DietOptimizationNotebook

March 30, 2019

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!)

In [1]:
```
#Begin by importing packages
from pulp import *
import numpy as np
import pandas as pd
```

In [2]:
```
#Import the data
data = pd.read_excel('diet.xls')

#Small file so visually inspected to find where the minimums and maximums were defined
mins = data.iloc[65, 3:]
maxes = data.iloc[66, 3:]

#Select only the subset with the foods and associated values for data
data = data.iloc[:64, :]

#Grab the column names
columns = list(data.columns)

#Confirm the bottom of the dataframe is still valid
data.tail()
```

Out[2]:
```
                    Foods  Price/ Serving   Serving Size  Calories  \
59        Neweng Clamchwd            0.75  1 C (8 Fl Oz)     175.7
60            Tomato Soup            0.39  1 C (8 Fl Oz)     170.7
61   New E Clamchwd,W/Mlk            0.99  1 C (8 Fl Oz)     163.7
62   Crm Mshrm Soup,W/Mlk            0.65  1 C (8 Fl Oz)     203.4
63   Beanbacn Soup,W/Watr            0.67  1 C (8 Fl Oz)     172.0

    Cholesterol mg  Total_Fat g  Sodium mg  Carbohydrates g  Dietary_Fiber g  \
59            10.0          5.0     1864.9             21.8              1.5
60             0.0          3.8     1744.4             33.2              1.0
61            22.3          6.6      992.0             16.6              1.5
62            19.8         13.6     1076.3             15.0              0.5
```

```
63            2.5        5.9     951.3           22.8            8.6

    Protein g  Vit_A IU  Vit_C IU  Calcium mg  Iron mg
59       10.9      20.1       4.8        82.8      2.8
60        4.1    1393.0     133.0        27.6      3.5
61        9.5     163.7       3.5       186.0      1.5
62        6.1     153.8       2.2       178.6      0.6
63        7.9     888.0       1.5        81.0      2.0
```

In [3]:
```
#Define the problem to be solved, and if it is to be min/max optimized
problem = LpProblem('Simple Diet Problem', LpMinimize)
```

In [4]:
```
#Make a vector of variables containing a variable for each food
x = [0]*len(data)
for i, row in data.iterrows():
    x[i] = LpVariable(row['Foods'], 0, None, LpContinuous)
```

In [5]:
```
#Add the objective function by multiplying each variable by its associated cost,
#and summing them all up
problem += sum(data['Price/ Serving']*x)
```

In [6]:
```
#Add the constraints (amounts < 0  were addressed in variable definitions)
for nutrient in columns[3:]:
    problem += sum(x*data[nutrient]) >= mins[nutrient]
    problem += sum(x*data[nutrient]) <= maxes[nutrient]
```

In [7]:
```
#Write the output file
problem.writeLP("SimpleDiet.lp")
```

In [8]:
```
#Solve the problem
problem.solve()
pulp.LpStatus[problem.status]
```

Out[8]: 'Optimal'

In [9]:
```
#Print out any food with a solved value greater than 0
for v in problem.variables():
    if v.varValue > 0.0:
        print(v.name, '=', v.varValue)
```

```
Celery,_Raw = 52.64371
Frozen_Broccoli = 0.25960653
Lettuce,Iceberg,Raw = 63.988506
Oranges = 2.2929389
Poached_Eggs = 0.14184397
Popcorn,Air_Popped = 13.869322
```

In [10]:
```
#The optimised cost value can be printed as well
print("\nTotal Cost of Ingredients per day = $",round(value(problem.objective),2))
```

```
Total Cost of Ingredients per day = $ 4.34
```

```
In [11]: #Also just for fun, if we do the same process but insist on having integer
         #values of servings, the results are almost identical
         #but there is a kiwi thrown in and a very small price increase

         #Define the problem to be solved, and if it is to be min/max optimized
         problem = LpProblem('Simple Diet Problem', LpMinimize)

         #Make a vector of variables containing a variable for each food
         x = [0]*len(data)
         for i, row in data.iterrows():
             x[i] = LpVariable(row['Foods'], 0, None, LpInteger)

         #Add the objective function by multiplying each variable by its associated cost,
         #and summing them all up
         problem += sum(data['Price/ Serving']*x)

         #Add the constraints (amounts < 0  were addressed in variable definitions)
         for nutrient in columns[3:]:
             problem += sum(x*data[nutrient]) >= mins[nutrient]
             problem += sum(x*data[nutrient]) <= maxes[nutrient]

         #Write the output file
         problem.writeLP("SimpleDiet.lp")

         #Solve the problem
         problem.solve()
         print(pulp.LpStatus[problem.status], '\n')

         #Print out any food with a solved value greater than 0
         print('Diet consists of: \n')
         for v in problem.variables():
             if v.varValue > 0.0:
                 print(v.name, '=', v.varValue)

         #The optimised cost value can be printed as well
         print("\nTotal Cost of Ingredients per day = $",round(value(problem.objective),2))
```

```
Optimal

Diet consists of:

Celery,_Raw = 41.0
Kiwifruit,Raw,Fresh = 1.0
Lettuce,Iceberg,Raw = 91.0
```

```
Oranges = 2.0
Poached_Eggs = 1.0
Popcorn,Air_Popped = 14.0

Total Cost of Ingredients per day = $ 4.89
```

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.)

In [12]:
```
#Define the upper/lower limits to be used for indicator variables
MIN = 0
MAX = 1000

#Define the problem to be solved, and if it is to be min/max optimized
a_problem = LpProblem('Simple Diet Problem - 2a', LpMinimize)
```

In [13]:
```
#Make a vector of variables containing a variable for each food
x = [0]*len(data)
for i, row in data.iterrows():
    x[i] = LpVariable(row['Foods'], 0, None, LpContinuous)
```

In [14]:
```
#And the binary variable for if they are included
incl = LpVariable.dicts('Included', x, 0, 1, LpInteger)
```

In [15]:
```
#Add the objective function by multiplying each variable by its associated cost,
#and summing them all up
a_problem += sum(data['Price/ Serving']*x)
```

In [17]:
```
#Add the constraints for minimum and maximum nutritional values
#(amounts < 0  were addressed in variable definitions)
for nutrient in columns[3:]:
    a_problem += sum(x*data[nutrient]) >= mins[nutrient]
    a_problem += sum(x*data[nutrient]) <= maxes[nutrient]

#Add a constraint so the binary variables actually work
for i,food in enumerate(x):
    a_problem += x[i] <= MAX*incl[x[i]]
    a_problem += x[i] >= MIN*incl[x[i]]

#Constrain the amount of food to be at least 0.1 serving if it is included,
#if it is not then incl[food]*0.1 returns 0
for i,food in enumerate(x):
    a_problem += x[i] >= incl[x[i]]*(0.1)
```

```
In [18]: #Write the output file
         a_problem.writeLP("SimpleDiet_A.lp")

         #Solve the problem
         a_problem.solve()

         #Print result
         print('Result is:', pulp.LpStatus[a_problem.status], '\n')
         #Print out any food with a solved value greater than 0
         print('Diet consists of: \n')
         for v in a_problem.variables():
             if v.varValue > 0.0:
                 print(v.name, '=', v.varValue)

         #The optimised cost value can be printed as well
         print("\nTotal Cost of Ingredients per day = $",round(value(a_problem.objective),2))
```

```
Result is: Optimal

Diet consists of:

Celery,_Raw = 52.64371
Frozen_Broccoli = 0.25960653
Included_Celery,_Raw = 1.0
Included_Frozen_Broccoli = 1.0
Included_Lettuce,Iceberg,Raw = 1.0
Included_Oranges = 1.0
Included_Poached_Eggs = 1.0
Included_Popcorn,Air_Popped = 1.0
Lettuce,Iceberg,Raw = 63.988506
Oranges = 2.2929389
Poached_Eggs = 0.14184397
Popcorn,Air_Popped = 13.869322

Total Cost of Ingredients per day = $ 4.34
```

No change as a result of this constraint, but I would not expect there to be. Our original solution already had all amounts over 0.1 serving to begin with.

b. Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.

This is easy since we already have made the included/not binary variable. Going to add this to the same code from above.

```
In [19]: b_problem = LpProblem('Simple Diet Problem - 2b', LpMinimize)

         #Make a vector of variables containing a variable for each food
         x = [0]*len(data)
```

```
for i, row in data.iterrows():
    x[i] = LpVariable(row['Foods'], 0, None, LpContinuous)

#And the binary variable for if they are included
incl = LpVariable.dicts('Included', x, 0, 1, LpInteger)

#Add the objective function by multiplying each variable by its associated cost,
#and summing them all up
b_problem += sum(data['Price/ Serving']*x), 'Minimize the cost'


#Add the constraints for minimum and maximum nutritional values
#(amounts < 0  were addressed in variable definitions)
for nutrient in columns[3:]:
    b_problem += sum(x*data[nutrient]) >= mins[nutrient]
    b_problem += sum(x*data[nutrient]) <= maxes[nutrient]

#Add a constraint so the binary variables actually work
for i,food in enumerate(x):
    b_problem += x[i] <= MAX*incl[x[i]]
    b_problem += x[i] >= MIN*incl[x[i]]

#Constrain the amount of food to be at least 0.1 serving if it is included,
#if it is not then incl[food]*0.1 returns 0
for i,food in enumerate(x):
    b_problem += x[i] >= incl[x[i]]*(0.1)

#Constrain broccoli or celery
b_problem += incl[x[0]] + incl[x[2]] <= 1

#Write the output file
b_problem.writeLP("SimpleDiet_B.lp")

#Solve the problem
b_problem.solve()

#Print result
print('Result is:', pulp.LpStatus[b_problem.status], '\n')

#Print out any food with a solved value greater than 0 which was included
print('Diet consists of: \n')
for v in range(0, len(x)):
    if x[v].varValue > 0.0:
        print(x[v], '=', x[v].varValue)

print('\nIncluded foods:\n')
for i in range(0, len(incl)):
    if incl[x[i]].varValue == 1:
```

```
            print(incl[x[i]])


    #The optimised cost value can be printed as well
    print("\nTotal Cost of Ingredients per day = $",round(value(b_problem.objective),2))
```

```
Result is: Optimal

Diet consists of:

Celery,_Raw = 43.154119
Lettuce,Iceberg,Raw = 80.919121
Oranges = 3.0765161
Poached_Eggs = 0.14184397
Peanut_Butter = 2.0464575
Popcorn,Air_Popped = 13.181772

Included foods:

Included_Celery,_Raw
Included_Lettuce,Iceberg,Raw
Included_Oranges
Included_Poached_Eggs
Included_Peanut_Butter
Included_Popcorn,Air_Popped

Total Cost of Ingredients per day = $ 4.49
```

So we end up with celery instead of the broccoli, increasing the amount of lettuce and oranges while reducing celery. The cost also rose slightly due to this new constraint.

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!]

So to begin the problem we should look over the foods and decide which ones are considered proteins, and which ones are not.

```
In [20]: data['Foods'][0:30]
```

```
Out[20]: 0           Frozen Broccoli
         1               Carrots,Raw
         2              Celery, Raw
         3               Frozen Corn
         4       Lettuce,Iceberg,Raw
         5      Peppers, Sweet, Raw
         6          Potatoes, Baked
```

```
7                            Tofu
8                 Roasted Chicken
9              Spaghetti W/ Sauce
10             Tomato,Red,Ripe,Raw
11                Apple,Raw,W/Skin
12                          Banana
13                          Grapes
14             Kiwifruit,Raw,Fresh
15                         Oranges
16                          Bagels
17                     Wheat Bread
18                     White Bread
19                 Oatmeal Cookies
20                       Apple Pie
21          Chocolate Chip Cookies
22                  Butter,Regular
23                  Cheddar Cheese
24             3.3% Fat,Whole Milk
25                  2% Lowfat Milk
26                       Skim Milk
27                    Poached Eggs
28                  Scrambled Eggs
29                  Bologna,Turkey
Name: Foods, dtype: object
```

```
In [21]: data['Foods'][30:]
```

```
Out[21]: 30         Frankfurter, Beef
         31    Ham,Sliced,Extralean
         32              Kielbasa,Prk
         33              Cap'N Crunch
         34                  Cheerios
         35     Corn Flks, Kellogg'S
         36      Raisin Brn, Kellg'S
         37             Rice Krispies
         38                 Special K
         39                   Oatmeal
         40          Malt-O-Meal,Choc
         41         Pizza W/Pepperoni
         42                      Taco
         43     Hamburger W/Toppings
         44            Hotdog, Plain
         45                  Couscous
         46                White Rice
         47              Macaroni,Ckd
         48             Peanut Butter
         49                      Pork
         50           Sardines in Oil
```

```
51      White Tuna in Water
52       Popcorn,Air-Popped
53     Potato Chips,Bbqflvr
54                 Pretzels
55            Tortilla Chip
56          Chicknoodl Soup
57         Splt Pea&Hamsoup
58           Vegetbeef Soup
59          Neweng Clamchwd
60              Tomato Soup
61     New E Clamchwd,W/Mlk
62     Crm Mshrm Soup,W/Mlk
63     Beanbacn Soup,W/Watr
Name: Foods, dtype: object
```

In [22]:
```
#Looking over the entire frame, I am going to ignore marginal protein sources
#and drinks (milks)

#From what I can see these are the indexes with large protein sources
data['Foods'].iloc[[7,8,27,28,29,30,31,32,43,44,49,50,51]]
```

Out[22]:
```
7                      Tofu
8           Roasted Chicken
27             Poached Eggs
28           Scrambled Eggs
29           Bologna,Turkey
30        Frankfurter, Beef
31     Ham,Sliced,Extralean
32             Kielbasa,Prk
43     Hamburger W/Toppings
44            Hotdog, Plain
49                     Pork
50          Sardines in Oil
51      White Tuna in Water
Name: Foods, dtype: object
```

In [23]:
```
#Make a new list with 0/1 flags for is protein.
#Since this is a base truth, I'm going to define it outside of the problem
p_inds = [7,8,27,28,29,30,31,32,43,44,49,50,51]

#Start with all 0 values
is_protein = [0]*len(data)

#Write a 1 to all locations which correspond to a protein in the data
for loc in p_inds:
    is_protein[loc] = 1
```

In [24]:
```
c_problem = LpProblem('Simple Diet Problem - 2c', LpMinimize)
```

```python
#Make a vector of variables containing a variable for each food
x = [0]*len(data)
for i, row in data.iterrows():
    x[i] = LpVariable(row['Foods'], 0, None, LpContinuous)

#And the binary variable for if they are included
incl = LpVariable.dicts('Included', x, 0, 1, LpInteger)

#Add the objective function by multiplying each variable by its associated cost,
#and summing them all up
c_problem += sum(data['Price/ Serving']*x), 'Minimize the cost'


#Add the constraints for minimum and maximum nutritional values
#(amounts < 0  were addressed in variable definitions)
for nutrient in columns[3:]:
    c_problem += sum(x*data[nutrient]) >= mins[nutrient]
    c_problem += sum(x*data[nutrient]) <= maxes[nutrient]

#Add a constraint so the binary variables actually work
for i,food in enumerate(x):
    c_problem += x[i] <= MAX*incl[x[i]]
    c_problem += x[i] >= MIN*incl[x[i]]

#Constrain the amount of food to be at least 0.1 serving if it is included,
#if it is not then incl[food]*0.1 returns 0
for i,food in enumerate(x):
    c_problem += x[i] >= incl[x[i]]*(0.1)

#Constrain broccoli or celery
c_problem += incl[x[0]] + incl[x[2]] <= 1

#Require at least 3 protein sources
#This multiplies the included flag with the externally defined protein flag.
#If something is included and a protein, the result will give a 1, otherwise a 0.
#When summing up the list we require it to be >= 3 so 3 sources minimum are required.
c_problem += lpSum([is_protein[i]*incl[x[i]] for i in range(0,len(x))]) >= 3

#Write the output file
c_problem.writeLP("SimpleDiet_C.lp")

#Solve the problem
c_problem.solve()

#Print result
print('Result is:', pulp.LpStatus[c_problem.status], '\n')

#Print out any food with a solved value greater than 0 which was included
```

```
    print('Diet consists of: \n')
    for v in range(0, len(x)):
        if x[v].varValue > 0.0:
            print(x[v], '=', x[v].varValue)
    print('\nIncluded foods:\n')
    for i in range(0, len(incl)):
        if incl[x[i]].varValue == 1:
            print(incl[x[i]])


    #The optimised cost value can be printed as well
    print("\nTotal Cost of Ingredients per day = $", round(value(c_problem.objective),2))
```

```
Result is: Optimal

Diet consists of:

Celery,_Raw = 42.399358
Lettuce,Iceberg,Raw = 82.802586
Oranges = 3.0771841
Poached_Eggs = 0.1
Scrambled_Eggs = 0.1
Kielbasa,Prk = 0.1
Peanut_Butter = 1.9429716
Popcorn,Air_Popped = 13.223294

Included foods:

Included_Celery,_Raw
Included_Lettuce,Iceberg,Raw
Included_Oranges
Included_Poached_Eggs
Included_Scrambled_Eggs
Included_Kielbasa,Prk
Included_Peanut_Butter
Included_Popcorn,Air_Popped

Total Cost of Ingredients per day = $ 4.51
```

So we get our bare minimum of 3 protein sources (Pork Kielbasa, Poached Eggs, Scrambled Eggs) and they were reduced as much as allowed by the constraints (0.1 serving size). This is likely due to the cost of meats being higher than sources like eggs.

```
In [ ]:
```