Introduction to Data Science

- Visualizing Data: Matplotlib & Descriptive Statistics -

Mauricio Molina

Keio University, Faculty of Economics

May 7, 2025

Contents

Importing Libraries

To begin, we import the essential libraries for data visualization.

Example Code

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Scatter Plot: Definition and Uses

A scatter plot is a type of data visualization that displays individual data points on a Cartesian coordinate system, where each point represents the values of two variables.

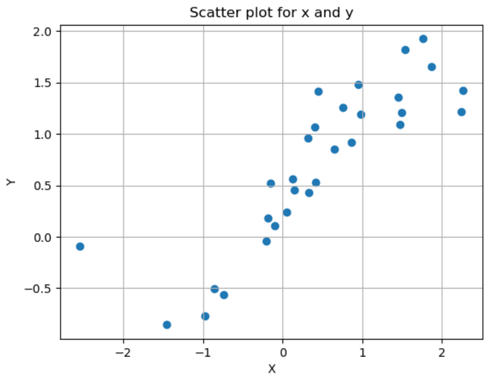
- **Definition:**
  - Each point corresponds to one observation, with its x-value plotted along the horizontal axis and its y-value plotted along the vertical axis.
- **Uses:**
  - **Identifying Relationships:** Useful for detecting correlations and patterns between variables.
  - **Outlier Detection:** Helps in spotting anomalies or unusual observations in the dataset.
  - **Trend Analysis:** Supports visualizing trends, clusters, or the spread of data, which is often a precursor for more detailed statistical analysis.
  - **Informing Decisions:** Serves as a tool for exploratory data analysis that informs hypothesis formulation and further modeling.

Scatter Plot Example

This example creates a scatter plot to display individual data points.

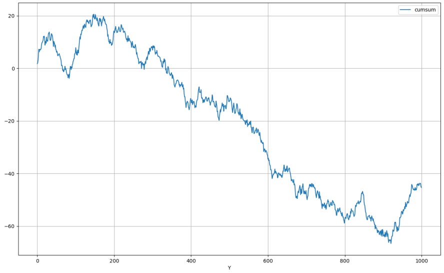
Example Code

```
# Set seed for generating random numbers
np.random.seed(0)

# Define values for x-axis
x = np.random.randn(30)
# Define values for y-axis
y = np.sin(x) + np.random.rand(30)

# Set the graph size
plt.figure(figsize=(20,6))

# Display the graph
plt.plot(x, y, 'o')
# Alternatively, you can use scatter
plt.scatter(x, y)

# Title
plt.title('Scatter plot for x and y')
# Name labels
plt.xlabel('X')
plt.ylabel('Y')
# Show grid
plt.grid(True)
```



Line Plot Example: X vs. Y

If we want to plot a continuous line, we omit the marker option inside the function `plot`.



Example Code

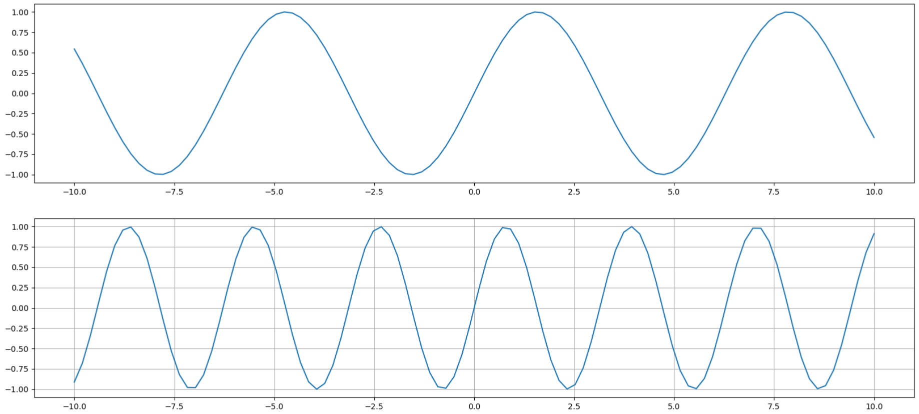
```
# Plot continuous line
np.random.seed(0)

# Domain of the data
x_data = np.arange(1000)
# Rank of data. Cumulative sum of samples of normal distribution.
y_data = np.random.randn(1000).cumsum()

# Plot size
plt.figure(figsize=(15,6))

# Set label as legend in the options of plot
plt.plot(x_data, y_data,label='cumsum')
plt.legend()

plt.xlabel('X')
plt.xlabel('Y')
plt.grid(True)
```

Subplots

We can plot graph several graphs inside one plot, using the `subplot` function.

Example Code

```
plt.figure(figsize=(20,9))

# Define the first subplot of a plot of 2 rows and 1 column
plt.subplot(2,1,1)

# Let's define the domain of out function from -10 to 10 with 100 subintervals
x = np.linspace(-10, 10, 100)
plt.plot(x, np.sin(x))

# The second subplot
plt.subplot(2,1,2)
plt.plot(x, np.sin(2*x))
plt.grid(True)
plt.savefig("output_graph_3.png")
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Histogram: Definition and Utility

A histogram is a graphical representation of data distribution where numerical data is grouped into bins and the frequency or count of data within each bin is displayed as vertical bars.

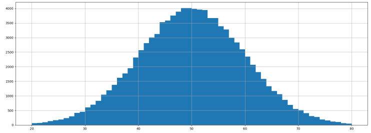
- **Definition:**
  - The x-axis represents the bins (ranges of values).
  - The y-axis shows the frequency or count of observations within each bin.
- **Utility:**
  - **Visualizing Data Distribution:** Quickly assess how data is spread over a range.
  - **Detecting Skewness and Outliers:** Identify asymmetries or unusual observations.
  - **Supporting Decision Making:** Assist in choosing statistical models by revealing data patterns.
  - **Comparing Groups:** Useful for comparing the distributions of different datasets.

Histogram Example

This histogram visualizes the distribution of a set of random data.

Example Code

```
# Fix the seed
np.random.seed(0)

# Define the size of the histogram
plt.figure(figsize=(20, 7))

# Define the data as a sample of a normal distribution with scale 10 and localization 50
data_norm = 10 * np.random.randn(10 ** 5) + 50

# Plot histogram
plt.hist(data_norm, bins = 60, range=(20, 80))

plt.grid(True)
plt.savefig("output_graph_4.png")
```



Main Options of `plt.hist`

The `plt.hist` function in Matplotlib offers several options to customize your histogram. Here are the key parameters:

- **bins**: Sets the number of bins or provides an array of bin edges.
- **range**: Specifies the lower and upper range of the bins.
- **density**: If set to `True`, normalizes the histogram so that the area under the histogram is equal to 1.
- **histtype**: Determines the type of histogram to draw (e.g., `"bar"`, `"barstacked"`, `"step"`, or `"stepfilled"`).
- **color**: Sets the color(s) of the bars.
- **alpha**: Controls the transparency level of the bars.
- **rwidth**: Specifies the relative width of the bars (a value between 0 and 1).
- **orientation**: Chooses the orientation of the bars (`"vertical"` or `"horizontal"`).
- **cumulative**: If set to `True`, computes and plots a cumulative histogram.

Dataset 1: Student Performance

This slide shows the modules used for downloading data from the internet and handling ZIP files.

Example Code

```
# Modules to get data from internet and handling zip files
import requests, zipfile
from io import StringIO
import io
```

This code downloads a ZIP file from the UCI repository and extracts its contents.

Example Code

```
# Set url from which data will be downloaded
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases
 /00356/student.zip'

# Obtain data from the url
r = requests.get(url, stream=True)

# Open zip file
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()
```

This code reads the student performance dataset in CSV format and displays the first 10 rows and summary information.

Example Code

```
student_data = pd.read_csv('student-mat.csv', sep=';')
student_data.head(10)
student_data.info()
```

For detailed information about each column, refer to: Student Performance Data Set Info

Getting basic information about DataSet

You can call methods of Pandas module on filtered data.

Example Code

```
# Getting the first values of the column 'sex'
student_data['sex'].head()

# Getting the first values of the column 'absences'
student_data['absences'].head()

# Compute the mean of 'age' grouped by 'sex'
student_data.groupby('sex')['age'].mean()
```

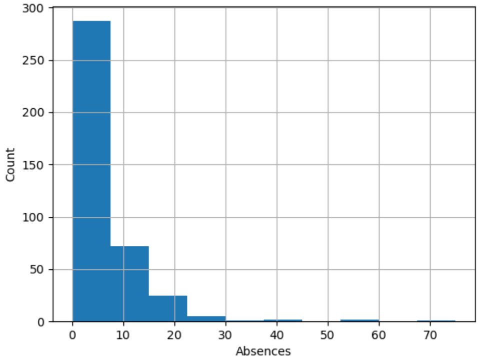
For detailed information about each column, refer to: Student Performance Data Set Info

Histogram for a variable of the DataSet

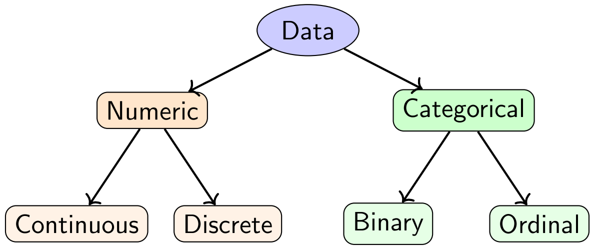
You can plot the corresponding Histogram for a variable in the dataset with the following code:

Example Code

```
# Histogram
plt.hist(student_data['absences'])

# Label each axis
plt.xlabel('Absences')
plt.ylabel('Count')

# Turn on grid
plt.grid(True)
```



Types of Data



Mean, median, mode and percentiles

Pandas also includes methods to compute basic descriptive statistics.

Example Code

```
# Mean
print(f"Mean: {student_data['absences'].mean()}")

# Median
print(f"Median: {student_data['absences'].median()}")

# Mode
print(f"Mode: {student_data['absences'].mode()}")
```

Measures of Center

Measures of center locates the center of the distribution along the horizontal axis.

Definition (Mean)

The **arithmetic mean** or **average** of a set of n measurements is equal to the sum of the measurements divided by n.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Definition (Median)

The **median** m of a set of n measurements is the value of x that falls in the middle position when the measurements are ordered from smallest to largest.

Another way to locate the center of a distribution is to look for the value of x that occurs with the highest frequency. This measure is called the **mode**.

Definition (Mode)

The **mode** is the category that occurs most frequently, or the most frequently occurring value of x. When measurements on a continuous variable have been grouped as a frequency or relative frequency histogram, the class with the highest peak or frequency is called the **modal class**, and the midpoint of that class is taken to be the mode.

The mode is generally used to describe large data sets, whereas the mean and median are used for both large and small data sets. Sometimes bimodal distributions of sizes or weights reflect a mixture of measurements taken on males and females. In any case, a set or distribution of measurements may have more than one mode.

Variance and Standard Deviation

Data sets may have the same center but look different because of the way the numbers *spread out* from the center. Variability is a very important characteristic of data. For example, if you were manufacturing bolts, extreme variation in the bolt diameters would cause a high percentage of defective products. Measures of variability can help you create a mental picture of the spread of the data.

Example Code

```
# Variance
print(f"Variance: {student_data['absences'].var()}")

# Standard Deviation
print(f"Standard Deviation: {student_data['absences'].std()}")
```

Measures of center locates the center of the distribution along the horizontal axis.

Definition (Variance)

The **variance** of a sample of n measurements is the sum of the squared deviations of the measurements about their mean $\bar{x}$ divided by $(n-1)$. The sample variance is denoted by s^2 and is given by the formula

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Definition (Standard Deviation)

The **standard deviation** of a set of measurements is equal to the positive square root of the variance.

$$s^2 = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Percentiles

A **percentile** (or quantile) is another measure of relative standing, most often used for large data sets.

Definition (Percentile)

A set of n measurements on the variable x has been arranged from smallest to largest. The p-th percentile ($= (p/100)$ quantile) is the value of x that is greater than $p\%$ of the measurements and is less than the remaining $(100 - p)\%$.

Example Code

```
# 10%, 25%, 50%, 75% and 90% percentile
print(f"Percentiles at: \n{student_data['absences'].quantile([0.1, 0.25, 0.50, 0.75, 0.9])}")
```

Definition

A set of n measurements on the variable x has been arranged from smallest to largest. The **lower quartile** (first quartile), Q_1, is the value of x that is greater than one-fourth of the measurements and is less than the remaining three-fourths. The **second quartile** is the **median**. The **upper quartile** (third quartile), Q_3, is the value of x that is greater than three-fourths of the measurements and is less than the remaining one-fourth.

Example Code

```
# Show the quartiles and more info
student_data['absences'].describe()
```

Definition

The **interquartile range (IQR)** for a set of measurements is the difference between the upper and lower quartiles; that is, $IQR = Q_3 - Q_1$.

Example Code

```
# Calculate the IQR
student_data['absences'].describe().iloc[6]-student_data['absences'].describe().iloc[4]
```

Five-number summary

Definition

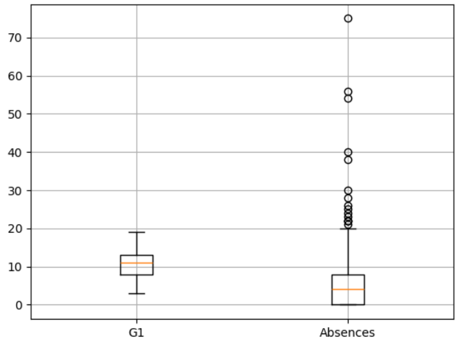
The **five-number summary** consists of the following numerical measures:

$$\text{min} \quad Q1 \quad \text{med} \quad Q3 \quad \text{max}$$

By definition, one-fourth of the measurements in the data set lie between each of the four adjacent pairs of numbers.
The five-number summary can be used to create a simple graph called a box plot(box-whisker plot) to visually describe the data distribution.

Let's note that the *pandas* function *pd.describe()* compute this summary.

- From the box plot, you can quickly detect any skewness in the shape of the distribution and see whether there are any outliers in the data set.
- An outlier may result from transposing digits when recording a measurement, from incorrectly reading an instrument dial, from a broken piece of equipment, or from other problems.
- Even when there are no recording errors, a data set may contain one or more measurements that are very different from the others in the set.
- These outliers can cause a distortion in commonly used numerical measures such as $\bar{x}$ and s.
- In fact, outliers may themselves contain important information not shared with the other measurements in the set.



Example Code

```
# Box plot for variables G1 and absences
plt.boxplot([student_data['G1'], \
 student_data['absences']], \
 labels=['G1', 'Absences'])
plt.grid(True)
plt.savefig("output_graph_6.png")
```

Isolating outliers, if they are present, is an important first step in analyzing a data set. The box plot is designed exactly for this purpose.

Box Plot: Outlier Detection Using IQR Fences

The box plot uses the interquartile range (IQR) to create imaginary "fences" that help separate outliers from the rest of the data:

- **Lower fence:** $Q_1 - 1.5 \times IQR$
- **Upper fence:** $Q_3 + 1.5 \times IQR$

Note: The fences are conceptually shown with lines in illustrative figures, but they are not typically drawn on a standard box plot.

Assignment Questions

Solve the following problems in your Jupyter Notebook. Ensure all code runs without errors. Upload the completed `.ipynb` file to K-LMS by next week's tuesday at midnight.

Q1: Download the Kaggle's Car Price Prediction Multiple Linear Regression dataset at:
`https://www.kaggle.com/datasets/hellbuoy/car-price-prediction`.

- Load to Python using Pandas.
- Print the first ten observations.
- Obtain all the info for the columns.

Q2: With the previous DataSet:

- Compute the five-number summary for all the applicable columns,
- Calculate the mean, median, variance, std, max and minimum value for the variable price. Compute the *IQR* for this variable.
- Calculate the same statistics of above separating the cars by number of doors.
- Draw a histogram for the variable price.

Q3: With the previous DataSet:

- Draw a box plot for the variable price.
- Explain if there is any outlier observation in the variable price.
- Draw a scatter plot for variables Engine and price.