Clustering Analysis of Vietnam War Casualties

Andrew Wang

Introduction

In this report, I will be conducted a clustering analysis on a set of Vietnam War casualty data (Source: Hartigan (1975)). I will be using various data clustering techniques, such as K-means clustering, 3D scatterplotting, heatmaps, and dendograms. However, before I can begin the clustering analysis I need to explain some background information regarding the Vietnam War and data clustering.

The Vietnam War

The Vietnam War was a 20 year long conflict in primarily in Vietnam, although the war reached portions of Laos and Cambodia as well. The Vietnam war was a long, costly, and highly divisive conflict, which saw the United States eventually pull out of South Vietnam in 1975. U.S. intervention in the region was an attempt to prevent the domino theory, which hypothesized that the fall of one country to Communism would lead to many others following into Communism. The war saw close to 60,000 U.S. soldiers die, and over 2 million Vietnamese. There is some controversy over whether or not the United States “won” the Vietnam War, as the casualty rates were extremely lopsided in America’s favor, but South Vietnam did fall to Communism.

The dataset

This dataset consists of 72 and 5 columns. The five columns are: the month, U.S. troop deaths, South Vietnamese troop deaths, third party troop deaths, and enemy troop deaths. Each row of the data has a month, so the casualty numbers are by month. This dataset contains six years worth of casulalty data, from January of 1966 to December of 1971.

Data Clustering

To begin with, clustering is to organize a set of objects with respect to their properties. A cluster is a collection of objects which share certain properties and are different from other clusters. In data clustering, there are many ways to separate data and conduct analysis on it. I will go over a few of the most common methods below.

Distance measures

Before I discuss data clustering methods I must first go over **distance measures**, which is how the distances are calculated between points, as distance measures is a common method for finding the differences between points (you just take the distance between them). (Source: Brownlee (2020))

Euclidean Distance

Euclidean distance is one of the most common methods for distance calculations, developed by Greek mathematician Euclid over 2,000 years ago. Euclidean distance is commonly used to find distances between two columns and their rows. The formula for Euclidean distance is: sqrt((x1-x2)^2 + (y1-y2)^2), in which (x1,y1) is a point and (x2,y2) is a point.

Manhattan Distance

Also called the Taxicab distance or the City Block distance, the Manhattan distance is best suited for datasets which describe objects in a uniform grid, hence the “City Block” name. It calculates the shortest to take while navigating on the grid. The formula for Manhattan distance is: |x1-x2| + |y1-y2|, in which (x1,y1) is a point and (x2,y2) is a point.

Correlation-based Distance

Correlation-based distance is exactly what it sounds like - the “distance” is based on how closely the objects are correlated. For example, two highly correlated objects, regardless of the geometric distance given by Euclidean or Manhattan distances, would be considered very close together by correlation-based distance.

Partition clustering

Partition clustering are a series of clustering methods used to classify observations within a data set into multiple groups based on similarity. It breaks down a data set into a set of clusters, constructing partitions, or groups, in the data set. There are different types of partitioning clustering methods. The most popular is the K-means clustering, while others include K-medoids clustering or PAM (Partitioning Around Medoids) clustering. A medoid is a representative object of a data set which serves as an average. However, it is different from an average because the medoid is restricted to only the values of the data set.

K-means clustering

While K-means clustering is a partition clustering method, because of its usage and importance in data clustering it receives its own section. There is a number, k, which describes the number of clusters to create. Based on the k value, k random points are selected out of a dataset. All other points are then sorted into clusters based on their distance to the selected points. After this, the average value for each cluster is calculated, and points are re-clustered according to their distances from the average value. This re-clustering occurs until there is no more change in the clusters. With the clusters found, you then add up the total variation in each cluster. However, because K-means clustering can’t find the best clustering method, it does many iterations to determine which points would be the best to cluster around, by sorting to the lowest variation. After running many iterations, it will take the iteration with the lowest variation, and use the points from that as your clusters.

Hierarchical clustering

Hierarchical clustering is often closely associated with heatmaps and dendograms. In hierarchical clustering, rows are ordered based on similarity. The difference between K-means clustering and hierarchical clustering is that there are no “clusters” in hierarchical clustering, whereas in K-means clustering all points are sorted into groups. In hierarchical clustering, rows (or columns) are essentially sorted into a list (a hierarchy). Starting at the first row, a similarity (based on distance) is calculated for each other row and the first one. Then, the row most similar with the first row is grouped together. This action of finding the similarity is then repeated until all the rows, and the new groups are treated as rows, have been grouped together. Based on the order in which rows were formed, a dendogram can be created. For instance, the shortest branches on the dendogram would be the the rows which got grouped first, because they are the most similar. Conversely, the longest rows in the dendogram are the rows which were grouped last, as they are the least similar.

Clustering analysis in R

Setup

In this section, I am going to review basic environment clean-up and imported packages, along with cleaning the data and making sure everything is ready for analysis. First, I clean up and set up the R environment with the working directory. Keep in mind that your directory structure is likely different from mine, and adjust accordingly. Furthermore, I turn warnings off in the effort of saving paper.

```
# Andrew Wang
# November 2
# Clustering Analysis of Vietnam War Casualties
#
# clean up and setup
rm(list=ls()) # clean up any old stuff in R
setwd("C:/Users/hyper/OneDrive/Desktop/Desktop Folders/Programming/R/Assignments/Week 9") # go t
o this folder
#load up myfunctions.R
source("C:/Users/hyper/OneDrive/Desktop/Desktop Folders/Programming/R/myfunctions.R")
options(warn = -1)
```

After this is done, I import the packages I’m using. Keep in mind that you will likely need to install these packages first before using them.

```
#library import
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(cluster)
library(scatterplot3d)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(NbClust)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

Getting the data in a .csv

Prior to importing the csv file into R, I had to do a series of formatting steps to get it into a state where it could be imported into R. I'll go over the steps here. First, I saved the .txt file to my hard drive. Next, I deleted the background data, leaving only the table data. I then opened up Excel, and began the important process, from under the data tab. While I would normally just sort by the delimination chracter of space(" "), that was not possible for this dataset because the spacing was inconsistent. However, because the number of characters in each column were the same I just split the table into columns through character counting. After the data was into a .csv file, I made some edits to the header names (for aesthetic) and then removed all the quotations from the dates. Now, the .csv file is ready for importation.

Setup (Part 2)

Finally, we can import our .csv file, and assign it to a variable. Notice that I've also added some optional arguments to my import, which specify the location of the row names in the importation. I then run some basic observations on the dataset as a starting point, in order to get a basic overview of the data we have. In the effort of saving paper, I have commented out the str() and summary() functions here because they have a large amount of output.

```
#basic cleaning + analysis
warData <- read.csv("warData.csv", header=TRUE, row.names = 1)
view(warData)
#summary(warData)
#str(warData)
print(sum(is.na(warData)))
```

```
## [1] 8
```

Notice something strange here, there are NA values in our dataset! After looking back into the .csv file, I realized that the NA values are merely the last two rows in the dataset, can I can simply remove them without repercussion.

```
#we need to get rid of the missing values
tail(warData)
```

```
##        US.deaths South.Vietnamese.deaths Third.party.deaths Enemy.deaths
## Sep-71        78                    1607                 27         6300
## Oct-71        29                    1574                 20         5744
## Nov-71        19                    1161                 14         4283
## Dec-71        17                     988                 26         4439
##               NA                      NA                 NA           NA
##               NA                      NA                 NA           NA
```

```
warData <- na.omit(warData)
print(sum(is.na(warData)))
```

```
## [1] 0
```

As the final step in setup, I set the seed, which is used in K-mean clustering.

```
#set seed
set.seed(123)
```

K-Means Clustering

To start off with, I want to conduct a relativly simple K-Means clustering on my data. First, I used the kmeans() function to conduct a K-Means cluster analysis on the dataset, and I set k = 8, with 25 iterations. I decided to make k = 8 because I felt it was an ample representation after testing with values from 3 to 10. After the cluster analysis, I print out the cluster column, which shows what cluster which row belongs to. Finally, I create a simple scatterplot out of U.S. deaths and enemy deaths, with appropriate labels and coloring based on the cluster group.

```
group <- kmeans(warData, centers = 8, nstart = 25)
group$cluster
```

```
## Jan-66 Feb-66 Mar-66 Apr-66 May-66 Jun-66 Jul-66 Aug-66 Sep-66 Oct-66 Nov-66
##      6      6      5      6      6      6      5      5      6      5      5
## Dec-66 Jan-67 Feb-67 Mar-67 Apr-67 May-67 Jun-67 Jul-67 Aug-67 Sep-67 Oct-67
##      6      5      8      3      5      3      8      8      5      5      5
## Nov-67 Dec-67 Jan-68 Feb-68 Mar-68 Apr-68 May-68 Jun-68 Jul-68 Aug-68 Sep-68
##      8      8      2      7      2      1      4      3      5      2      1
## Oct-68 Nov-68 Dec-68 Jan-69 Feb-69 Mar-69 Apr-69 May-69 Jun-69 Jul-69 Aug-69
##      8      3      3      3      1      4      1      2      2      3      1
## Sep-69 Oct-69 Nov-69 Dec-69 Jan-70 Feb-70 Mar-70 Apr-70 May-70 Jun-70 Jul-70
##      3      8      1      3      3      8      3      1      2      8      8
## Aug-70 Sep-70 Oct-70 Nov-70 Dec-70 Jan-71 Feb-71 Mar-71 Apr-71 May-71 Jun-71
##      5      5      5      5      5      5      1      4      3      3      8
## Jul-71 Aug-71 Sep-71 Oct-71 Nov-71 Dec-71
##      5      5      5      5      6      6
```

```
#simple plot
plot(warData$US.deaths, warData$Enemy.deaths, main = "US deaths vs. enemy deaths", xlab = "US de
aths", ylab = "Enemy deaths", col = group$cluster, pch = 20)
```

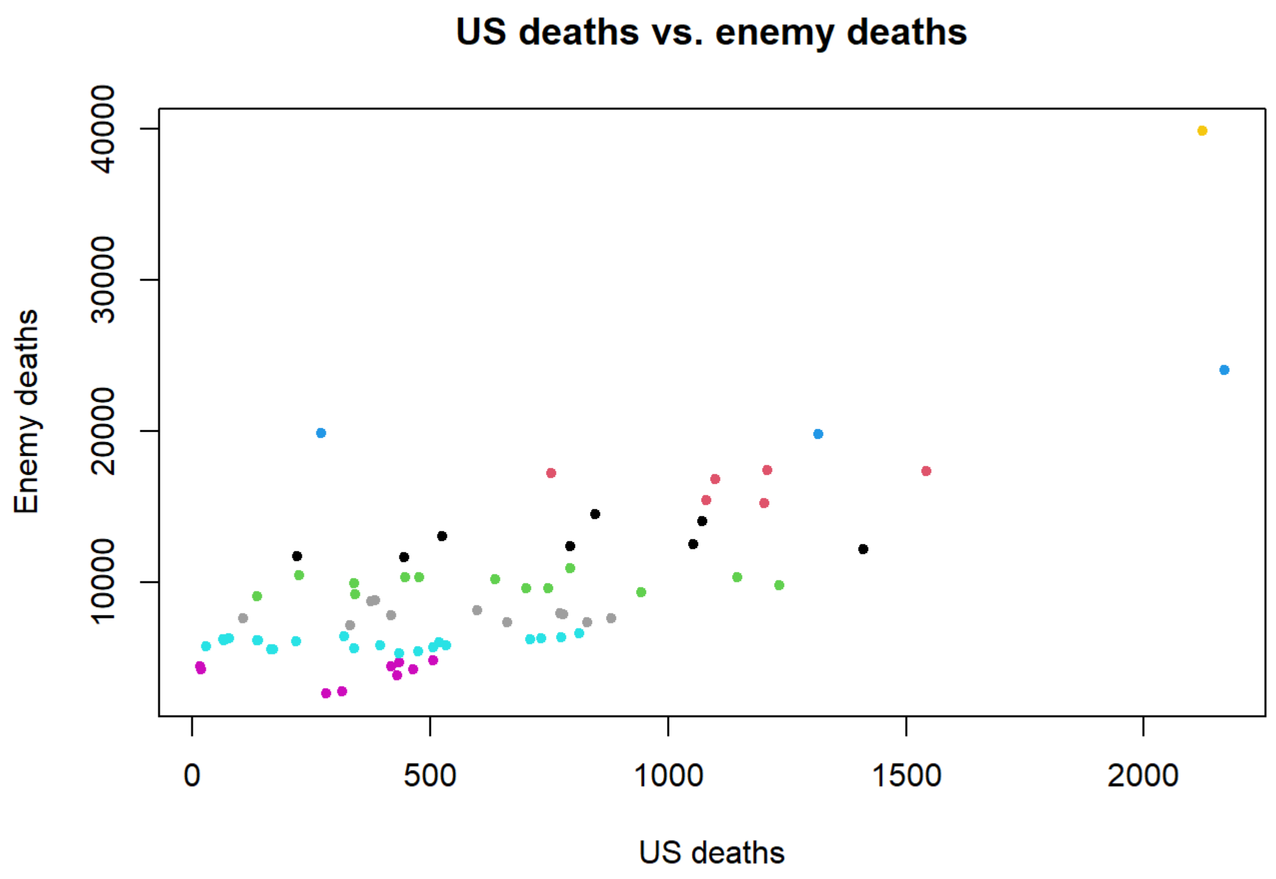

Figure 1: K-mean 2D Scatterplot

Looking at this scatterplot, the first thing I noticed was the scale of the two axes. The y-axis, enemy deaths, had a much large scale than U.S. deaths, which represented the efficiency of the United States military in the Vietnam War, the casualty ratio was heavily in their favor. Each month the enemy would take many times more casualties than the United States. Another thing that stood out was the variation with respect to the y-axis. As enemy deaths increased, the variation of datapoints for U.S. deaths also increased. Also, notice the one cluster at the top-right, its important to keep an eye on outliers.

3-D scatterplot

Next, I wanted to create a 3-D scatterplot with the same clustering data, so that the visualization of another dimension would aid in the analysis of the dataset. The dimension I added was South Vietnamese deaths, as I felt the numbers for third party deaths were not substantial enough to be of any analysis value. The code below for the 3-D scatterplot is essentially the same, with an added Z-axis and accompanying Z-axis label.

```
#3d plot
scatterplot3d(warData$US.deaths, warData$Enemy.deaths, warData$South.Vietnamese.deaths,  type =
"h", angle = 55, pch = 20,   main = "3D scatterplot Vietnam war deaths", xlab = "US deaths", yla
b = "Enemy deaths", zlab = "South Vietnamese deaths", grid = TRUE, col.grid = "blue", color = gr
oup$cluster)
```

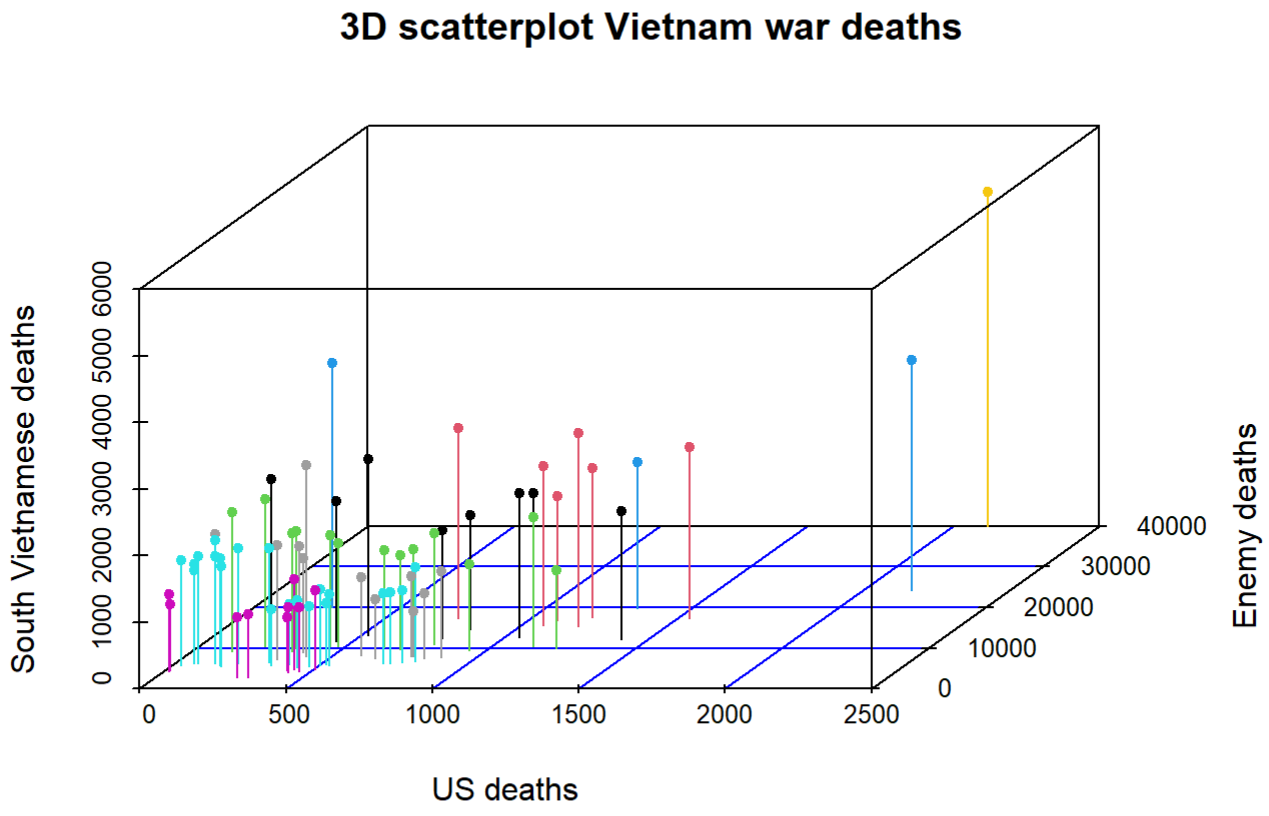

Figure 2: K-Mean 3D Scatterplot

Looking at the scatterplot, I noticed the relationship between South Vietnamese deaths and U.S. deaths: as U.S. deaths went up generally South Vietnamese deaths went up as well. Again I noticed the scale, in which enemy deaths still far outnumbered U.S. or south Vietnamese deaths. However, it seems that generally that more South Vietnamese soldiers died per month than U.S. soldiers, likely because the U.S. government had been training and equipping the South Vietnamese soldiers before any actual U.S. military forces arrived in Vietnam.

Euclidean Heatmap

For the next cluster analysis, I wanted to get a heatmap so I could look at the specific outliers in the data, as well as identify general trends in casualties. However, in order to maintain the readability of the heatmap I had to take a subset of the data, and therefore I only took the first 24 months (2 years). I used the euclidean distance method (discussed above) to calculate heatmap differences.

```
#gets sample for better heatmap
subset <- warData[1:25, ]

#eudclidian heatmap
eucl <- dist(subset, method = "euclidean")
fviz_dist(eucl)
```

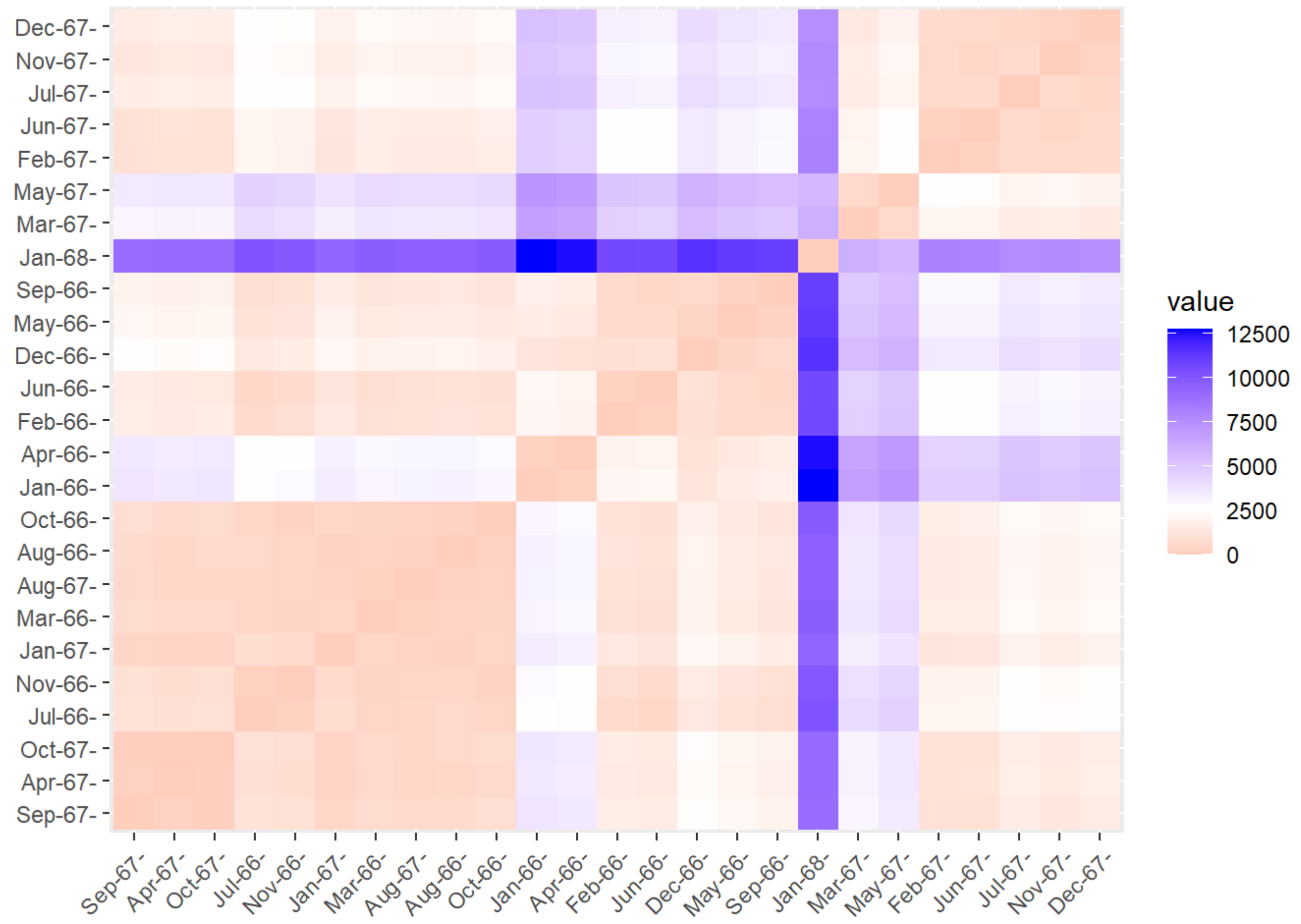

Figure 3: Euclidean Heatmap of data subset

The first thing I noticed from this graph was the blue line which was January of 1968. In the heat map, strong purples indicate many more casualties. January of 1968 was the outlier month in both the 3D and 2D scatterplot, and this makes sense because it corresponds with the Tet Offensive, in which the Viet Cong launched a massive counterattack across South Vietnam. Two other months, less outliers but still somewhat purple include March of 1967 and May of 1967. This is strange, considering April of 1967 didn't have large casualties and was mostly in line with other months.

Hierarchical Clustering

For my final cluster analysis graphic, I chose to do a dendogram. I deicded to do a dendogram because I want to get a picture of how different months related to each other, and how similar months in a year would be. I put the dataset into the fviz_dend() function, which is meant to create a dendogram. By setting k = 6, I create six "clusters" or groups, and turn the labels on so that months can be identified.

```
#hierarchical clustering dendogram
fviz_dend(hclust(dist(warData)), k = 6, as.ggplot = TRUE, show_labels = TRUE)
```

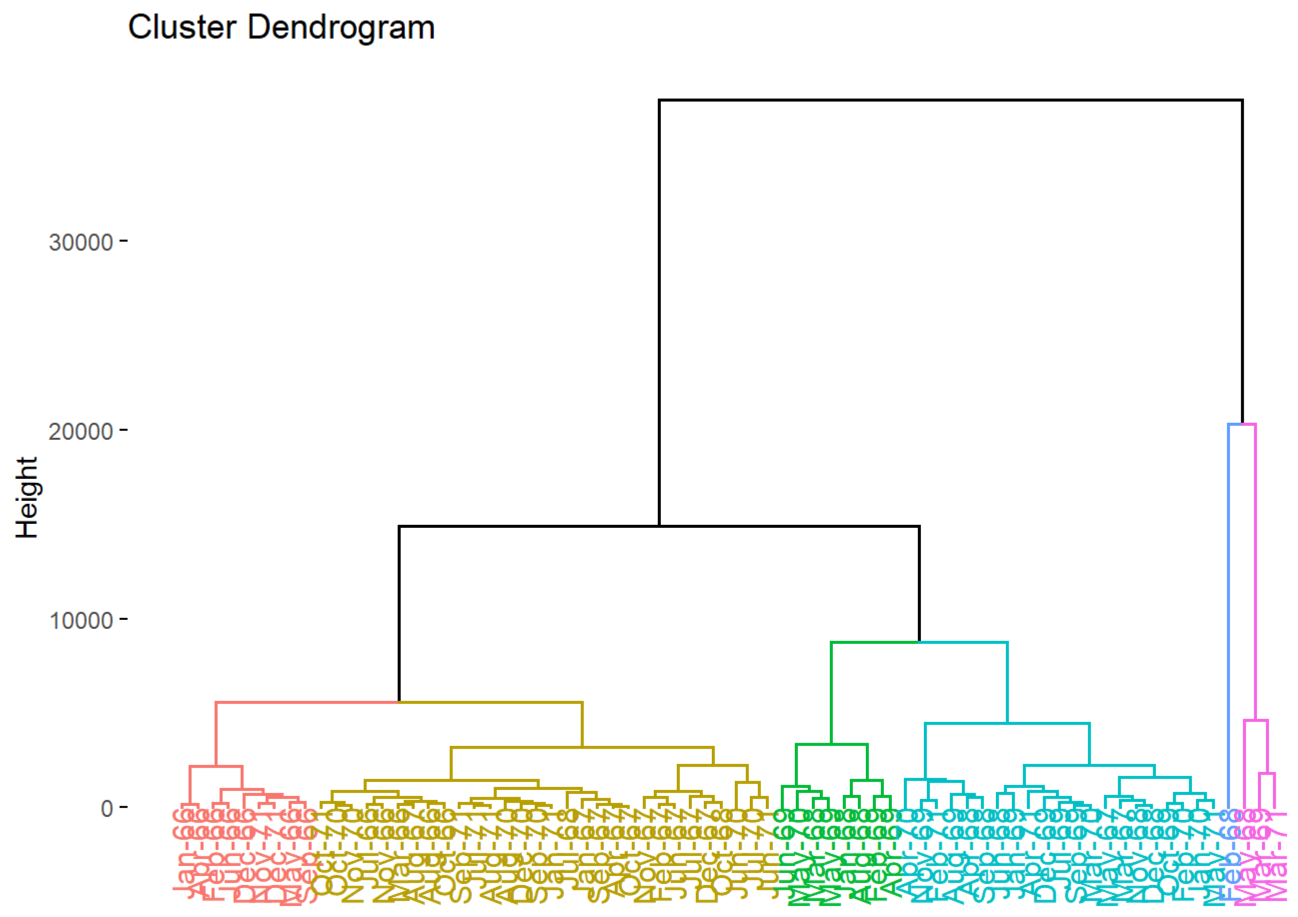

Figure 4: Dendogram

While the specific months may be hard to read, we can gather a few takeaways from this dendogram. Firstly, we can conclude that there was little variation among casualty rates for all factions for a majority of the years, as the lengths of the branches in the dendogram are mostly very short, indicated that the months are very close. Furthermore, we can again see the outlier month, but this time we can also see the other months closest to it, likely the blue dots from the original 2D scatterplot.

Conclusions

After a cluster analysis of this data, I am able to come away with a few conclusions:

- The United States had a large advantage in military equipment and training. This is evident in the large ratio between causalities of U.S. soldiers and enemy soldiers.
- Most months saw little variation in casualty rates. This suggests that there were no large tactical gains by either side. The obvious exception is the Tet Offensive.
- The Tet Offensive was a strategic win but a tactical loss for the North Vietnamese. In looking at the data surrounding the Tet Offensive, it is clear that the North Vietnamese lost many times more men than either the South Vietnamese or U.S. combined, yet the Tet Offensive is still considered a turning point in the Vietnam War. Therefore, while the North Vietnamese lost many soldiers during the Tet Offensive, it was a strategic victory as it causes Americans to shift their focus to trying to disengage from the conflict.

References

Brownlee, Jason. 2020. *4 Distance Measures for Machine Learning*. *Machine Learning Mastery*. https://machinelearningmastery.com/distance-measures-for-machine-learning/#:~:text=A%20distance%20measure%20is%20an,claim%2C%20or%20a%20diagnosis). (https://machinelearningmastery.com/distance-measures-for-machine-learning/#:~:text=A%20distance%20measure%20is%20an,claim%2C%20or%20a%20diagnosis).)

Hartigan, John. 1975. *Combat Deaths in Indochina*. *Clustering Algorithms*. https://people.sc.fsu.edu/~jburkardt/datasets/hartigan/file13.txt (https://people.sc.fsu.edu/~jburkardt/datasets/hartigan/file13.txt).