Biostatistic Final Project
Kaitlyn Terrell, Jacob Halle, Kristina Ivanov

Project 1.
==========

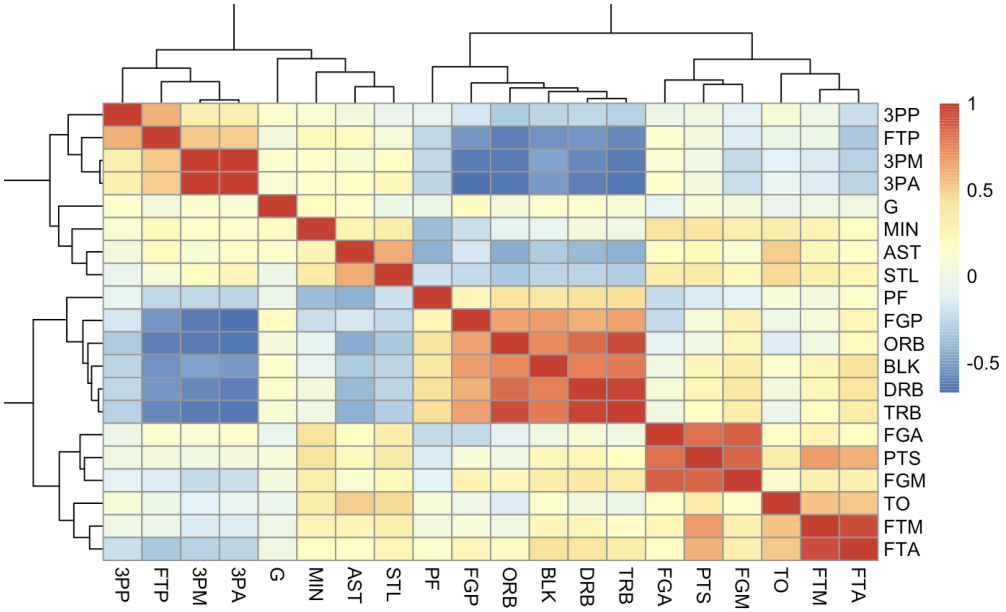


Figure 1a. In the analysis above, I used the "cor" function to calculate the Pearson correlation matrix for the scaled NBA player statistic. The "pheatmap" function is used to create a heatmap and the color representation ranges from blue (strong negative correlation), white (no correlation), to bright red (strong correlation).Within this heat map, we are able to see how different performance metrics correlate with each other. For example the 3PA and 3PM are bright red which indicate that they are highly correlated. This could be useful to see which metrics are not correlated and see which players stand out between different metrics.

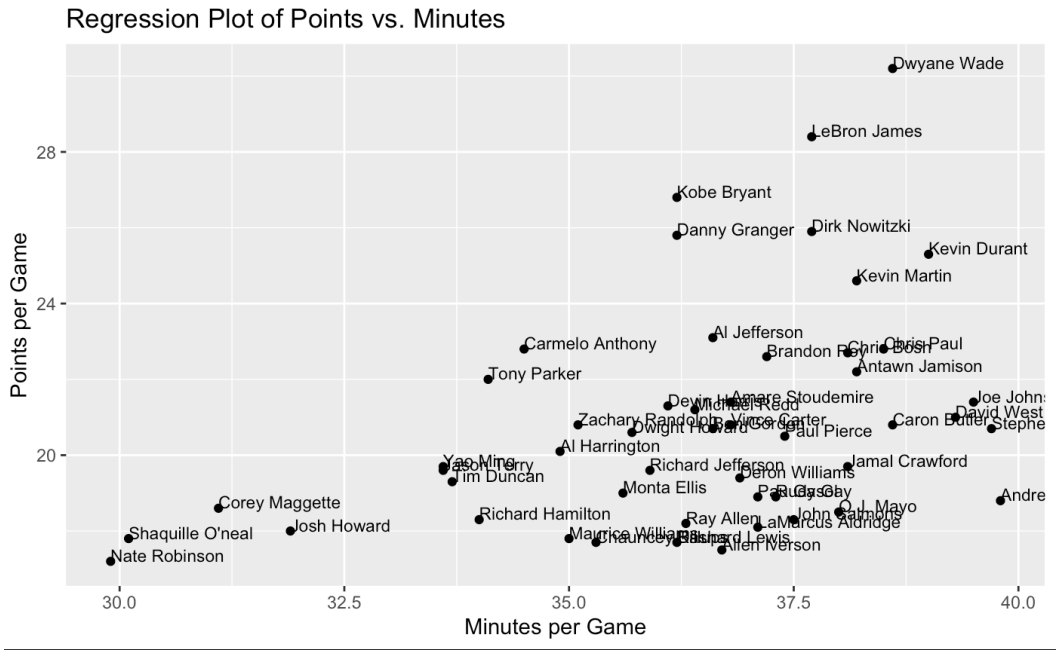


Figure 1b. This plot above is to show the amount of minutes played per game, versus the amount of points they scored per game. By looking at this data, it shows that the longer the player played the more points they scored.

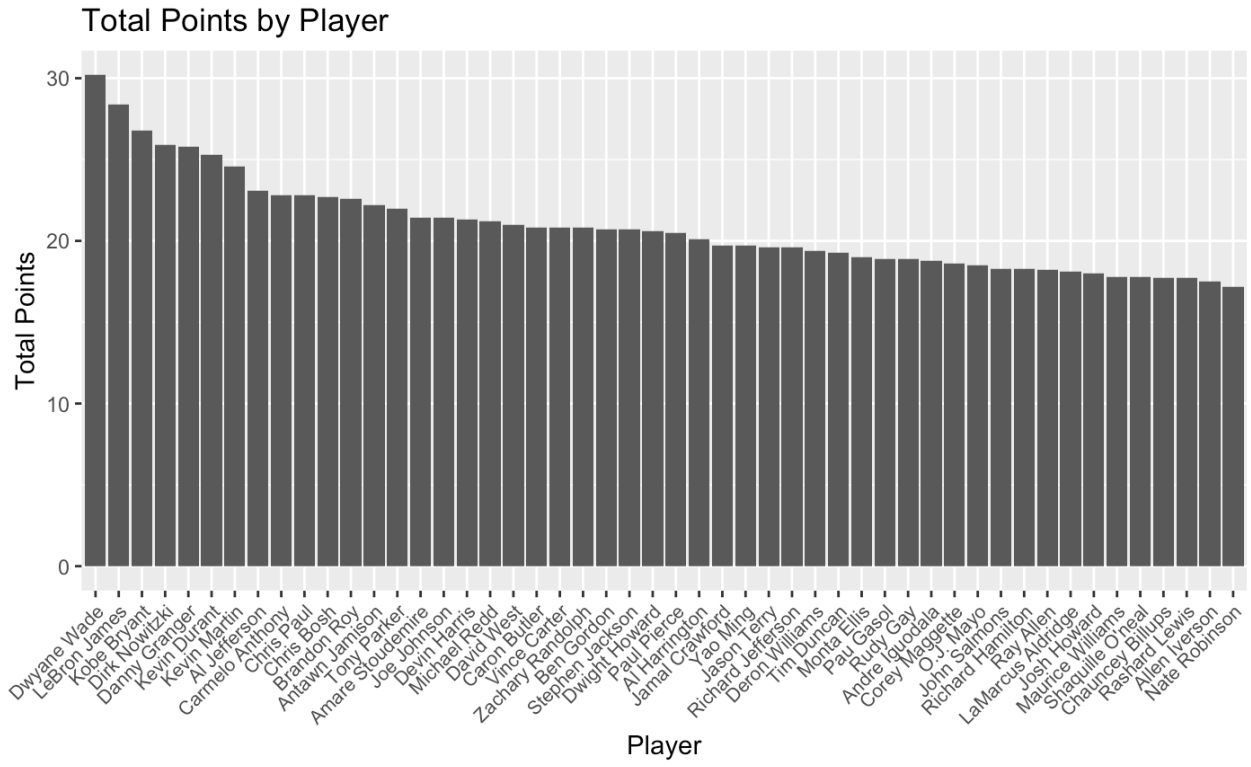


Figure 1c. Total points scored showed above. By viewing this, we can see the top few players that end up scoring the most points for their team. This could be a good indication of who will become a successful basketball player and continue growing in their career. It also seems like the average and total amount of scored points also has the same top players. This indicates that they have been consistent throughout the season, if the average is similar to the total.

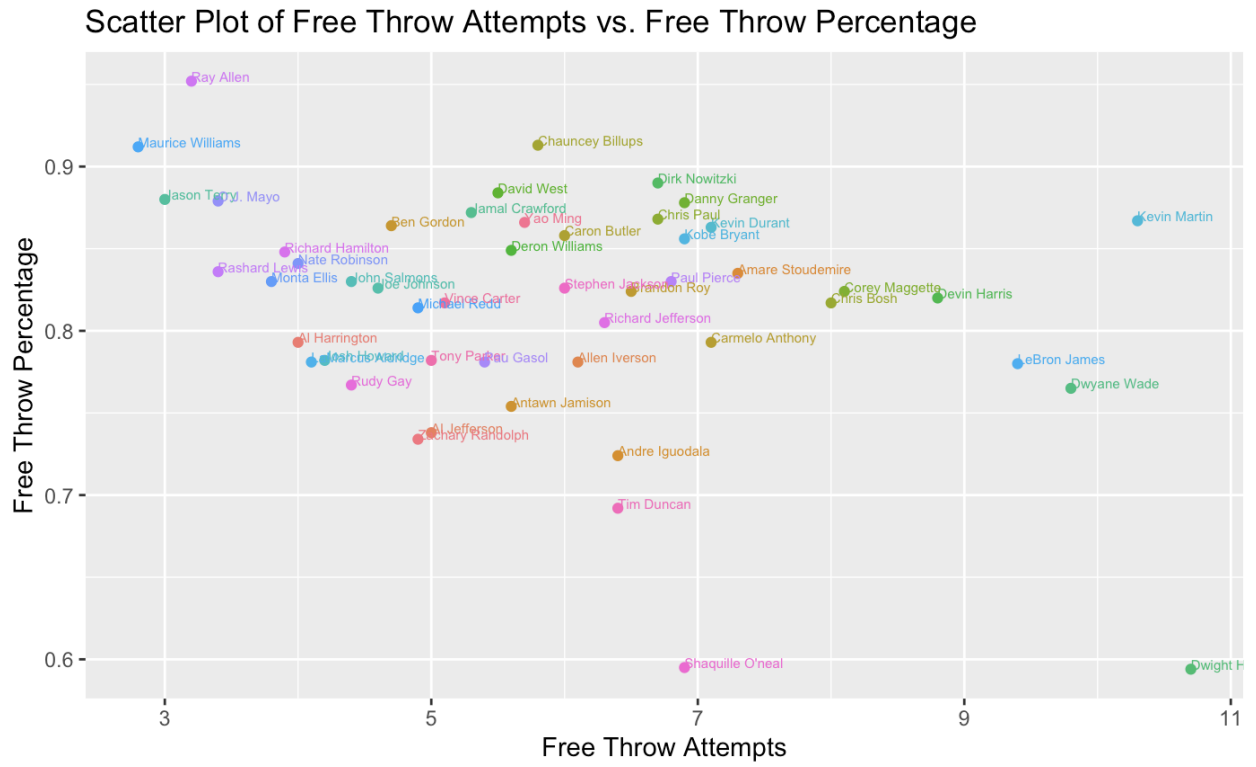


Figure 1d. The scatter plot above was created using the "ggplot" function and the x-axis represents the Free Throw Attempts and the y-axis represents the Free Throw Percentage. By using this plot, we are able to visualize the relationship between Free Throw Attempts and Free Throw Percentages for each player. With this in mind, we will get a better understanding of which players are able to convert the free throw attempts into succesful scoring throws. For example, when looking at the plot, it seems like Kevin Martin has thrown more free throws than the rest and has roughly ~86% success rate.

Name <chr>	success_rate <dbl>	free_throws_thrown <dbl>
Dwight Howard	0.594	10.7
Kevin Martin	0.867	10.3
Dwyane Wade	0.765	9.8
LeBron James	0.780	9.4
Devin Harris	0.820	8.8
Corey Maggette	0.824	8.1
Chris Bosh	0.817	8.0
Amare Stoudemire	0.835	7.3
Kevin Durant	0.863	7.1
Carmelo Anthony	0.793	7.1

Figure 1e. The data is organized with the highest amount of three throws first. Some of the players listed on here are also in the top point scoring list as well. This is another way to understand the players skills. This will be able to better understand the percentage and the number of free throws attempted.

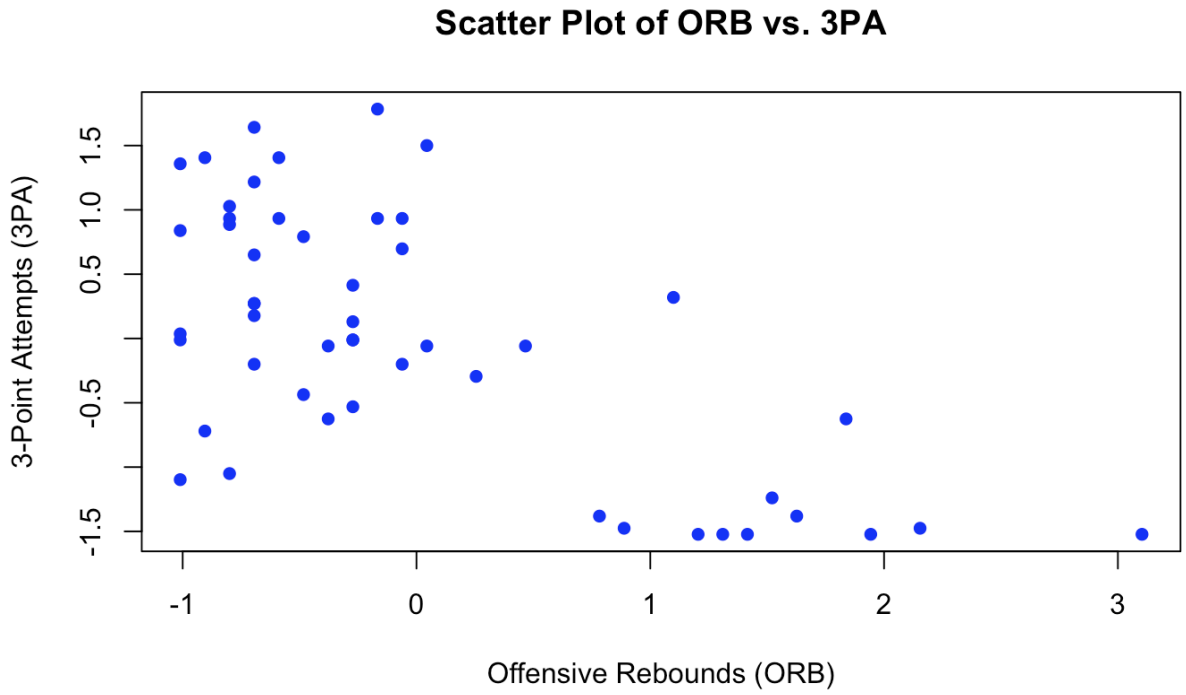


Figure 1f. The scatterplot above shows the Offensive Rebounds versus 3-Point Attempts which from the correlation matrix shows that they are strongly negatively correlated. This means that there are played that will excel in the 3PA and not excel in the ORB metric. In this case we can get a visualization of that in a scatterplot. We can see two groups of players who are good at 3PA's mostly and the other group more successful at ORB's.

Could you have predicted the successes and failures of some of the players, based on analyses of the data ? Maybe you could be a talent scout for an NBA team?

Based on the analysis performed above, it does seem apparent that there are players who did excel in certain categories than others. Based on the graph of the time played per game, it shows many players played for about ~35-40 minutes per game, and the players that stood out scoring points are Kobe Bryant, Dwyane Wade, LeBron James, Kevin Durrant, and Dirk Nowitzki. Bringing out outside knowledge to this assignment, many people are aware of these players present day especially like Kobe Bryant, Dwyane Wade, LeBron James, Kevin Durrant. These players have continuously excelled throughout their careers. Another interesting point is that it seems like there are players that are good at one type of feature and not so good at others. So in order to make a well rounded team, it seems like you would want to choose players that excel at different features within the game. One example is shown in a scatterplot above is the 3PA vs ORB. It is clear that there are players that are better in the 3PA than ORB. So it would be valuable to extract the players that are best performing in those two categories and have them on the same team in order to achieve a well balanced team.

Project 2.
==========

1) Use PCA to reduce dimensions. How many components do you need to keep to reproduce the digits reasonably well ? what is your final matrix ?

The first step is to read in the data, perform PCA, and get an idea of how the principal components explain the variance in the data set. Pixels that contained only 0s were not included in PCA.

Figure 2.1: Percentage of variance explained by principal components

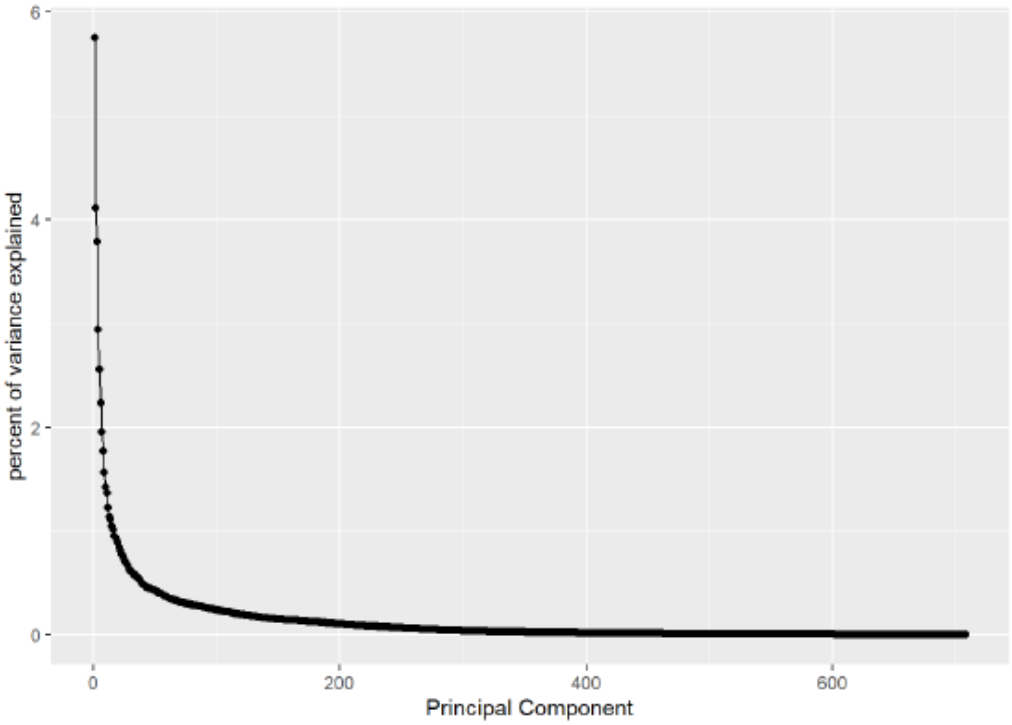


Figure 2.1 shows how much variance each principal component explains. There is a high number of PCs that contribute significantly (>1%) to the amount variance explained. Next, the loadings of each pixel on the PCs will be explored to see which pixels are contributing most to the PCs. These pixels will be important to keep if dimensions are to be reduced.

Figure 2.2: Density of Cumulative Loadings for Principal Components 2-10

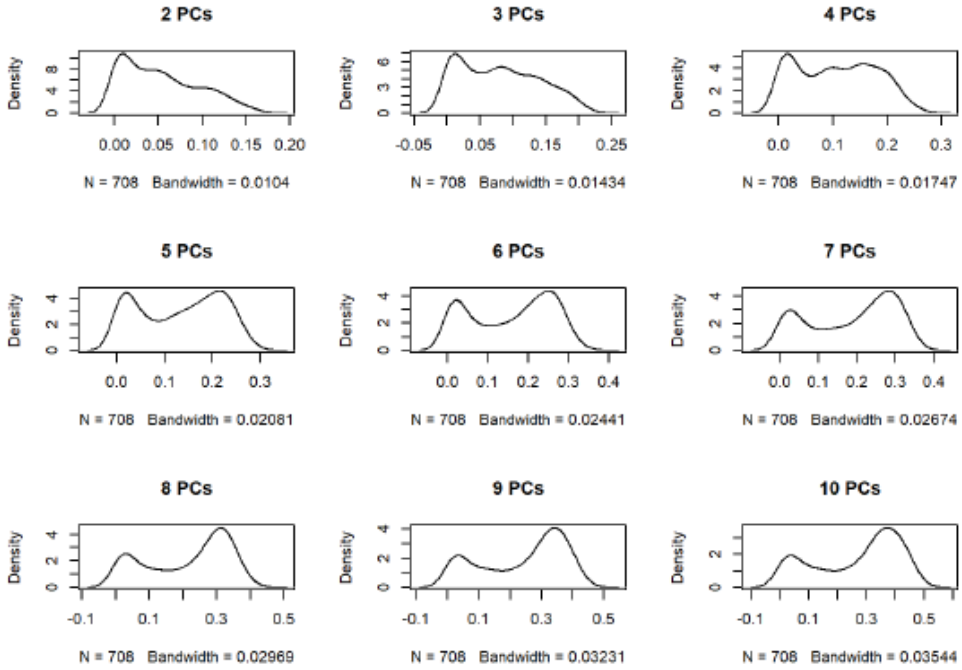
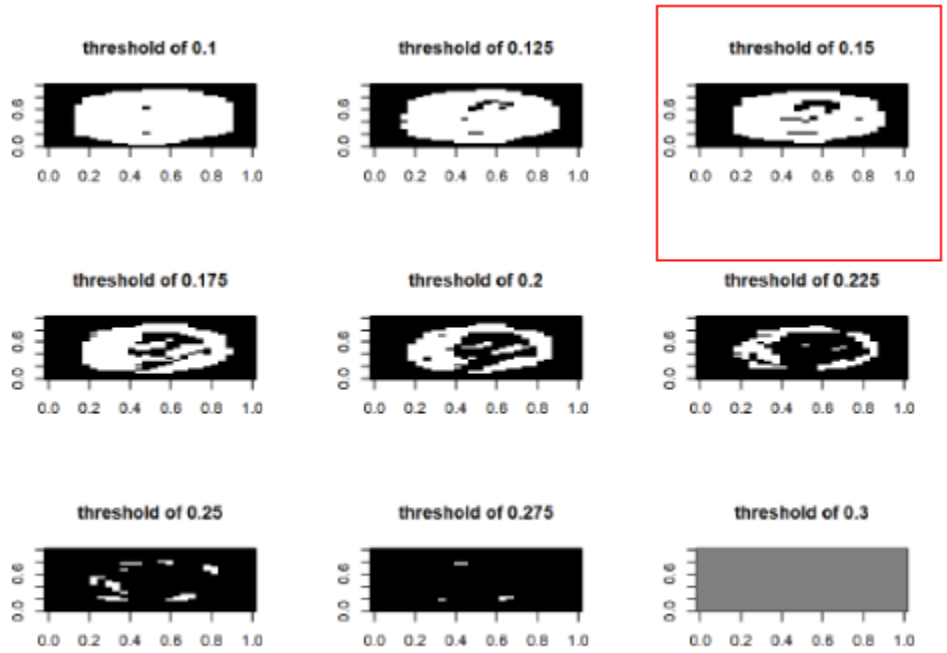


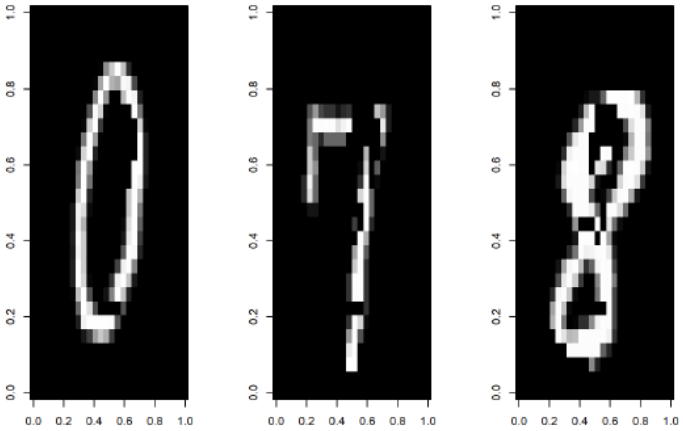
Figure 2.2 displays the density of cumulative loadings of each pixel for PCs 2-10. At 5 PCs, we see the formation of two distinct peaks. This could indicate a separation between pixels that contribute significantly to the top PCs. Next, the minimum cumulative loadings threshold for keeping a pixel in the image will be explored using the top 5 PCs.

Figure 2.3: The Effect of Increasing the Cumulative Loading Threshold for Pixel Inclusion



In Figure 2.3, if a pixel is above the threshold for cumulative loadings, it is colored white. This allows for the quick identification of included pixels. As the threshold increases, the less significant pixels begin to drop out. When using 5 PCs, the threshold of 0.15 total loadings gives a result that eliminates most pixels, but should still allow for identification of the digits. The pixels colored white when the threshold is set to 0.15 cumulative loadings for 5 PCs will constitute the final matrix.

Figure 2.4: Test Digits Using the Reduced Matrix

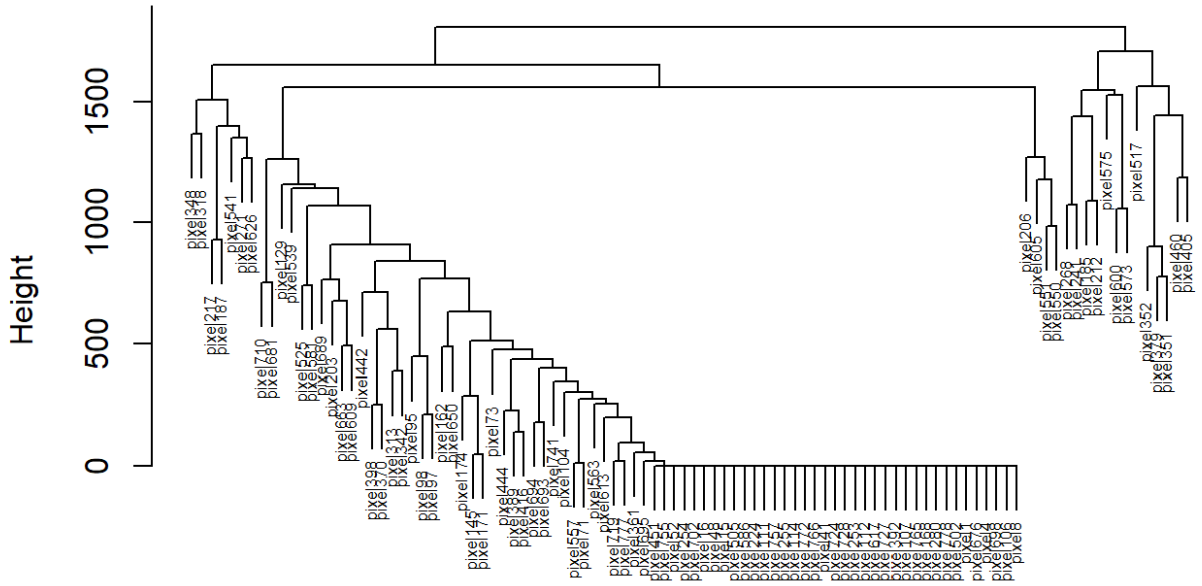


The numbers in Figure 2.4 contain only pixels included in the final matrix. This matrix contains only 343 pixels and the numbers are still recognizable. This shows that PCA can be used to reduce data down to only the most important dimensions. In this case, I used 5 PCs with a threshold of 0.15 total loadings to isolate the most relevant pixels and was left with 343.

2) Draw a tree of the pixels, and see if you can explain the results based on geometry of the pixels (how far apart are they in the 2-d space).
Try to Explain the PCA results in light of this.

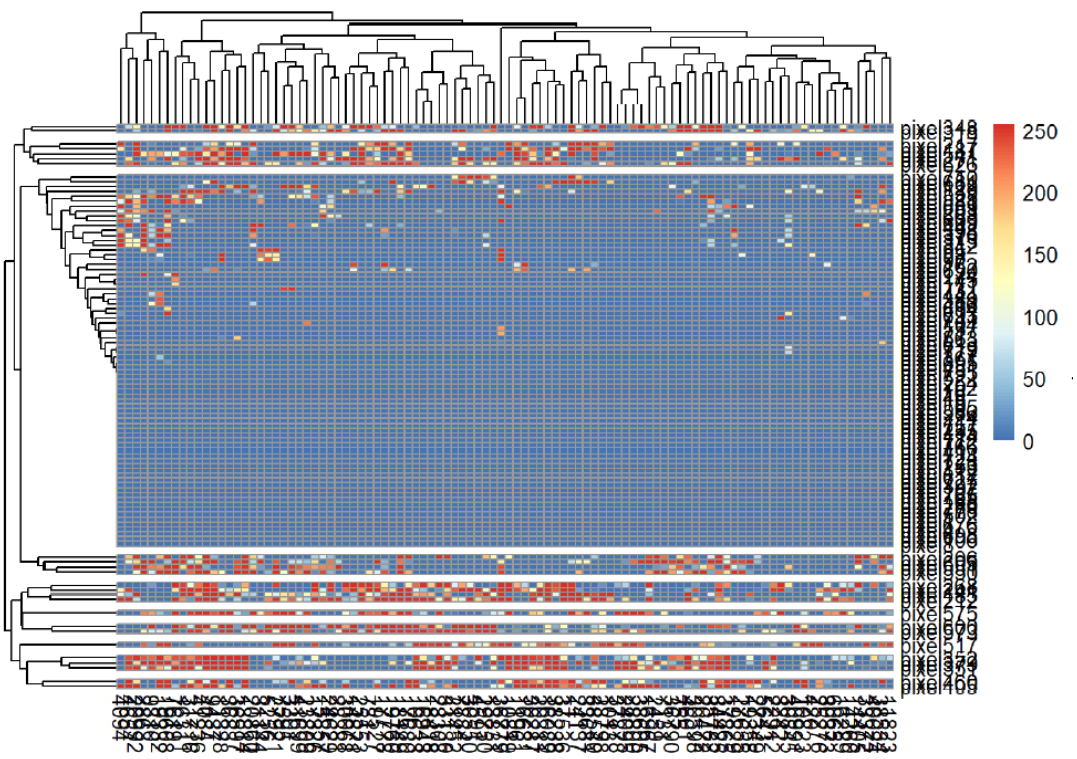
To obtain a readable tree, pixels were randomly sampled from the data set.

Figure 2.5: Cluster Dendrogram of Randomly Sampled Pixels



Some meaning can be derived from this dendrogram. The pixels that are on the edge of the image all consistently have low values, so they have short distances in the tree relative to each other. Other pixels that are in the center of the image have more variability and have a further distance in the tree relative to one another. A heat map will better reveal this

Figure 2.6: Heatmap of Randomly Sampled Pixels and Digits



The heat map with the rows cut further shows that the pixels that are variable tend to be further away from pixels that are consistently low or high. This can be related back to the 2D space of the image and the PCA results through the experiment in question 1 that dealt with increasing loading thresholds.

The order of pixels that are lost as the threshold for cumulative loading contribution is increased reveals which pixels contribute the most to the principal components, see figure 2.3. The pixels around the outside of the image are consistently 0, so they do not contribute much to the PCs. Then as the threshold increases, pixels in the middle of the image are removed. This is because these pixels often contain portions of the digits, so they do not contribute the most variance. It is the pixels that sometimes contain digits that are the most variable, so these pixels contribute the most variance and subsequently contribute the most to the PCs. As the tree and heatmap suggest as well, these are the pixels that are not in the middle, but not all the way on the outside.

3) Can you use some of the tools you have learnt to build a classifier, so if you get a new set of pixels you can predict what is in the picture. This is a the start of a real project, but you don't have all the tools (such as neural networks) which might be more suited for this task. Split your dataset into two (a training set and a test set), build your classifier and figure out how well it does with the test data in predicting the digits. Define the sensitivity and specificity of your classifier. How well does it recognize your own handwriting (make sure your handwriting is not in the training set)

I used two methods and created two classifiers. In the first method, the average value for each pixel for a given digit is calculated using the training set. Then the euclidean distances of the test numbers are calculated relative to each averaged digit (0-9). Then the test number is assigned to a class based on the shortest distance. This method was used to classify my handwritten digits as well. Please see the rMarkdown for the commented code. The results are below.

Figure 2.7: Classifier 1 Results

```
## [1] "success rate comparing test to training set using Euclidean dist = 79 %"
```

```
sensitivity = t(TP/(TP + FN))
specificity = t(TN/(TN+FP))
colnames(sensitivity) = nums
colnames(specificity) = nums
print(paste('average Sensitivity',mean(sensitivity)))
```

```
## [1] "average Sensitivity 0.789191919191919"
```

```
print(paste('average Specificity',mean(specificity)))
```

```
## [1] "average Specificity 0.971716725634392"
```

```
print(sensitivity)
```

```
##       0         1         2         3   4   5 6   7   8         9
## [1,] 0.9 0.9090909 0.8888889 0.6666667 0.9 0.5 1 0.8 0.6 0.7272727
```

```
print(specificity)
```

```
##             0        1 2         3         4         5         6         7 8
## [1,] 0.9859155 0.971831 1 0.9358974 0.9459459 0.9736842 0.971831 0.9726027 1
##             9
## [1,] 0.9594595
```

```
## [1] "success rate comparing my digits to training set using Euclidean dist = 80 %"
```

Using this method, I obtain 79% accuracy for the test set. The average sensitivity for each number is around 79%. My model performs the best with 6's and struggles the most with 5's. The average specificity for my model is 97%. The most common false positive was for number 3.

I attempt another method for classifying the test set. In this method, I find the average loadings for each pixel of each digit in n dimensional PC space where n is the number of PCs that explain 95% of the variance. Then I perform the same euclidean distance measurement for classification. The results are below.

Figure 2.8: Classifier 2 Results

```
## [1] "success rate comparing test to training set using Euclidean dist in PC space = 85 %"
```

```
sensitivity = t(TP/(TP + FN))
specificity = t(TN/(TN+FP))
colnames(sensitivity) = nums
colnames(specificity) = nums
print(paste('average Sensitivity',mean(sensitivity)))
```

```
## [1] "average Sensitivity 0.845555555555556"
```

```
print(paste('average Specificity',mean(specificity)))
```

```
## [1] "average Specificity 0.980973926135136"
```

```
print(sensitivity)
```

```
##       0 1         2         3 4   5   6   7   8 9
## [1,] 0.9 1 0.8888889 0.6666667 1 0.7 0.9 0.8 0.6 1
```

```
print(specificity)
```

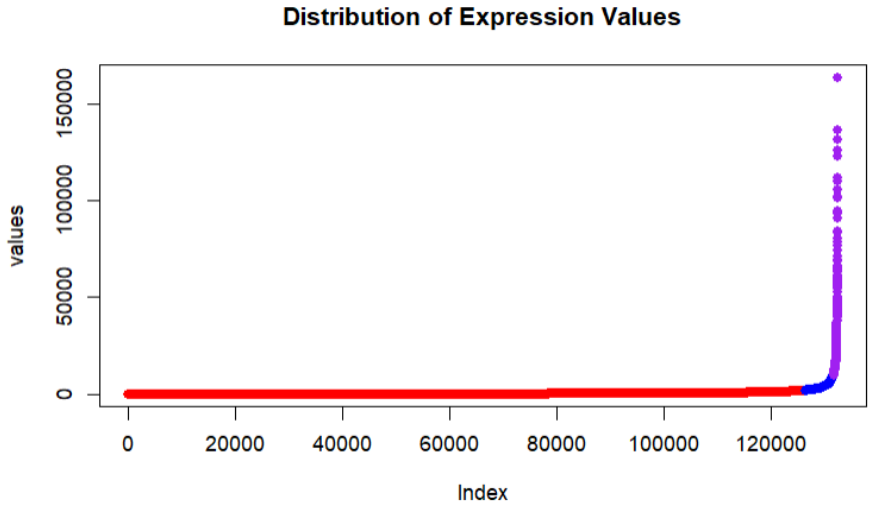
```
##             0         1         2         3         4         5         6
## [1,] 0.9620253 0.9866667 0.9871795 0.9518072 0.9868421 0.9873418 0.987013
##               7 8         9
## [1,] 0.9871795 1 0.9736842
```

The first finding of this method is that it takes 320 principal components to cover 95% of the variance. The euclidean distances between the test digits and the averaged training distances was then calculated in 320 dimensional PC space. Again, the digit that was closest to the test number was used to identify the digit. This method performed slightly better than the previous model. This method resulted in 85% accuracy. The average sensitivity and specificity are 84.5% and 98% respectively. This model correctly identified all 1's, 4's, and 9's and struggled the most with 8's. This model was also least sensitive to 3's.

==========
Project 3.
==========

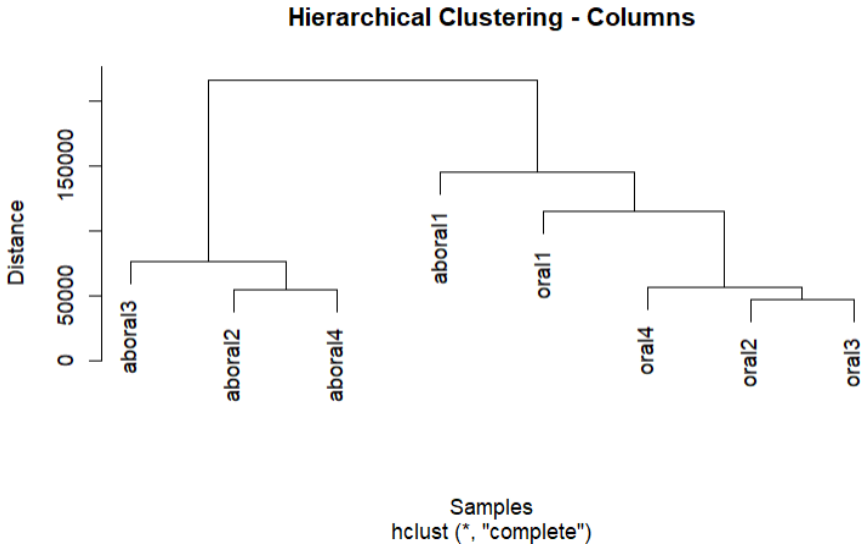
1) Build hierarchical trees based on the columns and for the rows (exclude rows that are "low" expression)

Figure 3.1: Expression Distribution of Raw Data



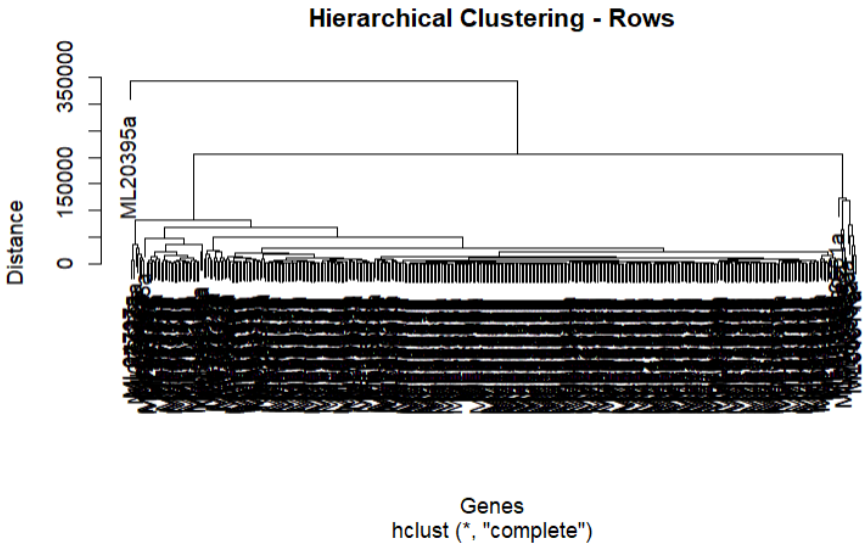
First, I determined what would be considered “low” expression by plotting the overall distribution of expression for all values, illustrated in the figure below. We see that the y-axis corresponds to the expression of each value, where we have a significant number of “low” expression values relative to the rest. I wanted to perform a similar analysis of this expression to what was done on the midterm, but instead of using the average expression of each row, I broke it down further to get a more precise representation of the overall expression in the dataset. While the trend is similar, I chose not to scale the data to avoid issues in analysis that will be discussed later on (Figure__). The threshold for low expression was set here at a cutoff of 2000 (red). This cutoff left us with a similar group of 392 genes to the midterm analysis.

Figure 3.2: Hierarchical Clustering of Columns



This hierarchical clustering of columns shows the relationship between oral2, 3, and 4, and how aboral1 shares a closer relationship to oral1 than aboral2, 3 and 4.

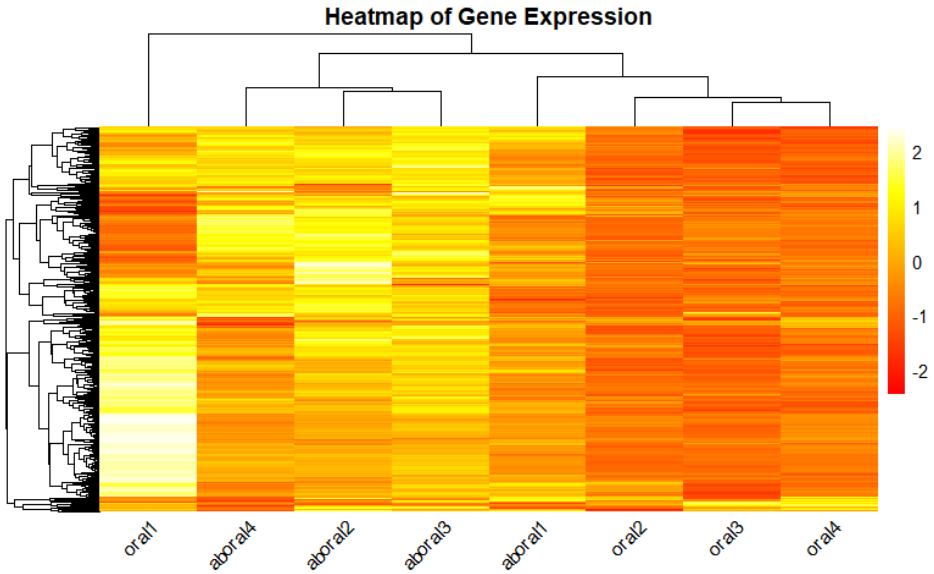
Figure 3.3: Hierarchical Clustering of Rows



This hierarchical clustering of rows shows the clustering of gene names based on expression levels, excluding low expression genes. We can see here that one gene (ML20395a) stands out from the rest in the tree. While many are difficult to pick out individually, we can observe the general trend of the genes, as they initially diverge into two major groups, suggesting a distinct pattern in gene expression. There are several genes in the left branch that have short or no distances between each other, indicating similarity in gene expression. The right branch has several nodes, or points of divergence, suggesting more variation in expression among these genes.

2) Draw a heat map of the expression data

Figure 3.4: Heat Map of Gene Expression



This heatmap of gene expression shows the distribution of expression across all samples. We can see that the oral 2, 3, 4 have lower expression, while the oral1 sample has the highest expression. The hierarchical tree at the top of the heat map (same as fig 3.3) illustrates how these differences in expression cause the divergence of oral1 from the rest of the oral samples in the tree. Similarly, the expression observed in aboral1 is generally lower than in the other three aboral samples as well, supporting it's divergence from the group as it shares more similarities in expression levels to the oral majority.

3) a) which are the most significantly changing genes in this dataset?

Figure 3.5: Most Significantly Changing Genes

```
log2 fold change (MLE): condition oral vs aboral
Wald test p-value: condition oral vs aboral
DataFrame with 10 rows and 6 columns
           baseMean log2FoldChange     lfcSE      stat      pvalue        padj
          <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
ML20758a    4405.03      -0.877224 0.0927147  -9.46154 3.03428e-21 1.18944e-18
ML073224a   9947.09      -2.279023 0.3032417  -7.51553 5.66790e-14 1.11091e-11
ML000314a   3498.83      -0.719653 0.1056977  -6.80860 9.85557e-12 1.28779e-09
ML034336a  32803.41      -2.474296 0.4211059  -5.87571 4.21038e-09 4.12618e-07
ML11032a   13918.96       2.000575 0.3542926   5.64667 1.63581e-08 1.28248e-06
ML306119a  16647.73       1.326031 0.2373890   5.58590 2.32497e-08 1.51898e-06
ML034337a  29101.71      -2.366997 0.4260080  -5.55623 2.75670e-08 1.54375e-06
ML04463a    6691.95      -1.790874 0.3311845  -5.40748 6.39172e-08 3.13194e-06
ML23836a    5613.81      -1.367709 0.2566911  -5.32823 9.91734e-08 4.31955e-06
ML047911a   3680.66       0.651533 0.1319643   4.93719 7.92565e-07 3.10685e-05
```

Figure 3.5 shows the results for the DESeq analysis on the dataset excluding the low expression genes. To perform this analysis, a DESeqDataSetFromMatrix was created to use the DESeq command and obtain the results, which were ordered by adjusted p-value. While log2foldchange is useful in performing analysis for directionality (upregulated/downregulated) of expression, adjusted p-value is more useful here to highlight the statistical significance of differential expression, where our lowest p-values indicate the highest confidence. The top 10 most significantly changing genes are: ML20758a, ML073224a, ML000314a, ML034336a, ML11032a, ML306119a, ML034337a, ML04463a, ML23836a, ML047911a.

b) which genes are most consistently highly expressed in these datasets they are the "house-keeping" genes

Figure 3.6: House-Keeping Genes

Gene <chr>	CV <dbl>
ML12239a	0.06226391
ML209114a	0.07771275
ML017310a	0.08128727
ML082318a	0.09133758
ML210018a	0.09603464
ML096826a	0.09742373
ML018039a	0.10338495
ML00017a	0.10561981
ML033237a	0.10999684
ML029520a	0.11422118

The housekeeping genes were determined by finding the coefficient variation (CV) of the DESeqDataSetFromMatrix data. The genes were then ordered by lowest CV. The CV is the ratio of standard deviation over the mean, where higher CV indicates higher variability and lower CV indicates more stability. Genes with low CV are more stable and have consistent gene expression, and are therefore our “house-keeping genes”. This is where the issue with scaled data came into play, the resulting gene list was the same, but the data had to be rounded to integers to be applied to the DESeq analysis, which resulted in many CV being 0.0. Because data is normalized in DESeq, we chose to keep the raw filtered data, as the results were the same and analysis could more definitively be performed here. The top 10 house-keeping genes are: ML12239a, ML209114a, ML017310a, ML082318a, ML210018a, ML018039a, ML018039a, ML00017a, ML033237a, ML029520a.

c) How consistent are these results with the analysis you did in the midterm project ?

I ran some code to compare the results from the midterm analysis gene lists to the lists from 3a and 3b. We found that there was one common gene between the top 5 expressed genes in the midterm data vs the housekeeping genes from the final analysis. This gene is ML00017a. I did some adjusting of the threshold to filter out the low expression genes, and we found that when set lower (250) on unscaled data, there were four genes in common between the midterm least variable genes and housekeeping genes, which makes sense since they are similar criteria. However, when we scale the data and set it to the same threshold (1) as in the midterm, we get the same single common gene between the top 5 expressed genes in the midterm and the final consistent expression (ML00017a). This was an indication to me that although the adjustment to our code to scale the data required rounding the data to perform DESeq, we were able to produce the same results from setting the threshold at the same position on the distribution graph with unscaled data.

d) What else can you say about the data in terms of consistency, and the results that you find from your analyses. The question is open-ended, think of this as your experiment, you need to write a paper based on this data so you have to figure out what kind of "story" you can tell based on this.

In the midterm project, we were tasked with filtering the data based on expression levels (low, medium, high) which we did by plotting the average distribution in each row and visually setting a cutoff for low, medium, and high expression. Before filtering the data, we found the 5

closest pairs of genes based on correlation. Additionally, we found the top 5 up regulated (ML000111a, ML000112a, ML000113a, ML000114a, and ML000115a) and down regulated genes (ML50721a, ML50771a, ML50791a, ML50792a, and ML50851a). We also determined the top 5 genes in all columns based on average expression using the same raw data. The resulting top 5 genes were: ML20395a, ML26358a, ML46651a, ML020045a, and ML00017a.

Comparatively, in the analysis for the final here, we were able to filter out the low expression genes using a similar method to the midterm. We plotted the distribution of the average expression by row, and visually set a cutoff point for low expression threshold (2000). Because this cutoff was based on the average expression in 8 columns, the mean was taken (250) and used to filter out rows possessing values matching that value. From here, we took our filtered data excluding genes with low expression, and determined the most significantly changing genes in the dataset by using the DESeq package, ordering results by adjusted p-value to observe genes with the most statistically significant differences.

We then found the top 10 "housekeeping" genes by determining which genes were most consistently highly expressed in the dataset. This was achieved by using the DESeq data and running a coefficient variance (CV) check per gene, the results were sorted by lowest CV. The coefficient variance is the ratio of standard deviation to mean, therefore, genes with lower CV mean higher stability and more consistent gene expression. These genes are: ML084210a, ML03197a, ML13052a, ML122316a, ML12239a, ML00121a, ML14881a, ML00747a, ML07089a, and ML210020a.

Therefore, although the analysis approach to determine these significant genes differed between the midterm and the final, we can look at the resulting genes to see which stood out. In the code for 3c, we can see that one (ML00017a) common gene was found between midterm analysis results and final analysis results. However, our approaches differed and the genes we produced were based on different parameters (ie. average expression vs most frequent genes).

e) what is the most interesting pathway or gene that is responding in this study?

Two genes stood out to me in this analysis, that is the outlier gene in the hierarchical tree, ML20395a, and the gene that was identified as common between midterm and final analysis data, ML00017a.

After looking up gene functions, we found that ML00017a is assocated with Elongation factor 2 OS in multiple other species. It has GO term association with translation, translational elongation, GTP catabolic process, and other functions. The elongation factor 2 protein can be found in humans and functions to catalyze GTP-dependent ribosomal translocation during translation elongation. Source for this human protein can be found here: https://www.uniprot.org/uniprotkb/P13639/entry

After looking up gene functions, we found that ML20395a is associated with the Elongation factor 1-alpha OS in several other species, and has GO term associations with translation, translational modification, and more.