Chapter 2 - Exploratory Data Analysis

Prelude

Early 2019, I was fortunate and came across the book “Applied Predictive Modeling” by Max Kuhn and Kjell Johnson.[1] The focus of this book is the implementation and optimization of predictive modeling. However “Applied Predictive Modeling” is also a demonstration of the power of the R package `caret` written by Max Kuhn and others.

> "The caret package, short for classification and regression training, focuses on simplifying model training and tuning across a wide variety of modeling techniques.[2]

The R package `caret` was written, in part, to overcome the syntactical differences between numerous R machine learning packages as it helps automate the process of model scouting and development.

I plan to discuss “Applied Predictive Modeling” and its tight connection to `caret` in this work since it outlines a methodical process for machine learning.

As one might suspect, many R machine learning packages having been developed by many different scientists have different variable input and output formats. In the spirit of harmonization and simplicity, `caret` standardizes the process of model development. As of August 2019, `caret` can pass information to/from 238 R packages.[3] What `caret` does to facilitate machine learning process development will be discussed in more detail.

Additionally, while searching for information regarding superior Machine Learning techniques, I became of aware of an article co-authored by Ross Quinlan and others, with the grand title, “Top 10 Algorithms in Data Mining”[4]. These 10 algorithms were identified by the IEEE, International Conference on Data Mining in December 2006. The algorithms were voted by the members of this organization as having the highest impact. As one would expect, these 10 algorithms are mainstays of Predictive Modeling. After further research I have also learned that this article and its companion book[5], have become the subject of an MIT course.[6]

At this point, I remembered a paper which used the amino acid sequences of Oxygen binding proteins to classify, using a Support Vector Machine, proteins into 6 categories, entitled “Oxypred: Prediction and Classification of Oxygen-Binding Proteins.”[7]

The “Oxypred” paper claimed that by using a set of 672 oxygen-binding proteins and 700 non-oxygen-binding proteins their model could classify the 7 categories of proteins to >95% using the percent amino acid composition of the 20 standard amino acids as features. Therefore only using 20 x percent amino acid composition predictors it was possible to obtain accuracies of >90% with a Support Vector

Machine.

The 672 proteins were taken from six different classes of oxygen-binding proteins consisting of 6 groups.

7 Class Accuracies Via Support Vector Machine Models

Protein Name	n	Accuracy (%)
Erythrocruorin	20	95.8
Hemerythrin	31	97.5
Hemocyanin	77	97.5
Hemoglobin	486	96.9
Leghemoglobin	13	99.4
Myoglobin	45	96.0
Control - Non-Oxygen binding proteins	700	n/a

I realized that this paper was great choice to model for several reasons. I postulated the high percent accuracies would very likely drop given a larger dataset of Oxygen binding proteins. Since the paper is from 2007, the number of proteins available to any researcher on Uniprot or on the Internet in general would have been small, 672 as mentioned above. As of July, 2019, the number of possible available number of proteins in the same class as Globins were 89,064.

For optimum scouting, Kuhn and Johnson suggest that a much broader framework of half a dozen or more machine learning techniques were commonly used in their opinion for screening in Bioinformaticists before finally choosing a good model candidate. The “Oxypred” paper did not discuss any model development therefore I speculated that a more thorough search using the “The Top Ten Algorithms in Data Mining” may provide an excellent chance to study the differences between the machine learning approaches.

By using these three papers, I would be able to easily generate data which should yield intuitive insights regarding the characteristics of the proteins of interest. But more importantly, the work would allow any reader to learn how a class of very diverse and impactful machine learning algorithms would stack up in the face of a small dataset. The preparation of the dataset used in this research is discussed in Appendix A.

Why use Exploratory Data Analysis?

The first steps a Data Scientist takes after data is chosen or provided is called *Exploratory Data Analysis* (EDA). EDA is a systematic process which can help uncover the most from your data. However, as you encounter more data over time you will face new problems. Consequently, this step should be taken patiently. New problems lead to new decisions which lead to new questions, new

approaches and even your further analysis.

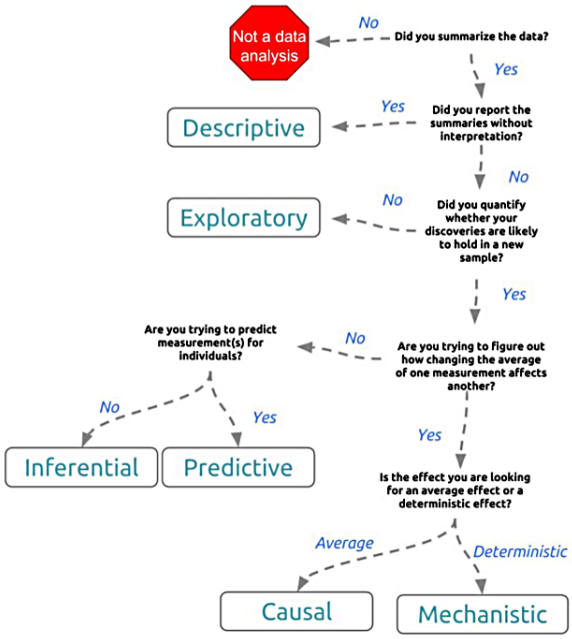
In Roger Peng's book, Exploratory Data Analysis with R (http://leanpub.com/exdata)[8] he outlines a simple ten step process for importing and exploring new data sets. This booklet is a great starter hands-on manual from a leader in the field. "EDA with R" is written with many R code examples. I recommend Exploratory Data Analysis with R for undergraduates and above unfamiliar with R. A FREE PDF file may be found at LeanPub.com.

Exploratory Data Analysis Checklist

1. Formulate your questions
2. Read in your data
3. Check the files
4. Use `str()`
5. Look at your data from `head` to `tail`
6. Check your n's & na's
7. Generate statistics / graphics
8. Challenge your solution
9. Follow up/Conclude

The list above was taken from 'Exploratory Data Analysis with R' and modified slightly to fit my needs.

I have also found Jeff Leek's book The Elements of Data Analytic Style (https://leanpub.com/datastyle) which discuss Data Science. The "data analysis question type flow chart" was also amenable to my own work. This book can also be found for free at Leanpub.com



Jeff Leek's Flow Chart

2.1 - Formulate your questions

After a cursorary view of the Uniprot data, I was interested to learn if the data would need to pre-processed or transformed before machine learning classification. Since the data from the proteins was already in values between [0 and 1], so scaling should not be an obvious problem. There are many pitfalls that can plague machine learning if the data is not pre-processed.

2.2 - Import data

- Inspect your data using command line interface using `less` .
- Question: Do we need a special library for importing data?
- Answer: NO, The data I produced is rectangular and in csv format.

Note: `read.csv()` by default reads character data as levels or factors, which is beneficial to us at this junction.

```
test_harness_paa <- read.csv("data/test_harness_paa.csv")
```

2.3 - Check your files

Dataset Problems?

When trying to determine if there are any problems with your data it is best to consider at least these four points.

1. Data types: check for characters where numbers should be and vice versa.
2. Numerical differences: use of the point versus the comma.
3. Missing values: Do the missing values have a pattern or not.
4. File structure: Some file types maybe in binary instead of ASCII or even the language might Cyrillic or bad Unicode.

Data Types

```
is.data.frame(test_harness_paa)
```

```
## [1] TRUE
```

```
class(test_harness_paa$Class)    # Col 1
```

```
## [1] "factor"
```

```
class(test_harness_paa$id)       # Col 2
```

```
## [1] "factor"
```

```
class(test_harness_paa$TotalAA)  # Col 3
```

```
## [1] "integer"
```

```
class(test_harness_paa$A)        # Col 4
```

```
## [1] "numeric"
```

```
# Column Name Pre-Processing Only
# Alphabetize the ordering of amino acids
Classes <- test_harness_paa[, 1]
th_paa_values <- test_harness_paa[,-c(1,2,3)]
ordered_paa_values_only <- th_paa_values[, order(colnames(th_paa_values))]
ordered_paa <- cbind(Classes, ordered_paa_values_only)
```

2.4 - Use `str()`

- Using `str()`, we see agreement with information provided by 'Check your files/Data Types.'

```
str(ordered_paa)
```

```
## 'data.frame':    3500 obs. of  21 variables:
##  $ Classes: Factor w/ 7 levels "Ctrl","Ery","Hcy",..: 1 1 1 1 1 1 1 1 1 1
...
##  $ A      : num  0.0592 0.0376 0.041 0.0625 0.0637 ...
##  $ C      : num  0.00402 0.0098 0 0.05208 0.01068 ...
##  $ D      : num  0.0723 0.0474 0.0462 0.0469 0.0577 ...
##  $ E      : num  0.0793 0.085 0.0846 0.0573 0.0854 ...
##  $ F      : num  0.0341 0.0261 0.0205 0.0521 0.033 ...
##  $ G      : num  0.0402 0.085 0.0718 0.0833 0.0677 ...
##  $ H      : num  0.0191 0.018 0.0103 0.0365 0.024 ...
##  $ I      : num  0.0673 0.0376 0.0359 0.026 0.0367 ...
##  $ K      : num  0.07028 0.0098 0.00513 0.05729 0.06807 ...
##  $ L      : num  0.0924 0.0572 0.059 0.0677 0.0801 ...
##  $ M      : num  0.012 0.0163 0.0103 0.0104 0.028 ...
##  $ N      : num  0.0462 0.067 0.0795 0.0104 0.047 ...
##  $ P      : num  0.0271 0.0458 0.0513 0.0625 0.0777 ...
##  $ Q      : num  0.0833 0.0458 0.0564 0.0365 0.0597 ...
##  $ R      : num  0.0572 0.1193 0.1513 0.0521 0.0594 ...
##  $ S      : num  0.0532 0.1716 0.1359 0.1302 0.0791 ...
##  $ T      : num  0.0572 0.0637 0.0821 0.0677 0.044 ...
##  $ V      : num  0.0693 0.0376 0.041 0.0521 0.0464 ...
##  $ W      : num  0.01004 0.0049 0.00513 0.01042 0.01034 ...
##  $ Y      : num  0.0462 0.0147 0.0128 0.026 0.021 ...
```

2.5 - Check your data with `head` & `tail`

```
head(ordered_paa, n=2)
```

```
##   Classes          A           C          D          E          F
## 1    Ctrl 0.05923695 0.004016064 0.07228916 0.07931727 0.03413655
## 2    Ctrl 0.03758170 0.009803922 0.04738562 0.08496732 0.02614379
##            G          H          I           K          L          M
## 1 0.04016064 0.01907631 0.06726908 0.070281124 0.09236948 0.01204819
## 2 0.08496732 0.01797386 0.03758170 0.009803922 0.05718954 0.01633987
##            N          P          Q          R          S          T
## 1 0.04618474 0.02710843 0.08333333 0.05722892 0.05321285 0.05722892
## 2 0.06699346 0.04575163 0.04575163 0.11928105 0.17156863 0.06372549
##            V           W          Y
## 1 0.06927711 0.010040161 0.04618474
## 2 0.03758170 0.004901961 0.01470588
```

```
tail(ordered_paa, n=2)
```

```
##      Classes         A C          D          E          F         G
## 3499     Mgb 0.1103896 0 0.05844156 0.07792208 0.04545455 0.0974026
## 3500     Mgb 0.0974026 0 0.05844156 0.09740260 0.03896104 0.0974026
##               H          I         K         L          M          N
## 3499 0.05844156 0.06493506 0.1168831 0.1103896 0.01948052 0.01298701
## 3500 0.03246753 0.03896104 0.1428571 0.1233766 0.02597403 0.02597403
##               P          Q           R          S          T          V
## 3499 0.02597403 0.05844156 0.019480519 0.03246753 0.02597403 0.03896104
## 3500 0.02597403 0.03896104 0.006493506 0.05194805 0.02597403 0.04545455
##               W          Y
## 3499 0.01298701 0.01298701
## 3500 0.01298701 0.01298701
```

2.6 - Check your n's and na's

- Dimensions: **Correct**

```
dim(ordered_paa)
```

```
## [1] 3500   21
```

- Number of Proteins Per Class: **Correct**

```
class_table <- table(ordered_paa$Class)
knitr::kable(class_table)
```

Var1	Freq
Ctrl	500
Ery	500
Hcy	500
Hgb	500
Hhe	500
Lgb	500
Mgb	500

- Checking for missing values, we see agreement **No missing values found.**

```
sapply(ordered_paa, function(x) sum(is.na(x)))
```

```
## Classes       A       C       D       E       F       G       H       I
##       0       0       0       0       0       0       0       0       0
##       K       L       M       N       P       Q       R       S       T
##       0       0       0       0       0       0       0       0       0
##       V       W       Y
##       0       0       0
```

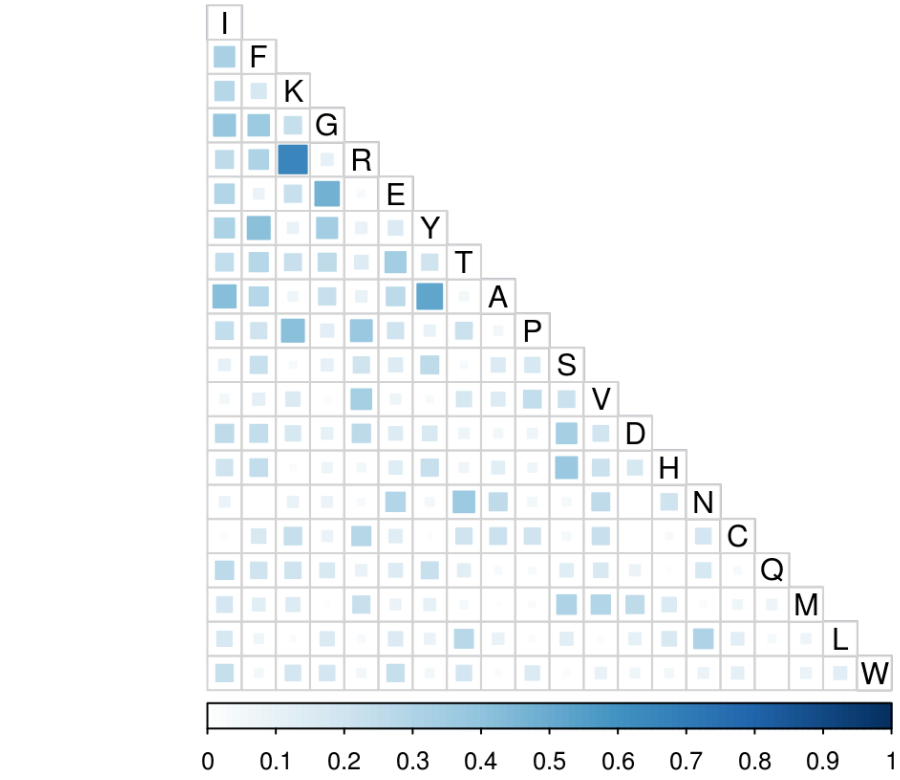
2.7 - Generate statistics and graphs

2.7.1 - Correlation Coefficients - Feature Reduction

A easily understandable test, is correlation 2D-plot for investigating multicollinearity or feature reduction. It is clear that fewer attributes “means decreased computational time and complexity. Secondly, if two predictors are highly correlated, this implies that they are measuring the same underlying information. Removing one should not compromise the performance of the model and might lead to a more parsimonious and interpretable model. Third, some models can be crippled by predictors with degenerate distributions”.[9]

Correlation Using `corrplot`

```
library("corrplot")
library("RColorBrewer")

thpaa_corr_mat = cor(ordered_paa[,c(2:21)], method = "p") # "p": Pearson test
for continous variables

corrplot(abs(thpaa_corr_mat),
         title = "Correlation Plot Of AA w/ 1st Principle Component Orderin
g",
         method = "square",
         type = "lower",
         tl.pos = "d",
         cl.lim = c(0, 1),
         addgrid.col = "lightgrey",
         cl.pos = "b", # Color legend position bottom.
         order = "FPC", # "FPC" = first principal component order.
         mar = c(1, 2, 1, 2),
         tl.col = "black")
```




```
thpaa_corr_mat["R", "K"]
```

```
## [1] -0.6590393
```

- There is **no reason to consider multicollinearity**.

If the correlation plot produced any values greater than or equal to (R >= |0.75|) then we could consider feature elimination. This interesting heuristic approach would be used for determining which feature to eliminate.[10]

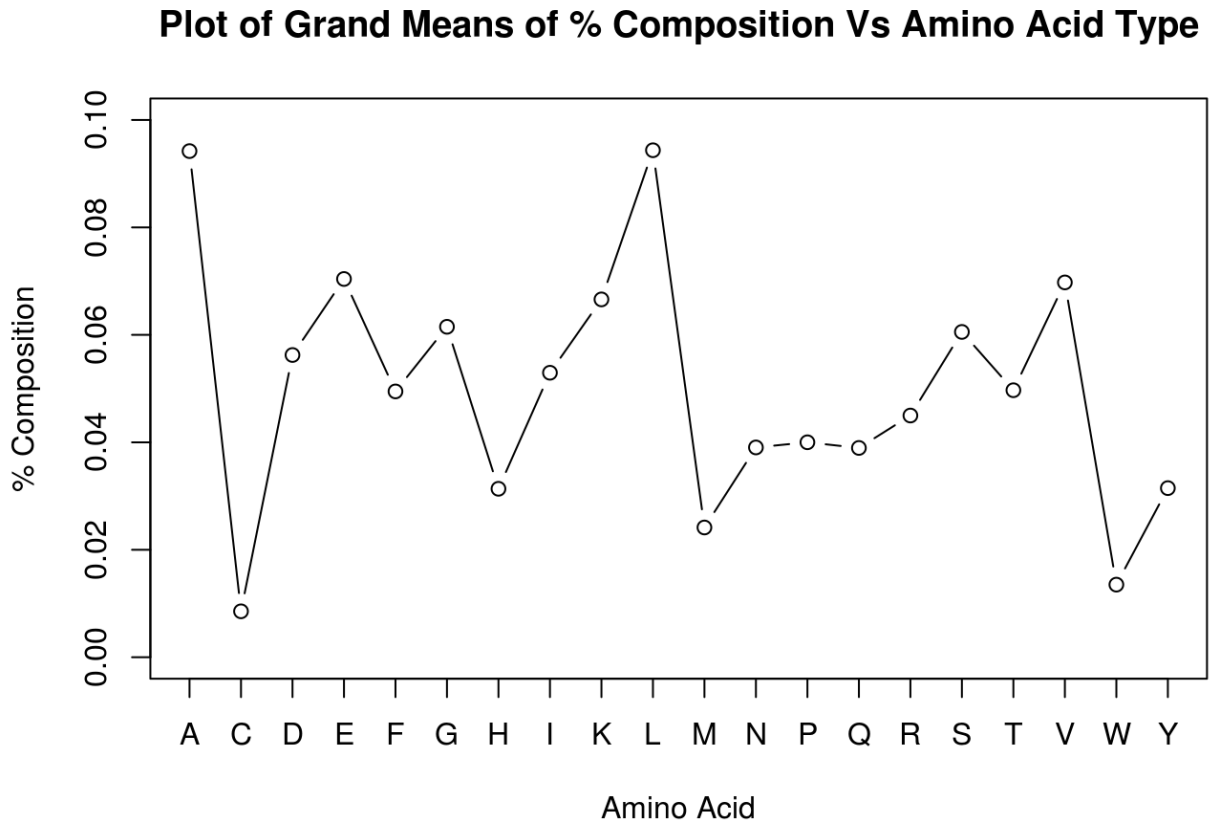
1. Calculate the correlation matrix of the predictors.
2. Determine the two predictors associated with the largest absolute pairwise correlation (R > |0.75|), call them predictors A and B.
3. Determine the average correlation between A and the other variables. Do the same for predictor B.
4. If A has a larger average correlation, remove it; otherwise, remove predictor B.
5. Repeat Steps 2–4 until no absolute correlations are above the threshold.

2.7.2 - First Moment Of A Distribution - Mean

2.7.2.1 - Investigate Grand Mean Vs Vs Amino Acid Type

- The top 3 peak values include Alanine (A), Glutamic acid (E), Leucine (L).

```
AA_ave <- colMeans(ordered_paa_values_only)
plot(AA_ave,
     main = "Plot of Grand Means of % Composition Vs Amino Acid Type",
     ylab = "% Composition",
     xlab = "Amino Acid",
     ylim = c(0, 0.1),
     type = "b",
     xaxt = "n")
axis(1, at = 1:20, labels = names(ordered_paa_values_only))
```



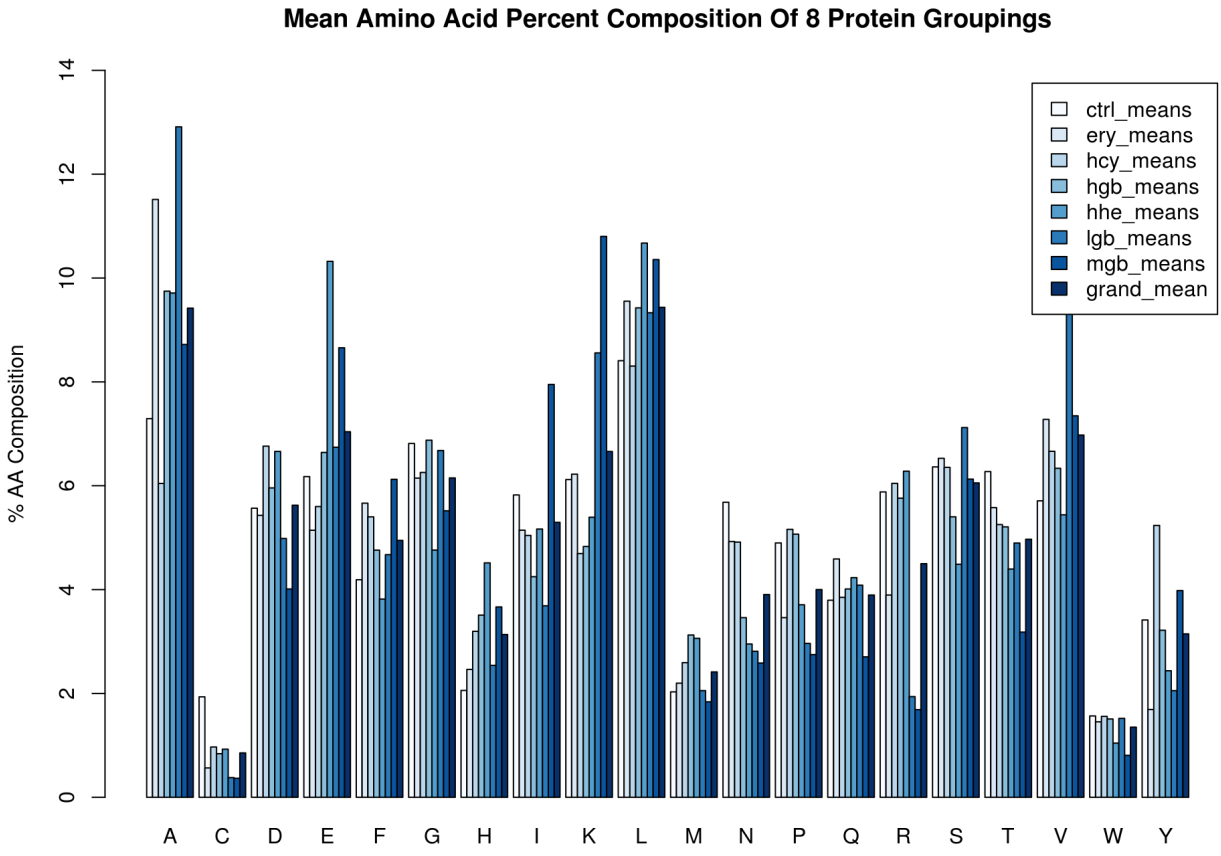
2.7.2.2 - Produce Grouped Barchart Of Amino Acids Vs. Group

Produce a *Grouped Barchart* of the mean percent of the percent amino acid composition of all 7 groups.

Pseudo-code:
1. Subset 7 protein groups, {Ctrl, Ery, Hcy, Hgb, Hhe, Lgb, Mgb} & Grand-Mean of All 7 sets
2. Determine column means for each protein class

3. Calculate percentage values
4. Produce Grouped Bar Plot

```
# 1. Subset 7 protein groups, {Ctrl, Ery, Hcy, Hgb, Hhe, Lgb, Mgb} & Grand-Me
an of All 7 sets
ctrl_set <- ordered_paa[ which(ordered_paa$Class == 'Ctrl'),]
ery_set  <- ordered_paa[ which(ordered_paa$Class == 'Ery'),]
hcy_set  <- ordered_paa[ which(ordered_paa$Class == 'Hcy'),]
hgb_set  <- ordered_paa[ which(ordered_paa$Class == 'Hgb'),]
hhe_set  <- ordered_paa[ which(ordered_paa$Class == 'Hhe'),]
lgb_set  <- ordered_paa[ which(ordered_paa$Class == 'Lgb'),]
mgb_set  <- ordered_paa[ which(ordered_paa$Class == 'Mgb'),]

# 2. Determine column means for each protein class
ctrl_means <- apply(ctrl_set[, 2:21], 2, mean)
ery_means  <- apply(ery_set[, 2:21], 2, mean)
hcy_means  <- apply(hcy_set[, 2:21], 2, mean)
hgb_means  <- apply(hgb_set[, 2:21], 2, mean)
hhe_means  <- apply(hhe_set[, 2:21], 2, mean)
lgb_means  <- apply(lgb_set[, 2:21], 2, mean)
mgb_means  <- apply(mgb_set[, 2:21], 2, mean)
grand_mean <- apply(ordered_paa[, 2:21], 2, mean)

# 3. Calculate percentage values
data = data.frame(ctrl_means, ery_means, hcy_means, hgb_means,
                  hhe_means, lgb_means, mgb_means, grand_mean)

percent_aa = as.matrix(t(100*data))

# 4. Produce *Grouped Bar Plot* of mean percent amino acid composition for 8
Classes of Proteins
barplot(percent_aa,
        ylim = c(0, 14),
        main = "Mean Amino Acid Percent Composition Of 8 Protein Groupings",
        ylab = "% AA Composition",
        col = colorRampPalette(brewer.pal(9,"Blues"))(8),
        legend = T, # Set legend to FALSE to see behind box
        beside = T)
```



2.7.3 - Second Moment Of A Distribution – Variance

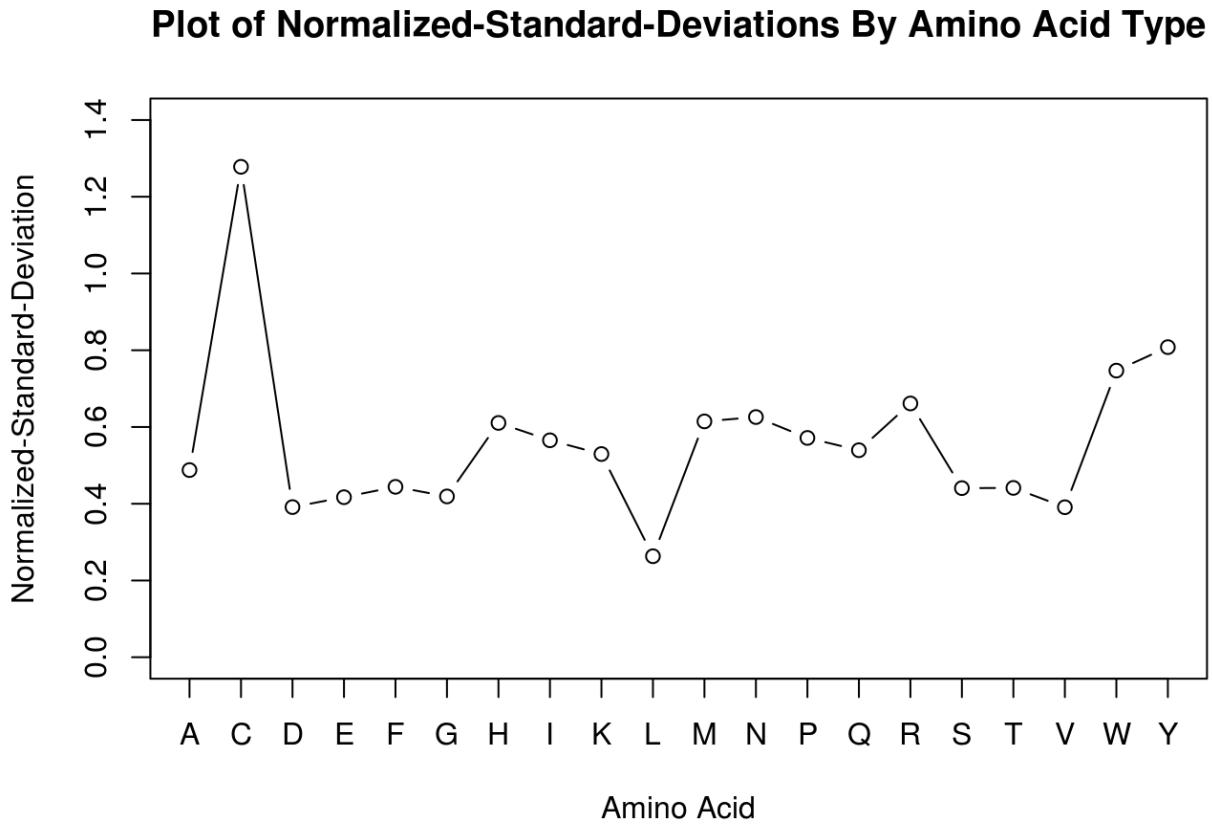
Investigating Normalized Standard Deviations

Standard deviations are sensitive to scale. Since I am trying to compare the standard deviations amongst different amino acid values, we can normalize them. This normalization is more commonly called coefficient of variation (CV).

$$\sigma_{normalized}(x) = \frac{\sigma(x)}{E[\,|x|\,]}$$

- It is interesting to see that Cysteine(C) has the largest Normalized-Standard-Deviation given its very low percent composition.

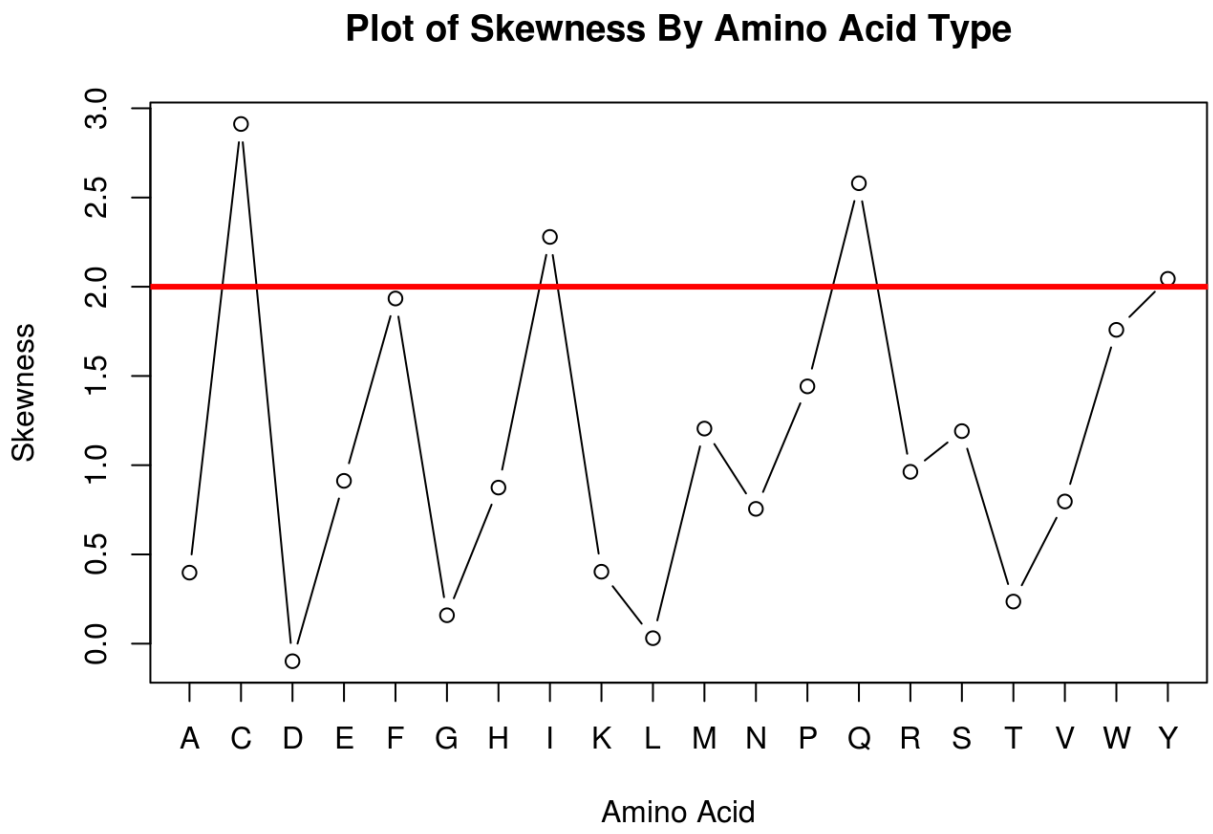
```
AA_var_norm <- (apply(ordered_paa_values_only, 2, sd)) / AA_ave
plot(AA_var_norm,
     main = "Plot of Normalized-Standard-Deviations By Amino Acid Type",
     ylab = "Normalized-Standard-Deviation",
     xlab = "Amino Acid",
     ylim = c(0, 1.4),
     type = "b",
     xaxt = "n")
axis(1, at = 1:20, labels = names(ordered_paa_values_only))
```



2.7.4 - Third Moment Of A Distribution - Skewness

- Skewness values for each A.A. by Class

```
AA_skewness <- (apply(ordered_paa_values_only, 2, e1071::skewness))
plot(AA_skewness,
     main = "Plot of Skewness By Amino Acid Type",
     ylab = "Skewness",
     xlab = "Amino Acid",
     type = "b",
     xaxt = "n")
axis(1, at = 1:20, labels = names(ordered_paa_values_only))
abline(h = 2, col = "red", lwd = 3)
```



> A general rule of thumb to consider is that skewed data whose ratio of the highest value to the lowest value is greater than 20 and skew > 20 have significant skewness.
>
> Applied Predictive Modeling, P.31

Generally Skew values above 2.0 are considered high.

- As we can see the amino acids {C, I, Q, Y} are above 2.0.

Amino acid	Skewness	Ratio of Max : Min
C, Cyteine	2.912	208.8
I, Isoleucine	2.279	75.2
Q, Glutamine	2.580	75.1
Y, Tyrosine	2.044	79.2

```
AA_skewness
```

```
##           A           C           D           E           F           G
##  0.39790430  2.91201697 -0.09855225  0.91222637  1.93418033  0.15907804
##           H           I           K           L           M           N
##  0.87451506  2.27919765  0.40305779  0.03054457  1.20517068  0.75528464
##           P           Q           R           S           T           V
##  1.44168086  2.58004855  0.96288759  1.19093700  0.23577331  0.79660722
##           W           Y
##  1.75853515  2.04425182
```

Cysteine: Ratio of values of Highest/Lowest value

```
max_val <- max(ordered_paa_values_only$C)
min_val <- min(ordered_paa_values_only$C[ordered_paa_values_only$C > 0])

ratio_Cys <- max_val / min_val
ratio_Cys
```

```
## [1] 208.7609
```

Isoleucine: Ratio of values of Highest/Lowest value

```
max_val <- max(ordered_paa_values_only$I)
min_val <- min(ordered_paa_values_only$I[ordered_paa_values_only$I > 0])

ratio_Iso <- max_val / min_val
ratio_Iso
```

```
## [1] 75.1875
```

Glutamine: Ratio of values of Highest/Lowest value

```
max_val <- max(ordered_paa_values_only$Q)
min_val <- min(ordered_paa_values_only$Q[ordered_paa_values_only$Q > 0])

ratio_Glu <- max_val / min_val
ratio_Glu
```

```
## [1] 75.05882
```

Tyrosine: Ratio of values of Highest/Lowest value

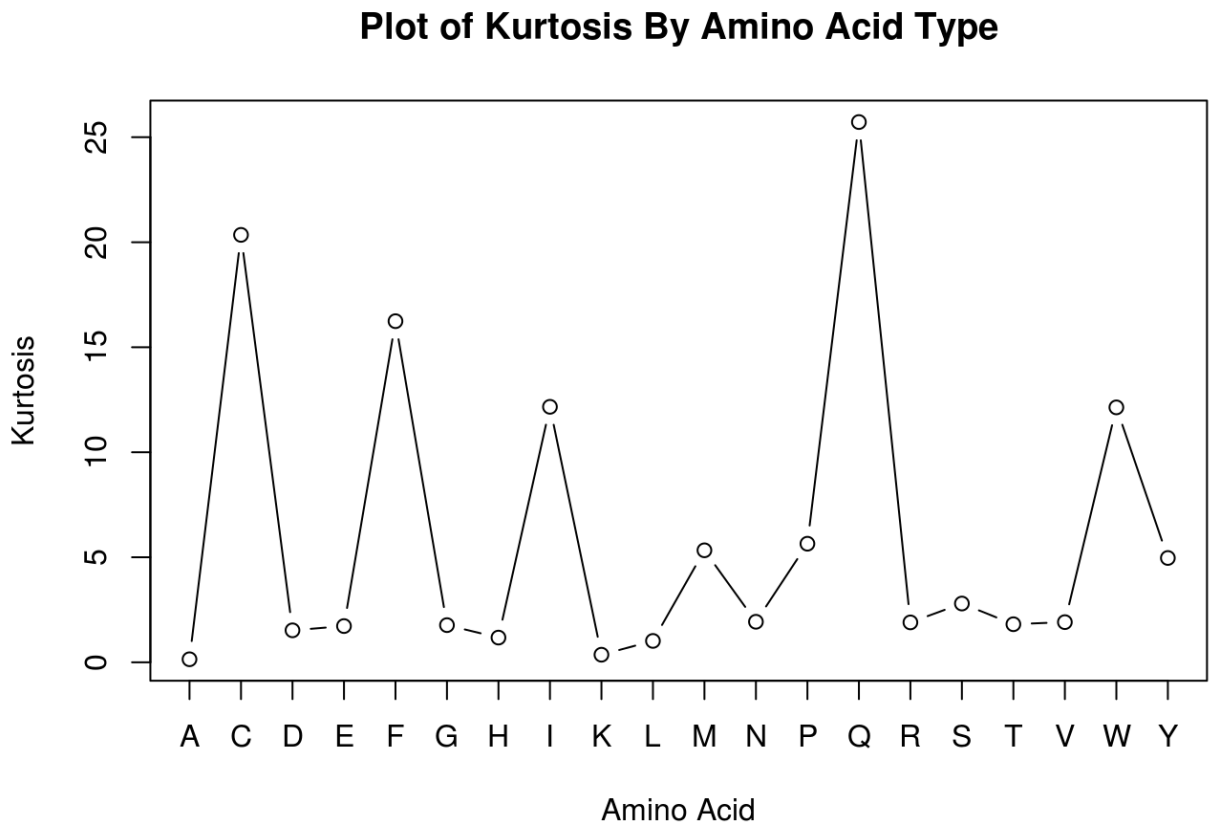
```
max_val <- max(ordered_paa_values_only$Y)
min_val <- min(ordered_paa_values_only$Y[ordered_paa_values_only$Y > 0])

ratio_Tyr <- max_val / min_val
ratio_Tyr
```

```
## [1] 79.24658
```

2.7.5 - Fourth Moment Of A Distribution - Kurtosis

Generate kurtosis values for each A.A. in Totality

```
AA_kurtosis <- (apply(ordered_paa_values_only, 2, e1071::kurtosis))
plot(AA_kurtosis,
     main = "Plot of Kurtosis By Amino Acid Type",
     ylab = "Kurtosis",
     xlab = "Amino Acid",
     type = "b",
     xaxt = "n")
axis(1, at = 1:20, labels = names(ordered_paa_values_only))
```



NOTES: 1. https://www.researchgate.net /post/What_is_the_acceptable_range_of_skewness_and_kurtosis_for_normal_distribution_of_data (https://www.researchgate.net /post/What_is_the_acceptable_range_of_skewness_and_kurtosis_for_normal_distribution_of_data)

I used indices for acceptable limits of ±2 (Trochim & Donnelly, 2006; Field, 2000 & 2009; Gravetter & Wallnau, 2014) Hope this helps!

- References:
- Trochim, W. M., & Donnelly, J. P. (2006). The research methods knowledge base (3rd ed.). Cincinnati, OH:Atomic Dog.
- Gravetter, F., & Wallnau, L. (2014). Essentials of statistics for the behavioral sciences (8th ed.). Belmont, CA: Wadsworth.
- Field, A. (2000). Discovering statistics using spss for windows. London-Thousand Oaks- New Delhi: Sage publications.
- Field, A. (2009). Discovering statistics using SPSS. London: SAGE.

2.7.6 - Data Transformations For 4 Skewed Features

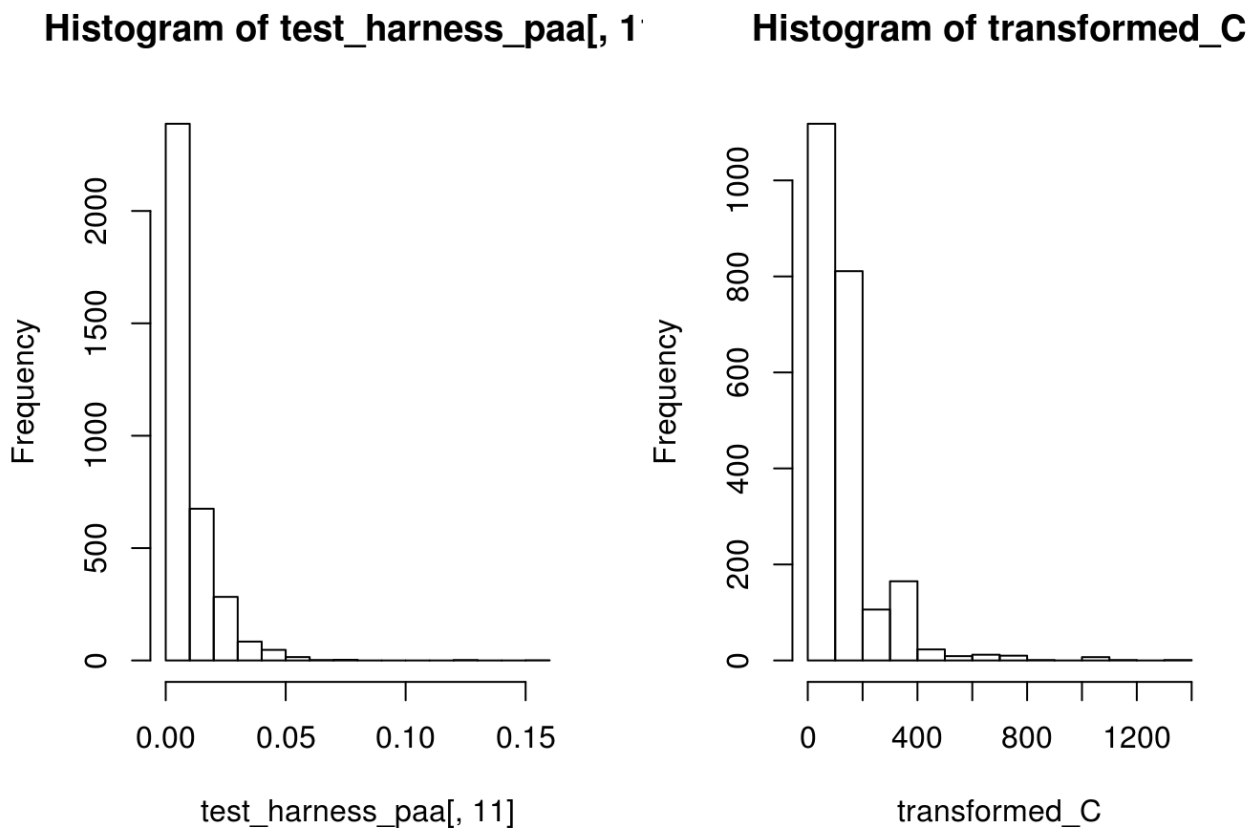
- Above we found in section 2.7.4, Skew was a found to a high degree and this suggested that data transformations should be carried out on the 4 attributes. These attributes were the amino acids {C, I, Q, Y} which had skew greater than 2.0 and a ratio of (maximum : minimum) greater than 20.

2.7.6.1 - Data Transformation of Cysteine

```
transformed_C <- 1/(test_harness_paa[,11])
transformed_C_skewness <- e1071::skewness(transformed_C)
transformed_C_skewness
```

```
## [1] NaN
```

```
# 2 figures arranged in 1 rows and 2 columns
par(mfrow=c(1,2))
hist(test_harness_paa[,11])
hist(transformed_C)
```



2.7. - Principle Component Analysis Using `princomp`

```
aa_PCA = princomp(ordered_paa_values_only)
names(aa_PCA)
```

```
## [1] "sdev"     "loadings" "center"   "scale"    "n.obs"    "scores"
## [7] "call"
```

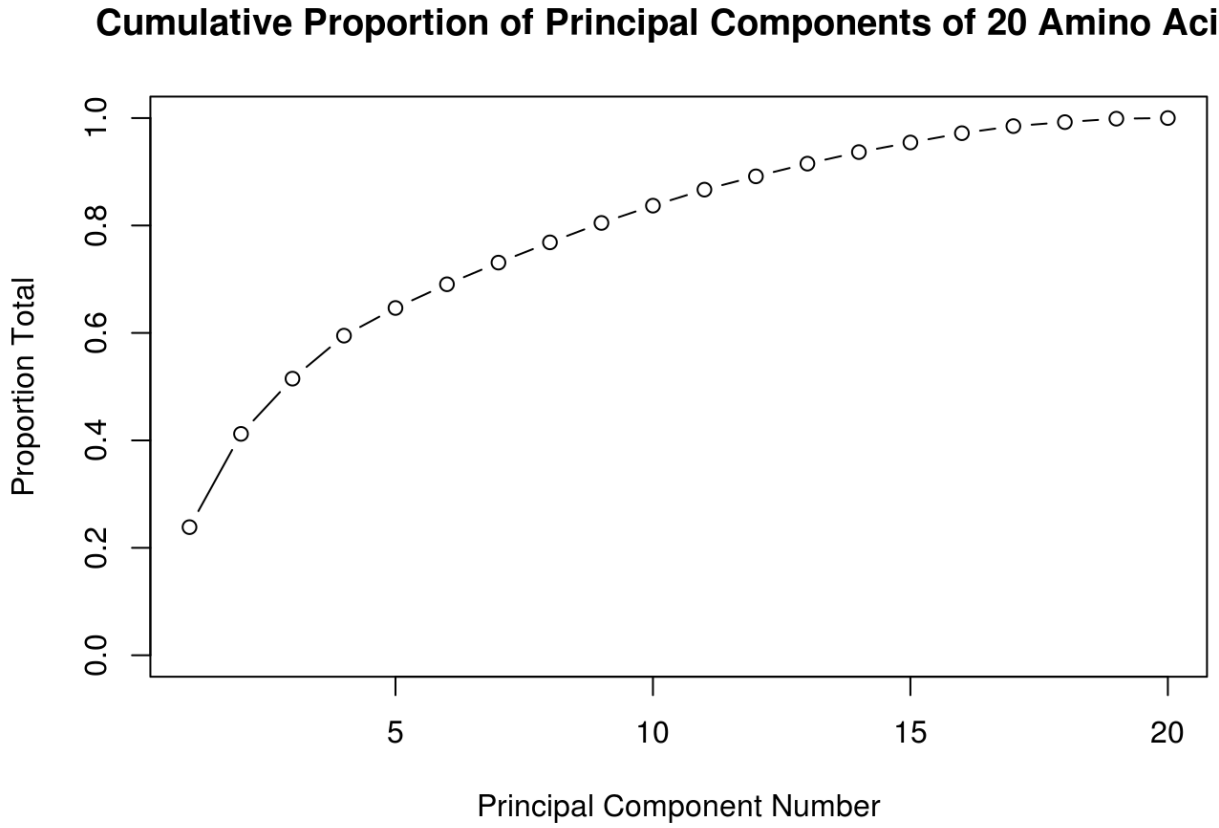
Investigate the cumulative proportion of the principle Components

```
cumsum(aa_PCA$sdev^2 / sum(aa_PCA$sdev^2))
```

```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7
## 0.2385014 0.4121097 0.5148883 0.5949217 0.6464432 0.6905978 0.7308998
##    Comp.8    Comp.9   Comp.10   Comp.11   Comp.12   Comp.13   Comp.14
## 0.7685861 0.8047531 0.8369133 0.8667231 0.8914812 0.9151191 0.9366097
##   Comp.15   Comp.16   Comp.17   Comp.18   Comp.19   Comp.20
## 0.9544947 0.9718148 0.9849630 0.9923993 0.9988034 1.0000000
```

Now, plot cumulative values.

```
plot(cumsum(aa_PCA$sdev^2 / sum(aa_PCA$sdev^2)),
     main = "Cumulative Proportion of Principal Components of 20 Amino Acid
s",
     xlab = "Principal Component Number",
     ylab = "Proportion Total",
     ylim = c(0,1),
     type = "b")
```



WRITE TAKEAWAY MESSAGE:

Var hi to lowest a, k, i, r, e + w alanine, lysine, isoleucine, arginine, glutamic acid, tryptophan

2.7.7 - Violin Plots

2.7.7.1 - c(C,I,Q,Y,D)

```
x_data <- as.data.frame(c(ordered_paa_values_only[,2],
                          ordered_paa_values_only[,8],
                          ordered_paa_values_only[,14],
                          ordered_paa_values_only[,20],
                          ordered_paa_values_only[,3]))
y_data <- c(rep("C", 3500),
               rep("I", 3500),
               rep("Q", 3500),
               rep("Y", 3500),
               rep("D", 3500))

tempdata <- data.frame(x_data, y_data)
colnames(tempdata)[1] <- "paa"
colnames(tempdata)[2] <- "AA"
#head(tempdata)
```

2.8 - Challenge your solution

2.9 - Follow up/Conclude

1. “Applied Predictive Modeling”, Max Kuhn and Kjell Johnson, Springer Publishing, 2018, (http://appliedpredictivemodeling.com/ (http://appliedpredictivemodeling.com/))↩
2. Building Predictive Models in R Using the caret Package, Max Kuhn, J. Stat. Soft., Nov. 2008, 28, 5, www.jstatsoft.org/v28/i05/paper↩
3. The caret Package (https://topepo.github.io/caret/index.html (https://topepo.github.io/caret/index.html))↩
4. “Top 10 Algorithms in Data Mining”, X. Wu etal, Knowl Inf Syst, 2008, 14:1–37, DOI 10.1007/s10115-007-0114-2↩
5. “The Top Ten Algorithms in Data Mining”, edited by X. Wu, V. Kumar, CRC Press, 2009, Book available on Google Books (https://books.google.com/books?id=_kcEn-c9kYAC&printsec=frontcover&source=gbs_ViewAPI#v=onepage&q&f=false)↩
6. Cynthia Rudin, 15.097 Prediction: Machine Learning and Statistics. Spring 2012. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu/ (https://ocw.mit.edu/), Creative Commons BY-NC-SA.↩
7. Exploratory Data Analysis with R, Roger D. Peng, Lean Publishing, 2016-05-18.↩
8. Exploratory Data Analysis with R, Roger D. Peng, Lean Publishing, 2016-05-18.↩
9. “Applied Predictive Modeling”, Max Kuhn and Kjell Johnson, Springer Publishing, 2018, P.43↩

10. “Applied Predictive Modeling”, Max Kuhn and Kjell Johnson, Springer Publishing, 2018, P.47 (http://appliedpredictivemodeling.com/ (http://appliedpredictivemodeling.com/))↩