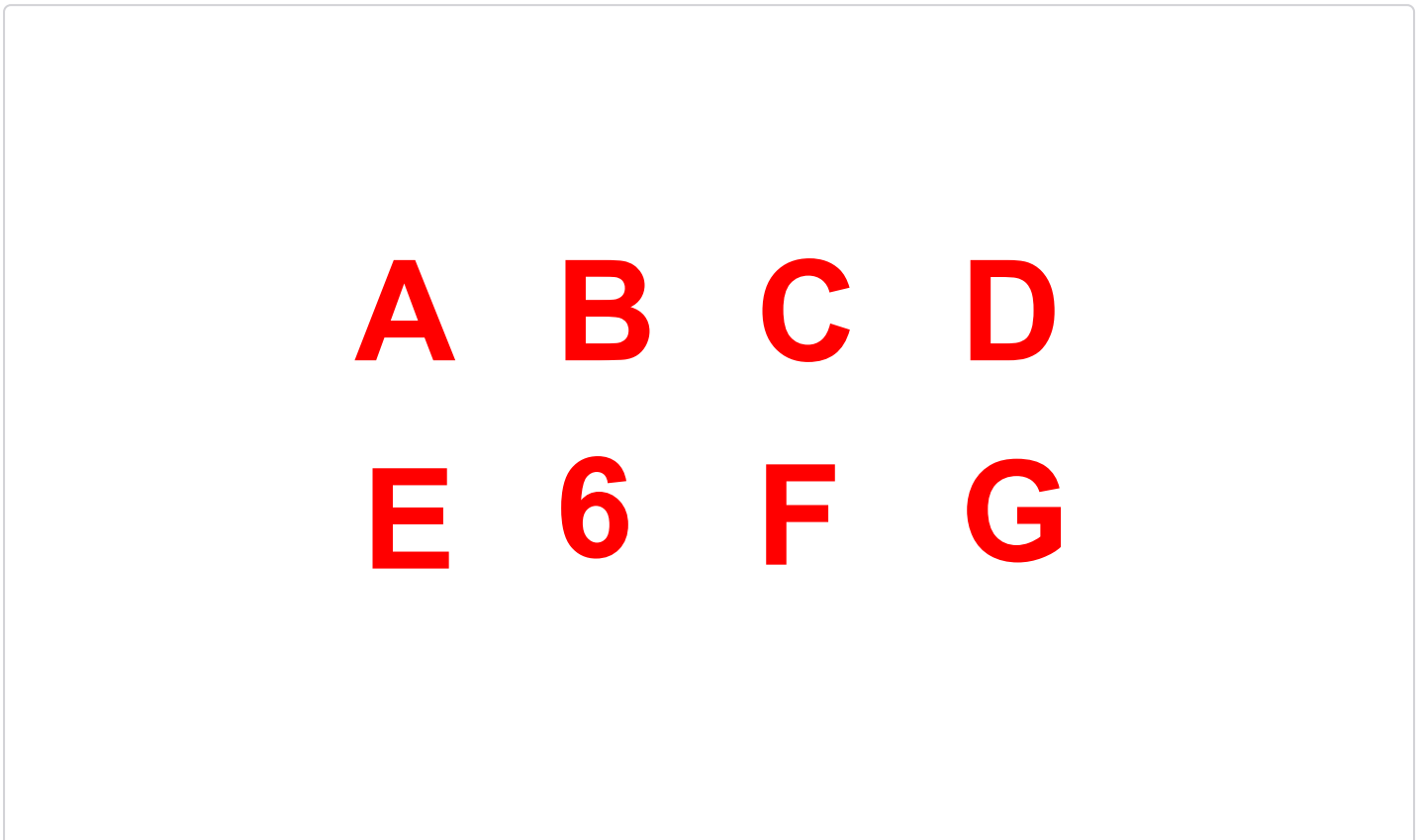
Finding Outliers in Data

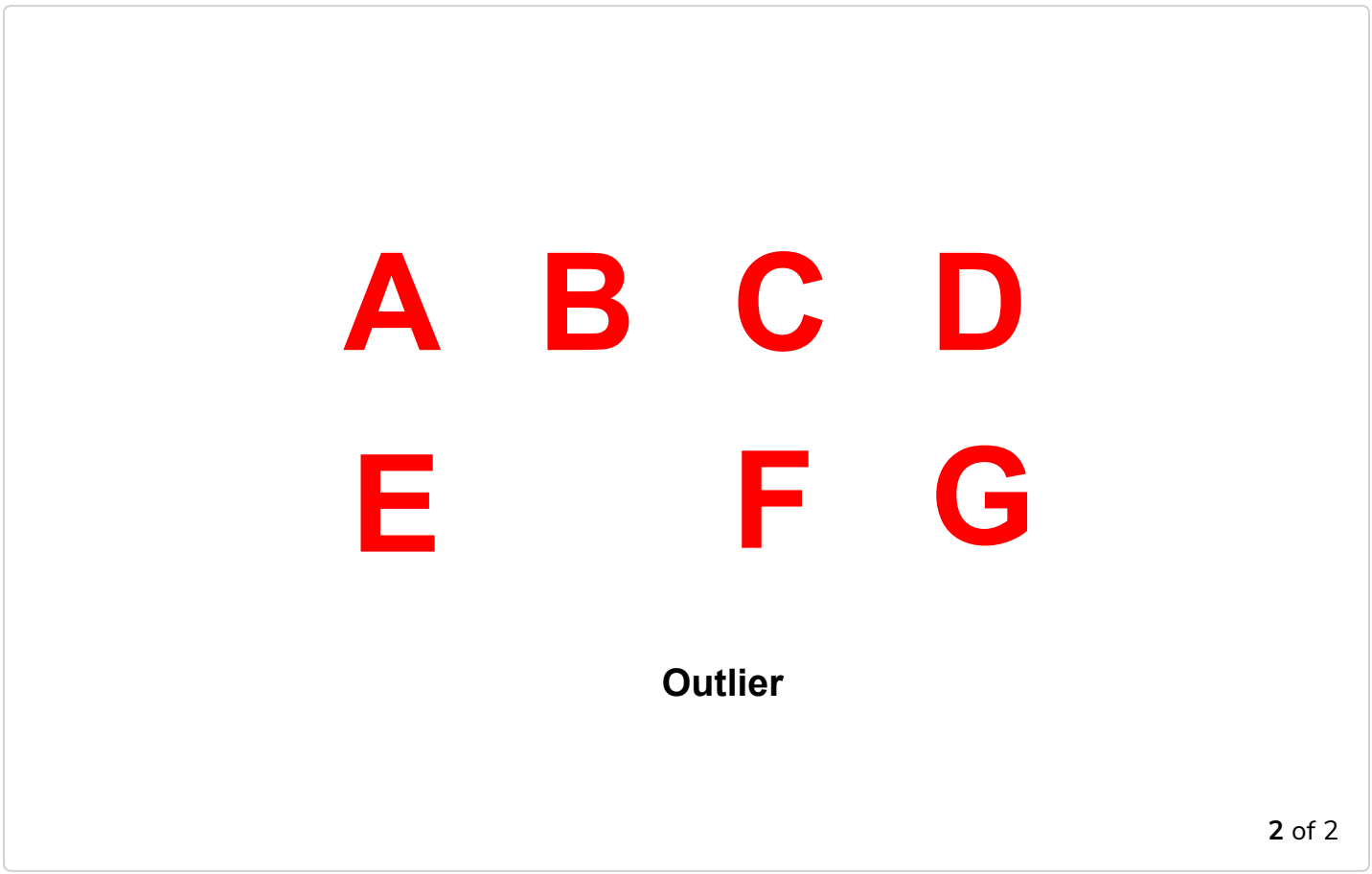
In this lesson, an explanation is provided on what outliers in data are and how to detect them.

We'll cover the following

What is an outlier?

Anything that lies outside the normal distribution of the provided dataset is known as an *outlier*. Let's suppose a list has these elements: `[32,30,39,35,31,4,37]`. It is quite evident that `4` is the outlier in this list because all the other elements lie around a mean value of `35`. Similarly, any data point that behaves differently from the rest of the set is known as an *outlier*.





Why do outliers exist?

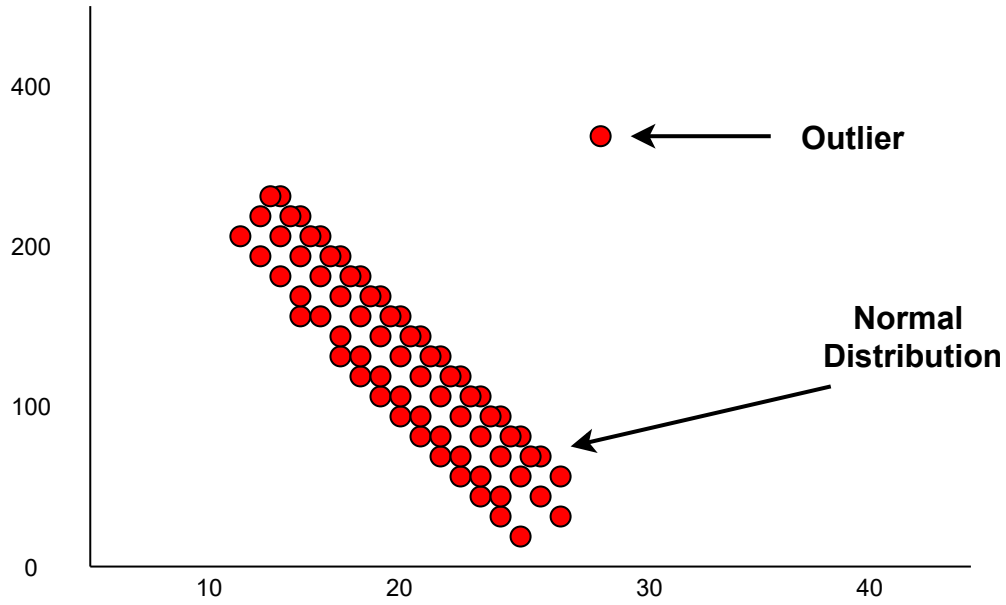
An *outlier* in any dataset mostly exists for the following two reasons:

1. **Variance in data**: There can always be anomalies and ambiguities in data, which can be quite different from the normal distribution.
2. **Entry error**: This occurs mainly due to human error while preparing the dataset or entering values.

Identifying outliers

There are two main methods used to identify outliers in any dataset:

1. **Visualization plots**: The outliers are clearly visible if we plot the data in a scatter, box, or histogram plot, as they are away from the center of the data. More about this will be discussed in the Visualization chapter.



Scatter plot with outlier

2. **Interquartile range (IQR) method**: This method uses the IQR and quantile values to define lower and upper bounds for the data. This method will be used in this lesson to detect outliers.

Interquartile range (IQR) method

The following steps are followed to obtain the *outliers* through *IQR*.

- Sort the data of the list in ascending order.
- Calculate the **1st (Q1)** & **3rd (Q3)** quartile and the **IQR (Q3 - Q1)**
- Calculate the lower bound **(Q1 - (IQR * 1.5))**.
- Calculate the upper bound **(Q3 + (IQR * 1.5))**.
- Any value that lies below the lower bound or above the upper bound is a potential *outlier*.

The `pandas` package provides built-in methods for both `Series` and `DataFrame` to find the *quantiles*. The *IQR*, *lower*, and *upper* bounds have to be calculated manually to check for the *outliers*.

> **Note**: It is advisable to find and remove the outliers column wise and not on

the entire dataframe because different columns can contain different data obtained under different constraints.

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.randn(900,3))

quantiles_df = (df.quantile([0.25,0.75]))

print("The 1st & 3rd quartiles of all columns:")
print(quantiles_df, '\n****************************************')

Q1 = quantiles_df[0][0.25]
Q3 = quantiles_df[0][0.75]

iqr = Q3 - Q1

lower_bound = (Q1 - (iqr * 1.5))
upper_bound = (Q3 + (iqr * 1.5))

print("The Lower bound for the first column:")
print(lower_bound, '\n****************************************')

print("The Upper bound for the first column:")
print(upper_bound, '\n****************************************')

col1 = df[0]

print("The outliers in the first column below the lower bound:")
print(col1[(col1 < lower_bound)], '\n****************************************')

print("The outliers in the first column above the upper bound:")
print(col1[(col1 > upper_bound)], '\n****************************************')
```

- On **line 6**, the `quantile()` method returns a `DataFrame` with the **Q1**, **Q2**, and **Q3** values of all the columns based on the parameters defined in it. The `0.25` gives the **Q1** value, and `0.75` gives **Q3** value.
- On **lines 11 & 12**, the Q1 and Q3 values of the first column from the `quantiles_df` variable are stored in the variables.
- On **line 14**, the **IQR** value is calculated using the above formula.
- On **lines 16 & 17**, the *lower* and *upper* bounds are calculated using the above formula.
- For this example, we are trying to filter outliers for only the first column so, on **line 25**, the first column is stored by reference in the `col1` variable to

avoid any confusion in the syntax.

- On **line 28**, those elements from the first column that are less than the *lower* bound are selected, as they are the outliers for this case.
- On **line 31**, those elements from the first column that are higher than the *upper bound* are selected, as they are the outliers for this case.

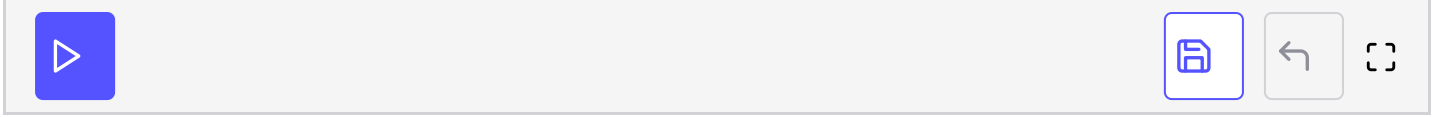
It can be seen from the output that the required outliers are displayed along with their row indexes.

Dealing with outliers

The *outliers* can have negative effects on the results of data and, therefore, need to be taken care of. In this case, we try to bring the outliers into our required range.

The following example demonstrates this task.

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.randn(900,3))

quantiles_df = (df.quantile([0.25,0.75]))

Q1 = quantiles_df[0][0.25]
Q3 = quantiles_df[0][0.75]

iqr = Q3 - Q1

lower_bound = (Q1 - (iqr * 1.5))
upper_bound = (Q3 + (iqr * 1.5))

col1 = df[0]

print("The outliers in the first column below the lower bound:")
print(col1[(col1 < lower_bound)], '\n****************************************')

print("The outliers in the first column above the upper bound:")
print(col1[(col1 > upper_bound)], '\n****************************************')

col1[(col1 < lower_bound)] = lower_bound

col1[(col1 > upper_bound)] = upper_bound

print("After Dealing with Outliers\n")

print("The outliers in the first column below the lower bound:")
print(col1[(col1 < lower_bound)], '\n****************************************')

print("The outliers in the first column above the upper bound:")
print(col1[(col1 > upper_bound)], '\n****************************************')
```



The code before **line 23** is the same. After that, the outliers generated from that output are processed to be inside our defined bounds.

- On **line 24**, the outliers less than the *lower* bound are assigned the *lower* bound value. Now, no value in the column is less than our defined *lower* bound.
- On **line 26**, the outliers greater than the *upper* bound are assigned to the *upper* bound value. Now, no value in the column is greater than our defined *upper* bound.

It can be seen that after dealing with *outliers*, the above code for finding *outliers* displays an empty column.

This is just one of many ways of dealing with outliers. Other techniques of detecting and dealing with outliers can be applied depending on the problem and the dataset.

In the next lesson, data grouping techniques are discussed.