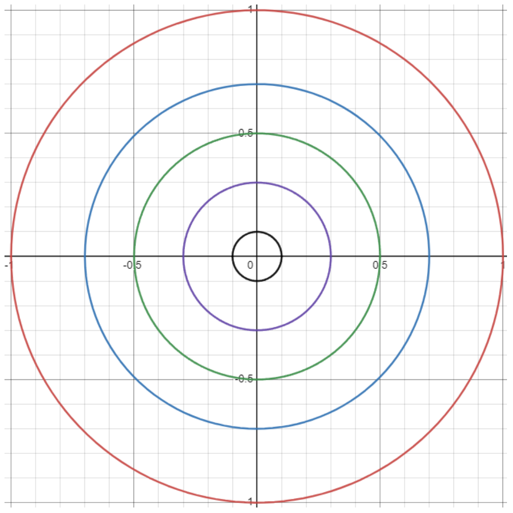
Homework Assignment (Problem Set) 4:

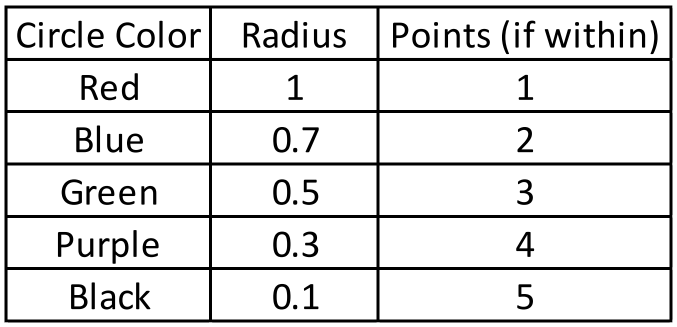
Note, Problem Set 3 directly focuses on Modules 7 and 8: Metaheuristic Algorithms and Monte Carlo Simulation

4 Questions

Rubric:
All questions worth 37.5 points
37.5 Points: Answer and solution are fully correct and detailed professionally.
25-37 Points: Answer and solution are deficient in some manner but mostly correct.
15-24 Points: Answer and solution are missing a key element or two.
1-14 Points: Answer and solution are missing multiple elements are significantly deficient/incomprehensible.
0 Points: No answer provided.

1. Perform a Monte Carlo simulation to estimate the probability of hitting each section of a dartboard (shown below) and the long-term average score of the player. Use the Monte Carlo simulation for both values – do not simply use probabilities to calculate the estimates. Assume that any dart thrown at the dartboard will hit somewhere on the space. Generate N pairs of random numbers (x,y) and use the equation of a circle ($x^2 + y^2 = r^2$) to determine which space a given dart hits. For example, if your random number is (0.4,0.6), we know that $r = \sqrt{0.4^2 + 0.6^2} = 0.721$, which equates to a 1 point shot between the blue and red circles. The radius and point value for each section is also shown below.





Circle Color	Radius	Points (if within)
Red	1	1
Blue	0.7	2
Green	0.5	3
Purple	0.3	4
Black	0.1	5

Circle Color	Radius	Points(if within)
Red	1	1
Blue	0.7	2
Green	0.5	3
Purple	0.3	4
Black	0.1	5

Part A: Determine the probability of hitting each section of the dartboard, to include hitting the section outside of the red circle

Optimal value of N: 12400
Probabilities of hitting each section:
Red: 0.3960
Blue: 0.1961
Green: 0.1238
Purple: 0.0623
Black: 0.0087
Outside: 0.2131

Part B: Determine the long-term average score per shot of a player.

Long-term average score per shot with optimal N: 1.4525

Python code is available in attached jupyter notebook which solves this problem.

2. A bicycle shop, Take a Bike, offers bonuses to its sales team for selling more than 4 bicycles in a day. Each salesperson can sell between 0 and 8 bikes per day and has a 40% probability of selling more than 4 bicycles in any given day (60% probability of selling 4 or fewer). If the salesperson sells more than 4 bikes, the probability of selling 5, 6, 7, or 8 bikes is shown below. The bonus that is paid is dependent on the model of each bike sold, each of which has a different probability of sale and bonus payout (also shown below). If the salesperson sells more than 4 bikes, the bonus is paid for each of the bikes sold, not just the number above 4. So if a salesperson sells 6 bikes, the bonus is paid for each of those 6 bikes, not just the 2 above the threshold. Develop a simulation model to calculate the bonus a salesperson can expect in a day. Do not simply calculate expected values for this – use random numbers to determine the number and types of bikes sold. Ensure you iterate your simulation multiple times (between 400 and 20,000).

Number sold	Probability
5	40%
6	35%
7	20%
8	5%

Model	Portion of total sales	Bonus
A	45%	$ 10
B	35%	$ 15
C	15%	$ 25
D	5%	$ 30

The mean daily bonus is $35.15

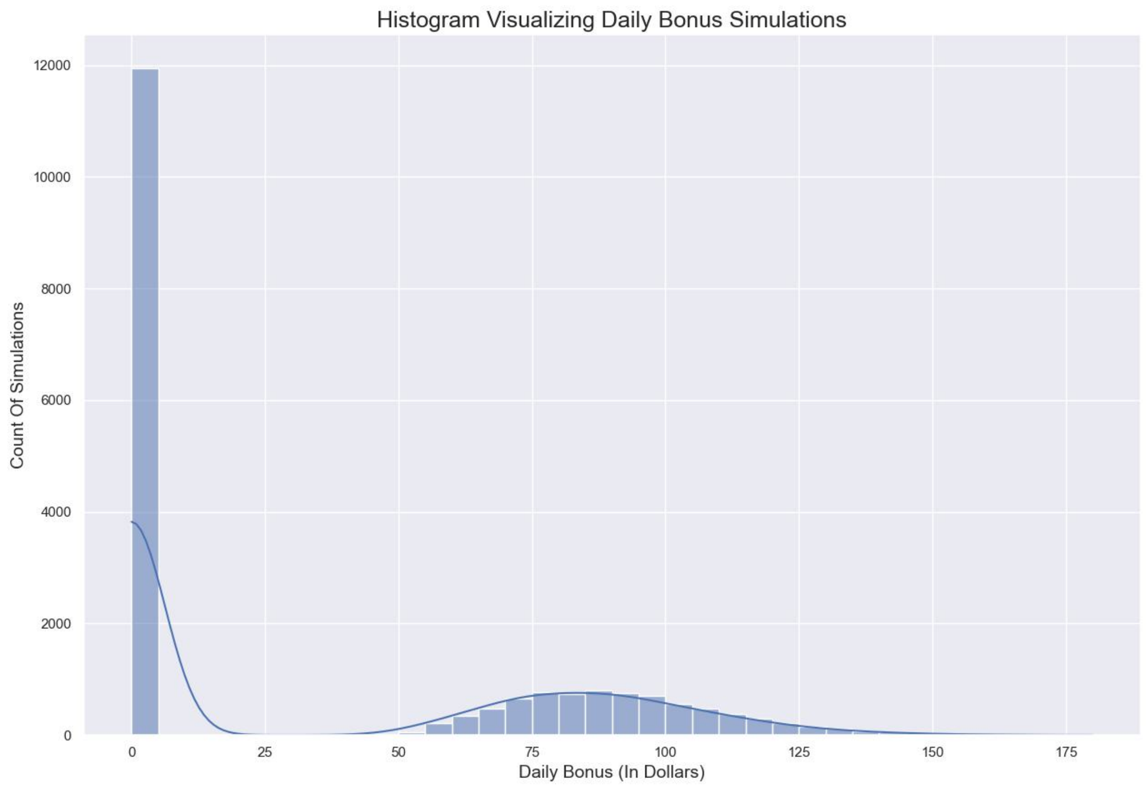


Fig 1: Daily Bonus Histogram for multiple simulations

Python code is available in attached jupyter notebook which solves this problem.

3. Jim is investing in his company's 401(k) retirement plan, funding 6% of his salary to get a 3% match (thus effectively investing 9% of his annual salary). He invests in each of the three available funds. 50% of his contributions go into investment A, which has an average return of 6.91% with a standard deviation of 12.89%. The rest of his contributions are equally divided between investment B, which has an average return of 8.94% with a standard deviation of 15.21%, and investment C, which has an average return of 9.88% with a standard deviation of 17.14%. Jim is currently 24 years old and earns $55,000 this year, but anticipates a pay raise of, on average, 2.83% with a standard deviation of 0.72%. Develop a simulation model to predict his 401(k) balance at age 60. Ensure you iterate your simulation multiple times (between 400 and 20,000).

The mean amount accrued in the 401ks in this simulation study is $1545940.79

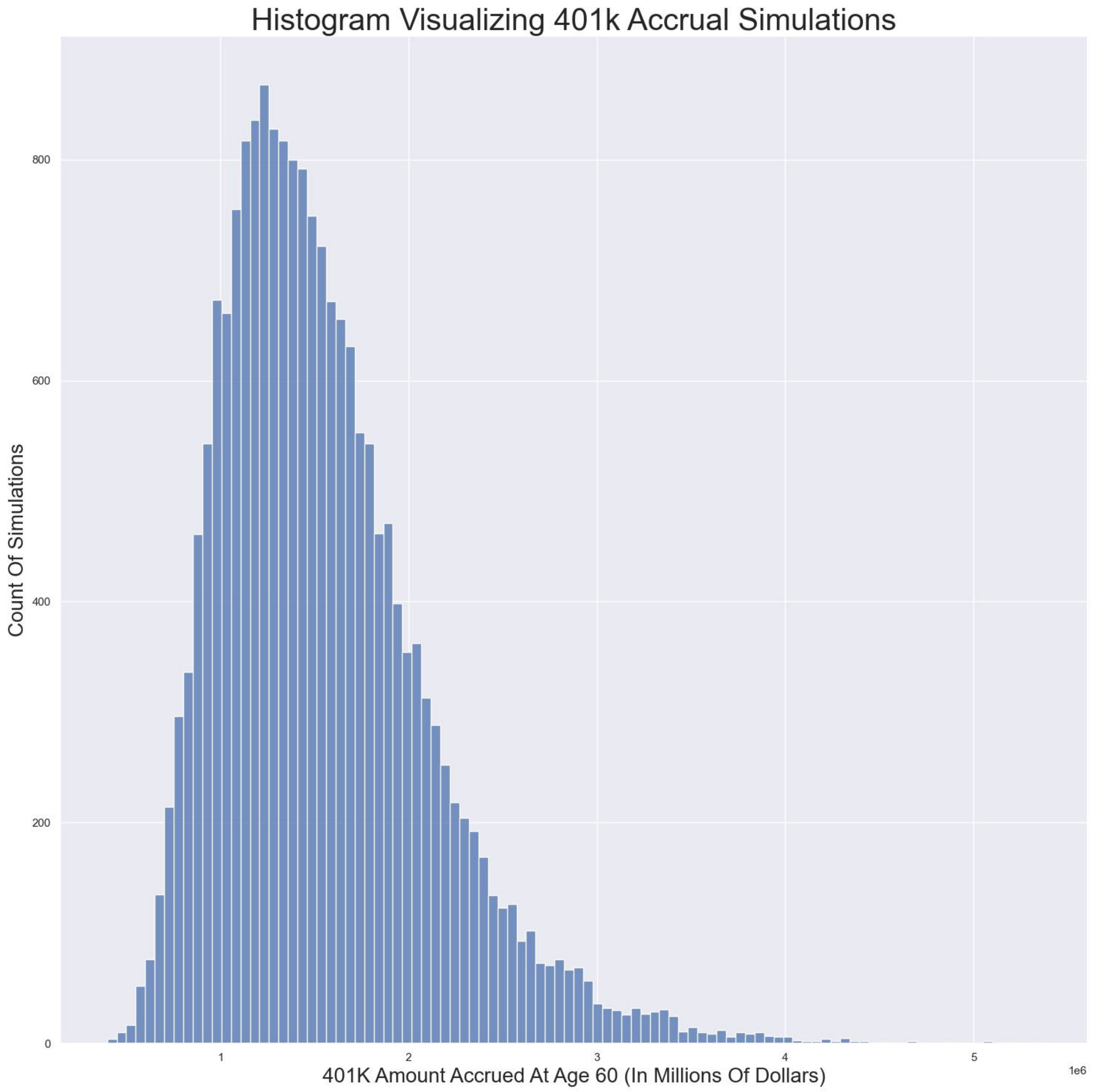


Fig 2: Histogram of 401K Amount Accrued at Age 60 using multiple simulation.

Python code is available in attached jupyter notebook which solves this problem.

4. Develop a simple heuristic (I recommend a construction or destruction heuristic) in either R or Python to solve the following knapsack problem: (Note, this problem can be solved to optimality using integer programming; however, the focus of this question is on developing a heuristic and metaheuristic). Then, develop a metaheuristic using your heuristic as a subroutine. You can iterate the heuristic *n* times and return the best solution, prevent previous solutions from being selected, etc. I am evaluating your ability to find a solution quickly with a heuristic and your approach to *improve* the heuristic with a metaheuristic.

Maximize	$12x_1 + 16x_2 + 22x_3 + 8x_4$
S.T.	$4x_1 + 5x_2 + 7x_3 + 3x_4 \leq 140$
	$0 \leq x_i \leq 10$ and x_i is integer

Initial Solution (Greedy Heuristic): [10, 10, 7, 0]
Objective Value (Greedy Heuristic): 434

Final Solution (Simulated Annealing Metaheuristic): [10, 10, 7, 0]
Objective Value (Simulated Annealing Metaheuristic): 434

Python code is available in attached jupyter notebook which solves this problem.

5. Develop a simulated annealing procedure in either R or Python to solve the same knapsack problem: (Note, this problem can be solved to optimality using integer programming; however, the focus of this question is on developing the simulated annealing method). Do not simply return a shell code from a web search, but try to implement the simulated annealing metaheuristic for this specific problem.

Maximize	$12x_1 + 16x_2 + 22x_3 + 8x_4$
S.T.	$4x_1 + 5x_2 + 7x_3 + 3x_4 \leq 140$
	$0 \leq x_i \leq 10$ and x_i is integer

final:

    x1: 10
    x2: 10
    x3: 7
    x4: 0
    objective: 434

Python code is available in attached jupyter notebook which solves this problem.