Diet Final Code

March 3, 2023

0.1 Function to solve the subsistence problem

The following code block defines a function `solve_subsistence_problem`, which takes as arguments a dataframe mapping different foods to nutrients; a series of prices for those same foods; a series giving dietary recommended intake (DRI) minimums; and a series giving dietary recommended maximums.

[1]:
```
from  scipy.optimize import linprog as lp
import numpy as np
import warnings

def␣
↪solve_subsistence_problem(FoodNutrients,Prices,dietmin,dietmax,max_weight=None,tol=1e-6):
↪
    """Solve Stigler's Subsistence Cost Problem.

    Inputs:
       - FoodNutrients : A pd.DataFrame with rows corresponding to foods,␣
↪columns to nutrients.
       - Prices : A pd.Series of prices for different foods
       - diet_min : A pd.Series of DRIs, with index corresponding to columns of␣
↪FoodNutrients,
                    describing minimum intakes.
       - diet_max : A pd.Series of DRIs, with index corresponding to columns of␣
↪FoodNutrients,
                    describing maximum intakes.
       - max_weight : Maximum weight (in hectograms) allowed for diet.
       - tol : Solution values smaller than this in absolute value treated as␣
↪zeros.

    """
    try:
        p = Prices.apply(lambda x:x.magnitude)
    except AttributeError:  # Maybe not passing in prices with units?
        warnings.warn("Prices have no units.  BE CAREFUL!  We're assuming␣
↪prices are per hectogram or deciliter!")
        p = Prices
```

```python
    p = p.dropna()

    # Compile list that we have both prices and nutritional info for; drop if
↪either missing
    use = p.index.intersection(FoodNutrients.columns)
    p = p[use]

    # Drop nutritional information for foods we don't know the price of,
    # and replace missing nutrients with zeros.
    Aall = FoodNutrients[p.index].fillna(0)

    # Drop rows of A that we don't have constraints for.
    Amin = Aall.loc[Aall.index.intersection(dietmin.index)]
    Amin = Amin.reindex(dietmin.index,axis=0)
    idx = Amin.index.to_frame()
    idx['type'] = 'min'
    #Amin.index = pd.MultiIndex.from_frame(idx)
    #dietmin.index = Amin.index

    Amax = Aall.loc[Aall.index.intersection(dietmax.index)]
    Amax = Amax.reindex(dietmax.index,axis=0)
    idx = Amax.index.to_frame()
    idx['type'] = 'max'
    #Amax.index = pd.MultiIndex.from_frame(idx)
    #dietmax.index = Amax.index

    # Minimum requirements involve multiplying constraint by -1 to make <=.
    A = pd.concat([Amin,
                   -Amax])

    b = pd.concat([dietmin,
                   -dietmax]) # Note sign change for max constraints

    # Make sure order of p, A, b are consistent
    A = A.reindex(p.index,axis=1)
    A = A.reindex(b.index,axis=0)

    if max_weight is not None:
        # Add up weights of foods consumed
        A.loc['Hectograms'] = -1
        b.loc['Hectograms'] = -max_weight

    # Now solve problem!  (Note that the linear program solver we'll use assumes
    # "less-than-or-equal" constraints.  We can switch back and forth by
    # multiplying $A$ and $b$ by $-1$.)

    result = lp(p, -A, -b, method='interior-point')
```

```
    result.A = A
    result.b = b

    if result.success:
        result.diet = pd.Series(result.x,index=p.index)
    else: # No feasible solution?
        warnings.warn(result.message)
        result.diet = pd.Series(result.x,index=p.index)*np.nan

    return result
```

0.2 Setup

[2]:
```
#!git reset --hard origin/master  # To revert to original
!pip install -r requirements.txt --upgrade
```

```
Requirement already satisfied: pint>=0.18 in /opt/conda/lib/python3.9/site-
packages (from -r requirements.txt (line 3)) (0.20.1)
Requirement already satisfied: requests>=2.26.0 in
/opt/conda/lib/python3.9/site-packages (from -r requirements.txt (line 6))
(2.28.2)
Requirement already satisfied: python-gnupg in /opt/conda/lib/python3.9/site-
packages (from -r requirements.txt (line 8)) (0.5.0)
Requirement already satisfied: eep153_tools in /opt/conda/lib/python3.9/site-
packages (from -r requirements.txt (line 10)) (0.11)
Requirement already satisfied: fooddatacentral in /opt/conda/lib/python3.9/site-
packages (from -r requirements.txt (line 12)) (1.0.9)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.9/site-packages (from requests>=2.26.0->-r
requirements.txt (line 6)) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.9/site-
packages (from requests>=2.26.0->-r requirements.txt (line 6)) (3.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.9/site-packages (from requests>=2.26.0->-r
requirements.txt (line 6)) (1.26.7)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.9/site-packages (from requests>=2.26.0->-r
requirements.txt (line 6)) (2.0.0)
```

0.3 USDA Food Central DataBase

API key to access USDA Food Central DataBase

[3]:
```
# API key for Gov; substitute your own!
apikey = "zjPzq6seNovgsh28xoUlw7NW7Ikj7Zidg1We4Ekv"
```

0.4 Data on Prices of Different Foods

We have a spreadsheet for the prices in each supermarket (Trader Joes, Walmart, Wholefoods, Target, and FoodMaxx) . We will be using the read_sheets function to import the data on prices into this notebook. You can find the spreadsheet here: (https://docs.google.com/spreadsheets/d/1dZW1__vvjZwcAfxfHiAkjqrLEpFbFsZ4s7__MzTG6Ezu0/edit#gid=11

Below we will show dataframes for each of the 5 supermarkets we obtained grocery prices from.

0.5 Trader Joes Price List

[4]:
```
import pandas as pd
from eep153_tools.sheets import read_sheets

df = read_sheets('https://docs.google.com/spreadsheets/d/
 ↪1dZW1_vvjZwcAfxfHiAkjqrLEpFbFsZ4s7_MzTG6Ezu0/edit#gid=0',sheet='Trader Joes')
df
```

```
Key available for students@eep153.iam.gserviceaccount.com.
```

[4]:
```
                       Food  Quantity  Units  Price      Date      Location  \
0                Banana,raw    120.00  grams   0.19  02/27/23  Trader Joe's
1               Apple, Gala      0.33  pound   1.29  02/27/23  Trader Joe's
2            Oranges, Navel      3.00  pound   3.69  02/27/23  Trader Joe's
3               Blueberries     11.00     oz   4.49  03/01/23  Trader Joe's
4               Raspberries      6.00     oz   3.99  03/01/23  Trader Joe's
5    Grapes, green, seedless      2.00  pound   5.99  03/01/23  Trader Joe's
6              Strawberries      1.00  pound   3.99  02/27/23  Trader Joe's
7              Lentils, red     17.63     oz   3.29  02/27/23  Trader Joe's
8                      Tofu     19.00     oz   2.29  03/01/23  Trader Joe's
9           Split peas, dry     12.00     oz   2.69  03/01/23  Trader Joe's
10         Chickpeas, canned      9.88     oz   1.99  03/01/23  Trader Joe's
11              Quinoa, dry     16.00     oz   3.99  02/27/23  Trader Joe's
12                   Hummus     16.00     oz   3.99  02/27/23  Trader Joe's
13       Black beans, canned     15.50     oz   1.09  02/27/23  Trader Joe's
14             Tomato, Roma      1.00  pound   2.99  02/27/23  Trader Joe's
15                 Broccoli     12.00     oz   2.49  03/01/23  Trader Joe's
16                     Kale     10.00     oz   2.99  02/27/23  Trader Joe's
17          Green beans, raw     24.00     oz   1.99  02/27/23  Trader Joe's
18           Onions, yellow      4.30     oz   0.99  03/01/23  Trader Joe's
19                   Celery     16.00     oz   2.79  03/01/23  Trader Joe's
20           Potato, Russet    200.00  grams   0.79  02/27/23  Trader Joe's
21                   Carrot      2.00  pound   1.99  02/27/23  Trader Joe's
22                 Cucumber      1.00  pound   2.49  02/27/23  Trader Joe's
23                  Avocado      5.40     oz   0.60  03/01/23  Trader Joe's
24          Lettuce, Romaine      5.00     oz   2.49  02/27/23  Trader Joe's
25                Mushrooms      6.00     oz   3.29  02/27/23  Trader Joe's
26                  Spinach     12.00     oz   2.49  03/01/23  Trader Joe's
27              Rice, White      3.00  pound   2.99  02/27/23  Trader Joe's
```

28	Tortillas, corn	12.00	oz	1.29	03/01/23	Trader Joe's
29	Oats, rolled	18.00	oz	2.99	02/27/23	Trader Joe's
30	Soy milk, unsweetened	32.00	oz	2.49	02/27/23	Trader Joe's
31	Peanut butter, creamy	16.00	oz	2.29	02/27/23	Trader Joe's
32	Rice, Brown	32.00	oz	3.29	03/01/23	Trader Joe's
33	Spaghetti, dry	1.00	pound	0.99	02/27/23	Trader Joe's

	FDC
0	1105314
1	1750341
2	746771
3	2346411
4	2346410
5	2346413
6	167762
7	174284
8	173788
9	172428
10	173800
11	168874
12	174289
13	175238
14	1999634
15	539572
16	1103116
17	2346400
18	790646
19	2346405
20	2060240
21	170393
22	2346406
23	171705
24	169247
25	1999629
26	1999632
27	168931
28	167535
29	2346396
30	1999630
31	2262072
32	169706
33	168927

0.6 Walmart Price List

[5]:
```python
import pandas as pd
from eep153_tools.sheets import read_sheets

walmartdf = read_sheets('https://docs.google.com/spreadsheets/d/
↪1dZW1_vvjZwcAfxfHiAkjqrLEpFbFsZ4s7_MzTG6Ezu0/edit#gid=0',sheet='Walmart')
walmartdf
```

```
Key available for students@eep153.iam.gserviceaccount.com.
```

[5]:
```
                         Food  Quantity Units  Price      Date Location  \
0                  Banana,raw       1.0   lbs   0.77  02/27/23  walmart
1                 Apple, Gala       1.0   lbs   1.67  02/27/23  walmart
2              Oranges, Navel       1.0   lbs   0.88  02/27/23  walmart
3                 Blueberries      11.0    oz   3.38  02/27/23  walmart
4                 Raspberries      12.0    oz   5.98  02/27/23  walmart
5     Grapes, green, seedless       1.0   lbs   1.98  02/27/23  walmart
6                Strawberries       1.0   lbs   2.78  02/27/23  walmart
7                Lentils, red      27.0    oz   9.55  03/01/23  walmart
8                        Tofu       1.0   lbs   2.12  02/27/23  walmart
9             Split peas, dry       1.0   lbs   1.48  02/27/23  walmart
10           Chickpeas, canned      15.5    oz   0.78  02/27/23  walmart
11                Quinoa, dry       1.0   lbs   3.72  02/27/23  walmart
12               Fresh Hummus      10.0    oz   3.47  02/27/23  walmart
13         Black beans, canned      15.0    oz   0.78  02/27/23  walmart
14               Tomato, Roma       1.0   lbs   1.28  02/27/23  walmart
15            Broccoli Florest      1.0    oz   4.15  02/27/23  walmart
16                       Kale      40.0    oz  11.78  02/27/23  walmart
17           Green beans, raw      12.0    oz   2.78  02/27/23  walmart
18             Onions, yellow       3.0    oz   2.24  02/27/23  walmart
19                     Celery      20.0    oz   2.98  02/27/23  walmart
20             Potato, Russet       5.0   lbs   3.47  02/27/23  walmart
21                     Carrot       1.0   lbs   0.98  02/27/23  walmart
22                   Cucumber       1.0    oz   0.68  02/27/23  walmart
23                    Avocado       1.0   lbs   1.09  02/27/23  walmart
24            Lettuce, Romaine      10.0    oz   2.98  02/27/23  walmart
25      Fresh whole Mushrooms       8.0    oz   2.08  02/27/23  walmart
26   Market side fresh Spinach     10.0    oz   2.28  02/27/23  walmart
27                Rice, White      14.0    oz   2.44  02/27/23  walmart
28            Tortillas, corn      25.0    oz   2.48  02/27/23  walmart
29        Organic Oats, rolled     24.0    oz  12.29  02/27/23  walmart
30       Soy milk, unsweetened     32.0  floz   3.24  02/27/23  walmart
31       Peanut butter, creamy     24.0    oz  12.56  02/27/23  walmart
32                Rice, Brown      32.0    oz   1.37  02/27/23  walmart
33             Spaghetti, dry       1.0    oz   0.09  02/27/23  walmart

        FDC     Note
```

```
0    1105073
1    1750341
2     746771
3    2346411
4    2346410
5    2346413
6     167762
7     174284
8     173788
9     172428    green
10    173800
11    168874  organic
12    174289
13    175238
14   1999634
15    539572
16   1103116
17   2346400
18    790646  organic
19   2346405
20   2060240
21   1103193    whole
22   2346406
23    815990
24    169247
25   1999629
26   1999632
27    168931
28    167535
29   2346396
30   1999630
31   2262072  organic
32    169706    Fresh
33    168927
```

0.7 Whole Foods Price List

[6]:
```
import pandas as pd
from eep153_tools.sheets import read_sheets

wholefoodsdf = read_sheets('https://docs.google.com/spreadsheets/d/
 ↪1dZW1_vvjZwcAfxfHiAkjqrLEpFbFsZ4s7_MzTG6Ezu0/edit#gid=0',sheet='Whole Foods')
wholefoodsdf
```

```
Key available for students@eep153.iam.gserviceaccount.com.
```

```
[6]:                      Food      FDC  Quantity Units  Price     Date  \
0                  Banana,raw  1105314       1.0   lbs   0.99  2/27/23
1                 Apple, Gala  1750341       3.0   lbs   6.49  2/27/23
2              Oranges, Navel   746771       1.0   lbs   2.39  2/27/23
3                 Blueberries  2346411      11.0    oz   6.99  2/27/23
4                 Raspberries  2346410      12.0    oz   8.69  2/27/23
5     Grapes, green, seedless  2346413       3.0   lbs   7.49  2/27/23
6                Strawberries   167762       1.0   lbs   4.49  2/27/23
7                Lentils, red   174284       1.0   lbs   2.99  2/27/23
8                        Tofu   173788      14.0    oz   2.99  2/27/23
9             Split peas, dry   172428       1.0   lbs   2.99  2/27/23
10           Chickpeas, canned   173800      15.5    oz   1.99  2/27/23
11                Quinoa, dry   168874       1.0   lbs   4.79  2/27/23
12                     Hummus   174289       1.0   lbs   5.29  2/27/23
13         Black beans, canned   175238      15.0    oz   1.99  2/27/23
14                Tomato, Roma  1999634       1.0   lbs   3.19  2/27/23
15                   Broccoli   539572       1.0   lbs   2.99  2/27/23
16                       Kale  1103116       1.0   lbs   4.99  2/27/23
17            Green beans, raw  2346400       1.0   lbs   2.99  2/27/23
18             Onions, yellow   790646       1.0   lbs   1.99  2/27/23
19                     Celery  2346405       1.0   lbs   2.69  2/27/23
20             Potato, Russet  2060240       1.0   lbs   1.69  2/27/23
21                     Carrot   170393       1.0   lbs   1.49  2/27/23
22                   Cucumber  2346406      14.0    oz   1.79  2/27/23
23                    Avocado   171705       8.0    oz   1.99  2/27/23
24           Lettuce, Romaine   169247      22.0    oz   2.99  2/27/23
25                  Mushrooms  1999629       8.0    oz   3.49  2/27/23
26                    Spinach  1999632       5.0    oz   3.99  2/27/23
27                Rice, White   168931       2.0   lbs   5.29  2/27/23
28            Tortillas, corn   167535       8.0    oz   1.99  2/27/23
29               Oats, rolled  2346396      42.0    oz   5.99  2/27/23
30     Soy milk, unsweetened  1999630      32.0    oz   5.39  2/27/23
31      Peanut butter, creamy  2262072      36.0    oz   6.79  2/27/23
32                Rice, Brown   169706      80.0    oz   5.99  2/27/23
33             Spaghetti, dry   168927      16.0    oz   1.59  2/27/23

       Location         Notes
0   Whole Foods
1   Whole Foods
2   Whole Foods
3   Whole Foods
4   Whole Foods  Only organic
5   Whole Foods
6   Whole Foods
7   Whole Foods
8   Whole Foods
9   Whole Foods
```

```
10  Whole Foods
11  Whole Foods  Only organic
12  Whole Foods
13  Whole Foods
14  Whole Foods
15  Whole Foods
16  Whole Foods
17  Whole Foods
18  Whole Foods
19  Whole Foods
20  Whole Foods
21  Whole Foods       Oragnic
22  Whole Foods
23  Whole Foods
24  Whole Foods
25  Whole Foods
26  Whole Foods
27  Whole Foods
28  Whole Foods
29  Whole Foods
30  Whole Foods       Organic
31  Whole Foods
32  Whole Foods
33  Whole Foods
```

0.8 FoodMaxx Price List

[7]:

```
import pandas as pd
from eep153_tools.sheets import read_sheets

foodmaxxdf = read_sheets('https://docs.google.com/spreadsheets/d/
↪1dZW1_vvjZwcAfxfHiAkjqrLEpFbFsZ4s7_MzTG6Ezu0/edit#gid=0',sheet='Food Max')
foodmaxxdf
```

```
Key available for students@eep153.iam.gserviceaccount.com.
```

[7]:

```
                        Food  Quantity Units  Price      Date  Location  \
0                 Banana,raw         1   lbs   0.69  02/27/23  FoodMaxx
1               Apple, Gala         1   lbs   0.99  02/27/23  FoodMaxx
2            Oranges, Navel         1   lbs   0.99  02/27/23  FoodMaxx
3                Blueberries         6    oz   3.49  02/27/23  FoodMaxx
4                Raspberries         1   lbs   4.99  02/27/23  FoodMaxx
5   Grapes, green, seedless         1   lbs   2.99  02/27/23  FoodMaxx
6               Strawberries         1   lbs   6.99  02/27/23  FoodMaxx
7               Lentils, red         1   lbs   1.99  02/27/23  FoodMaxx
8                       Tofu        14    oz   2.99  02/27/23  FoodMaxx
9            Split peas, dry         1   lbs   1.79  02/27/23  FoodMaxx
```

10	Chickpeas, canned	1	lbs	0.79	02/27/23	FoodMaxx
11	Quinoa, dry	1	lbs	4.99	02/27/23	FoodMaxx
12	Hummus	10	oz	3.99	02/27/23	FoodMaxx
13	Black beans, canned	15	oz	1.19	02/27/23	FoodMaxx
14	Tomato, Roma	1	lbs	1.39	02/27/23	FoodMaxx
15	Broccoli	1	lbs	1.89	02/27/23	FoodMaxx
16	Kale	8	oz	1.29	02/27/23	FoodMaxx
17	Green beans, raw	12	oz	3.29	02/27/23	FoodMaxx
18	Onions, yellow	5	lbs	4.69	02/27/23	FoodMaxx
19	Celery	1	lbs	1.29	02/27/23	FoodMaxx
20	Potato, Russet	1	lbs	1.89	02/27/23	FoodMaxx
21	Carrot	1	lbs	0.69	02/27/23	FoodMaxx
22	Cucumber	8	oz	0.79	02/27/23	FoodMaxx
23	Avocado	5	oz	0.50	02/27/23	FoodMaxx
24	Lettuce, Romaine	1	lbs	3.49	02/27/23	FoodMaxx
25	Mushrooms	8	oz	2.99	02/27/23	FoodMaxx
26	Spinach	8	oz	2.49	02/27/23	FoodMaxx
27	Rice, White	2	lbs	4.59	02/27/23	FoodMaxx
28	Tortillas, corn	30	oz	2.79	02/27/23	FoodMaxx
29	Oats, rolled	42	oz	3.99	02/27/23	FoodMaxx
30	Soy milk, unsweetened	32	oz	2.69	02/27/23	FoodMaxx
31	Peanut butter, creamy	16	oz	2.79	02/27/23	FoodMaxx
32	Rice, Brown	28	oz	2.99	02/27/23	FoodMaxx
33	Spaghetti, dry	1	lbs	1.69	02/27/23	FoodMaxx

	FDC
0	1105073
1	1750341
2	746771
3	2346411
4	2346410
5	2346413
6	167762
7	174284
8	173788
9	172428
10	173800
11	168874
12	174289
13	175238
14	1999634
15	539572
16	1103116
17	2346400
18	790646
19	2346405
20	2060240

```
21   1103193
22   2346406
23    815990
24    169247
25   1999629
26   1999632
27    168931
28    167535
29   2346396
30   1999630
31   2262072
32    169706
33    168927
```

0.9 Target Price List

[8]:
```
import pandas as pd
from eep153_tools.sheets import read_sheets

targetdf = read_sheets('https://docs.google.com/spreadsheets/d/
↪1dZW1_vvjZwcAfxfHiAkjqrLEpFbFsZ4s7_MzTG6Ezu0/edit#gid=0',sheet='Target')
targetdf
```

```
Key available for students@eep153.iam.gserviceaccount.com.
```

[8]:
```
                         Food  Quantity Units  Price      Date Location  \
0                  Banana,raw      2.00   lbs   1.99  02/27/23   Target
1                 Apple, Gala      0.66   lbs   5.99  02/27/23   Target
2              Oranges, Navel      4.00   lbs   5.99  02/27/23   Target
3                 Blueberries     11.20    oz   3.59  02/27/23   Target
4                 Raspberries     12.00    oz   8.29  02/27/23   Target
5    Grapes, green, seedless      1.50    lb   4.29  02/27/23   Target
6                Strawberries      1.00    lb   3.49  02/27/23   Target
7                Lentils, red      1.00    lb   1.59  02/27/23   Target
8                        Tofu     14.00    oz   3.29  02/27/23   Target
9            Split peas, dry      1.00    lb   1.39  02/27/23   Target
10         Chickpeas, canned     15.50    oz   0.85  02/27/23   Target
11               Quinoa, dry     48.00    oz   9.69  02/27/23   Target
12                    Hummus     10.00    oz   3.49  02/27/23   Target
13       Black beans, canned     15.50    oz   0.85  02/27/23   Target
14              Tomato, Roma     16.00    oz   1.99  02/27/23   Target
15                  Broccoli     12.00    oz   2.99  02/27/23   Target
16                      Kale     16.00    oz   3.49  02/27/23   Target
17          Green beans, raw     12.00    oz   2.99  02/27/23   Target
18            Onions, yellow      2.00    lb   3.49  02/27/23   Target
19                    Celery     20.00    oz   3.19  02/27/23   Target
20            Potato, Russet      5.00    lb   4.39  02/27/23   Target
```

21	Carrot	1.00	lb	1.29	02/27/23	Target
22	Cucumber	16.00	oz	2.69	02/27/23	Target
23	Avocado	32.00	oz	3.29	02/27/23	Target
24	Lettuce, Romaine	22.00	oz	4.29	02/27/23	Target
25	Mushrooms	8.00	oz	2.19	02/27/23	Target
26	Spinach	9.00	oz	2.29	02/27/23	Target
27	Rice, White	32.00	oz	2.19	02/27/23	Target
28	Tortillas, corn	25.00	oz	2.99	02/27/23	Target
29	Oats, rolled	42.00	oz	4.19	02/27/23	Target
30	Soy milk, unsweetened	64.00	oz	3.99	02/27/23	Target
31	Peanut butter, creamy	16.00	oz	1.79	02/27/23	Target
32	Rice, Brown	1.00	lb	1.19	02/27/23	Target
33	Spaghetti, dry	16.00	oz	0.99	02/27/23	Target

	FDC	notes
0	1105314	organic
1	1750341	organic
2	746771	
3	2346411	
4	2346410	
5	2346413	
6	167762	
7	174284	green
8	173788	
9	172428	
10	173800	
11	168874	organic
12	174289	
13	175238	
14	1999634	
15	539572	
16	1103116	
17	2346400	
18	790646	
19	2346405	
20	2060240	
21	170393	
22	2346406	
23	171705	
24	169247	
25	1999629	
26	1999632	
27	168931	
28	167535	
29	2346396	
30	1999630	
31	2262072	

```
32    169706
33    168927
```

0.9.1 Look up nutritional information for foods

Now we have a list of foods with prices. For the remainder of this code, we will be using data from Trader Joes. Do lookups on USDA database to get nutritional information.

[9]:
```python
import fooddatacentral as fdc
import warnings

D = {}
count = 0
for food in  df.Food.tolist():
    try:
        FDC = df.loc[df.Food==food,:].FDC[count]
        count+=1
        D[food] = fdc.nutrients(apikey,FDC).Quantity
    except AttributeError:
        warnings.warn("Couldn't find FDC Code %s for food %s." % (food,FDC))

FoodNutrients = pd.DataFrame(D,dtype=float)
FoodNutrients = FoodNutrients.fillna(0)
FoodNutrients
```

[9]:
```
                         Banana,raw  Apple, Gala  Oranges, Navel  Blueberries  \
 Ergosta-5,7-dienol             0.0          0.0           0.000          0.0
 Ergosta-7,22-dienol            0.0          0.0           0.000          0.0
Alanine                         0.0          0.0           0.028          0.0
Alcohol, ethyl                  0.0          0.0           0.000          0.0
Amino acids                     0.0          0.0           0.000          0.0
…                                …          …               …          …
cis-Lutein/Zeaxanthin           0.0          0.0           0.000          0.0
cis-Lycopene                    0.0          0.0           0.000          0.0
cis-beta-Carotene               1.0          0.0           0.000          0.0
trans-Lycopene                  0.0          0.0           0.000          0.0
trans-beta-Carotene             7.0          0.0           0.000          0.0

                        Raspberries  Grapes, green, seedless  Strawberries  \
 Ergosta-5,7-dienol             0.0                      0.0         0.000
 Ergosta-7,22-dienol            0.0                      0.0         0.000
Alanine                         0.0                      0.0         0.033
Alcohol, ethyl                  0.0                      0.0         0.000
Amino acids                     0.0                      0.0         0.000
…                                …                       …            …
cis-Lutein/Zeaxanthin           0.0                      0.0         0.000
cis-Lycopene                    0.0                      0.0         0.000
```

cis-beta-Carotene	0.0	0.0	0.000
trans-Lycopene	0.0	0.0	0.000
trans-beta-Carotene	0.0	0.0	0.000

	Lentils, red	Tofu	Split peas, dry	… \
Ergosta-5,7-dienol	0.000	0.0	0.000	…
Ergosta-7,22-dienol	0.000	0.0	0.000	…
Alanine	1.042	0.0	1.049	…
Alcohol, ethyl	0.000	0.0	0.000	…
Amino acids	0.000	0.0	0.000	…
…	…	…	…	…
cis-Lutein/Zeaxanthin	0.000	0.0	0.000	…
cis-Lycopene	0.000	0.0	0.000	…
cis-beta-Carotene	0.000	0.0	0.000	…
trans-Lycopene	0.000	0.0	0.000	…
trans-beta-Carotene	0.000	0.0	0.000	…

	Lettuce, Romaine	Mushrooms	Spinach	Rice, White \
Ergosta-5,7-dienol	0.000	5.841	0.0	0.000
Ergosta-7,22-dienol	0.000	1.543	0.0	0.000
Alanine	0.056	0.000	0.0	0.377
Alcohol, ethyl	0.000	0.000	0.0	0.000
Amino acids	0.000	0.000	0.0	0.000
…	…	…	…	…
cis-Lutein/Zeaxanthin	0.000	0.000	0.0	0.000
cis-Lycopene	0.000	0.000	0.0	0.000
cis-beta-Carotene	0.000	0.000	0.0	0.000
trans-Lycopene	0.000	0.000	0.0	0.000
trans-beta-Carotene	0.000	0.000	0.0	0.000

	Tortillas, corn	Oats, rolled	Soy milk, unsweetened \
Ergosta-5,7-dienol	0.000	0.0	0.0000
Ergosta-7,22-dienol	0.000	0.0	0.0000
Alanine	0.215	0.0	0.1394
Alcohol, ethyl	0.000	0.0	0.0000
Amino acids	0.000	0.0	0.0000
…	…	…	…
cis-Lutein/Zeaxanthin	0.000	0.0	0.9655
cis-Lycopene	0.000	0.0	0.0000
cis-beta-Carotene	0.000	0.0	0.0000
trans-Lycopene	0.000	0.0	0.0000
trans-beta-Carotene	0.000	0.0	0.0000

	Peanut butter, creamy	Rice, Brown	Spaghetti, dry
Ergosta-5,7-dienol	0.00	0.000	0.000
Ergosta-7,22-dienol	0.00	0.000	0.000
Alanine	1.16	0.437	0.438

```
Alcohol, ethyl                                      0.00        0.000            0.000
Amino acids                                         0.00        0.000            0.000
…                                                  …          …              …
cis-Lutein/Zeaxanthin                               0.00        0.000            0.000
cis-Lycopene                                        0.00        0.000            0.000
cis-beta-Carotene                                   0.00        0.000            0.000
trans-Lycopene                                      0.00        0.000            0.000
trans-beta-Carotene                                 0.00        0.000            0.000

[227 rows x 34 columns]
```

0.10 Units & Prices

Now, the prices we observe can be for lots of different quantities and units. The FDC database basically wants everything in either hundreds of grams (hectograms) or hundreds of milliliters (deciliters).
We use the `units` function to convert all foods to either deciliters or hectograms, to match FDC database:

[10]:
```python
# Convert food quantities to FDC units
df['FDC Quantity'] = df[['Quantity','Units']].T.apply(lambda x : fdc.
 ↪units(x['Quantity'],x['Units']))

# Now may want to filter df by time or place--need to get a unique set of food␣
 ↪names.
df['FDC Price'] = df['Price']/df['FDC Quantity']

df.dropna(how='any') # Drop food with any missing data

# To use minimum price observed
Prices = df.groupby('Food',sort=False)['FDC Price'].min()
Prices
```

```
/opt/conda/lib/python3.9/site-packages/pandas/core/dtypes/cast.py:1990:
UnitStrippedWarning: The unit of the quantity is stripped when downcasting to
ndarray.
  result[:] = values
```

[10]:
```
Food
 Banana,raw                0.15833333333333335 / hectogram
Apple, Gala                 0.8618070249045213 / hectogram
Oranges, Navel              0.2711685824873994 / hectogram
Blueberries                 1.4398189923056004 / hectogram
Raspberries                 2.3457184696470974 / hectogram
Grapes, green, seedless     0.6602844752437083 / hectogram
Strawberries                0.8796444261176615 / hectogram
Lentils, red                0.6582605491441835 / hectogram
```

```
Tofu                           0.4251440677081007 / hectogram
Split peas, dry                0.7907246470364275 / hectogram
Chickpeas, canned                0.71047757368082 / hectogram
Quinoa, dry                    0.8796444261176615 / hectogram
Hummus                         0.8796444261176615 / hectogram
Black beans, canned           0.24805560338737195 / hectogram
Tomato, Roma                    0.659182163932784 / hectogram
Broccoli                       0.7319347104537935 / hectogram
Kale                           1.0546914622924541 / hectogram
Green beans, raw              0.29247993449860427 / hectogram
Onions, yellow                 0.8121214495368513 / hectogram
Celery                         0.6150897114958084 / hectogram
Potato, Russet                              0.395 / hectogram
Carrot                        0.21935995087395316 / hectogram
Cucumber                       0.5489510328403452 / hectogram
Avocado                        0.3919329105508934 / hectogram
Lettuce, Romaine               1.7566433050891044 / hectogram
Mushrooms                      1.9341889135686592 / hectogram
Spinach                        0.7319347104537935 / hectogram
Rice, White                   0.21972738797759467 / hectogram
Tortillas, corn                0.3791950909579894 / hectogram
Oats, rolled                   0.5859397012735857 / hectogram
Soy milk, unsweetened          0.2744755164201726 / hectogram
Peanut butter, creamy          0.5048585804033696 / hectogram
Rice, Brown                   0.36266042129412357 / hectogram
Spaghetti, dry                0.21825763956302877 / hectogram
Name: FDC Price, dtype: object
```

0.11 Dietary Requirements

Next, we will create a function that takes as arguments the characteristics of a person (e.g., age, sex, activity level) and returns a pandas.Series of Dietary Reference Intakes (DRI's) or "Recommended Daily Allowances" (RDA) of a variety of nutrients appropriate for our population of interest.

Our data for this is based on US government recommendations available at https://www.dietaryguidelines.gov/sites/default/files/2021-03/Dietary_Guidelines_for_Americans-2020-2025.pdf

These data have been put into a google sheets that we will access using the read_sheets function defined earlier. You can access the spreadsheet here: https://docs.google.com/spreadsheets/d/1swR8k5x6GaRZd5DvfDF55oeAZZvOC4edB3ofOKov8nE/edit#gid=18

Note that we've tweaked the nutrient labels to match those in the FDC data.

Also note that the last two rows refer to the maximum requirements which involve multiplying the constraint by -1 to make <=.

[11]:
```
RDIs = read_sheets('https://docs.google.com/spreadsheets/d/
↪1swR8k5x6GaRZd5DvfDF55oeAZZvOC4edB3ofOKov8nE/edit#gid=188168169')
```

```
def recommended_diet(age, sex, activity_level):
    if activity_level == 'high' or activity_level == 'High':
        if sex == 'Female' or sex == 'F':
            if age in range(1,4):
                agerange = '1-3'
            if age in range(4,9):
                agerange = '4-8'
            if age in range(9,14):
                agerange = '9-13'
            if age in range(14,19):
                agerange = '14-18'
            if age in range(19,31):
                agerange = '19-30'
            if age in range(31,51):
                agerange = '31-50'
            if age in range(51,100000000):
                agerange = '51+'
            group = 'F ' + agerange
        if sex == 'Male' or sex == 'M':
            if age in range(1,4):
                agerange = '1-3'
            if age in range(4,9):
                agerange = '4-8'
            if age in range(9,14):
                agerange = '9-13'
            if age in range(14,19):
                agerange = '14-18'
            if age in range(19,31):
                agerange = '19-30'
            if age in range(31,51):
                agerange = '31-50'
            if age in range(51,100000000):
                agerange = '51+'
            group = 'M ' + agerange
        bmin = RDIs['high_activity'].set_index('Nutrition')[[group]]
        bmax = RDIs['high_max'].set_index('Nutrition')[[group]]
    if activity_level == 'moderate' or activity_level == 'Moderate':
        if sex == 'Female' or sex == 'F':
            if age in range(1,4):
                agerange = '1-3'
            if age in range(4,9):
                agerange = '4-8'
            if age in range(9,14):
                agerange = '9-13'
            if age in range(14,19):
                agerange = '14-18'
            if age in range(19,31):
```

```
                agerange = '19-30'
            if age in range(31,51):
                agerange = '31-50'
            if age in range(51,100000000):
                agerange = '51+'
            group = 'F ' + agerange
        if sex == 'Male' or sex == 'M':
            if age in range(14,19):
                agerange = '14-18'
            if age in range(19,31):
                agerange = '19-30'
            if age in range(31,51):
                agerange = '31-50'
            if age in range(51,100000000):
                agerange = '51+'
            group = 'M ' + agerange
        bmin = RDIs['moderate_activity'].set_index('Nutrition')[[group]]
        bmax = RDIs['moderate_max'].set_index('Nutrition')[[group]]
    b = pd.concat([bmin,-bmax])
    return b.squeeze()
```

Key available for students@eep153.iam.gserviceaccount.com.

Let's find our the recommended daily allowance of nutrient intake for a Male aged 52 years old with high activity levels.

[12]:
```
recommended = recommended_diet(52, 'M' , 'High')
recommended
```

[12]:
```
Nutrition
Energy                          2600.0
Protein                           56.0
Fiber, total dietary              28.0
Folate, DFE                      400.0
Calcium, Ca                     1000.0
Carbohydrate, by difference      130.0
Iron, Fe                           8.0
Magnesium, Mg                    420.0
Niacin                            16.0
Phosphorus, P                    700.0
Potassium, K                    4700.0
Riboflavin                         1.3
Thiamin                            1.2
Vitamin A, RAE                   900.0
Vitamin B-12                       2.4
Vitamin B-6                        1.7
Vitamin C, total ascorbic acid    90.0
Vitamin E (alpha-tocopherol)      15.0
```

```
Vitamin K (phylloquinone)          120.0
Zinc, Zn                            11.0
Sodium, Na                       -2300.0
Energy                           -3500.0
Name: M 51+, dtype: float64
```

For this code demonstration, we will use nutritional information based on high activity levels.

[13]:
```
from eep153_tools.sheets import read_sheets

DRI_url = "https://docs.google.com/spreadsheets/d/
↪1swR8k5x6GaRZd5DvfDF55oeAZZvOC4edB3ofOKov8nE/edit#gid=188168169"

DRIs = read_sheets(DRI_url)

# Define *minimums*
diet_min = DRIs['moderate_activity'].set_index('Nutrition')

# Define *maximums*
diet_max = DRIs['moderate_max'].set_index('Nutrition')
```

```
Key available for students@eep153.iam.gserviceaccount.com.
```

0.12 Using solve_subsistence_problem to analyze diet

Let's choose a particular group (type of person with particular dietary requirements) and solve the subsistence problem for them:

[14]:
```
group = 'F 19-30'
tol = 1e-6

result =␣
↪solve_subsistence_problem(FoodNutrients,Prices,diet_min[group],diet_max[group],tol=tol)

print("Cost of diet for %s is $%4.2f per day.\n" % (group,result.fun))

# Put back into nice series
diet = result.diet

print("\nDiet (in 100s of grams or milliliters):")
print(diet[diet >= tol])  # Drop items with quantities less than precision of␣
↪calculation.
print()

tab = pd.DataFrame({"Outcome":np.abs(result.A).dot(diet),"Recommendation":np.
↪abs(result.b)})
print("\nWith the following nutritional outcomes of interest:")
print(tab)
```

```python
print()

print("\nConstraining nutrients are:")
excess = tab.diff(axis=1).iloc[:,1]
print(excess.loc[np.abs(excess) < tol*100].index.tolist())
```

```
Cost of diet for F 19-30 is $4.69 per day.


Diet (in 100s of grams or milliliters):
 Banana,raw               1.589644
Oranges, Navel            0.338834
Black beans, canned       5.381961
Kale                      0.168344
Carrot                    0.359735
Avocado                   0.304517
Tortillas, corn           0.158310
Soy milk, unsweetened     6.105317
Peanut butter, creamy     1.773057
dtype: float64


With the following nutritional outcomes of interest:
                                    Outcome  Recommendation
Nutrition
Energy                          3229.221341          2100.0
Protein                          100.833751            46.0
Fiber, total dietary              55.844313            28.0
Folate, DFE                      400.000002           400.0
Calcium, Ca                     1000.000014          1000.0
Carbohydrate, by difference      192.369320           130.0
Iron, Fe                          18.000000            18.0
Magnesium, Mg                    731.531214           310.0
Niacin                            38.213151            14.0
Phosphorus, P                   1811.717738           700.0
Potassium, K                    4700.000010          4700.0
Riboflavin                         1.314460             1.1
Thiamin                            1.605426             1.1
Vitamin A, RAE                   700.000709           700.0
Vitamin B-12                       2.400000             2.4
Vitamin B-6                        1.824062             1.3
Vitamin C, total ascorbic acid    75.000001            75.0
Vitamin E (alpha-tocopherol)      15.000000            15.0
Vitamin K (phylloquinone)         90.000001            90.0
Zinc, Zn                          10.967731             8.0
Sodium, Na                      1499.705430          2300.0
Energy                          3229.221341          3300.0
```

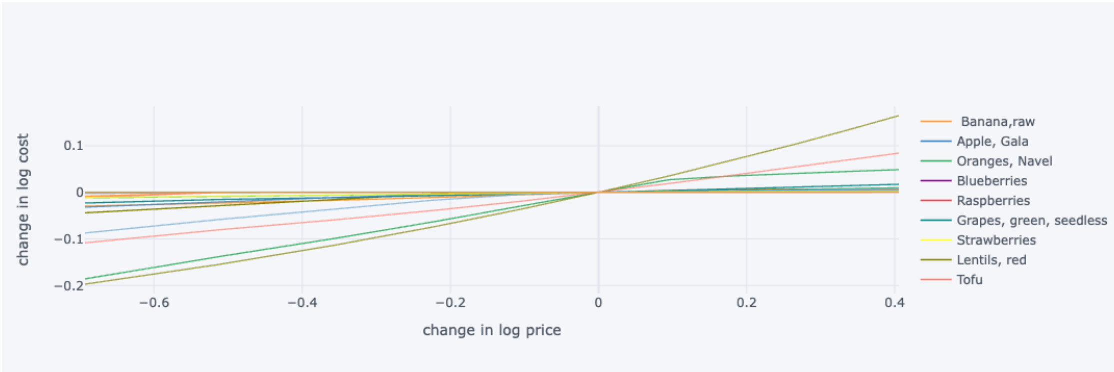
```
Constraining nutrients are:
['Folate, DFE', 'Calcium, Ca', 'Iron, Fe', 'Potassium, K', 'Vitamin B-12',
'Vitamin C, total ascorbic acid', 'Vitamin E (alpha-tocopherol)', 'Vitamin K
(phylloquinone)']
```

0.13 Effects of Price Changes on Subsistence Diet Cost

As prices change, we should expect the minimum cost diet to also change. The code below creates a graph which changes prices away from the 'base' case one food at a time, and plots changes in total diet cost.

[15]:
```
import cufflinks as cf
cf.go_offline()

scale = [.5,.6,.7,.8,.9,1.,1.1,1.2,1.3,1.4,1.5]

cost0 =␣
 ↪solve_subsistence_problem(FoodNutrients,Prices,diet_min[group],diet_max[group],tol=tol).
 ↪fun

Price_response={}
for s in scale:
    cost = {}
    for i,p in enumerate(Prices):
        my_p = Prices.copy()
        my_p[i] = p*s
        result =␣
 ↪solve_subsistence_problem(FoodNutrients,my_p,diet_min[group],diet_max[group],tol=tol)
        cost[Prices.index[i]] = np.log(result.fun/cost0)
    Price_response[np.log(s)] = cost

Price_response = pd.DataFrame(Price_response).T
Price_response.iplot(xTitle='change in log price',yTitle='change in log cost')
```

```
/opt/conda/lib/python3.9/site-packages/geopandas/_compat.py:111: UserWarning:

The Shapely GEOS version (3.10.3-CAPI-1.16.1) is incompatible with the GEOS
version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions between both
will be slow.
```



0.14 Effects of Price Changes on Subsistence Diet Composition

The code below creates a graph which changes prices just for *one* food, and traces out the effects of this change on all the foods consumed.

[16]:
```
import cufflinks as cf
cf.go_offline()

ReferenceGood = 'Kale'

scale = [0.5,0.75,0.9,1.,1.1,1.2,1.3,1.4,1.5,2,4]

cost0 = solve_subsistence_problem(FoodNutrients,Prices,diet_min[group],diet_max[group],tol=tol).fun

my_p = Prices.copy()

diet = {}
for s in scale:

    my_p[ReferenceGood] = Prices[ReferenceGood]*s
    result = solve_subsistence_problem(FoodNutrients,my_p,diet_min[group],diet_max[group],tol=tol)
    diet[my_p[ReferenceGood]] = result.diet

Diet_response = pd.DataFrame(diet).T
Diet_response.index.name = '%s Price' % ReferenceGood

Diet_response.reset_index(inplace=True)

# Get rid of units for index (cufflinks chokes)
Diet_response['%s Price' % ReferenceGood] = Diet_response['%s Price' % ReferenceGood].apply(lambda x: x.magnitude)
```

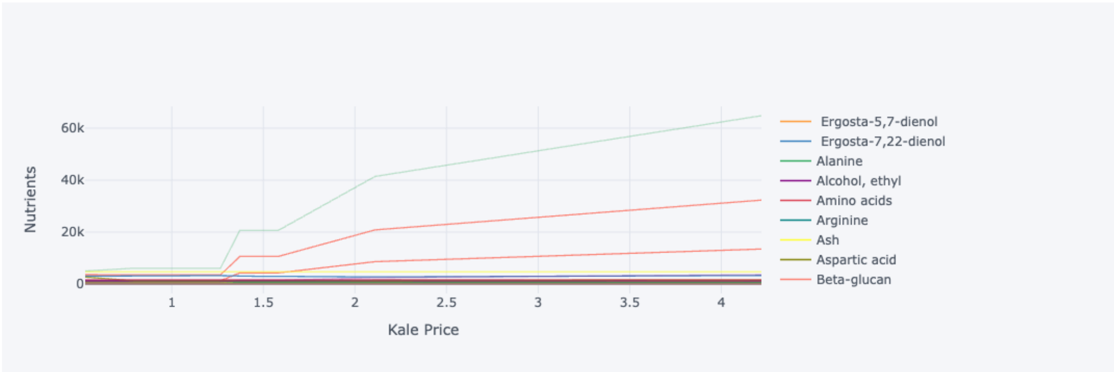
```
Diet_response = Diet_response.set_index('%s Price' % ReferenceGood)

# Just look at goods consumed in quantities greater than error tolerance
Diet_response.loc[:,(Diet_response>tol).sum()>0].iplot(xTitle='%s Price' %␣
    ↪ReferenceGood,yTitle='Hectograms')
```

```
/opt/conda/lib/python3.9/site-packages/pandas/core/dtypes/cast.py:1990:
UnitStrippedWarning:

The unit of the quantity is stripped when downcasting to ndarray.
```

0.15 Effects of Price Changes on Subsistence Diet Nutrition

The code below creates a graph which uses the food price changes described above, but maps into nutrients.

[17]:
```
# Matrix product maps quantities of food into quantities of nutrients
NutrientResponse = Diet_response@FoodNutrients.T

# Drop columns of missing nutrients
NutrientResponse = NutrientResponse.loc[:,NutrientResponse.count()>0]
NutrientResponse.iplot(xTitle='%s Price' % ReferenceGood,yTitle='Nutrients')
```



0.16 Adding Constraint on Total Weight

At least at some prices the minimum cost subistence diet involves eating unreasonable amounts of food (e.g., 10 kilograms of cabbage per day). We can easily add an additional constraint of the form

$$\sum x_i \leq \text{max weight}$$

to our linear programming problem since it's just another linear inequality, and this may give us more realistic results.

0.16.1 Price Changes and Subsistence Diet Composition with Weight Constraint

Re-do our analysis of changing prices, but with a constraint that total weight of diet must be less that 15 hectograms (1.5 kg).

[18]:
```
import cufflinks as cf
cf.go_offline()

ReferenceGood = 'Kale'

scale = [0.5,0.75,0.9,1.,1.1,1.2,1.3,1.4,1.5,2,4]

cost0 = solve_subsistence_problem(FoodNutrients,Prices,diet_min[group],diet_max[group],max_weight=15,tol=tol).fun

my_p = Prices.copy()

diet = {}
for s in scale:

    my_p[ReferenceGood] = Prices[ReferenceGood]*s
    result = solve_subsistence_problem(FoodNutrients,my_p,diet_min[group],diet_max[group],max_weight=15,tol=tol)
```

```
    diet[my_p[ReferenceGood]] = result.diet

Diet_response = pd.DataFrame(diet).T
Diet_response.index.name = '%s Price' % ReferenceGood

Diet_response.reset_index(inplace=True)

# Get rid of units for index (cufflinks chokes)
Diet_response['%s Price' % ReferenceGood] = Diet_response['%s Price' %␣
 ↪ReferenceGood].apply(lambda x: x.magnitude)

Diet_response = Diet_response.set_index('%s Price' % ReferenceGood)

# Just look at goods consumed in quantities greater than error tolerance
Diet_response.loc[:,(Diet_response>tol).sum()>0].iplot(xTitle='%s Price' %␣
 ↪ReferenceGood,yTitle='Hectograms')
```