Probability of Winning the Lottery

March 6, 2024
Zander Bonnet

Video Link:

https://vimeo.com/920182646/f9b4d8fb0e?share=copy

Consider a state lottery and the probability of winning the grand prize. The game is played by selecting a group of 6 numbers from {1, 2, 3, ..., 51}, and the state selects a group of 6 numbers from the same set. You win the grand prize if all 6 of your numbers match the state's. The probability of winning is so low that some people are trying different ways of increasing their odds.

In [1]:
```python
import math
```

1) Identify whether the probability of winning is discrete or continuous. Write a Python function that calculates the probability of winning the grand prize for the new game, given that a group of 6 numbers are selected from {1, 2, 3, ..., 51}.

The probability of winning the lottery is discrete becasue there are a finite sets of outcomes. In our case there are only 51 possible numbers that can be selected.

In our lottery there are 51 possible numbers that can be selected and each number can only be selected once. In this case we can use the combinations formula to calculate the total possible combinations ingnoring order and repeats.

In [2]:
```python
#Takes 2 int values
#nums = The number of ints to be drawn
#maxnum = The maximum number of numbers to be drawn from
#Calculates the probability of winning with those params
def problotwin(nums,maxnum):
    if not (isinstance(nums, int) & isinstance(maxnum, int)):
        raise TypeError('Input is not an int')
    if nums > maxnum:
        raise TypeError('nums cannot be larger than maxnum')

    return(1/math.comb(maxnum, nums))
```

In [3]:
```python
print('Probability to win with one ticket: {:.10f}%'.format(problotwin(6,51)*100
```

```
Probability to win with one ticket: 0.0000055526%
```

2) Alexandra buys 1 ticket every 24 hours of every 365 days per year for 80 years of her life. Each ticket costs $5 and she selects her numbers independently from one ticket to the next.

Write a Python program that calculates the probability of Alexandra winning the lottery at least one time and the amount of money she spends trying.

```
In [4]:
#Takes 3 int values
#drawings = the number of drawings you enter
#howManyDraw = The number of ints to be drawn
#maxNum = The maximum number of numbers to be drawn from
#Calculates the probability of winning at least once with those params
def winatleastonce(drawings, howManyDraw, maxNum):
    if not (isinstance(howManyDraw, int) & isinstance(maxNum, int) & isinstance(
        raise TypeError('Input is not an int')
    if howManyDraw > maxNum:
        raise TypeError('howManyDraw cannot be larger than maxnum')
    return 1 - math.pow(1 - problotwin(howManyDraw, maxNum),drawings)
```

Assume the drawing is drawn daily.

```
In [5]:
drawings = 365*80
cost = 5 * drawings

probwin = (winatleastonce(drawings, 6, 51))*100
print('Probability of winning at least once: {:.10f}%'.format(probwin))
print('Total Cost: ${:}'.format(cost))
```

```
Probability of winning at least once: 0.1620056437%
Total Cost: $146000
```

Assume the drawings is weekly

```
In [6]:
drawings = (365*80)//7
cost = 5 * 365 * 80

probwin = (winatleastonce(drawings, 6, 51))*100
print('Probability of winning at least once: {:.10f}%'.format(probwin))
print('Total Cost: ${:}'.format(cost))
```

```
Probability of winning at least once: 0.0231573693%
Total Cost: $146000
```

3) Amir decides to increase his chance of winning by organizing N of his friends to buy one ticket each for this week's drawing. Each friend buys a ticket independently, and the minimum value of N (Nmin(epsilon)) to ensure that the probability that there is at least one winner among the group is greater than or equal to epsilon needs to be calculated for various values of epsilon. Write a Python function that calculates the value of Nmin(epsilon) for epsilon = 1e-5, 1e-3, 0.1, and 0.5.

```
In [7]:
#Method 1 - Dr. Ofori
eps = [.00001, .001, 0.1, 0.5]
probLose = 1- problotwin(6,51)
nmins = []
for e in eps:
    n_min = math.ceil(math.log(1 - e) / math.log(probLose))  # Calculate N_min
    nmins.append(n_min)
print(eps)
print(nmins)
```

```
probs = [winatleastonce(i, 6, 51) for i in nmins] # Calculates the exact prob of
print(probs)
```

```
[1e-05, 0.001, 0.1, 0.5]
[181, 18019, 1897486, 12483207]
[1.0050223358426535e-05, 0.001000029386858281, 0.10000000306185353, 0.5000000258
236499]
```

In [8]:

```
#Method 2
eps = [.00001, .001, 0.1, 0.5]
nmins = []
for e in eps:
    mins = 0
    while (winatleastonce(mins, 6, 51)) < e: #Calculates the threshold of N-min
        mins += 1
    nmins.append(mins)
print(eps)
print(nmins)
probs = [winatleastonce(i, 6, 51) for i in nmins] # Calculates the exact prob of
print(probs)
```

```
[1e-05, 0.001, 0.1, 0.5]
[181, 18019, 1897486, 12483207]
[1.0050223358426535e-05, 0.001000029386858281, 0.10000000306185353, 0.5000000258
236499]
```

4) If Amir has gathered enough friends to win with probability 0.5 using the above strategy, write a Python program that determines a way that they can increase their win probability above 0.5 without adding more people to the group.

We can easily imporve the odds of winning the lottery if we make sure that there are no repeates. If we make sure that everyone has a unique set of numbers it will greatly increase their odds of winning.

In [9]:

```
print(nmins)
new = [(problotwin(6,51) * i) for i in nmins] #Calculates the prob if all entrie
print(new)
dif = [(new[i] - probs[i]) for i in range(len(new))] #Calculates the change in p
print(dif)
```

```
[181, 18019, 1897486, 12483207]
[1.0050273578441553e-05, 0.0010005297216018692, 0.10536051608432458, 0.693147212
6315836]
[5.022001501753129e-11, 5.003347435881666e-07, 0.00536051302247105, 0.1931471868
0793375]
```

In [10]:

```
print('Amir increased their odds to {:.2f}%'.format(new[3]*100))
```

```
Amir increased their odds to 69.31%
```

5) The state lottery commissioner introduces a consolation prize: If exactly 3 of your 6 numbers match any 3 of the state's, you win a smaller prize. Write a Python program that calculates the probability of winning the consolation prize and the expected value of the consolation prize. Is this consolation prize likely to increase sales?

If there can only be EXACTLY 3 numbers correct

In [11]:
```
#Takes 3 int values
#inCommon = the number of values that will match the winning ticket
#nums = The number of ints to be drawn
#maxNums = The maximum number of numbers to be drawn from
#Calculates the probability of winning with exactly the inCommon number
def probtogetx(inCommon, nums, maxNums):
    if not (isinstance(nums, int) & isinstance(maxNums, int) & isinstance(inComm
        raise TypeError('Input is not an int')
    if nums > maxNums:
        raise TypeError('nums cannot be larger than maxNums')
    if inCommon > nums:
        raise TypeError('inCommon cannot be greater than nums')

    return ((math.comb(nums,inCommon) #Ways to draw inCommon correct numbers
             * math.comb(maxNums - nums, nums-inCommon)) #ways to draw the remai
            /math.comb(maxNums,nums)) #Total number of ways to draw nums from ma
```

In [12]:
```
print('Probability to win the consolation prize: {:.10f}%'.format(probtogetx(3,
```

```
Probability to win the consolation prize: 1.5758384760%
```

In [13]:
```
PRIZE = 100_000_000

preex = (problotwin(6,51) * PRIZE) #ex of just main prize

postex = (problotwin(6,51) * PRIZE) + (probtogetx(3, 6, 51) * (PRIZE * .01)) #ex

preex, postex
```

Out[13]:
```
(5.552637336155554, 15763.937397345617)
```

In [14]:
```
print('The expected value of the original drawing is {:.3f}'.format(preex))
print('The expected value of the new drawing is {:.3f}'.format(postex))
```

```
The expected value of the original drawing is 5.553
The expected value of the new drawing is 15763.937
```

We would expect the addition of the consolation prize to increase sales. There is a large increase in expected return with the addition of the consolation prize. It will also feel as if they have a larger chance of winning something.

If there can be 3 or more numbers correct excluding getting all 6

In [15]:
```
#Takes 3 int values
#inCommon = the minimum number of values that will match the winning ticket
#nums = The number of ints to be drawn
#maxNums = The maximum number of numbers to be drawn from
#Calculates the probability of winning at least the inCommon number
def atleastxsame(inCommon, nums, maxNums):
    if not (isinstance(nums, int) & isinstance(maxNums, int) & isinstance(inComm
        raise TypeError('Input is not an int')
    if nums > maxNums:
        raise TypeError('nums cannot be larger than maxNums')
    if inCommon > nums:
        raise TypeError('inCommon cannot be greater than nums')
    prob = 0
    for x in range(inCommon, nums):
        prob += probtogetx(x, nums, maxNums)
    return prob
```

In [16]:

```
print('Probability to win the consolation prize: {:.10f}%'.format(atleastxsame(3
```

```
Probability to win the consolation prize: 1.6597943525%
```

In [17]:

```
PRIZE = 100_000_000

preex = (problotwin(6,51) * PRIZE)

postex = (problotwin(6,51) * PRIZE) + (atleastxsame(3, 6, 51) * (PRIZE*.01))

preex, postex
```

Out[17]:

```
(5.552637336155554, 16603.496162572337)
```

In [18]:

```
print('The expected value of the original drawing is {:.3f}'.format(preex))
print('The expected value of the new drawing is {:.3f}'.format(postex))
```

```
The expected value of the original drawing is 5.553
The expected value of the new drawing is 16603.496
```

We would expect the addition of the consolation prize to increase sales. There is a large increase in expected return with the addition of the consolation prize. It will also feel as if they have a larger chance of winning something.

References:

Ofori, E. (2024). Just Give DSC-510 a "Chance " Probability Week 3 Topics [Anouncement]. Halo. https://halo.gcu.edu/courses/DSC-510-O500-20240215/announcements

Rogel-Salazar, J. (2023). Statistics and Data Visualization with Python. CRC Press.