NUTRIOMICS CHALLENGE

Optimization of Dietary Nutrient Supplementation for Rational Rebalancing of Human Gut Microbiome

Zihan Wang, Ray Huang and Sang Yoon Byun

1. The Biological Problem

The human gut consists of microbes and bacterial species that form a unique and dynamic ecosystem called a microbiome. The gut microbiome serves multiple essential functions, including producing vitamins, defending against pathogens, and digesting fibre [1, 2]. In a healthy body, these species maintain homeostasis where the diverse bacterial composition is stable and well-balanced. However, once homeostasis is perturbed, the human body becomes vulnerable to a range of dysbiosis-related syndromes [3]. Therefore, rebalancing gut microbiome via nutritional intervention to enhance taxonomic diversity is essential for health. Human gut microbiomes vary massively from person to person–thus, it is vital to be able to generate personalized prebiotics to cater to each individual's needs [4].

2. The Algorithmic Challenge

1. Defining normal and abnormal microbiome compositions.

A major obstacle to our goal to rebalance the human gut microbiome is that we lack not only a standard definition of normal and abnormal microbiome compositions, but also how to quantitatively measure abnormality. To formulate this task into a computational problem, a crucial algorithmic challenge is to define what a normal microbiome composition is. Fortunately, we have access to a reference sample collection profiling human gut microbiome composition, which we can be use to define normality. However, it is important to note that the reference sample collection comes from a wide variety of people and should be better categorized later on.

2. Understanding how nutrients affect each species.

Since we plan to provide a diet to rebalance a human gut microbiome, we also have to define how the selected nutrients would affect the patient's microbiome composition and be able to quantitatively predict the outcome of nutrient supplementation on the relative abundance of any species. To tackle this, we could set up a nutrient impact score, quantifying the impact of selected nutrients on each ASV.

3. Optimal selection of nutrients for rebalancing taxonomic diversity.

Given the definition of normality in microbiome composition and the effect of nutrients on bacterial composition, the problem of selecting nutrients to rebalance the taxonomic diversity is well-defined. However, solving this optimization problem algorithmically remains a challenge. Since there are about 100 different nutrients, naively trying every combination of nutrients suggests that we need to consider approximately 2×10^{14} different combinations. If, for every combination, we also need to check the changed relative abundance for each species, we can end up with at least 4×10^{18} arithmetic operations. Such tasks will be computationally infeasible for a modern computer.

3. Definitions

Amplicon Sequences Variants (ASVs): A cluster of closely related microbial species.

Nutrient Impact (NI): A numerical value $[0, 1]$ representing the anticipated propensity for a given ASV to benefit from supplementation of a specific nutrient.

Nutrient Impact Matrix (NIM): An $m \times n$ matrix **NIM**, where m is the number of ASVs, and n is the number of nutrients, that aggregates the computed NI values for n nutrients over the entire set of ASVs.

Test Microbiome Sample (TMS): A $m \times 1$ vector MS, representing the proportion of the ASVs of a deviant sample.

Normal ASV Lower/Upper Bound: A vector $\mathbf{v}^{\text{low}}/\mathbf{v}^{\text{high}}$ denoting the lower/upper bounds of normal ASVs.

Intervention: An $n \times 1$ vector $\mathbf{x}$, with each value a positive real number, representing the nutritional intervention that increases nutrient abundance. The values in the vector represent the units of nutrients we provide in the dietary supplementation. Given a positive integer t, a vector $\mathbf{x}$ is classified as t-sparse if it has at most t non-zero elements.

Impact: A vector $\mathbf{b} = \mathbf{NIM} \times \mathbf{x}$ representing how the intervention alters the given TMS.

Failure Count: An integer representing the number of ASVs that do not fall into the normal abundance range. Let $c = c(\mathbf{NIM}, \mathbf{MS}, \mathbf{x}, \mathbf{v}^{\text{low}}, \mathbf{v}^{\text{high}})$ denote failure count, where c is defined as

$$c = m - |\{i \mid \mathbf{v}_i^{\text{low}} \leq \mathbf{MS}_i + \mathbf{b}_i \leq \mathbf{v}_i^{\text{high}}, 1 \leq i \leq m\}|$$

4. The Computational Problem

As a preliminary step, we compute the normal abundance range that each ASV in the TMS should fall into (see Preliminary Normal State Problem). We then focus on finding the optimal subset of nutrients that makes the maximal number of ASVs fall into the normal states. We allow each selected nutrient to have multiplicities, as providing more nutrients can activate more ASVs. The given NIM table decides how each ASV is impacted by the nutrients. We consider the changes in ASV values as the dot product of the NIs of the ASV with the nutritional intervention.

Dietary Nutrient Supplementation Optimization Problem:

Find a subset of nutrients that collectively minimizes the failure count.

Input: A matrix **NIM**, a TMS **u**, and a desired number of nutrients used t

Output: A t-sparse vector $\mathbf{x}$, that minimizes failure count $c(\mathbf{NIM}, \mathbf{MS}, \mathbf{x}, \mathbf{v}^{\text{low}}, \mathbf{v}^{\text{high}})$

Obtaining the Normal States:

Find a list of normal ASV abundance ranges for a particular TMS based on the RSC.

Reference Sample Collection (RSC): A collection of vectors describing microbiome samples that are presumed to be normal.

Input: A Reference Sample Collection *RSC* and a test microbiome sample *TMS*

Output: Two vectors $\mathbf{v}^{\text{low}}$ and $\mathbf{v}^{\text{high}}$ of a given *TMS* that defines the region of normal ASV states

Note: We deliberately omitted the interaction between a certain TMS to the abundance ranges (the TMS could affect the chosen ASVs) to simplify the whole problem. We define $\mathbf{v}^{\text{low}}$ and $\mathbf{v}^{\text{high}}$, as the 10%, 90% quantile among the records as the low and high values (this is just one possible heuristic). Since we believe that having abundant ASV is less harmful than having deviant ones, we further increased every value in vhigh by a unit of 5. With this set of vlow and vhighs, normal samples have on average 1-2 violations, while deviant samples have on average 11-12 violations.

5. Algorithmic Approach

5.1. *A Random Sampling Approach*

A simple solution is based on randomly sampling the nutrients to intervene with (and the amount of each nutrient). Formally, for a deviant sample and given a sparsity constraint of k, we randomly sample a vector that is k-sparse, where each non-zero element is randomly drawn from a 0-1 uniform distribution. We then do a grid search on a hyperparameter γ, which controls the scale of the nutrients (to compensate for the naive 0-1 uniform sampling). For each deviant sample, we can sample many trials and perform a grid search for γ for each trial. The best sampled vector is the solution given by this algorithm.

5.2. *A Linear Programming Approach*

Since the problem is composed of linear constraints (the $\mathbf{v}^{\text{low}}$ and $\mathbf{v}^{\text{high}}$), straightforward thinking is to model it as a linear programming (LP) problem, which is deterministically solvable with polynomial complexity [5]. The constraints can be turned into the following LP constraints:

$$-\sum_{j=1}^{n} NIM_{ij}x_j <= -\mathbf{v}^{\text{low}} + MS_i, \forall 1 \le i \le m \;\;\&\; \sum_{j=1}^{n} NIM_{ij}x_j <= \mathbf{v}^{\text{high}} - MS_i, \forall 1 \le i \le m$$

where x is the solution (intervention) we are optimizing for. However, we note that two ingredients are missing here: the sparsity of the solution and the minimization of the constraint violations instead of hard constraints. Both ingredients are hard to incorporate naively into an LP system. Hence, we take a step back and look for an algorithm that can approximately solve the problem by relaxing these two ingredients into easily optimized penalties.

Turning sparsity into a penalty

Sparsity constraints, in general, are hard to satisfy. Instead of finding a solution that satisfies a sparsity constraint, we turn to optimize a solution that is the most sparse. Simply speaking, we will additionally optimize for $||\mathbf{x}||_{l_0}$, where l_0 is defined as the non-zero elements of the solution vector $\mathbf{x}$. We introduce a hyperparameter α that controls the sparseness. The larger α is, the more sparse the solution has to be. Therefore, we can tweak α to obtain the desired sparseness.

However, the non-zero elements of a vector is not a linear function of the vector and is not easy to optimize. A (good) relaxation of the l_0 function is the l_1 norm [6, 7] that can also be integrated in an LP.

Turning constraint violations into a penalty

Minimizing constraint violations is also non-standard in linear programming. We address this by adding a gap variable $-y_i$ for each constraint, such as $y_i > 0$, when the constraint is unsatisfied and 0 otherwise. Then, instead of minimizing constraint violations, we can minimize the nonzero values of $\mathbf{y}$, which we can again approximate by minimizing the l_1 norm of $\mathbf{y}$.

By combining these two techniques, we turn the problem into an actual linear programming problem:

$$\begin{aligned} & min_{\mathbf{x},\mathbf{y}} \;\; \alpha \sum_{i=1}^{n} x_i + \beta \sum_{i=1}^{2m} y_i \\ & s.t. -\sum_{j=1}^{n} NIM_{ij}x_j - y_i \le -\mathbf{v}^{\text{low}} + MS_i, \forall 1 \le i \le m \\ & \sum_{j=1}^{n} NIM_{ij}x_j - y_{i+m} \le \mathbf{v}^{\text{high}} - MS_i, \forall 1 \le i \le m \\ & \mathbf{x} \ge 0, \mathbf{y} \ge 0 \end{aligned}$$

where α and β are two hyperparameters controlling how sparse we want the solution to be and how many constraint violations we wish to have. Note that since this LP always has a solution ($\mathbf{x} = \mathbf{0}, \mathbf{y} = \mathbf{inf}$), the linear program can always return an optimal solution.

Finding a Close to Optimal Solution with LP

Given the LP formulation and two values of α and β, we can (approximately) solve the problem. We want to find the solution among all possible α and β that is k sparse and produces the minimum number of violations. Now, we illustrate two ways to search for the α and β.

Grid Search: We can employ a grid search over all α and β pairs on a fixed range of α and βs. To achieve speed-up, note that the exact absolute values of α and β are unimportant since they are just a scale; what is important is the ratio $\alpha : \beta$. Therefore, we can skip the grid point when the slope is already searched in a previous grid point. We note that this cannot asymptotically speed up the algorithm, as the number of grid points needed to be calculated is the number of irreducible fractions, which is still quadratic to n, the number of points on one axis of the grid[1].

Double Binary Search: Now, we illustrate an algorithm that is aimed toward traversing the grid faster by skipping certain grid points. While it does not provably give the exact same solution as grid search (potentially less optimal), in practice, we find it much faster and also produces nearly the same solution as grid search.

The idea is based on binary search. We note that because both α and β are (inversely) proportional to the sparsity and constraint violations, we can perform a binary search on β, where for each trial of this binary search, we perform another binary search on α to obtain a solution that satisfies the sparsity constraints. While such double binary search does not necessarily produce the same solution as grid search (due to the coupling effect of α and β), in practice, it offers a good run time complexity of $O(\log^2 n)$, where n is the number of grid points.

6. Results

We performed grid search, binary search, and random search on all 142 deviant samples to examine the efficacy of nutritional interventions generated by these algorithms. All three methods significantly reduced the number of violations in every deviant sample, effectively placing large proportions of ASVs into normal states (Figure 1). We also tested our methods across varying numbers of nutrients (denoted as K) to examine the behaviors of the algorithms based on K. As can be seen in Figure 2, grid search performs poorly when selecting only a few nutrients, but performances of all three methods quickly converge after $K = 4$. Binary search method was about 3 times faster than grid search method across different K values. The exact running time of random search depends on the number trials. Increasing the number of trials would naturally increase the running time, but it would also improve the performance of random search.

7. Discussion

We formulated the nutrient supplementation problem by searching for a sparse vector that violates the least number of linear constraints. We presented two solutions, one based on random sampling and a linear programming formulation with a $l_0 \rightarrow l_1$ relaxed constraint. For the linear programming formulation, we further explored two hyperparameter searching strategies, one better performance grid

[1]About $(3 * n^2)/(\pi^2)$, ref: https://oeis.org/A002088

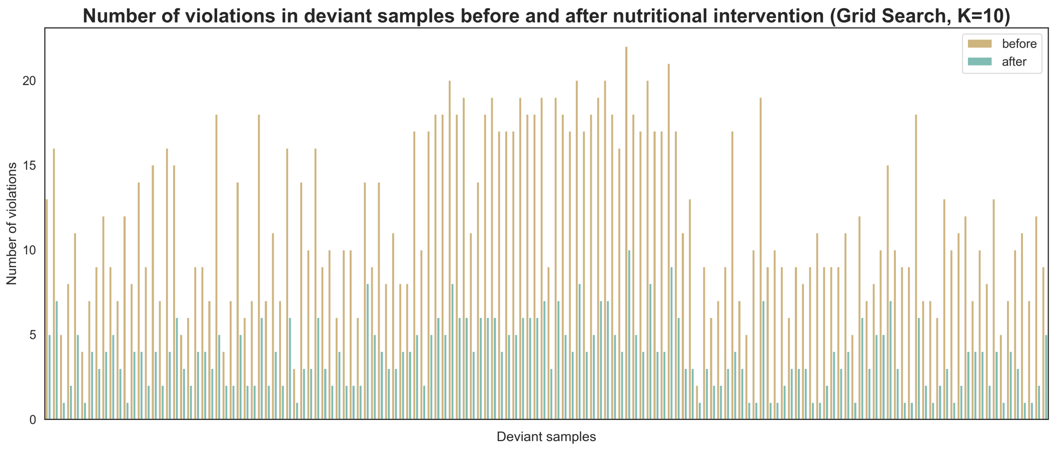

Figure 1. *Number of violations in all 142 deviant samples before and after a nutritional intervention is applied. Yellow and green bars represent the number of violations before and after selecting 10 nutrients as dietary supplementation using a grid search. Binary and random search produced similar results.*

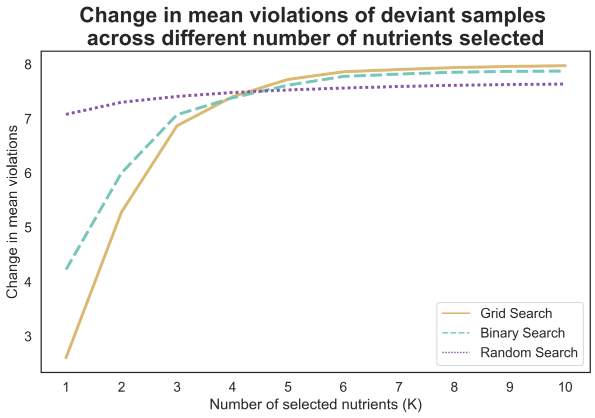

Figure 2. *Change in the mean violations of deviant samples based on a different number of nutrients selected for dietary supplementation. Grid search (yellow solid), binary search (green dashed), and random search (purple dotted) was used to select the optimal subset of nutrients. Random search here ran 1000 trials.*

search, and one more efficient double binary search. Both solutions are competitive and can reduce 60% - 70% violations of deviant samples.

One limitation of this work is the definition of the normal state. While we defined it with low and high boundaries, there could be other heuristics that we weren't able to explore.

Future work could look into more efficient sampling methods, as we demonstrated that random sampling is already effective. For example, one can apply simulated annealing upon randomly sampled interventions. Another direction is to improve the linear programming solution. Possible follow-ups include better search strategies for α and β and a formal proof on the proximity to the optimal solution.

References

[1] Anna Heintz-Buschart and Paul Wilmes. Human gut microbiome: function matters. *Trends in microbiology*, 26(7):563–574, 2018.

[2] Jason Lloyd-Price, Anup Mahurkar, Gholamali Rahnavard, Jonathan Crabtree, Joshua Orvis, A Brantley Hall, Arthur Brady, Heather H Creasy, Carrie McCracken, Michelle G Giglio, et al. Strains, functions and dynamics in the expanded human microbiome project. *Nature*, 550(7674):61–66, 2017.

[3] Stanislav N Iablokov, Pavel S Novichkov, Andrei L Osterman, and Dmitry A Rodionov. Binary metabolic phenotypes and phenotype diversity metrics for the functional characterization of microbial communities. *Frontiers in Microbiology*, 12:653314, 2021.

[4] Moul Dey. Toward a personalized approach in prebiotics research. *Nutrients*, 9(2):92, 2017.

[5] Vasek Chvatal, Vaclav Chvatal, et al. *Linear programming*. Macmillan, 1983.

[6] Carlos Ramirez, Vladik Kreinovich, and Miguel Argaez. Why l1 is a good approximation to l0: A geometric explanation. 2013.

[7] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.