11856.68   AICc=11856.69   BIC=11874.11
```

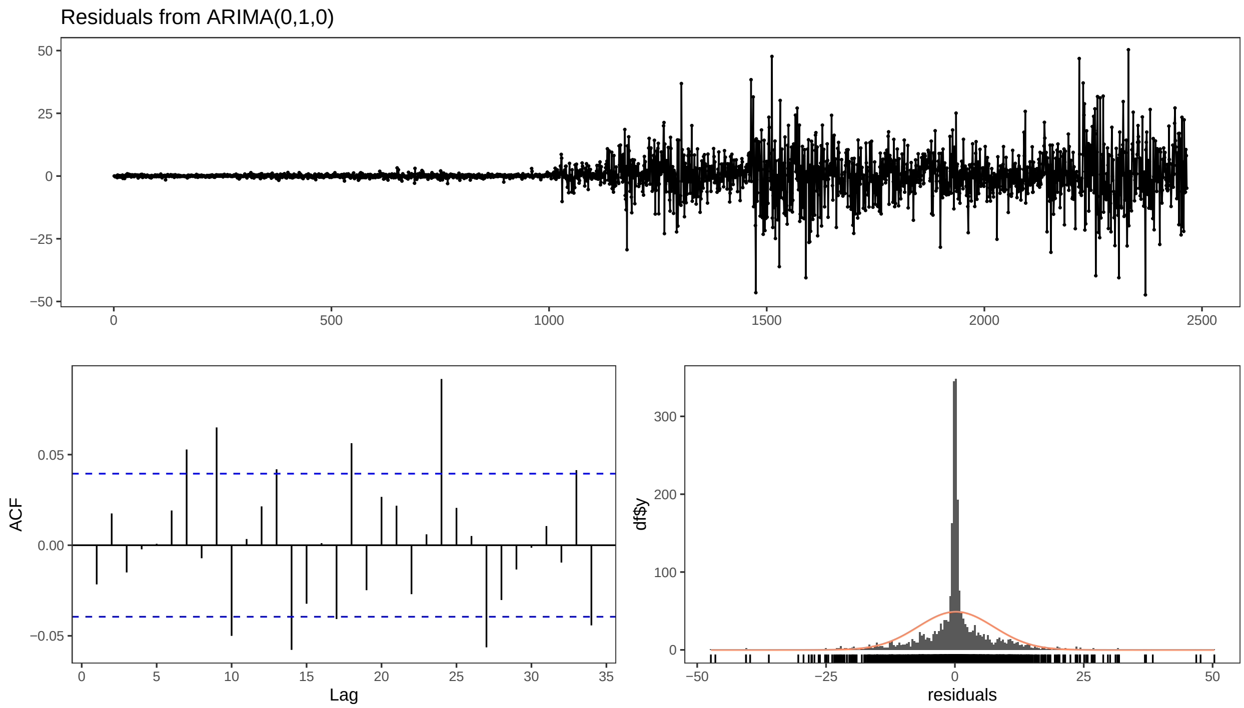



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 7.663, df = 9, p-value = 0.5684
##
## Model df: 1.   Total lags used: 10
##
##
```

```
##
## ### Residual checks for MSFT
## Series: train
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.0879  0.1878
## s.e.   0.0205  0.0748
##
## sigma^2 = 16.57:  log likelihood = -6954
## AIC=13914   AICc=13914.01   BIC=13931.43
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 18.38, df = 9, p-value = 0.03101
##
## Model df: 1.   Total lags used: 10
##
##
##
## ### Residual checks for TSLA
## Series: train
## ARIMA(0,1,0)
##
## sigma^2 = 53.87:  log likelihood = -8407.68
## AIC=16817.36   AICc=16817.37   BIC=16823.17
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 27.087, df = 10, p-value = 0.002523
##
## Model df: 0.   Total lags used: 10
```

Comparative Interpretation

Comparative Forecastability (Lower RMSE = More Predictable)

```r
# Stock that is easiest/hardest to forecast (based on best RMSE)
forecastability_tbl <- best_by_stock %>%
  arrange(RMSE) %>%
  mutate(forecastability_rank = row_number())

forecastability_tbl %>%
  knitr::kable(digits = 3,
               caption = "Comparative Forecastability of Stocks")
```

Table 3: Comparative Forecastability of Stocks

ticker	model	RMSE	MAE	MAPE	forecastability_rank
AAPL	ARIMA	8.931	7.903	2.899	1
AMZN	ARIMA	12.652	9.753	4.066	2
TSLA	Naive	23.215	17.863	4.040	3
MSFT	Naive	32.446	28.789	5.969	4

Deployment

Reproducible Forecast Function

A simple deployment-ready function that can be reused in a script, Shiny app, or scheduled job.

```
forecast_stock <- function(ticker, from = start_date, to = end_date, h = 20){
  x <- getSymbols(ticker, src = "yahoo", from = from, to = to, auto.assign = FALSE)
  close <- as.numeric(Cl(x))

  fit <- auto.arima(close)
  fc  <- forecast(fit, h = h)

  list(
    ticker = ticker,
    model  = fit,
    forecast = fc
  )
}

# Example:
demo <- forecast_stock("AAPL", from = start_date, to = end_date, h = 20)
autoplot(demo$forecast) + labs(title = "Deployment Demo: AAPL 20-Day Forecast")
```

Conclusion

Key Findings

- Trend & growth differed meaningfully across AAPL/MSFT/TSLA/AMZN.
- Volatility (rolling SD of returns) highlighted different risk regimes—typically TSLA exhibits higher volatility.
- Naive baseline is a strong benchmark; any model must beat it to justify complexity.
- ARIMA often improves on naive for some tickers, but not always—evaluation determines the winner.

Next Improvement

- Explore additional models (e.g., ETS, Prophet, machine learning).
- Incorporate exogenous variables (e.g., market indices, macroeconomic indicators).
- Extend forecast horizon and evaluate longer-term predictability.
- Automate regular updates and forecasts via scheduled scripts or dashboards.

References

- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.
- Yahoo Finance. (n.d.). Retrieved from https://finance.yahoo.com/
- R Documentation for `quantmod`, `forecast`, and `tseries` packages.

Thank You!