

# YData: Introduction to Data Science



Class 10: Data visualization

# Overview

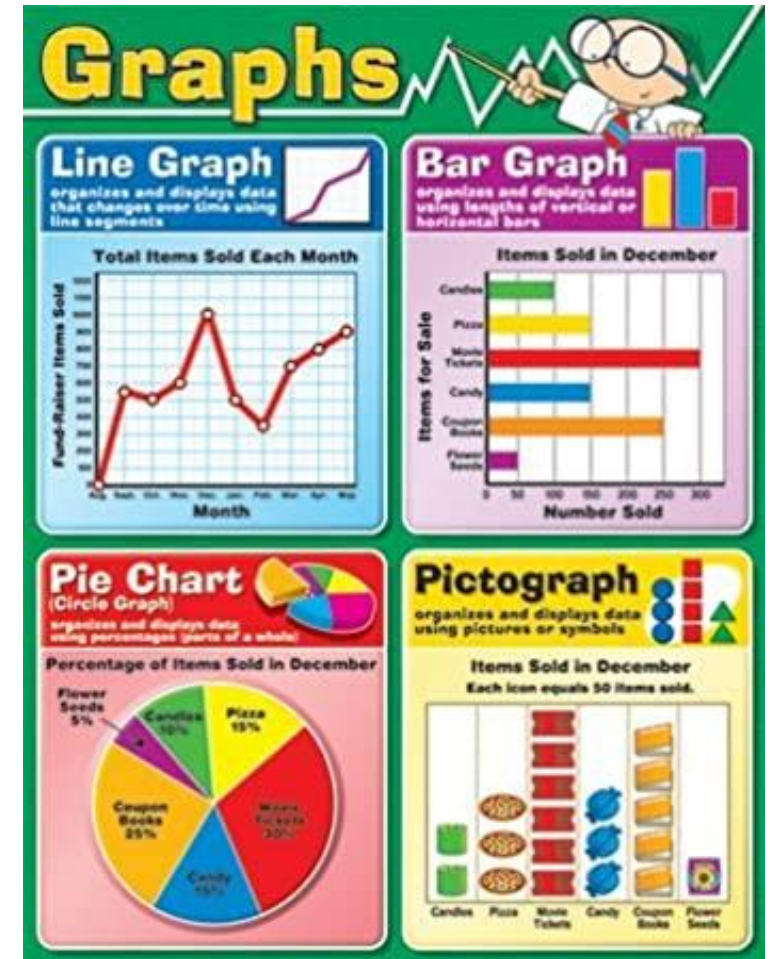
Quick review and warm-up exercise of pandas DataFrames

History of Data Visualization

Review and additional features of visualizing data with matplotlib

If there is time: Visualizing data with seaborn

Much of what we cover is a review and will be good practice for the midterm



# Announcement: Homework 5

Homework 5 has been posted!

It is due on Gradescope on **Sunday October 5<sup>th</sup> at 11pm**

- **Be sure to mark each question on Gradescope along with the **page that has the answers!****

Note: The homework is on the long side, but you already have the skills to do most of it, so please get started early

- It should be good practice/review for the midterm exam

# Midterm exam

Thursday October 9<sup>th</sup> **in person** during regular class time

- Exam is on paper

A practice exam has been posted

Also, please post a practice question to [Canvas discussions](#)

- If more than 50 students post reasonable questions, I will use one of the questions on the exam



# Midterm exam “cheat sheet”

You are allowed an exam “cheat sheet”

One page, double sided, that contains **only code**

- No code comments allowed
- E.g., `plt.plot(x, y, ".")`

Cheat sheet must be on a regular 8.5 x 11 piece of paper

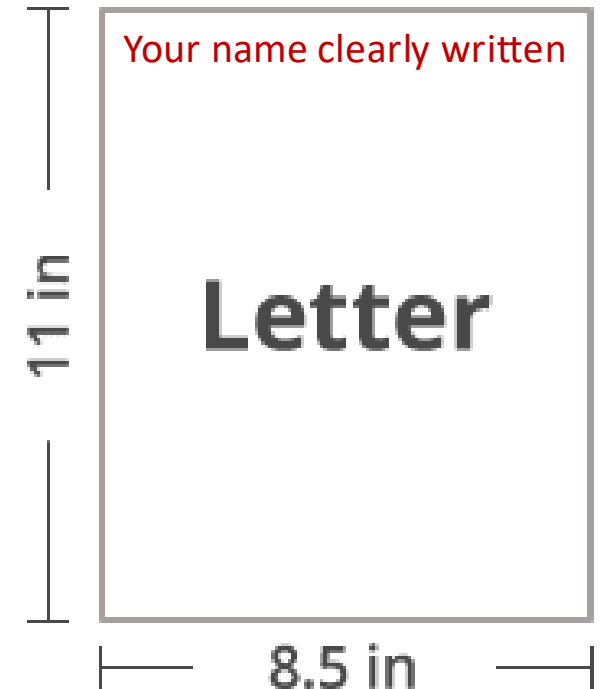
- Your name on the upper left of both sides of the paper

Strongly recommend making a typed list of all functions discussed in class and on the homework

- This will be useful beyond the exam

You must turn in your cheat sheet with the exam

- Failure to do so will result in a 20 point deduction



# Quick review: pandas DataFrames

Pandas DataFrames hold Table data

PLAYER	POSITION	TEAM	SALARY
str	str	str	f64
"Paul Millsap"	"PF"	"Atlanta Hawks"	18.671659
"Al Horford"	"C"	"Atlanta Hawks"	12.0
"Tiago Splitter..."	"C"	"Atlanta Hawks"	9.75625
"Jeff Teague"	"PG"	"Atlanta Hawks"	8.0
"Kyle Korver"	"SG"	"Atlanta Hawks"	5.746479

Selecting columns:

- `my_df[["col1", "col2"]]` # getting multiple columns using a list
- `my_df.drop(columns = ["col1", "col2"])` # removing columns

Extracting rows:

- `my_df.iloc[0]` # getting a row by number
- `my_df.loc["index_name"]` # getting a row by Index value
- `my_df[my_df["col_name"] == 7]` # getting rows using a Boolean mask
- `my_df.query("col_name == 7")` # getting rows using the query function

# Quick review: pandas DataFrames

Sorting rows of a DataFrame

```
my_df.sort_values("col_name", ascending = False)  # sort from largest to smallest
```

Adding a new:

```
my_df["new_col"] = values_array
```

Renaming a column:

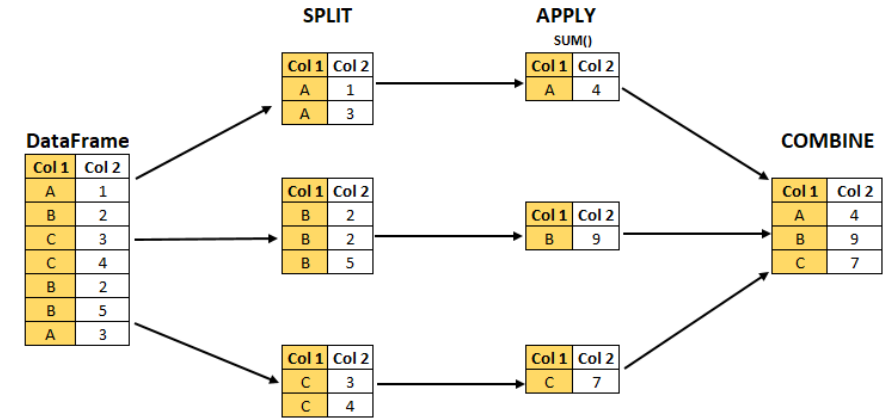
```
rename_dictionary = {"old_col_name": "new_col_name"}  
my_df.rename(columns = rename_dictionary )
```

# Quick review: Creating aggregate statistics by group

We can get statistics separately by group:

```
dow.groupby("Year").agg("max")
```

```
my_df.groupby("group_col_name").agg(  
    new_col1 = ('col_name', 'statistic_name1'),  
    new_col2 = ('col_name', 'statistic_name2'),  
    new_col3 = ('col_name', 'statistic_name3')  
)
```



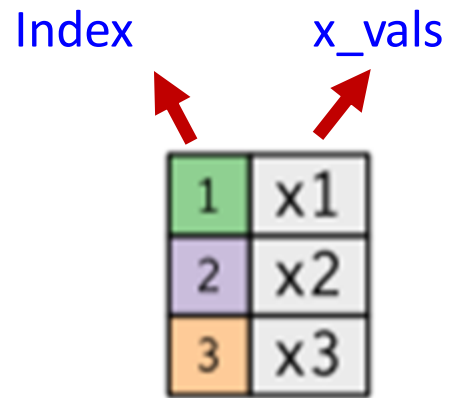
```
nba_salaries.groupby("TEAM").agg(  
    max_salary = ("SALARY", "max"),  
    min_salary = ("SALARY", "min"),  
    first_player = ("PLAYER", "min")  
)
```



# Review: Joining

Suppose we have two DataFrames (or Series) called **x\_df** and **y\_df**

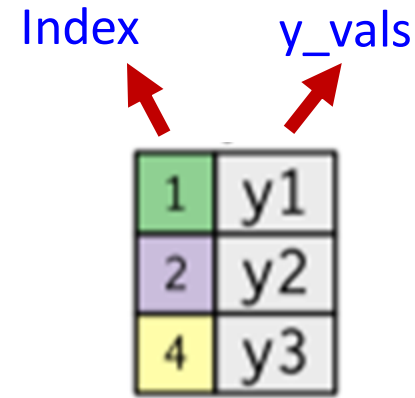
- **x\_df** have one column called **x\_vals**
- **y\_df** has one column called **y\_vals**



A diagram of a DataFrame with two columns. The first column is labeled 'Index' with a red arrow pointing to it. The second column is labeled 'x\_vals' with a red arrow pointing to it. The DataFrame contains three rows: the first row has index 1 and value x1; the second row has index 2 and value x2; the third row has index 3 and value x3.

Index	x_vals
1	x1
2	x2
3	x3

**DataFrame: x\_df**



A diagram of a DataFrame with two columns. The first column is labeled 'Index' with a red arrow pointing to it. The second column is labeled 'y\_vals' with a red arrow pointing to it. The DataFrame contains three rows: the first row has index 1 and value y1; the second row has index 2 and value y2; the third row has index 4 and value y3.

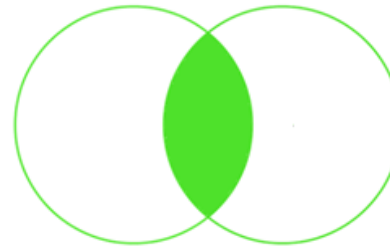
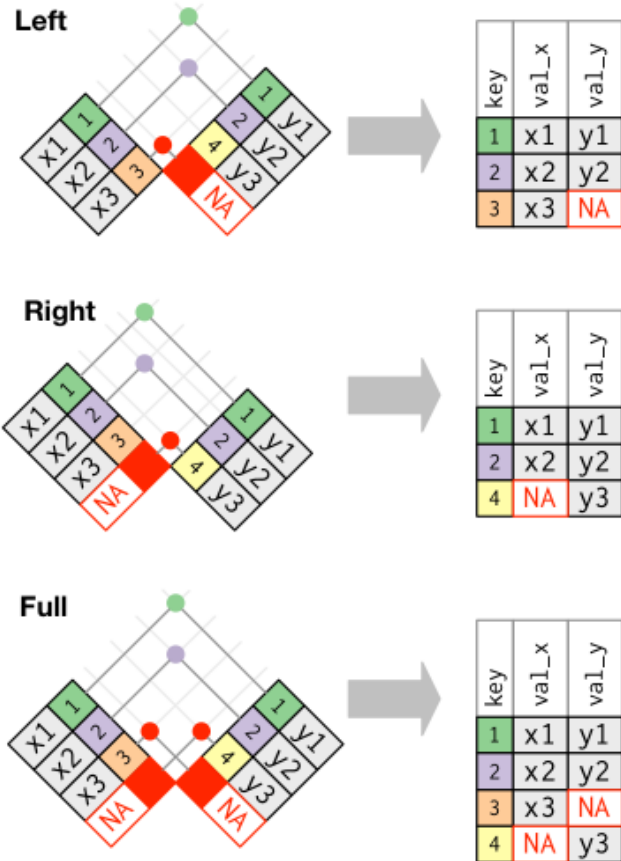
Index	y_vals
1	y1
2	y2
4	y3

**DataFrame: y\_df**

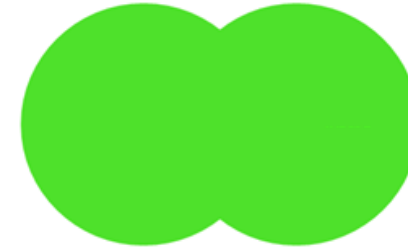
We can join these two DataFrames into a single DataFrame by aligning rows with the same Index value using the general syntax: **x\_df.join(y\_df)**

- i.e., the new joined data frame will have two columns: **x\_vals**, and **y\_vals**

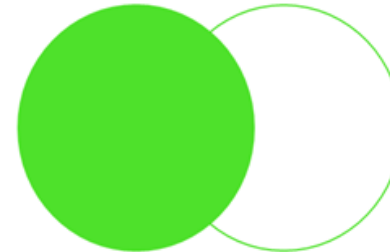
# Review: Joining



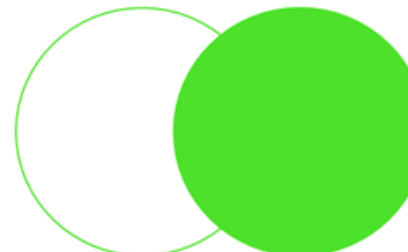
INNER JOIN



FULL OUTER JOIN



LEFT JOIN



RIGHT JOIN

`x_df.join(y_df, how = "left")`    # based on Index

`x_df.merge(y_df, how = "inner", left_on = "x_col", right_on = "y_col")`    # based on columns

# Questions?



Let's do a quick review and a few warm-up exercise in Jupyter!

# Data visualization!



# A very brief history of data visualization...

*Statistical Science*

2008, Vol. 23, No. 4, 502–535

DOI: 10.1214/08-STS268

© Institute of Mathematical Statistics, 2008

## **The Golden Age of Statistical Graphics**

**Michael Friendly**

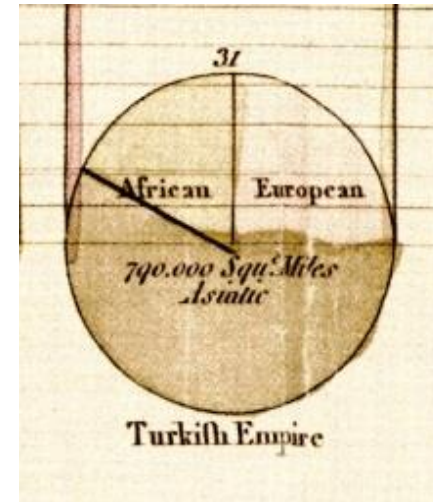
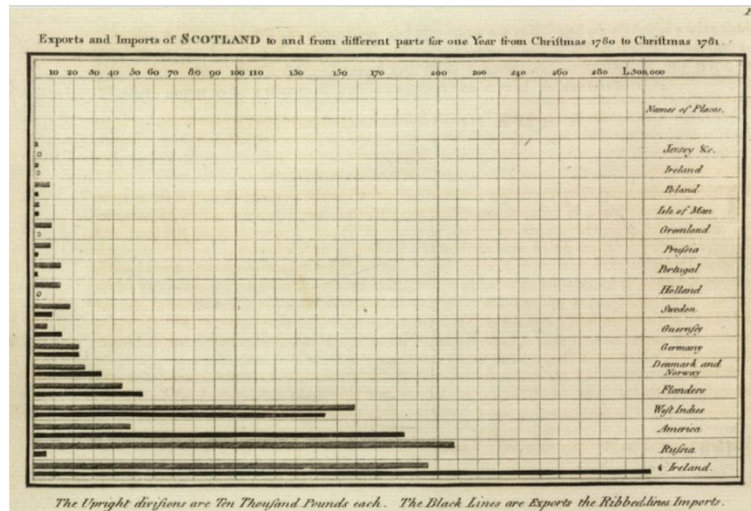
# Data visualization

What are some reasons we visualize data rather than just reporting statistics?

# A very brief history of data visualization

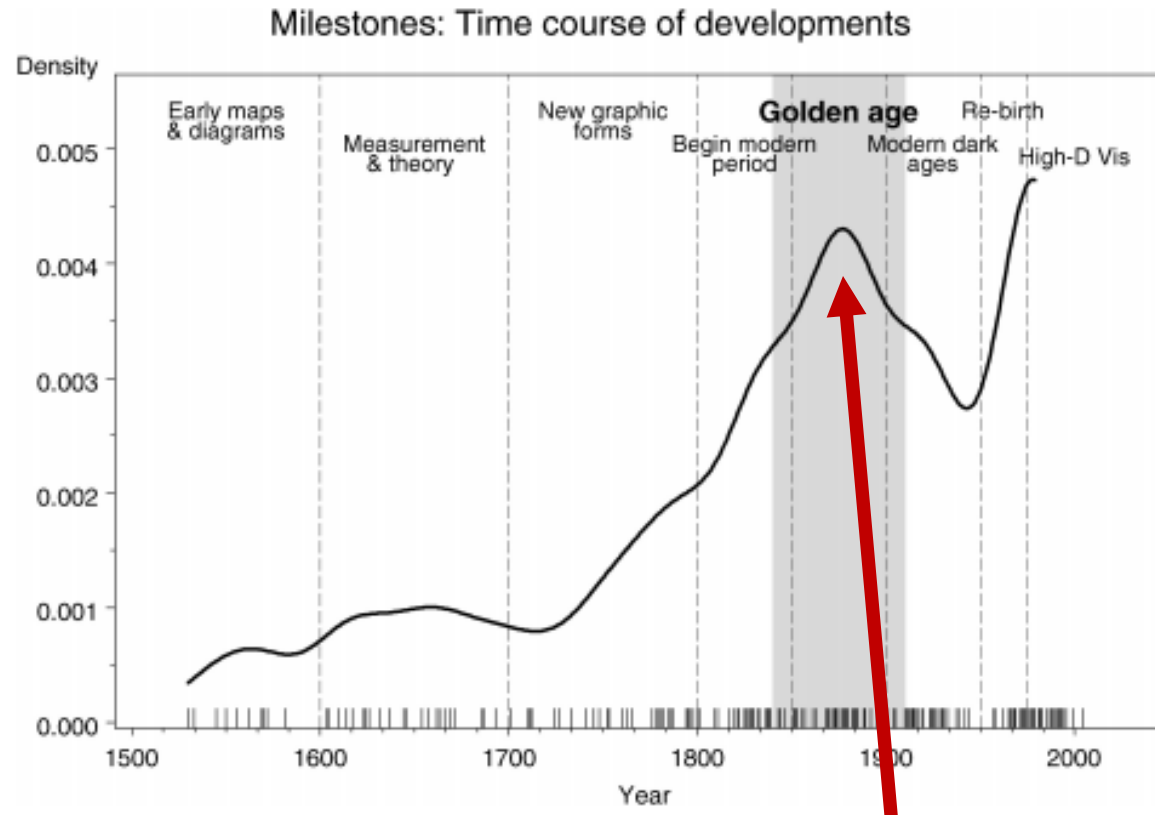
The age of modern statistical graphs began around the beginning of the 19<sup>th</sup> century

[William Playfair](#) (1759-1823) is credited with inventing the line graph, bar chart and pie chart



# A very brief history of data visualization

According to Friendly, statistical graphics researched its golden age between 1850-1900



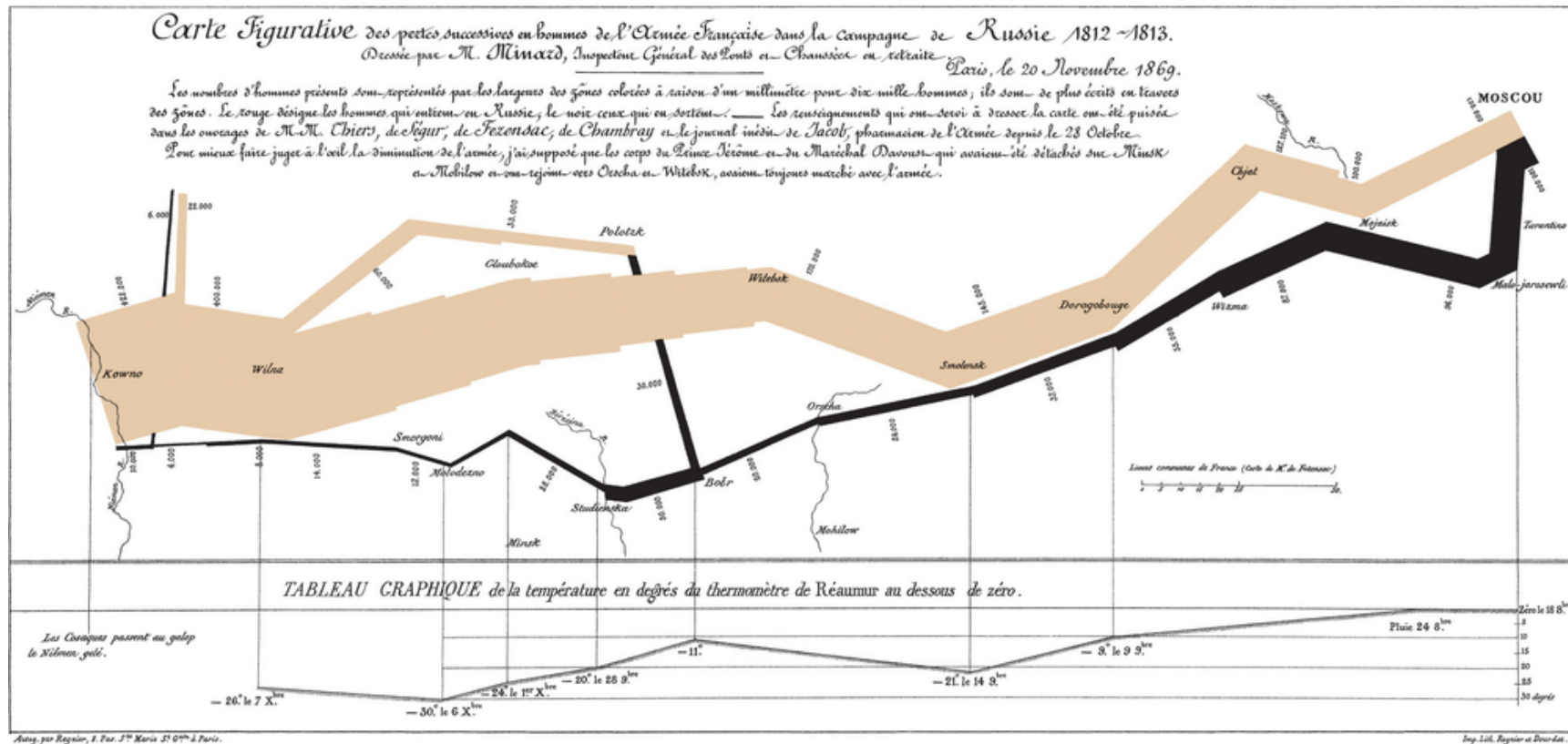


# A very brief history of data visualization

## Joseph Minard (1781-1870)

Map of Napoleon's march on Russia

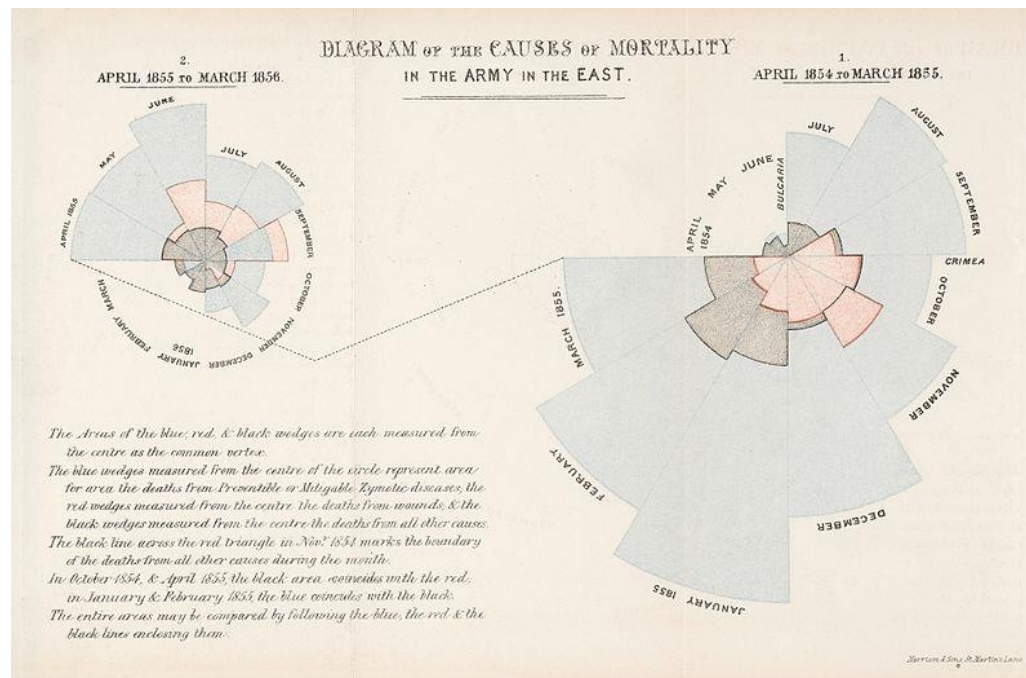
"It may well be the best statistical graphic ever drawn" – E. Tufte



# A very brief history of data visualization

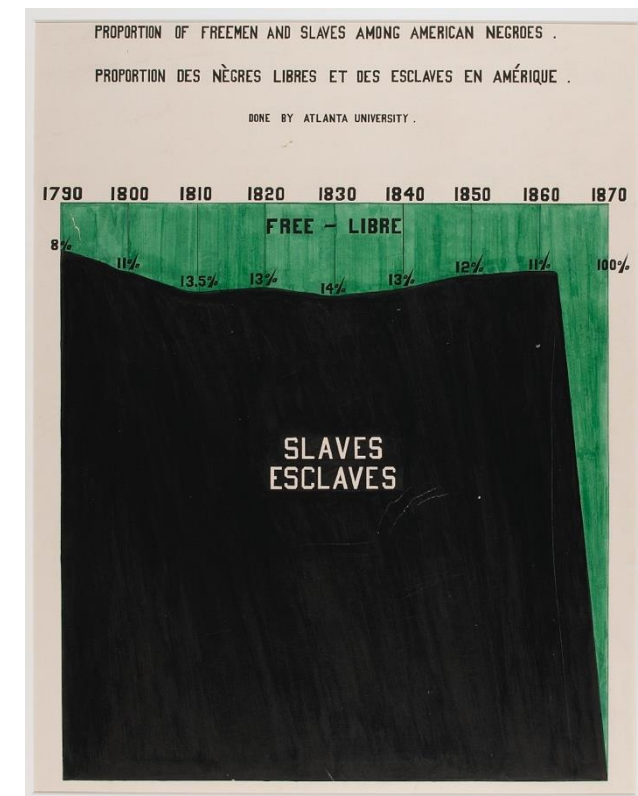
[Florence Nightingale](#) (1820-1910)

Causes of mortality in the army in the east



[W.E.B. Du Bois](#) (1868-1963)

Percent of African Americans who were slaves

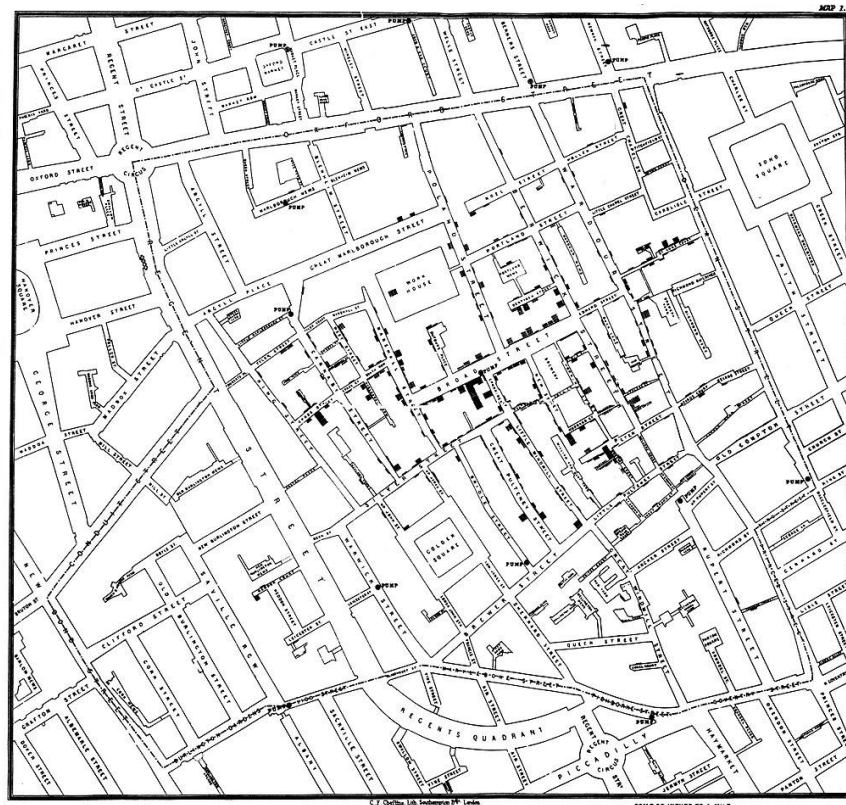


[See Du Bois Visualization Challenge](#)

# A very brief history of data visualization

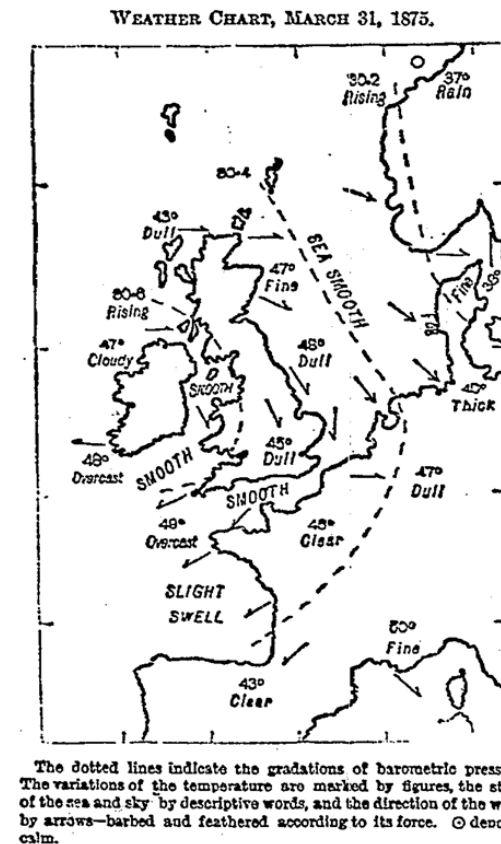
John Snow (1813-1858)

Clusters of cholera cases in London epidemic of 1854



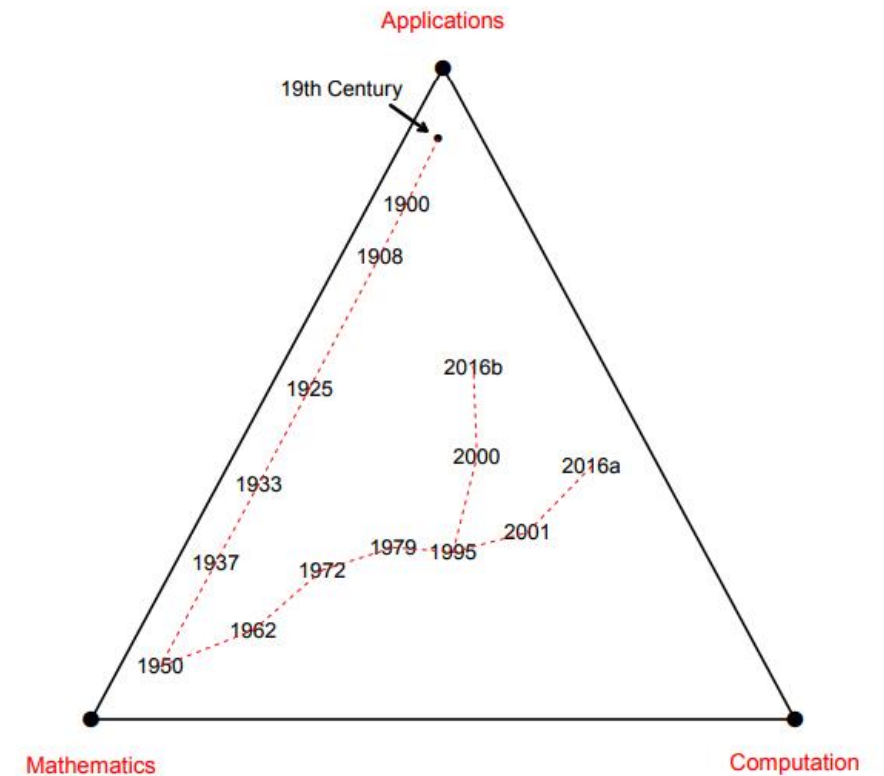
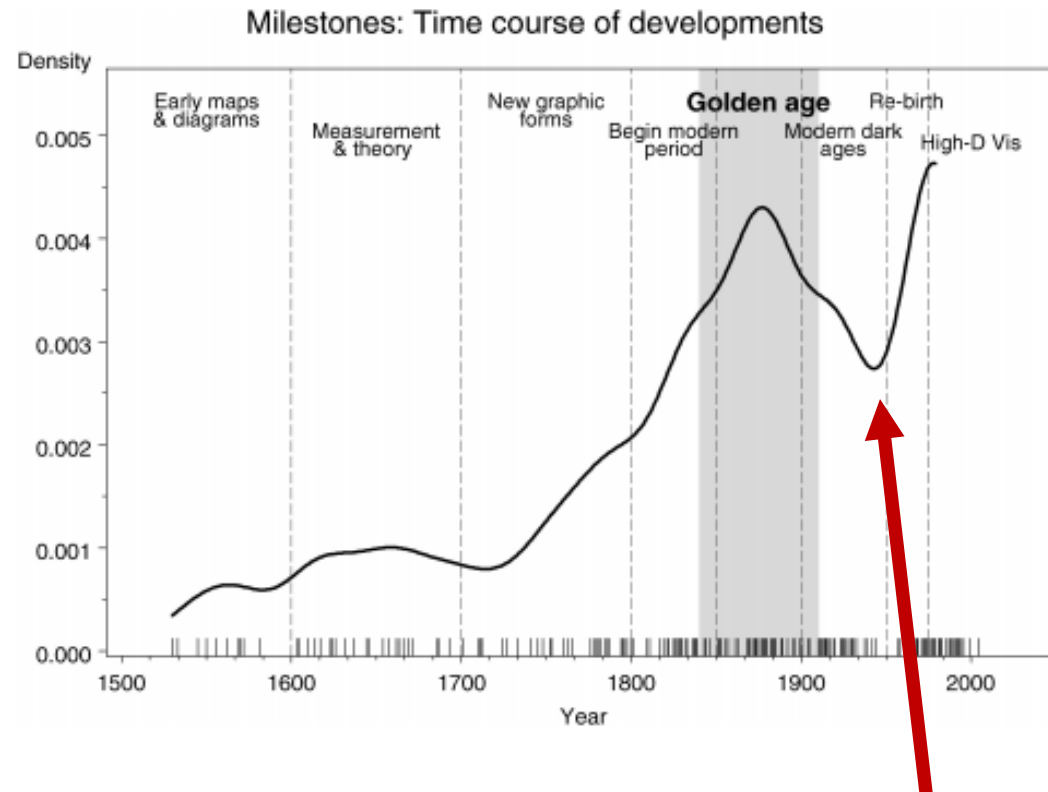
Francis Galton (1822-1911)

First weather map published in a newspaper (1875)



# A very brief history of data visualization

“Graphical dark ages” around 1950



Computer Age Statistical Inference, Efron and Hastie



# A very brief history of data visualization

Currently undergoing a “Graphical re-birth”

Box plot

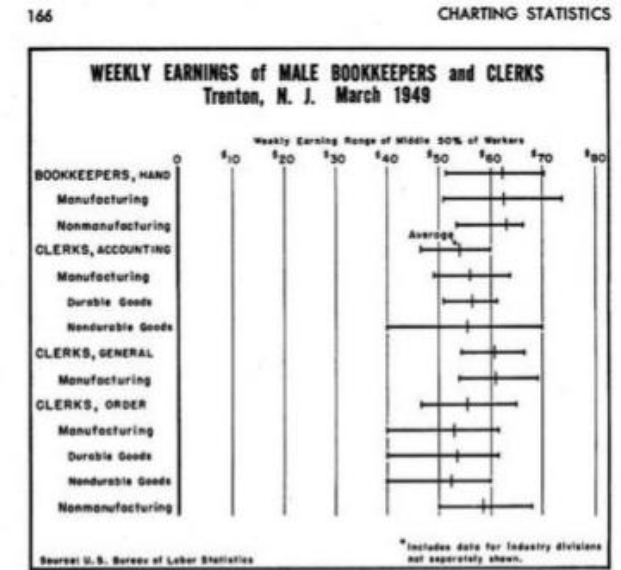
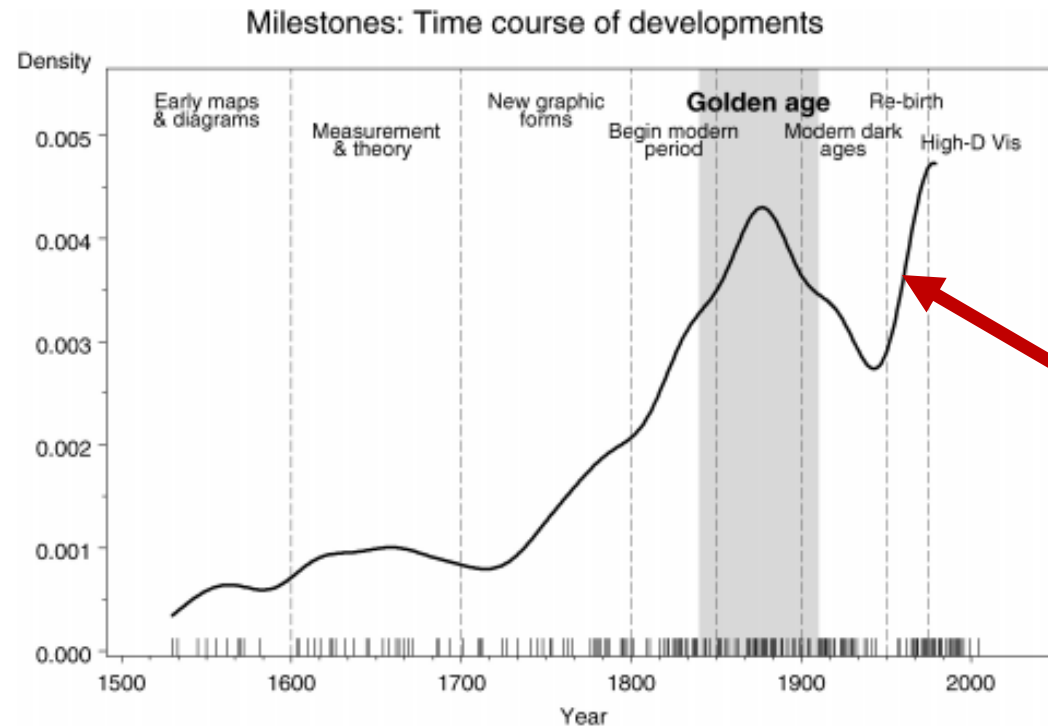


Fig. 6-23. The range bar and symbol.

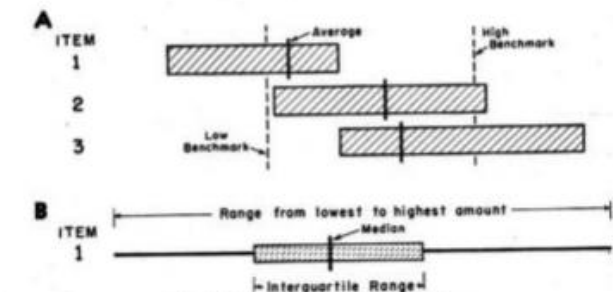
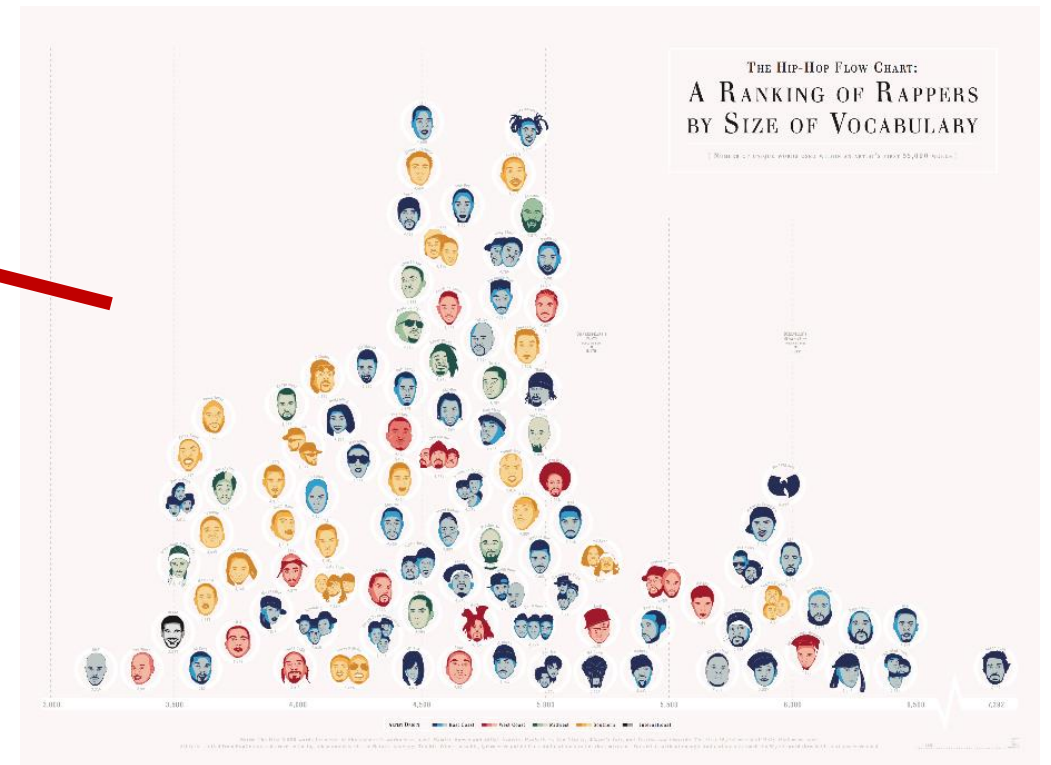
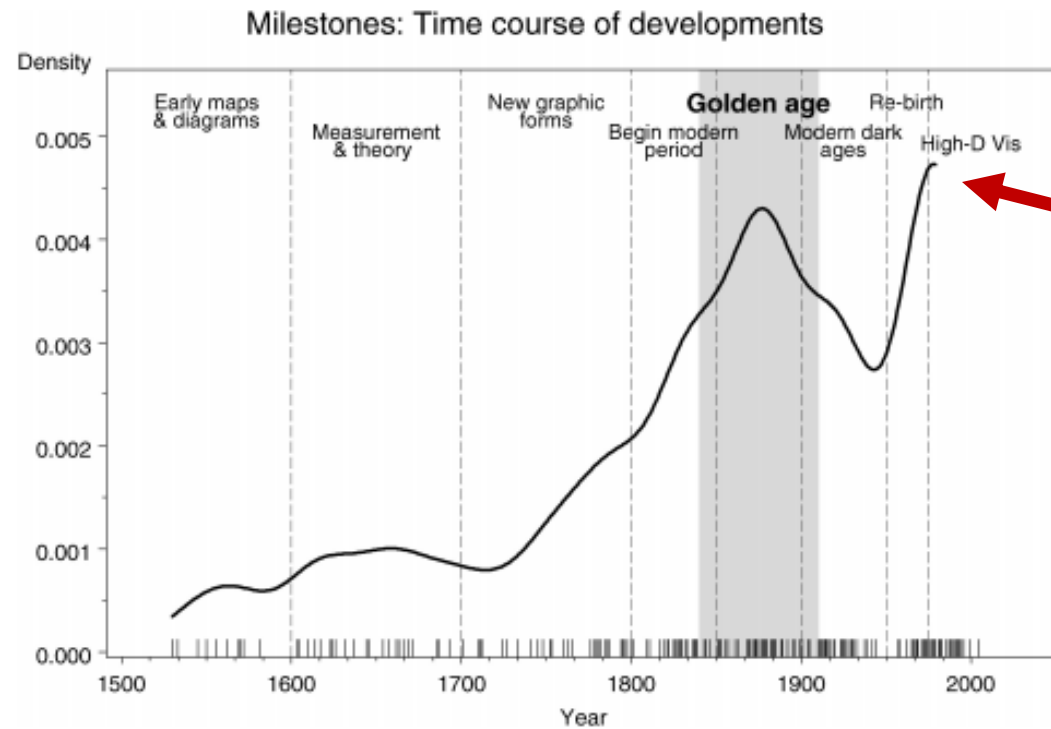


Fig. 6-24. Various uses of the range bar.

[Spear 1952](#), [Tukey 1970](#)

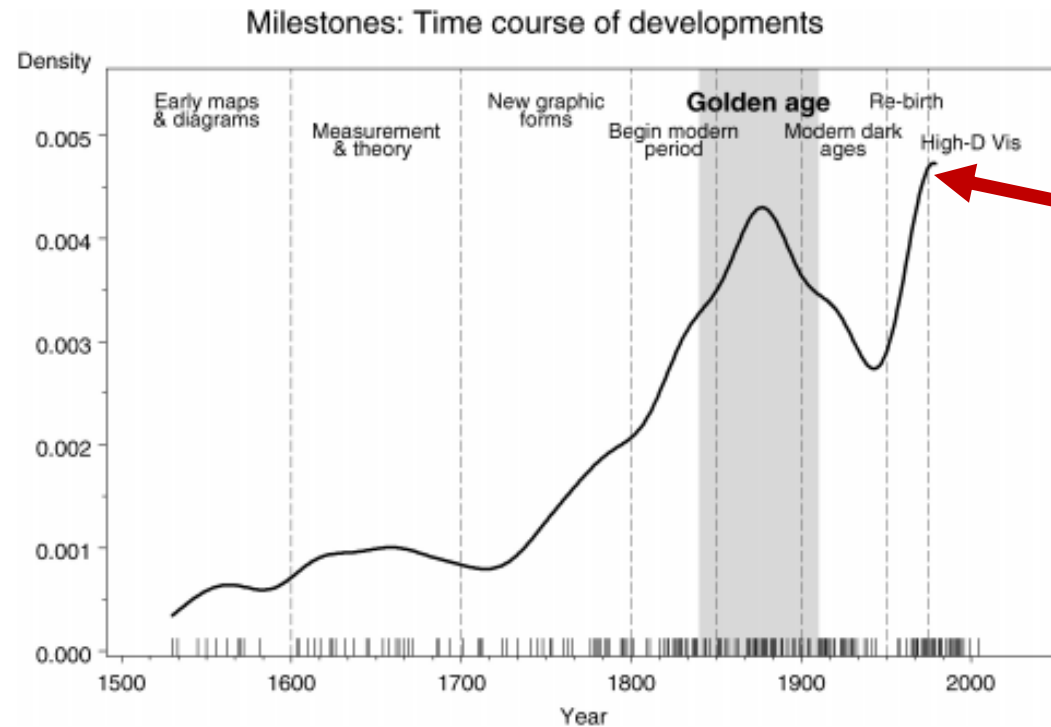
# A very brief history of data visualization

Currently undergoing a “Graphical re-birth”



# A very brief history of data visualization

Currently undergoing a “Graphical re-birth”



Hans Rosling's gapminder

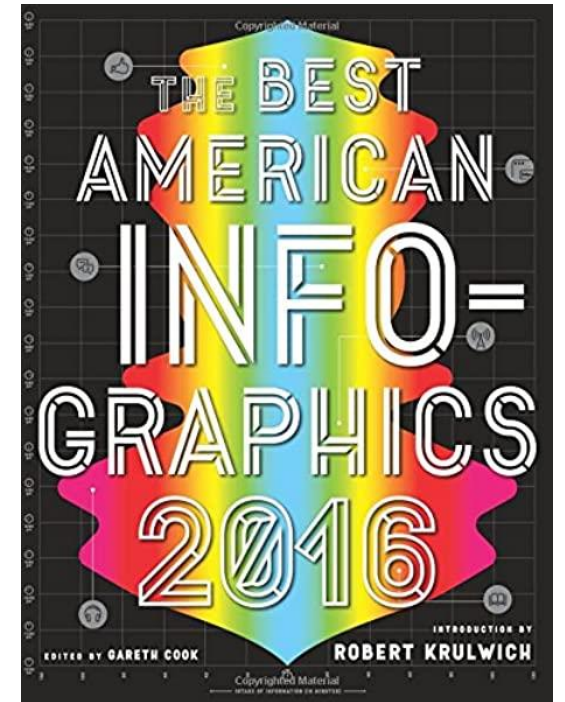
- [Simple version](#)
- [TV special effects](#)
- [Ted Talk](#)

# Coming up on homework 5: find an interesting data visualization...

Homework 5 : Find an interesting data visualization

- <https://www.reddit.com/r/dataisbeautiful/>
- <https://flowingdata.com/>

We will do a little show and tell in class





Review and continuation of data visualization

# Review of visualizing data with matplotlib

We have already discussed creating a few data visualizations using [matplotlib](#) which is a comprehensive library for creating static, animated, and interactive visualizations.



Let's now review and expand our use of this package

We will then discuss another visualization library called "seaborn" which makes it even easier to create beautiful looking graphics



# Review of visualizing one quantitative variable

Q: How can we visualize one quantitative variable?

A: Histograms!

A: Box plots!

```
plt.hist(data1, edgecolor = "k", alpha = .5);
```

```
plt.hist(data2, edgecolor = "k", alpha = .5);
```

```
plt.boxplot( [data1, data2], tick_labels = ["lab1", "lab2"])
```

# Visualizing time series

Q: How can we visualize a time series?

A: Line plot

```
plt.plot(x1, y1, '-o', label = 'First line');
```

```
plt.plot(x2, y2, '-o', label = 'Second line');
```

```
plt.legend();
```

Let's review this in Jupyter!

# Review of visualizing two quantitative variables

Q: How can we visualize two quantitative variables?

A: Scatter plot!

```
plt.plot(x_array, y_array, '.');
```

We can also use the matplotlib `plt.scatter()` function...

# Scatter plots

`plt.scatter(x, y)` has additional useful arguments such as:

- `s`: specified the size of each point
- `color`: specifies the color of each point
- `marker`: specifies the shape of each point

Let's explore this in Jupyter!

# Review of visualizing categorical data

Q: How can we visualize categorical data?

A: Bar plots and pie charts

```
import matplotlib.pyplot as plt  
plt.pie(data, labels = label_names)  
plt.bar(labels, data)
```

```
plt.xlabel("Drink type")  
plt.ylabel("Number of drinks")
```

# Subplots: pyplot interface

Matplotlib makes it easy to create multiple subplots within a larger figure

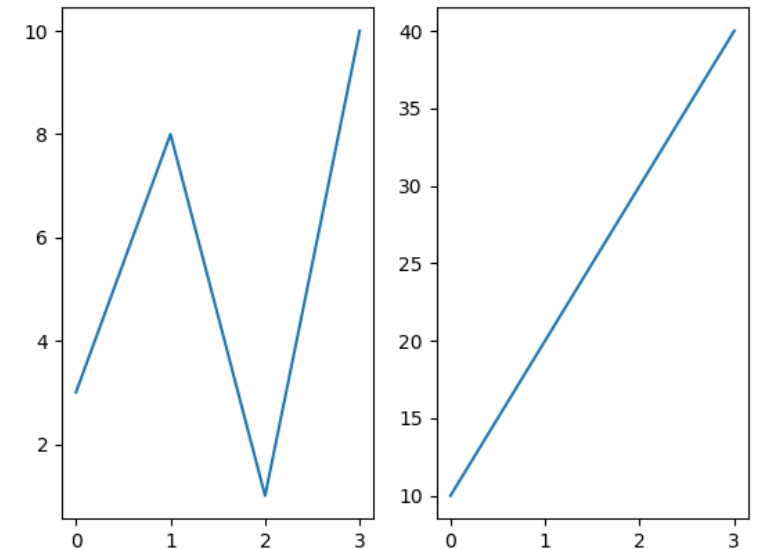
1 row →  
2 columns →

```
plt.subplot(1, 2, 1);  
plt.plot(x1, y1);
```

plot on the first subplot

```
plt.subplot(1, 2, 2);  
plt.plot(x2, y2);
```

plot on the second subplot



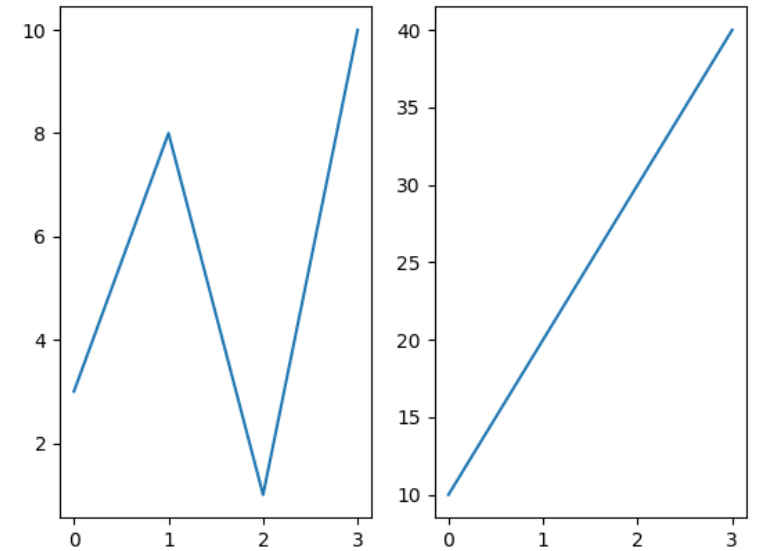


# Subplots: axes interface

Matplotlib makes it easy to create multiple subplots within a larger figure

```
fig, ax = plt.subplots(1, 2); # notice subplots
ax[0].plot(x1, y1);

ax[1].plot(x2, y2);
ax.set_ylabel("y label") # notice set_ylabel
```

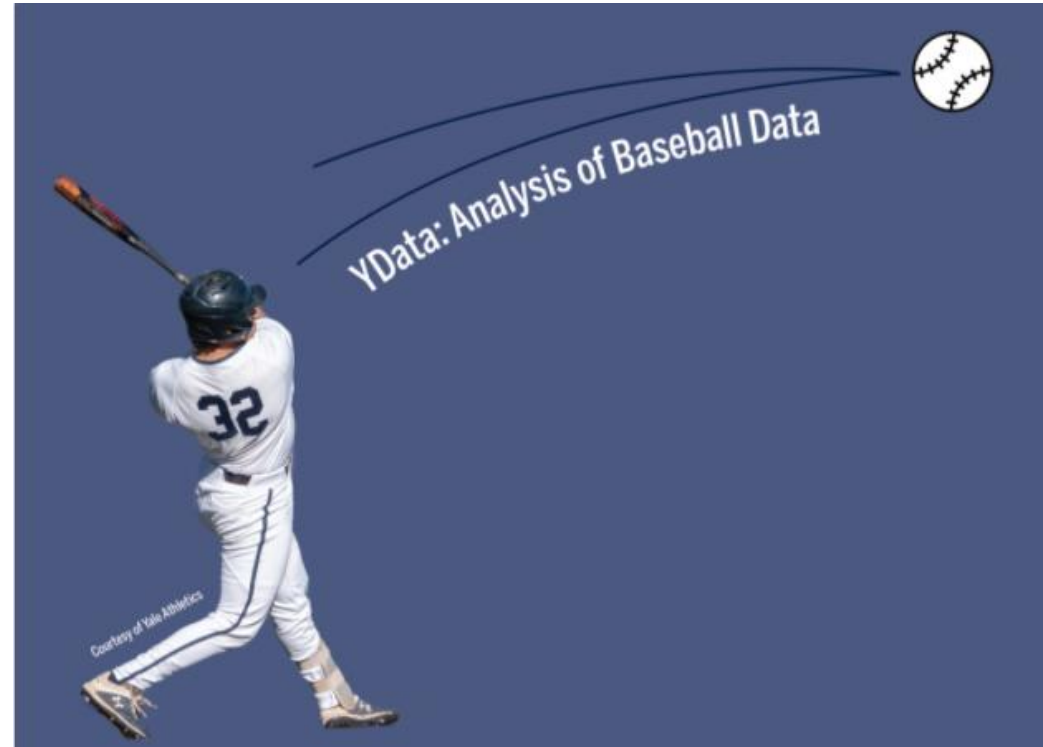


Let's explore this in Jupyter!

# Using matplotlib as a canvas

We can also use matplotlib as a canvas to create general figures

For example, in my Ydata baseball class, we drew a baseball diamond and illustrated where players were on base with red circles.



Let's briefly explore this in Jupyter!

# Seaborn

“Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.”

- i.e., it will create better looking plots that are easier to make

There are ways to create visualizations in seaborn:

1. **axes-level** functions that plot on a single axis
2. **figure-level** functions that plot across multiple axes

We will focus on figure level plots



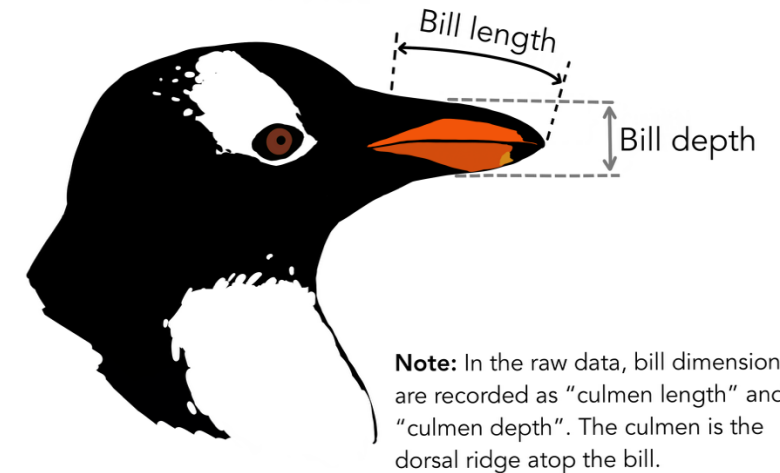
To make plots better looking we can set a theme

```
import seaborn as sns
```

```
sns.set_theme()
```

# Inspiration: Palmer penguins

To explore seaborn, let's look at some data on penguins!



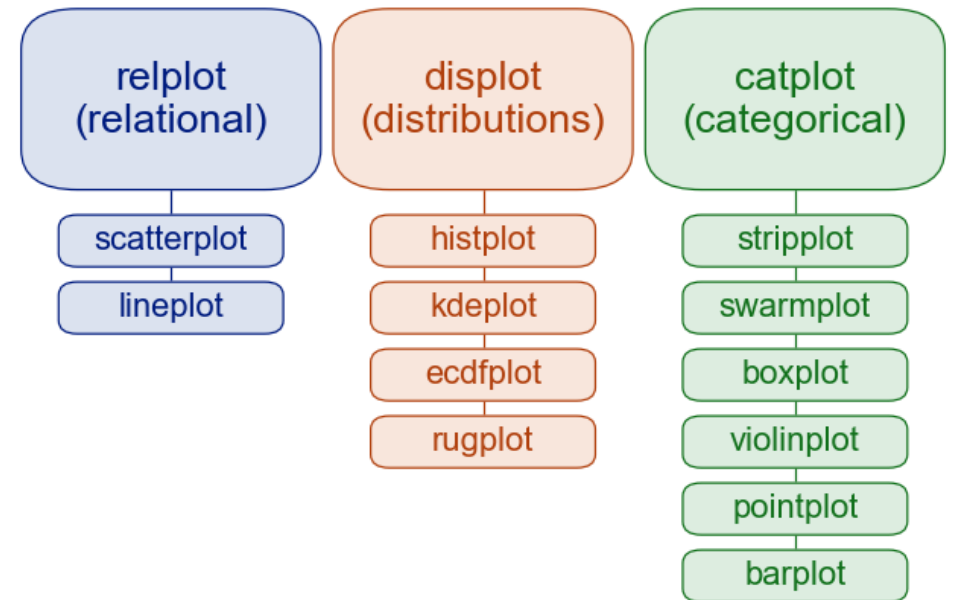
# Seaborn figure level plots

Figure level plots are grouped based on the types of variables being plotted

In particular, there are plots for:

1. Two quantitative variables
  - `sns.relplot()`
2. A single quantitative variable
  - `sns.displot()`
3. Quantitative variable compared across different categorical levels
  - `sns.catplot()`

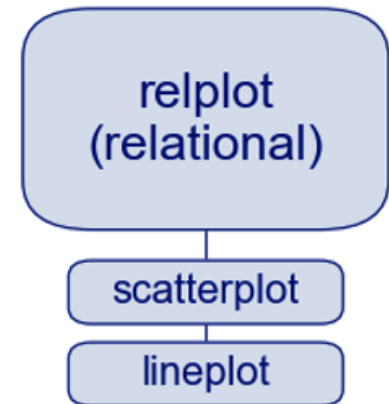
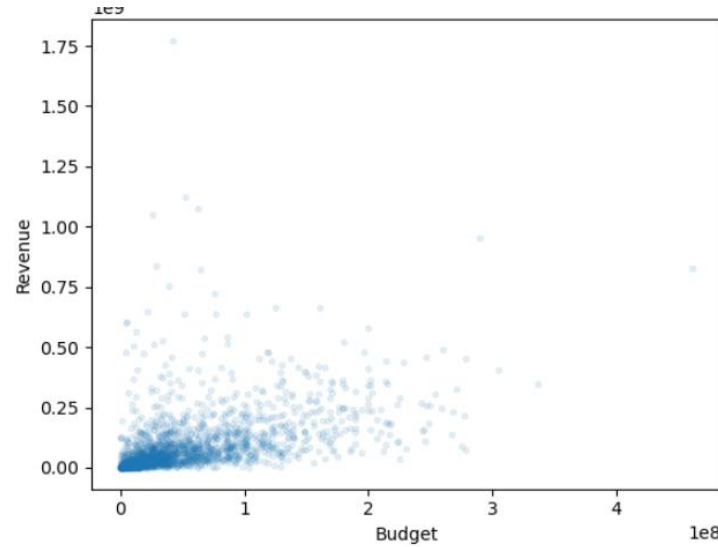
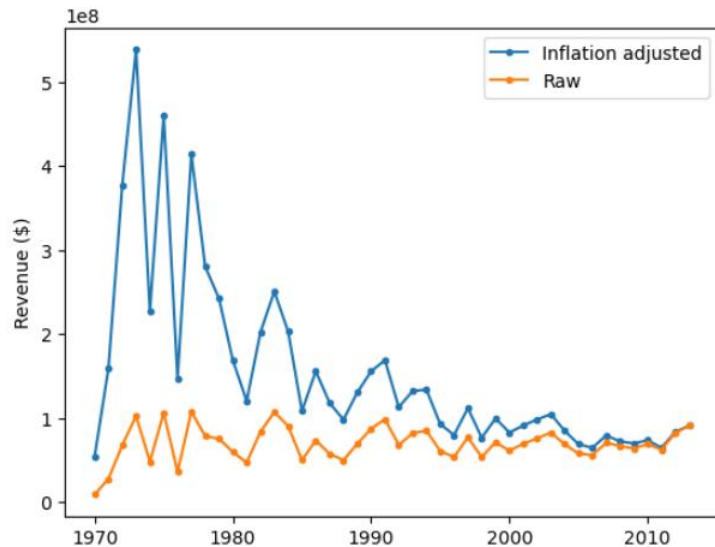
## Figure level plots



# Plots for two quantitative variable

What types of plots have we seen for assessing the relationships between two quantitative variable?

- Line plots and scatter plots!

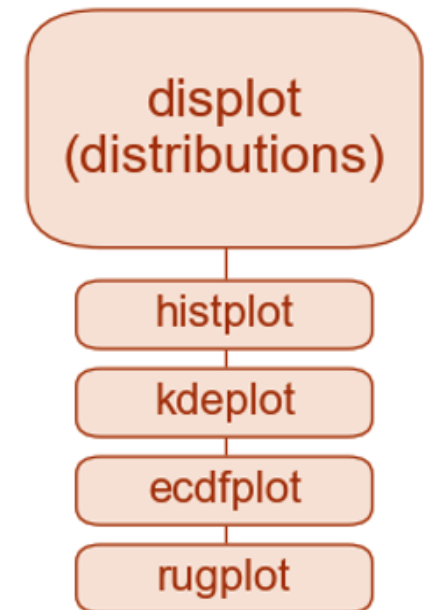
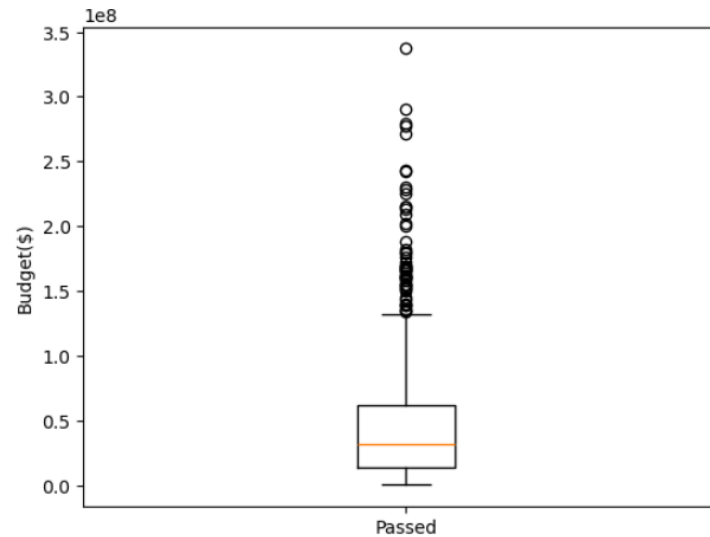
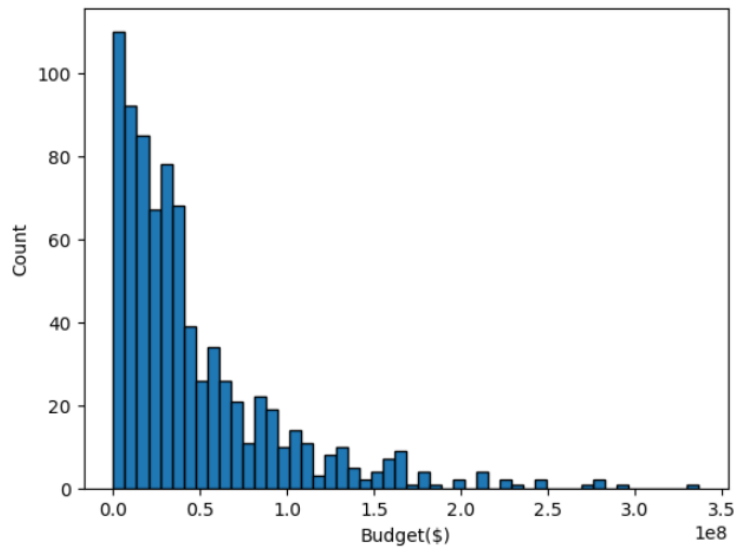


Let's explore this in Jupyter!

# Plots for a single quantitative variable

What types of plots have we seen for plotting a single quantitative variable?

- Histograms and boxplots

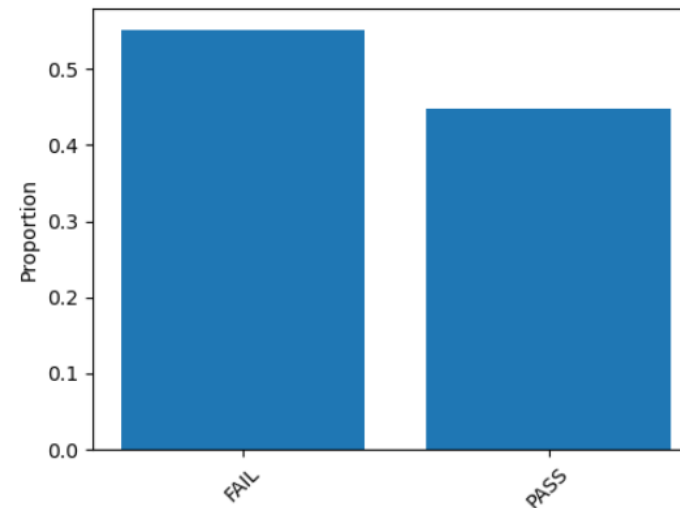
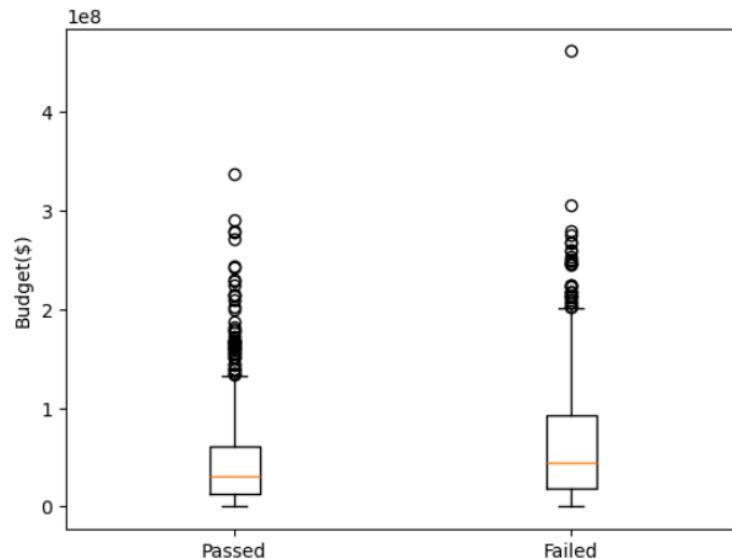


Let's explore this in Jupyter!

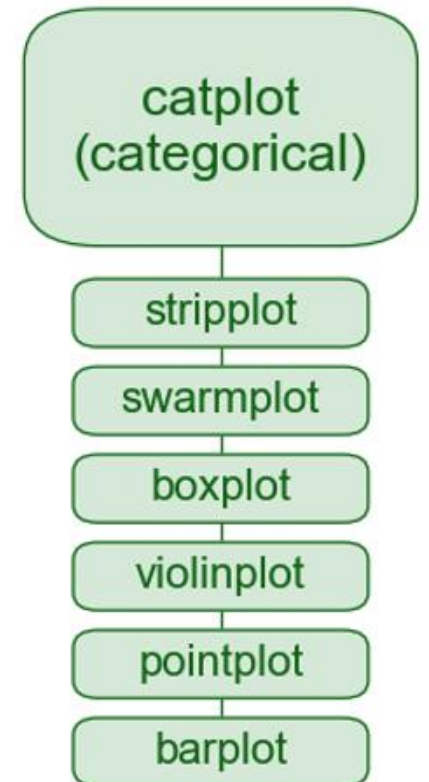
# Plots for quantitative data comparing across different categorical levels

What types of plots have we seen comparing quantitative data at different levels of a categorical variable?

- Side-by-side boxplots, barplots (sort of)



Let's explore this in Jupyter!





Next class: Interactive graphics