YData: Introduction to Data Science

Class 11: Intro to data visualization

Overview

Quick review of pandas DataFrames and joining DataFrames

History of Data Visualization

Visualizing data with matplotlib!

Announcement: Homework 5

Homework 5 has been posted!

It is due on Gradescope on Sunday February 26th at 11pm

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

Announcement: class project

The final project is a 6-10 page Jupyter notebook report where you analyze your own data to address a question that you find interesting

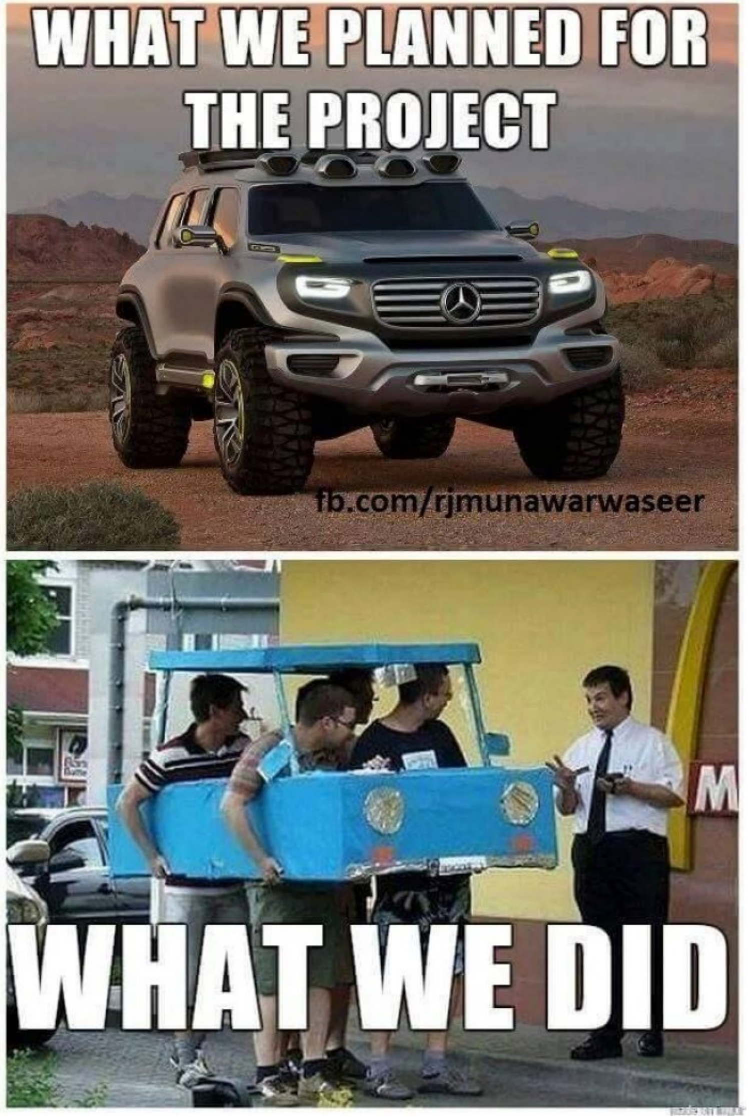
- It's a chance to practice everything you've learned in class!

The goal of is to present a clear and compelling data analysis showing a few interesting results!

- Ideally something you are proud of that you could show off to potential future employers, etc.

A few sources for data sets are listed on Canvas

- You can use data you collect as well. If you use data for another class your work must be unique for each class.



Announcement: class project

A draft of the project is due on April 7th

- I'm telling you about this early so:
    - If you want to think/work on the project over break you can
    - If don't want to think about it over break you don't have to

There will be a "peer review" period where you will give and receive feedback on three other projects

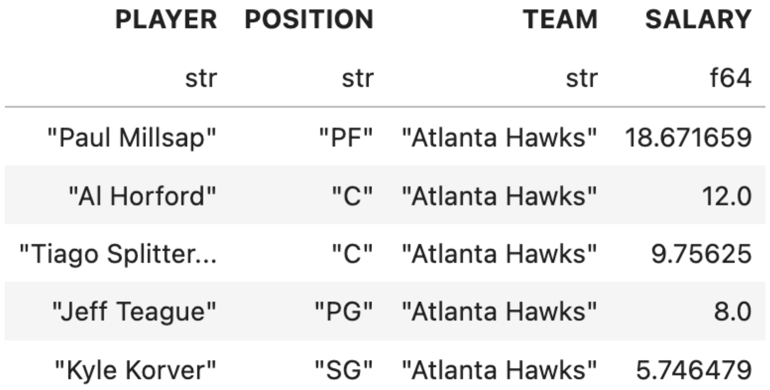
- Instructions for how to do the review will be given

The final version will be due on April 30th

A Jupyter notebook "template" project will be available soon

Questions?

Quick review: pandas DataFrames



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

Pandas DataFrame hold Table data

Selecting columns:

- my_df[[“col1”, “col2”]].copy() # getting multiple columns using a list

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

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

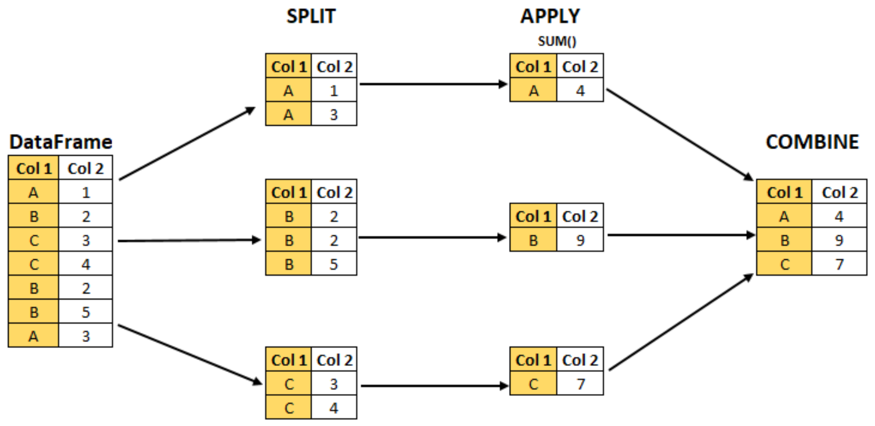
Renaming a column:

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

Creating aggregate statistics by group

We can get statistics separately by group:

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



DataFrame

Col 1	Col 2
A	1
B	2
C	3
C	4
B	2
B	5
A	3

SPLIT

Col 1	Col 2
A	1
A	3

Col 1	Col 2
B	2
B	2
B	5

Col 1	Col 2
C	3
C	4

APPLY

SUM()

Col 1	Col 2
A	4

Col 1	Col 2
B	9

Col 1	Col 2
C	7

COMBINE

Col 1	Col 2
A	4
B	9
C	7

```
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')
)
```

Creating a DataFrame "by hand"

We can create small DataFrames by entering data manually from a dictionary using the df.DataFrame() function

To do this we first create a dictionary, where:

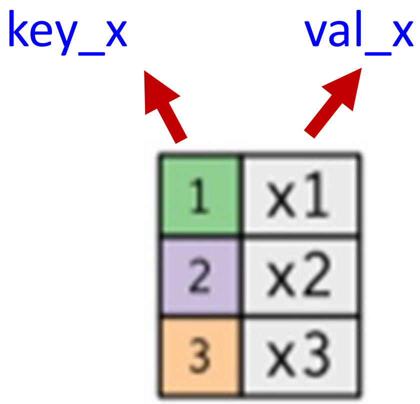
- The **keys** are the names of the columns
- The **values** are the data in the rows

```
my_dictionary = {"col1": ["a", "b", "c"], "col2": [1, 2, 3]}
my_df = pd.DataFrame(my_dictionary)
```

Review: Joining data frames

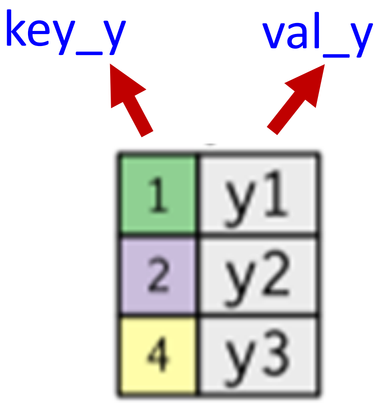
Suppose we have two DataFrames (or Series) called **x_df** and **y_df**

- x_df have two columns called key_x, and val_x
- y_df has two columns called key_y and val_y



key_x	val_x
1	x1
2	x2
3	x3

DataFame: x_df



key_y	val_y
1	y1
2	y2
4	y3

DataFrame y_df

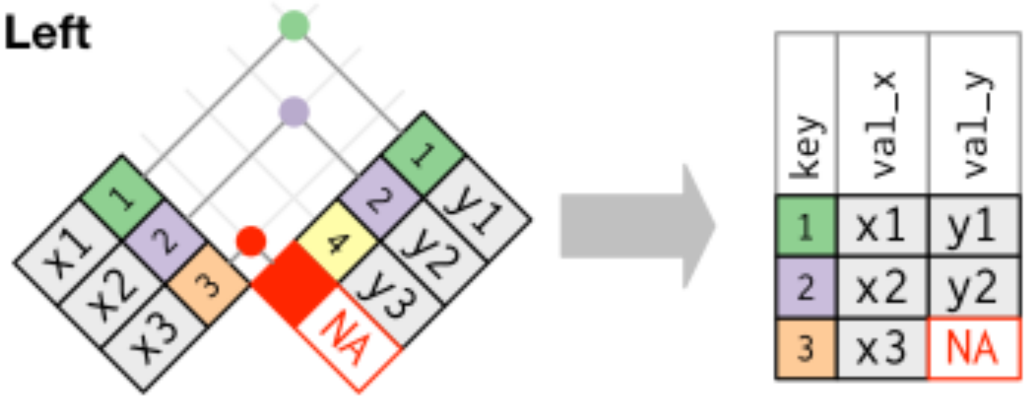
Joins have the general form:

```
x_df.merge(y_df, left_on = "key_x",  right_on = "key_y")
```

Left joins

Left joins keep all rows in the left table.

Data from right table is added when there is a matching key, otherwise NA as added.



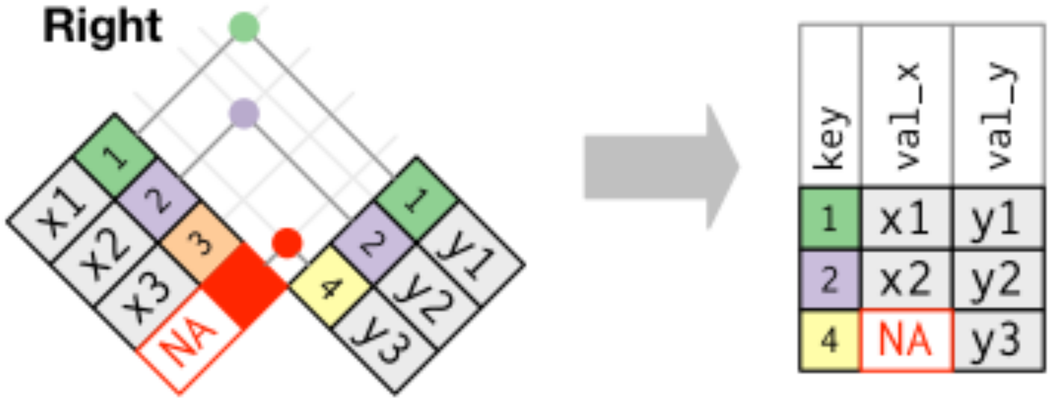
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

```
x_df.merge(y_df, how = "left", left_on = "key_x",  right_on = "key_y")
```

Right joins

Right joins keep all rows in the right table.

Data from left table added when there is a matching key, otherwise NA as added.

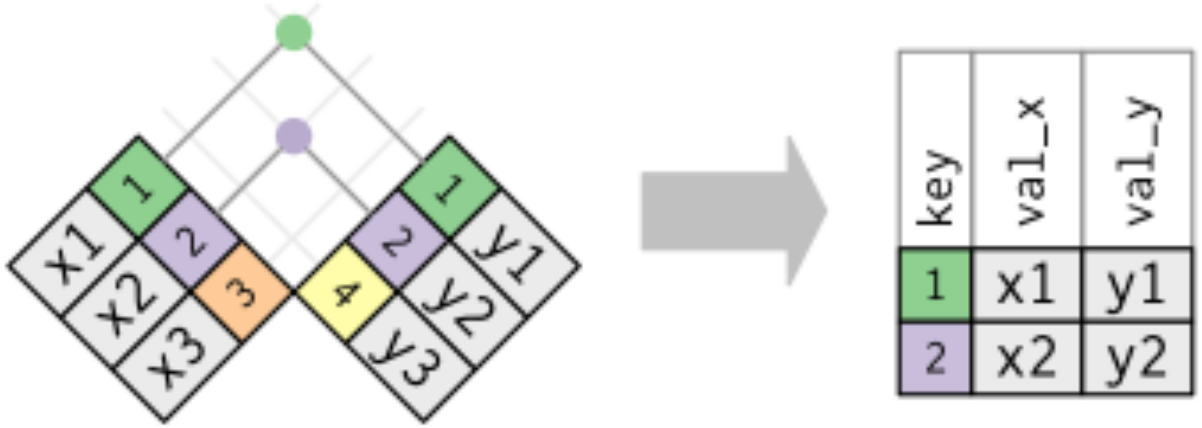


key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

```
x_df.merge(y_df, how = "right", left_on = "key_x",  right_on = "key_y")
```

Inner joins

Inner joins only keep rows in which there are matches between the keys in both tables.



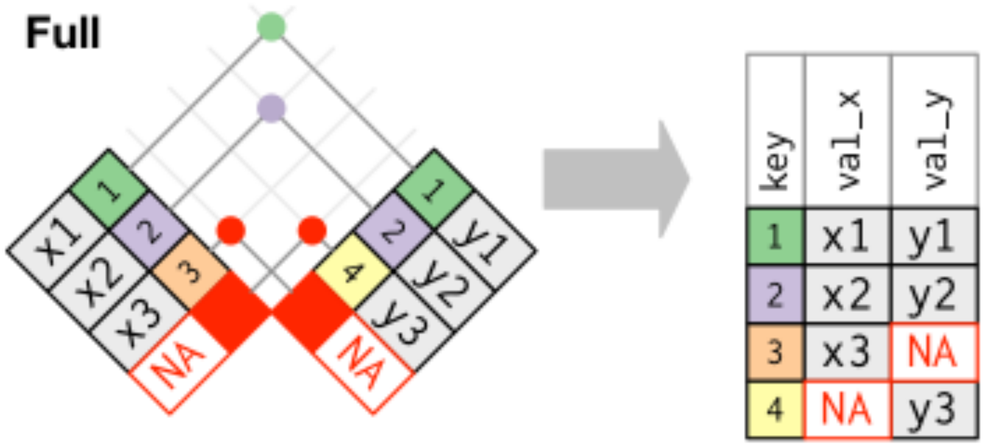
key	val_x	val_y
1	x1	y1
2	x2	y2

```
x_df.merge(y_df, how = "inner", left_on = "key_x",  right_on = "key_y")
```

Full (outer) joins

Full joins keep all rows in both table.

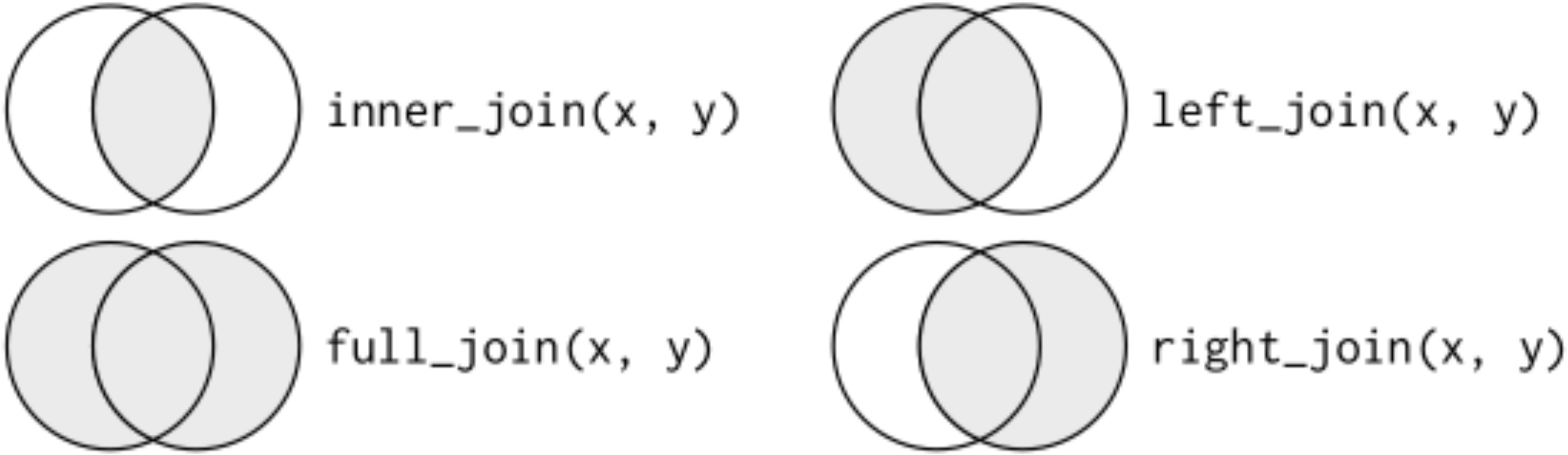
NAs are added where there are no matches.



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3

```
x_df.merge(y_df, how = "outer", left_on = "key_x",  right_on = "key_y")
```

Summary



Joining on Index values

If two DataFrames **have the same Index values** then we can join them using the .join() method instead of the .merge() method

The .join() method is very similar to .merge() except we don't need to specify left_on and right_on arguments since the DataFrames are being joined by their Indexes

An example of a left join would be:

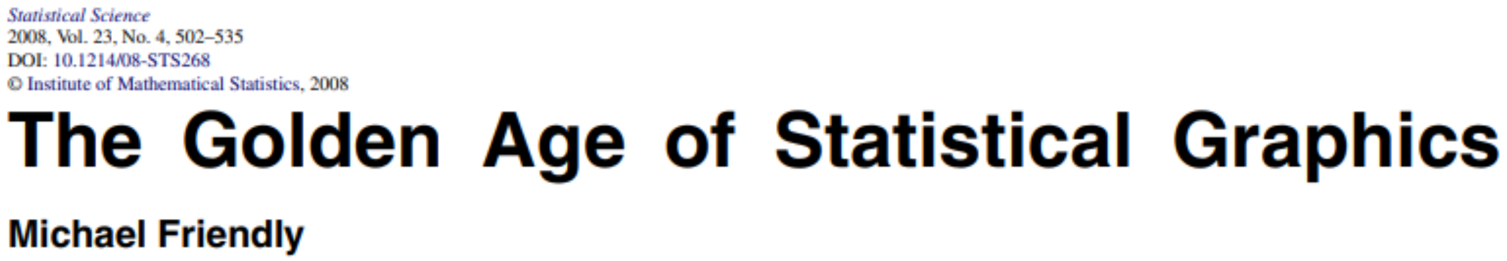
- x_df.join(y_df, how = "left")   # assuming x_df and y_df has the same Index values

Let's review this in Jupyter!

Questions?

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?

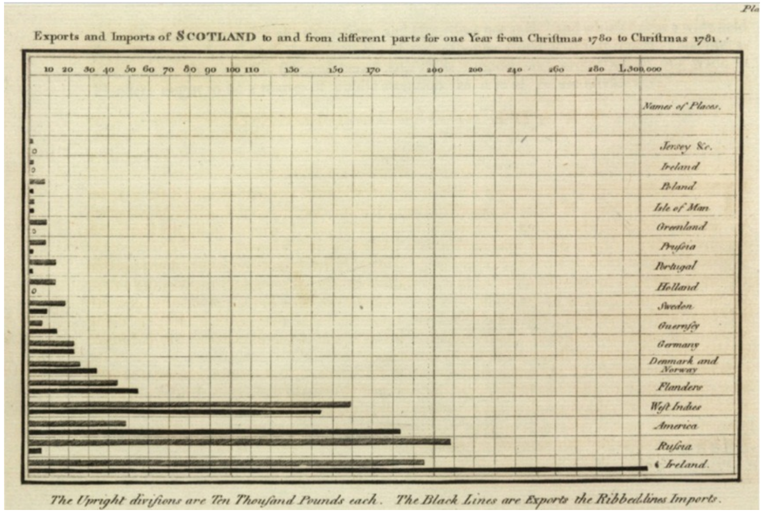
Whatever relates to extent and quantity may be represented by geometrical figures. Statistical projections which speak to the senses without fatiguing the mind, possess the advantage of fixing the attention on a great number of important facts.

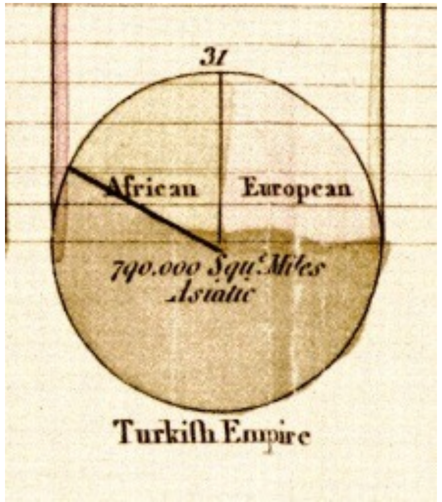
—Alexander von Humboldt, 1811

A very brief history of data visualization

The age of modern statistical graphs began around the beginning of the 19th century

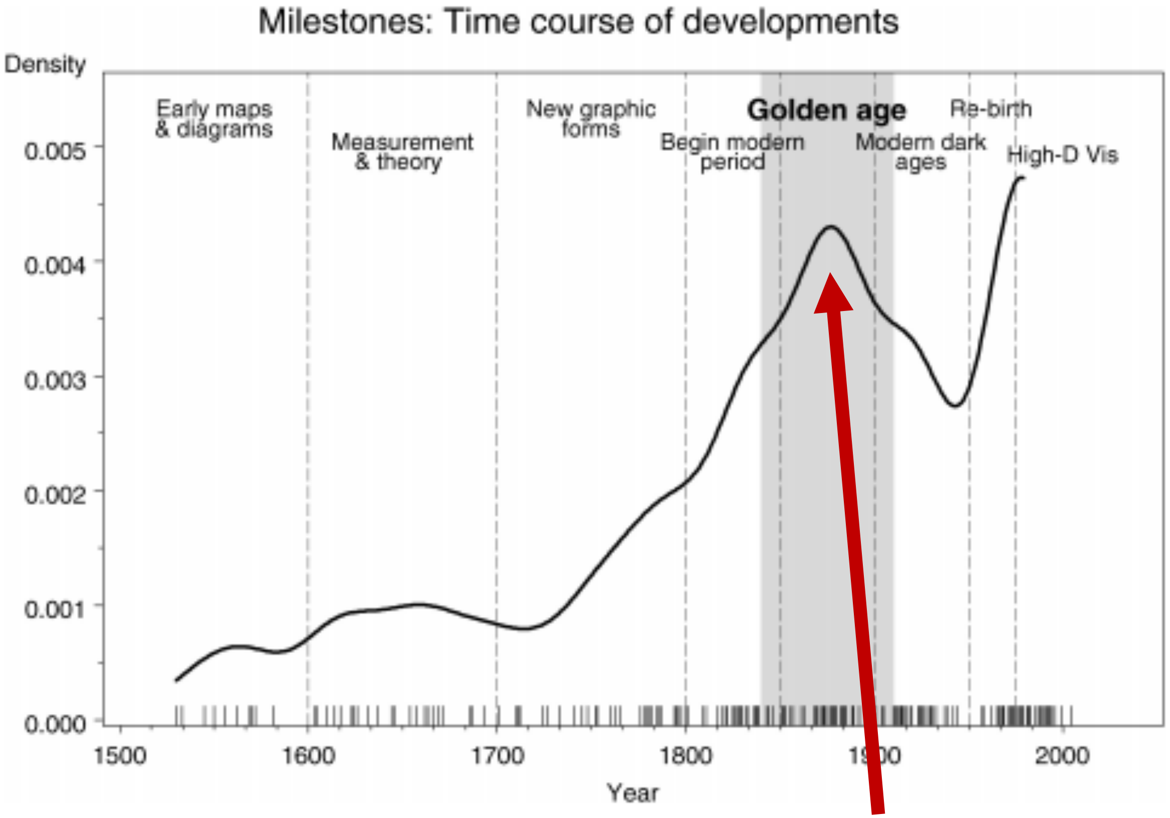
William Playfair (1759-1823) is credited with inventing the line graph, bar chart and pie chart





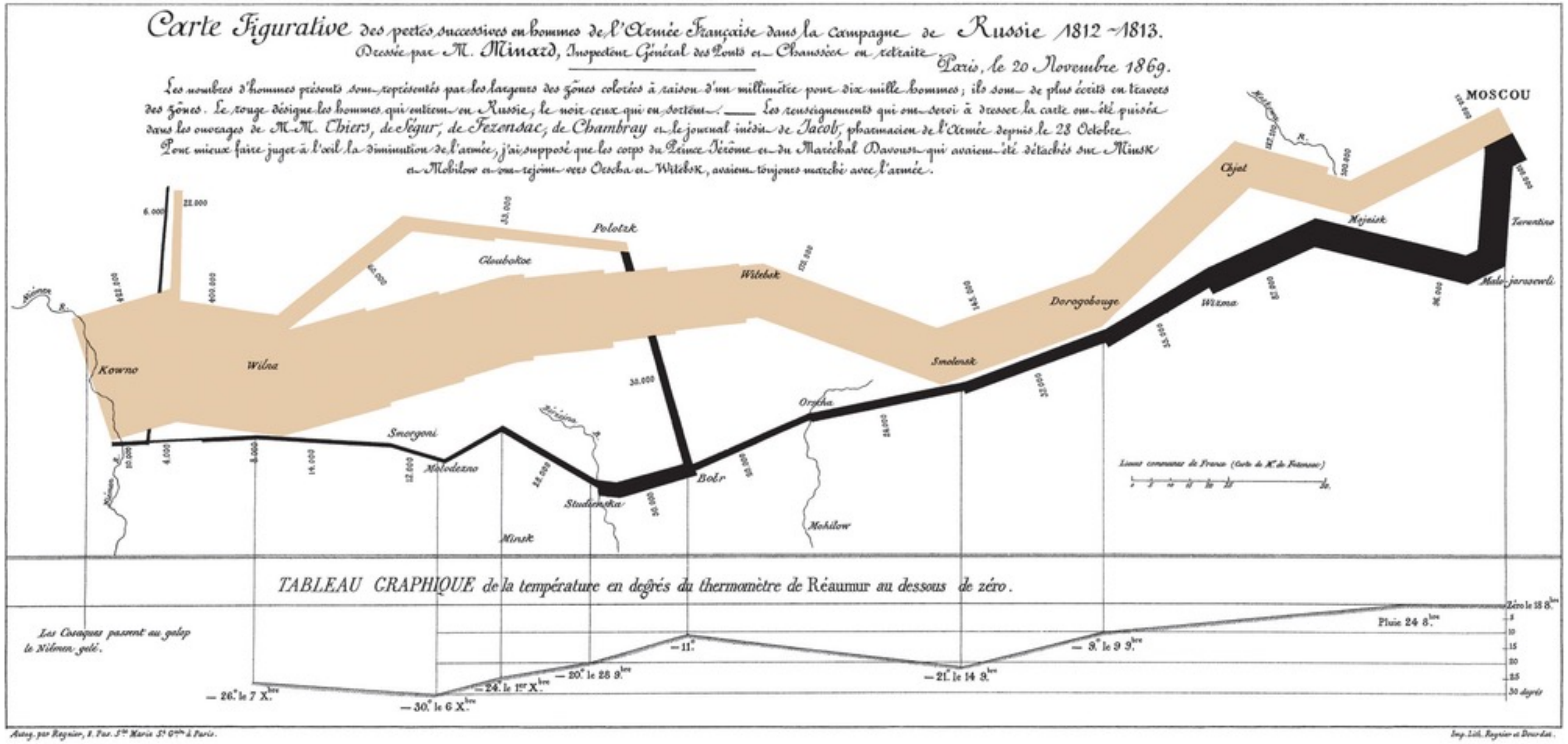
A very brief history of data visualization

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



A very brief history of data visualization

Joseph Minard (1781-1870)



Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812–1813.
Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.
Paris, le 20 Novembre 1869.

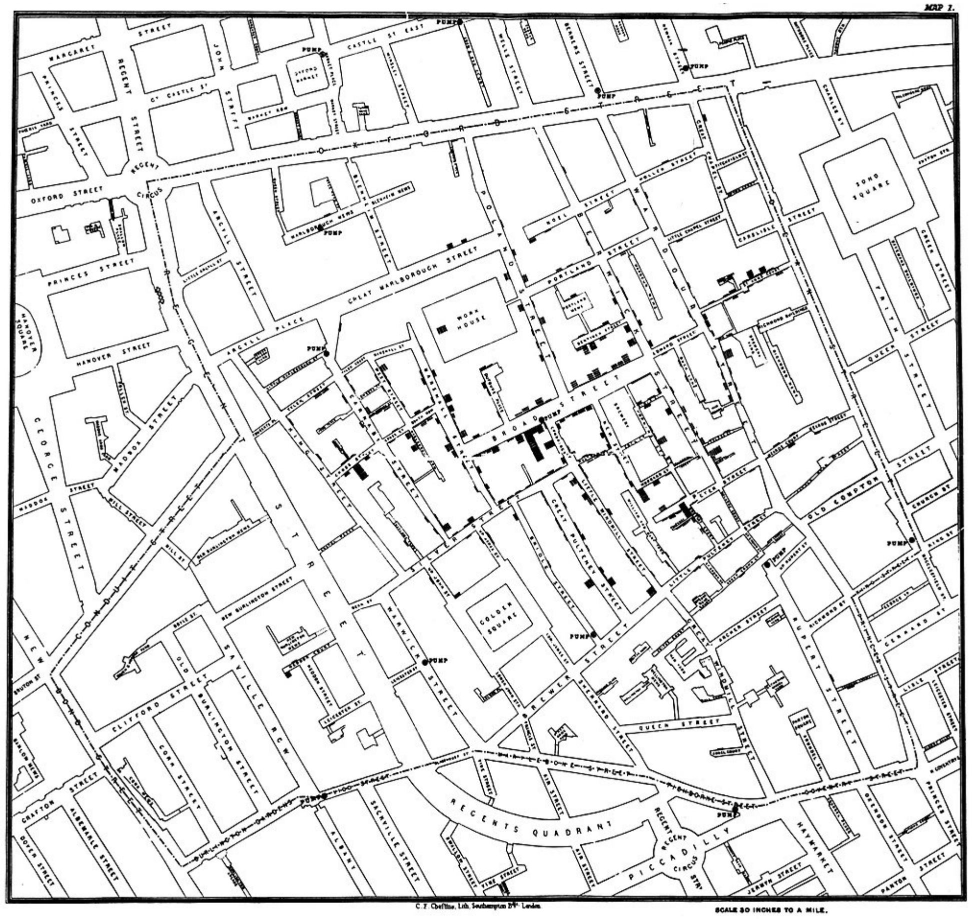
Les nombres d'hommes présents sont représentés par les largeurs des zones colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en travers des zones. Le rouge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. —— Les renseignements qui ont servi à dresser la carte ont été puisés dans les ouvrages de M.M. Thiers, de Ségur, de Fezensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre.

Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davoust qui avaient été détachés sur Minsk et Mobilow et ont rejoint vers Orscha et Witebsk, avaient toujours marché avec l'armée.

Map of Napoleon's march on Russia

A very brief history of data visualization

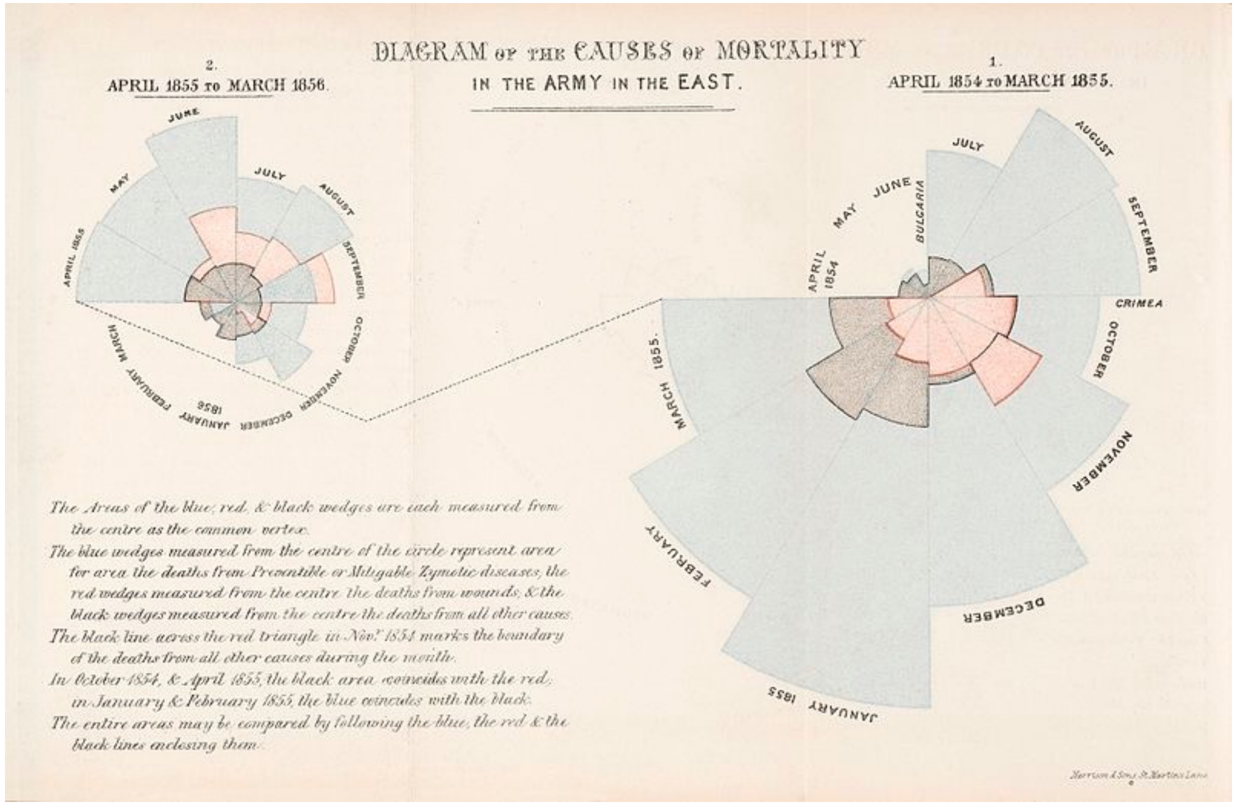
John Snow (1813-1858)



Clusters of cholera cases in London epidemic of 1854

A very brief history of data visualization

Florence Nightingale (1820-1910)



The Areas of the blue, red, & black wedges are each measured from the centre as the common vertex.

The blue wedges measured from the centre of the circle represent area for area the deaths from Preventible or Mitigable Zymotic diseases, the red wedges measured from the centre the deaths from wounds, & the black wedges measured from the centre the deaths from all other causes.

The black line across the red triangle in Nov.r 1854 marks the boundary of the deaths from all other causes during the month.

In October 1854, & April 1855, the black area coincides with the red; in January & February 1855, the blue coincides with the black.

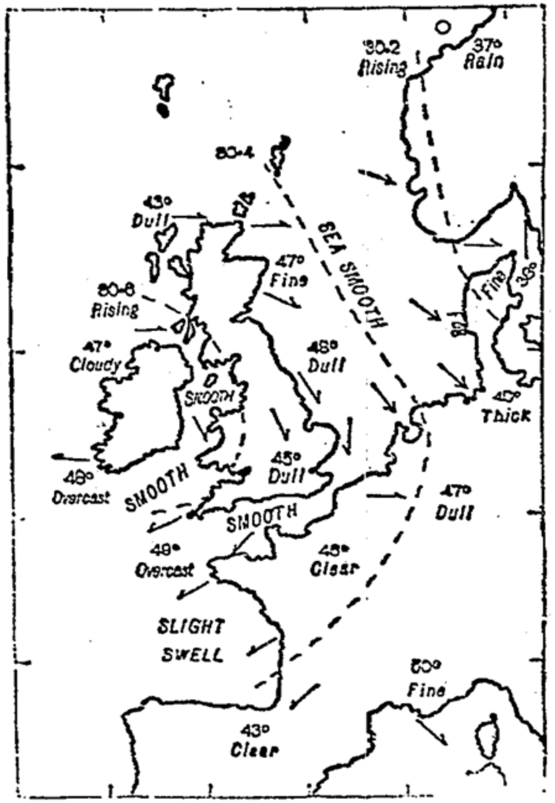
The entire areas may be compared by following the blue, the red & the black lines enclosing them.

Diagram of the causes of mortality in the army in the east

A very brief history of data visualization

Francis Galton (1822-1911)

Weather Chart, March 31, 1875.

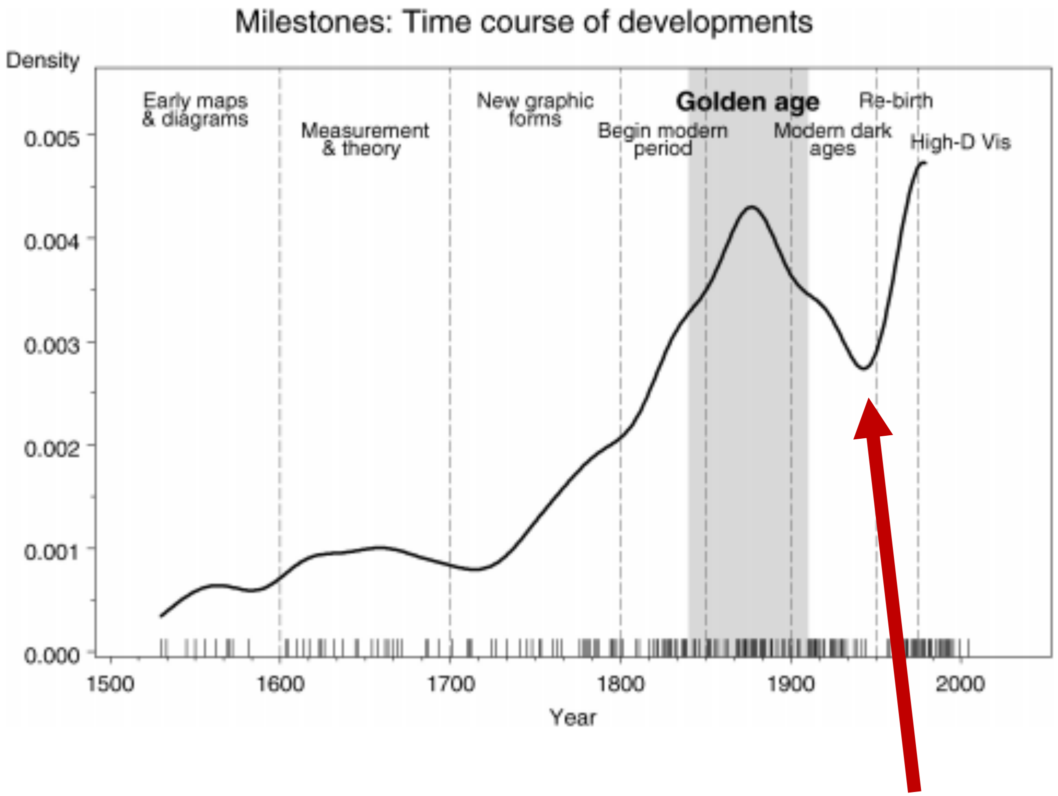


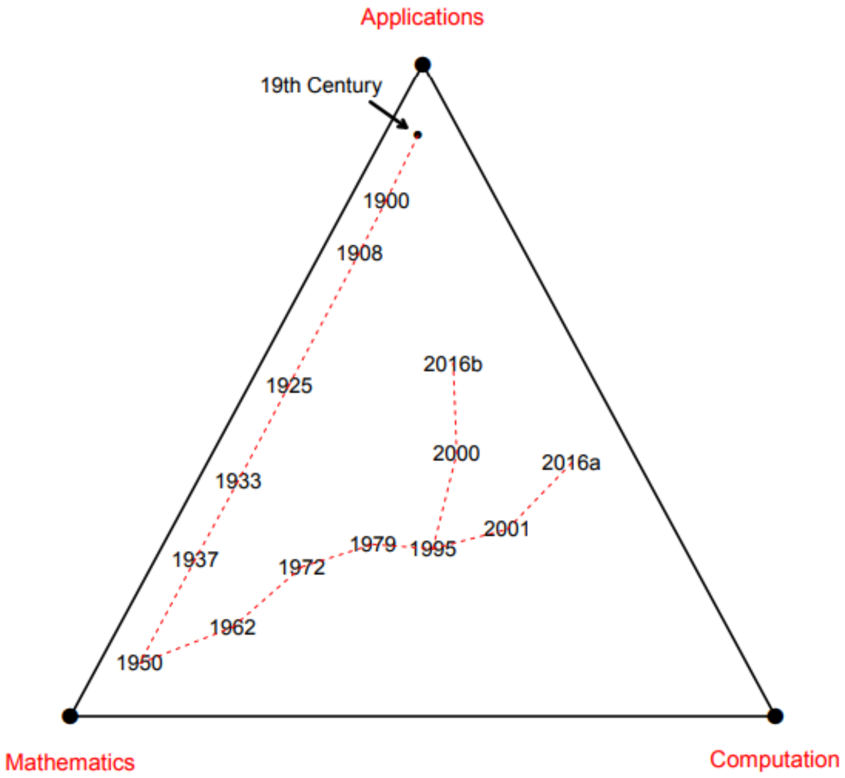
The dotted lines indicate the gradations of barometric pressure The variations of the temperature are marked by figures, the state of the sea and sky by descriptive words, and the direction of the wind by arrows—barbed and feathered according to its force. ⊙ denotes calm.

First weather map published in a newspaper (1875)

A very brief history of data visualization

“Graphical dark ages” around 1950

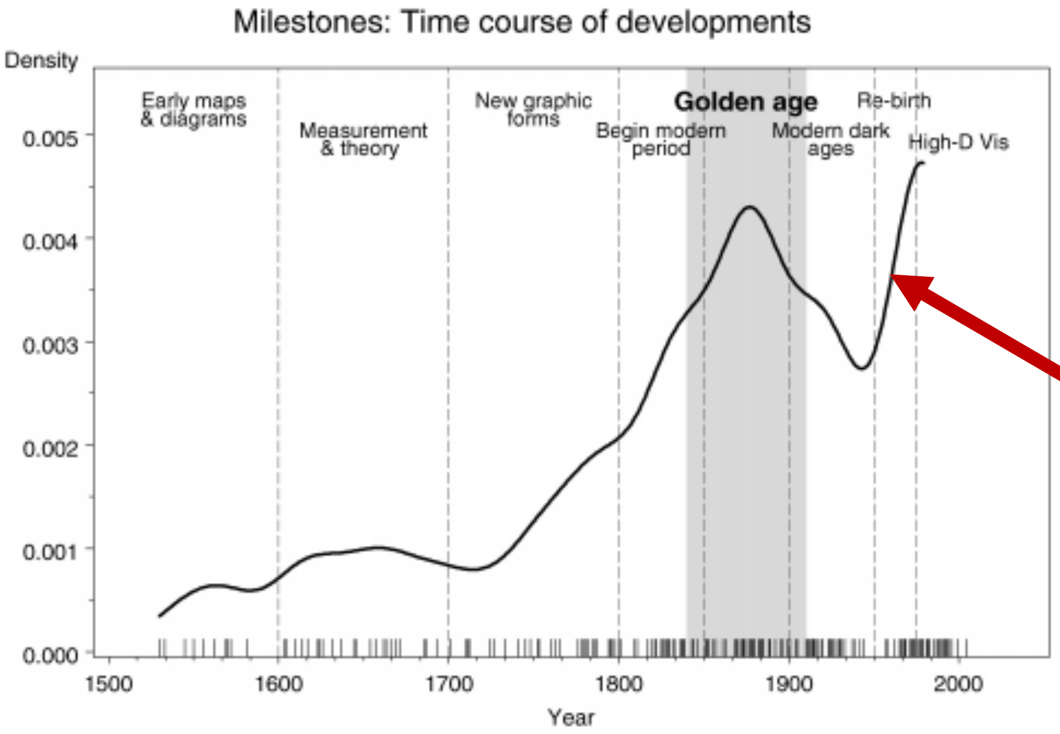




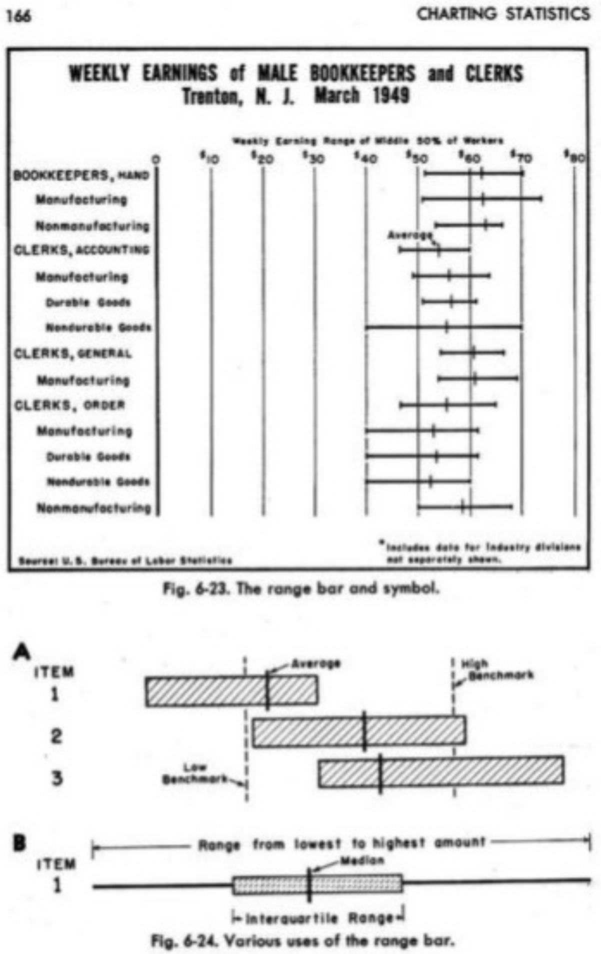
Computer Age Statistical Inference, Efron and Hastie

A very brief history of data visualization

Currently undergoing a “Graphical re-birth”



Box plot


166 CHARTING STATISTICS

WEEKLY EARNINGS of MALE BOOKKEEPERS and CLERKS
Trenton, N. J. March 1949

Source: U. S. Bureau of Labor Statistics

*Includes data for industry divisions not separately shown.

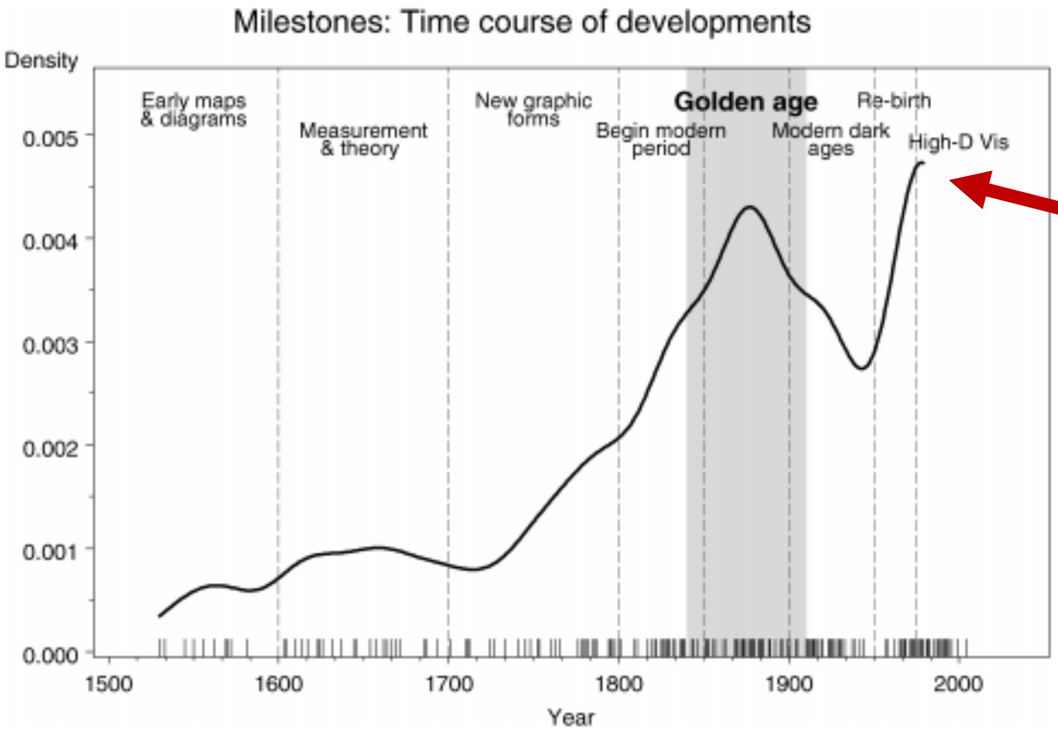
Fig. 6-23. The range bar and symbol.

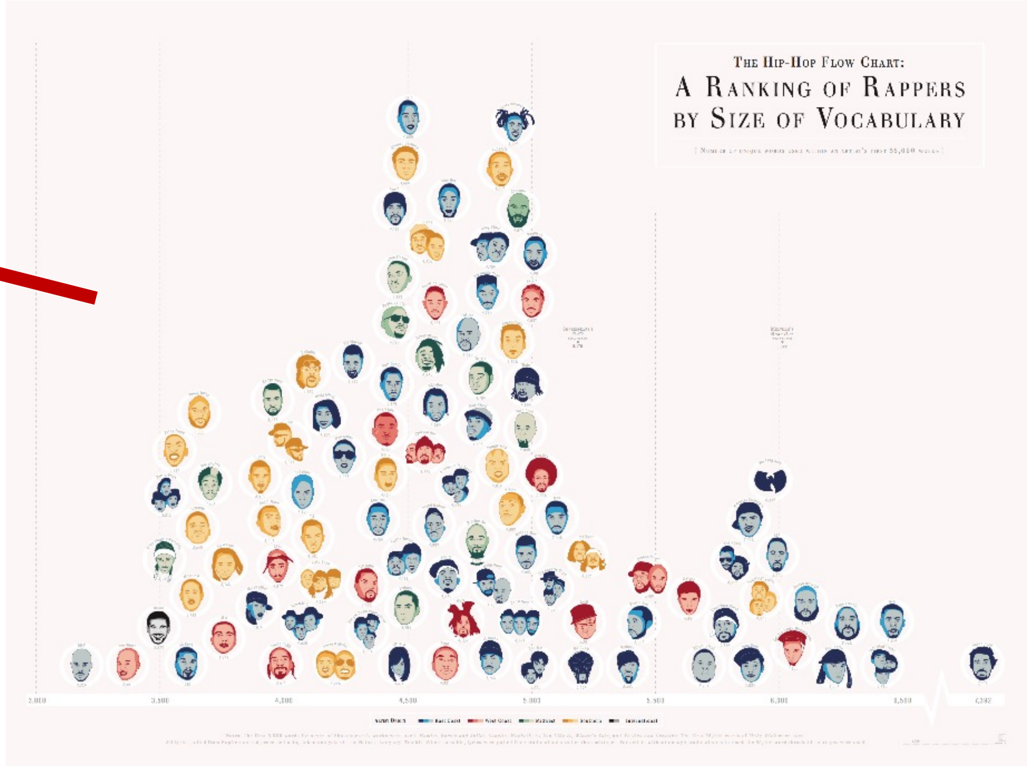
Fig. 6-24. Various uses of the range bar.

Spear 1952, Tukey 1970

A very brief history of data visualization

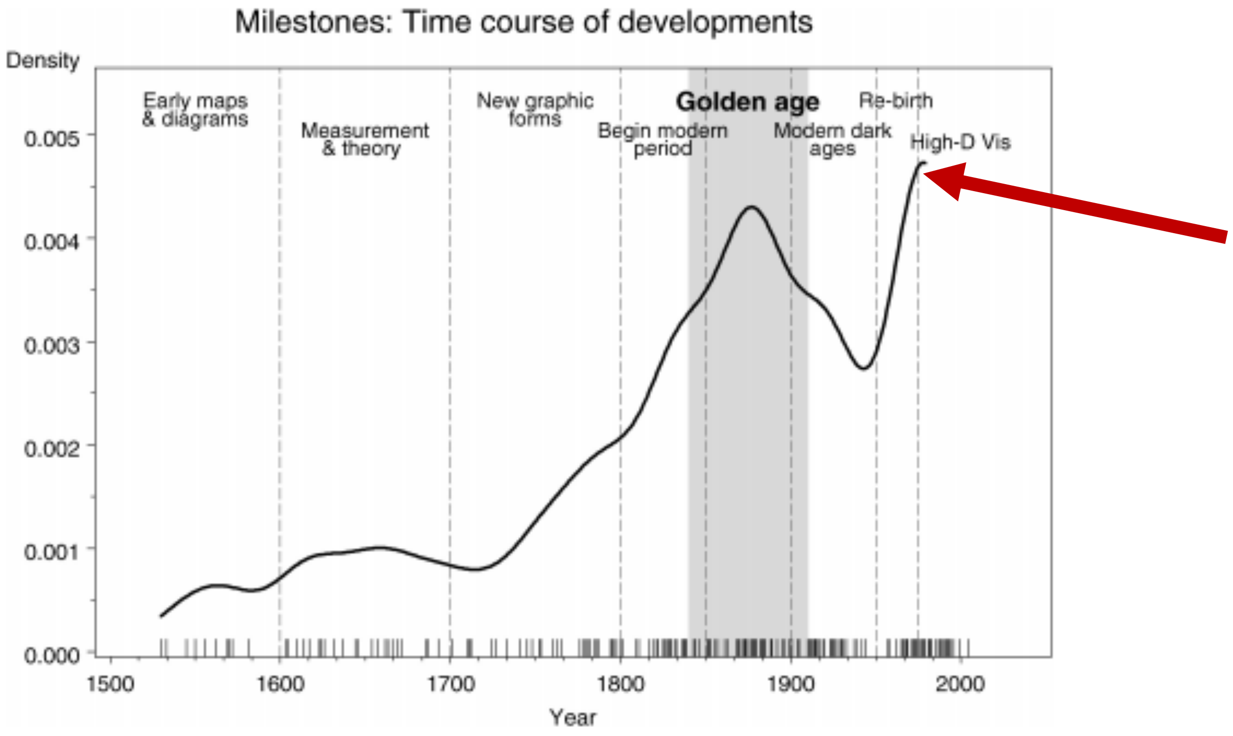
Currently undergoing a “Graphical re-birth”





A very brief history of data visualization

Currently undergoing a “Graphical re-birth”

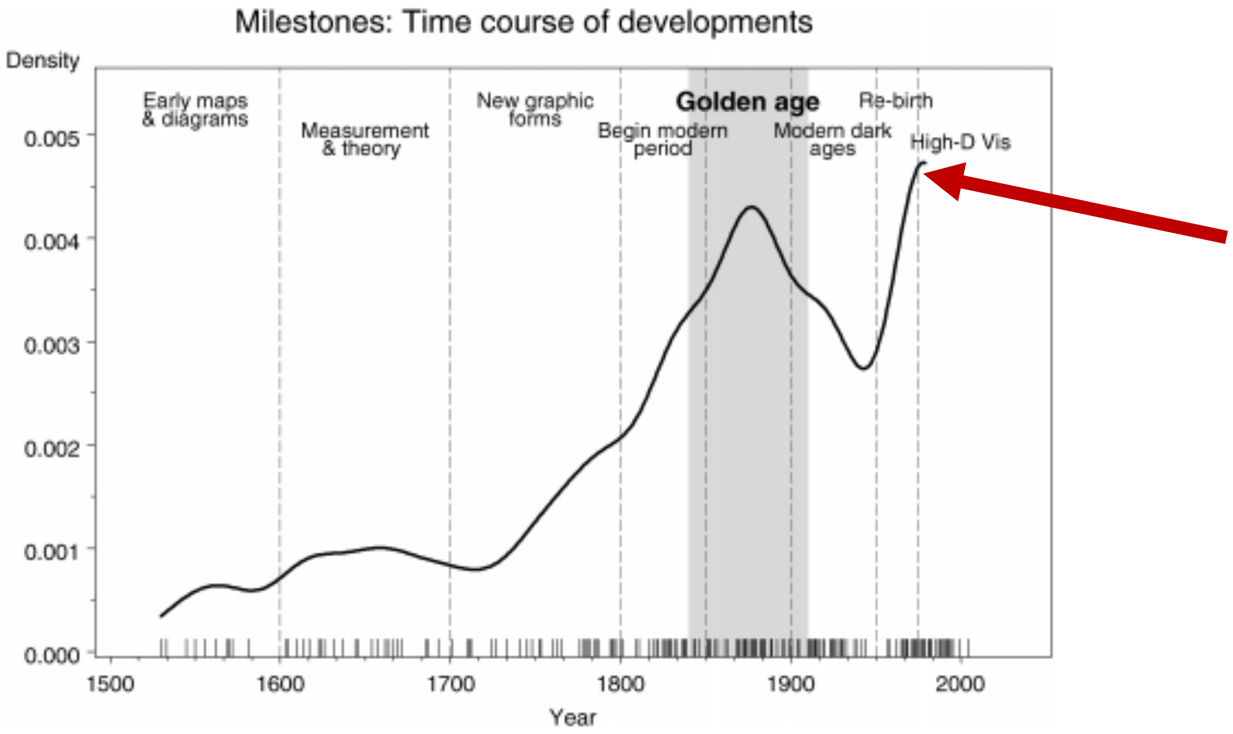


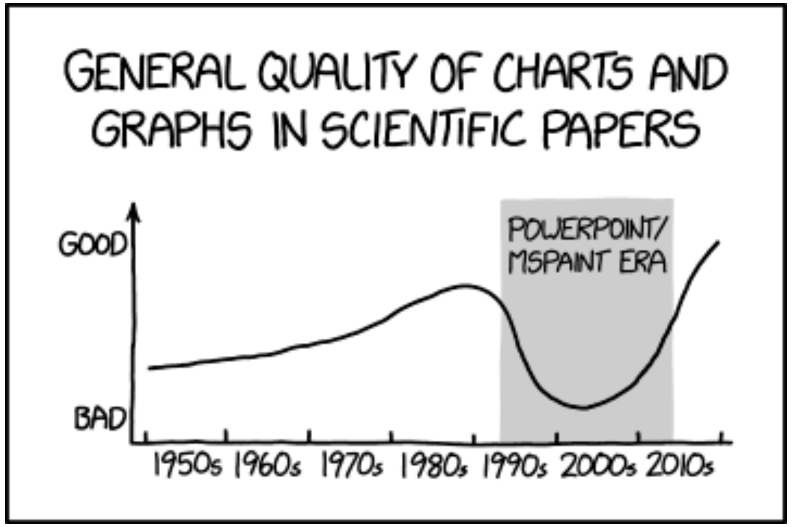
Hans Rosling’s gapminder

- Simple version
- TV special effects
- Ted Talk

A very brief history of data visualization

Currently undergoing a “Graphical re-birth”



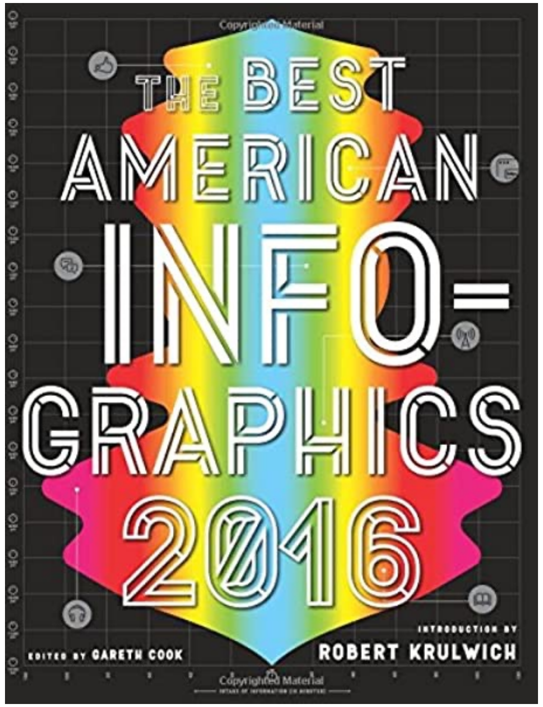


Coming up on homework 5: find an interesting data visualization…

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



Visualizing data with matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

- Matplotlib makes easy things easy and hard things possible.

Note: there are two different "interfaces" to matplotlib, that use slightly different syntax

- An explicit "Axes" interface
- An implicit "pyplot" interface
  - Just be aware if you are reading Stack overflow articles

We will use the pyplot interface (for now)

```
import matplotlib.pyplot as plt
```

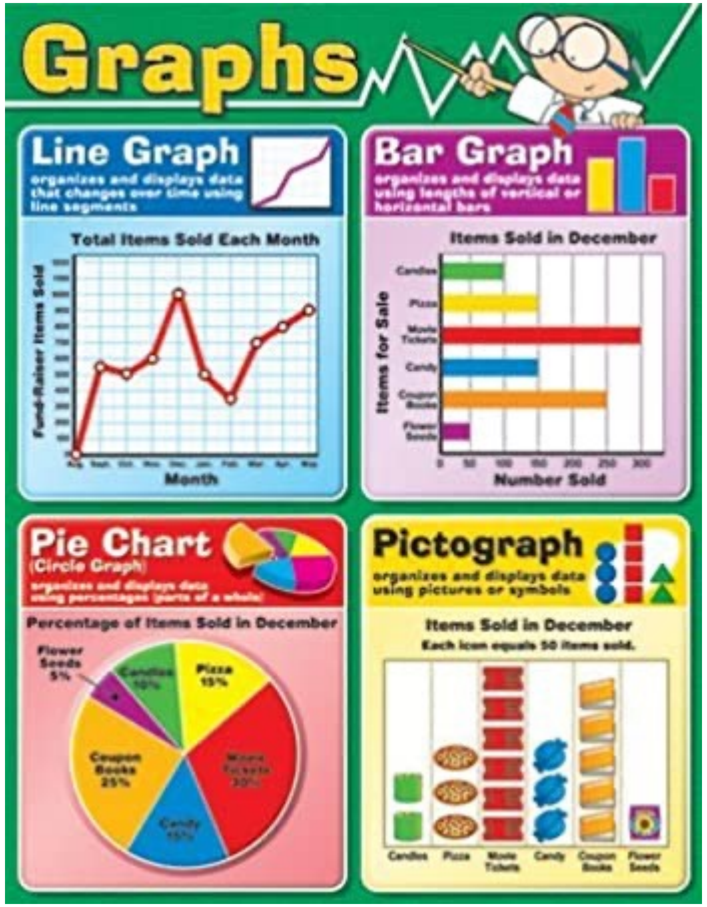
Note: we will discuss seaborn soon…

Types of plots

There are many different ways to plot data, many of which I am sure you have seen before

The type of plot you choose will depend on the type of data you have and what you want to emphasize

- i.e., There are different types of plots of categorical data, a single quantitative variable, two quantitative variables, etc.
  - This will become even more apparent when we look at the seaborn package

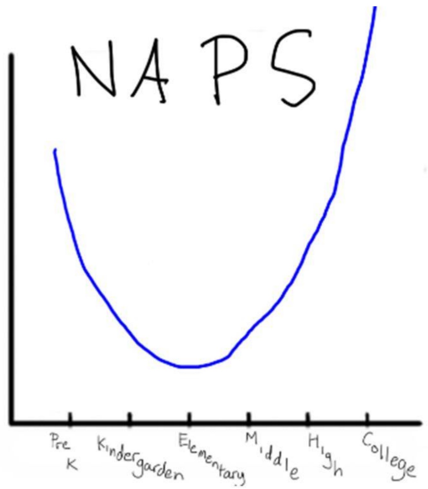


Line graph

Line graph (line chart, or curve chart) displays information as a series of data points called "markers" connected by straight line segments.

We can create line graphs in matplotlib using:

- plt.plot(y)   # create a line graph with successive integers as the x-values
- plt.plot(x, y)   # plots y as a function of x
- plt.plot(x, y, '-o') # creates lines with circle markers

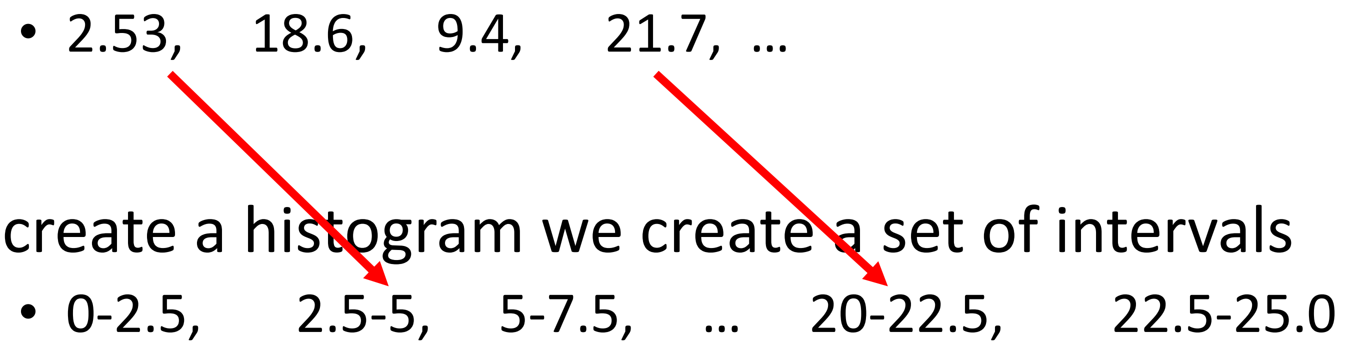


```
plt.ylabel(“y label”)

plt.xlabel(“x label”)

plt.title(“my title”)


plt.plot(x, y, label = “blah”)

plt.legend()
```

Let's explore this in Jupyter!

Visualizing quantitative data: histograms

Salaries of basketball players (in millions of dollars):



- 2.53, 18.6, 9.4, 21.7, …

To create a histogram we create a set of intervals

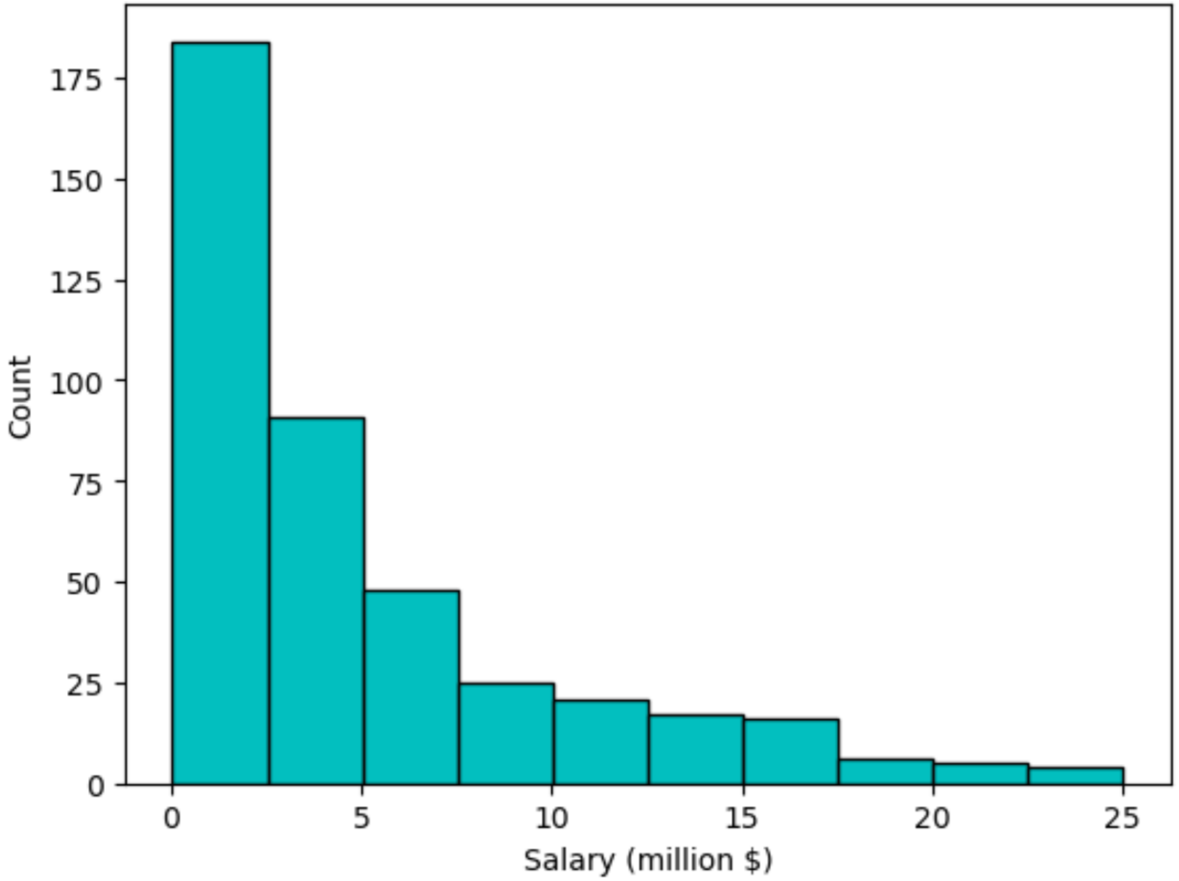
- 0-2.5, 2.5-5, 5-7.5, … 20-22.5, 22.5-25.0

We count the number of points that fall in each interval

We create a bar chart where the height of the bars is the counts in each bin

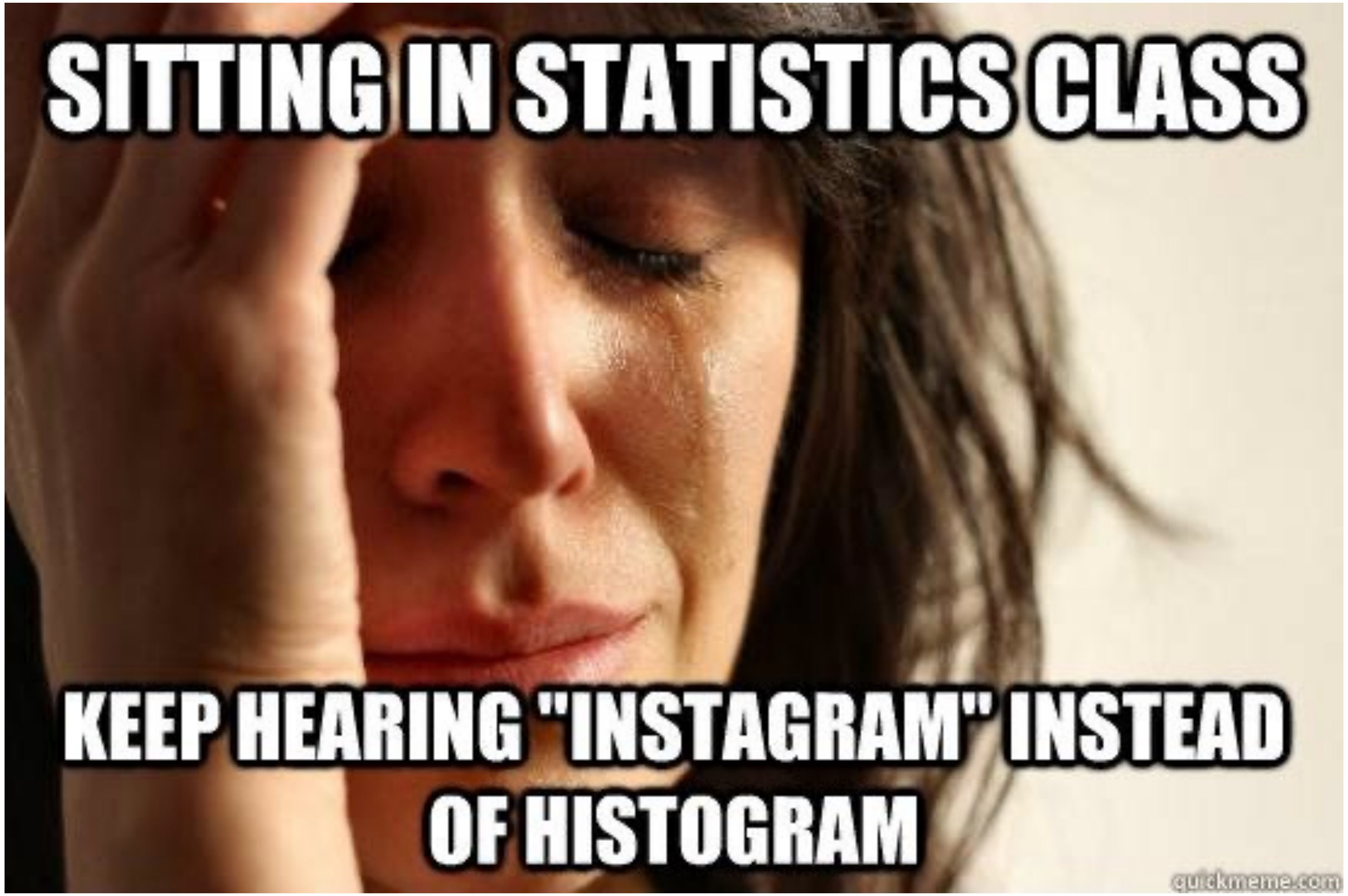
Histograms – countries life expectancy in 2007

Life Expectancy	Frequency Count
(0 – 2.5]	184
(2.5 – 5]	91
(5 – 7.5]	48
(7.5 – 10]	25
(10 – 12.5]	21
(12.5 – 15]	17
(15 – 17.5]	16
(17.5 – 20]	6
(20 – 22.5]	5
(22.5 – 25]	4



```
plt.hist(data)
```

Let's explore this in Jupyter!


SITTING IN STATISTICS CLASS
KEEP HEARING "INSTAGRAM" INSTEAD OF HISTOGRAM
quickmeme.com

Five number summary

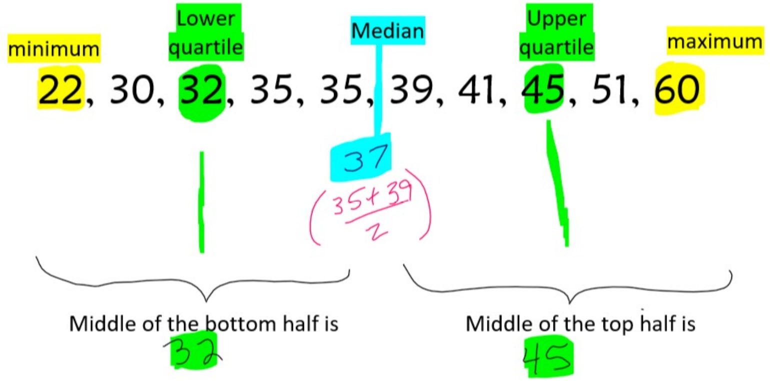
A **"Five Number Summary"** of quantitative data is defined as:

(minimum, Q_1, median, Q_3, maximum)

- Q_1 is the 25th percentile - i.e., the value such that 25% of the data is less than this value
- Q_3 is the 75th percentile - i.e., the value such that 75% of the data is less than this value

The Five number summary roughly splits the data into 4 equal parts

The **Interquartile range (IQR)** is defined as $Q_3 - Q_1$



minimum

Lower quartile

Median

Upper quartile

maximum

22, 30, 32, 35, 35, 39, 41, 45, 51, 60

37

$\left(\frac{35+39}{2}\right)$

Middle of the bottom half is 32

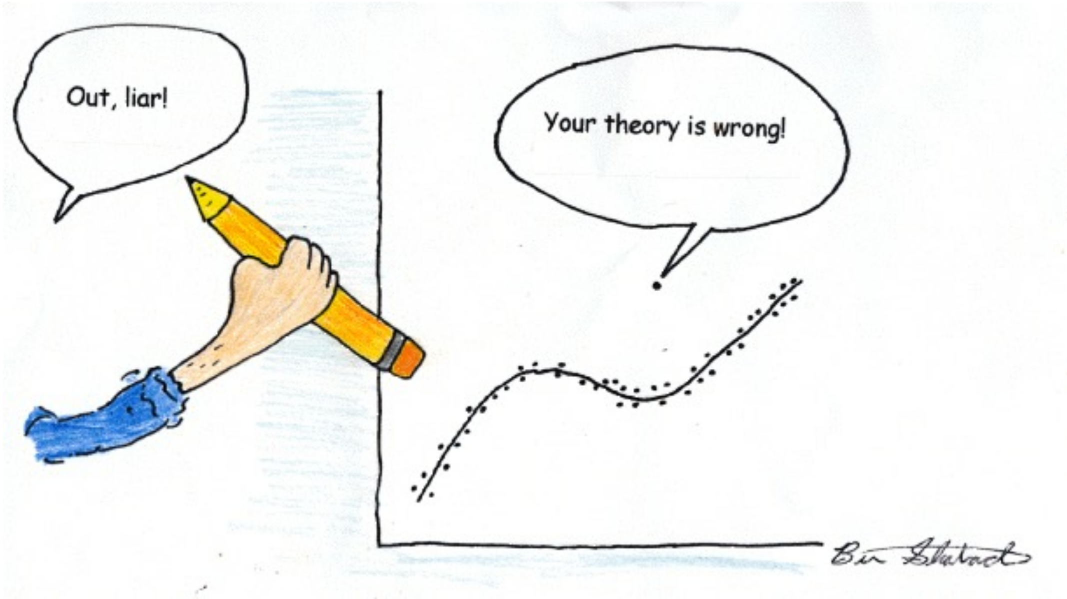
Middle of the top half is 45

Detecting of outliers

As a rule of thumb, we call a data value an **outlier** if it is:

Smaller than: Q_1 - 1.5 * IQR

Larger than: Q_3 + 1.5 * IQR



Boxplots

A **boxplot** is a graphical display of the 5 number summary and consists of:

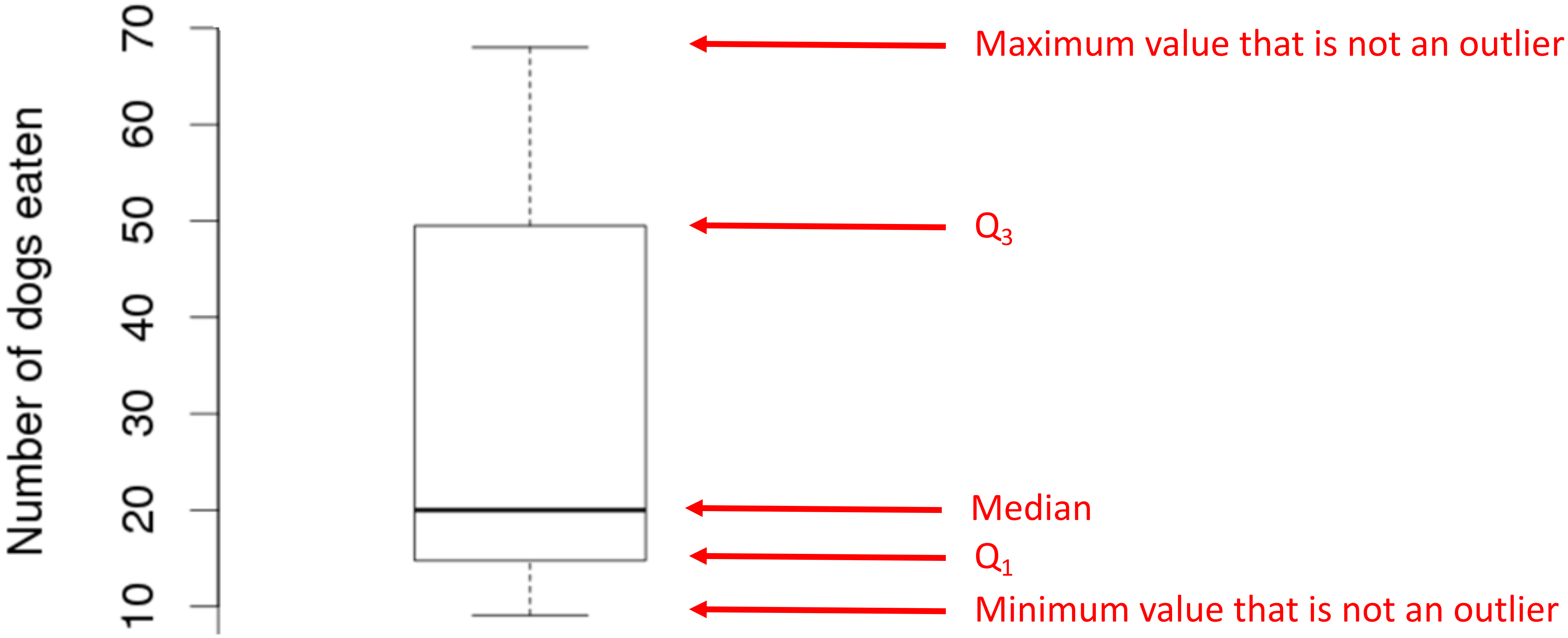
1. Drawing a box from Q_1 to Q_3

2. Dividing the box with a line (or dot) drawn at the median

3. Draw a line from each quartile to the most extreme data value that is not and outlier

4. Draw a dot/asterisk for each outlier data point.

Box plot of the number of hot dogs eaten by the men's contest winners 1980 to 2010



Maximum value that is not an outlier

Q_3

Median

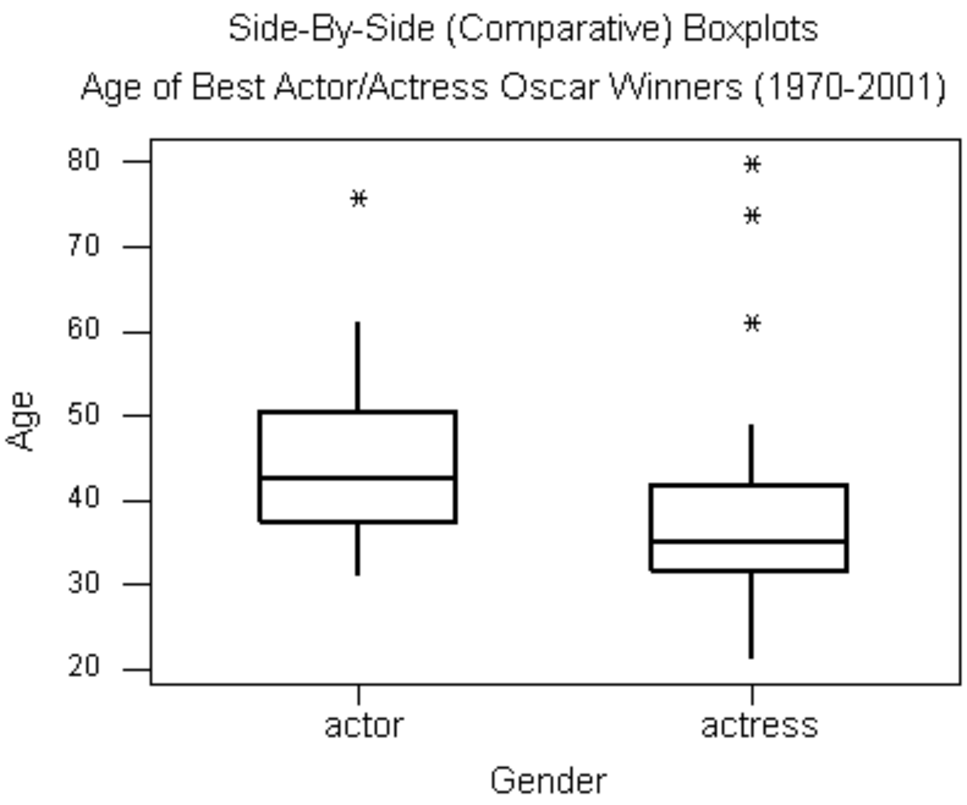
Q_1

Minimum value that is not an outlier

Comparing quantitative variables across categories

Often one wants to compare quantitative variables across categories

Side-by-Side graphs are a way to visually compare quantitative variables across different categories.



Let's explore this in Jupyter!

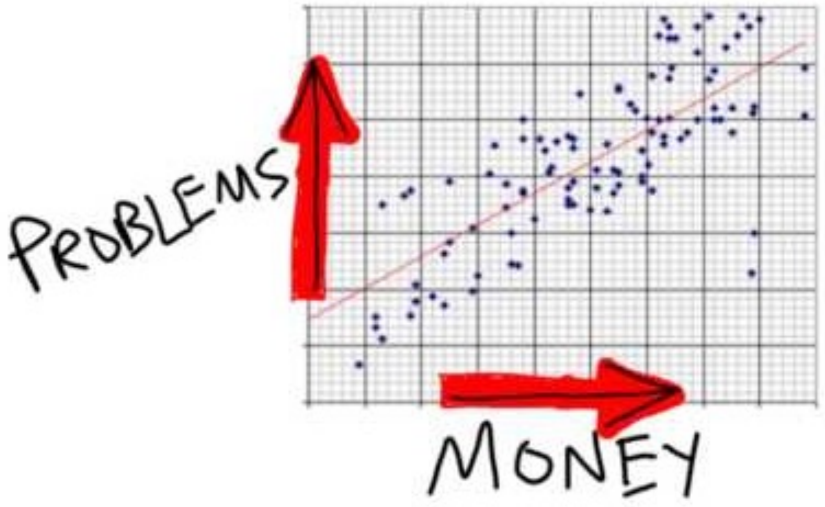
`plt.boxplot(data)`

Scatterplots

A **scatterplot** graphs the relationship between two variables

- Each axis represents the value of one variables
- Each point the plot shows the value for the two variables for a single data case

If there is an explanatory and response variable, then the explanatory variable is put on the x-axis and the response variable is put on the y-axis.



Scatterplots

There are two ways to create scatter plots in matplotlib:

1. Using plt.plot(x, y, '.')

2. Using plot.scatter(x, y)
   - This function 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

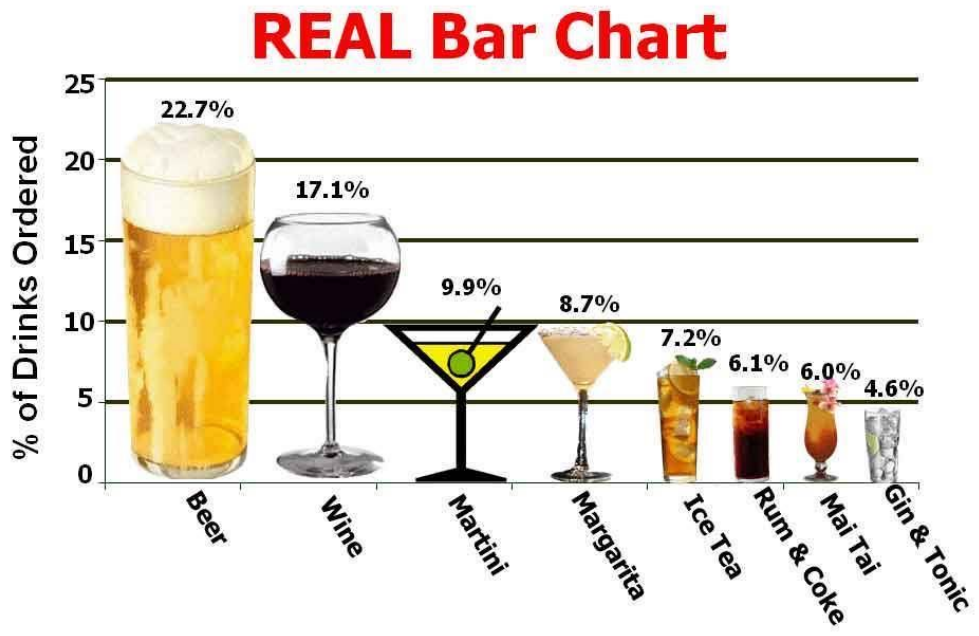
Let's explore this in Jupyter!

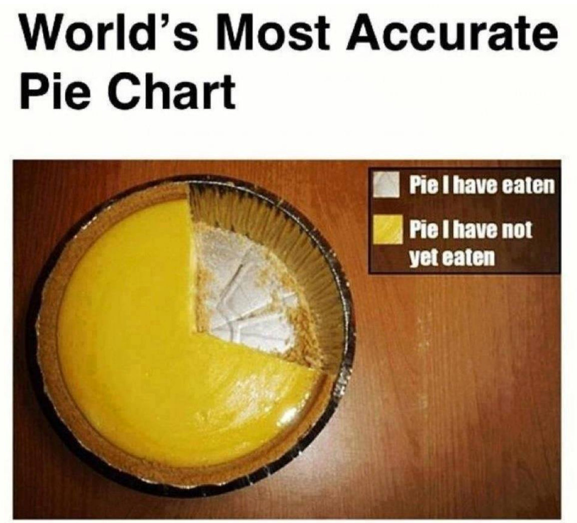
Bar plots and pie charts

Bar plots and pie charts are used to plot ***categorical data***

Bar plots, the heights of the bars indicate the number of items in each category

In pie charts, the angle of each segment indicates the proportion of items in each category





Let's explore this in Jupyter!

Subplots

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

1 row

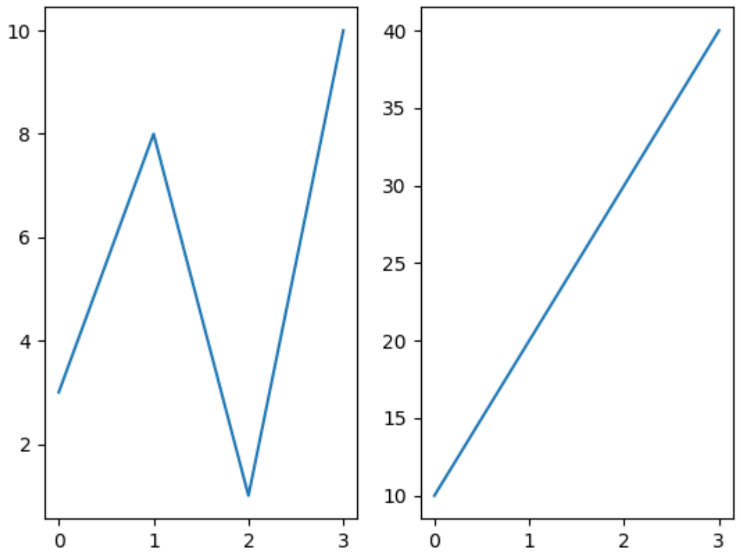
2 columns

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

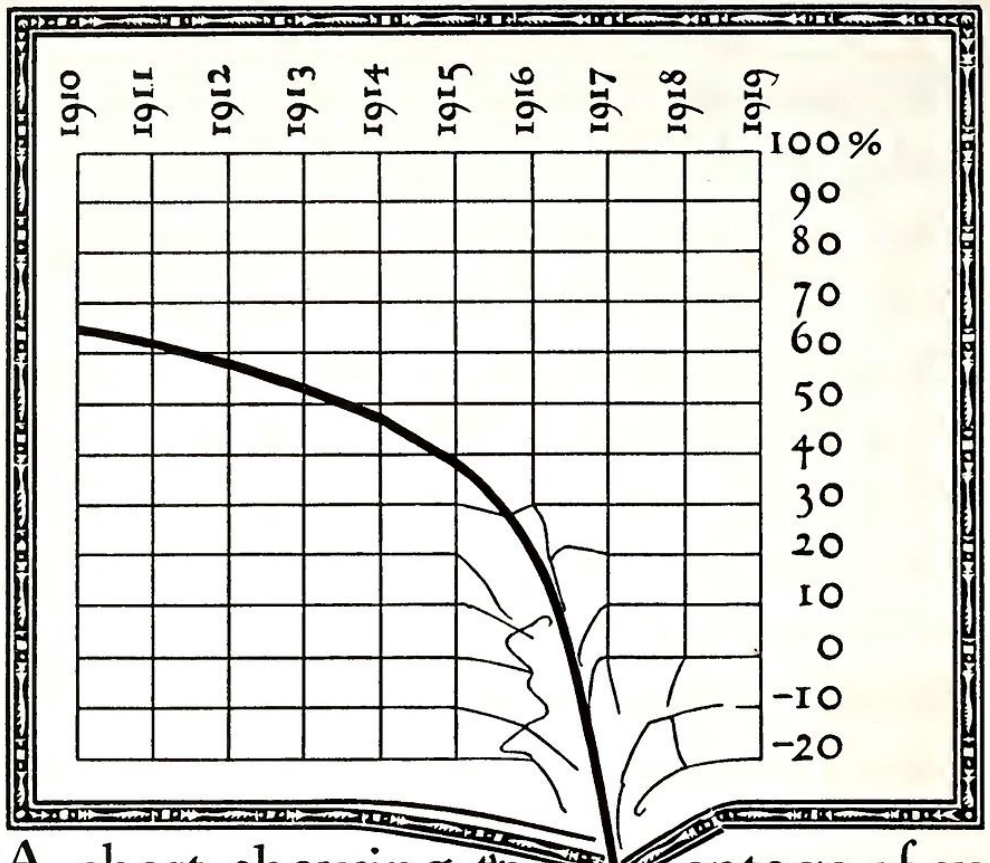
plot on the first subplot

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

plot on the second subplot



Let's explore this in Jupyter!



A chart showing the percentage of excellence in the physical properties of books published since 1910.