MSIS 5223: Tutorial 5 – Data Reduction and Exploration in R

This exercise covers Chapter 25 in Crawley's book *The R Book*.

Instructions

The purpose of this tutorial is to help you become familiar with using R. A difficulty of any project is narrowing down the data to be used. While narrowing the scope of a project can in turn narrow the data, this often doesn't reduce the data enough. Additionally, an individual may not be familiar enough with a dataset to choose the correct variables for a given analysis.

This exercise will go over three main analyses that provide a way to reduce the number of columns in a given dataset. Note, these techniques are not designed to reduce the number of rows in a dataset, necessarily. These techniques are principal components analysis (PCA), factor analysis (FA), and cluster analysis. These are all analyses that fall under the umbrella term *multivariate statistics*, because they deal with multiple variables.

1. Principal Components Analysis (PCA)

Principal components analysis, or PCA, is a technique that creates linear combinations of your columns. In other words, it determines which columns are similar and lumps them together. This process leads to the creation of new columns. If you have 12 columns of data, for example, it may determine that in reality you have only 5 unique columns of data. While this is oversimplified, it helps illustrate what PCA does.

In most cases, the results of PCA are not used. While PCA creates brand new columns of data, it is very hard to interpret what those columns represent. For example, assume you have a dataset on smartphone sales and two of the variables include *annual income* and *occupation*. PCA combines these into a single column based on their similarity. What does this new column represent? Additionally, the data points for each of those columns is now transformed into a linear combination creating new data points for a new column. What are these new data points? Can you interpret them?

The simple answer is "no." The new columns from a PCA can be used effectively within certain types of analyses, but that usage is beyond the scope of this class. Why even talk about it then? While you won't be using the newly created columns from PCA,

the results from PCA can inform your decision in selecting columns of data; i.e. which columns are considered redundant. This process provides a methodology in finding the number of redundant columns of data and removing them. This process is fairly easy to perform once you have learned how to perform it. It should be noted, though, that it must be performed in conjunction with factor analysis (FA) (see next section).

When performing a PCA followed by a FA you should split your data into two separate halves. You should never use the same sample of data for both. This is similar to creating subsamples for training, testing, and validation. Each one should have different variance from the other. The best approach is to split your data into 50%.

For this example the dataset datareduction.csv is used. This dataset resulted from an assessment of employees' adoption of a new system within an organization. Many different variables exist in the dataset, three of which will be used in this example:

- Perceived Usefulness – How useful an individual thinks the new technology is in performing job functions
- Perceived Ease of Use – How easy-to-use the new system appears to be
- Intention to Use – The extent to which an individual believes he/she will use the new system to perform his/her job

Each of these three variables are measured on a 7-point scale. A 1 represents strongly disagree while a 7 represents strongly agree. *Perceived Usefulness* (PU) and *Perceived Ease of Use* (PEOU) were measured with 6 different questions each while *Intention to Use* was measured with 3 different questions. While there are many other variables in the dataset (e.g. race, gender, religion), the three main measures of technology usage are the focus; thus, the other variables will be ignored for now.

Begin by separating out the columns of data that pertain to the analysis. This means we will need to create a new dataframe that only contains the measures of PU, PEOU, and Intention.

```
> #### PU, PEOU, and Intention to Use System
> reduction_data.pca = reduction_data[c("peruse01","peruse02","peruse03",
+ "peruse04","peruse05","peruse06","pereou01","pereou02","pereou03",
+ "pereou04","pereou05","pereou06","intent01","intent02","intent03")]
>
```

Figure 1 Creating a New Dataframe

A new dataframe is created called reduction_data.pca. This only contains the columns of data related to PU, PEOU, and Intention. The next step is to run the PCA and assess the eigenvalues, or the variance calculated from the PCA.

```
> pcamodel_reduc = princomp(reduction_data.pca,cor=FALSE)
> pcamodel_reduc$sdev^2
   Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7    Comp.8
9.1916085 3.7600060 2.2489006 0.7182259 0.5385546 0.4962996 0.4322337 0.3767315
   Comp.9   Comp.10   Comp.11   Comp.12   Comp.13   Comp.14   Comp.15
0.3229324 0.2994056 0.2738115 0.2599479 0.1785493 0.1581499 0.1331775
```

Figure 2 PCA Eigenvalues

Look at Figure 2, the results of the PCA. Notice that the output has created 15 components. You will always have the same number of components as the number of columns you put into the analysis. Recall that PU has 6 items, PEOU has 6 items, and Intention has 3. This results in 15 columns of data used as input. You may be confused at this moment, because you were told that the purpose of PCA was to reduce the number of columns; the output has produced 15 components. It appears that nothing was removed.

This brings up the next step of the analysis. Notice that each component has an associated value; the first component has the greatest value, 3.032 for Component 1, and the last component has the smallest value, 0.365 for Component 15. This is where we start to reduce this dataset.

The way this PCA is assessed is any value greater than 1.0 is retained, while anything less than 1.0 is thrown out. The first three components all have values greater than 1.0. Starting with Component 4, however, the values are all smaller than 1.0. What this assessment tells us, is that in reality we are only dealing with 3 variables, not 15.

We can run a Scree Plot to confirm the findings from the PCA. Below is the R code for a Scree Plot along with the resulting plot itself.

```
pcamodel_reduc = princomp(reduction_data.pca, cor=FALSE)
plot(pcamodel_reduc, main="Screeplot of PU, PEOU, Intention")
```

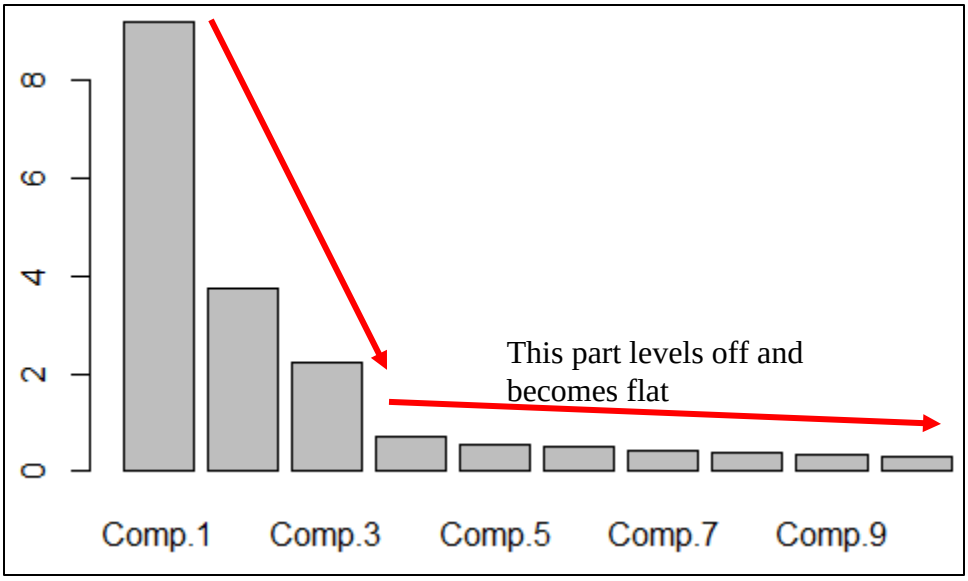

Figure 3 Scree Plot of PU, PEOU, and Intention

Scree plots are more subjective than assessing eigenvalues. The purpose of using the scree plot is to determine how many components, or columns of data, you are truly dealing with. The way to read a scree plot is determining where the plot levels off and becomes flat; anything prior to that leveling off is a component that remains.

Look at Figure 3. You can see that the first 3 components are not flat. Starting with Component 4 you can see the plot is level and flat. This agrees with the PCA results given on the previous page. Out of 15 variables, only 3 should be used for the analysis. This means that some combination of all 15 variables should yield just 3. The next logical question is, “Which variables are combined or removed to result in only 3?”

At this point the PCA cannot yield any more information. Unfortunately, the PCA can only suggest the number of columns to reduce. To determine which variables will be removed (or possibly combined with others) a factor analysis must be performed.

2. Factor Analysis (FA)

Just like PCA, a factor analysis helps reduce the variables within a given dataset. In the previous section we looked at three variables: Perceived Usefulness, perceived ease of use, and intention. The PCA suggested that out of 15 total measures, we truly only need 3. My hunch is that each of those three variables will only use one measure.

A factor analysis can be performed using different types of rotations. Some rotations assume the data has no relationship and forces them to be orthogonal (i.e. not correlated), like a promax rotation; others assume a relationship such as varimax (i.e. correlated). While this is an important concept to know, it is something that is beyond the scope of this class and no further detail or discussion will be given. This is mentioned here so that you are aware of its existence. For the sake of this exercise, only the varimax rotation will be used.

To perform a FA, use the factor analysis function provided by R. The following code is an analysis for PU, PEOU, and Intention. Note that the list of measures begins with a tilde, the ~ symbol; do not leave it off. There are some important things to notice about this function after the list of measures. Notice the argument "factors=3". You have to specify the exact number of factors (similar idea to components in PCA) for the analysis. It may be a good idea to decrement and increment this number by 1 or 2 from the PCA result just to confirm your results. While the PCA is pretty good, it is possible that a FA may suggest a completely different number.

```
> reduction_data.FA = factanal(~peruse01+peruse02+peruse03+peruse04+peruse05+peruse06+
+ pereou01+pereou02+pereou03+pereou04+pereou05+pereou06+
+ intent01+intent02+intent03,
+ factors=3,
+ rotation="varimax",
+ scores="none",
+ data=reduction_data)
```

Figure 4 Factor Analysis for PU, PEOU, and Intention

The next argument is the rotation value. Varimax is traditionally used and should be your default choice. Next is the argument for scores. The default is none, but accepted values include regression or Bartlett. Leave it as the default value. The last argument is the data you will be using for this analysis. You should be using the same items that were used in your PCA; however, this should be a different sample. Recall that you should have split your data into two halves; one half for the PCA and the other half for the FA.

The next step is to type look at the results of the FA. Because I saved the FA into a new object, reduction_data.FA, I simply need to type in the object into R and the results will be displayed. See Figure 5 on the next page.

The output contains a lot of information. Ignore the majority of the output. The main focus of your attention should be on the factor loadings. This is what helps determine which items to remove, combine, and filter. Notice the heading of the table contains Factor1, Factor2, and Factor3. If we had specified 4 factors in the factanal() function, this table would have a fourth heading for Factor4.

These factors are similar in idea to the components in PCA. What we are doing here is forcing all 15 items in our data into 3 imaginary items.

```
> reduction_data.FA

Call:
factanal(x = ~peruse01 + peruse02 + per

Uniquenesses:
peruse01 peruse02 peruse03 peruse04 per
   0.462    0.312    0.174    0.348
pereou06 intent01 intent02 intent03
   0.344    0.044    0.219    0.060

Loadings:
         Factor1 Factor2 Factor3
peruse01  0.709   0.148   0.119
peruse02  0.790   0.156   0.200
peruse03  0.883   0.192
peruse04  0.777   0.217
peruse05  0.736   0.162   0.222
peruse06  0.669   0.179   0.164
pereou01          0.725
pereou02  0.193   0.798
pereou03  0.207   0.825   0.121
pereou04  0.291   0.662   0.187
pereou05  0.269   0.642   0.152
pereou06  0.226   0.773
intent01  0.140   0.122   0.960
intent02  0.215   0.156   0.843
intent03  0.225   0.163   0.929
```

Figure 5 Results of the FA

I have drawn some red lines to make this table easier to read. Focus on the first column, Factor1. Notice that each of the 15 items has an associated score with each factor. These can range from -1.0 to 1.0. A negative value represents an inverse relationship. The closer a value is to 0, the less of a relationship that item has with that factor. For Factor1, all of the items for PU are high. In fact, the smallest value is 0.669. Look at the values for PEOU and Intention; these are all very small with the largest at 0.291. This tells me that Factor1, our imaginary column of data, is mainly comprised of PU.

Look at Factor 2. A similar story evolves. The items with the greatest scores come from PEOU while the items for PU and Intention have extremely low values. Factor3 presents a similar scenario, with several blank values listed (these values are smaller than 0.005 and are considered insignificant). The results of this FA confirm what the PCA provided: out of a total of 15 items in this dataset, we truly are dealing with just 3.

Now that we know we truly have 3 variables, how do we actually go about reducing them? One method that is often suggested is to take the relevant items within each of the factors and average their values. For example, Factor1 is comprised of peruse01, peruse02, peruse03, peruse04, peruse05, and peruse06; the other items have scores that are too low to be relevant. Create a new, blank column in your dataset and for each row in your dataset average the values for all 6 items and name the column peruse_average. Now, instead of using 6 different items in your analysis, you have 1 single item. Do this for Factor2 and Factor3 as well.

Another alternative solution is to pick the most representative item within each factor. The representative item has the largest value. For Factor1 that would be peruse03 with 0.883; for Factor2, it is pereou03 with 0.825; Factor3 is intent01 with 0.960. In your analysis, you would only use peruse03, pereou03, and intent01; the other 12 columns of data would be completely ignored or thrown out.

It is possible to further refine your FA. It is actually recommended that any item with a value less than 0.70 should be removed from the analysis for a given factor. Looking at Factor1 in Figure 5, we can see that peruse06 has a value less than 0.70; therefore, it should be removed. For Factor2, this includes items pereou4 and pereou5; for Factor3, all Intention items are greater than 0.70, so they remain. Run the analysis one more time (see Figure 6).

```
Loadings:
         Factor1 Factor2 Factor3
peruse01 0.728   0.127   0.127
peruse02 0.802   0.211   0.119
peruse03 0.881   0.110   0.170
peruse04 0.784           0.207
peruse05 0.722   0.236   0.134
pereou01                 0.704
pereou02 0.220           0.793
pereou03 0.218   0.139   0.846
pereou06 0.237   0.107   0.757
intent01 0.133   0.963
intent02 0.202   0.849   0.136
intent03 0.219   0.935   0.140
```

Figure 6 Refining the FA

The results seen in Figure 6 present a more refined version of the FA. Prior to the removal of peruse06, pereou04, and pereou05, I would not have averaged the scores. Some of the values were below 0.70 and are borderline for inclusion. With those items removed, you can see that the loadings are much cleaner; I would feel comfortable averaging these scores for each factor.

It should be noted that if you plan on averaging scores, they should have the same scale. For example, if you perform a FA and find that *income*, *price of house*, and

occupation load on the same factor, then you need to first normalize these three variables prior to averaging their scores. Each one has a different range and mean, which results in an incorrect average value; after normalizing each one, then you can average their scores together.

3. Cluster Analysis

In addition to performing a PCA and FA, a cluster analysis can also be used to reduce your data, though it isn't similar to the process like PCA and FA. In PCA and FA, you determine which variables are similar and remove redundant measures; in clustering, you merely choose the best variable to represent grouping. Cluster analysis, then, helps determine which variables are best for apportioning group membership. Like PCA and FA, cluster analysis is an unsupervised technique; there is no target variable. As discussed in class, there are many different types of clustering algorithms. For this exercise, K-Means and agglomerative analysis will be performed using kmeansdata.txt, taxon.txt, and pgfull.txt.

Recall that cluster analysis attempts to maximize the between-cluster variance and minimizing within-cluster variance. K-means, k-medians, k-modes all are centroid-based algorithms. That is, each cluster is created based on a central point in space in which its data points surround. Hierarchical clustering, which includes agglomerative and divisive, are based on distance of objects to one another, not a centroid.

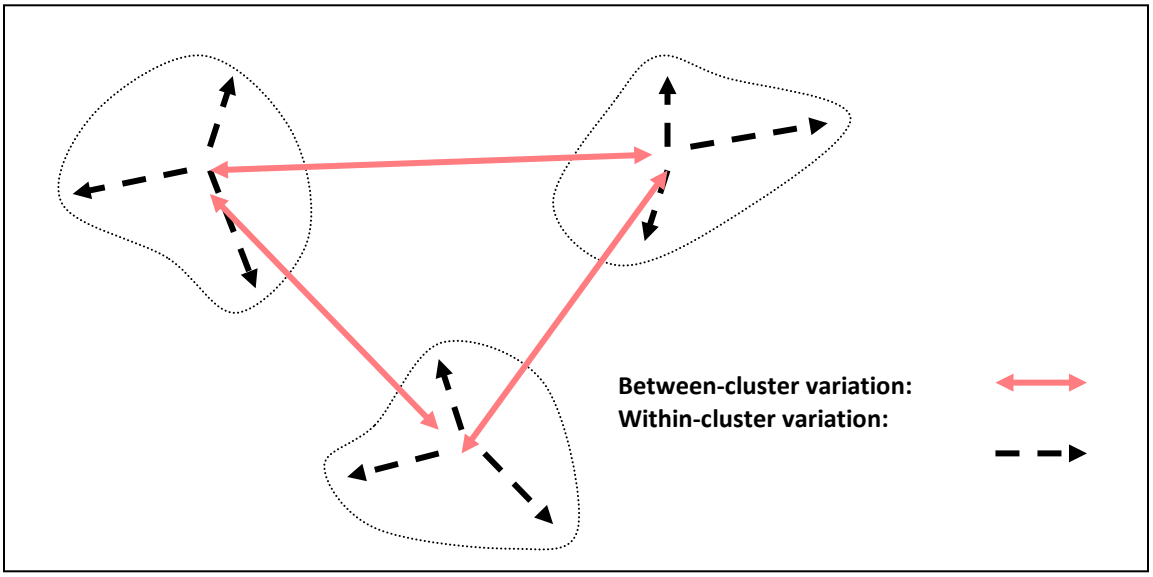

Figure 7 Variance for Cluster Analysis

The dataset kmeansdata.txt contains four variables, two of which are categorical: *group* and *grouping*. The variable *grouping* has values A, B, C, D, E, and F as shown in Figure 8. These represent natural groupings within the data. When using clustering, you would normally not have variables such as *grouping* or *group* because you are unaware of how the data group together. I am merely using this data to illustrate how clustering works.

```
> unique(kmeans_data$grouping)
[1] A B C D E F
Levels: A B C D E F
```

Figure 8 Grouping Variable Values

K-means is based on the idea that you choose the number of clusters *k* and the algorithm will determine, based on initial seeding, where those clusters are. To further understand how this works, first look at some plots based on the following R code.

```
> par(mfrow=c(1,3))
> #### Add color based on the "grouping" variable
> plot(kmeans_data$x, kmeans_data$y, pch=18, col=kmeans_data$grouping, main="Grouping Var.")
>
> #### Now, let us use the K-Means algorithm to plot the data.
> #### Restrict the data to 6 distinct groups
> km = kmeans(data.frame(kmeans_data$x, kmeans_data$y), 6)
> plot(kmeans_data$x, kmeans_data$y, col=km[[1]], main="6 KM Groups")
>
> #### Restrict it based on using 4 groups instead of 6
> km2 = kmeans(data.frame(kmeans_data$x, kmeans_data$y), 4)
> plot(kmeans_data$x, kmeans_data$y, col=km2[[1]], main="4 KM Groups")
```

Figure 9 Plots of kmeansdata.txt

The code will produce three plots, each one shown in Figure 10. The first plot isn't using any kind of clustering algorithm; the colors are based on the variable *grouping*. Again, if you were using clustering, you wouldn't have this kind of data. This is how the categorical variable *grouping* appears based on the data. If we apply a k-means algorithm on the data and specify 6 clusters, then the middle plot is given. The plot on the right is a k-means cluster analysis constrained to 4 groups.

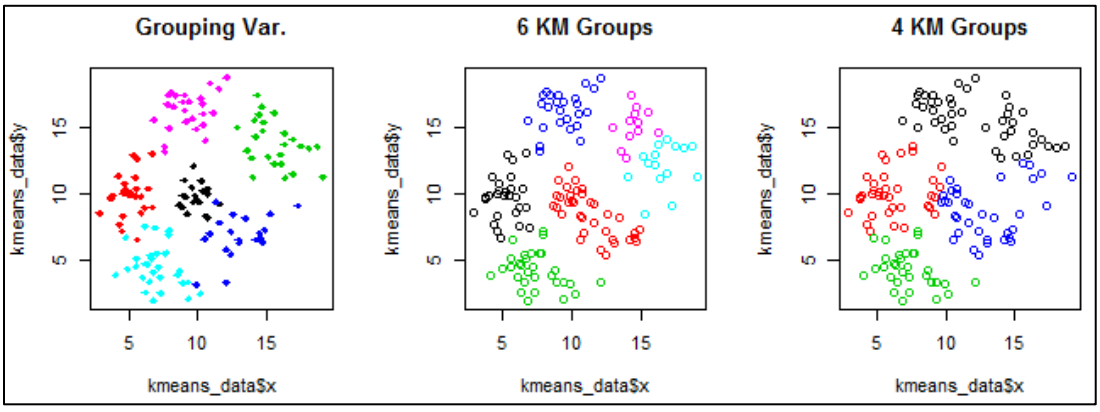

Figure 10 Plots Showing Groupings of Clusters

Notice the data points in the first plot overlap in some regions. The light-blue group overlaps with the red and navy groups. In the middle is a group with the color black, which appears to be very close with the navy-colored group. Look at the third plot, the one with four clusters. The group that was in the middle is now part of the red group; the group that was green (in the first plot) is now incorporated into one massive group (see third plot). What does this indicate? It is possible the grouping assigned by the researchers who collected the data are incorrect. It is possible that instead of having six groups, this data merely contains 4. You can see how well clustering performs with this data. The k-means algorithm was able to pick up the groups fairly well when constrained to 4 or 6 groups.

To view the misclassification of data, run the code in Figure 11 to obtain the table shown. You can see that Group A was perfectly assigned to Cluster 6. For Group B, however, one member was assigned to Cluster 1 with the rest placed in Cluster 4. Group C had the worst placement. Despite this, the clustering did a fairly good job with the overlapping data.

```
> table(km[[1]], kmeans_data$grouping)

    A  B  C  D  E  F
1   0  1  0  2 27  0
2   0  0 12  0  0  0
3   0  0 13  2  0  0
4   0 24  0  0  3  0
5   0  0  0  0  0 25
6 20  0  0 16  0  0
```

Figure 11 Misclassification of Groups

The next example for this exercise is to illustrate that clustering may not always perform well. In fact, when a decision tree is used on the data, the categorization is better than with clustering. Use the taxon.txt data. It contains 120 observed plants from 4 different islands. Specific characteristics, seven variables, for taxonomy were measured for all plants.

```
> km_tax = kmeans(taxon_data, 4)
> km_tax
```

Figure 12 K-Means Run on Taxon Data

A k-means algorithm is run on the data for 4 clusters (see Figure 12), where taxon_data is the dataframe the taxon data is saved to. The results are shown below in Figure 13. Look at the two highlighted sections of the figure, one in red and one in green. The first section contains two rows (the red rectangle); the top row is incremental starting at 1 and ending at 16 while the second row is comprised of the numbers 2 and 3 in what

appears to be a random order. The first row represents the rows of data, the observations, in the dataset. The numbers 1, 2, 3, 4 are the first, second, third, and fourth data points, respectively, from the dataset. The numbers directly below those—the 2s, and 3s—are the clusters assigned to those data points.

```
Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
  2   2   2   3   3   3   2   2   2   3   2   3   2   2   2   2
 17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32
  2   2   2   2   1   2   3   2   2   1   2   2   1   2   4   4
 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48
  1   1   4   4   4   3   4   4   1   1   4   4   3   1   4   4
 49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64
  4   4   4   3   4   4   1   4   1   4   1   4   1   3   4   3
 65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
  1   1   3   4   3   1   1   3   1   4   4   4   3   1   1   1
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96
  4   2   3   4   1   1   1   3   2   1   3   1   3   1   3   3
 97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112
  1   2   3   3   1   3   3   3   1   3   3   3   3   3   3   1
113 114 115 116 117 118 119 120
  1   3   3   3   1   3   1   1
```

Figure 13 Cluster Assignment of Taxon Data

Look closely at the second highlighted section (the green rectangle). The top row is a continuation of the ordered rows from the dataset; it begins with data point 17 and ends with 32. The row right below it represents the assigned clusters.

The data itself is sorted so that the first 30 observations belong to taxon I, the next 30 to taxon II, the next 30 to taxon III, etc. This cluster analysis was performed with 4 clusters; however, the results are not very good. Look at the first 30 records; the observations are placed in clusters 1, 2, 3, and 4, with the majority in 2 and 3. All of these observations should be just one single cluster. Obviously, clustering did not perform very well. If you run the clustering algorithm again with just 3 clusters, instead of 4, the results come out cleaner. This won't do, because we know this data contains 4 groups.

As an alternative, a hierarchical clustering technique can be performed on the data. Specifically, an agglomerative analysis. Agglomerative is an additive process. It starts out with each observation and slowly clusters them, based on Euclidean distance, until only one cluster remains. Go ahead and run the analysis and plot the results.

```
> plot(hclust(dist(taxon_data)), main="Taxonomy Cluster Analysis")
```

Figure 14 Running Agglomerative Analysis for Taxon Data

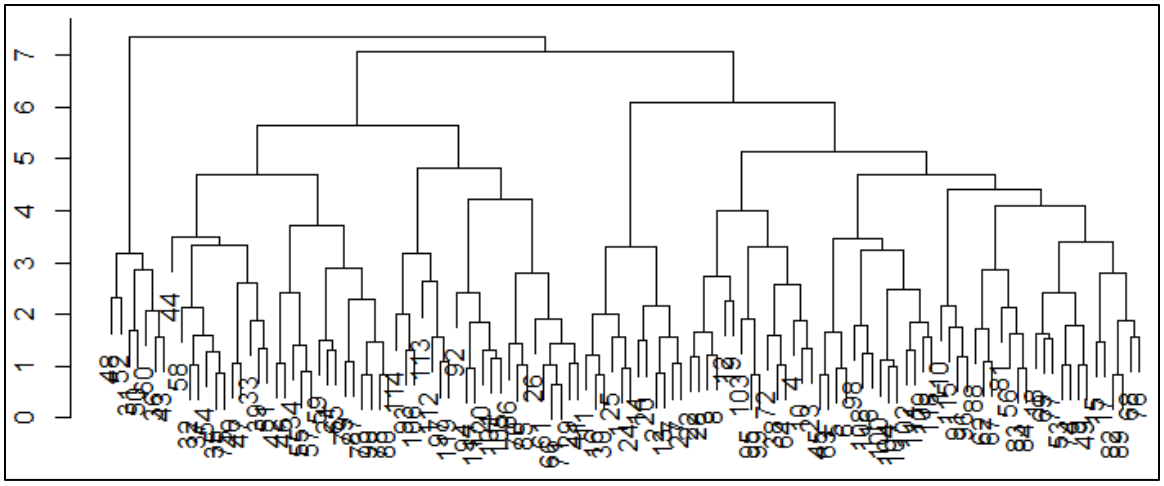

Figure 15 Plot of Cluster Results for Taxon Data

The results are not improved. No matter which break you look at, the clusters involve observations from more than one group. Clustering did not appear to perform very well with this data. This is an important lesson to learn about clustering. Clustering may not find a global optimum; in many situations it only finds the local optimum.

To help further illustrate this, see Figure 16 below. This is the result of a decision tree on the taxon data. You can see that the tree was able to partition the data into four separate groups based on *sepal*, *leaf*, and *petiole*. This is much cleaner than clustering.

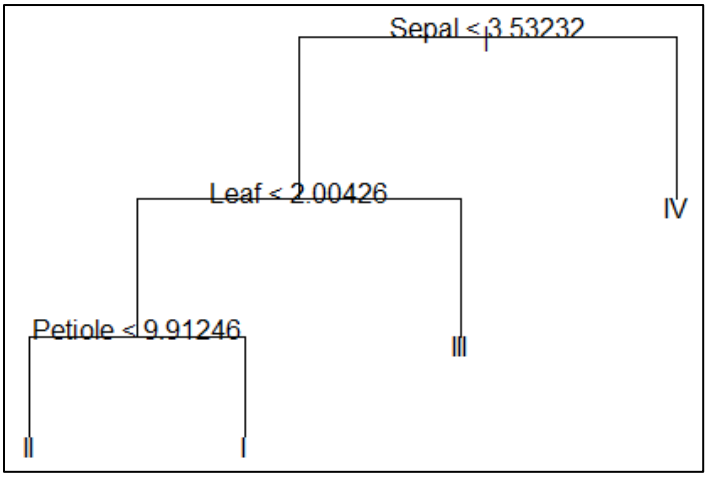

Figure 16 Decision Tree for Taxon Data

In various projects you engage in, it is often beneficial to attempt multiple types of statistical techniques. This process allows you to see where your analyses differ and agree. The temptation, typically, is to use a single process and then accept the results at face value. Don't just settle for using one type of analysis!

The last clustering example for this exercise covers hierarchical clustering. Specifically, agglomerative analysis. For this example, the data found in pgfull.txt is used. Below is the code to load it into R.

```
> #Load the Experimental Plant Plot data file
> temptable = paste(workingdirectory, "\\pgfull.txt", sep="")
> pg_data = read.table(temptable, header=T)
> pg_labels = paste(pg_data$plot, letters[pg_data$lime], sep="")
> ncol(pg_data)#59 columns of data!
[1] 59
```

Figure 17 Load the Plant Growth Data

This data provides observations on plant growth for 54 plant species on 89 plots. Calculate the Euclidean distance for all of the rows and columns. Note, only do this for the first 54 columns; the other 5 (plot, lime, species, hay, pH) are not considered for this. Once calculated, the results can be plotted.

```
> pg_dist = dist(pg_data[,1:54])
> pg_clust = hclust(pg_dist)
> plot(pg_clust, labels=pg_labels, main="Agglomerative Clustering")
```

Figure 18 Agglomerative Analysis Function

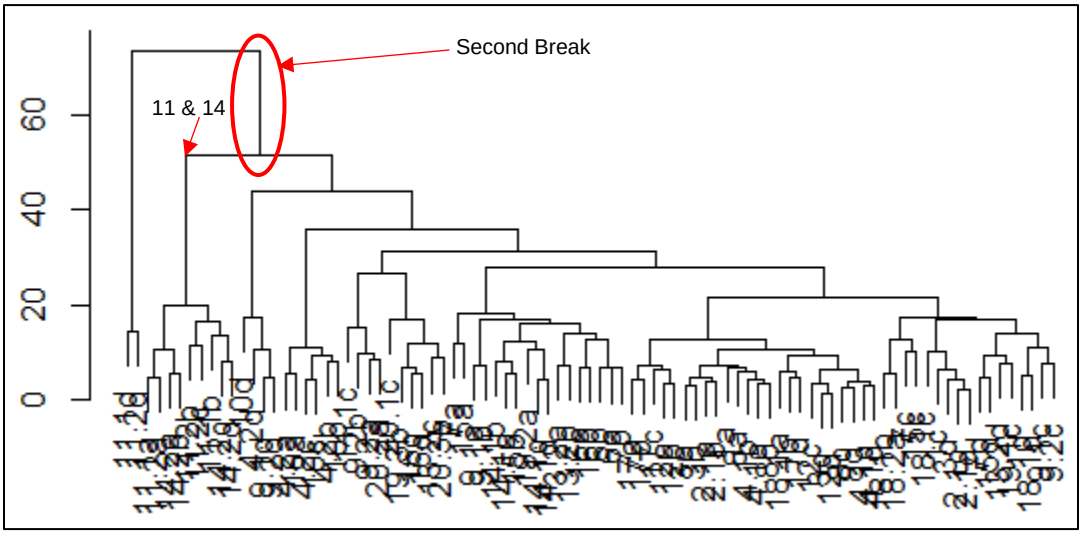

Figure 19 Hierarchical Clustering Results

Observe the second break in the clustering, the one circled in red. If you run this in R and stretch the plot as wide as your screen will allow, you will notice that the plants

on the left of the break belong to plots 11 and 14; those on the right are the rest. Plots 11 and 14 are high nitrogen plots receiving phosphorus.[1]

[1] See the following for more information on this data: Crawley et al. 2005. "Determinants of species richness in the Park Grass Experiment," *American Naturalist*, 165, pp. 348-362.