# PHYLOGENY – TME3

Academic Year 2023/2024

Marina Abakarova
marina.abakarova@sorbonne-universite.fr

19 October 2023

**General rules**

- Reports must be sent by e-mail, using the subject "`[PHYG] TME3`", including in the body the names of the persons who worked on it (maximum two students per group). The deadline is 26th of October.
- Multiple files should be grouped in a compressed archive (`.tar.gz` or `.zip`).
- Your report *must be* in PDF format and named `student1_student2_TME3.pdf`. It should be simple, clear and well organized. Answers should be given in an exhaustive manner. Consider adding at the beginning a summary indicating the page of each answer.
- Source code must be well explained, commented and, most importantly, it should work without errors. Provide all needed information (*e.g.,* compiler/interpreter version) in a `README` file.
- All required materials can be found in the repository `https://github.com/abakarovaMarina/PHYG2023`
- A discord server is created so we can exchange our questions, answers and comments `https://discord.gg/w7JUvJV4`.

## Exercise 1 – Genome evolution simulation

1. What is necessary to build a model for simulating the evolution of genomes? What kind of parameters the model has to take into account?
2. Why is the choice of parameters important?

3. Can we use the same parameters to simulate the evolution in different clades? Justify your answer.

4. Explain the Poisson distribution. What is the only parameter of this distribution? How is it related to the binomial distribution? Can we derive the probability mass function (PMF) of the Poisson distribution using the PMF of the binomial distribution? Justify your answer. The PMF of binomial and Poisson distributions are given below.

   $\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$

   $\mathbb{P}(X = k) = \frac{\lambda^k}{k!} \mathrm{e}^{-\lambda}$

## Exercise 2 – Implementation of simple events

Consider the implementation of a genome as a python dictionary where *keys* are *positive integers* (identifying chromosomes) and *values* are lists of *signed integers* (identifying genes). Gene lists also contain the value `0` (the centromere) exactly once. The following is an example of a genome `g` with two chromosomes containing 18 and 7 genes, respectively:

```
genome = {
   1: [1,0,2,3,4,5,6,7,8,-10,-9,11,-16,-15,-25,22,23,24,-21]
   2: [-20,0,-19,-18,-17,12,13,14]
}
```

Consider now the file `tme3.py` in which you can find the definition of the following functions:

- `poisson(lam)`: returns an integer value according to a Poisson distribution with parameter $\lambda =$ `lam`.
- `choose_coordinates(g)`: it returns a pair `(chrNum,chrPos)`, where `chrNum` identifies a (randomly chosen) chromosome of the input genome `g` and `chrPos` is a (random) index over the list of genes `g[chrNum]`.
- `inversion(g,mean_inv_len)`: it returns the genome which can be obtained after the application of an *inversion* event to the input genome `g`. The parameter `mean_inv_len` defines the $\lambda$ parameter of the Poisson distribution used to choose the number of genes involved in the inversion.
- `deletion(g,mean_del_len)`: it returns the genome which can be obtained after the application of a *deletion* event to the input genome `g`. The parameter `mean_del_len` defines the $\lambda$ parameter of the Poisson distribution used to choose the number of genes involved in the deletion.
- `fission(g)`: it returns the genome which can be obtained after the application of a *fission* event to the input genome `g`.

Using the first two functions (which are already defined), complete the python file `tme3.py`, defining the functions which implement *inversion*, *deletion*, and *fission* events (see file `simulation.pdf` for more details). In order to test your implementation, after completing all the three functions, the command

```
python tme3.py
```

should perform the three operations on a test genome.

## Exercise 3 – Small dataset simulation

Run the python script `sim.py` (which *requires* all functions of Exercise 2) to run 10 simulations. The script should output each time 5 species from a (randomly generated) ancestor genome with 100 genes, 4 chromosomes with 5 average events (run the command "`python sim.py -h`" for help running the script). The output of each simulation is written in the `results` sub-folder where you can find the simulated genomes (`genomes.txt`), some statistics (`statistics.txt`), and the simulated tree (`tree-newick.txt`).

1. Include the directory of the simulations to your report (as a `.zip` or `.tar.gz` file).
2. Write a table which contains, for each simulation, the number/type of events that have occurred. Which one is the most frequent? Which is the least frequent?
3. Is there any event that did not occur in any simulation? If yes, what is the reason?
4. Choose two simulations and reconstruct a phylogenetic tree using the Maximum Likelihood based Gene Order Analysis (MLGO). In order to achieve this task, you can simply go to the web server `http://www.geneorder.org` and use the file `genomes.txt` as *Input Data*. Then, choose the *Phylogenetic Tree Reconstruction*, and put your e-mail in the field *Notification*. As a results, you will receive the trees as files named `gene_order.tree`.
5. Use the web service `http://itol.embl.de/upload.cgi` to visualize both the simulated and reconstructed trees. Insert and compare them in your report.

## Exercise 4 – Large dataset simulation

1-4. Repeat all four points of Exercise 3, simulating this time 13 species from a (randomly generated) ancestor genome with 18 000 genes, 23 chromosomes with 1000 average events.

5. Compare the results obtained with MLGO in this exercise with those obtained in Exercise 3.