Unit I

Introduction

The Ascendance of Data

- We live in a world that's drowning in data.
  - Websites track every user's every click. Your smartphone is building up a record of your location and speed every second of every day.
  - "Quantified selfers" wear pedometers-on-steroids that are ever recording their heart rates, movement habits, diet, and sleep patterns.
  - Smart cars collect driving habits,
  - smart homes collect living habits, and
  - smart marketers collect purchasing habits.
- The Internet itself represents a huge graph of knowledge that contains (among other things) an enormous cross-referenced encyclopedia; domain-specific databases about movies, music, sports results, pinball machines, memes, and cocktails; and too many government statistics (some of them nearly true!) from too many governments to wrap your head around.
- Buried in these data are answers to countless questions that no one's ever thought to ask.

What Is Data Science?

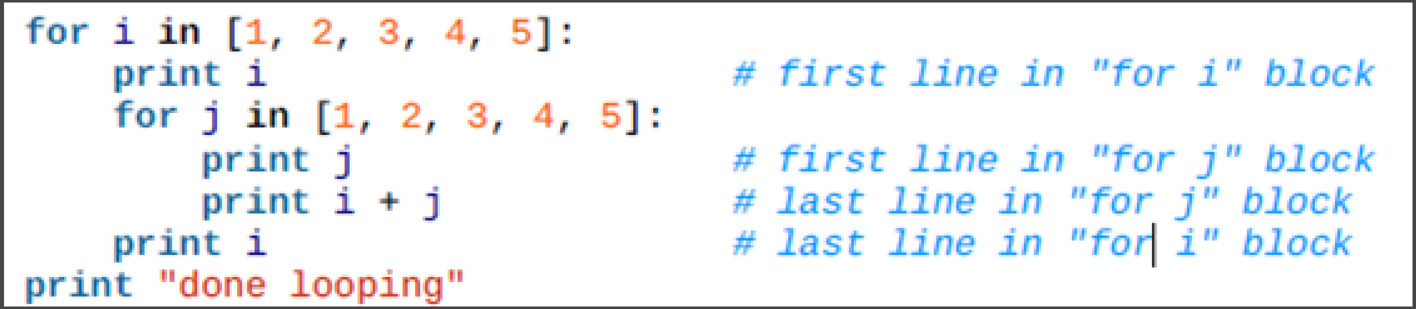
- There’s a joke that says a data scientist is someone who knows more statistics than a computer scientist and more computer science than a statistician
- In fact, some data scientists are — for all practical purposes — statisticians, while others are pretty much indistinguishable from software engineers. Some are machine-learning experts, Some are PhDs with impressive publication records.
- We define it as “***a data scientist is someone who extracts insights from messy data.”***
- Today’s world is full of people trying to turn data into insight.

- Facebook asks you to list your hometown and your current location, ostensibly to make it easier for your friends to find and connect with you. But it also analyzes these locations to identify global migration patterns.
- As a large retailer, Target tracks your purchases and interactions, both online and in-store.
- **Obama Case Study**
- Some data scientists also occasionally use their skills for good — using data to make government more effective, to help the homeless, and to improve public health.

A Crash Course in Python

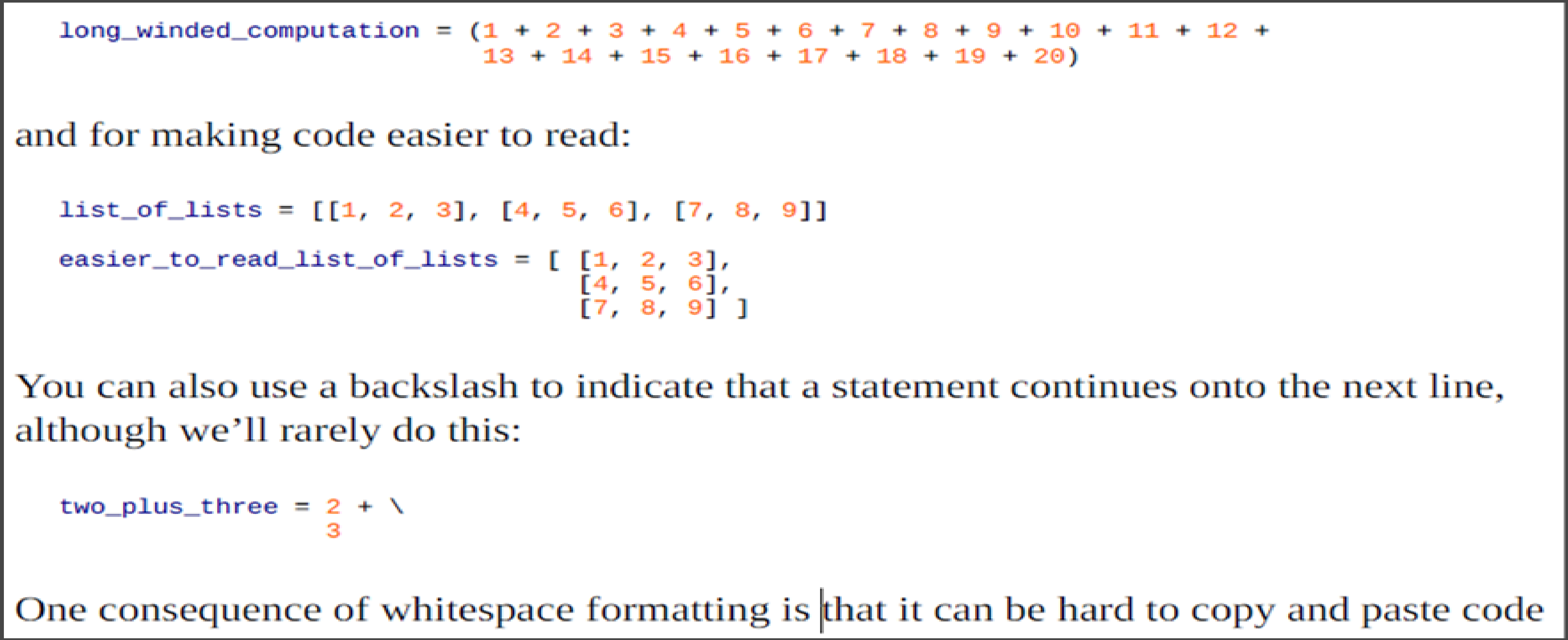
Whitespace Formatting

- Many languages use curly braces to delimit blocks of code. Python uses indentation:



```
for i in [1, 2, 3, 4, 5]:
    print i                     # first line in "for i" block
    for j in [1, 2, 3, 4, 5]:
        print j                 # first line in "for j" block
        print i + j             # last line in "for j" block
    print i                     # last line in "for i" block
print "done looping"
```

• This makes Python code very readable, but it also means that you have to be very careful with your formatting.
• Whitespace is ignored inside parentheses and brackets, which can be helpful for long-winded computations:



```
long_winded_computation = (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 +
                           13 + 14 + 15 + 16 + 17 + 18 + 19 + 20)
```

and for making code easier to read:

```
list_of_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

easier_to_read_list_of_lists = [ [1, 2, 3],
                                 [4, 5, 6],
                                 [7, 8, 9] ]
```

You can also use a backslash to indicate that a statement continues onto the next line, although we'll rarely do this:

```
two_plus_three = 2 + \
                 3
```

One consequence of whitespace formatting is that it can be hard to copy and paste code

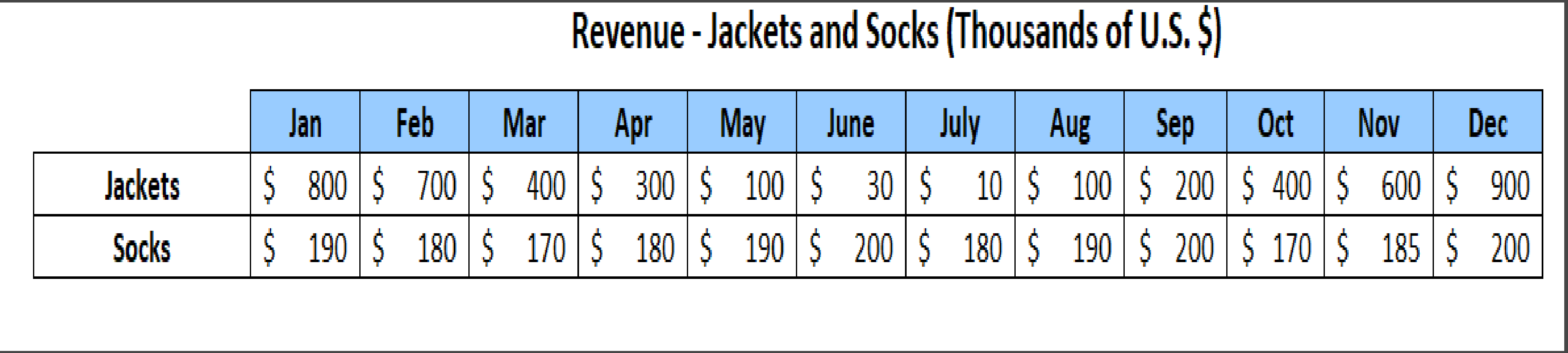
Data Visualization

- Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.
- **Data visualization** is the graphic representation of data. It involves producing images that communicate relationships among the represented data to viewers of the images.
- This communication is achieved through the use of a systematic mapping between graphic marks and data values in the creation of the visualization.

- In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.
- Increased amounts of data created by Internet activity and an expanding number of sensors in the environment are referred to as "big data" or Internet of things. Processing, analyzing and communicating this data present ethical and analytical challenges for data visualization.
- The field of data science and practitioners called data scientists help address this challenge.

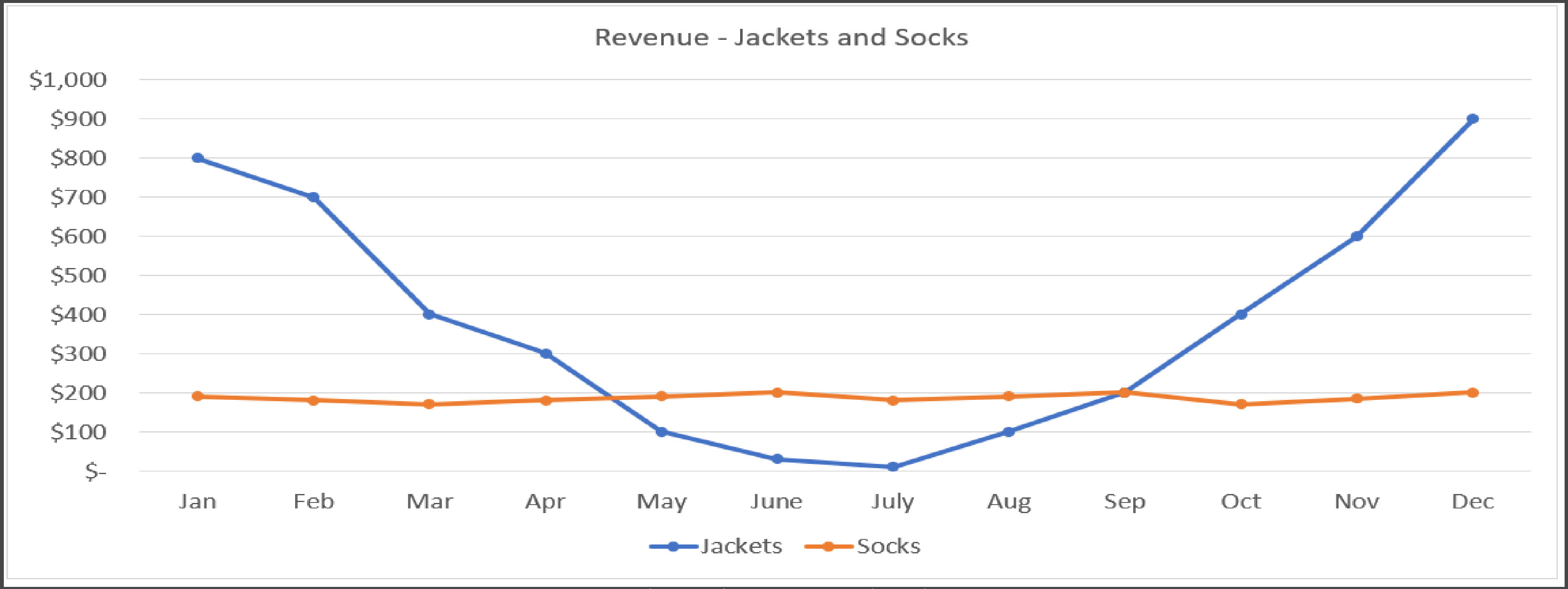
The advantages and benefits of good data visualization

- Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from circle.
- Our culture is visual, including everything from art and advertisements to TV and movies.
- Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It’s storytelling with a purpose.
- If you’ve ever stared at a massive spreadsheet of data and couldn’t see a trend, you know how much more effective a visualization can be.
- It gives you improved insight
- It helps faster decision making



Revenue - Jackets and Socks (Thousands of U.S. $)

	Jan	Feb	Mar	Apr	May	June	July	Aug	Sep	Oct	Nov	Dec
Jackets	$ 800	$ 700	$ 400	$ 300	$ 100	$ 30	$ 10	$ 100	$ 200	$ 400	$ 600	$ 900
Socks	$ 190	$ 180	$ 170	$ 180	$ 190	$ 200	$ 180	$ 190	$ 200	$ 170	$ 185	$ 200



Examples of diagrams used for data visualization

- Bar Chart
- Line Chart
- Histogram
- Scatter Plot
- Heat Map

matplotlib

- A wide variety of tools exists for visualizing data. We will be using the matplotlib library, which is widely used (although sort of showing its age).
- If you are interested in producing elaborate interactive visualizations for the Web, it is likely not the right choice, but for simple bar charts, line charts, and scatterplots, it works pretty well.
- In particular, we will be using the matplotlib.pyplot module.
- In its simplest use, pyplot maintains an internal state in which you build up a visualization step by step. Once you're done, you can save it (with savefig()) or display it (with show()).

Bar Charts

- Bar charts are such a popular graph visualization because of how easy you can scan them for quick information. Bar charts organize data into rectangular bars that make it a breeze to compare related data sets.
- Bar graph uses bars to compare data among different categories. It is well suited when you want to measure the changes over a period of time. It can be represented horizontally or vertically. Also, the important thing to keep in mind is that longer the bar, greater is the value.
- Matplotlib API provides the **bar()** function that can be used in the MATLAB style use as well as object oriented API.

- The signature of bar() function to be used is as follows:
  - bar(x, height, width, bottom, align)
- *bars are by default width 0.8*

```
from matplotlib import pyplot as plt
x = [5,8,10]
y = [12,16,6]
x2 = [6,9,11]
y2 = [6,15,7]
plt.bar(x, y)
plt.bar(x2, y2, color='g', align='center')
plt.title('Epic Info')
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```

- **Use a bar chart for the following reasons:**
  - You want to compare two or more values in the same category
  - You want to compare parts of a whole
  - You don't have too many groups (less than 10 works best)
  - You want to understand how multiple similar data sets relate to each other
- **Don't use a bar chart for the following reasons:**
  - The category you're visualizing only has one value associated with it
  - You want to visualize continuous data

- charts can be horizontal, to create a horizontal bar chart:
- import matplotlib.pyplot as plt

```
import numpy as np

objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]

plt.barh(y_pos, performance, align='center', alpha=0.5)
plt.yticks(y_pos, objects)
plt.xlabel('Usage')
plt.title('Programming language usage')

plt.show()
```

Line Charts

- As we saw already, we can make line charts using plt.plot(). These are a good choice for showing *trends*
- Line charts help to visualize data in a compact and precise format which makes it easy to rapidly scan information in order to understand trends.
- Line charts are used to show resulting data relative to a continuous variable - most commonly time or money.
- The proper use of color in this visualization is necessary because different colored lines can make it even easier for users to analyze information.

- **Use a line chart for the following reasons:**
  - You want to understand trends, patterns, and fluctuations in your data
  - You want to compare different yet related data sets with multiple series
  - You want to make projections beyond your data
- **Don’t use a line chart for the following reason:**
  - You want to demonstrate an in-depth view of your data

- import matplotlib.pyplot as plt

  ```
  t = arange(0.0, 20.0, 1);
  s = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];
  plt.plot(t, s);

  plt.xlabel('Item (s)');
  plt.ylabel('Value');
  plt.title('Python Line Chart: Plotting numbers');
  plt.grid(True);
  plt.show();
  ```
- t = arange(0.0, 20.0, 1)
  defines start from 0, plot 20 items (length of our array) with steps of 1.

Scatterplots

- A scatterplot is the right choice for visualizing the relationship between two paired sets of data.
- Another commonly used plot type is the simple scatter plot, a close cousin of the line plot. Instead of points being joined by line segments, here the points are represented individually with a dot, circle, or other shape.
- Matplot has a built-in function to create *scatterplots* called scatter(). A scatter plot is a type of plot that shows the data as a collection of points. The position of a point depends on its two-dimensional value, where each value is a position on either the horizontal or vertical dimension.
- A third variable can be set to correspond to the color or size of the markers, thus adding yet another dimension to the plot.

```
import matplotlib.pyplot as plt
girls_grades = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
boys_grades = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
grades_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.scatter(grades_range, girls_grades, color='r')
plt.scatter(grades_range, boys_grades, color='b')
plt.xlabel('Grades Range')
Plt.ylabel('Grades Scored')
Plt.title('scatter plot')
plt.show()
```

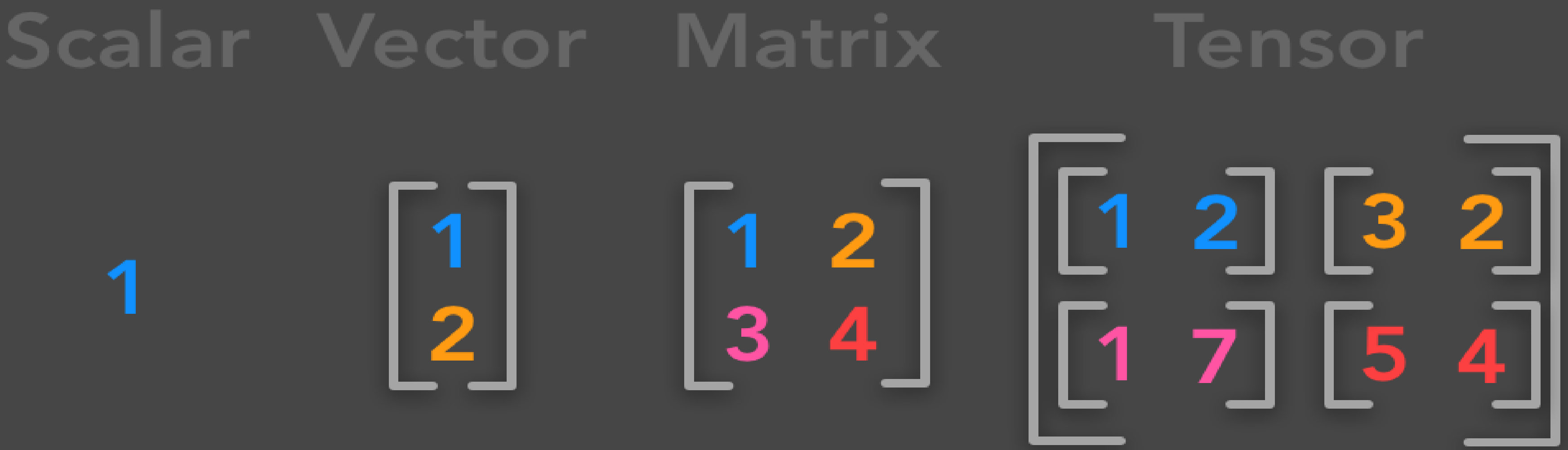
Linear Algebra

- *Linear algebra is the branch of mathematics concerning linear equations, linear functions and their representations through matrices and vector spaces.*
- The word *algebra* comes from the Arabi which means "*the reunion of broken parts*". This is collection of methods deriving unknowns from knowns in mathematics.
- *Linear Algebra* is the branch that deals with *linear equations* and *linear functions* which are represented through *matrices* and *vectors*.
- In simpler words, it helps us understand geometric terms such as planes, in higher dimensions, and perform mathematical operations on them.
- By definition, algebra deals primarily with scalars but Linear Algebra has vectors and matrices to deal with linear equations and functions.

- **Why Linear Algebra is significant in Data Science?**
- *Linear Algebra* is central to almost all areas of mathematics like *geometry* and *functional analysis*. Its concepts are a crucial prerequisite for understanding the theory behind *Data Science*.
- At some point, you may want to gain a better understanding of how the *different algorithms* really work under the hood.

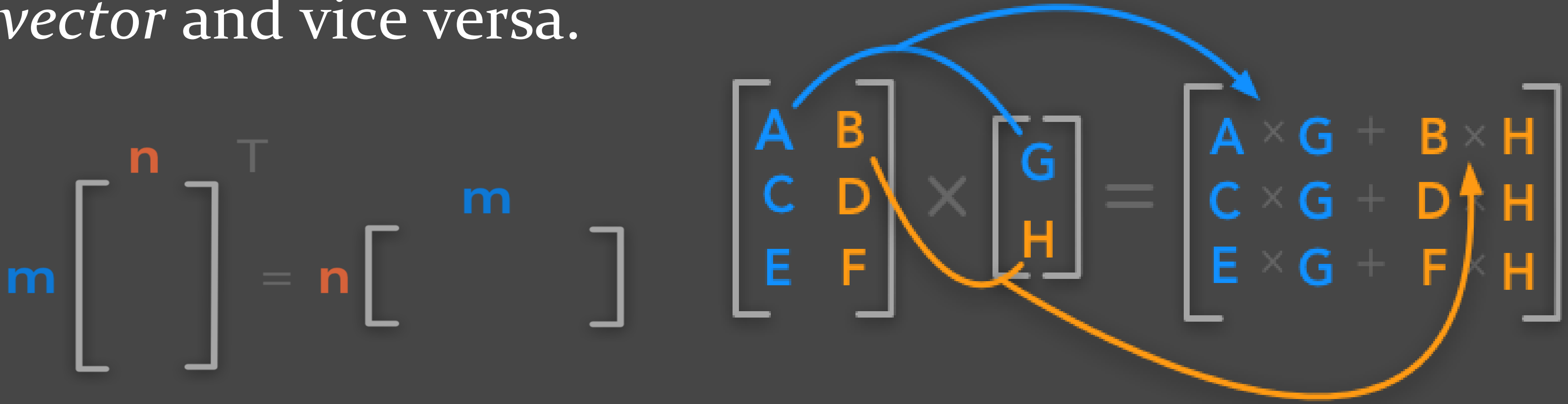
How Linear Algebra is used in Data Science?

- A *scalar* is a single number
- A *vector* is an array of numbers.
- A *matrix* is a 2-D array
- A *tensor* is a n-dimensional array with n>2



Some important concepts:

- *Transposition:* we can convert a *row vector* to a *column vector* and vice versa.



- **Multiplying Matrices and Vectors:** The dot product of matrices & vectors is used in every equation explaining data science algorithms. Matrices multiplication is *distributive*, *associative*, *NOT commutative*, while vector multiplication is *commutative*.

● **Identity and Inverse Matrices**

● The *identity matrix In* is a special matrix of shape (n×n) that is filled with 0 except the diagonal that is filled with 1. An *inverse matrix* is that results in the identity matrix when it is multiplied by its original form.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^{-1}A = I_n$$

- **Special Kinds of Matrices and Vectors**
- This section covers different interesting type of matrices with specific properties
  i.e. *Diagonal*, *Symmetric* & *Orthogonal* matrices.
  - A matrix Ai,j is diagonal if its entries are all zeros except on the diagonal (when i=j).
  - The matrix A is symmetric if it is equal to its transpose: $\mathbf{A=A^T}$. This concerns only square matrices.
  - A matrix is orthogonal if columns are mutually orthogonal and have a unit norm (orthonormal) and rows are mutually orthonormal and have unit norm.
  - $\mathbf{A^TA=1}$
-

Diagonal matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Symmetric matrix

$$\begin{bmatrix} 1 & 2 & 6 \\ 2 & 8 & 4 \\ 6 & 4 & 5 \end{bmatrix}$$

Orthogonal matrix

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Orthonormal rows

Orthonormal columns

numpy

- NumPy is a module for Python. The name is an acronym for "Numeric Python" or "Numerical Python".
- NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays.
- SciPy (Scientific Python) is often mentioned in the same breath with NumPy.
- SciPy needs Numpy, as it is based on the data structures of Numpy and furthermore its basic creation and manipulation functions.
- It extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier-transformation and many others.
- Both NumPy and SciPy are not part of a basic Python installation. They have to be installed after the Python installation. NumPy has to be installed before installing SciPy.

- Creating a Vector

```
# Load library
import numpy as np
# Create a vector as a row
vector_row = np.array([1, 2, 3])
# Create a vector as a column
vector_column = np.array([[1], [2], [3]])
```

- Creating a Matrix

```
import numpy as np
# Create a matrix
matrix = np.array([[1, 2], [1, 2], [1, 2]])
```

Selecting Elements

```
import numpy as np
vector = np.array([1, 2, 3, 4, 5, 6])
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
vector[2]
matrix[1,1]
# Select all elements of a vector
vector[:]
# Select everything up to and including the third element
   vector[:3]
# Select everything after the third element
vector[3:]
# Select the last element
vector[-1]
# Select the first two rows and all columns of a matrix
matrix[:2,:]
# Select all rows and the second column
matrix[:,1:2]
```

- Describing a Matrix
  - You want to describe the shape, size, and dimensions of the matrix.
  - Use shape, size, and ndim:

```
import numpy as np
matrix = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
# View number of rows and columns
matrix.shape
# View number of elements (rows * columns)
matrix.size
# View number of dimensions
matrix.ndim
```

- Finding the Maximum and Minimum Values
- Calculating the Average, Variance, and Standard Deviation
- Reshaping Arrays
- Transposing a Vector or Matrix
- Flattening a Matrix
- Getting the Diagonal of a Matrix
- Calculating the Trace of a Matrix
- Calculating Dot Products
- Adding and Subtracting Matrices
- Multiplying Matrices
- Inverting a Matrix
- Generating Random Values

Trace of matrix: sum of diagonal elements

matrix.trace()

np.mean()

np.median()

https://acadgild.com/blog/python-mean-median-mode

https://www.hackerearth.com/blog/developers/descriptive-statistics-python-numpy

https://towardsdatascience.com/applying-statistics-in-python-part-ii-a66c9b2ccabd

Statistics

- Describing a Single Set of Data,
- Central Tendencies,
- Dispersion,
- Correlation,
- Simpson’s Paradox,
- Some Other Correlational Caveats,
- Correlation and Causation.

- *Statistics refers to the mathematics and techniques with which we understand data.*
- Statistics is a Mathematical Science pertaining to data collection, analysis, interpretation and presentation.
- Statistics is used to process complex problems in the real world so that Data Scientists and Analysts can look for meaningful trends and changes in Data. In simple words, Statistics can be used to derive meaningful insights from data by performing mathematical computations on it.
- Several Statistical functions, principles and algorithms are implemented to analyse raw data, build a Statistical Model and infer or predict the result.

- The field of Statistics has an influence over all domains of life, the Stock market, life sciences, weather, retail, insurance and education are but to name a few.

Stock Market
Education
Life Sciences
Statistics
Insurance
Weather
Retail

- **Terminologies In Statistics – Statistics For Data Science**
- One should be aware of a few key statistical terminologies while dealing with Statistics for Data Science. I’ve discussed these terminologies below:
  - ***Population*** *is the set of sources from which data has to be collected.*
  - *A* ***Sample*** *is a subset of the Population*
  - *A* ***Variable*** *is any characteristics, number, or quantity that can be measured or counted. A variable may also be called a data item.*
  - *Also known as a statistical model, A statistical* ***Parameter*** *or population parameter is a quantity that indexes a family of probability distributions. For example, the mean, median, etc of a population.*

- **Mean:** For a dataset, mean is said to be the average value of all the numbers. It can sometimes be used as a representation of the whole data. For instance, if you have the marks of students from a class, and you asked about how good is the class performing. It would be irrelevant to say the marks of every single student, instead, you can find the mean of the class, which will be a representative for class performance.
- To find the mean, sum all the numbers and then divide by the number of items in the set. For example, if the numbers are 1,2,3,4,5,6,7,8,8 then the mean would be 44/9 = 4.89.

- **Median:** Median of a set of numbers is usually the middle value. When the total numbers in the set are even, the median will be the average of the two middle values. Median is used to measure the central tendency. To calculate the median for a set of numbers, follow the below steps:
  1. Arrange the numbers in ascending or descending order
  2. Find the middle value, which will be n/2 (where n is the numbers in the set)

- **Mode:** Mode is the most frequent value occuring in the population. It is a metric to measure the central tendency, i.e. a way of expressing, in a (usually) single number, important information about a random variable or a population. Mode can be calculated using following steps:
  - Count the number of time each value appears
  - Take the value which appears the most
  - Let us understand it with an example: Suppose we have a dataset having 10 data points, listed below:

    4,5,2,8,4,7,6,4,6,3

So we see that the value 4 is repeating the most. So mode of this dataset is 4

Variance: Variance is used to measure the spread of given set of numbers and calculated by the average of squared distances from the mean. Let’s take an example, suppose the set of numbers we have is (600, 470, 170, 430, 300)

To Calculate:

1) Find the Mean of set of numbers, which is (600 + 470 + 170 + 430 + 300) / 5 = 394
2) Subtract the mean from each value which is (206, 76, -334, 36, -94)
3) Square each deviation from the mean which is (42436, 5776, 50176, 1296, 8836)
4) Find the Sum of Squares which is 108520
5) Divide by total number of items (numbers) which is 21704

Informally, it measures how far the set of numbers are spread from their average values.

https://www.researchgate.net/post/What_are_the_different_types_of_applications_of_variance_and_standard_deviation

Standard Deviation: The standard deviation of a set of values helps us understand how spread out those values are. This statistic is more useful than the variance because it's expressed in the same units as the values themselves. Mathematically, the standard deviation is the square root of the variance of a set. It's often represented by the greek symbol sigma, σ.

https://www.businessinsider.in/stock-market/heres-what-statisticians-mean-by-a-standard-deviation/articleshow/45353386.cms

https://simple.wikipedia.org/wiki/Standard_deviation

https://examples.yourdictionary.com/examples-of-standard-deviation.html

Describing a Single Set of Data

- Data set is simply the data itself:
- num_friends = [100, 49, 41, 40, 25, # ... and lots more]
- for a larger data, we use statistics to distill and communicate relevant features of our data.
- For better representation of any set of data we generate some statistics. Probably the simplest statistic is simply the number of data points:
  - num_points = len(num_friends)
- You're probably also interested in the largest and smallest values:
  - largest_value = max(num_friends)
  - smallest_value = min(num_friends)

Central Tendencies

Usually, we’ll want some notion of where our data is centered. Most commonly we’ll use

the mean (or average), which is just the sum of the data divided by its count:

If you have two data points, the mean is simply the point halfway between them. As you

add more points, the mean shifts around, but it always depends on the value of every

point.

We'll also sometimes be interested in the median, which is the middle-most value (if the number of data points is odd) or the average of the two middle-most values (if the number of data points is even).

For instance, if we have five data points in a sorted vector x, the median is x[5 // 2] or x[2]. If we have six data points, we want the average of x[2] (the third point) and x[3] (the fourth point).

Notice that — unlike the mean — the median doesn't depend on every value in your data. For example, if you make the largest point larger (or the smallest point smaller), the middle points remain unchanged, which means so does the median.

Less commonly you might want to look at the mode, or most-common value

Dispersion

Dispersion refers to measures of how spread out our data is. Typically they’re statistics for which values near zero signify not spread out at all and for which large values (whatever that means) signify very spread out.

For instance, a very simple measure is the range, which is just the difference between the largest and smallest elements: The range is zero precisely when the max and min are equal, which can only happen if the elements of x are all the same, which means the data is as undispersed as possible.

Conversely, if the range is large, then the max is much larger than the min and the data is more spread out.

A more complex measure of dispersion is the variance.

Variance has units that are the square of the original units.

As it can be hard to make sense of these, we often look instead at the standard deviation: Standard Deviation is square root of variance.

Correlation

Correlation, statistical technique which determines how one variables moves/changes in relation with the other variable. It gives us the idea about the degree of the relationship of the two variables. It's a bi-variate analysis measure which describes the association between different variables. In most of the business it's useful to express one subject in terms of its relationship with others.
For example: Sales might increase if lot of money is spent on product marketing
Why it is useful?
1. If two variables are closely correlated, then we can predict one variable from the other.
2. Correlation plays a vital role in locating the important variables on which other variables depend.
3. It's used as the foundation for various modeling techniques.
4. Proper correlation analysis leads to better understanding of data.
5. Correlation contribute towards the understanding of causal relationship(if any).

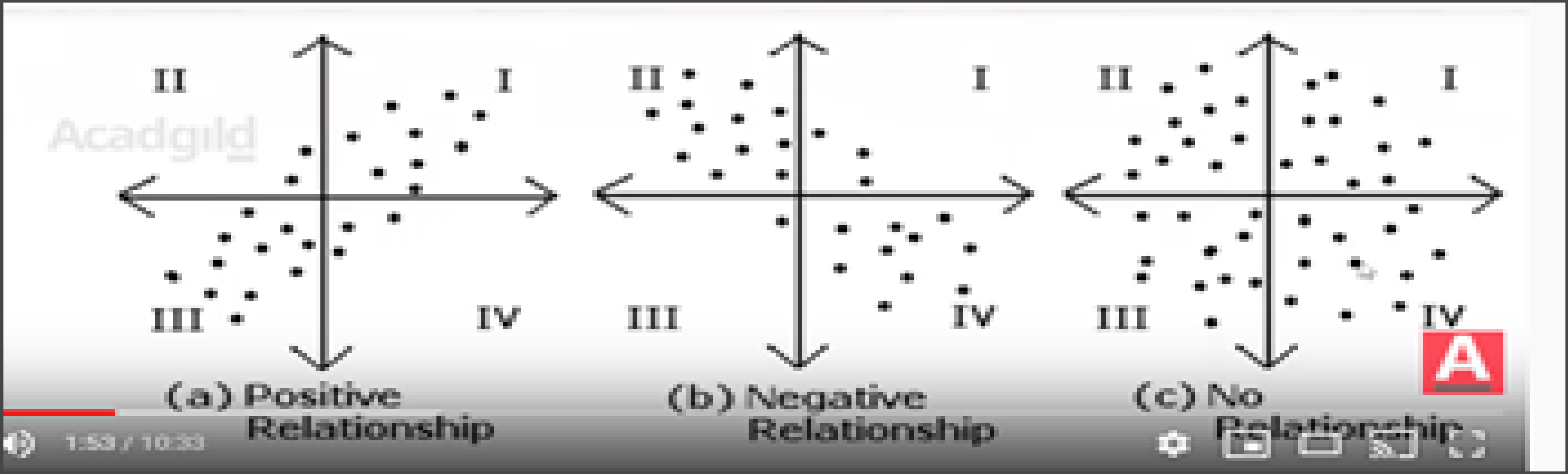
Covariance

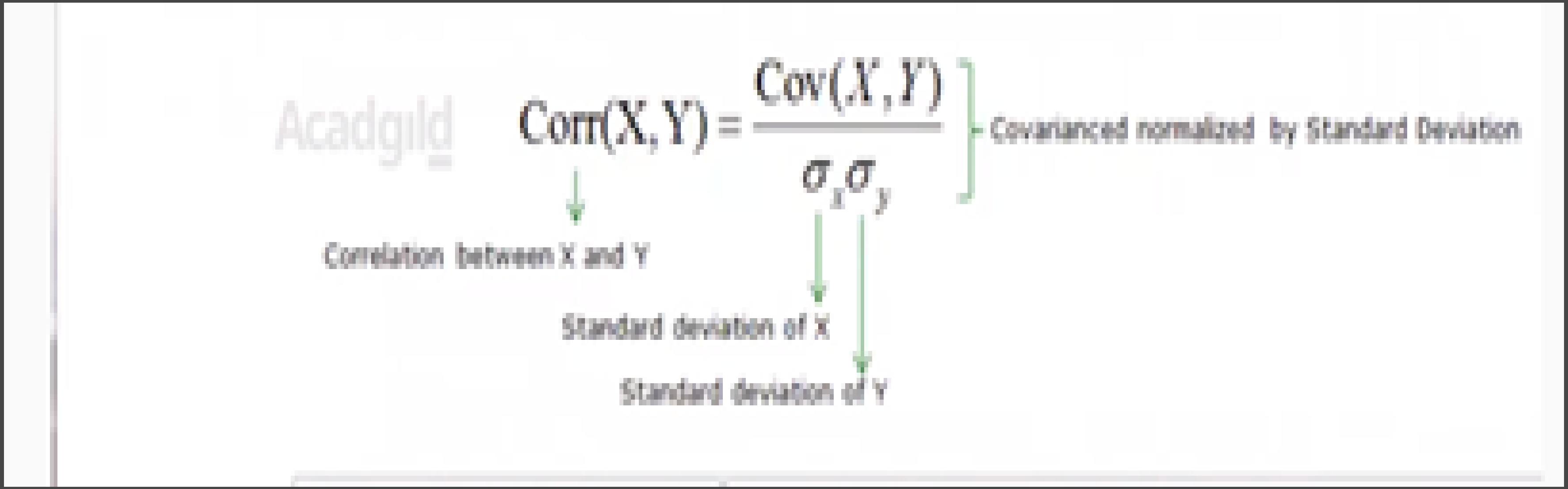
Two variables are related based on how these variables change in relation with each other.

But wait, is covariance same as correlation?

As covariance says something on same lines as correlation, correlation takes a step further than covariance and also tells us about the strength of the relationship.

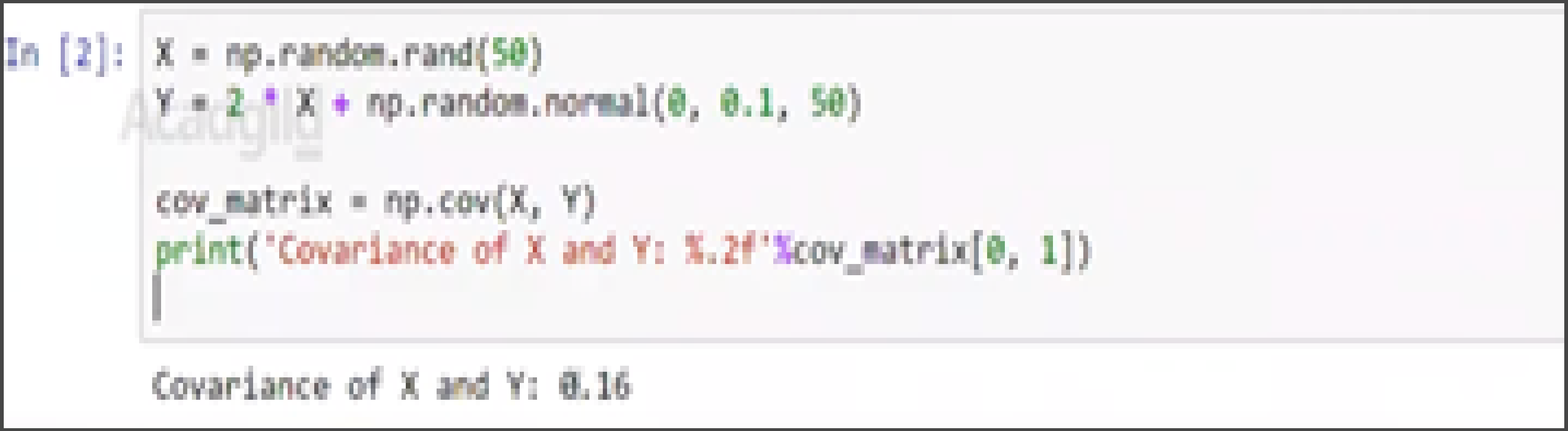
Both can be positive or negative. Covariance is positive if one increases other also increases and negative if one increases other decreases.



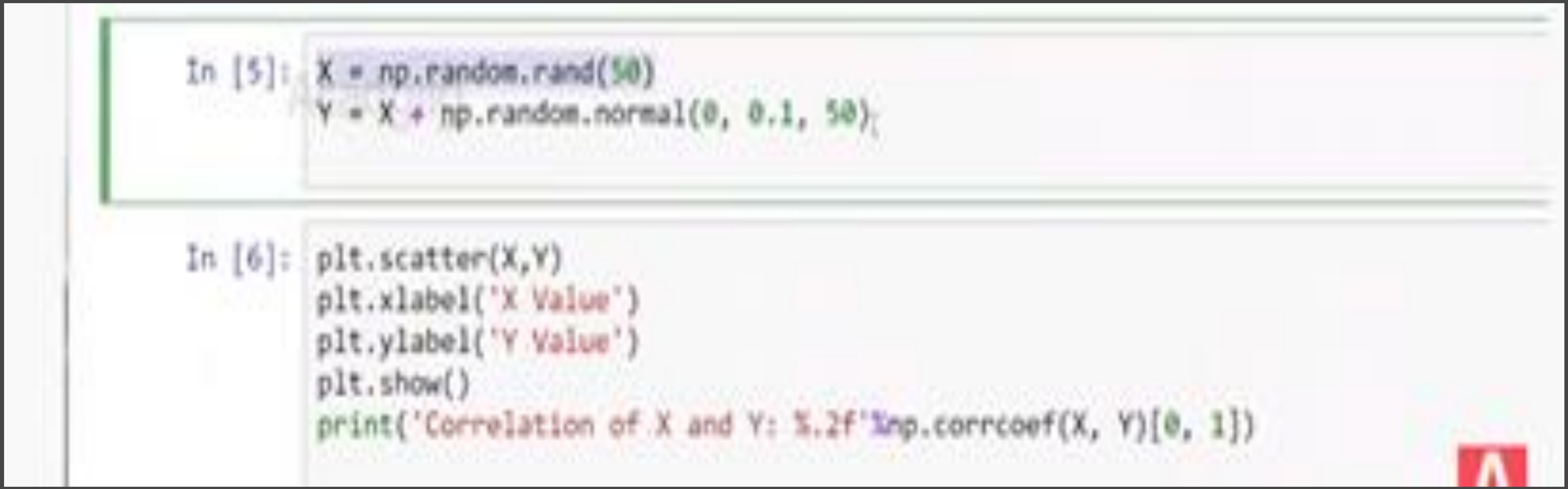


$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_x \sigma_y}$$

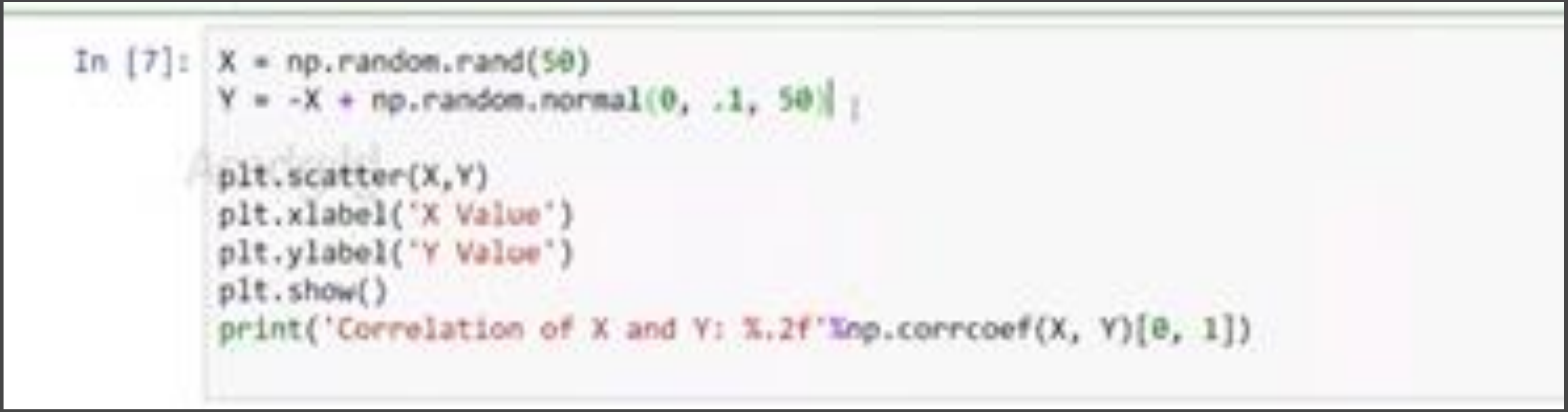
Acadgild
Covarianced normalized by Standard Deviation
Correlation between X and Y
Standard deviation of X
Standard deviation of Y



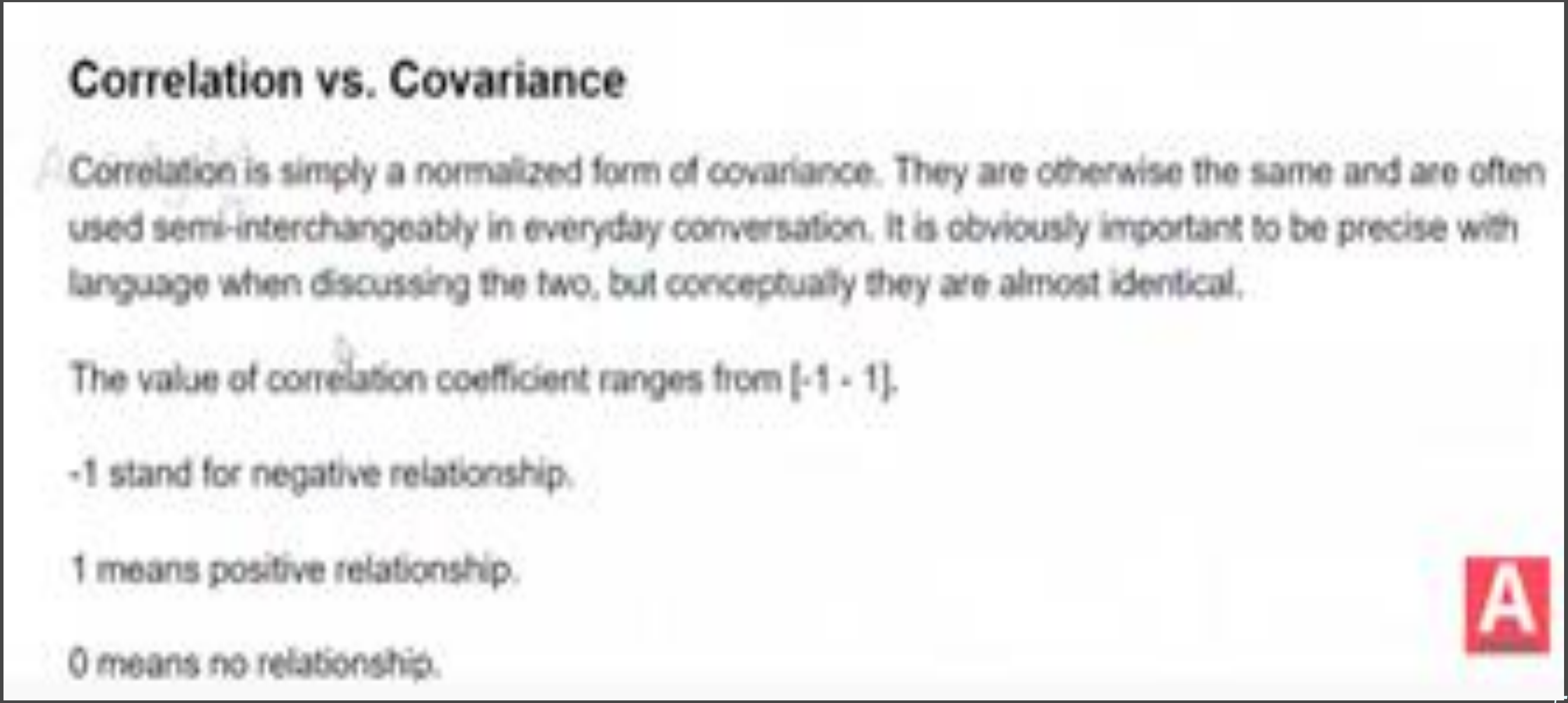
```
In [2]: X = np.random.rand(50)
        Y = 2 * X + np.random.normal(0, 0.1, 50)

        cov_matrix = np.cov(X, Y)
        print('Covariance of X and Y: %.2f'%cov_matrix[0, 1])

Covariance of X and Y: 0.16
```



```
In [5]: X = np.random.rand(50)
        Y = X + np.random.normal(0, 0.1, 50)

In [6]: plt.scatter(X,Y)
        plt.xlabel('X Value')
        plt.ylabel('Y Value')
        plt.show()
        print('Correlation of X and Y: %.2f'%np.corrcoef(X, Y)[0, 1])
```



```
In [7]: X = np.random.rand(50)
        Y = -X + np.random.normal(0, .1, 50)

        plt.scatter(X,Y)
        plt.xlabel('X Value')
        plt.ylabel('Y Value')
        plt.show()
        print('Correlation of X and Y: %.2f'%np.corrcoef(X, Y)[0, 1])
```



Correlation vs. Covariance

Correlation is simply a normalized form of covariance. They are otherwise the same and are often used semi-interchangeably in everyday conversation. It is obviously important to be precise with language when discussing the two, but conceptually they are almost identical.

The value of correlation coefficient ranges from [-1 - 1].

-1 stand for negative relationship.

1 means positive relationship.

0 means no relationship.

Simpson’s Paradox

https://www.youtube.com/watch?v=zj2QV7cfUfQ

One not uncommon surprise when analyzing data is Simpson’s Paradox, in which correlations can be misleading when confounding variables are ignored.

The key issue is that correlation is measuring the relationship between your two variables all else being equal.

If your data classes are assigned at random, as they might be in a well-designed experiment, “all else being equal” might not be a terrible assumption. But when there is a deeper pattern to class assignments, “all else being equal” can be an awful assumption.

The only real way to avoid this is by knowing your data and by doing what you can to make sure you’ve checked for possible confounding factors.

Good links

https://www.youtube.com/watch?v=ebEkn-BiW5k

https://www.youtube.com/watch?v=sxYrzzy3cq8