Data Analytics (PETR 6397)

Coding Environment and Data Wrangling

Dr. Ahmad Sakhaee-Pour

Petroleum Engineering Department

University of Houston

Spring 2025

Visualization is an essential part of data analytics

There are various libraries in Python

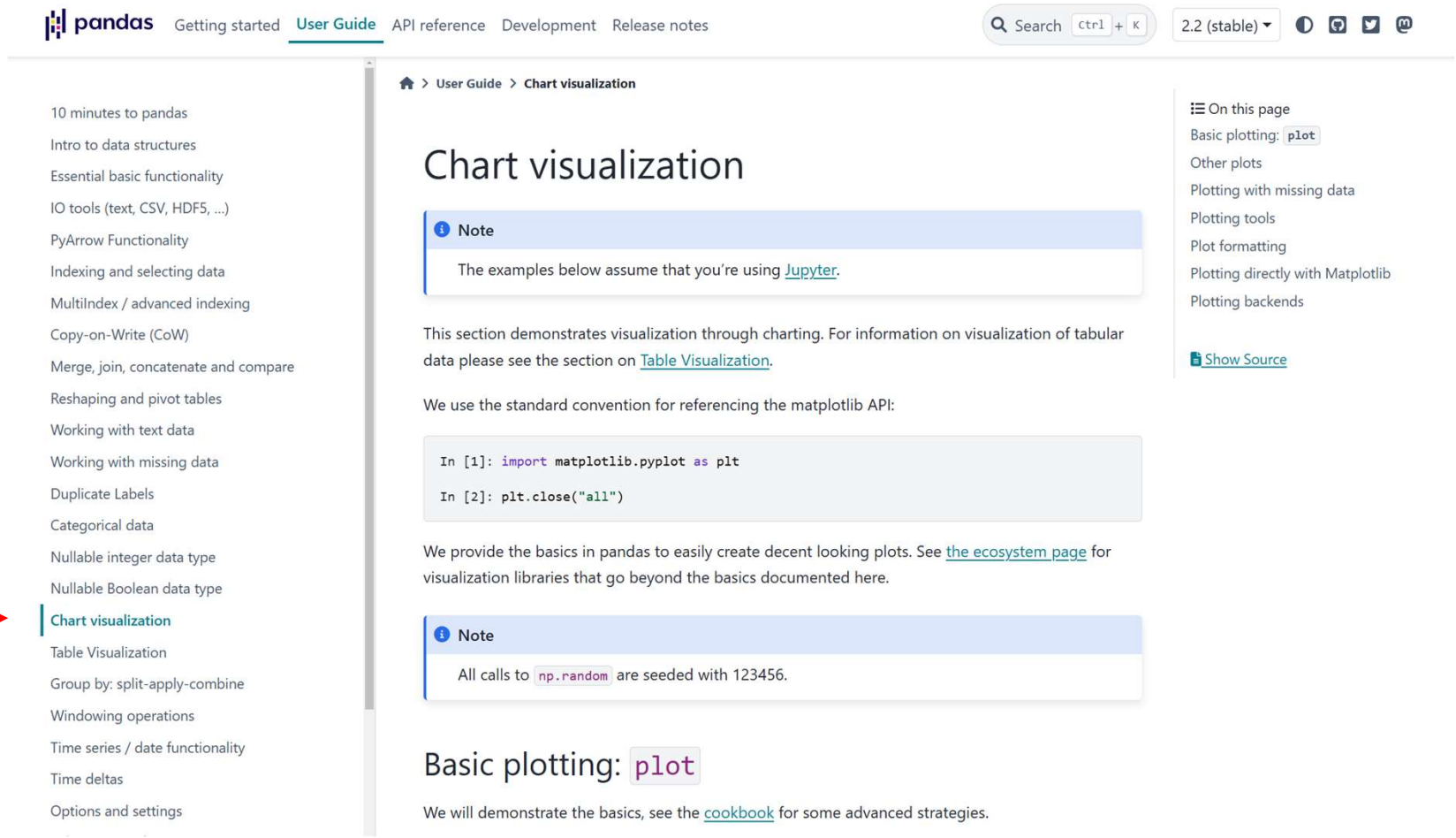
1. Pandas provides basic visualizations

2. Matplotlib has a visualization environment similar to MATLAB

3. Seaborn was built on Matplotlib, and it is more advanced

…

Pandas for visualization



Pandas for visualization (continued)

- Search for histogram and cross plot

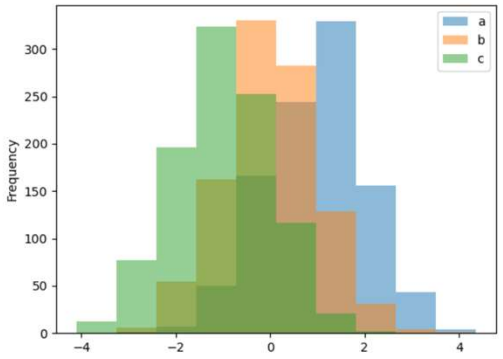
Histograms

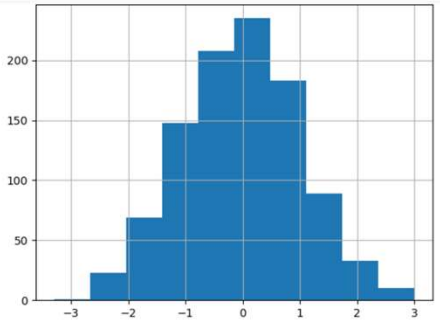
Histograms can be drawn by using the `DataFrame.plot.hist()` and `Series.plot.hist()` methods.

```
In [25]: df4 = pd.DataFrame(
   ....:     {
   ....:         "a": np.random.randn(1000) + 1,
   ....:         "b": np.random.randn(1000),
   ....:         "c": np.random.randn(1000) - 1,
   ....:     },
   ....:     columns=["a", "b", "c"],
   ....: )
   ....:

In [26]: plt.figure();

In [27]: df4.plot.hist(alpha=0.5);
```

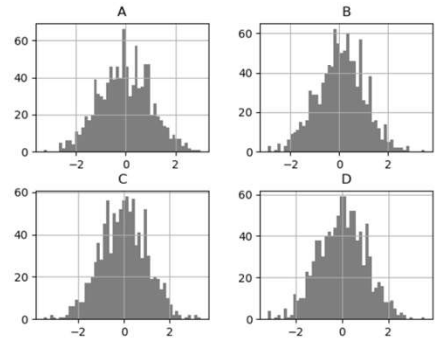




`DataFrame.hist()` plots the histograms of the columns on multiple subplots:

```
In [34]: plt.figure();

In [35]: df.diff().hist(color="k", alpha=0.5, bins=50);
```



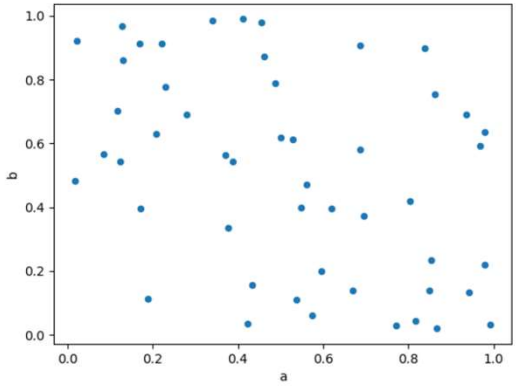
Scatter plot

Scatter plot can be drawn by using the `DataFrame.plot.scatter()` method. Scatter plot requires numeric columns for the x and y axes. These can be specified by the `x` and `y` keywords.

```
In [70]: df = pd.DataFrame(np.random.rand(50, 4), columns=["a", "b", "c", "d"])

In [71]: df["species"] = pd.Categorical(
   ....:     ["setosa"] * 20 + ["versicolor"] * 20 + ["virginica"] * 10
   ....: )
   ....:

In [72]: df.plot.scatter(x="a", y="b");
```



To plot multiple column groups in a single axes, repeat `plot` method specifying target `ax`. It is recommended to specify `color` and `label` keywords to distinguish each groups.

```
In [73]: ax = df.plot.scatter(x="a", y="b", color="DarkBlue", label="Group 1")

In [74]: df.plot.scatter(x="c", y="d", color="DarkGreen", label="Group 2", ax=ax);
```

Matplotlib for visualization

matplotlib

Plot types User guide Tutorials Examples Reference Contribute Releases 3.10 (stable)

Matplotlib cheatsheets

Support Matplotlib

Matplotlib 3.10.0 documentation

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

On this page
Install
Learn
Community
What's new
Contribute
About us

Install

pip conda other

```
pip install matplotlib
```

For more detailed instructions, see the installation guide.

Learn

How to use Matplotlib?

Quick start guide
User guide
Tutorials
Frequently Asked Questions

What can Matplotlib do?

Plot types
Examples

Check these

Matplotlib for visualization (continued)

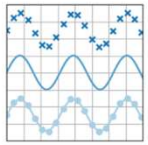
Plot types

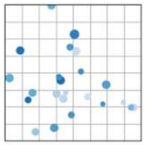
Overview of many common plotting commands provided by Matplotlib.

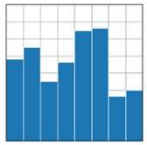
See the gallery for more examples and the tutorials page for longer examples.

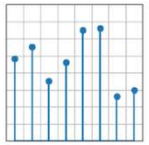
Pairwise data

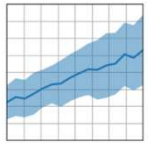
Plots of pairwise (x, y), tabular $(var\_0, \cdots, var\_n)$, and functional $f(x) = y$ data.


plot(x, y)

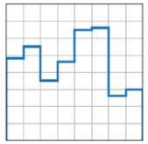

scatter(x, y)


bar(x, height)


stem(x, y)

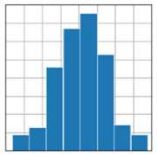

fill_between(x, y1, y2)

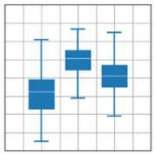

stackplot(x, y)

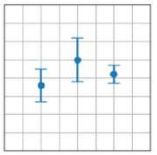

stairs(values)

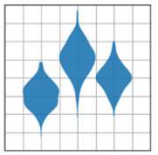
Statistical distributions

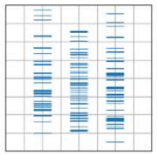
Plots of the distribution of at least one variable in a dataset. Some of these methods also compute the distributions.


hist(x)

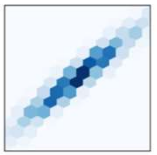

boxplot(X)


errorbar(x, y, yerr, xerr)

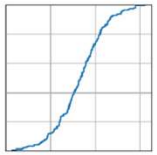

violinplot(D)


eventplot(D)

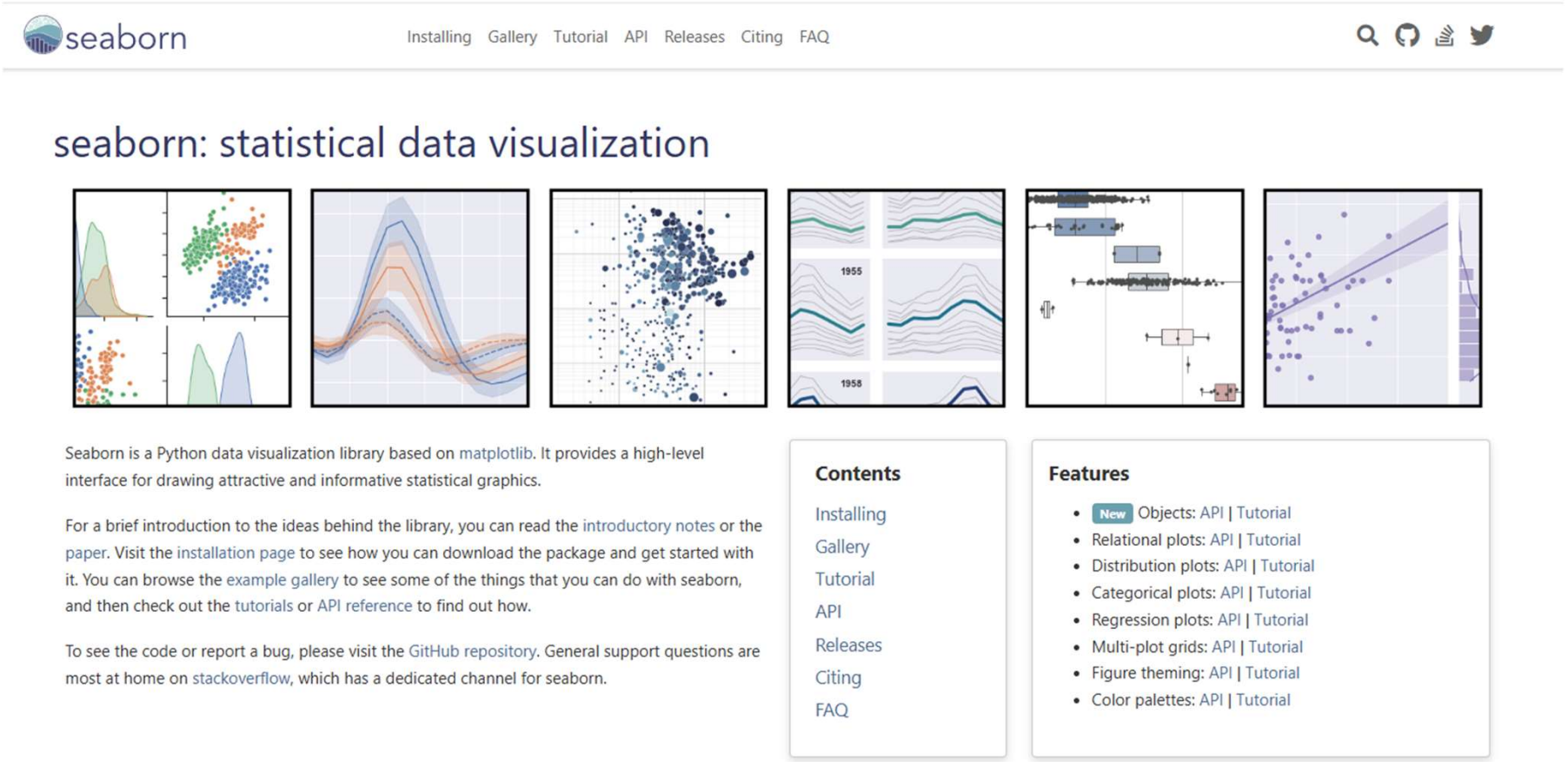

hist2d(x, y)


hexbin(x, y, C)


pie(x)


ecdf(x)

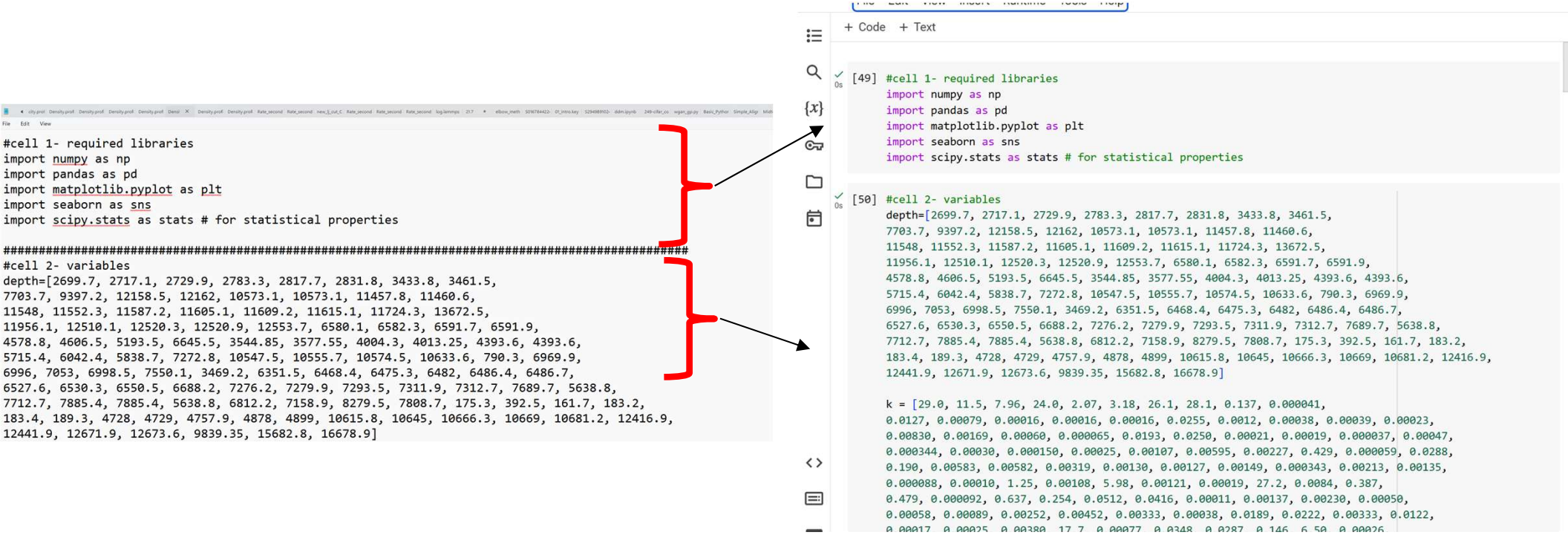
Seaborn is more advanced and equipped better for statistical analysis



First code in Google Colab

First visualization code

- Access vis1 on Canvas
- Divide and paste into cells
- Run the cells
- How would you interpret the data?
- How to improve the quality of the plots?



```
#cell 1- required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats # for statistical properties

##############################################################################################
#cell 2- variables
depth=[2699.7, 2717.1, 2729.9, 2783.3, 2817.7, 2831.8, 3433.8, 3461.5,
7703.7, 9397.2, 12158.5, 12162, 10573.1, 10573.1, 11457.8, 11460.6,
11548, 11552.3, 11587.2, 11605.1, 11609.2, 11615.1, 11724.3, 13672.5,
11956.1, 12510.1, 12520.3, 12520.9, 12553.7, 6580.1, 6582.3, 6591.7, 6591.9,
4578.8, 4606.5, 5193.5, 6645.5, 3544.85, 3577.55, 4004.3, 4013.25, 4393.6, 4393.6,
5715.4, 6042.4, 5838.7, 7272.8, 10547.5, 10555.7, 10574.5, 10633.6, 790.3, 6969.9,
6996, 7053, 6998.5, 7550.1, 3469.2, 6351.5, 6468.4, 6475.3, 6482, 6486.4, 6486.7,
6527.6, 6530.3, 6550.5, 6688.2, 7276.2, 7279.9, 7293.5, 7311.9, 7312.7, 7689.7, 5638.8,
7712.7, 7885.4, 7885.4, 5638.8, 6812.2, 7158.9, 8279.5, 7808.7, 175.3, 392.5, 161.7, 183.2,
183.4, 189.3, 4728, 4729, 4757.9, 4878, 4899, 10615.8, 10645, 10666.3, 10669, 10681.2, 12416.9,
12441.9, 12671.9, 12673.6, 9839.35, 15682.8, 16678.9]
```

```
+ Code  + Text

[49] #cell 1- required libraries
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import scipy.stats as stats # for statistical properties

[50] #cell 2- variables
     depth=[2699.7, 2717.1, 2729.9, 2783.3, 2817.7, 2831.8, 3433.8, 3461.5,
     7703.7, 9397.2, 12158.5, 12162, 10573.1, 10573.1, 11457.8, 11460.6,
     11548, 11552.3, 11587.2, 11605.1, 11609.2, 11615.1, 11724.3, 13672.5,
     11956.1, 12510.1, 12520.3, 12520.9, 12553.7, 6580.1, 6582.3, 6591.7, 6591.9,
     4578.8, 4606.5, 5193.5, 6645.5, 3544.85, 3577.55, 4004.3, 4013.25, 4393.6, 4393.6,
     5715.4, 6042.4, 5838.7, 7272.8, 10547.5, 10555.7, 10574.5, 10633.6, 790.3, 6969.9,
     6996, 7053, 6998.5, 7550.1, 3469.2, 6351.5, 6468.4, 6475.3, 6482, 6486.4, 6486.7,
     6527.6, 6530.3, 6550.5, 6688.2, 7276.2, 7279.9, 7293.5, 7311.9, 7312.7, 7689.7, 5638.8,
     7712.7, 7885.4, 7885.4, 5638.8, 6812.2, 7158.9, 8279.5, 7808.7, 175.3, 392.5, 161.7, 183.2,
     183.4, 189.3, 4728, 4729, 4757.9, 4878, 4899, 10615.8, 10645, 10666.3, 10669, 10681.2, 12416.9,
     12441.9, 12671.9, 12673.6, 9839.35, 15682.8, 16678.9]

     k = [29.0, 11.5, 7.96, 24.0, 2.07, 3.18, 26.1, 28.1, 0.137, 0.000041,
     0.0127, 0.00079, 0.00016, 0.00016, 0.00016, 0.0255, 0.0012, 0.00038, 0.00039, 0.00023,
     0.00830, 0.00169, 0.00060, 0.000065, 0.0193, 0.0250, 0.00021, 0.00019, 0.000037, 0.00047,
     0.000344, 0.00030, 0.000150, 0.00025, 0.00107, 0.00595, 0.00227, 0.429, 0.000059, 0.0288,
     0.190, 0.00583, 0.00582, 0.00319, 0.00130, 0.00127, 0.00149, 0.000343, 0.00213, 0.00135,
     0.000088, 0.00010, 1.25, 0.00108, 5.98, 0.00121, 0.00019, 27.2, 0.0084, 0.387,
     0.479, 0.000092, 0.637, 0.254, 0.0512, 0.0416, 0.00011, 0.00137, 0.00230, 0.00050,
     0.00058, 0.00089, 0.00252, 0.00452, 0.00333, 0.00038, 0.0189, 0.0222, 0.00333, 0.0122,
```

Basic statistical properties

- Always report these properties when you analyze variables (this is the last cell of the code)
- Min
- Max
- Mean
- Mode
- Median
- Standard deviation

```
#cell 9

# Calculate and print statistics for x1
print(f"Variable x1:")
print(f"  Minimum: {np.min(x1)}")
print(f"  Maximum: {np.max(x1)}")
print(f"  Mean: {np.mean(x1)}")
print(f"  Median: {np.median(x1)}")
print(f"  Mode: {stats.mode(x1)}")

# Similarly, calculate and print for x2 and x3
print(f"\nVariable x2:") # \n add a new line for better readability
print(f"  Minimum: {np.min(x2)}")
print(f"  Maximum: {np.max(x2)}")
print(f"  Mean: {np.mean(x2)}")
print(f"  Median: {np.median(x2)}")
print(f"  Mode: {stats.mode(x2)}")

print(f"\nVariable x3:")
print(f"  Minimum: {np.min(x3)}")
print(f"  Maximum: {np.max(x3)}")
print(f"  Mean: {np.mean(x3)}")
print(f"  Median: {np.median(x3)}")
print(f"  Mode: {stats.mode(x3)}")
```

```
Variable x1:
  Minimum: 161.7
  Maximum: 16678.9
  Mean: 7445.583018867925
  Median: 6997.25
  Mode: ModeResult(mode=4393.6, count=2)

Variable x2:
  Minimum: 3.7e-05
  Maximum: 30.7
  Mean: 2.386303575471698
  Median: 0.002825
  Mode: ModeResult(mode=0.00016, count=3)

Variable x3:
  Minimum: 1.3
  Maximum: 23.8
  Mean: 9.07264150943396
  Median: 7.8
  Mode: ModeResult(mode=10.2, count=5)
```

Second code

Second visualization code

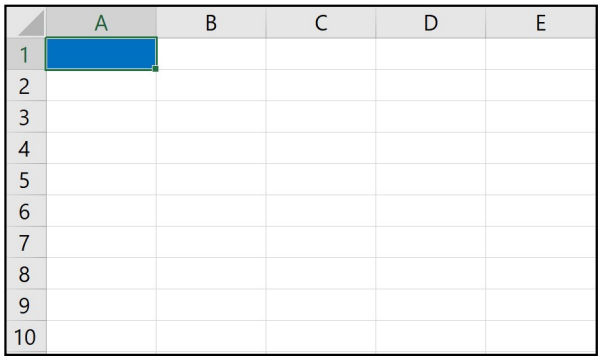
- Follow the same procedure using vis2
- What is the difference between the two codes?
- What is your conclusion?

Data structures in Python

- Data structure in Python is different from MATLAB
- We have to distinguish three specific structures (Python vs. NumPy vs. Pandas)

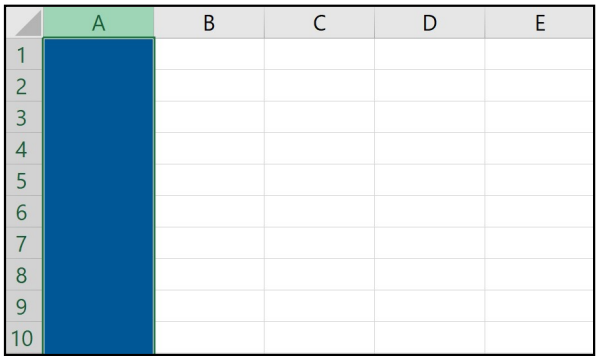
Data structures in Python and Excel

Scalar



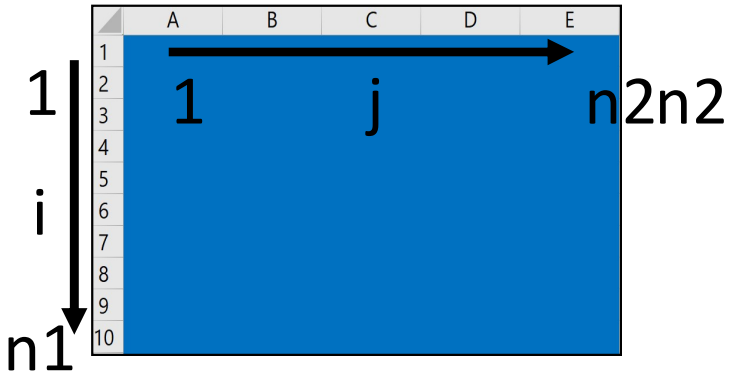
Integer
Float
String
Boolean (True or False)

Vector



Simple list

Array



It has an arbitrary number of dimensions (here is 2)

It has different sizes (n1, n2)

NumPy

- Matrix is referred to as an array in NumPy
- Array was designed specifically for large computations
- Array is ideal for linear algebra
- Array is similar to the matrix format and suitable for common operations in engineering

5	7	1	-2	9

1d array
shape: (5,)

5	7	-1	4
5	8	1	11

2d array
shape: (2,4)

ndarray format

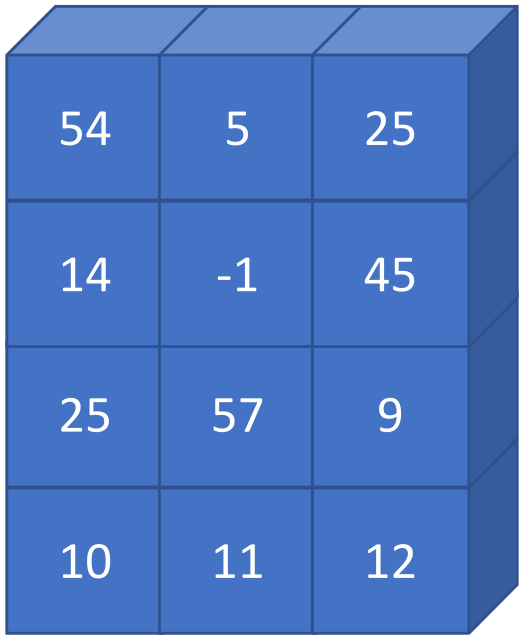
- Row and column numbering starts from 0
- ndarray can have m dimensions: A[n1,n2,n3,n4,…, nm]

A[0,0], A[0,1], A[0,2]

A[1,0], A[1,1], A[1,2]

A[2,0], A[2,1], A[2,2]

A[3,0], A[3,1], A[3,2]

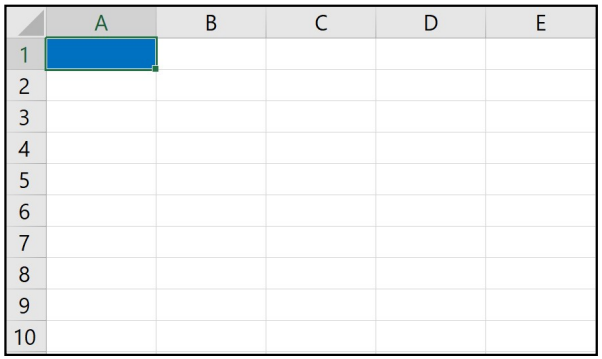


- Negative index counts in the reverse order:

A[0,2]= 25 A[7,5]= A[-1,-1] = A[-2,1] =

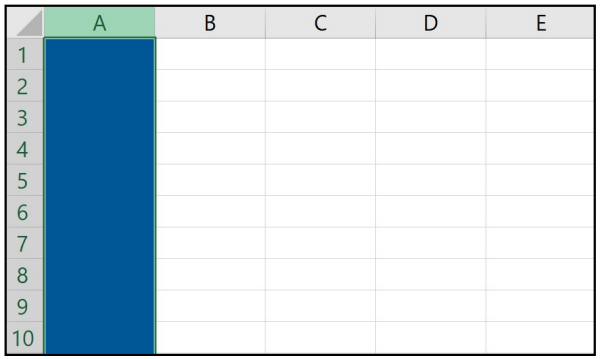
Data structures in NumPy and Excel

Scalar



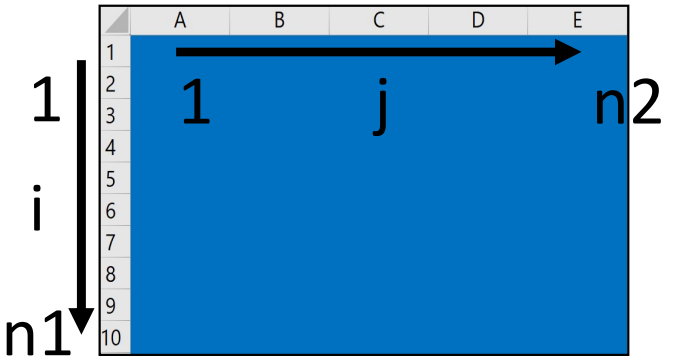
Integer
Float
String
Boolean (True/False)

Vector



Numpy vectors -
List -> np
Element wise
multiplication.

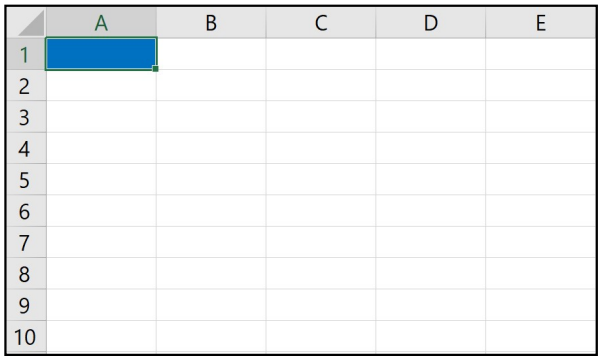
Array



We read them from CSV
We display them using
“imshow” command

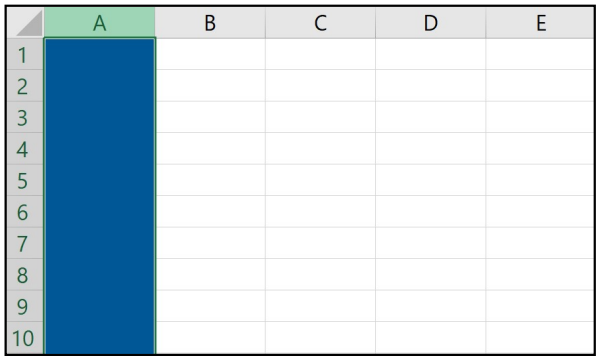
Current status quo in terms of data structures

Scalar



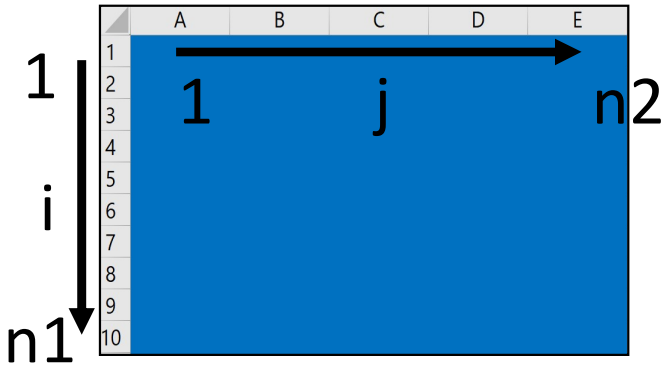
Integer
Float
String
Boolean (True or False)

Vector



Simple lists
Numpy vectors

Array



It has an arbitrary number of dimensions (here is 2)

It has different sizes (n1, n2)

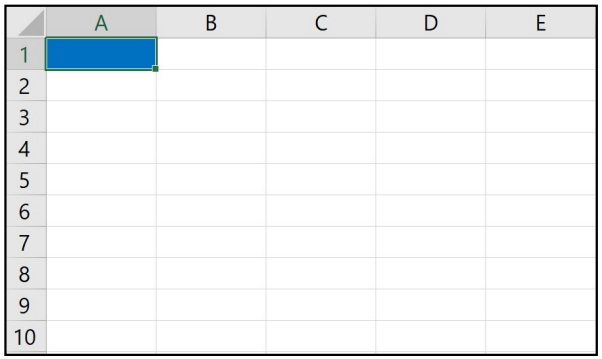
Pandas is a library that handles data more easily

- Mckinney developed the pandas on top of numpy
- It is great at filtering, reformatting, visualizing, and wrangling tabular data
- Its DataFrame (next slide) is the most common approach of dealing with tabular data
- Rows are “observations” and columns are “variables”.

Well_Name	DEPT	DPHZ	DT	GR_EDT	HCAL	HDRA	NPHI	PEFZ	RHOZ	RLA3	RLA4	RLA5
MIP_4H	122.5	0.4168	59.2228	50.0387	3.9599	0.2233	0.8214	5.6412	2.0006	0.0001	100000	0.0024
MIP_4H	123	0.4168	59.2228	51.8071	3.9599	0.2233	0.7914	5.6412	2.0006	0.0001	100000	0.0024
MIP_4H	123.5	0.4168	59.2228	50.3461	3.9656	0.2233	0.7087	5.6412	2.0006	0.0001	100000	0.0024
MIP_4H	124	0.4168	59.2228	48.3869	3.9543	0.2233	0.6884	5.6412	2.0006	0.0006	100000	0.0026
MIP_4H	124.5	0.4168	58.3158	48.663	3.9543	0.2233	0.7115	5.6412	2.0006	0.0001	100000	0.0019
MIP_4H	125	0.4168	59.0299	49.6893	3.9599	0.2233	0.7245	5.6412	2.0006	0.0004	100000	0.0027
MIP_4H	125.5	0.4168	58.5447	51.198	3.9599	0.2233	0.7274	5.6412	2.0006	100000	100000	0.0019
MIP_4H	126	0.4168	58.5314	49.679	3.9543	0.2233	0.7347	5.6412	2.0006	0.0011	100000	0.0025
MIP_4H	126.5	0.4168	58.1528	55.6172	3.9543	0.2233	0.7681	5.6412	2.0006	100000	100000	0.0016
MIP_4H	127	0.4168	57.4968	57.4487	3.9599	0.2233	0.8471	5.6412	2.0006	0.0002	100000	0.0021
MIP_4H	127.5	0.4168	56.8835	58.9678	3.9543	0.2233	0.8365	5.6412	2.0006	0.0012	100000	0.003
MIP_4H	128	0.4168	52.0731	56.3679	3.9656	0.2233	0.8532	5.6412	2.0006	0.002	100000	0.0037
MIP_4H	128.5	0.4166	58.5608	54.5506	3.9599	0.2248	0.807	5.6488	2.0009	100000	100000	0.0021
MIP_4H	129	0.4121	57.672	55.8407	3.9599	0.2295	0.7968	5.6588	2.0082	100000	100000	0.0023
MIP_4H	129.5	0.4074	57.856	52.6294	3.9599	0.233	0.7641	5.6624	2.0159	0.0021	100000	0.0041
MIP_4H	130	0.4031	57.6578	52.6515	3.9599	0.235	0.708	5.6554	2.023	0.0014	100000	0.0038

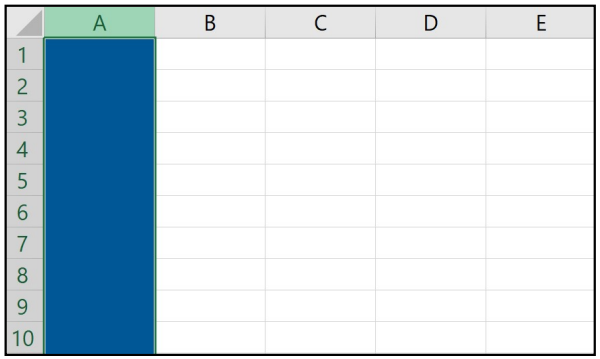
Series and DataFrames of pandas are convenient tools in data handling

Scalar



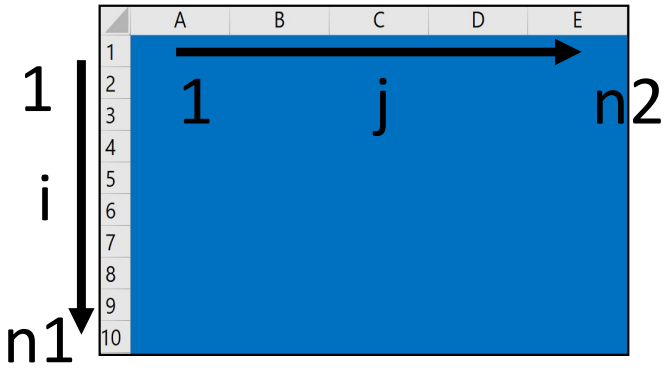
Integer
Float
String
Boolean (True/False)

Vector



Similar to numpy vector

Array



Similar to a bunch of numpy vectors where we specify the name of the column or “j”

Data wrangling definition and importance

- Data wrangling is cleaning and preprocessing data to prepare them for our task
- We have only one lecture on this topic but in practice the majority of time is invested in this area
- Remember that data and its good quality are the most important elements of machine learning

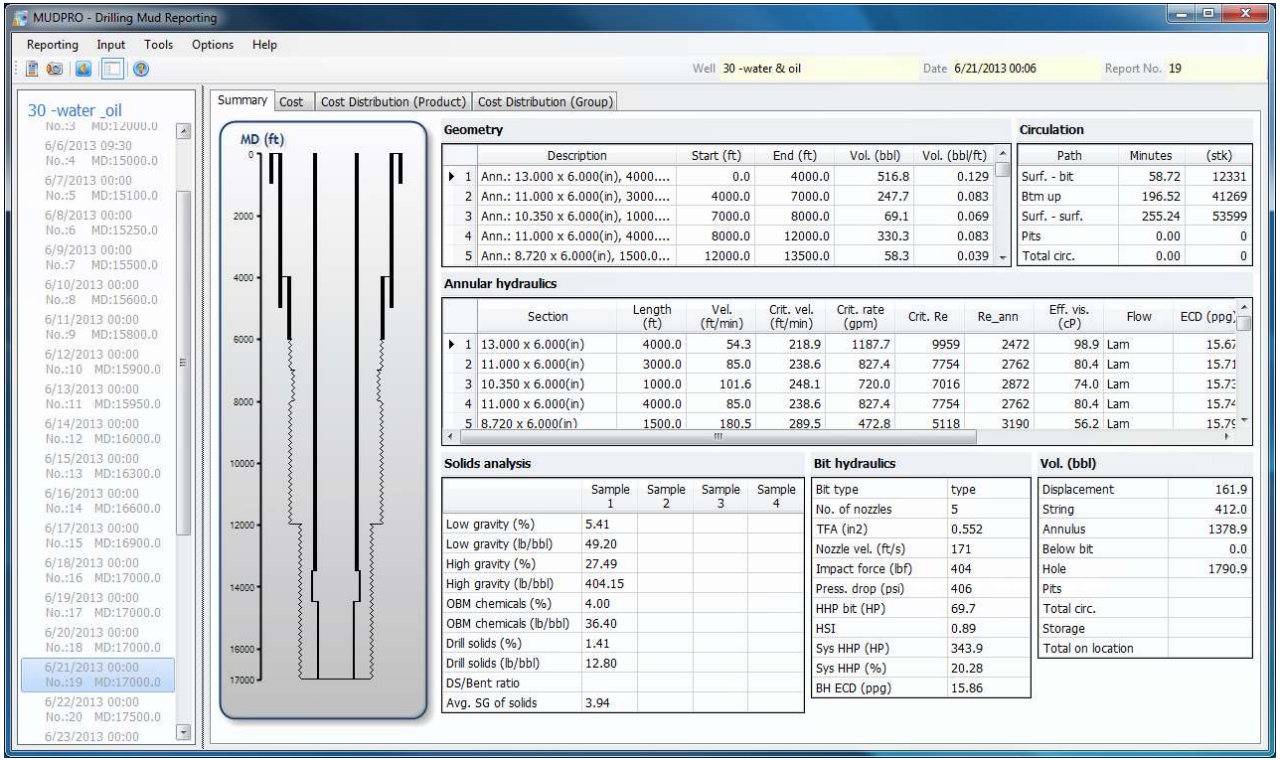
Steps of data wrangling

- Discuss with researchers in other areas to determine which data are needed
- Collect and extract the data
- Visualize the data (never analyze data without visualization)
- Clean the data

- Outcome:
  - Raw data
  - Cleaned data
  - Documentation on how to create clean data from raw
  - Data dictionary

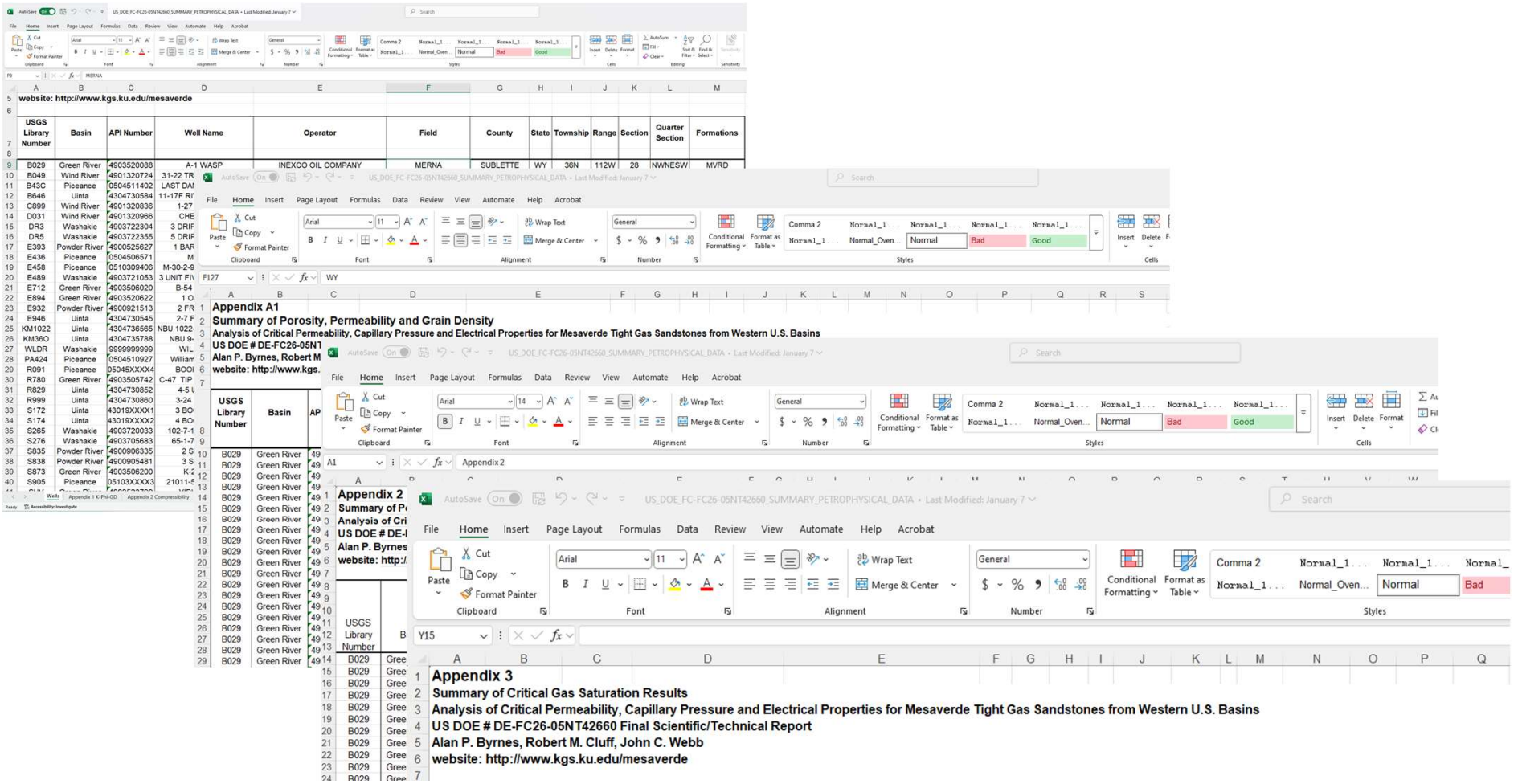
Example 1: raw data

- Raw data are collected directly from the source
- The data shown below are structured because they have a time column, date, wellbore geometry, and pumping schedule
- Data usually need some manipulation for each project, even if it is structured



Example 2: raw data of second homework

- We should spend time figuring out where and how the required data are stored


website: http://www.kgs.ku.edu/mesaverde

USGS Library Number	Basin	API Number	Well Name	Operator	Field	County	State	Township	Range	Section	Quarter Section	Formations
B029	Green River	4903520088	A-1 WASP	INEXCO OIL COMPANY	MERNA	SUBLETTE	WY	36N	112W	28	NWNESW	MVRD
B049	Wind River	4901320724	31-22 TR									
B43C	Piceance	0504511402	LAST DAI									
B646	Uinta	4304730584	11-17F RI									
C899	Wind River	4901320836	1-27									
D031	Wind River	4901320966	CHE									
DR3	Washakie	4903722304	3 DRIF									
DR5	Washakie	4903722355	5 DRIF									
E393	Powder River	4900525627	1 BAR									
E436	Piceance	0504506571	M									
E458	Piceance	0510309406	M-30-2-9									
E489	Washakie	4903721053	3 UNIT FI									
E712	Green River	4903506020	B-54									
E894	Green River	4903520622	1 O									
E932	Powder River	4900921513	2 FR									
E946	Uinta	4304730545	2-7 F									
KM1022	Uinta	4304736565	NBU 1022-									
KM36O	Uinta	4304735788	NBU 9-									
WLDR	Washakie	9999999999	WIL									
PA424	Piceance	0504510927	Willian									
R091	Piceance	05045XXXX4	BOOI									
R780	Green River	4903505742	C-47 TIP									
R829	Uinta	4304730852	4-5 U									
R999	Uinta	4304730860	3-24									
S172	Uinta	43019XXXX1	3 BO									
S174	Uinta	43019XXXX2	4 BO									
S265	Washakie	4903720033	102-7-1									
S276	Washakie	4903705683	65-1-7									
S835	Powder River	4900906335	2 S									
S838	Powder River	4900905481	3 S									
S873	Green River	4903506200	K-2									
S905	Piceance	05103XXXX3	21011-5									

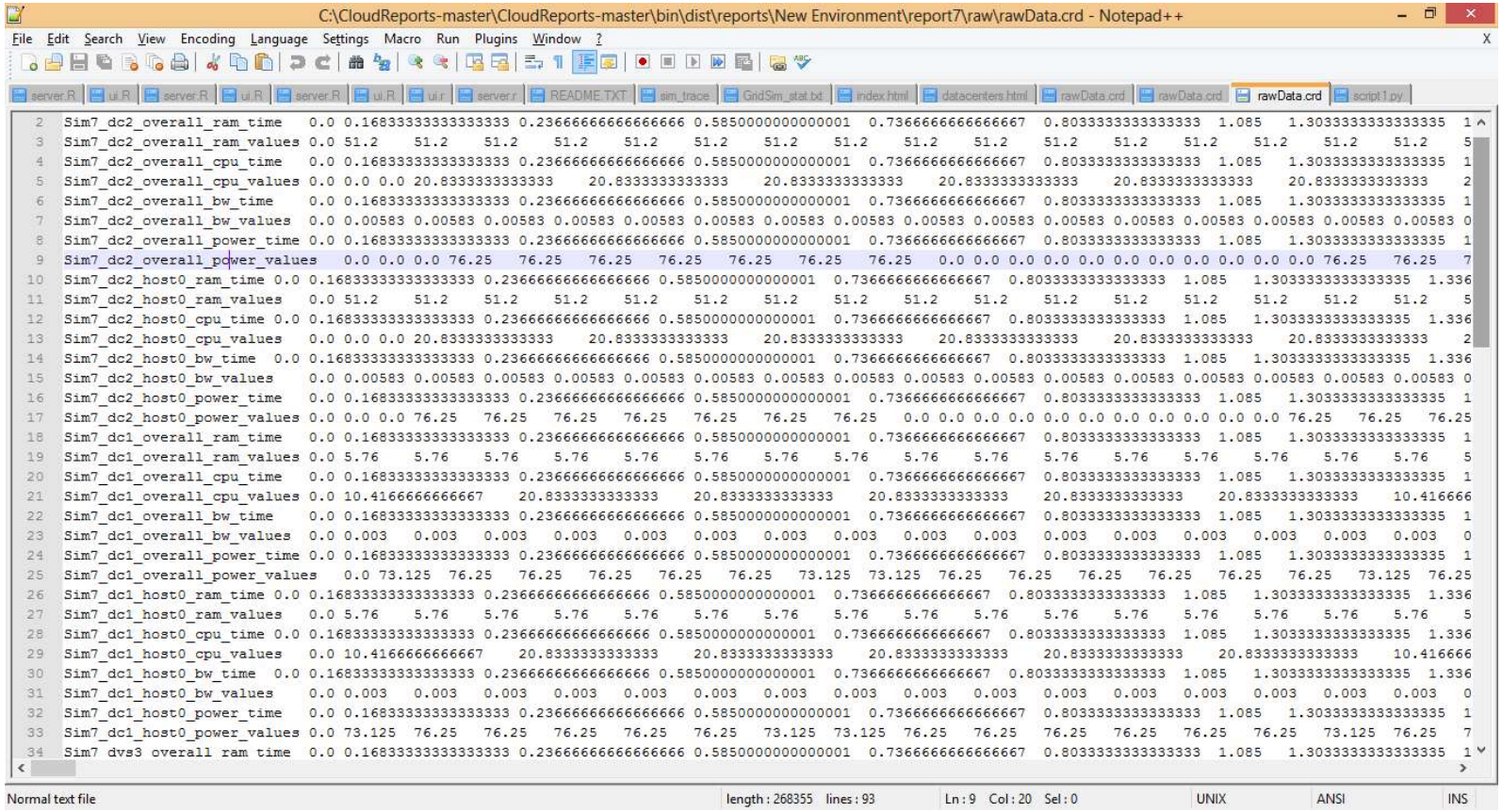
Appendix A1
Summary of Porosity, Permeability and Grain Density
Analysis of Critical Permeability, Capillary Pressure and Electrical Properties for Mesaverde Tight Gas Sandstones from Western U.S. Basins
US DOE # DE-FC26-05NT
Alan P. Byrnes, Robert M
website: http://www.kgs.

Appendix 2
Summary of Pe
Analysis of Cri
US DOE # DE-
Alan P. Byrnes
website: http:/

Appendix 3
Summary of Critical Gas Saturation Results
Analysis of Critical Permeability, Capillary Pressure and Electrical Properties for Mesaverde Tight Gas Sandstones from Western U.S. Basins
US DOE # DE-FC26-05NT42660 Final Scientific/Technical Report
Alan P. Byrnes, Robert M. Cluff, John C. Webb
website: http://www.kgs.ku.edu/mesaverde

Example 3: reservoir simulation results

- The benefits of using DataFrame is clear



Clean data are processed data ready for the analysis

- We remove irrelevancies, errors, outliers, and inaccuracies to clean the data
- Below is an example of clean data (again, notice the benefits of using a DataFrame to organize your data)
- Each column indicates the variable
- Each row represents the observation

Depth	TOC	Calcite	Porosity	Swirr	Quartz	RockClass
2773.88	4.39	0.00	7.06	7.11	40.63	SS
2776	4.39	6.36	6.34	9.36	51.61	SS
2778.13	3.45	2.64	4.70	5.93	41.46	SS
2780.25	0.31	67.33	6.32	11.78	4.07	LS
2782.48	4.06	0.62	5.69	7.38	38.09	SS
2784.49	4.00	2.89	6.19	7.59	40.53	SS
2786.62	3.52	6.03	0.70	8.62	29.14	DO

Data dictionary provides a clear description in business terms

- This will help everyone when the project is transferred
- Many recipients are not in your field
- There should be at least two columns

Variable	Description	Data Type	Source	Required
TVD	True vertical depth, ft	Integer	Drilling Report	Yes
MD	Measured depth, ft	Integer	CoreCad	Yes
TOC	Total organic content, %	Float	CoreCad	Yes
Calcite	Calcite content, %	Float	CoreCad	Yes
Swirr	Water saturation irreducible, %	Float	CoreCad	Yes
phi	Porosity, %	Float	CoreCad	Yes
Quartz	Quartz content, %	Float	CoreCad	Yes
Rock Class	Rock classification (SS: Sandstone, LS: Limestone,..)	Varchar	RockWare software/SME	Yes

Code documentation

- Code documentation explains how you went from raw to clean data
- It helps everyone especially you when the project is transferred
- It will help to add data or modify the project in the future
- If others could not replicate the results, you are unlikely to receive credit for the work (they may even question the integrity)

First example of code documentation

- Add comments (use#) to your code as much as possible
- This will help when you return to a code later

```
#cell 1- required libraries to visualize the data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats # for statistical properties
```

```
#############################################################################################
#cell 2- variables are defined here
depth=[2699.7, 2717.1, 2729.9, 2783.3, 2817.7, 2831.8, 3433.8, 3461.5,
7703.7, 9397.2, 12158.5, 12162, 10573.1, 10573.1, 11457.8, 11460.6,
11548, 11552.3, 11587.2, 11605.1, 11609.2, 11615.1, 11724.3, 13672.5,
11956.1, 12510.1, 12520.3, 12520.9, 12553.7, 6580.1, 6582.3, 6591.7, 6591.9,
4578.8, 4606.5, 5193.5, 6645.5, 3544.85, 3577.55, 4004.3, 4013.25, 4393.6, 4393.6,
5715.4, 6042.4, 5838.7, 7272.8, 10547.5, 10555.7, 10574.5, 10633.6, 790.3, 6969.9,
```

Second example of code documentation

- Docstrings are extended and more through comments in the code
- They provide a standardized way to define the utility, arguments, exceptions, and functions

```
# cell 1 - Required libraries for data visualization
"""
This cell imports the necessary libraries for data visualization and statistical analysis.

Libraries imported:
- numpy (as np): For numerical operations such as mean, median, and array manipulations.
- pandas (as pd): For creating and manipulating dataframes.
- matplotlib.pyplot (as plt): For creating static, animated, and interactive visualizations.
- seaborn (as sns): For advanced data visualization, built on top of matplotlib.
- scipy.stats: For statistical analysis and computing properties like mode, mean, median, etc.
"""
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats  # for statistical properties
```

```
# cell 2 - Variables are defined here
"""
This cell defines the datasets (depth, permeability, porosity) that will be used for visualiz

Variables:
- depth (list): Depth measurements at different intervals (in feet).
- k (list): Permeability values at corresponding depth measurements (in microdarcy).
- phi (list): Porosity values at corresponding depth measurements (in percentage).
"""
depth = [2699.7, 2717.1, 2729.9, 2783.3, 2817.7, 2831.8, 3433.8, 3461.5,
7703.7, 9397.2, 12158.5, 12162, 10573.1, 10573.1, 11457.8, 11460.6,
11548, 11552.3, 11587.2, 11605.1, 11609.2, 11615.1, 11724.3, 13672.5,
```

Other examples of code documentation

- Use meaningful names for variables. Be as explicit as possible. For instance, use poro for porosity instead of ab12. (This is just a good practice rather than documentation)

- Create README file to provide detailed info about the simulations

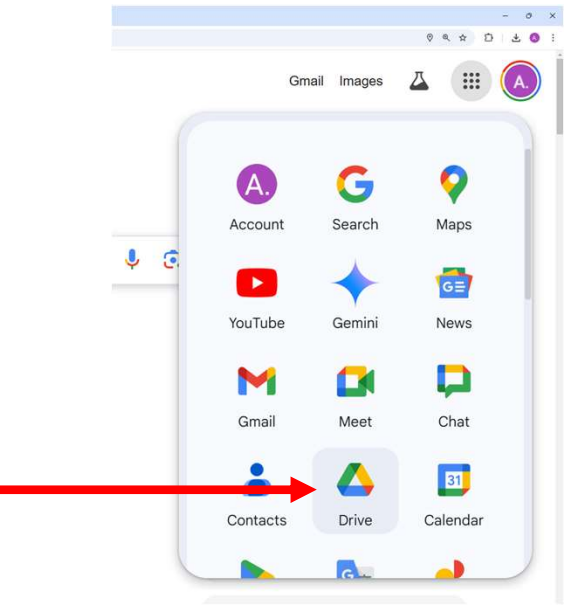
Data Preparation

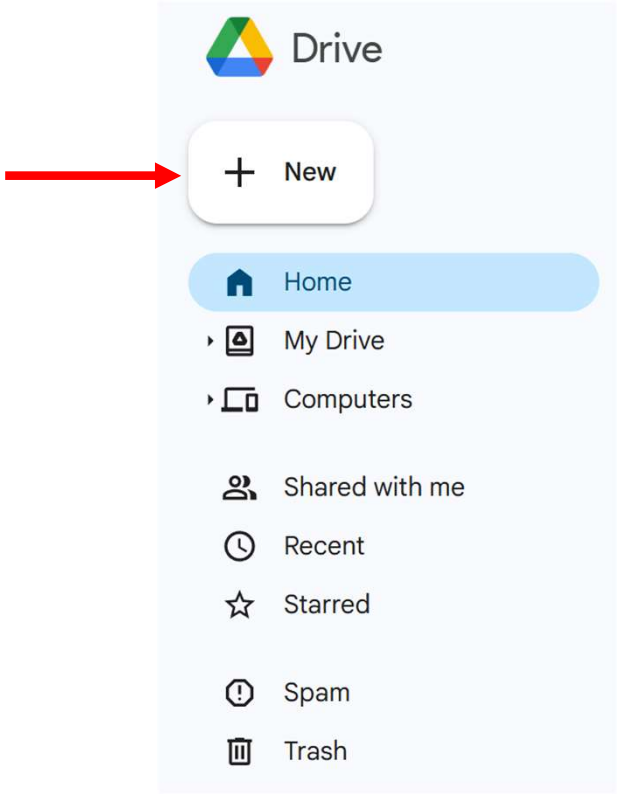
- Merging multiple data sources
- Transformation of raw data into meaningful information
- Drive engineered variables (new variables)
- Remove unwanted observations, duplicates, and outliers
- Fix structural and contradictory errors
- Type conversion
- Deal with missing data
- Normalize and standardize data
- Validate your dataset

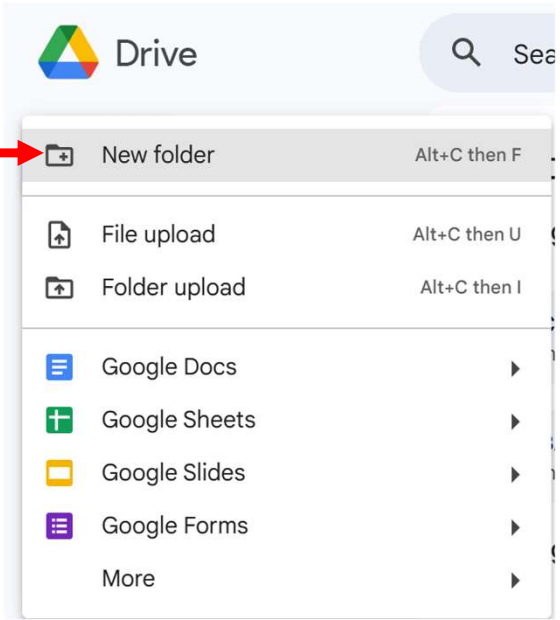
Third code

- Upload a csv file and access it from Google Drive

Next, we create a folder on Google Drive to use Google Colab



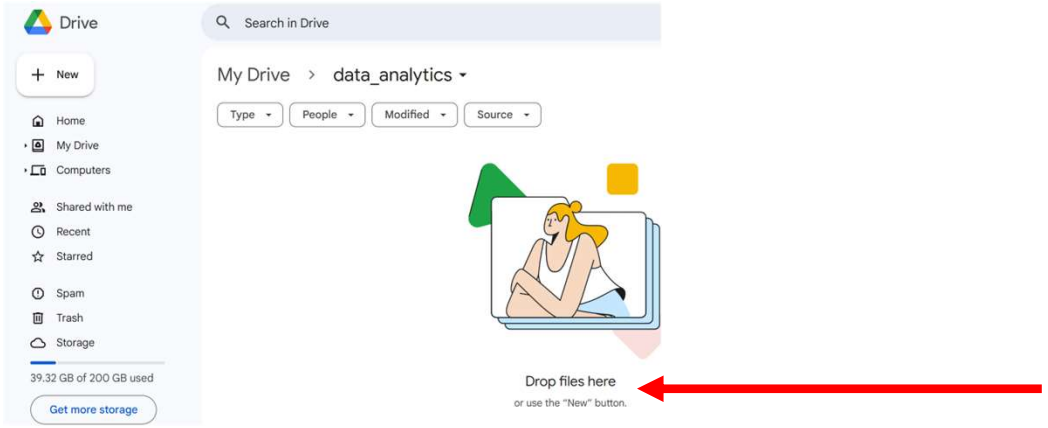


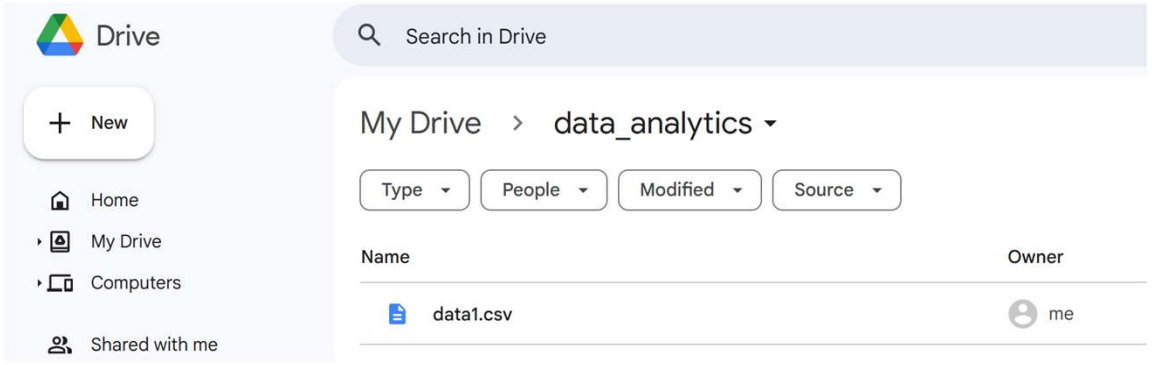


Please feel free to use another software

Upload the file

Locate data1.csv on Canvas and upload it to the folder you created on Google Drive





Use its address to update the path in wrangling1

- Locate the wrangling1.txt on Canvas
- Divide and insert its cells in a new Notebook
- Update this line of the code:

Check the uploaded data

- Print the imported data and compare it with the source file

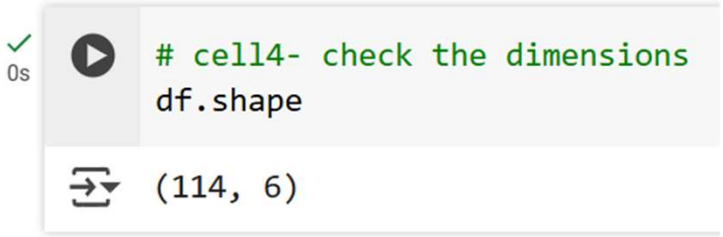
```
# cell3- udpated the following path to access the data
path='/content/drive/MyDrive/data_analytics/data1.csv'
df=pd.read_csv(path)

df
```

	sample	permeability(md)	porosity (%)	depth (ft)	pressure(psi)	production rate(bbl/day)
0	1	29.00000	21.7	2699.70	1112.950964	370.575096
1	2	11.50000	19.9	2717.10	3816.504868	369.718165
2	3	7.96000	19.2	2729.90	119.233582	133.720583
3	4	24.00000	22.1	2783.30	291.996216	104.202517
4	5	2.07000	19.8	2817.70	3987.438445	121.171415
...	...	...	...	...	...	...

Check the dimension of the uploaded data

- Is the printed dimension consistent with the data1.csv file?



Check if you have duplicate rows (data wrangling)

```
# cell 5
df.duplicated().sum() # summarizes the number of duplicates
```

```
8
```

```
[20] #cell 6- check if there are duplicate values
     df.duplicated() # shows which variables are duplicates
```

	0
0	False
1	False
2	False
3	False

Remove duplicate values and check the dimensions

```
# cell7 - remove duplicate variables and check the dimension
df=df.drop_duplicates()# drops duplciate variables
df.shape # check the dimension
```

```
(106, 6)
```

Missing data is another common issue addressed in data wrangling

- Types of missing data:
  - Missing entirely at random (MCAR)
  - Missing at random (MAR)
  - Missing not at random (nonignorable)

- Imputation is replacing missing value with another value

There are many methods to address the missing data

- Delete rows (or columns) containing missing values
- Impute the missing value using the mean or median
- Predict the missing values with algorithms such as KNN and linear regression
- Impute missing categorical values with the most frequent category or a new category
- Use Last Observation Carried Forward (LOCF) or Next Observation Carried Backward (NOCB) for imputation
- Interpolate missing values using timestamps before or after

These methods are easily accessible using python libraries (pandas, scikit)

pandas.DataFrame.dropna

```
DataFrame.dropna(*, axis=0, how=<no_default>, thresh=<no_default>, subset=None,
inplace=False, ignore_index=False)                                    [source]
```

Remove missing values.

sklearn.impute

Transformers for missing value imputation.

User guide. See the Imputation of missing values section for further details.

`IterativeImputer`	Multivariate imputer that estimates each feature from all the others.
`KNNImputer`	Imputation for completing missing values using k-Nearest Neighbors.
`MissingIndicator`	Binary indicators for missing values.
`SimpleImputer`	Univariate imputer for completing missing values with simple strategies.

Determine how many values are missing and remove them in our code

```
#cell 8 - report how many values are missing
number_of_missing_values = df.isna().sum().sum()
print(f"number of missing values: {number_of_missing_values}")
```

```
number of missing values: 6
```

```
#cell 9 - remove the rows with missing
df_without_missing = df.dropna()
df_without_missing.shape
```

```
(100, 6)
```

Outliers

Boxplot provides a convenient tool to visualize the outliers

Outliers are more than 3 standard deviations away from the mean

Median (50th percentile)

First quartile (25th percentile)

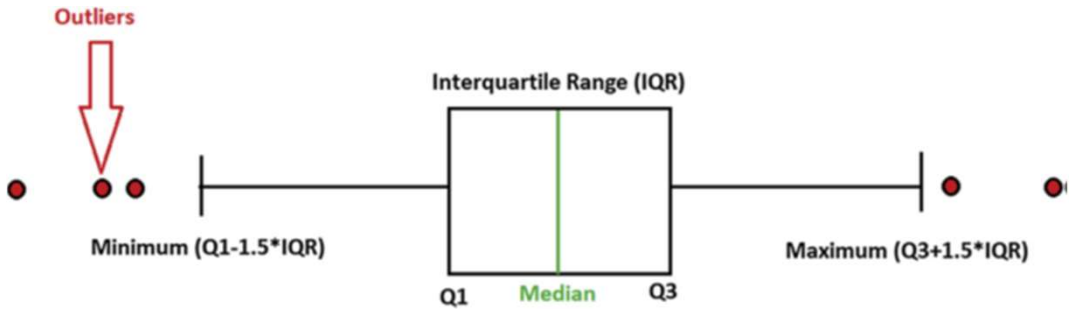
Third quartile (75th percentile)

Interquartile range (25th to 75th percentile)

Maximum (Q3+1.5*IQR): This is not the actual maximum in a dataset

Minimum (Q1-1.5*IQR): This is not the actual minimum in a dataset

Outliers are outside of defined "Min" and "Max"

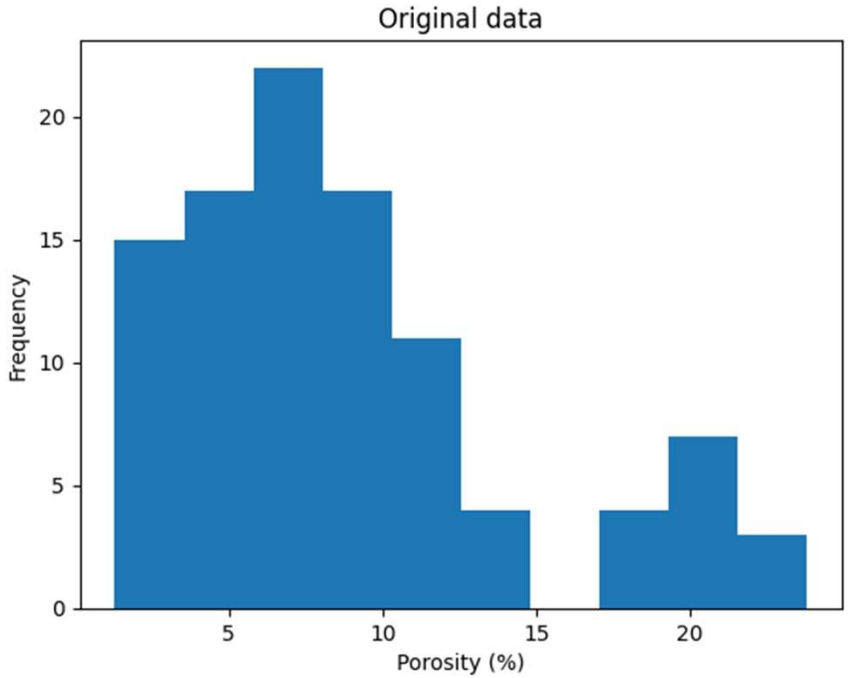


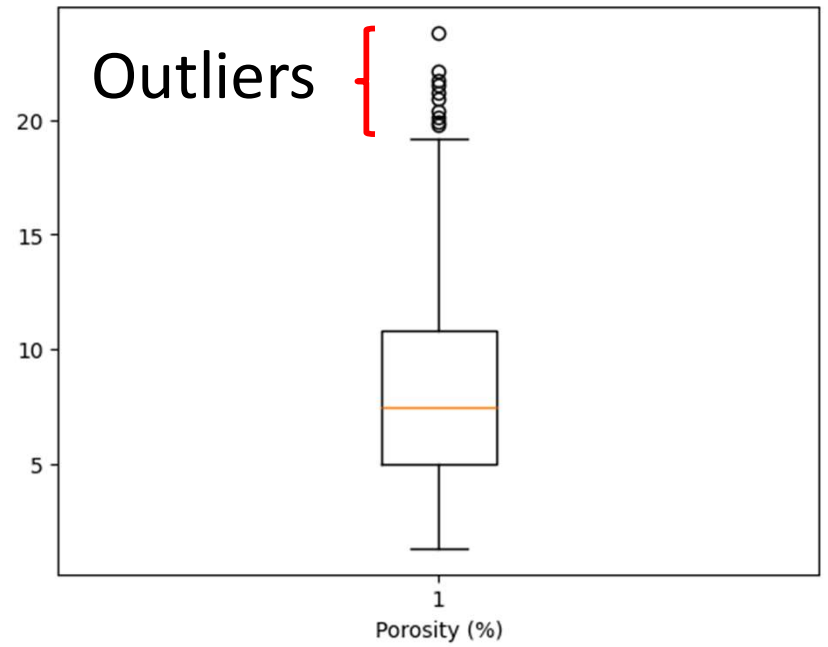
Outliers are also addressed in data wrangling

- Always plot histograms and boxplots, especially to address the outliers (there may be a long tale)

```
# cell 10 - histogram of porosity
plt.hist(df_without_missing['porosity (%)'], bins=10)
plt.xlabel('Porosity (%)')
plt.ylabel('Frequency')
plt.show()
```

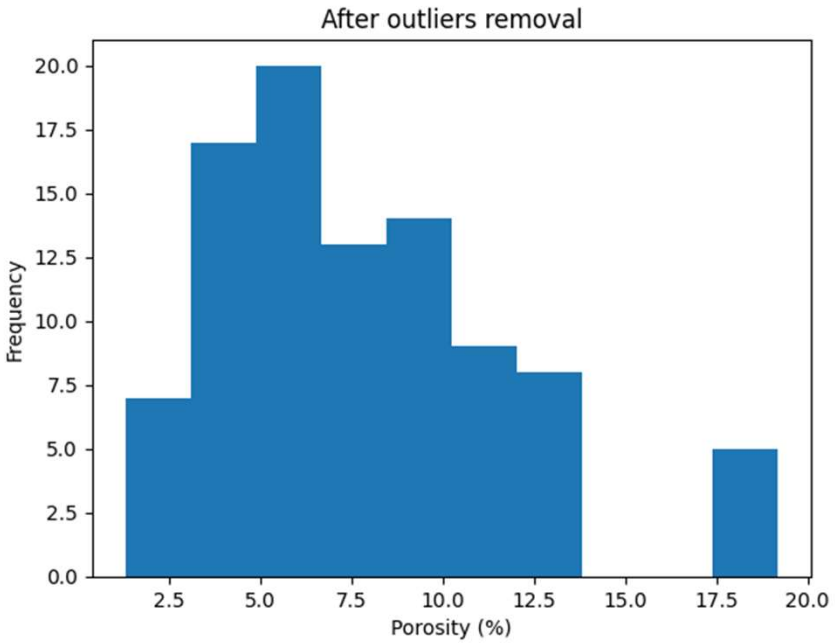
```
# cell 11 - boc plot of porosity
plt.boxplot(df_without_missing['porosity (%)'])
plt.xlabel('Porosity (%)')
plt.show()
```

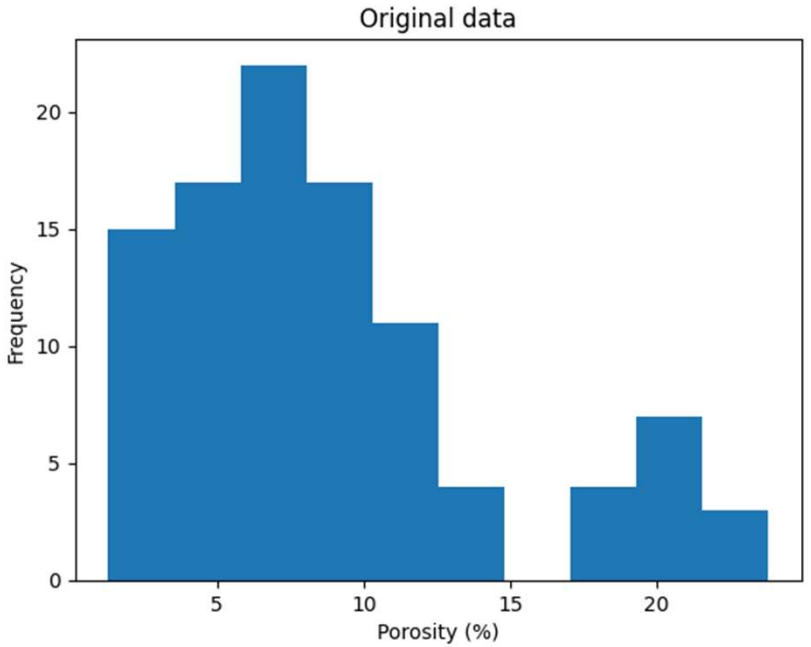




Remove the outliers and plot the histogram

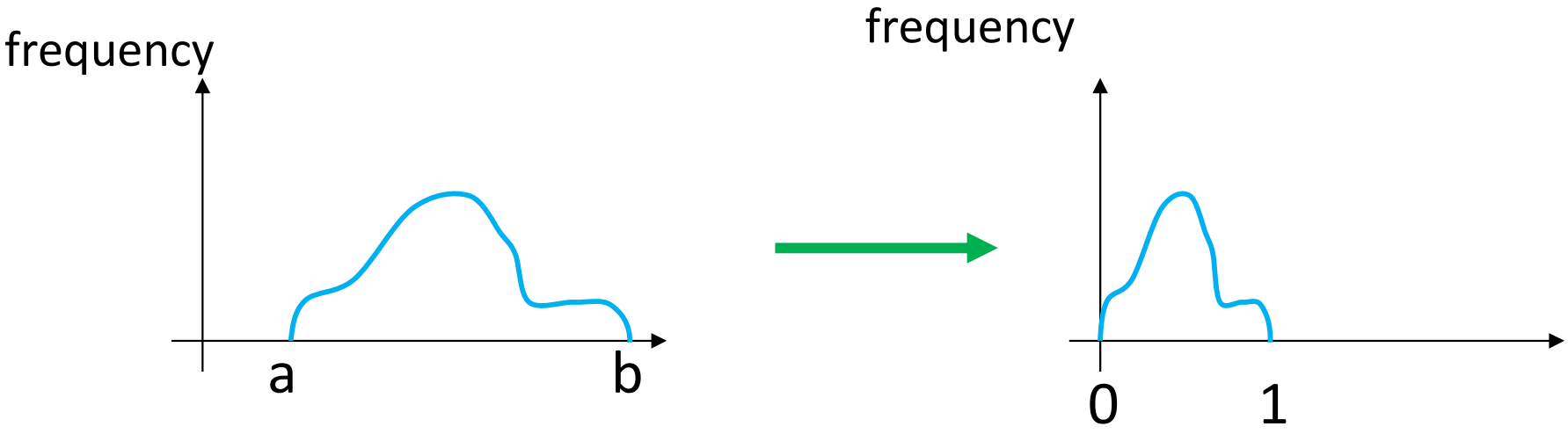
```
# cell 12- remove the outlier and plot the histogram
Q1 = df_without_missing['porosity (%)'].quantile(0.25)
Q3 = df_without_missing['porosity (%)'].quantile(0.75)
# Calculate IQR
IQR = Q3 - Q1
# Define outlier condition
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Remove outliers
df_cleaned = df[(df['porosity (%)'] >= lower_bound) & (df['porosity (%)'] <= upper_bound)]
plt.hist(df_cleaned['porosity (%)'], bins=10)
plt.xlabel('Porosity (%)')
plt.ylabel('Frequency')
```





Normalization (feature scaling)

- Normalization (min-max operation) is a common method to scale the data:

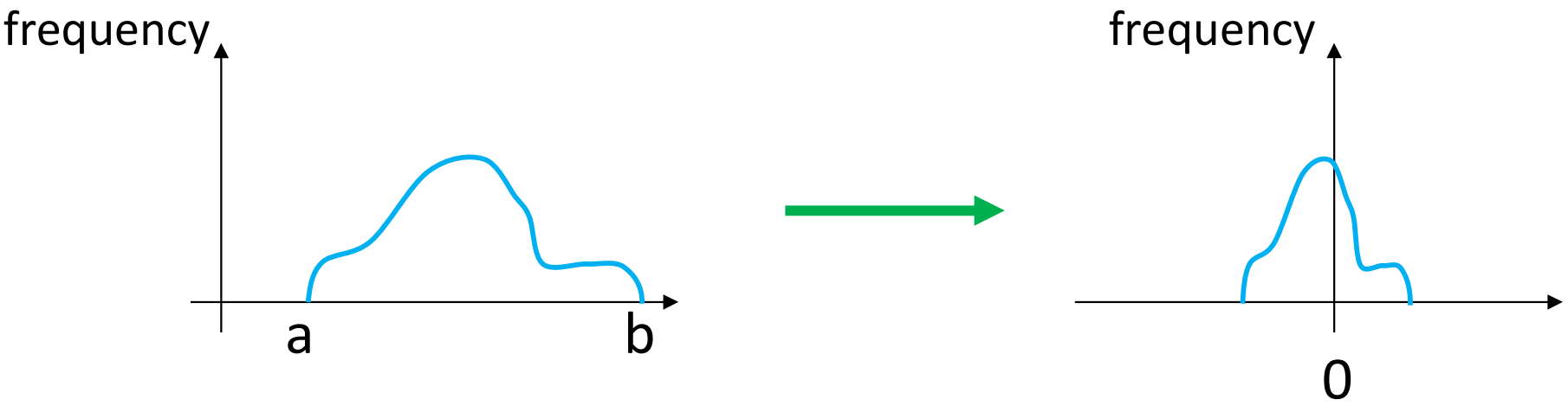


1. Subtract the min (a)
2. Divide the result by the difference (absolute value of b-a)

Command in Python: MinMaxScaler

Standardization (feature scaling)

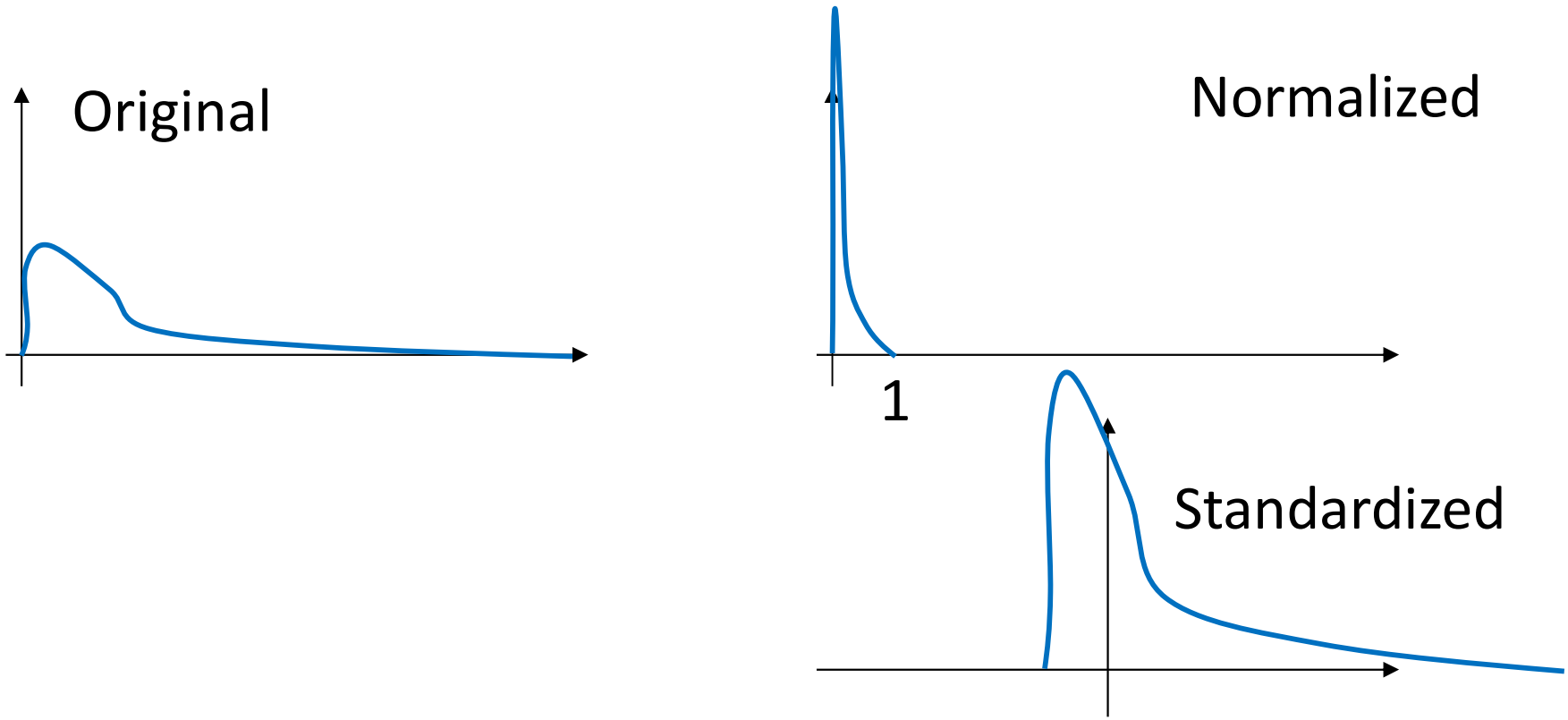
- Standardization is another common method for scaling
- Standardization is less affected by outliers



- 1. Subtract the min: (a+b)/2
- 2. Divide the result by the standard deviation

Command in Python: StandardScaler

Heavy tail has a negative impact on scaling



- Solution: transform the data before scaling (feature engineering focuses on this issue)

Feature engineering is another part of data wrangling

- Feature engineering uses domain knowledge to extract the main features
- Techniques:
  - Log Transform
  - One-hot encoding
  - Change scale or unit
  - Calculate a new feature
- Which one of these helps in the wrangling code?

Correlation coefficient shows a linear relationship

- We should always calculate the correlation coefficients of all pairs in our data
- The coefficient captures only the linear relation
- What is the meaning of correlation coefficient = -1, 0, 1?
- Q1: Does y depend on x in the following relations if -1<x<1?
    a) y=x
    b) y=x^2
    c) y=x^3
    d) y=x^4
    e) y=x^5
- Q2: Does the correlation coefficient capture the dependency?

Extend the code

1. Determine all the correlation coefficients

2. Plot all the cross plots (hint: use scatter_matrix from Pandas or sns.pairplot from Seaborn library)

Find the meanings of these phrases from the book

- Data pipeline
- Data snooping
- Ensemble method