# ‘Data Analysis’
# Computer Lab Session
# General Introduction to Data Analysis with Python

---

## Master 1 MLDM / CPS$^2$ / COSI / 3DMT
## Saint-Étienne, France

Ievgen Redko

## 1 Outline

1. Introduction to Python (1/2)
2. Introduction to Python (2/2)
3. **Probability, Random Variables and Probability Distributions**
4. Linear Algebra (1/2)
5. Linear Algebra (2/2)
6. Principal Component Analysis
7. Linear Regression (1/2)
8. Linear Regression (2/2)
9. Clustering (1/2)
10. Clustering (2/2)

### Outcome

The objective of this lab is to become familiar with Python functions for working with probabilities, random variables and probability distribution.

## 2 Warm-up

### Exercise 1: Heads or tails?

The probability of an event $E$ (possibly unknown), denoted by $\mathrm{P}(E)$, is defined to be the value approached by the relative frequency of occurrence of $E$ in a very long series of trials of a chance experiment.

With Python, we will represent the relative frequency of heads as a function of the number of tosses. In Python, the rand function will generate random values drawn from uniform distribution (it is part of the

numpy.random module to be imported) from the interval $[0, 1)$. We imagine that the coin is not loaded. They are 2 events: heads or tails, each having the same chance $= \frac{1}{2}$.

1. Print 10 random values 0 or 1 (0 for the event 'tails', 1 for the event 'heads') by using the rand and round functions. The latter is a built-in function.

2. Represent the relative frequency of the number of heads as a function of the number of tosses (for 1 to 1000 tosses). For this, create a vector called $proba$ and a variable $sum$ with its initial value set to 0. The $proba$ value will be assigned by the relative frequency of the number of heads computed by the number of heads (given by $sum$) divided by the number of tosses.

   Then plot this relative frequency and add a red horizontal line at the 0.5 level.

3. Compare your result with another run of the algorithm. Modify further your Python code for having the representation of the relative frequency of the number of heads as a function of the number of tosses for 1 to 10,000 tosses (instead of 1,000 tosses). What is the difference between the new result and the previous one?

**Remark 1** *In Python, you can use the numpy.random.seed command to force random function to give the same result for different runs of the algorithm.*

# 3 Random Variables and Probability Distributions

## Exercise 2: Probability Distribution for Discrete Random Variables

1. **One dice**: The total number of events from rolling a dice is equal to 6 for each side of the dice. The probability for having the value 1, 2, 3... or 6 is the same and equals to $\frac{1}{6}$. Represent the probability histogram of the results obtained by a dice over 1000 runs and compare it with the true probabilities.

   **Remark 2** *Use numpy.bincount command to count the number of each possible outcome, np.random.randint to generate the random integers from the prescribed range and matplotlib.pyplot.bar command to plot the corresponding probabilities.*

2. **Two dices**: Do the same for a problem with two dices. Before implementing the solution, first find out the total number of events and the maximal score that can be obtained with two dices.

3. **Mean and variance**: Implement two functions that calculate the mean and variance of a discrete random variable. These latter are defined as follows:

$$\mathbb{E}(X) = \sum_{\text{all possible } x \text{ values}} x \times \mathrm{P}(x)$$

$$\sigma_X^2 = \mathrm{V}(X) = \sum_{\text{all possible } x \text{ values}} [x - \mathbb{E}(X)]^2 \times \mathrm{P}(x)$$

   Apply them to the observations from the two cases considered above. Compare further the obtained results with the theoretical values.

4. **Correlation**: Generate a sample $X$ of 1000 random integers from $[0, 50]$ interval. Define further a variable $Y = X + \mathcal{N}(0, 10^2)$ using np.random.normal command. Calculate the correlation between $X$ and $Y$ using numpy.corrcoef command. What do you observe? What modification can you do to obtain negative correlation? No correlation? Use scatter plots to confirm your intuition.

# 4 Probability Distribution for Continuous Random Variables

## Exercise 3: Correlation Coefficient for Continuous Random Variables

1. **Load Iris data set**: Import (or install if not available) the seaborn library. Use it to load the famous Iris data set by executing the following command:

```
import seaborn as sns
iris = sns.load_dataset("iris")
```

   This data set contains the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris.

2. **Visualize data**: Use pairwise plots to visualize the Iris data set. For this, use the pairplot command. For more details, check this page `https://seaborn.pydata.org/generated/seaborn.pairplot.html`. The obtained chart contains a lot of information:

   - On the diagonal are the univariate distributions, plotted as histograms and kernel density plots (if diag_kind="kde" is used as option).
   - On the right of the diagonal are the pair-wise relationships allowing to see the correlation
   - On the left side of the diagonal is the scatter-plot matrix where you can illustrate the underlying relationship using linear regression fitting (kind="reg" option).

3. **Analysis**: What are the most correlated feature pairs? What feature pairs allow to separate the different classes of data?

## 4.1 Binomial, Geometric and Poisson Distributions

### 4.1.1 Binomial Distribution $\mathcal{B}(n, p)$

The *binomial random* variable $X$ is defined as $X =$ number of successes observed among $n$ trials. Then, if $X$ is a binomial variable (noted $X \equiv \mathcal{B}(n, p)$), we get:

$$\mathrm{P}(X = x) = p(x) = \binom{n}{x} p^x (1-p)^{n-x} = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

where $\binom{n}{x}$ is read as "$n$ choose $x$" and represents the number of ways of choosing $x$ items from a set of $n$. Note that for $n = 1$, the binomial variable is equivalent to the Bernoulli variable $\mathcal{B}(1, p)$.

$$\mathrm{E}(X) = np$$

$$\mathrm{V}(X) = np(1-p)$$

In Python, the relevant functions for a binomially distributed random variable $X$ for $n$ trials and with success probability $p$ are part of scipy.stats.binom module. Notably, one has:

- pmf$(k, n, p)$, to find $\mathrm{P}(X = k)$
- cdf$(k, n, p)$, to find $\mathrm{P}(X \leq k)$
- ppf$(q, n, p)$, to find $c$ such that $\mathrm{P}(X \leq c) = q$
- rvs(n,p), to generate $n$ independent values of $X$

Example: Compute the probability of getting four heads in six tosses of a fair coin.

```
binom.pmf(k = 4, n = 6, p = 0.5)
```

We will obtain the value 0.234375. Thus, $\mathrm{P}(X = 4) = 0.234$, when $X$ is a binomial random variable with $n = 6$ and $p = 0.5$.

Cumulative probabilities of the form $\mathrm{P}(X \leq x)$ can be computed using cdf(); this function takes the same arguments as pmf(). For example, we can calculate $\mathrm{P}(X \leq 4)$ where $X$ is the number of heads obtained in six tosses of a fair coin as:

```
binom.cdf(k = 4, n = 6, p = 0.5)
```

### 4.1.2 Geometric Distribution $\mathcal{G}(p)$

A *geometric random* variable $\mathcal{G}(p)$ is defined as $X$: number of trials until the first success is observed (including the success trial). The probability distribution of $X$ is called the geometric probability distribution. If $X$ is a geometric random variable with probability of success $p$ for each trial, i.e. $X \equiv \mathcal{G}(p)$, then

$$\mathrm{P}(X = x) = (1-p)^{x-1}p$$

$$\mathrm{E}(X) = \frac{1}{p} \text{ and } \mathrm{V}(X) = \frac{1-p}{p^2}$$

In Python, one can manipulate this distribution in the same manner as the binomial one. All the function related to it are provided as part of the module scipy.stats.geom.

## 4.2 Poisson Distribution $\mathcal{P}(\lambda)$

The Poisson distribution $\mathcal{P}(\lambda)$ is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event. The Poisson distribution can also be used for the number of events in other specified intervals such as distance, area or volume.

The Poisson distribution is the probability distribution of independent event occurrences in an interval. A discrete random variable $X$ is said to have a Poisson distribution with parameter $\lambda > 0$, if for $x = 0, 1, 2, \ldots$ the probability mass function of $X$ is given by:

$$\mathrm{P}(X = x) = \frac{\lambda^x e^{-\lambda}}{x!} \text{ where } x = 1, 2, 3, \ldots$$

The positive real number $\lambda$ is equal to the expected value of $X$ and also to its variance:

$$\lambda = \mathrm{E}(X) = \mathrm{V}(X).$$

In Python, the module scipy.stats.poisson can be used to work with Poisson distribution.

## 4.3 Normal Distribution $\mathcal{N}(\mu, \sigma)$

A Normal distribution $\mathcal{N}(\mu, \sigma)$ is bell-shaped and symmetric. It is characterized by a mean $\mu$ and standard deviation $\sigma$. $\mu$ describes where the corresponding curve is centered, and $\sigma$ describes how much the curve spreads out around that center.

$$\mathrm{P}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

If we let the mean $\mu = 0$ and the standard deviation $\sigma = 1$ in the *probability density function*, we get the probability density function for the *standard normal distribution*.

## Exercise 4: Calculating probabilities of events for different laws

1. Plot the probability mass function of the binomial distributions for $p = 1/2$ and $n = 5, 10, 15$ and 20, then the probability mass function of the binomial distributions for $n = 10$ and $p = 1/2, p = 1/3, p = 1/4, p = 1/5$.
2. Find the probability of having four or less correct answers from twelve multiple choice questions with five possible answers if a student attempts to answer every question at random.
3. Identify and plot the regions where a random variable from the standard normal distribution occurs within one, two and three standard deviations of the mean. Use interval(percent, mean, std) to do this.
4. If there are twelve cars crossing a bridge per minute on average, find the probability of having seventeen or more cars crossing the bridge in a particular minute.
5. A normal distribution with mean = 3500 grams and standard deviation = 600 grams is a reasonable model for the probability distribution of the continuous variable $X$: birth weight of a randomly selected full-term baby.
   - What proportion of birth weights are between 2900 and 4700 grams?
   - What birth weight $w$ is exceeded only in 2.5% of the cases?

## Exercise 5: Students data analysis

1. Download the “students_data.npy" file from *Claroline* (on your working directory) and import it into Python:

```
data = np.load("students_data.npy")
```

2. Print the coefficient correlations between all the continuous attributes. What are the significant correlations?
3. Plot the histogram of the size of the students (first variable).
4. Compute the mean and standard deviation of the size variable. Does it follow a normal distribution?
5. By using a normal approximation of the size, how many students do you expect to find in the classroom with a size larger or equal to 180 cm? How many are they in reality?
6. Plot the histogram of the age of the students calculated based on their year of birth (second column).
7. Compute the mean and standard deviation of the age variable. Does it follow a normal distribution?
8. By using a normal approximation of the age, how many students do you expect to find in the classroom with an age smaller that the mean less one standard deviation? And with an age smaller that the mean less two standard deviations? How many are they in reality?