

Growth Curves and Descriptive Statistics

Wendy Shane Balboa | Aretha Rgeel Castigon | Saintjoy Mandalinao | Ivy Ringcunada

This dataset contains historical daily stock prices for Apple (since 1980) and Microsoft (since 1986), adjusted for stock splits and dividends. The columns represent Opening, High, Low, and Closing prices for each trading day. The dataset follows a daily periodicity, excluding weekends and market holidays. The adjustments for stock splits and dividends ensure that past stock prices remain comparable to present-day values.

Sheet 1: Combined Data of Apple and Microsoft Historical Annual Stock Price Data[illegible]

Sheet 2: Apple and Microsoft Historical Annual Stock Price Data for Year 1986-2025

	A	B	C	D	E	F	G	H	I	J	K	L
1	Date	Apple_Open	App_High	App_Low	App_Close	Mic_Open	Mic_High	Mic_Low	Mic_Close			
2	3/13/1986	0.0853	0.0862	0.084	0.0853	0.0548	0.0628	0.0548	0.0602			
3	3/14/1986	0.0853	0.0905	0.0853	0.09	0.0602	0.0634	0.0602	0.0623			
4	3/17/1986	0.0896	0.0896	0.0874	0.0896	0.0623	0.0639	0.0623	0.0634			
5	3/18/1986	0.0896	0.0939	0.0892	0.0926	0.0634	0.0639	0.0612	0.0618			
6	3/19/1986	0.0926	0.0939	0.0909	0.0913	0.0618	0.0623	0.0602	0.0607			
7	3/20/1986	0.0965	0.1021	0.0965	0.0974	0.0607	0.0607	0.0585	0.0591			
8	3/21/1986	0.0974	0.0991	0.0948	0.0952	0.0591	0.0602	0.0564	0.0575			
9	3/24/1986	0.0952	0.0952	0.0909	0.0922	0.0575	0.0575	0.0553	0.0559			
10	3/25/1986	0.0922	0.0961	0.0922	0.0961	0.0559	0.0569	0.0553	0.0569			
11	3/26/1986	0.0961	0.0991	0.0961	0.0974	0.0569	0.0591	0.0564	0.0585			
12	3/27/1986	0.0974	0.0999	0.0974	0.0974	0.0585	0.0596	0.0585	0.0596			
13	3/31/1986	0.0974	0.0982	0.0965	0.0974	0.0596	0.0596	0.058	0.0591			
14	4/1/1986	0.0974	0.0974	0.0931	0.0939	0.0591	0.0591	0.0585	0.0585			
15	4/2/1986	0.0939	0.0943	0.0905	0.0939	0.0585	0.0602	0.0585	0.0591			
16	4/3/1986	0.0939	0.0952	0.0926	0.0931	0.0596	0.0612	0.0596	0.0596			
17	4/4/1986	0.0931	0.0931	0.0917	0.0922	0.0596	0.0602	0.0596	0.0596			
18	4/7/1986	0.0922	0.0948	0.0905	0.0929	0.0596	0.0602	0.0575	0.0585			

+

...

×

\$JAM

COMBINED DATA 1986 1987 YEARLY AVG OF HIGH +

Sheet 3: Apple and Microsoft Historical Annual Stock Price Data for Year 1987-2025

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Apple_Open	App_High	App_Low	App_Close	Mic_Open	Mic_High	Mic_Low	Mic_Close		
2	1/2/1987	0.1392	0.1417	0.1383	0.1409	0.1037	0.1047	0.1021	0.1026		
3	1/5/1987	0.1422	0.1491	0.1413	0.1482	0.1026	0.1096	0.1021	0.1085		
4	1/6/1987	0.1486	0.1516	0.1469	0.1508	0.1085	0.1107	0.108	0.1101		
5	1/7/1987	0.1512	0.1547	0.1504	0.1542	0.1101	0.1116	0.1101	0.1116		
6	1/8/1987	0.1542	0.1555	0.1534	0.1542	0.1166	0.1209	0.1166	0.1198		
7	1/9/1987	0.1542	0.1577	0.1529	0.1564	0.1198	0.1252	0.1192	0.1246		
8	1/12/1987	0.1568	0.1577	0.1542	0.1568	0.1246	0.1337	0.1235	0.1327		
9	1/13/1987	0.1555	0.1564	0.1538	0.1538	0.1316	0.1316	0.1268	0.1289		
10	1/14/1987	0.1538	0.1663	0.1534	0.1659	0.1289	0.1295	0.1262	0.1289		
11	1/15/1987	0.1663	0.1771	0.1654	0.1719	0.1289	0.1391	0.1289	0.1359		
12	1/16/1987	0.1723	0.1723	0.1646	0.168	0.1359	0.1359	0.13	0.1327		
13	1/19/1987	0.168	0.1831	0.165	0.1831	0.1327	0.1375	0.13	0.1375		
14	1/20/1987	0.1896	0.1921	0.1775	0.1779	0.1375	0.1429	0.137	0.1397		
15	1/21/1987	0.1754	0.1762	0.1689	0.1689	0.1397	0.1493	0.137	0.1456		
16	1/22/1987	0.1685	0.1814	0.1671	0.1809	0.1456	0.1606	0.1413	0.1601		
17	1/23/1987	0.1809	0.1827	0.1732	0.1732	0.1601	0.1665	0.145	0.145		
18	1/26/1987	0.1723	0.174	0.1706	0.1715	0.1461	0.1493	0.1413	0.1477		
19	1/27/1987	0.1723	0.1831	0.1719	0.1818	0.1477	0.1558	0.1477	0.1536		
20	1/28/1987	0.1827	0.1921	0.1797	0.1909	0.1536	0.1563	0.1493	0.1542		
21	1/29/1987	0.1926	0.1973	0.184	0.1866	0.1542	0.1558	0.1445	0.1477		
22	1/30/1987	0.1861	0.1926	0.1814	0.1913	0.1477	0.1574	0.1456	0.1571		

+

...

×

\$JAM

COMBINED DATA 1986 1987 YEARLY AVG OF HIGH +

Sheet 4: Computed Yearly Average of Apple and Microsoft Historical Annual Stock Price Data for Year 1987-2025

[illegible]

Reading CSV file

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Contains R code for installing packages, loading libraries, and reading a CSV file. The code is as follows:

```
1 #AMAT 132 Exercise 2
2 #(Growth Curves and Descriptive Statistics
3
4
5 #installing the packages
6 install.packages("ggplot2")
7 install.packages("dplyr")
8 install.packages("readr")
9
10
11 # Load necessary libraries
12 library(ggplot2)
13 library(dplyr)
14 library(readr)
15
16 # Read CSV file (Ensure correct file path)
17 df <- read_csv("C:/Users/saint/Documents/AMAT132/AMAT_132.csv")
18
19
20 # Display first few rows
21 head(df)
22
23 # Clean column names (Replace spaces with underscores, and handle other special characters)
24 colnames(df) <- make.names(colnames(df)) # More robust than gsub
25
26 # Convert necessary columns to numeric, handle potential errors
```
- Console:** Shows the execution of the code, including the output of `read_csv` and the `head(df)` command. The output for `read_csv` is:

```
> # Read CSV file (Ensure correct file path)
> df <- read_csv("C:/Users/saint/Documents/AMAT132/AMAT_132.csv")
Rows: 39 Columns: 3
  Column specification
Delimiter: ","
dbl (3): Date, App_High, Mic_High

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
> |
```
- Environment:** Shows the variable `df` with 39 observations and 3 variables.
- Viewer:** Displays a plot of the data, showing a scatter plot with a yellow background and a black line.

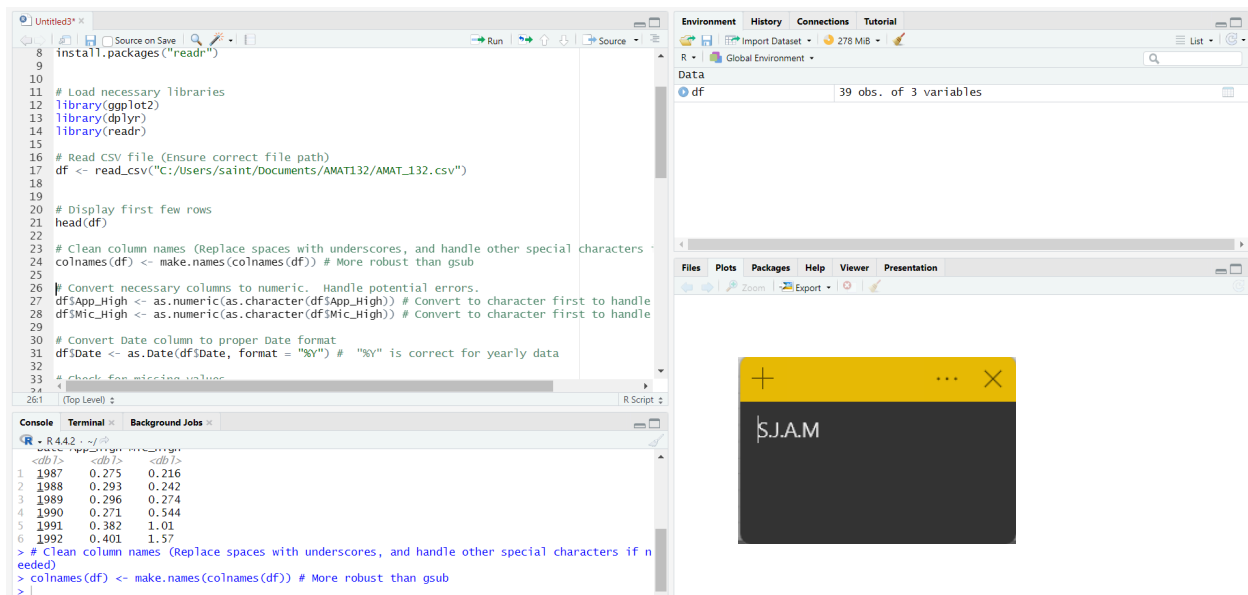
Displaying first few rows

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Contains the same R code as the previous screenshot, but with the `head(df)` command added to the script.
- Console:** Shows the execution of the code, including the output of `head(df)`. The output is:

```
> head(df)
# A tibble: 6 x 3
  Date App_High Mic_High
<dbl> <dbl> <dbl>
1 1987  0.275  0.216
2 1988  0.293  0.242
3 1989  0.296  0.274
4 1990  0.271  0.544
5 1991  0.382  1.01
6 1992  0.401  1.57
```
- Environment:** Shows the variable `df` with 39 observations and 3 variables.
- Viewer:** Displays a plot of the data, showing a scatter plot with a yellow background and a black line.

Clean column names (Replace spaces with underscores, and handle other special characters if needed)



The RStudio interface shows the following components:

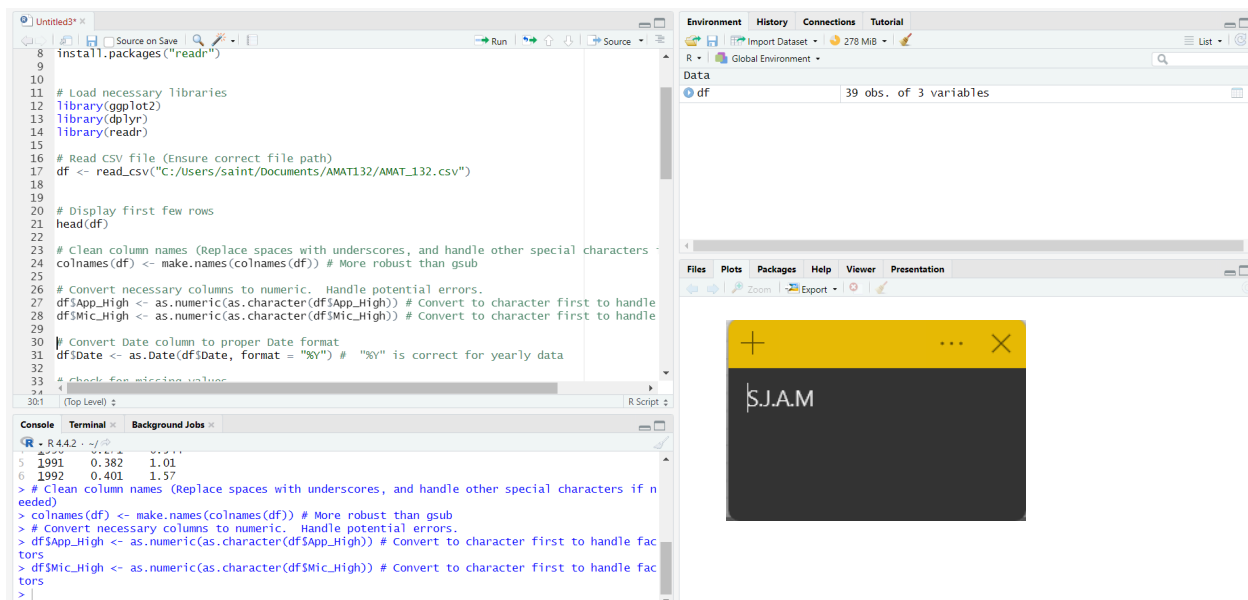
- Source Editor:** Contains R code for loading libraries, reading a CSV file, and cleaning column names.
- Environment Pane:** Shows the data frame 'df' with 39 observations and 3 variables.
- Console:** Displays the output of the R script, including the first few rows of the data frame and the result of the column name cleaning operation.

```
8 install.packages("readr")
9
10
11 # Load necessary libraries
12 library(ggplot2)
13 library(dplyr)
14 library(readr)
15
16 # Read CSV file (Ensure correct file path)
17 df <- read_csv("c:/Users/saint/Documents/AMAT132/AMAT_132.csv")
18
19 # Display first few rows
20 head(df)
21
22 # Clean column names (Replace spaces with underscores, and handle other special characters if needed)
23 colnames(df) <- make.names(colnames(df)) # More robust than gsub
24
25 # Convert necessary columns to numeric. Handle potential errors.
26 df$App_High <- as.numeric(as.character(df$App_High)) # Convert to character first to handle
27 df$Mic_High <- as.numeric(as.character(df$Mic_High)) # Convert to character first to handle
28
29 # Convert Date column to proper Date format
30 df$Date <- as.Date(df$Date, format = "%Y") # "%Y" is correct for yearly data
31
32 # Check for missing values
33
```

Console Output:

```
R - R 4.4.2 - ~/R
> head(df)
  <dbl> <dbl> <dbl>
1 1987  0.275  0.216
2 1988  0.293  0.242
3 1989  0.296  0.274
4 1990  0.271  0.544
5 1991  0.382  1.01
6 1992  0.401  1.57
> # Clean column names (Replace spaces with underscores, and handle other special characters if needed)
> colnames(df) <- make.names(colnames(df)) # More robust than gsub
>
```

Convert necessary columns to numeric. Handle potential errors.



The RStudio interface shows the following components:

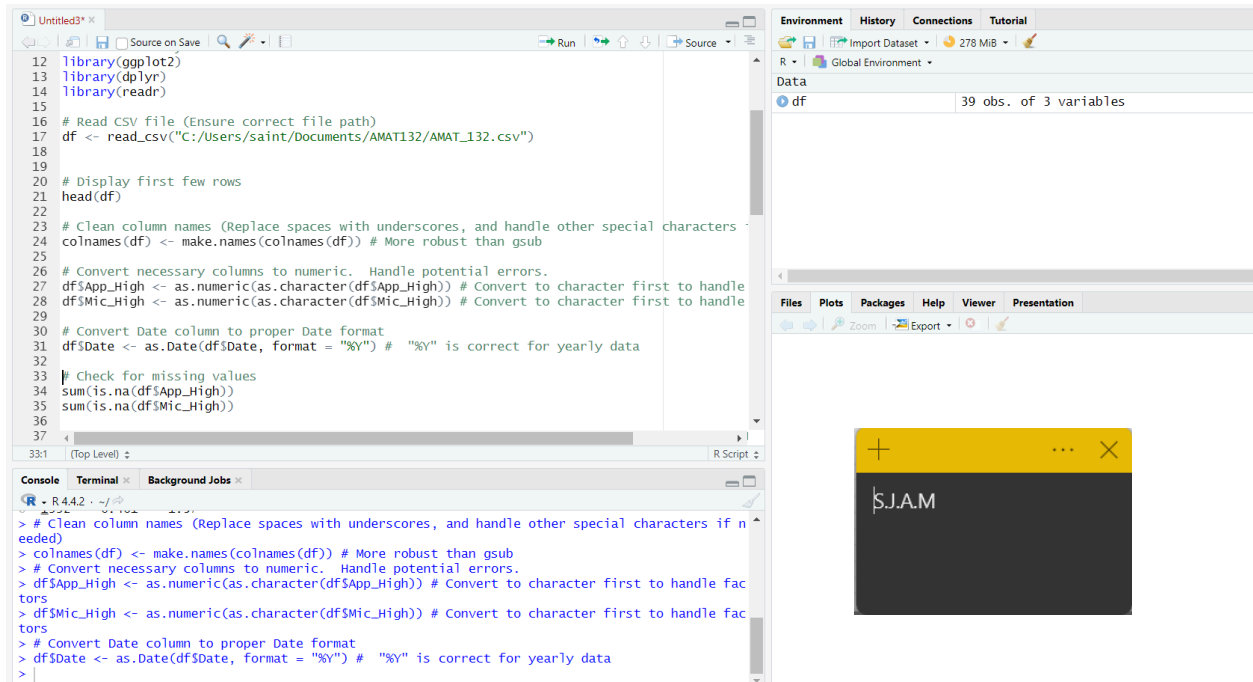
- Source Editor:** Contains R code for loading libraries, reading a CSV file, and converting columns to numeric.
- Environment Pane:** Shows the data frame 'df' with 39 observations and 3 variables.
- Console:** Displays the output of the R script, including the first few rows of the data frame and the result of the column conversion operation.

```
8 install.packages("readr")
9
10
11 # Load necessary libraries
12 library(ggplot2)
13 library(dplyr)
14 library(readr)
15
16 # Read CSV file (Ensure correct file path)
17 df <- read_csv("c:/Users/saint/Documents/AMAT132/AMAT_132.csv")
18
19 # Display first few rows
20 head(df)
21
22 # Clean column names (Replace spaces with underscores, and handle other special characters if needed)
23 colnames(df) <- make.names(colnames(df)) # More robust than gsub
24
25 # Convert necessary columns to numeric. Handle potential errors.
26 df$App_High <- as.numeric(as.character(df$App_High)) # Convert to character first to handle
27 df$Mic_High <- as.numeric(as.character(df$Mic_High)) # Convert to character first to handle
28
29 # Convert Date column to proper Date format
30 df$Date <- as.Date(df$Date, format = "%Y") # "%Y" is correct for yearly data
31
32 # Check for missing values
33
```

Console Output:

```
R - R 4.4.2 - ~/R
> head(df)
  <dbl> <dbl> <dbl>
5 1991  0.382  1.01
6 1992  0.401  1.57
> # Clean column names (Replace spaces with underscores, and handle other special characters if needed)
> colnames(df) <- make.names(colnames(df)) # More robust than gsub
> # Convert necessary columns to numeric. Handle potential errors.
> df$App_High <- as.numeric(as.character(df$App_High)) # Convert to character first to handle factors
> df$Mic_High <- as.numeric(as.character(df$Mic_High)) # Convert to character first to handle factors
>
```

Convert Date column to proper Date format

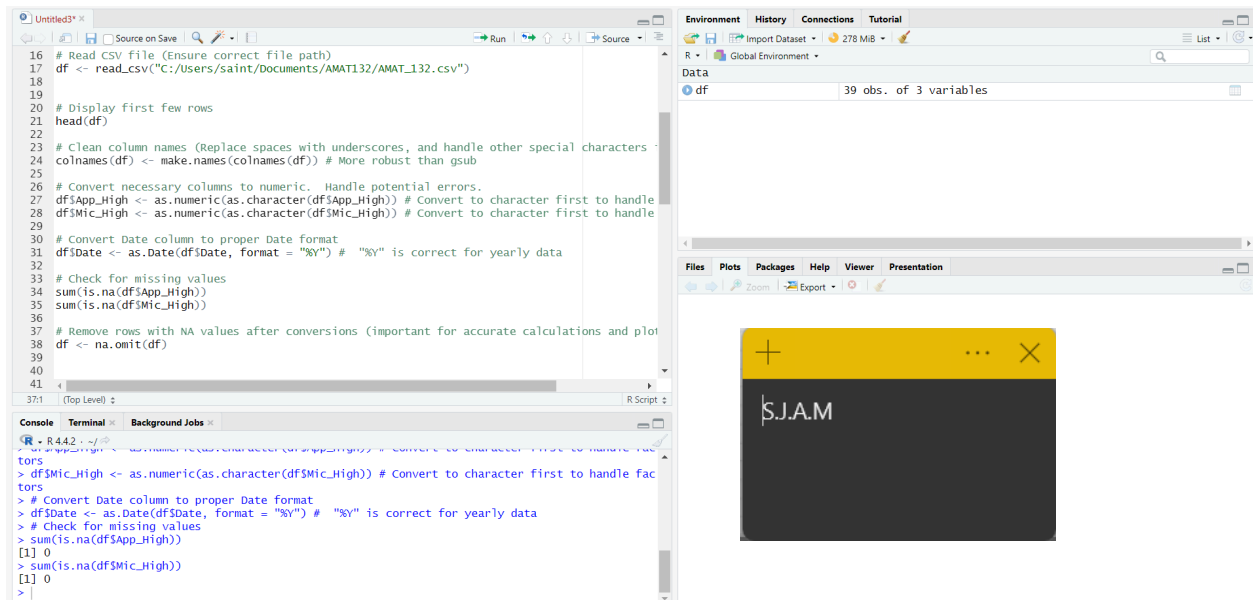


The screenshot shows the RStudio IDE with a script editor on the left and the Environment pane on the right. The script editor contains the following R code:

```
12 library(ggplot2)
13 library(dplyr)
14 library(readr)
15
16 # Read CSV file (Ensure correct file path)
17 df <- read_csv("C:/Users/saint/Documents/AMAT132/AMAT_132.csv")
18
19 # Display first few rows
20 head(df)
21
22 # Clean column names (Replace spaces with underscores, and handle other special characters)
23 colnames(df) <- make.names(colnames(df)) # More robust than gsub
24
25 # Convert necessary columns to numeric. Handle potential errors.
26 df$App_High <- as.numeric(as.character(df$App_High)) # Convert to character first to handle
27 df$Mic_High <- as.numeric(as.character(df$Mic_High)) # Convert to character first to handle
28
29 # Convert Date column to proper Date format
30 df$Date <- as.Date(df$Date, format = "%Y") # "%Y" is correct for yearly data
31
32 # Check for missing values
33 sum(is.na(df$App_High))
34 sum(is.na(df$Mic_High))
35
```

The Environment pane on the right shows the variable 'df' with 39 observations and 3 variables. The console at the bottom shows the execution of the code, including the conversion of the date column and the check for missing values.

Check for missing values



The screenshot shows the RStudio IDE with a script editor on the left and the Environment pane on the right. The script editor contains the following R code:

```
16 # Read CSV file (Ensure correct file path)
17 df <- read_csv("C:/Users/saint/Documents/AMAT132/AMAT_132.csv")
18
19 # Display first few rows
20 head(df)
21
22 # Clean column names (Replace spaces with underscores, and handle other special characters)
23 colnames(df) <- make.names(colnames(df)) # More robust than gsub
24
25 # Convert necessary columns to numeric. Handle potential errors.
26 df$App_High <- as.numeric(as.character(df$App_High)) # Convert to character first to handle
27 df$Mic_High <- as.numeric(as.character(df$Mic_High)) # Convert to character first to handle
28
29 # Convert Date column to proper Date format
30 df$Date <- as.Date(df$Date, format = "%Y") # "%Y" is correct for yearly data
31
32 # Check for missing values
33 sum(is.na(df$App_High))
34 sum(is.na(df$Mic_High))
35
36 # Remove rows with NA values after conversions (important for accurate calculations and plot)
37 df <- na.omit(df)
38
39
40
41
```

The Environment pane on the right shows the variable 'df' with 39 observations and 3 variables. The console at the bottom shows the execution of the code, including the conversion of the date column and the check for missing values.

Remove rows with NA values after conversions (important for accurate calculations and plots) df <- na.omit(df)

The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```
20 # Display first few rows
21 head(df)
22
23 # Clean column names (Replace spaces with underscores, and handle other special characters)
24 colNames(df) <- make.names(colNames(df)) # More robust than gsub
25
26 # Convert necessary columns to numeric. Handle potential errors.
27 df$App_High <- as.numeric(as.character(df$App_High)) # Convert to character first to handle
28 df$Mic_High <- as.numeric(as.character(df$Mic_High)) # Convert to character first to handle
29
30 # Convert Date column to proper Date format
31 df$Date <- as.Date(df$Date, format = "%Y") # "%Y" is correct for yearly data
32
33 # Check for missing values
34 sum(is.na(df$App_High))
35 sum(is.na(df$Mic_High))
36
37 # Remove rows with NA values after conversions (important for accurate calculations and plots)
38 df <- na.omit(df)
39
40
41 # Compute covariance and correlation (after removing NAs)
42 cov_value <- cov(df$App_High, df$Mic_High) # use.complete.obs is now the default
43 cor_value <- cor(df$App_High, df$Mic_High) # use.complete.obs is now the default
44
45
```

The console on the bottom left shows the output of the code:

```
> # Display first few rows
> head(df)
[1] 0
[1] 0
> # Check for missing values
> sum(is.na(df$App_High))
[1] 0
> sum(is.na(df$Mic_High))
[1] 0
> # Remove rows with NA values after conversions (important for accurate calculations and plots)
> df <- na.omit(df)
>
```

The Environment pane on the right shows the data frame 'df' with 39 observations and 3 variables. The Viewer pane at the bottom shows a preview of the data frame with a yellow header and a dark body containing the text 'S.J.A.M'.

Compute covariance and correlation (after removing NAs)

The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```
24 colNames(df) <- make.names(colNames(df)) # More robust than gsub
25
26 # Convert necessary columns to numeric. Handle potential errors.
27 df$App_High <- as.numeric(as.character(df$App_High)) # Convert to character first to handle
28 df$Mic_High <- as.numeric(as.character(df$Mic_High)) # Convert to character first to handle
29
30 # Convert Date column to proper Date format
31 df$Date <- as.Date(df$Date, format = "%Y") # "%Y" is correct for yearly data
32
33 # Check for missing values
34 sum(is.na(df$App_High))
35 sum(is.na(df$Mic_High))
36
37 # Remove rows with NA values after conversions (important for accurate calculations and plots)
38 df <- na.omit(df)
39
40
41 # Compute covariance and correlation (after removing NAs)
42 cov_value <- cov(df$App_High, df$Mic_High) # use.complete.obs is now the default
43 cor_value <- cor(df$App_High, df$Mic_High) # use.complete.obs is now the default
44
45 # Print results
46 cat("Covariance:", cov_value, "\nCorrelation:", cor_value, "\n")
47
48 # SCATTER PLOT: Relationship between Apple and Microsoft High Prices
49
```

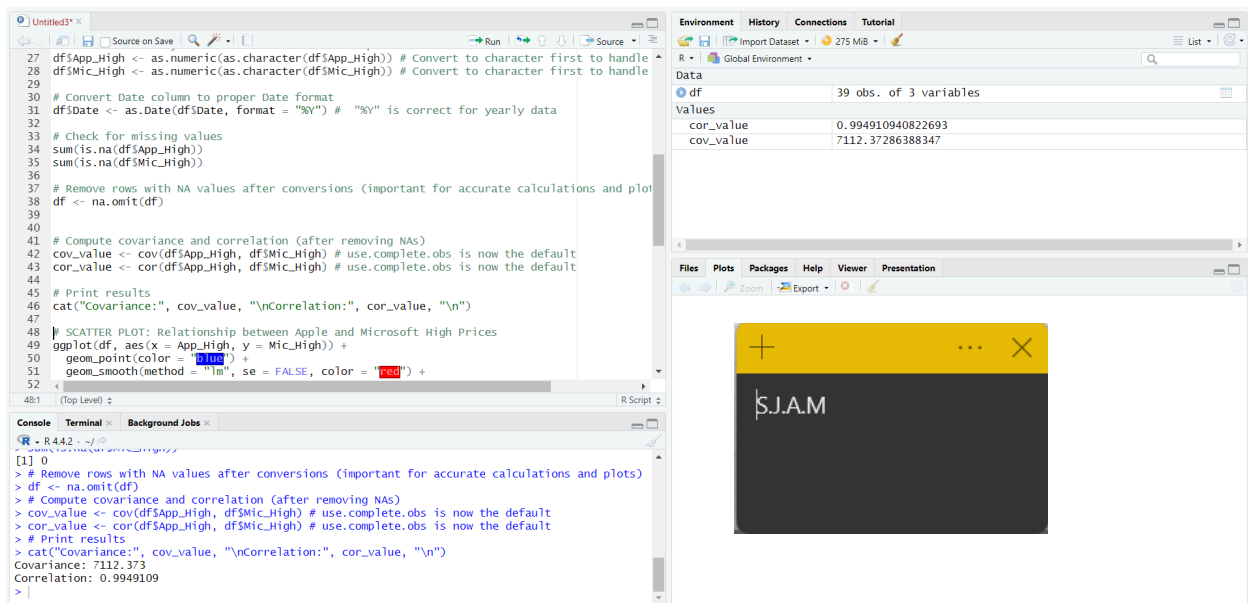
The console on the bottom left shows the output of the code:

```
> # Check for missing values
> sum(is.na(df$App_High))
[1] 0
> sum(is.na(df$Mic_High))
[1] 0
> # Remove rows with NA values after conversions (important for accurate calculations and plots)
> df <- na.omit(df)
> # Compute covariance and correlation (after removing NAs)
> cov_value <- cov(df$App_High, df$Mic_High) # use.complete.obs is now the default
> cor_value <- cor(df$App_High, df$Mic_High) # use.complete.obs is now the default
>
```

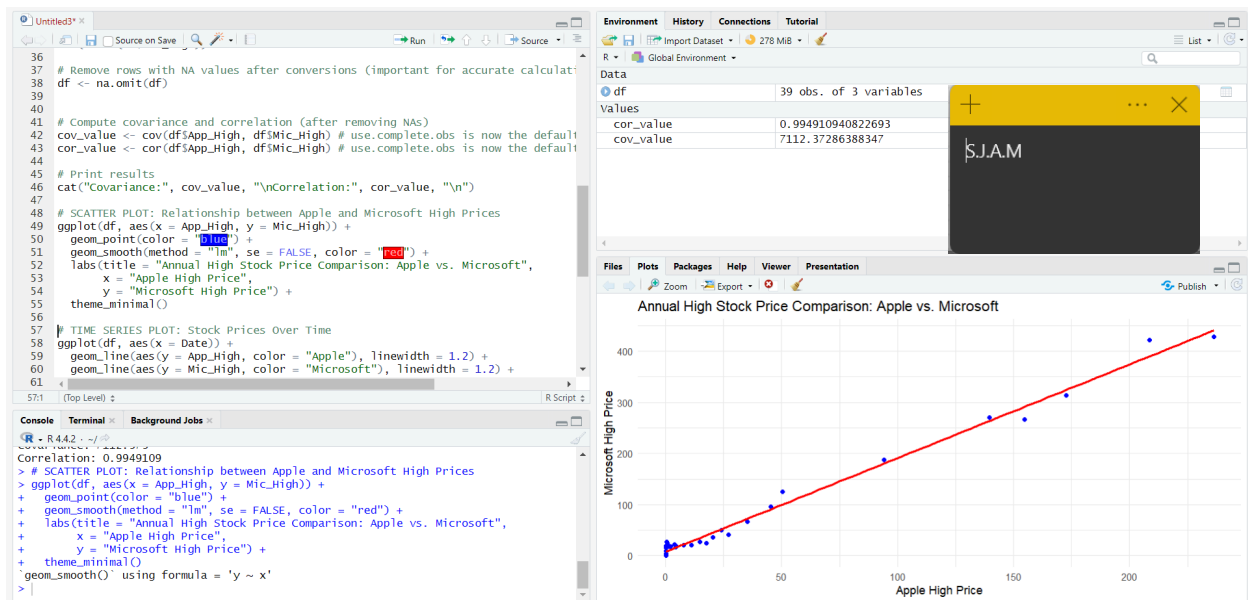
The Environment pane on the right shows the data frame 'df' with 39 observations and 3 variables. The Viewer pane at the bottom shows a preview of the data frame with a yellow header and a dark body containing the text 'S.J.A.M'.

Values	
cor_value	0.994910940822693
cov_value	7112.37286388347

Printing results for covariance and correlation



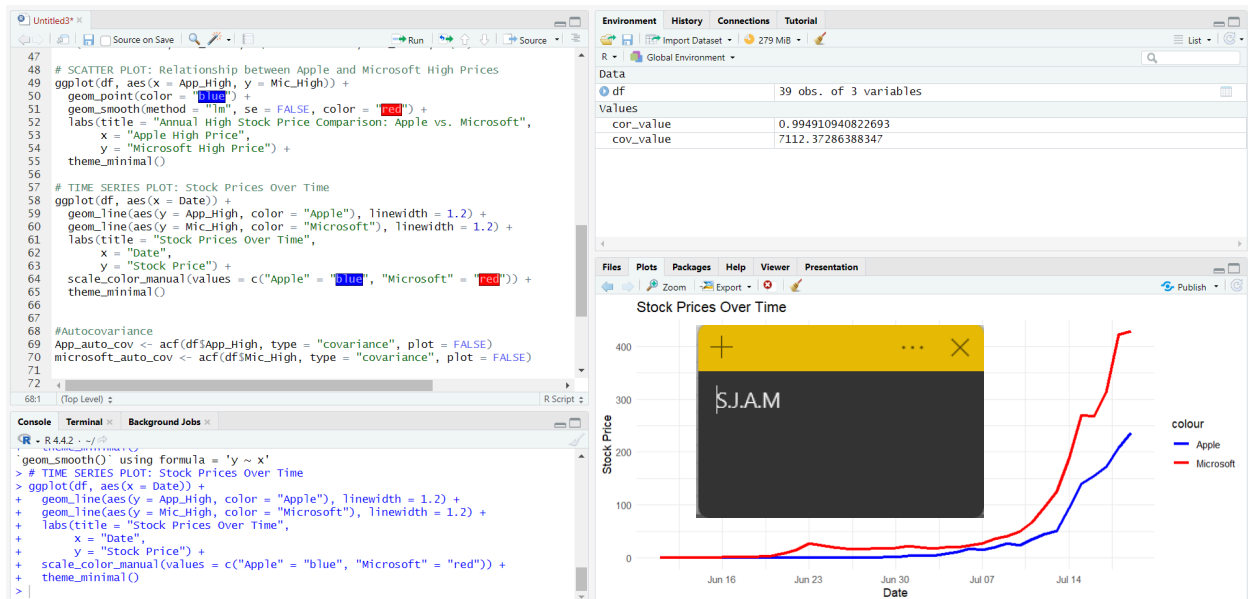
SCATTER PLOT: Relationship between Apple and Microsoft High Prices



The scatter plot shows how the high stock prices of Apple and Microsoft relate to each other, and it reveals a strong positive correlation. This means that when Apple's stock price goes up, Microsoft's tends to rise as well. You can see this trend in the upward slope of the data points, which aligns closely with the red regression line. This line indicates a linear relationship between the two companies' stock prices. There are a few points that stand out as outliers, indicating moments when one company's stock behaved differently—these could be due to specific market factors at play. Overall, the plot suggests

that Apple and Microsoft's stock prices generally move in tandem, likely influenced by larger market trends or conditions in the tech industry.

TIME SERIES PLOT: Stock Prices Over Time

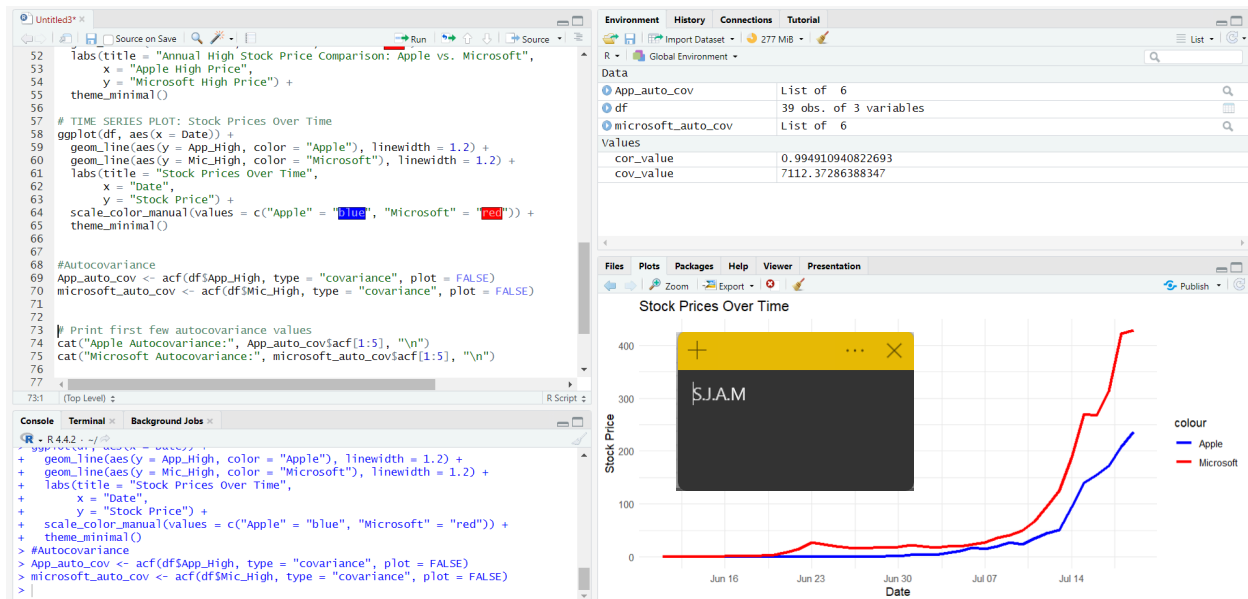


The time series plot offers a snapshot of Apple and Microsoft's stock prices over time. At first, both stocks show a fairly steady trend with just a few ups and downs. However, things take off dramatically in early July. During this spike, Microsoft (shown by the red line) has a more significant surge compared to Apple (shown by the blue line), which suggests that Microsoft is experiencing stronger growth or perhaps more volatility.

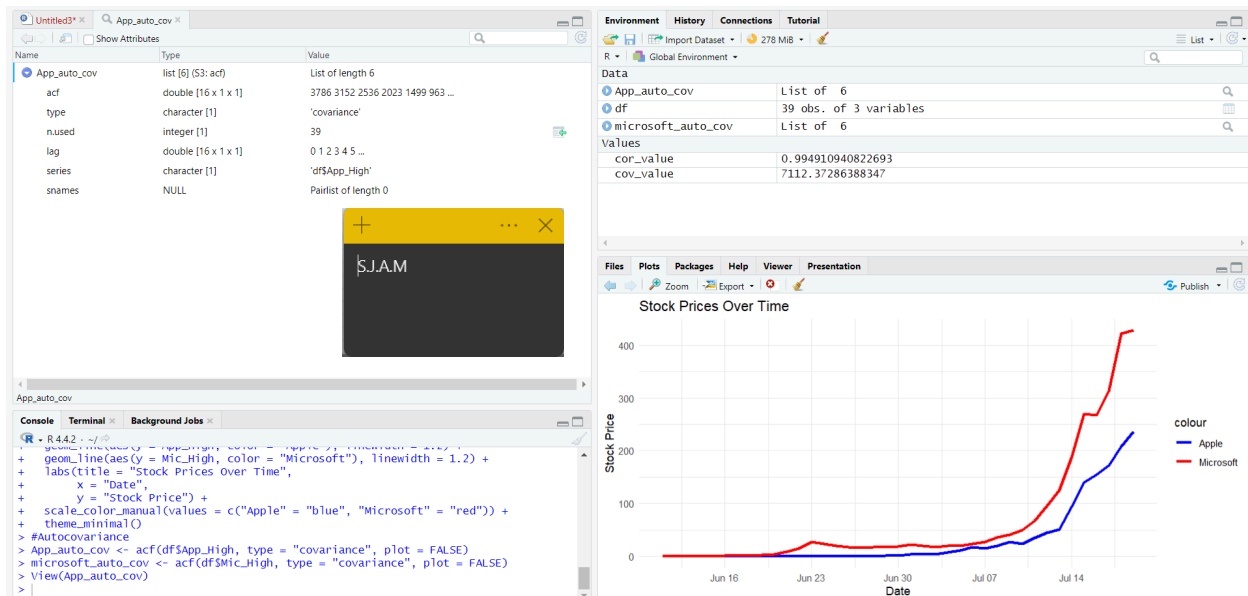
Interestingly, both companies share a similar upward trend, indicating that their stock prices are likely responding to similar market influences. However, it's clear that Microsoft's stock is climbing at a quicker pace. This pattern might be due to various external factors, like earnings reports, developments within their industries, or broader market movements that are impacting both tech giants.

Exercise 2.b Interpret Results

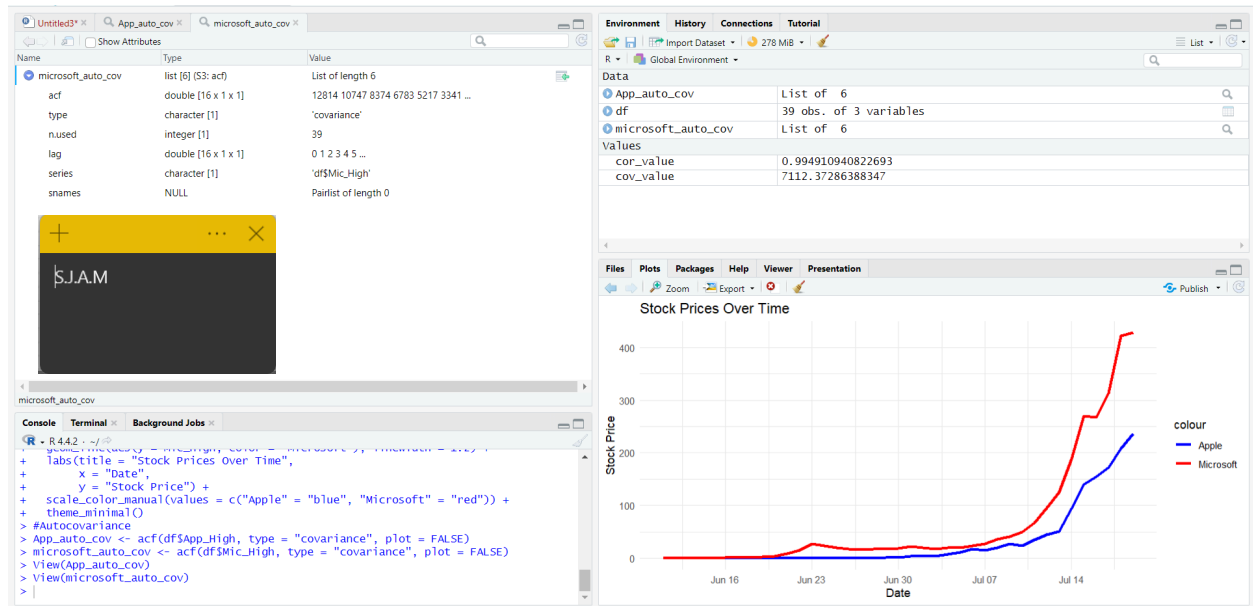
Compute the autocovariance (per lag) for each time series and plot the Autocorrelation Function (ACF) using the code below. Edit the field names when necessary. Interpret results.



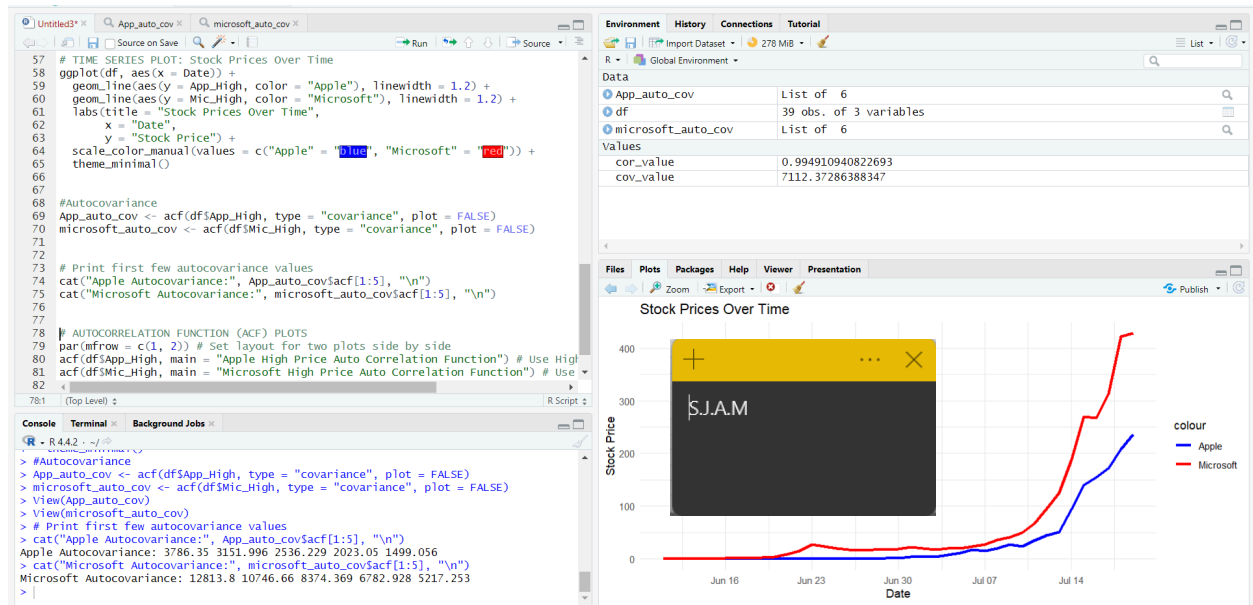
Apple auto covariance



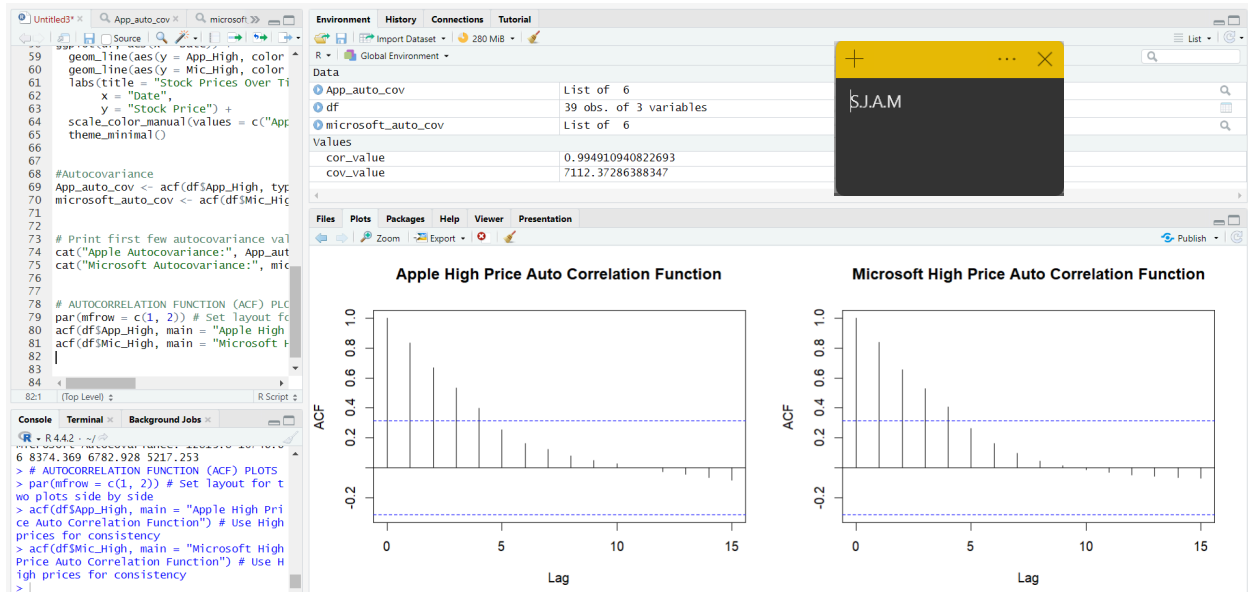
Microsoft auto covariance



Print first few autocovariance values



AUTOCORRELATION FUNCTION (ACF) PLOTS



The **Autocorrelation Function (ACF)** plots for the high stock prices of Apple and Microsoft reveal an interesting connection between past and future prices over time. Starting at a high positive value close to 1 at lag 0, the ACF for both stocks gradually decreases as the lag increases. This means there's a strong relationship between the current stock prices and their recent past values, showing that what happened before significantly affects future prices. However, this influence does weaken over time. The decreasing bar heights suggest that while past prices still play a role in shaping future prices, their impact diminishes as time goes on.

The dashed blue lines you see represent confidence intervals, and any ACF values that fall outside of these lines are statistically significant. For both Apple and Microsoft, the ACF values are significant up to around lag 5, after which they taper off toward zero. This indicates that both stocks have a sort of short-term memory—past prices influence future prices for a few time periods before their effects fade away.

In addition, both stocks have a rising trend, with Microsoft's stock price growing at a more rapid pace than Apple's, especially in the later dates. Both stocks initially stayed relatively consistent at lower price points, but as time went on, the prices of Microsoft rose more rapidly than those of Apple.

With respect to **autocovariance**, the study shows that both Microsoft and Apple stock prices are highly dependent on their immediate past values, as shown by high autocovariance at low lags. Microsoft's autocovariance values are, however, significantly higher than those of Apple, implying higher volatility and price fluctuations. As the lag gets larger, the autocovariance values decrease slowly, showing that past stock prices have decreasing effects on future prices for longer lags. This implies that although both shares exhibit short-term momentum, price movements of Microsoft are stronger and indicate greater general price swings.