Monte Carlo Simulation of a Golf Tournament

1 Introduction

In this exercise, we would like you to write code to perform a Monte Carlo simulation of a golf tournament. Your code should take a list of golfers, along with some parameters describing their golfing ability, and determine the probability of winning, and of finishing in the top five, for each of them. The scoring rules for a golf tournament, and the process of Monte Carlo simulation, are explained below.

Please do not spend more than one hour or so on this task. Your main focus should be on the correctness of the solution rather than, say, performance or code quality, although you may be asked about those things if you are invited to a follow up interview.

2 Structure of a Golf Tournament

Golf is a game in which a number of players attempt to hit a ball into a hole, using a club. A *round* of golf consists of 18 holes, and a golf *tournament* consists of 4 *rounds*. Each player scores a point every time they hit the ball with their club, and their score for the tournament is the sum of their scores for all holes in all rounds. The expected number of points per round, in general, is about 72, so a golfer's score for a whole tournament will be about 288.

The player with the lowest score wins the tournament.

In your simulation, you should assume that a golfer's score for a *round* is given by a normal (Gaussian) distribution. The mean and standard deviation of this distribution form our "rating" for that golfer. You should have received with this document a file containing the names and ratings of twenty golfers. That file should be the input to your simulation.

2.1 Determining the Winner and Top 5

Imagine that 8 golfers compete in a tournament and get the scores shown in table 1. In this tournament, every player got a unique score, so there are no ties. Rory McIlroy got the lowest score, and so wins outright. If we had placed a bet on McIlroy to win the tournament, we would get the entire payout, so we say his win fraction is 1, while everyone else's is 0.

Similarly, McIlroy, Spieth, Garcia, Woods, and Perry are the only players in the top five. If we had placed a bet on any of them to finish in the top 5, then we would again get the whole payout. Each of them has a "top 5 fraction" of 1, while everyone below them in the table has a "top 5 fraction" of 0.

The situation is a little more complicated if golfers are tied. Consider the leaderboard in table 2. McIlroy and Spieth are joint winners, each scoring 288. If we had bet on either of them to win, our payout would be reduced by one half. Each of them gets a "win fraction" of 1/2, because there is one spot shared between two players.

Table 1: Golf tournament with no ties

Golfer	Tournament Score	Win Fraction	Top 5 Fraction
Rory McIlroy	288	1	1
Jordan Spieth	289	0	1
Sergio Garcia	290	0	1
Tiger Woods	291	0	1
Kenny Perry	292	0	1
John Cook	293	0	0
Jaco Van Zyl	294	0	0
Justin Rose	295	0	0

Table 2: Golf tournament with tied scores

Golfer	Tournament Score	Win Fraction	Top 5 Fraction
Rory McIlroy	288	1/2	1
Jordan Spieth	288	1/2	1
Sergio Garcia	290	0	1
Tiger Woods	291	0	2/3
Kenny Perry	291	0	2/3
John Cook	291	0	2/3
Jaco Van Zyl	294	0	0
Justin Rose	295	0	0

We consider an N-way tie in an outright winner market to give 1/N wins to each player involved.

When it comes to considering the top 5 finish, we can see that McIlroy, Spieth, and Garcia should all definitely be in the top 5. The next 2 spots in the top five, however (positions four and five), must be shared between the three players who scored 291 - Woods, Perry, and Cook. Each of them gets a "top 5 fraction" of two thirds.

If N players are tied for P places in the top 5, each player is considered to be in the top 5 with fraction P/N.

3 Monte Carlo Simulations

A Monte Carlo simulation is a method to calculate properties of a probability distribution by randomly sampling values from it. In this exercise, we have a probability distribution for each golfer that describes their score in a single round. The algorithm for a single iteration of the Monte Carlo process is:

1. Generate a random score for a whole tournament for each golfer. The scores should be drawn according to the score probability distribution for each golfer.
2. Determine the win fraction and top 5 fraction for each player

The final output of your simulation should be the win fraction and top 5 fraction for each player, averaged over many such iterations. 1000 iterations should be enough for the purposes of this exercise.

4 Your Solution

You may use any language of your choice for this exercise, as well as any libraries that you wish – you should also feel free to look things up online while completing it.

In Python, to generate a random number that falls within a normal distribution, you can use random.normalvariate(mean, stdev) from the standard library. If the language you choose doesn't have a built-in function for generating normally-distributed random numbers, go ahead and copy any code you find online instead of writing the function yourself.

When you have completed your solution, please return to us, by e-mail:

- The complete source code of your solution
- Brief instructions on how we can run the solution. You should assume the solution will be read by a Python developer working on a Linux desktop. This should include a list of any packages that are required.
- Example output from your solution