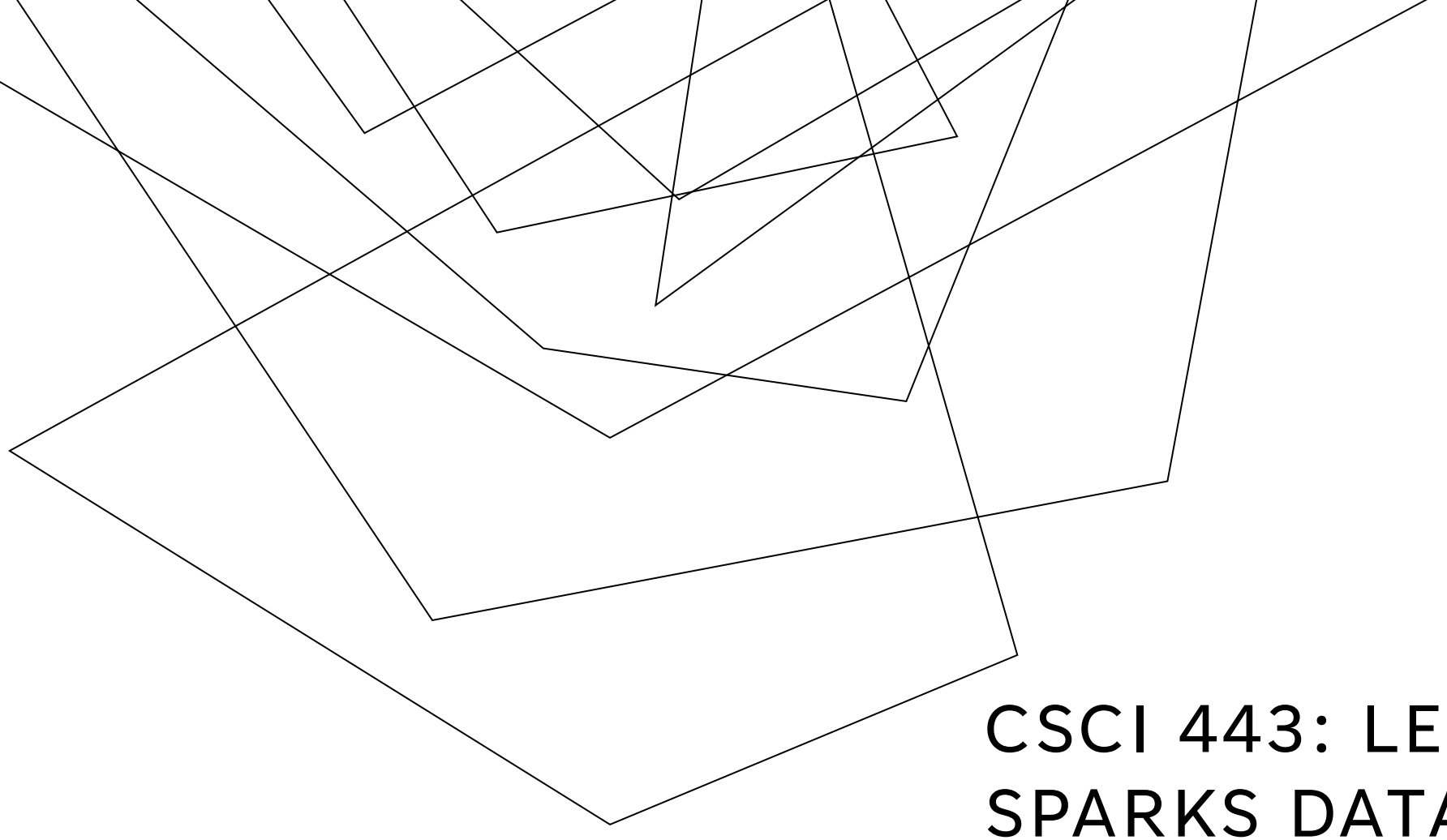


CSCI 443: LECTURE 3 SPARKS DATA FRAMES, CHAPTER 1

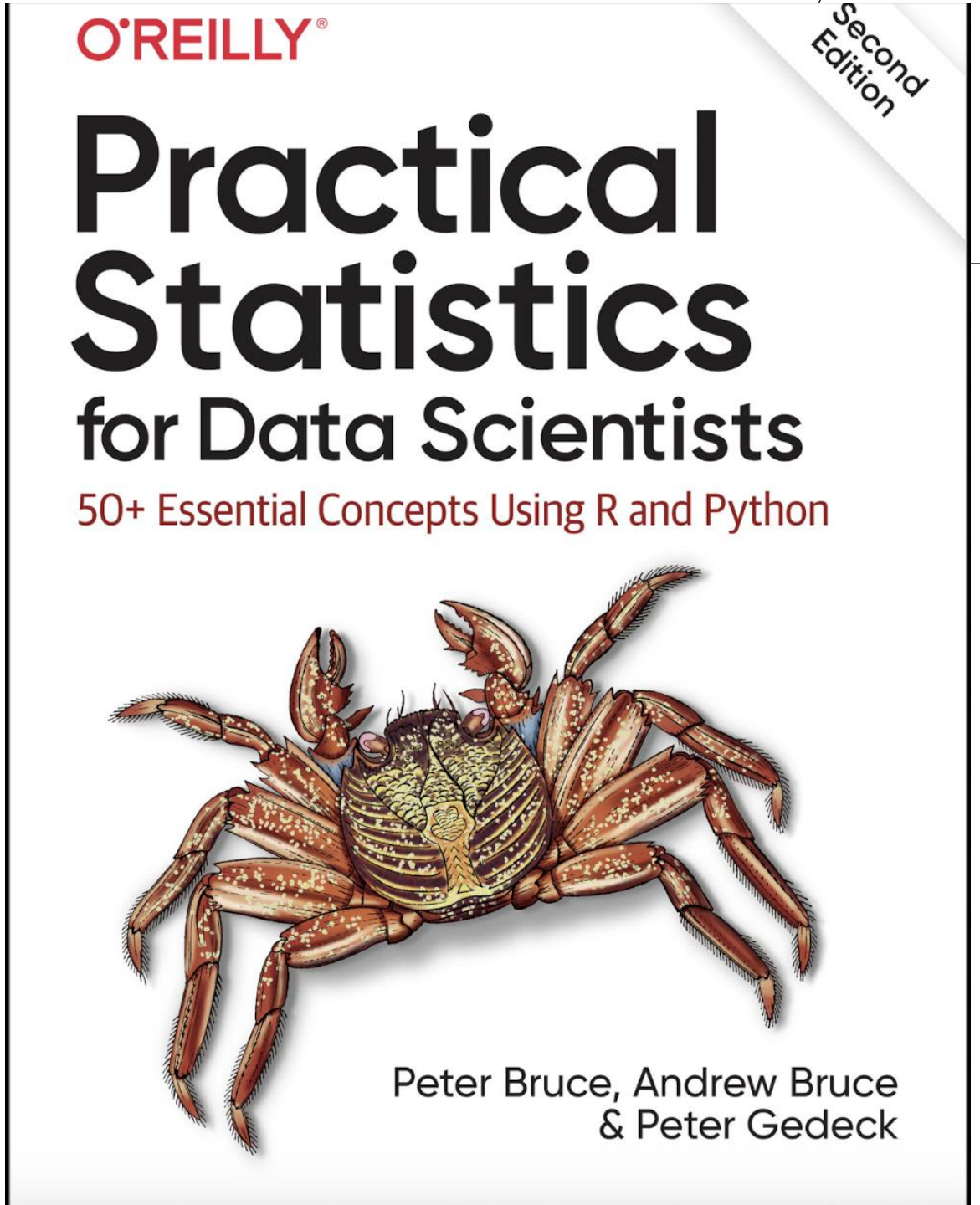
Professor David Harrison

TODAY

- DataFrames
- Chapter 1
- Spark DataFrames
- Computing common statistics with large data.

READING!

- Book provides examples in Python and R. We are using Python.
- Read Chapter 1: Exploratory Data Analysis.
- We need to learn the terminology.
  - Types of data, features, outcomes, records.
  - DataFrames, indices, ...



HOMEWORK 1

Due Thursday, January 30.

11 pm.

Focuses on

- setting up accounts,
- using github and Databricks
- Notebooks.

Submission:

- Submit archived Databricks Notebook to Blackboard.
- NOTE: Submission only needs to be the notebook. No README is necessary.

OFFICE HOURS

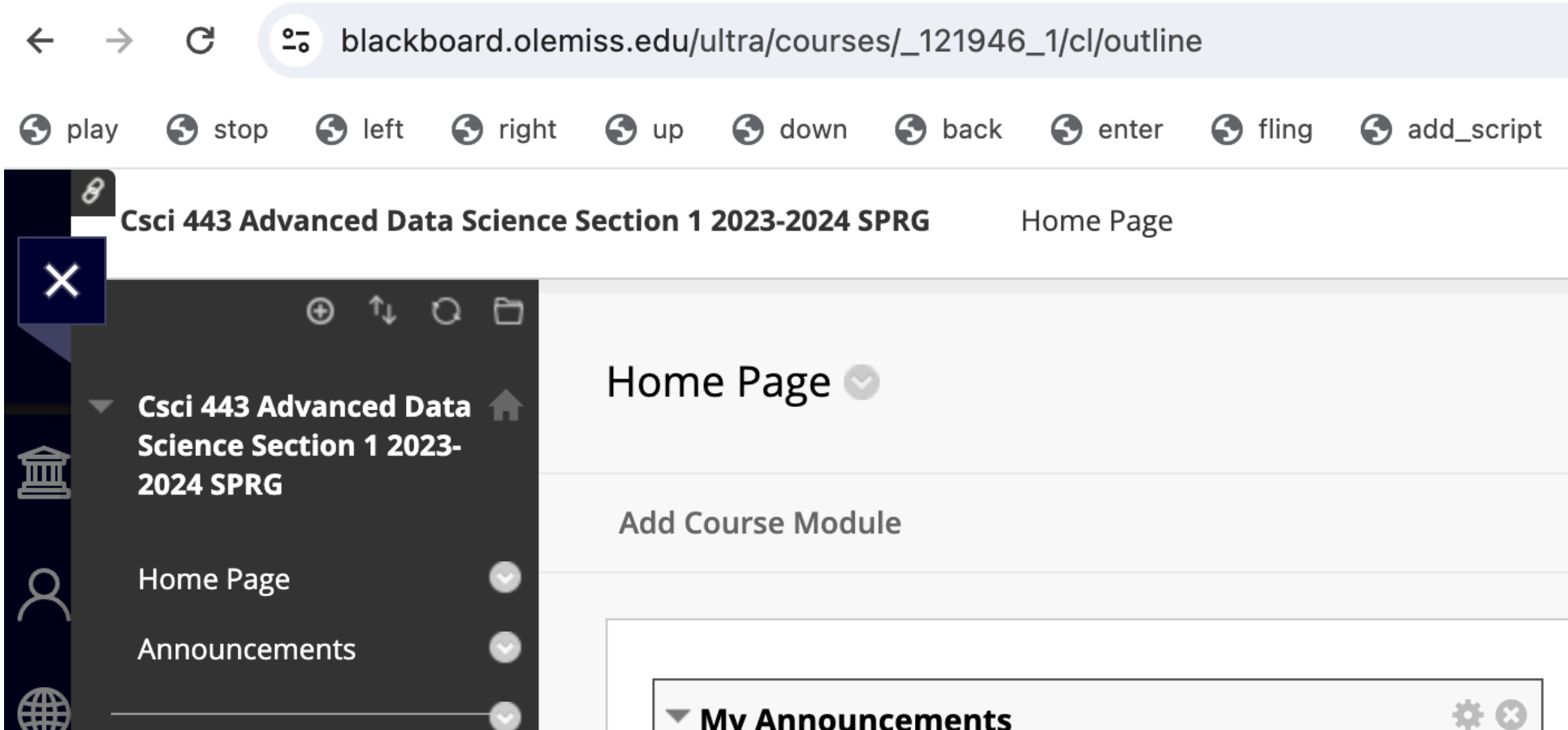
Due to scheduling conflict, office hours updated

Monday	1:00-2:00 PM
Tuesday	4-5 PM

.

BLACKBOARD

All lecture slides, homeworks, and solutions will appear on blackboard.

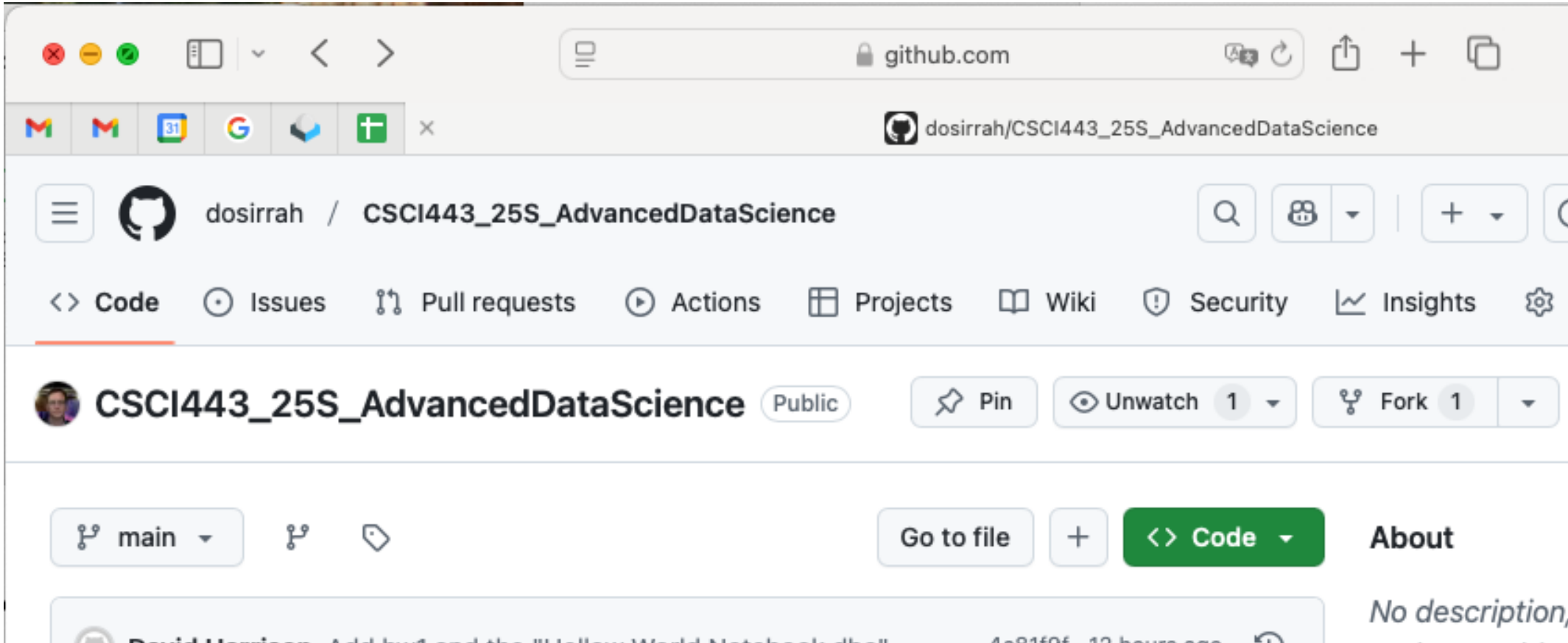


GITHUB

Lecture slides and examples have been committed to GitHub for lectures 1 and 2.

The project is at

https://github.com/dosirrah/CSCI443_25S_AdvancedDataScience



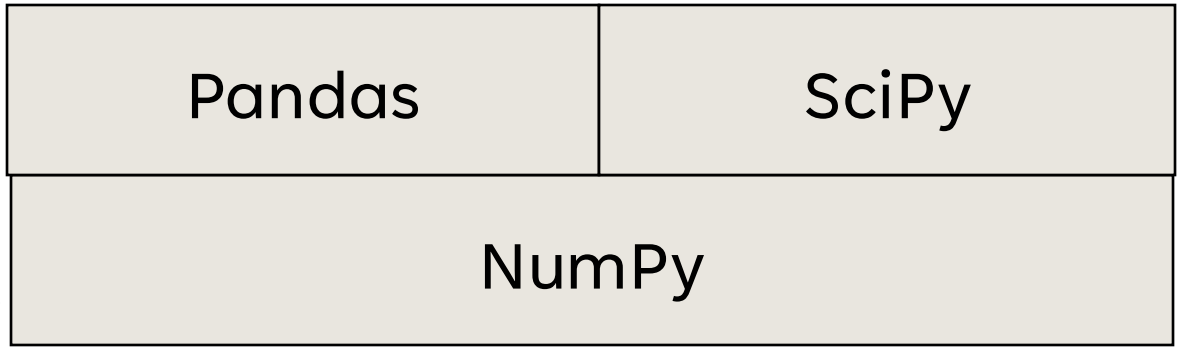
WHY NUMPY?

NumPy provides

- large, memory-efficient, multi-dimensional arrays
- Must faster than Python code.
- Great if you are using jupyter notebooksor python locally.

Downside:

- Doesn't use CUDA. No GPU acceleration
- Only supports basic data types.
- Doesn't exploit cluster-wide operations.
  (We'll come back to this)



WHY USE DATA FRAMES?

NumPy doesn’t provide an annotated tabular data type.

```
# lecture03/example1_data_frames.py
import pandas as pd

# Data to be represented in the DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [20, 21, 19, 22, 20],
    'Grade': [88, 92, 85, 90, 95]
}

# Create a DataFrame from the data
students_df = pd.DataFrame(data)

# Display the DataFrame
print(students_df)
```

CAN PERFORM MATH ON COLUMNS

Can add columns, multiply columns, …

- With other columns
- With a constant

```
x = np.arange(n)  # Generate index values for x
y = np.random.rand(n)  # Generates n random numbers
df = pd.DataFrame( data: {'y': y}, index=x)


start_time = time.time()
df["newy"] = df["y"] + c
end_time = time.time()
```

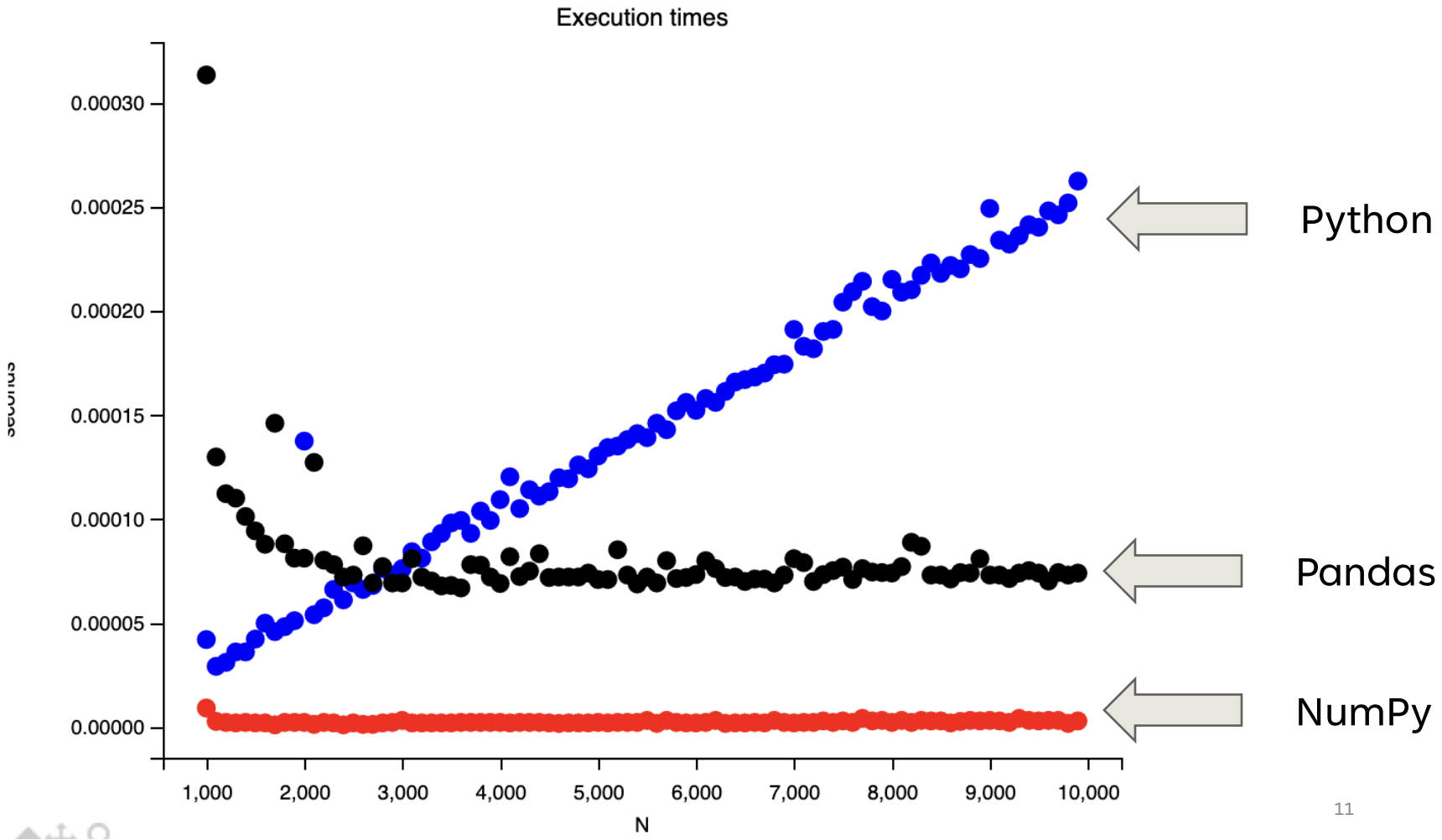
ARE DATA FRAMES FAST?

Pandas uses NumPy underneath.

Pandas = add constant to all elements in a column containing N elements.

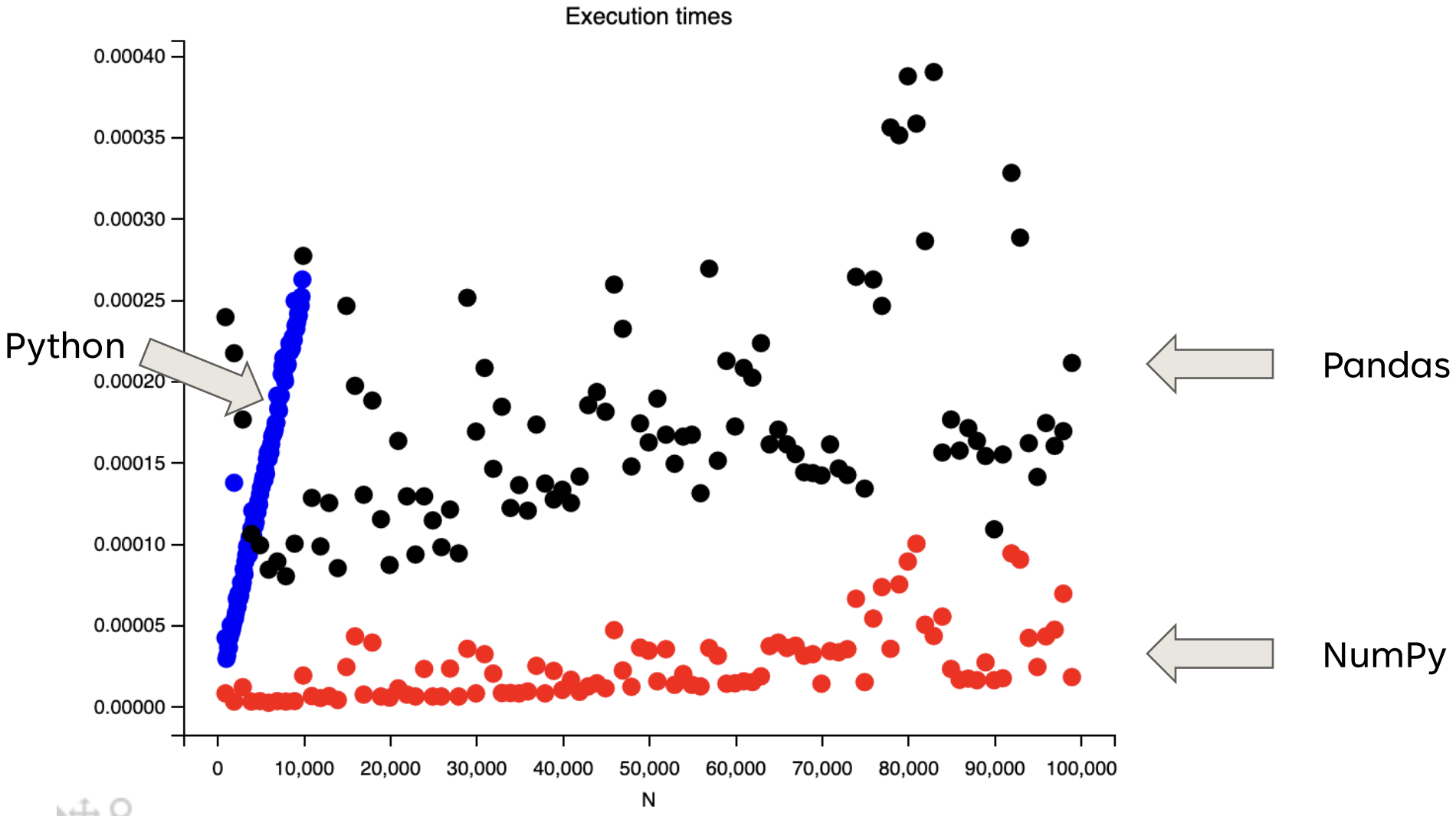
Python = add constant to all elements in a list of length N.

NumPy = add constant to all elements in a NumPy array of length N.



ARE DATA FRAMES FAST?

The tabular abstraction provided py Pandas comes with a cost.



COMMONLY USED OPERATIONS PROVIDED BY DATAFRAMES

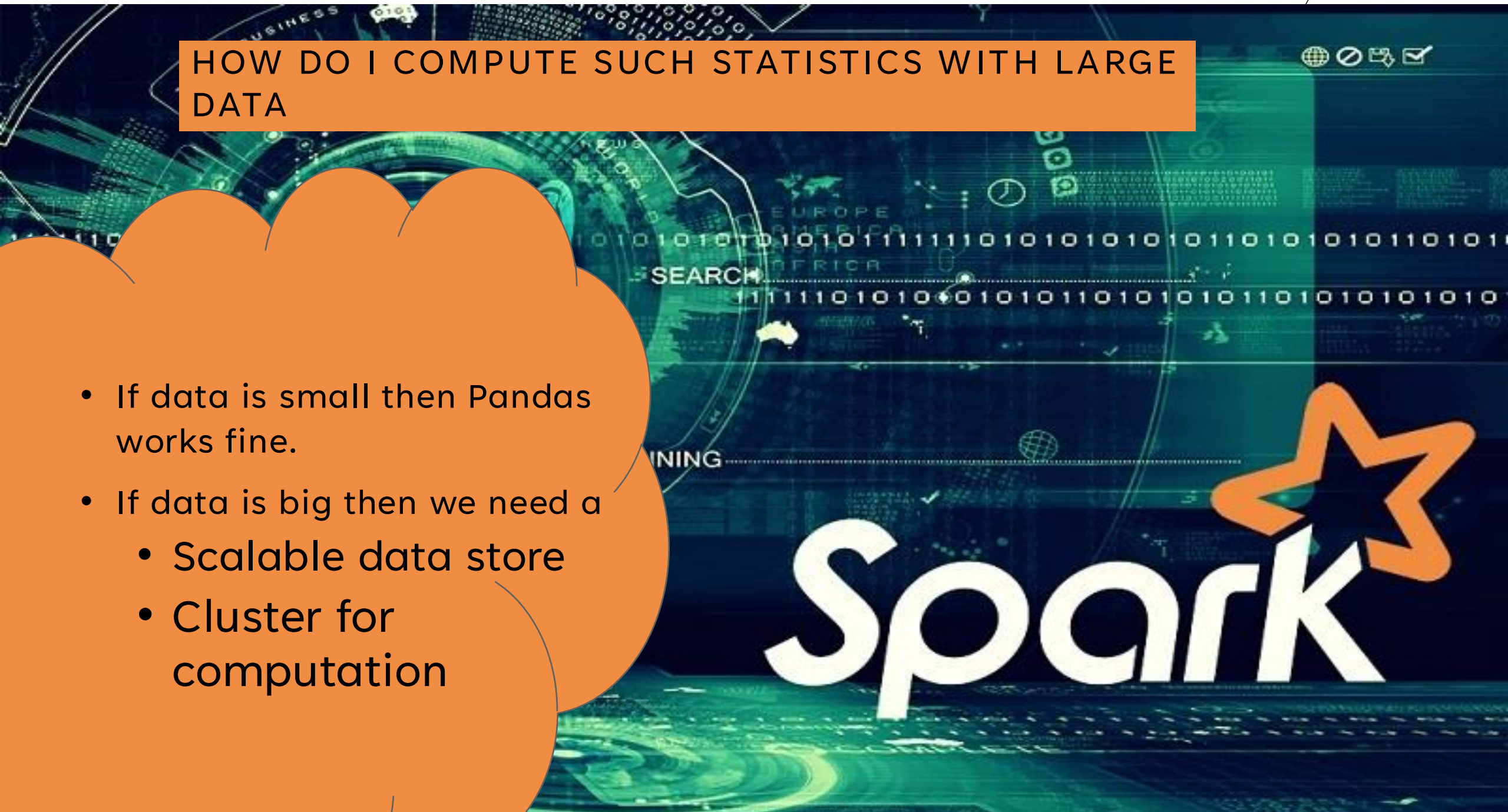
DataFrames provides many operations that can be computed efficiently over columns:

- min
- max
- median
- percentiles
- standard deviation

(see lecture03/example2_df_notebook.dbc in class github repository)

HOW DO I COMPUTE SUCH STATISTICS WITH LARGE DATA

- If data is small then Pandas works fine.
- If data is big then we need a
  - Scalable data store
  - Cluster for computation

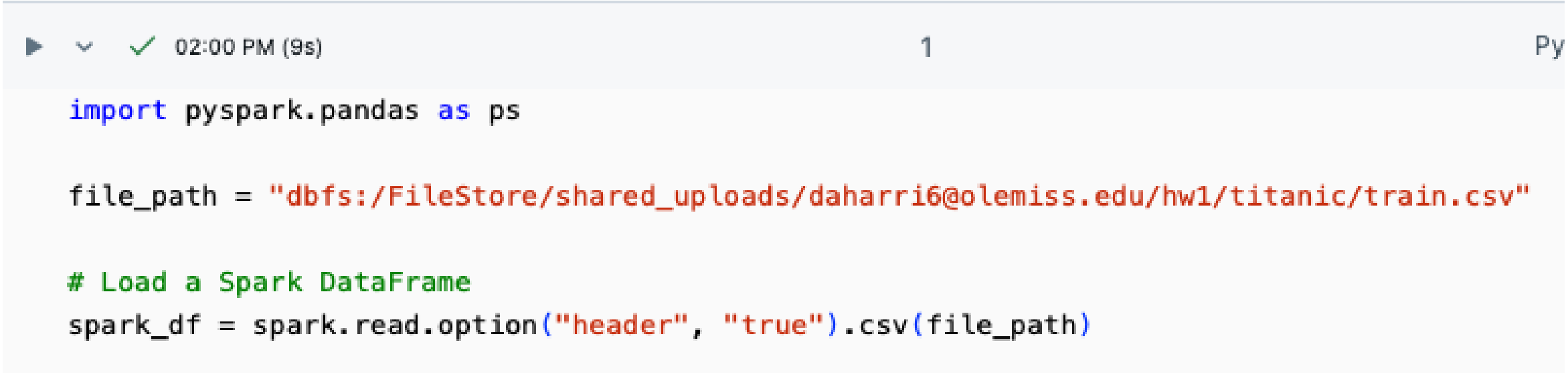


LOADING A DATAFRAME

Load a Spark dataframe.

This DataFrame DOES NOT load the data into loca node.

The DataFrame access the data on a Spark cluster.



LOADING INTO KOALAS

Provides a Pandas interface to DataFrames, but it ISN'T Pandas.

It is still using a Spark DataFrame and the data remains in Spark.

```
# Load a Spark DataFrame
spark_df = spark.read.option("header", "true").csv(file_path)

# Convert Spark DataFrame to a Koalas DataFrame (remains distributed)
df = koalas_df = ps.DataFrame(spark_df)

# Perform Pandas-like operations (but still in Spark)
df.describe()  # Runs in Spark, does NOT load into Pandas
#df["Age"].mean()  # Also runs in Spark
```

▸ (29) Spark Jobs

▸ spark_df: pyspark.sql.dataframe.DataFrame = [PassengerId: string, Survived: string ... 10 more fields]

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891	891	891	891	891	714	891	891	891	891	204	889
unique	891	2	3	891	2	88	7	7	681	248	147	3
top	296	0	3	"Watt, Mrs. James (Elizabeth ""Bessie"" Inglis...	male	None	0	0	1601	8.05	None	S
freq	1	549	491	1	577	177	608	678	7	43	687	644

DATA EXPLORATION

Provides a Pandas interface to DataFrames

```
ps_df.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	None	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	None	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	None	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	None	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	None	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	None	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	None	C

DATA EXPLORATION

Is there missing data?

Just now (1s) 2

```
from pyspark.sql.functions import col, count, when

# Count missing values in the "Age" column
missing_age_count = spark_df.select(
    count(when(col("Age").isNull(), 1)).alias("missing_age_count")
)

missing_age_count.show()


total_count = spark_df.count()
missing_count = spark_df.filter(col("Age").isNull()).count()
```

(6) Spark Jobs

missing_age_count: pyspark.sql.dataframe.DataFrame = [missing_age_count: long]

```
+-----------------+
|missing_age_count|
+-----------------+
|              177|
+-----------------+
```

DATA EXPLORATION

```
# Count missing values in the "Age" column
missing_age_count = spark_df.select(
    count(when(col("Age").isNull(), 1)).alias("missing_age_count")
)
```

- col(“Age”)
  - I want to act on column “Age”.
  - This is a deferred expression meaning that it doesn’t execute immediately.
  - We are building a plan (called a “logical plan”).
  - Spark does lazy evaluation.
  - col(“Age”) by itself is not yet associated with a dataframe.

DATA EXPLORATION

```
# Count missing values in the "Age" column
missing_age_count = spark_df.select(
    count(when(col("Age").isNull(), 1)).alias("missing_age_count")
)
```

- when(col(“Age”).isNull(), 1)
  - Further builds the logical plan.
  - Says to select all the values in the column that are null.
  - Replace all nulls with 1.
- count(when(col(“Age”).isNull(), 1))
  - Build more logical plan.
  - Count the 1’s.
- spark_df.select()
  - Select causes the locigal plan to be mapped onto a physical plan and then executed on the cluster.
- alias(“missing_age_count”)
  - Renames the output.

DEFINITIONS FROM CHAPTER 1

Feature

A column within a table is commonly referred to as a *feature*.

Synonyms

attribute, input, predictor, variable

Outcome

Many data science projects involve predicting an *outcome* — often a yes/no outcome (in Table 1-1, it is "auction was competitive or not"). The *features* are sometimes used to predict the *outcome* in an experiment or a study.

Synonyms

dependent variable, response, target, output

Records

A row within a table is commonly referred to as a *record*.

Synonyms

case, example, instance, observation, pattern, sample

Table 1-1. A typical data frame format

HYPOTHESIS TESTING

Often in Data Science we are trying to find the answer to a question:

- Is a drug safe?
- Is a drug effective?

Or prove a hypothesis

- H1: Drug A is safe.
- H2: Drug A is effective.

Or find a statistic.

- GDP increased by 3.5% in the 3rd quarter.

OUTCOMES

In a clinical trial testing the safety of Drug X, outcomes might include:

- Incidence of specific side effects,
- Changes in vital signs (blood pressure, heart rate),
- Laboratory test results (liver enzyme levels, blood cell counts),
- Reports of adverse events,
- Patient-reported symptoms or quality of life measures.

The hypothesis "Drug X is safe" is a statement that is tested against the collected outcome data.

The specific outcomes measured in answering a hypothesis like safety are called “endpoints.”

Data scientists (or researchers) analyze these outcomes to determine whether they support or refute the hypothesis.

OUTLIERS

All real-world data is subjected to noise. Noise can result in samples that land far from most of the other samples.

Some real-world processes also generate infrequent results far from the other samples.

Both are called ***outliers***.

Can we remove such outliers?

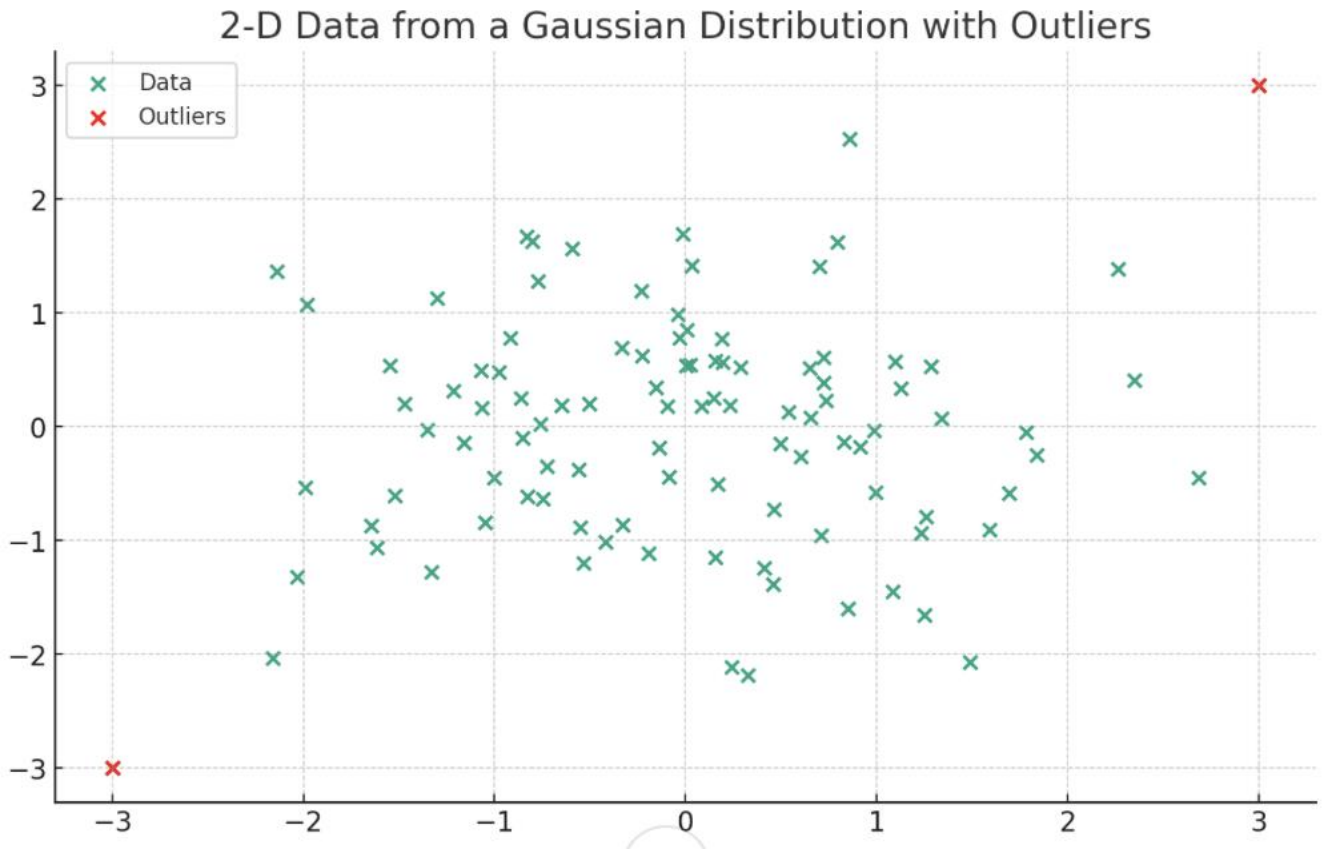
Are they due to ***natural variability***?

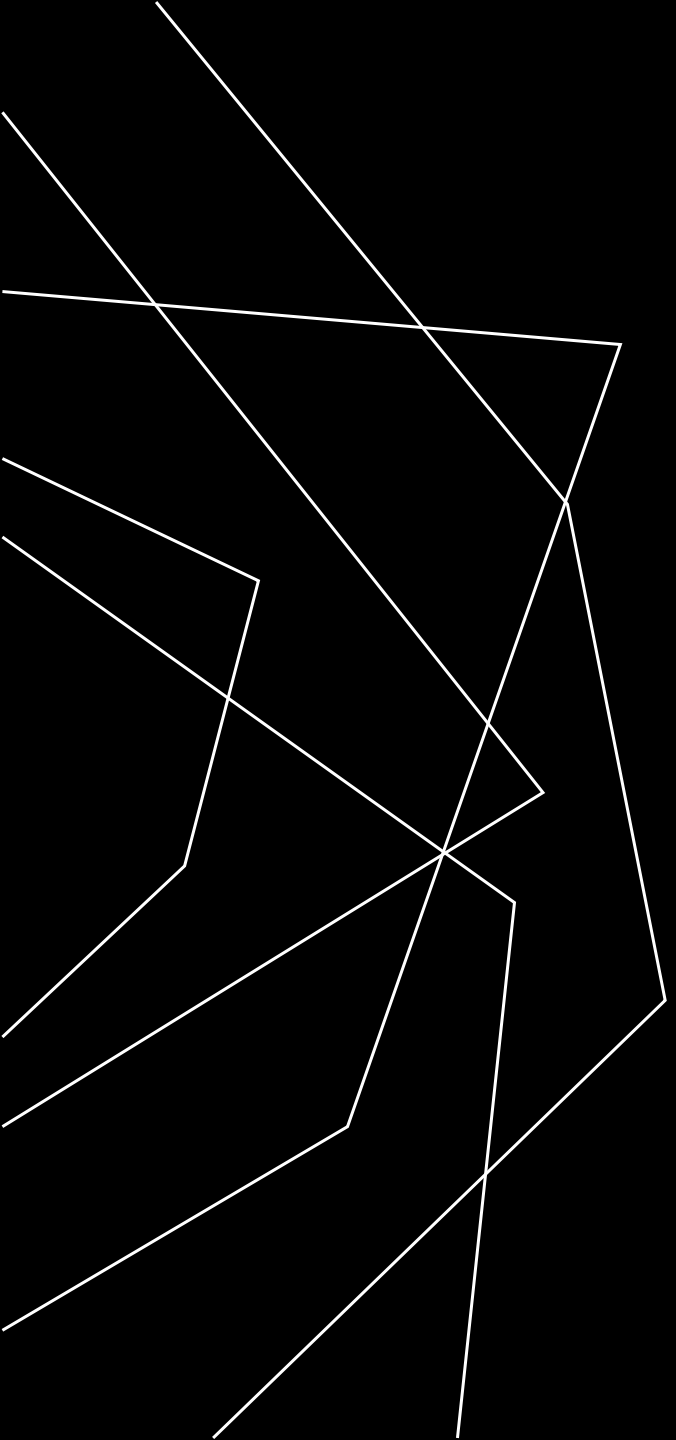
Or are they due to ***error*** in the data collection?

OUTLIERS

Are the outliers really due to error?

Can we remove them as spurious?





THANK YOU

David Harrison

Harrison@cs.olemiss.edu