Homework Eleven

Chelsea Hughes

Question 15.2

In the videos, we saw the "diet problem". (The diet problem is one of the first large-scale optimization problems to be studied in practice. Back in the 1930's and 40's, the Army wanted to meet the nutritional requirements of its soldiers while minimizing the cost.) In this homework you get to solve a diet problem with real data. The data is given in the file diet.xls. 1. Formulate an optimization model (a linear program) to find the cheapest diet that satisfies the maximum and minimum daily nutrition constraints, and solve it using PuLP. Turn in your code and the solution. (The optimal solution should be a diet of air-popped popcorn, poached eggs, oranges, raw iceberg lettuce, raw celery, and frozen broccoli. UGH!) 2. Please add to your model the following constraints (which might require adding more variables) and solve the new model: a. If a food is selected, then a minimum of 1/10 serving must be chosen. (Hint: now you will need two variables for each food i: whether it is chosen, and how much is part of the diet. You'll also need to write a constraint to link them.) b. Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected. c. To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected. [If something is ambiguous (e.g., should bean-and-bacon soup be considered meat?), just call it whatever you think is appropriate – I want you to learn how to write this type of constraint, but I don't really care whether we agree on how to classify foods!] If you want to see what a more full-sized problem would look like, try solving your models for the file diet_large.xls, which is a low-cholesterol diet model (rather than minimizing cost, the goal is to minimize cholesterol intake). I don't know anyone who'd want to eat this diet – the optimal solution includes dried chrysanthemum garland, raw beluga whale flipper, freeze-dried parsley, etc. – which shows why it's necessary to add additional constraints beyond the basic ones we saw in the video! [Note: there are many optimal solutions, all with zero cholesterol, so you might get a different one. It probably won't be much more appetizing than mine.]

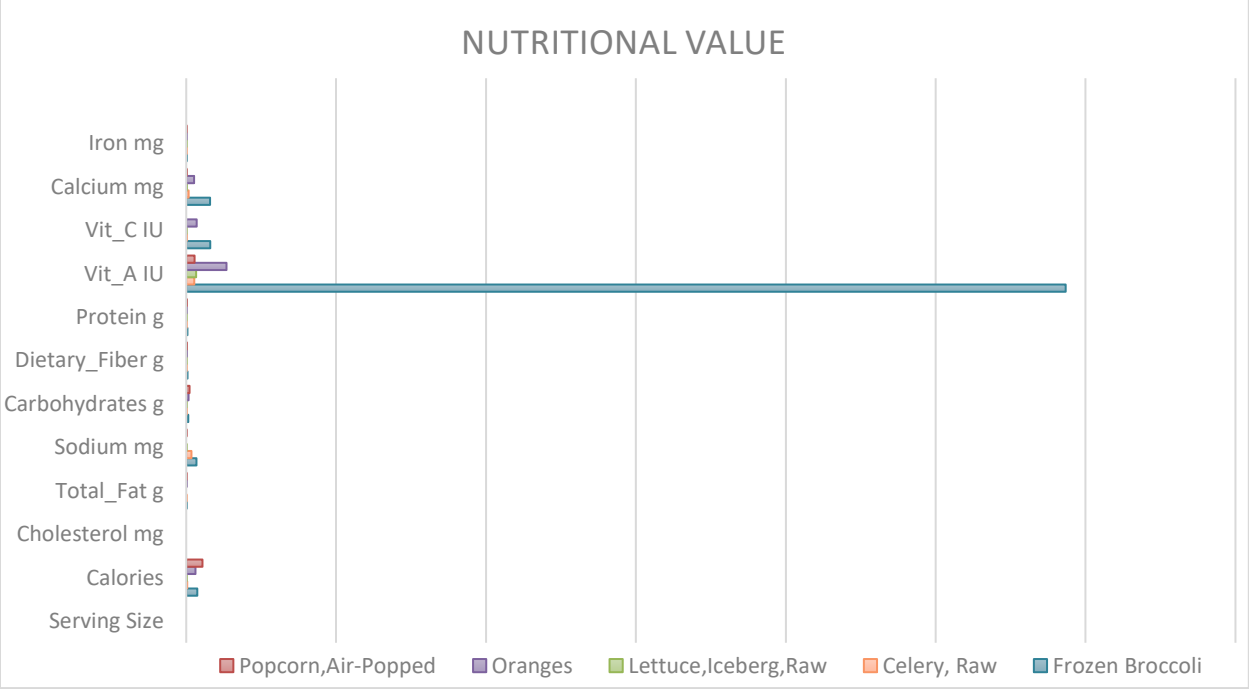
15.2.1

```
from pulp import *
import pandas as pd

#import data and convert to data frame
data1 = pd.read_csv('diet.csv')
data1 = data1.iloc[0:64,:] [1]

#visualize all foods
allfood1 = list(data1['Foods'])

#obtain price for all food (per serving size)
```

```
price = [float(price.replace('$','')) for price in data1['Price/ Serving']]
cost = {food:price for food, price in zip(allfood1,price)}

#visualize nutrients
nutrients1 = list(data1.columns[3:14])

#assign limits for each nutrient [2]
limit = data1.iloc[65:68,3::]
limitmin = {nutrient:limit for nutrient, limit in zip(nutrients1,limit.iloc[0,:])}
limitmax = {nutrient:limit for nutrient, limit in zip(nutrients1,limit.iloc[1,:])}

#prepare list to store nutrient content
servingsize = []

#store nutrient content
for i in nutrients1: [3]
    x = {food:nutrient for food, nutrient in zip(allfood1, list(data1[i]))}
    servingsize[i]=x

#initiate optimization problem
dietproblem = LpProblem(name='army_diet', sense=LpMinimize)

#create food dictionary
allfood2 = LpVariable.dicts("food",allfood1,0) [4]
selectedfood = LpVariable.dicts('selectedfood', allfood1, 0, 1, LpBinary)

#calculate the total cost of meal
dietproblem += lpSum([cost[i]*allfood2[i] for i in allfood1]), 'Total daily meal
cost' [5]

#add constraints for each nutrient
for nutrient in nutrients1:
    dietproblem += lpSum(servingsize[nutrient][food]*allfood2[food] for food in
allfood1)>=limitmin[nutrient],''.join([nutrient,'lower_req'])
    dietproblem += lpSum(servingsize[nutrient][food]*allfood2[food] for food in
allfood1)<=limitmax[nutrient],''.join([nutrient,'upper_req'])

#store optimization data
dietproblem.writeLP("dietproblem.lp")

#solve optimization problem
dietproblem.solve()
print ("Status:", LpStatus[dietproblem.status])
```

```
#visualize optimum value of each food [6]
for v in dietproblem.variables():
    if v.varValue !=0:
        print (v.name, "=", v.varValue)

#visualize optimum value of each meal
print ("Total daily meal cost = ", value(dietproblem.objective))
```

The optimum cost for a daily meal was $4.38 daily. With this cost, one would be allowed to consume *popcorn*, *eggs*, *oranges*, *lettuce*, *celery* and *broccoli*. Obviously, this diet would not contain nutritional value so I see the point of adding constraints to ensure proper nutrition as indicated in the chart below:

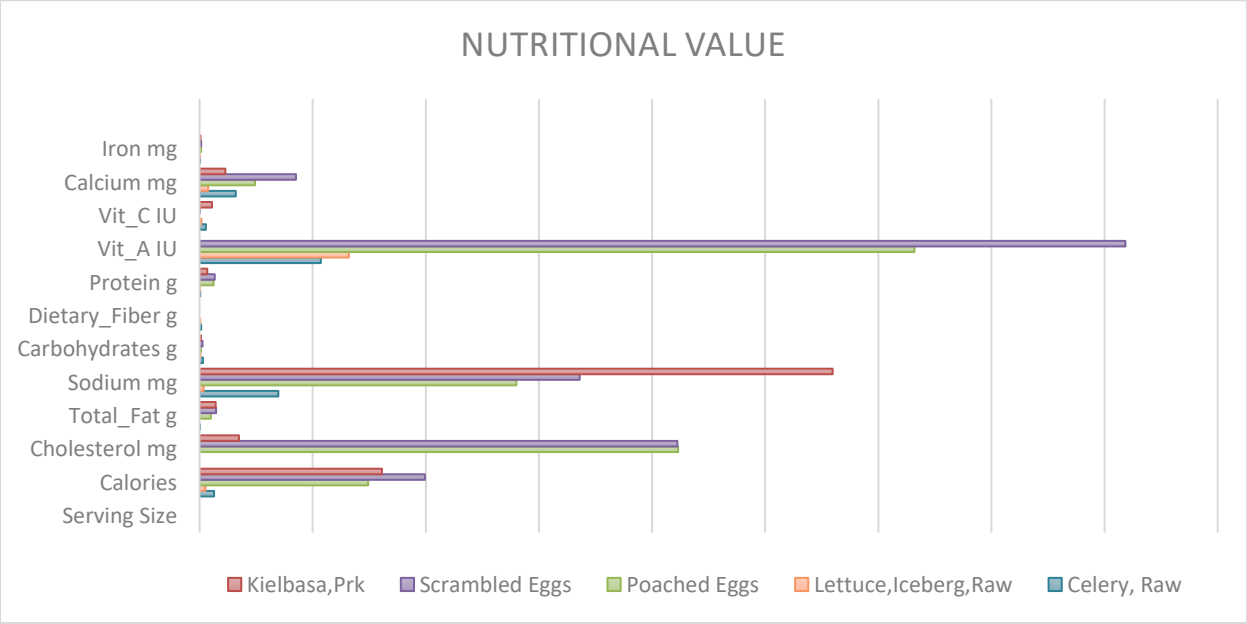


15.2.2

```
#ensure 1/10 serving size
for food in allfood1:
    dietproblem += allfood2[food] >= 0.1*selectedfood[food], ''.join([food,
'chosen'])
    dietproblem += selectedfood[food] >= allfood2[food]*0.0000001

#broccoli or celery [7]
dietproblem += selectedfood['Celery Raw'] + selectedfood['Frozen Broccoli'] <= 1
```

```
#choose three proteins
dietproblem += selectedfood['Bologna, Turkey'] + selectedfood['Frankfurter, Beef']
+ selectedfood['Ham,Sliced,Extralean'] + selectedfood['Hamburger W/Toppings'] +
selectedfood['Hotdog, Plain'] + selectedfood['Kielbasa,Prk'] +
selectedfood['Roasted Chicken'] + selectedfood['Poached Eggs'] +
selectedfood['Pork'] + selectedfood['Sardines in Oil'] + selectedfood['Scrambled
Eggs'] + selectedfood['Tofu'] + selectedfood['White Tuna in Water'] <=3

#solve optimization problem
dietproblem.solve()
print ("Status:", LpStatus[dietproblem.status])

#visualize optimum value of each food
for v in dietproblem.variables():
    if v.varValue !=0:
        print (v.name, "=", v.varValue)

#visualize optimum value of each meal
print ("Total cost with first round of constraints = ",
value(dietproblem.objective))
```

The optimum cost for a daily meal was $4.64 daily. The added constraints concluded to be *celery*, eggs (*poached* and *scrambled*), and *kielbasa pork*. Comparatively to 15.2.1, this has a much higher nutritional value per serving size cost; however, it is still not particularly enough so you would really need to add more constraints.



References

[1] https://www.geeksforgeeks.org/python-extracting-rows-using-pandas-iloc/

[2] http://benalexkeen.com/linear-programming-with-python-and-pulp-part-5/

[3] https://www.geeksforgeeks.org/create-a-list-from-rows-in-pandas-dataframe/

[4] https://www.coin-or.org/PuLP/pulp.html

[5] https://www.programcreek.com/python/example/96844/pulp.lpSum

[6] https://www.coin-or.org/PuLP/CaseStudies/a_blending_problem.html

[7] https://towardsdatascience.com/linear-programming-and-discrete-optimization-with-python-using-pulp-449f3c5f6e99

[2] pretty much designed my flow for me. It is an introduction to linear programming for Python beginners using PuLP. So, basically, it was like my second birthday of the year. Because I didn't know what I was doing, I was simultaneously working in Excel to visualize the data in a way that I knew to decipher it. The two graphs I thought my interesting were attached. I debated between those and the financial assessment pie charts (which were also very interesting).