My Comprehensive Evaluation

A Comprehensive Evaluation Report
Presented to
The Statistics Faculty
Amherst College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts
in
Statistics

Kaitlyn E. Haase

February 26 2019

Acknowledgements

I want to thank my family.

Table of Contents

List of Tables

List of Figures

Abstract

In recent years, the amount of geographic data has increased immensely. With new technology, the accuracy and complexity of data has also improved. This has provoked statisticians to create techniques to best analyze and draw conclusions from this new-found data. Earlier techniques of spatial data were not equipped to handle the complexity and quantity of the data. This project first explores how and why we analyze data based on geographic information. Next, I will explain some of the newer spatial data algorithms, including PAM (Partitioning Around Medoids), CLARA (Clustering LARge Applications), and CLARANS (Clustering Large Applications based on RANdomized Search). Example data will be used to demonstrate CLARA, and the project will conclude a model to predict cluster.

In recent years, the amount of geographic data has increased immensely. With new technology, the accuracy and complexity of data has also improved. This has provoked statisticians to create techniques to best analyze and draw conclusions from this new-found data. Earlier techniques of spatial data were not equipped to handle the complexity and quantity of the data. This project first explores how and why we analyze data based on geographic information. Next, I will explain some of the newer spatial data algorithms, including PAM (Partitioning Around Medoids), CLARA (Clustering LARge Applications), and CLARANS (Clustering Large Applications based on RANdomized Search). Example data will be used to demonstrate CLARA, and the project will conclude a model to predict cluster.

Introduction

As mentioned in the abstract, we have much more spatial data than we have had in the past. Spatial data analysis is analyzing data based on topological, geometric, and geographic information. Spatial data may include latitude and longitude, zip code, or street address.

0.1 Why Analyze Spatial Data?

We are interested in analyzing spatial data for many reasons, one being because there is so much of it available. Investigating spatial data can help us find dissimilarities and similarities among objects. This can aid us in allocating resources to areas that need them most, discovering changes over time, and categorizing new objects.

0.2 Analyzing Spatial Data Algorithms

There are many algorithms out there that handle spatial data; most algorithms are focused around clustering. To get a glimpse of the number of algorithms and strategies to analyze spatial data, the chart below provides some examples.

```
label(path = "clustering_methods.png",
      caption = "Clustering Methods",
      label = "Clustering", type = "figure", scale= 0.5)
```

As noted, the methods are aimed around clustering, which we will further explore in the next section.

0.2.1 Clustering

Clustering organizes a set of data items into groups so that items in the same group are similar to each other and different from those in other groups [Rec 1]. Clustering

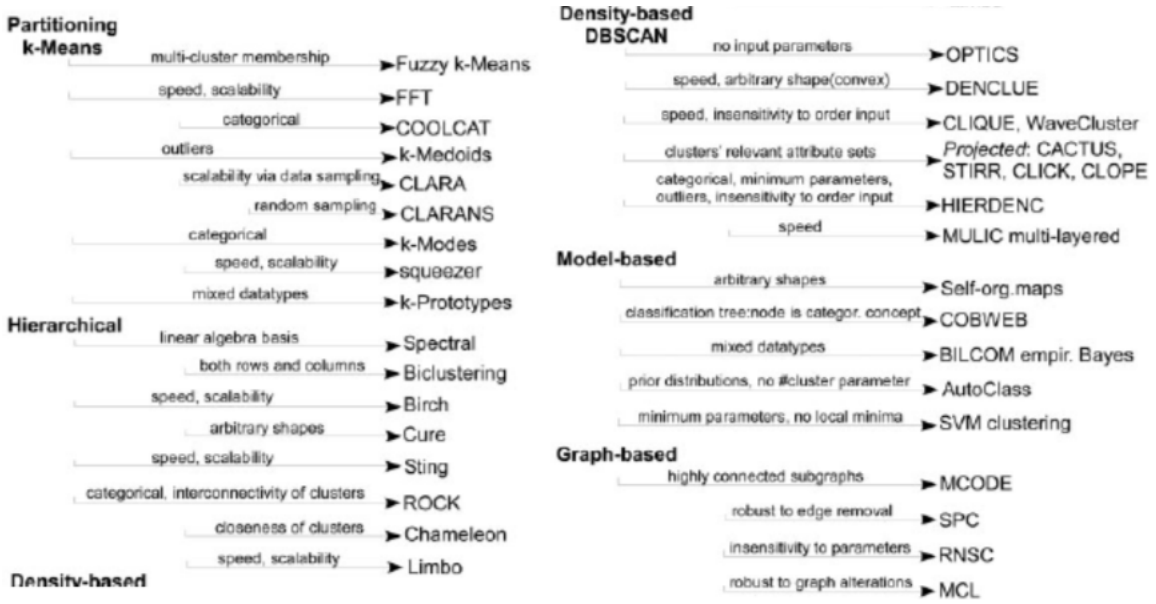


Figure 1: Clustering Methods

is helpful in finding patterns and similarities/differences between data points and groups; however it can be quite subjective. It is up to the statistician to determine how many clusters are appropriate for the data, as well as the cut off for what is considered "dissimilar" or "similar". Additionally, the statistician must choose which clustering algorithm is best to use... This will be discussed in more detail later on in the project.

Chapter 1

How to Cluster

There are many factors to consider when choosing a clustering algorithm, such as the application of the problem (what do you want to find out about this data?), quality vs speed trade off (the size of the data plays a role), characteristics of the data (i.e. numeric distance measures), dimensionality (typically as dimension increases the time it takes to run the method increases and quality of the data clusters decrease), and outliers (some methods are very sensitive to outliers) [Rec 2].

1.1 Types of Clustering: Partitioning

There are four main types of clustering: hierarchical, partitioning, density-based, and methods-based. Next, I'll dive into the partitioning clustering technique.

Partitioning cluster methods divide a set of data items into a number of non-overlapping clusters. A data item is typically assigned to a cluster based on a proximity or dissimilarity measure [Rec 2, p. 405].

Usually, there is a data set with n observations and the goal is to divide the data points into K clusters so that an objective function is optimized.

The most common objective function is the sum of squared errors (SSE), where c_k is the centroid or medoid of the cluster C_k.

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i C_k} ||x_i - c_k||^2$$

Partitioning clustering algorithms classify the data into K groups by satisfying both that each group has at least one data point, and that each data point belongs to exactly one group. [Rec 5, p. 18].

1.2 Methods to Create Clusters: *K*-Medoids

There are many ways to create clusters. The most basic method is the K-means algorithm, which was developed by MacQueen in 1967 [Rec 5, p. 18]. In response to K-means being very sensitive to outliers, the K-medoid algorithm was created in 1987 [Rec 5, p. 19]. Both partitioning methods use iterative processes to find K clusters; however, they use different ways to represent these clusters.

1.2.1 *K*-Means

K-means algorithm represents its n observations in k groups, with the center of the groups being the mean/average observation. The goal of the algorithm is to find k centroids, one for each cluster. In order to do this, we must minimize an *objective function*, which is the squared error function for k means. The objective function is:

$$O = \sum_{j=1}^{k} \sum_{i=1}^{j} ||X_i^{(j)} - C_j||^2$$

Where $|X_i^{(j)} - C_j|$ is an indicator of the distance of the data points from their cluster centers.

The steps of the algorithm are as follows:

1. Choose K points in the space to represent the centroid. This works best if they are chosen to be far apart from each other.
2. Assign each object in the data set to the cluster with the closest centroid.
3. When all of the clusters have been made, recalculate the positions of the K centroids.
4. Repeat steps 2 and 3 until the centroids no longer move.

This algorithm always terminates; however, it is sensitive both to outliers and to the initial randomly selected K cluster centers. Therefore, the algorithm should be run multiple times to reduce the effects from this sensitivity.

[Rec 5, p. 18]

In order to determine how well _K_means worked, we use the within cluster sum-of-squares to determine the compactness/“goodness” of the clustering (and we want it as small as possible).

We calculate the WSS by the following equation:

$$WSS = \sum_{k=1}^{k} \sum_{x_i = C_k} (x_i - mu_k)^2$$

Where x_i is a data point in cluster C_k and mu_k is the mean value assigned to the cluster C_k. [Rec 7].

1.2.2 *K* Medoids

On the contrary, instead of taking the mean value of the objects in a cluster, the k-medoid method uses the most centrally located object in a cluster to be the cluster center [Rec 2]. This causes the method to be less sensitive to outliers, but also requires more time to run.

Steps for K-medoids: 1. Initial guess for centers C_1, C_2,... C_k (i.e. randomly select k points from X_1, X_2,... X_n) 2. Minimize over C: for each i= 1, 2,... n, find the cluster center C_k closest to Xi and let C(i)=k. 3. Minimize over C_1, C_2,... C_k: for each k=1,... K, $C_k = X_k^*$, the medoid of points in cluster k. ie, the point Xi in the cluster k that minimizes

$$\sum_{c(j)=k} ||X_j - X_i||^2$$

Basically, _K_means and _K_medoids follow very similar algorithms; however, _K_medoids uses the most centrally located object (medoid) in a cluster to be the cluster center. This causes there to only be at most one center changed for each iteration (makes the algorithm run slower). [rec 2, p. 6].

1.3 How to Choose *K*

Now that we've discussed different kinds of _K_means and _K_medoids partitioning methods, we know how to find K clusters of data points; but how do we determine what K is?

Well, there are many ways to choose k, which is why these methods are so subjective.

I will describe two of the many ways to determine K, both of which use visuals to determine what value of k is appropriate for the data. The elbow method and silhouette method are common ways to find K when using the K means and K medoids algorithms.

1.3.1 Elbow Method

To start, the elbow method looks at the total within-cluster sum of squares (WSS) and determines when there are enough clusters so that the next cluster does not improve the total WSS very much. This would be the appropriate K to choose.

The steps for this algorithm are as follows:

1. Compute the clustering algorithm (i.e. _k_medoids method) for different values of k (i.e. k from 1 to 10).
2. For each k, calculate the total WSS. WSS can be calculated as:

$$WSS = \sum_{i=1}^{k} \sum_{x_i = C_k} ||x_i - c_k||^2$$

Where x_i is a data point in cluster C_k and c_k is the medoid assigned to the cluster C_k. [Rec 7].

3. Plot the curve of the total WSSs according to the number of clusters (k).
4. The location of the bend in the plot is generally considered an indicator for the appropriate number of clusters.

There will be an example of this method used in Chapter 3.

1.3.2 Silhouette Method

The Silhouette Method focuses on the quality of clustering. A high average silhouette width indicates a good clustering (how well each object lies within its cluster).

The steps of the Silhouette Algorithm are:

1. Compute clustering algorithm for different values of k (i.e. k from 1 to 10).
2. For each k, calculate the average silhouette of observations. There is a silhouette method in R that can calculate this for us...
3. Plot the curve of the average silhouettes according to the number of clusters (k).
4. The location of the maximum is considered the appropriate number of clusters.

There will also be an example of this method used in Chapter 3. –> datanovia website

Chapter 2

Clustering Methods Continued

2.1 PAM

Partitioning Around Medoids (PAM) is the most commonly used type of k-medoid clustering (Kaufmann & Rousseeuw, 1987).

It iterates through all the k cluster centers and tries to replace the center with one of the other objects (n-k possibilities) [Rec 2]. For a replacement to occur, the squared error function must decrease (if it does not decrease, there is no replacement). The algorithm eventually terminates with a local optimum.

The squared error function calculates the average dissimilarity

The total complexity of PAM in one iteration is

$$O(k(n-k)^2)$$

(O= each non-medoid data point, k= # of cluster centers,

$$(n-k)$$

objects to compare to, and

$$(n-k)$$

operations for calculating E). This makes for a costly computation when n is large. The algorithm works best when n= 100 and k=5.

Explanation of PAM, REC 6, P. 146–> 4 cases, and algorithm Rec 6 bibliography (Ng & Han, 2000)

2.2 CLARA

Because PAM does not scale well to large data sets, Clustering LARge Applications (CLARA) was developed to deal with larger data sets (Kaufmann & Rousseeuw, 1990).

CLARA is a sampling based method, meaning a sample of the data is used to represent the entire data set. Medoids are chosen from this sample data using PAM and then "the average dissimilarity is computed using the whole dataset" (**don't know what "average dissimilarity" means or how it is calculated). If a new set of medoids gives a lower dissimilarity than a previous best solution, then the best solution is replaced with a new set of medoids [Rec 2, p. 7].

Experiments indicate that 5 samples of size 40+ 2 k give satisfactory results [Rec 6, p. 146].

The steps for the algorithm are as follows: 1. For i= 1 to 5, repeat the following steps: 2. Draw a sample of 40+ 2 k objects from the entire data set, and use PAM to find k medoids of the sample. 3. For each object O_j in the entire data set, determine which of the k medoids are most similar to O_j. 4. Calculate the average dissimilarity of the clustering obtained in the previous step. If this value is less than the current minimum, use this value as the current minimum, and retain the k meoids found in Step 2 as the best medoids obtained so far. 5. Return to Step 1 to start the next iteration.

*PAM on samples

`https://www.coursera.org/lecture/cluster-analysis/3-4-the-k-medoids-clustering-method-nJ0Sb`

2.3 CLARANS

(Ng & Han, 1994)

*Randomized re-sampling, ensuring efficiency and quality

```
#How to insert a figure, make sure amherst.png is in main directory

label(path = "clustering_pic.png",
      caption = "CLARANS searching for a better solution",
      label = "CLARANS", type = "figure", scale= 0.5)
```

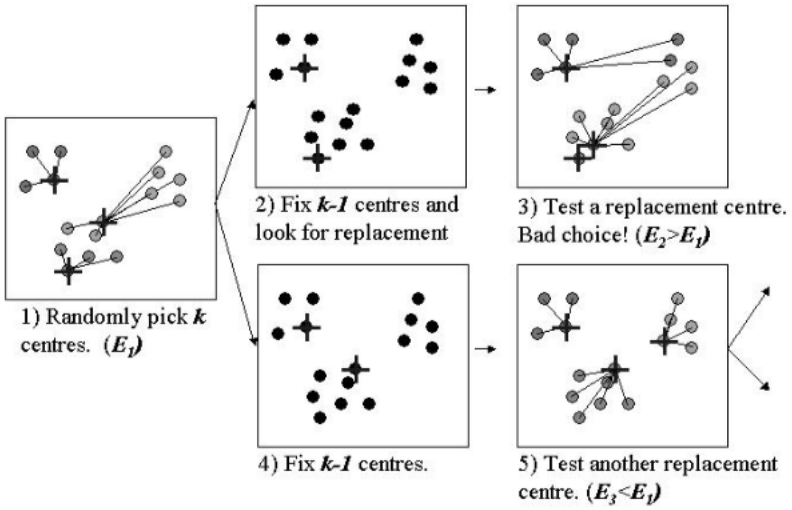


Figure 2.1: CLARANS searching for a better solution

Chapter 3

Example

```
#loading in packages
library(readr)
library(factoextra)
```

```
Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.
```

```
library(NbClust)
library(ggplot2)
library(cluster)
library(GGally)
```

```
Attaching package: 'GGally'
```

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

    nasa
```

3.1 Exploring the Data

In this example, I will further explore the data from my Stat 495 final project. The data is from DataUSA, which uses public US Government data to analyze and visualize relationships. In the project, we decided to use data from 2016 because of size restrictions. The data contains spatial information, quantitative, and a few categorical variables.

There is demographic information as well as variables that are indicators of health status. The health status variables include: poor to fair health (the percentage of adults reporting fair or poor health (age-adjusted)), poor physical health days (average number of physically unhealthy days reported in the past 30 days (age-adjusted)), physical inactivity (the percentage of adults aged 20 and over reporting no leisure-time physical inactivity), and adult obesity (the percentage of adults to report a BMI of greater than or equal to 30). In interpreting the health indicator variables, the higher the values for these variables, the less healthy a person is.

For our Stat 495 project, we used mapping techniques to visualize and analyze the data. We were unable to draw many conclusions from the map, which is why I am interested in analyzing the data through clustering. Since clustering utilizes spatial information, it may be helpful in finding patterns in the data.

My research question is to see whether there are clusters of people with exceptionally good or exceptionally poor health. This information could lead to further insights into what environmental or other factors are impacting peoples' health.

I plan to use the CLARA method, since I have more than 100 observations. The data set in fact has over 60,000 observations, so I will need to sample about 1000 observations in order to produce the best results using CLARA.

My first step is to import the data.

```
#using data from final stat 495 project
#library(readr)
data_subset <- read_csv("CopyOfdata_subset.csv")
```

```
Parsed with column specification:
cols(
  .default = col_double(),
  geo_name = col_character(),
  geo = col_character(),
  zip = col_character(),
  TRI.ID = col_character(),
  County.x = col_character(),
  County.y = col_character()
)

See spec(...) for full column specifications.
```

Next, I will take a random sample of 1000 observations. I assume the sample is representative of the data set because n=1000.

```
set.seed(1)
#getting a sample of 1000 observations
mysample <- data_subset[sample(1:nrow(data_subset), 1000,
   replace=FALSE),]
```

The data set I imported has 64 variables, which are too many for this example. Since my research question is focused around peoples' health, I will only include the health indicator variables and the latitude and longitude of the data (spatial information).

```
#only keeping the variables I want to look at
myvars <- c("Latitude_tri", "Longitude_tri", "poor_or_fair_health", "poor_physical
smallsample <- mysample[myvars]
```

The data is now ready to apply CLARA.

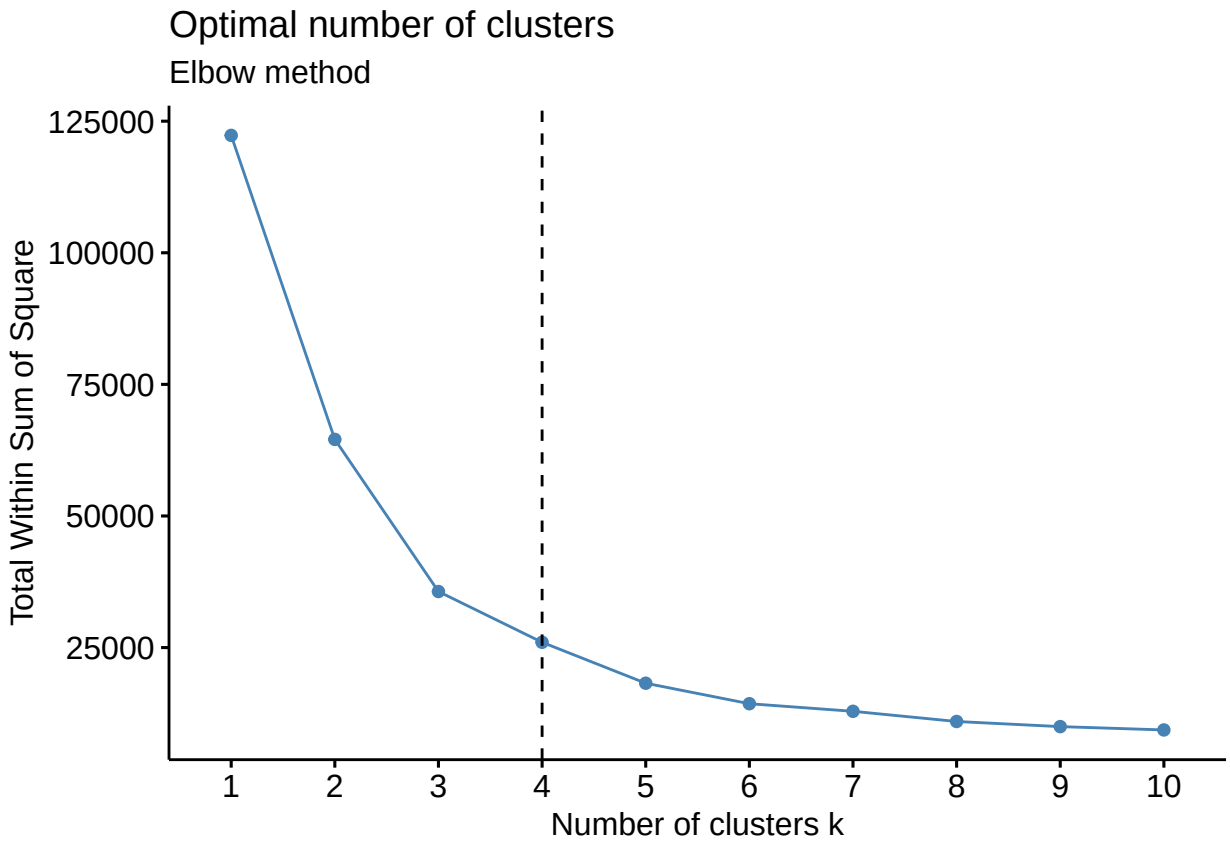
3.2 Applying CLARA

Step 1: Determining k.

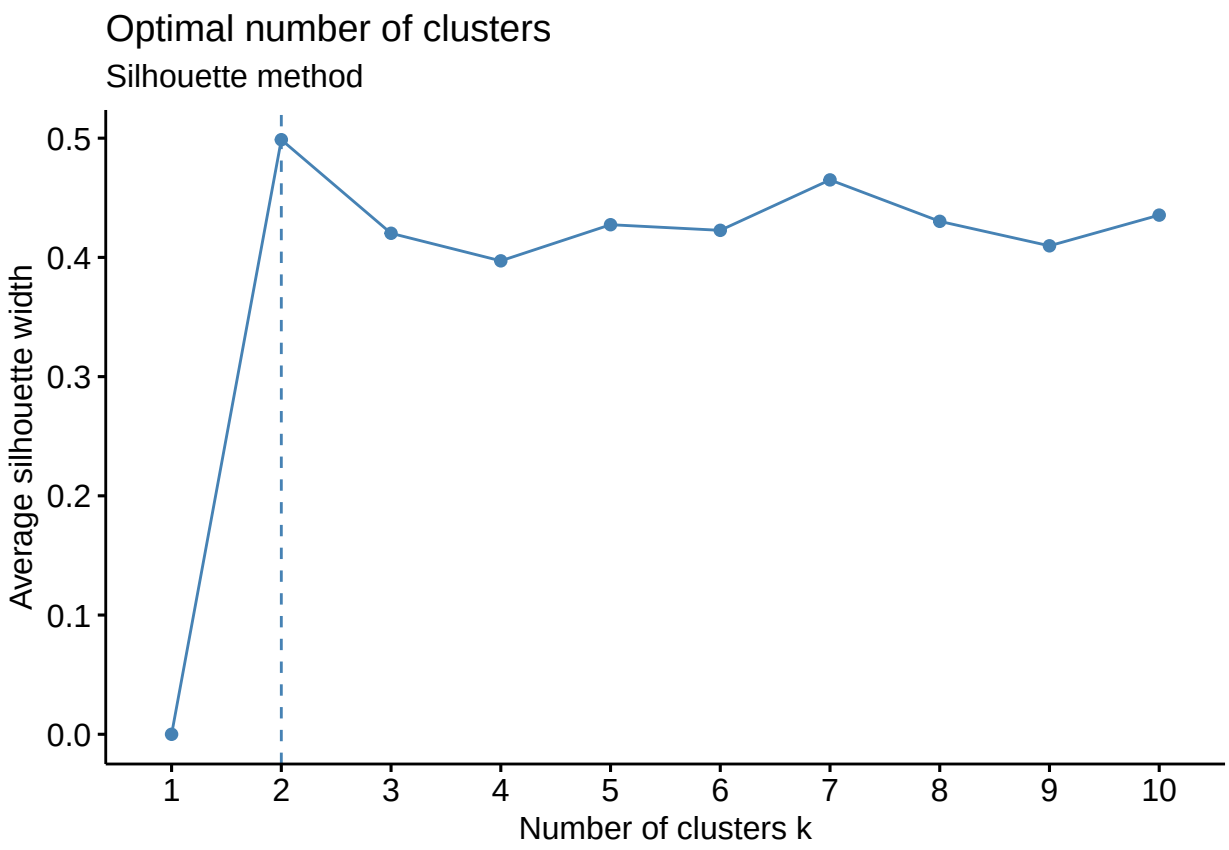
One of the major steps in clustering algorithms is determining how many k clusters is appropriate. In Chapter 1, I explained the Elbow and Silhouette methods of determining k. I will perform both methods on this data to start.

```
#finding k with project data, using Elbow Method
#pkgs <- c("factoextra",  "NbClust")
#install.packages(pkgs)

#library(factoextra)
#library(NbClust)
#library(ggplot2)
#first I must omit any of the missing data, so that the functions can run
new<- na.omit(smallsample)
# Elbow method
#fviz_nbclust(new, kmeans, method = "wss") +
```

```
 #   geom_vline(xintercept = 4, linetype = 2)+
  #labs(subtitle = "Elbow method")

fviz_nbclust(new, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```




```
fviz_nbclust(new, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette method")
```



According to the Elbow method, k should be 4 (where the elbow is in the graph). According to the Silhouette method, k should be 2 (the maximum point in the graph). Since there is variation in values of k for these methods I will take the average of the two to determine k.

Step 2: Run CLARA function

Next, I will run the CLARA algorithm on the data, using the criteria of k=3.

```
#library(cluster)
## run CLARA
clarasamp <- clara(new[1:6], 3)
```

```
## print components of clara
print(clarasamp)
```

```
Call:    clara(x = new[1:6], k = 3)
Medoids:
     Latitude_tri Longitude_tri poor_or_fair_health
[1,]      39.3265      -84.4388               0.155
```

```
[2,]      40.3973      -75.9357               0.165
[3,]      36.1336      -96.1039               0.196
     poor_physical_health_days physical_inactivity adult_obesity
[1,]                       3.7               0.232         0.289
[2,]                       3.7               0.245         0.308
[3,]                       4.6               0.353         0.355
Objective function:  5.659219
Clustering vector:   int [1:925] 1 2 1 3 1 3 1 1 1 1 1 3 1 2 1 3 1 3 ...
Cluster sizes:           457 205 263
Best sample:
 [1]   5  24  86 139 149 175 177 192 208 224 242 285 306 316 333 353 361
[18] 370 389 400 404 410 429 468 471 489 502 506 567 593 679 691 703 719
[35] 726 741 780 800 811 815 818 877 882 883 902 918

Available components:
 [1] "sample"     "medoids"    "i.med"      "clustering" "objective"
 [6] "clusinfo"   "diss"       "call"       "silinfo"    "data"
```

This output tells us a lot about the results of the clustering. To start, the information from the Medoids section show that cluster 3 contains people with the worst health, in comparison to cluster 1 and 2. For example, cluster 1 and 2 average 3.7 poor_physical_health_days, while cluster 3 averages 4.6. This difference was seen in all four health indicator variables.

The cluster sizes are also noted. There are 457 observations in cluster 1, 205 in cluster 2, and 263 in cluster 3.

```
#more output from CLARA

#cluster number for each observation
print(clarasamp$cluster)
```

```
  [1] 1 2 1 3 1 3 1 1 1 1 1 3 1 2 1 3 1 3 3 2 1 3 2 1 1 1 3 2 2 1 1 2 3 1 3
 [36] 2 1 1 3 3 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 1 1 1 1 2 2 3 3 1 2 1 3
 [71] 1 1 3 3 1 3 1 3 1 2 1 2 3 1 1 3 1 2 3 3 3 3 1 1 3 3 3 1 2 2 1 2 3 3 1
[106] 1 1 3 3 3 1 3 3 3 2 1 2 3 2 3 2 2 1 3 1 2 1 1 1 3 2 3 1 2 1 1 2 1 3 3
[141] 1 2 1 1 1 1 2 1 1 2 2 1 1 3 3 1 3 1 3 1 3 1 1 2 1 3 1 3 3 1 1 3 1 3 3
[176] 1 3 1 2 3 2 1 2 1 1 2 1 1 3 1 1 3 2 1 2 3 3 2 1 1 1 2 1 3 2 1 2 1 1 1
```

```
[211] 1 1 1 2 1 3 3 2 3 3 1 1 3 2 1 1 3 3 3 2 2 1 1 1 3 3 1 1 1 1 1 2 1 2 3
[246] 2 1 1 1 1 1 3 3 2 1 1 2 1 1 2 1 1 1 3 2 3 2 3 2 2 3 2 2 3 1 1 3 2 1 3
[281] 2 1 1 3 1 1 3 1 2 2 3 1 1 3 1 2 1 1 1 1 1 3 1 1 1 1 3 3 1 3 1 1 3 1 1
[316] 3 1 2 1 1 1 3 2 3 1 1 3 1 1 3 2 1 1 1 3 1 2 1 2 2 1 3 1 1 3 3 3 3 1 1
[351] 1 1 1 1 1 3 1 3 2 1 1 3 3 1 2 2 3 1 2 3 1 3 3 2 2 2 1 1 3 2 3 3 3 1 1
[386] 1 1 2 1 3 1 1 2 1 2 2 1 3 3 2 1 3 2 2 1 1 1 1 1 1 3 1 1 1 1 1 2 1 3 1
[421] 3 3 2 3 1 2 1 3 2 1 1 2 1 3 3 1 1 3 1 2 1 1 1 3 1 2 1 1 2 1 3 1 1 2 3
[456] 1 1 2 1 1 1 2 1 1 3 3 1 1 3 3 1 1 3 1 3 1 1 1 1 3 3 2 1 1 1 1 2 2 3 1
[491] 2 3 1 3 3 1 2 2 2 2 1 2 2 3 1 1 2 3 2 1 1 2 3 2 3 3 2 2 2 1 2 1 3 2 3
[526] 3 3 1 3 1 1 2 2 2 1 3 2 1 1 1 3 3 1 1 3 1 1 2 3 1 2 1 3 1 1 3 2 3 3 3
[561] 1 2 1 2 1 3 2 3 1 3 1 1 3 3 2 1 1 3 2 2 1 1 2 1 2 1 1 1 1 1 2 1 3 3 1
[596] 3 1 2 2 1 3 2 3 3 1 3 1 1 1 3 3 1 2 1 1 3 3 2 1 2 3 2 1 2 3 1 1 2 1 2
[631] 1 1 3 3 1 2 3 1 1 3 1 3 3 1 1 1 2 2 3 2 1 1 1 2 1 1 1 2 3 1 3 1 1 3 1
[666] 1 1 1 3 1 2 1 3 1 1 1 1 1 2 2 3 1 2 1 2 2 1 1 2 2 3 2 1 1 1 2 1 3 3 1
[701] 1 1 1 3 2 1 3 1 1 2 3 3 1 1 1 3 3 1 1 1 1 3 1 1 2 3 2 3 1 1 1 3 1 1 3
[736] 1 2 1 2 3 2 2 2 1 3 3 3 1 1 1 2 3 3 3 1 3 3 2 1 2 3 3 1 2 3 2 1 1 3 1
[771] 2 1 1 1 3 1 3 1 1 1 3 1 1 3 1 2 1 1 3 1 3 1 1 1 1 3 2 2 2 3 1 3 3 1 1
[806] 1 3 1 1 1 2 1 1 1 2 2 2 1 1 2 1 3 3 3 3 1 3 2 3 2 1 2 1 1 1 3 2 1 1 3
[841] 2 2 2 2 2 1 3 1 2 1 2 3 3 3 1 1 3 1 1 1 1 1 2 3 3 2 3 1 2 1 1 1 1 2 1
[876] 1 3 1 1 1 3 3 1 1 1 1 1 1 3 1 1 3 1 3 1 3 2 2 2 3 1 2 2 2 1 1 3 1 2 1
[911] 3 1 3 2 3 3 1 1 2 1 3 1 1 3 1
```

```
#silhouette width for each cluster
print(clarasamp$silinfo)
```

```
$widths
    cluster neighbor   sil_width
703       1        2  0.59850204
780       1        2  0.59563931
208       1        2  0.58522416
149       1        2  0.55721957
471       1        2  0.54671717
719       1        2  0.52773515
818       1        2  0.52638884
410       1        2  0.52317171
5         1        2  0.49272339
333       1        2  0.48262589
```

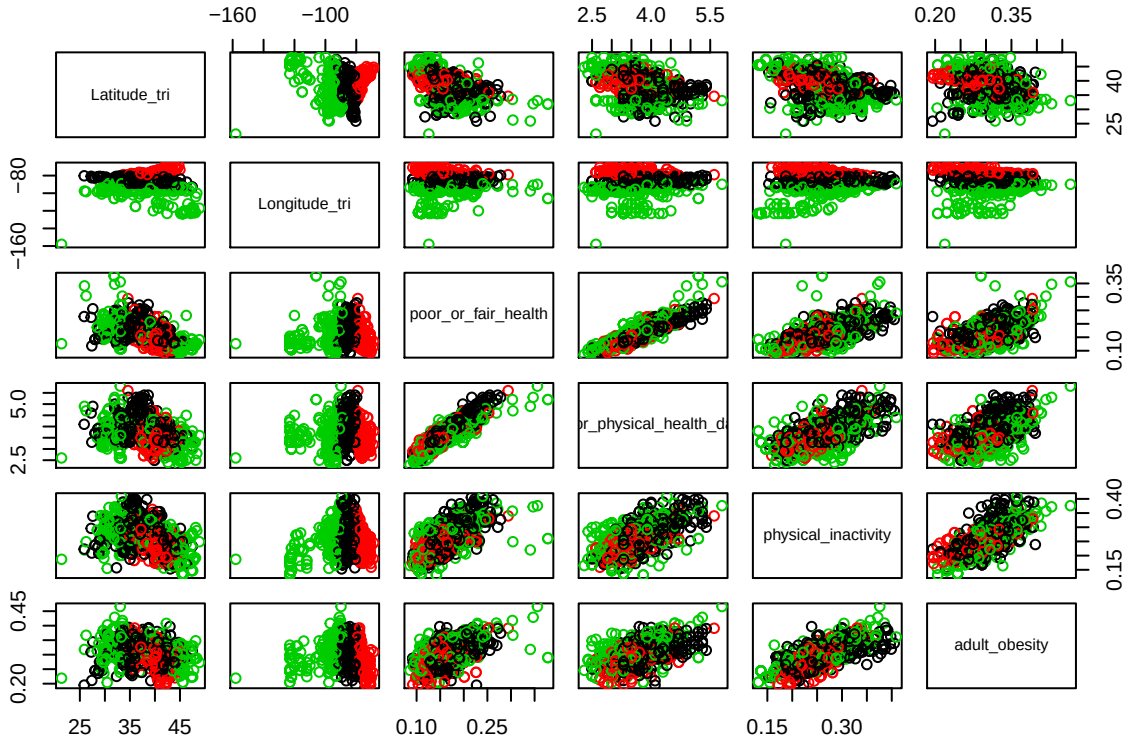
```
361         1         2  0.46877636
306         1         2  0.46546394
389         1         2  0.40532336
506         1         2  0.40333194
468         1         2  0.35920963
353         1         3  0.32935312
285         1         2  0.31204760
918         1         2  0.16440373
24          1         2  0.13034904
883         1         2 -0.03054377
224         2         1  0.75049734
679         2         1  0.74239308
404         2         1  0.73164794
902         2         1  0.73161340
242         2         1  0.73125890
400         2         1  0.73078958
502         2         1  0.65554544
741         2         1  0.64706730
567         2         1  0.62730880
811         2         1  0.46502193
815         2         1  0.40449435
429         2         1  0.39693747
882         3         1  0.33188295
139         3         1  0.31419706
726         3         1  0.31186219
800         3         1  0.31127063
691         3         1  0.29040279
370         3         1  0.28765117
489         3         1  0.28371053
177         3         1  0.26300486
175         3         1  0.23220663
86          3         1  0.06439635
593         3         1  0.03655278
877         3         1 -0.06313781
192         3         1 -0.11112090
316         3         1 -0.14469112
```

```
$clus.avg.widths
[1] 0.4221831 0.6345480 0.1720134

$avg.width
[1] 0.401444
```

This information tells us even more about the CLARA output. The first part gives us the categorizations of each data point to its cluster. The second part of information gives us the average silhouette width for each cluster. The silhouette widths were: 0.422183 for cluster 1, 0.634548 for cluster 2, and 0.172013 for cluster 3. The better the clustering is, the greater the silhouette width; so we can determine that cluster 2 was best compared to cluster 1 and 3.

Next, I will walk through some of the visualizations given this new clustering information.

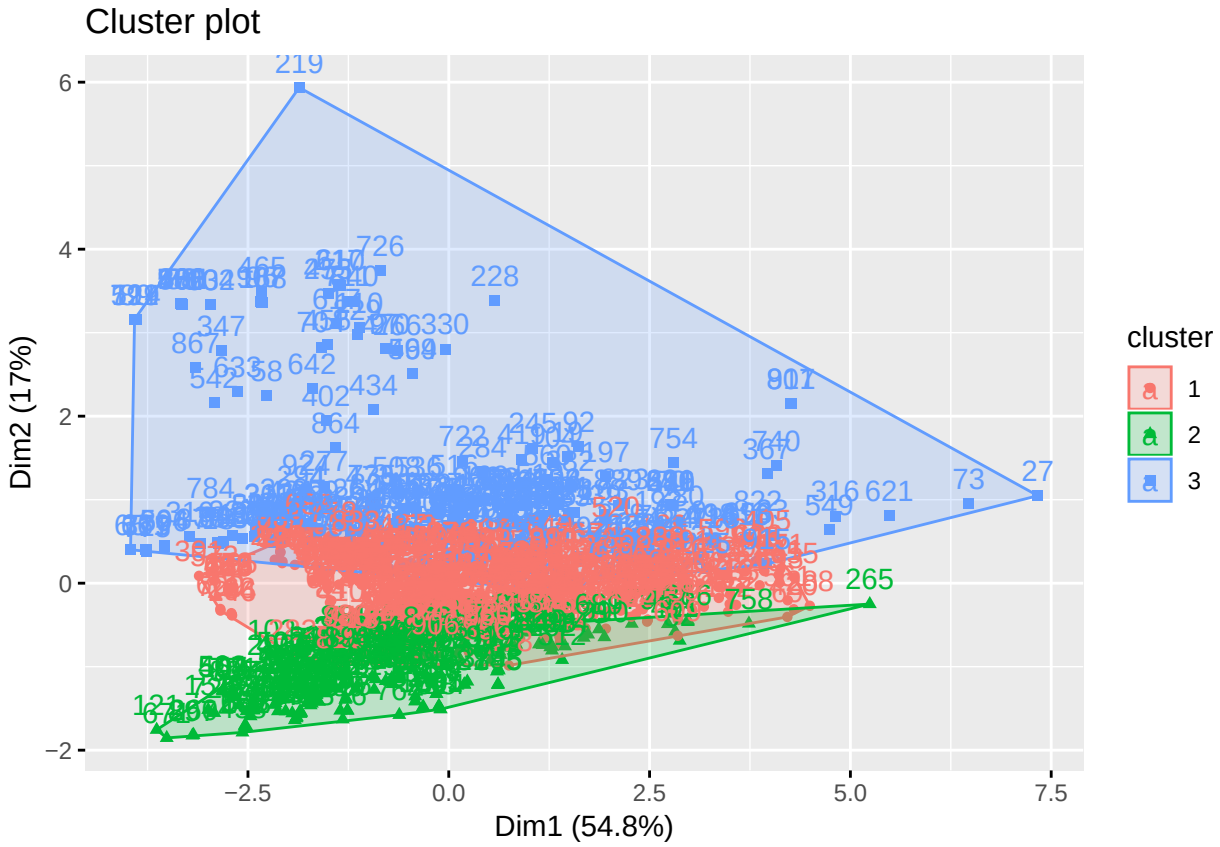
```
## plot clusters
plot(new, col = clarasamp$cluster)
## plot centers
points(clarasamp$centers, col = 1:2, pch = 8)
```



The plot of the clusters does not look great. Aside from comparing longitude with the other variables, the plots have entirely overlapping clusters. This indicates that the CLARA method was unable to find great patterns in the data.

Next, I will use a version of a ggplot to plot the clusters.

```
#plotting clara
factoextra::fviz_cluster(clarasamp)
```

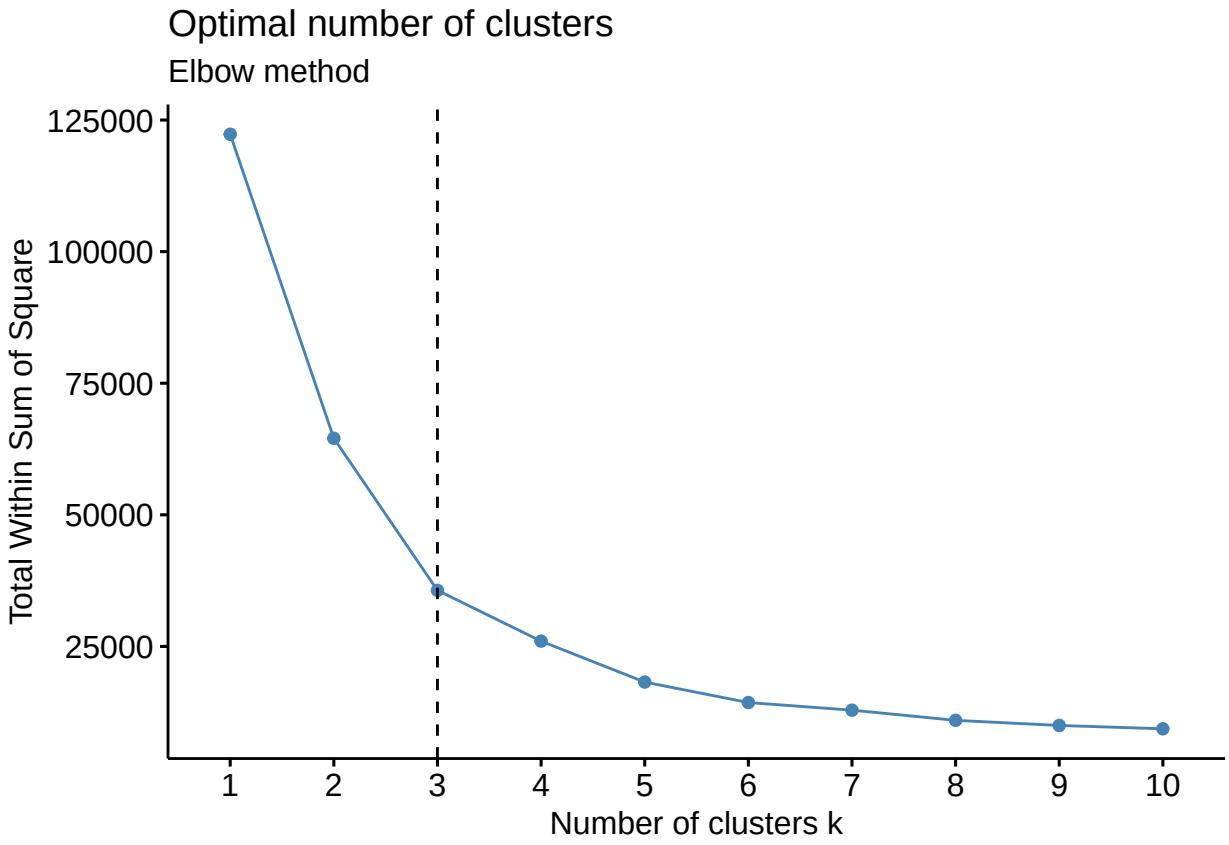


3.3 Evaluation of CLARA

There are multiple ways to determine the effectiveness of CLARA and the quality of its clusters. One of the ways to internally validate the method, is to look at its WSS. If there is a high WSS, it is likely the method did not work very well.

I plotted in the previous section in the Elbow method plot to determine the number of clusters to use. Since I decided to use k=3 clusters, I can now go back and calculate the WSS for the method.

```
elbow<- fviz_nbclust(new, kmeans, method = "wss") +
    geom_vline(xintercept = 3, linetype = 2)+
  labs(subtitle = "Elbow method")
elbow
```



The Elbow method when k=3, shows a WSS to be about 30,000. This is very high, which is a concern when interpreting the cluster results...

3.3.1 Model to Predict Cluster

The CLARA method found three clusters to group the health data. While the WSS value of over 30,000 indicated the clustering may not be very accurate or useful, I want to determine if I can predict the cluster number (1, 2, or 3), given the health indicators. This would be helpful information, say there is a new data observation and I want to categorize it into cluster 1, 2, or 3.

To start this process, I first had to include a variable with cluster number (from the CLARA method) to the original sample of the data set.

```
#adding each data point's cluster #
cluster<- clarasamp$clustering
cluster_data<- cbind(new, cluster)
```

Next, I looked at possible relationships between the health indicator variables and cluster number. To start, I quickly looked at a multivariate linear regression model to predict cluster using all of the possible variables.

```
kitchen_sink<- lm(cluster~., data=cluster_data)
summary(kitchen_sink)
```

```
Call:
lm(formula = cluster ~ ., data = cluster_data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.33635 -0.62036 -0.00065  0.59937  1.48615

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                0.383530   0.376180   1.020  0.30822
Latitude_tri              -0.007281   0.006479  -1.124  0.26141
Longitude_tri             -0.039276   0.002194 -17.905  < 2e-16 ***
poor_or_fair_health       10.089001   1.417808   7.116 2.24e-12 ***
poor_physical_health_days -1.081969   0.088359 -12.245  < 2e-16 ***
physical_inactivity        1.986841   0.730739   2.719  0.00667 **
adult_obesity              0.332835   0.816048   0.408  0.68347
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6793 on 918 degrees of freedom
Multiple R-squared:  0.3765,    Adjusted R-squared:  0.3724
F-statistic: 92.38 on 6 and 918 DF,  p-value: < 2.2e-16
```

According to this model, longitude, poor_or_fair_health, poor_physical_health_days, and physical_inactivity were strong predictors of cluster number. Overall, the model

seemed to fit the data fairly well. The model had a high F-statistic and a low p-value of <2e-16. The adjusted R-squared value was 0.372.

In analyzing this model I realized that latitude and longitude were used as quantitative variables instead of categorical. Since linear and multivariate regression predictive models only use quantitative or categorical variables, I realized that the latitude and longitude (spatial information) would not be helpful.

```
#taking out latitude and longitude
vars <- names(cluster_data) %in% c("Latitude_tri", "Longitude_tri")
cluster_data_new <- cluster_data[!vars]
```

I ran another kitchen sink model, with only the health variables and cluster information.

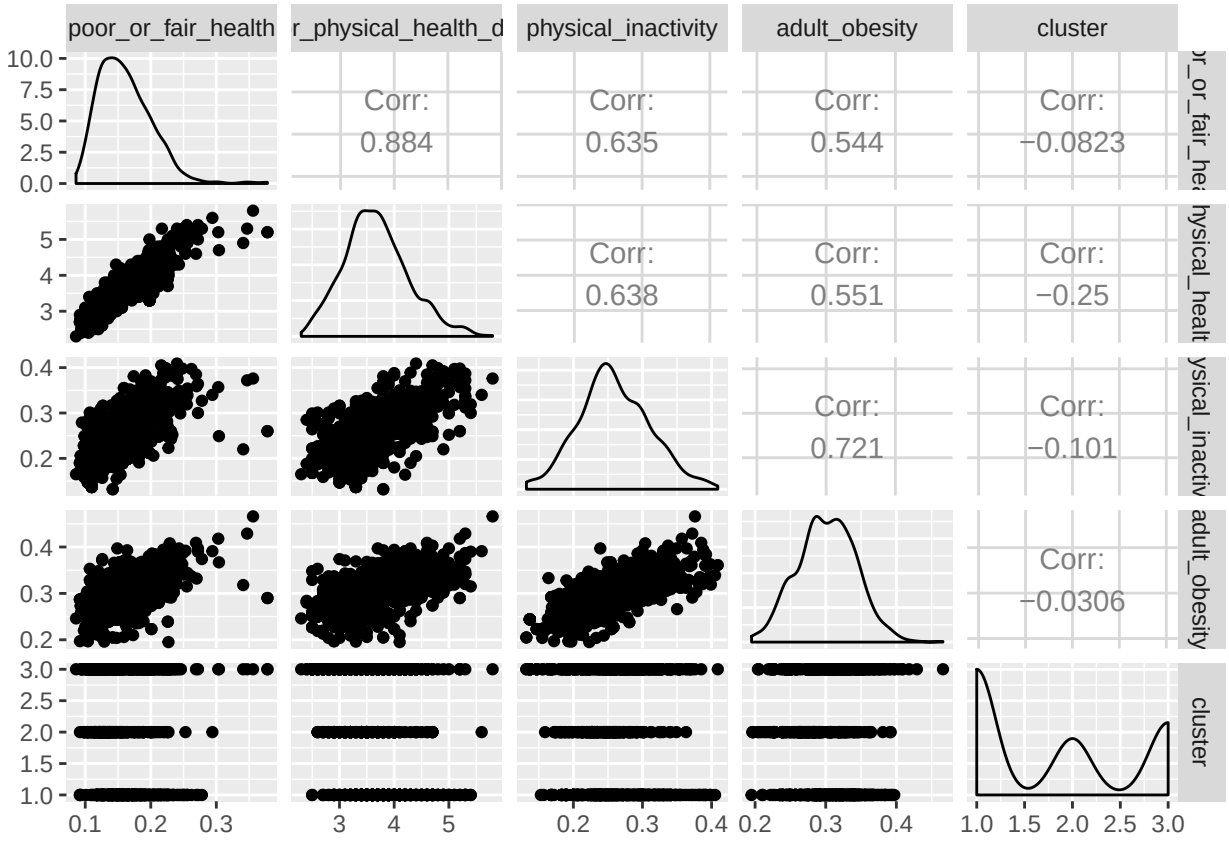
```
new_kitchen_sink<- lm(cluster~., data=cluster_data_new)
summary(new_kitchen_sink)
```

```
Call:
lm(formula = cluster ~ ., data = cluster_data_new)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4582 -0.6918 -0.1393  0.6406  2.2221

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                  3.5366     0.2297  15.399  < 2e-16 ***
poor_or_fair_health         13.0696     1.3988   9.343  < 2e-16 ***
poor_physical_health_days  -1.2089     0.0964 -12.540  < 2e-16 ***
physical_inactivity        -0.9774     0.8086  -1.209  0.22705
adult_obesity               2.8044     0.9247   3.033  0.00249 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7882 on 920 degrees of freedom
Multiple R-squared:  0.1587,    Adjusted R-squared:  0.1551
F-statistic: 43.39 on 4 and 920 DF,  p-value: < 2.2e-16
```

This model had three significant predictors (poor_or_fair_health, poor_physical_health_days, and adult_obesity), a high F-statistic, and a low p-value of <2e-16. The model did not fit the data very well, and had an adjusted R-squared value of 0.155.

Next, I looked at the possible correlations between cluster number and the health indicator variables. I predicted the healthier people (lower scores on the health indicator variables) would be in cluster 1, while the least healthy people (higher scores on health indicator variables) would be in cluster 3. I also predicted the health indicator variables would be highly correlated with each other, considering they all are aiding in predicting one's health.

```
ggpairs(cluster_data_new)
```



The correlation plot shows strong positive correlations between poor_or_fair_health, poor_physical_health_days, physical_inactivity, and adult_obesity, as I had predicted. The highest correlation was 0.884, between poor_physical_health_days and poor_to_fair_health. All of the variables in general show bell-shaped curves with a relatively even shape.

The plots comparing the variables to the cluster number are hard to interpret at first. To start, the poor_to_fair_health versus cluster plot shows that the highest values of poor_or_fair_health are in cluster 3. These look to be possible outliers, but regardless, it confirms the prediction that the unhealthy people (high health variable scores) are in cluster 3.

The poor_physical_health_days versus cluster number and adult_obesity versus cluster number show a couple of observations with high health variable scores in cluster 3 as well. It is again unclear if these points are outliers or not.

In general, the plots show that cluster 2 has the smallest range of health scores, which further confirms that cluster 2 had the highest quality of clustering (the largest silhouette width). In terms of correlation values, cluster number was shows to be sightly negatively correlated with poor_physical_health_days, with a correlation value of -0.25.

I had predicted the correlation to be positive, because the CLARA output revealed cluster 3 to have the most unhealthy people. This would mean the higher the health variable value, the higher the cluster number. Since the correlations are in fact slightly negative, I believe the reason for the higher health value mean score for cluster 3 was probably due to the outliers also shown in the plots.

All of correlations between the health variables and cluster number were negative, indicating that the clustering was not very effective and instead there may be outliers impacting the original analysis of CLARA.

Nevertheless, I will continue to explore possible relationships between health variables and cluster number. Based on the correlation plot, I will explore poor_or_fair_health (because of the plot), poor_physical_health_days (because of the correlation value), and adult_obesity (because of the plot).

I tried numerous combinations of the variables as well as interaction terms, because the variables are so highly correlated.

```
set.seed(2)
#exploring possible relationships between health variables and cluster number

fun1<- lm(cluster~ poor_or_fair_health + poor_physical_health_days + adult_obesity
#low adjusted R-squared (0.155), but significant predictors

fun2<- lm(cluster~ poor_or_fair_health, data= cluster_data_new)
fun3<- lm(cluster~ poor_physical_health_days, data= cluster_data_new)
```

```
fun4<- lm(cluster~ adult_obesity, data= cluster_data_new)
#low adjusted R-squared, highest of the 3 functions was 0.06

fun5<- lm(cluster~ poor_or_fair_health + poor_physical_health_days + adult_obesity + poo
#added an interaction, raised the adjusted R-squared to 0.165

fun6<- lm(cluster~ poor_or_fair_health + poor_physical_health_days + adult_obesity + poo
#tried a different interaction, about the same adjusted R-squared

fun7<- lm(cluster~ poor_or_fair_health + poor_physical_health_days + adult_obesity + poo
#last combination of an interaction, highest adjusted R-squared yet (0.175)!
#only predictor not significant was poor_or_fair_health

fun8<- lm(cluster~  poor_physical_health_days + adult_obesity + poor_or_fair_health:adul
#dropped poor_or_fair_health, about the same adjusted R-squared (0.174)

fun9<- lm(cluster~ poor_physical_health_days + adult_obesity + poor_physical_health_days
#tried a different interaction, low adjusted R-squared (0.0958)

fun10<- lm(cluster~ poor_physical_health_days + adult_obesity + poor_or_fair_health:poor
#tried last combination of interaction, adjusted R-squared= 0.166
```

Most of the model had significant predictors; however, the R-squared values were small; indicating that the models did not fit the data very well.

In comparing adjusted R-squared values and the number of predictors used, the best model ended up being:

```
summary(fun8)
```

```
Call:
lm(formula = cluster ~ poor_physical_health_days + adult_obesity +
    poor_or_fair_health:adult_obesity, data = cluster_data_new)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5661 -0.6691 -0.1815  0.5856  2.2755
```

```
Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                        5.73676    0.36071  15.904  < 2e-16 ***
poor_physical_health_days         -1.24651    0.08954 -13.921  < 2e-16 ***
adult_obesity                     -4.81191    1.07188  -4.489 8.05e-06 ***
adult_obesity:poor_or_fair_health 42.15924    4.03943  10.437  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7795 on 921 degrees of freedom
Multiple R-squared:  0.1763,    Adjusted R-squared:  0.1736
F-statistic: 65.71 on 3 and 921 DF,  p-value: < 2.2e-16
```

This model used three predictors and had about the same $R_squared$ value as the previous model, with one less predictor.

Conclusion

If we don't want Conclusion to have a chapter number next to it, we can add the `{.unnumbered}` attribute. This has an unintended consequence of the sections being labeled as 3.6 for example though instead of 4.1. The LaTeX commands immediately following the Conclusion declaration get things back on track.

More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Appendix A

The First Appendix

This first appendix includes all of the R chunks of code that were hidden throughout the document (using the `include = FALSE` chunk tag) to help with readibility and/or setup.

In the main Rmd file:

```
# This chunk ensures that the acstats package is
# installed and loaded. This acstats package includes
# the template files for the thesis and also two functions
# used for labeling and referencing
if(!require(devtools))
  install.packages("devtools", repos = "http://cran.rstudio.com")
if(!require(acstats)){
  library(devtools)
  devtools::install_github("Amherst-Statistics/acstats")
}
library(acstats)
```

In :

```
# This chunk ensures that the acstats package is
# installed and loaded. This acstats package includes
# the template files for the thesis and also two functions
# used for labeling and referencing
if(!require(devtools))
  install.packages("devtools", repos = "http://cran.rstudio.com")
if(!require(dplyr))
    install.packages("dplyr", repos = "http://cran.rstudio.com")
if(!require(ggplot2))
    install.packages("ggplot2", repos = "http://cran.rstudio.com")
```

```r
if(!require(acstats)){
  library(devtools)
  devtools::install_github("Amherst-Statistics/acstats")
}
```

Appendix B

The Second Appendix, for Fun

References

Angel, E. (2001a). *Batch-file computer graphics : A bottom-up approach with quicktime*. Boston, MA: Wesley Addison Longman.

Angel, E. (2001b). *Test second book by angel*. Boston, MA: Wesley Addison Longman.

Ng, R. T., & Han, J. (2000). *Efficient and effective clustering methods for spatial data mining*. San Francisco, CA: Morgan Kaufmann Publishers Inc.