# Utilizing Relative Codon Frequencies for Taxonomic Classification

Lilith Kotler

December 2025

**Abstract**

Here, we present three models trained to classify species from all domains of life, and beyond, based on the type of organism or biological entity they are. Among the three methods, naive Bayes classification, random forests, and neural networks, we found random forests to be the ideal balance between performance and interpretability. The ability to identify the most predictive features was crucial to contextualize the complex dynamics among the 64 codons and the many reasons why their frequencies can differ. The insights gained from this study contribute to a deeper understanding of the differential use of codons among groups of organisms.

## 1 Introduction & Motivation

The genetic code determines how genetic material is translated into proteins, and is nearly universal across all organisms. It uses 64 units, called codons, that are three nucleotide bases long to encode for 20 standard amino acids. This code is redundant, as most amino acids are encoded by more than one codon, but not ambiguous, as a given codon will always encode for the same amino acid, with few exceptions. Codons consist of four different nucleotides: adenine (A), guanine (G), cytosine (C), and uracil (U) in RNA and thymine (T) in DNA. Because DNA must first be transcribed into RNA prior to protein translation, codons are usually discussed in terms of A, G, C, and U.

Taxonomy, a science which seeks to classify all organisms in a specific and hierarchical manner, was first defined in 1758, much before the discovery of DNA (Padial and Miralles, 2010). But, as DNA sequencing has become more accurate, quicker, and cheaper, molecular data has become an increasingly important tool in evolutionary biology. This project seeks to use relative codon frequencies of over 13,000 organisms in order to predict the taxonomic identity of the samples (L. Hallee, 2020).

## 2 Data Exploration & Preprocessing

The data originally had 13,028 rows and 69 columns, of which two rows and four columns were removed, for 13,026 rows and 65 columns after cleaning. The data cleaning was using `dplyr` from the `tidyverse` R library (Wickham et al., 2019), and all coding done for this project used R version 4.5.2 (R Core Team, 2025). The two rows were removed due to parsing errors when initially loading the CSV. The names of these two species included commas, which caused missing values in some of their corresponding data columns. The

four columns removed contained the species' IDs and names, DNA type of the sample, and the number of codons used to calculate the frequencies.

The names and IDs of the species are unnecessary for predicting taxa and have the potential to bias the results because many species names, especially those of viruses, contain the name of their taxonomic group. The number of codons was removed because the relative codon frequencies were already calculated, and the absolute codon frequencies would no longer be comparable between organisms. Finally, the DNA type column was removed because this could strongly bias the results. The goal is to use only the codon frequencies for taxonomic predictions, but some DNA types correspond to a single taxon or even a single species. For example, there are many samples of chloroplast DNA, which are photosynthetic organelles inside plant cells, so all DNA from chloroplasts corresponds to the plant kingdom.

In order to visualize the overall distribution of codons across all samples, the mean frequency for each codon was calculated and then plotted in a bar chart (Fig. 1). It is apparent in Fig. 1 that there are large discrepancies in codon usage frequencies, straying greatly from a uniform distribution that may be naively assumed. Some of these discrepancies are to be expected, however, such as with UAA and UAG, which are stop codons in most organisms and therefore are typically present only once in an entire gene.

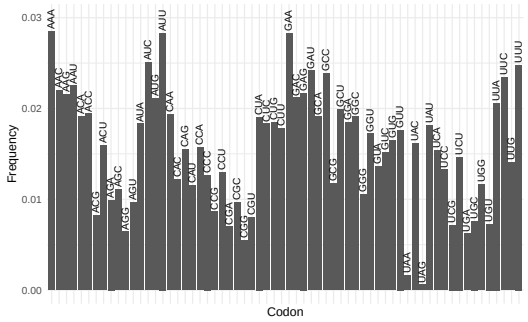


**Figure 1: Mean Relative Codon Frequencies.** Plot created with `ggplot2` from `tidyverse` R library (Wickham et al., 2019).

The dataset contains 11 classes of organisms: bacteria, viruses, plants, vertebrates, invertebrates, mammals, bacteriophages, rodents, primates, archaea, and plasmids. To determine whether or not the data is imbalanced, we made a bar chart to show the number of observations in each class (Fig. 2). Fig. 2 makes it clear that there are very few plasmid observations, so we decided to remove them from the data. With only 18 instances, plasmids made up less than 0.14% of all observations and could not be oversampled or bootstrapped without excessively amplifying noise.

Of the remaining 10 classes, not all taxa were mutually exclusive. Despite this, the authors of this data posit that they were able to develop models that could accurately classify samples into these groups (Hallee and Khomtchouk, 2023). To determine how well the data's variance can separate each class, we conducted a principal components analysis (PCA). We found that there are two clusters of observations, but also that the clusters were not separating taxa (Fig. 3).

We did a closer analysis of the two groups of non-mutually exclusive classes, beginning with rodents, primates, mammals, and vertebrates. Similarly to the analysis with

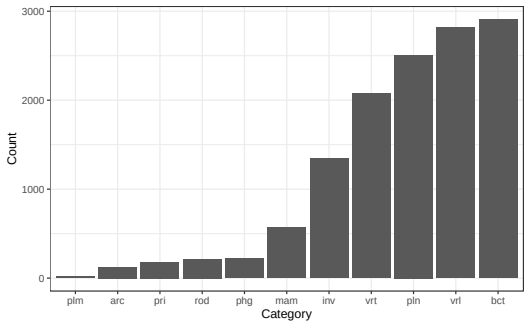


**Figure 2: Number of Observations in Each Class.** Plot created with `ggplot2` from `tidyverse` R library (Wickham et al., 2019).

all 10 classes, we found two distinct clusters, but each cluster contained observations from all four classes (Fig. S1). For this reason, we combined rodents, primates, mammals, and vertebrates into a single vertebrate class. We also did a closer analysis of bacteria, bacteriophages, and viruses and found that bacteriophages, which are a type of virus, clustered with bacteria rather than other viruses (Fig. 4). This is to be somewhat expected because bacteriophages infect bacteria and inject their genetic material into the DNA of the host bacterium, leading to a significant amount of DNA being shared. Because of this overlap, instead of combining bacteriophages with other viruses, we elected to remove them from our analysis entirely, especially because they are one of the minority classes.

At the end of our data processing, we were left with six classes in the response variable, 64 continuous predictors, and 12,698 observations.

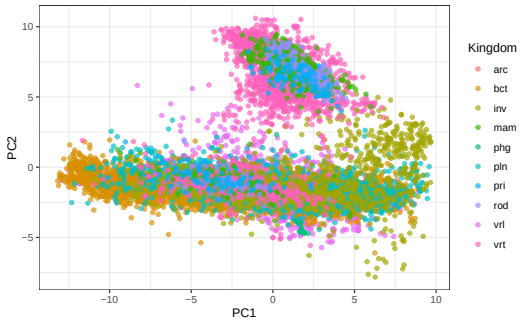


**Figure 3: PCA with All 10 Classes.** Plot created with `ggplot2` from `tidyverse` R library (Wickham et al., 2019).

# 3 Modeling Approaches

We used three machine learning approaches: naive Bayes classifiers, random forests, and neural networks. These methods were selected primarily for their ability to perform multiclass classification and, secondarily, for their ability to handle imbalanced data.

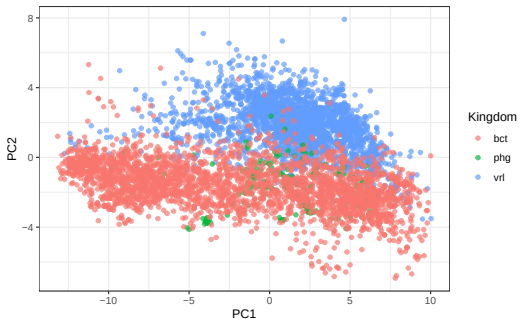


**Figure 4: PCA with Bacteria, Bacteriophages, and Viruses.** Plot created with `ggplot2` from `tidyverse` R library (Wickham et al., 2019).

Among the simple generative classification models, linear discriminant analysis (LDA), naive Bayes, and quadratic discriminant analysis (QDA), we determined that naive Bayes would be most appropriate. Due to the calculation of covariance matrices, LDA and especially QDA are much more computationally expensive than naive Bayes. With 64 features and a six-class response, LDA and QDA would have taken a prohibitively long time on a local machine. Additionally, the simplicity of naive Bayes means that it requires less data, making it better suited for the extreme imbalance in our data set.

Tree-based methods are inherently well-suited for multiclass classification, but single trees are prone to poor performance, especially on imbalanced data. However, as an ensemble method, random forests are a strong alternative. The use of a different set of predictors at each split decorrelates the trees, greatly decreasing the chances of converging to a local rather than global minimum. Although boosting, another ensemble method, can also provide strong results by iteratively improving the signal from previous trees, having six classes made it too computationally expensive for the scope of this project.

Neural networks can easily be applied to multinomial classification problems with the inclusion of a softmax output layer, making them an attractive choice for this data set. Additionally, they are good at handling data sets with large amounts of observations and predictors. Because our data does not involve sequences or images, we determined that feed-forward neural networks would be best suited for this application.

# 4 Model Evaluation

We used the same 60/20/20 training/validation/testing splits for all three models and ensured that the minority class, archaea, was present in each split. The training set has 7,782 observations and the validation and testing sets each have 2,593 observations. After using the validation set to tune hyperparameters or select predictors, we fit the final models with both the training and validation set, then evaluated the models' performances on the unseen testing set. We used the validation set method instead of cross-validation (CV) because CV was a computationally intractable method to train the random forests and neural networks due to the multiclass response variable. Although the naive Bayes model was initially trained using 10-fold CV, we elected to instead use the validation set method to maintain consistency with the other two models and ensure that their performances could be directly compared.

Utilizing the `e1071` R library, we fit the naive Bayes model using forward stepwise selection (Meyer et al., 2024). To provide a more robust analysis of the imbalanced data, we used macro F1 score as the evaluation metric instead of overall accuracy or error. We continued iteratively adding predictors until the macro F1 score as evaluated on the validation set stopped increasing, resulting in a final model with 13 out of the 64 predictors: AGG, CUA, UUU, UGU, AUA, ACA, UGA, UGC, UUC, AAG, CUU, CCC, and UAA.

For the random forest, we used the `randomForest` R library, which has two primary hyperparameters: `mtry`, which is the number of predictors used at each split, and `ntree`, which is the total number of trees used (Liaw and Wiener, 2002). We used a tuning grid with 28 entries, with `mtry` ranging from 6 to 12 and `ntree` values of 250, 500, 750, and 1000. The forest with 12 variables at each split and 250 trees had the highest macro F1 score on the validation set, so these values were used to fit the final random forest model.

We fit well over a dozen different feed-forward neural networks using the `keras` and `tensorflow` R libraries (Allaire and Chollet, 2025, Allaire and Tang, 2025). There is an inherent difficulty with training neural networks because there are infinitely many ways to create them without a clear way to identify which is the best. So, we started by fitting single-layer networks starting at 10 neurons in the hidden layer, increasing by 10 units at a time until reaching 50 units. We then tried adding dropout layers and additional hidden layers, but found that they worsened the model's performance. The single-layer network with 50 nodes had the lowest validation loss and highest validation accuracy, so we used that same structure to fit our final neural network.

| Model | Accuracy | Precision | Recall | Macro F1 Score | Macro AUC |
|---|---|---|---|---|---|
| Naive Bayes | 0.688 | 0.667 | 0.642 | 0.642 | 0.884 |
| Random Forest | 0.899 | 0.885 | 0.837 | 0.856 | 0.985 |
| Neural Network | 0.939 | 0.915 | 0.919 | 0.917 | 0.995 |

**Table 1: Evaluation Metrics for Overall Performance.** Accuracy, precision, recall, and macro F1 score calculated using `caret` (Kuhn and Max, 2008). Macro area under the curve (AUC) calculated by `multiROC` (Wei and Wang, 2018). LaTeX table created with assistance of `xtable` (Dahl et al., 2019).

Because the response has several imbalanced classes, we had to use various methods to evaluate the performance of the models. Table 1 has evaluation metrics for each model's performance across all classes, and tables S1, S2, and S3 have per-class metrics for the naive Bayes classifier, random forest, and neural network, respectively. We also created two types of visualizations: confusion matrices and receiver operating characteristic (ROC) curves, both using `ggplot2` from the `tidyverse` library (Wickham et al., 2019). The confusion matrices (Fig. 5, 6, 7) were made with code adapted from davedgd (2019) and the `caret` library and the one-versus-all (OVA) ROC curves (Fig. S2, S3, S4) used the `multiROC` library (Kuhn and Max, 2008, Wei and Wang, 2018).

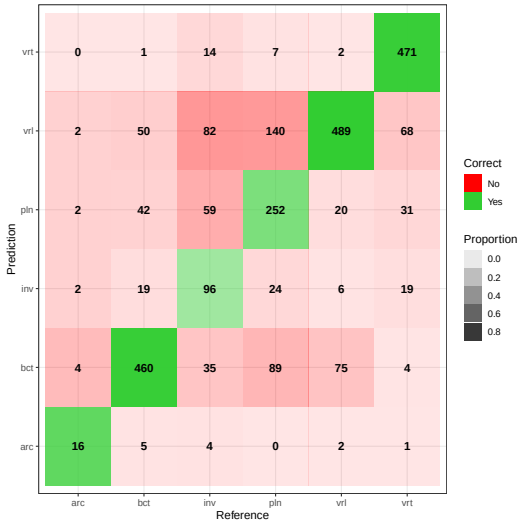


**Figure 5: Naive Bayes Confusion Matrix.** Confusion matrix created with `caret` and plotted using `ggplot2` (Kuhn and Max, 2008, Wickham et al., 2019). Code used for plotting adapted from davedgd, 2019.

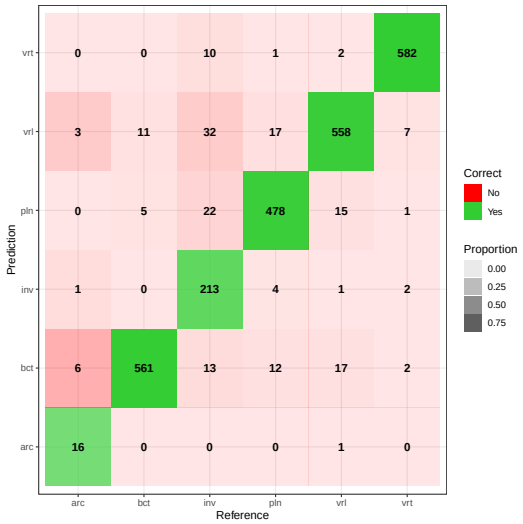


**Figure 6: Random Forest Confusion Matrix.** Confusion matrix created with `caret` and plotted using `ggplot2` (Kuhn and Max, 2008, Wickham et al., 2019). Code used for plotting adapted from davedgd, 2019.

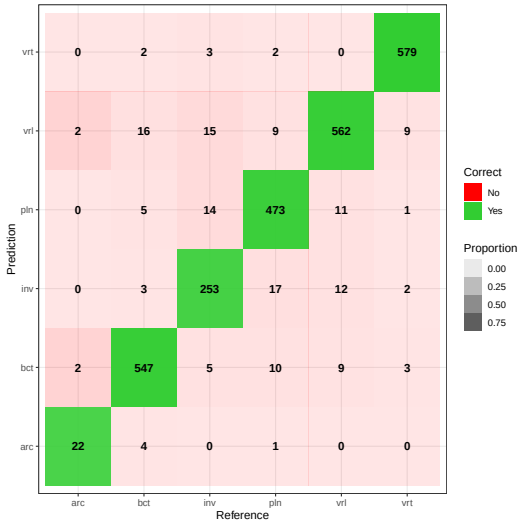


**Figure 7: Neural Network Confusion Matrix.** Confusion matrix created with `caret` and plotted using `ggplot2` (Kuhn and Max, 2008, Wickham et al., 2019). Code used for plotting adapted from davedgd, 2019.

# 5 Interpretation & Insights

From figures 5, 6, and 7, it is clear that the naive Bayes model performed significantly worse than the other two models and that the neural network performed the best overall. While it has the best predictive performance, it is also the least interpretable, with no ability to assess variable importance.

With the random forest, however, the importance of variables can easily be quantified and visualized. In analyzing important predictors for the random forest, we chose to consider only the mean decrease in Gini (MDG) and not the mean decrease in accuracy because MDG is more robust to imbalanced data. After the first 10 predictors, there is a drop-off in importance (Fig. S5), but among these there are six predictors that were also selected by the naive Bayes model: CUA, AGG, UGA, UGC, UGU, and AAG.

Notably, CUA and AGG encode for leucine and arginine, respectively, which are two of the three amino acids with six degenerate codons, with the third being serine. However, not a single serine codon was present among either set of significant predictors. A likely cause for this is the importance of GC bias in synonymous codon selection (Palidwor et al., 2010). GC bias is an enrichment in guanine (G) and cytosine (C) DNA bases, and as GC bias increases, we generally observe a higher amount of synonymous codons that end in G or C. But, leucine and arginine are the only amino acids that have codons where GC-changing point mutations are possible in both the first and third positions, making the dynamics of their codon frequencies much more complex (Palidwor et al., 2010). As a result, the reasons for differential leucine and arginine codon frequencies can not only vary widely between species among a single taxon, but also between regions within a single organism.

UGA is one of the reasons that the genetic code is nearly universal, but not universal. Codons can encode for atypical amino acids due to the evolution of novel tRNAs, which are special types of nucleic acids that transport amino acids to the correct codons during protein

translation. tRNAs contain an anticodon, which is what binds to the codon as the protein synthesis machinery adds the new amino acid to the growing peptide. Most organisms, including humans, do not have a tRNA corresponding to UGA, which is why it is typically a stop codon, but some organisms, particularly certain species of bacteria, have tRNAs that bring selenocysteine to UGA codons (Vargas-Rodriguez et al., 2018). In organisms such as these, UGA will inherently be used much more frequently because it is no longer constricted to being used once per gene. Because the presence of tRNAs with UGA's anticodon is most common among bacteria, this may influence the importance of UGA as a predictor.

It has even been found that some bacteria have evolved to use a cysteine-carrying tRNA for UGA instead of the more typical selenocysteine-carrying tRNA (Vargas-Rodriguez et al., 2018), adding a third cysteine-encoding codon. Interestingly, the two canonical cysteine codons, UGC and UGU, were both identified as significant predictors by the naive Bayes and random forest models. One study found that the bacterium *E. coli* differentially uses the two synonymous cysteine codons depending on whether or not a particular cysteine is supposed to form disulfide bridges, which are covalent bonds between two cysteines in a protein (Song et al., 2006). This has many implications for codon usage frequencies not only for bacteria but also other domains of life if they are found to exhibit the same codon usage bias. If this is a common trait, variations between taxa could also be caused by differential amounts of proteins containing disulfide bridges.

There are many potential causes for codon usage bias beyond what we have discussed here and the specifics can vary from species to species, which makes it incredibly difficult to identify exactly why some groups of organisms use one codon more than another, from both a machine learning and biological perspective. However, as the literature continues to expand, we hope to be able to better analyze our results through the context of molecular mechanisms.

# 6 Conclusion & Reflection

Overall, we determined that the random forest was the best model, despite having slightly worse performance than the neural network, because its high degree of interpretability makes it much more useful.

We found that each model erroneously classified observations from each non-viral class as viruses, but that the precision for the virus class was fairly high for the random forest and neural network. This is most likely due to the mechanism of viruses, which are not actually classified as living organisms because they cannot reproduce on their own. Instead, they infect cells and inject their genetic material into that of the host's, hijacking their cellular machinery to replicate. This inherently results in shared portions of the genomes of viruses and the type of organisms that they infect, which could be contributing to the misclassifications. Additionally, although we removed bacteriophages, which exclusively infect bacteria, early in our analysis, it is likely that there are several bacteriophages within the general virus class because the vast majority of all viruses are bacteriophages.

We also found fairly frequent misclassifications between archaea and bacteria, although this is to be somewhat expected. There are three domains of life, with the first two to evolve being Archaea and Bacteria, both of which are prokaryotic, meaning that they do not have membrane-bound organelles. Although the two have many differences, their similarities make them fundamentally different from the third domain, Eukarya, which consists of all fungi, animals, plants, and many other microorganisms. This, in addition to the small

archaea sample size, are likely significant contributors to the misclassifications.

One finding that was not expected and cannot be currently explained is the misclassifications of plants as invertebrates and vice versa. Invertebrates and plants are much too distantly diverged for there to be an ancestral explanation for similar codon frequencies, but we were unable to find any scientific literature exploring convergent evolution that could potentially explain this phenomenon.

There were several limitations in this project, with the most notable from a machine learning perspective being the lack of cross-validation. Due to the scope of our analysis, we were unable to use high-performance computing (HPC) resources, greatly hindering the robustness of our pipeline. From a biological point of view, this data set does not seem to include fungi, a unique and incredibly biodiverse group of organisms, and the taxonomic classes are not all on the same level, with some being domains, others kingdoms, and some even more specific, such as rodents and primates. In the future, we would like to do a more intricate and stratified analysis with significantly more data. This may include steps such as multiple hypothesis testing to determine, for each codon, which taxa have statistically significantly different mean frequencies, implementing backward stepwise naive Bayes variable selection, or conducting several sets of analysis based on the strict taxonomic classifications of organisms.

However, despite these pitfalls, in this paper we presented two models that, despite the significant variation in codon frequencies within taxa, species, and even individuals, achieved high mean classification accuracy with 10% error or less. This suggests that among the many layers of complexity lie signals predictive of taxonomy that are significant enough to be identified by even comparatively simple models.

# References


Allaire, J., & Chollet, F. (2025). *Keras: R interface to 'keras'* [R package version 2.16.0]. https://doi.org/10.32614/CRAN.package.keras

Allaire, J., & Tang, Y. (2025). *Tensorflow: R interface to 'tensorflow'* [R package version 2.20.0]. https://doi.org/10.32614/CRAN.package.tensorflow

Dahl, D. B., Scott, D., Roosen, C., Magnusson, A., & Swinton, J. (2019). *Xtable: Export tables to latex or html* [R package version 1.8-4]. https://doi.org/10.32614/CRAN.package.xtable

davedgd. (2019, November 30). *Re: Plot confusion matrix in r using ggplot* [Stack overflow]. https://stackoverflow.com/questions/37897252/plot-confusion-matrix-in-r-using-ggplot/59119854#59119854

Hallee, L., & Khomtchouk, B. B. (2023). Machine learning classifiers predict key genomic and evolutionary traits across the kingdoms of life. *Scientific Reports*, *13*(1), 2088. https://doi.org/10.1038/s41598-023-28965-7

Kuhn & Max. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software*, *28*(5), 1–26. https://doi.org/10.18637/jss.v028.i05

L. Hallee, B. K. (2020). Codon usage. https://doi.org/10.24432/C5KP6B

Liaw, A., & Wiener, M. (2002). Classification and regression by randomforest. *R News*, *2*(3), 18–22. https://CRAN.R-project.org/doc/Rnews/

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2024). *E1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), tu wien* [R package version 1.7-16]. https://doi.org/10.32614/CRAN.package.e1071

Padial, J. M., & Miralles, A. (2010). Review the integrative future of taxonomy. *Frontiers in Zoology*.

Palidwor, G. A., Perkins, T. J., & Xia, X. (2010). A general model of codon bias due to GC mutational bias. *PLoS ONE*, *5*(10), e13431. https://doi.org/10.1371/journal.pone.0013431

R Core Team. (2025). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Song, J., Wang, M., & Burrage, K. (2006). Exploring synonymous codon usage preferences of disulfide-bonded and non-disulfide bonded cysteines in the e. coli genome. *Journal of Theoretical Biology*, *241*(2), 390–401. https://doi.org/10.1016/j.jtbi.2005.12.004

Vargas-Rodriguez, O., Englert, M., Merkuryev, A., Mukai, T., & Söll, D. (2018). Recoding of the selenocysteine UGA codon by cysteine in the presence of a non-canonical tRNACys and elongation factor SelB. *RNA Biology*, *15*(4), 471–479. https://doi.org/10.1080/15476286.2018.1474074

Wei, R., & Wang, J. (2018). *Multiroc: Calculating and visualizing roc and pr curves across multi-class classifications* [R package version 1.1.1]. https://doi.org/10.32614/CRAN.package.multiROC

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

# Appendix

## Supplementary Figures

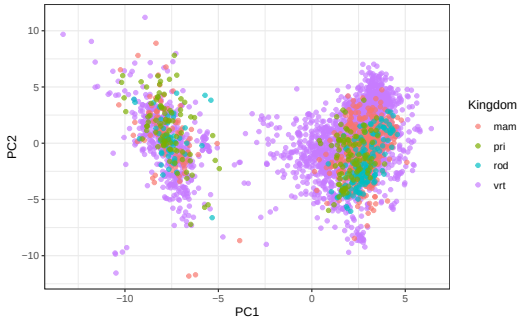


**Figure S1:** PCA with Vertebrates, Mammals, Rodents, and Primates

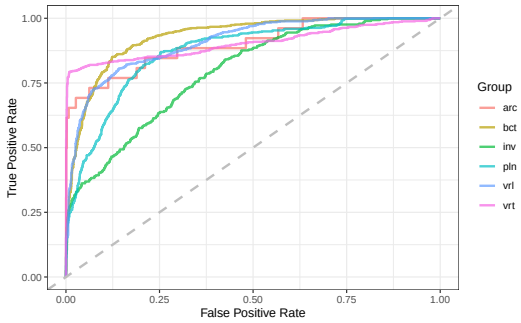


**Figure S2: Per-Class One-Versus-All ROC Curves for Naive Bayes Model.** ROC curves calculated with `multiROC` and plotted with `ggplot2`.

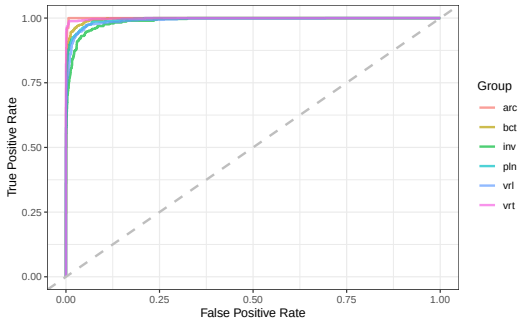


**Figure S3: Per-Class One-Versus-All ROC Curves for Random Forest Model.** ROC curves calculated with `multiROC` and plotted with `ggplot2`.

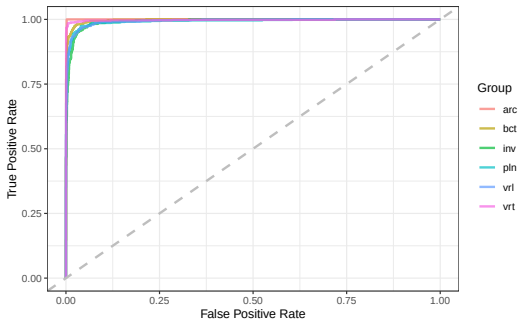


**Figure S4: Per-Class One-Versus-All ROC Curves for Neural Network Model.** ROC curves calculated with `multiROC` and plotted with `ggplot2`.

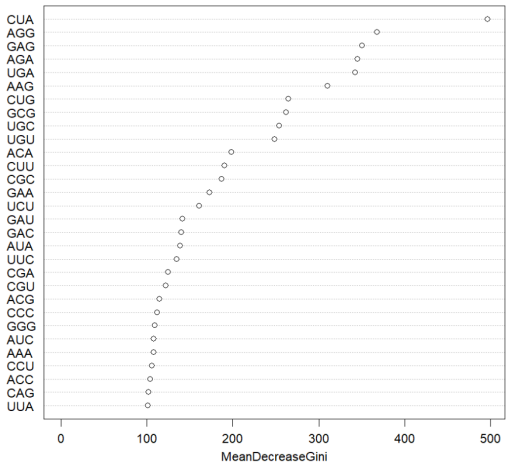


**Figure S5: Variance Importance Plot for Random Forest Model.**

## Supplementary Tables

| Class | Balanced Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| arc | 0.805 | 0.571 | 0.615 | 0.593 | 0.900 |
| bct | 0.847 | 0.690 | 0.797 | 0.740 | 0.929 |
| inv | 0.650 | 0.578 | 0.331 | 0.421 | 0.792 |
| pln | 0.709 | 0.621 | 0.492 | 0.549 | 0.872 |
| vrl | 0.826 | 0.588 | 0.823 | 0.686 | 0.906 |
| vrt | 0.890 | 0.952 | 0.793 | 0.865 | 0.904 |

**Table S1: Per-Class Evaluation Metrics for Naive Bayes Model.** Balanced accuracy, per-class precision and recall, and F1 scores calculated using `caret`. Per-class OVA AUC calculated with `multiROC`. LaTeX table created with assistance of `xtable`.

| Class | Balanced Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| arc | 0.807 | 0.842 | 0.615 | 0.711 | 0.994 |
| bct | 0.956 | 0.903 | 0.941 | 0.922 | 0.990 |
| inv | 0.839 | 0.852 | 0.693 | 0.764 | 0.961 |
| pln | 0.931 | 0.872 | 0.895 | 0.883 | 0.986 |
| vrl | 0.933 | 0.867 | 0.907 | 0.887 | 0.984 |
| vrt | 0.980 | 0.975 | 0.968 | 0.971 | 0.995 |

**Table S2: Per-Class Evaluation Metrics for Random Forest Model.** Balanced accuracy, per-class precision and recall, and F1 scores calculated using `caret`. Per-class OVA AUC calculated with `multiROC`. LaTeX table created with assistance of `xtable`.

| Class | Balanced Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| arc | 0.922 | 0.815 | 0.846 | 0.830 | 0.999 |
| bct | 0.967 | 0.950 | 0.948 | 0.949 | 0.996 |
| inv | 0.929 | 0.882 | 0.872 | 0.877 | 0.992 |
| pln | 0.954 | 0.938 | 0.924 | 0.931 | 0.993 |
| vrl | 0.960 | 0.917 | 0.946 | 0.931 | 0.993 |
| vrt | 0.986 | 0.988 | 0.975 | 0.981 | 0.998 |

**Table S3: Per-Class Evaluation Metrics for Neural Network Model.** Balanced accuracy, per-class precision and recall, and F1 scores calculated using `caret`. Per-class OVA AUC calculated with `multiROC`. LaTeX table created with assistance of `xtable`.

## Code Availability

Link to annotated code, figures, model fits, and altered data:
https://github.com/lilithbk/codon-freq-for-taxonomy.git