Lambton College
School of Computer Studies

AML-2103 Visualization for AI and ML

[Prepared By: Vahid Hadavi, PhD]

Lecture1

1. Explain the concept of data visualization

Major Topics:

- 1.1 Define data exploration
- 1.2 Identify different aspects of data visualization
- 1.3 Discuss data Wrangling
- 1.4 Explore Tools and Libraries for visualization using Python

1.1 Define data exploration

Data Exploration

- Data exploration is the initial step in data analysis.
- In this step, users explore a large data set in an unstructured way to uncover initial patterns, characteristics, and points of interest.
- This process is not meant to reveal every bit of information a dataset holds, but rather to help create a broad picture of important trends and major points to study in greater detail.
- data exploration uses visualization because it creates a more straightforward view of data sets than simply examining thousands of individual numbers or names.

1.2 Identify different aspects of data visualization

Data Visualization

- Data visualization is the graphical representation of information and data.
- Visual data is very easy to understand compared to data in any other form.
- Advantages of data visualization:
  - Complex data can be easily understood
  - A simple visual representation of outliers, target audiences, and future markets can be created
  - Storytelling can be done using dashboards and animations
  - Data can be explored through interactive visualizations

Data Visualization

- Most common types:
  - Bar Chart
  - Line Chart
  - Scatterplot
  - Pie Chart
  - Heat Map
  - Histogram
  - Box Plot

1.3 Discuss data Wrangling

Data Wrangling

- Data wrangling is the process of cleaning, structuring and enriching raw data into a desired format for better decision making in less time.
- Data has become more diverse and unstructured, demanding, increased time spent culling, cleaning, and organizing data ahead of broader analysis.
- Data Wrangling steps:
  - ➢ Discovering

In this step, the data is to be understood more deeply. Before implementing methods to clean it, you need to understand what is about.

  - ➢ Structuring

Raw data is given to you in a haphazard manner, in most cases is not any structure to it. This needs to be rectified, and the data needs to be restructured in a manner that better suits the analytical method used.

Data Wrangling

- Cleaning

All datasets have some outliers, which can skew the results of the analysis. Null values will have to be addressed. These will have to be cleaned, for the best results.

- Enriching

After cleaning, it will have to be enriched. This means that you will have to take stock of what is in the data and strategize whether you will have to augment it using some additional data in order to make it better. You should also brainstorm about whether you can derive any new data from the existing clean data set that you have.

Data Wrangling

- Validating

  Validation rules refer to some repetitive programming steps which are used to verify the consistency, quality and the security of the data you have.

- Publishing

  The prepared wrangled data is published so that it can be used further down the line.

1.4 Explore Tools and Libraries for visualization using Python

Data visualization tools

- Non-coding tools:
  - Datawrapper
  - Tableau
  - Plotly

- Coding tools:
  - Python
  - MATLAB
  - R

Data visualization with python

- Python packages:
  - Matplotlib

The first and the most commonly used library for data visualization. It is simple yet very powerful.

  - Seaborn

It is a popular data visualization library that is built on top of Matplotlib. Seaborn's default styles and color palettes are much more sophisticated than Matplotlib.

  - Plotly

Plotly is widely known as an online platform for data visualization, very few people know that it can be accessed from a Python notebook.

Any questions so far?
Any comments?

Lambton College
School of Computer Studies

AML-2103 Visualization for AI and ML

[Prepared By: Vahid Hadavi, PhD]

Lecture2

2. Describe the statistical concepts of AI and ML

Major Topics:

- 2.1 Discuss measures of Central Tendency
- 2.2 Explain measure of Dispersion
- 2.3 Discuss Probability and Probability Distribution
- 2.4 Describe Regression and Correlation

2.1 Discuss measures of Central Tendency

Measures of Central Tendency

- A measure of central tendency is a summary statistic that represents the center point or typical value of a dataset.
- These measures indicate where most values in a distribution fall and are also referred to as the central location of a distribution.
- You can think of it as the tendency of data to cluster around a middle value.
- In statistics, the three most common measures of central tendency:
  - Mean

    The arithmetic average that is computed by summing up all measurements and dividing the sum by the number of observations.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Measures of Central Tendency

- Median

  This is the middle value of the ordered dataset. If there is an even number of observations, the median will be the average of the two middle values. The median is less prone to outliers compared to the mean, where outliers are distinct values in data.

- Mode

  Mode is defined as the most frequent value. There may be more than one mode in cases where multiple values are equally frequent.

2.2 Explain measure of Dispersion

Measure of Dispersion

- A measure of spread, sometimes also called a measure of dispersion, is used to describe the variability in a sample or population.
- It is usually used in conjunction with a measure of central tendency, such as the mean or median, to provide an overall description of a set of data.
- A measure of spread gives us an idea of how well the mean, for example, represents the data.
- If the spread of values in the data set is large, the mean is not as representative of the data as if the spread of data is small.
- Common measures of dispersion:
  - Variance

The variance is the expected value of the squared deviation from the mean. It describes how far a set of numbers is spread out from their mean.

$$Var(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

Measure of Dispersion

- Standard deviation

  It's the square root of the variance.

- Range

  It is the difference between the largest and smallest values in a dataset.

- Interquartile range

  Also called the midspread or middle 50%, it is the difference between the 75th and 25th percentiles, or between the upper and lower quartiles.

2.3 Discuss Probability and Probability Distribution

Probability

- Probability is a measure of the likelihood that an event will occur.
- Probabilities lie between 0 and 1.
- Zero probability implies that something is impossible.
- A probability of 1 means something is certain.
- What does an intermediate probability imply, for example if we say that the probability of rain tomorrow is 0.25?
- Let A denote an event . The probability of that event is usually written P(A) or Pr(A).
- A probability of 0.25 (also expressed as 1/4, or as 25%) implies that we think that it is 3 times as likely not to rain as it is to rain. This is because

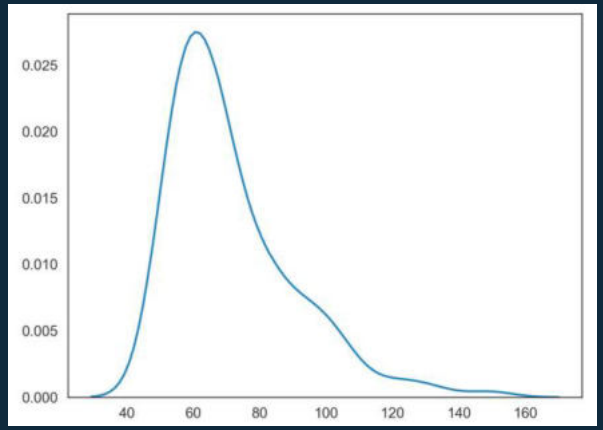
  P(no rain) = 1 - P(rain) = 0.75

  0.75/0.25 = 3

Probability distribution

- A probability distribution is a function that provides the probabilities for every possible event.
- Probability distribution is frequently used for statistical analysis.
- There are two types of probability distribution:
  - Continuous

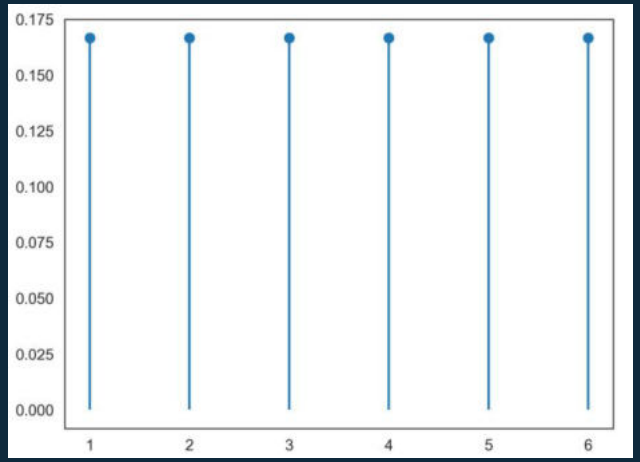
A continuous probability distribution defines the probabilities of each possible value of a continuous random variable.



Probability distribution

- Discrete

A discrete probability distribution shows all the values that a random variable can take, together with their probability.



Data Wrangling

- Cleaning

All datasets have some outliers, which can skew the results of the analysis. Null values will have to be addressed. These will have to be cleaned, for the best results.

- Enriching

After cleaning, it will have to be enriched. This means that you will have to take stock of what is in the data and strategize whether you will have to augment it using some additional data in order to make it better. You should also brainstorm about whether you can derive any new data from the existing clean data set that you have.

Data Wrangling

- Validating

  Validation rules refer to some repetitive programming steps which are used to verify the consistency, quality and the security of the data you have.

- Publishing

  The prepared wrangled data is published so that it can be used further down the line.

1.4 Describe Regression and Correlation

Correlation and Regression

Correlation

- Correlation describes the statistical relationship between two variables.
- In positive correlation, both variables move in the same direction.
- In negative correlation, the variables move in opposite directions.
- In zero correlation, the variables are not related.

Regression

- It determines the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).
- Simple linear regression uses one independent variable to explain or predict the outcome of the dependent variable Y.
- Multiple linear regression uses two or more independent variables to predict the outcome.
- Non-linear regression methods can capture non-linear relationships and be used for more complicated data and analysis.

Data visualization tools

- Non-coding tools:
  - Datawrapper
  - Tableau
  - Plotly

- Coding tools:
  - Python
  - MATLAB
  - R

Data visualization with python

- Python packages:
  - Matplotlib

The first and the most commonly used library for data visualization. It is simple yet very powerful.

  - Seaborn

It is a popular data visualization library that is built on top of Matplotlib. Seaborn's default styles and color palettes are much more sophisticated than Matplotlib.

  - Plotly

Plotly is widely known as an online platform for data visualization, very few people know that it can be accessed from a Python notebook.

Any questions so far?
Any comments?

Lambton College
In Toronto

Lambton College
School of Computer Studies

AML-2103 Visualization for AI and ML

[Prepared By: Vahid Hadavi, PhD]

Lecture3

3. Examine Python data exploration library, NumPy

Major Topics:

- 3.1 Discuss basic NumPy operations
- 3.2 Illustrate loading a simple dataset using Numpy
- 3.3 Compute central tendency (mean, median, mode etc.) using NumPy
- 3.4 Interpret the Dispersion of the dataset (variance, standard deviation etc.)
- 3.5 Apply Indexing, Slicing, Splitting and Iterating using NumPy
- 3.6 Demonstrate advanced NumPy operations: Filtering, Sorting, Combining and Reshaping

3.1 Discuss basic NumPy operations

Creating vectors and matrices

- We need to import numpy.

```
import numpy as np
```

- To create vector, we create one dimensional array.

```
vector_row = np.array([1, 2, 3])
```

- To create matrix, we create two-dimensional array (nested lists)

```
matrix = np.array([[1, 2],
                   [1, 2],
                   [1, 2]])
```

Describing a matrix

- Describing the shape, size, and dimension of the matrix.

```
matrix = np.array([[1, 2, 3, 4],
                   [5, 6, 7, 8],
                   [9, 10, 11, 12]])
```

```
matrix.shape
```

```
(3, 4)
```

```
matrix.size
```

```
12
```

```
matrix.ndim
```

```
2
```

- Dimension of a matrix, vector and scalar is 2,1 and 0, respectively.

Applying Operations to Elements

- Directly manipulating elements.

```
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
```

```
matrix + 100

array([[101, 102, 103],
       [104, 105, 106],
       [107, 108, 109]])
```

- Using vectorize method to apply pre-defined functions.

```
add_100 = lambda i: i + 100
```

```
vectorized_add_100 = np.vectorize(add_100)
```

```
array([[101, 102, 103],
       [104, 105, 106],
       [107, 108, 109]])
```

3.2 Illustrate loading a simple dataset using Numpy

Loading a dataset

- To load a dataset with Numpy, genfromtxt() method is used.
- The file name (path) and the delimiter type must be given.
- For the list of method parameters, click here

```
dataset = np.genfromtxt('./data/normal_distribution.csv', delimiter=',')
```

```
array([[ 99.14931546, 104.03852715, 107.43534677,  97.85230675,
         98.74986914,  98.80833412,  96.81964892,  98.56783189],
       [ 92.02628776,  97.10439252,  99.32066924,  97.24584816,
         92.9267508 ,  92.65657752, 105.7197853 , 101.23162942],
       [ 95.66253664,  95.17750125,  90.93318132, 110.18889465,
         98.80084371, 105.95297652,  98.37481387, 106.54654286],
       [ 91.37294597, 100.96781394, 100.40118279, 113.42090475,
        105.48508838,  91.6604946 , 106.1472841 ,  95.08715803],
       [101.20862522, 103.5730309 , 100.28690912, 105.85269352,
         93.37126331, 108.57980357, 100.79478953,  94.20019732],
       [102.80387079,  98.29687616,  93.24376389,  97.24130034,
         89.03452725,  96.2832753 , 104.60344836, 101.13442416],
       [106.71751618, 102.97585605,  98.45723272, 100.72418901,
        106.39798503,  95.46493436,  94.35373179, 106.83273763],
       [ 96.02548256, 102.82360856, 106.47551845, 101.34745901,
        102.45651798,  98.74767493,  97.57544275,  92.5748759 ],
       [105.30350449,  92.87730812, 103.19258339, 104.40518318,
        101.29326772, 100.85447132, 101.2226037 , 106.03868807],
       [110.44484313,  93.87155456, 101.5363647 ,  97.65393524,
         92.75048583, 101.72074646,  96.96851209, 103.29147111],
       [101.3514185 , 100.37372248, 106.6471081 , 100.61742813,
        105.0320535 ,  99.35999981,  98.87007532,  95.85284217],
       [ 97.21315663. 107.02874163. 102.17642112.  96.74630281,
```

3.3 Compute central tendency (mean, median, mode etc.) using NumPy

Mean, median and mode

- To calculate mean, median and mode of a Numpy array, mean(), median() and mode() methods are respectively used.
- The axis parameter should be set.
- Default value for axis is “None” that calculates the values for the flattened array.
- If we want to have the result for each row, we need to choose axis=1
- If we want to have the result for each column, we need to choose axis=0.

```
np.mean(dataset, axis=1)
```

```
array([100.17764752,  97.27899259, 100.20466135, 100.56785907,
       100.98341406,  97.83018578, 101.49052285,  99.75332252,
       101.89845125,  99.77973914, 101.013081  , 100.54961696,
        98.48256886,  98.49816126, 101.85956927,  97.05201872,
       102.62147483, 101.21177037,  99.58777968,  98.96533534,
       103.85792812, 101.89050288,  99.07192574,  99.34233101])
```

Mean, median, mode and std

```
np.mean(dataset, axis=0)
```

```
array([ 99.7674351 ,  99.61229127, 101.14584656, 101.8449316 ,
        99.04871791,  99.67838931,  99.7848489 , 100.44049274])
```

3.4 Interpret the Dispersion of the dataset (variance, standard deviation etc.)

Variance and standard deviation

- To calculate variance and standard deviation of a Numpy array, var() and std() methods are respectively used.
- The axis parameter should be set.
- Default value for axis is “None” that calculates the values for the flattened array.
- If we want to have the result for each row, we need to choose axis=1
- If we want to have the result for each column, we need to choose axis=0.

```
>>> a = np.array([[1, 2], [3, 4]])
>>> np.var(a)
1.25
>>> np.var(a, axis=0)
array([1.,  1.])
>>> np.var(a, axis=1)
array([0.25,  0.25])
```

3.5 Apply Indexing, Slicing, Splitting and Iterating using NumPy

Indexing and Slicing

- Indexing elements in a NumPy array works the same as with built-in Python.

```
dataset[0] # index single element in outermost dimension
dataset[-1] # index in reversed order in outermost dimension
dataset[1, 1] # index single element in two-dimensional data
dataset[-1, -1] # index in reversed order in two-dimensional data
```

- Slicing has also been adapted from Python's Lists.

```
dataset[1:3] # rows 1 and 2
dataset[:2, :2] # 2x2 subset of the data
dataset[-1, ::-1] # last row with elements reversed
dataset[-5:-1, :6:2] # last 4 rows, every other element up to index 6
```

Splitting and Iterating

- We can split an array into multiple sub-arrays.
- There are two ways of splitting your data, horizontally and vertically.

```
np.hsplit(dataset, (3)) # split horizontally in 3 equal lists
np.vsplit(dataset, (2)) # split vertically in 2 equal lists
```

- We can use nditer() method to step over the whole list of data, in an array, one after another, visiting every single element once.

```
for x in np.nditer(dataset):
print(x)
```

3.6 Demonstrate advanced NumPy operations: Filtering, Sorting, Combining and Reshaping

Filtering and Sorting

- Filtering can be used to clean up your data if you want to avoid outlier values.
  - We can use extract() or the shorthand notation

```
np.extract((dataset < 3), dataset)
```

```
dataset[dataset > 10]
```

```
dataset[(dataset > 5) & (dataset < 10)]
```

  - We can have access to the list of indices instead of values with where()

```
np.where(dataset > 5)
```

- We can sort rows (axis=0) or columns (axis=1).

```
np.sort(dataset, axis=0)
```

Combining and Reshaping

- We can stack rows and columns of two datasets with a same dimension.

```
np.vstack([dataset_1, dataset_2]) # combine datasets vertically
np.hstack([dataset_1, dataset_2]) # combine datasets horizontally
np.stack([dataset_1, dataset_2], axis=0) # combine datasets on axis 0
```

- We can reshape an array using reshape().
- Size of the array before and after reshaping must match

```
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9],
                   [10, 11, 12]])
```

```
matrix.reshape(2, 6)

array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12]])
```

One useful argument in reshape is -1, which effectively means “as many as needed,”.

```
matrix.reshape(1, -1)

array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]])
```

Any questions so far?
Any comments?

Lambton College | In Toronto

Lambton College

School of Computer Studies

AML-2103 Visualization for AI and ML

[Prepared By: Vahid Hadavi, PhD]

Lecture5

5. Evaluate Pandas

Major Topics:

- 5.1 Discuss Pandas
- 5.2 Operations with Pandas
- 5.3 Pandas vs. Numpy: Advantages and Disadvantages

Pandas basic operations

- We need to import pandas.
- We can load different types of files into a dataframe or series using pandas.

```
# Load library
import pandas as pd

# Create URL
url = 'https://tinyurl.com/titanic-csv'

# Load data
dataframe = pd.read_csv(url)
```

- The file name (path) is necessary. For more information about the parameters please refer to pandas documentation.
- We can use head() or tail() to check the first or last 5 (default) rows of the data frame.

```
dataframe.head(2)
```

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

Pandas basic operations

- Using .shape to check the dimensions of the dataframe.

```
dataframe.shape

(1313, 6)
```

- Using describe() to get descriptive statistics for any numeric columns.

	Age	Survived	SexCode
count	756.000000	1313.000000	1313.000000
mean	30.397989	0.342727	0.351866
std	14.259049	0.474802	0.477734
min	0.170000	0.000000	0.000000
25%	21.000000	0.000000	0.000000
50%	28.000000	0.000000	0.000000
75%	39.000000	1.000000	1.000000
max	71.000000	1.000000	1.000000

4.3 Apply data cleaning

Manipulating Data

- Using replace() to replace values.

```
dataframe['Sex'].replace(["female", "male"], ["Woman", "Man"]).head(5)

0    Woman
1    Woman
2      Man
3    Woman
4      Man
```

```
dataframe.replace(1, "One").head(2)
```

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29	female	One	One
1	Allison, Miss Helen Loraine	1st	2	female	0	One

Manipulating Data

- Using isnull() to find missing values.

```
dataframe[dataframe['Age'].isnull()].head(2)
```

	Name	PClass	Age	Sex	Survived	SexCode
12	Aubert, Mrs Leontine Pauline	1st	NaN	female	1	1
13	Barkworth, Mr Algernon H	1st	NaN	male	1	0

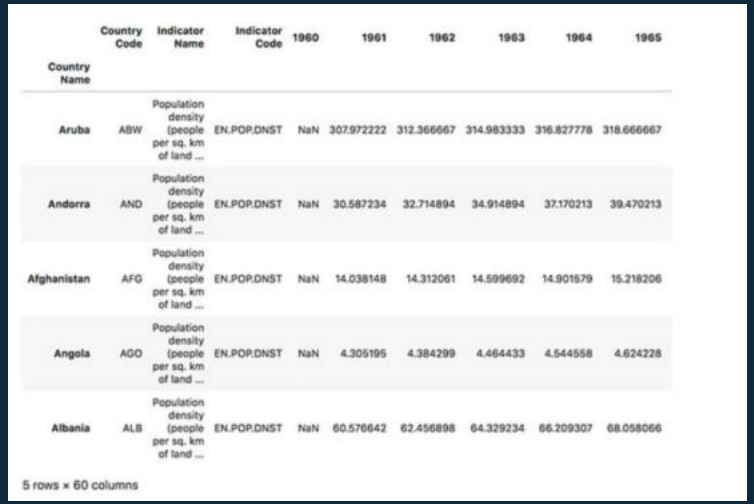
- Using replace() to replace missing values. Numpy must be imported.

```
dataframe['Sex'] = dataframe['Sex'].replace('male', np.nan)
```

4.4 Compute central tendency and dispersion using pandas

central tendency and dispersion

```
dataset = pd.read_csv('./data/world_population.csv', index_col=0)
```



Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965
Aruba	ABW	Population density (people per sq. km of land ...	EN.POP.DNST	NaN	307.972222	312.366667	314.983333	316.827778	318.666667
Andorra	AND	Population density (people per sq. km of land ...	EN.POP.DNST	NaN	30.587234	32.714894	34.914894	37.170213	39.470213
Afghanistan	AFG	Population density (people per sq. km of land ...	EN.POP.DNST	NaN	14.038148	14.312061	14.599692	14.901579	15.218206
Angola	AGO	Population density (people per sq. km of land ...	EN.POP.DNST	NaN	4.305195	4.384299	4.464433	4.544558	4.624228
Albania	ALB	Population density (people per sq. km of land ...	EN.POP.DNST	NaN	60.576642	62.456898	64.329234	66.209307	68.058066

5 rows × 60 columns

```
# calculating the mean for 1961 column
dataset["1961"].mean()
```

```
176.91514132840538
```

central tendency and dispersion

- Using axis parameter we can have the values per columns (axis=0) and per rows (axis=1).
- By default axis=0 is used.

```
# mean for each country (row)
dataset.mean(axis=1).head(10)
```

```
Country Name
Aruba                  413.944949
Andorra                106.838839
Afghanistan             25.373379
Angola                   9.649583
Albania                 99.159197
Arab World              16.118586
United Arab Emirates    31.321721
Argentina               11.634028
Armenia                103.415539
American Samoa         211.855636
dtype: float64
```

4.5 Apply Indexing, Slicing and Iterating using pandas

Indexing

- We can access columns with the single bracket.

```
dataset["2000"] # index the 2000 col
```

- We can access rows with index numbers, using iloc[]

```
dataset.iloc[-1] # index the last row
```

- We can access rows with index names, using loc[].

```
dataset.loc["Germany"] # index the row with index Germany
```

```
dataset[["2015"]].loc[["Germany"]] # index row Germany and column 2015
```

Slicing and Iterating

- We can slice dataframes using loc[] and iloc[].

```
dataset.iloc[0:10] # slice of the first 10 rows
```

```
dataset.loc[["Germany", "India"]] # slice of rows Germany and India
```

```
# subset of Germany and India with years 1970/90
dataset.loc[["Germany", "India"]][["1970", "1990"]]
```

- We can use iterrows() to iterate over rows and columns in pandas.

```
# iterating the whole dataset
for index, row in dataset.iterrows():
print(index, row)
```

4.2 Explain advantages and drawbacks of pandas over NumPy

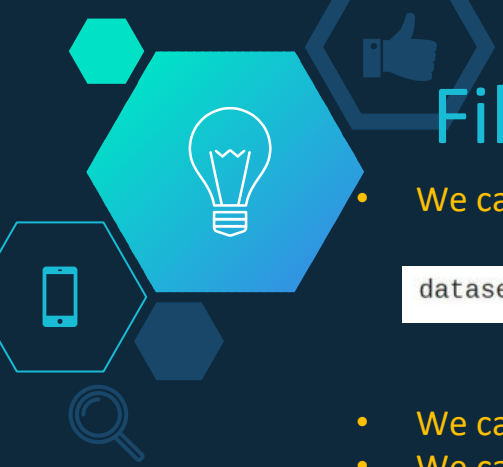
Advantages and Disadvantages

Advantages

- High level of abstraction
- Less intuition
- Faster processing
- Easy DataFrames design

Disadvantages

- Less applicable
- More disk space
- Performance problems
- Hidden complexity



Filtering and Sorting

- We can use the brackets-based conditional filtering.

```
dataset[(dataset["1990"] < 10)] # countries'population density < 10 in 1999
```

- We can sort a dataframe using sort_values().
- We can use “by” parameter to specift the column or columns we want to sort based on.

```
dataset.sort_values(by=["1999"]) # values sorted by 1999
```

- We can sort in descending order by setting “ascending” on False.

```
# values sorted by 1999 descending
dataset.sort_values(by=["1994"], ascending=False)
```

Reshaping

- We can use pivot() to reshape a dataframe.

```
dataset.pivot(index=["1999"] * len(dataset), columns="Country Code", values="1999")
```

- Reshaping is a very complex topic. If you want to dive deeper into it, refer to pandas documentation.
  https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.pivot_table.html

4.6 Demonstrate advanced pandas operations: Filtering, Sorting and Reshaping

Any questions so far?
Any comments?