Probability Code by Travis McVoy

This document contains all the code I've written for simulations, approximations, or what have you for various probabilistic systems and/or ideas. At the moment, the code is organized (just) into code used to supplement problem sets, some before the code made to supplement notes. In general, though, most of these simulations are assisted by resources in Sections or the problem sets; that is, I made them for fun simply to playing around with, only. Since I'm still learning, it very well may be the case that some of these calculations aren't actually representative of what's happening. If you see anything, feel free to email me (travismcvoy@berkeley.edu).

Code for Part 4

In the Part 4 problem set, there seem to have been a few problems that allowed for for simulations. These simulations are something I came up with on my own, and are therefore obviously not required. I originally wrote my code in Python, but I've been using Julia a lot lately, so I decided to transfer everything to one nice, self contained document.

Problem 23.

We are told that we are in a DMC, and that a certain commodity presently sells for \$2 at a source. Suppose that after one week, the commodity will sell for either \$1 or \$4 on source, with these two possibilities being equally likely. We are then asked to maximize the expected money that we have after the second week. We found that the optimal policy is to buy 1 if we have \$2 or \$4 at the source, and to do nothing if we have \$1. I then attempted the process of doing this repeatedly, since each 2 week period is independent of one another, and the total profits add up. Here, I

NOTE: In my code, I also generated a delayed drawing our profits after 50 iterations for 100 samples. However, that also slowed my code, and there's no reason to actually do that unless you want to see what happens in the long run. Also, I'm not sure if this is actually a good way to do this, but it seemed like it made sense. I'm sure there's a better way to do this, but I wanted to make sure that I checked the math myself, so there wasn't much. Consequently, I didn't understand a lot of what we did (which is why I'm learning probably now).

```
using Plots
function buyone(money, one) # buy
    money = money - 2
    one = one + 1
    return money, one
end

function sell(money, one) # sell
    money = money + 4
    one = one - 1
    return money, one
end

function simulate(n, money, one)
    for i in 1:n
        if money == 2
            money, one = buyone(money, one)
        elseif one == 1
            money, one = sell(money, one)
        end
    end
    return money
end

simulations, vec1, vec2 = 100, [], []
for i in 1:simulations
    money = 2
    one = 0
    money = simulate(50, money, one)
    push!(vec1, money)
    money = 2
    one = 0
    money = simulate(100, money, one)
    push!(vec2, money)
end
histogram(vec1, xlabel="Money", ylabel="Frequency", title="Buy 1, Sell When If", legend=false)
savefig("1")
histogram(vec2, xlabel="Money", ylabel="Frequency", title="Buy 1, Sell When If", legend=false)
savefig("2")
```

Problem 35.

We are told that a box contains 5 red and 5 blue marbles. Two marbles are withdrawn randomly. If they are the same color, then we win \$5; if they are different colors, we pay \$1. We are asked to calculate the expected winnings, which we found to be -0.5. This is another problem in which simulations of independent iterations

Consider the drawing of marbles as a single iteration in a game. After one draw, we replace the marbles and draw again during the next iteration. Repeating this process, we have a game in which we can mean winnings over time. We could hypothetically consider simulations to get the probabilities of winning, losing, or

$$P(X = x) = \frac{\binom{5}{x}\binom{5}{5-x}}{\binom{10}{5}}$$

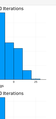
So the probability that we select 2 red marbles, with $x = 0, 1, 2$ (clearly, we can't do $x = 3$ or $x = 4$ ... $x = 5$). We can represent this in code by selecting a random number between 0 and 1, and then using the above formula to see if we win \$5 or lose \$1. Doing this for 100 iterations, we get a final random number in the range -10, we consider that to represent losing no money, so we lose \$10 (assuming 0 loss) and anything else we lose.

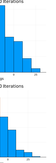
We can now plot our winnings in time using a simulation. Note I am assuming we start with no money. Maybe we spent it all at the game. After plotting our winnings vs time, we can extrapolate the average winnings and that winnings over many samples.

```
function simulate_one_time(time) # time
    money = 0
    for i in 1:time
        # get last iteration winnings because
        # we initial time step is zero
        # and we want to add winnings
        # the first iteration of game
        if i == 1
            last = simulate(0)
            money = money + last # stored in index zero
        else
            # do the same for 1 step machine you were
            # playing last time
            last = simulate(last) # x_i + x_{i-1}
            money = money + last
        end
        if mod(i, 10) == 0 # every 10th step
            # print out the current money
            # for the last 10 steps
            println("At step $i, I have $money")
        end
        # save current i and that step money
        # for next iteration
        last = money
        money = simulate(last) # x_i
    end
    return money
end

function simulate_many(iterations, time)
    final = []
    for i in 1:iterations # for each iteration
        # print out which iteration we are on
        # for sanity check
        print("Iteration $i\n")
        money = simulate_one_time(time)
        push!(final, money)
        print("Final Winnings: $money\n")
        print("Average Winnings: $(mean(final))\n")
    end
    return final
end

final = simulate_many(500, 300)
histogram(final, xlabel="Final Winnings", ylabel="Occurrences", title="500 samples of 300 Iterations", legend=false)
savefig("3")
```



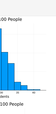


Since our expected winnings is -0.5 and we did 300 iterations, we expect to end at around -150. As you can see, sometimes that happens, sometimes it doesn't. To get a better sense of what's happening, we can use a bunch of samples and generate a histogram of the average

```
# Simple function to get final winnings
function simulate_one_time(time) # time
    money = 0
    for i in 1:time
        # get last iteration winnings because
        # we initial time step is zero
        # and we want to add winnings
        # the first iteration of game
        if i == 1
            last = simulate(0)
            money = money + last # stored in index zero
        else
            # do the same for 1 step machine you were
            # playing last time
            last = simulate(last) # x_i + x_{i-1}
            money = money + last
        end
        # save current i and that step money
        # for next iteration
        last = money
        money = simulate(last) # x_i
    end
    return last/iterations
end

function last_final_calculate(n, time)
    final = [] # store final winnings
    for i in 1:n # for each number of iterations
        for j in 1:samples
            simulate = simulate(time)
            push!(final, simulate)
        end
        average = mean(final)
        push!(last_final, average)
        empty!(final)
        print("Average: $average\n")
    end
    return last_final
end

last_final_samples, time = 500, 300
histogram(last_final_samples, xlabel="Average Winnings", ylabel="Occurrences", title="500 samples of 300 Iterations", legend=false)
savefig("4")
```



Note that we've shown now average winnings to be -0.5 (-150 in the last iteration). However, the above distributions match our expectations. One thing that I don't understand is why the distribution of final winnings seems to be centered around -150 (the expected value), but the distribution of average winnings seems to be centered around -150. However, I guess that the calculated standard deviation matches the values in the back of the book, so we come numbers, and the numbers matched my graphs. Do you guys know why the distribution of final winnings centers around the expected value? I checked the math myself, so there wasn't much. Consequently, I didn't understand a lot of what we did (which is why I'm learning probably now).

Bayes's Theorem

There were several examples in the section on Bayes's Theorem that I loved and wanted to explore via simulation. In the code that follows, we'll do just that.

Insurance Company

We are given the following two part problem statement.

Part 1

An insurance company believes that people can be divided into two classes: those who are accident prone and those who are not. The company's statistics show that an accident prone person will have an accident at some time within a fixed 1-year period with probability 0.4, whereas this probability decreases to 0.2 for a person who is not accident prone. If a randomly chosen new policyholder has had an accident within a fixed 1-year period, what is the probability that a new policyholder will have an accident within a year of purchasing a policy?

Part 2

Suppose that a new policyholder has an accident within a year of purchasing a policy. What is the probability that he or she is accident prone?

We found in part 1 that the desired probability is 0.28. I have it to the reader to verify. The part that I'm interested in is the second part of the question. Let's go ahead and re-state it, then we can use simulation to see the results in action.

To solve this problem, we first introduce some notation: Let A_1 be the event that a person is accident prone and let B be the event that a person had an accident within a year of purchasing a policy. We want to find

$$P(A_1|B)$$

To do so, we can use Bayes's rule. Namely, we know that A_1 and A_2^c partition the sample space so $P(A_1|B)$ is given by:

$$P(A_1|B) = \frac{P(B|A_1)P(A_1)}{P(B|A_1)P(A_1) + P(B|A_2^c)P(A_2^c)} = \frac{(0.4)(0.4)}{(0.4)(0.4) + (0.2)(0.6)} = \frac{0.16}{0.16 + 0.12} = \frac{0.16}{0.28} = 0.4\overline{571}$$

Let's do a simulation to see this result, and then we'll use Bayes's rule to see if we can do it quicker. Note that the insurance company should use this policy (Bayes's) in order to gain a profit!

Note: I actually simulated the probability that someone has an accident, not the conditional probability that someone is accident prone given that they had an accident. I'm not actually sure how to simulate conditional probability, I found a line that might be helpful, but truthfully, I should probably work on my homework for my classes. Anyway, I'm leaving it here for reader. https://stackoverflow.com/questions/4148960/probability-simulation-in-excel/4150417

```
# Simulate one person
# Assume person is an accident prone
# Simulate 1 year
function simulate(prob, n)
    accident_count = 0
    for i in 1:n
        prone = rand(0:1)
        if prone == 1 # accident prone
            # if it people are accident prone an
            # accident happens
            # 40% chance of an accident
            # 60% chance of no accident
            # 1 represents an accident
            # 0 represents no accident
            # simulating 100 and accident prone
            # 40% chance of an accident
            # 60% chance of no accident
            # 1 represents an accident
            # 0 represents no accident
            accident = rand(0:1)
            if accident == 1
                accident_count = accident_count + 1
            end
        end
    end
    return accident_count
end

# Run a number of samples
function run_samples(samples, n, prob) # for, map, filter
    last_accident_count = []
    for i in 1:samples
        accident_count = simulate(prob, n)
        push!(last_accident_count, accident_count)
        print("Accidents: $accident_count\n")
    end
    return last_accident_count
end

last_accident_count_samples = run_samples(100, 100, 0.4)
histogram(last_accident_count_samples, xlabel="Number of Accidents", ylabel="Occurrences", title="1000 Samples of 100 People", legend=false)
savefig("5")
```

What does the above left tail mean? In each sample we had 100 people, so we expect that of that sample, roughly 30 of them had an accident. Indeed, that's exactly what happened. We see that of the six samples, we have the number of accidents, and we plotted a histogram to see how it behaves for a normal distribution.

One interesting point is the percentage of samples in which 20% of policy holders had an accident in the first year. In particular, in our 10,000 and 100,000 sample experiments, roughly 4% of the sample experiments had 20% of individuals had an accident. Interestingly enough, in our 1,000 sample experiment that percentage was 0%. Although we can't say for certain why this was the case that the data isn't bin-valued (unlike, as the left tail majority of the data has between 25% and 35% of new policy holders having an accident.

Given that most of the samples, between 25% and 35% of policy holders had an accident in the first year, how much should the insurance company charge? Well, we would need some data on the distribution of cost of accidents, but suppose that on average, an accident costs a dollars. Then, suppose that the company has, on average, n policy holders, then the company should charge at least $0.3an$ dollars per policy holder. Why? Well, if we take the left tail, we see that 30% is also overwhelming that likely. Thus, to be on extremely safe mode, we would argue the company should charge a little bit more than $0.3an$, say, where $c > 0.3$.

At least, that's our first recommendation. Notice that our simulation corresponds to the first part of the last paragraph. Namely, we have shown that on average, 30% of new holders will have an accident. However, we also showed that, sometimes, almost 50% of them are accident prone. Meaning, of those 30%, roughly 70% are accident prone, so of the 30% that have an accident, roughly 5% will have an accident again the next year. That is, of new policy holders 5% of them will have an accident two years in a row. That's a lot of money! Suppose I'm not allowed now to include that. Only I think we could have going, but we might want to see if Markov chains would make the math easier.

Edit

Why does the above new line a binomial distribution? It's not easier enough to answer that right now, but it occurred to me that that should be considered. It might be the case that it's not as a coin among. It doesn't, that's a guess.

Blood Tests

Another example from Bayes's Theorem section that I really like which involves blood tests. In the example, we are given the following problem statement:

> Suppose a certain test is 95 percent effective in detecting a certain disease when it is in fact present. However, the test also yields a "false positive" result for 1 percent of the healthy persons tested. (That is, if a healthy person is tested, then, with probability .01, the test result will imply that he or she has the disease.) If 0 percent of the population actually has the disease, what is the probability that a person has the disease given that the test result is positive?

The solution to the problem is as follows:

We let D be the event that a person actually has the disease and p be the event that the test result is positive. The desired probability is then:

$$P(D|p) = \frac{P(D \cap p)}{P(p)} \tag{1}$$

$$= \frac{P(D)P(p|D)}{P(D)P(p|D) + P(D^c)P(p|D^c)} \tag{2}$$

$$= \frac{(0.005)(0.95)}{(0.005)(0.95) + (0.995)(0.01)} \tag{3}$$

$$= 0.324 \tag{4}$$

This is rather surprising. Even though the test is 95% effective and only yields a false positive 1% of the time, 0.5% of the population has the disease, then roughly 70% of the positive test results are false positives. Let's simulate this system to see it in action.

Since the finance sector sets us free roughly 30% of the positive tests results are true positives, we'll keep track of the number of positive tests, the number of true and false positives, the ratio of true positives to false positives (which is what estimates $P(D|p)$), and the number of infected people.

I have chosen to make each sample consists 10,000 people. The reason for this is that most of our probabilities are really smaller, so having a larger number will mean we mostly deal with integers (units) from the start. In each sample we should expect that:

- 50,000 ($0.005 \cdot 10,000$) people are infected.
- 95,000 ($0.95 \cdot 10,000$) people $+$ 50,000 ($0.005 \cdot 10,000$) people have the disease.
- 50,000 ($0.01 \cdot 10,000$) people $+$ 95,000 ($0.95 \cdot 10,000$) people test positive.
- 45,000 ($0.95 \cdot 50,000$) people have the disease and test positive.

and the ratio between the disease/non-disease positive results should be about $\frac{45}{5} = 9.0$.

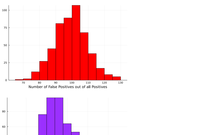
To model the system, we will need to be careful about how we contend with pure numbers to represent states within the system. First, we need to know if a person has the disease or not, so that changes the likelihood that they get a positive test. We have the probability that a person is sick is 0.005, with that a random number between 1 and 1,000. If the number is the single 1-5, then the person is sick, otherwise, they are healthy. Once we have determined the status of a given person's health, we need to know if they test positive. If they are sick, we need to generate a random number between 1 and 100 to determine if they test positive. If they are healthy, we need to generate a random number between 1 and 100. Likewise, given that they test positive, the probability of a false positive is 1% so we pick a random number between 1 and 100 and check if that number is 1.

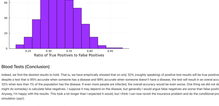
Now, in order for a true positive to occur, we must have that the person is sick and that they tested positive. We do this using flow of control. We first randomly assign the person to be sick or not. If they are sick, we randomly assign a test result. If they test the person is not sick, we again run a test result. We then add the appropriate probability simulation. As we do this, we update a count of the number of infections, positive test results, true positives, and false positives. From there, we'll keep everything we need. Let's write the simulation:

```
function in_simple_sweep(sample_size = 1e6)
    # Simulate a population of sample_size to test the disease
    # p_sick: P(D) in the notation above (probability of being sick)
    # p_false: P(p|D^c) false positive rate
    # p_true: P(p|D) true positive rate
    # p_infected: P(D)
    # p_positive: P(p)
    # p_true_pos: P(D|p)
    # Keep a 1-indexed array of the number of people
    p_sick = 0.005
    p_false = 0.01
    p_true = 0.95
    p_infected = 0
    p_positive = 0
    p_true_pos = 0
    for i in 1:sample_size
        # Randomly assign if the person is sick
        if rand(1:1000) <= 5
            p_infected += 1
            if rand(1:100) <= 95
                p_positive += 1
                p_true_pos += 1
        end
        # Randomly assign if false positive
        else
            if rand(1:100) <= 1
                p_positive += 1
            end
        end
    end
    p_pos = "Number of individuals tested and test result positive: $p_positive"
    p_inf = "Number of people infected: $p_infected"
    p_true = "Number of True Positives out of 100000: $p_true_pos"
    p_false = "Number of False Positives out of 100000: $(p_positive - p_true_pos)"
    p_ratio = "Ratio of True Positives to False Positives: $(p_true_pos/(p_positive - p_true_pos))"
    println(p_pos)
    print(p_inf)
    print(p_true)
    print(p_false)
    print(p_ratio)
end

in_simple_sweep(100000)
```

```
Number of individuals tested and test result positive: 9499
Number of people infected: 50000
Number of True Positives out of 100000: 45000
Number of False Positives out of 100000: 4499
Ratio of True Positives to False Positives: 9.000222222222222
```





Blood Tests Conclusions

Indeed, we find the desired results to hold. That is, we have empirically showed that on only 32% (roughly speaking) of positive test results will be true positives. So, despite a test that is 95% accurate when someone has a given disease and 99% accurate when someone doesn't, and a disease being only 0.5% of a population, only 32% when less than 1% of the population has the disease. Even more people are infected, the overall accuracy would be even worse. One thing we did not do (which I'm sure would be very easy) is to see how the ratio changes as the number of people who are infected increases. Intuitively, it should increase the ratio of true positives to false positives. Anyway, I'm happy with the results. This took a lot longer than I expected it would, but I think I can now revisit the insurance problem and do the conditional probability simulation (soon).

Multivariate Coefficients

In chapter 1, we saw that there are $\binom{n+r-1}{r}$ ways to form nonnegative integer vectors of dimension r that sum to n. In the script below, we will find those vectors. Why? Just for fun.

```
function nc_vec(n, r)
    # Find all vectors of dimension r that sum to n.
    # Assume n, r are positive integers.
    # Return a 2D array where each row is a vector.
    for k in 1:n-r+1
        for j in 1:r-1
            if j == k+1
            continue
            elseif j == r && n - k == 1
            continue
            end
            for s in 1:n-k-j
                for t in 1:n-k-j-s
                    if s + t == n - k - j
                    if s == t
                    continue
                    end
                    if s == 1 && t == 1
                    continue
                    end
                    if s == 1 && j == 1
                    continue
                    end
                    if t == 1 && j == 1
                    continue
                    end
                    if s == 1 && t == 2 && j == 2
                    continue
                    end
                    if s == 2 && t == 1 && j == 2
                    continue
                    end
                    if s == 2 && t == 2 && j == 2
                    continue
                    end
                    if s == 1 && t == 2 && j == 1
                    continue
                    end
                    if s == 2 && t == 1 && j == 1
                    continue
                    end
                    if s == 2 && t == 2 && j == 1
                    continue
                    end
                    if s == 1 && t == 1 && j == 2
                    continue
                    end
                    if s == 2 && t == 2 && j == 2
                    continue
                    end
                    if s == 1 && t == 1 && j == 1
                    continue
                    end
                    if s == 2 && t == 2 && j == 1
                    continue
                    end
                    if s == 1 && t == 2 && j == 2
                    continue
                    end
                    if s == 2 && t == 1 && j == 2
                    continue
                    end
                    if s == 2 && t == 2 && j == 2
                    continue
                    end
                    if s == 1 && t == 1 && j == 1
                    continue
                    end
                    if s == 2 && t == 2 && j == 1
                    continue
                    end
                    if s == 1 && t == 2 && j == 1
                    continue
                    end
                    if s == 2 && t == 1 && j == 1
                    continue
                    end
                    if s == 2 && t == 2 && j == 1
```