ISYE6501 Homework 11

Done By: Joel Quek

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.]

Part 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!)

```
from pulp import *
import numpy as np
import pandas as pd
```

```
diet = pd.read_excel('diet.xls')
```

```
diet.shape
```

```
(67, 14)
```

```
pd.set_option('display.max_rows', None)
diet.head()
```

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carb
0	Frozen Broccoli	0.16	10 Oz Pkg	73.8	0.0	0.8	68.2	
1	Carrots,Raw	0.07	1/2 Cup Shredded	23.7	0.0	0.1	19.2	
2	Celery, Raw	0.04	1 Stalk	6.4	0.0	0.1	34.8	

```
min_ = diet.iloc[65, 3:]
max_ = diet.iloc[66, 3:]
```

```
min_
```

```
Calories           1500.0
Cholesterol mg       30.0
Total_Fat g          20.0
Sodium mg           800.0
Carbohydrates g     130.0
Dietary_Fiber g     125.0
Protein g            60.0
Vit_A IU           1000.0
Vit_C IU            400.0
Calcium mg          700.0
Iron mg              10.0
Name: 65, dtype: object
```

```
max_
```

```
Calories            2500.0
Cholesterol mg       240.0
Total_Fat g           70.0
Sodium mg           2000.0
Carbohydrates g      450.0
Dietary_Fiber g      250.0
Protein g            100.0
Vit_A IU           10000.0
Vit_C IU            5000.0
Calcium mg          1500.0
Iron mg               40.0
Name: 66, dtype: object
```

```
max_.index
```

```
Index(['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg',
       'Carbohydrates g', 'Dietary_Fiber g', 'Protein g', 'Vit_A IU',
       'Vit_C IU', 'Calcium mg', 'Iron mg'],
      dtype='object')
```

```
max_['Calories']
```

```
2500.0
```

```
# Remove last 3 rows

diet = diet.iloc[:64,:]
diet.tail()
```

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carbohy
59	Neweng Clamchwd	0.75	1 C (8 Fl Oz)	175.7	10.0	5.0	1864.9	
60	Tomato Soup	0.39	1 C (8 Fl Oz)	170.7	0.0	3.8	1744.4	
61	New E Clamchwd W/Mlk	0.99	1 C (8 Fl Oz)	163.7	22.3	6.6	992.0	

```
# Define Decision Variables

food_vars = LpVariable.dicts("Foods", diet.index, lowBound=0, cat='Continuous') #if cat='Integer' a different output will be given


# Define the objective function

prob = LpProblem("Cheapest_Diet", LpMinimize) # create instance
prob += lpSum([diet.loc[i, 'Price/ Serving'] * food_vars[i] for i in diet.index]) # objective



# Define the constraints

# Nutrient intake constraints
for nutrient in ['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary_Fiber g', 'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron mg']:
    minimum = diet[nutrient].min()
    maximum = diet[nutrient].max()
    prob += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) >= min_[nutrient]
    prob += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) <= max_[nutrient]
'''
# Serving size constraints
for i in diet.index:
    prob += food_vars[i] * diet.loc[i, 'Serving Size'] >= 0.1  # each food item should be at least 0.1 serving
    prob += food_vars[i] * diet.loc[i, 'Serving Size'] <= 100  # each food item should be at most 100 servings
'''
```

```
'\n# Serving size constraints\nfor i in diet.index:\n    prob += food_vars[i] * diet.l
oc[i, 'Serving Size'] >= 0.1  # each food item should be at least 0.1 serving\n    pro
b += food_vars[i] * diet.loc[i, 'Serving Size'] <= 100  # each food item should be at
```

```
# Solve the optimization problem and print the results

prob.solve()
print("Status:", LpStatus[prob.status])
print("Total Cost of Diet = $", value(prob.objective))
for i in diet.index:
```

```
    if food_vars[i].varValue > 0:
        print(diet.loc[i, 'Foods'], ":", food_vars[i].varValue)
```

```
Status: Optimal
Total Cost of Diet = $ 4.337116797399999
Frozen Broccoli : 0.25960653
Celery, Raw : 52.64371
Lettuce,Iceberg,Raw : 63.988506
Oranges : 2.2929389
Poached Eggs : 0.14184397
Popcorn,Air-Popped : 13.869322
```

Part 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!]

Constraint 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.)

```
# Create Binary Variable for Foods
diet['Binary']=1
diet.head()
```

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carb
0	Frozen Broccoli	0.16	10 Oz Pkg	73.8	0.0	0.8	68.2	
1	Carrots,Raw	0.07	1/2 Cup Shredded	23.7	0.0	0.1	19.2	
2	Celery, Raw	0.04	1 Stalk	6.4	0.0	0.1	34.8	

```
foods = list(diet['Foods'])
# foods


# Decision Variables
food_vars = LpVariable.dicts("Foods", diet.index, lowBound=0, cat='Continuous') #if cat='Integer' a different output will be given

prob_a = LpProblem("Cheapest_Diet", LpMinimize)

# Problem Variable
prob_a += lpSum([diet.loc[i, 'Price/ Serving'] * food_vars[i] for i in diet.index])

# Define the constraints

# Nutrient intake constraints
for nutrient in ['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary_Fiber g', 'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron mg']:
    minimum = diet[nutrient].min()
    maximum = diet[nutrient].max()
    prob_a += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) >= min_[nutrient]
    prob_a += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) <= max_[nutrient]

# The 1/10 Constraint Equation
prob_a += lpSum([food_vars[i].varValue for i in diet.index]) >= 0.1

prob_a.solve()
# print("Status:", LpStatus[prob_a.status])
print("Total Cost of Diet = $", value(prob_a.objective))
for i in diet.index:
    if food_vars[i].varValue > 0:
        print(diet.loc[i, 'Foods'], ":", food_vars[i].varValue)
```

```
Total Cost of Diet = $ 4.337116797399999
Frozen Broccoli : 0.25960653
Celery, Raw : 52.64371
Lettuce,Iceberg,Raw : 63.988506
Oranges : 2.2929389
Poached Eggs : 0.14184397
Popcorn,Air-Popped : 13.869322
```

Constraint b

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

```
# Decision Variables
food_vars = LpVariable.dicts("Foods", diet.index, lowBound=0, cat='Continuous') #if cat='Integer' a different output will be given

# Problem Variable
prob_b = LpProblem("Cheapest_Diet", LpMinimize)

prob_b += lpSum([diet.loc[i, 'Price/ Serving'] * food_vars[i] for i in diet.index])
```

```
# Define the constraints

# Nutrient intake constraints
for nutrient in ['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary_Fiber g', 'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron mg']:
    minimum = diet[nutrient].min()
    maximum = diet[nutrient].max()
    prob_b += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) >= min_[nutrient]
    prob_b += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) <= max_[nutrient]
    # prob_b += lpSum([diet.loc[2, 'Binary'] + diet.loc[0, 'Binary']]) <= 1

# Add the constraint that at most one of celery and frozen broccoli can be selected
prob_b += lpSum([diet.loc[2, 'Binary'] + diet.loc[0, 'Binary']]) <= 1 #, "celery_or_broccoli_constraint"

prob_b.solve()
# print("Status:", LpStatus[prob_b.status])
print("Total Cost of Diet = $", value(prob_b.objective))
for i in diet.index:
    if food_vars[i].varValue > 0:
        print(diet.loc[i, 'Foods'], ":", food_vars[i].varValue)
```

```
Total Cost of Diet = $ 1.364863704
Frozen Broccoli : 2.4968789
Popcorn,Air-Popped : 24.134077
```

I can run Constraint B artificially by removing one or both of Broccoli and Celery

```
no_broccoli = diet.drop(0, axis=0)
no_celery = diet.drop(2, axis=0)
```

No Broccoli

```
# Decision Variables
food_vars = LpVariable.dicts("Foods", no_broccoli.index, lowBound=0, cat='Continuous') #if cat='Integer' a different output will be given

# Problem Variable
prob_b = LpProblem("Cheapest_Diet", LpMinimize)

prob_b += lpSum([no_broccoli.loc[i, 'Price/ Serving'] * food_vars[i] for i in no_broccoli.index])

# Define the constraints

# Nutrient intake constraints
for nutrient in ['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary_Fiber g', 'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron mg']:
    minimum = no_broccoli[nutrient].min()
    maximum = no_broccoli[nutrient].max()
    prob_b += lpSum([no_broccoli.loc[i, nutrient] * food_vars[i] for i in no_broccoli.index]) >= min_[nutrient]
    prob_b += lpSum([no_broccoli.loc[i, nutrient] * food_vars[i] for i in no_broccoli.index]) <= max_[nutrient]

prob_b.solve()
# print("Status:", LpStatus[prob_b.status])
print("Total Cost of Diet = $", value(prob_b.objective))
for i in no_broccoli.index:
```

```
    if food_vars[i].varValue > 0:
        print(no_broccoli.loc[i, 'Foods'], ":", food_vars[i].varValue)
```

```
Total Cost of Diet = $ 4.4878950176
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
```

No Celery

```
# Decision Variables
food_vars = LpVariable.dicts("Foods", no_celery.index, lowBound=0, cat='Continuous') #if cat='Integer' a different output will be given

# Problem Variable
prob_b = LpProblem("Cheapest_Diet", LpMinimize)

prob_b += lpSum([no_celery.loc[i, 'Price/ Serving'] * food_vars[i] for i in no_celery.index])

# Define the constraints

# Nutrient intake constraints
for nutrient in ['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary_Fiber g', 'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron mg']:
    minimum = no_celery[nutrient].min()
    maximum = no_celery[nutrient].max()
    prob_b += lpSum([no_celery.loc[i, nutrient] * food_vars[i] for i in no_celery.index]) >= min_[nutrient]
    prob_b += lpSum([no_celery.loc[i, nutrient] * food_vars[i] for i in no_celery.index]) <= max_[nutrient]

prob_b.solve()
# print("Status:", LpStatus[prob_b.status])
print("Total Cost of Diet = $", value(prob_b.objective))
for i in no_celery.index:
    if food_vars[i].varValue > 0:
        print(no_celery.loc[i, 'Foods'], ":", food_vars[i].varValue)
```

```
Total Cost of Diet = $ 21.985209176199998
Lettuce,Iceberg,Raw : 49.408397
Tofu : 3.4688109
Kiwifruit,Raw,Fresh : 38.501363
Poached Eggs : 0.12340829
Peanut Butter : 1.534422
Beanbacn Soup,W/Watr : 1.5596588
```

No Broccoli yields a more optimal solution.

Constraint c

To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected.

```
proteins_ = ['Tofu','Roasted Chicken','Butter,Regular','Cheddar Cheese','3.3% Fat,Whole Milk','2% Lowfat Milk','Skim Milk','Poached Eggs','Scrambled Eggs','Bologna,Turkey','Frankfurter, Beef','Ham,Sli
 # I excluded the soups


proteins_index=[]
for i in proteins_:
  for j in range(0,64):
    if diet.loc[j].at['Foods']== i:
      proteins_index.append(j)
proteins_index
```

```
[7, 8, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 49, 50, 51]
```

```
# Decision Variables
food_vars = LpVariable.dicts("Foods", diet.index, lowBound=0, cat='Continuous') #if cat='Integer' a different output will be given

# Problem Variable
prob_c = LpProblem("Cheapest_Diet", LpMinimize)

prob_c += lpSum([diet.loc[i, 'Price/ Serving'] * food_vars[i] for i in diet.index])

# Define the constraints

# Nutrient intake constraints
for nutrient in ['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary_Fiber g', 'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron mg']:
    minimum = diet[nutrient].min()
    maximum = diet[nutrient].max()
    prob_c += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) >= min_[nutrient]
    prob_c += lpSum([diet.loc[i, nutrient] * food_vars[i] for i in diet.index]) <= max_[nutrient]
    # prob_c += lpSum([diet.loc[i, 'Binary'] for i in proteins_index]) >= 3


# Add the constraint that at least three proteins are to be selected
prob_c += lpSum([diet.loc[i, 'Binary'] for i in proteins_index]) >= 3

prob_c.solve()
# print("Status:", LpStatus[prob_b.status])
print("Total Cost of Diet = $", value(prob_c.objective))
for i in diet.index:
    if food_vars[i].varValue > 0:
        print(diet.loc[i, 'Foods'], ":", food_vars[i].varValue)
```

```
Total Cost of Diet = $ 4.337116797399999
Frozen Broccoli : 0.25960653
Celery, Raw : 52.64371
Lettuce,Iceberg,Raw : 63.988506
Oranges : 2.2929389
Poached Eggs : 0.14184397
Popcorn,Air-Popped : 13.869322
```

0s completed at 12:18 AM