Lab 3: Data Visualization

2025-04-18

Why Visualize Data?

In an age of big data, it can be difficult to accurately convey a narrative through presenting plain numbers in tables

On the flip side, there are **BAD** ways to visualize that data

Let's create a list of axioms or goals that we want to stick to when creating data visualizations that help us convey what we want to be conveyed. Our goals:

- Be clear and concise (avoid over-complicating the figure)
- Accurately represent the data (use the *right* type of graph)
- Be purposeful (avoid doing things because it *looks neat*)
- Others?

Types of Visualizations

Here are a list of different types of graphs often used to express data (these are hyperlinks):

- Most common: Line, Scatter, Bar
- Less common: Density / Histogram, Box, Pie, Sankey
- Rare: Waterfall, Radar, Pareto (these are mostly stylistic variations or combinations of the above graphs, and are often pointless)

Which do I use?

- Non-answer: It depends on the data!

GGPLOT2

R's best friend for data visualization is the package *ggplot2*

```
library(pacman)
p_load(ggplot2)
```

In general, the code for line charts will look like:

```
ggplot(dataset, aes(x = x_variable, y = y_variable)) +
  geom_TYPE() +
  labs(x = "x variable label", y = "y variable label", title = "Graph title") +
  theme()
```

Line Chart

Let's graph the unemployment data over time, using

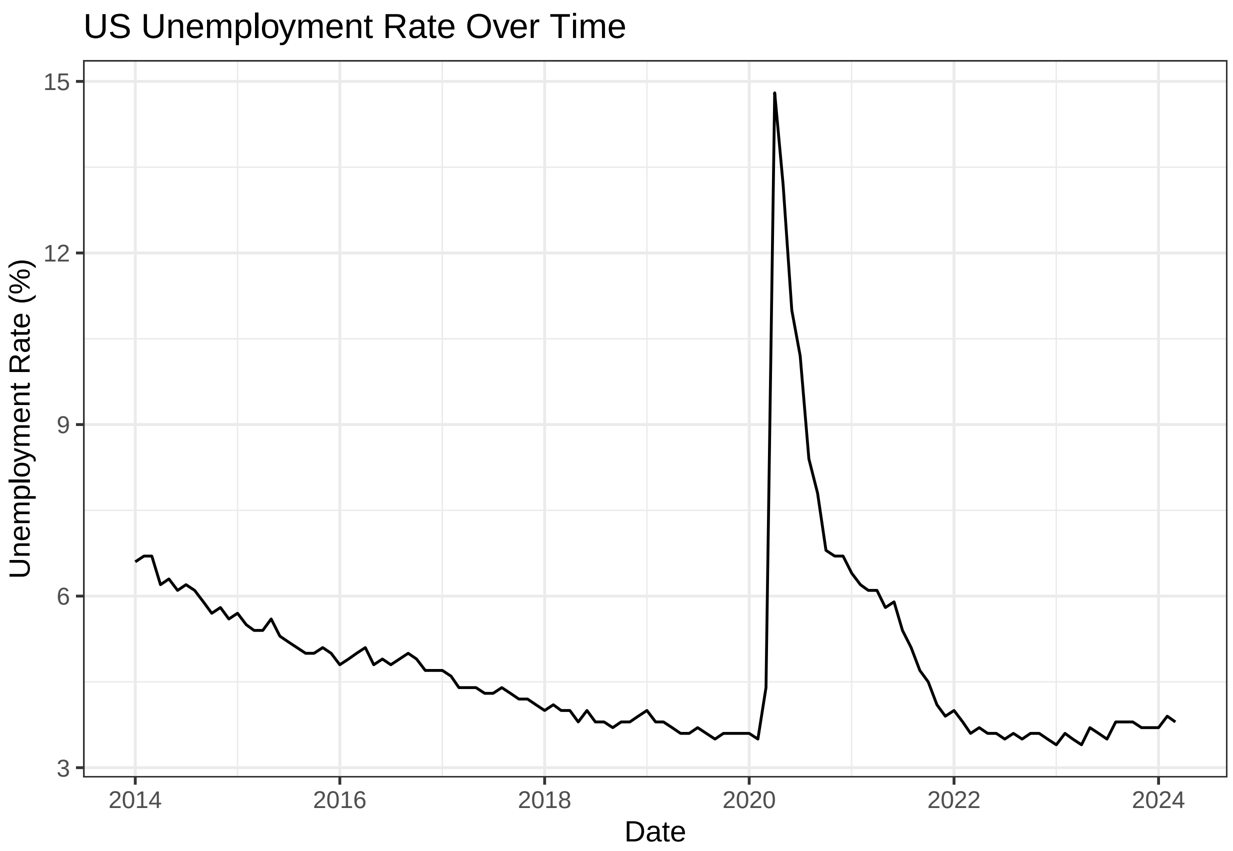
```
# Load package and data
p_load(readr)

url_time = "https://raw.githubusercontent.com/cmulholland217/Metrics_Lab_Spring2025/refs/heads/main/tim

time_data = read_csv(url(url_time))
```

```
## Rows: 123 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (3): UNRATE, FEDFUNDS, INF
## date (1): date
##
## 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.
```

```
# Graph:
ggplot(time_data, aes(x = date, y = UNRATE)) +
  geom_line() +
  labs(x = "Date", y = "Unemployment Rate (%)", title = "US Unemployment Rate Over Time") +
  theme_bw()
```



There are a lot of options we have for adjusting this visualization. Suppose we want:

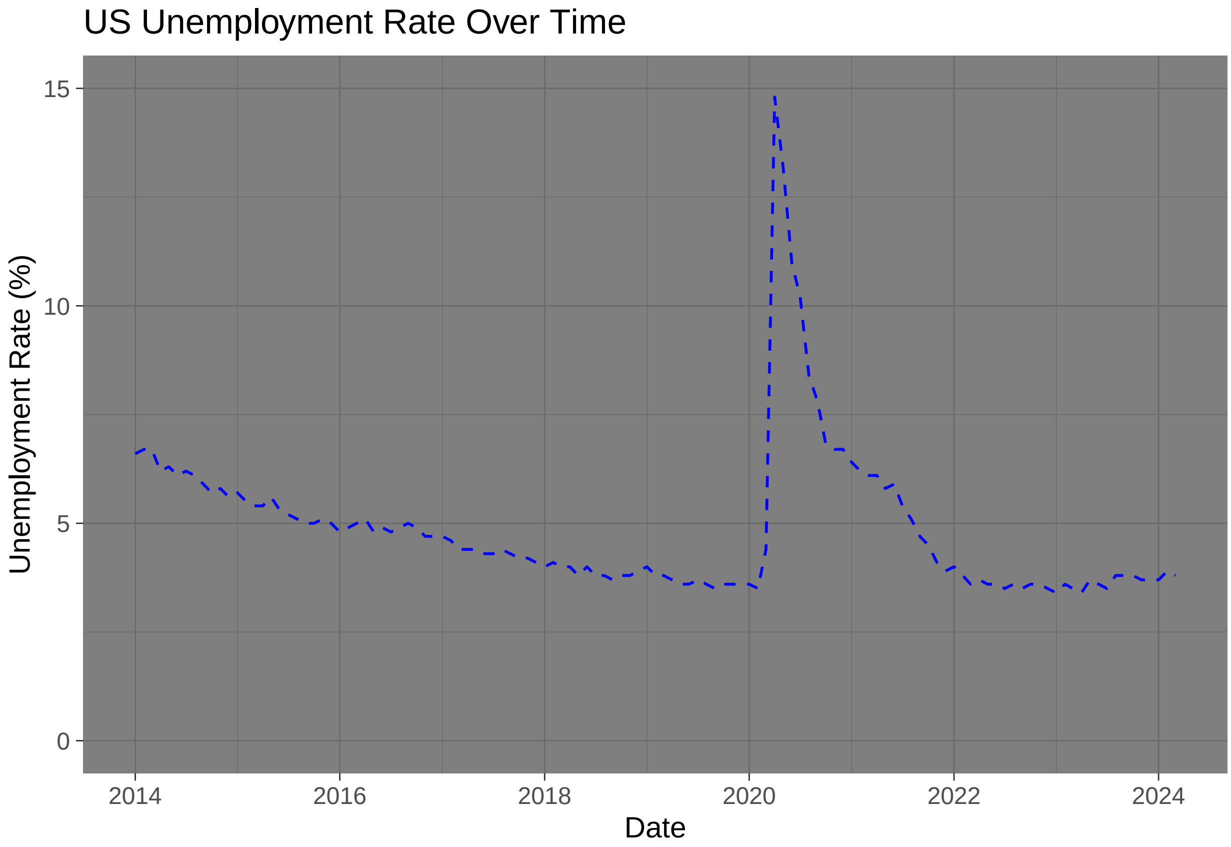
- The y-axis to span from 0 to 15
- The line to be dashed instead of solid
- The line to be blue
- Try theme_dark()

```
# Graph:
ggplot(time_data, aes(x = date, y = UNRATE)) +

  # Changes to the line are within the "geom_line" function
  geom_line(linetype = "dashed",
            color = "blue") +
  labs(x = "Date", y = "Unemployment Rate (%)", title = "US Unemployment Rate Over Time") +

  # Change the y-limits
  ylim(0,15) +

  # Try a new theme!
  theme_dark()
```

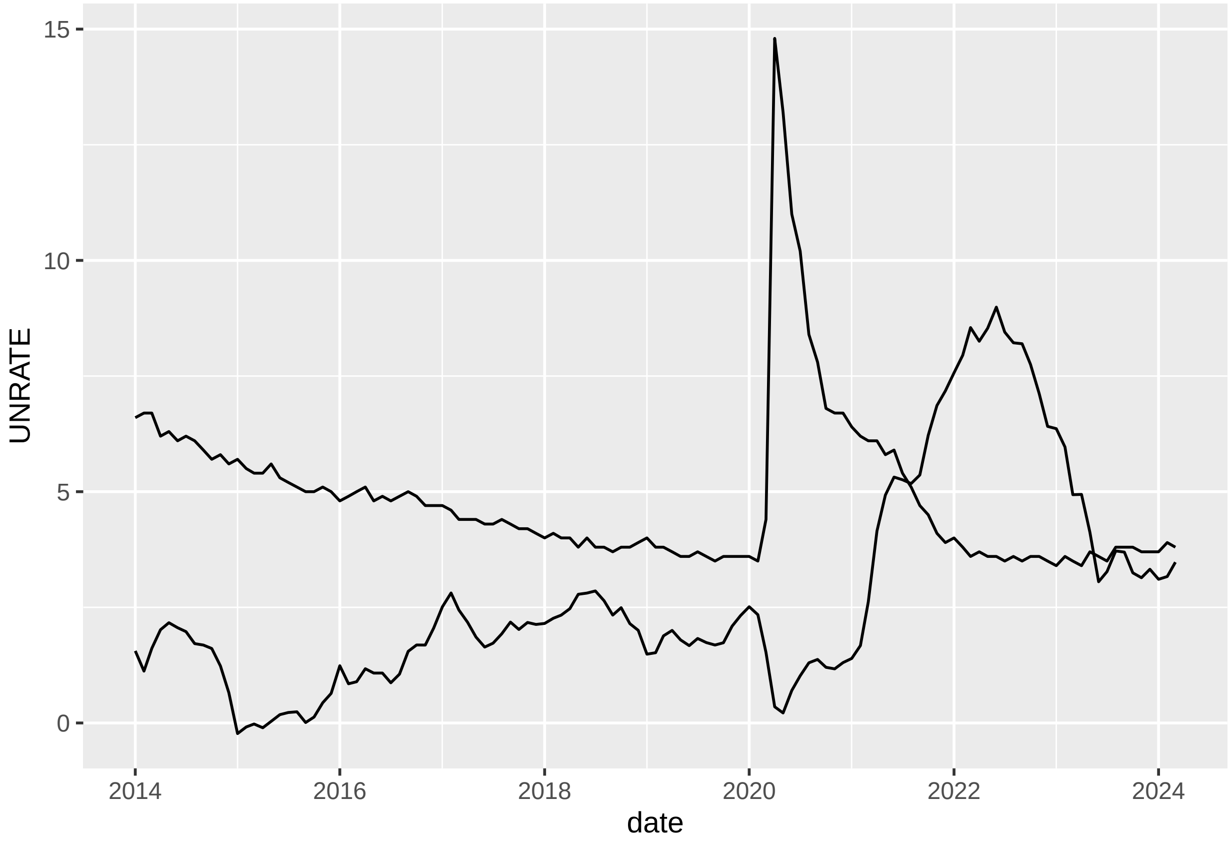


Is this a *better* data visualization? Admittedly, no...

Multiple Lines

Let's graph the unemployment rate (in percent) and the inflation rate (in percent)

```r
# Plainly:
ggplot(time_data, aes(x = date)) +
  geom_line(aes(y = UNRATE)) +
  geom_line(aes(y = INF))
```



Let's add some detail:

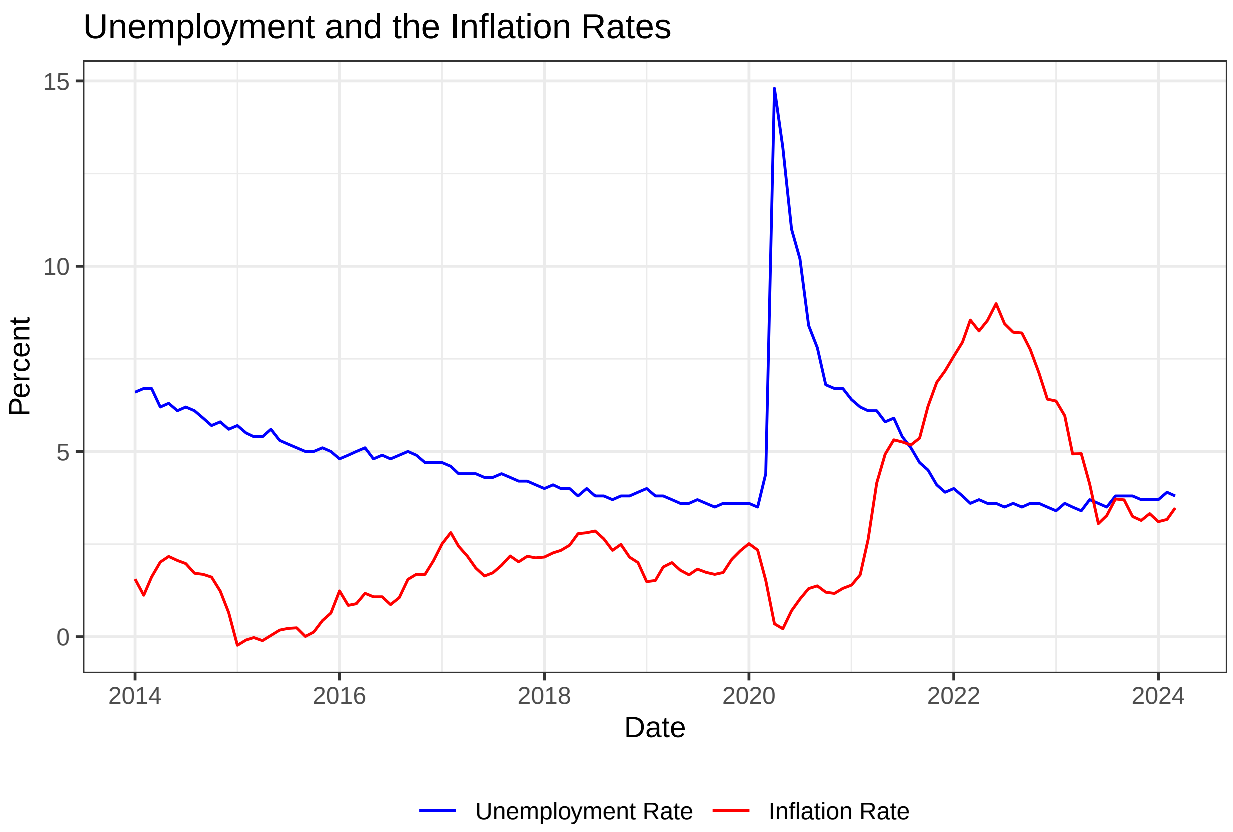
```
ggplot(time_data, aes(x = date)) + # SAME x variable

  # Separate "geom_line" for each variable
  geom_line(aes(y = UNRATE, color = "blue")) +
  geom_line(aes(y = INF, color = "red")) + # COLOR within aes()

  labs(title = "Unemployment and the Inflation Rates",
       x = "Date", y = "Percent") +

  # Legend:
  scale_color_identity(
      name = "",
      guide = "legend",
      labels = c("Unemployment Rate", "Inflation Rate")) +
  theme_bw() +

  # Position of legend
  theme(legend.position = "bottom")
```



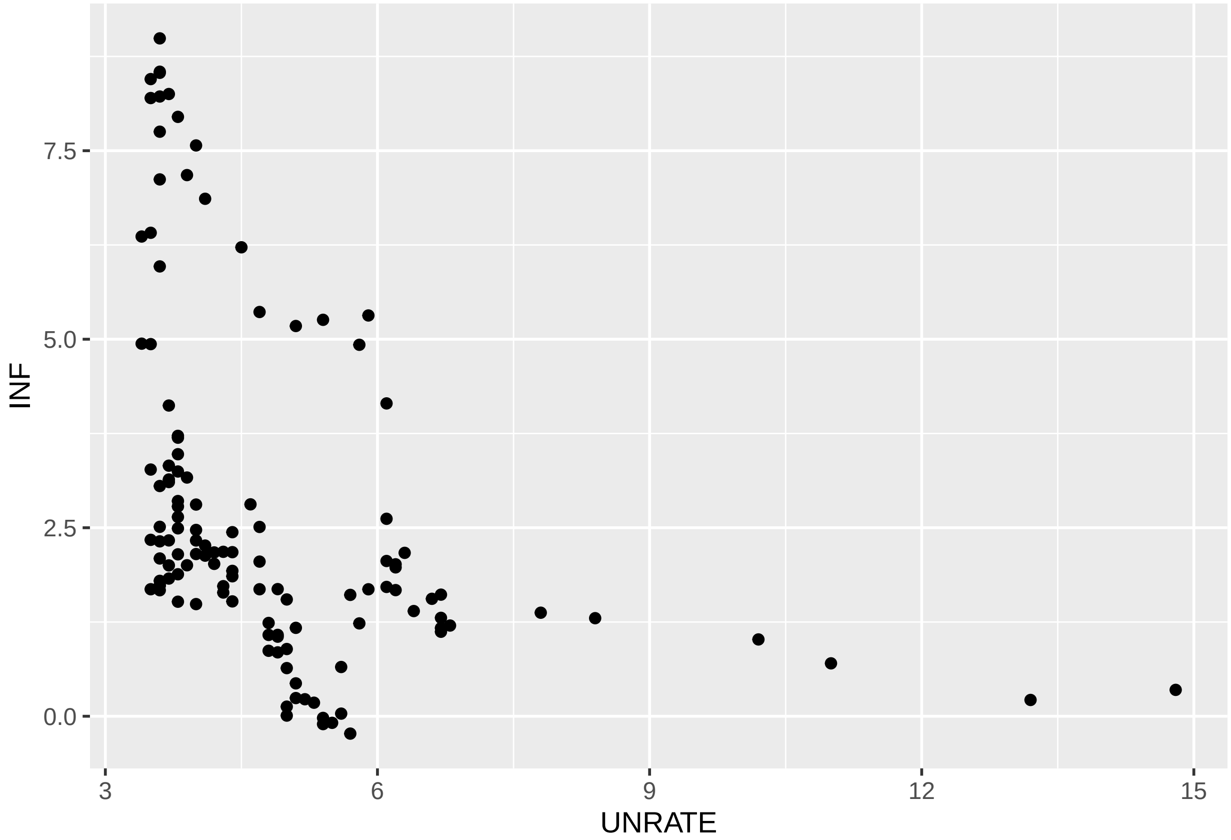
Notice:

- the colors are listed *within* the aes() function
- scale_color_identity() gives details in the legend. name = “ ” yields no name, as it’s unnecessary here
- theme(legend.position = “bottom”) is *after* theme_bw(), doing it the other way around won’t work

Scatter

Let’s graph the unemployment rate *against* the inflation rate (the Phillips Curve)

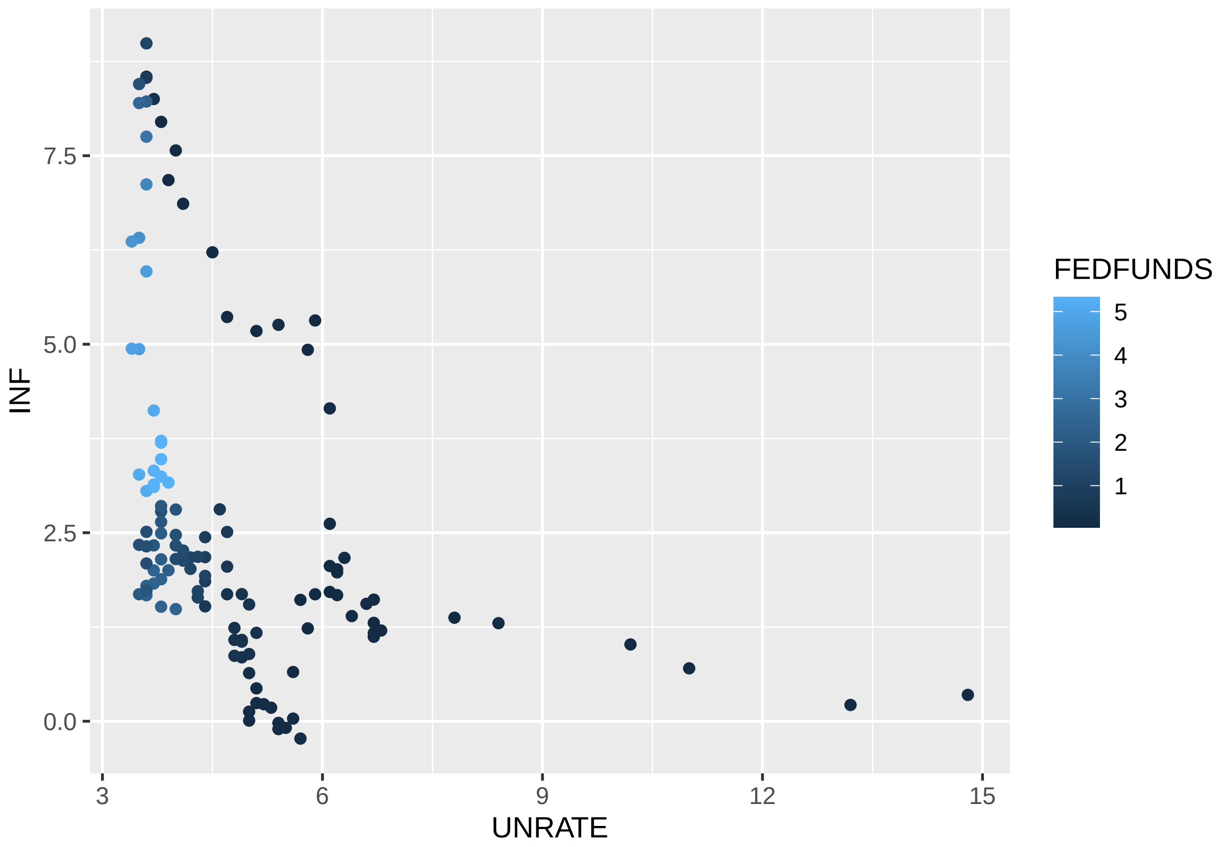
```r
# geom_point()
ggplot(time_data, aes(x = UNRATE, y = INF)) +
  geom_point()
```



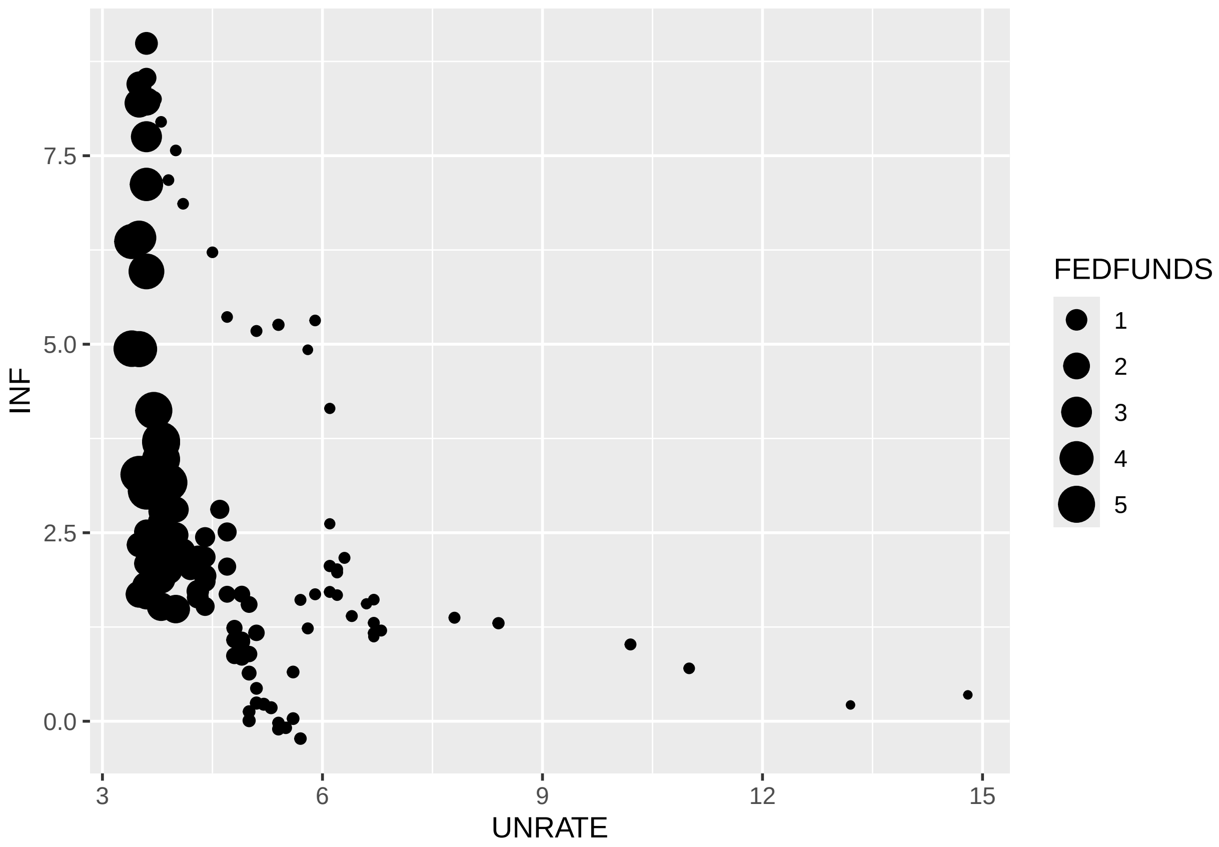
Problems start to arise when there are three variables. One common way to introduce a third variable is with *color* or *size*. Suppose we want to show what the effective federal funds rate was in these data points:

```
# Color
plot1 = ggplot(time_data, aes(x = UNRATE, y = INF)) +
  geom_point(aes(color = FEDFUNDS))

# Size
plot2= ggplot(time_data, aes(x = UNRATE, y = INF)) +
  geom_point(aes(size = FEDFUNDS))
plot1
```



```
plot2
```



Bar Graph

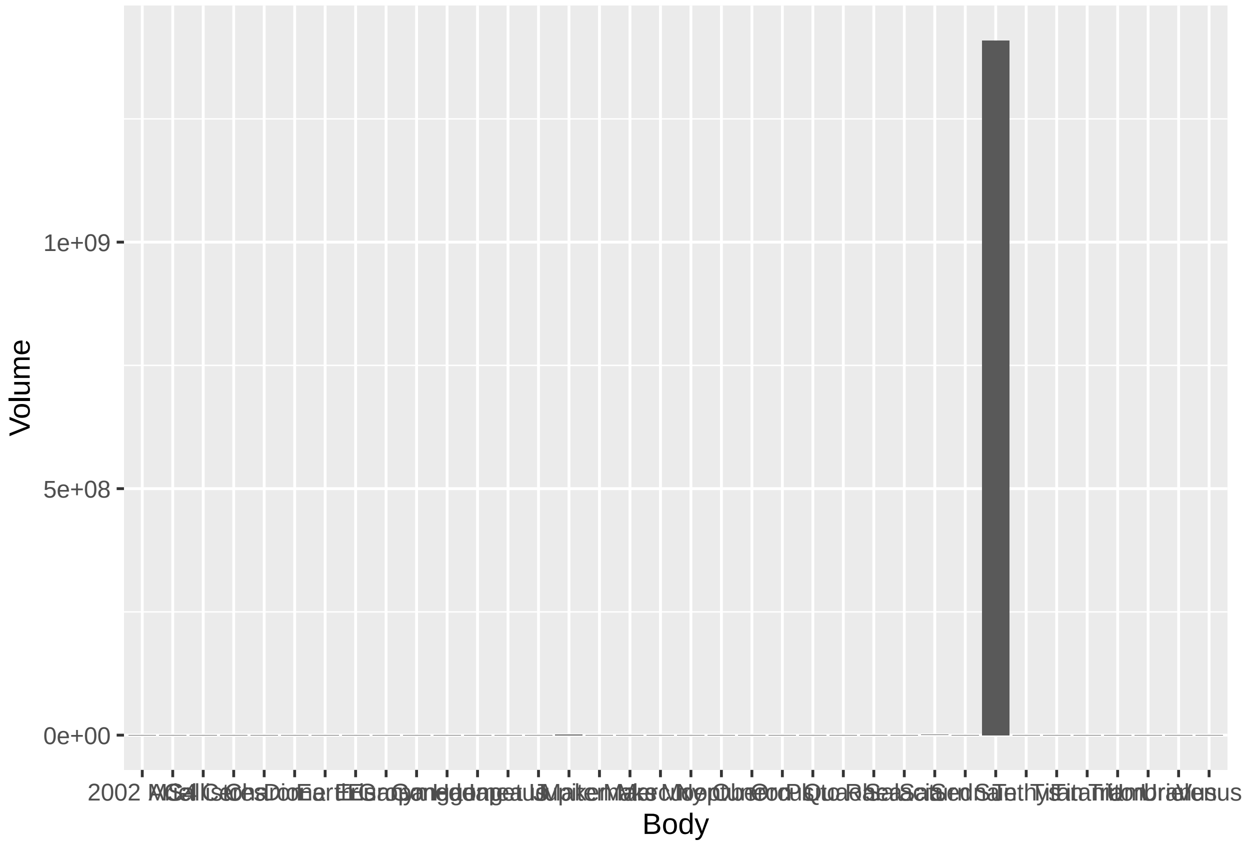
Grab the **space_data** from the GitHub page

```
# planets
url_space = "https://raw.githubusercontent.com/cmulholland217/Metrics_Lab_Spring2025/refs/heads/main/spa

space_data = read_csv(url(url_space))
```

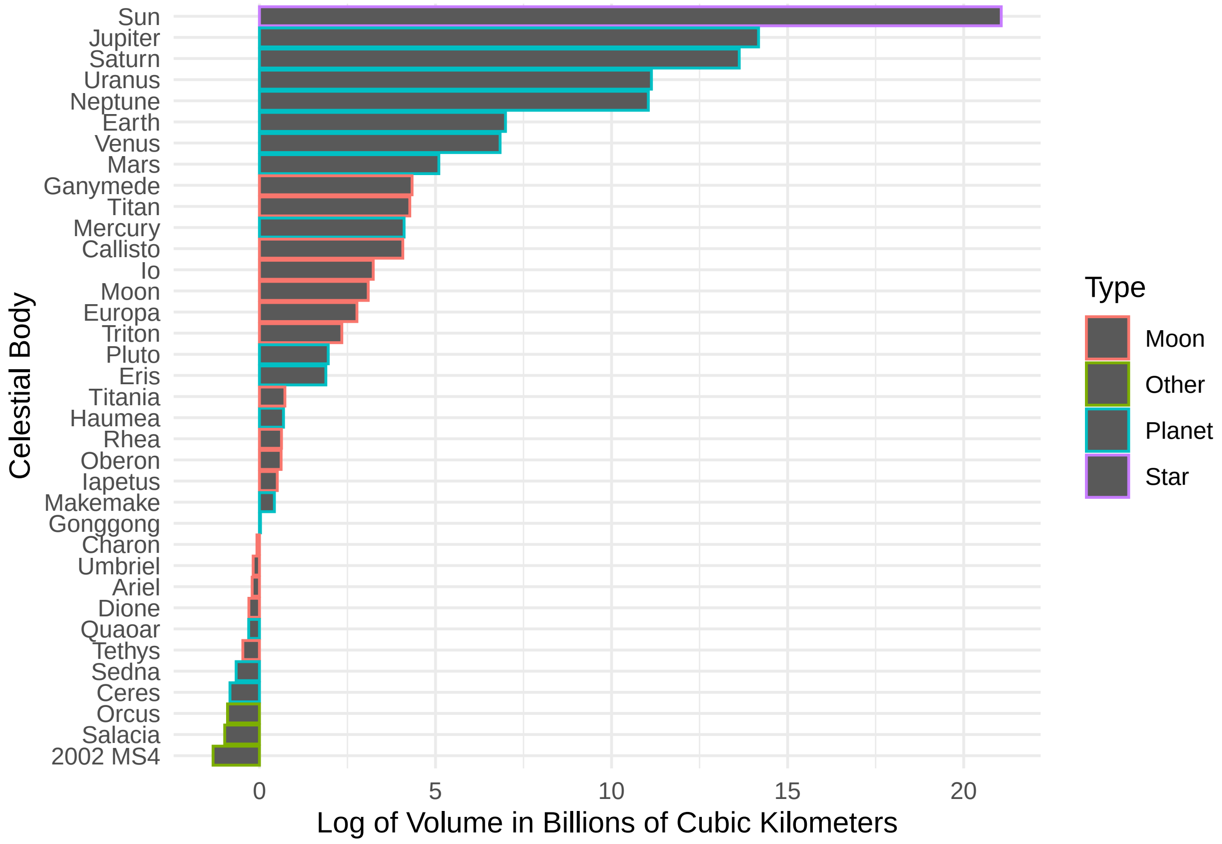
```
## Rows: 36 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Body, Type
## dbl (1): Volume
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ggplot(space_data, aes(x = Body, y = Volume)) +
  geom_bar(stat = "identity")
```



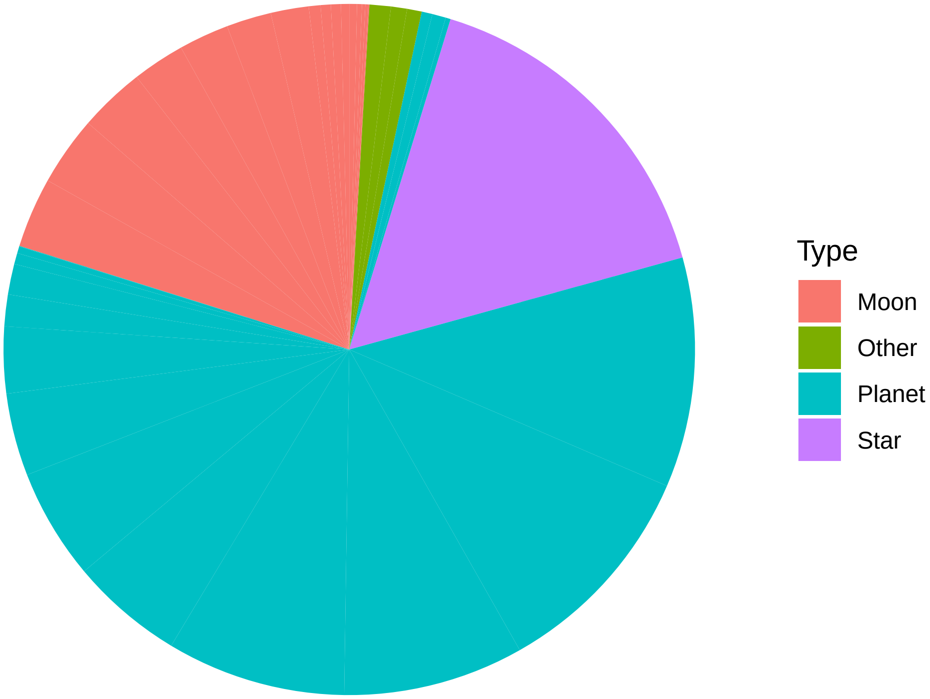
Yikes… Let’s clean this up. And

```r
ggplot(space_data, aes(x = reorder(Body,Volume), y = log(Volume))) +
  # Color for the type of celestial body
  geom_bar(aes(color = Type), stat = "identity") +
    # replace "color" with "fill" for different look
  coord_flip() +
  labs(x = "Celestial Body", y = "Log of Volume in Billions of Cubic Kilometers") +
  theme_minimal()
```



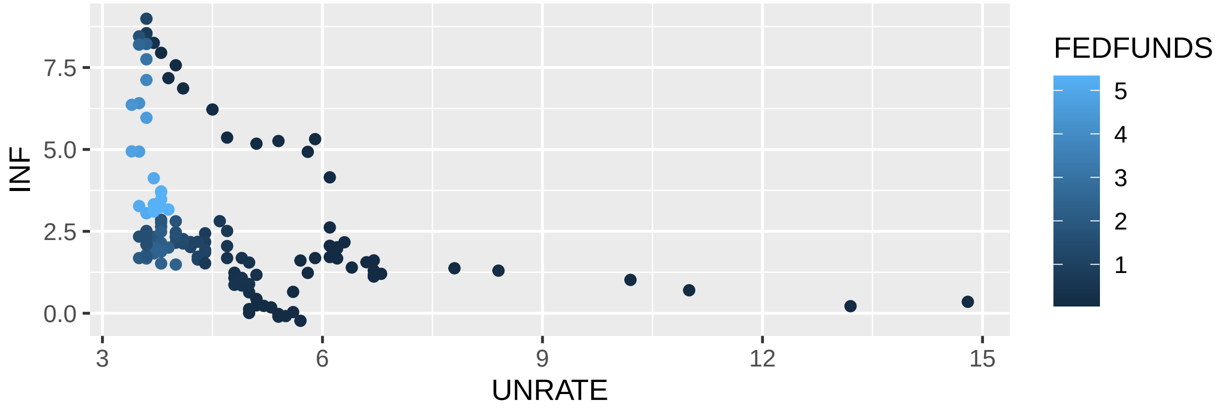
Pie Chart

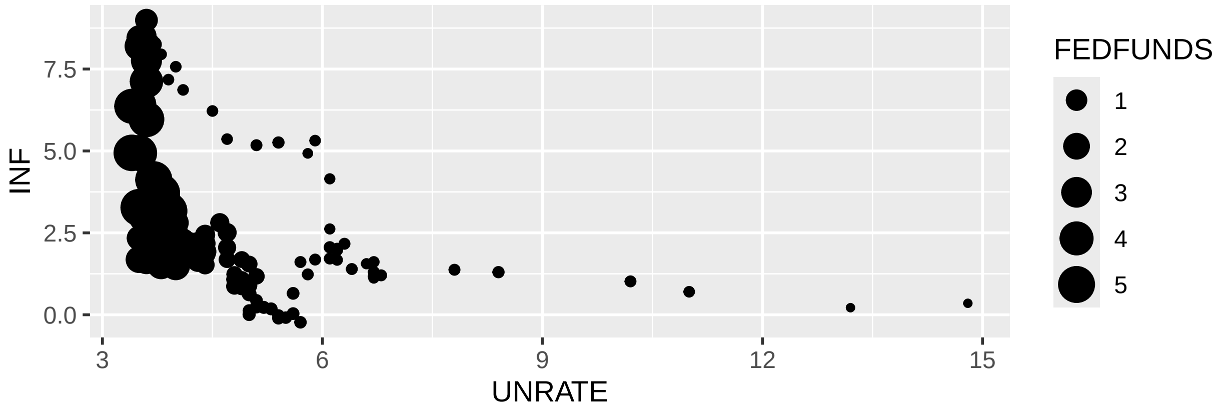
```
ggplot(space_data, aes(x="", y = log(Volume), fill=Type)) +
  geom_bar(stat="identity", width=1) +
  coord_polar("y") +
  theme_void()
```



Multiple Graphs

```
p_load(gridExtra)
grid.arrange(plot1, plot2)
```


INF
7.5
5.0
2.5
0.0
3
6
9
12
15
UNRATE
FEDFUNDS
5
4
3
2
1


INF
7.5
5.0
2.5
0.0
3
6
9
12
15
UNRATE
FEDFUNDS
● 1
● 2
● 3
● 4
● 5