YData: Introduction to Data Science

Class 03: Python basics continued

Overview

Review of an intro to Python

- Expressions, Names, Call expressions, Data types

More intro to Python

- Lists
- Dictionaries

Brief discussion of statistics

- Proportions, histograms, mean and median

If there is time

- For loops
- Conditional statements

Announcement

There is another way to access Jupyter Notebooks through the Yale Center for Research Computing (YCRC)

You can open a Jupyter notebook (and the ydat123_2023e conda environment) using: https://ydata123.ycrc.yale.edu/pun/sys/dashboard/batch_connect/sys/ydata123_jupyter/session_contexts/new

Note: you must be on Yale's network to access this cluster

- Use Cisco VPN to get on Yale's network from off campus

Important: everything stored on your account will be deleted 30 days after the end of the semester so be sure to back everything up at the end of the semester!

Announcement: Homework 1

Homework 1 has been posted!

It is due on Gradescope on Sunday January 29th 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, so titrate the difficulty of your analysis to challenge yourself appropriately
  - Ok, if you can't fully answer your question, we will be using easier/more powerful methods later in the semester.
- I think the problems are at a reasonable level of difficulty but not 100% sure
  - I will get your feedback to improve in the future…

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 review 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!

Dictionaries

Dictionaries

Dictionaries allow you to look up ***values*** based on a ***key***

- i.e., you supply a “key” and the dictionary returns the stored value

We can create dictionaries using the syntax:

- my_dict = { 'key1': 5, ‘key2’: 20}

We can retrieve dictionary values by supplying a key using square brackets []

- my_dict['key2’]

Let's explore this in Jupyter!

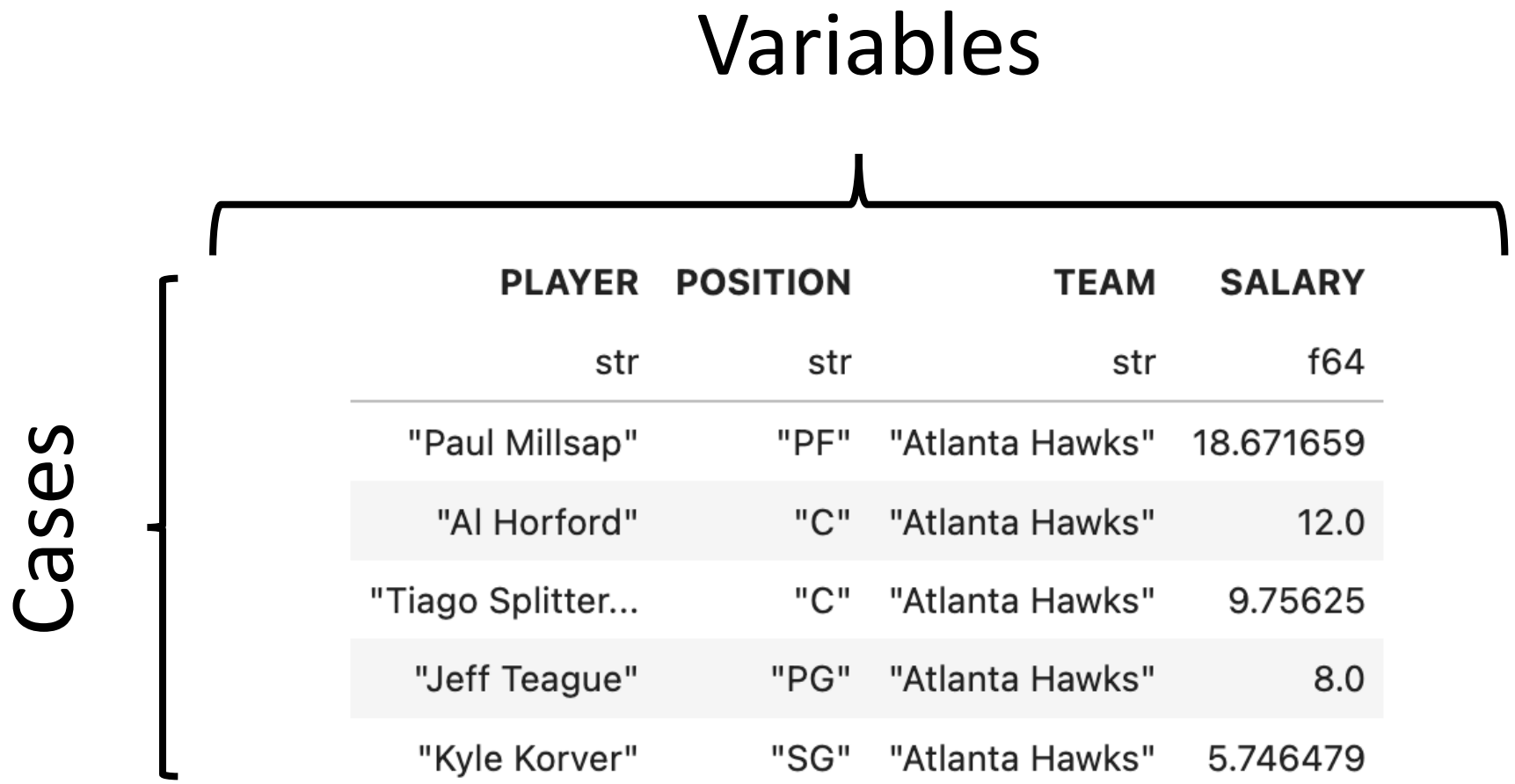
A brief introduction to statistics…

Example: NBA salaries

Let's explore salaries of NBA players (from the 2015-2016 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

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

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

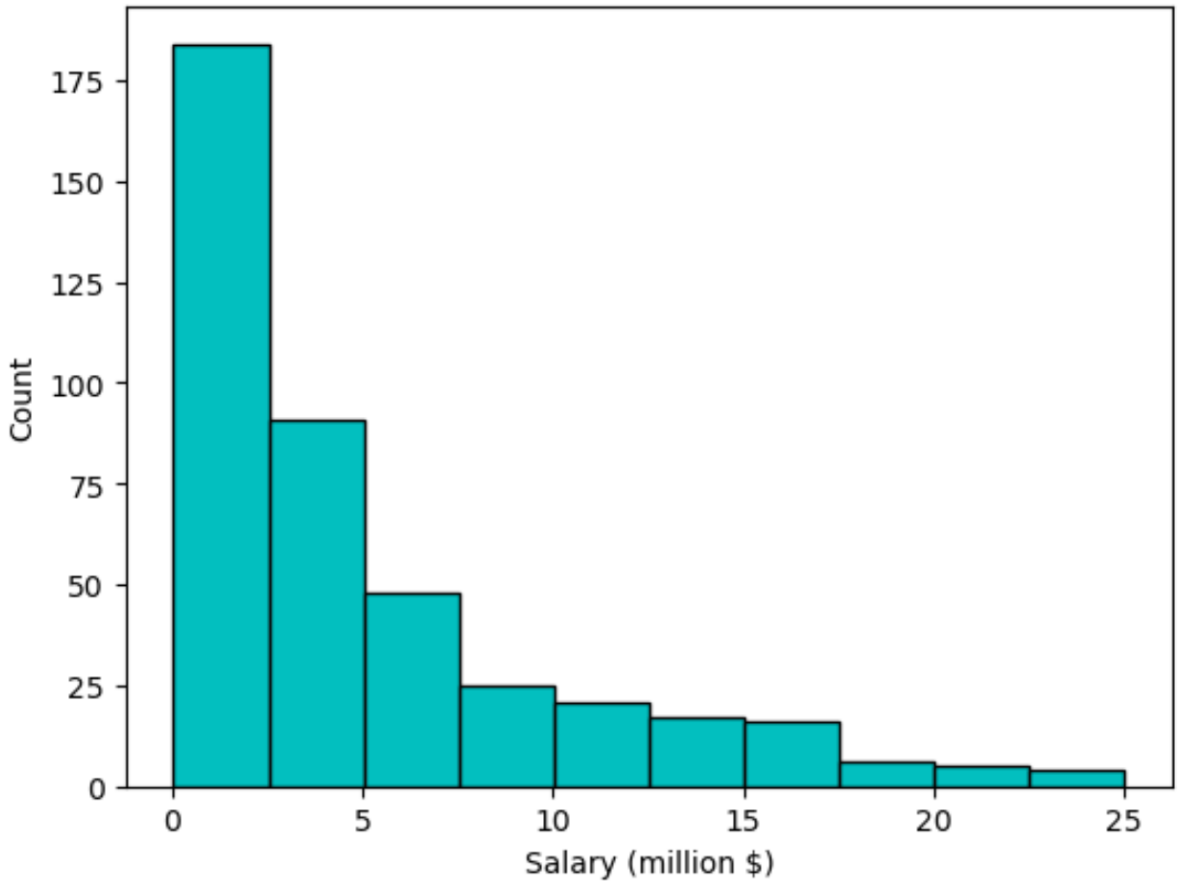
- 0-2.5, 2.5-5, 5-7.5, … 20-22.5, 22.5-25.0

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

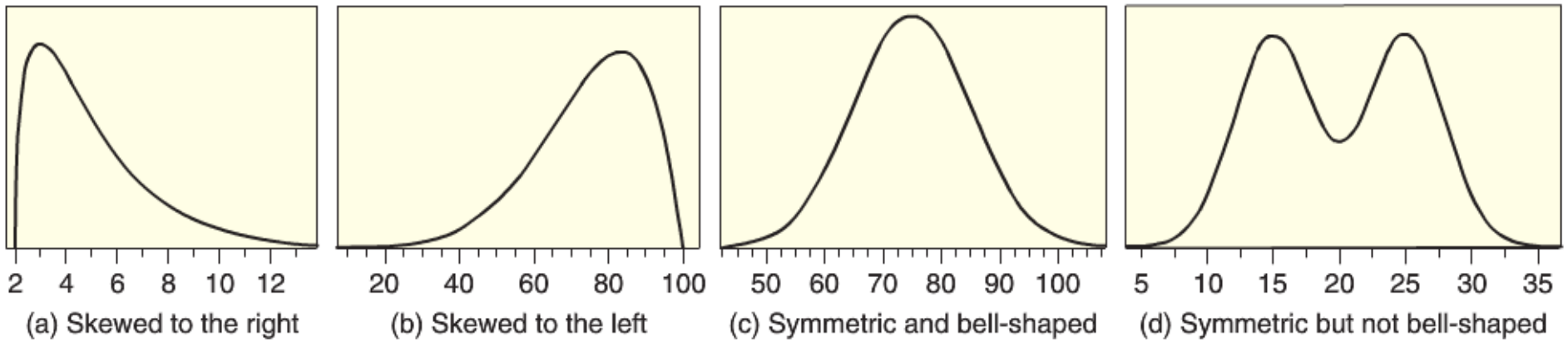
Histograms – countries life expectancy in 2007

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

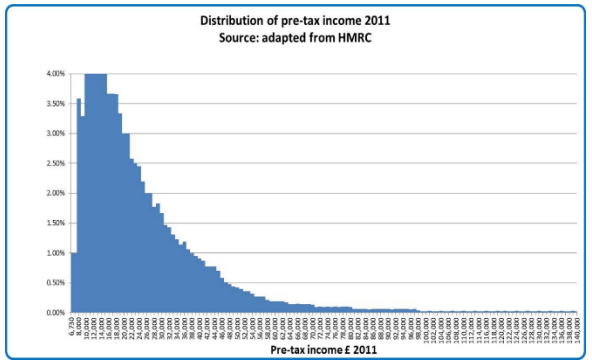


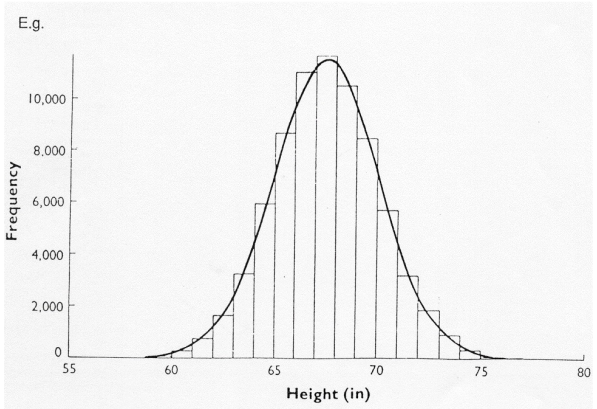
`Matplotlib: plt.hist(data)`

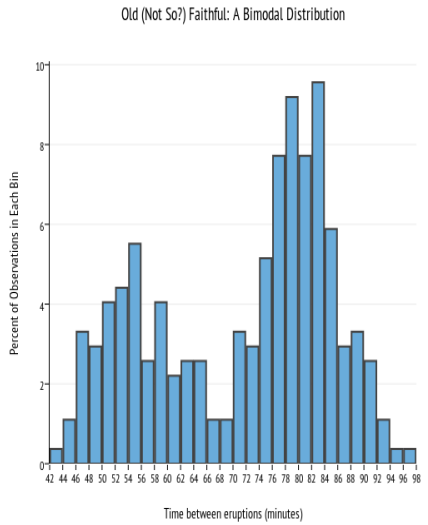
Common shapes of data distributions



Income distribution

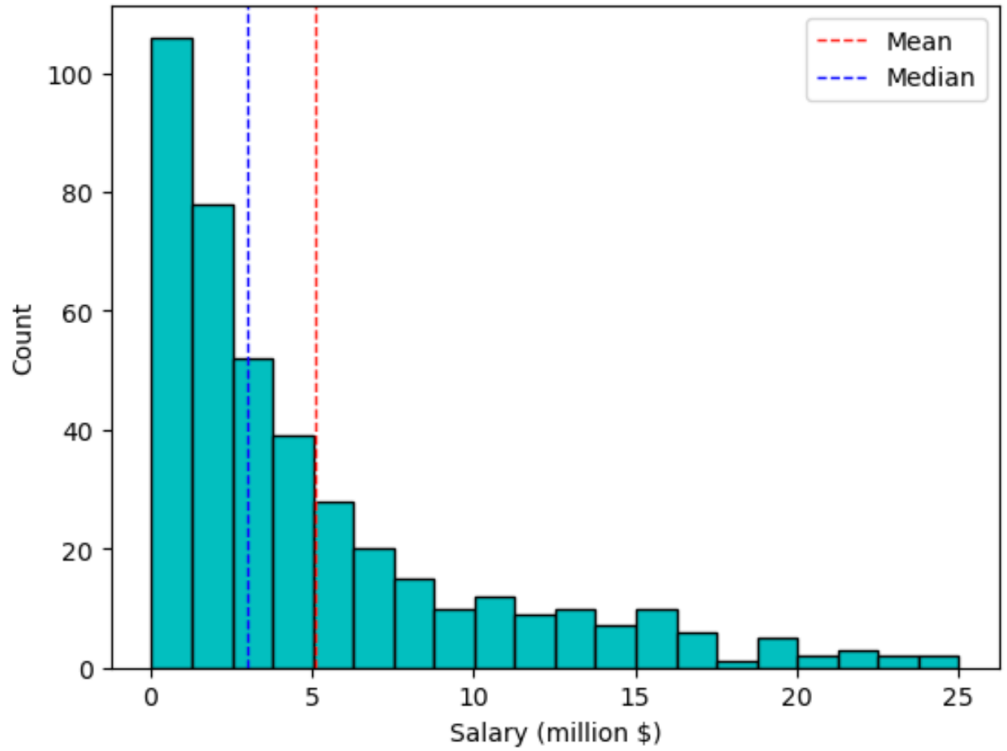






Quantitative data: statistics for central tendency

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

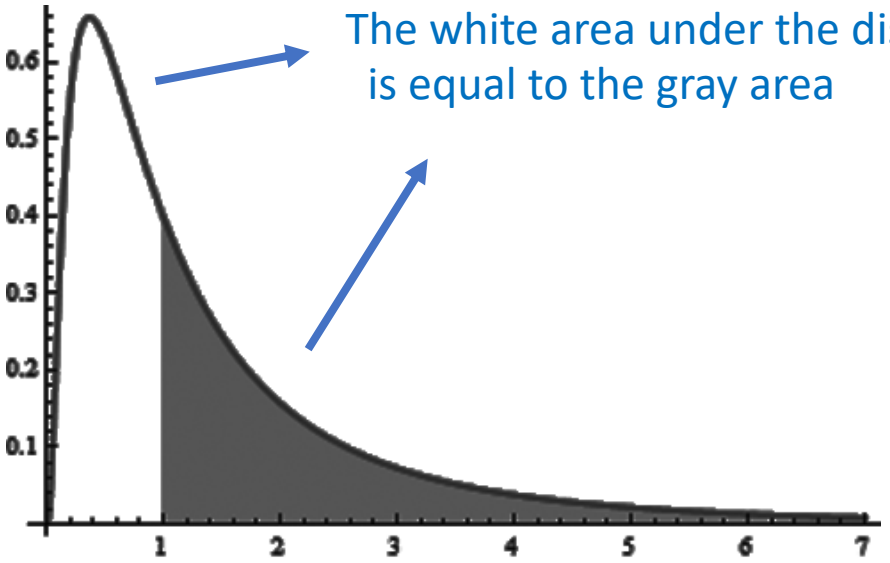
The **median** is a value that splits the data in half

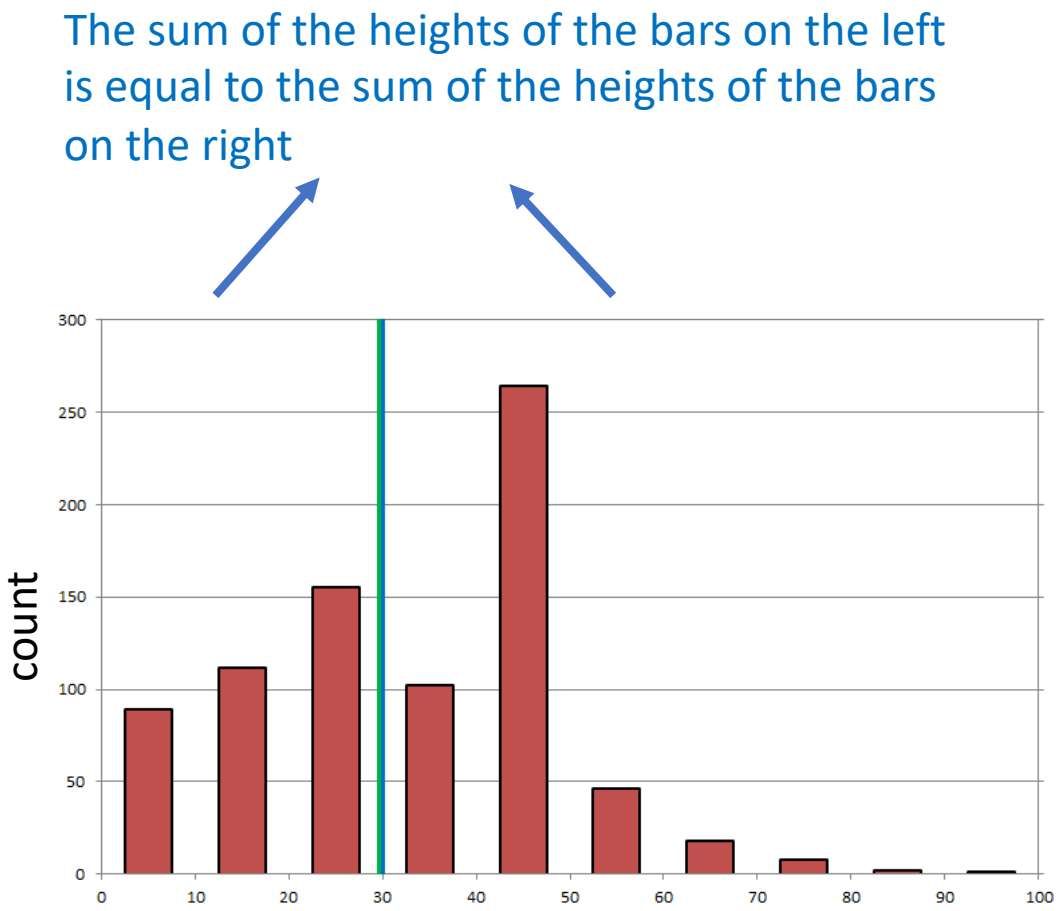
- i.e., half the values in the data are smaller than the median and half are larger

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

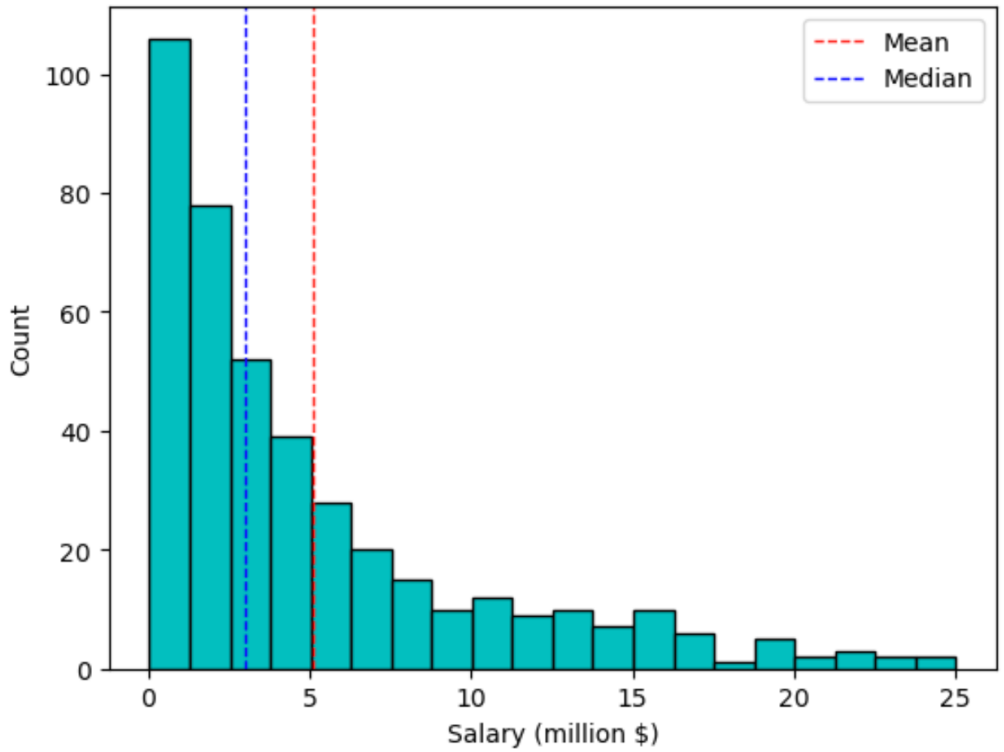
The median






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

Measure of central tendency: 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?

Back to Python…

Loops

For loops

For loops repeat a process many times, iterating over a sequence of items

- Often we are iterating over an array of sequential numbers

```
animals =  ["cat", "dog", "bat”]
for creature in animals:
    print(creature)

for i in np.arange(4):
    print(i**2)
```

Ranges

A range gives us a sequence of consecutive numbers

An sequence of increasing integers from 0 up to *end* - 1

- range(end)

An sequence of increasing integers from *start* up to *end* - 1

- range(start, end)

A sequence with step between consecutive values

- range(start, end, step)

The range always includes start but excludes end

Let's explore this in Jupyter!

Conditional statements

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 also make comparisons across elements in an array

Let's explore this in Jupyter!

Conditional statements

Conditional statements control the sequence of computations that are performed in a program

We use the keyword if to begin a conditional statement to only execute lines of code if a particular condition is met.

We can use elif to test additional conditions

We can use an else statement to run code if none of the if or elif conditions have been met.

```
num = 5
if num == 1:
    print("Monday")
elif num == 2:
    print("Tuesday")
elif num == 3:
    print("Wednesday")
elif num == 4:
    print("Thursday")
elif num == 5:
    print("Friday")
elif num == 6:
    print("Saturday")
elif num == 7:
    print("Sunday")
else:
    print("Invalid input")
```

Let's explore this in Jupyter!