Week 7 Homework

July 1, 2020

0.1 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.

Importing packages

[132]:
```
from pulp import *
import pandas as pd
import numpy as np
```

Read in Data

[12]:
```
#Please change file paths!

diet = pd.read_excel('C:/Users/ujjaw/OneDrive/Documents/GaTech/ISYE 6501/Week 7/
↪diet.xls')
diet_large = pd.read_excel('C:/Users/ujjaw/OneDrive/Documents/GaTech/ISYE 6501/
↪Week 7/diet_large.xls')
```

Binary Variable that specifies poultry

[13]:
```
# Creating a binary variable that specifies poultry or not

poultry = np.zeros(64)
poultry[[8, 27, 28, 29, 30, 31, 32, 41, 43, 44, 49, 50, 51, 56, 57, 58, 59, 61,␣
↪63]] = 1
diet['Poultry'] = pd.Series(poultry)
```

[123]:
```
diet.head()
```

[123]:
```
                   Foods  Price/ Serving      Serving Size  Calories  \
0      Frozen Broccoli            0.16        10 Oz Pkg      73.8
1          Carrots,Raw            0.07  1/2 Cup Shredded      23.7
2          Celery, Raw            0.04           1 Stalk       6.4
3          Frozen Corn            0.18           1/2 Cup      72.2
4  Lettuce,Iceberg,Raw            0.02            1 Leaf       2.6
```

```
   Cholesterol mg  Total_Fat g  Sodium mg  Carbohydrates g  Dietary_Fiber g  \
0             0.0          0.8       68.2             13.6              8.5   
1             0.0          0.1       19.2              5.6              1.6   
2             0.0          0.1       34.8              1.5              0.7   
3             0.0          0.6        2.5             17.1              2.0   
4             0.0          0.0        1.8              0.4              0.3   

   Protein g  Vit_A IU  Vit_C IU  Calcium mg  Iron mg  Poultry  
0        8.0    5867.4     160.2       159.0      2.3      0.0  
1        0.6   15471.0       5.1        14.9      0.3      0.0  
2        0.3      53.6       2.8        16.0      0.2      0.0  
3        2.5     106.6       5.2         3.3      0.3      0.0  
4        0.2      66.0       0.8         3.8      0.1      0.0  
```

Preparing Data for PuLP package

[129]:
```
diet_data = diet[0:64]
diet_data = diet_data.values.tolist()
foods = [x[0] for x in diet_data]
cost = {x[0]: float(x[1]) for x in diet_data}
cals = {x[0]: float(x[3]) for x in diet_data}
cholestrol = {x[0]: float(x[4]) for x in diet_data}
fat = {x[0]: float(x[5]) for x in diet_data}
sodium  = {x[0]: float(x[6]) for x in diet_data}
carbs   = {x[0]: float(x[7]) for x in diet_data}
fiber   = {x[0]: float(x[8]) for x in diet_data}
protein = {x[0]: float(x[9]) for x in diet_data}
Vit_A = {x[0]: float(x[10]) for x in diet_data}
Vit_C = {x[0]: float(x[11]) for x in diet_data}
Calcium = {x[0]: float(x[12]) for x in diet_data}
Iron = {x[0]: float(x[13]) for x in diet_data}
Poultry = {x[0]: float(x[14]) for x in diet_data}


# Script for preparing the data so it can be looped over. Unfortunately, looped␣
 ↪script generated an error.

# l = []
# names = []
# nutrient_restrictions = []
# for i in range(3, diet.shape[1]):
#     names.append(diet.columns.values[i])
#     l.append({x[0]: float(x[i]) for x in diet_data})
#     nutrient_restrictions.append([diet.iloc[65, i], diet.iloc[66, i]])
# l[0]
```

0.1.1 Question 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!)()

Writing out Optimization Problem

[130]:
```
problem = LpProblem('Diet Problem', LpMinimize)
selectVars = LpVariable.dicts('Picked', foods, 0, 1, cat = 'Integer')
amountVars = LpVariable.dicts('Amounts', foods, 0)


problem += lpSum([cost[i] * amountVars[i] for i in foods]), 'total cost'


# Script for looping over the nutrients which for some reason generated an␣
↪error.

# for a in range(len(l)):
#     problem += lpSum([l[a][i] * amountVars[i] for i in foods]) >=␣
↪nutrient_restrictions[a][1], 'max ' + names[a]
#     problem += lpSum([l[a][i] * amountVars[i] for i in foods]) <=␣
↪nutrient_restrictions[a][0], 'min ' + names[a]


problem += lpSum([cals[i] * amountVars[i] for i in foods]) >= diet.iloc[65, 3],␣
↪'min cals'
problem += lpSum([cals[i] * amountVars[i] for i in foods]) <= diet.iloc[66, 3],␣
↪'max cals'

problem += lpSum([cholestrol[i] * amountVars[i] for i in foods]) >= diet.
↪iloc[65, 4], 'min cholestrol'
problem += lpSum([cholestrol[i] * amountVars[i] for i in foods]) <= diet.
↪iloc[66, 4], 'max cholestrol'

problem += lpSum([fat[i] * amountVars[i] for i in foods]) >= diet.iloc[65, 5],␣
↪'min fat'
problem += lpSum([fat[i] * amountVars[i] for i in foods]) <= diet.iloc[66, 5],␣
↪'max fat'

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
↪6], 'min sodium'
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
↪6], 'max sodium'
```

```
problem += lpSum([carbs[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪7], 'min carbs'
problem += lpSum([carbs[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪7], 'max carbs'

problem += lpSum([fiber[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪8], 'min fiber'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪8], 'max fiber'

problem += lpSum([protein[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪9], 'min protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪9], 'max protein'

problem += lpSum([Vit_A[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪10], 'min Vit_A'
problem += lpSum([Vit_A[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪10], 'max Vit_A'

problem += lpSum([Vit_C[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪11], 'min Vit_C'
problem += lpSum([Vit_C[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪11], 'max Vit_C'

problem += lpSum([Calcium[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪12], 'min Calcium'
problem += lpSum([Calcium[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪12], 'max Calcium'

problem += lpSum([Iron[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪13], 'min Iron'
problem += lpSum([Iron[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪13], 'max Iron'
```

```
C:\Users\ujjaw\anaconda3\lib\site-packages\pulp\pulp.py:1114: UserWarning:
Spaces are not permitted in the name. Converted to '_'
  warnings.warn("Spaces are not permitted in the name. Converted to '_'")
```

Solving the Problem

[131]:
```
problem.solve()
varsDictionary = {}
for v in problem.variables():
    varsDictionary[v.name] = v.varValue
    if v.varValue >0:
        print ("{} = {}".format(v.name, v.varValue))
```

```
Amounts_Celery,_Raw = 52.64371
Amounts_Frozen_Broccoli = 0.25960653
Amounts_Lettuce,Iceberg,Raw = 63.988506
Amounts_Oranges = 2.2929389
Amounts_Poached_Eggs = 0.14184397
Amounts_Popcorn,Air_Popped = 13.869322
```

Results As we can see, we have a solution with the only constraints (other than minimum serving size being 0) is that the diet is above and below the minimum and maximum nutronal requirements. Doesn't look to be a good diet though… I certainly wouldn't want to eat it.

0.1.2 Question 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

Writing out the Optimization Problem

[121]:
```
problem = LpProblem('Diet Problem', LpMinimize)
selectVars = LpVariable.dicts('Picked', foods, 0, 1, cat = 'Integer')
amountVars = LpVariable.dicts('Amounts', foods, 0)



problem += lpSum([cost[i] * amountVars[i] for i in foods]), 'total cost'

problem += lpSum([cals[i] * amountVars[i] for i in foods]) >= diet.iloc[65, 3],␣
 ↪'min cals'
problem += lpSum([cals[i] * amountVars[i] for i in foods]) <= diet.iloc[66, 3],␣
 ↪'max cals'

problem += lpSum([cholestrol[i] * amountVars[i] for i in foods]) >= diet.
 ↪iloc[65, 4], 'min cholestrol'
problem += lpSum([cholestrol[i] * amountVars[i] for i in foods]) <= diet.
 ↪iloc[66, 4], 'max cholestrol'

problem += lpSum([fat[i] * amountVars[i] for i in foods]) >= diet.iloc[65, 5],␣
 ↪'min fat'
problem += lpSum([fat[i] * amountVars[i] for i in foods]) <= diet.iloc[66, 5],␣
 ↪'max fat'

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪6], 'min sodium'
```

```
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪6], 'max sodium'

problem += lpSum([carbs[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪7], 'min carbs'
problem += lpSum([carbs[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪7], 'max carbs'

problem += lpSum([fiber[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪8], 'min fiber'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪8], 'max fiber'

problem += lpSum([protein[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪9], 'min protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪9], 'max protein'

problem += lpSum([Vit_A[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪10], 'min Vit_A'
problem += lpSum([Vit_A[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪10], 'max Vit_A'

problem += lpSum([Vit_C[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪11], 'min Vit_C'
problem += lpSum([Vit_C[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪11], 'max Vit_C'

problem += lpSum([Calcium[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪12], 'min Calcium'
problem += lpSum([Calcium[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪12], 'max Calcium'

problem += lpSum([Iron[i] * amountVars[i] for i in foods]) >= diet.iloc[65,␣
 ↪13], 'min Iron'
problem += lpSum([Iron[i] * amountVars[i] for i in foods]) <= diet.iloc[66,␣
 ↪13], 'max Iron'


#Minimum 1/10 Serving Size
for i in foods:
    problem += amountVars[i] >= selectVars[i]*0.1
    problem += amountVars[i] <= selectVars[i]*100000

#Poultry Requirement
```

```
problem += lpSum([Poultry[i] * selectVars[i] for i in foods]) >= 3, 'poultry␣
 ↪requirement'

#Frozen brocolli and celery requirement
problem += selectVars['Celery, Raw'] + selectVars['Frozen Broccoli'] <= 1
```

Solve Optimization Problem

[127]:
```
problem.solve()
varsDictionary = {}
for v in problem.variables():
    varsDictionary[v.name] = v.varValue
    if v.varValue >0:
        print ("{} = {}".format(v.name, v.varValue))
```

```
Amounts_Celery,_Raw = 42.399358
Amounts_Kielbasa,Prk = 0.1
Amounts_Lettuce,Iceberg,Raw = 82.802586
Amounts_Oranges = 3.0771841
Amounts_Peanut_Butter = 1.9429716
Amounts_Poached_Eggs = 0.1
Amounts_Popcorn,Air_Popped = 13.223294
Amounts_Scrambled_Eggs = 0.1
Picked_Celery,_Raw = 1.0
Picked_Kielbasa,Prk = 1.0
Picked_Lettuce,Iceberg,Raw = 1.0
Picked_Oranges = 1.0
Picked_Peanut_Butter = 1.0
Picked_Poached_Eggs = 1.0
Picked_Popcorn,Air_Popped = 1.0
Picked_Scrambled_Eggs = 1.0
```

Results We also have a solution now with the following constraints: - Minimum of 1/10 serving is chosen - Celery and Frozen Brocolli are not both chosen at the same time. - 3 kinds of meat/poultry etc. are chosen.

Still does not look appetizing but the program has done it's job as the solution is both feasible and is the optimal solution.