Project 1: Which Urn is Which ? A Preliminary

Craig Brooks

Due February 6 2023

1 Introduction

There are two urns, and inside this urn are 100,000 marbles comprised of 3 colors: White, Green, and Black. In one urn, the group ratio for $White : Green : Black$ is $25 : 35 : 40$ and the second is $37 : 25 : 38$. Our task: To determine which urn contains which ratio by doing small random samples from each population by observing the median number of draws it will take to draw a green marble.

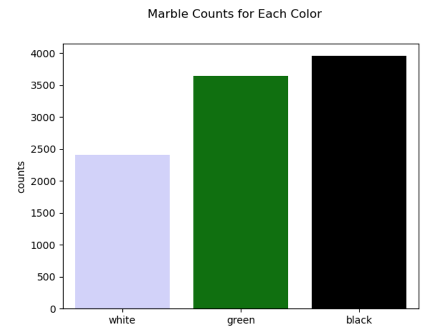
2 What We Know

If we simply want to find the distribution given an arbitrary amount of draws, our random variable will be $(X = number\ of\ successes\ drawn\ in\ n\ trials) \sim Binomial(X)$ in the case of 2 marble species if they are being drawn with replacement. In our case, we have 3 marble species. Thus, we can characterize our sampling with a multinomial distribution. In these situations, all draws are independent of each other, so the probability of success for each draw *does not change*.

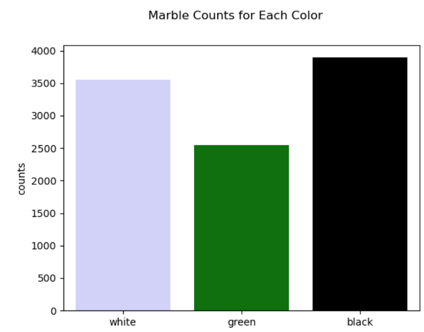
We can also determine the *number* of times we can expect to draw, on average, until we successfully draw a green marble, and this can be modeled with a geometric distribution. Unlike the multinomial distribution, where we are looking at the distribution of the successes, we are counting the number of *trials* until a successful outcome.

3 Outline of the Experiment

In our experiment, we will construct our sets of marbles with a multinomial distribution. Figure 1 shows a barplot representing the two urns with the counts of marbles of each color. From the generated dataset, we will then do a large number of trials, and pick (with replacement) until we achieve a positive result for each trial, in this case picking a green marble. Next, we can calculate the average and/or median number of draws it takes until drawing a green marble, then plot a histogram of the counts over n trials. From there, we will plot box-and-whisker plots for the trials, and therefore determine the interquartile



(a) 25/35/40 group ratio



(b) 38/25/37 group ratio

Figure 1: Two graphs representing the distribution of marbles in two urns

range + 1.5*median of n trials. Finally, we can calculate the log-likelihoods of drawing a green in x tries given a particular urn in n trials to a 95 % confidence interval.

4 Code

To do our experiment, we must come up with a way to draw our marbles. We must first generate the two "urns" filled with the marbles of the different ratios. For this, we use a function called `Category`, which is not listed here. However, once generated, we can begin taking samples from each urn. This snippet takes integer data generated in the `Category` function (not listed here), draws `Ntrial` random samples, and converts it into the strings `'White'`, `'Green'`, and `'Black'`. Thus, we generate our random samples to analyze directly, write to a file, or import into a DataFrame.

```
NTrial = 20
color_lists = []

for _ in range(20):
    samples = random.choices(outcomes, k=NTrial)
    color=[]
    for item in samples:
        if item == '1':
            color.append('White')
        elif item == '2':
            color.append('Green')
        else:
            color.append('Black')
    color_lists.append(color)
```

We will be reusing much of what has been used in previous assignments. However, there will be code to generate the data for the box-and-whisker plot. Below is a snippet that will produce a boxplot from data in a file. The file will contain rows of strings where each row contains the outcomes of the draws.

```
#This is context manager for all our file operations and plotting
with open('colors.csv', 'r') as my_file:
    x = my_file.readlines()
    # Converts the strings in the datafile to arrays that pandas can understand
    def Convert(string):
        li = list(string.split(" "))
        return li

    for i in range(len(x)):
        array.append(Convert(x[i].strip()))

    # Reads the file into a DataFrame
    f = pd.DataFrame(array)
    f.index = f.index + 1
    f.rename(columns={x:y for x,y in zip(f.columns,range(1,len(f.columns) + 1))})
```

```
# This segment renames the columns in the DataFrame
for key in f.keys():
    key = str(key)

# This effectively reshapes the DataFrame to make it more amenable to using the
# sns.boxplot(...) function
f_melted = pd.melt(f)
f_melted = f_melted.rename(columns={'variable': 'mean count', 'value': 'trials'})

fig3, ax3 = plt.subplots()

sns.boxplot(data=f_melted, x='mean count')
ax3.set_title('Median Count Until Green Marble for N Trials)
```