ISA 444: Business Forecasting

04: Time Series EDA

Fadel M. Megahed, PhD

Endres Associate Professor
Farmer School of Business
Miami University

@FadelMegahed
fmegahed
fmegahed@miamioh.edu
Automated Scheduler for Office Hours

Fall 2023

Quick Refresher from Last Class

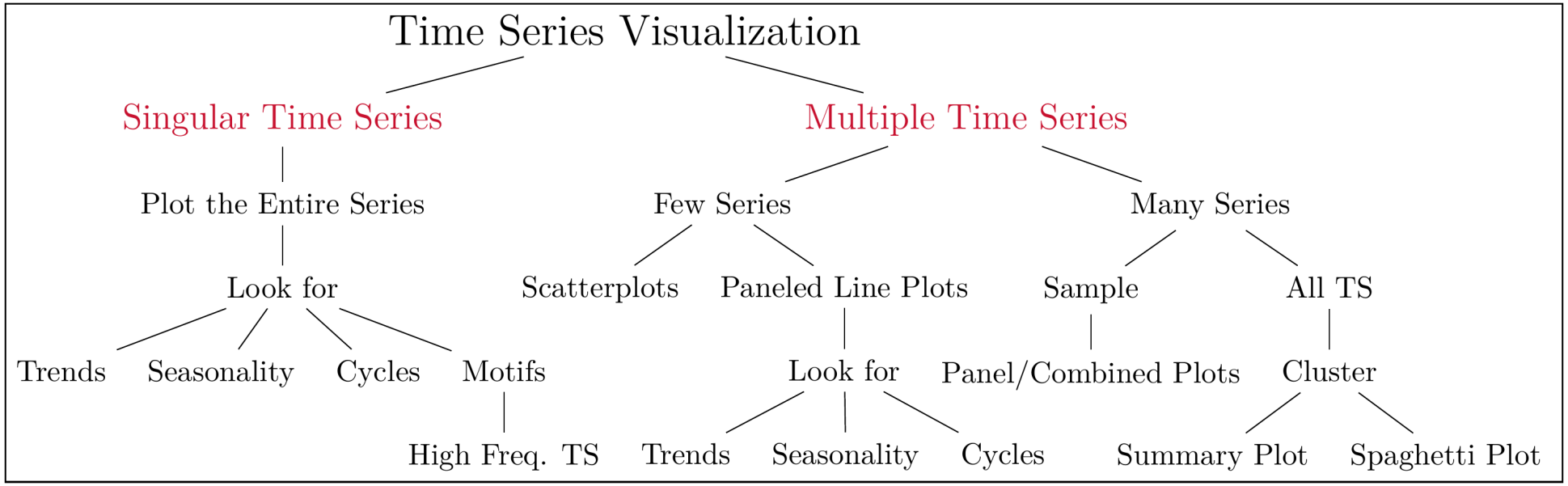
- ✅ Read CSV files in both R and Python.
- ✅ Construct line charts and seasonality plots in both R and Python.
- ✅ Utilize the project workflow in R and create R and Python scripts.
- ❌ Access, subset, and create `ts()` objects in R.

Learning Objectives for Today's Class

- Examine the goals of utilizing line charts in time-series analysis (i.e., detect trends, seasonality, and cycles).
- Develop a deeper understanding of the grammar of graphics, which we used to create time series plots in  and  .
- Use numerical summaries to describe a time series.
- Explain what do we mean by correlation.

A Taxonomy of Time Series Plots and their Interpretation

A Structured Approach for Time Series Viz

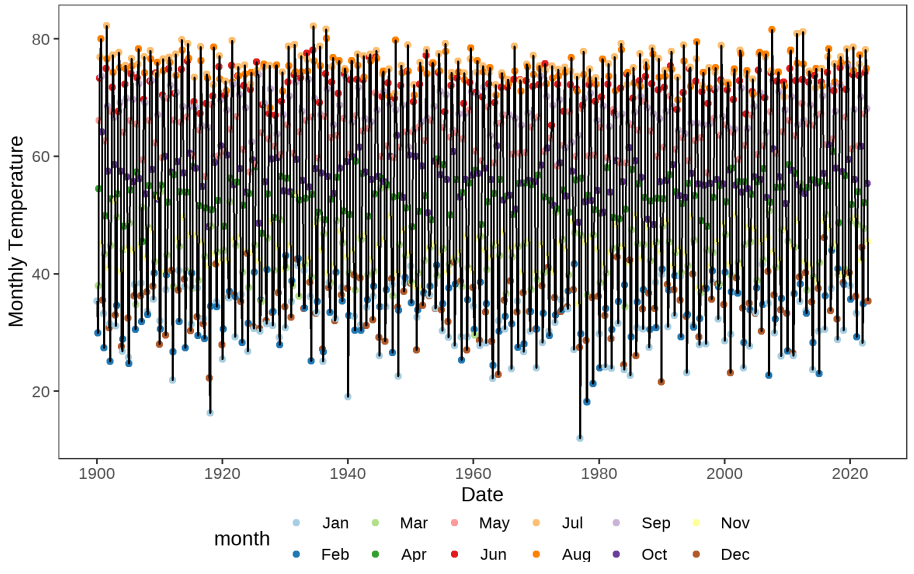


A Potential Framework for Time Series Visualization

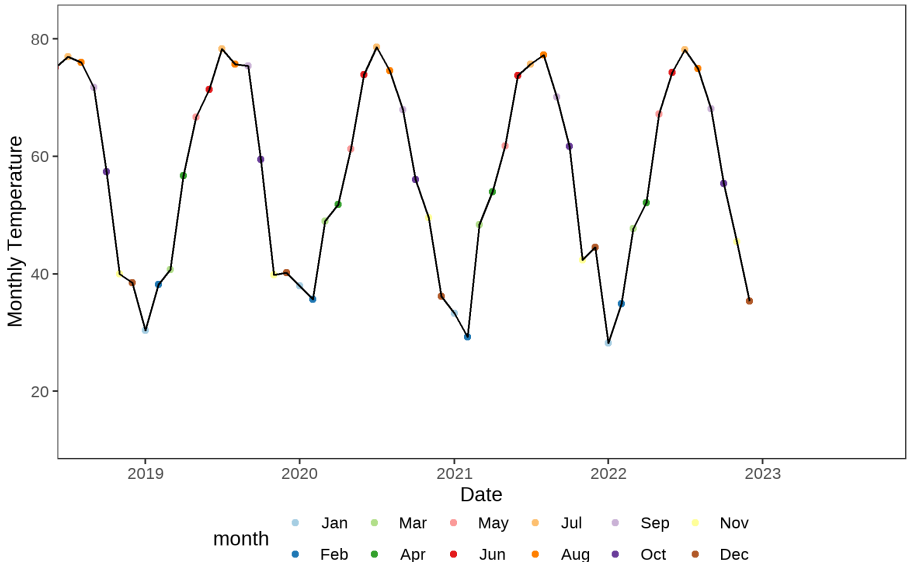
This is my best attempt to improve on the general advice provided in the previous slide. Many of the suggestions, presented in this flow chart, stem from my past and current research/consulting collaborations.

They are by no means a comprehensive list of everything that you can do.

The Line Chart



A plot of a long time-series for monthly weather in Cincinnati, with color denoting different months.



A snippet of the time-series (last 5 full years) for monthly weather in Cincinnati, with color denoting different months.

The Line Chart: Practical Considerations

Things to Consider: (Insert below)

- **Format your data:** ....
- **Entire time-series vs a snippet:** ....
- **On the use of color:** ....
- **On grouping the data:** ....

On the Interpretation of Line Charts

05:00

Activity | Book Stores | GDP 1 | GDP 2 | Key Points

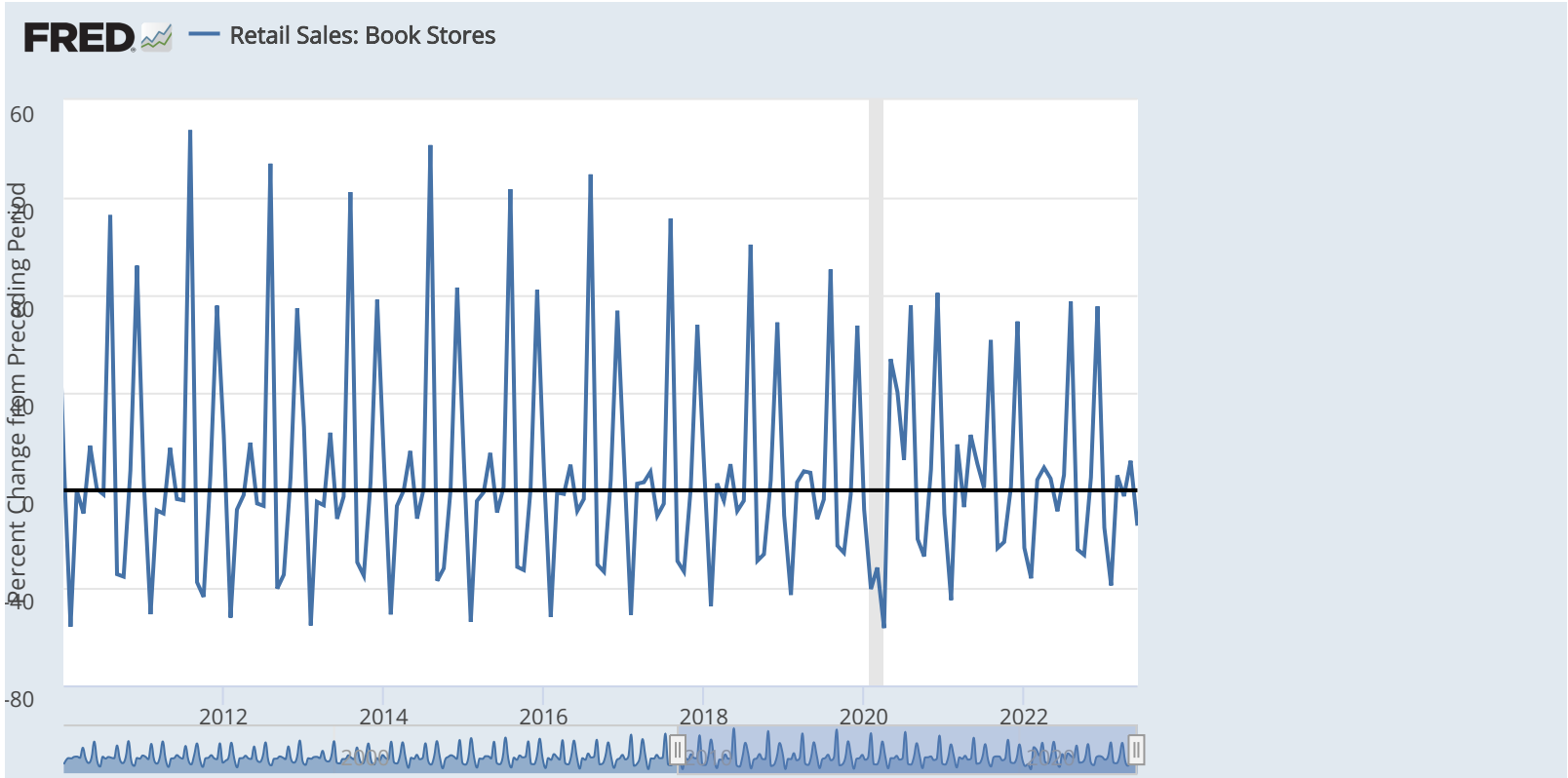
> Over the next 5 minutes, please identify what you have learned from the charts in each tab.

- Write down your answers in the last tab (it is editable).
- Discuss your answers with your neighboring classmates.
- Be prepared to share these answers with class.

On the Interpretation of Line Charts

05:00

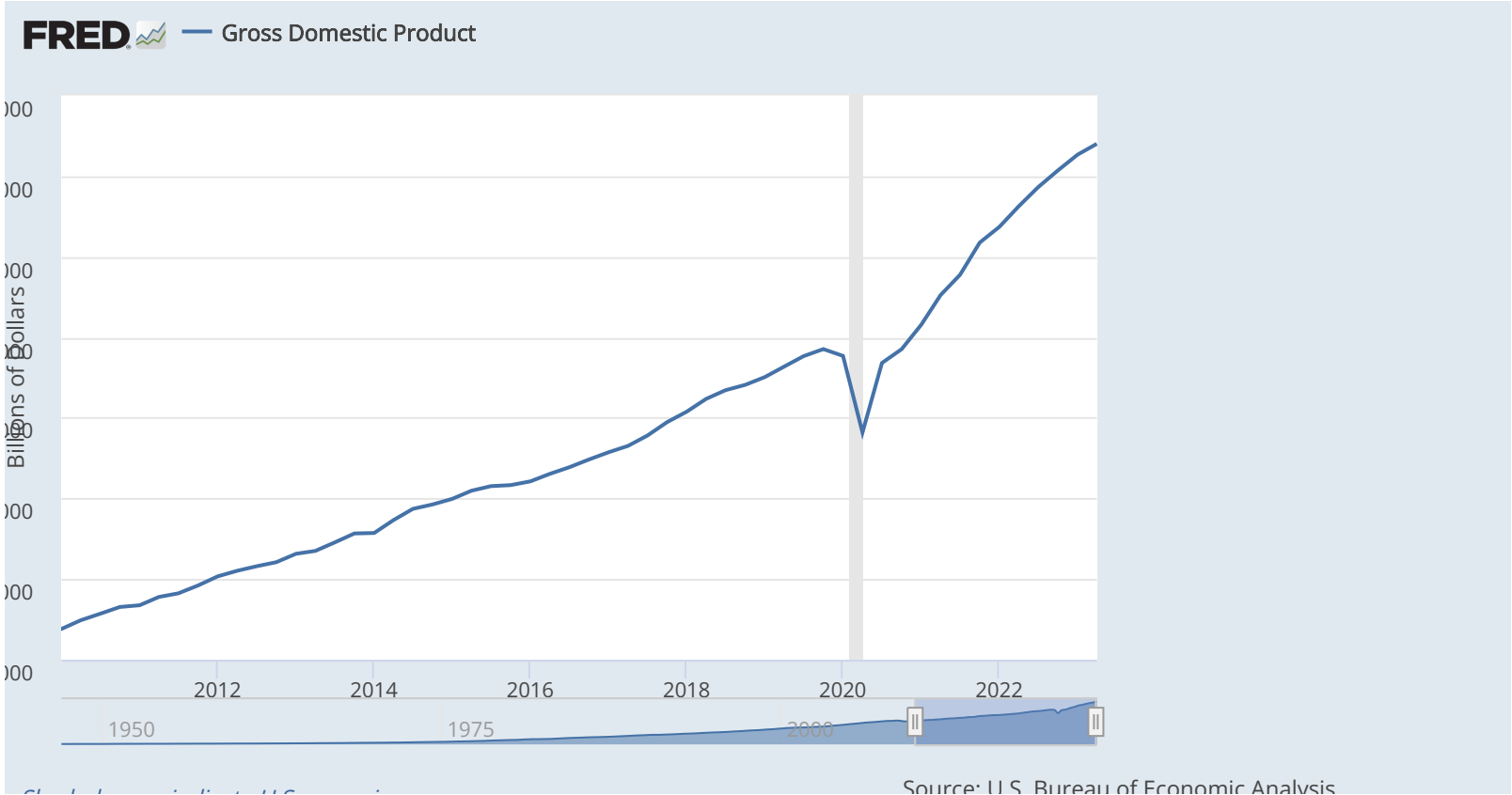
Activity | **Book Stores** | GDP 1 | GDP 2 | Key Points



On the Interpretation of Line Charts

05:00

Activity | Book Stores | GDP 1 | GDP 2 | Key Points



On the Interpretation of Line Charts

05:00

Activity | Book Stores | GDP 1 | GDP 2 | Key Points

On the Interpretation of Line Charts

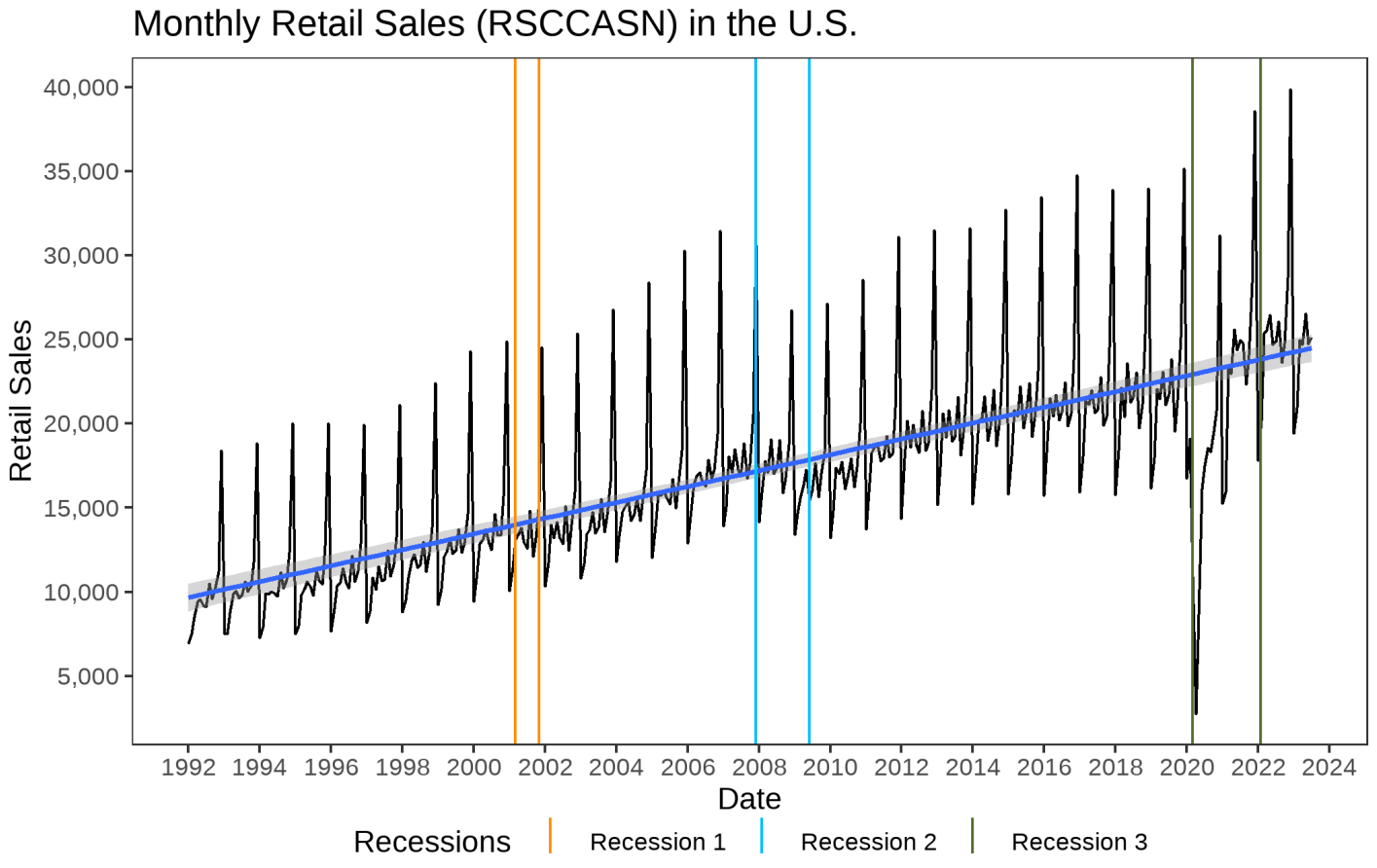
05:00

Activity | Book Stores | GDP 1 | GDP 2 | Key Points

Main Insight(s): (Insert below)

- **Book Stores:** Trend: ... | Seasonality: ... | Cycle: ...
- **GDP 1:** Trend: ... | Seasonality: ... | Cycle: ...
- **GDP 2:** Trend: ... | Seasonality: ... | Cycle: ...

Need Assistance with Trends

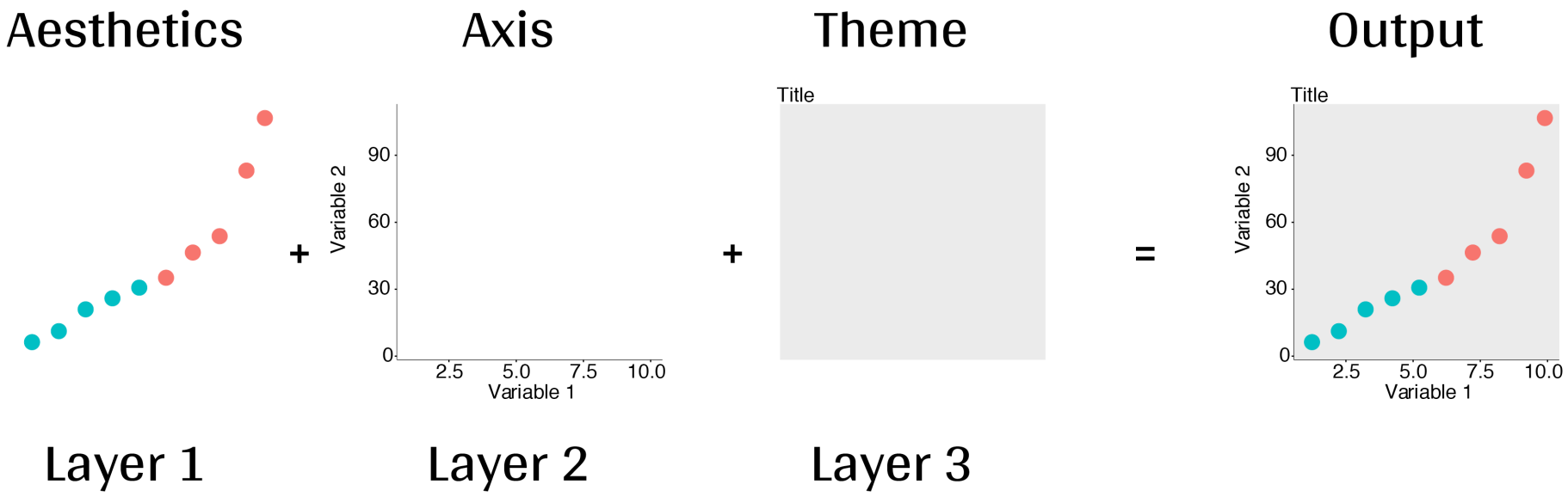


Need Assistance with Seasonality

In Section 2.2.1 of our reference book, the authors presented two approaches for considering seasonality. We can replicate them easily in R and Python. Refer to the discussion in the next section for more detail.

The Grammar of Graphics and the `ggplot2` package

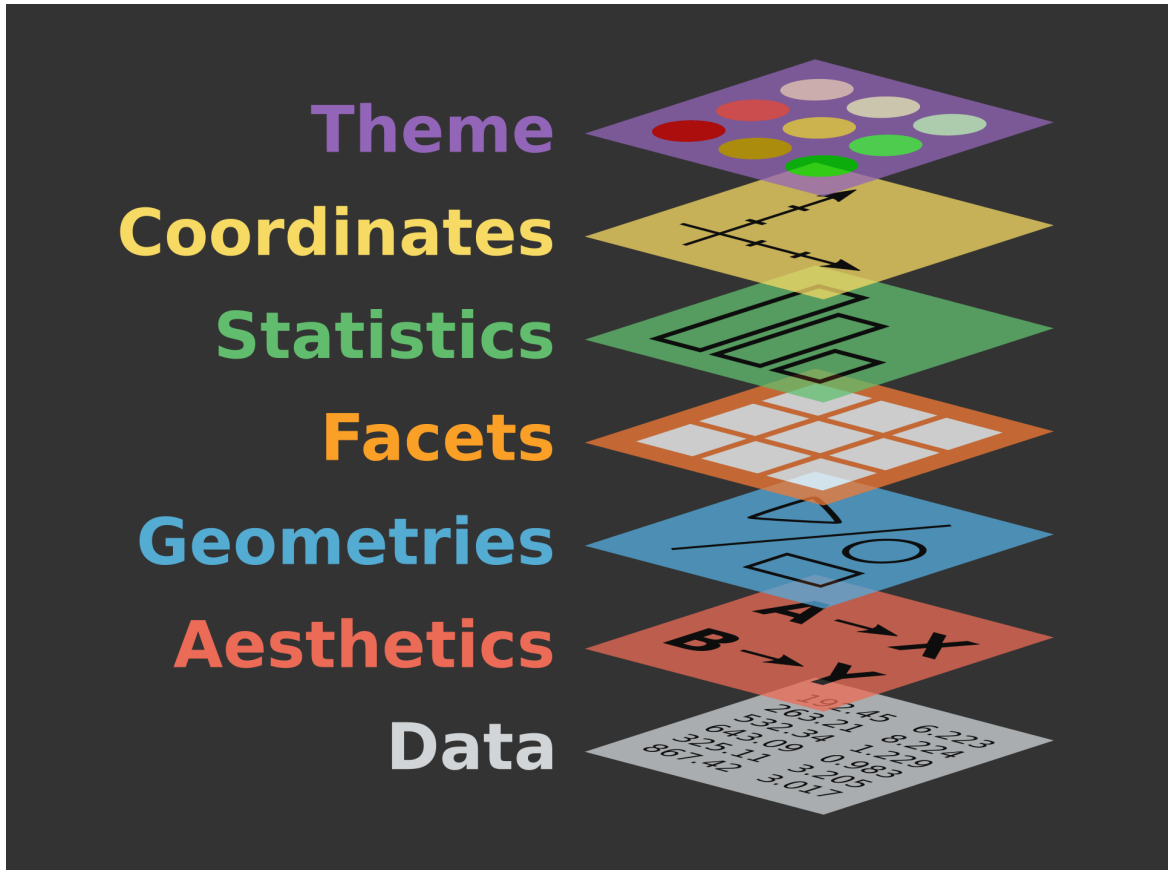
A Visual Introduction to Graph Layers



Schematic of some distinct layers in the Grammar of Graphics.

Figure from the QCBS R Workshop Series. It is from Chapter 05 of their third workshop.

The Grammar of Graphics



An overview of the layers introduced in the Grammar of Graphics.

Figure from the QCBS R Workshop Series. It is from Chapter 05 of their third workshop.

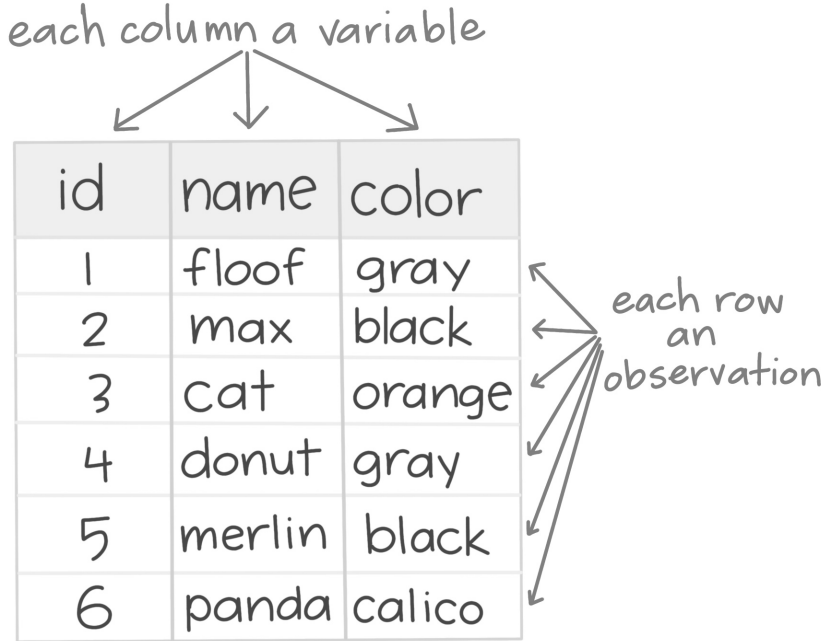
Grammar of Graphics Layers: Data

- Data needs to be in a **tidy** format (see next two slides).
- The dplyr and tidyr 📦 can help with `tidying` your data.

> "TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."
>
> —HADLEY WICKHAM

In tidy data:

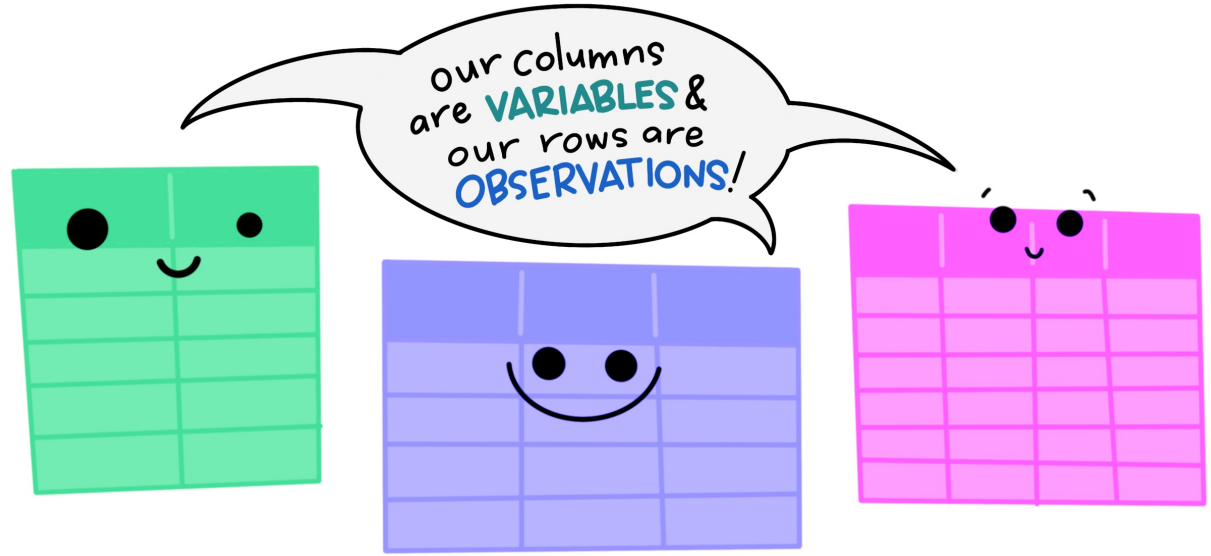
- each variable forms a column
- each observation forms a row
- each cell is a single measurement



Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

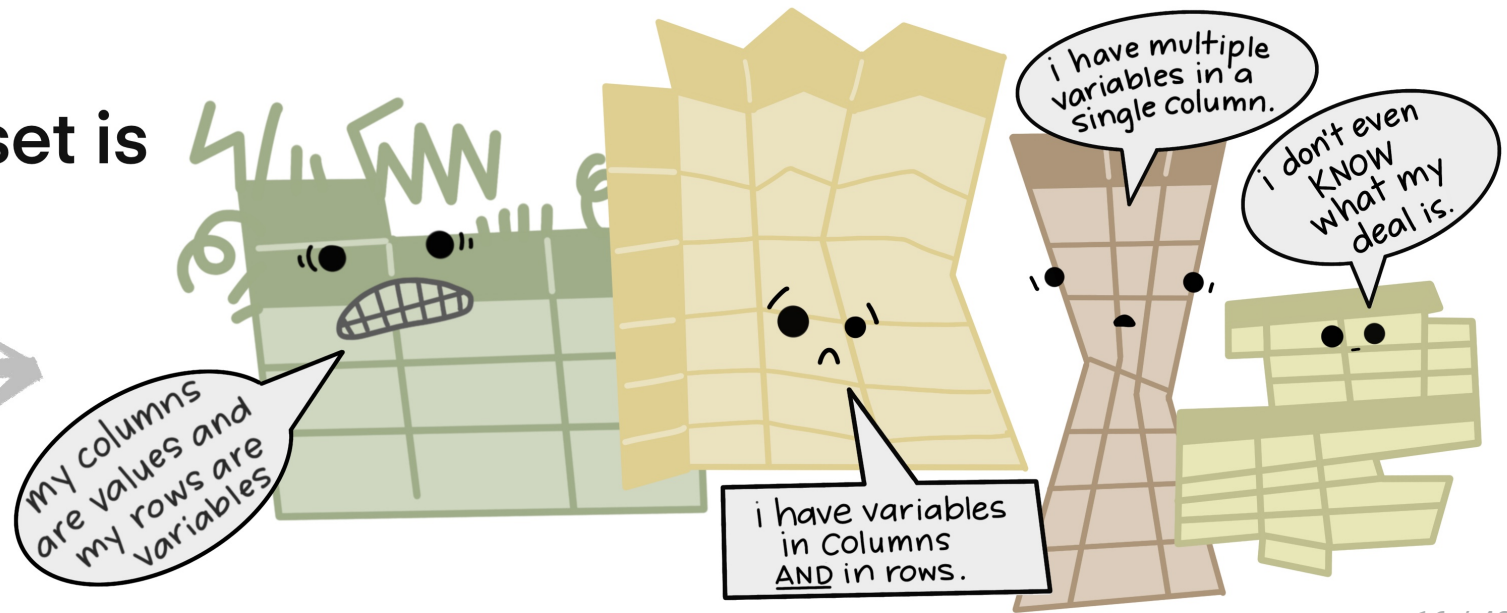
Source: Illustration is from the Openscapes blog Tidy Data for reproducibility, efficiency, and collaboration by Julia Lowndes and Allison Horst

The standard structure of tidy data means that "tidy datasets are all alike..."



"...but every messy dataset is messy in its own way."

–HADLEY WICKHAM



Source: Illustration is from the Openscapes blog Tidy Data for reproducibility, efficiency, and collaboration by Julia Lowndes and Allison Horst

Grammar of Graphics Layers: Aesthetics

Aesthetics (`ggplot2::aes()`) are used to make data visible. For example:

- `x`, `y`: variable to be plotted along the x and y axes.
- `color`: color of geoms (i.e., points, lines, etc) according to the data.
- `fill`: the inside color of the geom (useful for bar charts).
- `group`: what group a geom belongs to (useful in multiple ts).
- `shape`: the shape of the plotted point (circle, triangle, filled circle, etc).
- `linetype`: the type of line used (solid, dashed, etc).
- `size`: size scaling for an extra dimension.
- `alpha`: the transparency of the geom

Identify the Aesthetics Used in the Charts

06:00

Activity | Line Chart 1 | Line Chart 2 | Seasonal Chart 1 | Seasonal Chart 2

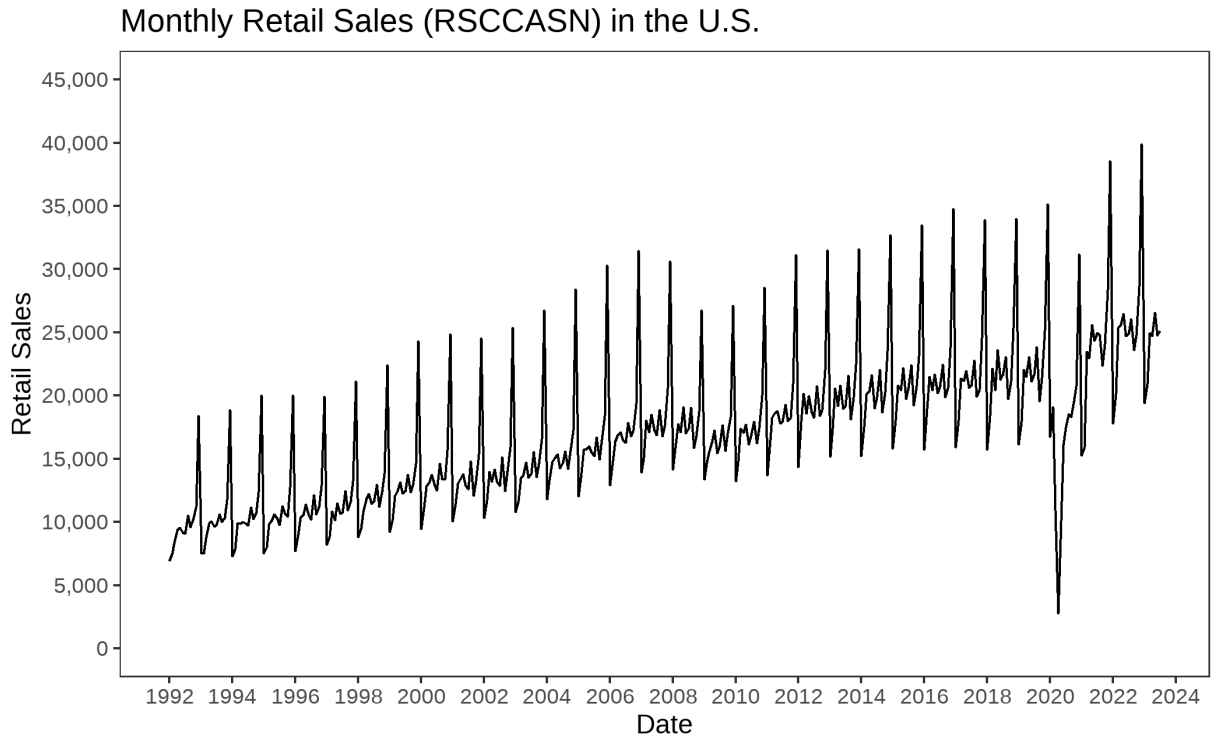
> Over the next 6 minutes, please identify the aesthetics used in each chart.

- Write down your answers in the right-side of each tab (it is editable).
- You can discuss your answers with your neighboring classmates.
- Be prepared to share these answers with class.

Identify the Aesthetics Used in the Charts

06:00

Activity | **Line Chart 1** | Line Chart 2 | Seasonal Chart 1 | Seasonal Chart 2



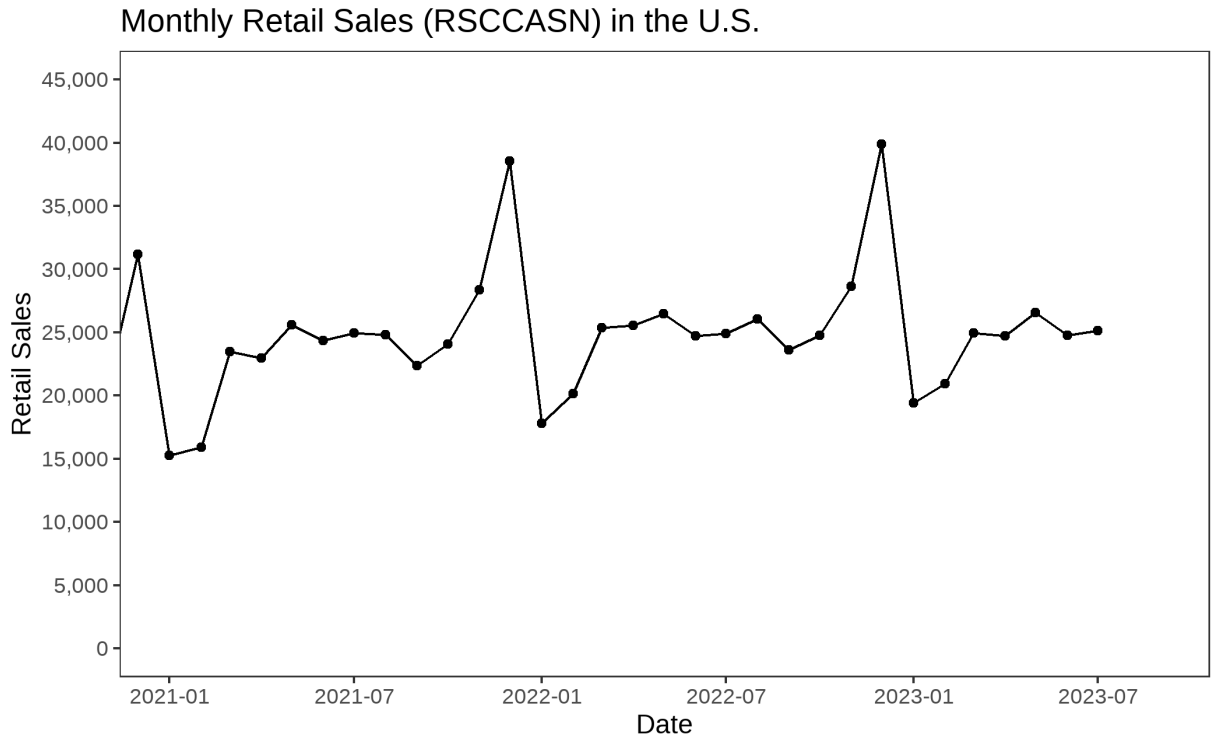
Main Aesthetics: (Insert below)

- `x`: .... and its class is: ....
- `y`: .... and its class is: ....
- `group`: .....
- `color`: .....

Identify the Aesthetics Used in the Charts

06:00

Activity | Line Chart 1 | **Line Chart 2** | Seasonal Chart 1 | Seasonal Chart 2



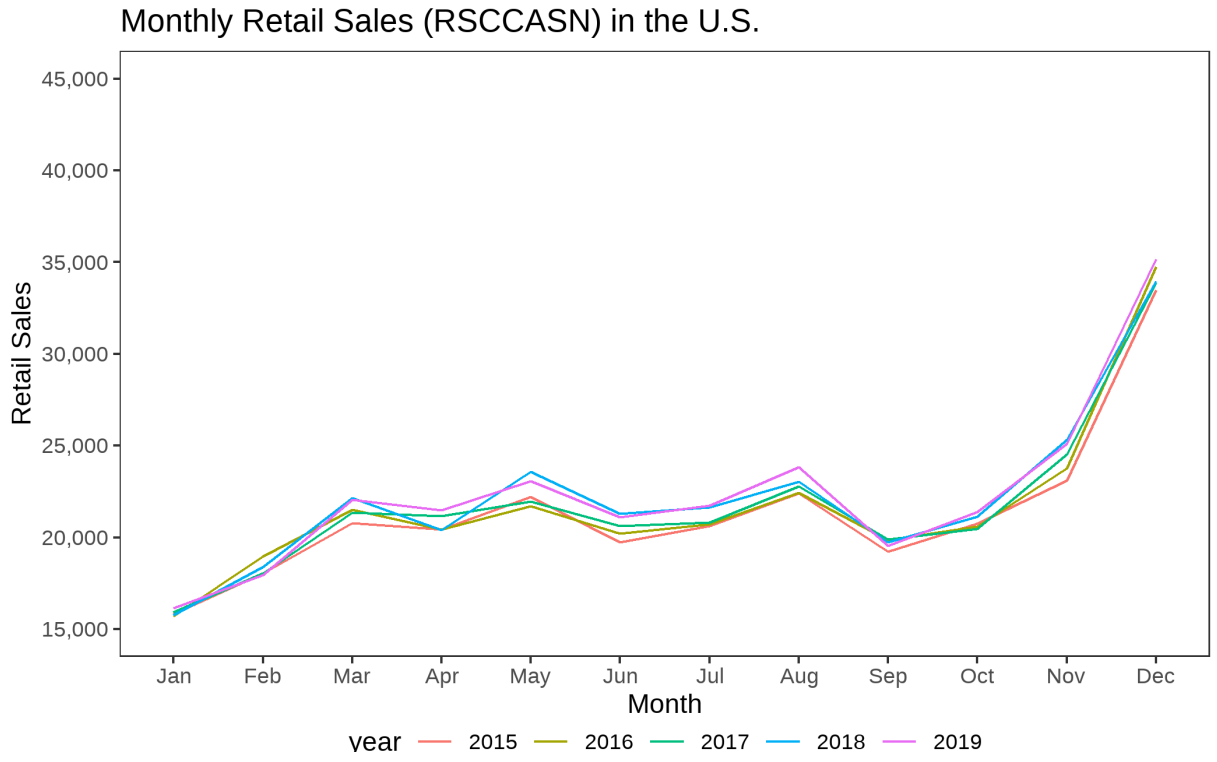
Main Aesthetics: (Insert below)

- `x`: .... and its class is: ....
- `y`: .... and its class is: ....
- `group`: .....
- `color`: .....

Identify the Aesthetics Used in the Charts

06:00

Activity | Line Chart 1 | Line Chart 2 | **Seasonal Chart 1** | Seasonal Chart 2



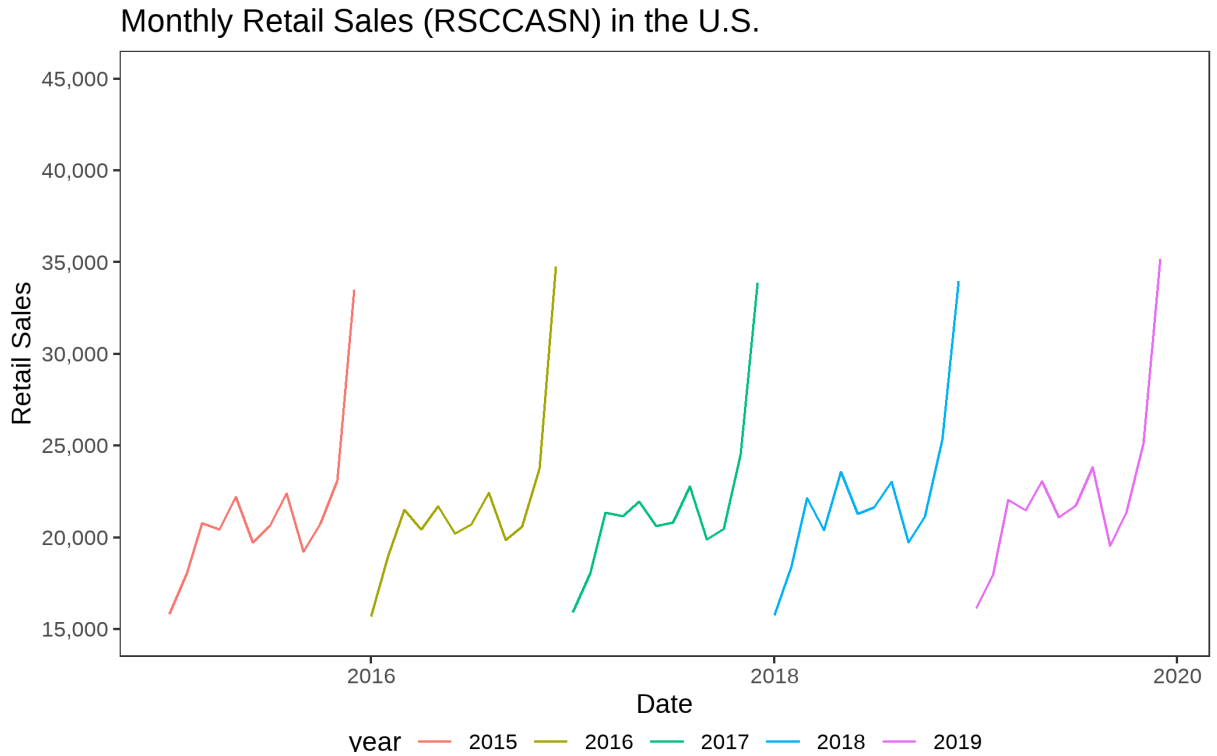
Main Aesthetics: (Insert below)

- `x`: .... and its class is: ....
- `y`: .... and its class is: ....
- `group`: .....
- `color`: .....

Identify the Aesthetics Used in the Charts

06:00

Activity | Line Chart 1 | Line Chart 2 | Seasonal Chart 1 | **Seasonal Chart 2**



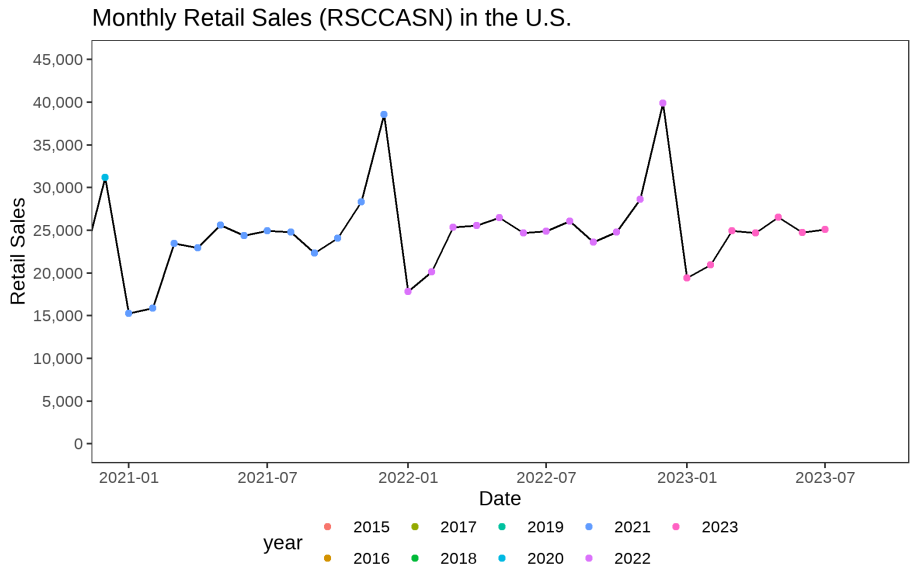
Main Aesthetics: (Insert below)

- `x`: .... and its class is: ....
- `y`: .... and its class is: ....
- `group`: .....
- `color`: .....

Grammar of Graphics Layers: Aesthetics

- Assigned **globally** to the entire plot via `ggplot2::ggplot(ggplot2::aes())`, or to **specific geoms** (e.g., `ggplot2::geom_point(ggplot2::aes())`.

The color is passed globally through ggplot(aes(x=date, y=price, color=year)).



The color is passed as an argument within the layer as geom_point(aes(color = year)).

Grammar of Graphics Layers: Individual Geoms

Geometric objects i.e., geoms help determine the type of plot. In this class, we will typically use one or more of the following *geoms*:

- `ggplot2::geom_point()`: scatterplot or points in a line graph.
- `ggplot2::geom_line()`: lines connecting points by increasing value of x.
- `ggplot2::geom_smooth()`: to fit a function line (e.g., linear regression line) based on data.

Grammar of Graphics Layers: Facets

We can use `ggplot2::facet_wrap()` to create small multiples based on a single variable. Arguments for `ggplot2::facet_wrap()` include:

- `facets` which takes the variable of interest in quotes (i.e., `facets = 'symbol'`);
- `nrow` and/or `ncol` which take numeric inputs for the number of rows and columns; and
- `scales`, where we typically use `free_y` to denote that different `ylim` for each panel.

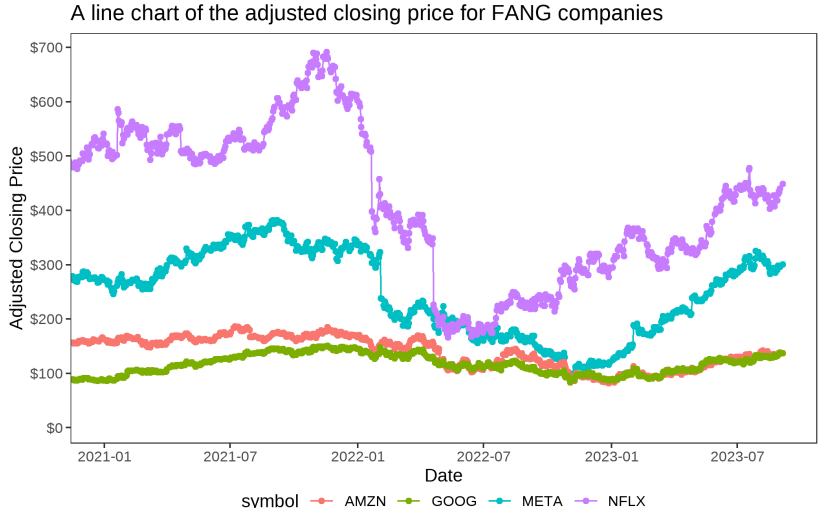
```
fang_df = tidyquant::tq_get(
  x = c('META', 'AMZN', 'NFLX', 'GOOG'),
  from = '2010-01-01', to = Sys.Date()
)

colnames(fang_df)
```

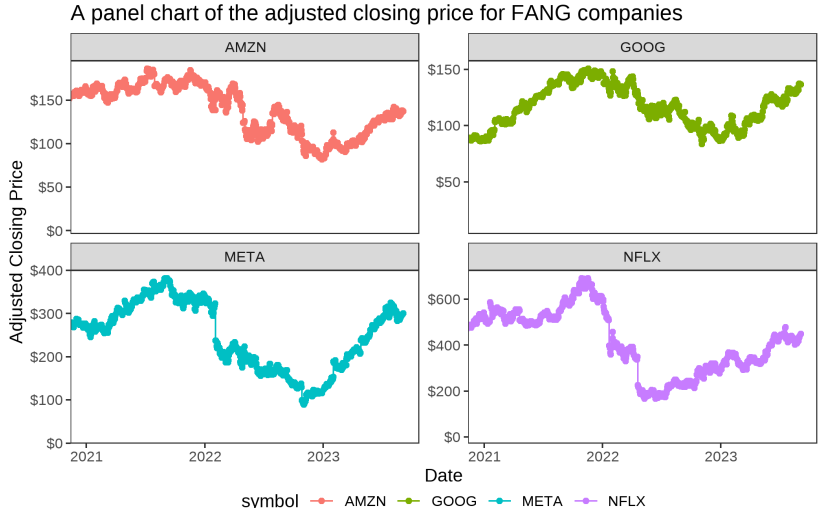
```
## [1] "symbol"   "date"     "open"     "high"     "low"      "close"    "volume"
## [8] "adjusted"
```

Grammar of Graphics Layers: Facets

We can use `ggplot2::facet_wrap()` to create small multiples based on a single variable. Arguments for `ggplot2::facet_wrap()` include:

- `facets` which takes the variable of interest in quotes (i.e., `facets = 'symbol'`);
- `nrow` and/or `ncol` which take numeric inputs for the number of rows and columns; and
- `scales`, where we typically use `free_y` to denote that different `ylim` for each panel.





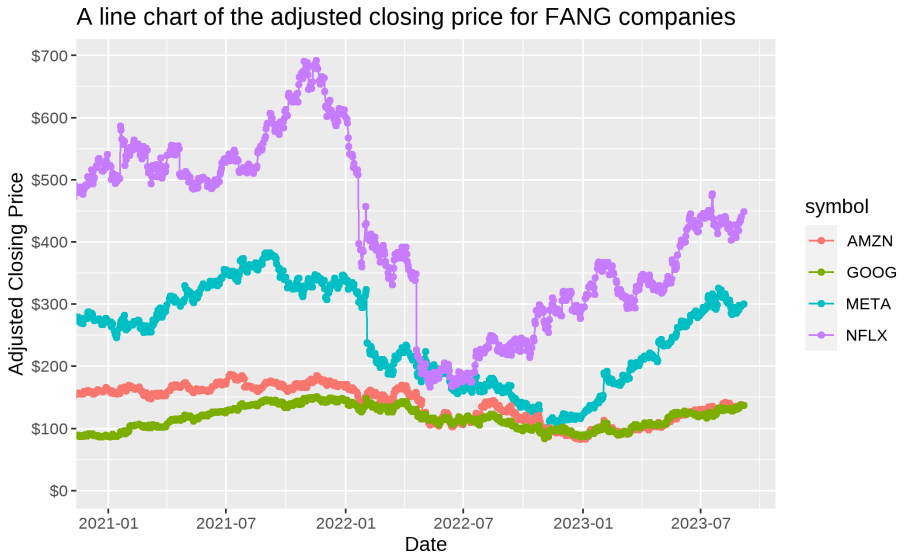
Grammar of Graphics Layers: Coordinates

In class, we will use the following two functions to create **snapshots** of the data:

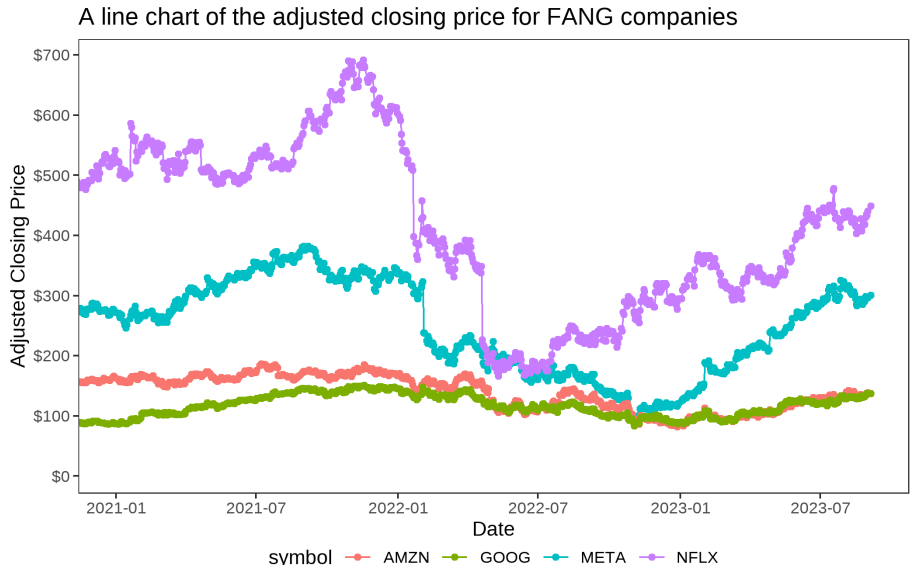
- `ggplot2::coord_cartesian()` to set limits. We will specicially use its xlim argument to create a snapshot when the x axis contains a continous variable (e.g., year). See ggplot2 documentation for more detail.
- `tidyquant::coord_x_date()` to set limits. We will specicially use its xlim argument to create a snapshot when the x axis contains a date variable (with a `class` of date). See tidyquant documentation for more detail.

Grammar of Graphics: Themes

Themes control the overall visual defaults. There are some themes built within the `ggplot2` 📦 (see the complete themes guide). For additional themes, please feel free to play with the ggthemes 📦.



Default ggplot2 theme



A modified theme (no gridlines, no gray background and caption is place below).

Putting it all together

A Singular TS: AAPL's Adj. Close

```
if(require(tidyquant)==F) install.packages("tidyquant") # install if needed
if(require(tidyverse)==F) install.packages('tidyverse') # install if needed

aapl = # get AAPL stock data from 1st trading day after Jan 1, 2020 to now
  tidyquant::tq_get(x = 'AAPL', from = '2020-01-01', to = Sys.Date() ) |>
  # select (i.e., keep) only the variables below
  dplyr::select( c(date, symbol, adjusted) ) |>
  # create the following variables: year and month
  dplyr::mutate(
    # date has to be of class Date if not use lubridate::ymd (mdy, dmy, etc)
    # to convert the string variable to date
    year = lubridate::year(date),
    month = lubridate::month(date, label = T)
  )
tail(aapl, n = 1) # print the last obs  to see what we have
```

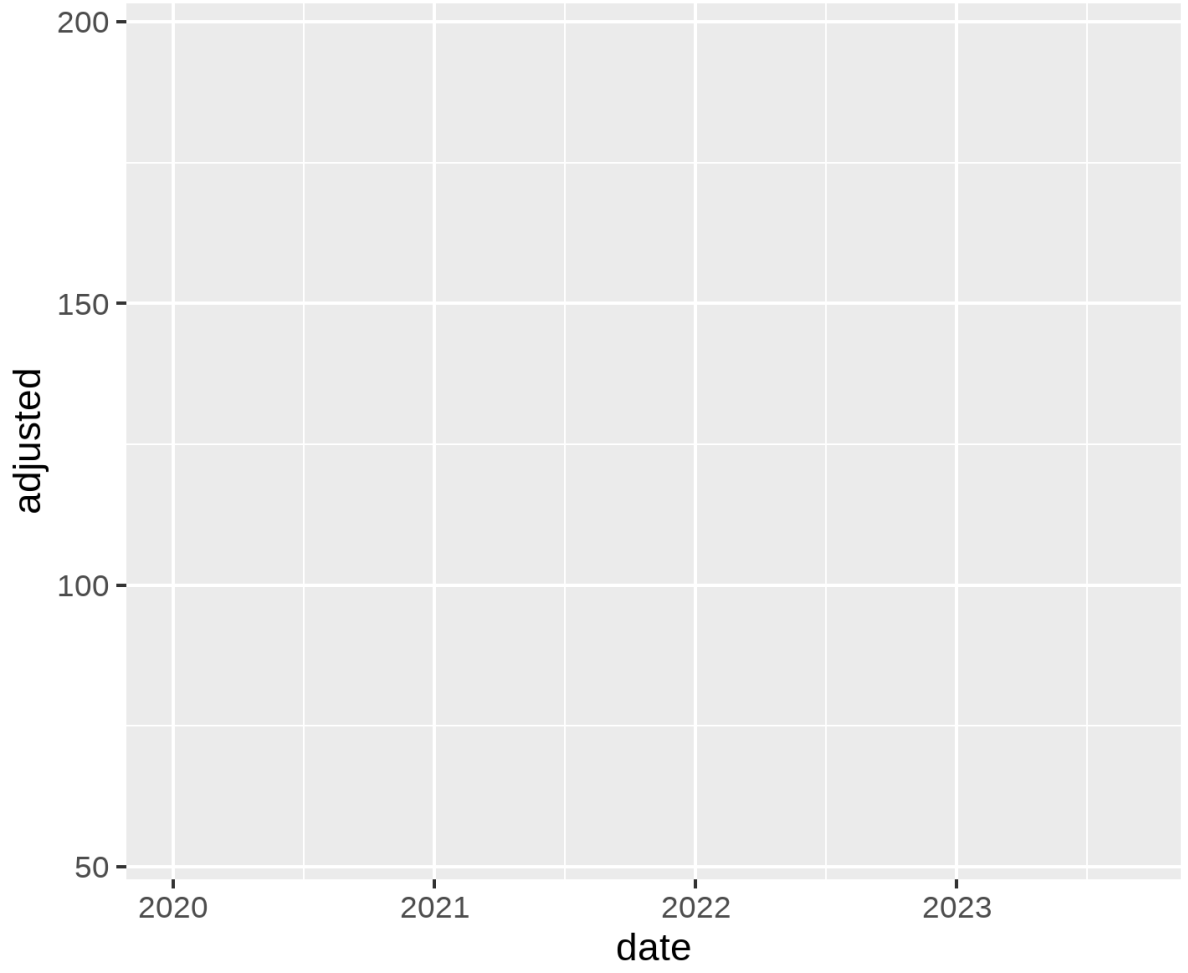
```
## # A tibble: 1 × 5
##   date       symbol adjusted  year month
##   <date>     <chr>     <dbl> <dbl> <ord>
## 1 2023-09-05 AAPL       190.  2023 Sep
```

A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(aapl)
```

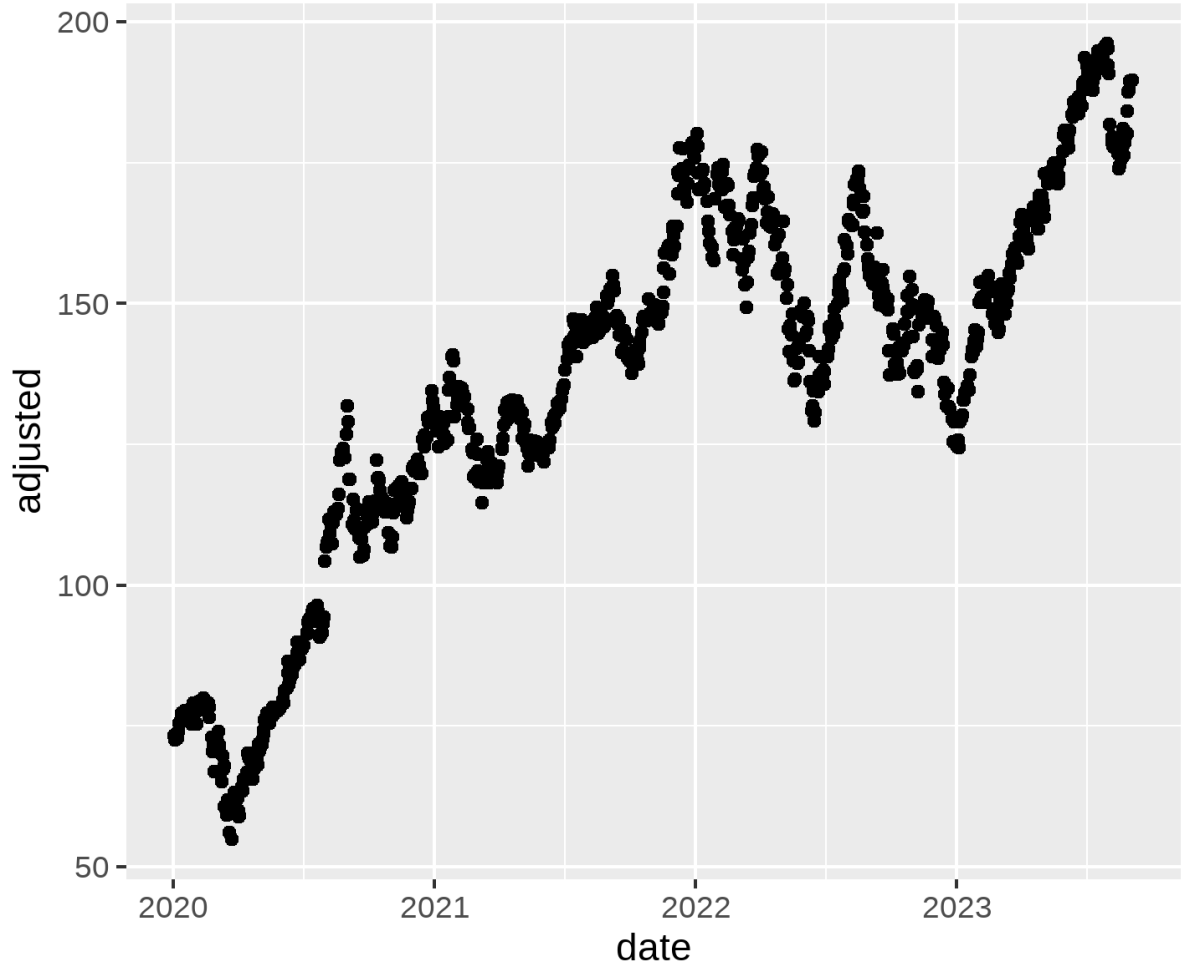
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
)
```



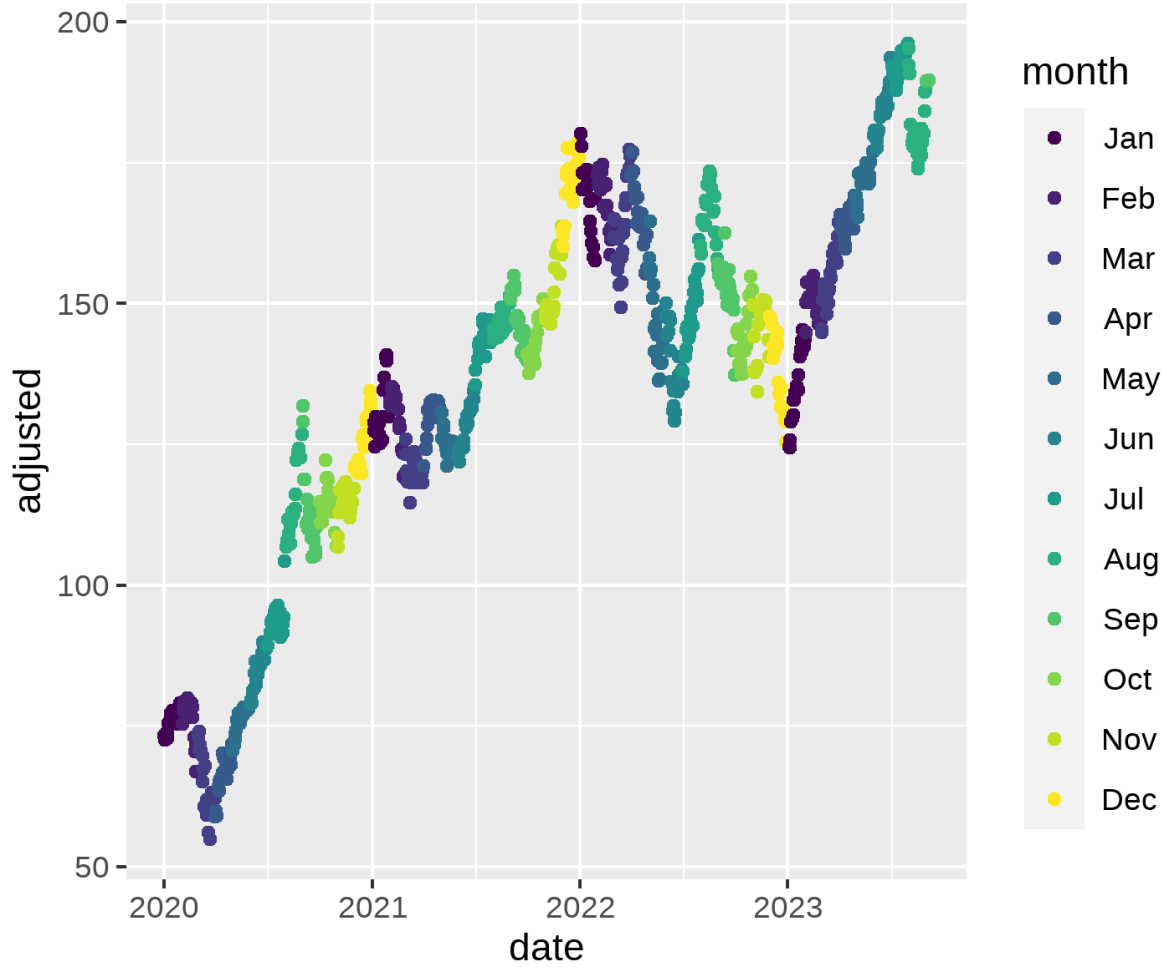
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point()
```



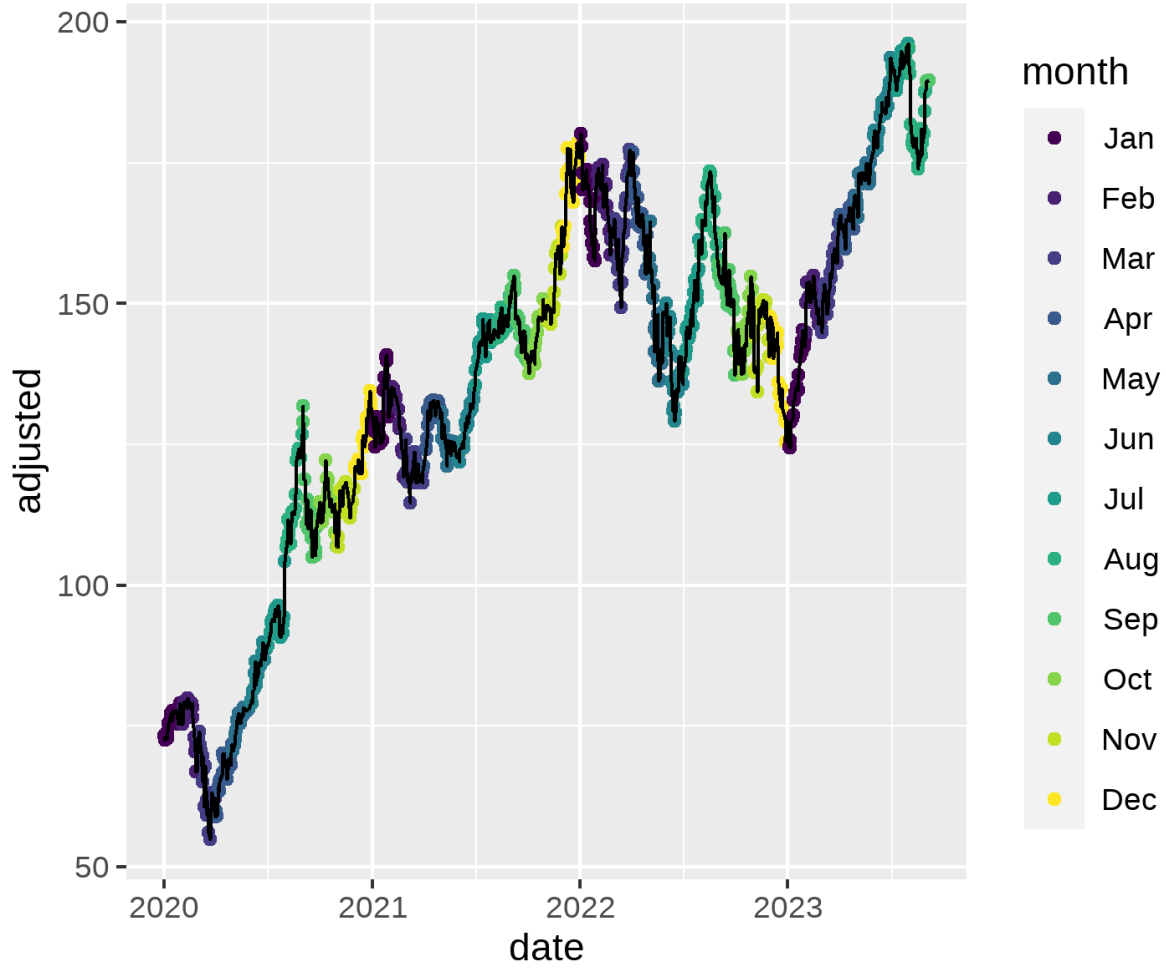
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point(
    ggplot2::aes(color = month)
  )
```



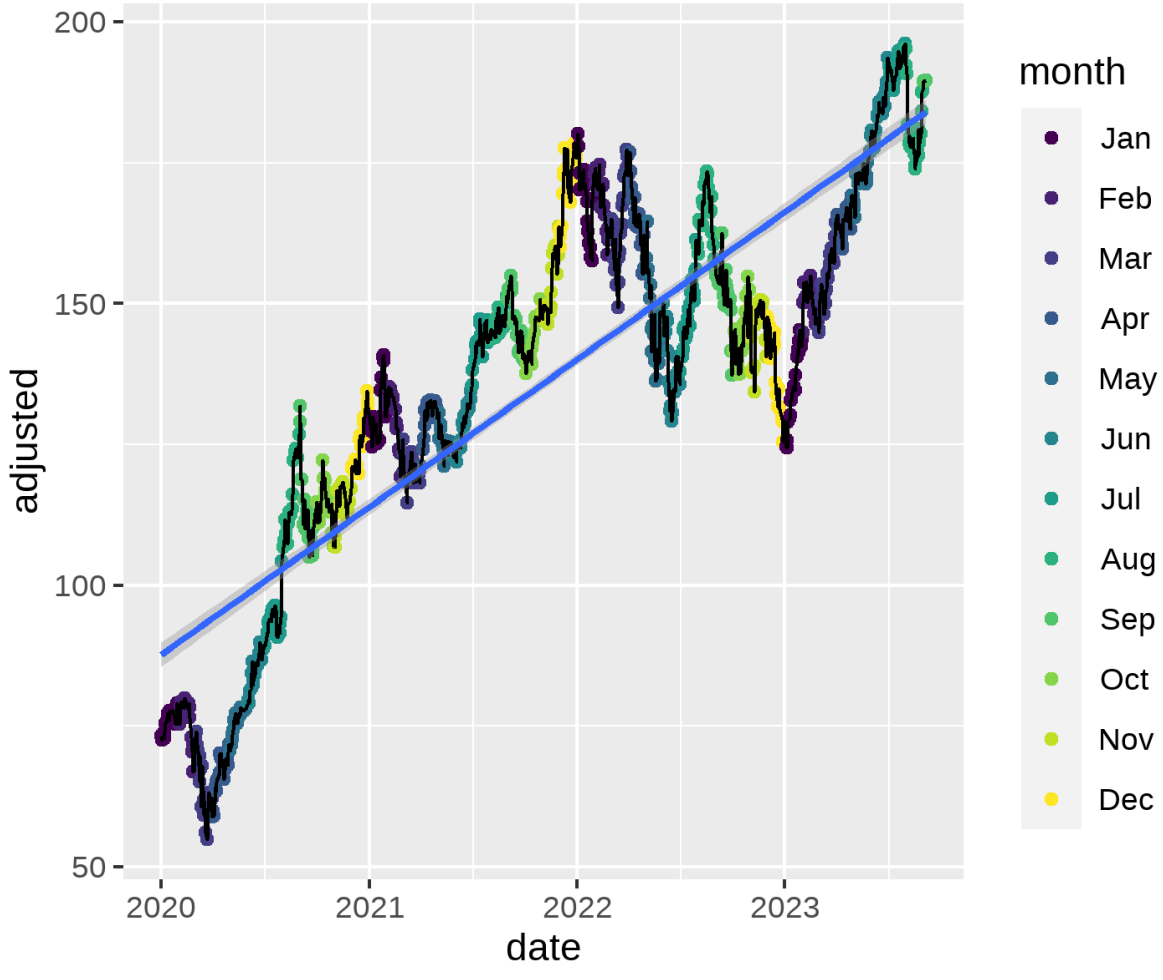
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point(
    ggplot2::aes(color = month)
  ) +
  ggplot2::geom_line()
```



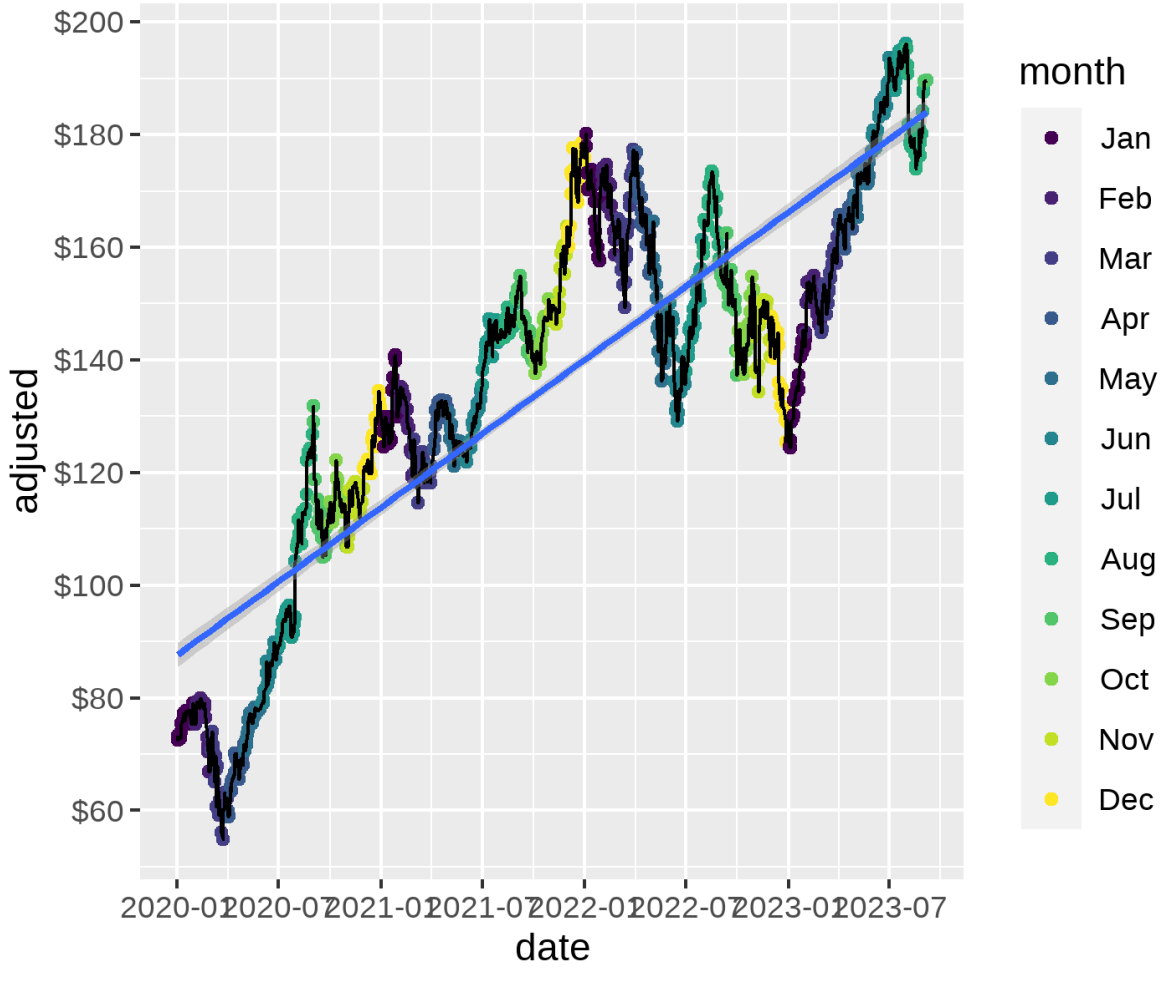
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point(
    ggplot2::aes(color = month)
  ) +
  ggplot2::geom_line() +
  ggplot2::geom_smooth(
    method = lm, formula = 'y ~ x'
    )
```



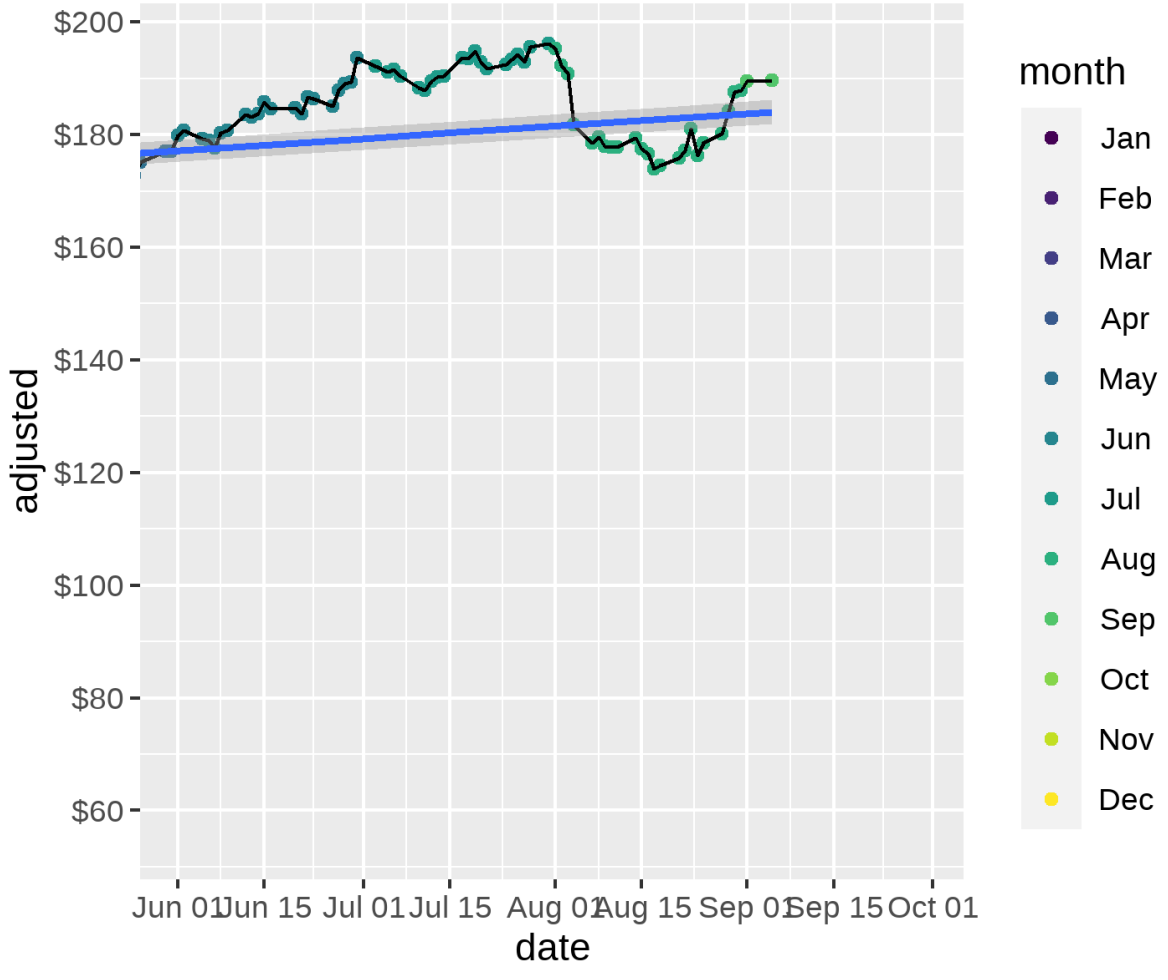
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point(
    ggplot2::aes(color = month)
  ) +
  ggplot2::geom_line() +
  ggplot2::geom_smooth(
    method = lm, formula = 'y ~ x'
    )  +
  ggplot2::scale_x_date(
    breaks = scales::pretty_breaks(n=8)
    ) +
  ggplot2::scale_y_continuous(
    breaks = scales::pretty_breaks(n=6),
    labels = scales::dollar)
```



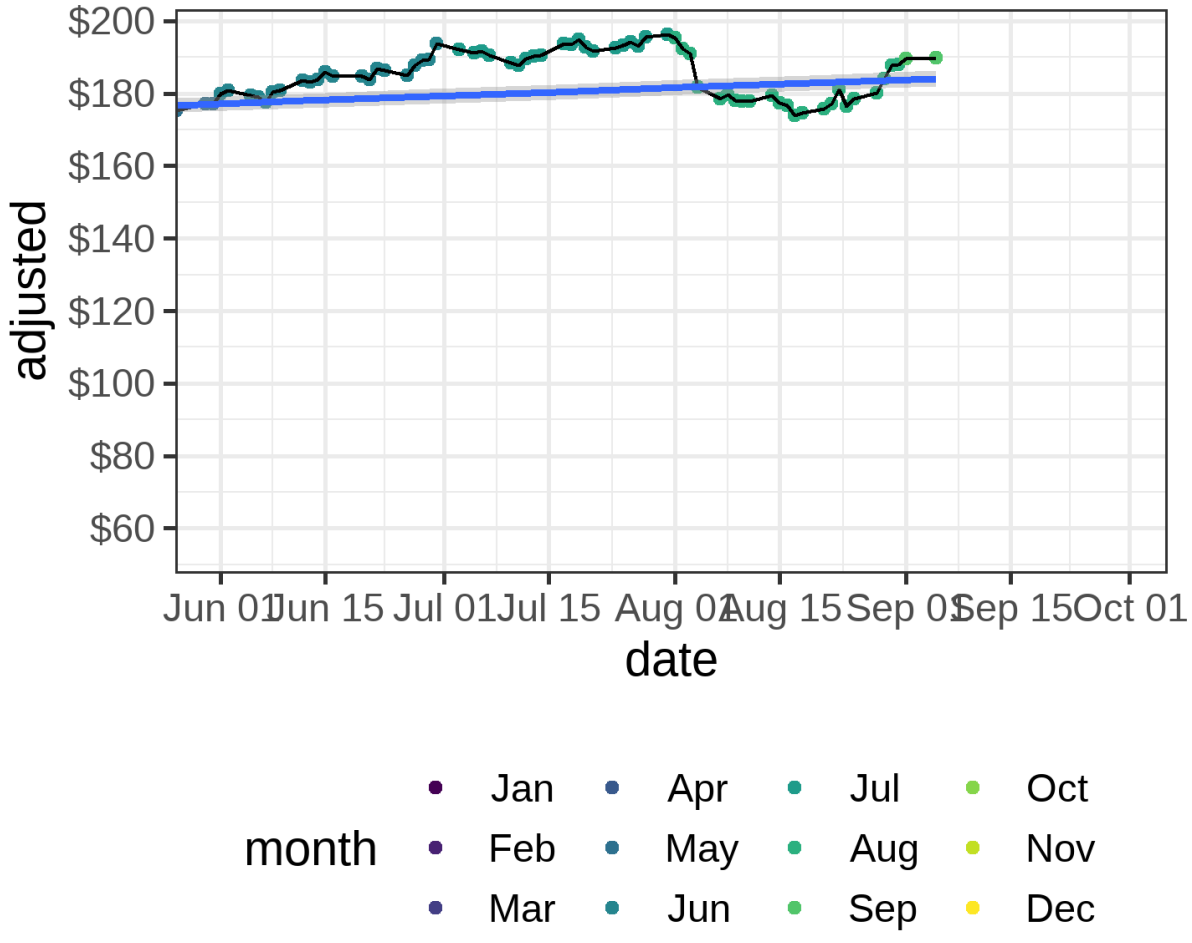
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point(
    ggplot2::aes(color = month)
  ) +
  ggplot2::geom_line() +
  ggplot2::geom_smooth(
    method = lm, formula = 'y ~ x'
    )  +
  ggplot2::scale_x_date(
    breaks = scales::pretty_breaks(n=8)
    ) +
  ggplot2::scale_y_continuous(
    breaks = scales::pretty_breaks(n=6),
    labels = scales::dollar) +
  tidyquant::coord_x_date(
    xlim = c('2023-06-01', '2023-09-30')
    )
```



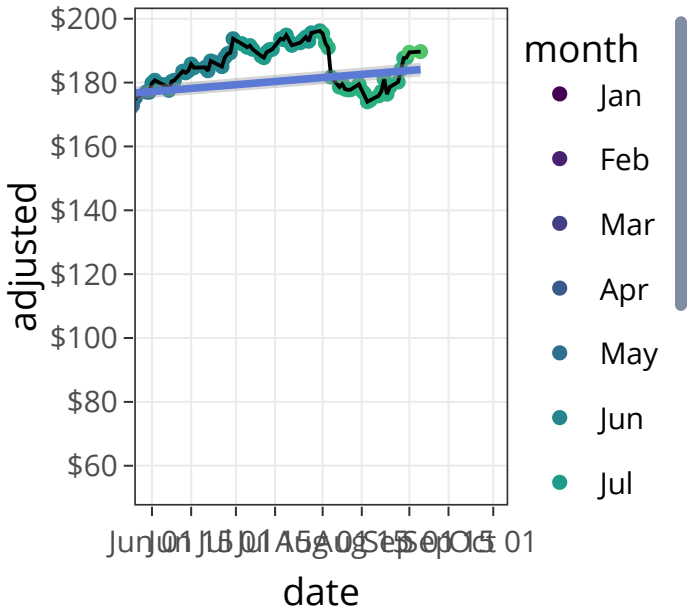
A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point(
    ggplot2::aes(color = month)
  ) +
  ggplot2::geom_line() +
  ggplot2::geom_smooth(
    method = lm, formula = 'y ~ x'
    )  +
  ggplot2::scale_x_date(
    breaks = scales::pretty_breaks(n=8)
    ) +
  ggplot2::scale_y_continuous(
    breaks = scales::pretty_breaks(n=6),
    labels = scales::dollar) +
  tidyquant::coord_x_date(
    xlim = c('2023-06-01', '2023-09-30')
    ) +
  ggplot2::theme_bw(base_size = 14) +
  ggplot2::theme(
    legend.position = 'bottom'
    )
```



A Singular TS: The GG Layers

```
# layers are + in ggplot2
ggplot2::ggplot(
  aapl,
  ggplot2::aes(x = date, y = adjusted)
) +
  ggplot2::geom_point(
    ggplot2::aes(color = month)
  ) +
  ggplot2::geom_line() +
  ggplot2::geom_smooth(
    method = lm, formula = 'y ~ x'
    )  +
  ggplot2::scale_x_date(
    breaks = scales::pretty_breaks(n=8)
    ) +
  ggplot2::scale_y_continuous(
    breaks = scales::pretty_breaks(n=6),
    labels = scales::dollar) +
  tidyquant::coord_x_date(
    xlim = c('2023-06-01', '2023-09-30')
    ) +
  ggplot2::theme_bw(base_size = 14) +
  ggplot2::theme(
    legend.position = 'bottom'
    ) -> aapl_plot

plotly::ggplotly(p = aapl_plot)
```

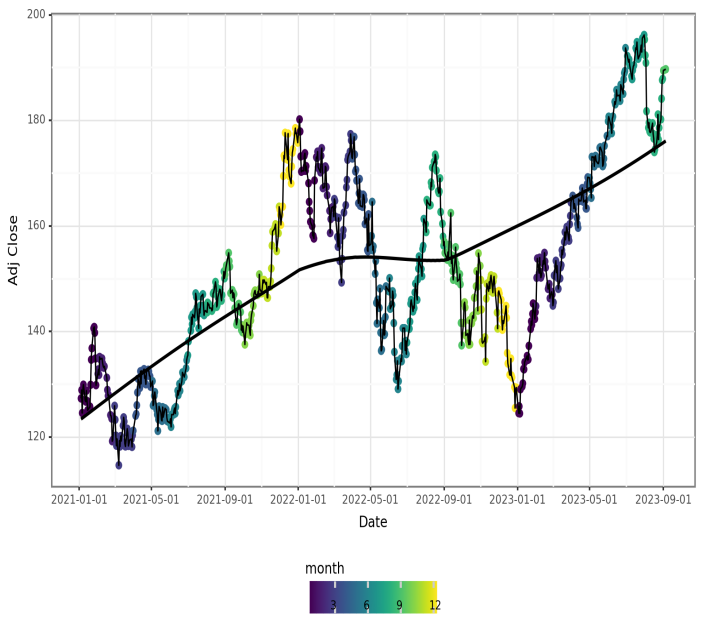


A Singular TS: Implementation

```
import yfinance as yf
import datetime as dt
from plotnine import ggplot, aes, theme,

# extracting the data using yfinance libr
aapl = yf.download(tickers=['AAPL'], star

# creating features
aapl.reset_index(inplace = True) # conver
aapl['month'] = aapl['Date'].dt.month
aapl['year'] = aapl['Date'].dt.year

# plotting using plotnine
(
  ggplot(aapl, aes(x = 'Date', y = 'Adj C
  geom_point( aes(color = 'month') ) +
  geom_line() +
  geom_smooth() +
  theme_bw(base_size = 8) +
  theme(legend_position = 'bottom')
  )
```

```
##
[**********************100%%**********************]  1
```

```
## <Figure Size: (640 x 480)>
```



Demo: `ts()` in  & `yfinance` in  ?

A Summarizing Time-Series Data

Measures of Average

Mean: Given a set of n values $Y_1, Y_2, \ldots, Y_n$, the arithmetic mean can be computed as:

$$\bar{Y} = \frac{Y_1 + Y_2 + \cdots + Y_n}{n} = \frac{1}{n} \sum_{i=1}^{i=n} Y_i.$$

Order Statistics: Given a set of n values $Y_1, Y_2, \ldots, Y_n$, we place them in an ascending order to define the order statistics, written as $Y_{(1)}, Y_{(2)}, \ldots, Y_{(n)}$.

Median:

- If n is odd, $n = 2m + 1$ and the median is $Y_{(m+1)}$.
- If n is even, $n = 2m$ and the median is the average of the two middle numbers, i.e., $\frac{1}{2}[Y_{(m)} + Y_{(m+1)}]$.

Measures of Variation

The **range** denotes the difference between the largest and smallest value in a sample:

$$Range = Y_{(n)} - Y_{(1)}.$$

The **deviation** is defined as the difference between a given observation Y_i and the mean $\bar{Y}$.

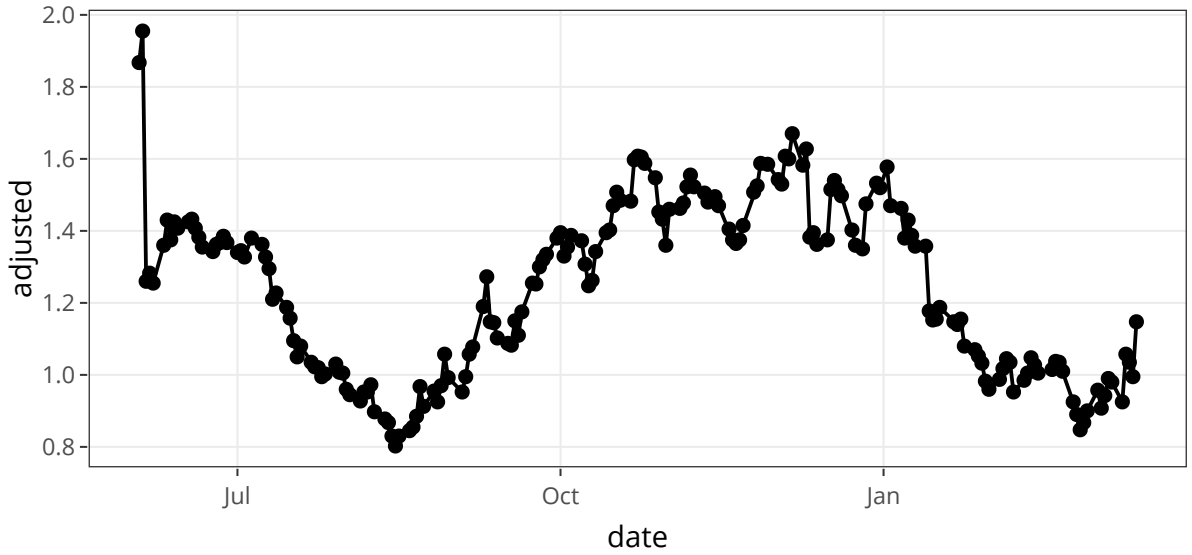
The **mean absolute deviation (MAD)** is the average deviations about the mean, irrespective of their sign:

$$\text{MAD} = \frac{\sum_{i=1}^{i=n} |d_i|}{n}.$$

The **variance** is the average of the squared deviations around the mean:

$$S^2 = \frac{\sum_{i=1}^{i=n} d_i^2}{n-1}.$$

The GameStop Short Squeeze



Note: Click Gamestop Short Squeeze for more details.

Summarizing the GME Short Squeeze: Avg/Var Measures

```
gme_get =
  tidyquant::tq_get(x = 'GME', from = '2019-06-01', to = '2020-03-16'
  dplyr::select(date, adjusted) |>
  dplyr::mutate(
    year = lubridate::year(date),
    month = lubridate::month(date, label = T)
  )

gme_get
```

```
## # A tibble: 198 × 4
##    date       adjusted  year m
##    <date>        <dbl> <dbl> <
##  1 2019-06-03     1.87  2019 J
##  2 2019-06-04     1.96  2019 J
##  3 2019-06-05     1.26  2019 J
##  4 2019-06-06     1.28  2019 J
##  5 2019-06-07     1.25  2019 J
##  6 2019-06-10     1.36  2019 J
##  7 2019-06-11     1.43  2019 J
##  8 2019-06-12     1.38  2019 J
##  9 2019-06-13     1.42  2019 J
## 10 2019-06-14     1.41  2019 J
## # ℹ 188 more rows
```

Summarizing the GME Short Squeeze: Avg/Var Measures

```
gme_get =
  tidyquant::tq_get(x = 'GME', from = '2019-06-01', to = '2020-03-16'
  dplyr::select(date, symbol, adjusted) |>
  dplyr::mutate(
    year = lubridate::year(date),
    month = lubridate::month(date, label = T)
  )

gme_summary =
  gme_get |>
  dplyr::group_by(symbol)

gme_summary
```

```
## # A tibble: 198 × 5
## # Groups:   symbol [1]
##    date       symbol adjusted
##    <date>     <chr>     <dbl>
##  1 2019-06-03 GME        1.87
##  2 2019-06-04 GME        1.96
##  3 2019-06-05 GME        1.26
##  4 2019-06-06 GME        1.28
##  5 2019-06-07 GME        1.25
##  6 2019-06-10 GME        1.36
##  7 2019-06-11 GME        1.43
##  8 2019-06-12 GME        1.38
##  9 2019-06-13 GME        1.42
## 10 2019-06-14 GME        1.41
## # ℹ 188 more rows
```

Summarizing the GME Short Squeeze: Avg/Var Measures

```
gme_get =
  tidyquant::tq_get(x = 'GME', from = '2019-06-01', to = '2020-03-16'
  dplyr::select(date, symbol, adjusted) |>
  dplyr::mutate(
    year = lubridate::year(date),
    month = lubridate::month(date, label = T)
  )

gme_summary =
  gme_get |>
  dplyr::group_by(symbol) |>
  dplyr::summarise(
    ajusted_avg = mean(adjusted),
    adjusted_med = median(adjusted),
    adjusted_var = var(adjusted),
    adjusted_sd = sd(adjusted)
  )

gme_summary |> t() # transposing for printout
```

```
##                [,1]
## symbol         "GME"
## ajusted_avg    "1.240871"
## adjusted_med   "1.2775"
## adjusted_var   "0.0562454"
## adjusted_sd    "0.2371611"
```

Summarizing the GME Short Squeeze: Avg/Var Measures

```
gme_get =
  tidyquant::tq_get(x = 'GME', from = '2019-06-01', to = '2020-03-16'
  dplyr::select(date, symbol, adjusted) |>
  dplyr::mutate(
    year = lubridate::year(date),
    month = lubridate::month(date, label = T)
  )

gme_summary =
  gme_get |>
  dplyr::group_by(symbol, year) |>
  dplyr::summarise(
    ajusted_avg = mean(adjusted),
    adjusted_med = median(adjusted),
    adjusted_var = var(adjusted),
    adjusted_sd = sd(adjusted)
  )

gme_summary
```

```
## # A tibble: 2 × 6
## # Groups:   symbol [1]
##   symbol  year ajusted_avg adj
##   <chr>  <dbl>       <dbl>
## 1 GME     2019        1.29
## 2 GME     2020        1.09
```

Summarizing the GME Short Squeeze: Avg/Var Measures

```
gme_get =
  tidyquant::tq_get(x = 'GME', from = '2019-06-01', to = '2020-03-16'
  dplyr::select(date, symbol, adjusted) |>
  dplyr::mutate(
    year = lubridate::year(date),
    month = lubridate::month(date, label = T)
  )

gme_summary =
  gme_get |>
  dplyr::group_by(symbol, year, month) |>
  dplyr::summarise(
    ajusted_avg = mean(adjusted),
    adjusted_med = median(adjusted),
    adjusted_var = var(adjusted),
    adjusted_sd = sd(adjusted)
  )

print(gme_summary, n=15)
```

```
## # A tibble: 10 × 7
## # Groups:   symbol, year [2]
##    symbol  year month ajusted_
##    <chr>  <dbl> <ord>       <d
##  1 GME     2019 Jun         1.
##  2 GME     2019 Jul         1.
##  3 GME     2019 Aug         0.
##  4 GME     2019 Sep         1.
##  5 GME     2019 Oct         1.
##  6 GME     2019 Nov         1.
##  7 GME     2019 Dec         1.
##  8 GME     2020 Jan         1.
##  9 GME     2020 Feb         0.
## 10 GME     2020 Mar         0.
```

A Demo in 

Correlation

The Pearson Correlation Coefficient

- **Correlation:** measures the strength of the **linear relationship** between two quantitative variables.
- It can be computed using the `cor()` from base R. Mathematically speaking, the pearson correlation coefficient, r, can be computed as

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

- Do **not** use the Pearson Correlation coefficient if both variables are not quantitative. Instead, refer to the `mixed.cor()` from the psch package to compute the correlations for mixtures of continuous, polytomous, and/or dichotomous variables.
- You should supplement **any descriptive summaries with visualizations** to ensure that you are able to interpret the computations correctly.

Supplement Summaries with Viz: Anscombe's Dataset

In a seminal paper, Anscombe stated:

> **Few of us escape being indoctrinated with these notions:**
>
> - numerical **calculations are exact, but graphs are rough**;
> - for any particular kind of **statistical data there is just one set of calculations constituting a correct statistical analysis**;
> - performing **intricate calculations is virtuous**, whereas **actually looking at the data is cheating**.

He proceeded by stating that

> a computer should **make both calculations and graphs**. Both sorts of output should be studied; each will contribute to understanding.

Now, let us consider his four datasets, each consisting of eleven (x,y) pairs.

Supplement Summaries with Viz: Anscombe's Dataset

x1	x2	x3	x4	y1	y2	y3	y4
10	10	10	8	8.04	9.14	7.46	6.58
8	8	8	8	6.95	8.14	6.77	5.76
13	13	13	8	7.58	8.74	12.74	7.71
9	9	9	8	8.81	8.77	7.11	8.84
11	11	11	8	8.33	9.26	7.81	8.47
14	14	14	8	9.96	8.1	8.84	7.04
6	6	6	8	7.24	6.13	6.08	5.25
4	4	4	19	4.26	3.1	5.39	12.5
12	12	12	8	10.84	9.13	8.15	5.56
7	7	7	8	4.82	7.26	6.42	7.91
5	5	5	8	5.68	4.74	5.73	6.89

Showing 1 to 11 of 11 entries

Previous 1 Next

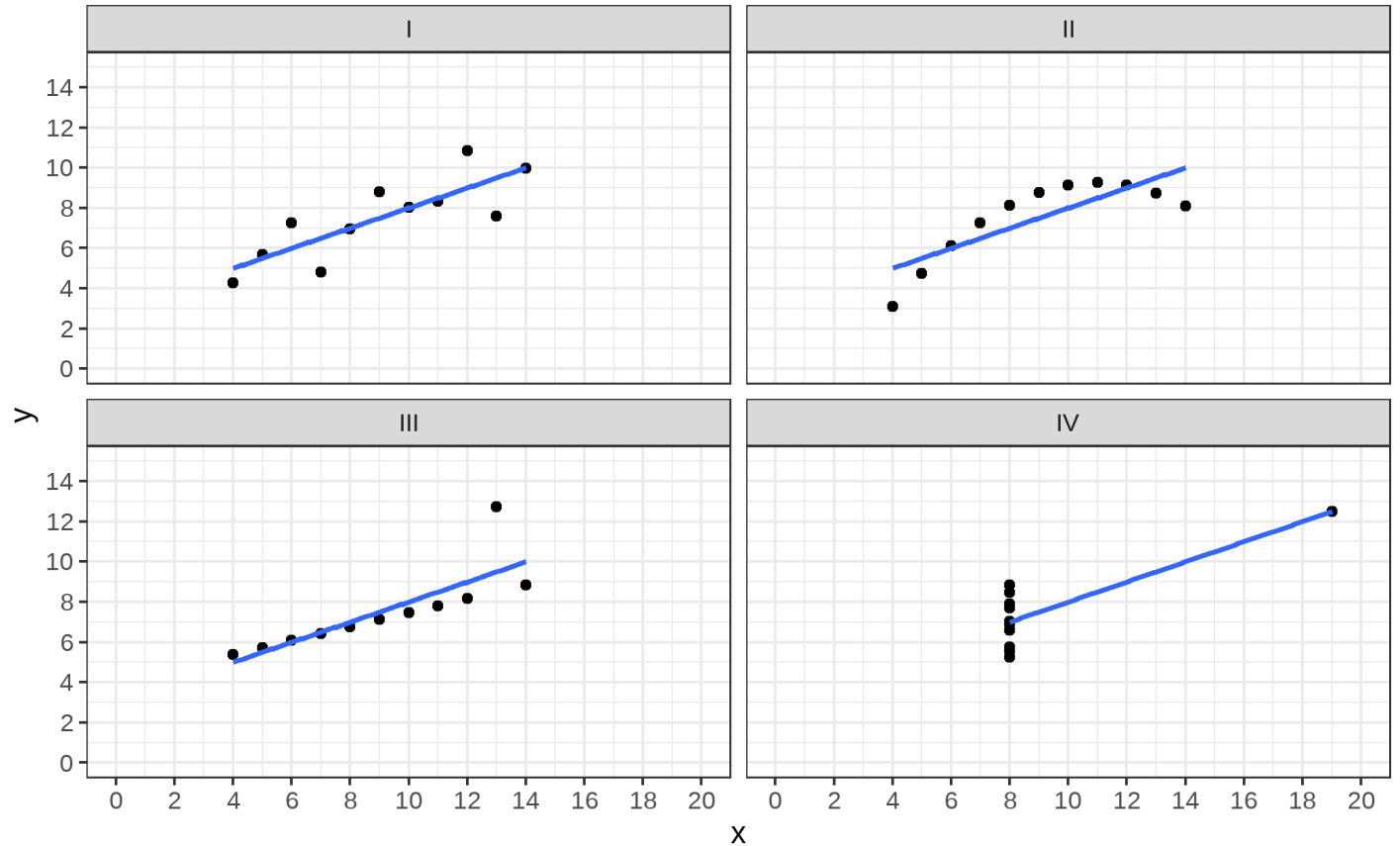
Supplement Summaries with Viz: Anscombe's Dataset

set	x.mean	x.sd	y.mean	y.sd	corr
I	9	3.32	7.5	2.03	0.82
II	9	3.32	7.5	2.03	0.82
III	9	3.32	7.5	2.03	0.82
IV	9	3.32	7.5	2.03	0.82

Showing 1 to 4 of 4 entries

Previous 1 Next

Supplement Summaries with Viz: Anscombe's Dataset



Recap

Summary of Main Points

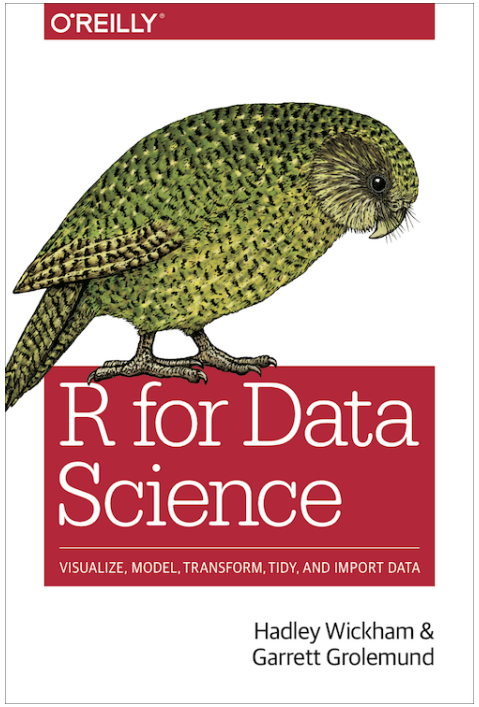
By now, you should be able to do the following:

- Examine the goals of utilizing line charts in time-series analysis (i.e., detect trends, seasonality, and cycles).
- Develop a deeper understanding of the grammar of graphics, which we used to create time series plots in R and Python.
- Use numerical summaries to describe a time series.
- Explain what do we mean by correlation.

Review and Clarification

- **Class Notes**: Take some time to revisit your class notes for key insights and concepts.
- **Zoom Recording**: The recording of today's class will be made available on Canvas approximately 3-4 hours after the session ends.
- **Questions**: Please don't hesitate to ask for clarification on any topics discussed in class. It's crucial not to let questions accumulate.

📖 Required Readings 📖



- Data Visualization
- Graphics for Communication
- Dates and Times

🎯 Assignment 🎯

- Go over your notes and complete Assignment 02 on Canvas.