Assignment 1: Game theory

General remarks:

- The deadline to hand in the assignment is 01-11-2020 by 23:59.
- You must submit your assignment through GitHub classroom. You **MUST** connect to the link https://classroom.github.com/a/Ni9tBx9k , and push your assignment to your provide repository within the required time.
- Creating the repository might take some time! If it does not appear to work, refresh the page! If you do not manage to create the repository, send us an email (elias.fernandez.domingos@vub.be).
- Provide a single (self-contained) *.PDF file, named <Surname>_<Name>_<studentID>_<affiliation>.pdf. Your name and surname should start with capital letter (e.g. Wim).
- Provide a file called "moran_process.[ext]", where [ext] is a placeholder for the programming language.
- Put your name, student ID and your affiliation (VUB/ULB) inside the document.
- Don't send in photos of your results, scan them if necessary!
- Any doubts should be emailed to Elias Fernández <eliferna@vub.be>.

Note: If these guidelines are not respected, your assignment will not be corrected and your grade will be 0. Also, You will not be able to push any new changes to your private repository after the deadline.

Guidelines to use the provided code:

The code consists on 2 main files:

1. **CGT-Exercise.ipynb** : contains a jupyter notebook with the explanation of the parts of the assignment that require a computational approach.
2. **EGTtools** package: This package is included as a git submodule in the repository provided to you. You may find the original repository at https://github.com/Socrats/EGTTools and you can import it in your project with "import egttools". When you clone your repository, you need to use "git clone –recurse-submodules -j8 <link to repository>" so that the submodules are also downloaded. If you have problems with this contact us.

You will also find some extra files

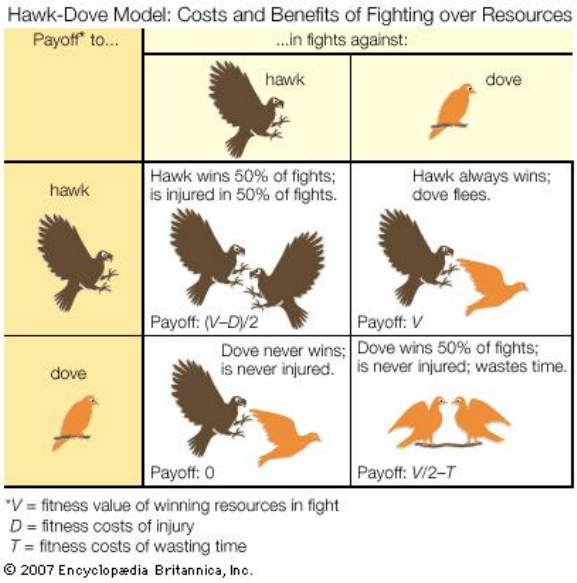
3. requirements.txt : contains the names of the packages required to run the code
4. README.md : contains some extra information on how to run the code
5. .gitignore : contains the files that will be ignored by git

- In Exercise 3, you will need write some code. Please do this in a **separate** file that **MUST** be named "moran_process.[extension]". You may write your code in Python, C, C++, Java or Julia. If you do not use python, you must also include all files required to compile your code. In case of C/C++/Java you may include extra files, but the "main" function should still be contained in a

file name “moran_process.[extension]”. You must submit these files together with your assignment.

- To be able to run the code, you need to have Python installed. We recommend Python 3.7, but it should work in previous versions as well.
- The EGT library requires the Numpy, Matplotlib, NashPy and SciPy, Jupyter and ipywidgets, packages to be installed. We recommend you to install them in a virtual environment by running in the command line “pip install requirements.txt”.

The Hawk-Dove game (3 pts)



The Hawk-Dove game is a coordination game formulated by John Maynard Smith and Georg Price. The aim of the game was to understand the resolution of conflicts by fighting in the animal kingdom. The game consists of two players. Each have the choice between two possible actions; either they take time to display (dove) before fighting or they can escalate immediately and fight (hawk). When both players escalate (hawk), they have a 50% risk of being injured (-D/2) and 50% of wining (V/2). When a dove fights another dove she also wins 50% of the time (V/2) but only after a period of mutual displays to show of strength (-T). Hawks always win against doves, resulting in a benefit for one (V) and not for the other (0).

1. Find all the (mixed strategy) Nash equilibria of this game. How do the results change when the order of the parameters V, D and T is changed (V>D, D>T, etc.)?
2. Under which conditions does displaying become more beneficial than escalating? Draw the set of all mixed strategies.
3. Validate your results using NashPy. You may use the example provided in the **CGT-Exercise.ipynb**. Indicate here the Nash equilibria found for V=2, D=3 and T = 1.

Which social dilemma? (3 pts)

Player A knows he’s confronted with one of three social dilemma’s; a prisoner’s dilemma, a snowdrift game or stag-hunt game (see above). In each game he needs to decide whether to cooperate (C) or defect (D), yet he is not sure in which he actually is. He’s sure that each game is equally likely. The other player, player B, knows in which game he’s playing. Determine the pure Nash equilibria using the Bayesian game analysis discussed in the course.

Prisonners dilemma

	C	D
C	2,2	0,5
D	5,0	1,1

Stag-Hunt game

	C	D
C	5,5	0,2
D	2,0	1,1

Snowdrift game

	C	D
C	2,2	1,5
D	5,1	0,0

Evolutionary Dynamics in the Hawk-Dove game (4 pts)

For this exercise, you will need to use the code provided to you, and follow the **CGT-Exercise.ipynb**. Follow the instructions indicated in the notebook, and provide answers to the questions stated here. You are going to study the evolutionary dynamics of the Hawk-Dove game both in infinite and finite populations.

1. Look at the plot of the gradient of selection for infinite populations, explain which saddle points are stable and which aren't, and why. Do the results here agree with those found in Exercise 1? Do you expect any changes if the population is finite?
2. For finite populations, how are the dynamics affected when changing:
    a. Population size (Z)
    b. Intensity of selection (β)
    c. Probability of mutation (μ)

    Reason each answer, explaining why a change should be expected.
3. Find the stationary distribution of the Hawk-Dove game for finite populations numerically. Assume that $\beta = 10$ and $\mu = 10^{-3}$. For this, you will need to
    a. Implement the Moran process with pairwise-comparison explained in CGT-Exercise.ipynb (and in the course).
    b. Implement a Monte Carlo simulation that estimates the stationary distribution:
        i. Start from a random population state
        ii. Run the Moran process for a transitory period of 10^3 generations.
        iii. Afterwards, run the process for 10^5 generations and count how often the population passes through each possible state.
        iv. Repeat the simulation 10 times and average the results
    c. Plot the stationary distribution

    Compare the results with the analytical approach. Do you find any differences? Why? Please follow the guidelines indicated in the Jupyter notebook.

Note: Exercise 3.3 can take a long time to run (if implemented in Python). Please, bear this in mind and finish it with time!

Success!