HIV Protease Prediction Project

Collin Leonard

6/21/2020

Overview

The HIV-1 protease cleavage data set, by Thorsteinn Rögnvaldsson, can be found on the *UCI Machine Learning Repository* website. The full data set folder can be downloaded here, and a more detailed description of the set and past publications can be found here.

According to this study, HIV-1 protease is an enzyme that plays an essential part in the development of the HIV virus. Cleavage sites are places on the virus where the enzyme suceeds in furthering the progress of the virus toward replication, and is the principle agent in the immune deficiency AIDS. Thus, one method in the treatment of HIV/AIDS is the understanding of HIV-1 protease, and its inhibitors. Each octamer in the data used is a possible cleavage site for the enzyme, and a successful prediciton of the cleavage site allows for progess in the development of cleavage site specific HIV-1 protease inhibitors.

The project data has only two attributes:

> An 8 letter string (octamer) and a label that tells whether this string represents a site in a peptide (or protein) where the HIV-1 protease cleaves it (+1 if yes, -1 if no). The 8 letter string can also be viewed as 8 independent attributes.

For reference, each attribute is a letter denoting an amino acid. G (Glycine, Gly); P (Proline, Pro); A (Alanine, Ala); V (Valine, Val); L (Leucine, Leu); I (Isoleucine, Ile); M (Methionine, Met); C (Cysteine, Cys); F (Phenylalanine, Phe); Y (Tyrosine, Tyr); W (Tryptophan, Trp); H (Histidine, His); K (Lysine, Lys); R (Arginine, Arg); Q (Glutamine, Gln); N (Asparagine, Asn); E (Glutamic Acid, Glu); D (Aspartic Acid, Asp); S (Serine, Ser); T (Threonine, Thr).

The goal of this project is to perform machine learning technique to predict the binary result of the cleavage site, with a goal of maximizing accuracy of prediction (while accounting for the specificity and sensitivity). The F_2 metric will be used to quanitify this goal.

Methods & Analysis

Loading and manipulating the data

The first step in the project is gathering and combining the data into a sinlge manageable data frame. The data when downloaded from the *UCI Machine Learning Repository* is contained in *.txt* files from four different sources. The breakdown of the files is as follows:

Data sets	Observations	Cleavage Sites	Non-cleavage Sites
746	746	401	345
1625	1625	374	1251
Schilling	3272	434	2838
Impens	947	149	798

The decision was made to combine all of these sets into one large data set. This sacrifices some of the individual predictive power of the unique experiments, but allows for a more robust result using the algorithms. To combine these sets, a loop reads each individual file from the data folder in the HIV folder. It reads the *.txt* format, separates the variables by comma, and binds each to a larger matrix. The function `load_dat` performs all the actions for the user automatically.

```
load_dat <- function(name_vector){
  # Initialize empty data frame
  dat <- data.frame(octamer = character(), flag = factor())

  # For loop takes names of each file, extracts the data, and fills dat with it
  for (name in name_vector) {
    dat_name <- paste("./data/",name, sep = "")
    temp_dat <- read.delim2(dat_name, header = FALSE,sep = ",", col.names = c("octamer","flag"))
    dat <- rbind(dat, temp_dat)
  }

 # Returns only the unique values of dat, (leaves behind octamers that produce conflicting results in d
 return(unique(dat, by = octamer))
}
```

The resulting data set contains only the unique combinations of octamer and cleavage indication. The next hurdle was manipulating the octamer combinations. The choice was made to expand the octamers into a matrix of letter counts, similar to a genre matrix in a movie recommendation system. The function `expand_octamers` was written to produce a data frame that has the cleavage information, followed by 20 columns of letter data. The function code can be seen below:

```
expand_octamers <- function(data_mat){
  ## Create matrix of individual octamers
  my_frame <- as.data.frame(data_mat$octamer, stringsAsFactors = FALSE)
  split_frame <- as.data.frame(tstrsplit(my_frame[,1], '', type.convert = TRUE), stringsAsfactors = FALS
  name_list<- names(table(split_frame[,1]))

  ## Create blank matrix of octamer letters
  blank_mat <- matrix(0,nrow(data_mat)+1, length(name_list))
  blank_mat[1,] <- name_list
  colnames(blank_mat) <- name_list

  ## Fill in blank matrix with data
  # (INSPIRED BY DATA FLAIR'S GENRE MATRIX FUNCTION
  for (index in 1:nrow(split_frame)) {
    for (col in 1:ncol(split_frame)) {
      temp_col <- which(blank_mat[1,] == split_frame[index,col])
      y <- as.numeric(blank_mat[index+1,temp_col])
      blank_mat[index+1,temp_col] <- y + 1
    }
  }

  ## Add flags
  new_dat <- as.data.frame(blank_mat[-1,], stringsAsFactors=FALSE)
  new_dat <- cbind(data_mat$flag, new_dat)
  row_vec <- as.character(data_mat$octamer)
  rownames(new_dat) <- row_vec
  colnames(new_dat)[1] <- "flag"
```

```
  return(mutate_all(new_dat, function(x) as.numeric(as.character(x))))
}
```

Once those two functions are loaded, it is a simple task to generate data to be used easily.

```
data_names <- c("schillingData.txt", "impensData.txt", "746Data.txt", "1625Data.txt")
dat <- load_dat(data_names)
```

Ten sets of repeating octamers were left after the `load_dat` function, because they have conflicting results for cleavage. In these cases, both occurences were excluded from analysis, though they might be a point of interest in later findings. The data is then binarized and fed into the expansion function:

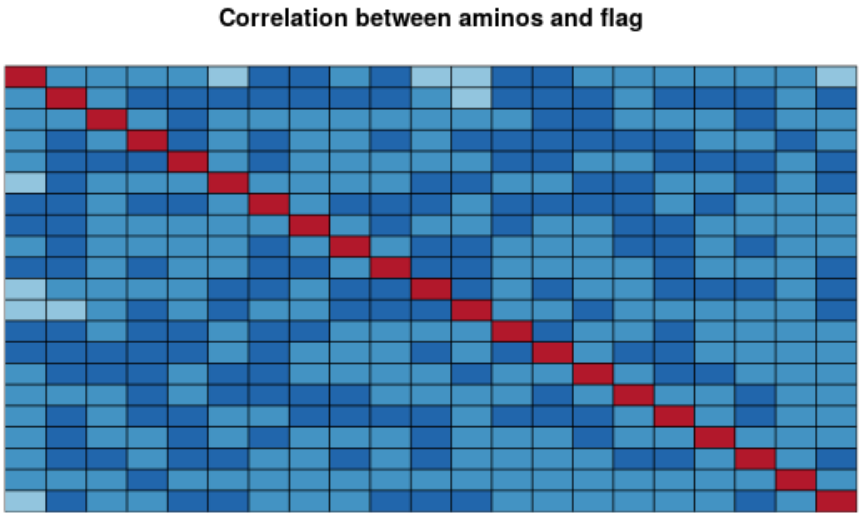
```
# Remove octamers that show up in different data sets, with different flag results
rep_ind <- which(duplicated(dat$octamer) == TRUE)
dat <- dat[-rep_ind,]

# Binarize flagging data (1 = cleavage, 0 = no cleavage)
dat$flag <- (dat$flag+1)/2

# Expand data to create octamer matrix
exp_dat <- expand_octamers(dat)
```

Exploring the data

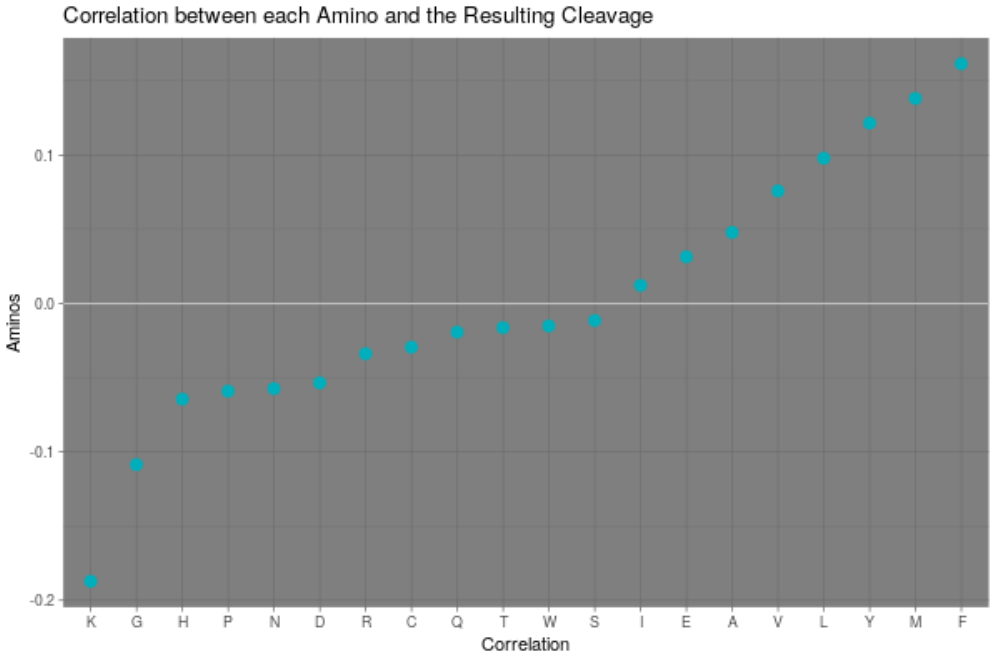
The first calculation performed was finding the ratio of cleavage sites to non-cleavage sites.

$$\mu = 0.171$$

With this number, an initial prediction vector made of only *0's* would have an accuracy of $1 - 0.171 = 0.829$. This can act as a baseline, but obviously the sensitivity to cleavage sites is 0%, which makes the prediction useless. To gain an understanding of the interactions between the matrix of letters with the 'flags' (cleavage sites), a visualization of the correlations can be very helpful.



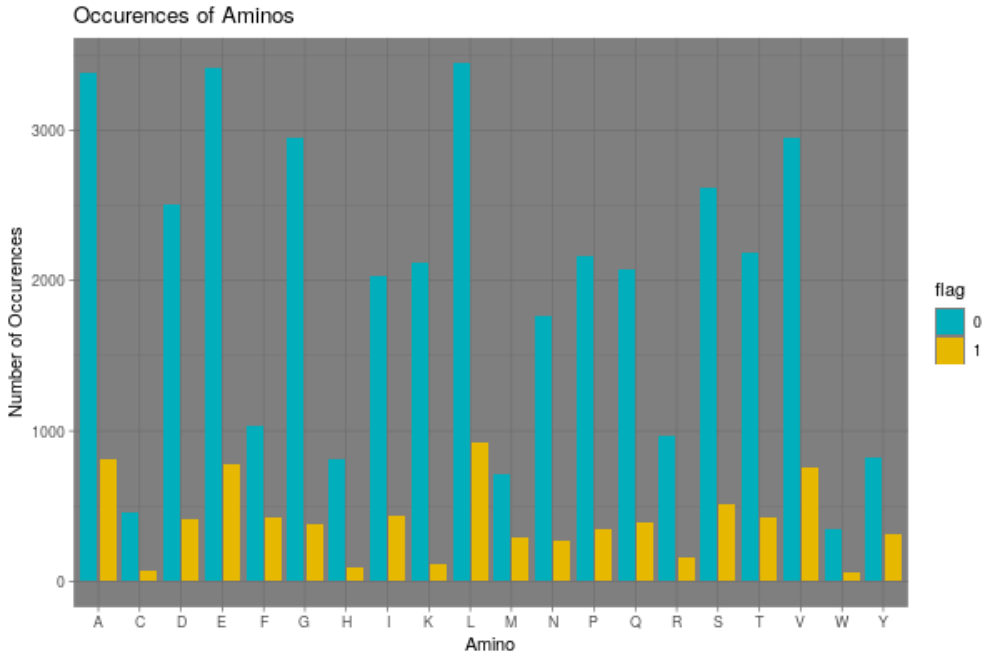
Because we are only concerned with the correlations between the letters and the flag value, only the first row of the heatmap needs to be examined. Extracting and plotting those values, we can see what amino acids are most closely connected with the resulting cleavage site.



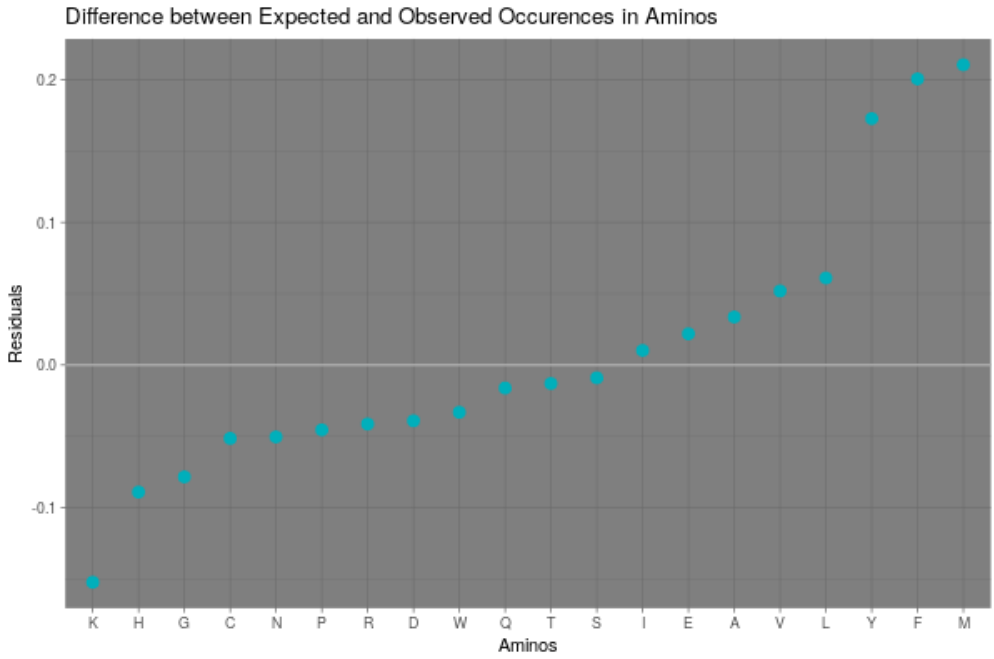
While the correlations are very weak, the five aminos with the highest correlations (absolute value) are:

Aminos	Correlations
F	0.1615793
G	0.1085164
K	0.1871195
M	0.138163
Y	0.1216108

From this table, we expect the importance of these variables to be higher than others. The second method of looking for patterns in the data, was collecting the data into a data frame of column sums, grouped by flag value. The hope was that, differences in the number of occurences (the lowest level of analysis when thinking about amino acid interactions) would show noteable and meaningful features. The following bar plot shows the number of occurences of each amino grouped by flag.



Theoretically, there should be a linear scaling factor, where the occurences of the amino acids have a predictable ratio based on the ratio between cleavage and non-cleavage sites. Using this assumption, the differences in occurences can tell us if an amino acid has noteable importance in the data.



The plot shows there are some amino acids with more cleavage sites than expected, and some with less. The amino acids farthest from zero can be assumed to have more importance when predicting the cleavage of a site than the middle acids. A table shows the aminos with the largest absolute residuals, based on a global estimate of the predicted ratio.

Aminos	Correlations
F	0.1615793
H	0.1085164
K	0.1871195
M	0.138163
Y	0.1216108

It is no coincidence that four of five are in the top of both charts! This provides more evidence towars the hypothesis that these aminos are more closely tied to the activity of the cleavage sites.

Developing and Testing Algorithms

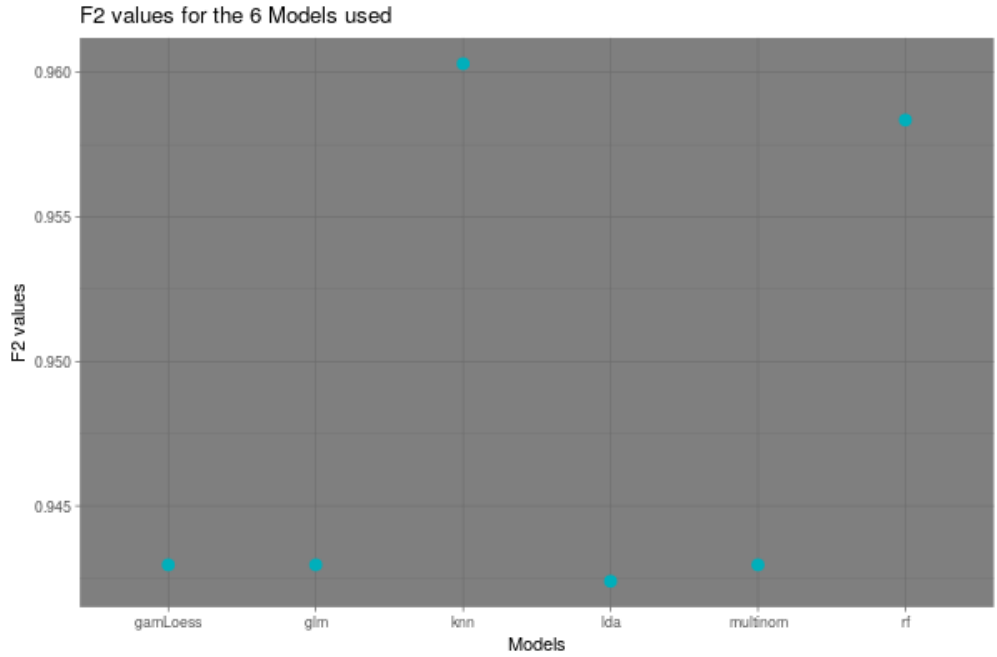
Before train algoriths, a test and train set must be calculated. The test set will be 10% of the train set. The original data has information on the separation of test and train sets, from the individual files. In this project, since the data is treated as one group, a new pair of sets should be created.

```
set.seed(719)
test_index <- createDataPartition(exp_dat$flag, times = 1, p = 0.2, list = FALSE)
test_set <- exp_dat[test_index,]
train_set <- exp_dat[-test_index,]
```

To find which training methods are best for this application, six were chosen for an initial test. Using `lapply`, a vector of the names of the methods were fed into a list of fits, which were then used in another loop to generate corresponding predictions. From there it was simple to isolate the accuracy and F2 values. The results are shown in the following table:

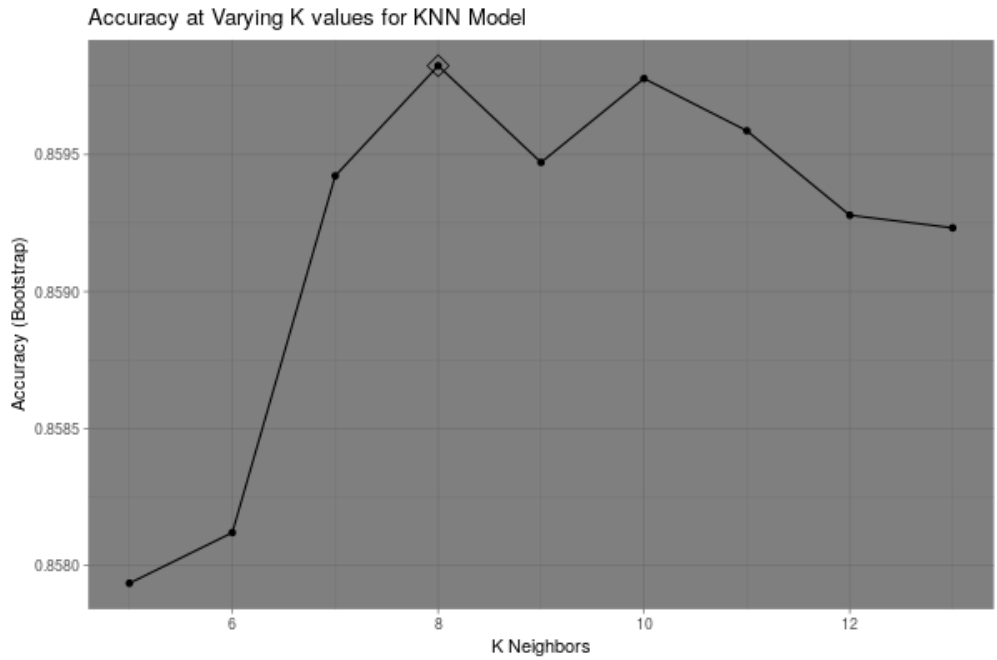
Model	Accuracy	F_2 Value
glm	0.8340462	0.9429604
lda	0.8314799	0.9423926
knn	0.8725406	0.9602983
gamLoess	0.8340462	0.9429604
multinom	0.8340462	0.9429604
rf	0.8776732	0.9583502

From the results of the initial six method training sequence, we can plot the F_2 Values to see the rankings more clearly:



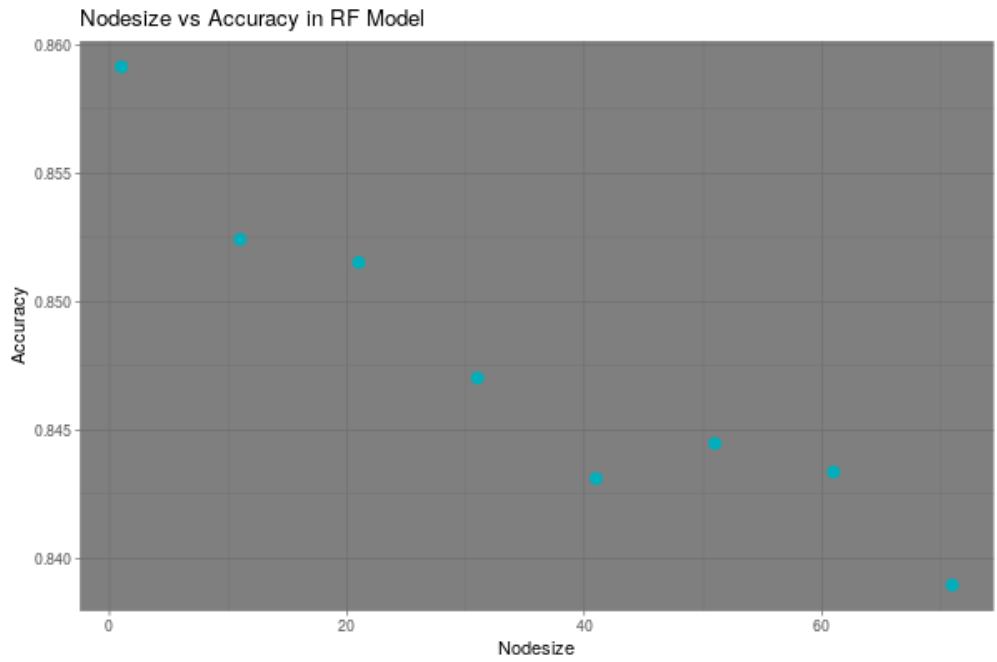
From both figures, it is clear, the *k nearest neighboors* and the *random forest* produce the highest accuracies and best F_2 values. Knowing this, we can easily optimize the fit models by using cross validation. Using the following training code, we can decide on an optimal k value for the final model.

```
set.seed(719)
train_knn <- train(flag ~ ., method = "knn",
                   data = train_set,
                   tuneGrid = data.frame(k = seq(5, 13, 1)))
```



Using a k of 8 the model can be optimized for a final comparison.

In the *random forest* model, the nodesize can be optimized using a similar technique to *knn's* k value. The follow are the accuracies for a range of nodesizes:



Finally, using the nodesize with the maximum accuracy, the optimized rf model can be calculated:

```
train_rf_2 <- randomForest(flag ~ ., data=train_set,
                           nodesize = nodesize[which.max(acc)])

rf_preds <- predict(train_rf_2, test_set)
```

Results

The final two methods for the analysis, *k nearest neighboor* and *random forest*, produced the following results:

Metric	KNN	RF
Accuracy	87.68%	87.43%
Sensitivity	98.56%	98.56%
Specificity	35.00%	33.50%
Balanced Accuracy	66.78%	66.03

We can see that the accuracy well exceeds are beginning baseline, and has a high sensitivity. Unfortunately, both methods have relatively low specificity percentages, making the balanced accuracy very low. This is a disappointing result from the optimized models, but when dealing with occurences that happen rarely, it is difficult to use these training methods to acheive better results than this.

Conclusion

From the results obtained, it is clear that further investigation must be done to acheive a method of predicting with high sensitivity which sites the HIV-1 protease with cleave. The shortcomings in this project start with the method of interacting with the amino acids. In reality, each combination of acids result in a highly distinct site that is impacted by the order of the octamer as well as which acids are present. This project attempted to use the data to simply infer from the number of certain amino acids where the site will be. A background in biochemistry would be helpful to understand and quantify these interactions. Additionally, the data should remain split in further investigations, to isolate the differences in each set, and not allow biases to be absorbed into a larger set. The primary goal of this project is to serve as a learning experience, and it has been very effective in that area.

References

1. https://data-flair.training/blogs/data-science-r-movie-recommendation/ Methods of achieving an expanded amino matrix
2. https://www.datanovia.com/en/blog/ggplot-colors-best-tricks-you-will-love/ Plotting help
3. https://www.r-bloggers.com/make-a-bar-plot-with-ggplot/ Bar plot help
4. https://archive.ics.uci.edu/ml/datasets/HIV-1+protease+cleavage# Machine Learning Institute data page
5. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5259813/ Primary reference material on HIV-1 Protease enzyme