Program I
CS355/555
SP Term, 2023

Background

Random Hands in Bridge

Usually between games in the card game bridge the deck of 52 cards is shuffled three times, thinking that this will produce a randomly ordered deck, so that the hands dealt will be randomly ordered. Sets of four hands produced using exactly three shuffles usually have suits ordered with four cards of one suit and three cards of the other suits, so this result is not surprising.

Completely Random Hands

Several experiments using computer experiments to produce completely random hands have been conducted, and most of them do not produce the same results as exactly three shuffles. Most sets of hands produced by randomly generated ordering of the deck have different numbers of cards in each suit, so that 4-4-2-1 or 5-5-3-0 results are much more common. There are various explanations for why three shuffles may or may not produce very random hands.

Assignment

Analysis

Using a deck of cards numbered 1 - 52, shuffle the cards fifteen times using several different methods described below to produce a permutation of the original deck. Calling the original order i and the new deck y_i, calculate the following function for each of the fifteen runs.

$$r = \frac{n \sum_{i=1}^{n} i y_i - 1378^2}{2507960 - 1378^2}$$

This is a specific instance of a correlation calculation, which we will learn about in more detail later in the course. The full calculation is given by:

$$r = \frac{n \sum_{i=1}^{n} i y_i - (\sum_{i=1}^{n} i)\,(\sum_{i=1}^{n} y_i)}{\sqrt{(n \sum_{i=1}^{n} i^2 - (\sum_{i=1}^{n} i)^2)\,(n \sum_{i=1}^{n} y^2 - (\sum_{i=1}^{n} y)^2)}}$$

Note that $\sum_{i=1}^{n} i y_i$ changes from shuffle to shuffle, but all of the other pieces of the formula remain the same since i and y are the numbers 1 through 52, and adding the digits and the squares of the digits produces the same value.

$$\sum_{i=1}^{n} i = \sum_{i=1}^{n} y_i = 1378$$

This computation is the **correlation coefficient**, which we will see in more detail later in the course. The correlation coefficient is a measure of how related two sets of numbers are. If $r = 0$ or if r is close to 0 then the two variables are completely independent. If $r = 1$, then the i values and the y_i values are all equal, $i = y_i$. If $r = -1$ then the y values are backward, so that $i = y_{53-i}$

First Run

Given a deck of cards $cards[]$ with 52 entries, produce a new deck by dividing the deck into two pieces, $cards[1-26]$ and $cards[27-52]$, then pick $card[1]$, followed by $card[27]$, followed by $card[2]$, then $card[28]$, etc.

Do this 15 times, calculating the value of r for each shuffle, and answer the following questions:

1. Plot *r* with respect to the times of shuffling. After how many shuffles are the cards in the most random order? (That is, when is r at a minimum value?)
2. Do the cards return to their original order? After how many runs?
3. Is a total of 15 runs enough to return to the original order?
4. If not, does it appear that the cards will return to their original order in a relatively small number of shuffles?

Second Run

In the First Run, the first card will always be 1 and the last card will always be 52. Change the shuffling so that the first card in the next deck is $card[27]$, the second card is $card[1]$, etc., so that the last two cards are $card[52]$ followed by $card[26]$. Answer the same questions as in the First Run.

Third Run

Repeat the First Run with a deck of cards of 104 members.

Fourth Run

Repeat the Second Run with a deck of cards of 104 members.

Programming Advice

Programming Style for the Computation of r

As noted above, when writing the program for the first run you can substitute constants for most of the expressions. If you do this, you will have to recalculate by hand for the third run. You could define a variable with a name like sumi and set sumi=1378 then use sumi for $\sum_{i=1}^{n} i$ and for $\sum_{i=1}^{n} y_i$. Similarly, sumisq would be the sum of squares.

You can have the program calculate the sums each time they are required. There are some obvious tradeoffs in computation time, effort to change n from 52 to 104,

etc. Which is easier, computing the sums by hand or having the program compute them? Is the savings in time worth the time spent hand calculating?

Some Good Advice Close to the first line of the program, set a value $n = 52$. Then, right after that, calculate $sumi = \sum_{i=1}^{n} i$ and $sumsq = \sum_{i=1}^{n} i^2$ (using correct Python syntax, of course). Then set $sqsum = sumi * sumi$. This reduces the computation of r to

$$r = \frac{n \sum_{i=1}^{n} iy_i - sqsum}{n * sumsq - sqsum}$$

Deliverables:

a. A well written project report
   i. Briefly explain the functionality of your code.
   ii. Based on the outputs, answer questions for each of the four runs.
   iii. Put the screenshots of your outputs whenever necessary.
b. Submit the working version of your python code.
c. Independent Completion Form

Note: 1. Don't forget to include your names in the report.
2. If it is a group project, only one submission is needed. The independent completion form should have two names on it.