Wk 7 HOMEWORK SOLUTION 15.2 1

Question 15.2 1

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.

Answer

Optimization Problem

- Diet Problems: Satisfy soldiers' nutritional requirements at minimum cost
- Available data:
    - a_{ij} = amount of nutrient j per unit of food i
    - m_j = minimum daily intake of nutrient j
    - M_j = maximum daily intake of nutrient j
    - c_i = per unit cost of food i

Variables should be use:

x_i = amount of food i in diet
y_i = binary varialbe indicating if food i is used

Constraints

$\sum_i a_{ij} x_i \geq mj$ - The total amount of food should meet the total min daily intake

$\sum_i a_{ij} x_i \leq Mj$ - The total amount of food should meet less than or equal to the total max daily intake

$x_i \geq 0$ - Make sure the amount of food is great or equal to zero to avoid system assign negative value

Objective Function

$Minimize \sum_i c_i x_j$ - minimize the totoal cost of the food selected

(as office hour suggetsed as equivalent: max negative) $Maximize - \sum_i c_i x_j$

Examples: Binary Variables   (note from Office hour)

If you want to eat hot dogs, you at least need to eat 2

$x_{hotdog} \geq 2y_{hotdog}$ - if we choose to eat, the qty should be 2

$x_{hotdog} \leq 1000000y_{hotdog}$ - if we choose not to eat, x should be zero

Examples: Binary Variables (note from Office hour)

You can eat poached eggs or scrambled eggs, but not both:

$y_{poached} + y_{scrambled} \leq 1$

You have to eat at least one portion of apples in any form:

$y_{rawapple} + y_{applepie} \geq 1$

Python Solution

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

Read File

When opened in Excel, the contraints are at the last 3 rows.
Read all data into pandas DataFrame then split the tail to create contraints

```
In [2]: # Read excel data into pandas
        df_diet = pd.read_excel('diet_large.xls', skiprows=1)
        df_diet.head()
```

Out[2]:

	Long_Desc	Protein	Carbohydrate, by difference	Energy	Water	Energy.1	Calcium, Ca	Iron, Fe	Magnesiu
0	Butter, salted	0.85	0.06	717	15.87	3000.0	24	0.02	2
1	Butter, whipped, with salt	0.85	0.06	717	15.87	2999.0	24	0.16	2
2	Butter oil, anhydrous	0.28	0	876	0.24	3665.0	4	0	0
3	Cheese, blue	21.4	2.34	353	42.41	1477.0	528	0.31	23
4	Cheese, brick	23.24	2.79	371	41.11	1552.0	674	0.43	24

5 rows × 31 columns

In [3]:
```
df_diet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7150 entries, 0 to 7149
Data columns (total 31 columns):
Long_Desc                         7146 non-null object
Protein                           7149 non-null object
Carbohydrate, by difference       7149 non-null object
Energy                            7149 non-null object
Water                             7145 non-null object
Energy.1                          7148 non-null float64
Calcium, Ca                       6986 non-null object
Iron, Fe                          7019 non-null object
Magnesium, Mg                     6635 non-null object
Phosphorus, P                     6736 non-null object
Potassium, K                      6791 non-null object
Sodium, Na                        7066 non-null object
Zinc, Zn                          6610 non-null object
Copper, Cu                        6548 non-null object
Manganese, Mn                     5735 non-null object
Selenium, Se                      5885 non-null object
Vitamin A, RAE                    6199 non-null object
Vitamin E (alpha-tocopherol)      3998 non-null object
Vitamin D                         481 non-null object
Vitamin C, total ascorbic acid    6815 non-null object
Thiamin                           6703 non-null object
Riboflavin                        6705 non-null object
Niacin                            6700 non-null object
Pantothenic acid                  5909 non-null object
Vitamin B-6                       6467 non-null object
Folate, total                     6450 non-null object
Vitamin B-12                      6438 non-null object
Vitamin K (phylloquinone)         3647 non-null object
Cholesterol                       6902 non-null float64
Fatty acids, total trans          481 non-null float64
Fatty acids, total saturated      6852 non-null float64
dtypes: float64(4), object(27)
memory usage: 1.7+ MB
```

In [4]:

```
# Show the contraints row at tail
df_diet.tail()
```

Out[4]:

	Long_Desc	Protein	Carbohydrate, by difference	Energy	Water	Energy.1	Calcium, Ca	Iron, Fe
7145	Turtle, green, raw	19.8	0	89	78.5	372.0	118	1.4
7146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7147	NaN	56	130	2400	3700	2400.0	1000	8
7148	NaN	g/d	g/d	kcal	g	NaN	mg/d	mg/d
7149	NaN	1000000	1000000	1000000	1000000	1000000.0	2500	45

5 rows × 31 columns

From above, the data item is from 0 to 7145.
The min is row 7147, max is row 7149

In [13]:

```
# Split df_diet to item and contraints
# For those NaN value, fill with 0
df_diet_item = df_diet.iloc[0:7146].fillna(0)
df_diet_min = df_diet.iloc[7147,1:].dropna()
df_diet_max = df_diet.iloc[7149,1:].dropna()
```

In [14]:

```
df_diet_item.tail()
```

Out[14]:

	Long_Desc	Protein	Carbohydrate, by difference	Energy	Water	Energy.1	Calcium, Ca	Iron, Fe	Magn
7141	Frog legs, raw	16.4	0.0	73	81.9	305.0	18	1.5	20
7142	Fish, mackerel, salted	18.5	0.0	305	43.0	1276.0	66	1.4	60
7143	Mollusks, scallop, (bay and sea), cooked, steamed	23.2	0.0	112	73.1	469.0	115	3.0	55
7144	Mollusks, snail, raw	16.1	2.0	90	79.2	377.0	10	3.5	250
7145	Turtle, green, raw	19.8	0.0	89	78.5	372.0	118	1.4	20

5 rows × 31 columns

By checking constraint lists, it can be seen the last 3 nutrients have no constraints. It is safe to remove those 3 constraints from the constraints list

In [15]:

```
# Convert Pandas series to dictionary
df_diet_min = df_diet_min.to_dict()
df_diet_max = df_diet_max.to_dict()
```

In [16]:

```
# Show all the nutrients, later will call those dictionaries by nutrient
s
nutrients = df_diet_item.columns.tolist()[1:]
nutrients
```

Out[16]:

```
['Protein',
 'Carbohydrate, by difference',
 'Energy',
 'Water',
 'Energy.1',
 'Calcium, Ca',
 'Iron, Fe',
 'Magnesium, Mg',
 'Phosphorus, P',
 'Potassium, K',
 'Sodium, Na',
 'Zinc, Zn',
 'Copper, Cu',
 'Manganese, Mn',
 'Selenium, Se',
 'Vitamin A, RAE',
 'Vitamin E (alpha-tocopherol)',
 'Vitamin D',
 'Vitamin C, total ascorbic acid',
 'Thiamin',
 'Riboflavin',
 'Niacin',
 'Pantothenic acid',
 'Vitamin B-6',
 'Folate, total',
 'Vitamin B-12',
 'Vitamin K (phylloquinone)',
 'Cholesterol',
 'Fatty acids, total trans',
 'Fatty acids, total saturated']
```

Solution with Pulp

In [18]:

```
# Create problem - as Diet Problem, object as Minimize
diet_problem = LpProblem('Diet Problem', LpMinimize)
```

Create Variables

Set Variables as food long description

In [19]:

```
# use the first column food names as variables. variables should be no l
ess than 0
food_list = df_diet_item['Long_Desc'].tolist()
amount_Vars = LpVariable.dict('food', food_list, 0)
```

Create Objective Function

The goal is to minimize cholesterol intake

In [20]:
```
# Create dictionary of Cholesterol of each food
Chol = dict(zip(df_diet_item['Long_Desc'],df_diet_item['Cholesterol']))
```

In [21]:
```
# Objective function to calculate the total cholesterol intake
diet_problem += lpSum([amount_Vars[i]*Chol[i] for i in food_list]), 'Total Cholesterol'
```

Create Constraints

Use the value in df_diet_min and df_diet_max

In [24]:

```
# Loop through each nutrient in nutrient list
for item in nutrients:
    # Create dictionary of food name and nutrient
    nutri_dict = dict(zip(df_diet_item['Long_Desc'],df_diet_item[item]))
    # Add min and max constraints when the constraints is not null
    if item in df_diet_min:
        print('Min '+item)
        diet_problem += lpSum([amount_Vars[i]*nutri_dict[i] for i in food_list]) >= df_diet_min[item],'min '+item
    if item in df_diet_max:
        print('Max '+item)
        diet_problem += lpSum([amount_Vars[i]*nutri_dict[i] for i in food_list]) <= df_diet_max[item],'max '+item
```

```
Min Protein
Max Protein
Min Carbohydrate, by difference
Max Carbohydrate, by difference
Min Energy
Max Energy
Min Water
Max Water
Min Energy.1
Max Energy.1
Min Calcium, Ca
Max Calcium, Ca
Min Iron, Fe
Max Iron, Fe
Min Magnesium, Mg
Max Magnesium, Mg
Min Phosphorus, P
Max Phosphorus, P
Min Potassium, K
Max Potassium, K
Min Sodium, Na
Max Sodium, Na
Min Zinc, Zn
Max Zinc, Zn
Min Copper, Cu
Max Copper, Cu
Min Manganese, Mn
Max Manganese, Mn
Min Selenium, Se
Max Selenium, Se
Min Vitamin A, RAE
Max Vitamin A, RAE
Min Vitamin E (alpha-tocopherol)
Max Vitamin E (alpha-tocopherol)
Min Vitamin D
Max Vitamin D
Min Vitamin C, total ascorbic acid
Max Vitamin C, total ascorbic acid
Min Thiamin
Max Thiamin
Min Riboflavin
Max Riboflavin
Min Niacin
Max Niacin
Min Pantothenic acid
Max Pantothenic acid
Min Vitamin B-6
Max Vitamin B-6
Min Folate, total
Max Folate, total
Min Vitamin B-12
Max Vitamin B-12
Min Vitamin K (phylloquinone)
Max Vitamin K (phylloquinone)
```

Solve Problem

```
In [25]: diet_problem.solve()

Out[25]: 1
```

```
In [41]: varDictionary = {}
         for v in diet_problem.variables():
             varDictionary[v.name]=v.varValue
```

```
In [37]: var_final = dict(filter(lambda elem: elem[1]>0,varDictionary.items()))
         var_final

Out[37]: {'food_Beans,_adzuki,_mature_seeds,_raw': 0.22022038,
          'food_Cocoa_mix,_no_sugar_added,_powder': 0.84893381,
          'food_Egg,_white,_dried,_flakes,_glucose_reduced': 0.039094142,
          'food_Infant_formula,_MEAD_JOHNSON,_LOFENALAC,_with_iron,_powder,_no
         t': 0.68965517,
          'food_Infant_formula,_NESTLE,_GOOD_START_ESSENTIALS__SOY,__with_iro
         n,': 0.78896111,
          'food_Margarine,_industrial,_non_dairy,_cottonseed,_soy_oil_(partial
         l': 0.13576474,
          'food_Noodles,_chinese,_chow_mein': 0.2225939,
          'food_Oil,_vegetable,_sunflower,_linoleic,_(hydrogenated)': 0.8778226
         8,
          'food_Peas,_split,_mature_seeds,_raw': 0.33925794,
          'food_Peppers,_hot_chile,_sun_dried': 0.04727465,
          'food_Seeds,_sunflower_seed_kernels,_oil_roasted,_without_salt': 0.007
         027573,
          'food_Snacks,_potato_chips,_fat_free,_made_with_olestra': 0.11833411,
          'food_Spaghetti,_spinach,_dry': 0.048892766,
          'food_Tomatoes,_sun_dried,_packed_in_oil,_drained': 0.41665491,
          'food_Water,_bottled,_non_carbonated,_CALISTOGA': 9999.5923,
          'food_Wheat,_durum': 0.045432375}
```

```
In [40]: # Calculate the total Cholesterol
         value(diet_problem.objective)

Out[40]: 0.0
```

As Above, we have the values for the variables and we achieve the min goal of Cholesterol as 0