Exercise 1

Consider the set of all possible five-card poker hands dealt fairly from a standard deck of fifty-two cards. a. How many atomic events are there in the joint probability distribution (i.e., how many five-card hands are there)?

There are 52 cards to begin with and every card is distinct, so (52\*51\*50\*49\*48), but since we do not consider the order relevant we have to divide with 5! As that is the possible combination of each unic hand. So (52\*51\*50\*49\*48)/5! = 2598960

b. What is the probability of each atomic event?

Since each are equally likely, every hand has a probability of 1/2598960.

c. What is the probability of being dealt a royal straight flush? Four of a kind?

There are only 4 distinct card combinations that give a royal straight flush so the probability necessarily is 4/2598960 or 1/649740. “Four of a kind” has 13 types it can be in, with the fifth card being any of the remaining 48 cards, the probability is therefore (13\*48)/ 2598960 = 1/4165.

Exercise 2

Deciding to put probability theory to good use, we encounter a slot machine with three independent wheels, each producing one of the four symbols BAR, BELL, LEMON, or CHERRY with equal probability. The slot machine has the following payout scheme for a bet of 1 coin (where “?” denotes that we don’t care what comes up for that wheel):

BAR/BAR/BAR pays 20 coins

BELL/BELL/BELL pays 15 coins

LEMON/LEMON/LEMON pays 5 coins

CHERRY/CHERRY/CHERRY pays 3 coins

CHERRY/CHERRY/? pays 2 coins

CHERRY/?/? pays 1 coin

a. Compute the expected “payback” percentage of the machine. In other words, for each coin played, what is the expected coin return?

4\*4\*4 = 64 which is the total combinations possible, so the payback percentage will be the payout times the probability for each summed; 20\*1/64+15\*1/64+5\*1/64+3\*1/64+2\*3/64+1\*9/64=58/64 or 90.625%.

b. Compute the probability that playing the slot machine once will result in a win.

1/64+1/64+1/64+1/64+3/64+9/64=16/64=1/4 or 25% if we consider “CHERRY/?/? pays 1 coin” as a win.

If not then: 1/64+1/64+1/64+1/64+3/64 = 7/64 or 10.9%

c. Estimate the mean and median number of plays you can expect to make until you go broke, if you start with 10 coins. Run a simulation in Python to estimate this. Add your results to your PDF report.

import random

options = ["Bar", "Lemon", "Cherry", "Bell"]

liste = []

for i in range(1000000):

    coins = 10

    rounde = 0

    while(coins>0):

        rounde += 1

        coins -= 1

        wheel1 = options[random.randint(0, 3)]

        wheel2 = options[random.randint(0, 3)]

        wheel3 = options[random.randint(0, 3)]

        if(wheel1==wheel2==wheel3=="Bar"):

            win = 20

        elif(wheel1==wheel2==wheel3=="Bell"):

            win = 15

        elif(wheel1==wheel2==wheel3=="Lemon"):

            win = 5

        elif(wheel1==wheel2==wheel3=="Cherry"):

            win = 3

        elif(wheel1==wheel2=="Cherry"):

            win = 2

        elif(wheel1=="Cherry"):

            win = 1

        else:

            win = 0

        coins += win

        #print("Round", rounde, " |", wheel1, wheel2, wheel3, "| Wow this gives:", win, " in winnings. Total coins is now at:", coins)

    liste.append(rounde)

liste = sorted(liste)

print("liste-len:", len(liste), "max:", max(liste), "avg():", sum(liste)/len(liste), "median:", liste[int(len(liste)/2)] )

From a single run:

Text

Description automatically generated

From 100000 runs:



From 1 million runs:



Exercise 3

This exercise consists of two parts that ask you to run simulations to compute the answers instead of trying to compute exact answers. Add your answers to your PDF report.

Part 1

Peter is interested in knowing the possibility that at least two people from a group of N people have a birthday on the same day. Your task is to find out what N has to be for this event to occur with at least 50% chance. We will disregard the existence of leap years and assume there are 365 days in a year that are equally likely to be the birthday of a randomly selected person.

a. Create a function that takes N and computes the probability of the event via simulation.

import random

def happybday(n):

    liste = []

    for x in range(10000):

        liste2 = []

        for i in range(n):

            liste2.append(random.randint(1, 365))

        if(len(set(liste2)) != len(liste2)):

            liste.append(1)

        else:

            liste.append(0)

    return sum(liste)/len(liste)

b. Use the function created in the previous task to compute the probability of the event given N in the interval [10, 50]. In this interval, what is the proportion of N where the event happens with the least 50% chance? What is the smallest N where the probability of the event occurring is at least 50%?

count = 9

while 8<count<50 :

    count += 1

    k = happybday(count)

    print("count:", count, "prob:", k)

Using 10000 runs to generate the probability:

A screenshot of a computer

Description automatically generated with low confidence

23 is the lowest N which returns equal or over 50% probability.

Part 2

Peter wants to form a group where every day of the year is a birthday (i.e., for every day of the year, there must be at least one person from the group who has a birthday). He starts with an empty group, and then proceeds with the following loop:

1. Add a random person to the group.

2. Check whether all days of the year are covered.

3. Go back to step 1 if not all days of the year have at least one birthday person from the group.

a. How large a group should Peter expect to form? Make the same assumption about leap years as in Part 1.

import random

import numpy as np

liste2 = []

for i in range(1000):

    liste = []

    while True:

        liste.append(random.randint(1, 365))

        if(len(set(liste))==365):

            break

    liste2.append(len(liste))

liste2 = sorted(liste2)

print("avg:", np.sum(liste2)/len(liste2), "median:", liste2[int(len(liste2)/2)])

For 1000 runs per:



For 10000 runs per:

