Inferring essential genes using only genomic data

Abstract / TLDR

I'd like to present a project I've been working on for my portfolio. This project, completed over four weeks in my free time, showcases my ability to quickly learn new concepts and develop practical skills. Initially, I aimed to conduct a genome-wide association study but shifted focus due to the scarcity of publicly available genome-phenotype datasets.

Inspired by an intriguing paper which posits that many traits are highly polygenic because mutations in genes with significant effects would be swiftly eliminated by natural selection, I asked the following question:

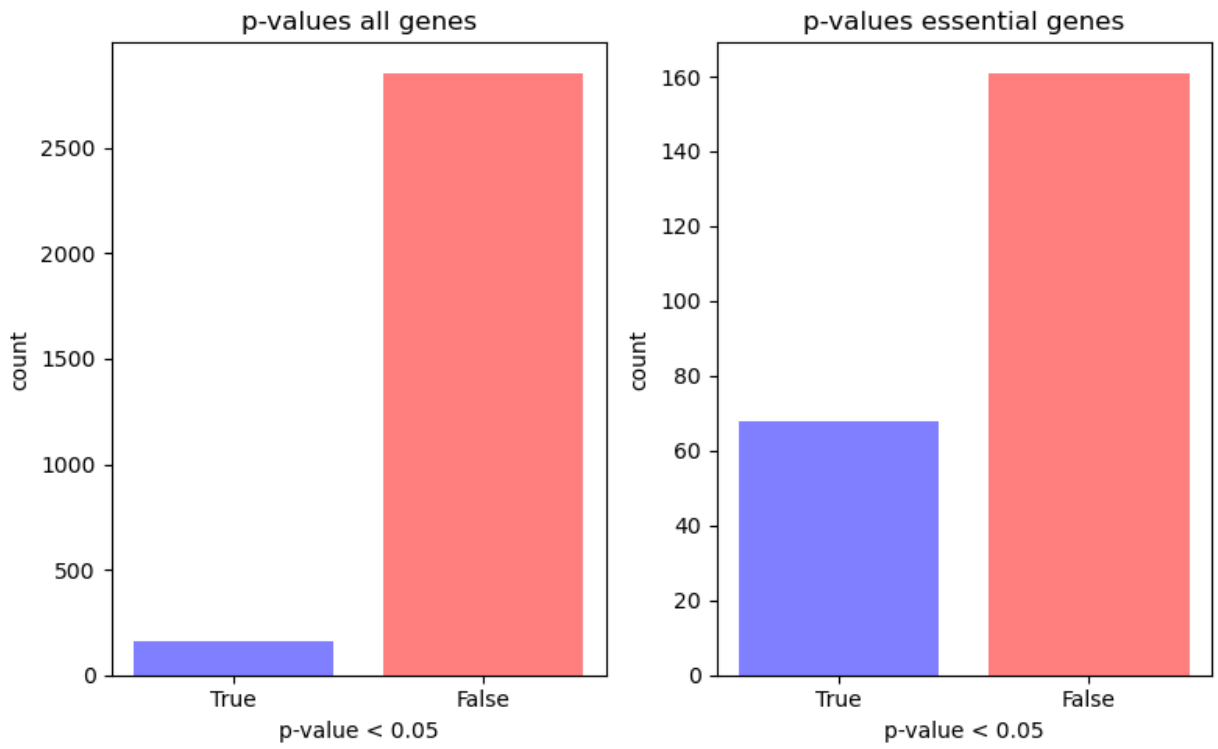
> Is it possible to look at the genomes of a population and infer a-priori which genes are most critical for the survival of the organism, based solely on the frequency of mutations in each gene?

I.e. any critically important gene wouldn't tolerate any mutations but less critical ones would.

For this, I used a dataset of ~3,500 complete, annotated *E. coli* genomes from the NCBI GenBank.

The plan was to identify genes with the lowest rate of mutations per base pair across the whole population, and then cross reference those against a known list of essential e-coli genes found here.

The results validated the hypothesis:



Where the p-value is our estimated probability of the gene having such a low mutation rate under the null. Amongst the essential genes, significantly more are assigned a low p-value when compared to the whole sample. However we can see from the confusion matrix that the predictor still has the majority of positives being false positives, and we also miss the majority of essential genes:

	Essential	Non-essential
Predicted Essential	68	96
Predicted Non-essential	161	2695

Nonetheless, there is signal and the methods could be improved in future work if desired.

Notes:

- My biology background is limited to a first-semester university course.
- I chose an original research question, disregarding whether it might be naïve, already solved, or statistically infeasible. My aim was to showcase my

creativity and independent thinking.

- I opted to implement various components myself, even when existing packages might have solved the problem. I felt that simply downloading and running a sequence of open-source tools wouldn't effectively demonstrate my skills, nor would it have been a good learning experience.

You can check out the code for the project here:
https://github.com/MartinDupont/mutation-rates

Methods

Math

We will refer to each separate sample genome as a specimen, and reserve the word ‘sample’ to describe a sequence for a single gene for any particular specimen.

Lets state our assumptions explicitly:

- Across the population, the most common variant is the optimal variant
- Across the population, at any given locus there is a certain probability of there being a mutation present. We refer to this as the ‘mutation rate’ for lack of a better phrase. (It *isn’t* the probability of a mutation newly occurring in a generation)
- Null hypothesis: This mutation rate is constant at every gene
    - We don’t really believe this hypothesis, but it helps us identify promising candidates
- Alternative hypothesis, different genes have different mutation rates
- The dump of 3500 e-coli genomes is a representative sample.
- Each specimen and each gene is an independent sample of the mutation rate for that gene
    - We expect this to be violated to some degree because the organisms are related. We will see later to what degree this assumption is violated.

- Mutation rate of a gene is inversely related to its importance.
    - This is probably wrong to a degree as we, as many genes probably have sections where they can tolerate changes to their shape and others not. I.e. mutation rate represents flexibility in structure.

The plan is as follows: First, we'll estimate the mutation rate under the null hypothesis. Then, we'll identify genes with the lowest p-values under this null. These genes will be our candidates for anomalously low mutation rates—potentially indicating their importance.

For a given gene, let L represent its length in base pairs, and f be the per-base-pair mutation rate. If f is small and L is large, observing the gene sequence of one sample will approximate a draw from a Poisson distribution. I.e. the probability of k mutations occurring in a gene sequence of length L will be

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad \text{for} \quad \lambda = Lf,$$

Suppose we now have N samples for that gene, and thus have mutation counts $\{k_i\}$ for $i \in \{1 \dots N\}$.

We want to combine these into an estimate of the mutation rate for the gene. If Bayes rule for a single sample is

$$P(\lambda|k) = \frac{P(k|\lambda)P(\lambda)}{P(k)}$$

Following the derivation in the appendix, it turns out that the estimator of λ for N samples with a flat prior is:

$$P(\lambda|k_1, k_2 \dots) \propto \lambda^{\sum k_i} e^{-N\lambda}$$

which is the conjugate prior for a poisson distribution with k' $= \sum k$ and λ' $= N\lambda$.

I.e. we can just pool all the counts together and use them as if they were a single draw from a poisson distribution.

What if now we don’t just take samples across individual genes, but we take across all organisms and all genes in the population? The problem is that the genes are of different lengths $\{L_i\}$. We need to estimate for f, not lambda. We derive in the appendix that we have

$$P(f|k_1, k_2, \dots) \propto (f \sum L_i)^{\sum_i k_i} e^{-f \sum_i L_i} \propto (fL')^{k'} e^{'fL'}$$

which is the conjugate prior for a poisson distribution with k’ $= \sum k$ and λ’ $= f \sum L_i$.

i.e. we can just pool the counts *and* the lengths together to estimate f.

So now our estimate $\hat{f}$ for f, the per base-pair mutation rate across the whole genome is then, somewhat unsurprisingly

$$\hat{f} = \frac{\sum_i k_i}{\sum_i L_i}$$

Our null hypothesis is then that the mutation rate across the whole genome is a constant possion distribution in $\hat{f}$.

More explicitly, the plan is:

- We'll analyze all samples and genes, tallying mutations and summing gene lengths. This allows us to calculate our estimate $\hat{f}$.
- For each gene, we'll sum mutations across all samples. We'll then apply a Poisson distribution to this data. Similar to a GWAS approach, we'll identify genes with unusually low p-values—those with surprisingly few mutations per unit length.

It's worth noting that we're primarily using p-values as a metric for low mutation rates. Consequently, our assumptions can be somewhat flexible without significantly impacting our results.

Data

The dataset comprises all complete, annotated, and assembled Escherichia coli genomes from the NCBI GenBank. This amounts to 3,581 specimens. I focus solely on coding sequences.

Working with E. coli offered several advantages. As a prokaryote, its genomes are small and haploid. Also, the pre-annotated dataset allowed me to bypass various steps in the genomics pipeline, such as assembly, alignment, and annotation.

The files are in FASTA format. Each sequence is tagged with a gene ID and a protein description, followed by the complete sequence. This structure conveniently enables matching sequences between specimens by gene IDs.

The rough workflow is as follows:

- Read each .fasta file
- Process each line in the file, which corresponds to a known coding gene, marked with identifiers that are (mostly) consistent across samples
- Collect each (sample, gene, sequence) triple and store it in a memory-efficient way
    - There is an enormous amount of duplicated sequences, so we can save a lot of memory + cpu by storing each unique sequence only once
        - This already indicates that the independent samples assumption is off. If samples were all independent, given that L is large, we would expect most sequences to be unique
- For each gene, find the most common variant.
- Iterate over all and estimate genome-wide mutation rate with equations above
- For each gene find mutation rate. Sort by p value under null.
- Match all genes against the list of known essential genes, and see if there is a trend

We actually calculate difference on the amino acid sequences, to account for synonymous mutations. Translating into amino acid sequences means that indels, despite being a small change, are read as a large change as they change the reading frame of all the amino acids that follow it. Indels normally result in a

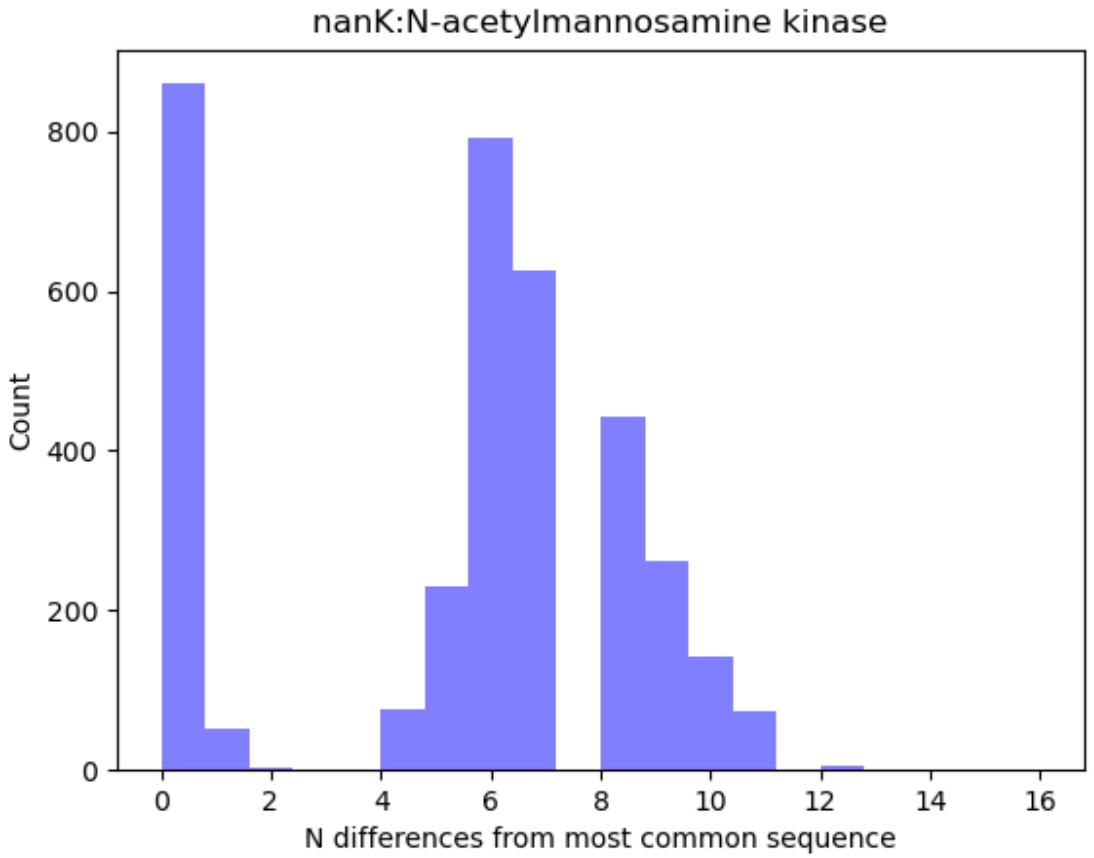
completely different protein, so in this sense we think it’s a good choice, because an indel shows up as a “large” distance in our model, matching the large functional difference.

We also measure the number of mutations using levenshtien distance a.k.a. edit distance. This allows us to deal with sequences of different lengths. The above framework doesn’t account for two copies of the same gene having different lengths, but the actual differences in length are negligibly small in comparison to total length..

Results

So, one realizes a few things pretty quickly.

Samples are strongly non-independent. It turns out that population structure is very important. Half the population might have the most common sequence, a quarter some other sequence, a quarter a different one. Genes of lengths in thousands of base pairs may have less than 10 variants across the sample. Here’s a plot of the counts for an arbitrarily chosen gene:



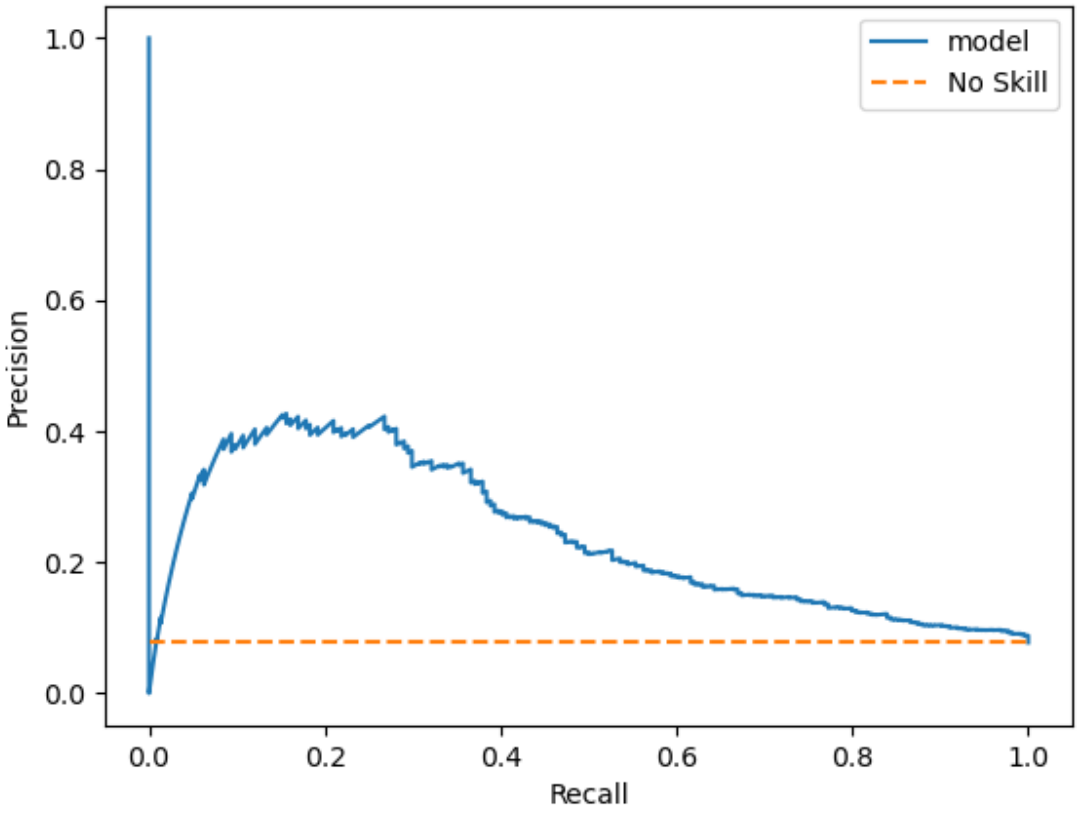
It's quite far from being poisson distributed.

For each gene, the sample size was also large enough to conclusively reject or accept the null hypothesis, and the p-values get driven to 0 or 1. The statistical model was fairly strongly violated.

I pivoted to a simpler approach which worked much better: pick a threshold mutation rate such that across all genomes, about 5% of p-values are less than 0.05, and use that as a selection criteria for flagging essential / non-essential genes. In this sense, the p-value isn't really a statistical estimator anymore, it's rather just being used as a threshold for a classifier.

This approach produced the bar chart and the confusion matrix presented in the introduction.

In addition, we can also take a variable threshold and produce a precision recall curve:



The curve performs significantly better than random chance at almost every threshold. Initially it dips below the no-skill curve because there are a handful of non-essential genes that have a mutation rate of 0, thus at threshold 0 we have terrible performance.

Thus, we can say that the hypothesis has indeed been validated, and we can indeed guess at which genes are critical for an organisms survival just by looking at the mutation rate. This could be useful to discover new biologically important pathways in various organisms.

Appendix: More math

Our model states that mutations should be poisson distributed. By observing the actual mutation counts, we can estimate the mutation rate. We want to calculate how to combine multiple measurements.

Suppose we now have N samples, so we have mutation counts $\{k_i\}, \quad i \in \{1 \dots N\}$.

If Bayes rule for a single sample is

$$P(\lambda|k) = \frac{P(k|\lambda)P(\lambda)}{P(k)}$$

We pick a flat prior i.e. $P(\lambda) = 1$. Then

$$P(\lambda|k_1, k_2, \dots) = \frac{P(k_1, k_2, \dots|\lambda)}{P(k_1, k_2, \dots)}$$

But we said the samples were independent so

$$\frac{P(k_1, k_2, \dots|\lambda)}{P(k_1, k_2, \dots)} = \frac{P(k_1|\lambda)}{P(k_1)} \frac{P(k_2|\lambda)}{P(k_2)} \dots$$

$$P(\lambda|k_1, k_2, \dots) \propto \prod_i \frac{\lambda^{k_i} e^{-\lambda}}{k_i!} \propto \lambda^{\sum k_i} e^{-N\lambda}$$

Remembering that we seek a distribution of lambda values, thus we can drop any terms dependent on k as irrelevant normalization factors.

We can see that this equivalent to the estimator for lambda if we had had $\sum k$ counts over a period N times as long. Great.

Now, the null hypothesis is that the mutation rate is constant across the genome. Thus we want to combine estimates from each specimen and each gene per specimen. The question is how to combine samples when the genes differ in length.

We recall that $\lambda = fL$, i.e. the expected count is the frequency times the length.

Combining samples of different lengths turns out to be easy:

$$P(f|k_1, k_2 \dots) \propto \prod_i \frac{\lambda_i^{k_i} e^{-\lambda_i}}{k_i!} = \prod_i \frac{(fL_i)^{k_i} e^{-fL_i}}{k_i!}$$

Remember that we're looking for P(f) with fixed k discard the k's in the denominator

$$P(f|k_1, k_2 \dots) \propto \prod_i (fL_i)^{k_i} e^{-fL_i}$$

$$\propto f^{\sum_i k_i} e^{-f\sum_i L_i} (\prod_i L_i^{k_i}) \propto f^{\sum_i k_i} e^{-f\sum_i L_i} \frac{(\sum_i L_i)^{\sum k_i}}{(\sum_i L_i)^{\sum k_i}} (\prod_i L_i^{k_i})$$

$$\propto (f\sum_i L_i)^{\sum_i k_i} e^{-f\sum_i L_i} \frac{\prod_i L_i^{k_i}}{(\sum_i L_i)^{\sum k_i}}$$

We can drop the last term because it doesn't depend on f:

$$P(f|k_1, k_2, \dots) \propto (f\sum_i L_i)^{\sum_i k_i} e^{-f\sum_i L_i}$$

which is the conjugate prior for a poisson distribution with $k' = \sum k_i$ and $\lambda' = f\sum L_i$.

The estimate for f is then just going to be :

$$\hat{f} = \frac{\sum_i k_i}{\sum_i L_i}$$