# Hw-7

July 4, 2017

## 1 Homework 7

I will store the data in a pandas dataframe.

```
In [11]: import pandas as pd
         from pulp import *

         data = pd.read_excel('diet.xls')
         reqs = data.iloc[65:67,2:data.shape[1]]
         options = data.iloc[0:64]
         options = options.drop('Serving Size',1)
```

Maintain a list of food.

```
In [12]: foods = []
         for x in range(len(options.iloc[:,1])):
             foods.append(options.iloc[x,0])
```

So this is our key data structure, a dict with the food name as key containing a list of all the nutritional information

```
In [13]: # build dictionary indexed by food name
         nut = {}
         for food,x in zip(foods,range(len(foods))):
             nut[food]=options.iloc[x,1:].values.tolist()
```

Now we get into using the LP library. This is new to me. First we set the LP problem object and then create the variables. The variables here are with the foods as key and the no value initially.

```
In [14]: # set problem and define variables
         prob = LpProblem("The Diet Problem",LpMinimize)
         varis = LpVariable.dicts("Foods",foods,0)
```

Here we have set the the objective funtion, minimise total cost

```
In [15]: # objective function
         prob += lpSum([nut[f][0] * varis[f] for f in foods]), 'Total Cost'
```

Set the constraints from the second df

```
In [16]: # note: the way we have constructed our lists means that calories is indexed at 1 and
         for j in range(1,11):
             min_value="min "+reqs.columns[j]
             max_value="max "+reqs.columns[j]
             prob += lpSum([nut[f][j] * varis[f] for f in foods]) \
                     >= reqs.iloc[0,j], min_value
             prob += lpSum([nut[f][j] * varis[f] for f in foods]) \
                     <= reqs.iloc[1,j], max_value
```

```
In [17]: prob.writeLP("diet_first.lp")
         prob.solve()
         for var in prob.variables():
             if var.varValue >0:
                 print(str(var.varValue)+" units of "+var.name)

         print ("Total Cost of ingredients = ", value(prob.objective))
```

```
52.64371 units of Foods_Celery,_Raw
0.25960653 units of Foods_Frozen_Broccoli
63.988506 units of Foods_Lettuce,Iceberg,Raw
2.2929389 units of Foods_Oranges
0.14184397 units of Foods_Poached_Eggs
13.869322 units of Foods_Popcorn,Air_Popped
Total Cost of ingredients =  4.337116797399999
```

Heyy bingo. This is what the suggested answer in the homework sheet is.

## 1.1 Adding Constraints

Now we will add constraints. With more time I might have written some code that defines protein by the amount of protein in it rather than picking out items, alas I did not.

```
In [18]: #Establish the food select variable to determine if the item was chosen or not
         selected=LpVariable.dicts("selected",foods,0,1,LpBinary)

         #Atleast 1/10 of a server if the food is selected
         for food in foods:
             prob += varis[food] >= 0.1*selected[food]

         for food in foods:
             prob += selected[food] >= varis[food]*0.0000001

         #Either Celery or Broccoli
         prob += selected['Frozen Broccoli'] + selected['Celery, Raw'] <= 1

         #Atleast 3 different Meat, Poultry, Egg, or Fish
         prob += selected['Roasted Chicken'] + \
         selected['Poached Eggs'] + \
         selected['Scrambled Eggs'] + \
         selected['Bologna,Turkey'] + \
         selected['Frankfurter, Beef'] + \
         selected['Ham,Sliced,Extralean'] + \
         selected['Hamburger W/Toppings'] + \
         selected['Hotdog, Plain'] + \
         selected['Pork'] + \
         selected['Sardines in Oil'] + \
         selected['White Tuna in Water'] + \
         selected['Chicknoodl Soup'] + \
         selected['Splt Pea&Hamsoup'] + \
         selected['Vegetbeef Soup'] + \
         selected['Neweng Clamchwd'] + \
         selected['New E Clamchwd,W/Mlk'] + \
         selected['Beanbacn Soup,W/Watr'] >=3
```

```
In [19]: prob.solve()
         for var in prob.variables():
             if var.varValue >0 and "selected" not in var.name:
                 print(str(var.varValue)+" of "+var.name)

         print ("Total Cost = ", value(prob.objective))
```

```
0.1 of Foods_Bologna,Turkey
42.423026 of Foods_Celery,_Raw
82.673927 of Foods_Lettuce,Iceberg,Raw
3.0856009 of Foods_Oranges
1.9590978 of Foods_Peanut_Butter
0.1 of Foods_Poached_Eggs
13.214473 of Foods_Popcorn,Air_Popped
0.1 of Foods_Scrambled_Eggs
Total Cost =  4.5129554810000005
```

Still a pretty rubbish diet.

## 1.2 Full Problem

So I will not comment through each code snippit but say that I have: * We minimise against cholesterol that is at index 27 * There is an increase in iterations through the loops because there are more features

Othere than that it is the same as problem 1.

```
In [20]: %reset -f
```

```
In [21]: from pulp import *
         import pandas as pd
         raw_large = pd.read_excel('diet_large.xls',header=1)
         raw_large = raw_large.fillna(0)
```

```
In [22]: #pull constraint info out
         reqs = raw_large.iloc[[7147,7149],1:31]

         #trim to df
         options = raw_large.iloc[0:7146,0:31]
         # options.tail(3)
         # reqs

In [23]: foods = []
         for x in range(len(options.iloc[:,0])):
             foods.append(options.iloc[x,0])

In [24]: # build dictionary indexed by food name
         nut = {}
         for food,x in zip(foods,range(len(foods))):
             nut[food]=options.iloc[x,1:].values.tolist()

In [25]: # set problem and define variables
         prob = LpProblem("The Diet (Cholesterol) Problem",LpMinimize)
         varis = LpVariable.dicts("Foods",foods,0)

In [26]: # objective function
         prob += lpSum([nut[f][27] * varis[f] for f in foods]), 'Total Cost'

In [27]: # note: the way the list is we need to reduce the index by one
         for j in range(30):
             min_value="min "+reqs.columns[j]
             max_value="max "+reqs.columns[j]
             prob += lpSum([nut[f][j] * varis[f] for f in foods]) \
                     >= reqs.iloc[0,j], min_value
             prob += lpSum([nut[f][j] * varis[f] for f in foods]) \
                     <= reqs.iloc[1,j], max_value
             #print(j,reqs.columns[j], reqs.iloc[0,j],reqs.iloc[1,j],nut['Butter, salted'][j])

In [28]: prob.writeLP("diet_large.lp")
         prob.solve()
         for var in prob.variables():
             if var.varValue >0:
                 print(str(var.varValue)+" units of "+var.name)

         print ("Total Cholesterol of ingredients = ", value(prob.objective))
```

```
1.1642403 units of Foods_Cereals_ready_to_eat,_NATURE'S_PATH,_OPTIMUM_SLIM
0.7518797 units of Foods_Cereals_ready_to_eat,_composite_character_cereals_(movies,_TV),
0.57206552 units of Foods_Chiton,_leathery,_gumboots_(Alaska_Native)
0.21465428 units of Foods_Egg,_white,_dried
0.45547635 units of Foods_Jew's_ear,_(pepeao),_dried
0.16709335 units of Foods_KRAFT,_Sugar_Free_COUNTRY_TIME_Pink_Lemonade_Mix,_with_vitamin_
0.17539672 units of Foods_Leavening_agents,_cream_of_tartar
```

```
0.27078841 units of Foods_Lettuce,_red_leaf,_raw
0.34185148 units of Foods_Margarine,_regular,_hard,_soybean_(hydrogenated)_and_cottonseed
0.64387872 units of Foods_Oil,_whale,_beluga_(Alaska_Native)
9999.0273 units of Foods_Water,_bottled,_non_carbonated,_CALISTOGA
0.015495182 units of Foods_Whale,_beluga,_meat,_air_dried,_raw_(Alaska_Native)
Total Cholesterol of ingredients =  0.0
```

Delicous. The next steps in improving this code would be removing the determinations of where each is so you can auto-import any xls with the same format e.g. you make the constraints df trim from the length of the originial less 2 etc.