LINEAR PROGRAMMING: A RESPONSE TO COVID-19

Vineet Parvathala

ABSTRACT

The purpose of Math 214: Linear Algebra is to begin to teach students how model natural phenomena and solve problems involving them efficiently. Linear programming is an optimization method that takes various linear inequalities relating to a situation and uses them to find the best or "optimal" solution. This was the primary content learned in the months leading up to the coronavirus pandemic so it only seemed fitting to use linear programming to study the current COVID-19 situation for our final project.

INTRODUCTION

In this project, I will be dealing with the availability of ventilators in hospitals in India. With population of 1.353 billion, India will face difficult challenges in their handling of patients and their hospital supplies like masks, ventilators, and testing swabs. A hospital-grade ventilator can be costly, ranging from $5,000 to $25,000. With the economic fears lurking, governments have to balance resisting large expenditures while providing necessary medical resources for citizen. Ventilators have become a key factor in treatment of the virus and is required when the lungs of patients suffering from COVID-19 tire out and need support through ventilators to assist with breathing.

LINEAR PROGRAMMING

Linear Programming is based upon a number of constraints and is a popular methodology for determining how to properly optimize and allocate resources in a given situation. Our project goal is to relate linear programming to the ongoing situation regarding COVID-19, by optimizing life-years for patients in India based on data regarding ventilators and patient profiles.

COMPUTATIONS

I used Python to program this project since it has several libraries that have uses in machine learning, data science, and linear algebra. In this project, I utilized a linear programming library called puLP to aid in our calculations. As seen in the code, I will utilize several functions outlined in puLP. These include LpMaximize, LpProblem, LpVariable, and LpStatus. Since this is a linear programming maximization problem, I will be using LpMaximize instead of LpMinimize.

DATA

I aimed to keep our data as current as possible and did so by pulling the number of active cases directly from a Coronavirus Data API that monitors both John Hopkins and Worldometer data. In order to make hypothetical conclusions, I assumed survival rates with ventilators and survival rates without ventilators. I also had to gather data on life expectancy in India and the distribution of cases based on age ranges in India. I utilized a normal distribution model in order to emulate a "first-come first-serve" situation to allocate ventilators.

Active Cases → Coronavirus Data API

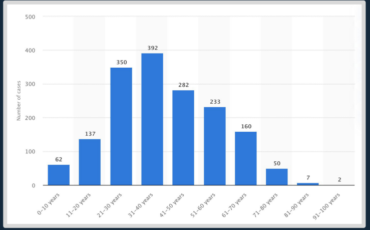
Life Expectancy → 68.7 years

Example Calculation: 31-40 Age Bracket

68.7 - (40+31)/2 = 33.2 years

Age Bracket	Survival Rate with Ventilator	Survival Rate w/o Ventilator	Average Life Expectancy
0-10	80%	40%	63.2
11-20	75%	35%	53.2
21-30	70%	35%	43.2
31-40	65%	30%	33.2
41-50	60%	25%	23.2
51-60	25%	12%	13.2
61-70	23%	15%	3.2
71-80	16%	8%	0
81-90	16%	8%	0

Random Ventilator Usage is used to simulate the "First-Come First-Serve" rule. The Random number generator correlates to the Normal Distribution that's centered around 35 years with a standard deviation of 17 years. Random number generator is then run the same number of times as the number of ventilators available.. For each result, if it is within a age bracket range (i.e. between 31-40), the ventilator usage for that bracket is increased by 1. This is the graph →



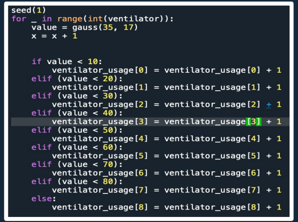
PROGRAM

The user is prompted to enter the number of ventilators available. Then, using the current active cases and percentage of cases for each age bracket, the estimated number of cases in each age bracket is displayed. It sets aside ventilators for those with severe and critical cases and works to determine a distribution of ventilators for those with mild cases. A "first-come first-serve" ventilators usage model displays how many ventilators were distributed to each age group. The total number of life years expected with this ventilator distribution is displayed. Our linear programming model then optimizes the number of life-years through set constraints and displays the optimized distribution of ventilators. The optimized number of total life-years is then displayed. It then computes how much the optimization model was able to increase life-years.

THE CODE

Random Ventilator Usage to Simulate "First-Come First-Serve": Random number generator correlating to Normal Distribution centered around 35 years with a standard deviation of 17 years.

Random number generator is run # ventilators times. For each result, if it is within a age bracket range (i.e. between 31-40), the ventilator usage for that bracket is increased by 1.



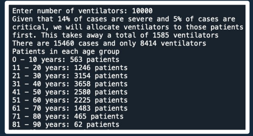
```
seed(1)
for _ in range(int(ventilator)):
    value = gauss(35, 17)
    x = x + 1


    if value < 10:
        ventilator_usage[0] = ventilator_usage[0] + 1
    elif (value < 20):
        ventilator_usage[1] = ventilator_usage[1] + 1
    elif (value < 30):
        ventilator_usage[2] = ventilator_usage[2] + 1
    elif (value < 40):
        ventilator_usage[3] = ventilator_usage[3] + 1
    elif (value < 50):
        ventilator_usage[4] = ventilator_usage[4] + 1
    elif (value < 60):
        ventilator_usage[5] = ventilator_usage[5] + 1
    elif (value < 70):
        ventilator_usage[6] = ventilator_usage[6] + 1
    elif (value < 80):
        ventilator_usage[7] = ventilator_usage[7] + 1
    else:
        ventilator_usage[8] = ventilator_usage[8] + 1
```

SAMPLE RUN

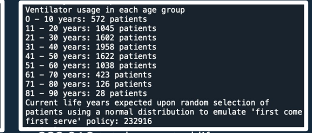
The user entered 10000 for the number of ventilators.

Patients in Each Age Group:



```
Enter number of ventilators: 10000
Given that 14% of cases are severe and 5% of cases are
critical, we will allocate ventilators to those patients
first. This takes away a total of 1585 ventilators
There are 15460 cases and only 8414 ventilators
Patients in each age group
0 - 10 years: 563 patients
11 - 20 years: 1246 patients
21 - 30 years: 3154 patients
31 - 40 years: 3658 patients
41 - 50 years: 2580 patients
51 - 60 years: 2225 patients
61 - 70 years: 1483 patients
71 - 80 years: 465 patients
81 - 90 years: 62 patients
```

Ventilator Usage in Each Age Group:

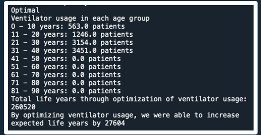


```
Ventilator usage in each age group
0 - 10 years: 572 patients
11 - 20 years: 1045 patients
21 - 30 years: 1602 patients
31 - 40 years: 1958 patients
41 - 50 years: 1622 patients
51 - 60 years: 1038 patients
61 - 70 years: 423 patients
71 - 80 years: 126 patients
81 - 90 years: 28 patients
Current life years expected upon random selection of
patients using a normal distribution to emulate 'first come
first serve' policy: 232916
```

232,916 total expected life years

Optimal Ventilator Usage in Each Age Group

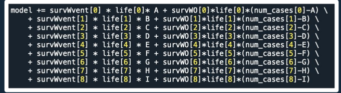
260,520 expected life years. Increase of 27,604 life years through linear programming model



```
Optimal
Ventilator usage in each age group
0 - 10 years: 563.0 patients
11 - 20 years: 1246.0 patients
21 - 30 years: 3154.0 patients
31 - 40 years: 3451.0 patients
41 - 50 years: 0.0 patients
51 - 60 years: 0.0 patients
61 - 70 years: 0.0 patients
71 - 80 years: 0.0 patients
81 - 90 years: 0.0 patients
Total life years through optimization of ventilator usage:
260520
By optimizing ventilator usage, we were able to increase
expected life years by 27604
```

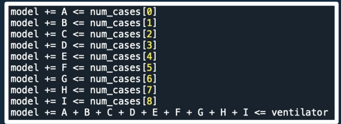
ALGORITHM

Linear Maximizing Model →
This equation is responsible for maximizing the total number of life years possible through linear programming (LpMaximize)



```
model += survWvent[0] * life[0]* A + survWO[0]*life[0]*(num_cases[0]-A) \
    + survWvent[1] * life[1] * B + survWO[1]*life[1]*(num_cases[1]-B) \
    + survWvent[2] * life[2] * C + survWO[2]*life[2]*(num_cases[2]-C) \
    + survWvent[3] * life[3] * D + survWO[3]*life[3]*(num_cases[3]-D) \
    + survWvent[4] * life[4] * E + survWO[4]*life[4]*(num_cases[4]-E) \
    + survWvent[5] * life[5] * F + survWO[5]*life[5]*(num_cases[5]-F) \
    + survWvent[6] * life[6] * G + survWO[6]*life[6]*(num_cases[6]-G) \
    + survWvent[7] * life[7] * H + survWO[7]*life[7]*(num_cases[7]-H) \
    + survWvent[8] * life[8] * I + survWO[8]*life[8]*(num_cases[8]-I)
```

Constraints & Variables →
A, B, C, D, E, F, G, H, and I are the number of ventilators in optimized model. The total number of ventilators must not surpass "ventilator" which is the user-inputted number of ventilators available



```
model += A <= num_cases[0]
model += B <= num_cases[1]
model += C <= num_cases[2]
model += D <= num_cases[3]
model += E <= num_cases[4]
model += F <= num_cases[5]
model += G <= num_cases[6]
model += H <= num_cases[7]
model += I <= num_cases[8]
model += A + B + C + D + E + F + G + H + I <= ventilator
```

CONCLUSION

Using this model, I was able to project an optimized value of life-years saved, compared to a random selection of patients requiring ventilators arriving at hospitals in India. The model significantly favors younger populations, as their life expectancies are much greater than those of greater ages. By utilizing the model created, hospitals would run into ethical/moral dilemmas, leaving older patients to die while helping younger patients. This is a common situation that has arisen in real hospitals all over the world, as ventilators are not in huge supply and strategic decisions must be made for the greater good.

SOURCES

Percentages of mild, severe, and critical cases:
https://www.cdc.gov/coronavirus/2019-ncov/hcp/clinical-guidance-management-patients.html
Percentages of each age group affected by Coronavirus:
https://www.statista.com/statistics/1110522/india-number-of-coronavirus-cases-by-age-group/
Assistance on determining a realistic numbers for our survival rates with and without ventilators:
https://medium.com/swlh/optimised-ventilator-deployment-95c2a6cf98b7
Link to Github Repository: https://github.com/vparv/covid19-ventilators