Project2

March 2, 2023

[1]:
```
!pip install -r requirements.txt
```

```
Collecting pint>=0.18
  Using cached Pint-0.20.1-py3-none-any.whl (269 kB)
Requirement already satisfied: requests>=2.26.0 in
/opt/conda/lib/python3.9/site-packages (from -r requirements.txt (line 6))
(2.26.0)
Collecting python-gnupg
  Using cached python_gnupg-0.5.0-py2.py3-none-any.whl (18 kB)
Collecting eep153_tools
  Using cached eep153_tools-0.11-py2.py3-none-any.whl (4.4 kB)
Collecting fooddatacentral
  Using cached fooddatacentral-1.0.9-py3-none-any.whl
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: 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: charset-normalizer~=2.0.0 in
/opt/conda/lib/python3.9/site-packages (from requests>=2.26.0->-r
requirements.txt (line 6)) (2.0.0)
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)
Installing collected packages: python-gnupg, fooddatacentral, eep153_tools, pint
  Attempting uninstall: pint
    Found existing installation: Pint 0.17
    Uninstalling Pint-0.17:
      Successfully uninstalled Pint-0.17
Successfully installed eep153_tools-0.11 fooddatacentral-1.0.9 pint-0.20.1
python-gnupg-0.5.0
```

[2]:
```
# API key for Gov;
apikey = "bwFohFv0W79JagEjhjfy121CHf29UEljzOOYel1N"
```

[3]:
```
# read in dietary requirements (max and min)
import pandas as pd
dri_max = pd.read_csv("Dietary Requirements Max.csv").set_index('Nutrition')
```

```
dri_min = pd.read_csv("Dietary Requirements Min.csv").set_index('Nutrition')

# convert kcal to kJ
temp = dri_max.loc['Energy']
temp.iloc[1:] = temp.iloc[1:] * 4.184
dri_max.loc['Energy'] = temp

temp = dri_min.loc['Energy']
temp.iloc[1:] = temp.iloc[1:] * 4.184
dri_min.loc['Energy'] = temp
```

```
/opt/conda/lib/python3.9/site-packages/pandas/core/indexing.py:1965:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj._check_is_chained_assignment_possible()
/opt/conda/lib/python3.9/site-packages/pandas/core/indexing.py:1732:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

[3]:

Nutrition	Source	C 1-3	F 4-8	M 4-8	F 9-13	M 9-13
Energy	---	4184.0	5020.8	5857.6	6694.4	7531.2
Protein	RDA	13.0	19.0	19.0	34.0	34.0
Fiber, total dietary	---	14.0	16.8	19.6	22.4	25.2
Folate, DFE	RDA	150.0	200.0	200.0	300.0	300.0
Calcium, Ca	RDA	700.0	1000.0	1000.0	1300.0	1300.0
Carbohydrate, by difference	RDA	130.0	130.0	130.0	130.0	130.0
Iron, Fe	RDA	7.0	10.0	10.0	8.0	8.0
Magnesium, Mg	RDA	80.0	130.0	130.0	240.0	240.0
Niacin	RDA	6.0	8.0	8.0	12.0	12.0
Phosphorus, P	RDA	460.0	500.0	500.0	1250.0	1250.0
Potassium, K	AI	3000.0	3800.0	3800.0	4500.0	4500.0
Riboflavin	RDA	0.5	0.6	0.6	0.9	0.9
Thiamin	RDA	0.5	0.6	0.6	0.9	0.9
Vitamin A, RAE	RDA	300.0	400.0	400.0	600.0	600.0
Vitamin B-12	RDA	0.9	1.2	1.2	1.8	1.8
Vitamin B-6	RDA	0.5	0.6	0.6	1.0	1.0
Vitamin C, total ascorbic acid	RDA	15.0	25.0	25.0	45.0	45.0
Vitamin E (alpha-tocopherol)	RDA	6.0	7.0	7.0	11.0	11.0

Vitamin K (phylloquinone)	AI	30.0	55.0	55.0	60.0	60.0
Zinc, Zn	RDA	3.0	5.0	5.0	8.0	8.0

	F 14-18	M 14-18	F 19-30	M 19-30	F 31-50	\
Nutrition						
Energy	7531.2	9204.8	8368.0	10041.6	7531.2	
Protein	46.0	52.0	46.0	56.0	46.0	
Fiber, total dietary	25.2	30.8	28.0	33.6	25.2	
Folate, DFE	400.0	400.0	400.0	400.0	400.0	
Calcium, Ca	1300.0	1300.0	1000.0	1000.0	1000.0	
Carbohydrate, by difference	130.0	130.0	130.0	130.0	130.0	
Iron, Fe	15.0	11.0	18.0	8.0	18.0	
Magnesium, Mg	360.0	410.0	310.0	400.0	320.0	
Niacin	14.0	16.0	14.0	16.0	14.0	
Phosphorus, P	1250.0	1250.0	700.0	700.0	700.0	
Potassium, K	4700.0	4700.0	4700.0	4700.0	4700.0	
Riboflavin	1.0	1.3	1.1	1.3	1.1	
Thiamin	1.0	1.2	1.1	1.2	1.1	
Vitamin A, RAE	700.0	900.0	700.0	900.0	700.0	
Vitamin B-12	2.4	2.4	2.4	2.4	2.4	
Vitamin B-6	1.2	1.3	1.3	1.3	1.3	
Vitamin C, total ascorbic acid	65.0	75.0	75.0	90.0	75.0	
Vitamin E (alpha-tocopherol)	15.0	15.0	15.0	15.0	15.0	
Vitamin K (phylloquinone)	75.0	75.0	90.0	120.0	90.0	
Zinc, Zn	9.0	11.0	8.0	11.0	8.0	

	M 31-50	F 51+	M 51+
Nutrition			
Energy	9204.8	6694.4	8368.0
Protein	56.0	46.0	56.0
Fiber, total dietary	30.8	22.4	28.0
Folate, DFE	400.0	400.0	400.0
Calcium, Ca	1000.0	1200.0	1000.0
Carbohydrate, by difference	130.0	130.0	130.0
Iron, Fe	8.0	8.0	8.0
Magnesium, Mg	420.0	320.0	420.0
Niacin	16.0	14.0	16.0
Phosphorus, P	700.0	700.0	700.0
Potassium, K	4700.0	4700.0	4700.0
Riboflavin	1.3	1.1	1.3
Thiamin	1.2	1.1	1.2
Vitamin A, RAE	900.0	700.0	900.0
Vitamin B-12	2.4	2.4	2.4
Vitamin B-6	1.3	1.5	1.7
Vitamin C, total ascorbic acid	90.0	75.0	90.0
Vitamin E (alpha-tocopherol)	15.0	15.0	15.0
Vitamin K (phylloquinone)	120.0	90.0	120.0

```
Zinc, Zn                          11.0     8.0    11.0
```

[7]:
```
import re
# function for age-sex specific DRI
def dri(age,sex,dietmin,dietmax):
    if age <= 3:
        index = 2

    else:
        for i, j in enumerate(list(dietmax)[3:]):
            if j[0] == sex:
                interval = re.findall(r'\d+', j)
                if len(interval) == 1:
                    if age >= int(interval[0]):
                        index = i + 3
                        break
                else:
                    if age >= int(interval[0]) and age <= int(interval[1]):
                        index = i + 3
                        break

    df = pd.DataFrame({'Nutrition': dietmin.iloc[:, 0],
                       'Max/Min': 'Minimum',
                       'Age & Sex': list(dietmin)[index],
                      'Intake': dietmin.iloc[:, index]})
    df_max = pd.DataFrame({'Nutrition': dietmax.iloc[:, 0],
                       'Max/Min': 'Maximum',
                       'Age & Sex': list(dietmax)[index],
                      'Intake': dietmax.iloc[:, index]})

    df = pd.concat([df, df_max], axis=0)

    return df
```

[8]:
```
# test
dri(10, 'F', dri_min, dri_max)
```

[8]:

	Nutrition	Max/Min	Age & Sex	Intake
Nutrition				
Energy	---	Minimum	F 9-13	1600.0
Protein	RDA	Minimum	F 9-13	34.0
Fiber, total dietary	---	Minimum	F 9-13	22.4
Folate, DFE	RDA	Minimum	F 9-13	300.0
Calcium, Ca	RDA	Minimum	F 9-13	1300.0
Carbohydrate, by difference	RDA	Minimum	F 9-13	130.0
Iron, Fe	RDA	Minimum	F 9-13	8.0

```
Magnesium, Mg                          RDA  Minimum    F 9-13    240.0
Niacin                                 RDA  Minimum    F 9-13     12.0
Phosphorus, P                          RDA  Minimum    F 9-13   1250.0
Potassium, K                            AI  Minimum    F 9-13   4500.0
Riboflavin                             RDA  Minimum    F 9-13      0.9
Thiamin                                RDA  Minimum    F 9-13      0.9
Vitamin A, RAE                         RDA  Minimum    F 9-13    600.0
Vitamin B-12                           RDA  Minimum    F 9-13      1.8
Vitamin B-6                            RDA  Minimum    F 9-13      1.0
Vitamin C, total ascorbic acid         RDA  Minimum    F 9-13     45.0
Vitamin E (alpha-tocopherol)           RDA  Minimum    F 9-13     11.0
Vitamin K (phylloquinone)               AI  Minimum    F 9-13     60.0
Zinc, Zn                               RDA  Minimum    F 9-13      8.0
Sodium, Na                              UL  Maximum    F 9-13   2200.0
Energy                                 NaN  Maximum    F 9-13  11715.2
```

[90]:
```
# get food list from five restaurants
food_list_total = pd.read_csv("FoodList.csv")
food_list_total = food_list_total.astype({"FDC": str})
food_list_total
```

[90]:
```
    Restaurant              Dish  Ingredients  Dish_Price  Quantity  Unit  \
0      Chipotle   Chicken burrito      chicken        8.85  132.0000  gram
1      Chipotle   Chicken burrito   white rice        8.85  120.0000  gram
2      Chipotle   Chicken burrito  black beans        8.85   60.0000  gram
3      Chipotle   Chicken burrito      lettuce        8.85   60.0000  gram
4      Chipotle   Chicken burrito        salsa        8.85   40.0000  gram
..          …                 …           …           …        …     …
364    Poke Bar  Wazzup Poke Bowl     cucumber       15.95   28.3500  gram
365    Poke Bar  Wazzup Poke Bowl  green onion       15.95    7.0875  gram
366    Poke Bar  Wazzup Poke Bowl        ponzu       15.95   35.4375  gram
367    Poke Bar  Wazzup Poke Bowl       wasabi       15.95  452.1825  gram
368    Poke Bar  Wazzup Poke Bowl         mayo       15.95   35.4375  gram

          FDC  Calorie/100g  Ingredient_Price/100 gm
0      331960         237.0                    1.320
1      790214         359.0                    0.274
2      747444         180.0                    0.440
3     2346389          21.0                    1.110
4      746777          34.0                    0.590
..        …             …                        …
364    168409          15.0                    0.880
365    170006          27.0                    0.440
366   2451144          33.0                    1.900
367    171831         292.0                    8.780
368    171002         334.0                    0.560
```

```
[369 rows x 9 columns]
```

[92]:
```
# split food_list by restaurant
grouped = food_list_total.groupby(food_list_total.Restaurant)
restaurants = ['Chipotle', 'Thai Basil', 'Ttoust', 'IB', 'Poke Bar']

food_list_res = []

for r in restaurants:
    food_list_res.append(grouped.get_group(r))
```

[70]:
```
import fooddatacentral as fdc
import warnings
from collections import defaultdict

# get nutritional information for ingredients
ing_res = []
for i in range(5):
    L = []
    D = {}
    items = []
    count = 0
    food_list = food_list_res[i]
    for food in food_list.Ingredients.tolist():
        try:
            FDC = food_list.iloc[count,:].FDC
            count+=1
            temp = fdc.nutrients(apikey,FDC)
            key = temp.Units
            # convert units if necessary
            if 'Energy' in key.index:
                if key['Energy'] != 'kJ':
                    temp.Quantity['Energy'] = temp.Quantity['Energy']*4.184

            L.append(temp.Quantity)
            D[food] = temp.Quantity
            items.append(food)
        except AttributeError:
            warnings.warn("Couldn't find FDC Code %s for food %s." % (food,FDC))

    FoodNutrients_Ing = pd.DataFrame(L,dtype=float)
    FoodNutrients_Ing.index = items
    FoodNutrients_Ing = FoodNutrients_Ing.fillna(0)
    FoodNutrients_Ing = FoodNutrients_Ing.transpose()

    #FoodNutrients_Ing
```

```
    FoodNutrients_Ing_d = pd.DataFrame(D,dtype=float)
    FoodNutrients_Ing_d = FoodNutrients_Ing_d.fillna(0)
    ing_res.append(FoodNutrients_Ing_d)
```

```
/tmp/ipykernel_29/329324121.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  temp.Quantity['Energy'] = temp.Quantity['Energy']*4.184
```

[75]:
```
ing_res[4]
```

[75]:

	salmon	white rice	sweet onion	green onion	\
Alanine	1.271	0.332	0.017	0.00	
Alcohol, ethyl	0.000	0.000	0.000	0.00	
Amino acids	0.000	0.000	0.000	0.00	
Arginine	1.221	0.516	0.111	0.00	
Ash	1.130	0.610	0.340	0.51	
…	…	…	…	…	
Vitamin K (Menaquinone-4)	0.000	0.000	0.000	0.00	
Vitamin K (phylloquinone)	0.500	0.000	0.300	156.30	
Vitamins and Other Components	0.000	0.000	0.000	0.00	
Water	64.890	11.890	91.240	92.32	
Zinc, Zn	0.360	0.800	0.130	0.20	

	kale	shoyu	spicy mayo	ahi tuna	cucumber	\
Alanine	0.147	0.0	0.0	1.331	0.024	
Alcohol, ethyl	0.000	0.0	0.0	0.000	0.000	
Amino acids	0.000	0.0	0.0	0.000	0.000	
Arginine	0.163	0.0	0.0	1.316	0.044	
Ash	1.540	0.0	0.0	1.300	0.380	
…	…	…	…	…	…	
Vitamin K (Menaquinone-4)	0.000	0.0	0.0	0.000	0.000	
Vitamin K (phylloquinone)	389.600	0.0	0.0	0.000	16.400	
Vitamins and Other Components	0.000	0.0	0.0	0.000	0.000	
Water	89.630	0.0	0.0	70.580	95.230	
Zinc, Zn	0.390	0.0	0.0	0.820	0.200	

	sweet chili	house dressing	edamame	ponzu	\
Alanine	0.0	0.0	0.00	0.0	
Alcohol, ethyl	0.0	0.0	0.00	0.0	
Amino acids	0.0	0.0	0.00	0.0	
Arginine	0.0	0.0	0.00	0.0	
Ash	0.0	0.0	0.00	0.0	
…	…	…	…	…	
Vitamin K (Menaquinone-4)	0.0	0.0	0.00	0.0	

```
Vitamin K (phylloquinone)              0.0             0.0    28.10    0.0
Vitamins and Other Components          0.0             0.0     0.00    0.0
Water                                  0.0             0.0    70.80    0.0
Zinc, Zn                               0.0             0.0     1.33    0.0

                                 tofu  wasabi   mayo
Alanine                          0.00    0.00    0.0
Alcohol, ethyl                   0.00    0.00    0.0
Amino acids                      0.00    0.00    0.0
Arginine                         0.00    0.00    0.0
Ash                              1.05    9.04    2.1
…                                …       …       …
Vitamin K (Menaquinone-4)        0.00    0.00    0.0
Vitamin K (phylloquinone)        0.00    3.50  155.1
Vitamins and Other Components    0.00    0.00    0.0
Water                           82.87   31.70   55.4
Zinc, Zn                         0.00    0.61    0.0

[144 rows x 16 columns]
```

[7]:
```
# construct table of nutritional information for dishes

dishes = food_list.groupby('Dish',sort=False)['Ingredients'].count()

FoodNutrients = pd.DataFrame(columns=dishes.keys().tolist(),
                  index=FoodNutrients_Ing.index, )
start = 0
end = 0
for i, column in enumerate(FoodNutrients):

    FoodNutrients[column] = FoodNutrients_Ing.iloc[:, start:start+dishes[i]].
 ↪sum(axis=1)
    start = start+dishes[i]
FoodNutrients
```

[7]:
```
                              Chicken burrito  steak burrito   \
Proximates                       0.000000e+00    0.000000e+00
Water                            3.795900e+02    3.856900e+02
Energy                           7.384437e+16    7.384437e+16
Nitrogen                         6.626300e+00    1.496300e+00
Protein                          9.385688e+01    8.445687e+01
…                                         …               …
Delta-5-avenasterol              0.000000e+00    0.000000e+00
Delta-7-Stigmastenol             0.000000e+00    0.000000e+00
Ergothioneine                    0.000000e+00    0.000000e+00
MUFA 18:1-11 t (18:1t n-7)       0.000000e+00    0.000000e+00
Sugars, added                    0.000000e+00    0.000000e+00
```

```
                           barbacoa burrito   carnitas burrito   \
Proximates                     0.000000e+00       0.000000e+00
Water                          3.707900e+02       4.275900e+02
Energy                         7.384437e+16       7.387602e+16
Nitrogen                       1.496300e+00       1.496300e+00
Protein                        8.765687e+01       9.775688e+01
…                                     …                  …
Delta-5-avenasterol            0.000000e+00       0.000000e+00
Delta-7-Stigmastenol           0.000000e+00       0.000000e+00
Ergothioneine                  0.000000e+00       0.000000e+00
MUFA 18:1-11 t (18:1t n-7)     0.000000e+00       0.000000e+00
Sugars, added                  0.000000e+00       0.000000e+00

                           sofritas burrito   veggie burrito   \
Proximates                     0.000000e+00     0.000000e+00
Water                          3.736900e+02     4.466900e+02
Energy                         7.387602e+16     7.387602e+16
Nitrogen                       1.496300e+00     1.496300e+00
Protein                        7.315687e+01     7.510688e+01
…                                     …                …
Delta-5-avenasterol            0.000000e+00     0.000000e+00
Delta-7-Stigmastenol           0.000000e+00     0.000000e+00
Ergothioneine                  0.000000e+00     0.000000e+00
MUFA 18:1-11 t (18:1t n-7)     0.000000e+00     0.000000e+00
Sugars, added                  0.000000e+00     0.000000e+00

                           chicken quesadilla   steak qusadilla   \
Proximates                       0.000000e+00      0.000000e+00
Water                            4.593900e+02      4.654900e+02
Energy                           7.387602e+16      7.387602e+16
Nitrogen                         5.516300e+00      3.863000e-01
Protein                          1.065169e+02      9.711687e+01
…                                       …                 …
Delta-5-avenasterol              0.000000e+00      0.000000e+00
Delta-7-Stigmastenol             0.000000e+00      0.000000e+00
Ergothioneine                    0.000000e+00      0.000000e+00
MUFA 18:1-11 t (18:1t n-7)       0.000000e+00      0.000000e+00
Sugars, added                    0.000000e+00      0.000000e+00

                           barbacoa quesadilla    carnita quesadilla  …  \
Proximates                       0.000000e+00        0.000000e+00  …
Water                            4.505900e+02        3.885900e+02  …
Energy                           7.387602e+16        7.384437e+16  …
Nitrogen                         3.863000e-01        3.863000e-01  …
Protein                          1.003169e+02        8.761687e+01  …
…                                       …                   …  …
```

Delta-5-avenasterol	0.000000e+00	0.000000e+00	…
Delta-7-Stigmastenol	0.000000e+00	0.000000e+00	…
Ergothioneine	0.000000e+00	0.000000e+00	…
MUFA 18:1-11 t (18:1t n-7)	0.000000e+00	0.000000e+00	…
Sugars, added	0.000000e+00	0.000000e+00	…

	Veggie Combo	IB's Original(chicken)	\
Proximates	0.000000	0.000000	
Water	599.720000	497.990000	
Energy	129317.731131	129265.731131	
Nitrogen	0.080000	5.130000	
Protein	50.160000	79.820000	
…	…	…	
Delta-5-avenasterol	0.000000	0.000000	
Delta-7-Stigmastenol	0.000000	0.000000	
Ergothioneine	0.000000	0.000000	
MUFA 18:1-11 t (18:1t n-7)	0.000000	0.000000	
Sugars, added	0.000000	0.000000	

	IB's Original(beef)	IB's Original(lamb)	\
Proximates	0.000000	0.000000	
Water	506.390000	492.160000	
Energy	129082.731131	129750.731131	
Nitrogen	0.000000	0.000000	
Protein	71.120000	64.280000	
…	…	…	
Delta-5-avenasterol	0.000000	0.000000	
Delta-7-Stigmastenol	0.000000	0.000000	
Ergothioneine	0.000000	0.000000	
MUFA 18:1-11 t (18:1t n-7)	0.000000	0.000000	
Sugars, added	0.000000	0.000000	

	Original Salmon Poke Bowl	Firecracker Poke Bowl	\
Proximates	0.000000	0.000000	
Water	349.970000	349.370000	
Energy	14278.841536	50999.296902	
Nitrogen	0.000000	0.000000	
Protein	42.170000	30.610000	
…	…	…	
Delta-5-avenasterol	0.000000	0.000000	
Delta-7-Stigmastenol	0.000000	0.000000	
Ergothioneine	0.000000	0.000000	
MUFA 18:1-11 t (18:1t n-7)	0.000000	0.000000	
Sugars, added	0.000000	13.300000	

	Sunset House Poke Bowl	The O.G. Poke Bowl	\
Proximates	0.000000	0.00000	

```
Water                                424.540000          336.83000
Energy                             42197.319302        44274.83239
Nitrogen                               0.000000            0.00000
Protein                               56.240000           44.55000
…                                           …                   …
Delta-5-avenasterol                    0.000000            0.00000
Delta-7-Stigmastenol                   0.000000            0.00000
Ergothioneine                          0.000000            0.00000
MUFA 18:1-11 t (18:1t n-7)             0.000000            0.00000
Sugars, added                         13.300000           13.30000

                           Goodie Mob Poke Bowl  Wazzup Poke Bowl
Proximates                             0.000000          0.000000
Water                                444.350000        448.360000
Energy                              5629.147776       5466.693248
Nitrogen                               0.000000          0.000000
Protein                               41.900000         33.200000
…                                           …                 …
Delta-5-avenasterol                    0.000000          0.000000
Delta-7-Stigmastenol                   0.000000          0.000000
Ergothioneine                          0.000000          0.000000
MUFA 18:1-11 t (18:1t n-7)             0.000000          0.000000
Sugars, added                          0.000000          0.000000

[213 rows x 46 columns]
```

[8]:
```
# Convert food quantities to FDC units
food_list['FDC Quantity'] = food_list[['Quantity','Unit']].T.apply(lambda x :␣
 ↪fdc.units(x['Quantity'],x['Unit']))

# Now may want to filter df by time or place--need to get a unique set of food␣
 ↪names.
food_list['FDC Price'] = food_list['Dish_Price']/food_list.
 ↪groupby('Dish',sort=False)['FDC Quantity'].sum()

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

# To use minimum price observed
Dish_Prices = food_list.groupby('Dish',sort=False)['Dish_Price'].min()/
 ↪food_list.groupby('Dish',sort=False)['FDC Quantity'].sum()
```

```
/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
```

[9]:
```
Dish_Prices
```

[9]:
```
Dish
Chicken burrito                1.644981412639405 / hectogram
steak burrito                 1.9702602230483268 / hectogram
barbacoa burrito              1.9702602230483268 / hectogram
carnitas burrito               1.765799256505576 / hectogram
sofritas burrito               2.179802955665025 / hectogram
veggie burrito                0.4925650557620817 / hectogram
chicken quesadilla            1.7472118959107805 / hectogram
steak qusadilla               2.0724907063197024 / hectogram
barbacoa quesadilla           2.0724907063197024 / hectogram
carnita quesadilla            1.7472118959107805 / hectogram
sofritas quesadilla           2.3152709359605916 / hectogram
chicken salad                  1.644981412639405 / hectogram
steak salad                   1.9702602230483268 / hectogram
barbacoa salad                1.9702602230483268 / hectogram
carnitas salad                 1.765799256505576 / hectogram
sofritas salad                 2.179802955665025 / hectogram
veggie salad                  1.5102389078498295 / hectogram
chicken taco                   1.682509505703422 / hectogram
steak taco                    1.4763231197771587 / hectogram
barbacoa taco                  1.715210355987055 / hectogram
carnitas taco                 1.5372168284789645 / hectogram
sofritas taco                 1.8209876543209877 / hectogram
kid's meal quesadilla         0.8544303797468354 / hectogram
Pineapple Fried Rice           2.326666666666666 / hectogram
Basil Eggplant Chicken        1.9074074074074074 / hectogram
Pad Kee Mow Chicken            2.653225806451613 / hectogram
Pad Thai Chicken & Prawns      1.852497096399535 / hectogram
Combo Garlic Pepper           1.8911335578002246 / hectogram
Spicy Basil Catfish           1.8994708994708995 / hectogram
Bibim Bop                     3.3369330453563713 / hectogram
Kimchi Soup                   1.6732283464566928 / hectogram
Gaibi Short Rib               2.6473509933774837 / hectogram
Spicy Rice Cake               1.6846543001686343 / hectogram
BBQ Chicken                   1.3934426229508197 / hectogram
Cheeseburger                   2.472527472527472 / hectogram
Avogobble sandwich            3.1055900621118013 / hectogram
Veggie Combo                   3.535353535353536 / hectogram
IB's Original(chicken)        2.2569444444444446 / hectogram
IB's Original(beef)           2.2569444444444446 / hectogram
IB's Original(lamb)           2.2569444444444446 / hectogram
Original Salmon Poke Bowl     3.2149155958679763 / hectogram
Firecracker Poke Bowl          6.082272748939414 / hectogram
Sunset House Poke Bowl        2.6790963298899806 / hectogram
The O.G. Poke Bowl            1.7337757450751807 / hectogram
Goodie Mob Poke Bowl          1.7472367368847699 / hectogram
Wazzup Poke Bowl              1.6819438842358623 / hectogram
```

```
dtype: object
```

[76]:
```
# Convert food quantities to FDC units
ing_prices = []
for r in food_list_res:
    food_list = r
    food_list['FDC Quantity'] = food_list[['Quantity','Unit']].T.apply(lambda x␣
 ↪: fdc.units(x['Quantity'],x['Unit']))

    # Now may want to filter df by time or place--need to get a unique set of␣
 ↪food names.
    food_list['FDC Price'] = food_list['Ingredient_Price/100 gm']

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

    # To use minimum price observed
    Ingredient_Prices = food_list.groupby('Ingredients',sort=False)['FDC␣
 ↪Price'].min()
    ing_prices.append(Ingredient_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
/tmp/ipykernel_29/2842241641.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  food_list['FDC Quantity'] = food_list[['Quantity','Unit']].T.apply(lambda x :
fdc.units(x['Quantity'],x['Unit']))
/tmp/ipykernel_29/2842241641.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  food_list['FDC Price'] = food_list['Ingredient_Price/100 gm']
```

[81]:
```
ing_prices[4]
```

[81]:
```
Ingredients
salmon           2.20
white rice       0.28
sweet onion      0.44
green onion      0.44
```

```
kale              0.88
shoyu             1.69
spicy mayo        1.38
ahi tuna          2.36
cucumber          0.88
sweet chili       1.05
house dressing    3.17
edamame           0.56
ponzu             1.90
tofu              0.74
wasabi            8.78
mayo              0.56
Name: FDC Price, dtype: float64
```

[86]:
```
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

    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)
    #print(Aall)
    # Drop rows of A that we don't have constraints for.
    Amin = Aall.loc[Aall.index.intersection(dietmin.index)]
    #print(dietmin)
    #print(Amin)
    Amin = Amin.reindex(dietmin.index,axis=0)
    #print(dietmin.index)
    #print(Amin)
    idx = Amin.index.to_frame()
    #print(Amin)
    idx['type'] = 'min'
    #print(Amin)
    #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
    #print(p)
    #print(A)
    #print(b)
    # 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', options={'presolve': True,
                                                             'cholesky':False,
                                                             'sym_pos':False,
                                                             'lstsq':True})

    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
```

[99]:
```
# dish-based result
group = 'M 31-50'
tol = 1e-6
#FoodNutrients_t = FoodNutrients.iloc[0:10,]
#print(FoodNutrients_t)
result =␣
 ↪solve_subsistence_problem(FoodNutrients,Dish_Prices,dri_min[group],dri_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)
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 M 31-50 is $6.75 per day.


Diet (in 100s of grams or milliliters):
Basil Eggplant Chicken       1.631341
Pad Thai Chicken & Prawns    1.817563
Cheeseburger                 0.108056
dtype: float64


With the following nutritional outcomes of interest:
                                    Outcome  Recommendation
Nutrition
Energy                         12970.399698          9204.8
Protein                          215.554768            56.0
Fiber, total dietary              34.452112            30.8
Folate, DFE                      728.109424           400.0
Calcium, Ca                     1187.572224          1000.0
Carbohydrate, by difference      269.608350           130.0
Iron, Fe                          23.351720             8.0
Magnesium, Mg                    714.713210           420.0
Niacin                            45.940157            16.0
Phosphorus, P                   2394.471782           700.0
Potassium, K                    5358.507778          4700.0
Riboflavin                         2.308153             1.3
Thiamin                            2.641435             1.2
Vitamin A, RAE                   899.999776           900.0
Vitamin B-12                       3.055109             2.4
Vitamin B-6                        2.978857             1.3
Vitamin C, total ascorbic acid   478.494956            90.0
Vitamin E (alpha-tocopherol)      15.000003            15.0
Vitamin K (phylloquinone)        709.607968           120.0
Zinc, Zn                          24.585825            11.0
Sodium, Na                      1215.458895          2300.0
Energy                         12970.399698         12970.4


Constraining nutrients are:
['Vitamin E (alpha-tocopherol)']
```

```
/tmp/ipykernel_30/3506648488.py:80: OptimizeWarning: Solving system with option
'cholesky':True failed. It is normal for this to happen occasionally, especially
as the solution is approached. However, if you see this frequently, consider
```

```
setting option 'cholesky' to False.
  result = lp(p, -A, -b, method='interior-point')
/tmp/ipykernel_30/3506648488.py:80: OptimizeWarning: Solving system with option
'sym_pos':True failed. It is normal for this to happen occasionally, especially
as the solution is approached. However, if you see this frequently, consider
setting option 'sym_pos' to False.
  result = lp(p, -A, -b, method='interior-point')
/tmp/ipykernel_30/3506648488.py:80: OptimizeWarning: Solving system with option
'sym_pos':False failed. This may happen occasionally, especially as the solution
is approached. However, if you see this frequently, your problem may be
numerically challenging. If you cannot improve the formulation, consider setting
'lstsq' to True. Consider also setting `presolve` to True, if it is not already.
  result = lp(p, -A, -b, method='interior-point')
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=3.32494e-64): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=4.34568e-55): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=4.06036e-54): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=4.55428e-23): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=1.44453e-23): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=4.87971e-24): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=1.95512e-23): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
/opt/conda/lib/python3.9/site-packages/scipy/optimize/_linprog_ip.py:117:
LinAlgWarning: Ill-conditioned matrix (rcond=1.14032e-22): result may not be
accurate.
  return sp.linalg.solve(M, r, sym_pos=sym_pos)
```

[88]:
```
# ingredient-based results
group = 'M 19-30'
tol = 1e-6
#FoodNutrients_t = FoodNutrients.iloc[0:10,]
#print(FoodNutrients_t)
results = []
for i in range(5):
    result =␣
 ↪solve_subsistence_problem(ing_res[i],ing_prices[i],dri_min[group],dri_max[group],tol=tol)

    print("Result for restaurant: " + restaurants[i])
    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)
    print()

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

```
Result for restaurant: Chipotle
Cost of diet for M 19-30 is $0.00 per day.


Diet (in 100s of grams or milliliters):
Series([], dtype: float64)


With the following nutritional outcomes of interest:
                              Outcome  Recommendation
Nutrition
Energy                            NaN         10041.6
Protein                           NaN            56.0
Fiber, total dietary              NaN            33.6
Folate, DFE                       NaN           400.0
Calcium, Ca                       NaN          1000.0
```

```
Carbohydrate, by difference         NaN          130.0
Iron, Fe                            NaN            8.0
Magnesium, Mg                       NaN          400.0
Niacin                              NaN           16.0
Phosphorus, P                       NaN          700.0
Potassium, K                        NaN         4700.0
Riboflavin                          NaN            1.3
Thiamin                             NaN            1.2
Vitamin A, RAE                      NaN          900.0
Vitamin B-12                        NaN            2.4
Vitamin B-6                         NaN            1.3
Vitamin C, total ascorbic acid      NaN           90.0
Vitamin E (alpha-tocopherol)        NaN           15.0
Vitamin K (phylloquinone)           NaN          120.0
Zinc, Zn                            NaN           11.0
Sodium, Na                          NaN         2300.0
Energy                              NaN        12970.4


Constraining nutrients are:
[]
Result for restaurant: Thai Basil
Cost of diet for M 19-30 is $6.69 per day.


Diet (in 100s of grams or milliliters):
egg         4.947712
carrots     3.846580
rice        0.179987
broccoli    1.176471
peanuts     3.770187
tofu        1.708200
dtype: float64


With the following nutritional outcomes of interest:
                                  Outcome  Recommendation
Nutrition
Energy                       12970.399989         10041.6
Protein                        182.869415            56.0
Fiber, total dietary            48.402655            33.6
Folate, DFE                    901.074617           400.0
Calcium, Ca                   1000.000000          1000.0
Carbohydrate, by difference    130.000000           130.0
Iron, Fe                        22.193402             8.0
Magnesium, Mg                  777.823342           400.0
Niacin                          53.041100            16.0
Phosphorus, P                 2591.924648           700.0
```

Potassium, K	4700.000234	4700.0
Riboflavin	3.066529	1.3
Thiamin	3.199720	1.2
Vitamin A, RAE	900.000000	900.0
Vitamin B-12	5.046667	2.4
Vitamin B-6	2.410318	1.3
Vitamin C, total ascorbic acid	109.290786	90.0
Vitamin E (alpha-tocopherol)	24.908895	15.0
Vitamin K (phylloquinone)	120.000000	120.0
Zinc, Zn	24.454724	11.0
Sodium, Na	1108.886424	2300.0
Energy	12970.399989	12970.4

Constraining nutrients are:
['Calcium, Ca', 'Carbohydrate, by difference', 'Vitamin A, RAE', 'Vitamin K (phylloquinone)', 'Energy']
Result for restaurant: Ttoust
Cost of diet for M 19-30 is $0.00 per day.

Diet (in 100s of grams or milliliters):
Series([], dtype: float64)

With the following nutritional outcomes of interest:

Nutrition	Outcome	Recommendation
Energy	NaN	10041.6
Protein	NaN	56.0
Fiber, total dietary	NaN	33.6
Folate, DFE	NaN	400.0
Calcium, Ca	NaN	1000.0
Carbohydrate, by difference	NaN	130.0
Iron, Fe	NaN	8.0
Magnesium, Mg	NaN	400.0
Niacin	NaN	16.0
Phosphorus, P	NaN	700.0
Potassium, K	NaN	4700.0
Riboflavin	NaN	1.3
Thiamin	NaN	1.2
Vitamin A, RAE	NaN	900.0
Vitamin B-12	NaN	2.4
Vitamin B-6	NaN	1.3
Vitamin C, total ascorbic acid	NaN	90.0
Vitamin E (alpha-tocopherol)	NaN	15.0
Vitamin K (phylloquinone)	NaN	120.0
Zinc, Zn	NaN	11.0

```
Sodium, Na                          NaN          2300.0
Energy                              NaN         12970.4


Constraining nutrients are:
[]
Result for restaurant: IB
Cost of diet for M 19-30 is $0.00 per day.


Diet (in 100s of grams or milliliters):
Series([], dtype: float64)


With the following nutritional outcomes of interest:
                                Outcome  Recommendation
Nutrition
Energy                              NaN         10041.6
Protein                             NaN            56.0
Fiber, total dietary                NaN            33.6
Folate, DFE                         NaN           400.0
Calcium, Ca                         NaN          1000.0
Carbohydrate, by difference         NaN           130.0
Iron, Fe                            NaN             8.0
Magnesium, Mg                       NaN           400.0
Niacin                              NaN            16.0
Phosphorus, P                       NaN           700.0
Potassium, K                        NaN          4700.0
Riboflavin                          NaN             1.3
Thiamin                             NaN             1.2
Vitamin A, RAE                      NaN           900.0
Vitamin B-12                        NaN             2.4
Vitamin B-6                         NaN             1.3
Vitamin C, total ascorbic acid      NaN            90.0
Vitamin E (alpha-tocopherol)        NaN            15.0
Vitamin K (phylloquinone)           NaN           120.0
Zinc, Zn                            NaN            11.0
Sodium, Na                          NaN          2300.0
Energy                              NaN         12970.4


Constraining nutrients are:
[]
Result for restaurant: Poke Bar
Cost of diet for M 19-30 is $17.20 per day.


Diet (in 100s of grams or milliliters):
```

```
salmon         1.437845
white rice     6.306049
kale          13.942400
dtype: float64
```

With the following nutritional outcomes of interest:

Nutrition	Outcome	Recommendation
Energy	12970.400170	10041.6
Protein	107.593601	56.0
Fiber, total dietary	72.298361	33.6
Folate, DFE	927.037001	400.0
Calcium, Ca	3617.370820	1000.0
Carbohydrate, by difference	566.929153	130.0
Iron, Fe	25.003825	8.0
Magnesium, Mg	719.632765	400.0
Niacin	45.253694	16.0
Phosphorus, P	1729.907707	700.0
Potassium, K	5853.152909	4700.0
Riboflavin	5.193306	1.3
Thiamin	2.743360	1.2
Vitamin A, RAE	3443.513531	900.0
Vitamin B-12	4.644240	2.4
Vitamin B-6	5.713440	1.3
Vitamin C, total ascorbic acid	1307.827800	90.0
Vitamin E (alpha-tocopherol)	15.000000	15.0
Vitamin K (phylloquinone)	5432.678145	120.0
Zinc, Zn	11.000000	11.0
Sodium, Na	823.780089	2300.0
Energy	12970.400170	12970.4

```
Constraining nutrients are:
['Vitamin E (alpha-tocopherol)', 'Zinc, Zn']
```

```
/tmp/ipykernel_29/2046845037.py:22: UserWarning: Prices have no units.  BE
CAREFUL!  We're assuming prices are per hectogram or deciliter!
  warnings.warn("Prices have no units.  BE CAREFUL!  We're assuming prices are
per hectogram or deciliter!")
/tmp/ipykernel_29/2046845037.py:91: UserWarning: The solution does not satisfy
the constraints within the required tolerance of 3.16E-04, yet no errors were
raised and there is no certificate of infeasibility or unboundedness. Check
whether the slack and constraint residuals are acceptable; if not, consider
enabling presolve, adjusting the tolerance option(s), and/or using a different
method. Please consider submitting a bug report.
  warnings.warn(result.message)
/tmp/ipykernel_29/2046845037.py:22: UserWarning: Prices have no units.  BE
```

```
CAREFUL!  We're assuming prices are per hectogram or deciliter!
  warnings.warn("Prices have no units.  BE CAREFUL!  We're assuming prices are
per hectogram or deciliter!")
/tmp/ipykernel_29/2046845037.py:22: UserWarning: Prices have no units.  BE
CAREFUL!  We're assuming prices are per hectogram or deciliter!
  warnings.warn("Prices have no units.  BE CAREFUL!  We're assuming prices are
per hectogram or deciliter!")
/tmp/ipykernel_29/2046845037.py:91: UserWarning: The solution does not satisfy
the constraints within the required tolerance of 3.16E-04, yet no errors were
raised and there is no certificate of infeasibility or unboundedness. Check
whether the slack and constraint residuals are acceptable; if not, consider
enabling presolve, adjusting the tolerance option(s), and/or using a different
method. Please consider submitting a bug report.
  warnings.warn(result.message)
/tmp/ipykernel_29/2046845037.py:22: UserWarning: Prices have no units.  BE
CAREFUL!  We're assuming prices are per hectogram or deciliter!
  warnings.warn("Prices have no units.  BE CAREFUL!  We're assuming prices are
per hectogram or deciliter!")
/tmp/ipykernel_29/2046845037.py:91: UserWarning: The solution does not satisfy
the constraints within the required tolerance of 3.16E-04, yet no errors were
raised and there is no certificate of infeasibility or unboundedness. Check
whether the slack and constraint residuals are acceptable; if not, consider
enabling presolve, adjusting the tolerance option(s), and/or using a different
method. Please consider submitting a bug report.
  warnings.warn(result.message)
/tmp/ipykernel_29/2046845037.py:22: UserWarning: Prices have no units.  BE
CAREFUL!  We're assuming prices are per hectogram or deciliter!
  warnings.warn("Prices have no units.  BE CAREFUL!  We're assuming prices are
per hectogram or deciliter!")
```

[ ]: