

# Preference-Based Daily Diet Optimization

Group 6

Baihan Lin, Daehyun Kim, Xinyuan Liu, Yijun Ma

# Problem

- Find an optimal diet for our group members with maximum happiness
  - Food dataset from Public Health England's governmental website
  - Based on preference of different categories of food
  - Constraints: fulfilling nutrient requirements
- Method: linear programming
- Modeled as a modified knapsack problem

# Background

Related diet problem:

Stifler Diet (1939): wanted to find the amount of each of 77 items of food to be eaten per day in order to at least meet the requirements of dietary allowances suggested by National Research Council with minimal cost.

# Model

## Mathematical Model

Objective:

$$\mathbf{H} = \sum_{i=0}^{2834} p_i x_i$$

$\mathbf{H}$ : happiness score

$p_i$ : each food item's preference rating

$x_i$ : amount of each food item in 100 grams

# Model

Constraints:

$$\sum_{j=0}^{2834} n_j X_j \leq N_{max}$$
$$\sum_{j=0}^{2834} n_j X_j \geq N_{min}$$

$n_j$ : amount of nutrient in 100 grams of each food item  $X_j$

$N_{max} / N_{min}$ : maximum/minimum daily recommended values for each nutrient

Nutrients include: protein (g), fat (g), carb (g), energy (kcal), sat\_fat (g), trans\_fat (g), cholesterol (mg), sodium (mg), potassium (mg), calcium (mg), Vitamin D (mcg), Vitamin E (mg), Vitamin B6 (mg), Vitamin B12 (mcg), and Vitamin C (mg)

# Model

Constraints:

$$\sum_{j=0}^{2834} x_j \leq B_{max} \text{ for all } x_j \text{'s in the group of beverages and alcoholic beverages}$$

where  $B_{max} = 30$  hundred grams ( $\approx 96$  ounces)

$$x_j \leq 30 \quad (j = 0, 1, 2, \dots, 2834)$$

# Model

## **Implementation**

1. Data Arrangement
2. Lpsolve Script Generation
3. Lpsolve Analysis

# Model

## Implementation

### 1. Data Arrangement

Food sub-group codes	Codes	Subject 1	Subject 2	Subject 3	Subject 4	average(n/average)
<b>Cereals and cereal products A</b>	<b>A</b>	<b>6</b>	<b>4</b>	<b>3</b>	<b>4</b>	<b>5</b>
Flours, grains and starches	AA	5	2	3	4	3.5
Sandwiches	AB	9	7	3	2	8
Rice	AC	8	4	3	4	6
Pasta	AD	4	5	3	3	4.5
Pizzas	AE	7	6	2	1	6.5
Breads	AF	3	2	4	5	2.5
Rolls	AG	2	2	4	4	2
Breakfast cereals	AI	1	1	5	4	1.5
Infant cereal foods	AK	1	3	1	3	2
Biscuits	AM	3	4	4	5	3.5
Cakes	AN	5	9	5	6	7
Pastry	AO	1	5	3	4	3.5
Buns and pastries	AP	1	7	2	5	4
Puddings	AS	1	8	5	4	4.5
Savouries	AT	1	3	1	5	2
<b>Milk and milk products B</b>	<b>B</b>	<b>4</b>	<b>4</b>	<b>4</b>	<b>4</b>	<b>4</b>
Cows milk	BA	3	3	4	4	3
Breakfast milk	BAB	3	3	4	5	3.5
Skimmed milk	BAE	3	1	5	4	2
Semi-skimmed milk	BAH	3	1	5	4	2
Whole milk	BAK	4	6	3	5	5

Food Cod	Food Name	Description	Group	Male	Female	Previous	Main data reference	Footnote	Water (g)			TOTAL nitri		PROT		Fat (g)		CarbO		CHO		ENERGY (KJ)		Starch	OLIGO	Total sug		Glucose		Galactose		Fructose		SUCR		MALT		LACTOSE		ALCOH		NSP (g)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
									WATER	TOTAL	nitri	PROT	FAT	CHO	KCAL	KJ	CarbO	CHO	KCAL	KJ	TOTAL	GLUC	TOTAL			GLUC	GALACT	FRUCT	SUCR	MALT	LACT	ALCOH	ENFIB																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
									Water	0.46	2.9	15.2	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625			Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151	625	Tr	0.8	151

```

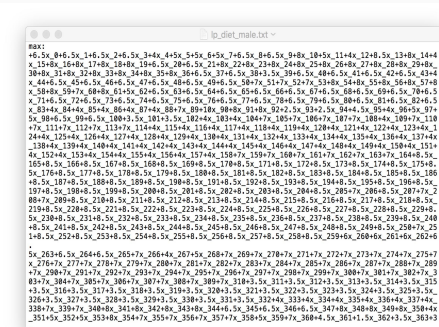
1 #!/usr/local/bin/
2 # Bashin Lin
3 # October 2016
4
5 # The following code was to generate proper data format of
6 # two columns of preference data for male and female
7 # corresponding to the food items, these two columns
8 # are later copied and pasted into excel sheet of all data.
9
10 # Part I: Generate pattern matching arguments
11 # input file: FoodGroup.list, pref.list, prefW.list
12 # output file: pref_ind_switch_final.sh
13
14 # pref: run:
15 # pref_ind_switch.sh
16 # pref_ind_switch_final.sh
17 # new pref.out
18
19 # pref: list pref: run:
20 sed -i -e 's/"/\|/g' pref.list;
21
22 # prefW: list pref: run:
23 sed -i -e 's/"/\|/g' prefW.list;
24
25 paste pref: run pref: run; pref_ind_switch.sh;
26 sed -i -e 's/"/\|/g' pref.out; pref_ind_switch.sh;
27
28 cat pref_ind_switch.sh | tr -d '\011' > pref_ind_switch_final.sh;
29
30 sed -i -e 's/"/\|/g' pref_ind_switch_final.sh;
31 sed -i -e 's/"/\|/g' pref_ind_switch_final.sh;
32
33 # Part II: Generate columns with matched preference data
34 # input file: pref_ind_switch_final.sh
35 # output file: new pref.out
36
37 for j in $(cat FoodGroup.list | do
38   # pref: list pref: run:
39   # prefW: list pref: run:
40   # pref: list pref: run:
41   # prefW: list pref: run:
42   # pref: list pref: run:
43   # prefW: list pref: run:
44   # pref: list pref: run:
45   # prefW: list pref: run:
46   # pref: list pref: run:
47   # prefW: list pref: run:
48   # pref: list pref: run:
49   # prefW: list pref: run:
50   # pref: list pref: run:
51   # prefW: list pref: run:
52   # pref: list pref: run:
53   # prefW: list pref: run:
54   # pref: list pref: run:
55   # prefW: list pref: run:
56   # pref: list pref: run:
57   # prefW: list pref: run:
58   # pref: list pref: run:
59   # prefW: list pref: run:
60   # pref: list pref: run:
61   # prefW: list pref: run:
62   # pref: list pref: run:
63   # prefW: list pref: run:
64   # pref: list pref: run:
65   # prefW: list pref: run:
66   # pref: list pref: run:
67   # prefW: list pref: run:
68   # pref: list pref: run:
69   # prefW: list pref: run:
70   # pref: list pref: run:
71   # prefW: list pref: run:
72   # pref: list pref: run:
73   # prefW: list pref: run:
74   # pref: list pref: run:
75   # prefW: list pref: run:
76   # pref: list pref: run:
77   # prefW: list pref: run:
78   # pref: list pref: run:
79   # prefW: list pref: run:
80   # pref: list pref: run:
81   # prefW: list pref: run:
82   # pref: list pref: run:
83   # prefW: list pref: run:
84   # pref: list pref: run:
85   # prefW: list pref: run:
86   # pref: list pref: run:
87   # prefW: list pref: run:
88   # pref: list pref: run:
89   # prefW: list pref: run:
90   # pref: list pref: run:
91   # prefW: list pref: run:
92   # pref: list pref: run:
93   # prefW: list pref: run:
94   # pref: list pref: run:
95   # prefW: list pref: run:
96   # pref: list pref: run:
97   # prefW: list pref: run:
98   # pref: list pref: run:
99   # prefW: list pref: run:
100  # pref: list pref: run:
101  # prefW: list pref: run:
102  # pref: list pref: run:
103  # prefW: list pref: run:
104  # pref: list pref: run:
105  # prefW: list pref: run:
106  # pref: list pref: run:
107  # prefW: list pref: run:
108  # pref: list pref: run:
109  # prefW: list pref: run:
110  # pref: list pref: run:
111  # prefW: list pref: run:
112  # pref: list pref: run:
113  # prefW: list pref: run:
114  # pref: list pref: run:
115  # prefW: list pref: run:
116  # pref: list pref: run:
117  # prefW: list pref: run:
118  # pref: list pref: run:
119  # prefW: list pref: run:
120  # pref: list pref: run:
121  # prefW: list pref: run:
122  # pref: list pref: run:
123  # prefW: list pref: run:
124  # pref: list pref: run:
125  # prefW: list pref: run:
126  # pref: list pref: run:
127  # prefW: list pref: run:
128  # pref: list pref: run:
129  # prefW: list pref: run:
130  # pref: list pref: run:
131  # prefW: list pref: run:
132  # pref: list pref: run:
133  # prefW: list pref: run:
134  # pref: list pref: run:
135  # prefW: list pref: run:
136  # pref: list pref: run:
137  # prefW: list pref: run:
138  # pref: list pref: run:
139  # prefW: list pref: run:
140  # pref: list pref: run:
141  # prefW: list pref: run:
142  # pref: list pref: run:
143  # prefW: list pref: run:
144  # pref: list pref: run:
145  # prefW: list pref: run:
146  # pref: list pref: run:
147  # prefW: list pref: run:
148  # pref: list pref: run:
149  # prefW: list pref: run:
150  # pref: list pref: run:
151  # prefW: list pref: run:
152  # pref: list pref: run:
153  # prefW: list pref: run:
154  # pref: list pref: run:
155  # prefW: list pref: run:
156  # pref: list pref: run:
157  # prefW: list pref: run:
158  # pref: list pref: run:
159  # prefW: list pref: run:
160  # pref: list pref: run:
161  # prefW: list pref: run:
162  # pref: list pref: run:
163  # prefW: list pref: run:
164  # pref: list pref: run:
165  # prefW: list pref: run:
166  # pref: list pref: run:
167  # prefW: list pref: run:
168  # pref: list pref: run:
169  # prefW: list pref: run:
170  # pref: list pref: run:
171  # prefW: list pref: run:
172  # pref: list pref: run:
173  # prefW: list pref: run:
174  # pref: list pref: run:
175  # prefW: list pref: run:
176  # pref: list pref: run:
177  # prefW: list pref: run:
178  # pref: list pref: run:
179  # prefW: list pref: run:
180  # pref: list pref: run:
181  # prefW: list pref: run:
182  # pref: list pref: run:
183  # prefW: list pref: run:
184  # pref: list pref: run:
185  # prefW: list pref: run:
186  # pref: list pref: run:
187  # prefW: list pref: run:
188  # pref: list pref: run:
189  # prefW: list pref: run:
190  # pref: list pref: run:
191  # prefW: list pref: run:
192  # pref: list pref: run:
193  # prefW: list pref: run:
194  # pref: list pref: run:
195  # prefW: list pref: run:
196  # pref: list pref: run:
197  # prefW: list pref: run:
198  # pref: list pref: run:
199  # prefW: list pref: run:
200  # pref: list pref: run:
201  # prefW: list pref: run:
202  # pref: list pref: run:
203  # prefW: list pref: run:
204  # pref: list pref: run:
205  # prefW: list pref: run:
206  # pref: list pref: run:
207  # prefW: list pref: run:
208  # pref: list pref: run:
209  # prefW: list pref: run:
210  # pref: list pref: run:
211  # prefW: list pref: run:
212  # pref: list pref: run:
213  # prefW: list pref: run:
214  # pref: list pref: run:
215  # prefW: list pref: run:
216  # pref: list pref: run:
217  # prefW: list pref: run:
218  # pref: list pref: run:
219  # prefW: list pref: run:
220  # pref: list pref: run:
221  # prefW: list pref: run:
222  # pref: list pref: run:
223  # prefW: list pref: run:
224  # pref: list pref: run:
225  # prefW: list pref: run:
226  # pref: list pref: run:
227  # prefW: list pref: run:
228  # pref: list pref: run:
229  # prefW: list pref: run:
230  # pref: list pref: run:
231  # prefW: list pref: run:
232  # pref: list pref: run:
233  # prefW: list pref: run:
234  # pref: list pref: run:
235  # prefW: list pref: run:
236  # pref: list pref: run:
237  # prefW: list pref: run:
238  # pref: list pref: run:
239  # prefW: list pref: run:
240  # pref: list pref: run:
241  # prefW: list pref: run:
242  # pref: list pref: run:
243  # prefW: list pref: run:
244  # pref: list pref: run:
245  # prefW: list pref: run:
246  # pref: list pref: run:
247  # prefW: list pref: run:
248  # pref: list pref: run:
249  # prefW: list pref: run:
250  # pref: list pref: run:
251  # prefW: list pref
```



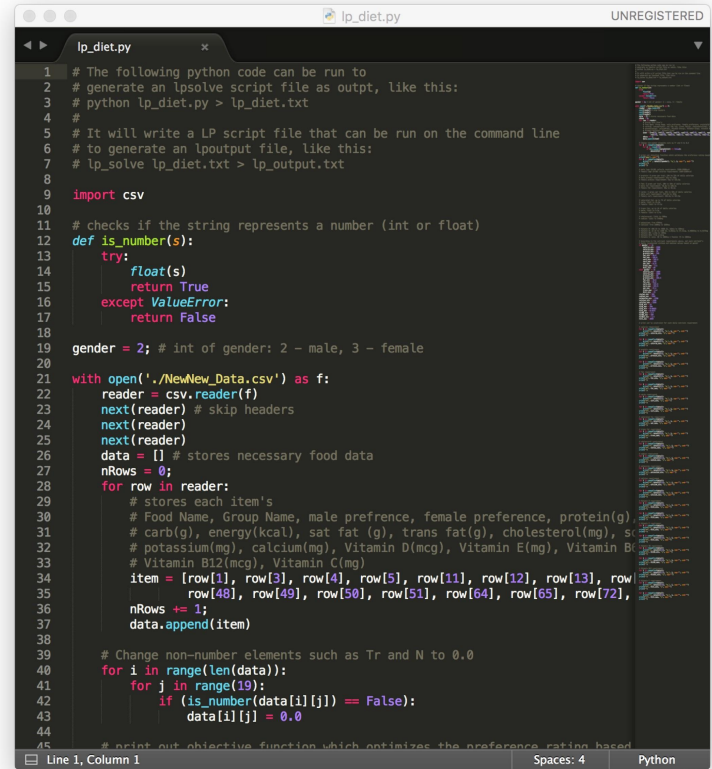
# Model

## Implementation

1. Data Arrangement
2. Lpsolve Script Generation
3. Lpsolve Analysis



```
max:
6.5x 846.5x 145.5x 246.5x 344x 445x 545x 645x 745.5x 846.5x 948x 105x 1144x 1248.5x 1348x 1444
x 1548x 1648x 1748x 1848x 1948.5x 2046.5x 2148x 2248x 2348x 2448x 2548x 2648x 2748x 2848x 2948x
3048x 3148x 3248x 3348x 3448x 3548x 3646.5x 3746.5x 3843.5x 3946.5x 4046.5x 4146.5x 4246.5x 4344
x 4446.5x 4546.5x 4646.5x 4746.5x 4846.5x 4946.5x 5047x 5147x 5247x 5348x 5448x 5548x 5648x 5748
x 5848x 5947x 6048x 6145x 6246.5x 6346.5x 6446.5x 6546.5x 6646.5x 6746.5x 6846.5x 6946.5x 7046.5
x 7146.5x 7246.5x 7346.5x 744x 7546.5x 7646.5x 7746.5x 7846.5x 7946.5x 8048x 8146.5x 8246.5
x 8346.5x 8446.5x 8546.5x 8646.5x 8746.5x 8847x 8948x 9048x 9148x 9246.5x 9346.5x 9446.5x 9546.5x 9646.5x 9746.5x 9846.5x 9946.5x 10046.5x 10146.5x 10246.5x 10346.5x 10446.5x 10546.5x 10646.5x 10746.5x 10846.5x 10946.5x 11046.5x 11146.5x 11246.5x 11346.5x 1144x 11546.5x 11646.5x 11746.5x 11846.5x 11946.5x 12046.5x 12146.5x 12246.5x 12346.5x 1244x 12546.5x 12646.5x 12746.5x 12846.5x 12946.5x 13046.5x 13146.5x 13246.5x 13346.5x 13446.5x 13546.5x 13646.5x 13746.5x 13846.5x 13946.5x 14046.5x 14146.5x 14246.5x 14346.5x 1444x 14546.5x 14646.5x 14746.5x 14846.5x 14946.5x 15046.5x 15146.5x 15246.5x 15346.5x 15446.5x 15546.5x 15646.5x 15746.5x 15846.5x 15946.5x 16046.5x 16146.5x 16246.5x 16346.5x 16446.5x 16546.5x 16646.5x 16746.5x 16846.5x 16946.5x 17046.5x 17146.5x 17246.5x 17346.5x 17446.5x 17546.5x 17646.5x 17746.5x 17846.5x 17946.5x 18046.5x 18146.5x 18246.5x 18346.5x 18446.5x 18546.5x 18646.5x 18746.5x 18846.5x 18946.5x 19046.5x 19146.5x 19246.5x 19346.5x 19446.5x 19546.5x 19646.5x 19746.5x 19846.5x 19946.5x 20046.5x 20146.5x 20246.5x 20346.5x 20446.5x 20546.5x 20646.5x 20746.5x 20846.5x 20946.5x 21046.5x 21146.5x 21246.5x 21346.5x 21446.5x 21546.5x 21646.5x 21746.5x 21846.5x 21946.5x 22046.5x 22146.5x 22246.5x 22346.5x 22446.5x 22546.5x 22646.5x 22746.5x 22846.5x 22946.5x 23046.5x 23146.5x 23246.5x 23346.5x 23446.5x 23546.5x 23646.5x 23746.5x 23846.5x 23946.5x 24046.5x 24146.5x 24246.5x 24346.5x 24446.5x 24546.5x 24646.5x 24746.5x 24846.5x 24946.5x 25046.5x 25146.5x 25246.5x 25346.5x 25446.5x 25546.5x 25646.5x 25746.5x 25846.5x 25946.5x 26046.5x 26146.5x 26246.5x 26346.5x 26446.5x 26546.5x 26646.5x 26746.5x 26846.5x 26946.5x 27046.5x 27146.5x 27246.5x 27346.5x 27446.5x 27546.5x 27646.5x 27746.5x 27846.5x 27946.5x 28046.5x 28146.5x 28246.5x 28346.5x 28446.5x 28546.5x 28646.5x 28746.5x 28846.5x 28946.5x 29046.5x 29146.5x 29246.5x 29346.5x 29446.5x 29546.5x 29646.5x 29746.5x 29846.5x 29946.5x 30046.5x 30146.5x 30246.5x 30346.5x 30446.5x 30546.5x 30646.5x 30746.5x 30846.5x 30946.5x 31046.5x 31146.5x 31246.5x 31346.5x 31446.5x 31546.5x 31646.5x 31746.5x 31846.5x 31946.5x 32046.5x 32146.5x 32246.5x 32346.5x 32446.5x 32546.5x 32646.5x 32746.5x 32846.5x 32946.5x 33046.5x 33146.5x 33246.5x 33346.5x 33446.5x 33546.5x 33646.5x 33746.5x 33846.5x 33946.5x 34046.5x 34146.5x 34246.5x 34346.5x 34446.5x 34546.5x 34646.5x 34746.5x 34846.5x 34946.5x 35046.5x 35146.5x 35246.5x 35346.5x 35446.5x 35546.5x 35646.5x 35746.5x 35846.5x 35946.5x 36046.5x 36146.5x 36246.5x 36346.5x 36446.5x 36546.5x 36646.5x 36746.5x 36846.5x 36946.5x 37046.5x 37146.5x 37246.5x 37346.5x 37446.5x 37546.5x 37646.5x 37746.5x 37846.5x 37946.5x 38046.5x 38146.5x 38246.5x 38346.5x 38446.5x 38546.5x 38646.5x 38746.5x 38846.5x 38946.5x 39046.5x 39146.5x 39246.5x 39346.5x 39446.5x 39546.5x 39646.5x 39746.5x 39846.5x 39946.5x 40046.5x 40146.5x 40246.5x 40346.5x 40446.5x 40546.5x 40646.5x 40746.5x 40846.5x 40946.5x 41046.5x 41146.5x 41246.5x 41346.5x 41446.5x 41546.5x 41646.5x 41746.5x 41846.5x 41946.5x 42046.5x 42146.5x 42246.5x 42346.5x 42446.5x 42546.5x 42646.5x 42746.5x 42846.5x 42946.5x 43046.5x 43146.5x 43246.5x 43346.5x 43446.5x 43546.5x 43646.5x 43746.5x 43846.5x 43946.5x 44046.5x 44146.5x 44246.5x 44346.5x 44446.5x 44546.5x 44646.5x 44746.5x 44846.5x 44946.5x 45046.5x 45146.5x 45246.5x 45346.5x 45446.5x 45546.5x 45646.5x 45746.5x 45846.5x 45946.5x 46046.5x 46146.5x 46246.5x 46346.5x 46446.5x 46546.5x 46646.5x 46746.5x 46846.5x 46946.5x 47046.5x 47146.5x 47246.5x 47346.5x 47446.5x 47546.5x 47646.5x 47746.5x 47846.5x 47946.5x 48046.5x 48146.5x 48246.5x 48346.5x 48446.5x 48546.5x 48646.5x 48746.5x 48846.5x 48946.5x 49046.5x 49146.5x 49246.5x 49346.5x 49446.5x 49546.5x 49646.5x 49746.5x 49846.5x 49946.5x 50046.5x 50146.5x 50246.5x 50346.5x 50446.5x 50546.5x 50646.5x 50746.5x 50846.5x 50946.5x 51046.5x 51146.5x 51246.5x 51346.5x 51446.5x 51546.5x 51646.5x 51746.5x 51846.5x 51946.5x 52046.5x 52146.5x 52246.5x 52346.5x 52446.5x 52546.5x 52646.5x 52746.5x 52846.5x 52946.5x 53046.5x 53146.5x 53246.5x 53346.5x 53446.5x 53546.5x 53646.5x 53746.5x 53846.5x 53946.5x 54046.5x 54146.5x 54246.5x 54346.5x 54446.5x 54546.5x 54646.5x 54746.5x 54846.5x 54946.5x 55046.5x 55146.5x 55246.5x 55346.5x 55446.5x 55546.5x 55646.5x 55746.5x 55846.5x 55946.5x 56046.5x 56146.5x 56246.5x 56346.5x 56446.5x 56546.5x 56646.5x 56746.5x 56846.5x 56946.5x 57046.5x 57146.5x 57246.5x 57346.5x 57446.5x 57546.5x 57646.5x 57746.5x 57846.5x 57946.5x 58046.5x 58146.5x 58246.5x 58346.5x 58446.5x 58546.5x 58646.5x 58746.5x 58846.5x 58946.5x 59046.5x 59146.5x 59246.5x 59346.5x 59446.5x 59546.5x 59646.5x 59746.5x 59846.5x 59946.5x 60046.5x 60146.5x 60246.5x 60346.5x 60446.5x 60546.5x 60646.5x 60746.5x 60846.5x 60946.5x 61046.5x 61146.5x 61246.5x 61346.5x 61446.5x 61546.5x 61646.5x 61746.5x 61846.5x 61946.5x 62046.5x 62146.5x 62246.5x 62346.5x 62446.5x 62546.5x 62646.5x 62746.5x 62846.5x 62946.5x 63046.5x 63146.5x 63246.5x 63346.5x 63446.5x 63546.5x 63646.5x 63746.5x 63846.5x 63946.5x 64046.5x 64146.5x 64246.5x 64346.5x 64446.5x 64546.5x 64646.5x 64746.5x 64846.5x 64946.5x 65046.5x 65146.5x 65246.5x 65346.5x 65446.5x 65546.5x 65646.5x 65746.5x 65846.5x 65946.5x 66046.5x 66146.5x 66246.5x 66346.5x 66446.5x 66546.5x 66646.5x 66746.5x 66846.5x 66946.5x 67046.5x 67146.5x 67246.5x 67346.5x 67446.5x 67546.5x 67646.5x 67746.5x 67846.5x 67946.5x 68046.5x 68146.5x 68246.5x 68346.5x 68446.5x 68546.5x 68646.5x 68746.5x 68846.5x 68946.5x 69046.5x 69146.5x 69246.5x 69346.5x 69446.5x 69546.5x 69646.5x 69746.5x 69846.5x 69946.5x 70046.5x 70146.5x 70246.5x 70346.5x 70446.5x 70546.5x 70646.5x 70746.5x 70846.5x 70946.5x 71046.5x 71146.5x 71246.5x 71346.5x 71446.5x 71546.5x 71646.5x 71746.5x 71846.5x 71946.5x 72046.5x 72146.5x 72246.5x 72346.5x 72446.5x 72546.5x 72646.5x 72746.5x 72846.5x 72946.5x 73046.5x 73146.5x 73246.5x 73346.5x 73446.5x 73546.5x 73646.5x 73746.5x 73846.5x 73946.5x 74046.5x 74146.5x 74246.5x 74346.5x 74446.5x 74546.5x 74646.5x 74746.5x 74846.5x 74946.5x 75046.5x 75146.5x 75246.5x 75346.5x 75446.5x 75546.5x 75646.5x 75746.5x 75846.5x 75946.5x 76046.5x 76146.5x 76246.5x 76346.5x 76446.5x 76546.5x 76646.5x 76746.5x 76846.5x 76946.5x 77046.5x 77146.5x 77246.5x 77346.5x 77446.5x 77546.5x 77646.5x 77746.5x 77846.5x 77946.5x 78046.5x 78146.5x 78246.5x 78346.5x 78446.5x 78546.5x 78646.5x 78746.5x 78846.5x 78946.5x 79046.5x 79146.5x 79246.5x 79346.5x 79446.5x 79546.5x 79646.5x 79746.5x 79846.5x 79946.5x 80046.5x 80146.5x 80246.5x 80346.5x 80446.5x 80546.5x 80646.5x 80746.5x 80846.5x 80946.5x 81046.5x 81146.5x 81246.5x 81346.5x 81446.5x 81546.5x 81646.5x 81746.5x 81846.5x 81946.5x 82046.5x 82146.5x 82246.5x 82346.5x 82446.5x 82546.5x 82646.5x 82746.5x 82846.5x 82946.5x 83046.5x 83146.5x 83246.5x 83346.5x 83446.5x 83546.5x 83646.5x 83746.5x 83846.5x 83946.5x 84046.5x 84146.5x 84246.5x 84346.5x 84446.5x 84546.5x 84646.5x 84746.5x 84846.5x 84946.5x 85046.5x 85146.5x 85246.5x 85346.5x 85446.5x 85546.5x 85646.5x 85746.5x 85846.5x 85946.5x 86046.5x 86146.5x 86246.5x 86346.5x 86446.5x 86546.5x 86646.5x 86746.5x 86846.5x 86946.5x 87046.5x 87146.5x 87246.5x 87346.5x 87446.5x 87546.5x 87646.5x 87746.5x 87846.5x 87946.5x 88046.5x 88146.5x 88246.5x 88346.5x 88446.5x 88546.5x 88646.5x 88746.5x 88846.5x 88946.5x 89046.5x 89146.5x 89246.5x 89346.5x 89446.5x 89546.5x 89646.5x 89746.5x 89846.5x 89946.5x 90046.5x 90146.5x 90246.5x 90346.5x 90446.5x 90546.5x 90646.5x 90746.5x 90846.5x 90946.5x 91046.5x 91146.5x 91246.5x 91346.5x 91446.5x 91546.5x 91646.5x 91746.5x 91846.5x 91946.5x 92046.5x 92146.5x 92246.5x 92346.5x 92446.5x 92546.5x 92646.5x 92746.5x 92846.5x 92946.5x 93046.5x 93146.5x 93246.5x 93346.5x 93446.5x 93546.5x 93646.5x 93746.5x 93846.5x 93946.5x 94046.5x 94146.5x 94246.5x 94346.5x 94446.5x 94546.5x 94646.5x 94746.5x 94846.5x 94946.5x 95046.5x 95146.5x 95246.5x 95346.5x 95446.5x 95546.5x 95646.5x 95746.5x 95846.5x 95946.5x 96046.5x 96146.5x 96246.5x 96346.5x 96446.5x 96546.5x 96646.5x 96746.5x 96846.5x 96946.5x 97046.5x 97146.5x 97246.5x 97346.5x 97446.5x 97546.5x 97646.5x 97746.5x 97846.5x 97946.5x 98046.5x 98146.5x 98246.5x 98346.5x 98446.5x 98546.5x 98646.5x 98746.5x 98846.5x 98946.5x 99046.5x 99146.5x 99246.5x 99346.5x 99446.5x 99546.5x 99646.5x 99746.5x 99846.5x 99946.5x 100046.5x
```



```
lp_diet.py
1 # The following python code can be run to
2 # generate an lpsolve script file as output, like this:
3 # python lp_diet.py > lp_diet.txt
4
5 # It will write a LP script file that can be run on the command line
6 # to generate an lpsolve script file, like this:
7 # lp_solve lp_diet.txt > lp_output.txt
8
9 import csv
10
11 # checks if the string represents a number (int or float)
12 def is_number(s):
13     try:
14         float(s)
15         return True
16     except ValueError:
17         return False
18
19 gender = 2; # int of gender: 2 - male, 3 - female
20
21 with open('NewData.csv') as f:
22     reader = csv.reader(f)
23     next(reader) # skip headers
24     next(reader)
25     next(reader)
26     data = [] # stores necessary food data
27     nRows = 0;
28     for row in reader:
29         # stores each item's
30         # Food Name, Group Name, male preference, female preference, protein(g),
31         # carb(g), energy(kcal), sat fat (g), trans fat(g), cholesterol(mg), s
32         # potassium(mg), calcium(mg), Vitamin D(mcg), Vitamin E(mg), Vitamin B
33         # Vitamin B12(mcg), Vitamin C(mg)
34         item = [row[1], row[3], row[4], row[5], row[11], row[12], row[13], row
35         row[48], row[49], row[50], row[51], row[64], row[65], row[72],
36         nRows += 1;
37         data.append(item)
38
39 # Change non-number elements such as Tr and N to 0.0
40 for i in range(len(data)):
41     for j in range(19):
42         if (is_number(data[i][j]) == False):
43             data[i][j] = 0.0
44
45 # print out objective function which optimizes the preference ratio based
```

Script: Python  
OS: Windows 10  
SW: Eclipse, Python 3.5.1

# Model

## Implementation

1. Data Arrangement
2. Lpsolve Script Generation
3. Lpsolve Analysis

```
lp_diet_male.txt ~
max:
x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_10, x_11, x_12, x_13, x_14, x_15, x_16, x_17, x_18, x_19, x_20, x_21, x_22, x_23, x_24, x_25
x_0+0.5x_1+0.5x_2+0.5x_3+0.5x_4+0.5x_5+0.5x_6+0.5x_7+0.5x_8+0.5x_9+0.5x_10+0.5x_11+0.5x_12+0.5x_13+0.5x_14+0.5x_15+0.5x_16+0.5x_17+0.5x_18+0.5x_19+0.5x_20+0.5x_21+0.5x_22+0.5x_23+0.5x_24+0.5x_25
30x_0+31x_1+32x_2+33x_3+34x_4+35x_5+36x_6+37x_7+38x_8+39x_9+40x_10+41x_11+42x_12+43x_13+44x_14+45x_15+46x_16+47x_17+48x_18+49x_19+50x_20+51x_21+52x_22+53x_23+54x_24+55x_25
50x_0+50x_1+50x_2+50x_3+50x_4+50x_5+50x_6+50x_7+50x_8+50x_9+50x_10+50x_11+50x_12+50x_13+50x_14+50x_15+50x_16+50x_17+50x_18+50x_19+50x_20+50x_21+50x_22+50x_23+50x_24+50x_25
71x_0+72x_1+73x_2+74x_3+75x_4+76x_5+77x_6+78x_7+79x_8+80x_9+81x_10+82x_11+83x_12+84x_13+85x_14+86x_15+87x_16+88x_17+89x_18+90x_19+91x_20+92x_21+93x_22+94x_23+95x_24+96x_25
83x_0+84x_1+85x_2+86x_3+87x_4+88x_5+89x_6+90x_7+91x_8+92x_9+93x_10+94x_11+95x_12+96x_13+97x_14+98x_15+99x_16+100x_17+101x_18+102x_19+103x_20+104x_21+105x_22+106x_23+107x_24+108x_25
107x_0+108x_1+109x_2+110x_3+111x_4+112x_5+113x_6+114x_7+115x_8+116x_9+117x_10+118x_11+119x_12+120x_13+121x_14+122x_15+123x_16+124x_17+125x_18+126x_19+127x_20+128x_21+129x_22+130x_23+131x_24+132x_25
133x_0+134x_1+135x_2+136x_3+137x_4+138x_5+139x_6+140x_7+141x_8+142x_9+143x_10+144x_11+145x_12+146x_13+147x_14+148x_15+149x_16+150x_17+151x_18+152x_19+153x_20+154x_21+155x_22+156x_23+157x_24+158x_25
159x_0+160x_1+161x_2+162x_3+163x_4+164x_5+165x_6+166x_7+167x_8+168x_9+169x_10+170x_11+171x_12+172x_13+173x_14+174x_15+175x_16+176x_17+177x_18+178x_19+179x_20+180x_21+181x_22+182x_23+183x_24+184x_25
185x_0+186x_1+187x_2+188x_3+189x_4+190x_5+191x_6+192x_7+193x_8+194x_9+195x_10+196x_11+197x_12+198x_13+199x_14+200x_15+201x_16+202x_17+203x_18+204x_19+205x_20+206x_21+207x_22+208x_23+209x_24+210x_25
211x_0+212x_1+213x_2+214x_3+215x_4+216x_5+217x_6+218x_7+219x_8+220x_9+221x_10+222x_11+223x_12+224x_13+225x_14+226x_15+227x_16+228x_17+229x_18+230x_19+231x_20+232x_21+233x_22+234x_23+235x_24+236x_25
237x_0+238x_1+239x_2+240x_3+241x_4+242x_5+243x_6+244x_7+245x_8+246x_9+247x_10+248x_11+249x_12+250x_13+251x_14+252x_15+253x_16+254x_17+255x_18+256x_19+257x_20+258x_21+259x_22+260x_23+261x_24+262x_25
263x_0+264x_1+265x_2+266x_3+267x_4+268x_5+269x_6+270x_7+271x_8+272x_9+273x_10+274x_11+275x_12+276x_13+277x_14+278x_15+279x_16+280x_17+281x_18+282x_19+283x_20+284x_21+285x_22+286x_23+287x_24+288x_25
289x_0+290x_1+291x_2+292x_3+293x_4+294x_5+295x_6+296x_7+297x_8+298x_9+299x_10+300x_11+301x_12+302x_13+303x_14+304x_15+305x_16+306x_17+307x_18+308x_19+309x_20+310x_21+311x_22+312x_23+313x_24+314x_25
315x_0+316x_1+317x_2+318x_3+319x_4+320x_5+321x_6+322x_7+323x_8+324x_9+325x_10+326x_11+327x_12+328x_13+329x_14+330x_15+331x_16+332x_17+333x_18+334x_19+335x_20+336x_21+337x_22+338x_23+339x_24+340x_25
341x_0+342x_1+343x_2+344x_3+345x_4+346x_5+347x_6+348x_7+349x_8+350x_9+351x_10+352x_11+353x_12+354x_13+355x_14+356x_15+357x_16+358x_17+359x_18+360x_19+361x_20+362x_21+363x_22+364x_23+365x_24+366x_25
```

Script: Lpsolve  
OS: Mac X  
SW: Lpsolve

```
lp_output_male.txt ~
Value of objective function: 1794.33880959

Actual values of the variables:
x_0 0
x_1 0
x_2 0
x_3 0
x_4 0
x_5 0
x_6 0
x_7 0
x_8 0
x_9 0
x_10 0
x_11 0
x_12 0
x_13 0
x_14 0
x_15 0
x_16 0
x_17 0
x_18 0
x_19 0
x_20 0
x_21 0
x_22 0
x_23 0
x_24 0
x_25 0
```

```
Value of objective function: 1794.33880959
```

Actual values of the variables:

Food Name	Group Name	Variable	100g of food to consume daily (opt value)
Chicory, pale variety, boiled in unsalted water	DG	x_716	30
Cranberries	FA	x_853	30
Duck, raw, meat only, weighed with fat, skin and bone	MCC	x_1049	2.94369
Fennel, Florence, boiled in unsalted water	DG	x_1089	8.29889
Fruit juice drink/squash, no sugar added, diluted	PCC	x_1162	30
Gourd, ridge, raw	DG	x_1199	21.6347
Jackfish, raw	JC	x_1316	0.46012
Jelly, sugar free, made with water	BR	x_1327	30
Lemon juice, fresh, weighed as whole fruit	FC	x_1492	30
Lime juice, fresh	FC	x_1520	15.6523
Marrow, boiled in unsalted water	DG	x_1575	30
Mushrooms, white, raw	DG	x_1699	3.7122
Orange juice, ambient, UHT	FC	x_1760	0.0189024
Passion fruit, flesh and pips, weighed with skin	FA	x_1817	30
Rock Salmon/Dogfish, raw	JA	x_2346	0.990351

Script: Python  
OS: Windows 10  
SW: Eclipse, Python 3.5.1

1. Data Arrangement
2. Lpsolve Script Generation
3. **Lpsolve Analysis**

```

Value of objective function: 1794.33880959

Actual values of the variables:
x_0      0
x_1      0
x_2      0
x_3      0
x_4      0
x_5      0
x_6      0
x_7      0
x_8      0
x_9      0
x_10     0
x_11     0
x_12     0
x_13     0
x_14     0
x_15     0
x_16     0
x_17     0
x_18     0
x_19     0
x_20     0
x_21     0
x_22     0
x_23     0
x_24     0
x_25     0

```

```

1 # The following python code can be run with
2 # an Ipsolve output file as input, like this:
3 # python lp_output_process.py < lp_output.txt > lp_trimmed.txt
4
5 # It will extract the lines giving the non-zero values
6 # and prints out the corresponding food name, group name, and the lp output
7 #
8
9 import fileinput
10 import csv
11 import re
12
13 with open('NewNew_Data.csv') as f:
14     reader = csv.reader(f)
15     next(reader) # skip headers
16     next(reader)
17     next(reader)
18     data = [] # stores necessary food data
19     nRows = 0;
20     for row in reader:
21         # stores each item's
22         # Food Name, Group Name, male preference, female preference, protein(g),
23         # carb(g), energy(kcal), sat fat (g), trans fat(g), cholesterol(mg), so
24         # potassium(mg), calcium(mg), Vitamin D(mcg), Vitamin E(mg), Vitamin B6
25         # Vitamin B12(mcg), Vitamin C(mg)
26         item = [row[1], row[3], row[4], row[5], row[11], row[12], row[13], row[
27 row[48], row[49], row[58], row[51], row[64], row[65], row[72],
28
29         nRows += 1;
30         data.append(item)
31
32 # get the results from the input file
33 for line in fileinput.input():
34     if line.startswith('x'):
35         output = line.split()
36         # find lines that have non-zero values
37         if output[1] != '0':
38             var_num = int(re.search(r'(\d+)', output[0]).group(1))
39             # find the corresponding food name and group name
40             item = [data[var_num][0], data[var_num][1], output[0], output[1]]
41             # align the columns
42             print ('{:0[0]:<60}{0[1]:<45}{0[2]:<15}{0[3]:<30}'.format(item))
43
44 else:
45     if line.startswith('Actual'):
46         print(line)
47         # prints out the header

```

Line 23, Column 57

Spaces: 4

Python

# Solution: Male

Value of objective function: **1794.33880959**

Actual values of the variables:

<b>Food Name</b>	<b>Group Name</b>	<b>Variable</b>	<b>100g of food to consume daily (opt value)</b>
Chicory, pale variety, boiled in unsalted water	DG	x_716	30
Cranberries	FA	x_853	30
Duck, raw, meat only, weighed with fat, skin and bone	MCC	x_1049	2.94369
Fennel, Florence, boiled in unsalted water	DG	x_1089	8.29889
Fruit juice drink/squash, no sugar added, diluted	PCC	x_1162	30
Gourd, ridge, raw	DG	x_1199	21.6347
Jackfish, raw	JC	x_1316	0.46012
Jelly, sugar free, made with water	BR	x_1327	30
Lemon juice, fresh, weighed as whole fruit	FC	x_1492	30
Lime juice, fresh	FC	x_1520	15.6523
Marrow, boiled in unsalted water	DG	x_1575	30
Mushrooms, white, raw	DG	x_1699	3.7122
Orange juice, ambient, UHT	FC	x_1760	0.0189024
Passion fruit, flesh and pips, weighed with skin	FA	x_1817	30
Rock Salmon/Dogfish, raw	JA	x_2346	0.990351

# Solution: Female

Value of objective function: **1639.81681676**

Actual values of the variables:

<b>Food Name</b>	<b>Group Name</b>	<b>Variable</b>	<b>100g of food to consume daily (opt value)</b>
Chicory, pale variety, boiled in unsalted water	DG	x_716	30
Cranberries	FA	x_853	30
Fennel, Florence, boiled in unsalted water	DG	x_1089	9.78434
Gourd, ridge, raw	DG	x_1199	2.39609
Jackfish, raw	JC	x_1316	0.809946
Jelly, sugar free, made with water	BR	x_1327	30
Lemon juice, fresh, weighed as whole fruit	FC	x_1492	30
Limes, flesh only, weighed with peel and pips	FA	x_1522	20.8073
Marrow, boiled in unsalted water	DG	x_1575	30
Mushrooms, white, raw	DG	x_1699	3.72073
Nutmeg, ground	H	x_1731	0.269869
Orange juice, ambient, UHT	FC	x_1760	0.0146341
Passion fruit, flesh and pips, weighed with skin	FA	x_1817	25.8045
Rock Salmon/Dogfish, raw	JA	x_2346	0.490755
Tea, black, infusion, average	PAA	x_2647	30

# Variations

- Unhappy case: minimize our satisfaction but in a healthy way
  - Try to minimize our happiness and keep enough daily nutrition
  - the value of objective functions for male (8.33620465) and female (15.27183148) both decrease sharply
  - the numbers of food in the optimal food list for male and female are less than original model
  - Male: seven kinds of food (originally fifteen)
  - Female: nine kinds of food (originally fifteen)

# Variations

- Vegetarian case: maximize our preference with a vegetarian diet
  - We modify our model and add some constraints that set the amount of all the food that made with meat and fish to zero
  - optimal happiness scores  $H_{\max} = 1454.78168382$  for male,  
 $H_{\max} = 1296.64981001$  for female
  - substituting meat and fish will reduce our happiness when we have the same nutrient constraints as before
  - consume more cereals and dairy to fulfill the nutrient need

# Variations

- Diversity case: maximize our happiness with plenty of diversity of food items
  - We modify the original model to enlarge our possibility to getting more different types of food
  - we bound each group with an upper limit so that we cannot eat more than 1000 grams of any food for a single day
  - The maximum happiness score drop from 1794.33880959 to 618.16189370 for male and from 1639.81681676 to 558.85880275 for female
  - male can consume food from 14 kind of groups, female can consume food from 13 different groups, originally both male and female eat food from 8 different groups



# Conclusions

## From original study

- To maximize our **happiness**, a daily diet should consist of mostly fruits and vegetables.
- For non-vegetarians, fish seems to be the most nutritious meat type.
- Debatably, from our results, male seem to be more easily satisfied than female.

## From variation studies

- To make oneself unhappy, breakfast cereal and milk/cheese might be good choices for male and female respectively.
- For vegetarians, with vegetables and fruits they could never be alone and sick.
- Having a diverse diet → a more balanced diet + a lower happiness level.

# Conclusions

## **Limitations:**

- Preference scores based on the small survey within our group (n=4), not representative
- The survey was inevitably focused on each food group, instead of individual food.
- Based on the dataset by Public Health England, cannot cover all U.S. foods

## **Further studies:**

- Preference scores based on bigger survey randomly sampled from different populations.
- Survey focusing on individual food.
- Based on the dataset by U.S. Department of Agriculture
- Personal diet based on individuals

# References

1. American Heart Association, "Know Your Facts"  
[http://www.heart.org/HEARTORG/Conditions/Cholesterol/PreventionTreatmentofHighCholesterol/Know-Your-Fats\\_UCM\\_305628\\_Article.jsp](http://www.heart.org/HEARTORG/Conditions/Cholesterol/PreventionTreatmentofHighCholesterol/Know-Your-Fats_UCM_305628_Article.jsp)
2. ConsumerLab, "Recommended Daily Intakes and Upper Limits for Nutrients" <http://www.consumerlab.com/RDAs/>
3. Sourceforge, "Ipsolve" <https://sourceforge.net/projects/ipsolve/>
4. NEOS, "The Diet Problem" <http://www.neos-guide.org/content/diet-problem>
5. Public Health England, "McCance and Widdowson's The Composition of Foods Integrated Dataset 2015"  
<https://www.gov.uk/government/publications/composition-of-foods-integrated-dataset-cofid>
6. SF Gate, "Daily Amounts of Carbs, Fat, Fiber, Sodium & Protein"  
<http://healthyeating.sfgate.com/daily-amounts-carbs-fat-fiber-sodium-protein-4230.html>
7. Sparkpeople, "Healthy Beverage Guidelines" [http://www.sparkpeople.com/resource/nutrition\\_articles.asp?id=605](http://www.sparkpeople.com/resource/nutrition_articles.asp?id=605)
8. US Department of Agriculture, "Estimated Calorie Needs per Day by Age, Gender, and Physical Activity Level"  
[https://www.cnpp.usda.gov/sites/default/files/usda\\_food\\_patterns/EstimatedCalorieNeedsPerDayTable.pdf](https://www.cnpp.usda.gov/sites/default/files/usda_food_patterns/EstimatedCalorieNeedsPerDayTable.pdf)
9. US Food and Drug Administration, "Guidance for Industry: A Food Labeling Guide (14. Appendix F: Calculate the Percent Daily Value for the Appropriate Nutrients)"  
<http://www.fda.gov/Food/GuidanceRegulation/GuidanceDocumentsRegulatoryInformation/LabelingNutrition/ucm064928.htm>
10. Vegetarian Resource Group, "Veganism in a Nutshell" <http://www.vrg.org/nutshell/vegan.htm#what>
11. Wikipedia, "Knapsack problem" [https://en.wikipedia.org/wiki/Knapsack\\_problem](https://en.wikipedia.org/wiki/Knapsack_problem)
12. Wikipedia, "Stigler Diet" [https://en.wikipedia.org/wiki/Stigler\\_diet](https://en.wikipedia.org/wiki/Stigler_diet)

Questions?



thank you!