15.2.2

March 28, 2018

In [1]:
```
#Import pulp and Pandas
from pulp import *
import pandas as pd
import xlrd
```

In [ ]:
```
#test Pulp
pulp.pulpTestAll()
```

In [2]:
```
#read in data
data = pd.read_excel('diet.xls',
                     skip_footer=3)
```

In [3]:
```
#convert to list
data = data.values.tolist()
```

In [4]:
```
#read in requirements
requirements = pd.read_excel('diet.xls',
                             skiprows=list(range(1,66)),
                             usecols=[2,3,4,5,6,7,8,9,10,11,12,13],
                             header=0,
                             index_col=0)
```

In [5]:
```
requirements
```

Out[5]:

	Calories	Cholesterol mg	Total_Fat g	Sodium mg	\
Serving Size					
Minimum daily intake	1500	30	20	800	
Maximum daily intake	2500	240	70	2000	

	Carbohydrates g	Dietary_Fiber g	Protein g	Vit_A IU	\
Serving Size					
Minimum daily intake	130	125	60	1000	
Maximum daily intake	450	250	100	10000	

	Vit_C IU	Calcium mg	Iron mg
Serving Size			
Minimum daily intake	400	700	10
Maximum daily intake	5000	1500	40

```
In [6]: # Use .loc to get specific requirements when building model
        # Ex. 'Calories' and 'Minimum daily intake'
        requirements.loc['Minimum daily intake','Calories']

Out[6]: 1500

In [7]: requirements.loc['Maximum daily intake','Calories']

Out[7]: 2500

In [8]: #create list of foods, then dictionaries for each nutrient with food as key and amount

        foods = [x[0] for x in data]
        cost = dict([(x[0], float(x[1])) for x in data])
        calories = dict([(x[0], float(x[3])) for x in data])
        cholesterol = dict([(x[0], float(x[4])) for x in data])
        fat = dict([(x[0], float(x[5])) for x in data])
        sodium = dict([(x[0], float(x[6])) for x in data])
        carbs = dict([(x[0], float(x[7])) for x in data])
        fiber = dict([(x[0], float(x[8])) for x in data])
        protein = dict([(x[0], float(x[9])) for x in data])
        vita = dict([(x[0], float(x[10])) for x in data])
        vitc = dict([(x[0], float(x[11])) for x in data])
        calcium = dict([(x[0], float(x[12])) for x in data])
        iron = dict([(x[0], float(x[13])) for x in data])

In [9]: # Initialize Pulp Optimization Object
        diet = LpProblem('diet', LpMinimize)

In [10]: #Create food variables
         foodvars = LpVariable.dict('Foods', foods, 0)

In [11]: #create binary variables
         chosenvars = LpVariable.dicts("Chosen", foods, 0, 1, 'Binary')

In [12]: #Objective Function
         diet += lpSum([cost[f] * foodvars[f] for f in foods]), 'Total Cost'

In [13]: # add in additional constraints for max and min nutrients
         diet += lpSum([calories[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum
         diet += lpSum([calories[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum

In [14]: diet += lpSum([cholesterol[f] * foodvars[f] for f in foods]) >= requirements.loc['Mini
         diet += lpSum([cholesterol[f] * foodvars[f] for f in foods]) <= requirements.loc['Maxi

In [15]: diet += lpSum([fat[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum dai
         diet += lpSum([fat[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum dai

In [16]: diet += lpSum([sodium[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum
         diet += lpSum([sodium[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum
```

In [17]:
```
diet += lpSum([carbs[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum da
diet += lpSum([carbs[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum da
```

In [18]:
```
diet += lpSum([fiber[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum da
diet += lpSum([fiber[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum da
```

In [19]:
```
diet += lpSum([protein[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum
diet += lpSum([protein[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum
```

In [20]:
```
diet += lpSum([vita[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum dai
diet += lpSum([vita[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum dai
```

In [21]:
```
diet += lpSum([vitc[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum dai
diet += lpSum([vitc[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum dai
```

In [22]:
```
diet += lpSum([calcium[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum
diet += lpSum([calcium[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum
```

In [23]:
```
diet += lpSum([iron[f] * foodvars[f] for f in foods]) >= requirements.loc['Minimum dai
diet += lpSum([iron[f] * foodvars[f] for f in foods]) <= requirements.loc['Maximum dai
```

In [24]:
```
# Add constraint so that at least .1 servings of a food are required if it is selecte
for f in foods:
    diet += foodvars[f] >= .1*chosenvars[f]
# and a constraint for max of any food
for f in foods:
    diet += foodvars[f] <= 999999999*chosenvars[f]
```

In [25]:
```
# Add constraint so that celery and brocolli both aren't selected
diet += chosenvars['Frozen Broccoli'] + chosenvars['Celery, Raw'] <= 1, 'Broc/Celery
```

In [26]:
```
# Add constraint so that at least 3 proteins are selected
diet += chosenvars['Roasted Chicken'] + chosenvars['White Tuna in Water']
+ chosenvars['Poached Eggs'] + chosenvars['Scrambled Eggs'] + chosenvars['Bologna,Turk
+ chosenvars['Frankfurter, Beef'] + chosenvars['Ham,Sliced,Extralean']
+ chosenvars['Kielbasa,Prk'] + chosenvars['Hamburger W/Toppings']
+ chosenvars['Hotdog, Plain'] + chosenvars['Taco'] + chosenvars['Pork']
+ chosenvars['White Tuna in Water'] + chosenvars['Sardines in Oil']
+ chosenvars['Chicknoodl Soup'] + chosenvars['Splt Pea&Hamsoup']
+ chosenvars['Vegetbeef Soup'] + chosenvars['Neweng Clamchwd']
+ chosenvars['Beanbacn Soup,W/Watr'] >= 3, 'Protein Req'
```

In [30]:
```
#Solve and check solution
diet.solve()
LpStatus[diet.status]
```

Out[30]:
```
'Optimal'
```

In [31]:
```
#Print foods and their amounts for an optimal diet
for v in diet.variables():
    if v.varValue>0:
        print (v.name, "=", v.varValue)
```

```
Chosen_Celery,_Raw = 1.0
Chosen_Kielbasa,Prk = 1.0
Chosen_Lettuce,Iceberg,Raw = 1.0
Chosen_Oranges = 1.0
Chosen_Peanut_Butter = 1.0
Chosen_Poached_Eggs = 1.0
Chosen_Popcorn,Air_Popped = 1.0
Chosen_Scrambled_Eggs = 1.0
Foods_Celery,_Raw = 42.399358
Foods_Kielbasa,Prk = 0.1
Foods_Lettuce,Iceberg,Raw = 82.802586
Foods_Oranges = 3.0771841
Foods_Peanut_Butter = 1.9429716
Foods_Poached_Eggs = 0.1
Foods_Popcorn,Air_Popped = 13.223294
Foods_Scrambled_Eggs = 0.1
```