YData: Introduction to Data Science

Class 03: Python basics continued and descriptive statistics

Overview

Review of an intro to Python

- Expressions, Names, Call expressions, Data types

More intro to Python

- Comparisons
- Lists

Basic statistics and data visualizations

- Categorical data: Proportions, bar plots, pie charts
- Quantitative data: mean and median, histograms
- If there is time: plots for 2 quantitative variables

Announcement: Homework 1

Homework 1 has been posted!

```
import YData
YData.download.download_homework(1)
```

It is due on Gradescope on Sunday January 28th at 11pm

- **Be sure to mark each question on Gradescope!**

Notes:

- There is an ~18 page reading from the book "Data and the American Dream" that you need to do, so I recommend you get started on this soon.
- The last problem consist of answering a question that come up with
  - At the moment, you will be a little limited in answering questions given we have only started to learn Python, but see if you can come up with anything interesting and challenge yourself appropriately

Announcement: learning groups!

Learning groups are informal groups of 3-4 students where you can get together (independent of TAs) to work on the homework and other class projects.

If you are interested in being part of a learning group please sign up!

- A link to sign up will be put on Canvas and has been sent out as an announcement.
- Please sign up by 11:59pm on Wednesday 1/24

Review: Basics of Python

Last class we discussed the basics of Python including…

Expressions:

- 2 + 3
- 2**3

Names and assignment:

- class_number = 123

Numerical representations:

- my_int = 2
- my_float = 3.14159

Review: Basics of Python

We also discussed…

<u>Types of objects</u>

- my_string = “hello world!” # this is a comment about a string
- my_Boolean = True
- type(my_Boolean)
- int('2') # type conversion from a string to an int

<u>Call expressions:</u>

- abs(-5)

Let’s quickly practice this in Jupyter!

Comparisons

We can use mathematical operators to compare numbers and strings

- Results return Boolean values True and False

Comparison	Operator	True example	False Example
Less than	<	2 < 3	2 < 2
Greater than	>	3 > 2	3 > 3
Less than or equal	<=	2 <= 2	3 <= 2
Greater or equal	>=	3 >= 3	2 >= 3
Equal	==	3 == 3	3 == 2
Not equal	!=	3 != 2	2 != 2

We can compare strings alphabetically

Let's explore this in Jupyter!

Lists

Lists

Lists are ways to store multiple items

We can create lists using square brackets []

- my_list = [2, 3, 4]

We can also access list items using square brackets []

- my_list[2]

Lists can contain elements of different types

- my_list2 = [5, 6, 'seven']

Let's explore this in Jupyter!

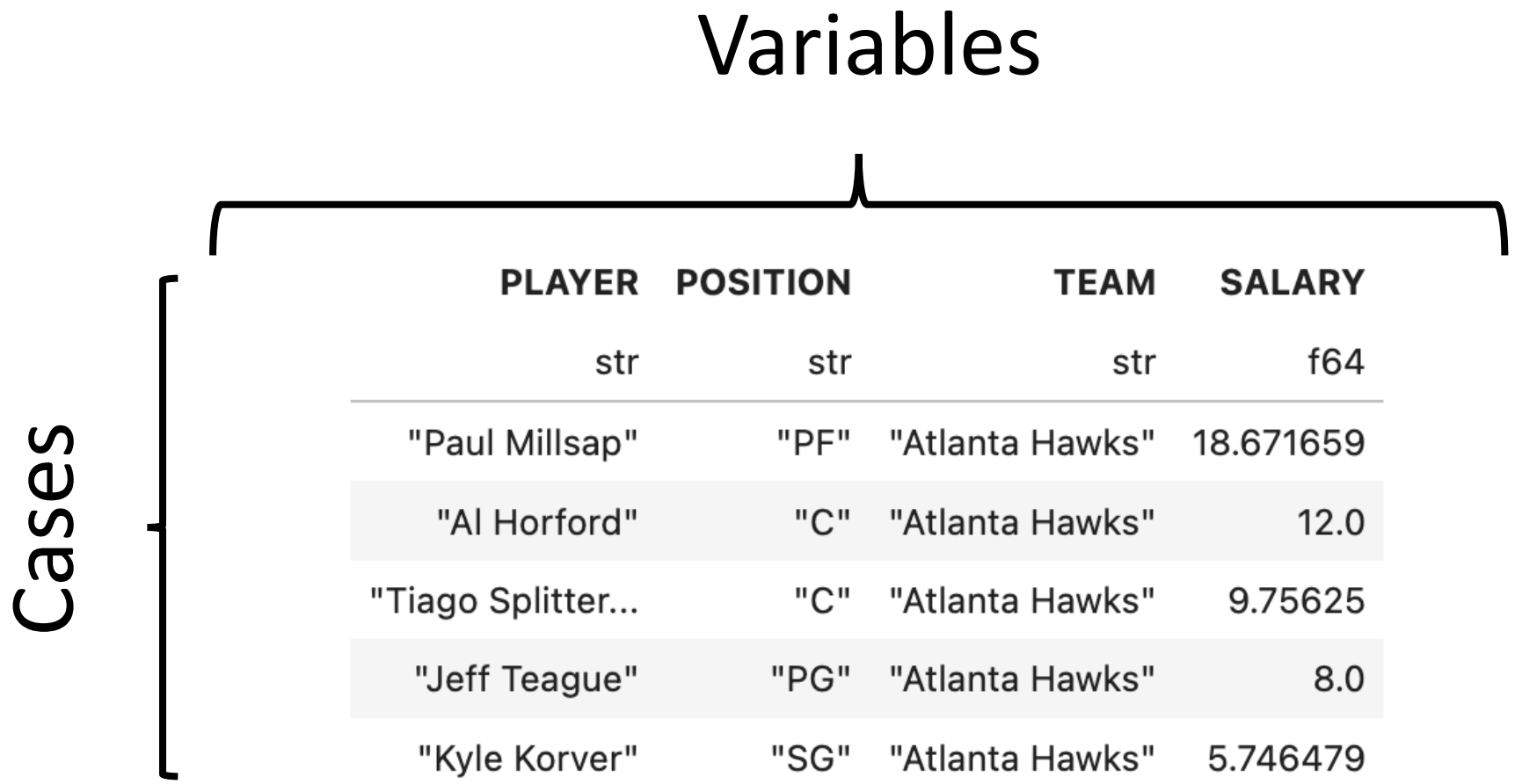
A brief introduction to statistics…

Example: NBA salaries

Let's explore salaries of NBA players (from the 2022-2023 season)

PLAYER	POSITION	TEAM	SALARY
Paul Millsap	PF	Atlanta Hawks	18.6717
Al Horford	C	Atlanta Hawks	12
Tiago Splitter	C	Atlanta Hawks	9.75625
Jeff Teague	PG	Atlanta Hawks	8
Kyle Korver	SG	Atlanta Hawks	5.74648
Thabo Sefolosha	SF	Atlanta Hawks	4
Mike Scott	PF	Atlanta Hawks	3.33333
Kent Bazemore	SF	Atlanta Hawks	2
Dennis Schroder	PG	Atlanta Hawks	1.7634
Tim Hardaway Jr.	SG	Atlanta Hawks	1.30452

Example Dataset – NBA player statistics



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

Categorical and Quantitative Variables

Cases

Categorical Variable

Quantitative Variable

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

Let's explore this in Jupyter!

Categorical data

statistics

A ***statistic*** is a number computed from a sample of data

Quantitative Variable

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

5.07

Proportions

For a *single* ***categorical variable***, the main ***statistic*** of interest is the *proportion* in each category

- E.g., the proportion of basketball players that are Centers (C)

$$\text{Proportion in a category} = \frac{\text{number in that category}}{\text{total number}}$$

Proportions

For a *single* ***categorical variable***, the main ***statistic*** of interest is the *proportion* in each category

- E.g., the proportion of basketball players that are Centers (C)

$$\text{Proportion centers} = \frac{\text{number of centers}}{\text{total number}}$$

Categorical Variable

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

Plotting data

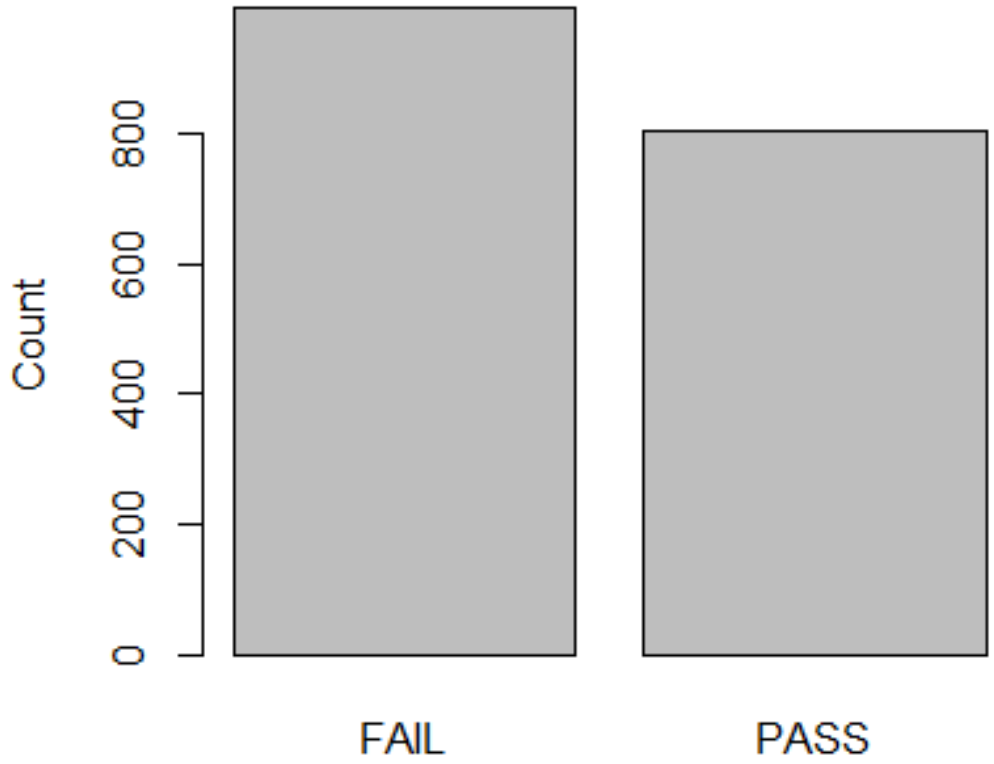
To create basic visualizations in Python we can use the matplotlib library

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

We can then create plots using functions such as:

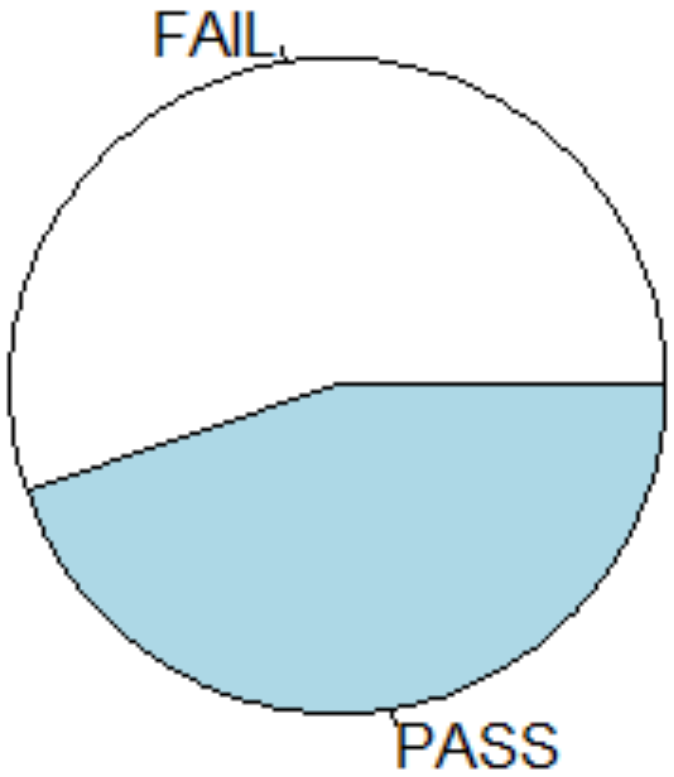
- plt.plot()
- plt.bar()
- plt.hist()
- etc.

Visualizing categorical data: The Bar Chart



Matplotlib: `plt.bar(labels, data)`

Visualizing categorical data: The Pie Chart



Matplotlib: `plt.pie(data)`

World's Most Accurate Pie Chart

Let's explore this in Jupyter!

Quantitative data

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

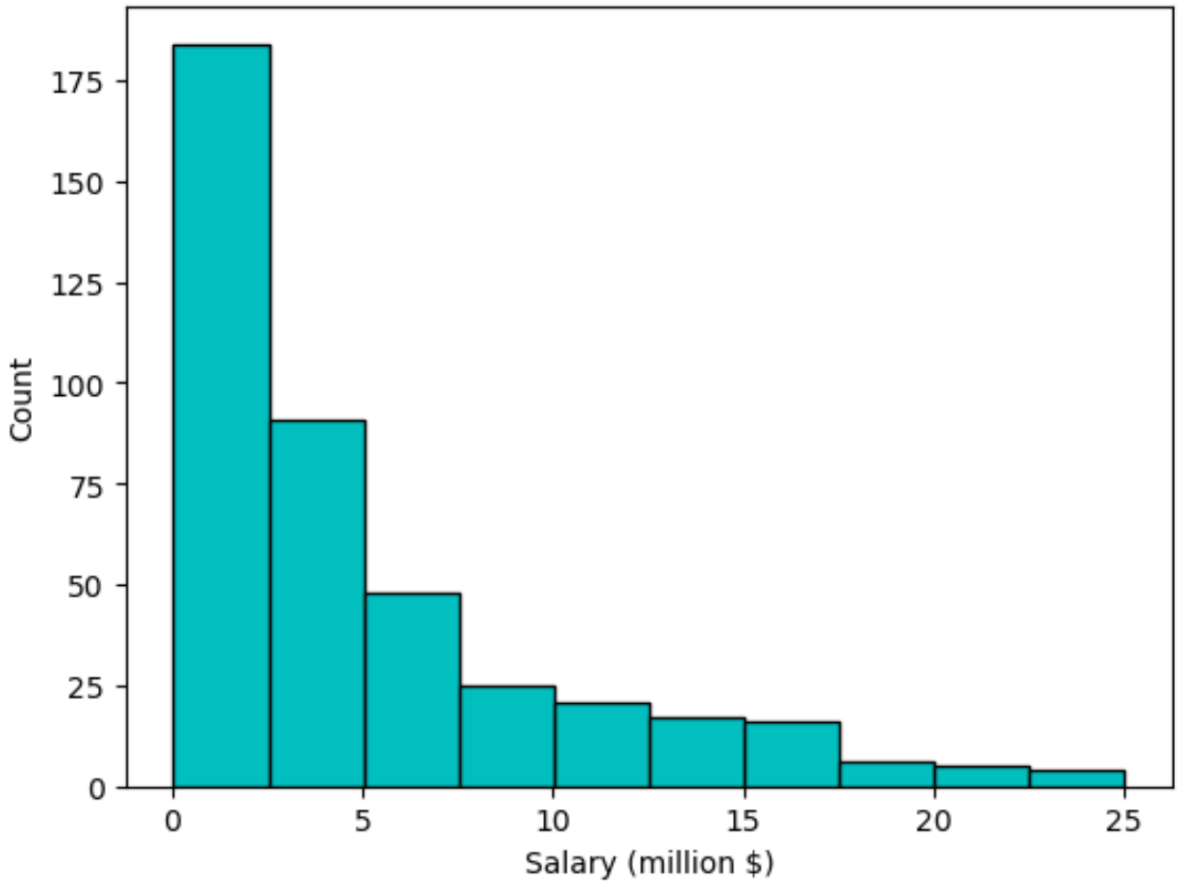
- 0-2.5, 2.5-5, 5-7.5, … 20-22.5, 22.5-25.0

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

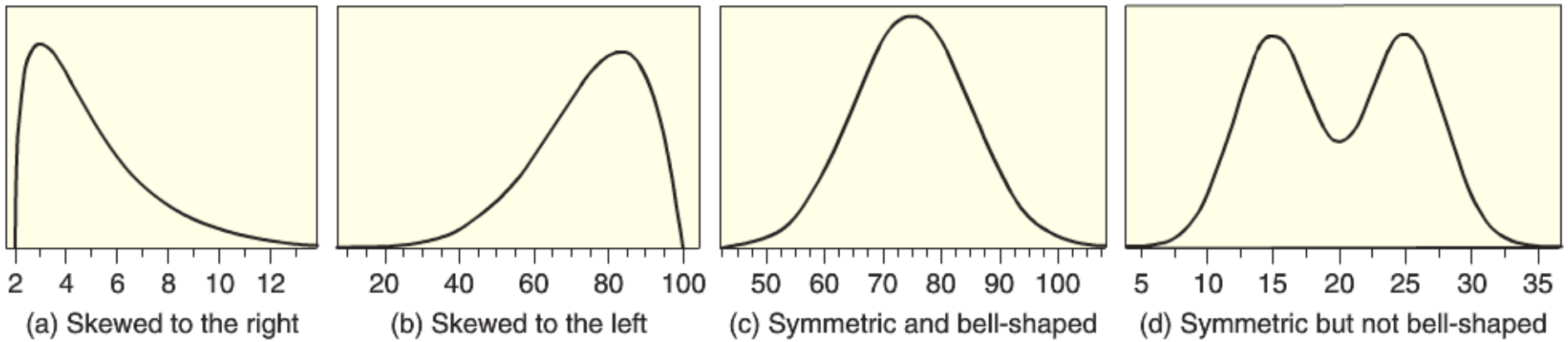
Histograms – countries life expectancy in 2007

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

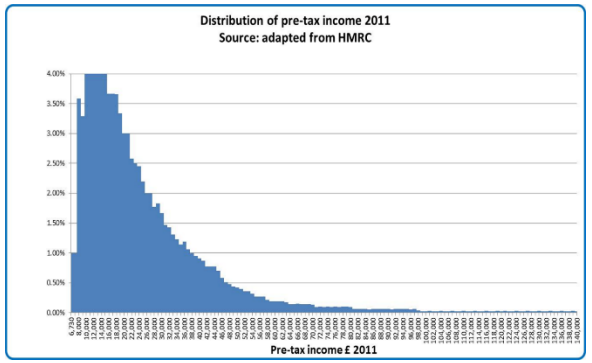


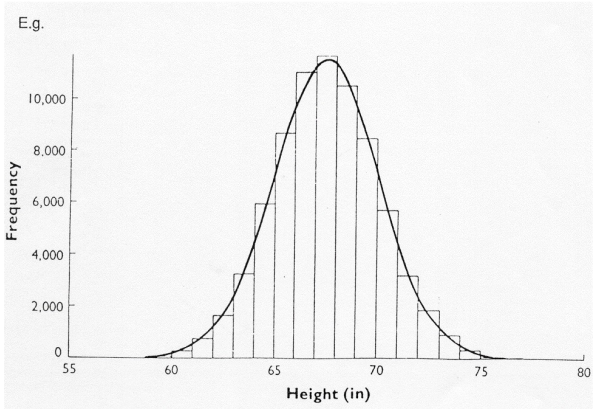
Matplotlib: `plt.hist(data)`

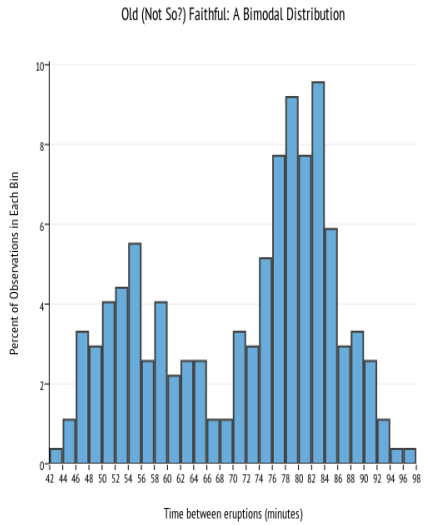
Common shapes of data distributions



Income distribution

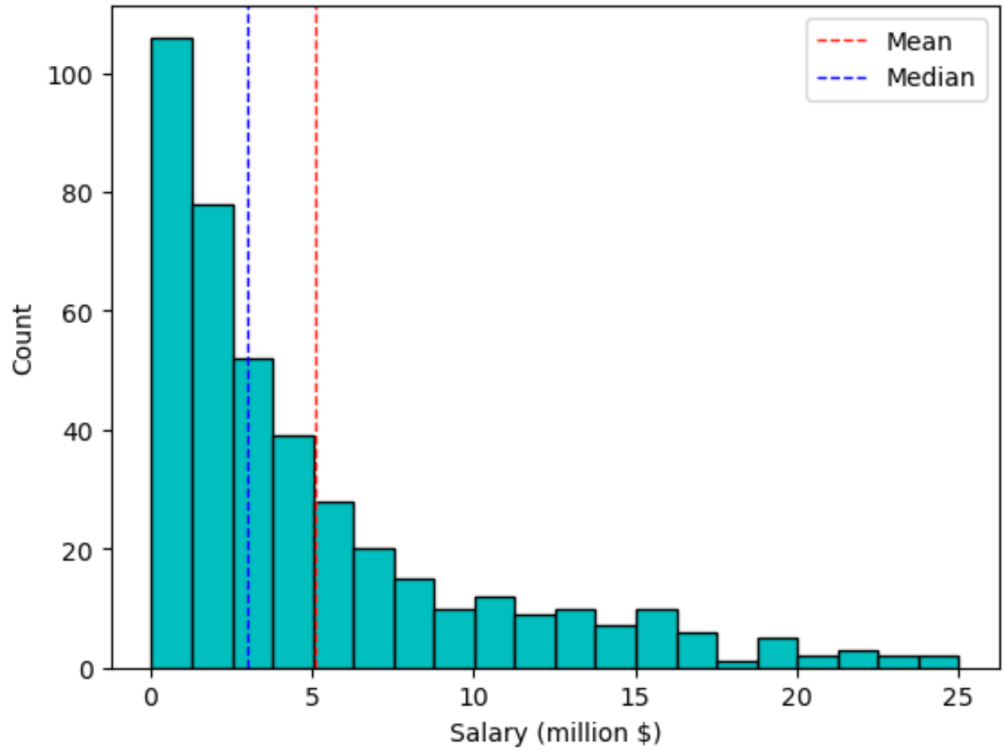






Quantitative data: statistics for central tendency

Two statistics for measuring the “central value” of a sample of quantitative data are the ***mean*** and the ***median***



The mean

$$\text{Mean} = \frac{\text{Sum of all data values}}{\text{Number of data values}}$$

$$\text{Mean} = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{n} = \sum_{i=1}^{n} \frac{x_i}{n} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

```
import statistics
statistics.mean(data_list)
```

The median

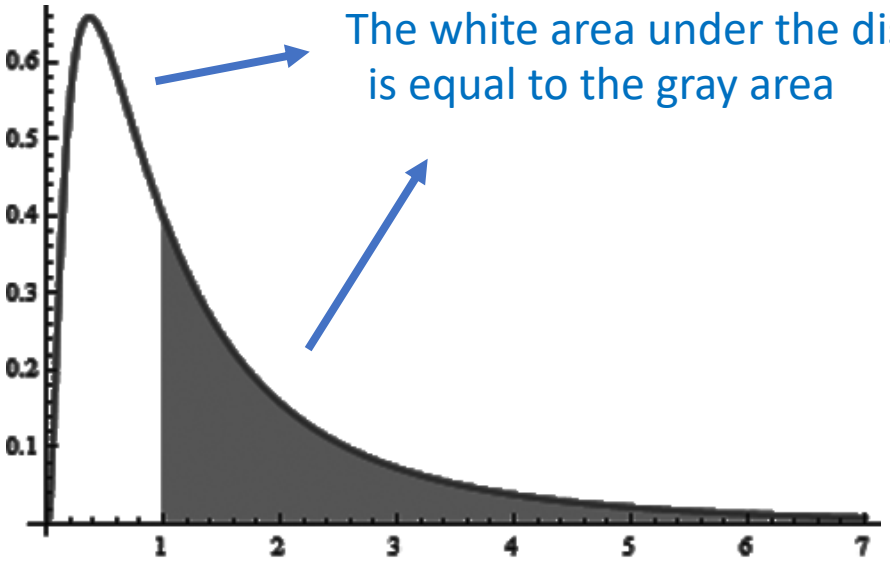
The **median** is a value that splits the data in half

- i.e., half the values in the data are smaller than the median and half are larger

To calculate the median for a data sample of size *n*, sort the data and then:

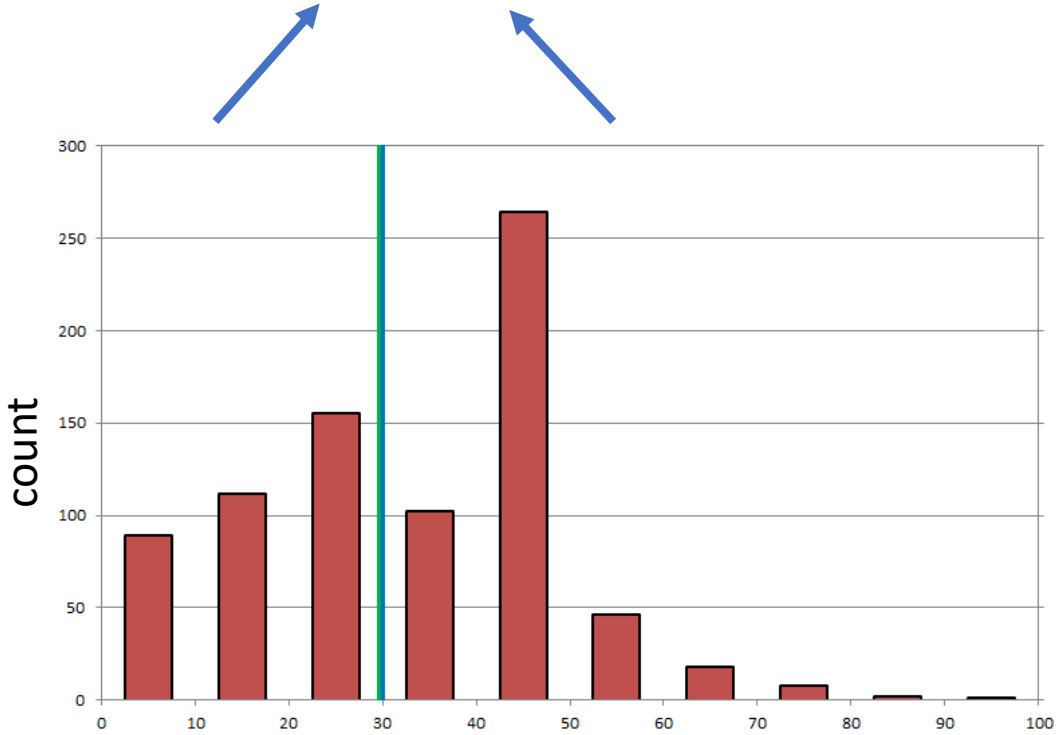
- If n is odd: The middle value of the sorted data
- If n is even: The average of the middle two values of the sorted data

The median



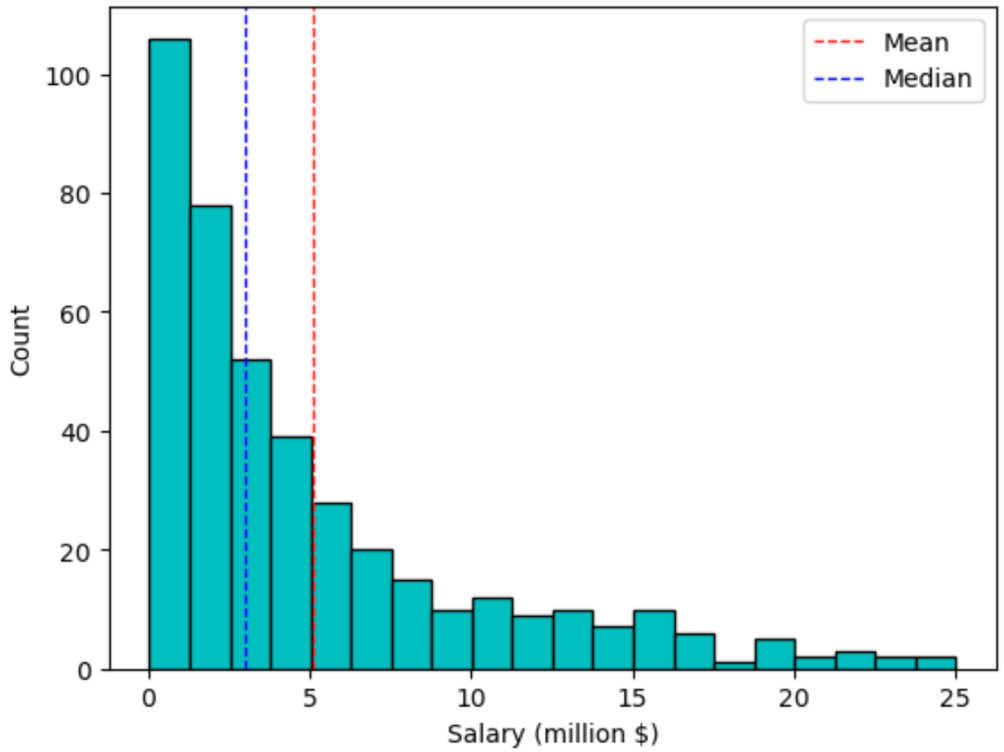
The white area under the distribution is equal to the gray area

The sum of the heights of the bars on the left is equal to the sum of the heights of the bars on the right




```
import statistics
statistics.median(data_list)
```

Labeling axes!



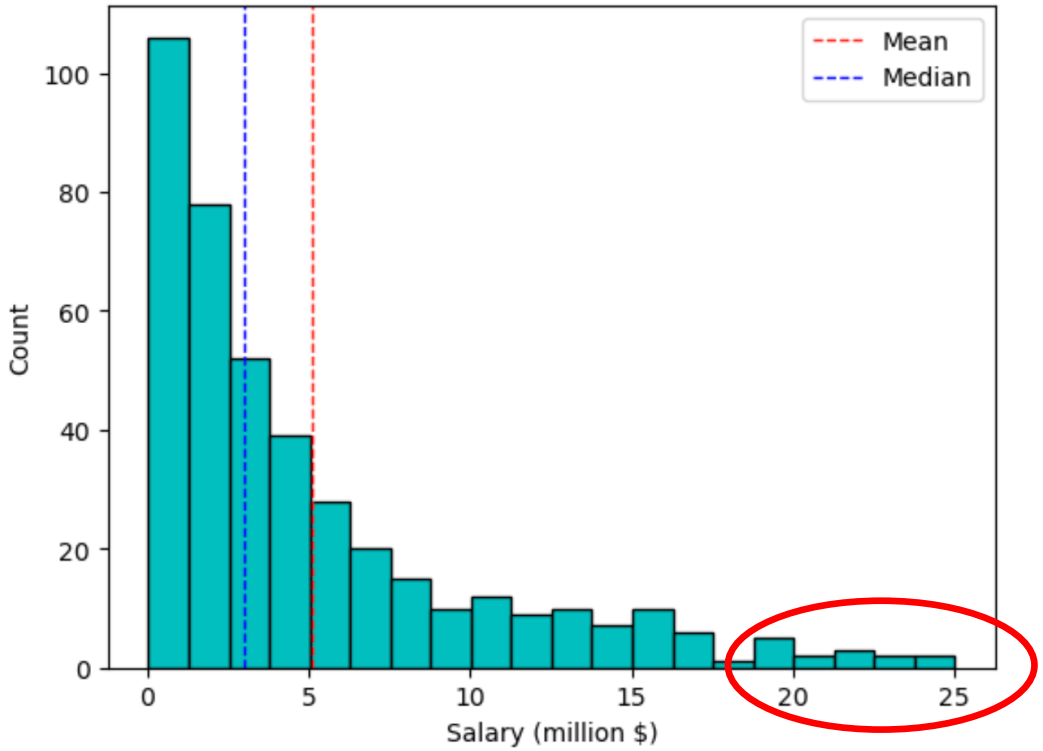

```
plt.hist(salary_list, edgecolor = "k");

plt.ylabel("Count")

plt.xlabel("Salary (million $)");
```

Outliers

An **outlier** is an observed value that is notably distinct from the other values in a dataset by being much smaller or larger than the rest of the data.

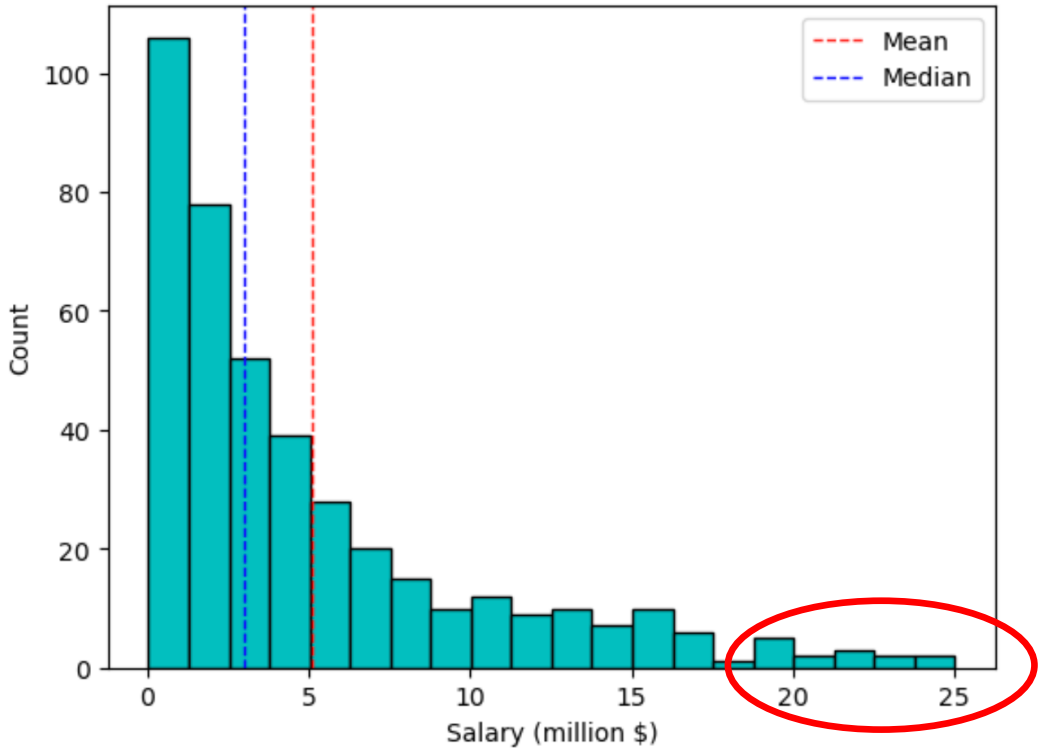


Outliers can potentially have a large influence on the statistics you calculate

One should examine outliers to understand what is causing them

- If there are due to an error, remove them
- Otherwise, need to think about how to treat them
  - Could be interesting phenomenon
  - Could restrict data to a particular range of values
  - Etc.

Outliers' impact on mean and median



The median is *resistant* to outliers

- i.e., not affected much by outliers

The mean is not resistant to outliers

What is the mean and median of the data: 1, 2, 3, 4, 990?

- Mean = 200
- Median = 3

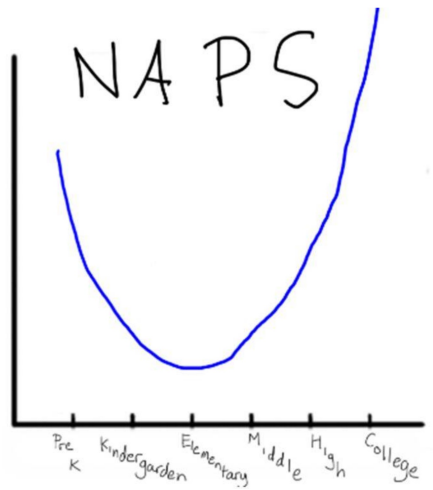

ANY
QUESTIONS?

Visualizing sequences: 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:

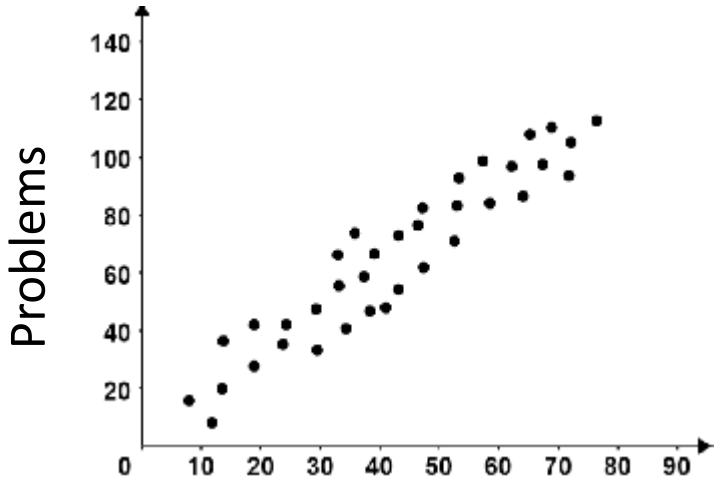
- plt.plot(y) # create a line graph with successive integers as the x-values
- plt.plot(x, y) # plots y as a function of x
- plt.plot(x, y, '-o') # creates lines with circle markers



Visualizing two quantitative variables: 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.

```
plt.plot(x, y, '.')
```


```
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!