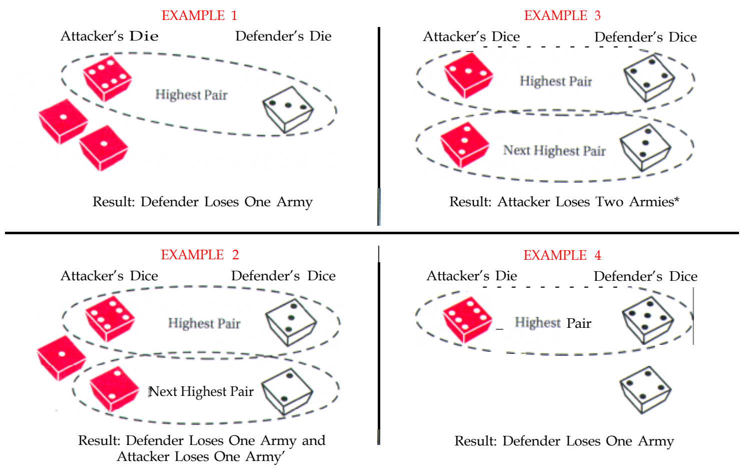
Physics 102 Spring 2018

4/5 in Class – More Simulations

More practice with using the random number generator as part of simulating things with randomness built in.

1. **A simplified Risk attack**: (This problem was number 4 on the worksheet from 4/3.) Risk is a game where you use dice to determine the outcome when one player attacks the other. The number of armies you control dictate the number of dice you can roll, up to a limit of three dice for the attacker (red) and two dice for the defender (white). (Advantage to the attacker here, they might get an extra die.) Let's say that you have to have one army for each die that you roll, and that you must roll the number of dice = to the number of armies you have, up to three dice for the attacker and two for defender. Once the attacker is down to two armies, they can only roll two dice. Once the defender is down to one, they can only roll one die. (These last three sentences are an oversimplification of the real rules to make it simpler to code.)

   Both players roll at the same time. You compare the dice in pairs to determine the winner. First you take the highest of each players rolls and compare them. If the attacker's highest roll is higher than the defender, the defender loses one army. If the defender's roll is equal or higher, the attacker loses an army. (Advantage to defender here, since they win on a tie.) Then you take the next highest of the attacker, and the next highest (only other one) of the defender and do the same comparison.



   Let's consider the case where the attacker has 5 armies and the defender has 5 armies. (You might not attack in the real game if you had a choice, but let's see what happens.)

   (a) Go to the board and work together to write a detailed structure plan for how you will code a simulation of this attack.

   (b) Code it. Call it `riskAttack.m`

   (c) Run the simulation of this case lots of times and predict who will win and what is the probability of that person winning. (The winner is the one who keeps an army at the end.) If you didn't plan on this at the beginning, how will you add it now?

Hint: at some point, you might want to use the `while` command. It is a bit like a `for` loop, but you don't need to know how many times it will run. For example, you can keep doing something `while` the number of armies > 0.

2. A muon is a particle that is often created during collisions between cosmic rays and the earth's atmosphere. A muon is a bit like a heavy electron (about 200 times heavier, with the same charge). Unlike an electron, a muon is unstable, meaning it decays. The muon has a mean lifetime, τ, of 2.197×10^{-6} s. The decay rate per unit time, Γ, is equal to the inverse of the mean lifetime.

   $$\Gamma = \frac{1}{\tau}$$

   The probability (P) that any given muon will decay in some time interval, dt, is $P = \Gamma dt$. Imagine starting with 1000 muons. Your task is to run a simulation using the random number generator to determine if a given muon decays, and plot the number of muons as a function of time as detailed below.

   (a) Go to the board and work together to write a detailed structure plan for how you will code a simulation of the decay of 1000 muons. Be sure your structure plan answers:

      i. What do you think is a reasonable size for the time interval, dt?

      ii. What do you think is a reasonable time scale for the entire simulation to run?

      iii. What constants do you need?

      iv. Will you use for loops or vectorize? Explain how you will implement either in words.

   (b) Code it. This could be iterative. Go back to your group and your structure plan if you need a new one! Call it `muonDecay.m`

   (c) Run the simulation of this case and plot the number of particles as a function of time.

   (d) Write out your data for t and N as two columns in a data file called `muonDecay.txt`

3. Use the information given in the last problem to show that

   $$N(t) = N_0 e^{-\Gamma t}$$

   where $N(t)$ is the number of particles left at any time, t, N_0 is the number you start with, and Γ is the decay rate per unit time. Please do this on the board with a group. Hint: the probability (given in last problem) is equal to the fraction of the sample that will decay in that time interval, or,

   $$\frac{dN}{N} = -\Gamma dt$$

   The negative sign indicates the number is going down.

   Plot this function on top of your simulated data from the last problem. How does it look?

4. Now let's get some more practice with `fminsearch`. Let's fit an exponential to your simulated data, using the least squares method, and see how close we get the parameters of N_0 and Γ.

   (a) Go to the board and work together to write a detailed structure plan for how you will code this.

   (b) Code it. Call it `fitExp.m`

   (c) Run the code and see how close you get for N_0 and Γ.