Tentative title: Power analysis of Genome-Wide Association Studies (GWAS) in bacterial pathogens using machine-learning

# Abstract:

Methods for GWAS developed for eukaryotic organisms are not easily translated into bacterial pathogens due to high level of clonality. Using simulation, we explored the statistical power of detecting SNP-associated phenotype traits (e.g., antibiotic resistance or MIC) using a machine learning approach. We first designed an algorithm to simulate SNPs associated with a continuous variable. We then applied machine learning techniques to identify the false positive and false negative errors. The result shows a large number of strains are needed to detect causal SNPs. We further developed methods to simulate phylogeny-associated SNPs, which increased the false positive detection rates. The results show the importance of taking phylogenetic relationship and clonality into account when performing GWAS for bacterial pathogens. Our algorithms and tools could be used for designing GWAS studies in pathogenic bacterial.

# Introduction

GWAS inspect dissimilarities from a reference to detect patterns.

# Material & Methods

## Data sets & pre-processing [tentative head]

## CDG measurements, genes, and SNPs

Original data was shared from xyz containing two tables. One of which, is a table log of repeated experimental samples collected from *Pseudomonas aeruginosa* strains at various tissue sites (Table 1). The use of this particular dataset are the expression levels of c-di-GMP, which in brief, is a secondary messenger related to decision making of whether to form biofilm or leave residence in search of a more favorable environment 11. The c-di-GMP levels serve as our continuous variable Y (predictor variable), which we later merge into a bigger matrix once the values have been normalized for the sheer convenience of a scaled number set.

The other table is a binary SNP matrix of 30 rows and 8,532 columns with rows and columns representing, in respective order, number of observed *P. aeruginosa* strains and SNP sites (Table 2). The binary indicates change (1) or no change (0). SNPs are mined and converted into binary using a pipeline that runs through a series of steps connecting the relationship between nucleotides in their corresponding gene and their alleles in each of the 30 strains. These SNPs will be our independent variable X also known as covariates. It is important to note that coherency of this matrix lies in the information that these genes, house-keeping genes, are either directly or indirectly related to inducing c-di-GMP level expression, and building on their possible importance, there seem to be a large number of variants across the strains and with so much information it can unravel a story about how the bacterium makes its decisions.

To hone in on the pathogenic variant(s), we first combine these two tables with respect to their matching strain names. However, due to the repeated process of collecting c-di-GMP levels in a strain, we resorted to averaging the values by the number of experiments performed for that particular strain. It was necessary to proceed with the averaged value in order for the number of row names to match.

In surveying the merged matrix, we quickly realized that the astronomical dimensions would likely exhaust any model or classifier we use, we decided to reduce the number of covariates. By doing so, it would not only use less computer RAM but also allow us to retrieve faster results in our analysis. Subsequently, we shrunk the columns from 8,532 to 1,078 by removing based on column sums equal to 0, 30, and redundant SNP patterns within genes (these are results). The reason being is that if the SNP site is the same across all 30 strains then its effect on the overall analysis will be futile and would in fact, add more redundancy to the matrix. It would also allow the data to have less variance and in this type of prediction analysis, it is not beneficial to have less spread. Though, this might raise concerns of whether it is legal to reduce based on redundant SNP patterns because [insert reason] but the removal is *within* genes and not its neighbors.

While we did successfully reduce the dimensions of our dataset, it was not ready to be fed into our models. If we blindly use this novel data, there would be no way of interpreting the result thus making it as good as a guess. Therefore, we strategized an alternative by using simulations to test the accuracy of our models before we move forward to testing the real data (discussions).

## Simulation of trait-associated SNPs

Simulation modeling sheds light on the uncertainties of randomness that occur even in genomics. Our approach is to first emulate a small dataset laddering onto the actual dimensions of our initial matrix in order to gauge and optimize a working frame. Second, we use a decomposition algorithm, Cholesky decomposition, to artificially inject a correlated variable (X) to the predictor variable (Y). This step conditionally creates a known relationship between the xy variables. We ran the algorithm several times to produce different sized tables with varying numbers of covariates ranging from [insert numerical range]. Lastly, we used a machine learning algorithm to identify such covariates.

**Simulating SNP-associated phenotypes**

**Tools (R packages) & steps**

The following method is used to simulate SNP-associated phenotypes. For illustration purposes, we only generate two cdg level correlated SNPs in this case. But this method can be extended to generating any number of cdg level correlated SNPs. It is also possible to set the inter-correlation between SNPs.

First, we generate 3 independent vectors of standard normal random numbers with equal length. One vector is for the cdg level, one vector is for the first SNP state, one vector is for the second SNP state. We assume that a scaled cdg level follows the standard normal distribution. Although the SNP state should be either 0 or 1, we can still assume it follows the standard normal distribution and then replace all negative values with 0 and all positive values with 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Cdg\_level | 1.58102905 | 0.80859597 | 1.00966938 | … |
|  | SNP1 | -2.38290958 | -2.11974195 | 2.22425003 | … |
|  | SNP2 | -0.50069548 | 1.31667049 | -0.09234325 | … |

The task is to correlates the SNP states with Cdg levels. To correlate three standard normal random variables, we can apply Cholesky decomposition to a desired covariance matrix and multiply the decomposed matrix with these three standard normal variables. Denote cdg level, SNP1 state, SNP2 state as random variable with standard deviation . The covariance matrix A can be written as follows:

is the covariance between and

In this case, all X’s are standard normal random variables. Therefore, they have a standard deviation of 1 and the covariance is the same as correlation. Matrix A can also be written as follows:

is the correlation between and

Apply Cholesky decomposition 2 to matrix to A, we get a lower triangle matrix L and its transpose LT.

Therefore, we get 3 correlated random variable with a covariance matrix A (in this case also correlation matrix) by doing the following:

are the three random arrays we generated in the beginning for the cdg level, first SNP state and second SNP state. Therefore, becomes the corresponding sequences for cdg level, SNP1 and SNP2. Since the SNP states can only be 0 or 1, we can replace the positive numbers in by 1 and let everything else be 0.

The following table is an output sample with cdg levels, SNP states. The input covariance is 0.5 between SNPs and cdg level, and 0.2 for the inter correlation between SNPs. We calculated the actual covariance for simulated data. As can be seen from the simulated covariance table, the covariance is very close to what we assigned in the beginning.

Output Sample

|  |  |  |  |
| --- | --- | --- | --- |
| Strains | cdg level | SNP1 | SNP2 |
| 1 | 1.963056 | 1 | 1 |
| 2 | -0.74528 | 0 | 1 |
| 3 | 0.839563 | 0 | 1 |
| 4 | -2.06734 | 0 | 0 |
| 5 | -0.00585 | 1 | 0 |

Input Covariance Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Correlation | cdg | SNP1 | SNP2 |
| cdg | 1 | 0.5 | 0.5 |
| SNP1 | 0.5 | 1 | 0.2 |
| SNP2 | 0.5 | 0.2 | 1 |

Covariance of Simulated Data

|  |  |  |  |
| --- | --- | --- | --- |
| Correlation | cdg | SNP1 | SNP2 |
| cdg | 1 | 0.502134 | 0.441385 |
| SNP1 | 0.502134 | 1 | 0.2078699 |
| SNP2 | 0.441385 | 0.2078699 | 1 |

Applying this method, we can simulate SNPs with desired correlations, and the correlation between SNPs can also be manipulated. However, in order to apply Cholesky decomposition, the covariance matrix (in this case also correlation matrix) should be positive definite[[1]](#footnote-1), which means the correlation input should be carefully selected. This method can be easily extended to generate any number of SNPs with desired correlations with the cdg level and reasonable inter-correlation among SNPs simply by enlarging the covariance matrix.

**Simulate tree-associated phenotypes**

In this method of tree-based simulation, we use Cholesky decomposition again to get our correlated variable. However, it will slightly differ from our first method as this takes in consideration of evolutionary relationships and that we only simulated one correlated variable. First, we created a tree object with the APE package in the RStudio interface 4 and used that to simulate a variance-covariance matrix based on the evolved traits of the tree. The topology and relationship of the tree tips can easily be seen and described in the produced matrix. For illustration purposes, we will use only a sample of the real dataset.

V: Tree-based covariance matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PA14 | M37351 | M1608 | F34365 | H47921 | T6313 |
| PA14 | 0.01110802 | 0.01082449 | 0.01082449 | 0.00478838 | 0 | 0 |
| M37351 | 0.01082449 | 0.01125653 | 0.01119291 | 0.00478838 | 0 | 0 |
| M1608 | 0.01082449 | 0.01119291 | 0.01134418 | 0.00478838 | 0 | 0 |
| F34365 | 0.00478838 | 0.00478838 | 0.00478838 | 0.00903998 | 0 | 0 |
| H47921 | 0 | 0 | 0 | 0 | 0.00696731 | 0.00312532 |
| T6313 | 0 | 0 | 0 | 0 | 0.00312532 | 0.00978805 |

In the above matrix, it’s evident that (PA14, M37351, M1608, F34365) are in the same clade whereas (H47921, T6313) are distantly related in a separate clade. The diagonal values are rate estimates of self, and the non-diagonal values are rate estimates between two traits. The values are estimated using a Brownian motion model capturing the convergence of evolutionary relationships which are then verified using generalized least square models 5.

Next, we created two additional matrices that are essential for matrix multiplication. One, a randomized standard normal matrix with n number of (characters/tree tips) to mold the shape of the output. Another is a stand-alone variance-covariance matrix with no association with the tree. is what we use to set correlation between tip and trait or in other words, cdg-level and SNP. The correlation coefficient between the cdg-level and the SNP is set at 0.9.

X: Standard normal matrix

|  |  |
| --- | --- |
| 0.18802917 | -0.5907137 |
| 1.13847969 | -0.5096937 |
| -1.2191201 | -1.8405232 |
| 0.58918957 | -0.1485317 |
| -1.1600177 | 1.93584266 |
| -0.1380378 | -1.397533 |

R: Tree-free var-covar matrix

|  |  |
| --- | --- |
| 1 | 0.9 |
| 0.9 | 1 |

Second, we apply Cholesky decomposition on and to get our coefficients. We then use that to perform matrix multiplication on three matrices , , and , with and transposed. [yinheng can you plz draw an matrix multiplication: dim(2 6) x dim(6 6)]

|  |  |  |
| --- | --- | --- |
| **PA14** | 0.019817256 | -9.302113e-03 |
| **M37351** | 0.049611505 | 1.229244e-02 |
| **M1608** | 0.029258089 | -1.709763e-02 |
| **F34365** | 0.061979149 | 3.705461e-02 |
| **H47921** | -0.096827124 | -1.671091e-02 |
| **T6313** | -0.056074563 | -7.465812e-02 |

The output above describes two vectors that are strongly correlated with one another. We proceed to binarize the second vector using pnorm to [yinheng plz complete this] and expending those values in a binomial distribution with one success trial and 50% probability to yield a discrete number. We repeat this discretize step to generate n number of random false positive and true negative SNPs. The first vector is left at its continuous state acting as our predictor variable Y or cdg-level.

Finally, we merge the randomized SNP matrix with the two correlated variables to use as our xgboost input. The correlation between the continuous variable Y and discrete variable X is calculated, using Pearson’s method, prior to running the algorithm so we know what to expect of the xgboost output. The sensitivity of the algorithm becomes apparent when the two target variables are highly correlated with a coefficient of >= 0.6 [don’t have the exact number but roughly around that range]. Statistical analyses are performed using CRAN’s core statistic package.

**Tools (R packages) & steps**

## Identification of causal SNPs using xgboost

What is xgboost algorithm with reference;

your customized scripts to run xgboost (description in words & equations & then commands/scripts as supplemental material)

Even after we reduce the dimension of our dataset, that is from 8,532 columns to 1,078 columns, the dataset still has too many features than observations. Based on this fact, it is impossible to implement regular regression methods on this dataset. Therefore, we decided to try machine learning techniques to see if they can produce some significant outcomes on this dataset.

The first machine learning method we tried is XGBoost. XGBoost is a scalable end-to-end tree boosting system which is widely used on many machine learning challenges. It can be used in classification problems as well as regression-based problems. In our dataset, we used it in a regression-based fashion. We treat the cdg level as the dependent variable and the SNP states as the independent variable. Slightly different from regression analysis, we are not trying to estimate the parameters for explanatory variables. Instead, our goal is to estimate which independent variable (SNP) has the most significant impact on the dependent variable.

Being a gradient boosted trees algorithm, XGBoost takes the parameters such as ‘max\_depth’, ‘eta’, ‘gamma’, ‘num\_rounds’. After running through the whole dataset, it can produce the significance level of each independent variable with regard to the dependent variable. In our case, it is able to output the SNPs which are most associated with the cdg level.

In determining the effectiveness and accuracy of XGBoost, we used simulated data as a benchmark. The method is stated in section (above). We first simulate two SNPs whose states are correlated with the cdg level. Then we mix them with other SNPs which have completely random states. We feed the simulate dataset to XGBoost and check the feature\_importance function to see if those two correlated SNPs are assigned high importance in XGBoost. Feature importance in XGBoost is measured in three ways, namely “cover”, “gain” and “frequency.” In our case, we use “gain” as the main measurement for feature importance. We also generated different sized dataset to evaluate the performance of XGBoost by and to determine how many observants it require to achieve a satisfactory accuracy. (Here is the result)

The picture and more discussion in the picture.

Based on the same idea, we used a simulated dataset to tune the parameters of XGBoost. We kept adjusting different parameters until it reaches the maximum predication accuracy in the simulated dataset. The optimized parameter turns out to be

'max\_depth':3, 'eta':0.03, 'gamma':0.3, ‘num\_rounds’:3

**Dimension reduction methods based on normalized mutual information**

In order to obtain an optimal selection of features and reduce the dimension of our dataset, we can extract features with largest dependency from the existing feature set. Filtering features based on their mutual information scores could be an effective way in removing redundant SNPs and improving XGBoost performance.

Mutual Information (MI) is a nonparametric measure of relevance between two variables. It reflects the shared information in two variables. The idea of using MI to reduce the dimension is that by calculation the mutual information between all features, we want to exclude features which carry the similar information as the features we have already selected from our dataset. In this way, the size of the dataset is reduced. Mutual Information ***I*** of two discrete random variables *X* and *Y* can be defined as6:

(1)

where is the joint probability function of *X* and *Y*, and and are the marginal probability distribution functions of *X* and *Y* respectively.

When *X* is a continuous random variable and *Y* is a discrete variable, Mutual information ***I*** is calculated in this way7:

(2)

To better evaluate MI between SNP states in our case, we used a scaled MI which is called normalized mutual information (NMI). It is defined as:

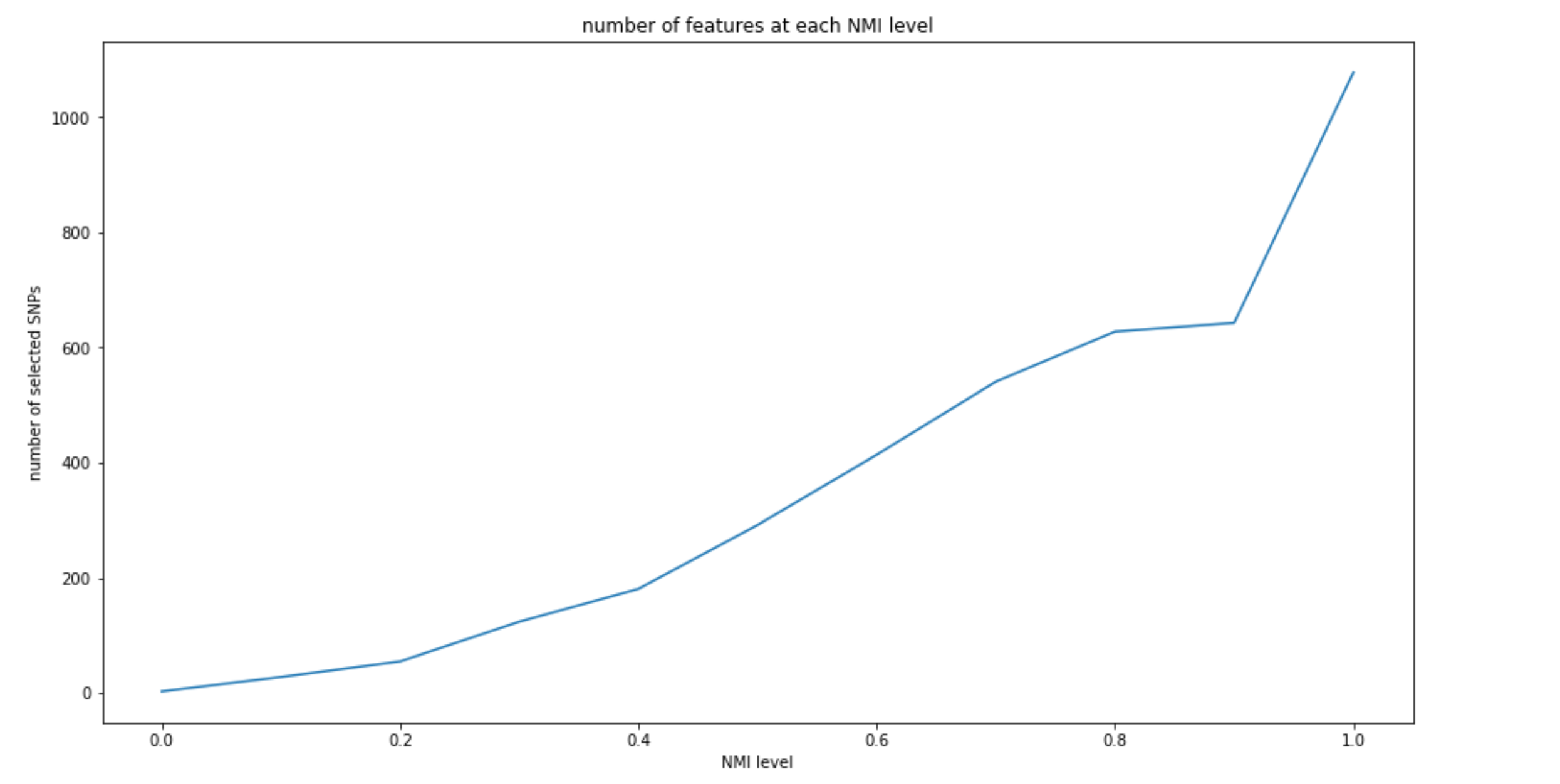
(2)

where denote the entropy of *X* and . In this way, the NMI is always between 0 and 1, and as desired. Next, we apply NMI dimension reduction method to our dataset to see if we can remove more SNPs from our features and improve the XGBoost performance.

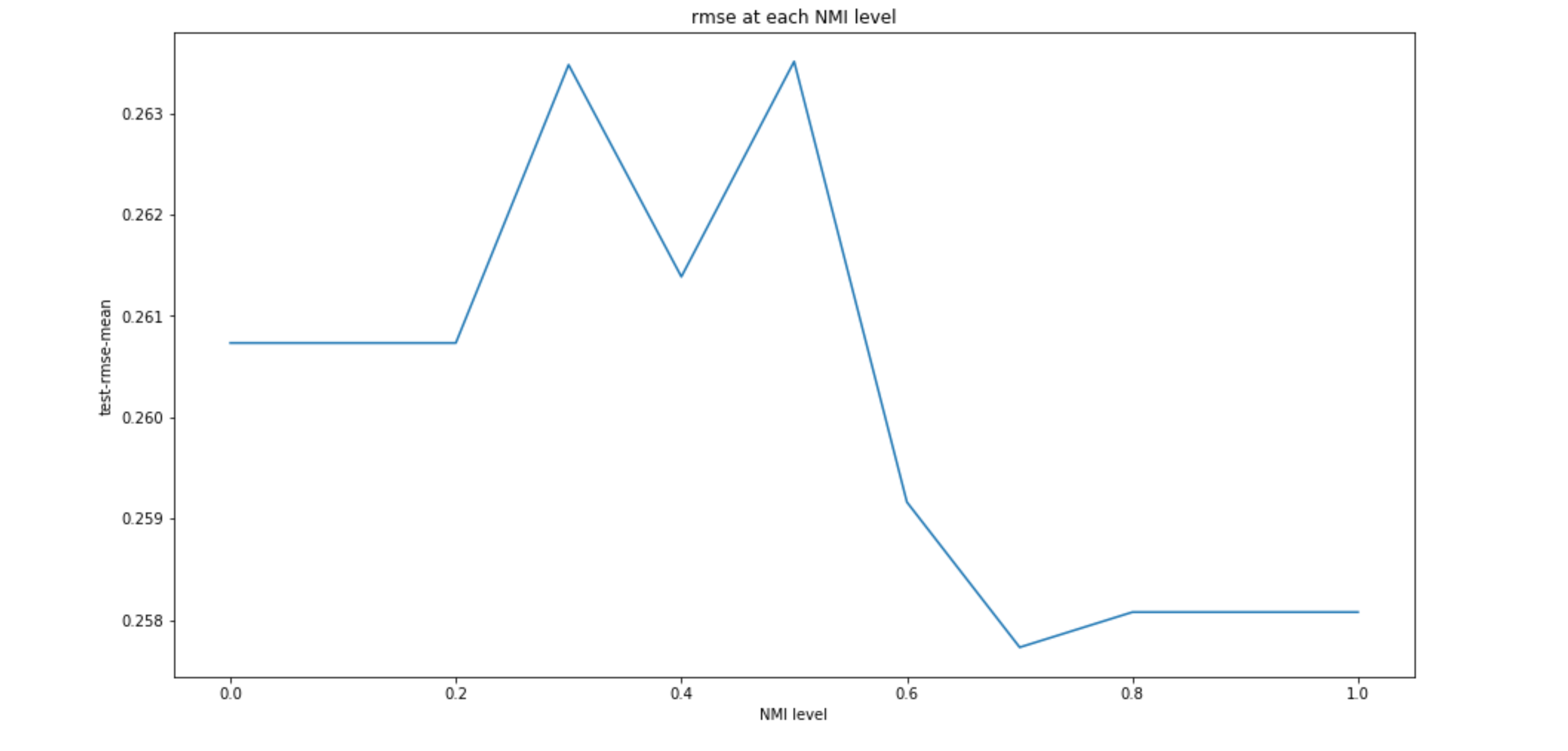
Here are the steps:

1. Initialization: We select the SNP which has the highest NMI with the cdg level as the first feature SNP in the selected feature set. The reason we select the first SNP in this way is that we believe the highest NMI with the cdg level indicates a higher chance that this SNP is influential to the cdg level.
2. Adding SNPs to selected-feature set: First, we set an NMI benchmark. Based on the benchmark, we calculate the NMI of each unselected SNP with each selected SNP. We add the unselected SNP to our selected-feature set only if it has the NMI which is lower than the benchmark with all the selected SNPs. In this way, the benchmark excludes all SNPs with a high NMI. In other words, it excludes SNPs which carry similar information.

To check the effectiveness of this dimension reduction method, we run XGBoost on reduced datasets with different NMI benchmark levels. The original dataset is the one with NMI=1.0. As the graph shows, when NMI decreases, the size of the dataset also decreases.



Then we use cross-validation to see the prediction error with different NMI levels. This graph shows the root mean square error in the test from XGBoost at each NMI level. One can easily see the error reaches the minimum when NMI=0.7, that is, when we exclude all SNPs with , the prediction error reaches the minimum.



It is not guaranteed that a dimension-reduced dataset will always perform better in XGBoost, but still it is worth trying to reduce the dimension while implementing machine learning algorithms on a dataset, as it not only removes the noise in the dataset, but also speeds up the training and testing process in those algorithms.

# Results

# Discussion

# Conclusion

# References Cited

1. Yan, J. *et al.* Bow-tie signaling in c-di-GMP: Machine learning in a simple biochemical network. *PLOS Comput. Biol.* **13,** e1005677 (2017).

# Table 1.Two phenotypes: Strains and their normalized c-di-GMP levels & antimicrobial sensitivity indices (Mei – should I make in a separate table)

|  |  |  |  |
| --- | --- | --- | --- |
| Strain | Body site | Cdg-level (mean) | Cdg-level (standard deviation) |
| PA14 | NA | 2.11071438 | 0.11249807 |
| M55212 | Blood | 0.36094968 | 0.14645814 |
| F22031 | Pubic bone | -1.66607404 | 0.20298575 |
| W25637 | Sputum | -0.16582713 | 0.05295834 |

# Table 3. List of c-di-GMP genes (Mei)

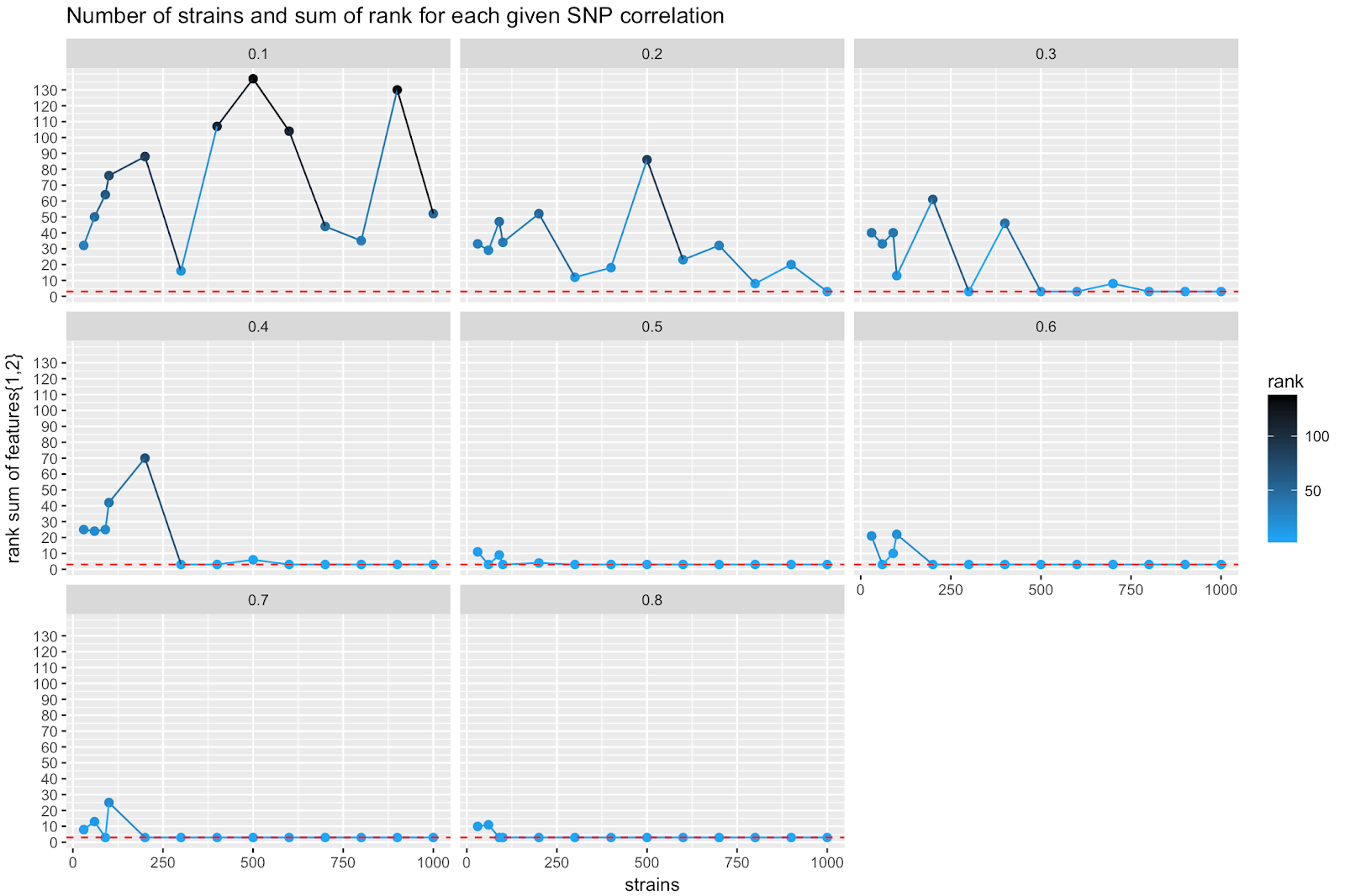
|  |  |  |
| --- | --- | --- |
| Gene symbol | Gene annotation | Num of SNPs |
| B136-33\_52\_00507 | HDOD domain || Metal-dependent hydrolase HDOD | 727 |
| B136-33\_52\_01564 | CSS motif domain associated with EAL || Putative cyclic diguanylate phosphodiesterase, CSS motif-containing domain | 1622 |
| B136-33\_52\_01815 | GlnD PII-uridylyltransferase || PII-uridylyltransferase/Glutamine-synthetase adenylyltransferase | 494 |

# Table 4. List of antibiotic-resistant genes (Mei)

|  |  |  |
| --- | --- | --- |
| Gene symbol | Gene annotation | Num of SNPs |
| NalD / PA3574 | NalD is a repressor of MexAB-OprM. Mutations lead to multidrug resistance and MexAB-OprM overexpression. | 5 |
| MexA / PA0425 | MexA is the membrane fusion protein of the MexAB-OprM multidrug efflux complex. | 4 |
| OprD / PA0958 | Basic amino acid, basic peptide and imipenem outer membrane porin OprD precursor | 27 |

# Figure 1. Flowchart of analytical protocol

# Figure 2. Power analysis using simulated data (show simulated datasets)



# Figure 3. SNPs & genes associated with cyclic-di-GMP expression

1. Yan, J. *et al.* Bow-tie signaling in c-di-GMP: Machine learning in a simple biochemical network. *PLOS Comput. Biol.* **13,** e1005677 (2017).

2. Paradis, E. *Analysis of phylogenetics and evolution with R*. (Springer, 2012).

3. van den Bos, A. *Parameter Estimation for Scientists and Engineers*. (John Wiley & Sons, Inc., 2007). doi:10.1002/9780470173862

4. Paradis, E. *et al.* *ape: Analyses of Phylogenetics and Evolution*. (2018).

5. Garland, Jr., T. & Ives, A. R. Using the Past to Predict the Present: Confidence Intervals for Regression Equations in Phylogenetic Comparative Methods. *Am. Nat.* **155,** 346–364 (2000).

6. COVER, T. M. & THOMAS, J. A. ELEMENTS OF INFORMATION THEORY. 774

7. Shadvar, A. Dimension Reduction by Mutual Information Feature Extraction. *Int. J. Comput. Sci. Inf. Technol.* **4,** 13–24 (2012).

1. In linear algebra, a symmetric n × n real matrix M is said to be positive definite if the scalar is strictly positive for every non-zero column vector z of n real numbers. Here denotes the transpose of 3. [↑](#footnote-ref-1)