Optimal Diet for Physical Illnesses

By Jaspal Khanuja, Sammy Rakeen Fairad, and Evelyn Pan

Introduction

This project focuses on optimizing diets tailored for specific illnesses by utilizing an updated database that includes supplements. Our group initially brainstormed project ideas during class presentations of individual concepts, ultimately deciding on this one due to its intriguing nature and potential real-world applications. We also chose to optimize diets for specific illnesses to build upon the concept of an optimal diet while aiming to enhance the project's future utility and relevance.

We will pull data from this website: https://fdc.nal.usda.gov/download-datasets (specifically the "Foundational Foods" database)

Mathematical Model

What are the decision variables?

- The decision variables in this model will be the food and supplements from the dataset

What are the constraints?

- We will have disease specific constraints (i.e. diabetics cannot eat too much sugar)

What are the parameters?

- Filtering through foods/supplements for specific daily necessities based on objective

What is the objective function?

- Finding foods that cover the most amount of required vitamins possible for the given illness

Analysis

- This will be a non-linear model
- We will nly keep track of foods that fit parameter, so sensitivity (food item with >= 50 mg

sodium etc)

Solution

Below is the Julia code implementation of this solution. We first clean up the data and fit it to our needs, then go through the fitted data using our model.

In [1]:
```julia
#Pkg.add(["DataFrames", "CSV"])
using Pkg
using Statistics
using CSV
using DataFrames
```

We first define the paths for the database that we are using (pulled from https://fdc.nal.usda.gov/download-datasets "Foundation Foods" dataset)

In [2]:
```julia
# Define full paths to your files
food_path = "data/food.csv"
food_nutrient_path = "data/food_nutrient.csv"
nutrient_path = "data/nutrient.csv"

# Load CSV files
food = CSV.read(food_path, DataFrame)
food_nutrient = CSV.read(food_nutrient_path, DataFrame)
nutrient = CSV.read(nutrient_path, DataFrame)

# Preview the data
println(first(food, 5))
println(first(food_nutrient, 5))
println(first(nutrient, 5))
```

```
5×5 DataFrame
 Row │ fdc_id  data_type           description            food_category_id  publication_date
     │ Int64   String31            String                 Int64?            String15
─────┼──────────────────────────────────────────────────────────────────────────────────────
   1 │ 319874  sample_food         HUMMUS, SABRA CLASSIC                16  2019-04-01
   2 │ 319875  market_acquisition  HUMMUS, SABRA CLASSIC                16  2019-04-01
   3 │ 319876  market_acquisition  HUMMUS, SABRA CLASSIC                16  2019-04-01
   4 │ 319877  sub_sample_food     Hummus                               16  2019-04-01
   5 │ 319878  sub_sample_food     Hummus                               16  2019-04-01
5×11 DataFrame
 Row │ id       fdc_id  nutrient_id  amount   data_points  derivation_id  min      max      median   footnote  min_year_acquired
     │ Int64    Int64   Int64        Float64?  Int64?       Int64?         Float64?  Float64?  Float64?  String?  Int64?
─────┼──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
   1 │ 2201847  319877         1051     56.3            1              1  missing   missing   missing   missing  missing
   2 │ 2201845  319877         1002      1.28           1              1  missing   missing   missing   missing  missing
   3 │ 2201846  319877         1004     19.0            1              1  missing   missing   missing   missing  missing
   4 │ 2201844  319877         1007      1.98           1              1  missing   missing   missing   missing  missing
   5 │ 2201852  319878         1091    188.0            1              1  missing   missing   missing   missing  missing
5×5 DataFrame
 Row │ id     name                              unit_name  nutrient_nbr  rank
     │ Int64  String                            String7    Float64?      Float64?
─────┼──────────────────────────────────────────────────────────────────────────
   1 │  2047  Energy (Atwater General Factors)  KCAL             957.0     280.0
   2 │  2048  Energy (Atwater Specific Factors) KCAL             958.0     290.0
   3 │  1001  Solids                            G                201.0     200.0
   4 │  1002  Nitrogen                          G                202.0     500.0
   5 │  1003  Protein                           G                203.0     600.0
```

We then fit the data to our needs. This is done by renaming columns and joining them together to produce a more dataframe with all the relevant data.

In [3]:
```julia
# Rename `:id` in `nutrient` to `:nutrient_id`
rename!(nutrient, :id => :nutrient_id)

# Merge food and food_nutrient on `fdc_id`
food_with_nutrients = innerjoin(food, food_nutrient, on=:fdc_id)

# Merge food_with_nutrients with nutrient on `nutrient_id`
full_data = innerjoin(food_with_nutrients, nutrient, on=:nutrient_id)

# Select relevant columns
relevant_data = select(full_data, :description, :name, :amount, :unit_name)

# Pivot the table to have nutrients as columns
pivoted_data = unstack(
    relevant_data,
    :name,       # Column to pivot into separate columns
    :amount,     # Values to fill the columns
    combine=mean # Aggregate duplicate values by taking the mean
)

# Display a sample of the processed dataset
println("Processed Data:")
println(first(pivoted_data, 5))

# Save the processed data to a new CSV file (optional)
CSV.write("data/processed_data.csv", pivoted_data)
```

```
Processed Data:
5×227 DataFrame
 Row │ description         unit_name  Water         Nitrogen       Total lipid (fat)  Ash           Phosphorus, P  Manganese, Mn  Potassium, K  Calcium, Ca   Sodium, Na   Magnesium, Mg  Iron, Fe       Copper, Cu      Zinc, Zn       Pantothenic acid  SFA 14:0    SFA 24:0     SFA 11:0    SFA 18:0     PUFA 22:6 n-3 (DHA)  SFA 6:0     PUFA 22:4  SFA 15:0     PUFA 20:5 n-3 (EPA)  PUFA 18:2 CLAs  PUFA 20:4    SFA 4:0     SFA 16:0     TFA 16:1 t  PUFA 18:4  SFA 22:0     PUFA 22:5 n-3 (DPA)  TFA 18:1 t   PUFA 20:3 n-9  MUFA 15:1  MUFA 18:1 c  MUFA 20:1 c  PUFA 18:3 n-3 c,c,c (ALA)  SFA 20:0     TFA 22:1 t  PUFA 22:2  PUFA 20:2 n-6 c,c  MUFA 22:1 n-9  SFA 10:0    SFA 17:0    SFA 12:0    SFA 8:0     TFA 18:2 t not further defined  MUFA 16:1 c  PUFA 20:3 n-6  TFA 18:3 t  PUFA 18:3 n-6 c,c,c  PUFA 18:2 n-6 c,c  MUFA 14:1 c  MUFA 17:1     MUFA 24:1 c  PUFA 20:3 n-3  Starch     Folate, total  Selenium, Se  Riboflavin  Niacin          Fiber, total dietary  Thiamin  Vitamin B-6  Lutein + zeaxanthin  Carotene, beta  Cryptoxanthin, beta  Lycopene  Carotene, alpha  Tocopherol, delta  Tocopherol, beta  Tocopherol, gamma  Vitamin E (alpha-tocopherol)  Tocotrienol, alpha  Tocotrienol, beta  Tocotrienol, gamma  Tocotrienol, delta  Vitamin C, total ascorbic acid  Lactose    Glucose    Sucrose    Maltose    Galactose  Fructose  Vitamin K (Menaquinone-4)  Vitamin K (Dihydrophylloquinone)  Vitamin K (phylloquinone)  Choline, from glycerophosphocholine  Betaine    Choline, from phosphocholine  Choline, from phosphotidyl choline  Choline, free  Choline, total
```

```
Choline, from sphingomyelin  Vitamin D2 (ergocalciferol)  25-hydroxycholecalciferol  Vitamin D3 (cholecalciferol
)  Cholesterol  Iodine, I  SFA 21:0  SFA 9:0   SFA 7:0   MUFA 22:1 n-11  PUFA 22:3  Fatty acids, total trans
SFA 23:0  TFA 14:1 t  Fatty acids, total monounsaturated  Fatty acids, total polyunsaturated  TFA 20:1 t  MUFA 1
2:1  SFA 5:0   Fatty acids, total saturated  Serine    Valine    Leucine   Arginine  Glutamic acid  Alanine
Cysteine  Histidine  Hydroxyproline  Phenylalanine  Aspartic acid  Isoleucine  Proline   Tryptophan  Threonine
Glycine   Methionine  Tyrosine  Lysine    Vitamin B-12  Retinol   Zeaxanthin  trans-beta-Carotene  trans-Lycopen
e  Lutein    cis-beta-Carotene  cis-Lycopene  Cryptoxanthin, alpha  cis-Lutein/Zeaxanthin  Protein   MUFA 20:1
Fatty acids, total trans-dienoic  Total fat (NLEA)  Fatty acids, total trans-monoenoic  Energy    Carbohydrate,
by summation  PUFA 20:2 c  Vitamin A, RAE  Sugars, Total  PUFA 20:3 c  PUFA 22:5 c  MUFA 17:1 c  Fatty acids, to
tal trans-polyenoic  PUFA 18:2 c  PUFA 18:3 c  PUFA 20:5c  PUFA 22:6 c  Carbohydrate, by difference  PUFA 20:4c
MUFA 22:1 c  Energy (Atwater General Factors)  Energy (Atwater Specific Factors)  Vitamin D (D2 + D3)  Vitamin D
(D2 + D3), International Units  Phytofluene  Phytoene  Fiber, insoluble  Fiber, soluble  Cystine   MUFA 18:1
PUFA 18:2  PUFA 20:3  PUFA 18:3  PUFA 18:3i  Specific Gravity  MUFA 22:1  5-Formyltetrahydrofolic acid (5-HCOH4
10-Formyl folic acid (10HCOFA)  5-methyl tetrahydrofolate (5-MTHF)  Molybdenum, Mo  Sulfur, S  Cobalt, Co
Boron, B  Nickel, Ni  Biotin    Beta-sitostanol  Campesterol  Campestanol  Phytosterols, other  Beta-sitosterol
Brassicasterol  Delta-5-avenasterol  Stigmasterol  Total dietary fiber (AOAC 2011.25)  Malic acid  Citric acid
TFA 18:2 t  Genistin  Daidzein  Genistein  Glycitin  Daidzin   Carotene, gamma  Total Sugars  Stigmastadiene
Delta-7-Stigmastenol  Glutathione  Beta-glucan  Ergosta-7,22-dienol  Ergosterol  Vitamin D4  Ergosta-7-enol
Ergosta-5,7-dienol  Ergothioneine  Raffinose  Stachyose  Verbascose  Oxalic acid  Quinic acid  High Molecular We
ight Dietary Fiber (HMWDF)  Low Molecular Weight Dietary Fiber (LMWDF)  Pyruvic acid
     │ String             String7    Float64?      Float64?       Float64?          Float64?       Float64?
Float64?      Float64?      Float64?      Float64?     Float64?       Float64?       Float64?        Float64?
Float64?         Float64?    Float64?     Float64?   Float64?     Float64?            Float64?   Float64?
Float64?     Float64?            Float64?        Float64?     Float64?   Float64?    Float64?    Float64?
Float64?     Float64?            Float64?     Float64?       Float64?   Float64?     Float64?     Float64?
Float64?     Float64?    Float64?   Float64?          Float64?       Float64?   Float64?     Float64?
Float64?   Float64?                           Float64?     Float64?       Float64?    Float64?            Float64?
Float64?     Float64?     Float64?     Float64?       Float64?   Float64?       Float64?      Float64?
Float64?        Float64?              Float64?  Float64?      Float64?            Float64?       Float64?
Float64?  Float64?         Float64?          Float64?         Float64?          Float64?
Float64?           Float64?           Float64?            Float64?            Float64?
Float64?  Float64?  Float64?  Float64?  Float64?   Float64?  Float64?                Float64?
Float64?                 Float64?                              Float64?   Float64?
Float64?                          Float64?      Float64?       Float64?                     Float64?
Float64?                 Float64?                    Float64?     Float64?   Float64?  Float64?  Float64?
Float64?        Float64?   Float64?                Float64?  Float64?    Float64?
Float64?                          Float64?    Float64?   Float64?  Float64?                     Float64?
Float64?  Float64?  Float64?  Float64?       Float64?  Float64?  Float64?   Float64?       Float64?
Float64?       Float64?    Float64?  Float64?    Float64?   Float64?  Float64?    Float64?  Float64?  Float64?
Float64?  Float64?    Float64?            Float64?        Float64?  Float64?           Float64?      Float64?
Float64?              Float64?  Float64?   Float64?                          Float64?         Float64?
Float64?  Float64?                    Float64?     Float64?        Float64?       Float64?     Float64?
Float64?     Float64?                          Float64?     Float64?     Float64?    Float64?     Float64?
Float64?    Float64?     Float64?                          Float64?                          Float64?
Float64?                                Float64?     Float64?  Float64?         Float64?        Float64?
Float64?   Float64?    Float64?    Float64?    Float64?    Float64?         Float64?   Float64?
Float64?                        Float64?                            Float64?        Float64?   Float64?
Float64?  Float64?    Float64?  Float64?         Float64?     Float64?     Float64?             Float64?
Float64?        Float64?              Float64?      Float64?                            Float64?   Float64?
Float64?    Float64?  Float64?  Float64?   Float64?  Float64?  Float64?         Float64?      Float64?
Float64?              Float64?     Float64?     Float64?              Float64?    Float64?    Float64?
Float64?            Float64?       Float64?   Float64?   Float64?     Float64?      Float64?     Float64?
Float64?                                    Float64?
─────┼────────────────────────────────────────────────
```

──
──
──
──

1 │ Hummus G 58.6818 1.17636 17.1364 1.96909 missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing
2 │ Hummus MG missing missing missing missing
166.091 1.05909 289.273 40.7273 438.364 71.1091 2.41091 0.347818
1.37909 0.3175 missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
0.115 0.948333 missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing 1.1 0.2 23.0
0.2 22.3 46.6 0.0 missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing missing missing missing missing
missing missing missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing
3 │ Hummus - NFY121400 G missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing 0.01 0.022 0.0 0.771 0.0 0.0 0.0
0.003 0.0 0.003 0.006 0.0 1.76 0.0 0.0 0.0
44 0.0 0.006 0.0 0.0 5.03 0.048 0.
725 0.077 0.0 0.0 0.005 0.0 0.0 0.014 0.0
0.0 0.013 0.018 0.0 0.0 0.013 8
.56 0.0 0.007 0.0 0.0 missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing missing
missing missing missing missing missing
missing missing missing missing missing missing missing missing
missing missing missing missing missing

```
missing        missing         missing                      missing                        missing
missing                       missing      missing    missing   missing   missing   missing         missing
missing                   missing   missing     missing                             missing
missing     missing    missing   missing                       missing   missing   missing   missing   missing
missing   missing   missing    missing         missing        missing        missing     missing   missing
missing    missing   missing     missing   missing   missing       missing   missing     missing
missing         missing   missing            missing       missing               missing                missing
missing    missing                           missing           missing                             missing
missing                     missing      missing         missing        missing      missing      missing
missing                              missing      missing      missing     missing      missing
missing     missing      missing                           missing                            missing
missing                                   missing      missing   missing           missing         missing
missing    missing    missing    missing    missing     missing           missing    missing
missing                         missing                             missing         missing    missing
missing   missing     missing   missing          missing      missing      missing              missing
missing         missing              missing       missing                             missing     missing
missing     missing   missing   missing    missing   missing   missing          missing       missing
missing               missing      missing      missing              missing     missing     missing
missing             missing        missing    missing    missing     missing      missing      missing
missing                                     missing       missing
   4 │ Hummus - NFY12140P  G          missing       missing             missing       missing          missing
missing         missing      missing       missing       missing       missing        missing         missing
missing       missing     missing      missing    missing                missing    missing    missing
missing                missing       missing      missing      missing    missing      missing    missing
missing                missing    missing          missing    missing     missing     missing
missing      missing       missing    missing          missing          missing    missing    missing
missing    missing                        missing      missing          missing     missing            missing
missing       missing    missing        missing        missing          8.2   missing         missing
missing      missing                        missing   missing      missing              missing         missing
missing   missing          missing            missing          missing            missing
missing             missing            missing             missing             missing
missing   missing   missing   missing   missing    missing   missing                    missing
missing                    missing                             missing    missing                        missing
missing        missing         missing                      missing                        missing
missing                       missing      missing    missing   missing   missing   missing         missing
missing                   missing   missing     missing                             missing
missing     missing    missing   missing                       missing   missing   missing   missing   missing
missing   missing   missing    missing         missing        missing        missing     missing   missing
missing    missing   missing     missing   missing   missing       missing   missing     missing
missing         missing   missing            missing       missing               missing                missing
missing    missing                           missing           missing                             missing
missing                     missing      missing         missing        missing      missing      missing
missing                              missing      missing      missing     missing      missing
missing     missing      missing                           missing                            missing
missing                                   missing      missing   missing           missing         missing
missing    missing    missing    missing    missing     missing           missing    missing
missing                         missing                             missing         missing    missing
missing   missing     missing   missing          missing      missing      missing              missing
missing         missing              missing       missing                             missing     missing
missing     missing   missing   missing    missing   missing   missing          missing       missing
missing               missing      missing      missing              missing     missing     missing
missing             missing        missing    missing    missing     missing      missing      missing
missing                                     missing       missing
   5 │ Hummus              UG         missing       missing             missing       missing          missing
missing         missing      missing       missing       missing       missing        missing         missing
missing       missing     missing      missing    missing                missing    missing    missing
missing                missing       missing      missing      missing    missing      missing    missing
missing                missing    missing          missing    missing     missing     missing
missing      missing       missing    missing          missing          missing    missing    missing
missing    missing                        missing      missing          missing     missing            missing
missing       missing    missing        missing        missing    missing         36.3333         16.15
missing      missing                        missing   missing      missing              missing         missing
missing   missing          missing            missing          missing            missing
missing             missing            missing             missing             missing
missing   missing   missing   missing   missing    missing   missing                    missing
missing                    missing                             missing    missing                        missing
missing        missing         missing                      missing                        missing
missing                       missing      missing    missing   missing   missing   missing         missing
missing                   missing   missing     missing                             missing
missing     missing    missing   missing                       missing   missing   missing   missing   missing
missing   missing   missing    missing         missing        missing        missing     missing   missing
missing    missing   missing     missing   missing   missing       missing   missing     missing
missing         missing   missing            missing       missing               missing                missing
missing    missing                           missing           missing                             missing
missing                     missing      missing         missing        missing      missing      missing
missing                              missing      missing      missing     missing      missing
missing     missing      missing                           missing                            missing
missing                                   missing      missing   missing           missing         missing
missing    missing    missing    missing    missing     missing           missing    missing
missing                         missing                             missing         missing    missing
missing   missing     missing   missing          missing      missing      missing              missing
missing         missing              missing       missing                             missing     missing
missing     missing   missing   missing    missing   missing   missing          missing       missing
```

```
missing              missing      missing      missing              missing     missing     missing
missing            missing        missing    missing    missing     missing      missing      missing
missing                                      missing       missing
```

Out[3]: "data/processed_data.csv"

We can then take the processed data and clean it so that the model can read through and process the inputs properly.

In [4]:
```
# Define file paths
input_path = "data/processed_data.csv"
output_path = "data/processed_data_cleaned.csv"
# Load the processed data
processed_data = CSV.read(input_path, DataFrame)
# Replace `missing` with `0` in each column
for col in names(processed_data)
if eltype(processed_data[!, col]) == Union{Missing, T} where T
replace!(processed_data[!, col], missing => 0)
end
end
# Save the cleaned dataset
CSV.write(output_path, processed_data)
println("Cleaned data saved to: ", output_path)
```

```
Cleaned data saved to: data/processed_data_cleaned.csv
```

Model

The cell below shows our model. We take the cleaned data above as our source data, then define 3 groups of constraints to represent diets that help with diabetes, heart disease, and osteoporosis. We then go through the data and get optimized diets, which are then saved into a csv file. We tried using a JuMP model for this model, but had difficulties implementing it so we opted to define the model in the way shown below.

In [5]:
```
# Load the cleaned dataset
input_path = "data/processed_data_cleaned.csv"
cleaned_data = CSV.read(input_path, DataFrame)

# Define nutrient thresholds for various diseases
disease_nutrient_thresholds = Dict(
    "Diabetes" => Dict(
        "Vitamin D (D2 + D3)" => 5,
        "Vitamin C, total ascorbic acid" => 10,
        "Vitamin E (alpha-tocopherol)" => 2,
        "Magnesium, Mg" => 50,
        "Zinc, Zn" => 2,
        "Fiber, total dietary" => 3
    ),
    "Heart Disease" => Dict(
        "Omega-3 fatty acids" => 0.5,  # Example: EPA/DHA in grams
        "Potassium, K" => 350,
        "Magnesium, Mg" => 50,
        "Vitamin C, total ascorbic acid" => 10,
        "Fiber, total dietary" => 3
    ),
    "Osteoporosis" => Dict(
        "Calcium, Ca" => 200,
        "Vitamin D (D2 + D3)" => 5,
        "Vitamin K" => 0.1,
        "Magnesium, Mg" => 50,
        "Phosphorus, P" => 100
    ),
    # Add other diseases similarly
)

# Filter foods for each disease
function filter_foods_for_diseases(data::DataFrame, disease_thresholds::Dict)
    results = Dict()
    for (disease, thresholds) in disease_thresholds
        # Minimum number of thresholds to meet (majority)
        min_thresholds = Int(ceil(length(thresholds) / 2))

        # Filter rows based on nutrient thresholds
        filtered_foods = filter(row -> begin
            count_met = 0
            for (nutrient, threshold) in thresholds
                if nutrient in names(data)
                    if coalesce(row[nutrient], 0) >= threshold
                        count_met += 1
                    end
                end
            end
            count_met >= min_thresholds
        end, data)
```

```
        results[disease] = filtered_foods
    end
    return results
end

# Apply the function
filtered_foods_by_disease = filter_foods_for_diseases(cleaned_data, disease_nutrient_thresholds)

# Save results to CSV
output_dir = "results"
for (disease, foods) in filtered_foods_by_disease
    output_path = joinpath(output_dir, "$disease.csv")
    CSV.write(output_path, foods)
    println("Filtered foods for $disease saved to: $output_path")
end
```

```
Filtered foods for Heart Disease saved to: results\Heart Disease.csv
Filtered foods for Osteoporosis saved to: results\Osteoporosis.csv
Filtered foods for Diabetes saved to: results\Diabetes.csv
```

Results and discussion

Below are example tables of our output.

In [6]:
```
filtered_foods_by_disease["Heart Disease"]
```

Out[6]: 4×227 DataFrame *127 columns omitted*

Row	description	unit_name	Water	Nitrogen	Total lipid (fat)	Ash	Phosphorus, P	Manganese, Mn	Potassium, K	Calcium, Ca	Sodium, Na
	String?	String7	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?
1	Spinach, baby	MG	*missing*	*missing*	*missing*	*missing*	39.05	0.4884	581.8	68.35	111.4
2	Spinach, mature	MG	*missing*	*missing*	*missing*	*missing*	40.56	0.4257	459.7	66.64	106.6
3	SPINACH, REGULAR (MATURE)	MG	*missing*	*missing*	*missing*	*missing*	40.5562	0.425687	459.688	66.6438	106.644
4	SPINACH, BABY	MG	*missing*	*missing*	*missing*	*missing*	39.05	0.488375	581.75	68.35	111.425

In [7]:
```
filtered_foods_by_disease["Diabetes"]
```

Out[7]: 7×227 DataFrame *127 columns omitted*

Row	description	unit_name	Water	Nitrogen	Total lipid (fat)	Ash	Phosphorus, P	Manganese, Mn	Potassium, K	Calcium, Ca	Sodium, Na
	String?	String7	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?	Float64?
1	Nuts, almonds, dry roasted, with salt added	MG	*missing*	*missing*	*missing*	*missing*	456.0	2.02	684.0	273.0	256.0
2	Peanut butter, smooth style, with salt	MG	*missing*	*missing*	*missing*	*missing*	339.0	1.68	564.0	49.0	429.0
3	Seeds, sunflower seed kernels, dry roasted, with salt added	MG	*missing*	*missing*	*missing*	*missing*	750.0	2.89	689.0	78.0	532.0
4	Peanut butter, creamy	MG	*missing*	*missing*	*missing*	*missing*	393.2	1.681	654.0	49.85	220.7
5	Almond butter, creamy	MG	*missing*	*missing*	*missing*	*missing*	506.9	2.135	745.4	263.8	0.9963
6	almond butter, creamy	MG	*missing*	*missing*	*missing*	*missing*	506.875	2.135	745.375	263.75	0.99625
7	peanut butter, creamy	MG	*missing*	*missing*	*missing*	*missing*	393.188	1.68112	654.0	49.85	220.688

As seen above, we can find what foods would be part of an optimal diet for anyone with a specific illness according to nutritional content. From the tables above, we can see that the model outputs a completely different list of foods depending on the input. We can then surmise the nutritional content of these foods (measured in miligrams).

For heart disease, we are shown that spinach is a very good source of nutrition, while for diabetes we get a more widespread result of food in our diet. This data shows that for each illness, we will have different amounts of options for an optimized and health diet according to the restrictions. These results also show us that there is usually a trend with the type of food that would generally be good in a diet for people with these illnesses (i.e. salty/creamy food for people with diabetes).

Limitations of model:

- Does not take into account the prices of food items
- Returns a table that includes every possible nutrient, even if not included in actual food item

Conclusion

In conclusion, every illness comes out with an entirely different optimized diet that have a unique amount of food options and types of food. We were able to use the data from the USDA, then implemented it in model, which can then be fitted for any illness by changing the constraints. A future direction this project can take is getting a different dataset for food prices and take that into account when optimizing a diet, as we tried doing so but encountered difficulty due to missing data in our chosen dataset. We can also try cleaning up the output some more for more accessible reading of the results.

Student Contributions:

- Jaspal Khanuja: Found and cleaned data, wrote code, decided problem focus, discussed mathematical model
- Sammy Rakeen Fairad: Reviewed and cleaned up code, analyzed results, wrote report, set up meetings, discussed mathematical model
- Evelyn Pan: Discussed problem ideas and mathematical model