Nathan Burwig
Math 87 HW 6
Due 11/02/2022

---

1 You can't handle the truth!

Let's look at the impact of a juror's likelihood to vote correctly and the chance of a wrongful conviction. A jury consists of 12 members, and each member must vote either guilty or not guilty. Assume that the vote of each member is independent Suppose that the probability that a juror votes correctly (i.e. in line with the actual innocence/guilt of the person on trial) is 70%. Disclaimer: these numbers are made up, though there are many studies done on this topic.

1. **What is the probability that there is an inconclusive (tied) voting outcome**

   For much of this problem, we will be using the binomial distribution. We take the probability of some event then to be...

   $$P(X = m) = \binom{n}{m} p^m (1-p)^{n-m}$$

   In the case of a tied voting outcome, we know that $m = 6$ and $n = 6$ with a p-value of 0.7. This gives the following outcome...

   $$P(X = 6) = \binom{12}{6} (0.7)^6 (0.3)^6 = 0.079$$

2. **If there is an inconclusive verdict, then a judge breaks the tie. Suppose a judge votes correctly 80% of the time. What is the probability of a tie and a judge voting correctly?**

   The probability of this happening is a compound probability. We know that there is a 0.079 probability of the first event occuring, so that, combined with the following probability associated with the judge voting correctly...

   $$P(X = 6) \times 0.8 = 0.079 \times 0.8 = 0.063$$

   I don't entirely know if I'm doing this correctly (I've always been bad at probability) but if this is correct, these numbers are scary.

3. **A verdict is reached if at least 7 jury members vote the same way, or a tie is broken in the manner given above. What is the probability that the correct verdict is reached?**

   So we know we can reach the correct verdict a few different ways. Either a) it is reached after a tie, followed by the correct verdict by the judge, or b) anywhere between 7-12 jurors vote correctly. This can be represented the following way.

   $$P(X = 6) * 0.8 + \sum_{m=7}^{12} \binom{12}{m} (0.7)^m (0.3)^{12-m}$$

   Which, when evaluated results in the value 0.9455 or 94.55%

4. **Of the times where the jury reaches the incorrect verdict, suppose 90% of people on trial are guilty but not convicted, and 10% are innocent but convicted. What is the probability of a wrongful conviction?**

   It is given that an innocent person is convicted only 10% of the time. We know that the probability of an incorrect verdict is $1 - 0.9445$ or 0.0545, so that means the probability of an innocent conviction is $0.0545 * 0.10 = 0.00545$ or 0.545%.

5. **What is the expected number of people wrongfully convicted in 2 million trials?**

   We get this number simply by multiplying 2 million by the number percentage chance of a wrongful conviction which we found in the previous part of this problem. We get $2,000,000 * 0.00545 = 10,900$ so approximatly 11000 people will be wrongly convicted out of 2 million.

6. **Write a `python` function that takes in the likelihood of a juror to vote correctly and returns the probability of a wrongful conviction. You may use `scipy.stats.binom` or compute your own binomial distribution. Include your code in your report.**

   Below is the code I used to calculate this for the given values.

```python
        from scipy.stats import binom

        ## a function to determine wrongful conviction rates
        def wrongful_conviction(p):
            ## initial variables
            prob = 0
            a = range(7, 13)

            ## loop through to get probabilities associated with problem
            for i in a:
              prob = prob + binom.pmf(12-i, 12, p)

            ## the probability there is a tie times probability judge
            ## gives incorrect verdict
            tie = binom.pmf(6, 12, p) *.2

            ## sum the two associated probabilites
            prob = (tie + prob) * .1

            return(prob)

```

Listing 1: The Program

7. **Test your function on likelihoods 50%, 60%, 70%, 80%, and 90%. Report both the probability of a wrongful conviction, and the expected number of people wrongfully convicted in 2 million trials. Discuss your results.**

   When I run my code, I get the following output.

P-Correct Verdict	P-Wrongful Conv.	# Wrongful Conv.
0.5	0.0432	86464
0.6	0.0194	38705
0.7	0.0054	10890
0.8	0.0007	1400
0.9	1.484e-5	29

8. **What are some flaws in the assumptions of the problem?**

   The most blatant flaws to me are the assumptions about the probability of a correct verdict. The ability to judge this depends on the judge, the trial, the defense, the prosecution, all sorts of variables that are very difficult to keep track of. Trying to reduce the complexity of a court room to simple probabilities is bound to be innacurate.

I also feel like it is very difficult to actually come up with accurate probabilities for wrongfully convicted persons. There isn't really a way to know how many people have been wrongfully convicted, so tryign to ascribe probabilities to these types of things is also bound to be innacurate.

2 Geometry feat. Monte Carlo

Let C be the circle defined by $(x - 1.25)^2 + (y - 1.25)^2 = 0.4$. We know it has interior area of $.4\pi$. Assume we don't know this, and use Monte Carlo Rejection Sampling to determine the area of this circle.

1. **Run this experiment with sample count (1, 5, 10, 50, 100, 1000, 5000, 10000, 50000, 100000, 500000, 1000000). Compute the error (relative the true area) both as a value and as a percentage**

   In order to run this experiment, there are a couple quantities that we would like to know. Namely, we want to know what equation we can use to find the area of the circle. We can do this by taking the number of points that land within the circle, and comparing it to the total number of points within the square. This ratio will give us the percentage area taken up in the square, so if we multiply by the area of the given square, we know the area of the circle within the region.

   We also want to know the actual area (given) and the radius ($\sqrt{.4}$) along with how to calculate error. Our equations of interest, then, are the following...

$$\text{Area} = \text{Area of Square} \cdot \frac{\text{\# of points within circle}}{\text{\# of points in total}}$$
$$\text{Error} = \frac{\text{calculated - expected}}{\text{expected}}$$

   With all of this in mind, we can begin working on the actual problem. We simulate points in our system according to the layour I've given above, and "reject" any points that are outside the area given by the formula for our circle. The region we are simulating in is the square from $0 \leq x \leq 2$ and $0 \leq y \leq 2$

   I'll give the code I used to do this in case the above explanation was unclear.

```python
import random, math

## iter = number of iterations (integer only)
## R    = radius of circle (integer or otherwise)
## ran  = range of region to check (ie [0, 2] -- only 2D range accepted)
def circle(iter, R, ran):
    total, in_circle = 0, 0

    for i in range(iter):
        ## generate random points uniformly over the given interval 'ran'
        x = random.uniform(ran[0], ran[1])
        y = random.uniform(ran[0], ran[1])

        ## sample rejection: remove points not within given area
        if (x - 1.25) ** 2 + (y - 1.25) ** 2 <= R ** 2:
            in_circle += 1
        total += 1

    ## calculate the area via ratio of area to volume and expected area
    # area is area of square times ratio of dots in circle vs out
    area = (ran[1] ** 2) * in_circle / total

    # expected area is just area of the circle of radius R
    expected = math.pi * R ** 2

    ## check for errors (on small samples no point lands in
    ## region so area = 0 causing divide by 0)
    if(area != 0):
        error = abs((area-expected)/expected)
    else:
        error = "err, div by 0"

    ## print relevant info
    print("Given", iter, R, ran)
    print("  expected:\t", expected)
    print("  calculated:\t", area)
    print("  error:\t", error)
```

Listing 2: Monte Carlo Rejection Sampling for Area of Circle

Using the above code we can generate as many points in the region we want. When we do this, we get the following for the given samples.

Samples	Calculated	Expected	Error
1	4.0	1.2566	2.18309
5	1.6	1.2566	0.27323
10	1.2	1.2566	0.04507
50	1.68	1.2566	0.33690
100	1.08	1.2566	0.14056
1000	1.316	1.2566	0.04723
5000	1.2568	1.2566	0.00012
10000	1.2492	1.2566	0.00591
50000	1.25256	1.2566	0.00324
100000	1.26056	1.2566	0.00312
500000	1.258504	1.2566	0.00148
1000000	1.2518	1.2566	0.00384

So we can see that we get fairly close as we get larger and larger in our sample count. I find it interesting that we only get *so* close. The last several steps only seemed to increase the accuracy by a little, and even then, it seemed largely random.

Also, it is worth noting the `err`. When you have such a small number of samples, it becomes increasingly likely that the one or five points you cast onto your system will actually land within the circle, so your area becomes zero which causes a divide by zero error. It took several attempts of running this before it gave a non-error result for 5 and 10 samples.

2. **Let C' be the cirlce defined by $(x-.75)^2+(y-.75)^2=.3$. Find the area of the union of these two points using a modified version of the above algorithm**

   We know that we want to find the area contained between the two circles. The first issue here is that I don't know how to calculate that normally, so I don't actually have anything to compare against. Looking up some code online I was able to find (`https://scipython.com/book/chapter-8-scipy/problems/p84/overlapping-circles/`) as well as some help from wolfram, I was able to find the *intersection* area of the two circles, and I know that the combined area will just be the addition of the two individual areas minus the intersection area)

   We will use an almost identical version of the code I have shown above, apart from two lines. Our rejection line (points not within the circle) now specifies whether is within either of the two circles. Our `expected` is now the sum of the two areas minus the intersection as calculated by wolfram and the code given in the provided link.

   With this, we get the following...

Samples	Calculated	Expected	Error
1000000	1.8867396	1.888137	0.00039

   We know the sum of the two areas to be 2.19911, so having a value of ≈ 1.88 makes sense as it is less than the total area, and is within a very small error of the actual value.