CLASSIFICATION METHODS TO PREDICT TAXONOMIC GROUPS FROM AMINO ACID COMPOSITIONS

Xuran Feng

Supervised by Dr. Edward Susko

Submitted in partial fulfillment of the requirements for the Degree of Bachelor of Science: Combined Honors Co-op in Statistics and Actuarial Science

Dalhousie University

Halifax, NS

December 23rd, 2020

Table of Contents

Acknowledgments

I would like to express my deepest appreciate to my supervisor, Dr. Edward Susko, who introduced me the basic background of Evolutionary Biology, this thesis would not exist without his support, inspiration and patient advice, especially during COVID-19 pandemic. I am pleased to do my undergraduate studies in the Department of Mathematics and Statistics at Dalhousie University. I would like to thank Dr. Bruce Smith and Dr. Gábor Lukács for their guidance and constructive suggestions in my second- and third-year study. Thanks should also go to Dr. Toby Kenny and Instructor Iain Beaton, for their support, communications, and valuable insights into my actuarial science study.

I would like to extend my gratitude to my friends, my parents, for their unwavering support, encouragement, and love.

1. Introduction

Proteins are built by amino acids and can be considered as a string of amino acids. The genome of a species includes all protein coding genes. These genes are in 1-1 correspondence with the proteins they produce. There are 20 standard amino acids with corresponding one-letter codes are $A, R, \ldots, V$. The goal of this paper is to predict taxonomic groups from the genome-wide amino acid compositions of species and to find the key predictors by using different statistical methods. We let $\boldsymbol{X}$ denote our predictor vector, giving the amino acid composition for a species. Specifically, the entries are the proportions of times each of these amino acids came up in a subset of genes for the taxonomic group. Consequently, the sum of the $\boldsymbol{X}$ entries always equals 1. There are totally 10 taxonomic groups and 765 observation points in our study data set, and the majority of the data points are belonging to taxonomic group B (Bacteria candidate phyla), We will use the short forms given in the 2nd column of *Table 1* to represent taxonomic groups in the following analyses for ease of interpretation.

An alternative set of predictors that we will consider are GARP and FYMINK. These are created by summing the specific amino acid frequencies given in their names. The codons, or three-letter DNA codes for GARP (glycine, alanine, arginine, and proline) have guanine (G) and cytosine (C) in the first two codon positions. The codons for FYMINK (phenylalanine, tyrosine, methionine, isoleucine, asparagine, and lysine) have adenine (A) and thymine (T) in the first two codon positions. We can use likelihood ratio test or compare the cross-validation errors between the full model (predictors: 20 amino acids) and the reduced model (predictors: GARP and FYMINK) to check if the reduced model is acceptable for us.

Taxonomic group name	Short forms used in analyses	Number of representatives
Thermophiles	T	37
Chloroflexi	C	28
Cyanobacteria/Melainabacteria group	C/M	40
Firmicutes	F	66
Actinobacteria	X	69
Bacteria candidate phyla	B	363
PVC group	P	57
Spirochaetes	S	11
FCB group	f	84
Acidobacteria	Y	10
		Total = 765

Table 1: Taxonomic groups in the study and number of representatives of each group

2. Methods

2.1. Tree-Based Method (Classification Tree)

Tree-based method are one of powerful predictive modelling approaches used in statistics, data mining and machine learning, we would use it to start our analysis. Whether Regression tree or classification tree will be built depends on whether the response variable is continuous or categorical. We would like to predict the taxonomic group labels from the amino acid compositions. Since our response variable is categorical, a classification tree is desired and will be constructed in the following.

2.1.1 Partitioning Method

Recursive binary partitions can be used to grow the tree, as shown in *Figure 1*, we need to choose predictor variable and split-point at each branch so that one split-point can split into two

regions in order to achieve best fit. For example, we first split at $X_1 = t_1$. Then the region $X_1 \leq t_1$ is split at $X_2 = t_2$ and the region $X_1 > t_1$ is split at $X_1 = t_3$. Finally, the region $X_1 > t_3$ is split at $X_2 = t_4$. The result of this process is a partition into the five regions $R_1, R_2, \cdots, R_5$ shown in the figure.

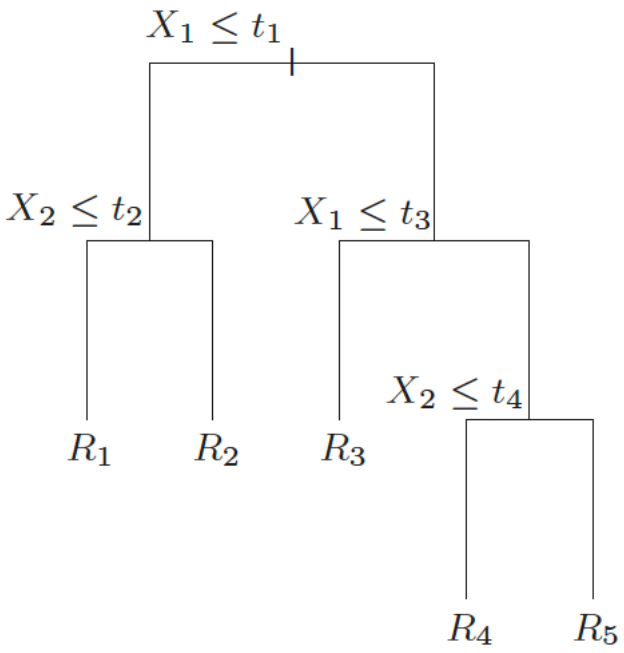

Figure 1: Simple example of decision tree

We now turn to the question of how to choose the predictor variables and split points. In the classification tree, there are 3 node impurity measures to decide the optimal split from a root node, and subsequent splits, which are misclassification error, Gini index, and cross-entropy/deviance. We want the split point to make two separate nodes such that class labels tend to differ in the two resulting nodes. Small node impurity measure tends to correspond to well separate classes. We minimize the node impurity measure used in each node to grow the optimal tree. The cross-entropy formula is

$$-\sum_{k=1}^{K} \hat{p}_{mk}\, log(\hat{p}_{mk}),$$

where $\hat{p}_{mk}$ is the proportion of class k observations in node m (i.e., included both internal nodes and terminal nodes).

In this paper, we choose cross-entropy as our impurity measure, since it is more related likelihood ratio test, based on the equation above, minimizing cross-entropy is equivalent to maximizing the likelihood statistics for a test with null hypothesis is two sister nodes are homogeneous. Large likelihood ratio suggests the two nodes are very different.

2.1.2 Pruning the Tree

Using the partitioning method alone usually results in larger and difficult to interpret trees. It usually fits the training set too well and reduces accuracy in the test subset. To avoid overfitting, Breiman *et al.* (1984) developed the methodology, called cost-complexity pruning to cut size of trees. Suppose $\alpha \geq 0$ be a real number called the complexity parameter (cp). Let R be a criterion value for the tree, such as the sum of cross-entropy in each terminal node and let *size* of tree be the number of leaves (terminal nodes). Then Breiman *et al.* define

$$R_\alpha = R + \alpha \cdot size$$

to be cost of the tree and minimize this cost-complexity measure. Small α results in larger trees and potential overfitting, large α in small trees and potential underfitting. We need a way to choose the optimal α. If a separate validation set exists, we can use it as the test set and compute the deviance versus α for the pruned tress. Then, this will have the considered range of trees and we can choose the smallest tree whose deviance is close to the lower bound of the range.

Otherwise, cross-validation can be used, this is what we do in the study. Suppose we randomly split our data into 10 equally sized subsets, then use 9 subsets to train and the rest of one to test, this can be done by 10 ways, finally average the 10 performance measures (i.e., cross-entropy). The α is the one that minimizes the cross-validation error. This process called 10-fold cross-validation.

2.1.3 Implementation in rpart

The rpart algorithm (Terry & Beth, 2019) that we used to fit classification trees is part of the base R implementation, and it works by splitting the dataset recursively to grow a tree until a predetermined criterion reached. In the classification tree, rpart function offers Gini index (default impurity criterion) and entropy index methods as choices, entropy index can be used by adding the argument parms = list(split = "information"). Besides, rpart runs 10-fold cross-validation by default, the optimal cp value can be found to obtain the cost of tree R_α by running 10-fold cross-validation. Growing a tree can be easily overfit our data, 10-fold cross-validation aims to avoid overfitting.

We can use plotcp() to extract cross-validation results. *Figure 2* is the results using a small maximum cp of 0.001 to keep the tree as large as possible first.

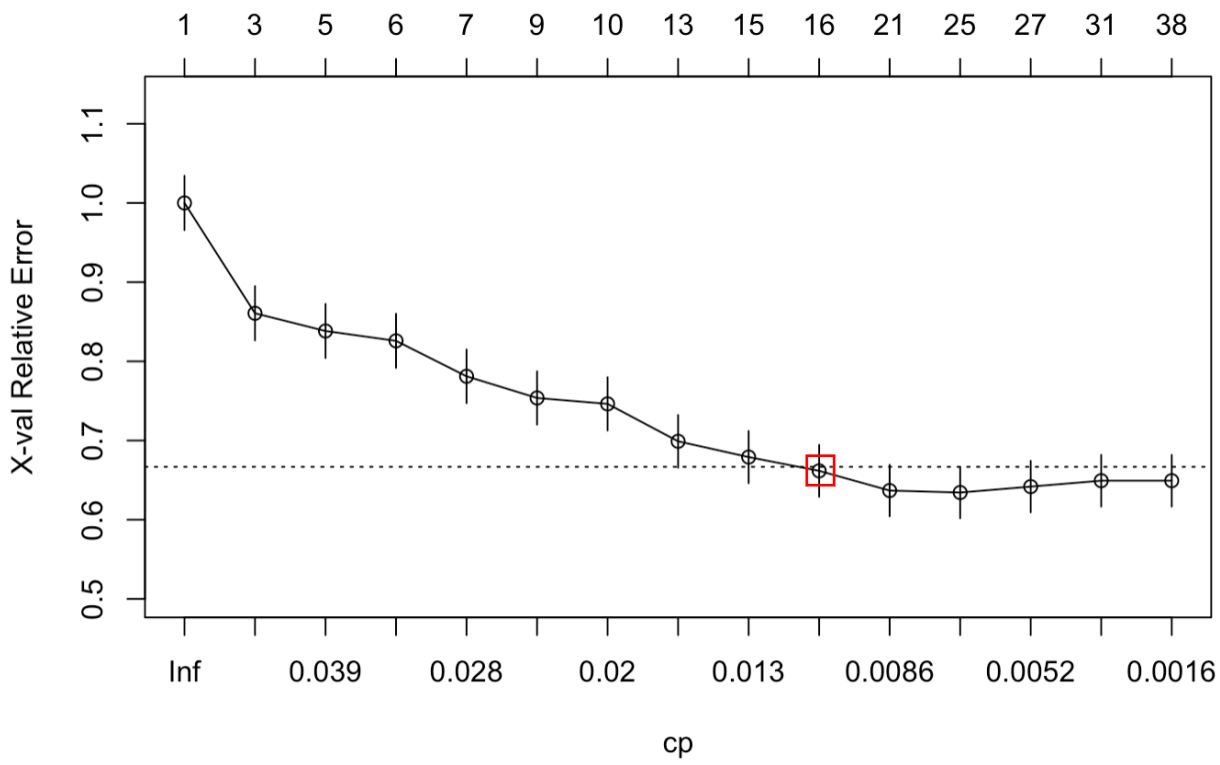

Figure 2: Plot by plotcp (upper horizontal axis is size of tree, y-axis is cross-validation error)

The dashed line in *Figure 2* is the sum of the cross-validation and its standard deviation within minimum cp value, it is common to use the smallest tree within 1 standard error (SE) of the minimum cross-validation error, this is called the 1-SE rule. Then we apply 1-SE rule to check the suggested size of tree and choose a tree with 16 leaves.

rpart.plot (Stephen, 2020) can be used to visualize the tree, but it constructs the tree with size corresponding to the last point from plotcp, e.g., rpart.plot will build a tree with size 38 instead of 16 if cp is still 0.001. To comply with 1-SE rule, we adjust cp value to 0.01 to grow a tree with size 16, see *Figure 3*.

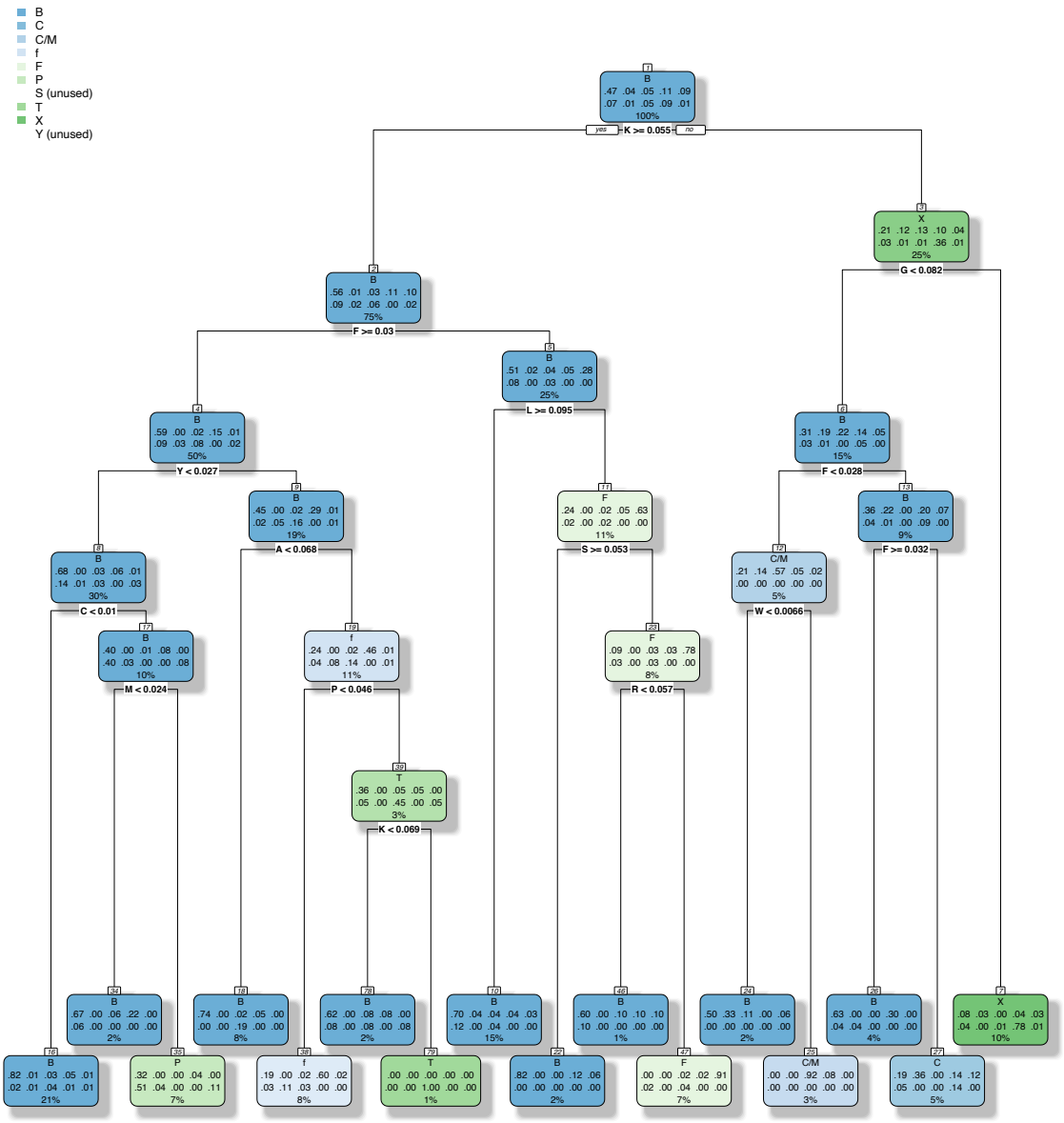

Figure 3: Classification tree with size 16 (predicted probabilities of each group shown in each box, probability order follows the order in top left legend)

According to *Figure 3,* most of the classification rules lead to group B (Bacteria candidate phyla). This is in part because Bacteria candidate phyla is the most common taxonomic group in our study dataset. Almost all groups can be predicted by the tree, except Spirochaetes (S) and

Acidobacteria (Y), these two groups are never predicted by this tree, and thus are marked unused in the legend. We also calculate cross-validation (CV) error by writing a function, detailed R code can be seen in Appendix. The basic idea is using different 9 node tree built separately from each training set and then tested on the test set, finally average the 10 cross-entropy. CV error under classification tree is 38.4%. We will use other methods to analyze our data and compare CV error among them in the following sections.

2.1.4 New Predictors: GARP & FYMINK

We will use classification tree methods with new predictors in this section. The response variable is still taxonomic group, but the predictor variables are GARP and FYMINK, which are two combinations of amino acid compositions. GARP is value of sum of amino acid frequencies G, A, R and P, similarly for FYMINK.

Similarly, by using impurity measure cross-entropy and choosing cp = 0.001 to keep the tree as large as possible first, we got *Figure 4*. According to 1-SE rule, the suggested tree size here is 3. Then, we adjust the cp value to 0.02, and use rpart.plot to view the pruned tree (*Figure 5*).

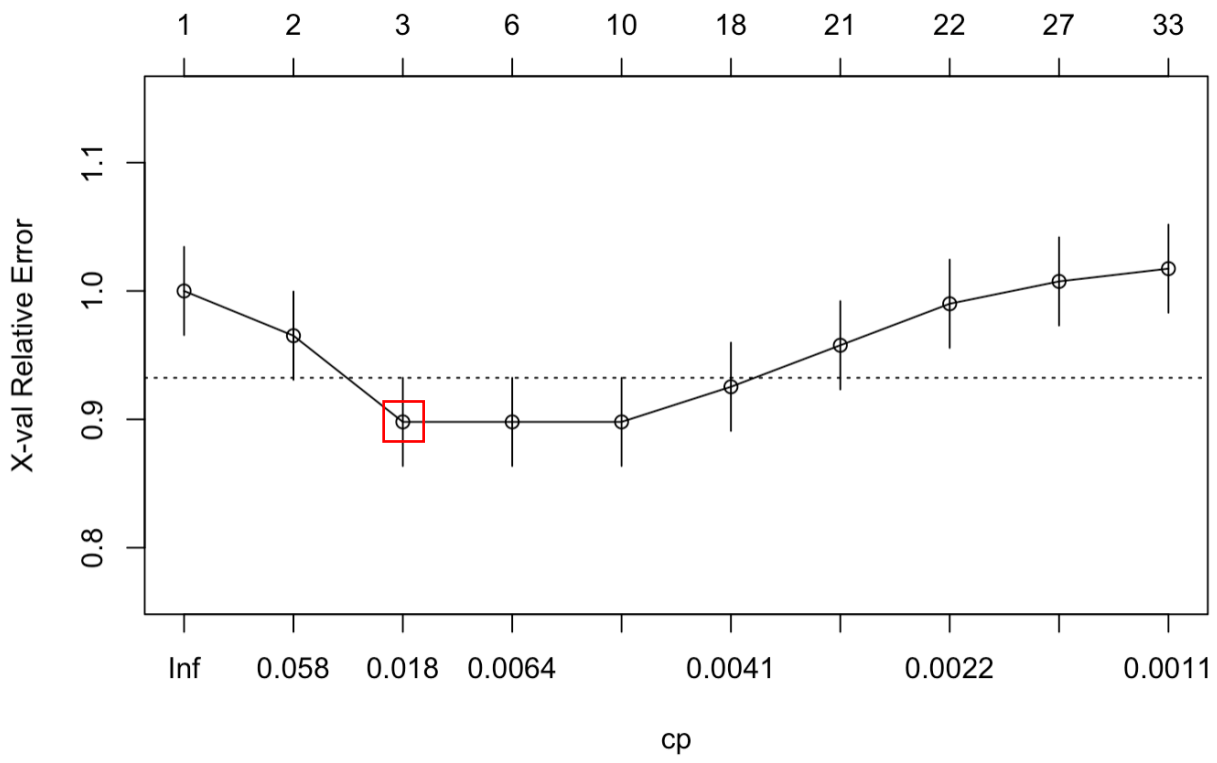

Figure 4: Plot by plotcp of predictors GARP and FYMINK

Based on Figure 5, it is simpler tree than with regular predictors, we can conclude that group B (Bacteria candidate phyla) and X (Actinobacteria) are two common taxonomic groups. The cross-validation error here is 47.2%, which is higher than CV error of the regular predictors.

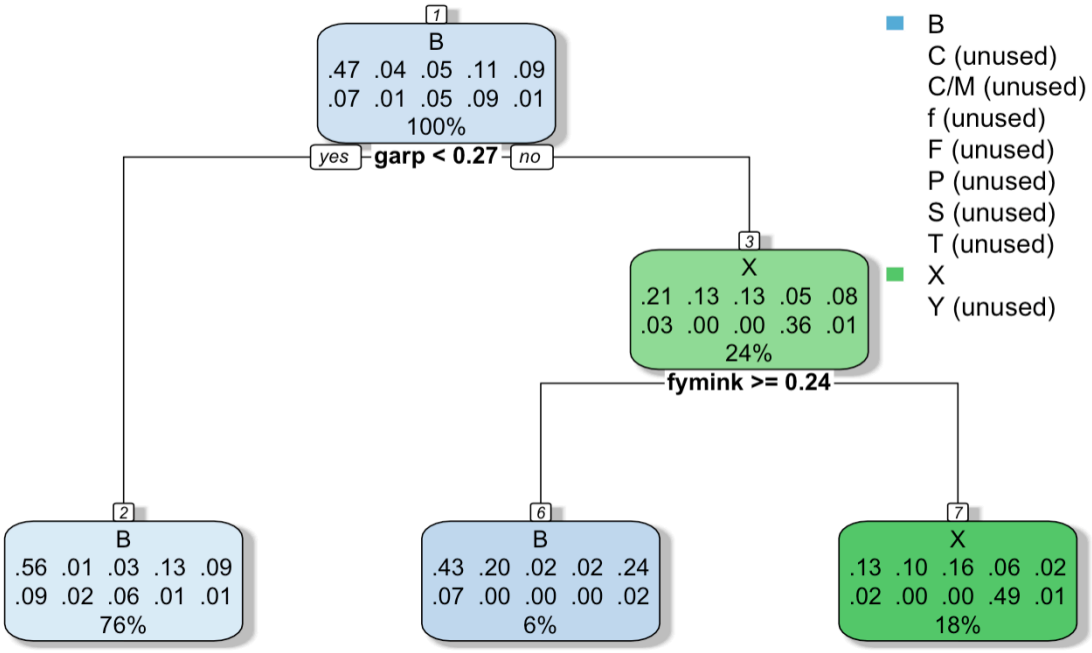

Figure 5: Classification tree with predictors GARP and FYMINK (predicted probabilities of each group shown in each box, probability order follows the order in top right legend)

2.2. Principal Component Analysis (PCA)

PCA is one of the dimension reduction methods. It aims to summarize, in an optimal way, the variation in a set of unordered and correlated variables $\boldsymbol{X}$ by constructing a new set of ordered and uncorrelated variables by a few linear combinations $Z_i = \boldsymbol{c_i'X}, i = 1, \dots, p$, such that

i^{th} principal component (PC) = linear combination $\boldsymbol{c_i'X}$ that maximizes

$$\text{Var}(Z_i) = \boldsymbol{c_i'\Sigma c_i} \text{ subject to } \boldsymbol{c_i'c_i} = 1, \text{ and}$$

$$\text{Cov}(Z_i, Z_j) = \boldsymbol{c_i'\Sigma c_i} = 0, \ j < i$$

Since covariance matrix Σ is positive definite matrix, there exists an orthogonal matrix $\mathbf{E}$ such that $\mathbf{E}'\mathbf{\Sigma}\mathbf{E} = \mathbf{\Lambda}$, where $\mathbf{E}'\mathbf{E} = \mathbf{E}\mathbf{E}' = \mathbf{I}_p$, and $\mathbf{\Lambda}$ is diagonal matrix of eigenvalues in decreasing order, i.e., $\mathbf{\Lambda} = diag(\lambda_1, \dots, \lambda_p), \lambda_1 \geq \cdots \geq \lambda_p \geq 0$.

Equivalently, i^{th} principal component is $Z_i = \boldsymbol{e_i'X}$, where $\boldsymbol{e_i}$ is i^{th} column of $\mathbf{E}$, which is the i^{th} normalized eigenvector of $\mathbf{\Sigma}$, also referred to as the principal component loadings for the i^{th} principal component. Here are three important properties of PCA,

$$\mathrm{Var}(Z_i) = \lambda_i$$
$$\mathrm{Cov}(Z_i, Z_j) = 0, \text{for } i \neq j$$
$$\mathrm{Var}(Z_1) \geq \mathrm{Var}(Z_2) \geq \cdots \geq \mathrm{Var}(Z_p) \geq 0$$

Back to our data set, unfortunately, PCA does not work well. The first PC only explains about 40% of total variance (*Figure 6*), and based on the red curve, the first 5 PCs (> 70% of total variance) can be used to reveal the structure of data.

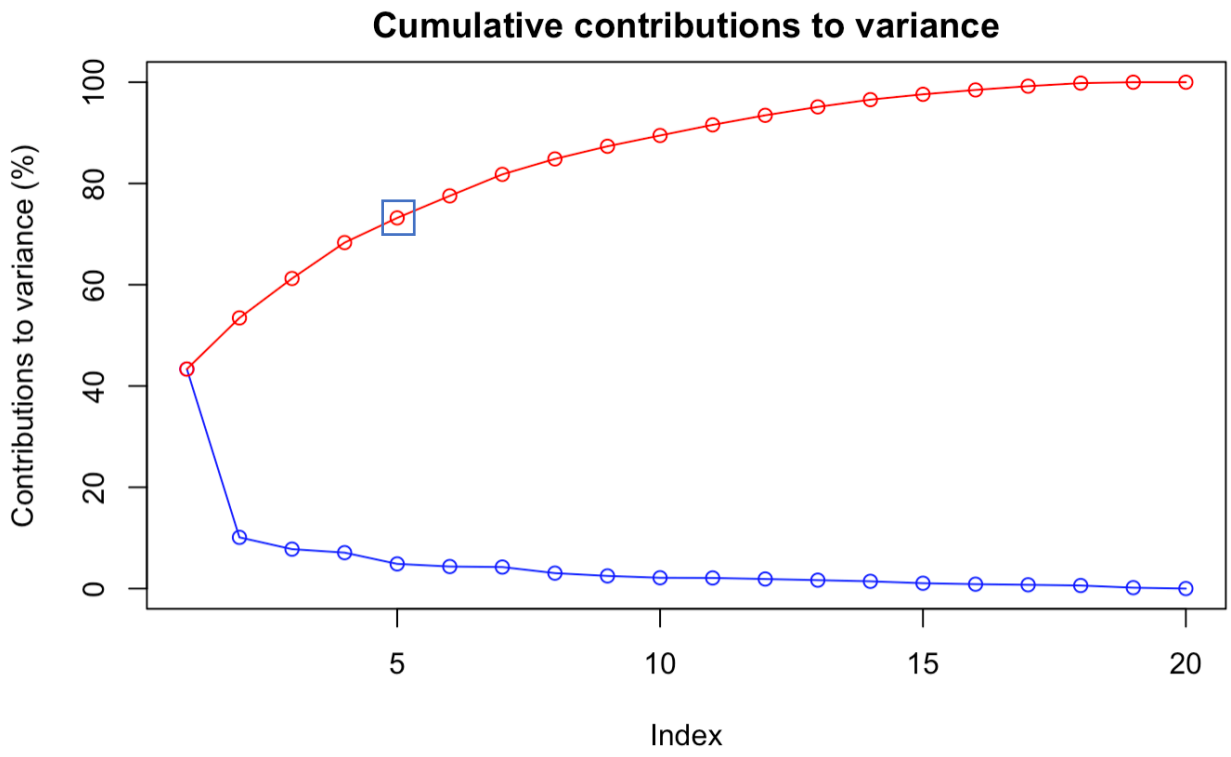

Figure 6: Scree plot (blue line that displays how much simple variation each PC captures from the data) and cumulative contributions to variance plot (red) based on covariance matrix

According to *Figure 7*, ambiguous clusters structure exists, but there is no clear separation of the groups perhaps with the exception of group X. There are some outliers in taxonomic group B, C/M, and f. The first two PCs thus does not seem able to recognize the taxonomic structure of our dataset. Because PCA is a clustering technique without using class labels, CV error cannot be used to assess performance.

Since PCA does not work very well as expected, we will next consider another dimension reduction method, linear discriminant analysis, that uses class labels.

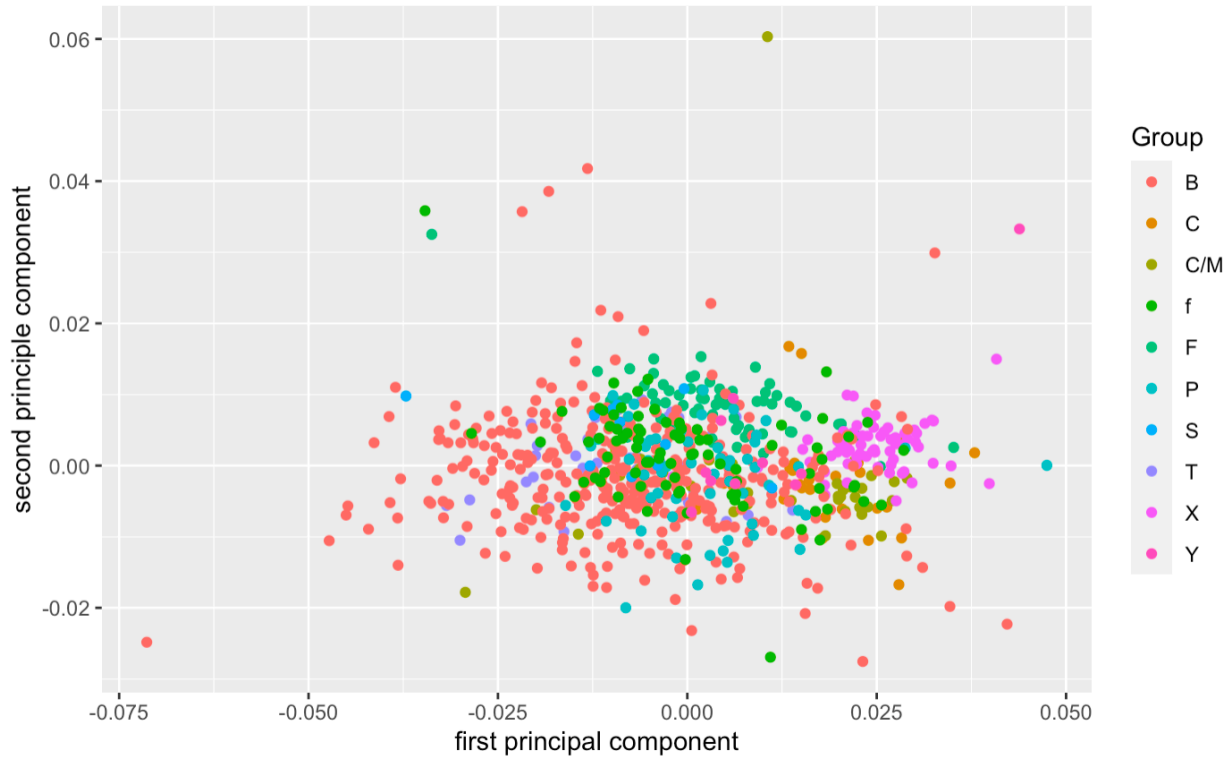

Figure 7: First two principal components based on PCA from covariance matrix

2.3. Linear Discriminant Analysis (LDA)

LDA is another dimension reduction method, similar to PCA, it seeks axes that maximize the separation between multiple classes while maintaining relatively small variation within classes. Suppose we have g classes. Let W denote the variance within classes, and B denote the variance between classes.

$$W = \frac{1}{n-g}\sum(x_{ij} - \overline{x_j})^2,$$

where x_{ij} is i^{th} observation point in j^{th} class, $\overline{x_j}$ is mean of j^{th} class.

$$B = \frac{1}{n-1}\sum(\overline{x_j} - \overline{x})^2,$$

where $\overline{x}$ is overall mean, n is number of observation points. LDA aims to maximizes the ratio B/W.

Compared with *Figure 7* and *Figure 8,* we can say that LDA is superior to PCA in our case, even though it is still hard to see clear cluster structure based on taxonomic grouping. The data points are more spread out. For example, the majority of taxonomic group B data points are in the left half of *Figure 8*, with negative first linear discriminant value. Also, group X data points are more in the bottom right, with positive first linear discriminant and negative second linear discriminant values. Additionally, the CV error of regular 20 predictors is 28.4%, which is lower than tree-based method, but CV error for the model that using GARP and FYMINK rises to 47.6%.

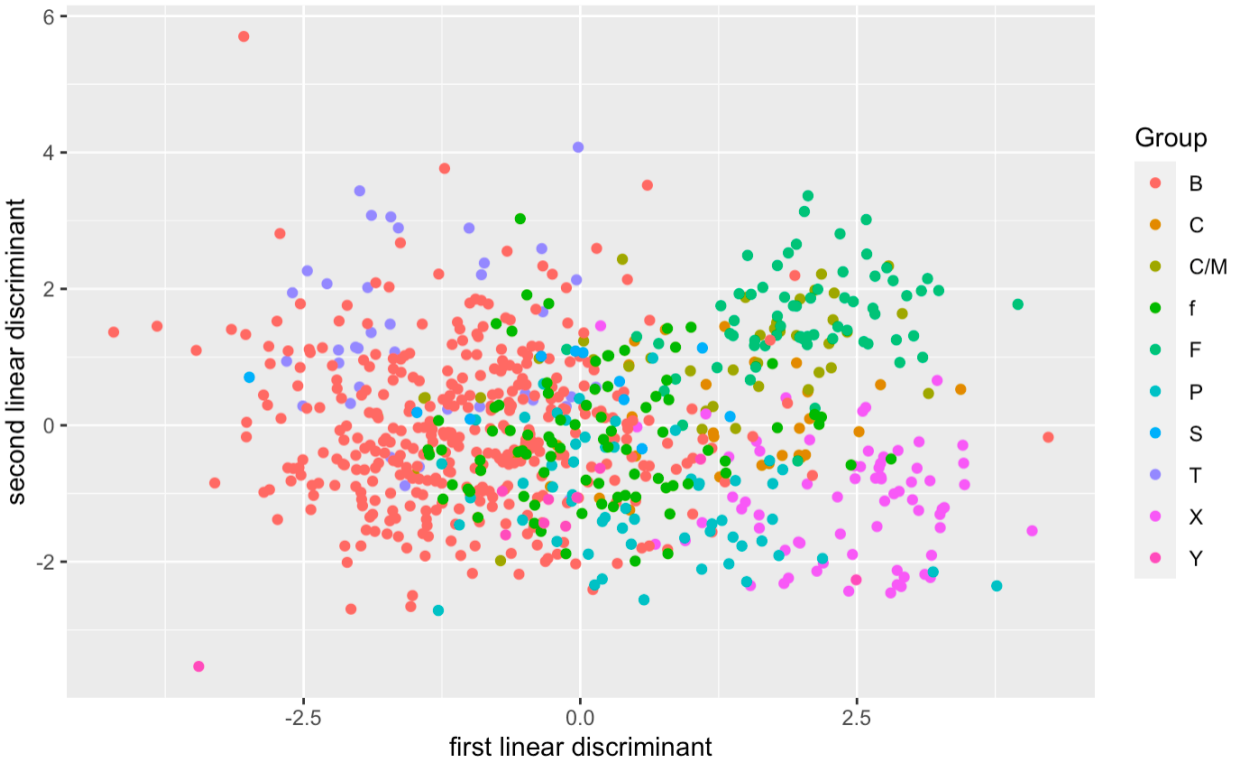

Figure 8: Plot of first two linear discriminants

2.4. Multinomial Logistic Regression

Multinomial analysis is a classification method that generalize logistic regression to multiple classes problem, i.e., we have 10 possible taxonomic groups. Multinomial model can be constructed to predict the probabilities of different possible outcomes (taxonomic groups) by given a set of predictor variables (amino acid species). We can develop multinomial model based on linear regression and logistic regression. In linear regression,

$E(\boldsymbol{Y}|\boldsymbol{X}) = \boldsymbol{\beta}'\boldsymbol{X}$, where $\boldsymbol{\beta}$ is regression coefficients matrix

Suppose we only have two possible discrete outcomes (i.e., Y = 0 or 1), and by definition of logistic regression, the model is

$$log(\frac{Pr(Y=1|X)}{1-Pr(Y=1|X)}) = \boldsymbol{\beta}'\boldsymbol{X}$$

$$\Longrightarrow Pr(Y=1|X) = \frac{e^{\boldsymbol{\beta}'\boldsymbol{X}}}{1+e^{\boldsymbol{\beta}'\boldsymbol{X}}},$$

$$Pr(Y=0|X) = \frac{1}{1+e^{\boldsymbol{\beta}'\boldsymbol{X}}}$$

Since we have 10 possible taxonomic group labels (i.e., $Y = 0, 1, 2, \ldots, 9$), for 10 possible outcomes, we need to extend the logistic regression model. The multinomial model includes logistic regression as a special case and sets

$$Pr(Y=i|X) = \frac{e^{\boldsymbol{\beta}_i'\boldsymbol{X}}}{1+\sum_{j=1}^{9} e^{\boldsymbol{\beta}_j'\boldsymbol{X}}} \ , i = 1,\ldots,9$$

$$Pr(Y=0|X) = \frac{1}{1+\sum_{j=1}^{9} e^{\boldsymbol{\beta}_j'\boldsymbol{X}}}$$

In our dataset, as you can see from *Table 1*, most of observation points are belonging to group B, we used it as reference group $Y = 0$. We used the routine multinom(), part of the R package nnet

(Venables & Ripley, 2002) for fitting. The logarithm of the ratio between probabilities, amino acid coefficient can be simplified to give

$$Log\left(\frac{\Pr(Y=i|X)}{\Pr(Y=0|X)}\right) = Log\left(\frac{e^{\boldsymbol{\beta}_i'\boldsymbol{X}}}{1+\sum_{j=1}^{9} e^{\boldsymbol{\beta}_j'\boldsymbol{X}}} \times \frac{1+\sum_{j=1}^{9} e^{\boldsymbol{\beta}_j'\boldsymbol{X}}}{1}\right)$$

$$= Log\left(e^{\boldsymbol{\beta}_i'\boldsymbol{X}}\right)$$

$$= \boldsymbol{\beta}_i'\boldsymbol{X}$$

Suppose we change from x_j to $x_j + 1$, which means one percent increase in the frequency of the j^{th} amino acid, and holding all other variables fixed. Then the difference of the log probabilities (i.e., coefficient of amino acid *j* in taxonomic group *i*) is

$$\Delta\, Log\left(\frac{\Pr(Y=i|X)}{\Pr(Y=0|X)}\right) = \sum_{k\neq j} \beta_{ik}x_k + \beta_{ij}\cdot\left(x_j+1\right) - \sum_{k\neq j}\beta_{ik}x_k - \beta_{ij}x_j$$

$$= \beta_{ij}$$

Please note that the following table results do not include the reference group B, since the significant amino acids from all other groups are compared with the reference group B. For example, there are 16 significant amino acids in group C, which means that these 16 amino acids are significant to predict whether an amino acid composition vector was from group C or group B.

Furthermore, if we want to test whether an amino acid composition vector was from group *i* or group *k*, the logarithm of the ratio becomes

$$Log\left(\frac{\Pr(Y=i|X)}{\Pr(Y=k|X)}\right) = Log\left(\frac{e^{\boldsymbol{\beta}_i'\boldsymbol{X}}}{1+\sum_{j=1}^{9} e^{\boldsymbol{\beta}_j'\boldsymbol{X}}} \times \frac{1+\sum_{j=1}^{9} e^{\boldsymbol{\beta}_j'\boldsymbol{X}}}{e^{\boldsymbol{\beta}_k'\boldsymbol{X}}}\right)$$

$$= Log\left(e^{(\boldsymbol{\beta}_i-\boldsymbol{\beta}_k)'\boldsymbol{X}}\right)$$

$$= (\boldsymbol{\beta_i} - \boldsymbol{\beta_k})'\boldsymbol{X}$$

So, if $\boldsymbol{\beta_i} = \boldsymbol{\beta_k}$, then none of amino acids are valuable for predicting group *i* or group *k*, but they could be significant for predicting group *i* or group 0.

2.4.1 Regular Predictors

By doing multinomial analysis, we aim to find the significant predictor variables (amino acids), such as how many significant amino acids for each taxonomic group and what are they?

To avoid multicollinearity issues arising from the sum of amino acid frequencies equaling 1, we remove the last amino acid V. In order to check the significance of each amino acid, we construct 95% confidence interval for each amino acid coefficient in each taxonomic group. If the confidence interval of an amino acid contains 0, then this amino acid is not significant for the prediction of certain taxonomic group *i* versus group B. We summarize the results in *Table 2* and *Table 3*. The significant amino acids in each taxonomic group can be seen in *Table 2*.

From *Table 2*, all amino acids are significant in taxonomic group Y, group S also requires all amino acids, except amino acid L for prediction. On the other hand, there are only 10 of amino acids are significant in taxonomic group F. Thus, it is much complex and need more information to predict taxonomic group Y and S versus group B compared with group F versus group B.

More generally, we can conclude the most important amino acid species from *Table 3*, amino acid A and W are significant in all taxonomic groups, also, species D, Q, H, K, P and S are significant in 8 of 9 taxonomic group. So, these amino acids are the important variables for us and necessary factors for prediction. However, amino acid L is significant in only 4 taxonomic group, in general, we can say that it is the least important predictor variable we have.

CV error here is 30.2%, which is better than tree method, but slightly higher than LDA.

Taxonomic Group	Significant amino acids	Number of significant amino acids
C	A, N, D, C, Q, E, G, H, I, K, M, P, S, T, W, Y	16
C/M	A, R, N, D, C, Q, E, H, I, L, M, P, S, W	14
f	A, R, Q, G, H, K, F, P, S, T, W	11
F	A, N, D, C, G, H, K, P, T, W	10
P	A, R, D, Q, E, H, I, L, K, F, P, S, T, W	14
S	A, R, N, D, C, Q, E, G, H, I, K, M, F, P, S, T, W, Y	18
T	A, R, N, D, Q, E, I, L, K, M, F, P, S, T, W, Y	16
X	A, R, D, C, Q, E, H, K, M, F, S, W, Y	13
Y	A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y	19

Table 2: Significant amino acid species in each taxonomic group

Amino acid species	Number of taxonomic groups which amino acid is significant
A	9
R	7
N	6
D	8
C	6
Q	8
E	7
G	5
H	8
I	6
L	4
K	8
M	6
F	6
P	8
S	8
T	7
W	9
Y	5

Table 3: Number of taxonomic groups for which the amino acid was significant

In each grid of *Figure 9*, we can observe the dot plot of distribution of coefficients in different taxonomic groups for a certain amino acid. Please note that we used the actual coefficient value divided by 100 to indicate percentage change, and vertical lines in each grid represent coefficient is 0.

Coefficients in amino acid G and C are more widely distributed. Majority coefficients in amino acid S and L are negative, and the coefficients in amino acid L are more spread out than S. In addition, coefficients in rest of the amino acids do not have much fluctuation, especially A and W, coefficients in these two amino acids are more concentrated on the vertical line 0.

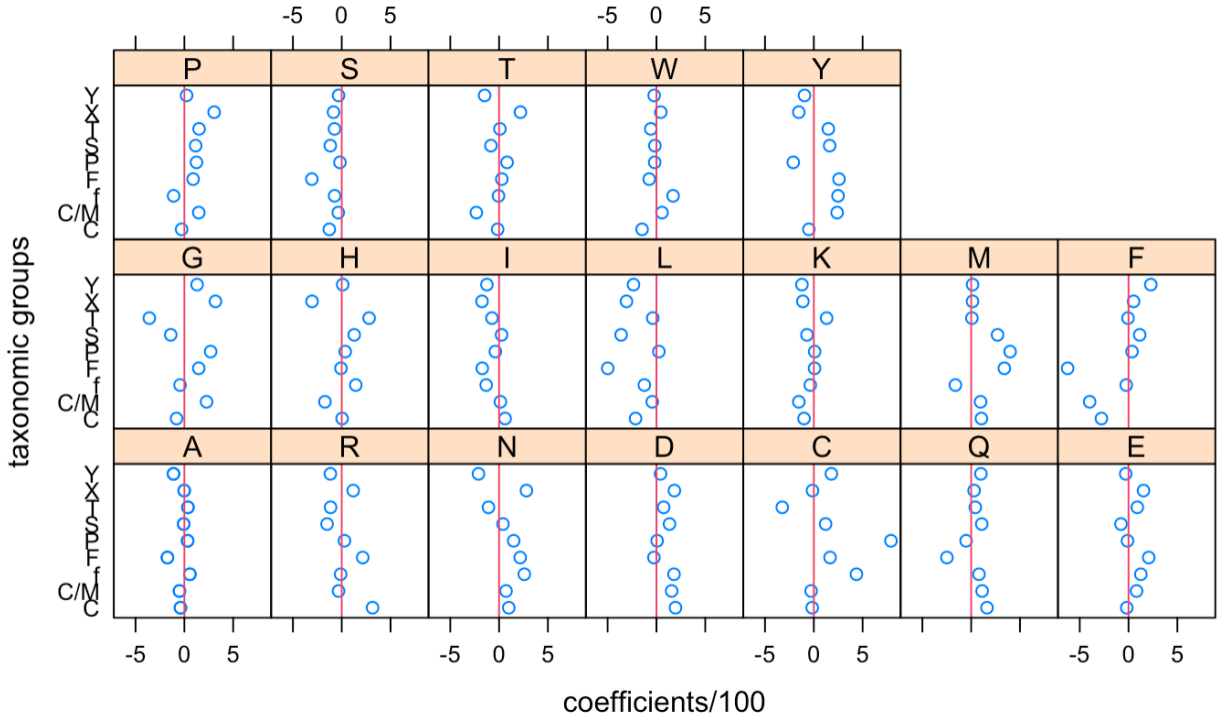

Figure 9: Dot plot of amino acids' coefficients in each taxonomic group

2.4.2 New Predictors: GARP & FYMINK

In this section, we consider multinomial models with predictors GARP and FYMINK. By *Table 4*, the two amino acid combinations are both significant in group f and group T, but insignificant in both group F and group X. This indicates that in taxonomic group f and T, the amino acids that within the two combinations are more important than amino acids that not in the combinations, and vice versa in taxonomic group F and T. In the rest of taxonomic group, there is only one amino acid combination is significant, so, we may more focus on the species that in the significant combination rather than other amino acids in the further studies.

Taxonomic Group	Significant amino acid combinations	Number of significant amino acid combinations
C	FYMINK	1
C/M	FYMINK	1
f	GARP, FYMINK	2
F	None	0
P	FYMINK	1
S	GARP	1
T	GARP, FYMINK	2
X	None	0
Y	GARP	1

Table 4: Significant amino acid combinations in each taxonomic group

Amino acid species	Number of taxonomic groups which amino acid combinations is significant
GARP	4
FYMINK	5

Table 5: Number of taxonomic groups for which the amino acid combination was significant

We cannot really say that which amino acid combination is more important than the other from *Table 5*, but more detailed comparison can be seen by *Figure 10*. The coefficients in GARP are more widely distributed, and only the coefficient in group T is negative. For FYMINK, the

coefficients are more stable. Five of coefficients are very near the vertical line 0. The CV error for the model using GARP and FYMINK is 46.8%.

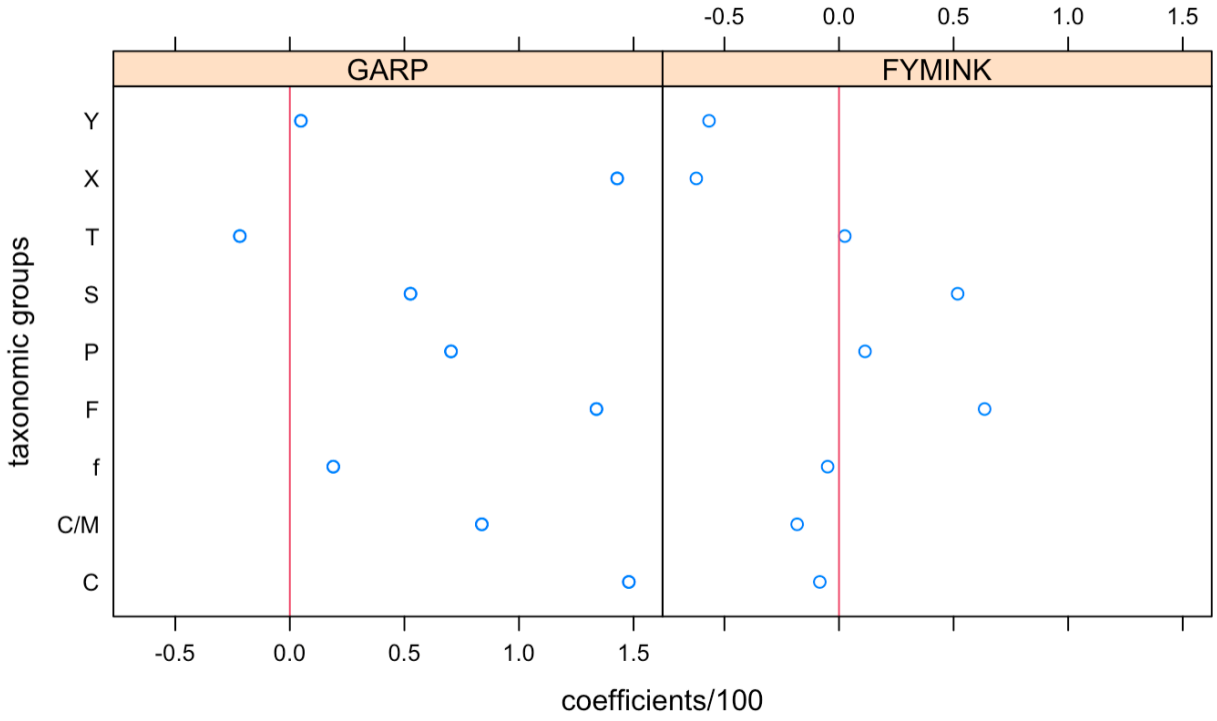

Figure 10: Dot plot of amino acids combinations' coefficients in each taxonomic group

2.4.3 Likelihood Ratio Test

A Likelihood ratio test can be applied to test if using all of the amino acids gives significantly better predictions than using GARP and FYMINK.

Full model:

$$log(\frac{Pr(\mathrm{Y}=i|X)}{1-Pr(\mathrm{Y}=0|X)}) = \beta_{0i} + \beta_{1i}\boldsymbol{X}_1 + \cdots + \beta_{19i}\boldsymbol{X}_{19}\,, i = 1, \cdots, 9$$

Reduced model:

$$log(\frac{Pr(\mathrm{Y}=i|X)}{1-Pr(\mathrm{Y}=0|X)}) = \beta_{0i} + \beta_{1i}\boldsymbol{X}_{GARP} + \beta_{2i}\boldsymbol{X}_{FYMINK}\,, i = 1, \cdots, 9$$

Likelihood ratio test:

H_0: Reduced model is true vs. H_A: Full model is true

The test statistic is

$$2 \times (l_{Full} - l_{Reduced}),$$

where l_{Full} is log-likelihood of full model, $l_{Reduced}$ is log-likelihood of reduced model.

To calculate test statistic, we need to use deviance residual, deviance of full model is $D_{Full} = 2 \times (l_{Saturated} - l_{Full})$, similar as deviance of reduced model.

So,

$$\begin{aligned} D_{Reduced} - D_{Full} &= 2 \times (l_{Saturated} - l_{Reduced}) - \big(2 \times (l_{Saturated} - l_{Full})\big) \\ &= 2 \times (l_{Full} - l_{Reduced}) \\ &= \text{Test statistic} \end{aligned}$$

By summary(), we get deviance of reduced model is 2252.256, and deviance of full model is 1093.266. So, test statistic is 2317.98, as we know, it follows chi-square distribution with degree of freedom $p - q$, p is number of parameters in full model, which is $9 \times 19 = 171$, q is number of parameters in reduced model, $9 \times 2 = 18$. Thus, test statistic follows χ^2_{153}, and the p-value is very near to 0, we can conclude that there is a significant evidence to reject the null hypothesis.

3. Cross-validation Results/Conclusion

The summary CV error results for all of the methods and predictors considered is given in *Table 6*. We compare CV errors among 3 methods with regular predictors or new predictors. As we can see, there are not much difference in the reduced models. CV errors are all approximately equal to 0.5, which are all higher than the CV errors in the full models. So, using GARP and FYMINK as our predictors is not a good choice in our case. On the other hand, CV errors have

some fluctuation in the full models, and we can use the full model and choose LDA and multinomial logistic regression as the main approaches in the future studies to further analyze the data since their CV errors are smaller than tree method.

Regular predictors	**CV Error**
Tree method	38.4%
LDA	28.4%
Multinomial Logistic Regression	30.2%
New predictors: GARP & FYMINK	**CV Error**
Tree method	47.2%
LDA	47.6%
Multinomial Logistic Regression	46.8%

Table 6: Summary of Cross-validation results

In summary, Bacteria candidate phyla is the most common taxonomic group in the study data set. Another two common taxonomic groups are FCB group (f) and Actinobacteria (X), and based on multinomial logistic regression model, 11 and 13 amino acids are significant predictors respectively in each taxonomic group for the predictions that versus the reference taxonomic group Bacteria candidate phyla (B). In addition, taxonomic groups Acidobacteria (Y) and Spirochaetes (S) require more significant amino acids than other taxonomic groups for prediction. Furthermore, amino acid A, D, Q, H, K, P and S are the common significant ones in all taxonomic groups.

References

Amit. (2017, February 14). *Cross validation*. https://rstudio-pubs-static.s3.amazonaws.com/250194_57c5e6eb80c44b35be939cb233293972.html

Breiman, L., Friedman, J. H., Olshen, R. A., & Store, C. J. (1984). *Classification and Regression Trees.* New York: Chapman & Hall / CRC Press. (Formerly Monterey: Wadsworth and Brooks/Cole.). [251, 257]

Hastie, T., Tibshirani, R., & Friedman, J. (2nd ed.). *The Elements of Statistical Learning*. Springer. DOI: 10.1007/b94608

Richard, A. J., & Dean, W. W. (2007). *Applied Multivariate Statistical Analysis. Sixth Edition.* Pearson Prentice Hall.

Stephen M. (2020). *rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'*. R package version 3.0.9. https://CRAN.R-project.org/package=rpart.plot

Terry T., & Beth A. (2019). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15. https://CRAN.R-project.org/package=rpart

Venables, W. N. & Ripley, B. D. (2002). *Modern Applied Statistics with S. Fourth Edition*. Springer, New York. ISBN 0-387-95457-0

Wu, J. H. (2010). *Distance Method Adjustments and a Test for General Heterotachy in Phylogenetic Estimation*. [10, 16]

Appendix

Selected R codes attached here, and the entire codes will be available upon request.

CV error functions in 3 methods

```r
#Tree Method CV error
cv.rpart<-function(data, model=Initial~ .,yname="Initial",K=10, seed=123){
  n<-nrow(data)
  set.seed(seed)
  datay=data[,yname]
  library(MASS)
  f<-ceiling(n/K)
  s<-sample(rep(1:K,f),n)
  CV=NULL
  for(i in 1:K){
    test.index<-seq_len(n)[(s==i)]
    train.index<-seq_len(n)[(s!=i)]
    rpart.fit=rpart(model,data=data[train.index,],cp=0.02,parms = list(split="information"))
    rpart.y<-data[test.index,yname]
    rpart.predy=predict(rpart.fit,data[test.index,],type="class")
    #print(lda.predy)
    error=mean(rpart.y!=rpart.predy)
    CV=c(CV,error)
  }
  list(call=model,K=K,rpart_error_rate=mean(CV),seed=seed)
}
newdata1<-newdata[,-20:-21]
#CV error with regular predictors = 0.3844101 (cp = 0.01)
rpart_cv=cv.rpart(data=newdata1,model=Initial~.,yname="Initial",K=10,seed=123)
rpart_cv$rpart_error_rate
#CV error with new predictors = 0.4719745 (cp = 0.02)
newrpart_cv=cv.rpart(data=sample,model=Group~.,yname="Group",K=10,seed=123)
newrpart_cv$rpart_error_rate
#Note that we use different cp values for different predictors, because it needs to use same cp as previous used,
# i.e., we use cp = 0.01 in regular predictors section, but 0.02 in GARP & FYMINK section.
```

```{r}
#LDA CV error
cv.lda<-function(data, model=Initial~ .,yname="Initial",K=10, seed=123){
  n<-nrow(data)
  set.seed(seed)
  datay=data[,yname]
  library(MASS)
  f<-ceiling(n/K)
  s<-sample(rep(1:K,f),n)
  CV=NULL
  for(i in 1:K){
    test.index<-seq_len(n)[(s==i)]
    train.index<-seq_len(n)[(s!=i)]
    lda.fit=lda(model,data=data[train.index,])
    lda.y<-data[test.index,yname]
    lda.predy=predict(lda.fit,data[test.index,])$class
    error=mean(lda.y!=lda.predy)
    CV=c(CV,error)
  }
  list(call=model,K=K,lda_error_rate=mean(CV),seed=seed)
}
newdata1<-newdata[,-20:-21]
#CV error with regular predictors = 0.283995
er.lda=cv.lda(data=newdata1,model=Initial~.,yname="Initial",K=10,seed=123)
er.lda$lda_error_rate
#CV error with new predictors = 0.4757842
er.lda1=cv.lda(data=sample,model=Group~.,yname="Group",K=10,seed=123)
er.lda1$lda_error_rate
```

```r
#multinomial cv error
cv.multi<-function(data, model=Initial~ .,yname="Initial",K=10, seed=123){
  n<-nrow(data)
  set.seed(seed)
  datay=data[,yname]
  library(MASS)
  f<-ceiling(n/K)
  s<-sample(rep(1:K,f),n)
  CV=NULL
  for(i in 1:K){
    test.index<-seq_len(n)[(s==i)]
    train.index<-seq_len(n)[(s!=i)]
    multi.fit=multinom(model,data=data[train.index,])
    multi.y<-data[test.index,yname]
    multi.predy=predict(multi.fit,data[test.index,])
    error=mean(multi.y!=multi.predy)
    CV=c(CV,error)
  }
  list(call=model,K=K,multi_error_rate=mean(CV),seed=seed)
}
#cv error with regular predictors = 0.3021075
newdata1<-newdata[,-20:-21]
multi_cv=cv.multi(data=newdata1,model=Initial~.,yname="Initial",K=10,seed=123)
multi_cv$multi_error_rate
#cv error with new predictors = 0.4679749
multi_cv1=cv.multi(data=sample,model=Group~.,yname="Group",K=10,seed=123)
multi_cv1$multi_error_rate
```