Computing Methods for Physics 1

Lecturer: Francesco Pannarale

Hands-on lab session 4A, A.Y. 2021-22

The goal of this session is to practice writing and handling classes in Python and performing a quick fit with `scipy.optimize.curve_fit`.

Once again you will simulate games of Kniffel and CLR, just as in LAB 3A. This should help you recycle several lines of code so you can focus on class syntax. As a reminder:

- **Kniffel** — This game is played with 5 dice with 6 sides numbered 1 through 6 all rolled at once. Our simplified version of the game requires rolling the 5 dice until each of the following six special combinations is achieved: the dice show $1, 2, 3, 4, 5$, they show $2, 3, 4, 5, 6$, they all show the same number (this is called a kniffel), only 3 of them show the same number, only 4 of them show the same number, 3 of them show the same number and the remaining 2 show another number [this is a full house, e.g., $(2, 2, 2, 5, 5)$].
- **CLR** — This game is played by $n > 2$ players with c coins each, to begin with. The players roll the dice in turns and the outcome of the dice roll determines the fate of their coins. A player that has 3 or more coins rolls 3 dice, a player with 2 coins left rolls 2 dice, a player with 1 coin left rolls 1 die, and a player without coins cannot roll. Each dice has 6 faces: 1 showing a C, 1 showing an L, 1 showing and R, and 3 showing a dot. For every L/R a player rolls, a coin goes to the player on their left/right; for every C a player rolls, one of their coins is removed from the game; rolling a dot has no consequences. The last player left with coins wins.

Instructions

Classes

- Build a class to represent a die. In both games there are 6 faces to a die, but the values are different: the tuple of face values will be an attribute, and a fair roll will be a method. You can add other features, but the two described are the essential ones you must have. **Use `random` and not `numpy.random` to roll the dice: it is less sophisticated and hence faster.**
- Build a class to simulate a single Kniffel game. Make sure you have a `__str__` method to help you debug your code. Make sure a game is played using an instance (or instances, you decide) of the class written to represent a die.
- Build a class to simulate a single CLR game: the same tips as for Kniffel apply, but make sure the class is flexible enough for you to simulate games with any $n > 2$ and any $c > 0$.
- Run a game of each kind. You should be well below 100 ms (which you can check in Jupyter with the magic command `%% time`).

Data production

- Run 10^4 Kniffel games and 10^5 CLR games to gather information on the number of rounds each game took to come to an end. The runtimes will be higher than those achieved in LAB 3A, but our focus are classes this time. **As for previous programs, start with smaller numbers to check your runtimes are not too long.**

Data processing

- Plot the density distribution of number of rounds obtained as a histogram.
- Fit the CLR density distribution with a Gaussian funtion and determine the expectation value of number of rounds to be played, as well as the (1 sigma) uncertainty on this number. If your density distribution does not look like a Gaussian, something is wrong with your simulations.
- Fit the Kniffel density distribution with a Poisson function and determine the mean number of rounds to be played. If your density distribution does not look like a Poisson function, something is wrong with your simulations.
- Overlay the density distributions and your fits. Mark the relevant interesting quantities you calculated on the plots.

Recommendations

- Keep in mind that CLR will be faster to run on average, but harder to write because you have to track update the coins the players have, check carefully for a winner, decide how many dice to roll, etc.
- Comment your notebook by providing your classes and thier methods with help strings (the messages you put at the very top within 3 quotation marks).
- Break down a game into many methods (roll, play a round, etc.
- Use numpy arrays as much as you can.
- Label your plots and their axes.

If you are done and bored

- Finish any lab session work you have left undone.
- Animate a CLR game. Create a histogram showing the number of coins per player and animate it as the game is carried out.