15.2_solution

July 5, 2018

0.0.1 15.2 part 1

This problem gave me more trouble than I care to admit! But I feel good about eventually figuring it out.

In [42]:
```
from pulp import *
import pandas as pd
import numpy as np
import random

# Read data
data=pd.read_excel("dietSummer2018.xls")
data=data[0:64]

#Convert dataframe to list
data=data.values.tolist()

#Gather data values to be used in constraints
foods = [x[0] for x in data]
calories = dict([(x[0], float(x[3])) for x in data])
chole = dict([(x[0], float(x[4])) for x in data])
totalfat = dict([(x[0], float(x[5])) for x in data])
sodium = dict([(x[0], float(x[6])) for x in data])
carb = 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])

#Cost dict for use in objective function
cost = dict([(x[0], float(x[1])) for x in data])


#Define problem
prob = LpProblem('15.2.1', LpMinimize)
```

```
#Variables
chosenVars=LpVariable.dicts("Chosen", foods, 0, 1, cat='Binary')
foodVars=LpVariable.dicts("Serving", foods, 0, cat='Continuous')

#Objective function
prob += lpSum([cost[f] * foodVars[f] for f in foods])

#Constraints

prob += lpSum([calories[f] * foodVars[f] for f in foods]) >= 1500, 'min Calories'
prob += lpSum([calories[f] * foodVars[f] for f in foods]) <= 2500, 'max Calories'

prob += lpSum([chole[f] * foodVars[f] for f in foods]) >= 30, 'min Cholestrol'
prob += lpSum([chole[f] * foodVars[f] for f in foods]) <= 240, 'max Cholestrol'

prob += lpSum([totalfat[f] * foodVars[f] for f in foods]) >= 20, 'min Fat'
prob += lpSum([totalfat[f] * foodVars[f] for f in foods]) <= 70, 'max Fat'

prob += lpSum([sodium[f] * foodVars[f] for f in foods]) >= 800, 'min sodium'
prob += lpSum([sodium[f] * foodVars[f] for f in foods]) <= 2000, 'max sodium'

prob += lpSum([carb[f] * foodVars[f] for f in foods]) >= 130, 'min carb'
prob += lpSum([carb[f] * foodVars[f] for f in foods]) <= 450, 'max carb'

prob += lpSum([fiber[f] * foodVars[f] for f in foods]) >= 125, 'min fiber'
prob += lpSum([fiber[f] * foodVars[f] for f in foods]) <= 250, 'max fiber'

prob += lpSum([protein[f] * foodVars[f] for f in foods]) >= 60, 'min protien'
prob += lpSum([protein[f] * foodVars[f] for f in foods]) <= 100, 'max protein'

prob += lpSum([vitA[f] * foodVars[f] for f in foods]) >= 1000, 'min vitamin A'
prob += lpSum([vitA[f] * foodVars[f] for f in foods]) <= 10000, 'max vitamin A'

prob += lpSum([vitC[f] * foodVars[f] for f in foods]) >= 400, 'min vitamin C'
prob += lpSum([vitC[f] * foodVars[f] for f in foods]) <= 5000, 'max vitamin C'

prob += lpSum([iron[f] * foodVars[f] for f in foods]) >= 10, 'min iron'
prob += lpSum([iron[f] * foodVars[f] for f in foods]) <= 40, 'max iron'

prob += lpSum([calcium[f] * foodVars[f] for f in foods]) >= 700, 'min calcium'
prob += lpSum([calcium[f] * foodVars[f] for f in foods]) <= 1500, 'max calcium'


#Solve the optimization problem
prob.solve()

for var in prob.variables():
```

```
            if var.varValue>0:
                if str(var).find('Chosen'):
                    print(str(var.varValue)+" servings of " + str(var) )
        print("Total cost of food = $%.2f" % value(prob.objective))
```

```
52.64371 servings of Serving_Celery,_Raw
0.25960653 servings of Serving_Frozen_Broccoli
63.988506 servings of Serving_Lettuce,Iceberg,Raw
2.2929389 servings of Serving_Oranges
0.14184397 servings of Serving_Poached_Eggs
13.869322 servings of Serving_Popcorn,Air_Popped
Total cost of food = $4.34
```

0.0.2 Part 2 - Additional constraints

```
In [43]: #part a
         for f in foods:
             prob += foodVars[f] >= 0.1 * chosenVars[f]
             prob += foodVars[f] <= 10000 * chosenVars[f]

         #part b
         prob+=chosenVars['Celery, Raw']+chosenVars['Frozen Broccoli']<=1


         #part c
         protein_foods=['Roasted Chicken','Poached Eggs','Scrambled Eggs','Bologna,Turkey','Fra
                        'Kielbasa,Prk','Taco','Hamburger W/Toppings','Hotdog, Plain','Peanut Bu
                        'White Tuna in Water','Chicknoodl Soup','New E Clamchwd,W/Mlk']
         rand=random.sample(protein_foods, 3)

         prob+=chosenVars[rand[0]]+chosenVars[rand[1]]+chosenVars[rand[2]]>=3

         #Solve the optimization problem again
         prob.solve()

         for var in prob.variables():
             if var.varValue>0:
                 if str(var).find('Chosen'):
                     print(str(var.varValue)+" servings of " + str(var) )
         print("Total cost of food = $%.2f" % value(prob.objective))
```

```
41.58054 servings of Serving_Celery,_Raw
0.1 servings of Serving_Frankfurter,_Beef
84.453184 servings of Serving_Lettuce,Iceberg,Raw
3.083672 servings of Serving_Oranges
1.8727459 servings of Serving_Peanut_Butter
0.11200946 servings of Serving_Poached_Eggs
```

```
13.27105 servings of Serving_Popcorn,Air_Popped
0.1 servings of Serving_White_Tuna_in_Water
Total cost of food = $4.58
```

0.0.3 Part 3 - Personal modifications

It might be rather impractical to include 0.1 hamburger with toppings, or 0.1 poached egg in a ration. Maybe we can fix this by making the servings to be an integer? Let's find out!

```
In [51]: #Define problem
         prob = LpProblem('15.2.1', LpMinimize)


         #Variables
         chosenVars=LpVariable.dicts("Chosen", foods, 0, 1, cat='Binary')
         foodVars=LpVariable.dicts("Serving", foods, 0,100, cat='Integer')

         #Objective function
         prob += lpSum([cost[f] * foodVars[f] for f in foods])

         #Constraints

         prob += lpSum([calories[f] * foodVars[f] for f in foods]) >= 1500, 'min Calories'
         prob += lpSum([calories[f] * foodVars[f] for f in foods]) <= 2500, 'max Calories'

         prob += lpSum([chole[f] * foodVars[f] for f in foods]) >= 30, 'min Cholestrol'
         prob += lpSum([chole[f] * foodVars[f] for f in foods]) <= 240, 'max Cholestrol'

         prob += lpSum([totalfat[f] * foodVars[f] for f in foods]) >= 20, 'min Fat'
         prob += lpSum([totalfat[f] * foodVars[f] for f in foods]) <= 70, 'max Fat'

         prob += lpSum([sodium[f] * foodVars[f] for f in foods]) >= 800, 'min sodium'
         prob += lpSum([sodium[f] * foodVars[f] for f in foods]) <= 2000, 'max sodium'

         prob += lpSum([carb[f] * foodVars[f] for f in foods]) >= 130, 'min carb'
         prob += lpSum([carb[f] * foodVars[f] for f in foods]) <= 450, 'max carb'

         prob += lpSum([fiber[f] * foodVars[f] for f in foods]) >= 125, 'min fiber'
         prob += lpSum([fiber[f] * foodVars[f] for f in foods]) <= 250, 'max fiber'

         prob += lpSum([protein[f] * foodVars[f] for f in foods]) >= 60, 'min protien'
         prob += lpSum([protein[f] * foodVars[f] for f in foods]) <= 100, 'max protein'

         prob += lpSum([vitA[f] * foodVars[f] for f in foods]) >= 1000, 'min vitamin A'
         prob += lpSum([vitA[f] * foodVars[f] for f in foods]) <= 10000, 'max vitamin A'

         prob += lpSum([vitC[f] * foodVars[f] for f in foods]) >= 400, 'min vitamin C'
```

```
        prob += lpSum([vitC[f] * foodVars[f] for f in foods]) <= 5000, 'max vitamin C'

        prob += lpSum([iron[f] * foodVars[f] for f in foods]) >= 10, 'min iron'
        prob += lpSum([iron[f] * foodVars[f] for f in foods]) <= 40, 'max iron'

        prob += lpSum([calcium[f] * foodVars[f] for f in foods]) >= 700, 'min calcium'
        prob += lpSum([calcium[f] * foodVars[f] for f in foods]) <= 1500, 'max calcium'

        #part a
        for f in foods:
            prob += foodVars[f] >= 1*chosenVars[f]
            prob += foodVars[f] <= 100000 * chosenVars[f]

        #part b
        prob+=chosenVars['Celery, Raw']+chosenVars['Frozen Broccoli']<=1


        #part c
        protein_foods=['Roasted Chicken','Poached Eggs','Scrambled Eggs','Bologna,Turkey','Fra
                       'Kielbasa,Prk','Taco','Hamburger W/Toppings','Hotdog, Plain','Peanut Bu
                       'White Tuna in Water','Chicknoodl Soup','New E Clamchwd,W/Mlk']
        rand=random.sample(protein_foods, 3)

        #2 proteins instead of 3 - need less since we get full servings
        prob+=chosenVars[rand[0]]+chosenVars[rand[1]]+chosenVars[rand[2]]>=2

        #Solve the optimization problem
        prob.solve()

        for var in prob.variables():
            if var.varValue>0:
                if str(var).find('Chosen'):
                    print(str(var.varValue)+" servings of " + str(var) )
        print("Total cost of food = $%.2f" % value(prob.objective))
```

```
39.0 servings of Serving_Celery,_Raw
1.0 servings of Serving_Kielbasa,Prk
1.0 servings of Serving_Kiwifruit,Raw,Fresh
91.0 servings of Serving_Lettuce,Iceberg,Raw
2.0 servings of Serving_Oranges
1.0 servings of Serving_Poached_Eggs
14.0 servings of Serving_Popcorn,Air_Popped
1.0 servings of Serving_Tofu
Total cost of food = $5.27
```

A bit more expensive, but I think this makes more sense! :)