# Data Science Foundations

Master in Big Data Solutions 2017-2018

Francisco Gutierres
francisco.gutierres@bts.tech

Ludovico Boratto
ludovico.boratto@bts.tech

Liana Napalkova
liana.napalkova@bts.tech

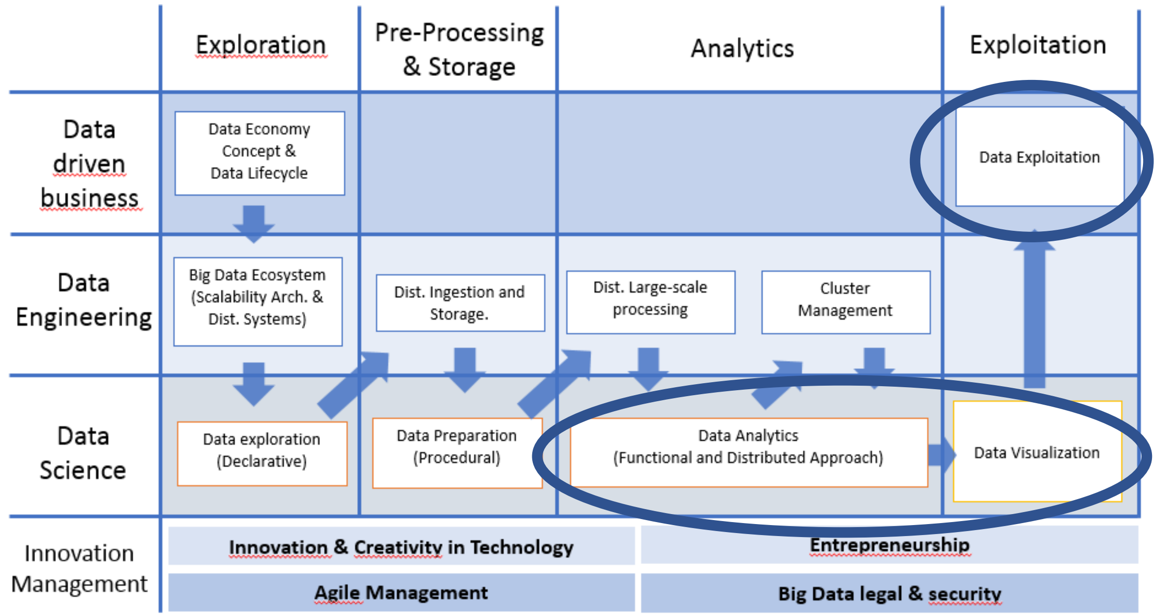
# Session 6 – Descriptive Statistics

Francisco Gutierres

# What we will learn

**Session1: Descriptive Statistics**

Introduction to descriptive statistics using Pandas' DataFrame, NumPy and Matplotlib library.




We will learn how to:

- Implement of several Phyton libraries (Pandas, NumPy and Matplotlib) to perform descriptive statistics with data using DataFrames.

# Today's Objective

What will we learn today?

- How to perform descriptive statistics.
- How to interpret results.

# Contents

**I. Activities in the class (Demos and Exercises)**

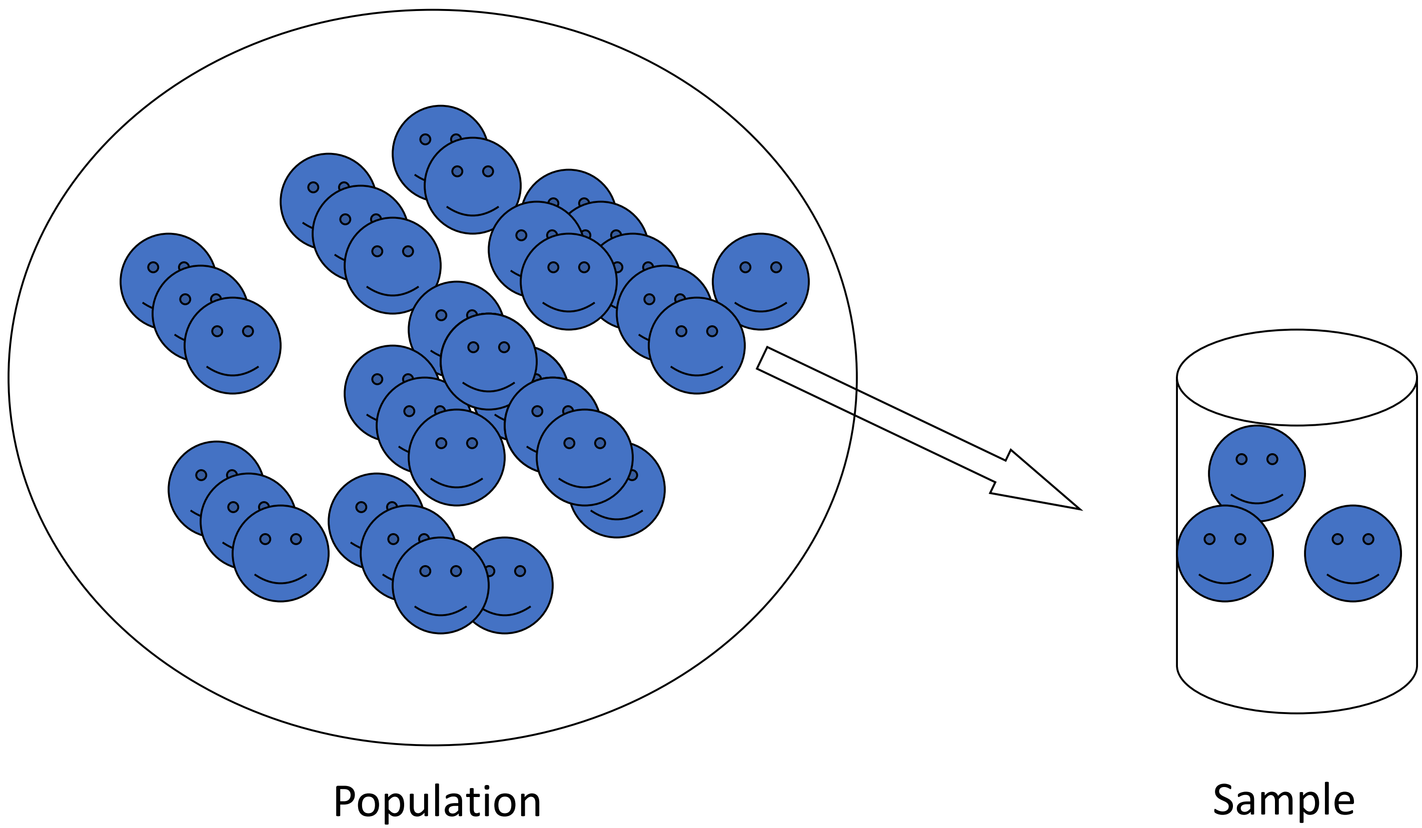
**II. Individual Assignment (will be explained at the end of the class)**

Use a dataset (called “1_Titanic “) in Pandas DataFrame:

- Perform Descriptive Statistics.
- Commit scripts to your Git.
- Create your repository “DataScienceFoundations”.

# What is Descriptive Statistics?

- Descriptive Statistics are used by researchers to report on **Populations** and **Samples**.
- In Sociology: **summary descriptions** of measurements (**variables**) taken about a **group of people**.
- By summarizing information, Descriptive Statistics **speed up** and **simplify comprehension** of a **Group's Characteristics**.



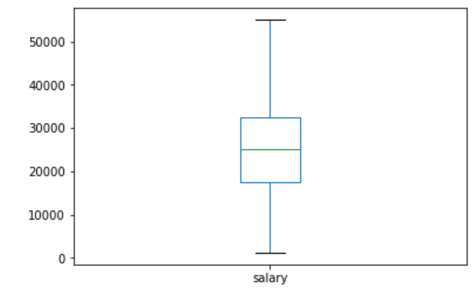
# What is Descriptive Statistics?

- One of the most basic ways to split statistics is to break it into two categories: **descriptive** and **inferential**.
- **Inferential statistics** - takes part of a population and attempts to infer something about the entire population. For example, I interview 100 likely voters about who they're going to vote for, and infer who is going to win an election.
- **Descriptive statistics** - describes only the numbers you have right in front of you. For example, I have a list of all the planes that took off from the airport yesterday, and they were on average ten minutes late.
- We're going to be doing some basic descriptive statistics, because we sure aren't going to release our entire dataset to our clients. **Summing it all up** into a **few numbers** works much more nicely.
- You use descriptive statistics all the time! Averages! Maximums! Minimums!
- We can break down descriptive statistics into a few major concepts, we'll talk about **central tendency** and **variability**.

# Types of Descriptive Statistics

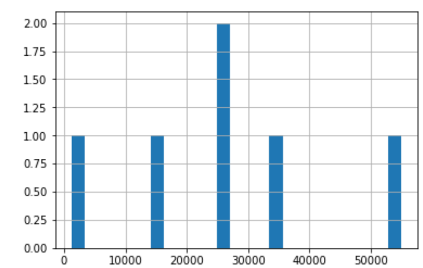
- **Organize Data**:

✓ Tables (Frequency, descriptive summary).
✓ Graphs (Histogram, Box-and-whisker plots).




- **Summarize Data**:

✓ Central Tendency.
✓ Variation.




| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| salary | 7.000 | 36600.000 | 32530.810 | 1200.000 | 20000.000 | 25000.000 | 45000.000 | 100000.000 |

# Types of Descriptive Statistics

- **Central Tendency (or Groups' "Middle Values")**
  - ✓ Mean
  - ✓ Median
  - ✓ Mode

- **Variation (or Summary of Differences Within Groups)**
  - ✓ Range
  - ✓ Interquartile Range (IQR)
  - ✓ Variance
  - ✓ Standard Deviation

# Data

- **Two major categories of data:**
  - ✓ qualitative/categorical - things that aren't numbers. Whether you're married or single, live in Barcelona or Madrid, or have blue eyes or brown eyes.
  - ✓ quantitative/numerical - is, obviously, based on numbers.

- **Kinds of numeric data:**
  - ✓ Continuous - data can be broken down into smaller and smaller numerical pieces. For example, is the temperature in your apartment 69°F, or 69.4°F, or 69.123°F?
  - ✓ Discrete - data are still numbers, but they can only have certain values. For example, Yelp ratings are 1, 2, 3, 4 or 5 stars.

# Demos

# Demo 1

- **Objective:** Mastering skills about Descriptive Statistics.
- **Phyton libraries for Descriptive Statistics**: Pandas, numpy and Matplotlib.

- Create a new notebook and put your code in the final.
- Export your Demo 1 *.ipynb notebook and push it to your repository "DataScienceFoundations".

## 1. Import modules

```
import pandas as pd
import numpy as np
%matplotlib inline
```

## 2. Create a dataframe

```
data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'age': [42, 52, 36, 24, 73],
        'preTestScore': [4, 24, 31, 2, 3],
        'postTestScore': [25, 94, 57, 62, 70]}
df = pd.DataFrame(data, columns = ['name', 'age', 'preTestScore', 'postTestScore'])
df
```

Out[2]:

| | name | age | preTestScore | postTestScore |
|---|---|---|---|---|
| 0 | Jason | 42 | 4 | 25 |
| 1 | Molly | 52 | 24 | 94 |
| 2 | Tina | 36 | 31 | 57 |
| 3 | Jake | 24 | 2 | 62 |
| 4 | Amy | 73 | 3 | 70 |

# Demo 1

## 3. Basic statistics

### 3.1.1. The sum of all the age

*df[‘variable'].sum()*

**Question: value?**

**Answer:**

```
In [3]: df['age'].sum()
Out[3]: 227
```

### 3.1.2. Cumulative sum of age, moving from the rows from the top

*df['variable '].cumsum()*

**Question: Cumulative sum?**

**Answer:**

```
In [4]: df['age'].cumsum()
Out[4]: 0     42
        1     94
        2    130
        3    154
        4    227
        Name: age, dtype: int64
```

### 3.1.3. Count the number of non-NA values

*df['variable'].count()*

**Question: value?**

**Answer:**

```
In [5]: df['age'].count()
Out[5]: 5
```

BTS | Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

# Demo 1

## 3.1.4. Count the number of NA values

*count_nan = len(df) - df.count()*

*count_nan*

**Question: value?**

**Answer:**

```
In [6]: count_nan = len(df) - df.count()
        count_nan

Out[6]: name             0
        age              0
        preTestScore     0
        postTestScore    0
        dtype: int64
```

## 3.1.5. Minimum value of age

*df[' 'variable'].min()*

**Question: value?**

**Answer:**

```
In [7]: df['age'].min()

Out[7]: 24
```

## 3.1.6. Maximum value of age

*df['variable'].max()*

**Question: value?**

**Answer:**

```
In [8]: df['age'].max()

Out[8]: 73
```

# Demo 1

## 3.1.7. Range of age

*df['variable'].max() - df['variable'].min()*

**Question: value?**

**Answer:**

```
In [9]: df['age'].max() - df['age'].min()

Out[9]: 49
```

## 3.1.8. Frequency table of age

*counts = df['variable'].value_counts()*

*counts*

**Question: value?**

**Answer:**

```
In [10]: counts = df['age'].value_counts()
         counts

Out[10]: 52    1
         73    1
         42    1
         36    1
         24    1
         Name: age, dtype: int64
```

Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

# Demo 1

## 4. Central Tendency

**Question: value?**

**Answer:**

```
In [11]: df['age'].mean()
Out[11]: 45.4
```

### 4.1.1. MEAN

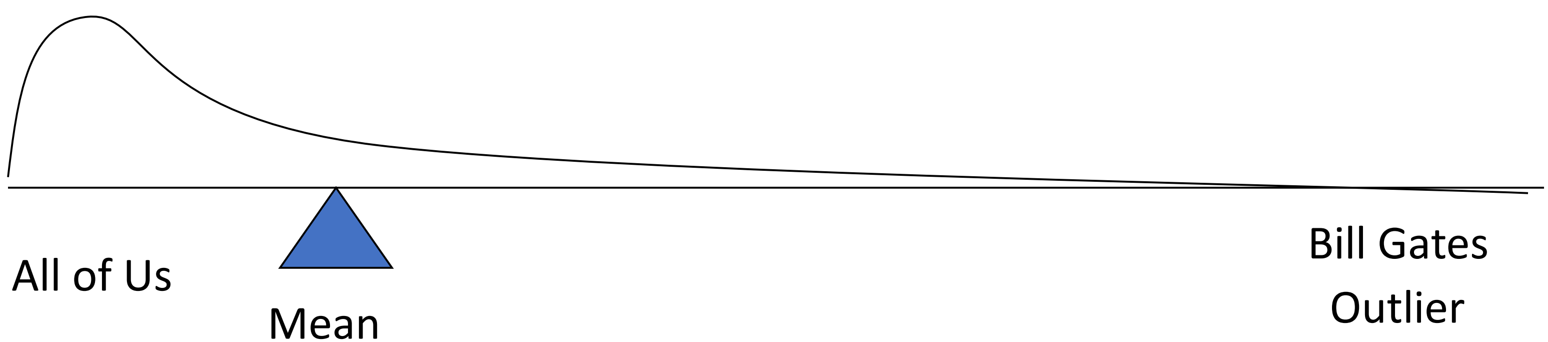
Most commonly called the "average."

Add up the values for each case and divide by the total number of cases.

1. Means can be badly affected by outliers (data points with extreme values unlike the rest).
2. Outliers can make the mean a bad measure of central tendency or common experience.

*df['variable'].mean()*

Income in the U.S.



# Demo 1

## 4.1.2. MEDIAN

The middle value when a variable's values are ranked in order; the point that divides a distribution into two equal halves.

When data are listed in order, the median is the point at which 50% of the cases are above and 50% below it.

The 50$^{th}$ percentile.

*df['variable'].sort_values()*

or

*df['variable'].median()*

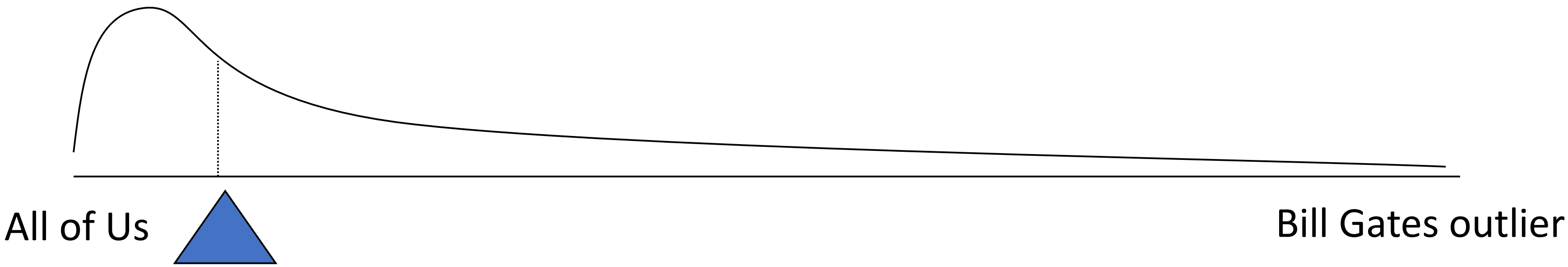
**Question: value?**

**Answer:**

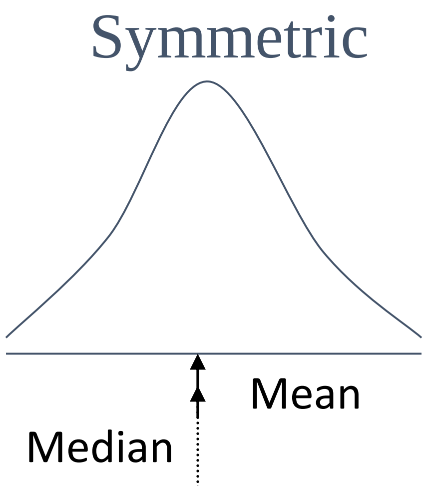
```
In [12]: df['age'].sort_values()
Out[12]: 3    24
         2    36
         0    42
         1    52
         4    73
         Name: age, dtype: int64

In [13]: df['age'].median()
Out[13]: 42.0
```

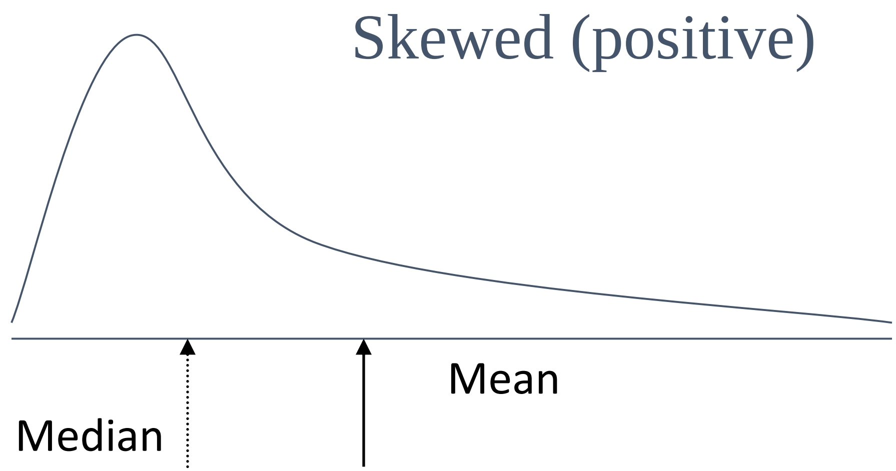
# Demo 1

1. The median is unaffected by outliers, making it a better measure of central tendency, better describing the “typical person” than the mean when data are skewed.




2. If the recorded values for a variable form a symmetric distribution (normal), the median and mean are identical.
3. In skewed data, the mean lies further toward the skew than the median.



# Demo 1

## 4.1.2. MODE

The most common data point is called the mode.

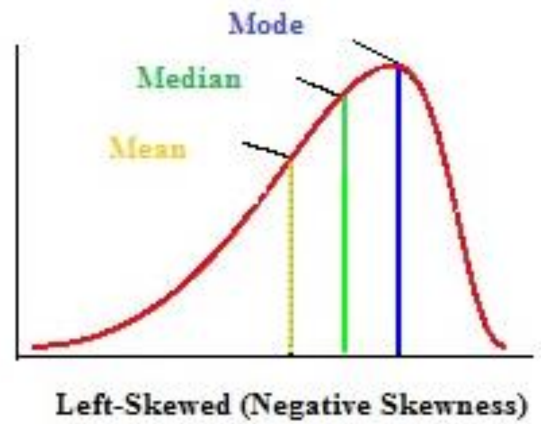
*df['variable'].mode()*

**Question: value?**

**Answer:**

```
In [15]: df['age'].mode()

Out[15]: Series([], dtype: int64)
```

*In a left skewed distribution, the mean is to the left of the peak.*



# Demo 1

## 5. Variation

### 5.1.1. RANGE

**Question: value?**

**Answer:**

```
In [18]: df['age'].max() - df['age'].min()
Out[18]: 49
```

Range is the difference between the largest and smallest number.

*df['variable'].max() - df['variable'].min()*

### 5.1.2. Interquartile Range (IQR)

**Question: value?**

**Answer:**

```
In [20]: Q1 = df['age'].quantile(0.25)
         Q3 = df['age'].quantile(0.75)
         IQR = Q3 - Q1
         IQR
Out[20]: 16.0
```

Equal to the difference between 75th and 25th percentiles,
or between upper and lower quartiles

*Q1 = df['variable'].quantile(0.25)*

*Q3 = df['variable'].quantile(0.75)*

*IQR = Q3 - Q1*

*IQR*

# Demo 1

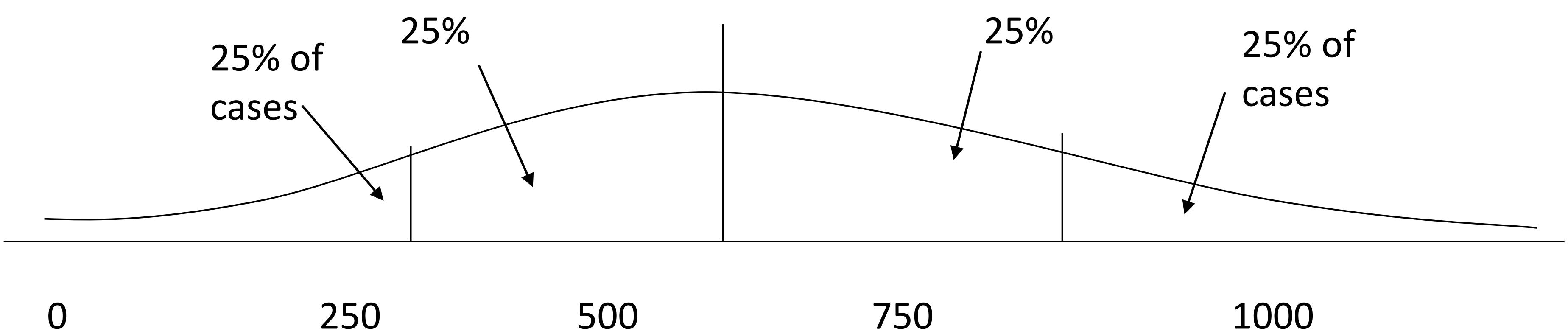
## 5.1.2. Interquartile Range (IQR)

A quartile is the value that marks one of the divisions that breaks a series of values into four equal parts.

The median is a quartile and divides the cases in half.

$25^{th}$ percentile is a quartile that divides the first ¼ of cases from the latter ¾.
$75^{th}$ percentile is a quartile that divides the first ¾ of cases from the latter ¼.

The interquartile range is the distance or range between the $25^{th}$ percentile and the $75^{th}$ percentile. Below, what is the interquartile range?



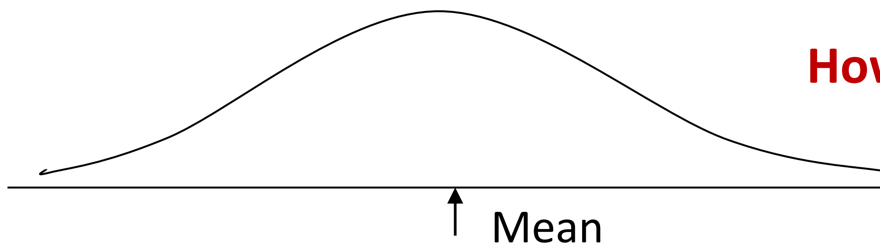
# Demo 1

## 5.1.3. Variance

Variance is difference between each data point and the mean, squared.

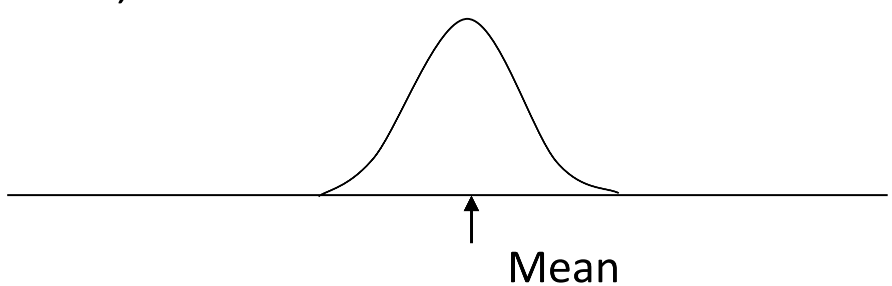
A measure of the spread of the recorded values on a variable.  A measure of dispersion.

The larger the variance, the further the individual cases are from the mean.




**How helpful is that???**

The smaller the variance, the closer the individual scores are to the mean.




**Question: value?**

**Answer:**

```
In [21]: df['age'].var()
Out[21]: 340.79999999999995
```

*df['variable'].var()*

# Demo 1

## 5.1.4. Standard Deviation

To convert variance into something of meaning, let's create standard deviation.

The square root of the variance reveals the average deviation of the observations from the mean.

*df['variable'].std()*

**Question: value?**

**Answer:**

```
In [22]: df['age'].std()

Out[22]: 18.46076921474292
```

## 6. Summary of age

### 6.1.1. Describe

*df['variable'].describe()*

**Question: values?**

**Answer:**

```
In [23]: df['age'].describe()

Out[23]: count     5.000000
         mean     45.400000
         std      18.460769
         min      24.000000
         25%      36.000000
         50%      42.000000
         75%      52.000000
         max      73.000000
         Name: age, dtype: float64
```

**Question: median value?**

# Demo 1

## 7. Graphs

### 7.1. Histogram

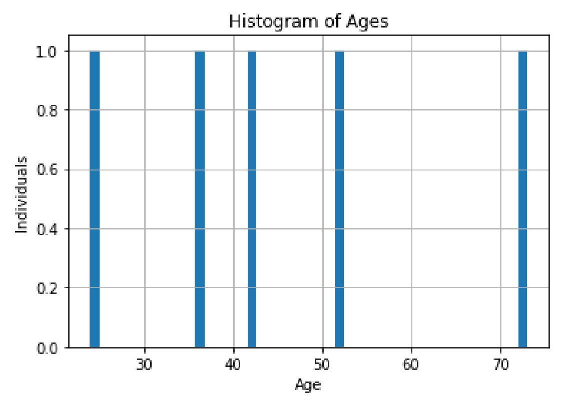
A histogram is an accurate graphical representation of the distribution of numerical data.

*import matplotlib.pyplot as plt*
*df['variable'].hist(bins=50)*
*plt.title("Histogram of Ages")*
*plt.xlabel("Age")*
*plt.ylabel("Individuals")*

**Question: value?**

**Answer:**

```
In [32]: import matplotlib.pyplot as plt
         df['age'].hist(bins=50)
         plt.title("Histogram of Ages")
         plt.xlabel("Age")
         plt.ylabel("Individuals")

Out[32]: <matplotlib.text.Text at 0xa817c50>
```



# Demo 1

## 7. Graphs

### 7.2. Box-and-whisker plots

A way to graphically portray almost all the descriptive statistics at once is the box-plot.

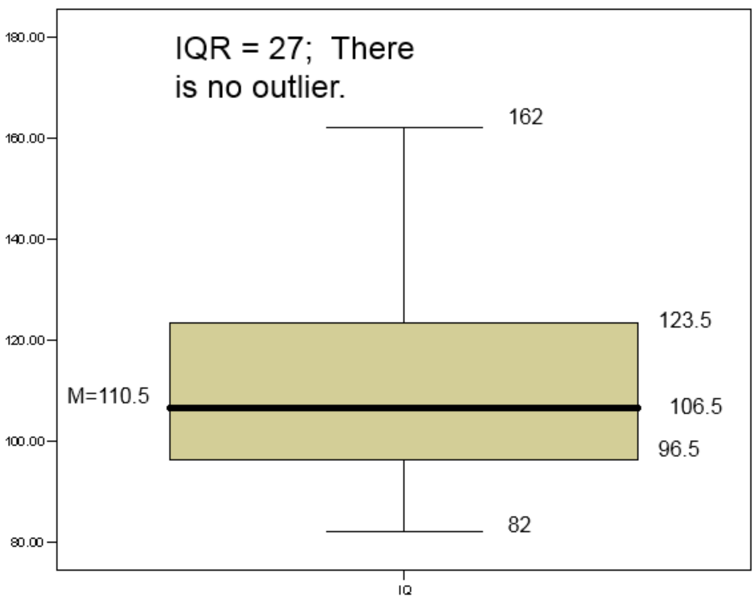
A box-plot shows: Upper and lower quartiles

Mean

Median

Range

Outliers (1.5 IQR) - That is, if a data point is below Q1 – 1.5×IQR or above Q3 + 1.5×IQR, it is viewed as being too far from the central values to be reasonable.



# Demo 1

## 7. Graphs

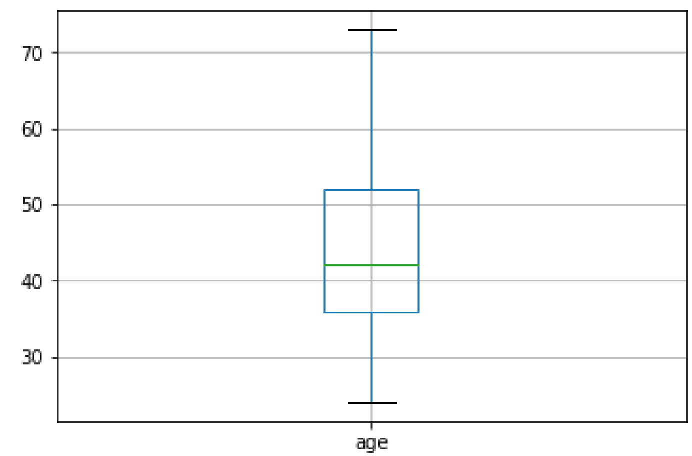
### 7.2. Box-and-whisker plots

*df.boxplot(column='variable', sym='o', return_type='axes')*

**Question: value?**

**Answer:**

```
In [35]:  df.boxplot(column='age', sym='o', return_type='axes')

Out[35]:  <matplotlib.axes._subplots.AxesSubplot at 0xa802278>
```



BTS Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

# Demo 1

## 8. Finding outliers

Standard deviation is helpful because it describes how far away from the mean your data generally is. We can use this to find data points that are usually far from the mean. These are outliers!

*df['variable_std'] = ((df['variable'] - df['variable'].mean()).apply(abs) / df['variable'].std())*
*df.sort_values(by='variable', ascending=False).head(6)*

**Question: value?**

**Answer:**

**How helpful is that???**

In [43]:
```
df['age_std'] = ((df['age'] - df['age'].mean()).apply(abs) / df['age'].std())
df.sort_values(by='age', ascending=False).head(6)
```

Out[43]:

| | name | age | preTestScore | postTestScore | age_std |
|---|---|---|---|---|---|
| 4 | Amy | 73 | 3 | 70 | 1.495062 |
| 1 | Molly | 52 | 24 | 94 | 0.357515 |
| 0 | Jason | 42 | 4 | 25 | 0.184174 |
| 2 | Tina | 36 | 31 | 57 | 0.509188 |
| 3 | Jake | 24 | 2 | 62 | 1.159215 |

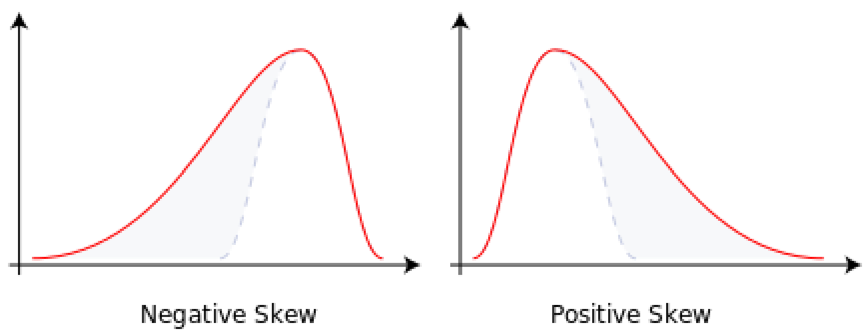
Generally, 3.0 is considered a extreme outlier. 1.5 is considered maybe an outlier, but probably not really. We can see how many standard deviations they are away from the mean.

# Demo 1

## 8. Skewness

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative, or undefined.

- Negative skew: The left tail is longer; the mass of the distribution is concentrated on the right of the figure.
- Positive skew: The right tail is longer; the mass of the distribution is concentrated on the left of the figure.




**Question: value?**

**Answer:**

```
In [44]: df['age'].skew()
Out[44]: 0.70478411035663524
```

*df['variable'].skew()*

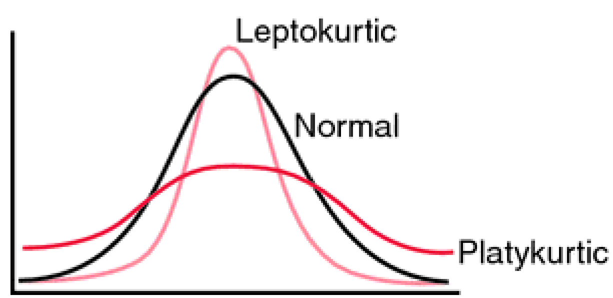
## 9. Kurtosis

# Demo 1

Kurtosis is a measure of the "tailedness" of the probability distribution of a real-valued random variable. In a similar way to the concept of skewness, kurtosis is a descriptor of the shape of a probability distribution and, just as for skewness, there are different ways of quantifying it for a theoretical distribution and corresponding ways of estimating it from a sample from a population.

- The kurtosis of any univariate normal distribution is 3.
- It is common to compare the kurtosis of a distribution to this value.
- Distributions with kurtosis less than 3 are said to be platykurtic, although this does not imply the distribution is "flat-topped" as sometimes reported. Rather, it means the distribution produces fewer and less extreme outliers than does the normal distribution.
- Distributions with kurtosis greater than 3 are said to be leptokurtic.

*df['variable'].kurt ()*




**Question: value?**

**Answer:**

```
In [45]: df['age'].kurt()
Out[45]: 0.6082774747074237
```

# Demo 2

- **Objective:** Mastering skills about Descriptive Statistics.
- **Phyton libraries for Descriptive Statistics**: Pandas, numpy and Matplotlib.

- Clone the Git repository to get an initial code:

https://github.com/FGutierresBTS/BTS_MasterInBigData.git

- Once you downloaded the repository to your local file system, go to the folder "BTS_MasterInBigData/ DataScienceFoundations".
- Copy the folder "Session_6_DSF" into your local folder "DataScienceFoundations".
- In the folder "Session_6_DSF" you will see the files called:
  - *BTS_DataScienceFoundation_Session6_DescriptiveStatistics_Demo2. ipynb*
- Import this file into Jupyter Notebook using the "Upload" button.
- Open the imported script and put your code in the final.
- Export your Demo 2 *.ipynb notebook and push it to your repository "DataScienceFoundations".

# Class Exercices

# Exercise 1

**Exercise 1 - Descriptive Statistics For pandas Dataframe "Advertising Data"**

- Considering the Descriptive Statistics in Demo 2 develop the Descriptive Statistics for the variables “TV”, “radio” and “newspaper”.
  - Dataset: "Advertising" (available at http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv).
  - Interpret and discuss the Results.
  - Commit scripts in your GitHub account. You should export your solution code (.ipynb notebook) and push it to your repository “DataScienceFoundations”.
- The following are the tasks that should complete and synchronize with your repository “DataScienceFoundations” until October 25. Please notice that none of these tasks is graded, however it’s important that you correctly understand and complete them in order to be sure that you won’t have problems with further assignments.

**Guidelines:**

- Clone the Git repository to get an initial code:
  https://github.com/FGutierresBTS/BTS_MasterInBigData.git
- Once you downloaded the repository to your local file system, go to the folder “BTS_MasterInBigData/Session_6_DSF”.
- Copy the folder “Session_6_DSF” into your local folder “DataScienceFoundations”.
- In the folder “Session_6_DSF” you will see the files called:
  - *BTS_DataScienceFoundation_Session6_DescriptiveStatistics_Exercise1.ipynb*
- Import these files into Jupyter Notebook using the “Upload” button.
- Open the imported script and put your code inside the notebook.
- Export your Exercise1 *.ipynb notebook and push it to your repository “DataScienceFoundations”.

# Exercise 2

## Exercise 1 - Descriptive Statistics For pandas Dataframe "Indicadores de fecundidad"

- Considering the Descriptive Statistics in Demo 2 develop the Descriptive Statistics for the variable “Taxa bruta de natalidade”.

  - Dataset: "Indicadores de fecundidad" (available at https://www.europeandataportal.eu/data/en/dataset/http---datos-gob-es-catalogo-a12002994-indicadores-de-fecundidad).
  - Interpret and discuss the Results.
  - Commit scripts in your GitHub account. You should export your solution code (.ipynb notebook) and push it to your repository “DataScienceFoundations”.
- The following are the tasks that should complete and synchronize with your repository “DataScienceFoundations” until October 25. Please notice that none of these tasks is graded, however it’s important that you correctly understand and complete them in order to be sure that you won’t have problems with further assignments.

## Guidelines:

- Clone the Git repository to get an initial code:

  https://github.com/FGutierresBTS/BTS_MasterInBigData.git
- Once you downloaded the repository to your local file system, go to the folder “BTS_MasterInBigData/Session_6_DSF”.
- Copy the folder “Session_6_DSF” into your local folder “DataScienceFoundations”.
- In the folder “Session_6_DSF” you will see the files called:

  - *BTS_DataScienceFoundation_Session6_DescriptiveStatistics_Exercise2.ipynb*
  - *"2_hdat724.csv"*.
- Import these files into Jupyter Notebook using the “Upload” button.
- Open the imported script and put your code inside the notebook.
- Export your Exercise2 *.ipynb notebook and push it to your repository “DataScienceFoundations”.

# Individual assignment

# Individual assignment

- Considering the Descriptive Statistics presented in Demo 2 develop a descriptive statistics for the data set "Titanic".
- Commit scripts in your GitHub account. You should export your your solution code (.ipynb notebook) and push it to your repository "DataScienceFoundations".
- The following are the tasks that should complete and synchronize with your repository "DataScienceFoundations" until October 25.

**<u>Guidelines:</u>**

- Clone the Git repository to get an initial code:
  https://github.com/FGutierresBTS/BTS_MasterInBigData.git
- Once you downloaded the repository to your local file system, go to the folder "BTS_MasterInBigData/ Session_6_DSF".
- Copy the folder "Session_6_DSF" into your local folder "DataScienceFoundations".
- In the folder "Session_6_DSF" you will see the files called:
  - *BTS_DataScienceFoundation_Session6_DescriptiveStatistics_Assignment.ipynb*
  - *"1_titanic_dataset.csv" (note: this dataset must be pre-processed).*
- Import these files into Jupyter Notebook using the "Upload" button.
- Open the imported script and put your code inside the notebook.
- Export your Assignment *.ipynb notebook and push it to your repository "DataScienceFoundations".


# Thank you

Barcelona, 2017