Week 3A

Pricilla

2025-09-05

I used a CSV that I downloaded from https://www.kaggle.com/datasets/sunnykusawa/stock-price-dataset-eod to create a stock_price_data table in PGADMIN. This data is from 2020. It is 1600 rows of stock prices over a duration of 5 months.

```
library(RSQLite)
library(DBI)
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
library(RODBC)
library(odbc)
library(crayon)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(RPostgres)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:crayon':
##
##     %+%
```

1. Find a dataset that includes time series for two or more separate items. For example, you could use end of day stock or cryptocurrency prices since Jan 1, 2022 for several instruments.

Connected to PgAmin to create Stock_df.

```
# Connect to the database
con <- dbConnect(odbc::odbc(), "Post")


# Run your query
Stock <- dbGetQuery(con, "SELECT * FROM stock_price_data;")
```

I will be using the adj_close for my window functions.

```
stock_df <- Stock
glimpse(stock_df)
```

```
## Rows: 1,600
## Columns: 13
## $ open       <dbl> 1737.27, 1765.00, 1787.23, 1744.91, 1729.00, 1723.93, 1729.~
## $ high       <dbl> 1757.50, 1767.76, 1788.47, 1787.00, 1742.41, 1744.11, 1732.~
## $ low        <dbl> 1736.09, 1728.00, 1755.11, 1741.82, 1724.35, 1721.20, 1705.~
## $ close      <dbl> 1752.64, 1736.25, 1757.76, 1773.96, 1734.16, 1728.23, 1720.~
## $ volume     <dbl> 1053479, 1051308, 986287, 1379642, 465638, 1148684, 1018829~
## $ adj_high   <dbl> 1757.50, 1767.76, 1788.47, 1787.00, 1742.41, 1744.11, 1732.~
## $ adj_low    <dbl> 1736.09, 1728.00, 1755.11, 1741.82, 1724.35, 1721.20, 1705.~
## $ adj_close  <dbl> 1752.64, 1736.25, 1757.76, 1773.96, 1734.16, 1728.23, 1720.~
## $ adj_open   <dbl> 1737.27, 1765.00, 1787.23, 1744.91, 1729.00, 1723.93, 1729.~
## $ adj_volume <dbl> 1053479, 1051308, 986287, 1379642, 465638, 1148684, 1018829~
## $ symbol     <chr> "GOOGL", "GOOGL", "GOOGL", "GOOGL", "GOOGL", "GOOGL", "GOOG~
## $ exchange   <chr> "XNAS", "XNAS", "XNAS", "XNAS", "XNAS", "XNAS", "XNAS", "XN~
## $ date       <date> 2020-12-31, 2020-12-30, 2020-12-29, 2020-12-28, 2020-12-24~
```

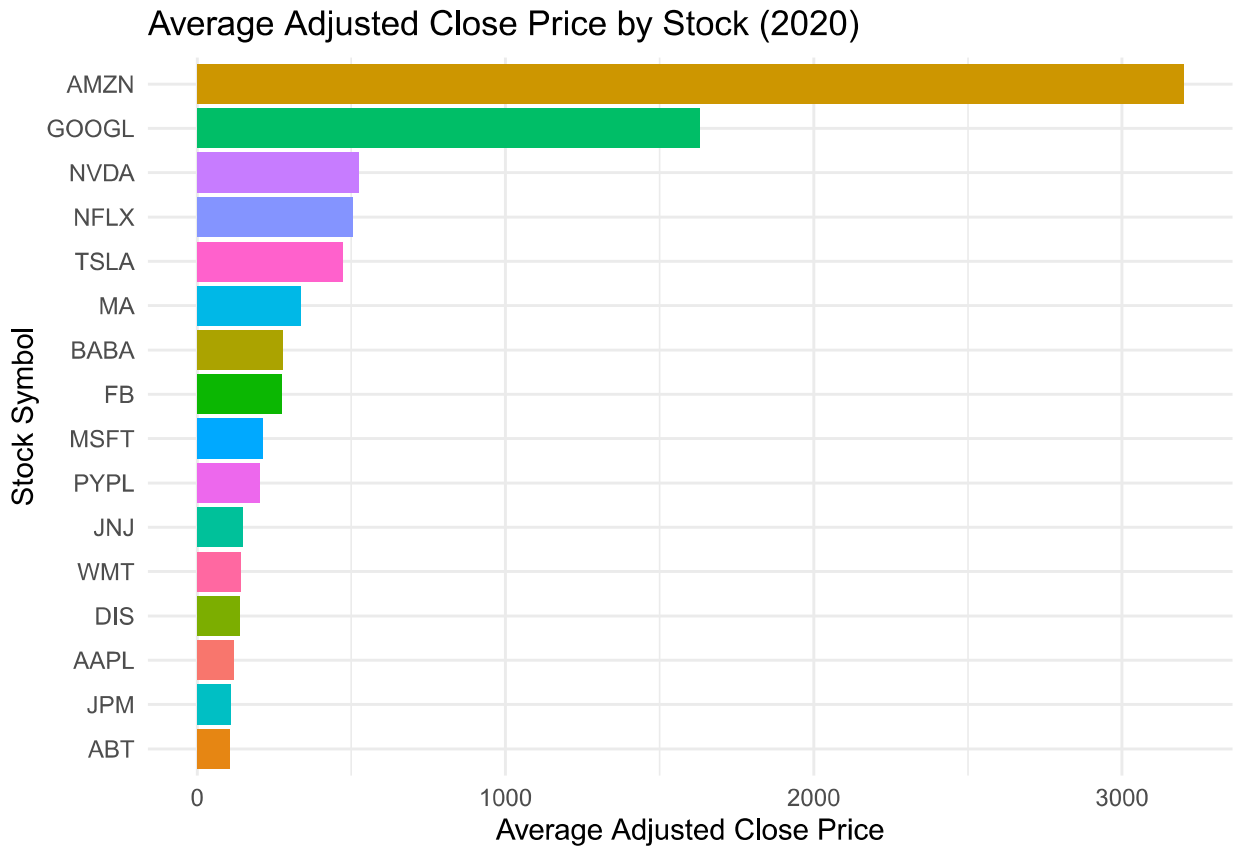
Used SQL to find Average close cost by the year to see which stock did well.

```
YearlyAvg <- dbGetQuery(con, "
  SELECT
    symbol,
    EXTRACT(YEAR FROM date) AS year,
    AVG(adj_close) AS avg_adj_close
  FROM stock_price_data
  GROUP BY symbol, EXTRACT(YEAR FROM date)
  ORDER BY symbol, year
")
```

```r
glimpse(YearlyAvg)
```

```
## Rows: 16
## Columns: 3
## $ symbol        <chr> "AAPL", "ABT", "AMZN", "BABA", "DIS", "FB", "GOOGL", "JN~
## $ year          <dbl> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 20~
## $ avg_adj_close <dbl> 119.0005, 107.1233, 3198.7356, 276.2280, 138.6336, 272.3~
```

```r
# Plot
ggplot(YearlyAvg, aes(x = reorder(symbol, avg_adj_close), y = avg_adj_close, fill = symbol)) +
  geom_col() +
  coord_flip() +  # Flip for easier readability
  labs(
    title = "Average Adjusted Close Price by Stock (2020)",
    x = "Stock Symbol",
    y = "Average Adjusted Close Price"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```



AMAZON AND GOOGLE are the best performing stocks in this dataset.

2. Use window functions (in SQL) to calculate the year-to-date average

```
WinYrAvg <- dbGetQuery(con, "

SELECT
    symbol,
    date,
    EXTRACT(YEAR FROM date) AS year,
    adj_close,
    AVG(adj_close) OVER (
        PARTITION BY symbol, EXTRACT(YEAR FROM date)
        ORDER BY date
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS ytd_avg_adj_close
FROM stock_price_data
")
```

```
glimpse(WinYrAvg)
```

```
## Rows: 1,600
## Columns: 5
## $ symbol            <chr> "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAP~
## $ date              <date> 2020-08-11, 2020-08-12, 2020-08-13, 2020-08-14, 202~
## $ year              <dbl> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020~
## $ adj_close         <dbl> 109.19, 112.82, 114.81, 114.71, 114.41, 115.36, 115.~
## $ ytd_avg_adj_close <dbl> 109.1900, 111.0050, 112.2733, 112.8825, 113.1880, 11~
```

```
WinYrAvg %>%
  filter(symbol %in% c("GOOGL", "AMZN")) %>%
  group_by(symbol) %>%
  slice_head(n = 10)
```

```
## # A tibble: 20 x 5
## # Groups:   symbol [2]
##    symbol date        year adj_close ytd_avg_adj_close
##    <chr>  <date>     <dbl>     <dbl>             <dbl>
##  1 AMZN   2020-08-11  2020     3081.             3081.
##  2 AMZN   2020-08-12  2020     3162.             3121.
##  3 AMZN   2020-08-13  2020     3161.             3135.
##  4 AMZN   2020-08-14  2020     3148.             3138.
##  5 AMZN   2020-08-17  2020     3182.             3147.
##  6 AMZN   2020-08-18  2020     3312.             3174.
##  7 AMZN   2020-08-19  2020     3260.             3187.
##  8 AMZN   2020-08-20  2020     3297.             3201.
##  9 AMZN   2020-08-21  2020     3285.             3210.
## 10 AMZN   2020-08-24  2020     3307.             3220.
## 11 GOOGL  2020-08-11  2020     1481.             1481.
## 12 GOOGL  2020-08-12  2020     1507.             1494.
## 13 GOOGL  2020-08-13  2020     1517.             1501.
## 14 GOOGL  2020-08-14  2020     1505.             1502.
## 15 GOOGL  2020-08-17  2020     1516.             1505.
## 16 GOOGL  2020-08-18  2020     1556.             1514.
## 17 GOOGL  2020-08-19  2020     1545.             1518.
## 18 GOOGL  2020-08-20  2020     1576.             1525.
```

```
## 19 GOOGL  2020-08-21  2020     1576.             1531.
## 20 GOOGL  2020-08-24  2020     1585.             1536.
```

The six-day moving averages for each item..

```
Day6Avg <- dbGetQuery(con, "

SELECT
    symbol,
    date,
    adj_close,
    AVG(adj_close) OVER (
        PARTITION BY symbol
        ORDER BY date
        ROWS BETWEEN 5 PRECEDING AND CURRENT ROW
    ) AS day6_adj_close
FROM stock_price_data
ORDER BY symbol, date
")
```

```
glimpse(Day6Avg)
```

```
## Rows: 1,600
## Columns: 4
## $ symbol         <chr> "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL",~
## $ date           <date> 2020-08-11, 2020-08-12, 2020-08-13, 2020-08-14, 2020-0~
## $ adj_close      <dbl> 109.19, 112.82, 114.81, 114.71, 114.41, 115.36, 115.51,~
## $ day6_adj_close <dbl> 109.1900, 111.0050, 112.2733, 112.8825, 113.1880, 113.5~
```

```
Day6Avg %>%
  filter(symbol %in% c("GOOGL", "AMZN")) %>%
  group_by(symbol) %>%
  slice_head(n = 10)
```

```
## # A tibble: 20 x 4
## # Groups:   symbol [2]
##    symbol date       adj_close day6_adj_close
##    <chr>  <date>         <dbl>          <dbl>
##  1 AMZN   2020-08-11     3081.          3081.
##  2 AMZN   2020-08-12     3162.          3121.
##  3 AMZN   2020-08-13     3161.          3135.
##  4 AMZN   2020-08-14     3148.          3138.
##  5 AMZN   2020-08-17     3182.          3147.
##  6 AMZN   2020-08-18     3312.          3174.
##  7 AMZN   2020-08-19     3260.          3204.
##  8 AMZN   2020-08-20     3297.          3227.
##  9 AMZN   2020-08-21     3285.          3248.
## 10 AMZN   2020-08-24     3307.          3274.
## 11 GOOGL  2020-08-11     1481.          1481.
## 12 GOOGL  2020-08-12     1507.          1494.
## 13 GOOGL  2020-08-13     1517.          1501.
## 14 GOOGL  2020-08-14     1505.          1502.
```

```
## 15 GOOGL  2020-08-17     1516.          1505.
## 16 GOOGL  2020-08-18     1556.          1514.
## 17 GOOGL  2020-08-19     1545.          1524.
## 18 GOOGL  2020-08-20     1576.          1536.
## 19 GOOGL  2020-08-21     1576.          1546.
## 20 GOOGL  2020-08-24     1585.          1559.
```

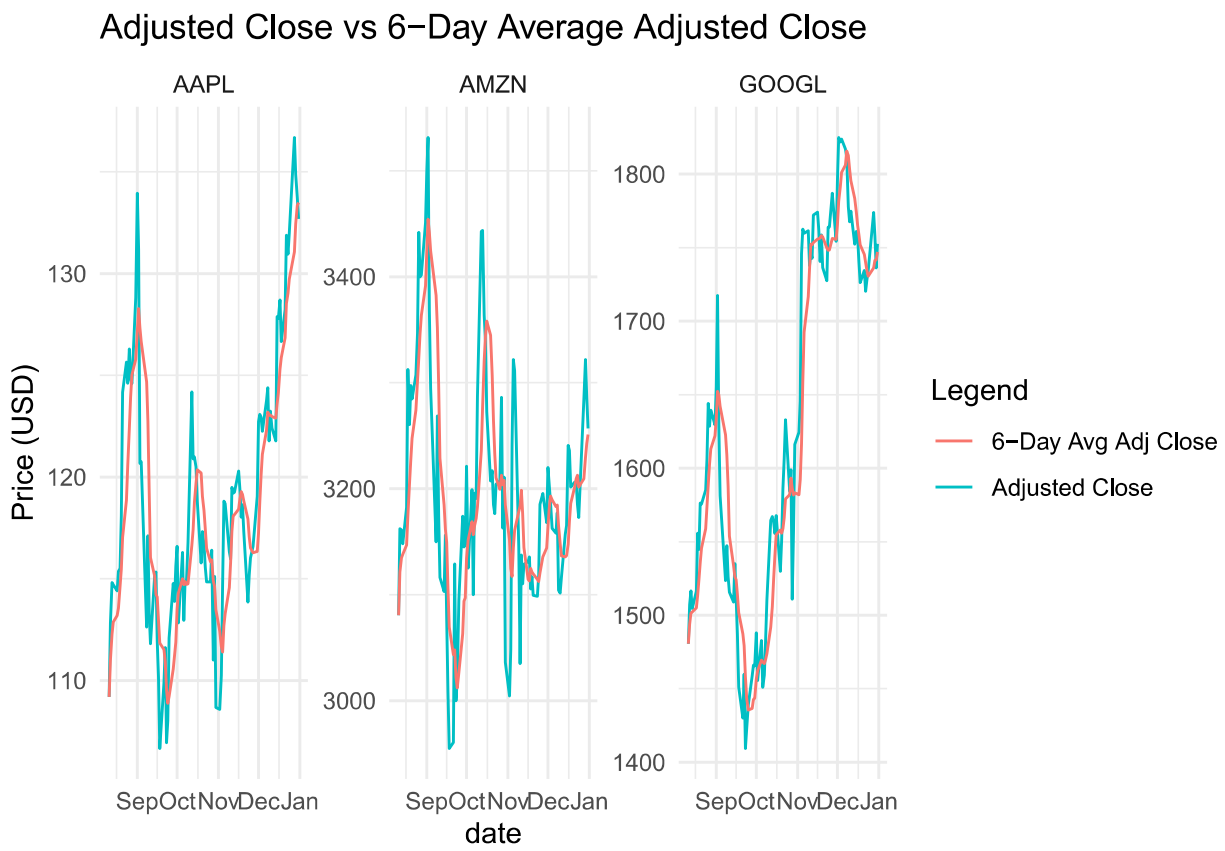
SNAPSHOT OF 6 DAY MOVING AVG FOR AAPL, AMZN & GOOGL.

The 6 day Moving Average Smooths volatility and reduces "noise" from daily price swings.

It helps highlight the general direction (upward/downward) over time.

Traders often use moving averages to identify buy/sell points (e.g., when price crosses the average line).

```
Day6Avg %>%
  filter(symbol %in% c("AAPL", "AMZN", "GOOGL")) %>%  # Filter for some symbols to keep plot readable
  ggplot(aes(x = date)) +
  geom_line(aes(y = adj_close, color = "Adjusted Close")) +
  geom_line(aes(y = day6_adj_close, color = "6-Day Avg Adj Close")) +
  facet_wrap(~symbol, scales = "free_y") +
  labs(
    title = "Adjusted Close vs 6-Day Average Adjusted Close",
    y = "Price (USD)",
    color = "Legend"
  ) +
  theme_minimal()
```



AAPL

The blue and red lines follow each other closely, but you can see when the blue line spikes or dips suddenly, the red line moves more gradually.

AMZN & GOOGL are Similar, smoothing is visible, the red line delays in reacting to sharp blue line movements, making it useful for identifying trends rather than daily changes.

Use window functions (dplyr) to calculate the year-to-date average

```
YearAVGdf <- stock_df %>%
  mutate(year = year(date)) %>%        # Extract year from date
  arrange(symbol, date) %>%            # Ensure proper order for cumulative average
  group_by(symbol, year) %>%
  mutate(
    ytd_avg_adj_close = cummean(adj_close)  # Cumulative average within each symbol-year
  ) %>%
  ungroup()
```

```
YearAVGdiff <- stock_df %>%
  mutate(year = lubridate::year(date)) %>%   # Extract year from date (ensure lubridate is loaded)
  arrange(symbol, date) %>%                  # Ensure proper order for cumulative average
  group_by(symbol, year) %>%
  mutate(
    ytd_avg_adj_close = cummean(adj_close),  # Cumulative average within each symbol-year
    diff_from_avg = adj_close - ytd_avg_adj_close  # Difference between actual and average
  ) %>%
  ungroup()
```

```
glimpse(YearAVGdf)
```

```
## Rows: 1,600
## Columns: 15
## $ open              <dbl> 447.88, 441.99, 457.72, 459.32, 464.25, 457.41, 463.~
## $ high              <dbl> 449.93, 453.10, 464.17, 460.00, 464.35, 464.00, 468.~
## $ low               <dbl> 436.43, 441.19, 455.71, 452.18, 455.85, 456.03, 462.~
## $ close             <dbl> 437.50, 452.04, 460.04, 459.63, 458.43, 462.25, 462.~
## $ volume            <dbl> 46975594, 41486205, 52520516, 41391302, 29431414, 26~
## $ adj_high          <dbl> 112.29, 113.08, 115.84, 114.80, 115.89, 115.80, 116.~
## $ adj_low           <dbl> 108.92, 110.11, 113.73, 112.85, 113.77, 113.81, 115.~
## $ adj_close         <dbl> 109.19, 112.82, 114.81, 114.71, 114.41, 115.36, 115.~
## $ adj_open          <dbl> 111.78, 110.31, 114.23, 114.63, 115.86, 114.16, 115.~
## $ adj_volume        <dbl> 187902376, 165944820, 210082064, 165565208, 11772565~
## $ symbol            <chr> "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAP~
## $ exchange          <chr> "XNAS", "XNAS", "XNAS", "XNAS", "XNAS", "XNAS", "XNA~
## $ date              <date> 2020-08-11, 2020-08-12, 2020-08-13, 2020-08-14, 202~
## $ year              <dbl> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020~
## $ ytd_avg_adj_close <dbl> 109.1900, 111.0050, 112.2733, 112.8825, 113.1880, 11~
```

```
YearAVGdf %>%
  filter(symbol %in% c("GOOGL", "AMZN")) %>%
  group_by(symbol) %>%
  slice_head(n = 10)
```

```
## # A tibble: 20 x 15
## # Groups:   symbol [2]
##     open  high   low close volume adj_high adj_low adj_close adj_open adj_volume
##    <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>   <dbl>     <dbl>    <dbl>      <dbl>
##  1 3113. 3159. 3073  3081. 3.72e6    3159.   3073      3081.    3113.    3718057
##  2 3108  3174. 3101. 3162. 3.53e6    3174.   3101.     3162.    3108     3527229
##  3 3183. 3218. 3155  3161. 3.15e6    3218.   3155      3161.    3183.    3149043
##  4 3178. 3178. 3120  3148. 2.75e6    3178.   3120      3148.    3178.    2751723
##  5 3173. 3195. 3154. 3182. 2.68e6    3195.   3154.     3182.    3173.    2675890
##  6 3212  3320  3206. 3312. 5.35e6    3320    3206.     3312.    3212     5345974
##  7 3303. 3316. 3256  3260. 4.19e6    3316.   3256      3260.    3303.    4185137
##  8 3252  3313. 3238  3297. 3.33e6    3313.   3238      3297.    3252     3332478
##  9 3295  3314. 3275. 3285. 3.58e6    3314.   3275.     3285.    3295     3575862
## 10 3310. 3380. 3258. 3307. 4.67e6    3380.   3258.     3307.    3310.    4666258
## 11 1494  1510. 1478. 1481. 1.55e6    1510.   1478.     1481.    1494     1554853
## 12 1487. 1512. 1485  1507. 1.13e6    1512.   1485      1507.    1487.    1126560
## 13 1508. 1537. 1508. 1517. 1.12e6    1537.   1508.     1517.    1508.    1119711
## 14 1514. 1520. 1499  1505. 1.10e6    1520.   1499      1505.    1514.    1097059
## 15 1516. 1524. 1505  1516. 9.98e5    1524.   1505      1516.    1516.     998010
## 16 1526. 1557. 1522. 1556. 1.42e6    1557.   1522.     1556.    1526.    1418850
## 17 1552. 1569. 1540  1545. 1.52e6    1569.   1540      1545.    1552.    1523592
## 18 1540. 1580. 1534. 1576. 1.32e6    1580.   1534.     1576.    1540.    1319131
## 19 1572. 1592. 1562. 1576. 1.74e6    1592.   1562.     1576.    1572.    1742275
## 20 1592. 1609. 1575. 1585. 1.28e6    1609.   1575.     1585.    1592.    1281893
## # i 5 more variables: symbol <chr>, exchange <chr>, date <date>, year <dbl>,
## #   ytd_avg_adj_close <dbl>
```

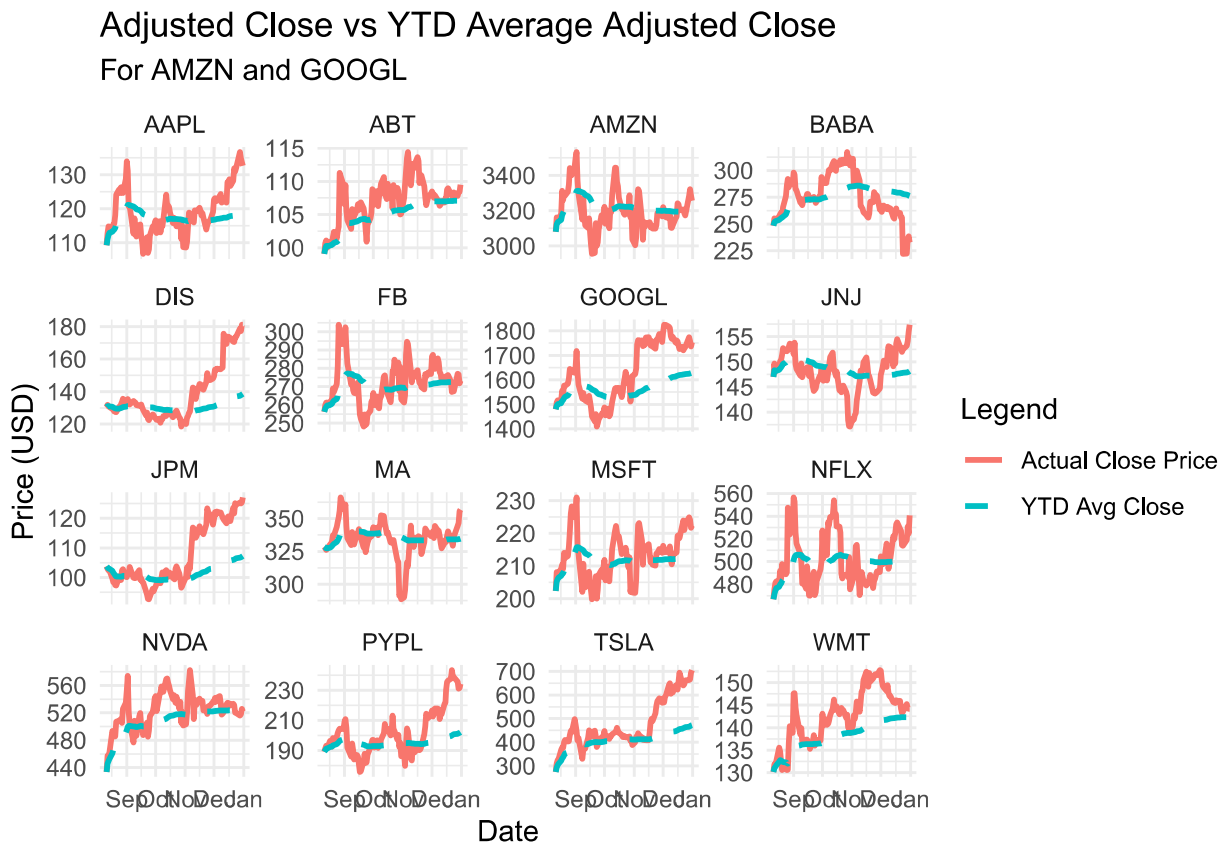
```
YearAVGdf %>%
  filter(symbol %in% c("GOOGL", "AMZN")) %>%
  group_by(symbol) %>%
  slice_head(n = 10) %>%
  select(symbol, date, adj_close, ytd_avg_adj_close)
```

```
## # A tibble: 20 x 4
## # Groups:   symbol [2]
##    symbol date       adj_close ytd_avg_adj_close
##    <chr>  <date>         <dbl>             <dbl>
##  1 AMZN   2020-08-11     3081.             3081.
##  2 AMZN   2020-08-12     3162.             3121.
##  3 AMZN   2020-08-13     3161.             3135.
##  4 AMZN   2020-08-14     3148.             3138.
##  5 AMZN   2020-08-17     3182.             3147.
##  6 AMZN   2020-08-18     3312.             3174.
##  7 AMZN   2020-08-19     3260.             3187.
##  8 AMZN   2020-08-20     3297.             3201.
##  9 AMZN   2020-08-21     3285.             3210.
## 10 AMZN   2020-08-24     3307.             3220.
## 11 GOOGL  2020-08-11     1481.             1481.
## 12 GOOGL  2020-08-12     1507.             1494.
## 13 GOOGL  2020-08-13     1517.             1501.
## 14 GOOGL  2020-08-14     1505.             1502.
## 15 GOOGL  2020-08-17     1516.             1505.
## 16 GOOGL  2020-08-18     1556.             1514.
## 17 GOOGL  2020-08-19     1545.             1518.
```

```
## 18 GOOGL  2020-08-20     1576.             1525.
## 19 GOOGL  2020-08-21     1576.             1531.
## 20 GOOGL  2020-08-24     1585.             1536.
```

```r
ggplot(YearAVGdf, aes(x = date)) +
  geom_line(aes(y = adj_close, color = "Actual Close Price"), size = 1) +
  geom_line(aes(y = ytd_avg_adj_close, color = "YTD Avg Close"), linetype = "dashed", size = 1) +
  facet_wrap(~symbol, scales = "free_y") +  # One panel per stock
  labs(
    title = "Adjusted Close vs YTD Average Adjusted Close",
    subtitle = "For AMZN and GOOGL",
    x = "Date",
    y = "Price (USD)",
    color = "Legend"
  ) +
  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.
```



Three stock snapshot Explanation

AMZN (Amazon)

Red Line (Actual): Shows significant fluctuations with a general upward trend.

Blue Line (YTD Average): Smoothly rises over time, reflecting Amazon's growing performance throughout the year.

The stock had sharp peaks and dips, likely due to market volatility or earnings releases.

Despite volatility, the YTD average increases steadily, suggesting consistent overall growth.

Amazon consistently traded above the YTD average, indicating strong investor confidence.

Google

Red Line and Actual: Starts lower and gradually increases with fewer sharp fluctuations compared to AMZN.

Blue Line: Steady linear climb, indicating a healthy and consistent rise in adjusted close prices.

GOOGL experienced less volatility than AMZN.

The price often returned to the YTD average, showing alignment between market price and intrinsic valuation.

Late in the year, the actual price diverged above the average signalling a strong bullish signal.

TSLA-Tesla

Red Line Actual price show high volatility with steep increases and dips.

The Blue Line which is YTD is Steady but much lower than the actual price for most of the timeline.

Tesla's stock saw explosive growth, which isn't matched by the YTD average until late.

The gap between actual price and YTD average indicates rapid, recent price surges.

Ideal example of a high momentum stock with elevated risk.

When I Calculate the difference between Average Price and adj_close it makes sense to not time the market in hopes of getting a good deal but to just continue buying stock because for my favorite stock AMZN I am not seeing a significant dip in prices that warrants timing the market for a good deal. 2020-08-18 to 2020-08-24 would have been a good day to sell because the average Price was high.

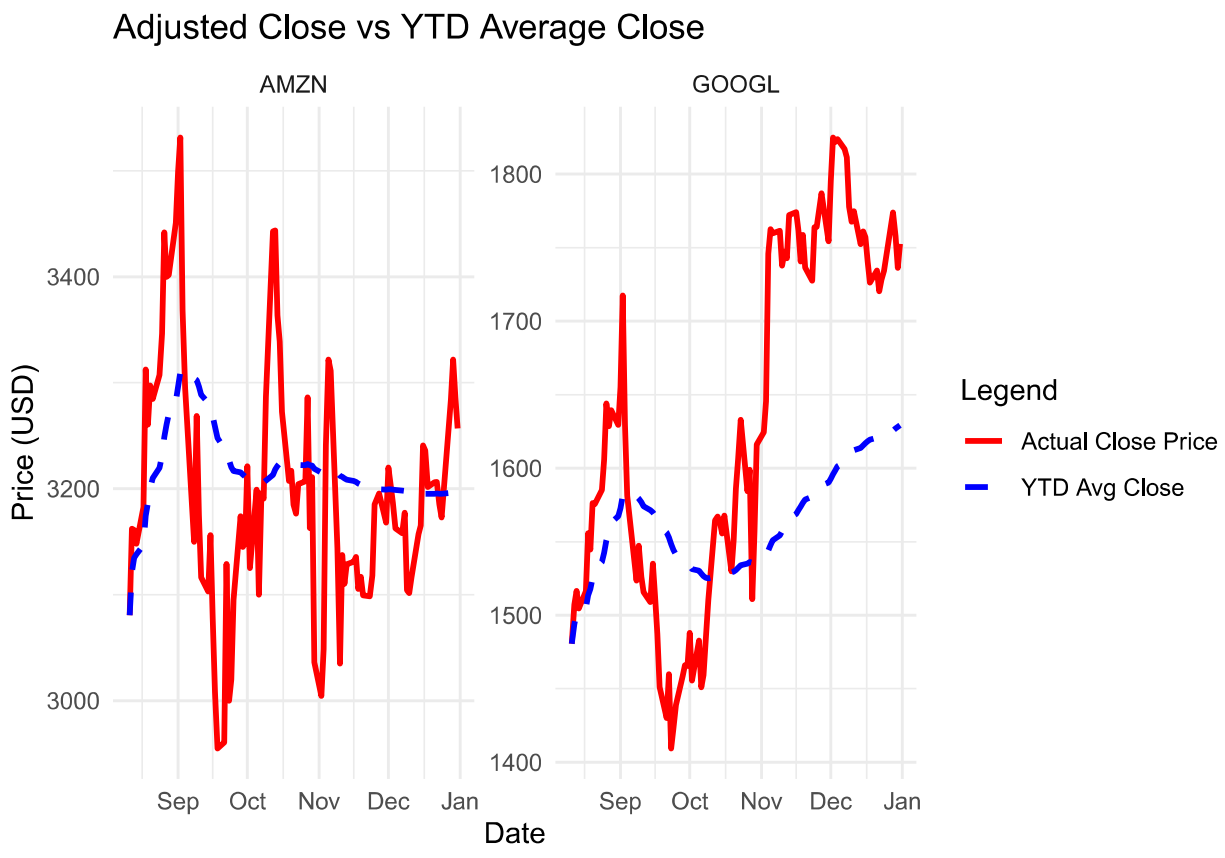
```
YearAVGdiff %>%
  filter(symbol %in% c("GOOGL", "AMZN")) %>%
  group_by(symbol) %>%
  slice_head(n = 10) %>%
  select(symbol, date, adj_close, ytd_avg_adj_close, diff_from_avg)
```

```
## # A tibble: 20 x 5
## # Groups:   symbol [2]
##    symbol date       adj_close ytd_avg_adj_close diff_from_avg
##    <chr>  <date>         <dbl>             <dbl>         <dbl>
##  1 AMZN   2020-08-11     3081.             3081.          0
##  2 AMZN   2020-08-12     3162.             3121.         40.8
##  3 AMZN   2020-08-13     3161.             3135.         26.4
##  4 AMZN   2020-08-14     3148.             3138.         10.0
##  5 AMZN   2020-08-17     3182.             3147.         35.5
##  6 AMZN   2020-08-18     3312.             3174.        138.
##  7 AMZN   2020-08-19     3260.             3187.         73.7
##  8 AMZN   2020-08-20     3297.             3201.         96.8
##  9 AMZN   2020-08-21     3285.             3210.         74.8
## 10 AMZN   2020-08-24     3307.             3220.         87.8
## 11 GOOGL  2020-08-11     1481.             1481.          0
```

```
## 12 GOOGL  2020-08-12     1507.             1494.         13.4
## 13 GOOGL  2020-08-13     1517.             1501.         15.2
## 14 GOOGL  2020-08-14     1505.             1502.          2.37
## 15 GOOGL  2020-08-17     1516.             1505.         11.2
## 16 GOOGL  2020-08-18     1556.             1514.         42.3
## 17 GOOGL  2020-08-19     1545.             1518.         26.7
## 18 GOOGL  2020-08-20     1576.             1525.         51.0
## 19 GOOGL  2020-08-21     1576.             1531.         44.7
## 20 GOOGL  2020-08-24     1585.             1536.         48.9
```

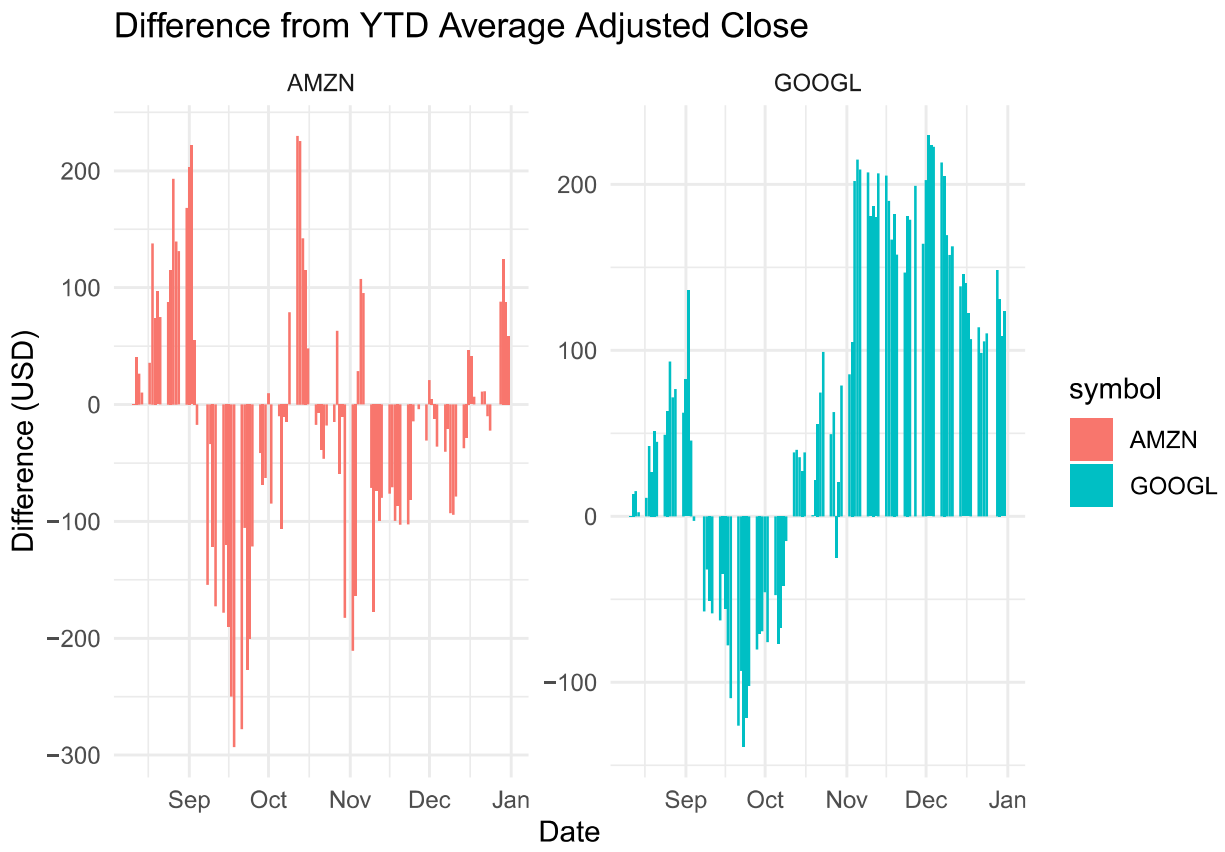
```r
# Filter only AMZN and GOOGL
filtered_df <- YearAVGdiff %>%
  filter(symbol %in% c("AMZN", "GOOGL"))

# Plot: Actual vs YTD Avg Close
ggplot(filtered_df, aes(x = date)) +
  geom_line(aes(y = adj_close, color = "Actual Close Price"), size = 1) +
  geom_line(aes(y = ytd_avg_adj_close, color = "YTD Avg Close"), size = 1, linetype = "dashed") +
  facet_wrap(~symbol, scales = "free_y") +
  labs(
    title = "Adjusted Close vs YTD Average Close",
    x = "Date", y = "Price (USD)",
    color = "Legend"
  ) +
  theme_minimal() +
  scale_color_manual(values = c("Actual Close Price" = "red", "YTD Avg Close" = "blue"))
```



Plot for how far the actual close was from the YTD average This Plot shows the difference between the daily Adjusted Close price and the Year-to-Date (YTD) Average Adjusted Close for two stocks. AMZN (Amazon) and GOOGL over the latter part of 2020.

```
ggplot(filtered_df, aes(x = date, y = diff_from_avg, fill = symbol)) +
  geom_col(position = "dodge") +
  labs(
    title = "Difference from YTD Average Adjusted Close",
    x = "Date", y = "Difference (USD)"
  ) +
  facet_wrap(~symbol, scales = "free_y") +
  theme_minimal()
```



AMZN

The differences swing widely between +250 and -300 USD showing high volatility.

In early August, AMZN traded above its YTD average.

In late September to mid-October, AMZN consistently traded below the YTD average — suggesting a slump relative to its average performance.

Later, it bounced back but still fluctuated around the YTD average.

GOOGL

Google showed a more consistent uptrend compared to AMZN.

After some minor dips in August and September, GOOGL showed mostly positive differences. Closing well above its YTD average for most of November and December.

This suggests strong, steady performance in the second half of the year.