

Statistics for data science:

This section will cover the **foundational statistics** needed for effective data science. Understanding core statistical principles enables data scientists to interpret data more accurately, identify trends, and make predictions. We will dive into the following topics:


1. Central Tendency Measures
2. Measures of Spread
3. Data Distribution
4. Correlation Analysis
5. Sampling Techniques

The dataset we will use in this section is the **World Happiness Score for 2023**:

Dataset context: The World Happiness Report dataset for 2023 offers a comprehensive examination of happiness metrics and the factors influencing well-being on a global scale. This dataset is designed to provide valuable insights for policymakers, researchers, and individuals interested in understanding the dynamics of happiness and well-being worldwide.

```
In [1]: import pandas as pd
df = pd.read_csv("data/WHR_2023.csv")
df.head()
```

Out[1]:	country	region	happiness_score	gdp_per_capita	social_support	healthy_life_ex
0	Finland	Western Europe	7.804	1.888	1.585	
1	Denmark	Western Europe	7.586	1.949	1.548	
2	Iceland	Western Europe	7.530	1.926	1.620	
3	Israel	Middle East and North Africa	7.473	1.833	1.521	
4	Netherlands	Western Europe	7.403	1.942	1.488	

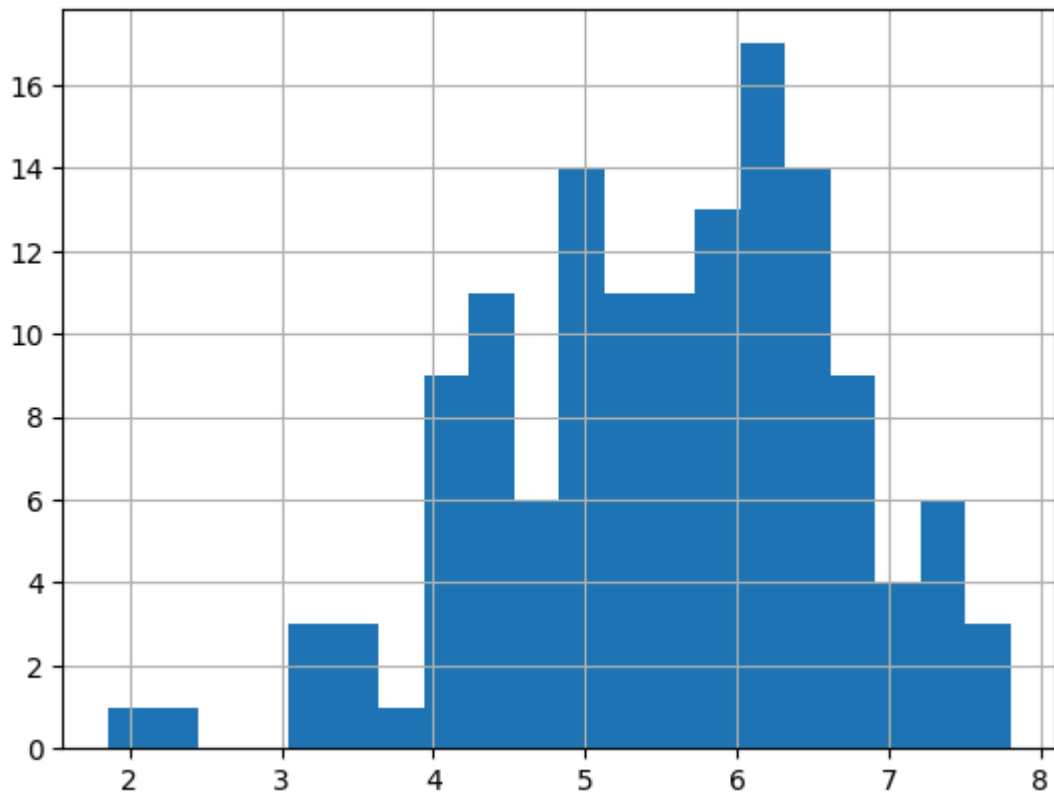


Let's start by a simple question:

Describe the distribution of happiness score in the world?

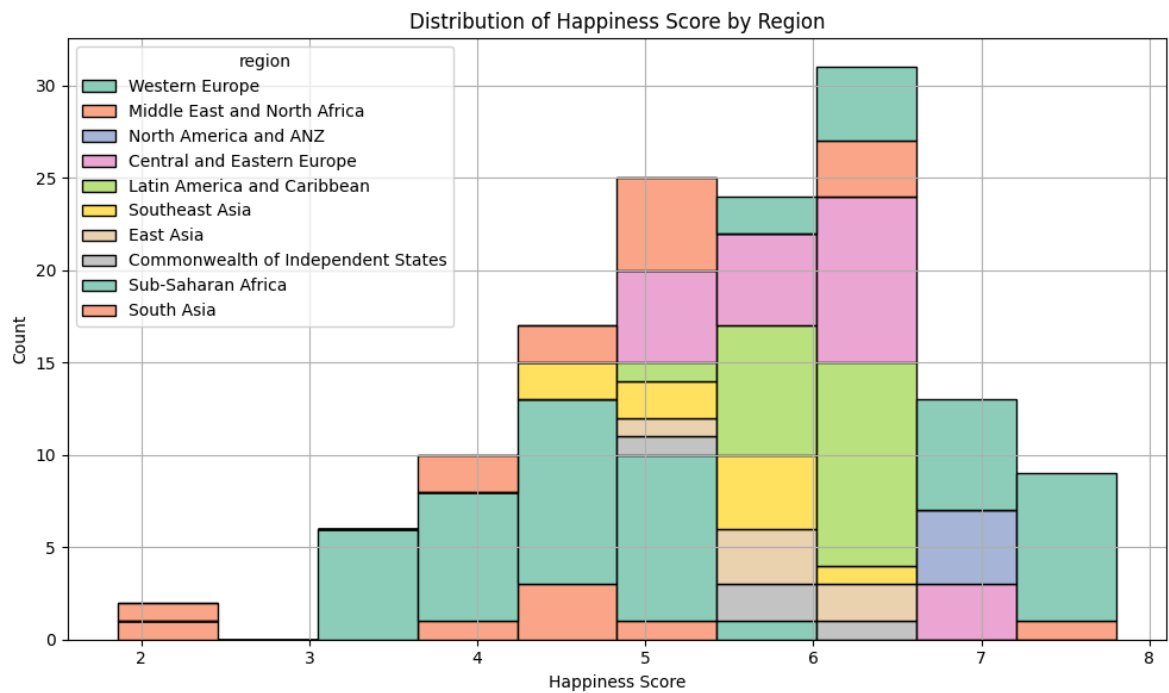
As we learned in previous sections, we can plot a **histogram** that can show the distribution of a Pandas column.

```
In [5]: df['happiness_score'].hist(bins=20)
```



```
In [14]: import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='happiness_score', hue='region', multiple='stack', palette='magma')
plt.title('Distribution of Happiness Score by Region')
plt.xlabel('Happiness Score')
plt.ylabel('Count')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Now that we have the histogram, the question becomes:

How can we accurately describe its distribution using statistical measures?

The answer lies in two main categories of statistical measures:

Describe the distribution by:

1. Measures of Central Tendency

These tell us about the central or "typical" value around which data points tend to cluster. Common measures include the mean, median, and mode.

2. Measures of Spread

These quantify the variability within the dataset, indicating how spread out or close together data points are around the central tendency. Common measures of spread include the range, variance, standard deviation, and quartiles.

By combining measures of central tendency and spread, we gain a more complete understanding of the data's distribution. This combination helps us answer key questions such as where the data is centered, how much variation exists, and how individual data points compare to the overall dataset.

Central tendency measures

Central tendency measures describe the center or average of a data set. The main measures include the **mean**, **median**, and **mode**.

Mean

The mean is a measure of central tendency, calculated by **summing all data values and dividing by the count of observations**. It is commonly referred to as the **arithmetic mean** to distinguish it from other types of averages, such as geometric and harmonic means.

For a dataset with n values x_1, x_2, \dots, x_n the arithmetic mean μ (for a population) or \bar{x} (for a sample) is calculated as:

$$\text{Mean } (\mu \text{ or } \bar{x}) = \frac{1}{n} \sum_{i=1}^n x_i$$

Types of Means:

- **Population mean** (μ): Calculated when data includes every member of a **population**.
- **Sample mean** (\bar{x}): Calculated from a subset (**sample**) of the population. The sample mean is used to estimate the population mean.

Advantage:

Easy to calculate and understand.

But

Highly influenced by **extreme values**, potentially giving a misleading central tendency in skewed distributions

Influenced by extreme values:

Suppose we have the salary data of employees in two companies:

- **Company A:** A small tech startup where salaries are more evenly distributed.
 - Salaries: [3000, 3200, 3500, 4000, 4500]\$
 - Mean: 3650
- **Company B:** A large corporation with some very high executive salaries.
 - Salaries: [3000, 3200, 3500, 4000, 50000]\$
 - Mean: 12740

In **Company A**, all employees earn between \$3000 and \$4500, reflecting a more uniform salary structure. In Company B, four employees earn typical salaries, but one executive

has a significantly higher salary of \$50000, creating an extreme value.

When calculating the mean for both companies, **Company B** has mean salary of \$12740, which is significantly higher than the majority of salaries and doesn't accurately reflect what most employees earn. This difference is due to the outlier (the \$50000 salary).

The inflated mean in **Company B** might give a **misleading** impression of higher typical salaries if we only report the mean.

How to calculate mean on Pandas?

```
In [3]: df['happiness_score'].mean()
```

```
Out[3]: 5.539795620437956
```

How can we interpret a mean world happiness score of 5.5?

Since the mean provides us with a central or "average" of the data then a mean happiness score of 5.5 (on a scale from 0 to 10) suggests that the overall happiness level is slightly above the midpoint, indicating that, on average, people around the world rate their happiness as moderate.

Median

The **median** is a measure of central tendency that represents the middle value in a sorted list of numbers. It's especially useful in datasets that may **contain outliers**, as the median is **resistant to extreme values**.

How to calculate the median?

1. For odd numbers of data points:

- When there's an odd number of data points, the median is the middle value in the sorted list.
- Example: For the dataset [3000 , 3500 , 4000], the median is 3500 , as it's the central value.

2. For even numbers of data points:

- When there's an even number of data points, the median is the average of the two middle values.
- Example: For the dataset [3000, 3500, 4000, 4500], the median is calculated as:
$$(3500 + 4000)/2 = 3750$$

Returning to the previous salary example, let's see how the median compares across two companies, one with and one without outliers.

Suppose we have the salary data of employees in two companies

- **Company A:** A small tech startup where salaries are more evenly distributed.
 - Salaries: [3000, 3200, 3500, 4000, 4500]\$
 - Median: 3500
- **Company B:** A large corporation with some very high executive salaries.
 - Salaries: [3000, 3200, 3500, 4000, 50000]\$
 - Median: 3500

Here, even though Company B has an extreme salary of 50000 , the median (3500) remains the same as for Company A. This highlights that the median is less influenced by outliers and better represents the typical salary in **Company B** than the mean does.

How to calculate median in Pandas?

```
In [4]: df['happiness_score'].median()
```

```
Out[4]: 5.684
```

Mode

The **mode** is a measure of central tendency that represents the **most frequently occurring value(s)** in a dataset.

Unlike the mean and median, which rely on averaging or positioning within a sorted list, the mode simply identifies the value that appears most often. This makes the mode particularly useful for categorical data but applicable to numerical data as well.

The **mode** is valuable in situations where we need to identify the most common category or preference. It is often used in fields like **marketing**, **economics**, and **psychology** to determine trends or the most popular choices within a group.

How to calculate mode in Pandas?

Python makes it easy to calculate the mode using libraries like Pandas:

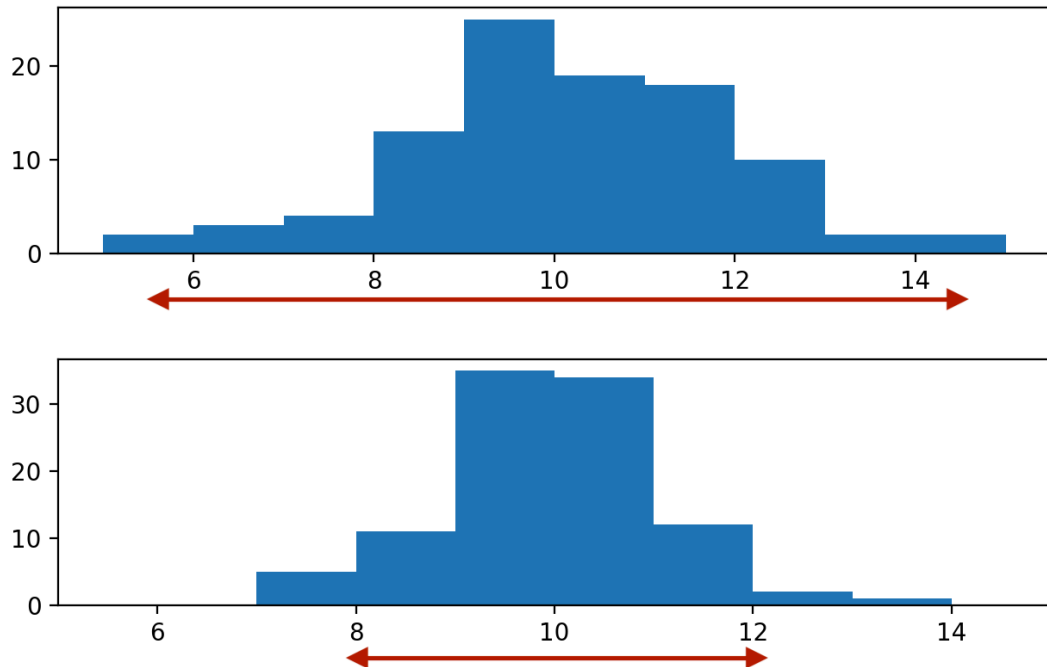
```
In [5]: df['region'].mode()
```

```
Out[5]: 0    Sub-Saharan Africa  
Name: region, dtype: object
```

This indicates that the Sub-Saharan region is the most commonly occurring region in the dataset, meaning it has the highest number of countries represented. Since this data includes happiness scores for all countries worldwide in a specific year, this suggests that Sub-Saharan Africa has more countries than other regions in the dataset.

Measures of spread

While **central tendency** (mean, median, and mode) provides information about the “**center**” or typical values in the data, it doesn’t account for other important aspects of the dataset’s distribution, such as **variability** or **shape**.

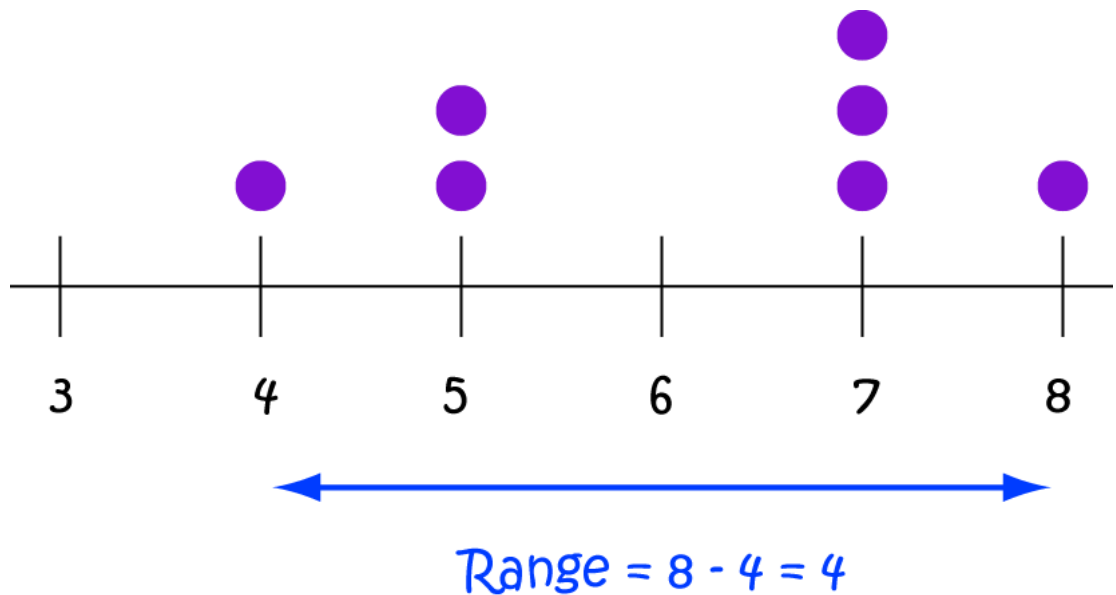


Spread measures describe how much variability there is in the data. Key measures include:

- **Range:** The difference between the maximum and minimum values.
- **Variance:** The average squared deviation from the mean.
- **Standard Deviation:** The square root of variance, providing spread in the same units as the data.
- **Quartiles:** The spread of the data in four equal parts

Range

The **range** is the simplest measure of spread, representing the **difference** between the **largest** and **smallest** values in a dataset. It provides a basic idea of the overall spread of data, showing **how far the extremes are from each other**.



What is the range of world happiness score?

```
In [6]: print(f"Range of happiness score is [{df['happiness_score'].min()} , {df['happiness_score'].max()} - df['happiness_score'].min()"])
```

Range of happiness score is [1.859 , 7.804]

Out[6]: 5.945

Limitations of the range:

- The range can be misleading if there are **extreme values/outliers**.

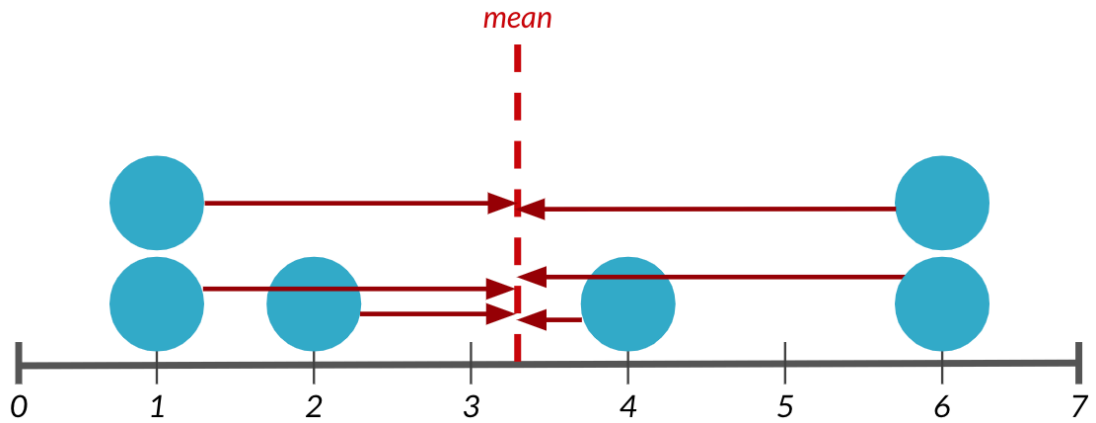
For example, if we add an unusually high score (e.g., 99) to the dataset, the range would increase significantly, even if most scores are close to each other.

- Range only considers the two extreme values and **ignores how data points are distributed between them**.

Because of these limitations, the range is often supplemented with other, more robust measures of spread, such as the **variance** and **quartiles**, which consider all data points to give a fuller picture of variability

Variance

Variance quantifies how spread out the values in a dataset are from the mean by calculating the **average of the squared differences** between each data point and the mean. **Unlike the range**, which only considers the extremes, variance **takes all data points into account**, giving a comprehensive picture of variability.



Formula:

For a dataset with n values x_1, x_2, \dots, x_n and mean μ :

$$\text{Variance (Population)} = \sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

For sample variance, we divide by $n-1$ instead of n :

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

Example:

Consider the following dataset representing the number of hours students studied per week: **5,7,8,4,9**

1. **Mean** $= \bar{x} = \frac{5+7+8+4+9}{5} = 6.6$

2. **Squared Differences:**

- $(5 - 6.6)^2 = 2.56$
- $(7 - 6.6)^2 = 0.16$
- $(8 - 6.6)^2 = 1.96$
- $(4 - 6.6)^2 = 6.76$
- $(9 - 6.6)^2 = 5.76$

3. **Variance (Sample):**

$$s^2 = \frac{2.56 + 0.16 + 1.96 + 6.76 + 5.76}{4} = 4.3$$

So, the sample variance for this dataset is 4.3.

Interpretation:

- **Higher variance** means greater variability, indicating data points are spread out from the mean.
- **Lower variance** indicates data points are closer to the mean, showing less variability.

Limitations of the variance:

- Sensitive to outliers

Squaring the differences amplifies the impact of extreme values, potentially skewing the variance.

- Unit interpretation

The variance for the quiz 1 score in the class is 9

What is the unit of the 9?

The average squared difference in quiz scores is 9 points².

Since variance is in squared units, it can be difficult to interpret directly without taking its square root

How to calculate variance in Pandas?

The code below shows how to calculate variance using Pandas:

```
In [7]: df['happiness_score'].var()
```

```
Out[7]: 1.299437546157149
```

Standard Deviation

Standard deviation is a measure of spread that addresses the interpretational challenge posed by variance. While variance gives the average squared deviation from the mean, it is expressed in squared units, making it hard to intuitively grasp what that variability means in the context of the original data.

- For a sample:

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

- For a population:

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

Interpretation:

Standard deviation provides an estimate of how much individual data points deviate from the mean in the original units of the dataset. If the standard deviation is small, the data points are close to the mean; if it is large, data points are more spread out.

For instance, if we calculate the standard deviation of heights in centimeters, **a standard deviation of 7.91 cm** means that, on average, heights vary by about **7.91 cm from the mean**.

How to calculate standard deviation in Pandas?

The code below shows how to calculate standard deviation using Pandas:

```
In [8]: df['happiness_score'].std()
```

```
Out[8]: 1.139928746087732
```

Z-score

What is your best and worst performance ?

Test	Mean	Standard Deviation	Your Score
<i>Math</i>	82	6	80
<i>Verbal</i>	75	3	75
<i>Science</i>	60	5	70
<i>Logic</i>	70	7	77

The **z-score**, also known as the standard score, tells us **how many standard deviations a particular data point is from the mean of the dataset**.

Formula for Z-Score:

The z-score of a value x is calculated as:

$$z = \frac{x - \text{mean}}{\text{standard deviation}}$$

Question:

The mean score for a science exam is 72, with a standard deviation of 4. Your score on the exam is 80.


- How many standard deviations above the mean is your score?
- If you had a score of 70, how many standard deviations below the mean would your score be?

Create a new column containing the z-score for each 'Happiness Score' value ?

```
In [9]: df['z_happiness_score'] = (df['happiness_score'] - df['happiness_score'].mean())
df.tail()
```

Out[9]:

	country	region	happiness_score	gdp_per_capita	social_support	healthy_life_
132	Congo (Kinshasa)	Sub-Saharan Africa	3.207	0.531	0.784	
133	Zimbabwe	Sub-Saharan Africa	3.204	0.758	0.881	
134	Sierra Leone	Sub-Saharan Africa	3.138	0.670	0.540	
135	Lebanon	Middle East and North Africa	2.392	1.417	0.476	
136	Afghanistan	South Asia	1.859	0.645	0.000	



Quartiles

Quartiles are values that divide a dataset into equal parts. By separating data into quartiles, we can identify specific portions of the data distribution, providing insight into **how data points are spread across different ranges.**

Quartiles help summarize the distribution and identify regions with high or low concentrations of data:

- Quartiles help detecting extreme values outside the normal range.
- It helps understand variation across different segments of the data.

Quartiles: It is one type of quantiles that divide data into four equal parts (each representing 25% of the data):

- Q_1 (1st Quartile): 25th percentile, where 25% of data points fall below this value.
- Q_2 (2nd Quartile): 50th percentile, which is also the median (half of data points are below and half are above).
- Q_3 (3rd Quartile): 75th percentile, where 75% of data points fall below this value.

Example calculation of quartiles:

For a sample sorted dataset: [4,7,9,13,16,18,20,24,26]

```
In [7]: import numpy as np

np.quantile([4,7,9,13,16,18,20,24,26], [0.25, 0.5, 0.75]) #Using Numpy
```

Out[7]: array([9., 16., 20.])

- Q_1 (25th percentile): Value below which 25% of the data falls is 9.
- Q_2 (50th percentile / median): Middle value, which is 16.
- Q_3 (75th percentile): Value below which 75% of the data falls is 20.

How to calculate quartiles in Pandas?

Here's a quick demonstration of calculating quartiles using numpy and Pandas:

```
In [10]: import numpy as np

np.quantile(df['happiness_score'], [0.25, 0.5, 0.75, 0.8, 0.1]) #Using Numpy
```

```
Out[10]: array([4.724 , 5.684 , 6.334 , 6.489 , 4.1084])
```

```
In [11]: df['happiness_score'].describe() #Using Pandas
```

```
Out[11]: count    137.000000
mean         5.539796
std          1.139929
min          1.859000
25%          4.724000
50%          5.684000
75%          6.334000
max          7.804000
Name: happiness_score, dtype: float64
```

How we can interpret the above statistical measures?

Let's break down these statistical measures to get a better sense of the global distribution of happiness:

- **Count of 137:** This indicates that there are 137 countries in the dataset with recorded happiness scores.
- **Mean of 5.54:** The average happiness score across these 137 countries is about 5.54. This suggests that, globally, the "typical" happiness level is just above the midpoint of the possible happiness score range, indicating a moderate average.
- **Standard Deviation of 1.14:** This measure shows how much the scores vary around the mean. A standard deviation of 1.14 indicates that happiness scores across countries are moderately spread out. Scores that fall within one standard deviation of the mean (approximately 5.54 ± 1.14) between 4.40 and 6.68) account for a significant portion of the data.
- **Minimum of 1.86:** The lowest recorded happiness score is 1.86, suggesting that some countries report very low levels of happiness, well below the global average.
- **25% (First Quartile - 4.72):** This quartile indicates that 25% of countries have happiness scores of 4.72 or lower, showing that a substantial portion of countries score below the mean.

- **Median (50% - 5.68):** The median happiness score is 5.68, which is close to the mean. This suggests a relatively symmetric distribution of scores, as the average and middle values are close.
- **75% (Third Quartile - 6.33):** This measure shows that 75% of countries have happiness scores below 6.33, with only the top 25% scoring above this.
- **Maximum of 7.80:** The highest happiness score in the dataset is 7.80, representing the happiest country.

Interquartile Range (IQR)

The **Interquartile Range (IQR)** is another important measure of spread that focuses specifically on the middle 50% of a dataset. It represents the range within which the central half of the data lies. It is calculated as the difference between the third quartile (Q_3) and the first quartile (Q_1):

$$IQR = Q_3 - Q_1$$

How IQR is useful?

- **Resilience to outliers**

IQR is not affected by extreme values, unlike the range or even the standard deviation.

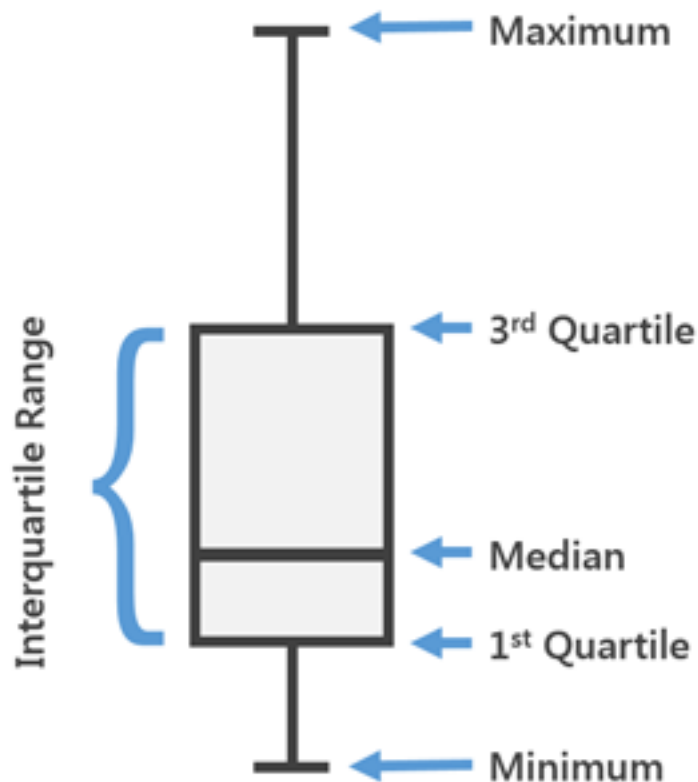
- **Identifying outliers**

Values are typically considered **outliers** if they lie:

- Below $Q_1 - 1.5 \times IQR$, or
- Above $Q_3 + 1.5 \times IQR$

Boxplot:

A **boxplot** is a helpful visual tool for summarizing the distribution of a dataset and identifying key measures, such as the Interquartile Range (IQR) and potential outliers.



Key components of a boxplot:

- **Box:**

The main part of the boxplot is the rectangular box, which represents the *IQR* (Interquartile Range) — the range between the Q_1 and Q_3 . This box shows where the middle 50% of the data lies, giving insight into the central spread of the data.

- **Median Line:**

Inside the box, a line represents the median (Q_2), showing the middle of the dataset.

- **"Whiskers":**

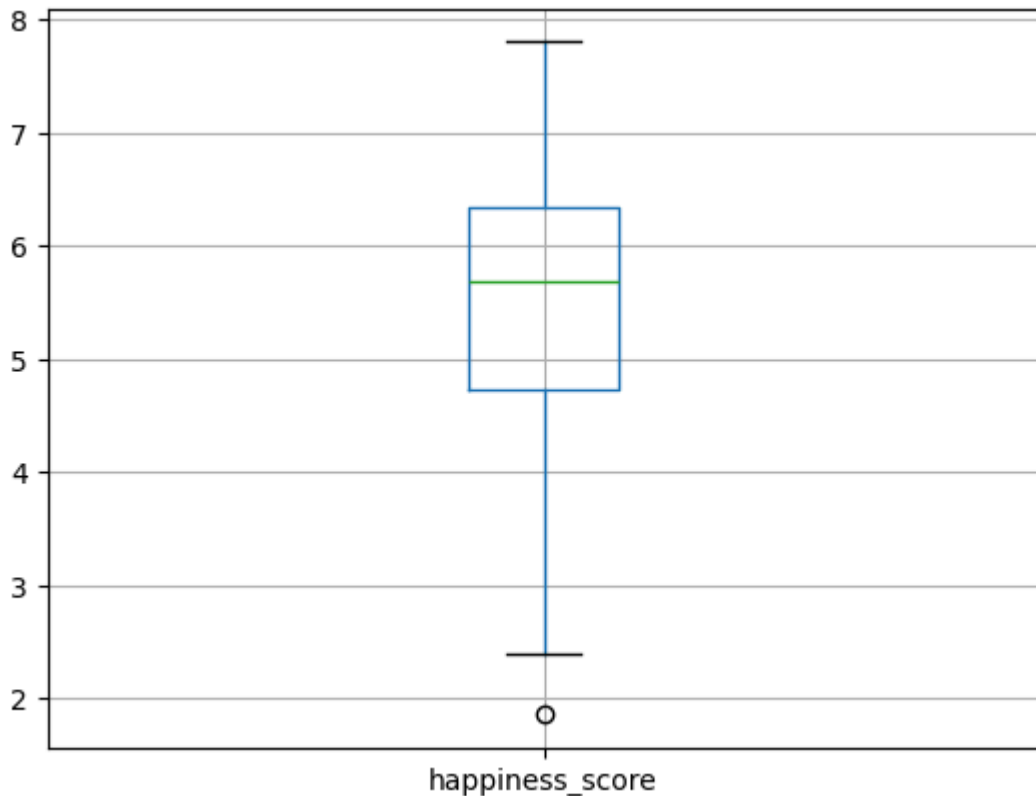
Lines extending from the box (whiskers) go to the smallest and largest values within a defined range. The whiskers typically extend to 1.5 times the *IQR* above Q_3 and below Q_1 . This range is where most data points are expected to fall if they're not outliers.

- **Outliers:**

Points that fall outside the range of the whiskers (either above $Q_3 + 1.5IQR$ or below $Q_1 - 1.5IQR$) are considered outliers and are often shown as individual dots. These points highlight values that are unusually high or low compared to the rest of the data.

Example: Boxplot for happiness score

```
In [12]: df.boxplot(['happiness_score']);
```



Shape of the distribution

Visualizing the distribution is a key step to better understand the shape and characteristics of your data beyond just numeric summaries. By plotting the data, we can observe not only where the data clusters (central tendency) and how it spreads (dispersion) but also its shape, which can reveal patterns such as **skewness** and **kurtosis**.

Distribution plots

- **Numerical data:**

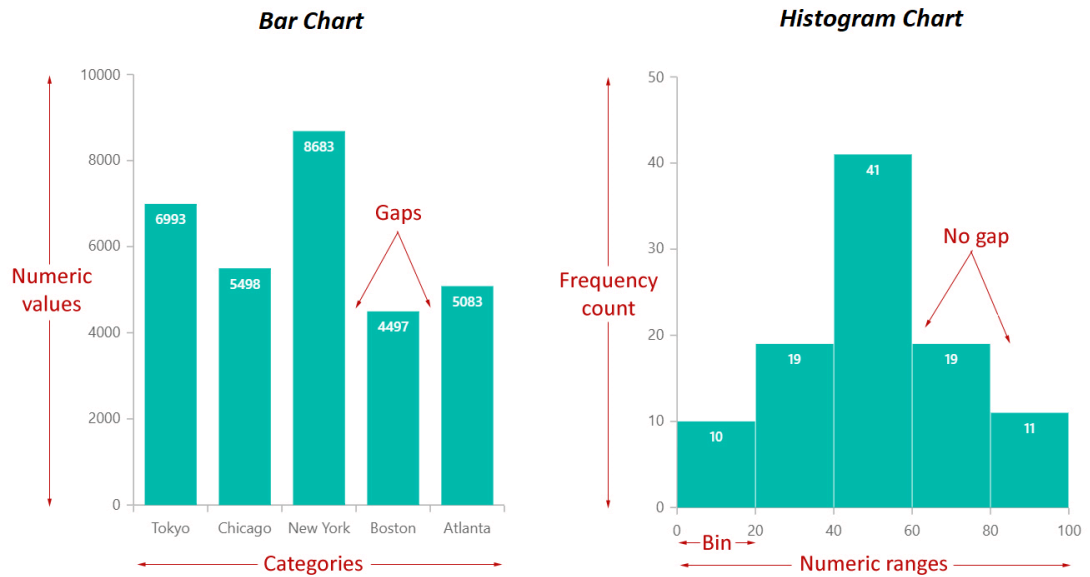
Histogram

Histogram is an effective way to view the frequency distribution of continuous data. This graph uses bars to show the frequency of data points within specified ranges (bins).

- **Categorical data:**

Count/Bar plot

For discrete data, a **count plot** is more suitable. This chart displays the frequency of each category in a dataset, helping us compare categorical occurrences.

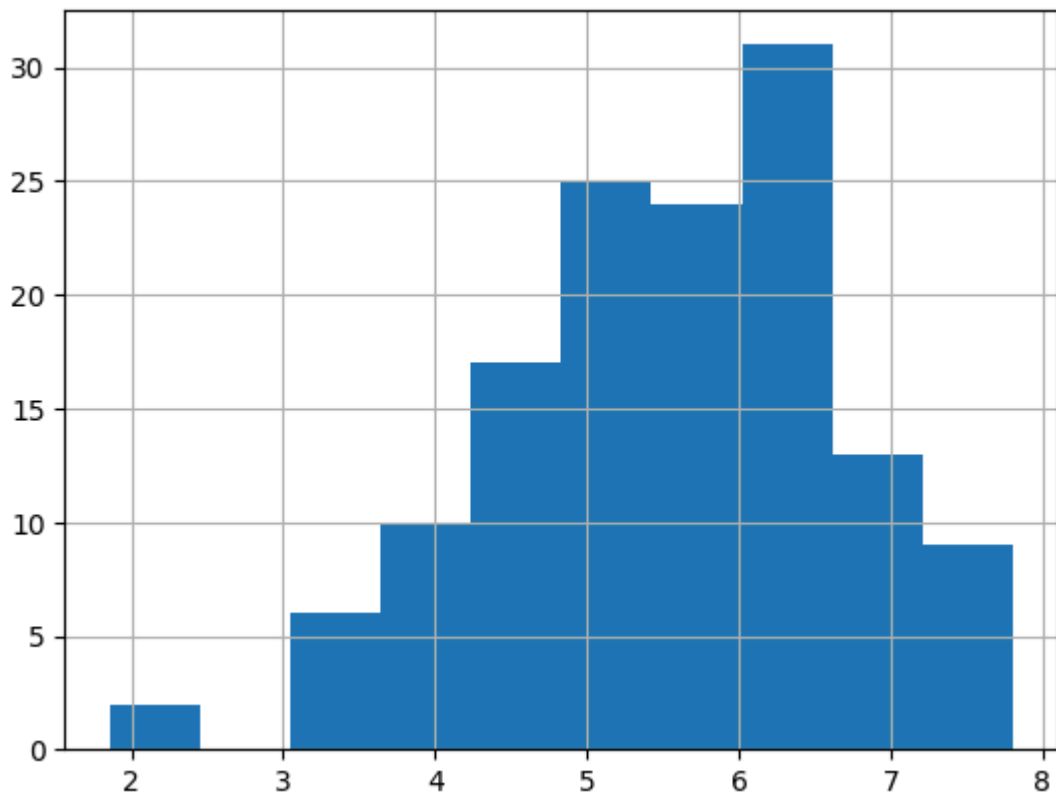


Example:

Consider the dataset of **World Happiness Scores**. For numerical happiness scores, a histogram shows how many countries fall within each happiness range. For categorical data like regions, a count plot displays the number of countries per region.

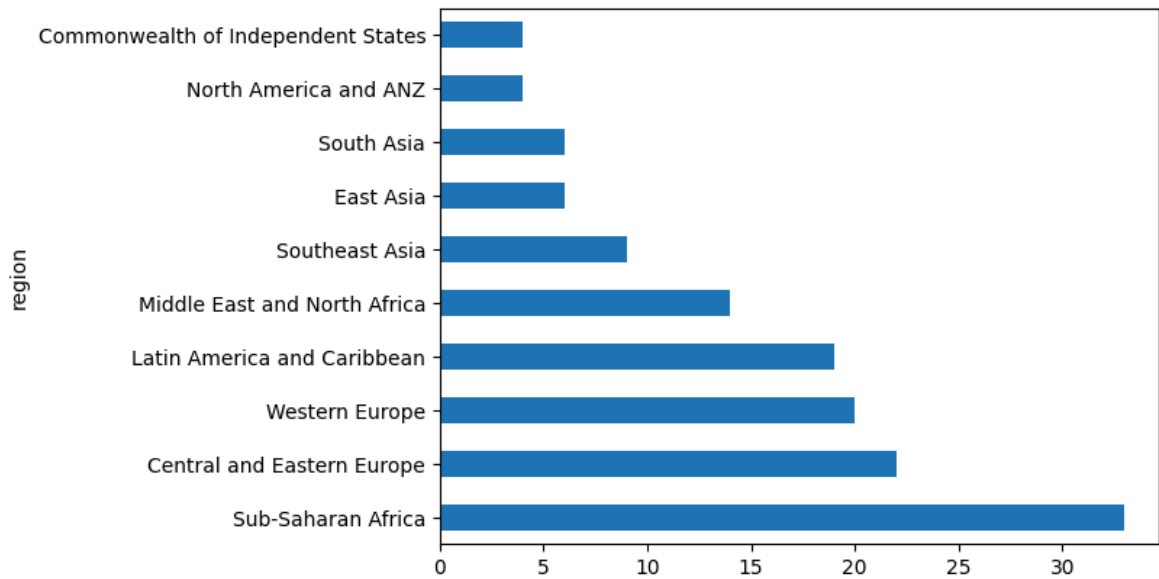
- **Happiness score**

```
In [13]: df['happiness_score'].hist();
```



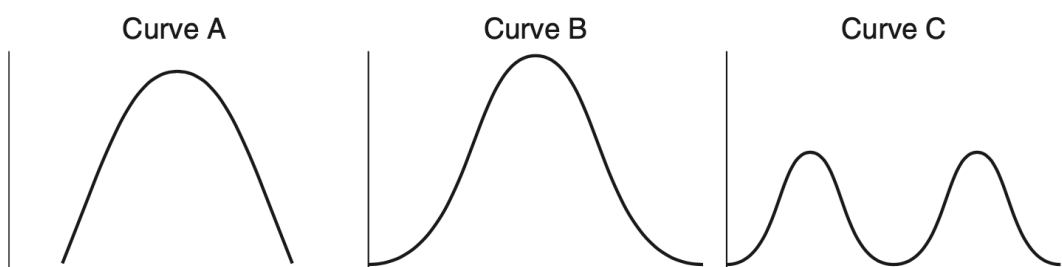
- **Region:**

```
In [14]: df['region'].value_counts().plot(kind = 'barh');
```



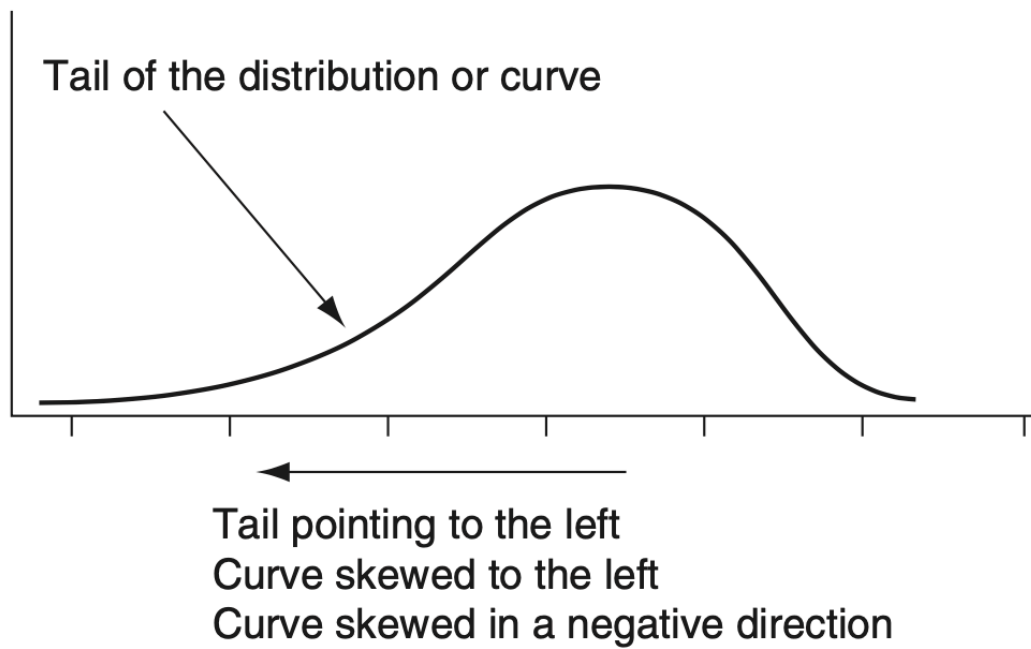
Skewed vs. Symmetrics distribution

- **Symmetric distribution:** is one in which the two halves of the distribution are mirror images of each other.
- Data is evenly spread around the center.
- The histogram shows a balanced shape, where both sides mirror each other.
- Mean \approx Median
- Example: Heights, weights



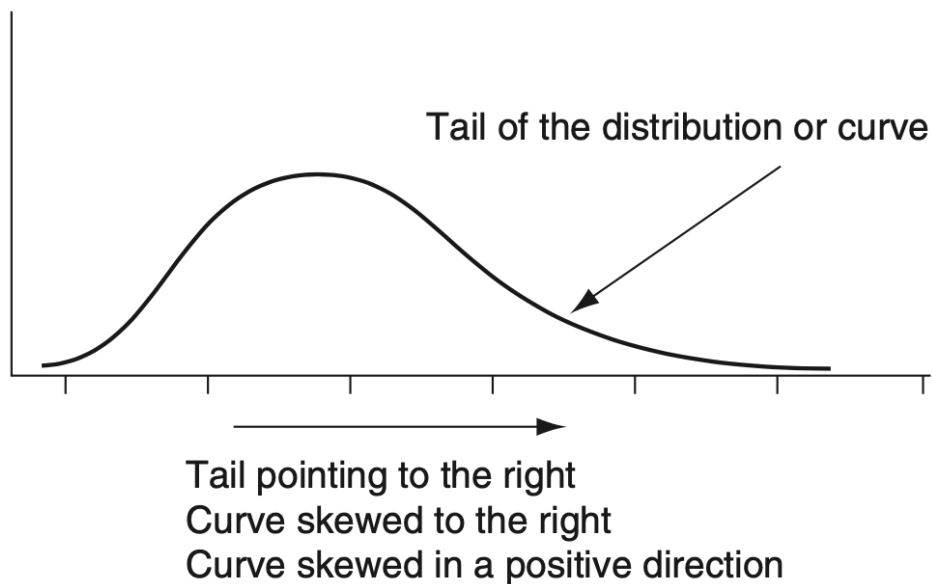
- **Skewed Distributions:** When a dataset isn't symmetric, it is considered skewed. It shows if data leans more towards one side of the mean

1. Left (negative) skewed:



- The left tail of the distribution is longer, and most data points are concentrated on the higher end.
- $\text{Mean} < \text{Median}$, as the mean is influenced by extreme low values.
- Example: Age at retirement in a dataset where some people retire very early.

- **Right (positive) skewed:**

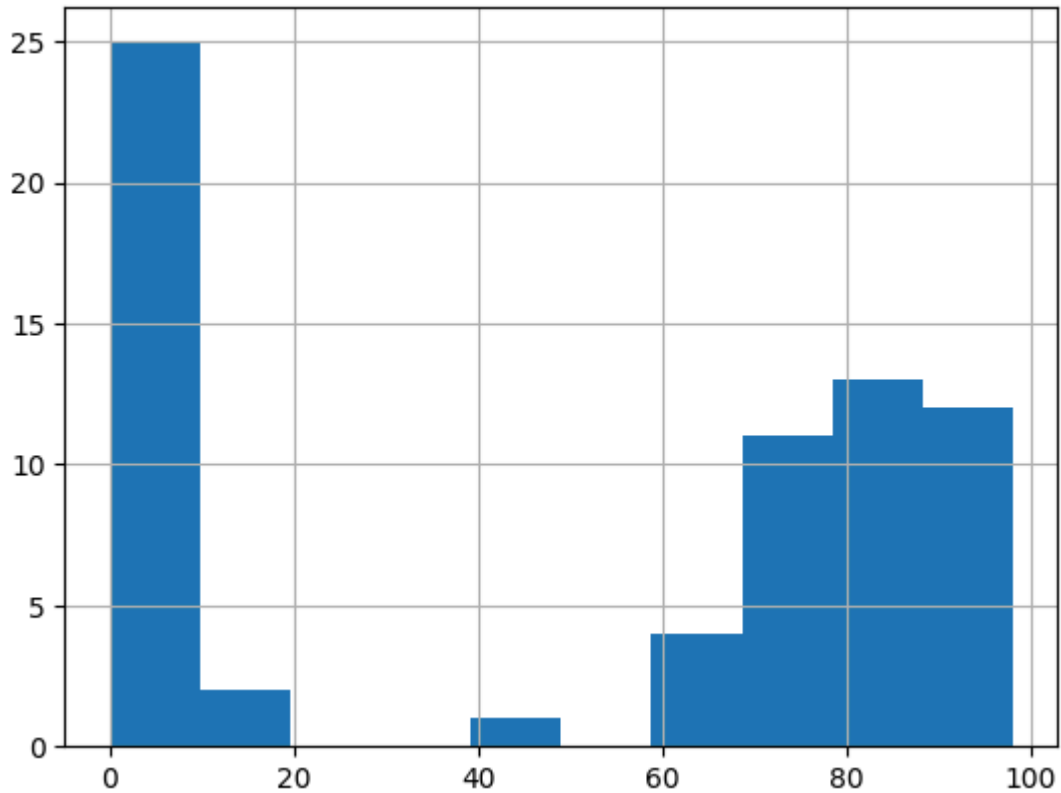


- The right tail of the distribution is longer, with most data points concentrated on the lower end.
- $\text{Mean} > \text{Median}$, as the mean is influenced by extreme high values.

- Example: Income distribution in most countries, where a small number of people earn significantly higher than the majority.

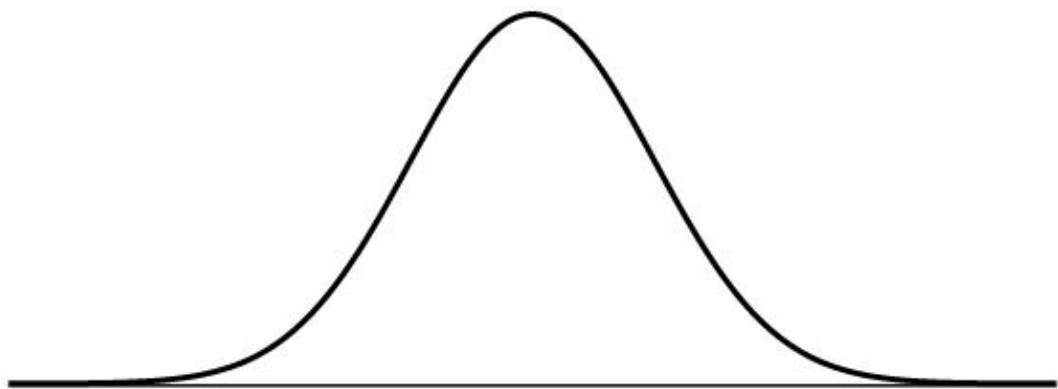
What is the distribution of Assignment #1 scores?

```
In [15]: scores = pd.read_csv("data/scores.csv")
scores['score'].hist();
```



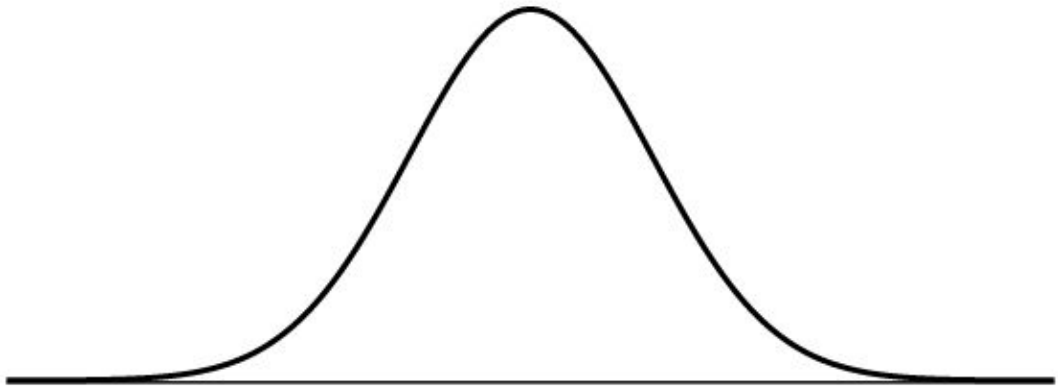
Normal distribution

The **normal distribution** is a **symmetric, bell-shaped** distribution with most data points clustering around the center and fewer as you move further from it.



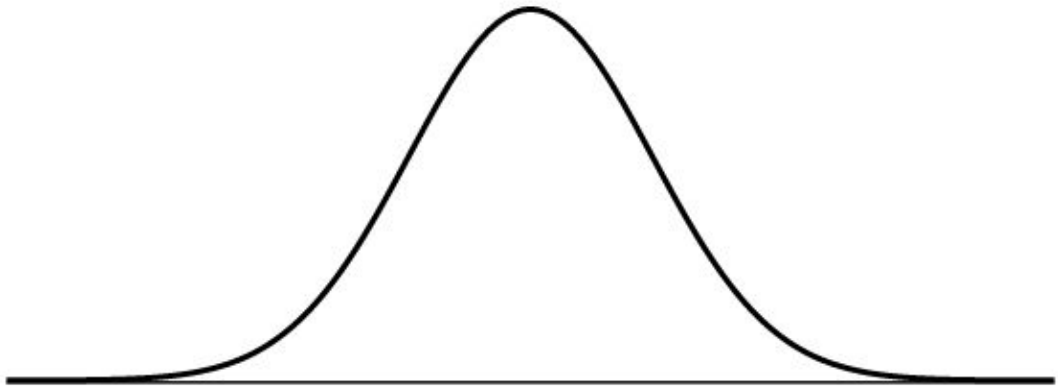
Key properties:

- **Symmetry:**



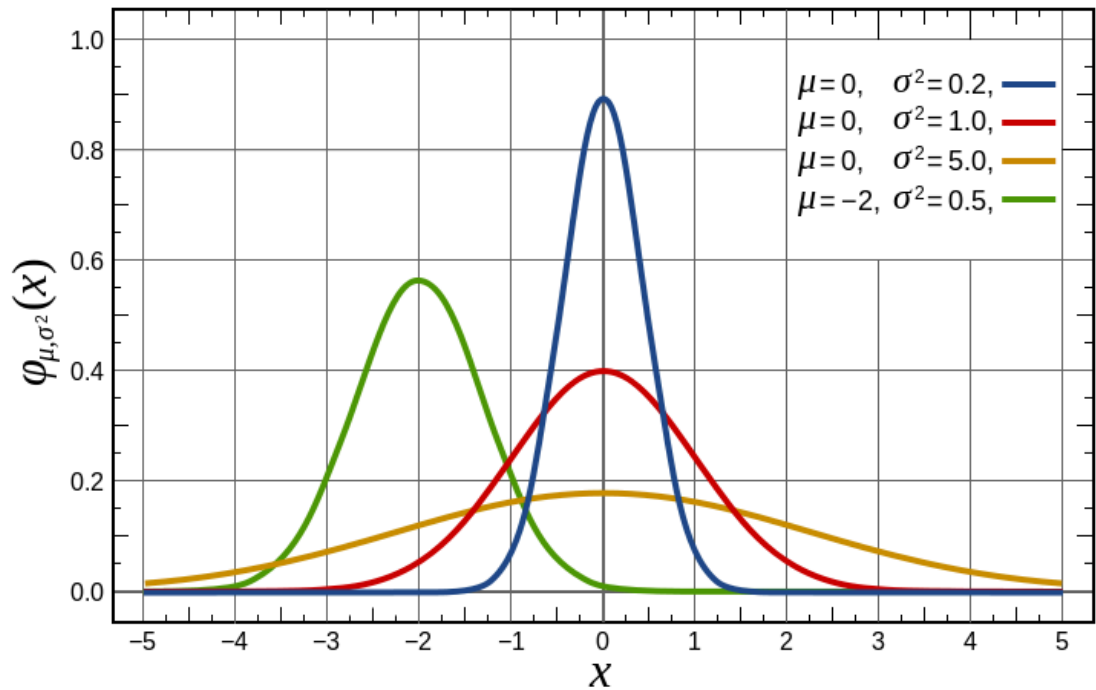
The normal distribution is perfectly symmetrical about its mean, meaning that the left and right sides of the curve mirror each other.

- **Mean, Median, and Mode are aligned:**



In a normal distribution, the mean, median, and mode are all equal and located at the center of the distribution.

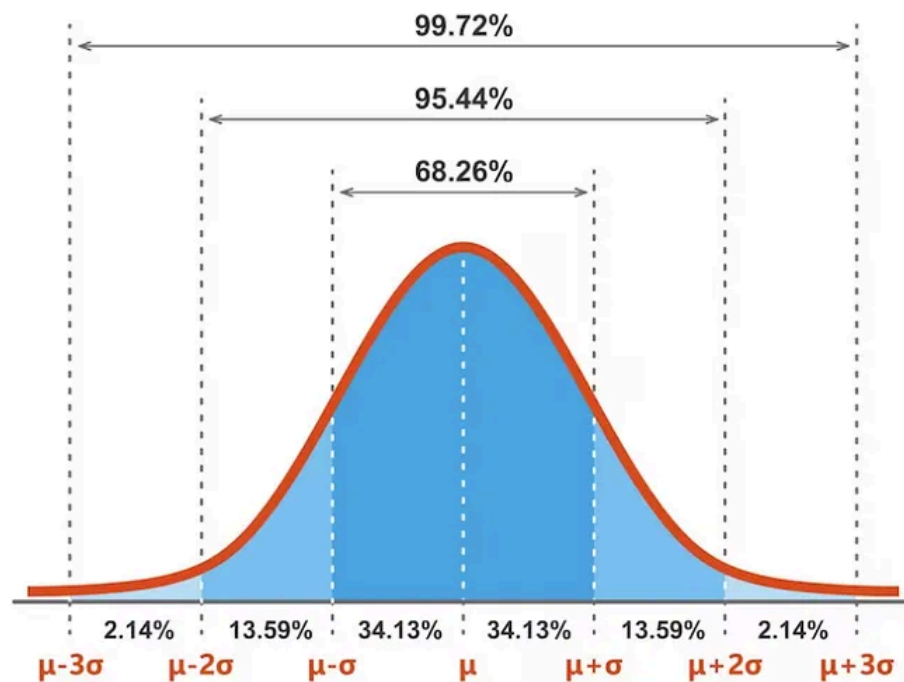
- **The mean and the standard deviation will define the exact shape of a normal curve.**



- **Empirical Rule**

The spread of data points around the mean follows a predictable pattern: - 68% of data falls within one standard deviation (σ) of the mean. - 95% within two standard deviations. - 99.7% within three standard deviations.

This pattern is known as the **Empirical Rule** or **68-95-99.7 rule** or **1-2-3 rule**.

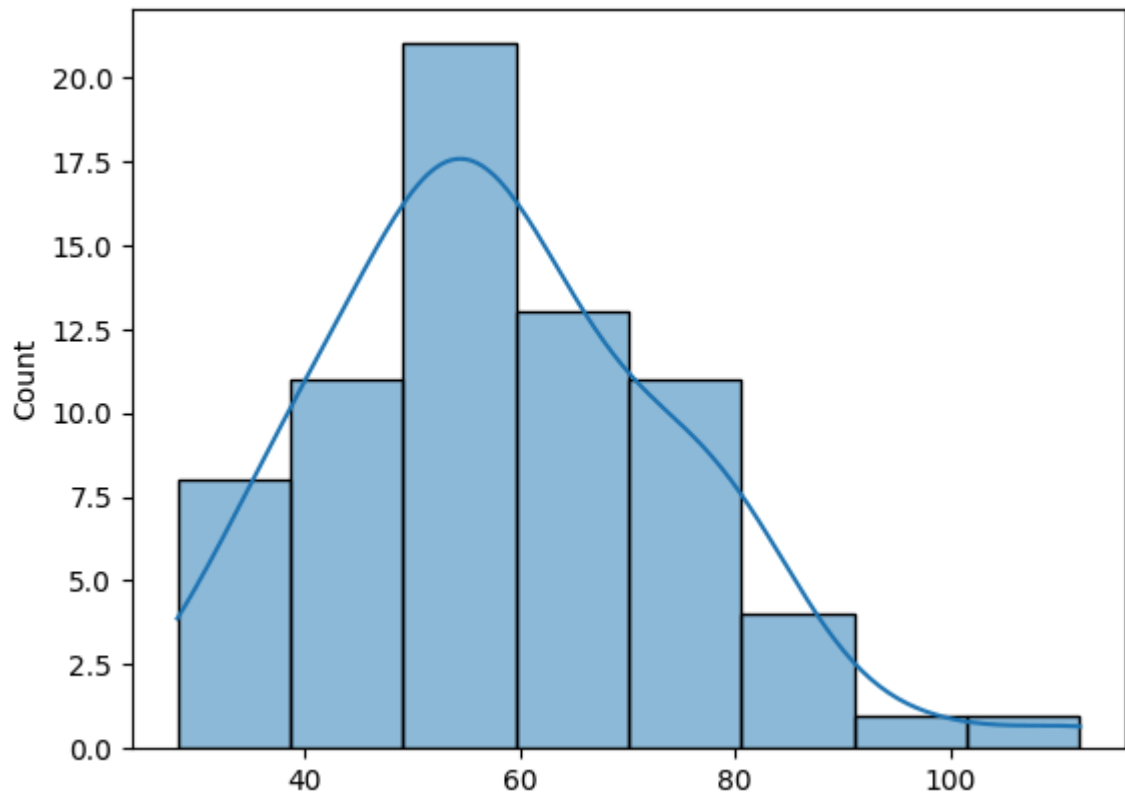


A good way to solidify understanding is to visualize the normal distribution using Python:

```
In [16]: import numpy as np
import seaborn as sns

# Generate random data following a normal distribution
data = np.random.normal(loc=60, scale=20, size=70)
data.mean()
# Plot the distribution

sns.histplot(data, kde=True);
```



Correlation Analysis

Correlation analysis is a statistical method used to understand the **relationship between two or more variables**. It measures how **changes in one variable are associated with changes in another**, and is essential in exploring and interpreting relationships within data.

Correlation can be assessed using: 1. Scatter plot 2. Correlation coefficient

The data will use:

```
In [17]: import pandas as pd
df = pd.read_csv("data/Fifa 23 Players Data.csv")
df.head(5)
```

Out[17]:

	Known As	Full Name	Overall	Potential	Value(in Euro)	Positions Played	Best Position	Nation
0	L. Messi	Lionel Messi	91	91	54000000	RW	CAM	Arge
1	K. Benzema	Karim Benzema	91	91	64000000	CF,ST	CF	Fr
2	R. Lewandowski	Robert Lewandowski	91	91	84000000	ST	ST	Pc
3	K. De Bruyne	Kevin De Bruyne	91	91	107500000	CM,CAM	CM	Bel
4	K. Mbappé	Kylian Mbappé	91	95	190500000	ST,LW	ST	Fr

5 rows × 89 columns

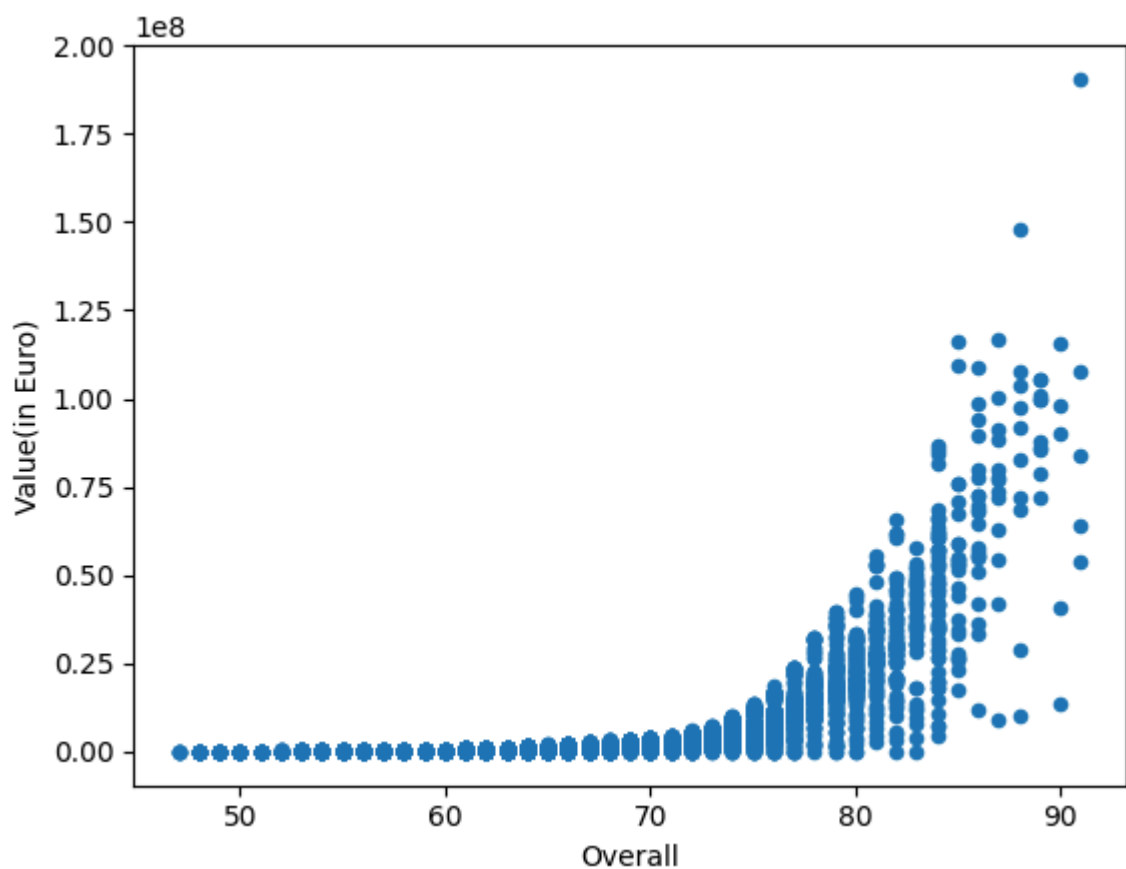


1. Correlation using Scatter plots:

Question

Does the *Euro value* of the player effected by the *overall* score ?

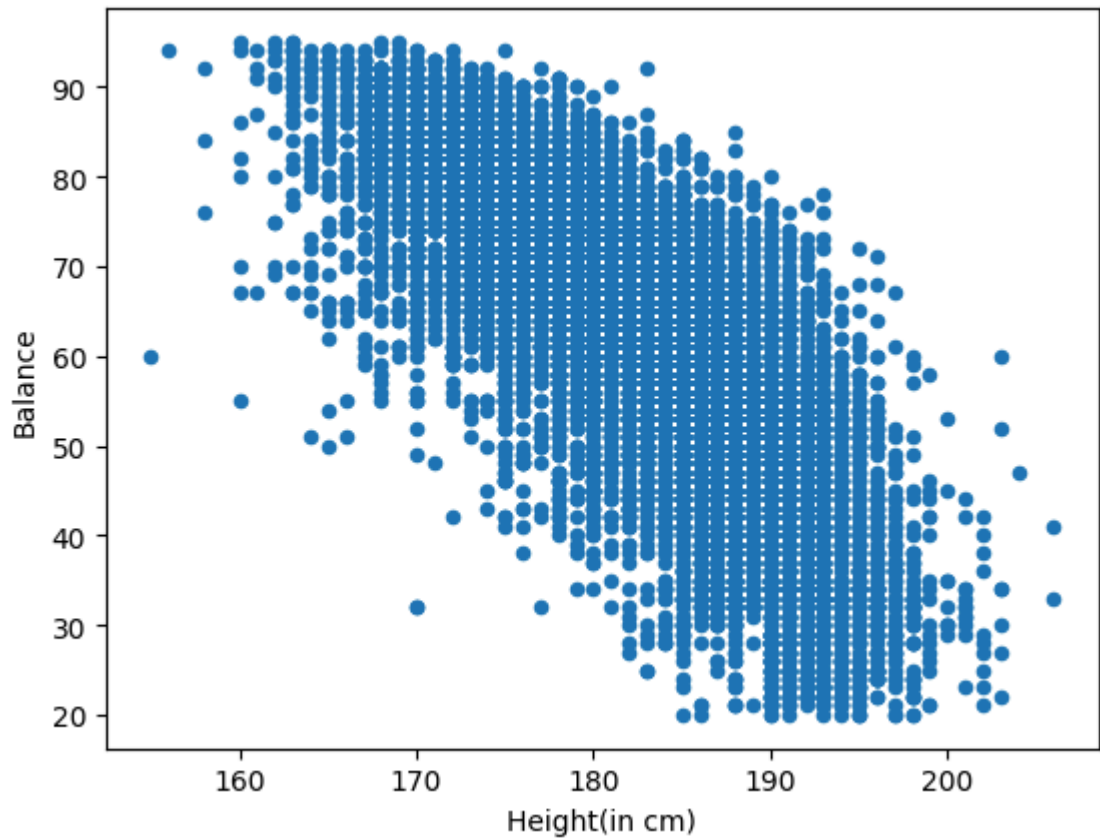
```
In [18]: df.plot.scatter(x='Overall', y='Value(in Euro)');
```



Question

Does the *Balance* skill of the effected by his *height* ?

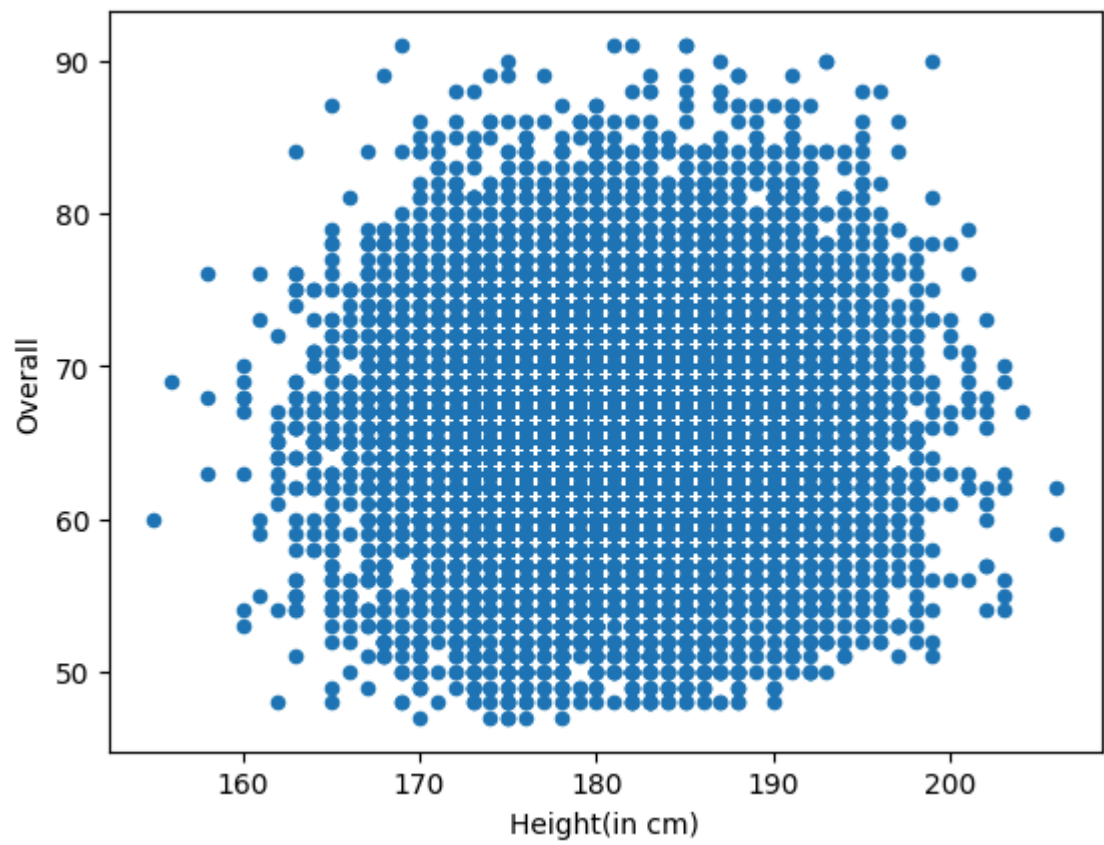
```
In [19]: df.plot.scatter(x='Height(in cm)', y='Balance');
```



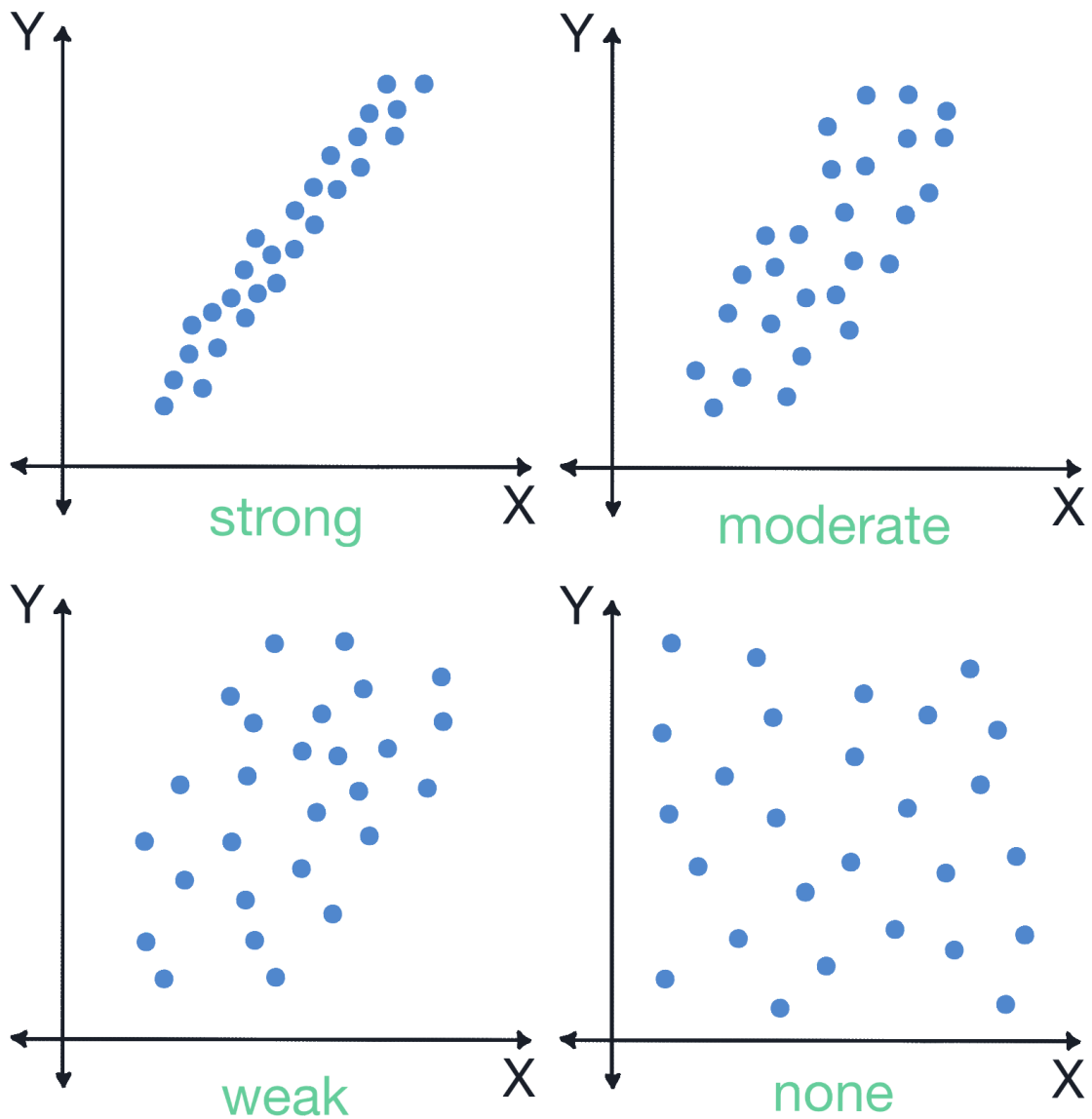
Question

Does the *Overall* score of the player effected by his *height*?

```
In [20]: df.plot.scatter(x='Height(in cm)', y='Overall');
```



Magnitude of the correlation using scatter plot



2. Correlation coefficient:

Pearson's correlation coefficient

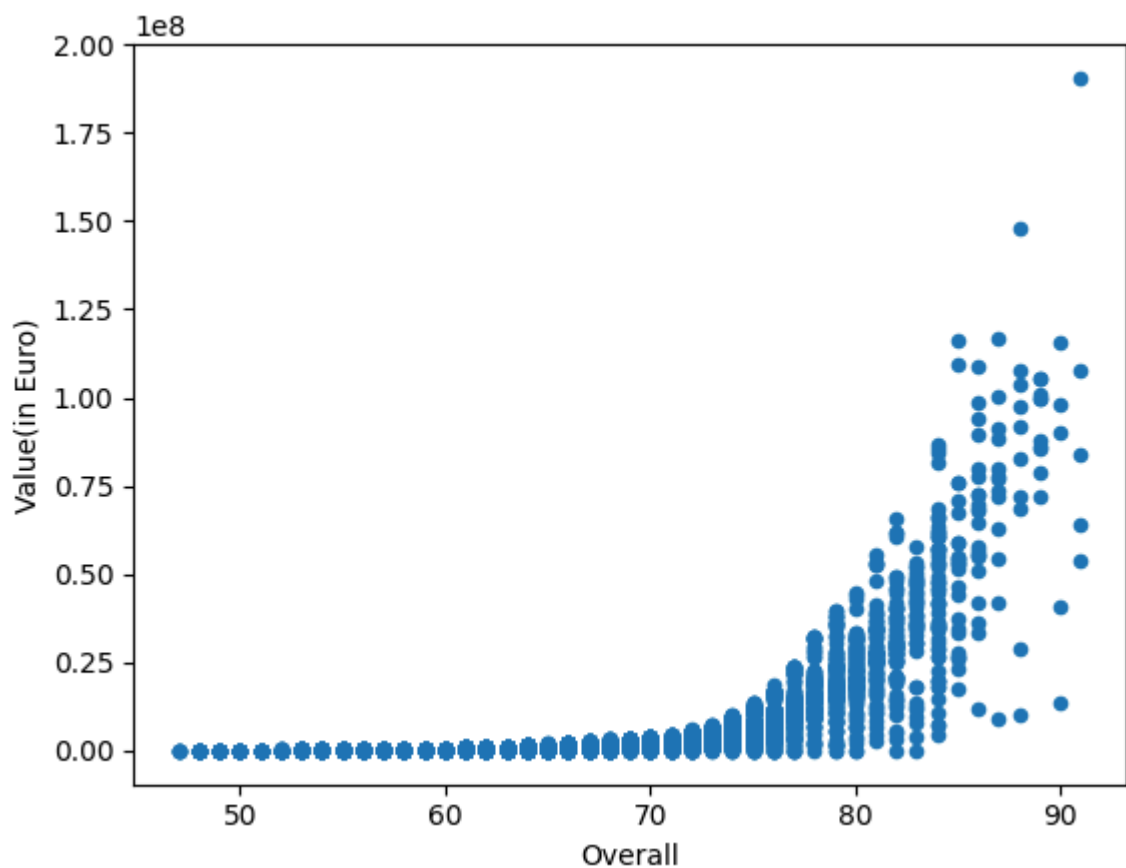
The most commonly used measure of correlation is **Pearson's** correlation coefficient (denoted as r), which quantifies the linear relationship between two continuous variables:

- Range of Values:
 - r ranges from -1 to +1.
 - $r > 0$: indicates a positive relationship.
 - $r < 0$: indicates a negative relationship.
- Usually:
 - 0.0 - 0.3: Weak correlation
 - 0.3 - 0.7: Moderate correlation
 - 0.7 - 1.0: Strong correlation

Note:

Pearson's correlation coefficient assumes the data is **linearly related** and **normally distributed**. For non-linear relationships, other measures, such as **Spearman's** or **Kendall's** rank correlation, may be more appropriate.

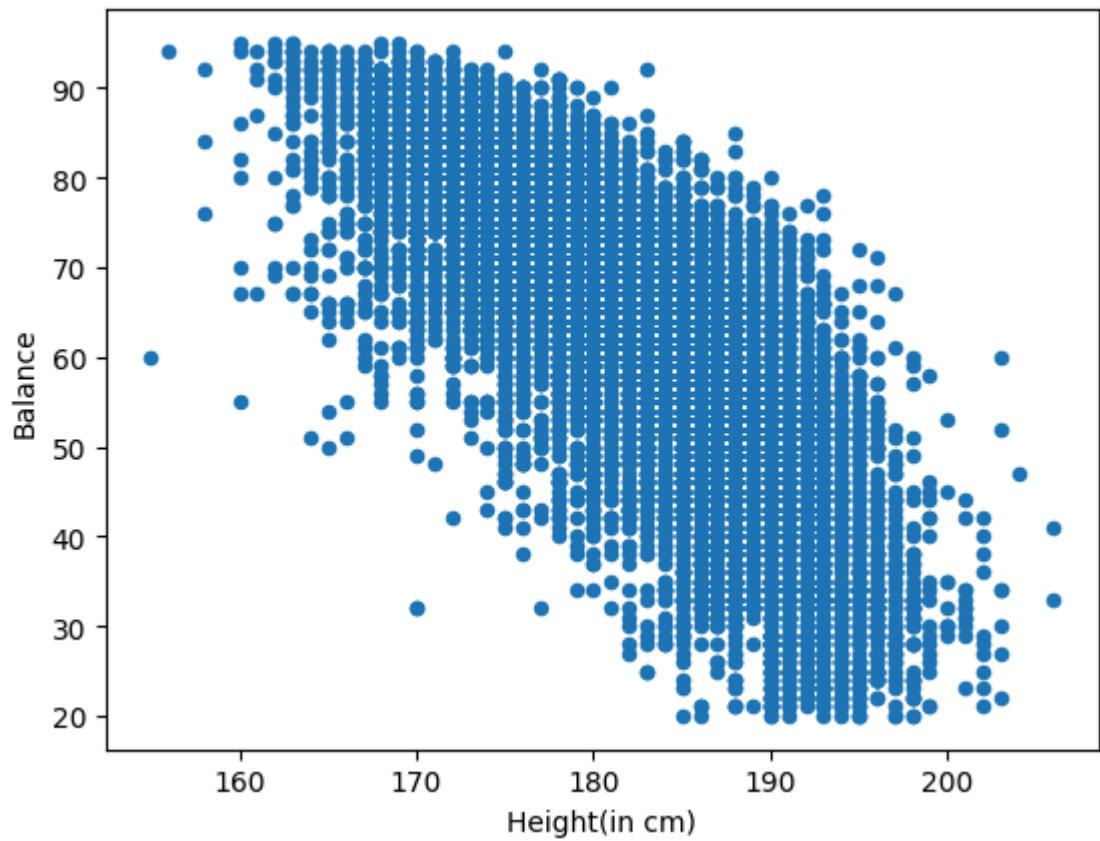
```
In [21]: df.plot.scatter(x='Overall', y="Value(in Euro)");
```



```
In [22]: df["Overall"].corr(df["Value(in Euro)"])
```

```
Out[22]: 0.5616422900267184
```

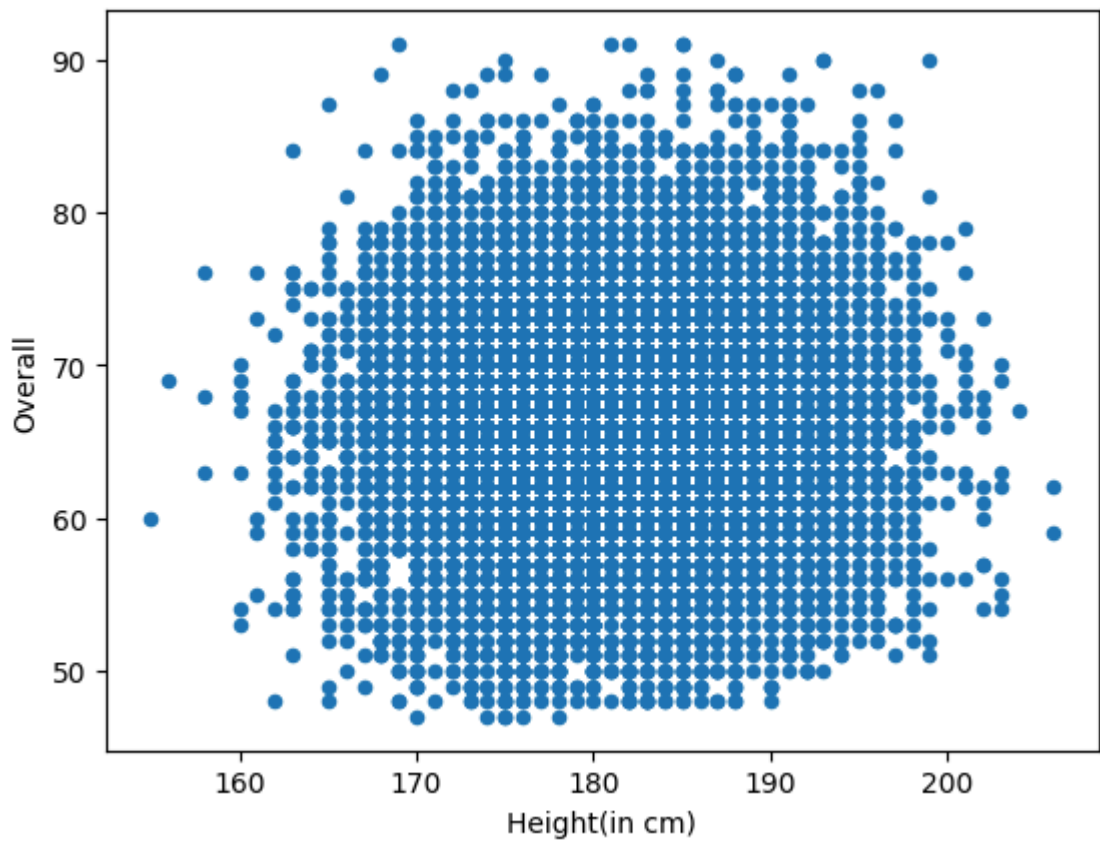
```
In [23]: df.plot.scatter(x='Height(in cm)', y='Balance');
```



```
In [24]: df['Balance'].corr(df['Height(in cm)'])
```

```
Out[24]: -0.7672474950683251
```

```
In [25]: df.plot.scatter(x='Height(in cm)', y='Overall');
```



```
In [26]: df['Overall'].corr(df['Height(in cm)'])
```

```
Out[26]: 0.03313746045613548
```

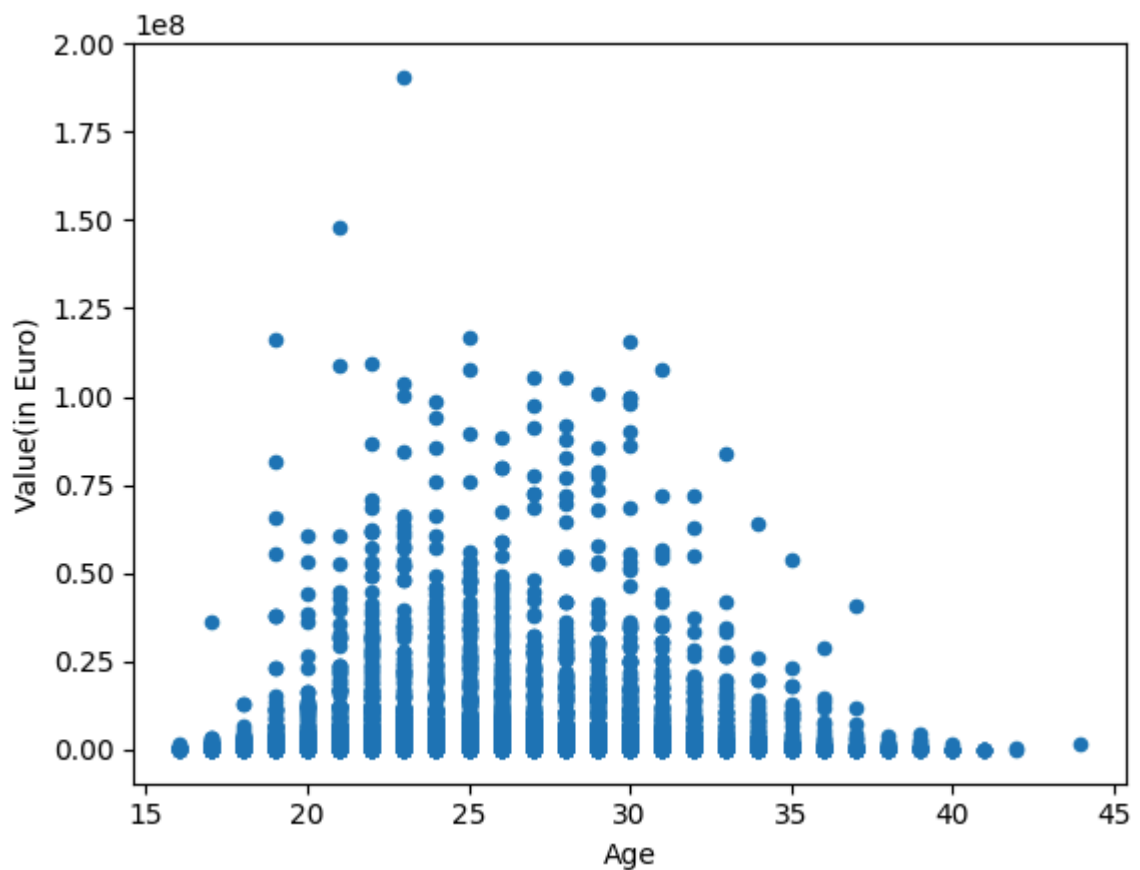
Question

Does the Euro value of the player effected by his age?

```
In [27]: df["Value(in Euro)"].corr(df['Age'])
```

```
Out[27]: 0.030661683255622994
```

```
In [28]: df.plot.scatter(x='Age', y='Value(in Euro)');
```



Correlation coefficient only measures the strength and direction of a **linear relationship**

Correlation matrix:

Shows the correlation coefficients between all numeric features in the data

```
In [29]: df.corr(numeric_only = True)
```

Out[29]:

	Overall	Potential	Value(in Euro)	Age	Height(in cm)	Weight(in kg)	TotalStats	Ba
Overall	1.000000	0.660630	0.561642	0.442369	0.033137	0.131420	0.608002	0
Potential	0.660630	1.000000	0.532835	-0.263686	0.017489	-0.003973	0.372469	0
Value(in Euro)	0.561642	0.532835	1.000000	0.030662	0.008028	0.027106	0.341551	0
Age	0.442369	-0.263686	0.030662	1.000000	0.066684	0.217895	0.246279	0
Height(in cm)	0.033137	0.017489	0.008028	0.066684	1.000000	0.754845	-0.382480	-0
...
RWB Rating	0.451583	0.291078	0.252712	0.141274	-0.321388	-0.257192	0.885559	0
LB Rating	0.428254	0.270494	0.234933	0.142184	-0.271875	-0.215549	0.840970	0
CB Rating	0.400783	0.227643	0.202809	0.183151	-0.088695	-0.044569	0.720341	0
RB Rating	0.428254	0.270494	0.234933	0.142184	-0.271875	-0.215549	0.840970	0
GK Rating	0.044462	-0.002057	0.011339	0.132966	0.369929	0.344316	-0.648967	0

71 rows × 71 columns



Data Sampling



Definitions and concepts

Population:

A **population** (or **universe**) is all possible cases that meet certain criteria. It's the total collection of cases that you're interested in studying

What is the population in the following cases?

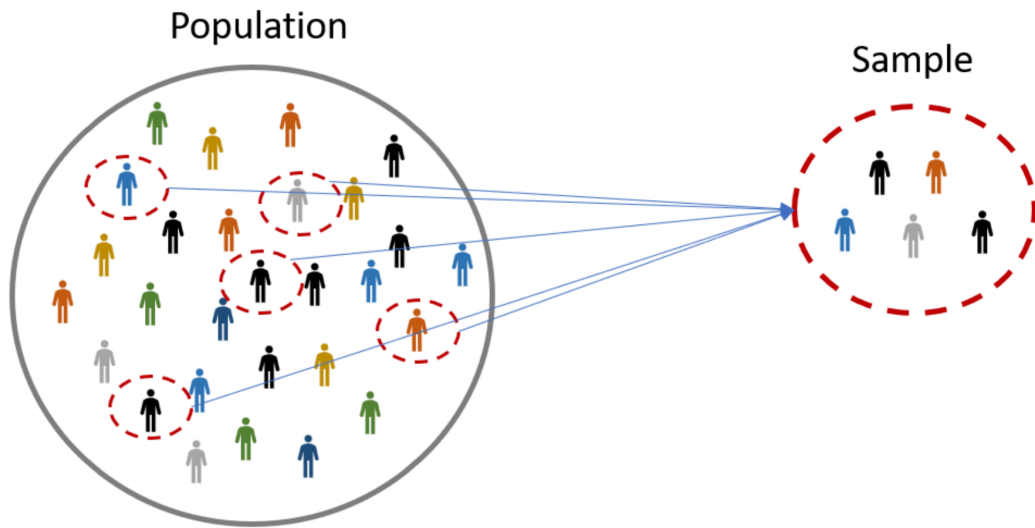
- Estimate university students average scores
- The percentage of female students who got honor last semester
- Does the performance of computer science students effectedd by remote vs. in-campus learning?

Can we collect data for all cases in the population ?

1. Getting data from the entire population is **too expensive** or would take **long time**.
2. The population is **constantly changing**

Sample:

A sample is the subset of a population



Sampling:

Sampling is the process of selecting a subset of data from a larger population to estimate characteristics of the whole population.

It is an essential technique used in statistical analysis to make inferences about a larger group based on a smaller sample

Sampling simulation

Let's simulate the sampling process

```
In [30]: import pandas as pd
import seaborn as sns
df = pd.read_csv("data/Fifa 23 Players Data.csv")
df
```

Out[30]:

	Known As	Full Name	Overall	Potential	Value(in Euro)	Positions Played	Best Position	N
0	L. Messi	Lionel Messi	91	91	54000000	RW	CAM	
1	K. Benzema	Karim Benzema	91	91	64000000	CF,ST	CF	
2	R. Lewandowski	Robert Lewandowski	91	91	84000000	ST	ST	
3	K. De Bruyne	Kevin De Bruyne	91	91	107500000	CM,CAM	CM	
4	K. Mbappé	Kylian Mbappé	91	95	190500000	ST,LW	ST	
...	
18534	D. Collins	Darren Collins	47	56	110000	ST,RM	CAM	R
18535	Yang Dejiang	Dejiang Yang	47	57	90000	CDM	CDM	
18536	L. Mullan	Liam Mullan	47	67	130000	CM	RM	
18537	D. McCallion	Daithí McCallion	47	61	100000	CB	CB	R
18538	N. Rabha	Nabin Rabha	47	50	60000	LB	LB	

18539 rows × 89 columns



Define population and populate sample

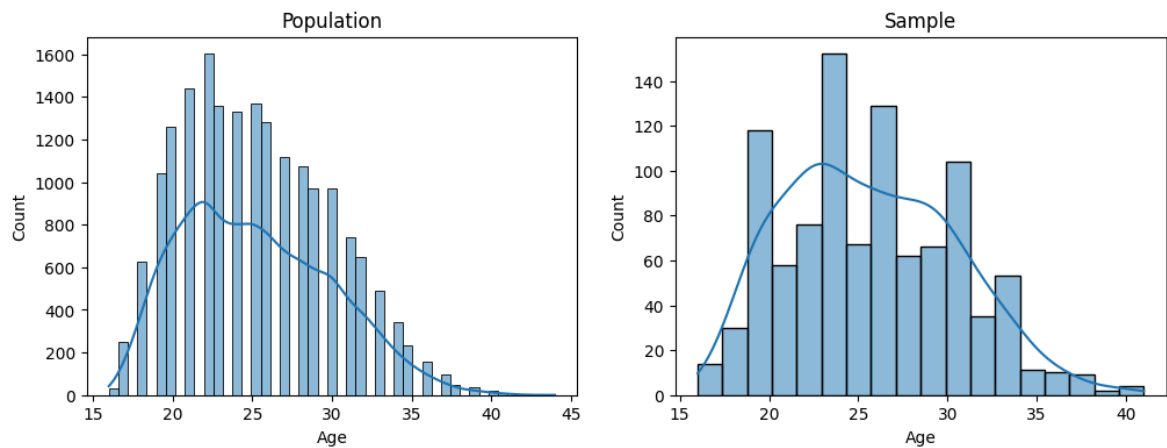
```
In [31]: population = df['Age']
sample = population.sample(1000)
print(sample[:10])
```

12510 21
6377 22
3401 28
2365 26
867 28
79 28
17448 19
8115 22
9755 26
5440 24
Name: Age, dtype: int64

Compare Population vs. Sample distribution

```
In [32]: import matplotlib.pyplot as plt
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

sns.histplot(ax = axes[0],data=population, kde=True).set_title("Population");
sns.histplot(ax = axes[1],data=sample, kde=True).set_title("Sample");
```



Are they similar ?

Population vs. Sample mean

```
In [33]: print(f"Population mean = {population.mean()}")
          print(f"Sample mean = {sample.mean()}")
```

Population mean = 25.24041210421274

Sample mean = 25.655

Population metric is called **Parameter**

Sample metric is called **Statistic**

The difference between the population parameter and the sample statistic is called

Sampling Error

The sampling error for the above sample is:

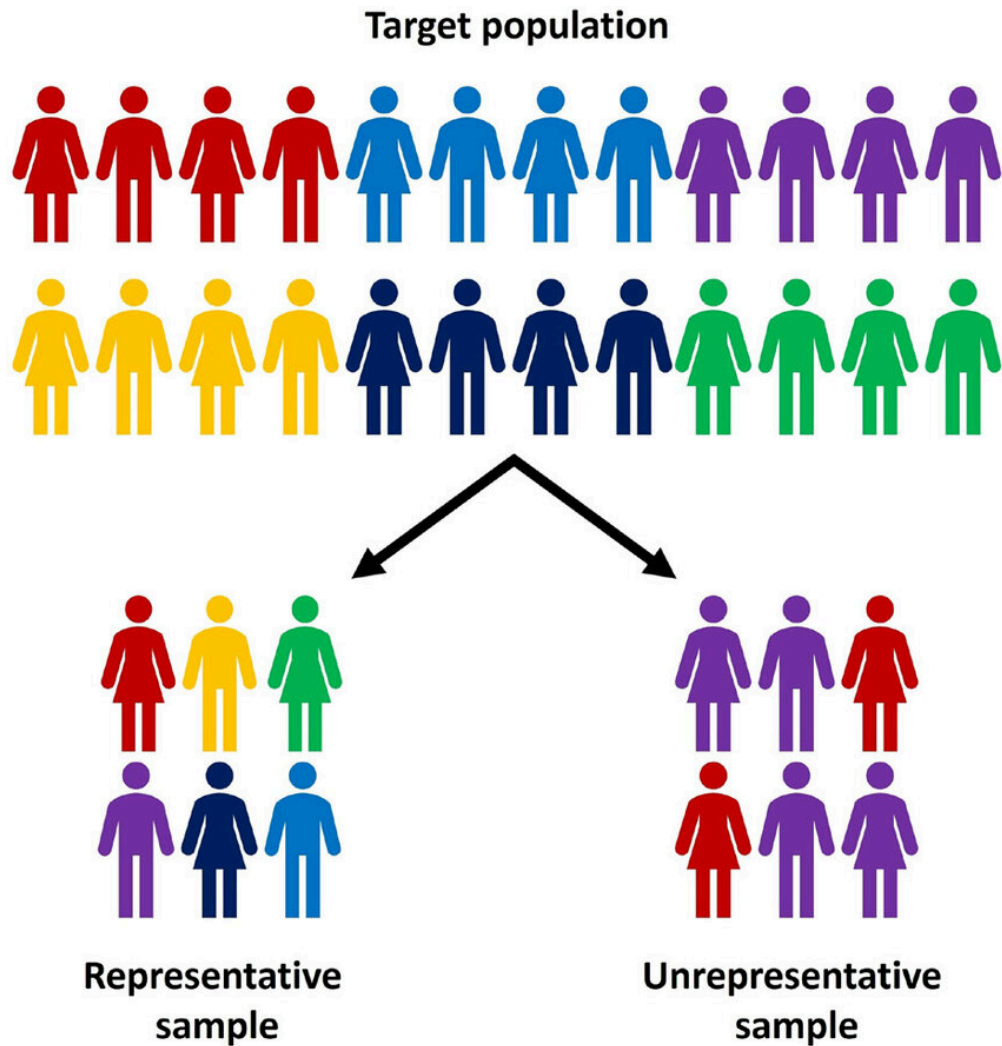
```
In [34]: population.mean() - sample.mean()
```

Out[34]: -0.4145878957872604

Representative sample

A representative sample is a sample that **accurately represents the characteristics of the population.**

For example, in a warehouse with a sample of 1,000 people split equally into 500 males and 500 females, a smaller group of 100 males and 100 females could generate a representative sample of the larger group.



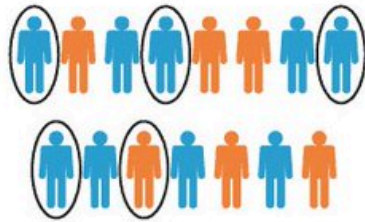
Example: Imagine a population that has a male/female split, or ratio, of 60%/40%. If a sample of the population is representative, you'd expect it to have a male/female split very close to 60%/40%.

Sampling methods

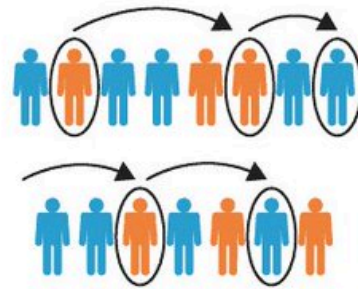
There are four main types of sampling:

- Simple random sampling (SRS)
- Stratified sampling
- Systematic sampling
- Cluster sampling

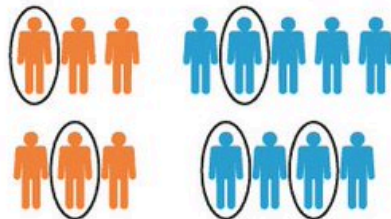
Simple random sample



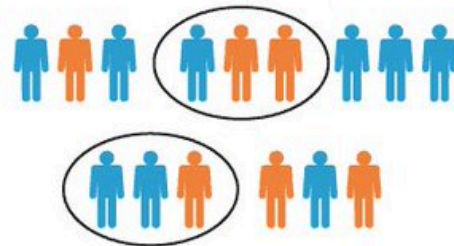
Systematic sample



Stratified sample



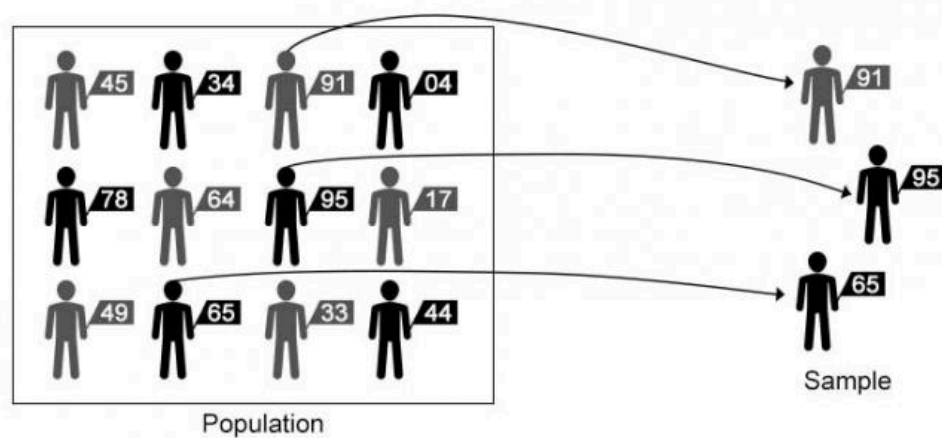
Cluster sample



Simple Random Sampling (SRS)

In this method, **each member** of the population **has an equal chance of being selected** in the sample. To conduct this type of sampling you can apply random number generator.

For example, you want to select a simple random sample of 1000 employees of a social media marketing company. You assign a number to every employee in the company database from 1 to 1000, and use a random number generator to select 100 numbers.



Simple Random Sampling

Advantages:

- Easy to understand and implement.
- Usually yeild to representative sample when the population is homogeneuos

Disadvantages:

- May not represent the population if certain groups are small.

Python implementation:

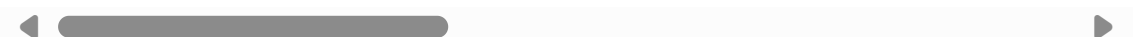
Sample randomly 200 players from Fifa 23 datasets

```
In [35]: sample_size = 160

srs = df.sample(n=sample_size)
srs
```

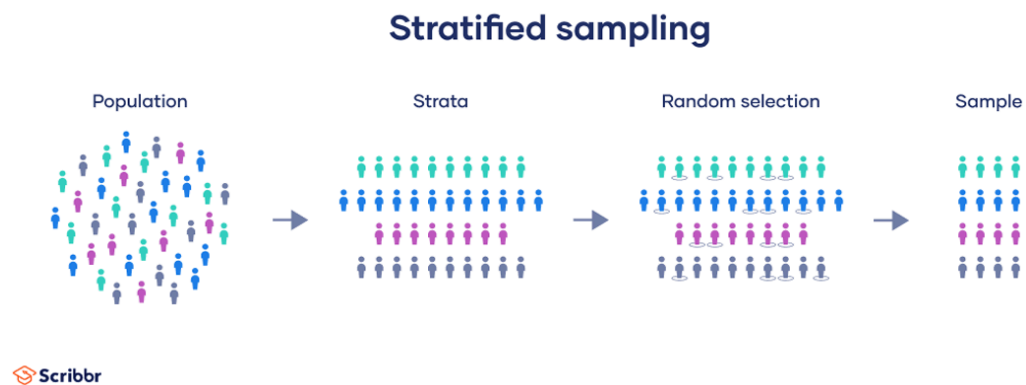
	Known As	Full Name	Overall	Potential	Value(in Euro)	Positions Played	Best Position	National
14786	Du Jia	Jia Du	61	61	220000	GK	GK	China
8582	João Queirós	João Ricardo Pereira Queirós	66	73	1600000	CB	CB	Portug
9527	A. Famewo	Akin Famewo	65	71	1000000	CB	CB	Engla
13993	M. Esponda	Marcelo Esponda	62	77	900000	CDM	RWB	Argenti
2862	K. Roofe	Kemar Roofe	73	73	3000000	ST,CF	ST	Jama
...
1556	Escudero	Sergio Escudero Palomo	75	75	3900000	LB,LM	LB	Spa
2514	M. Mancosu	Marco Mancosu	73	73	1900000	CAM	CAM	Ita
6209	S. Tigges	Steffen Tigges	68	73	1800000	ST	ST	Germa
11157	J. de Lange	Jeffrey de Lange	64	70	750000	GK	GK	Netherlan
839	M. Thuram	Marcus Thuram	78	83	21500000	ST,LM	ST	Fran

160 rows × 89 columns



Stratified Sampling

In this methods, the population is divided into strata (subgroups), and a random sample is taken from each.



Advantages:

- Ensures representation of all subgroups.
- Higher precision if each stratum is internally homogenous.

Disadvantages:

- More complex to organize and implement.
- Requires knowledge of population strata.

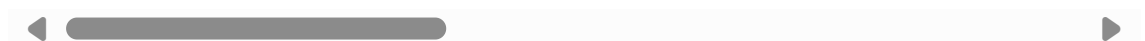
Sample single player from each nationality

```
In [36]: stratified = df.groupby('Nationality').sample(n = 1)
stratified
```

Out[36]:

	Known As	Full Name	Overall	Potential	Value(in Euro)	Positions Played	Best Position	Nat
12149	R. Akbari	Rahmat Akbari	63	72	1100000	CM,ST	RM	Af
3317	K. Asllani	Kristjan Asllani	72	84	5500000	CDM,CM	CM	
783	R. Bensebaini	Ramy Bensebaini	78	79	15000000	LB,CB	LB	
13768	Iker Álvarez	Iker Álvarez de Eulate	62	74	850000	GK	GK	
8762	Nélson da Luz	Nélson Conceição da Luz	66	72	1400000	LW	LM	
...
11058	G. Contreras	Germán Contreras	64	71	1100000	RB,LB	RB	\
9433	Nguyễn Quang Hải	Quang Hải Nguyễn	66	71	1200000	CM	CAM	
11120	L. Isgrove	Lloyd Isgrove	64	64	575000	RW,RM	RW	
8559	L. Musonda	Lubambo Musonda	66	66	900000	RM,RWB,LWB	RM	
3507	N. Mushekwi	Nyasha Mushekwi	72	72	1100000	ST	ST	Z

160 rows × 89 columns



Print the number of different nationalities in each sample

```
In [37]: print(stratified['Nationality'].unique())
print(srs['Nationality'].unique())
```

160

53

```
In [38]: import numpy as np
np.setdiff1d(stratified['Nationality'].unique(), srs['Nationality'].unique())
```

```
Out[38]: array(['Afghanistan', 'Algeria', 'Andorra', 'Angola',
               'Antigua and Barbuda', 'Armenia', 'Azerbaijan', 'Barbados',
               'Belarus', 'Benin', 'Bermuda', 'Bolivia', 'Bosnia and Herzegovina',
               'Bulgaria', 'Burkina Faso', 'Burundi', 'Cameroon',
               'Cape Verde Islands', 'Central African Republic', 'Chad',
               'Chinese Taipei', 'Comoros', 'Congo', 'Congo DR', 'Costa Rica',
               'Croatia', 'Cuba', 'Cyprus', 'Dominican Republic', 'El Salvador',
               'Equatorial Guinea', 'Estonia', 'Ethiopia', 'Faroe Islands',
               'Fiji', 'Gambia', 'Georgia', 'Gibraltar', 'Grenada', 'Guam',
               'Guatemala', 'Guinea', 'Guyana', 'Haiti', 'Honduras', 'Hong Kong',
               'Indonesia', 'Iran', 'Iraq', 'Israel', 'Jordan', 'Kazakhstan',
               'Kenya', 'Korea DPR', 'Korea Republic', 'Kosovo', 'Latvia',
               'Lebanon', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania',
               'Luxembourg', 'Madagascar', 'Mali', 'Malta', 'Mauritania',
               'Mauritius', 'Moldova', 'Montenegro', 'Montserrat', 'Mozambique',
               'Namibia', 'New Zealand', 'Nigeria', 'North Macedonia',
               'Palestine', 'Panama', 'Papua New Guinea', 'Philippines',
               'Puerto Rico', 'Qatar', 'Saint Kitts and Nevis', 'Saint Lucia',
               'Serbia', 'Sierra Leone', 'Singapore', 'Slovenia', 'South Sudan',
               'Sudan', 'Suriname', 'Syria', 'São Tomé e Príncipe', 'Tanzania',
               'Thailand', 'Togo', 'Trinidad and Tobago', 'Tunisia', 'Uganda',
               'Ukraine', 'United Arab Emirates', 'Uzbekistan', 'Venezuela',
               'Vietnam', 'Wales', 'Zambia', 'Zimbabwe'], dtype=object)
```

We can observe that the nationalities in the stratified sample represent all nationalities in the population, whereas several nationalities are absent in the sample when using simple random sampling.

Exercise:

Sample ~300 player using:

- SRS
- Stratified sample by **Best Position**

Calculate the **CF Rating** for each sample, which one shows higher sampling error?

Calculate the population mean

```
In [46]: pop_mean = df['CF Rating'].mean()
pop_mean
```

```
Out[46]: 55.714925292626354
```

Calculate using SRS

```
In [47]: srs_mean = df['CF Rating'].sample(n=300).mean()
srs_mean
```

```
Out[47]: 54.61
```

Calculate using stratified sampling

```
In [48]: str_s = df.groupby('Best Position').sample(frac=0.016)
str_mean = str_s['CF Rating'].mean()
str_mean
```

Out[48]: 55.5959595959596

Calculate sampling errors

```
In [49]: print(pop_mean)
          print(srs_mean)
          print(str_mean)

          srs_sampling_error = pop_mean - srs_mean
          sts_sampling_error = pop_mean - str_mean

          print(srs_sampling_error)
          print(sts_sampling_error)
```

55.714925292626354

54.61

55.5959595959596

1.1049252926263549

0.1189656966667556

We can notice that stratified sample yeild to lower samoling error

</Statistics for data science>