# ISyE 6501 HW11

October 31, 2019

**Group Member**
Jinyu, Huang | jhuang472@gatech.edu | GTID: 903522245
Chengqi, Huang | chengqihuang@gatech.edu | GTID: 903534690
Yuefan, Hu | yuefanhu@gatech.edu | GTID:903543027
Jingyu, Li | alanli@gatech.edu | GTID: 903520148

### 0.0.1 Question 15.1

### 0.0.2 Question 15.2

**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.**

[455]:
```
import numpy as np
import pandas as pd
import math
from pulp import *
```

[456]:
```
d = pd.read_excel(r'C:\Users\alanl\Desktop\Gatech MSiA\Courses\ISyE 6501␣
↪Introduction to Analytics Modeling\hw11\data 15.2\diet.xls')
```

The first step is to split the DataFrame into two parts: nutrition table and constraint values.

[457]:
```
d_c = d.iloc[65:67, :].dropna(axis=1) # Dataframe of constraint values
```

[458]:
```
d_n = d.iloc[:64, :]
```

We build a function to create new variables (i.e. number of units for each kind of food) and a function to add constraints to the model.

[459]:
```
def food(data):
    food_volume = {}
    nrow = data.shape[0]
    for i in range(nrow):
        foodname = data.iloc[i,]['Foods']
        food_volume[foodname] = LpVariable(foodname, 0, None, LpContinuous)
    return food_volume

def nutrition_constraints(problem, nutritionlist, food_volume, data,␣
↪data_constraint):
    for nutrition in nutritionlist:
        minimum = data_constraint[nutrition].iloc[0]
```

```
        maximum = data_constraint[nutrition].iloc[1]
        total = 0
        for foodname in food_volume:
            n = data.loc[data['Foods']==foodname, nutrition].to_list()[0]
            total += n * food_volume[foodname]
        problem += total <= maximum
        problem += total >= minimum
    return problem
```

Then we build the linear optimization model to minimize the total cost while all nutrition constraints are satisfied.

[460]:
```
prob = LpProblem('p1', LpMinimize)
food_volume = food(d_n)
nutritionlist = d_n.columns.to_list()[3:]
cost = 0
for foodname in food_volume:
    cost += food_volume[foodname] * d_n.loc[d_n['Foods']==foodname, 'Price/␣
 ↪Serving'].to_list()[0]
prob += cost
prob = nutrition_constraints(prob, nutritionlist, food_volume, d_n, d_c)
status = prob.solve()
```

[461]:
```
for foodname in food_volume:
    need = value(food_volume[foodname])
    if need > 0:
        print('{} = {:.3f}'.format(foodname,need))

print('\nObj = {:.3f}'.format(value(prob.objective)))
```

```
Frozen Broccoli = 0.260
Celery, Raw = 52.644
Lettuce,Iceberg,Raw = 63.989
Oranges = 2.293
Poached Eggs = 0.142
Popcorn,Air-Popped = 13.869

Obj = 4.337
```

The results show above is that the the optimal solution is nearly: 0.26 units of Frozen Broccoli, 52.64 units of Raw Celery, 63.99 units of Raw Iceberg Lettuce, 2.29 units of Oranges, 0.14 units of Poached Eggs and 13.87 units of Air-Popped Popcorn. And the objective value is 4.34.

**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.

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.

We introduce another group of variables to indicate whether or not a kind of food is selected (binary variable).

[462]:
```
def select(data):
    food_select = {}
    nrow = data.shape[0]
    for i in range(nrow):
        foodname = data.iloc[i,]['Foods']
        food_select[foodname] = LpVariable(foodname+'_s', 0, 1, LpBinary)
    return food_select
```

For new constraints a: If a food is selected, then a minimum of 1/10 serving must be chosen. The best way to connect the binary variable of a food being chosen or not and the continuous varible of the volume of a food here is two contraints combinations:

1). volume = volume * chosen

2). volume >= 1/10 * chosen

These two constraints mean that if chosen = 0, then volume should be 0, and if chosen = 1, then volume should be at least 1/10 serving.

But the problem here is that the first constraint is quadratic. One way to solve it is that we can change the first constraint into volume <= C * chosen, in which C is an extremely large number so that volume will not equal to C in optimal solution (chosen=1). But when chosen = 0, the this new constraint will become volume <= 0. Combining with volume >= 0, we can get volume = 0.

[463]:
```
def minimum_volume(problem, food_volume, food_select):
    for foodname in food_volume:
        problem += food_volume[foodname] <= food_select[foodname] * 10000000
            # ensure if a food is not selected, the volume will be zero
        problem += food_volume[foodname] >= 1/10 * food_select[foodname]
            # ensure if a food is selected, the volume will be at least 1/10␣
 ↪units
    return problem
```

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

[464]:
```
def celery_broccoli(problem, food_select):
    problem += food_select['Frozen Broccoli']+food_select['Celery, Raw'] <= 1
    return problem
```

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

[465]:
```
def meat_kind(problem, food_select, meat_list):
    total = 0
    for meat in meat_list:
        total += food_select[meat]
    problem += total >= 3
    return problem
```

Following is a list of foods that are classified as meat/poultry/fish/eggs. And we build the model.

[466]:
```
meat_list = ['Roasted Chicken', 'Poached Eggs', 'Scrambled Eggs',␣
 ↪'Bologna,Turkey', 'Frankfurter, Beef',
```

```
             'Ham,Sliced,Extralean', 'Kielbasa,Prk', 'Pizza W/Pepperoni',␣
↪'Taco', 'Hamburger W/Toppings',
             'Hotdog, Plain', 'Pork', 'Sardines in Oil', 'White Tuna in Water',␣
↪'Chicknoodl Soup',
             'Splt Pea&Hamsoup', 'Vegetbeef Soup', 'Neweng Clamchwd', 'New E␣
↪Clamchwd,W/Mlk', 'Beanbacn Soup,W/Watr']

prob = LpProblem('p2', LpMinimize)
food_volume = food(d_n)
food_select = select(d_n)
nutritionlist = d_n.columns.to_list()[3:]
cost = 0
for foodname in food_volume:
    cost += food_volume[foodname] * d_n.loc[d_n['Foods']==foodname, 'Price/␣
↪Serving'].to_list()[0]
prob += cost
prob = nutrition_constraints(prob, nutritionlist, food_volume, d_n, d_c)
prob = minimum_volume(prob, food_volume, food_select)
prob = celery_broccoli(prob, food_select)
prob = meat_kind(prob, food_select, meat_list)
status = prob.solve()
```

[467]:
```
for foodname in food_volume:
    choose = value(food_select[foodname])
    if choose > 0:
        print(foodname+' is selected:')
    need = value(food_volume[foodname])
    if need > 0:
        print('volume = {:.3f}\n'.format(need))

print('Obj = {:.3f}'.format(value(prob.objective)))
```

```
Celery, Raw is selected:
volume = 42.399

Lettuce,Iceberg,Raw is selected:
volume = 82.803

Oranges is selected:
volume = 3.077

Poached Eggs is selected:
volume = 0.100

Scrambled Eggs is selected:
volume = 0.100

Kielbasa,Prk is selected:
```

```
volume = 0.100

Peanut Butter is selected:
volume = 1.943

Popcorn,Air-Popped is selected:
volume = 13.223

Obj = 4.513
```

Above is the optimal solution for the problem if we add three constraints. Compared to this problem, the former one can be regarded as a relaxation of this problem, so the minimized objective of this problem is 4.51, which is higher than the former one.

Between Celery and Broccoli, Celery is included while the other one is excluded. And the solution contains three kinds of meat/poultry/fish/eggs (e.g. Poached Eggs, Scrambled Eggs and Kielbasa, Prk), but all of them are at the low bound of the value.

**3. 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).**

[468]:
```
dl = pd.read_excel(r'C:\Users\alanl\Desktop\Gatech MSiA\Courses\ISyE 6501␣
↪Introduction to Analytics Modeling\hw11\data 15.2\diet_large.xls')
```

[469]:
```
dl_c = dl.iloc[[7147,7149], :].dropna(axis=1) # Dataframe of constraint values
```

[470]:
```
dl_n = dl.iloc[:7146, :].drop(['Fatty acids, total trans', 'Fatty acids, total␣
↪saturated'], axis=1)
dl_n.fillna(0, inplace=True)
```

[471]:
```
def food_large(data):
    food_volume = {}
    nrow = data.shape[0]
    for i in range(nrow):
        foodname = data.iloc[i,]['Long_Desc']
        food_volume[foodname] = LpVariable(foodname, 0, None, LpContinuous)
    return food_volume

def nutrition_constraints_large(problem, nutritionlist, food_volume, data,␣
↪data_constraint):
    for nutrition in nutritionlist:
        minimum = data_constraint[nutrition].iloc[0]
        maximum = data_constraint[nutrition].iloc[1]
        total = 0
        for foodname in food_volume:
            n = data.loc[data['Long_Desc']==foodname, nutrition].to_list()[0]
            total += n * food_volume[foodname]
        problem += total <= maximum
        problem += total >= minimum
    return problem
```

```python
prob = LpProblem('p3', LpMinimize)
food_volume = food_large(dl_n)
nutritionlist = dl_n.columns.to_list()[1:-1]
cholesterol = 0
for foodname in food_volume:
    cholesterol += food_volume[foodname] * dl_n.
 ↪loc[dl_n['Long_Desc']==foodname, 'Cholesterol'].to_list()[0]
prob += cholesterol
prob = nutrition_constraints_large(prob, nutritionlist, food_volume, dl_n, dl_c)
status = prob.solve()
```

[472]:
```python
for foodname in food_volume:
    need = value(food_volume[foodname])
    if need > 0:
        print('{} = {:.3f}'.format(foodname,need))

print('\nObj = {:.3f}'.format(value(prob.objective)))
```

```
Spices, mustard seed, yellow = 0.226
Infant formula, WYETH-AYERST, store brand soy, with iron, powde = 0.660
Oil, vegetable, nutmeg butter = 0.762
Soup, clam chowder, manhattan style, dehydrated, dry = 0.061
Water, bottled, non-carbonated, CALISTOGA = 9999.870
Lupins, mature seeds, raw = 0.049
Peanut flour, low fat = 0.256
Soybeans, mature seeds, raw = 0.359
Peanut butter, chunky, vitamin and mineral fortified = 0.230
Leavening agents, baking powder, low-sodium = 0.080
Leavening agents, baking soda = 0.002
Leavening agents, yeast, baker's, active dry = 0.006
Snacks, potato chips, plain, salted = 0.959
Gelatin desserts, dry mix, reduced calorie, with aspartame, add = 0.066
Oil, bearded seal, (oogruk oil) (Alaska Native) = 0.194
Flaxseed oil = 0.102
Celery flakes, dried = 0.091

Obj = 0.000
```

We get a solution with zero cholesterol intake, which seems to be a unattractive diet.