Diet Problem in 74 Different Developing Countries

Matthew Grattan* George Ho†

EID/ChE488: Convex Optimization Techniques

The Cooper Union for the Advancement of Science and Art

December 19, 2018

*grattan@cooper.edu

†ho@cooper.edu

Contents

Abstract

The diet problem is a classic linear programming problem. Here, we solve the diet problem for 1457 markets across 74 different developing countries. We use a constraint set of four nutrients (carbohydrates, fats, protein, and fiber) and 85 unique foods—not all of these foods are available in each country, however. We perform a closer inspection of the optimal solution for markets in Lori, Armenia; Antioquia, Colombia; and Kachin, Myanmar and find that these efficient diets consist of only a few foods. It is important to note that we use a simple model, which neglects micronutrients, personal preference for a variety of foods, and fluctuations in supply and demand.

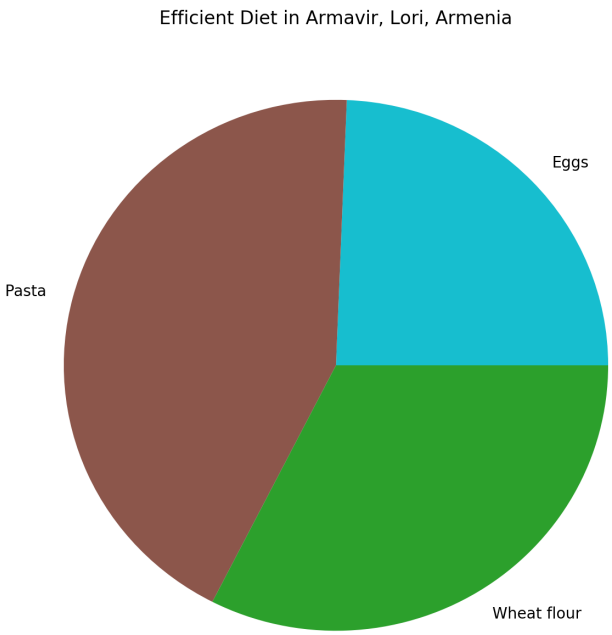

Figure 1: Efficient diets contain only a few foods.

1 Introduction

The diet problem is a classic linear programming problem (LP) whose solution is cheapest diet that satisfies a set of nutritional constraints. The problem was first posed by Jerry Cornfield for the US Army during World War II[1] to find "a low cost diet that would meet the nutritional needs of a GI soldier."[2] George Stigler, an economist, offered a heuristic solution to the diet problem for 77 foods in 1945, but it was not necessarily the minimum cost possible.[2,3] In 1947, Jack Laderman solved Stigler's diet problem with George Dantzig's simplex method,[2] one of the first algorithms to solve LPs efficiently.[4] LPs, including the diet problem, were important for military logistics, and their solutions helped minimize costs associated with the allocation of supplies. Many solution methods were developed during World War II.[4]

The diet problem is interesting to anyone trying to find the cheapest diet given a set of foods and nutrients. The problem is of interest because the constraints are quite flexible; they can be as simple as calories or macronutrients, but can also include vitamins and minerals, total food weight, and variety of foods in the solution set. Contemporary approaches to the diet problem account for specific health concerns like glycemic load for people with diabetes,[5] or seek to minimize environmental impact by accounting for carbon emissions, nitrogen release, water usage, and land usage.[6] Other novel approaches employ fuzzy logic to model uncertainty, like fluctuations in pricing.[5]

In this paper, we are particularly interested in solving the diet problem for different developing countries. We want to minimize the cost of food, subject to: regional availability of foods and the daily recommended intake of carbohydrates, fat, protein and fiber. Food supplies vary based on region, so the regional availability of certain foods further constrains the problem. We would like to minimize the cost of subsistence in 1457 markets across 74 developing countries. We are constraining the problem with only four nutrients (carbohydrates, fat, protein, and fiber) and a set of 85 unique foods. Not all of the foods are available in each market.

2 Problem Description

Table 1: Diet Problem Parameters

Parameter	Value
Daily protein requirement	50
Daily fat requirement	65
Daily carbohydrate requirement	300
Daily fiber requirement	25
Food prices	Vary by country, locality and market

We acquired a dataset on world food prices compiled by the World Food Program from Kaggle.[7] The

data set contained the prices of 321 commodities from 1457 markets across 74 developing countries. We removed the commodities that were not foods and consolidated subtypes of foods into one entry (i.e. we treated "Bread (wheat)" and "Bread (rye)" as simply "Bread"). This data processing resulted in a final list of 85 foods.

We acquired nutrition data on carbohydrates, fat, protein, and fiber for the reduced set of 85 foods from the USDA Branded Food Products Database.[8] We chose not to include vitamins and minerals to focus what would likely be staple foods—and for convenience as the data was entered by hand.

While the data sets are too large to reproduce in our appendices, we have given the first few rows of our data files as examples. Please refer to our Appendix for both example data and our code.

In general, LPs can be solved using a variety of methods, depending on the size of the problem. Brute-force is conceptually the simplest but also the most time consuming: A basic solution is computed for each of the n vertices of the feasible set, resulting $n!$ linear systems.[4] A popular solution method for LPs is the simplex algorithm, which finds the minimum value by calculating the objective starting at one vertex and moves in the direction of the adjacent vertex with the lowest objective value. Interior-point methods were developed more recently in the 1980s and are effective for very large LPs.[4]

Our problem is actually a set of 1457 LPs, each with 85 variables and four constraints. We chose the simplex algorithm as our solution method and implemented it using the built-in LP solver from the SciPy package for Python (`scipy.optimize.linprog`).[9]

We can pose the problem in the following way:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) = c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$. For our problem specifically, $n = 85$ and $m = 4$. The vector c represents the cost of each of the 85 foods, the vector b represents the recommended daily intake of each nutrient, and the matrix A represents the amount of each nutrient in each food. The decision variable $x \in R^n$ represents the amount of food purchased in kilograms. Examples of A, b, and c are given below.

$$A = \begin{bmatrix} 106.7 & 39.9 & \dots & 174.4 \\ 45.3 & 3.4 & \dots & 81.0 \\ 475.4 & 213.4 & \dots & 0.4 \\ 40.0 & 18.0 & \dots & 0.0 \end{bmatrix}$$

$$b = \begin{bmatrix} 50 & 65 & 300 & 25 \end{bmatrix}^\top$$

$$c = \begin{bmatrix} 51.72 & 67.00 & 29.00 & \dots & 50.00 \end{bmatrix}^\top$$

Let $f(x)$ be the cost function (literally the cost of food) we seek to minimize, and define the feasible set as

$$\Omega = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$$

To prove the existence of a feasible solution x^* to our problem, we first apply the Weierstrass Extreme Value Theorem.[4]

The feasible set Ω is nonempty because it contains at least the zero vector. Suppose $x = 0$, then

$$Ax = 0 \leq B$$

All inequalities are nonstrict, so Ω is a closed polytope in $\mathbb{R}^n$. The set Ω is also bounded. Let

$$\epsilon = 1 + \max_i |b_i|$$

and define the open ball of radius ϵ centered at the origin as

$$B_\epsilon(0) = \{x \in R^n : \|x\| < \epsilon\}$$

then $\forall x \in \Omega, x \in B_\epsilon(0)$ also. This implies that $\Omega \subseteq B_\epsilon(0)$. Therefore, Ω is compact, as it is closed and bounded.

The objective $f(x) = c^\top x$ is a linear functional, that maps vectors in $\mathbb{R}^n$ to scalar values in $\mathbb{R}$. $\Omega \subset \mathbb{R}^n$, so f is defined on Ω. Matrix multiplication is continuous; thus, f is continuous.

The conditions of the Weierstrass theorem are satisfied, so there exists $x^* \in \Omega$ that minimizes $f(x)$.

We next show that x^* lies on the boundary of Ω by applying the first-order necessary conditions for interior points. $f \in \mathcal{C}^1$ because f is simply matrix multiplication. $\nabla f(x) = c \; \forall x$, but for interior points that satisfy the first order necessary conditions, $\nabla f(x^*) = 0$. Therefore, $x^* \in \partial\Omega$.

3 Results

3.1 Solver

LPs are extremely well-studied and well-understood, with open-source implementations of solvers available. We use the SciPy implementation of the simplex algorithm (`scipy.optimize.linprog`) to solve our optimization problems.[9]

The problems were fairly small. We considered 85 possible foods, and 4 possible nutrients: this means that each problem had 85 variables and 4 constraints. Note that even though not all markets supply all foods, we set up the problem as if they did, but with a unit price of 999999. Thus, each problem has exactly 85 variables and 4 constraints. The bulk of the computation time comes down to the number of problems we solved: there are 1457 unique markets in developing countries for which we have data. In total, solving all optimization problems took less than 2 minutes.

Our code and data can be found on GitHub.[10]

3.2 Optimal Solution

Since there are 1457 different optimization problems (each with a minimum cost as expressed in the local currency, and not normalized to one single currency), it is infeasible to go over each optimal solution. Nevertheless, we make a few remarks. No optimization problem was infeasible, nor were any problems unbounded: the optimization algorithm terminated successfully for all problems.

Furthermore, all solutions are optimal. All LPs have their optimal solution on the boundary of the feasible set, and the simplex algorithm is guaranteed to find the optimal solution.[4] For a rigorous proof of this, please refer to section 2, Problem Description.

Finally, the solutions tended to be very sparse: that is, most foods would not be bought at all, and all nutrients would be supplied by buying some quantity of one or two foods. This suggests how unrealistic our problem formulation is: if everybody ate the efficient diet, then only one or two foods would be demanded in each country. Since these foods are scarce (as all foods are), there would quickly be food shortages or sharp price increases of the relevant foods, which would in turn change the LP that we need to solve.

3.3 Sensitivity Analysis

It is important to gain some sense of how sensitive the solution is to changes in the parameters. Ordinarily with a general optimization problem, it would be very difficult to perform a manual, perturbation-style sensitivity analysis of each individual problem (i.e. perturbing each price/nutrition parameter in turn), due

to the sheer number of optimization problems, and the large number of variables within each one. Happily however, our problems are all LPs, and there exists a simple method to perform a sensitivity analysis on an LP: simply solve the dual problem. The solution to the dual problem quantifies the sensitivity of the problem to its parameters: each dual variable corresponds to one constraint, and the value of the dual variable indicates how sensitive that constraint is to a change in parameters. If the dual variable is zero, then the constraint is strictly satisfied; if the dual variable is not zero, it is satisfied with equality and its magnitude indicates how sensitive it is to a change in parameters.

The default behavior of `scipy.optimize.linprog` is to solve not only the primal problem, but also the dual problem: it returns the dual variables as "slack" variables. Thus, little to no changes are actually required in terms of the codebase. In what follows, we present a statistical analysis of these dual variables.

Recall that all markets have four constraints, one for each of the four main nutritional requirements: protein, fat, carbohydrates and fiber. Each dietary requirement is expressed as an inequality constraint, which may be active or inactive. It would be interesting to understand the statistical distribution of the dual variables.

Far and away the most remarkable trait of this distribution of dual variables is the fact that over 90% of the optimization problems have an inactive fat inequality constraint. This means that most markets (in most localities and countries) do not struggle to provide a diet with sufficient fat, and that they would still be able to provide such a fat-sufficient diet even upon a small change of parameters. It is difficult to tell why this is the case. It is true that there are a lot of fat-rich foods in the data set: for example, most markets stock fish, which has a high fat content, and some markets even sell ghee, which is clarified butter (whether it makes sense to include ghee as a food, since it is unlikely to be consumed on its own, is a question that we did not foresee while creating the data set). However, it must also be that the fat requirement is just low relative to the average fat content of foods available in most markets, and that the foods available in most markets can easily meet such a low requirement. Most likely some combination of these factors result in this phenomenon.

On the other hand, it appears that the limiting factor is instead the other three inequality constraints: protein, carbohydrates and fiber. Whereas 90% of markets have the fat requirement as an inactive inequality constraint, the corresponding numbers for protein, carbohydrates and fiber are 20%, 30% and 20%, respectively. In words, this means that the diets available in many markets (in many localities and countries) only satisfy the constraints for protein, carbohydrates and fiber with equality, and are thus most in need of cheaper sources of these three nutrients. This statistical observation could speak either to the general lack of cheap, nutritious foods in the African continent, or to the inadequacy of our data set (which we elaborate on in the Future Work section).

4 Conclusion

4.1 Discussion

It is infeasible to discuss the numerical solutions to each of the 1457 problems: there are simply far too many problems to be able to analyze each solution by hand. It is also very difficult to analyze all the solutions statistically (e.g. provide summary statistics of all optimization problems), as not all numbers are expressed in the same units: in other words, prices in different countries are expressed in the local currency, and even within one country, different markets may not sell foods in the same units (e.g. one market may sell eggs by the dozen, whereas another market may sell individual eggs). It is clear that it is impossible to compare apples to apples without writing a "conversion" program that converts all prices to one common currency (e.g. USD), and all food quantities to common units.

In light of these difficulties, we will limit ourselves to discussing three particular solutions, doing all necessary conversions by hand. The interpretation of these solutions within the context of the original problem (the diet problem) is the same across all such solutions. Here, we consider the efficient diet for markets in the following locations

- Armavir, in Lori, Armenia
- San Vicente, in Antioquia, Colombia
- Sadung, in Kachin, Myanmar

As discussed above, all costs have been converted to USD, and all quantities of food have been expressed in the same units, as far as possible (e.g. it was impossible convert between "boxes" of eggs, to dozens of eggs).

Table 2: Efficient diets for markets in Lori, Antioquia, and Kachin

Market	Locality	Country	Cost of diet (ν^*)	Diet (x^*)
Armavir	Lori	Armenia	0.4 USD	0.23 kg of pasta, 0.13 boxes of eggs
San Vicente	Antioquia	Colombia	33.95 USD	25 kg of beans
Sadung	Kachin	Myanmar	5.68 USD	19 kg of rice

For Armavir, Lori, Armenia, the solution prescribes 0.23 kilogram of pasta, and 0.13 boxes of eggs, with a cost of 0.4 USD per day. Immediately, a problem surfaces: it should be impossible to buy 0.13 boxes of anything. We elaborate on this shortcoming below.

For San Vicente, Antioquia, Colombia, the solution prescribes 25 kilograms of beans, with a cost of nearly 34 USD per day. It is interesting that the diet consists entirely of beans: however, upon closer inspection, it

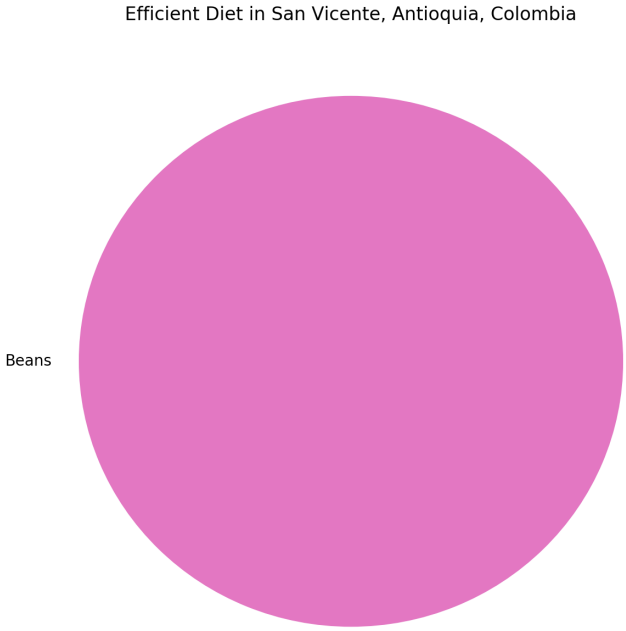

Figure 2: An efficient diet in San Vicente, Antioquia, Colombia consists of only beans.

turns out that according to our data set, San Vicente only sells beans, and therefore it makes sense that the optimal diet would consist of only beans: after all, it is the only thing that can be bought! We elaborate on this shortcoming below.

For Sadung, Kachin, Myanmar, the solution prescribes 19 kilograms of rice, with a cost of around 6 USD per day. While this diet consists of only one food (much like San Vicente), Sadung sells more than just rice: onions, potatoes and tomatoes are all sold, but rice is the cheapest option out of the four. It is therefore interesting that the optimal solution consists of only the cheapest food.

Put in this light, it is painfully clear that diets prescribed by the solutions to LPs are not sustainable: it is probably neither appealing nor ideal to eat the same food every day, and there may not even be a recipe to cook a proper meal with just one or two ingredients. We discuss this, and many other shortcomings and failings of our project, below.

We also make several other remarks on the solutions: we find that the most demanded foods (assuming that everybody follows the efficient diet) are rice, cowpeas, maize and wheat, each of which have a demand

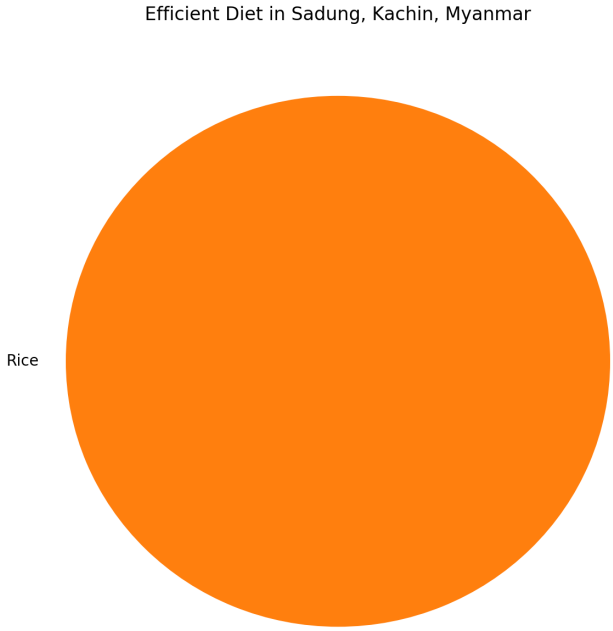

Figure 3: An efficient diet in Sadung, Kachin, Myanmar consists of only rice.

in excess of 1000 kilograms every day. If domestic supply of these foods fall short of this figure, then the developing country will need import them, which is not ideal. Furthermore, more than half of all foods under consideration are not demanded at all, from any market.

4.2 Future Directions

There are several shortcomings of this project, and several other directions for future work. We discuss each one in turn.

Firstly, we used food price data from developing countries, but we use used nutrition data from the USDA. There is some incongruity there: we are effectively assuming that food in developing countries has the same nutritional content as their counterparts in developed countries. This is probably true (or true enough) for staple foods such as chicken and beef. However, foods such as rice (or any other crop) is known to have different nutritional content depending on the nutritional content of the soil in which it is grown. Thus, it would be best to obtain a data set that is representative of food in developing countries, instead of

having to stitch together two separate data sets.

Secondly, our original data set leaves much to be desired: it only covers 85 foods. It would be ideal to obtain a more comprehensive data set of all foods available in developing countries. For example, it was described above that the market in San Vicente, Antioquia, Colombia only sells beans: it is worth corroborating if this is indeed the case, or if that is merely a result of an incomplete data set.

Thirdly, we implicitly assume that it possible to buy fractional amounts of each food. This is clearly not true: one cannot buy anything but an integer number of chickens, and it is improbable that one can buy any quantity of eggs. Thus, we should treat the diet problem as an integer linear problem (ILP), or, in the worst case, a mixed-integer linear problem (MILP), in order to account for the specific quantities in which one purchases food.

Fourthly, in the interest of simplicity, we simply discarded any additional information about the food supplied in parenthetical statements. Most such parenthetical statements disambiguated whether or not the particular food was imported, or the exact species of the food, and we thought it justifiable to discard this information. However, sometimes these parenthetical statements differentiated between the kind of poultry (e.g. chicken, turkey, duck etc.), or even different kinds of meats (e.g. beef, pork, lamb etc.). These distinctions are very important to make, and is something we should take into account.

Finally, we only consider the four most important nutrients. However, having seen that all markets are capable of supplying a feasible diet, it is now more interesting to see if it is possible for people in developing countries to maintain a diet that is considered healthy and recommended in a developed country: in other words, include the recommended daily intake of all vitamins and nutrients into the optimization problem.

References

[1] van Dooren, C. A Review of the Use of Linear Programming to Optimize Diets, Nutritiously, Economically and Environmentally. *Frontiers in Nutrition* **2018**, *5*, 48.

[2] Dantzig, G. B. The Diet Problem. *Interfaces* **1990**, *20*, 43–47.

[3] Stigler, G. J. The Cost of Subsistence. *Journal of Farm Economics* **1945**, *27*, 303–314.

[4] Chong, E. K.; Zak, S. H. An introduction to optimization.

[5] Bas, E. A robust optimization approach to diet problem with overall glycemic load as objective function. *Applied Mathematical Modelling* **2014**, *38*, 4926 – 4940.

[6] Gephart, J. A.; Davis, K. F.; Emery, K. A.; Leach, A. M.; Galloway, J. N.; Pace, M. L. The environmental cost of subsistence: Optimizing diets to minimize footprints. *Science of The Total Environment* **2016**, *553*, 120 – 127.

[7] Boysen, J. World Food Prices. `https://www.kaggle.com/jboysen/global-food-prices`.

[8] USDA Food Composition Databases. `https://ndb.nal.usda.gov/ndb/search/list`.

[9] `scipy.optimize.linprog`. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html`.

[10] Ho, G. diet-problem. `https://github.com/eigenfoo/diet-problem`.

Appendix

All our code can be found on the GitHub repository.[10] In the interest of brevity we attach excerpts from `nutrition.csv` and `pricing.csv` and the code of `all_countries.ipynb` here.

	Protein	Fat	Carb	Fiber
Bread	106.7	45.3	475.4	40.0
Rice	39.9	3.4	213.4	18.0
Wheat	74.9	12.7	425.3	11.0
Livestock	174.4	81.0	0.4	0.0
Apples	9.3	3.2	658.9	87.0

Table 3: First few rows of `nutrition.csv`

	country_name	locality_name	market_name	commodity	currency	market_type	units	price
0	Afghanistan	Daykundi	Nili	Bread	AFN	Retail	KG	51.72
1	Afghanistan	Daykundi	Nili	Rice	AFN	Retail	KG	67.00
2	Afghanistan	Daykundi	Nili	Wheat	AFN	Retail	KG	29.00
3	Afghanistan	Badakhshan	Fayzabad	Bread	AFN	Retail	KG	50.00
4	Afghanistan	Badakhshan	Fayzabad	Livestock	AFN	Retail	Unit	5500000.00

Table 4: First few rows of `pricing.csv`

In [1]:
```
'''
Solving the diet problem for all countries.

This script reads nutrition and pricing data, groups the data by country,
locality and market (in that order)
and solves the diet problem for each group, and writes the results to an output csv.
The results include the
minimum, the minimum value, the optimization status, and the slack
 (a.k.a. dual) variables.

Author: George Ho (gh@eigenfoo.xyz)
''';

import numpy as np
import pandas as pd
from scipy.optimize import linprog

NUTRITION_FILE = '../data/nutrition.csv'
PRICING_FILE = '../data/pricing.csv'

# If a particular market does not have a food, we set
# its price to be "infinity" (i.e. 999999). np.nan
# does not suffice, as we must have float datatypes.
INFINITY = 9999999999

# Read in data csvs
nutrition = pd.read_csv(NUTRITION_FILE, index_col=0)
pricing = pd.read_csv(PRICING_FILE, index_col=0)

# Group by country, locality and market, in that order.
grouped = pricing.groupby(['country_name', 'locality_name', 'market_name'])

solns = {}

# Solve diet problem for all groups
# (i.e. all markets in all localities in all countries)
for market, idx in grouped.groups.items():
    df = pricing.loc[idx]

    # Form A matrix (nutritional values of each food) and
    # b vector (dietary requirements for each nutrient).
    # Dietary requirements: protein, fat, carbs, fiber, in that order.
    # From http://www.netrition.com/rdi_page.html
    A_ub = -np.transpose(nutrition.values)
    b_ub = -np.array([50, 65, 300, 25])

    # Construct c vector appropriately (i.e. add INFINITYs to
    # the appropriate foods)
```

```
    c = pd.Series(data=INFINITY*np.ones(84), index=nutrition.index)
    c.loc[df.commodity_name] = df.price.values
    c = c.values

    # Solve LP using the default simplex algorithm.
    solns[market] = linprog(c, A_ub, b_ub)

# Collect optimization status, minimum value and minimum into one array
data = np.hstack([
    np.transpose(
        np.vstack([[soln.status for soln in solns.values()],
                   [soln.fun for soln in solns.values()]])
    ),
    [soln.x for soln in solns.values()],
    [soln.slack for soln in solns.values()]
])

# Cast array in a dataframe and save to a csv.
cols = (['status', 'fun']
        + nutrition.index.tolist()
        + ['protein_ineq', 'fat_ineq', 'carbs_ineq', 'fiber_ineq'])

df = pd.DataFrame(data=data,
                  index=solns.keys(),
                  columns=cols)
df.to_csv('all_countries.csv')
```