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

February 2, 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("2026-01-31")
```

Here we define the study universe (AAPL, MSFT, TSLA, AMZN) and the historical window (**2016-01-01 to 2026-01-31**). The study period is fixed from January 1, 2016 to January 31, 2026 to ensure consistent temporal coverage across all securities. Using a common observation window allows differences in trends, volatility, and forecast accuracy to be attributed to underlying stock behavior rather than discrepancies in data availability or sampling periods.

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      48.20236
## 2016-01-06     54.32     54.40    53.64      54.05    39518900      47.32676
## 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      45.82070
## 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
```

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

```
## $AAPL
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
## 2026-01-23    247.32    249.41   244.68     248.04    41689000        248.04
## 2026-01-26    251.48    256.56   249.80     255.41    55969200        255.41
## 2026-01-27    259.17    261.95   258.21     258.27    49648300        258.27
## 2026-01-28    257.65    258.86   254.51     256.44    41288000        256.44
## 2026-01-29    258.00    259.65   254.41     258.28    67253000        258.28
## 2026-01-30    255.17    261.90   252.18     259.48    92352600        259.48
##
## $MSFT
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
## 2026-01-23    451.87    471.10   450.53     465.95    38000200        465.95
## 2026-01-26    465.31    474.25   462.00     470.28    29291200        470.28
## 2026-01-27    473.70    482.87   473.16     480.58    29213900        480.58
## 2026-01-28    483.21    483.74   478.00     481.63    36875400        481.63
## 2026-01-29    439.99    442.50   421.02     433.50   128855300        433.50
## 2026-01-30    439.17    439.60   426.45     430.29    58470400        430.29
##
## $TSLA
##            TSLA.Open TSLA.High TSLA.Low TSLA.Close TSLA.Volume TSLA.Adjusted
## 2026-01-23    447.43    452.43   444.04     449.06    56771400        449.06
## 2026-01-26    445.00    445.04   434.28     435.20    49397400        435.20
```

```
## 2026-01-27    437.41    437.52   430.69     430.90    37733100         430.90
## 2026-01-28    431.91    438.26   430.10     431.46    54857400         431.46
## 2026-01-29    437.80    440.23   414.62     416.56    81686100         416.56
## 2026-01-30    425.35    439.88   422.70     430.41    82483000         430.41
##
## $AMZN
##            AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
## 2026-01-23    234.96    240.45   234.57     239.16    33778500         239.16
## 2026-01-26    239.98    240.95   237.54     238.42    32825500         238.42
## 2026-01-27    239.69    244.88   238.08     244.68    38029200         244.68
## 2026-01-28    246.37    247.78   241.53     243.01    40882700         243.01
## 2026-01-29    242.82    243.00   236.74     241.73    47229600         241.73
## 2026-01-30    239.89    243.32   237.64     239.30    46535400         239.30
```

This chunk retrieves daily OHLCV data from Yahoo Finance into a named list of `xts` objects—one per ticker. Inspecting the `head()`/`tail()` snapshots is a quick sanity check that:

1. Dates are ordered correctly
2. Fields exist as expected (Open/High/Low/Close/Volume/Adjusted)
3. Price scales differ substantially across tickers—reinforcing why later we use **indexing (start=100)** and **returns** for apples-to-apples comparison.

Extract Adjusted Close Prices

```r
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
```

```r
tail(close_xts)
```

```
##              AAPL   MSFT   TSLA   AMZN
## 2026-01-23 248.04 465.95 449.06 239.16
## 2026-01-26 255.41 470.28 435.20 238.42
## 2026-01-27 258.27 480.58 430.90 244.68
## 2026-01-28 256.44 481.63 431.46 243.01
## 2026-01-29 258.28 433.50 416.56 241.73
## 2026-01-30 259.48 430.29 430.41 239.30
```

Adjusted closing prices are extracted for each ticker and merged into a single aligned `xts` object. This structure enables consistent downstream computation of returns, rolling statistics, and forecasting models. The merged dataset confirms that all series share the same trading calendar and exhibit strong upward trends over the analysis horizon, indicating non-stationarity in price levels.

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] 2534    4
```

```
## [1] 2534    4
```

Financial markets operate on an irregular calendar due to weekends and holidays. After merging all series, `na.omit()` is applied to retain only dates where all tickers are simultaneously observed. Since the dimensionality remains unchanged, the merged dataset contains a complete intersection of trading days with no missing values across assets.

Data Preparation

Convert to Tidy Data

```
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: 10,136
## 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.~
```

The wide `xts` object is reshaped into a long, tidy format with one observation per date and ticker. This structure facilitates grouped transformations, visualization, and comparative analysis using `ggplot2` and `tidyverse` workflows. Because `close_xts_aligned` contains **2,534** trading days and there are **4** tickers, the tidy dataset contains **10,136** rows (2,534 $\times$ 4).

Create Returns (Risk/Volatility Lens)

Returns analysis is essential for risk assessment.

```
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)
```

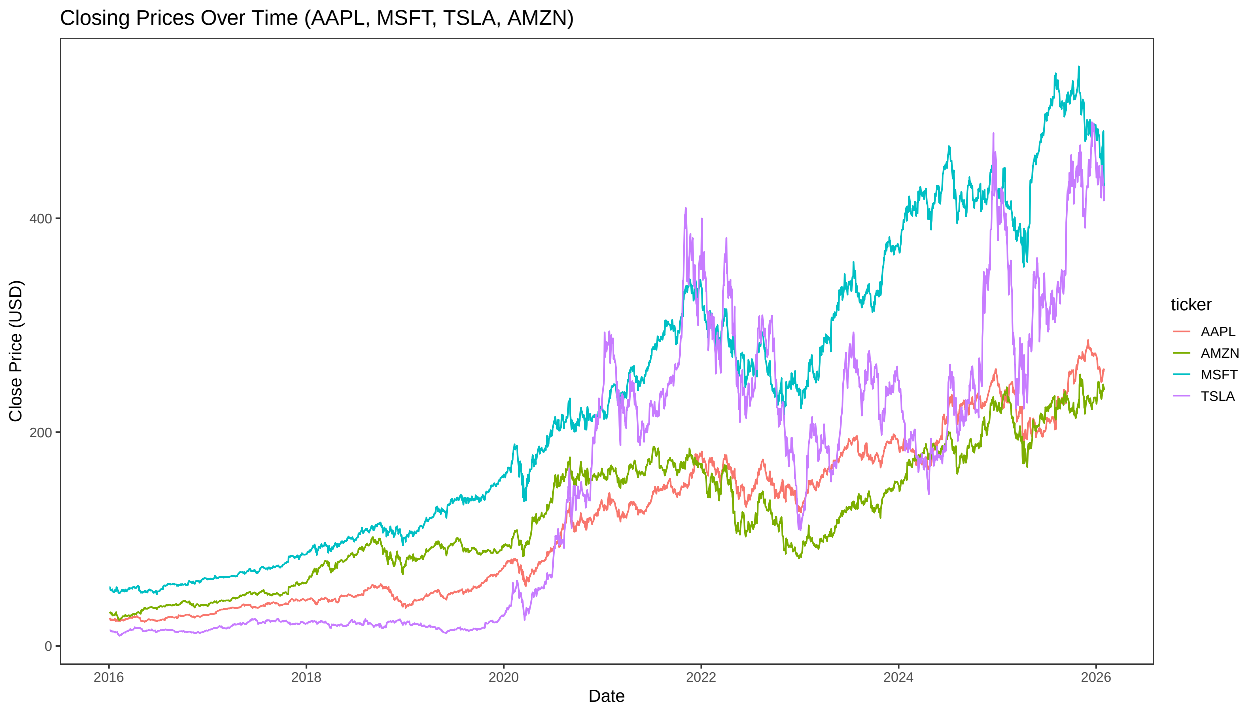
```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.2365179 -0.0091694  0.0010903  0.0009602  0.0119478  0.2044906
```

Daily log returns are computed to transform non-stationary price series into a scale-free representation suitable for volatility and risk analysis. The empirical return distribution exhibits heavy tails, with occasional extreme positive and negative values. Such behavior is consistent with financial time series and motivates the use of rolling volatility measures and robust forecasting benchmarks.

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)"
  )
```



The time series of closing prices illustrates long-term growth patterns and major structural shifts across all securities. Absolute price levels differ substantially between tickers, limiting direct comparability. This visualization primarily supports individual trend inspection rather than relative performance analysis.

Normalize Prices (Indexed to 100)

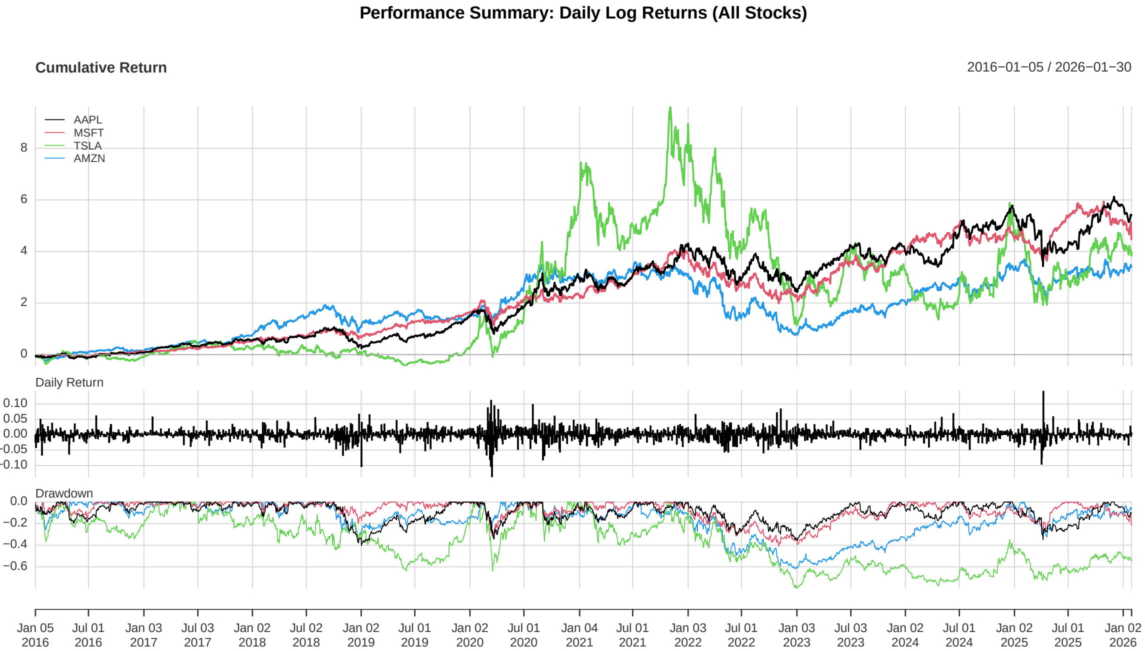
```
close_indexed <- close_df %>%
  group_by(ticker) %>%
  arrange(date) %>%
  mutate(index_100 = 100 * close / first(close)) %>%
  ungroup()

close_indexed %>%
  ggplot(aes(date, index_100, color = ticker)) +
  geom_line() +
```

```
  labs(
    title = "Indexed Closing Prices (Start = 100)",
    x = "Date", y = "Index"
  )
```

Price series are indexed to a common baseline value of **100** at the first observation for each ticker. This transformation standardizes scale and enables direct comparison of relative growth trajectories. The indexed series highlight substantial divergence in cumulative performance across stocks over the study period.

Return & Drawdown Summary

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

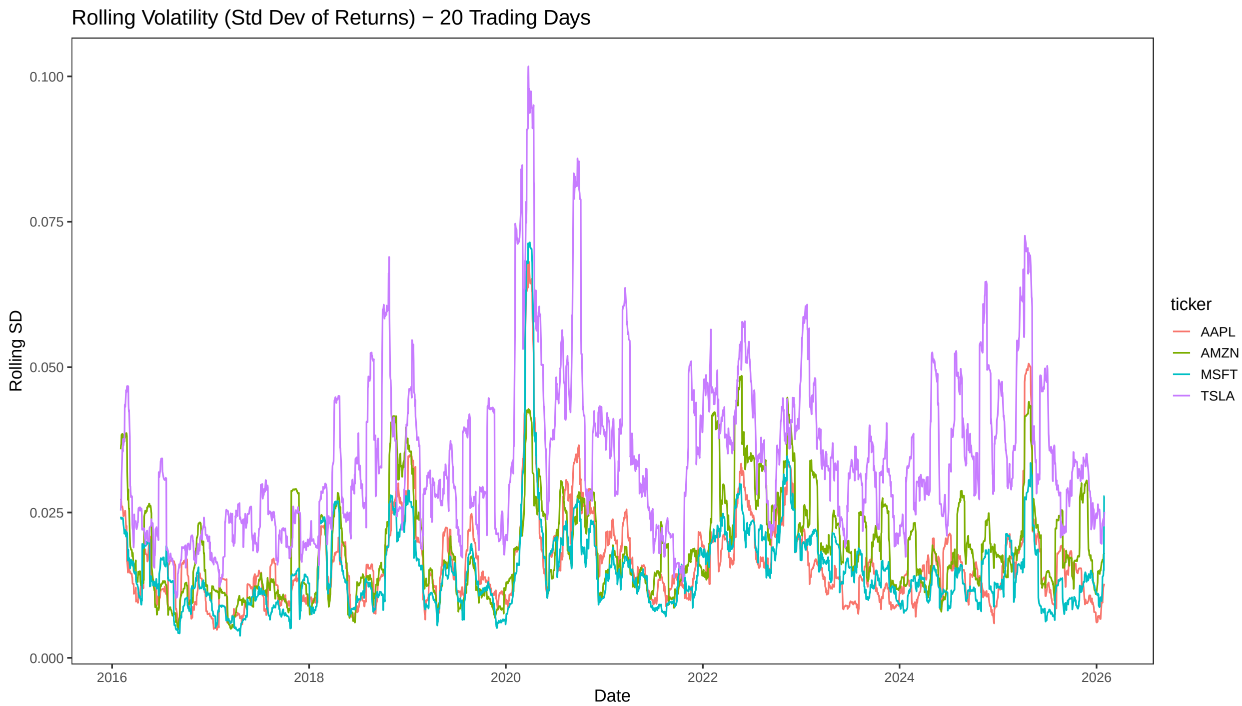


The performance summary visualizes cumulative returns, daily return distributions, and drawdowns in a unified framework. This representation captures both long-term compounding behavior and short-term risk characteristics. Differences in drawdown depth and return variability reflect heterogeneity in volatility regimes across securities.

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"
  )
```



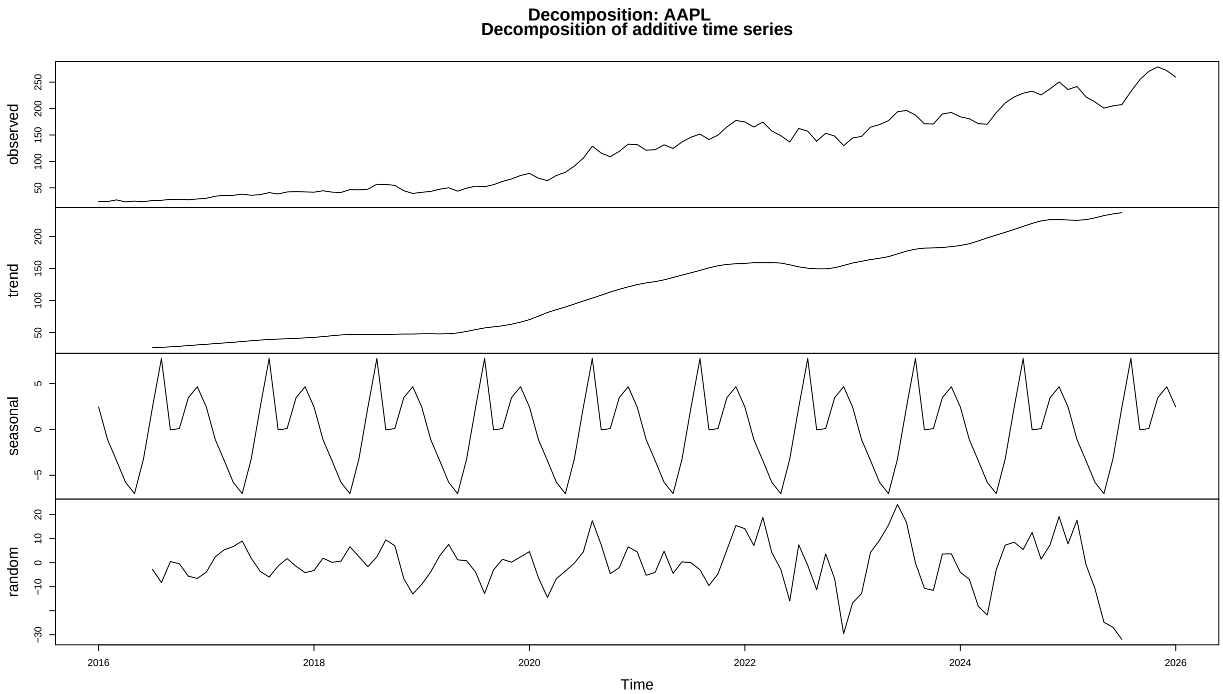
Rolling 20-day standard deviations of log returns are computed as a proxy for monthly realized volatility. The resulting series display volatility clustering, a common characteristic of equity markets. Higher and more frequent volatility spikes indicate increased uncertainty and reduced short-horizon predictability for certain securities.

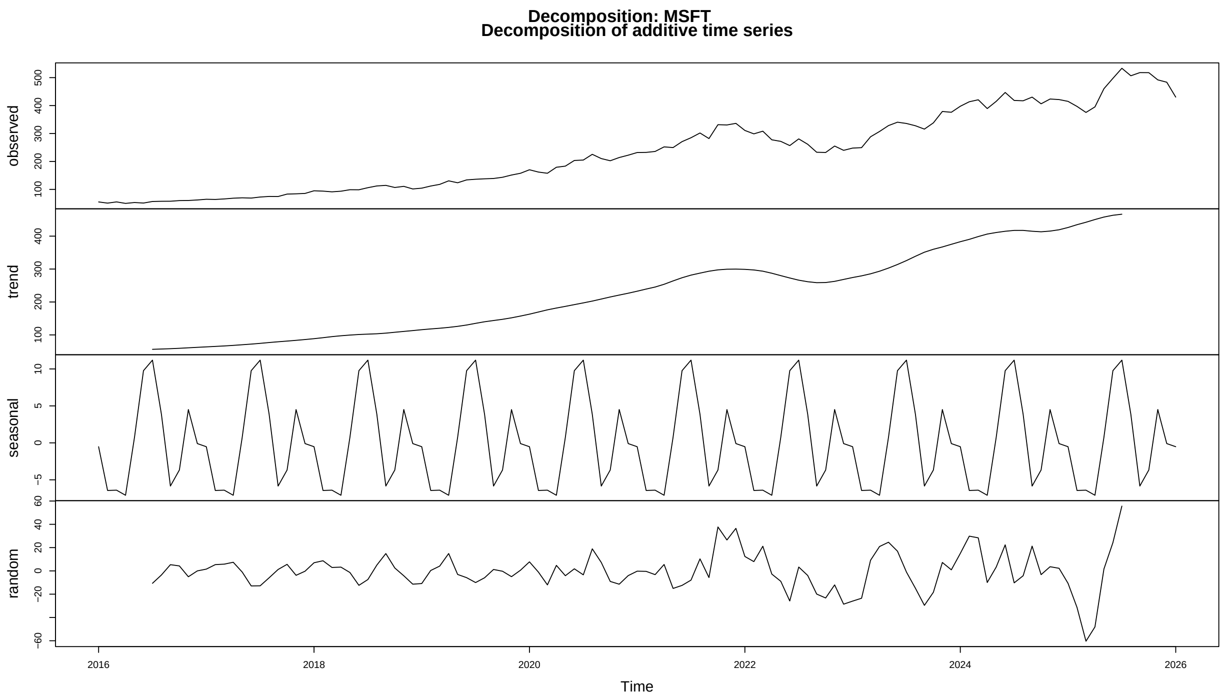
Decomposition (Monthly Aggregation)

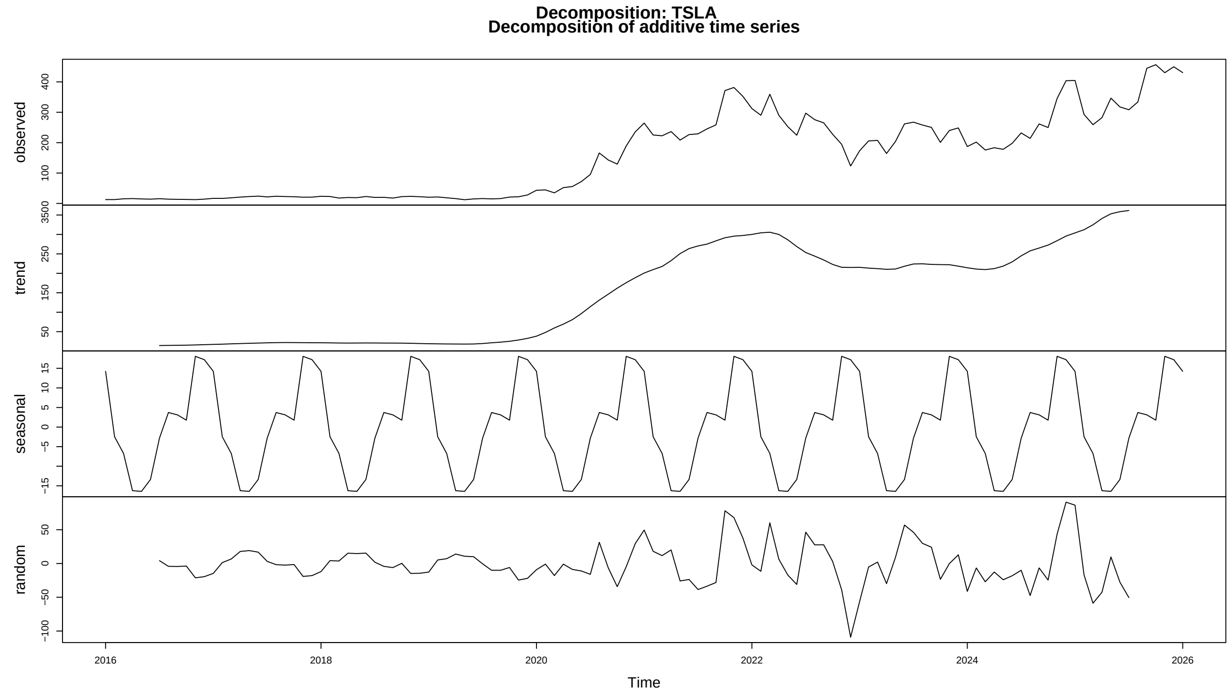
Classic decomposition is easiest on regular seasonal frequency. Converted daily close to monthly close and decompose per stock.

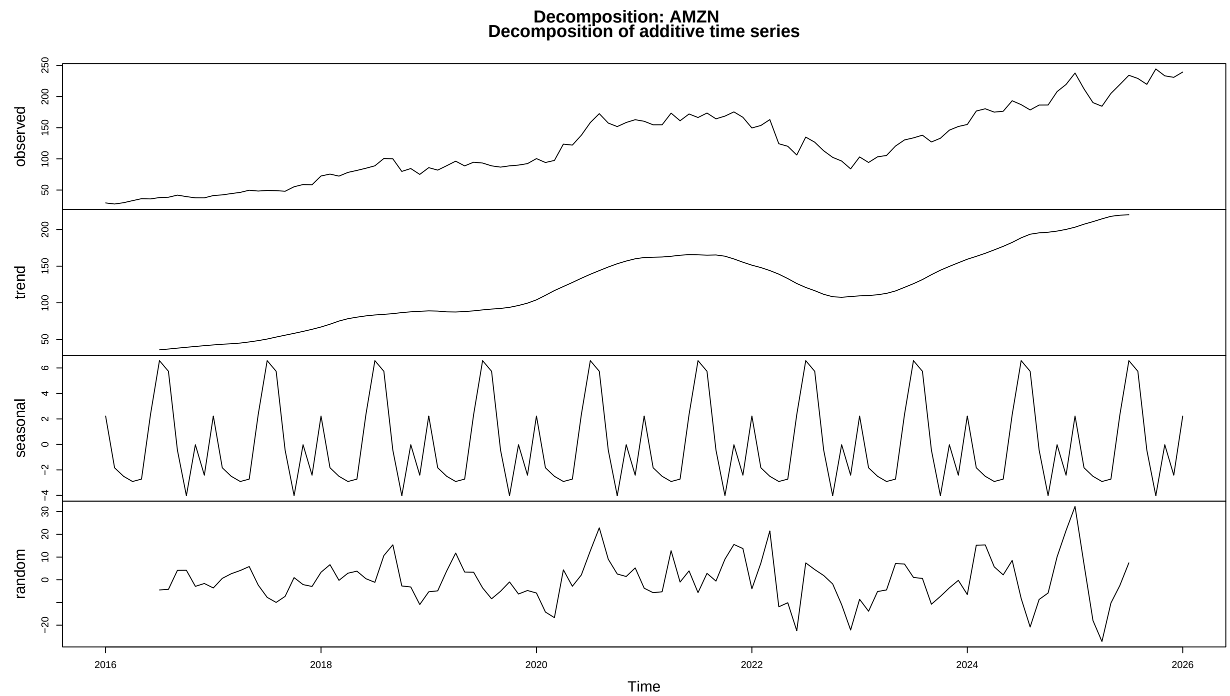
```
monthly_close_list <- map(close_list, ~ Cl(to.monthly(.x)) )
names(monthly_close_list) <- tickers

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)
}
```

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









Daily closing prices are aggregated to monthly frequency to facilitate classical additive decomposition. Across all securities, the trend component dominates the series, while seasonal effects are comparatively weak and unstable. The remainder component captures irregular fluctuations associated with firm-specific and macroeconomic events. These results suggest that forecasting models should prioritize trend handling rather than fixed seasonal structure.

Modeling

Forecasting Design

The focus of forecasting design is short-term movement using:

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

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

Helper Functions

```
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
}
```

Short-term forecasting is conducted using two benchmark approaches: a naive persistence model and an automatically selected ARIMA model. Model performance is evaluated using a fixed holdout horizon and time-series-appropriate error metrics. This design ensures consistent and comparable evaluation across all securities.

Build Models per Stock

```
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)
```

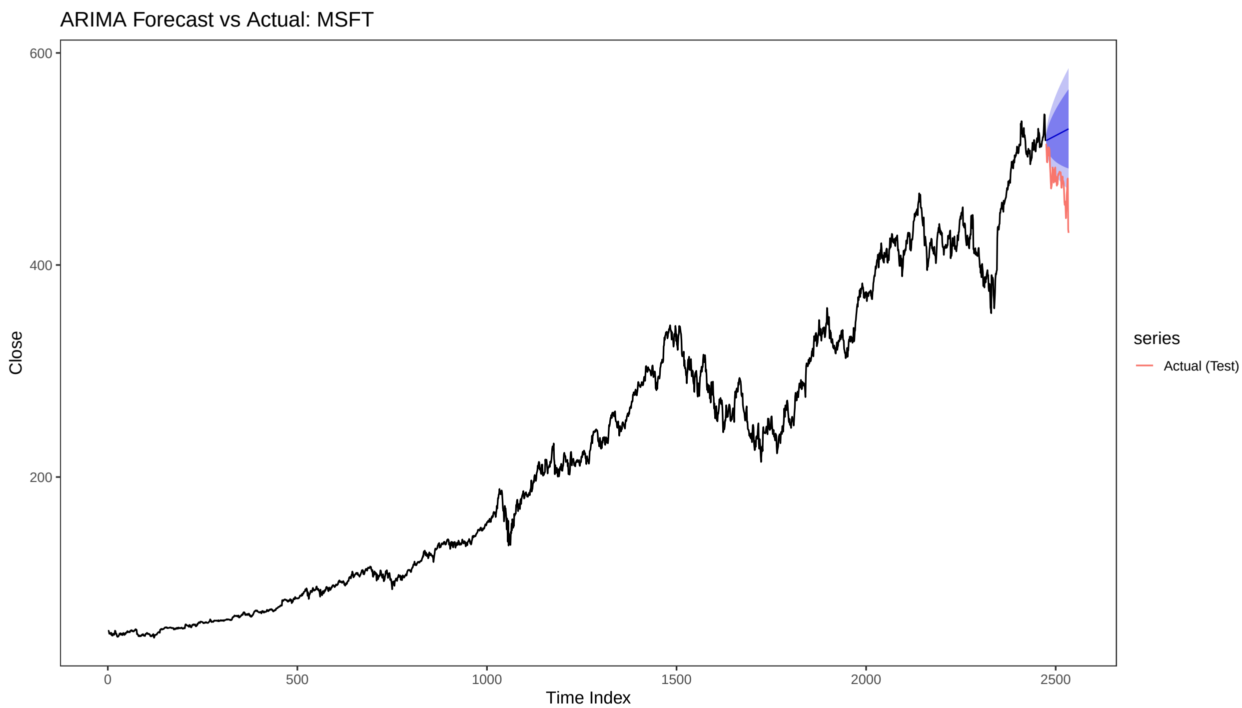
```
## # A tibble: 8 x 5
##   ticker model  RMSE   MAE  MAPE
##   <chr>  <chr> <dbl> <dbl> <dbl>
## 1 AAPL   Naive  9.38  7.59  2.86
## 2 AAPL   ARIMA 11.1   8.10  3.09
## 3 AMZN   Naive 21.3  19.7   8.52
## 4 AMZN   ARIMA 23.5  22.1   9.54
## 5 MSFT   Naive 39.8  35.9   7.60
## 6 MSFT   ARIMA 46.2  41.7   8.84
## 7 TSLA   Naive 35.5  30.8   7.18
## 8 TSLA   ARIMA 35.5  30.8   7.18
```

Forecast accuracy varies by security and model, but in this holdout window the **naive (random-walk/persistence)** baseline is extremely difficult to beat. In the reported results, ARIMA does **not** improve RMSE/MAE/MAPE for AAPL, AMZN, or MSFT, and TSLA is effectively a tie. This is common for short-horizon price-level forecasting, where prices often behave close to a random walk and incremental structure is limited.

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),
```

```r
               series = "Actual (Test)") +
    labs(title = paste("ARIMA Forecast vs Actual:", tk),
         x = "Time Index", y = "Close")
}

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



Forecast-versus-actual plots provide a visual assessment of model performance over the holdout period. Deviations between predicted and observed values highlight challenges associated with abrupt price movements and regime shifts. These plots complement numerical accuracy metrics by illustrating dynamic tracking behavior.

Evaluation

Accuracy Leaderboard (Model Comparison)

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

```
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	Naive	9.381	7.587	2.860	1
AAPL	ARIMA	11.092	8.097	3.088	2
AMZN	Naive	21.271	19.656	8.518	1
AMZN	ARIMA	23.500	22.056	9.540	2
MSFT	Naive	39.808	35.896	7.602	1
MSFT	ARIMA	46.206	41.737	8.837	2
TSLA	Naive	35.506	30.770	7.184	1
TSLA	ARIMA	35.506	30.770	7.184	2

This chunk ranks models within each ticker by RMSE and presents a concise leaderboard. In this run, the **naive** baseline ranks #1 for every ticker (with TSLA essentially tied), which reinforces an important finance lesson: any alternative model must **consistently** beat the random-walk baseline to justify extra complexity. The table is still portfolio-friendly because it demonstrates principled, time-ordered evaluation rather than selecting a model purely because it looks better in-sample.

Best Model per Stock

```
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	Naive	9.381	7.587	2.860

ticker	model	RMSE	MAE	MAPE
AMZN	Naive	21.271	19.656	8.518
MSFT	Naive	39.808	35.896	7.602
TSLA	Naive	35.506	30.770	7.184

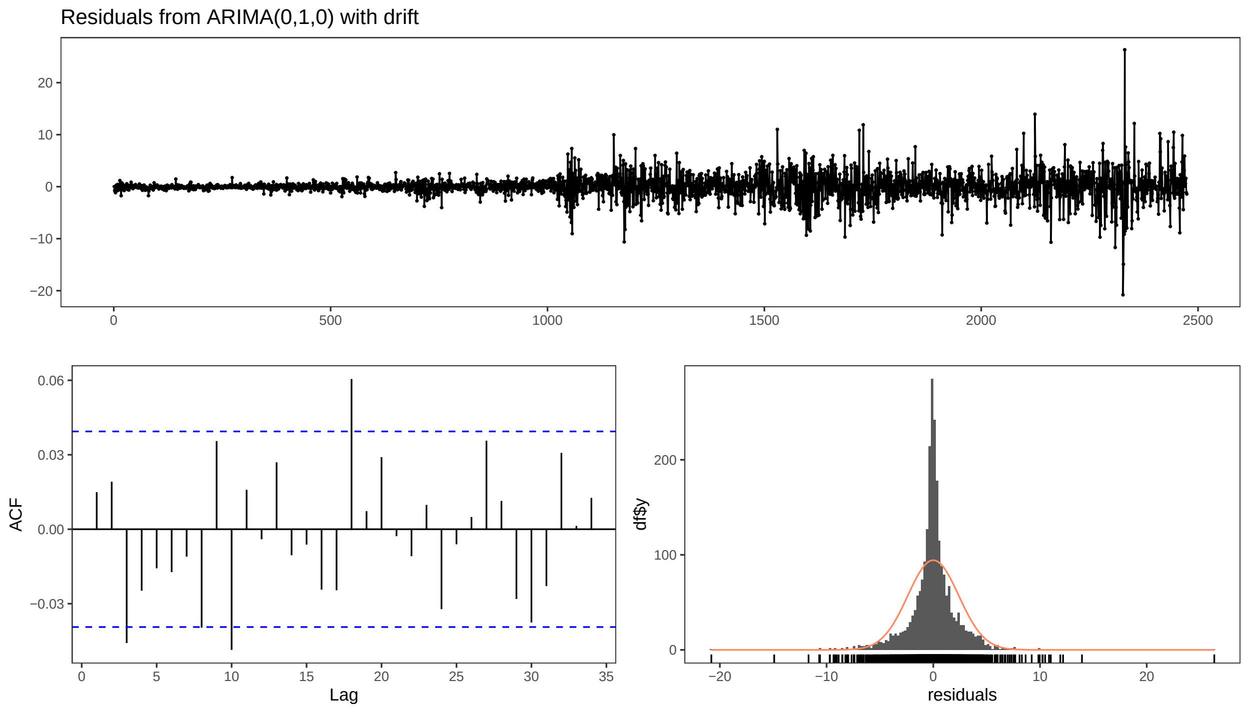
Selecting the best model per stock (by RMSE) yields **Naive** for **AAPL**, **AMZN**, **MSFT**, and **TSLA** (TSLA is effectively a tie, but `slice_min(..., with_ties = FALSE)` returns the first minimum, which is Naive here). For stakeholders, this is still a valuable outcome: under this design and horizon, **simplicity wins**, and it suggests that short-horizon **price levels** are hard to forecast beyond a random-walk benchmark.

Residual Diagnostics for ARIMA Models

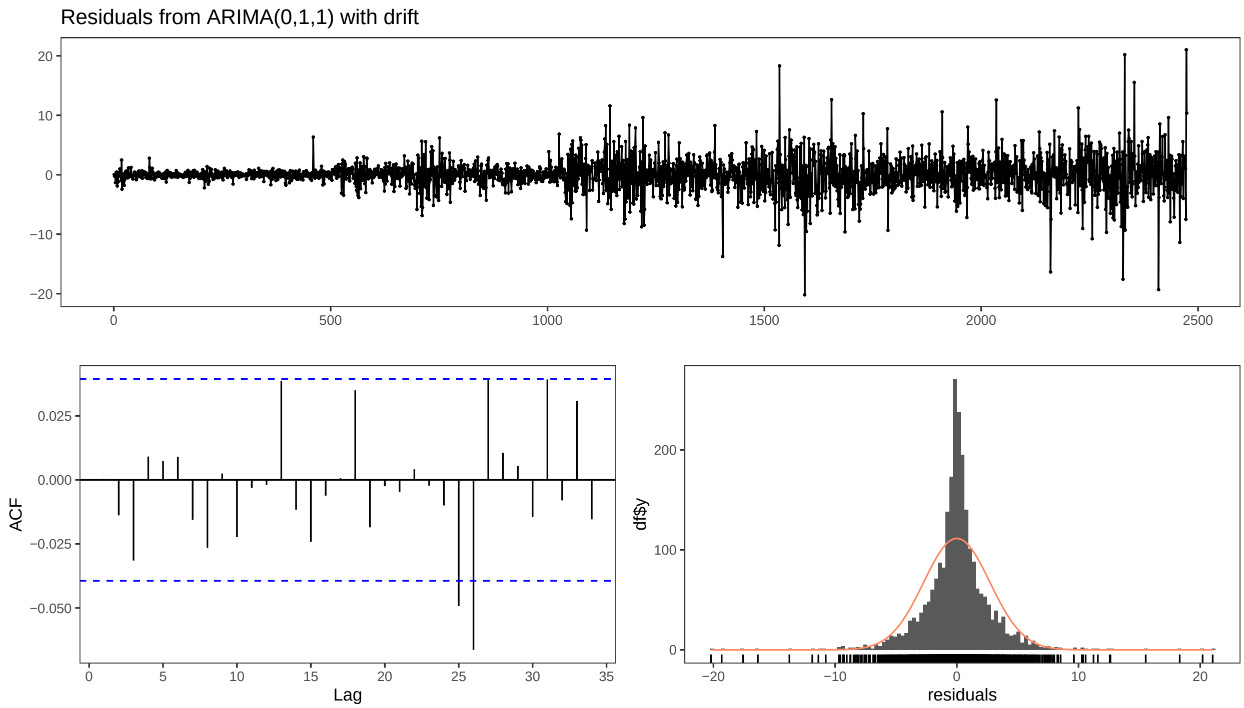
Even when Naive is the winner, it is useful to check ARIMA residuals to see whether any systematic autocorrelation remains (ACF, histogram, Ljung–Box).

```
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)
```

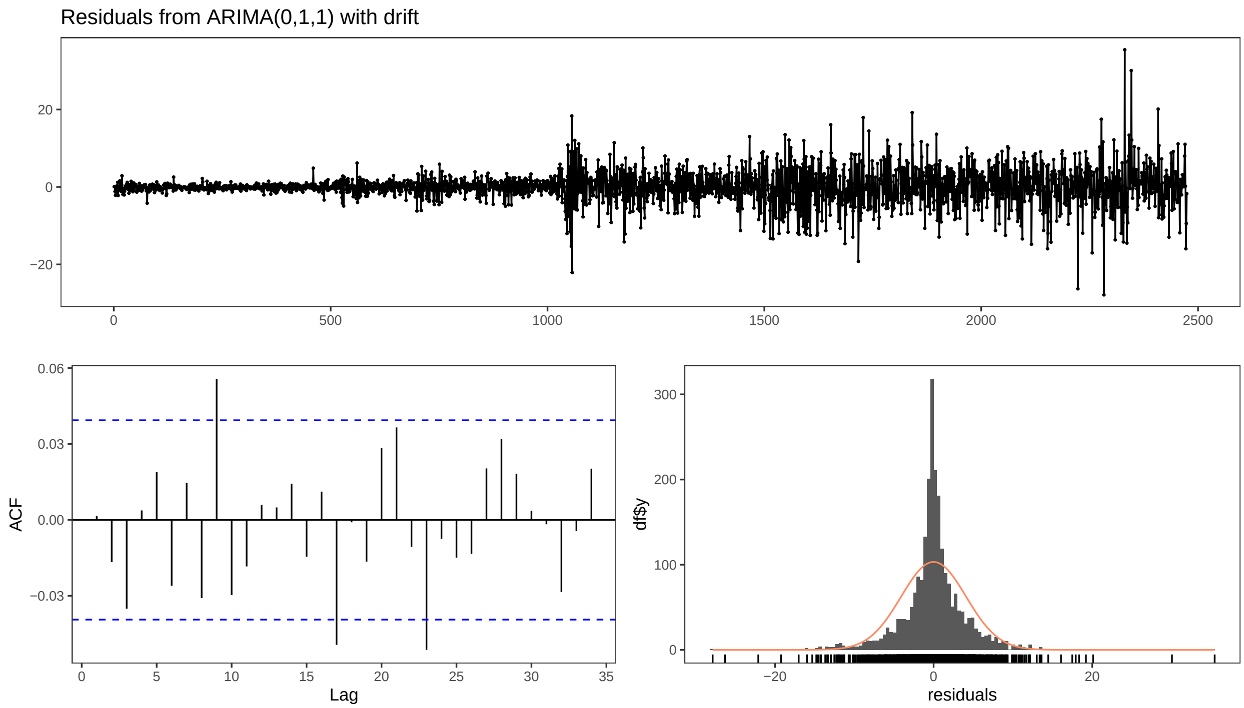
```
##
##
## ### Residual checks for AAPL
## Series: train
## ARIMA(0,1,0) with drift
##
## Coefficients:
##        drift
##       0.0981
## s.e.  0.0479
##
## sigma^2 = 5.678:  log likelihood = -5655.76
## AIC=11315.52   AICc=11315.52   BIC=11327.14
```



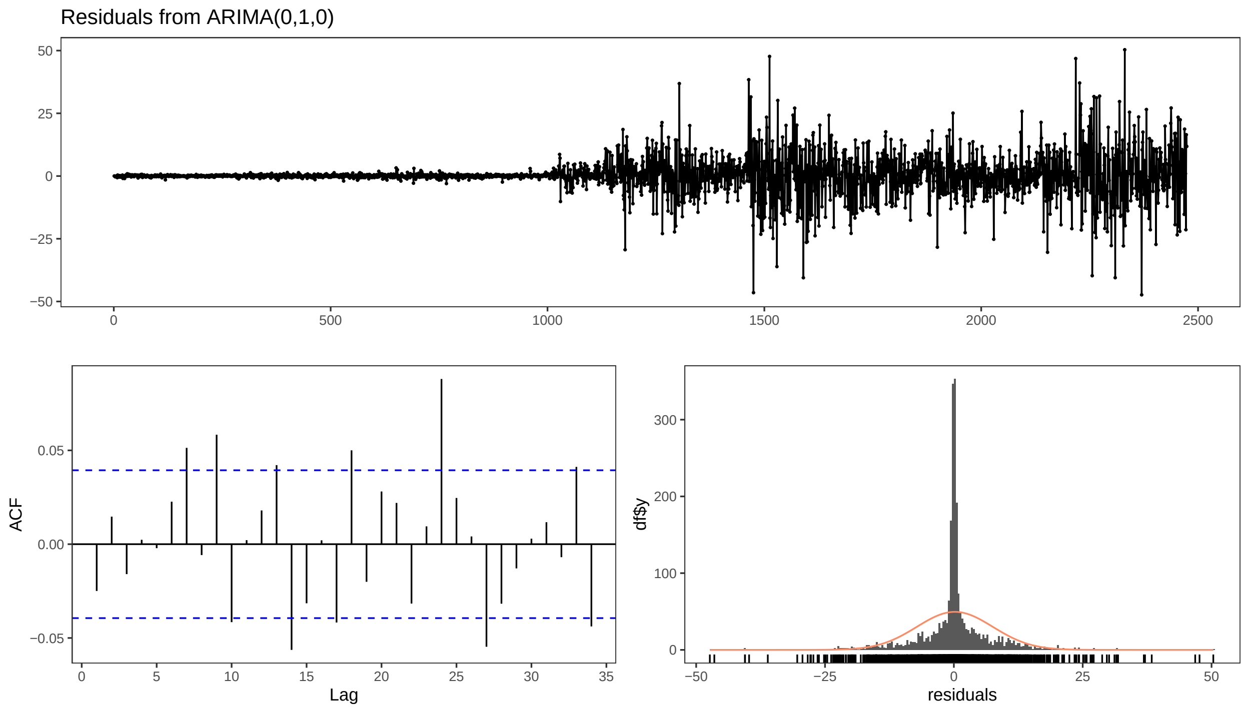

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0) with drift
## Q* = 22.767, df = 10, p-value = 0.01164
##
## Model df: 0.   Total lags used: 10
##
##
##
## ### Residual checks for AMZN
## Series: train
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.0327  0.0898
## s.e.   0.0205  0.0530
##
## sigma^2 = 7.427:  log likelihood = -5987.39
## AIC=11980.77   AICc=11980.78   BIC=11998.21
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 7.0892, df = 9, p-value = 0.6278
##
## Model df: 1.   Total lags used: 10
##
##
##
## ### Residual checks for MSFT
## Series: train
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.0811  0.1869
## s.e.   0.0205  0.0756
##
## sigma^2 = 16.73:  log likelihood = -6991.42
## AIC=13988.85   AICc=13988.86   BIC=14006.29
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 19.156, df = 9, p-value = 0.0239
##
## Model df: 1.   Total lags used: 10
##
##
##
## ### Residual checks for TSLA
## Series: train
## ARIMA(0,1,0)
##
## sigma^2 = 54.33:  log likelihood = -8448.99
## AIC=16899.98   AICc=16899.99   BIC=16905.8
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 23.389, df = 10, p-value = 0.009397
##
## Model df: 0.   Total lags used: 10
```

Residual diagnostics evaluate whether systematic structure remains unexplained by the selected models. Autocorrelation patterns and Ljung–Box test results indicate that residuals for some securities deviate from white noise assumptions. This suggests potential benefits from alternative model classes or extensions incorporating additional dynamics.

Comparative Interpretation

Comparative Forecastability (Lower RMSE = More Predictable)

```
# 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	Naive	9.381	7.587	2.860	1
AMZN	Naive	21.271	19.656	8.518	2
TSLA	Naive	35.506	30.770	7.184	3
MSFT	Naive	39.808	35.896	7.602	4

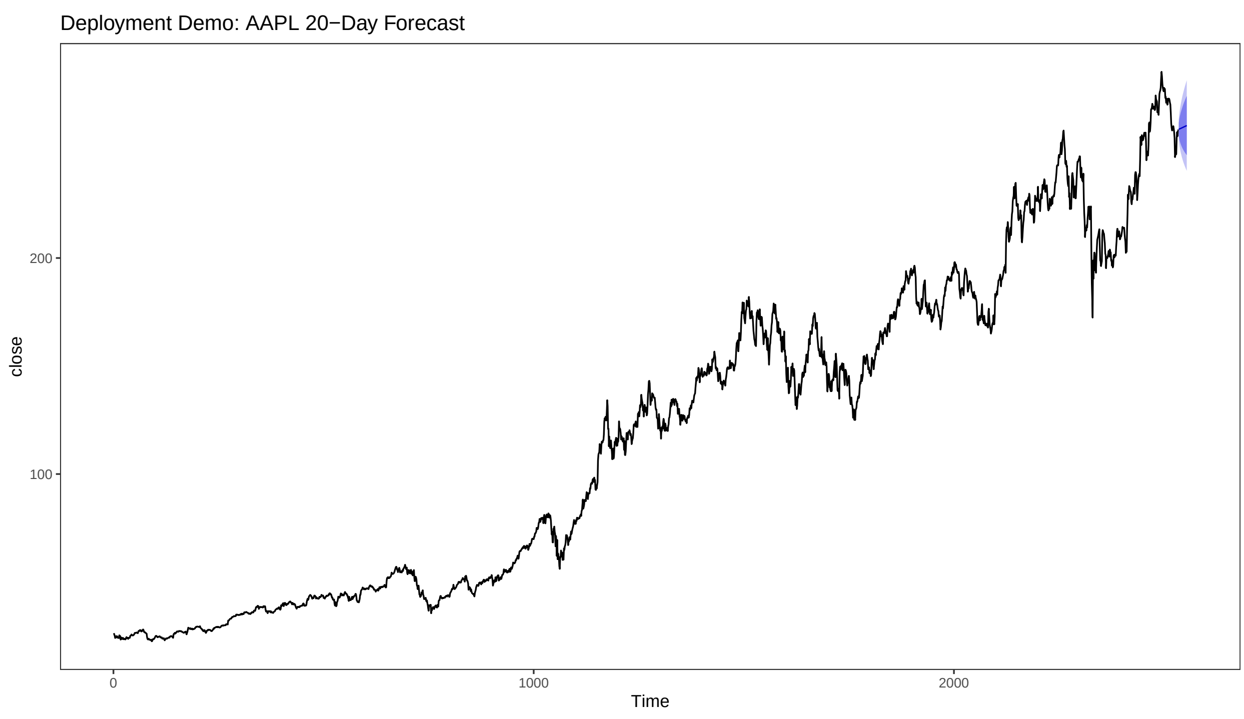
Securities are ranked by minimum out-of-sample RMSE as a proxy for forecastability. Lower error values indicate more stable short-horizon dynamics under the evaluated modeling framework. The resulting ranking highlights heterogeneity in predictability across technology stocks.

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")
```



The forecasting workflow is encapsulated in a reusable function that retrieves recent market data, fits an ARIMA model, and generates short-term forecasts. This modular structure supports integration into dashboards, scheduled pipelines, or interactive applications, subject to appropriate production safeguards.

Conclusion

Key Findings

The analysis demonstrates meaningful differences in trend behavior, volatility regimes, and forecast accuracy across major technology stocks. Model effectiveness is asset-specific, and simple baselines remain competitive in several cases.

Next Improvement

Future work may incorporate alternative model families, exogenous predictors, extended forecast horizons, and automated deployment pipelines to enhance predictive performance and operational usability.

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.
- Wickham, H., et al. (2019). *Welcome to the tidyverse*. Journal of Open Source Software, 4(43), 1686.
- Brockwell, P. J., & Davis, R. A. (2016). *Introduction to time series and forecasting*. Springer.
- Chatfield, C. (2003). *The analysis of time series: an introduction*. CRC press.
- Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.
- Tsay, R. S. (2010). *Analysis of financial time series*. John Wiley & Sons.
- Shumway, R. H., & Stoffer, D. S. (2017). *Time series analysis and its applications: with R examples*. Springer.
- ChatGPT (2024). *Assistance with R programming and time series analysis*.

Thank You!