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 longtitude, 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 amoung objects. This can aid us in allocating resources to areas that need them most, discovering changes over time, and categorizing new objects.

0.2 Big Picture Analyzing Spatial Data Algorithms

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

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

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

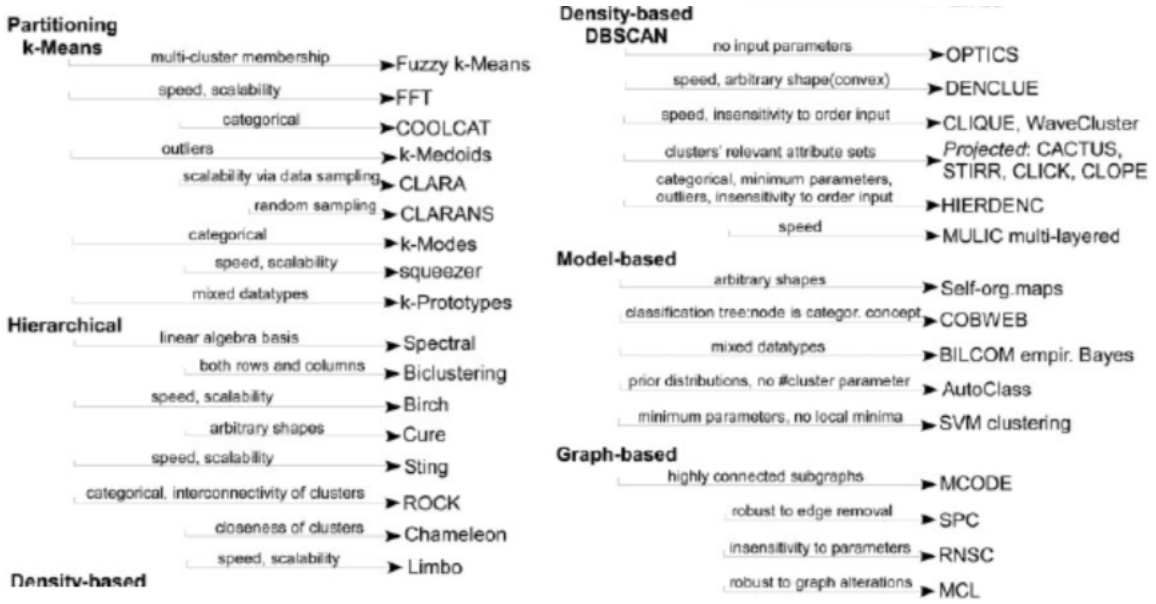


Figure 1: Clustering Methods

0.2.1 Clustering

Two categories of how to analyze spatial data include classification and clustering. Classification groups data items together into categories according to their properties. It is considered supervised classification because it needs a training dataset to fit the classification model and a test dataset to evaluate the model.

Clustering is organizing 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 is helpful in finding patterns and similarities/differences between data points and groups; however it can be quite subjective, as we will discuss later on in the project.

Chapter 1

How to Cluster

There are many factors to consider when chosing 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 (size of 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

Two of the main types of clustering are partitioning and hierarchial.

Hierarchial clustering organizes data items into a hierarchy with a sequence of nested partions or groupings [Rec 1, p. 405]. There is the bottom-up approach: There is also the top-down approach:

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]. Partitioning clustering algorithms classifies 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 How to Create Clusters: *K*-Means vs *K*-Medoids

K-means algorithm and k-medoid algorithm are two examples of partitioning algorithms. They both ise iterative processes to find K clusters; however, they use different

ways to represent these clusters.

1.2.1 *K*-Means (maybe don't include this)

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 eachother.
2. Assign each object 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].

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.

same steps as K-means except BLANK (p. 6 in Rec 2) steps in Rec 5, p. 19 for k-medoids

Rec 5 p. 18-19 ***

```
#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)
```

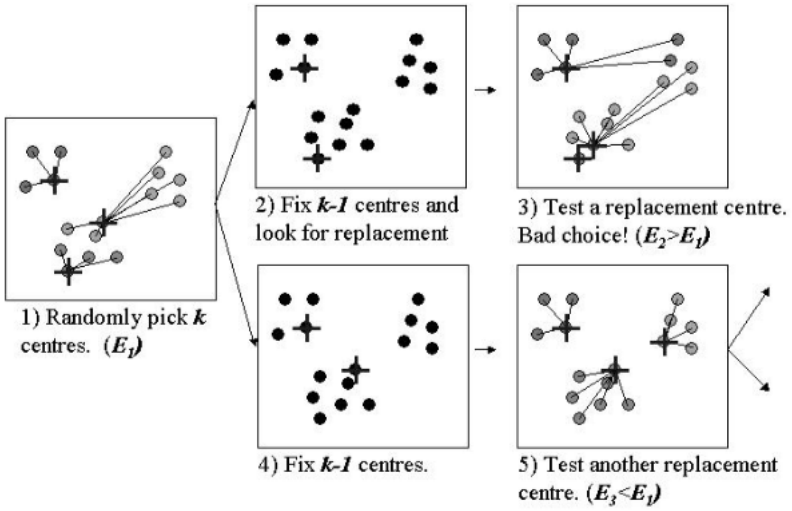


Figure 1.1: CLARANS searching for a better solution

1.3 How to Choose *K*

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.

The steps for this algorithm are as follows:

1. Compute the clustering algorithm for different values of k (i.e. k from 1 to 10).
2. For each k, calculate the total WSS.
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.

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 algoritm for different values of k (i.e. k from 1 to 10).
2. For each k, calculate the average silhouette of observations.
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.

–> datanovia website

Chapter 2

Spatial Clustering Methods

2.1 PAM

Partitioning Around Medoids (PAM) is the most common realisation of k-medoid clustering.

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 total complexity of PAM in one iteration is **formula:

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

(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. Works best for n= 100, 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.

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].

2.3 CLARANS (?)

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

Data came from Stat 495 final project. (use info from project...). Needed a sample of 1000...

Importing 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.
```

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

Picking variables to focus on–> expanding conclusions from Stat 495 project

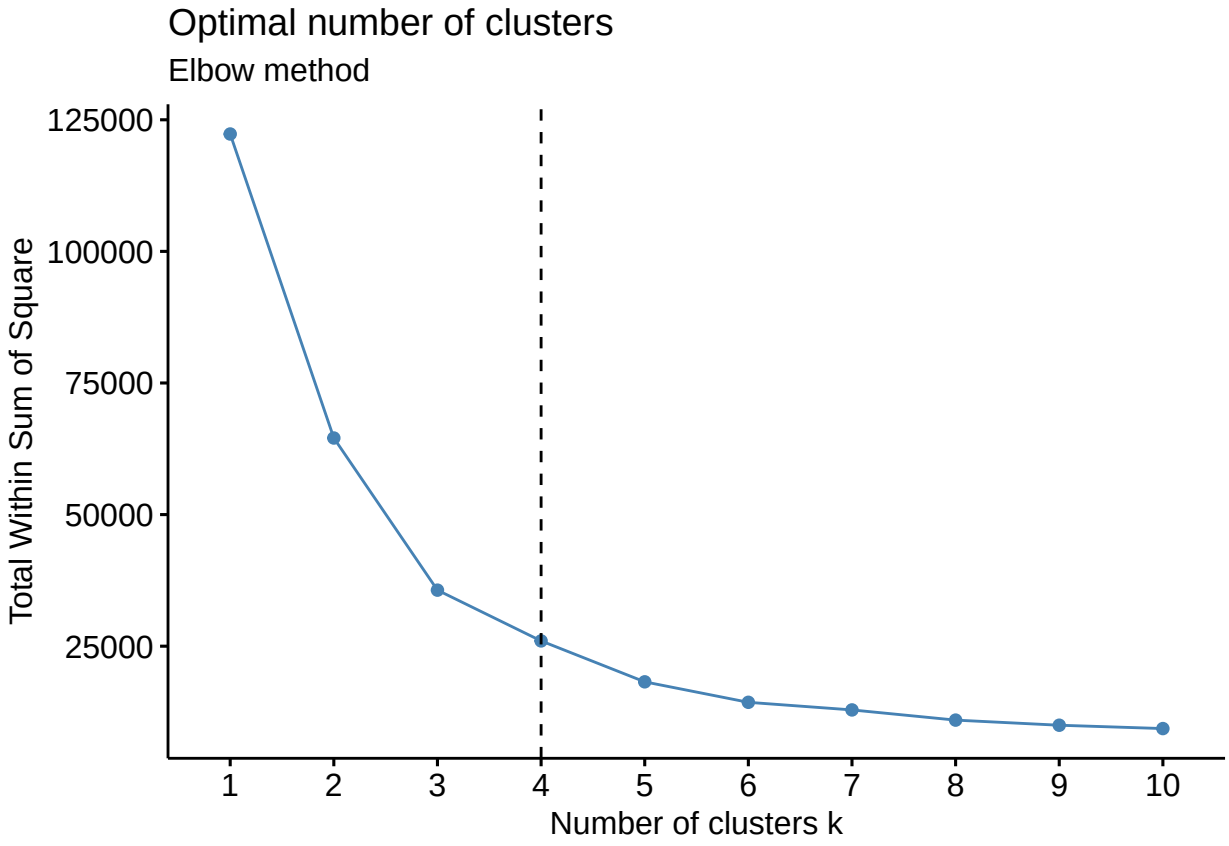
```
#only keeping the variables I want to look at
myvars <- c("Latitude_tri", "Longitude_tri", "poor_or_fair_health", "poor_physical_healt
smallsample <- mysample[myvars]
```

3.2 Applying CLARA

Step 1: finding k

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

```
#library(factoextra)
#library(NbClust)
#library(ggplot2)
new<- na.omit(smallsample)
# Elbow method
fviz_nbclust(new, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```



Step 2: Run CLARA function

```
#new<- na.omit(smallsample)
#library(cluster)
## run CLARA
clarasamp <- clara(new[1:6], 4)
```

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

```
Call:    clara(x = new[1:6], k = 4)
Medoids:
     Latitude_tri Longitude_tri poor_or_fair_health
[1,]      38.6364      -83.6929               0.200
[2,]      40.3973      -75.9357               0.165
[3,]      36.1335      -96.0532               0.196
[4,]      45.4342     -123.0000               0.110
     poor_physical_health_days physical_inactivity adult_obesity
[1,]                       4.4               0.299         0.283
[2,]                       3.7               0.245         0.308
[3,]                       4.6               0.353         0.355
[4,]                       3.3               0.137         0.244
Objective function:  4.691022
Clustering vector:   int [1:925] 1 2 1 3 1 3 1 1 1 1 1 3 1 2 1 3 3 3 ...
Cluster sizes:           453 195 230 47
Best sample:
 [1]   5  11  24  86 139 149 162 175 177 192 208 224 242 285 306 311 316
[18] 353 361 370 389 400 404 410 429 468 471 478 489 506 589 679 691 703
[35] 719 726 736 741 800 811 815 818 877 882 883 895 902 918

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

```
summary(clarasamp)
```

```
Object of class 'clara' from call:
 clara(x = new[1:6], k = 4)
Medoids:
     Latitude_tri Longitude_tri poor_or_fair_health
[1,]      38.6364      -83.6929               0.200
[2,]      40.3973      -75.9357               0.165
[3,]      36.1335      -96.0532               0.196
```

```
[4,]      45.4342     -123.0000                0.110
     poor_physical_health_days physical_inactivity adult_obesity
[1,]                       4.4               0.299         0.283
[2,]                       3.7               0.245         0.308
[3,]                       4.6               0.353         0.355
[4,]                       3.3               0.137         0.244
Objective function:   4.691022
Numerical information per cluster:
     size  max_diss  av_diss isolation
[1,]  453 13.228882 4.578281  1.656594
[2,]  195  8.392191 2.971767  1.050917
[3,]  230 14.554453 6.249473  1.153918
[4,]   47 42.497226 5.284278  1.489171
Average silhouette width per cluster:
[1] 0.2863797 0.6457187 0.4655863 0.9673973
Average silhouette width of best sample: 0.4306859

Best sample:
 [1]   5  11  24  86 139 149 162 175 177 192 208 224 242 285 306 311 316
[18] 353 361 370 389 400 404 410 429 468 471 478 489 506 589 679 691 703
[35] 719 726 736 741 800 811 815 818 877 882 883 895 902 918
Clustering vector:
  [1] 1 2 1 3 1 3 1 1 1 1 1 3 1 2 1 3 3 3 3 2 1 3 2 1 1 3 3 2 2 1 1 2 3 1 3
 [36] 1 1 1 3 3 1 1 1 1 1 1 1 1 1 2 1 1 3 1 1 1 1 4 1 1 1 1 2 2 3 3 1 2 1 3
 [71] 1 1 3 3 1 3 1 4 1 2 1 2 4 1 1 3 1 2 4 3 3 3 1 1 3 4 4 1 2 2 1 2 3 3 1
[106] 1 1 3 3 4 1 3 3 4 2 1 2 3 2 3 2 2 1 3 1 2 1 1 1 3 2 4 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 4 3 1 1 3 1 3 3
[176] 1 3 1 1 4 2 1 1 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 4 2 4 3 1 1 3 2 1 1 3 4 3 2 2 1 1 1 3 3 1 1 3 1 1 2 1 2 3
[246] 2 1 1 3 1 1 3 4 2 1 1 2 3 1 2 1 1 1 3 1 4 2 3 2 1 3 2 2 3 1 1 3 2 1 3
[281] 1 1 3 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 4 2 1 1 1 3 1 2 1 2 2 1 3 1 1 3 3 4 3 1 1
[351] 1 1 1 1 1 3 1 3 1 1 1 3 3 1 2 1 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 4 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 1 1 4 4 1 1 3 1 2 1 1 1 3 1 2 1 1 2 1 3 1 1 2 4
[456] 1 1 2 1 1 1 2 1 1 4 3 1 1 4 4 1 1 3 1 4 1 1 1 1 3 4 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 3 2 1 3 2 3
[526] 3 3 1 4 1 1 2 2 2 1 3 2 1 1 1 3 4 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 4 1 1 3 3 2 1 1 3 2 2 1 1 2 1 2 1 1 1 1 3 2 1 3 4 1
[596] 3 1 1 2 1 3 2 3 4 1 3 1 1 1 4 3 1 2 1 1 3 4 2 1 2 3 2 1 2 3 1 1 2 1 2
[631] 1 1 4 3 1 2 3 1 1 3 1 4 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 3 2 2 1 1 2 2 3 2 1 1 1 2 1 3 3 1
[701] 1 1 1 4 2 1 3 1 1 2 4 3 1 1 1 3 3 1 1 1 1 3 1 1 2 4 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 4 2 1 1 3 1
[771] 2 1 1 1 3 1 3 3 1 1 3 3 1 3 1 2 1 1 3 1 4 1 1 1 1 3 2 2 2 4 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 4 2 1 2 3 1 1 3 2 1 1 4
[841] 2 2 2 2 2 1 3 1 2 1 2 4 3 3 1 1 3 1 1 1 1 1 2 3 3 2 4 1 2 1 1 1 1 2 1
[876] 1 3 1 1 1 3 3 1 1 1 3 1 1 3 1 1 3 1 3 1 3 2 2 2 3 1 2 2 2 1 1 4 1 2 1
[911] 3 1 3 2 3 3 1 1 2 1 3 1 1 3 1
```

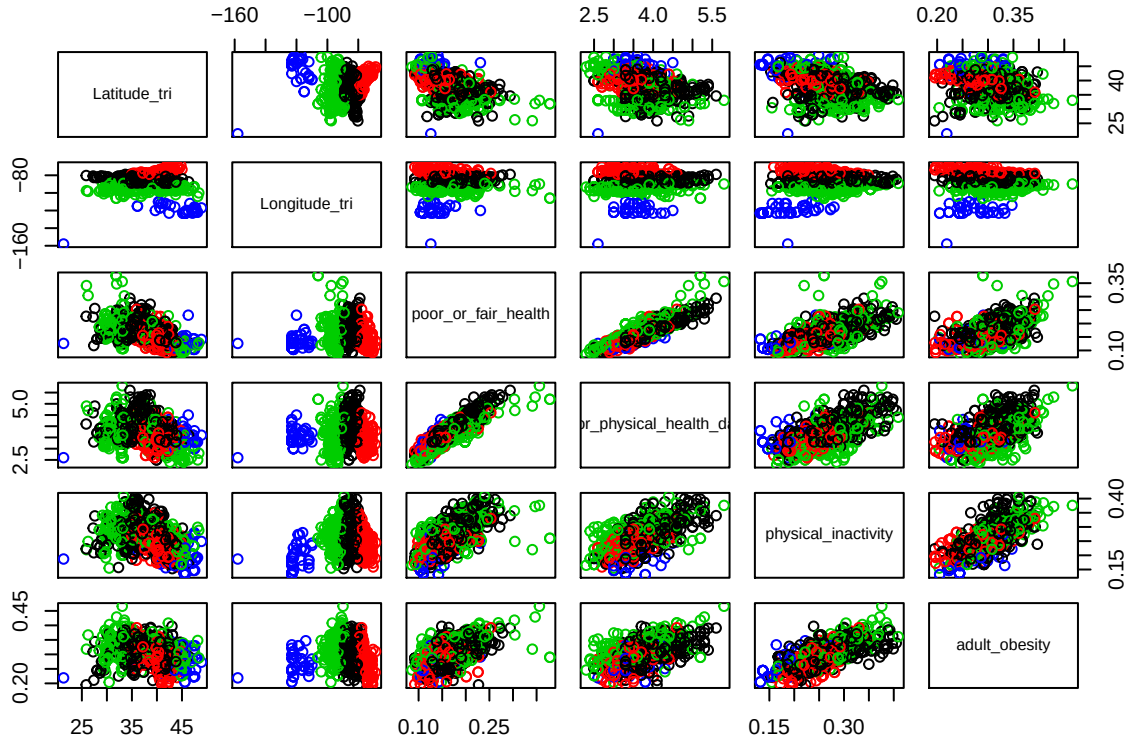
Silhouette plot information for best sample:

	cluster	neighbor	sil_width
11	1	2	0.51160188
703	1	2	0.50784463
149	1	3	0.50590891
208	1	2	0.48693148
162	1	3	0.47468187
471	1	2	0.44956360
410	1	2	0.42362707
389	1	2	0.42157038
5	1	2	0.41457596
719	1	2	0.41289559
468	1	2	0.37020730
361	1	2	0.35061984
306	1	2	0.34569720
285	1	2	0.34506744
818	1	3	0.30770633
506	1	3	0.21671937
589	1	2	0.20444489
918	1	2	0.19851903
736	1	3	0.18549695
478	1	2	0.16554915

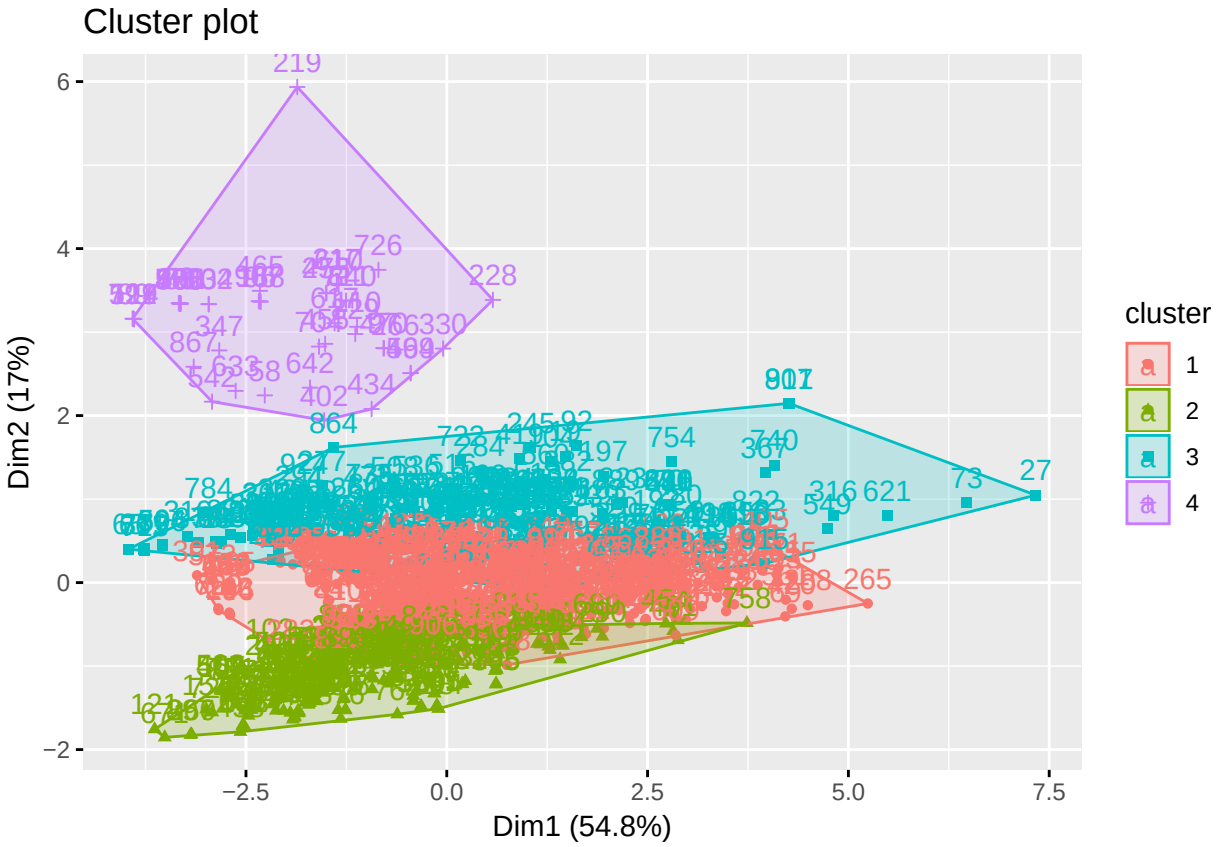
```
24        1        2  0.15565598
311       1        2  0.04768370
353       1        3  0.03173421
883       1        2 -0.12930919
895       1        2 -0.24550081
224       2        1  0.76643863
679       2        1  0.75957431
902       2        1  0.74802681
404       2        1  0.74508892
242       2        1  0.73417881
400       2        1  0.73283435
741       2        1  0.61737358
811       2        1  0.47833894
815       2        1  0.43844208
429       2        1  0.43689016
691       3        1  0.62256512
370       3        1  0.62175433
489       3        1  0.61988506
882       3        1  0.60671130
139       3        1  0.58178390
177       3        1  0.55053388
175       3        1  0.43464488
86        3        1  0.29525285
192       3        1  0.28445790
316       3        1  0.27777790
877       3        1  0.22608234
800       4        3  0.96743579
726       4        3  0.96735884

1128 dissimilarities, summarized :
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0507  6.3776 10.6340 12.8320 15.7130 51.8530
Metric :  euclidean
Number of objects : 48

Available components:
```

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

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




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



3.3 Evaluation of CLARA

3.3.1 Model to Predict Cluster

First, had to include a cluster variable in the original data set, using the data provided by the CLARA function.

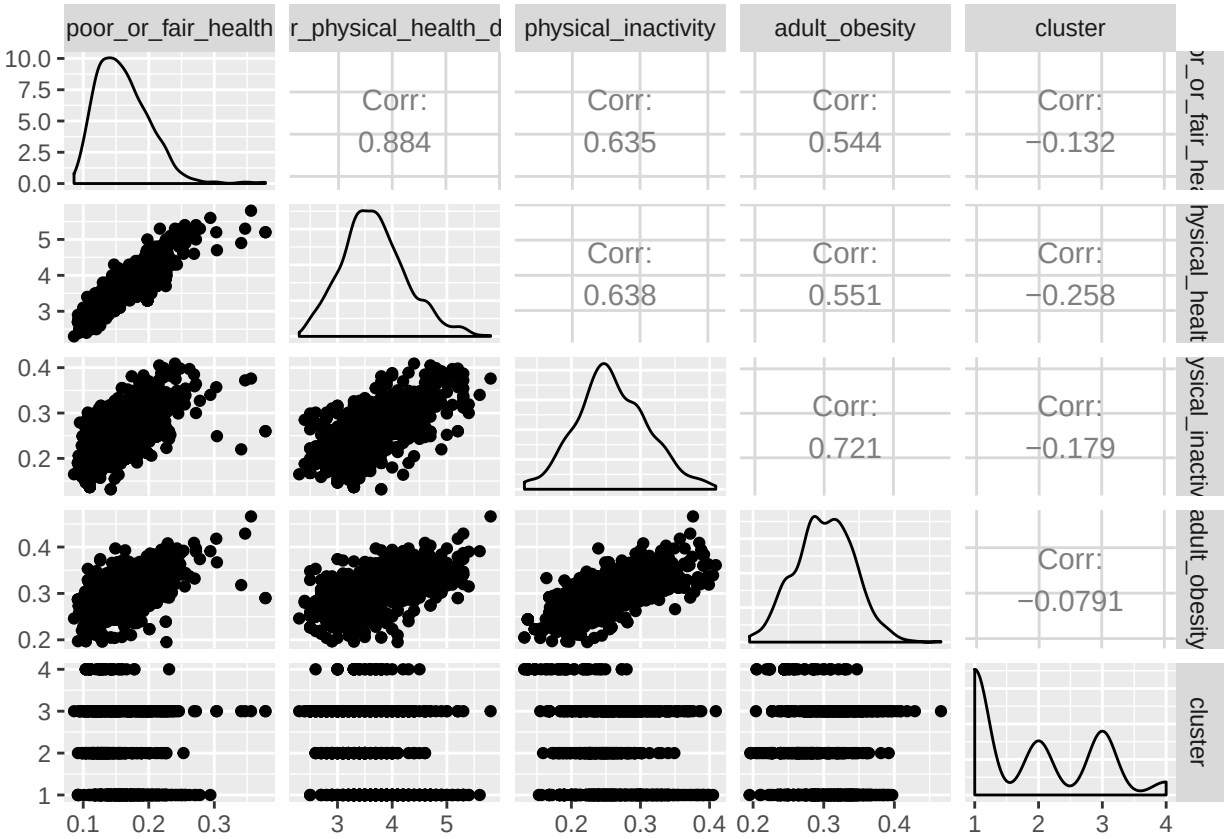
```
#adding each data point's cluster #
cluster<- clarasamp$clustering
cluster_data<- cbind(new, cluster)
```

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

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

```
Residuals:
    Min      1Q  Median      3Q     Max
-2.3742 -0.6013  0.0600  0.6334  1.6092

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                -1.373221   0.386769  -3.550 0.000404 ***
Latitude_tri                0.004183   0.006661   0.628 0.530140
Longitude_tri              -0.055900   0.002255 -24.786  < 2e-16 ***
poor_or_fair_health         8.194392   1.457716   5.621 2.51e-08 ***
poor_physical_health_days -0.913749   0.090846 -10.058  < 2e-16 ***
physical_inactivity         1.395680   0.751308   1.858 0.063536 .
adult_obesity              -0.419444   0.839018  -0.500 0.617249
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6984 on 918 degrees of freedom
Multiple R-squared:  0.4751,    Adjusted R-squared:  0.4717
F-statistic: 138.5 on 6 and 918 DF,  p-value: < 2.2e-16
```

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

ggpairs(cluster_data_new)
```




```
fun1<- lm(cluster~ poor_physical_health_days, data= cluster_data_new)
summary(fun1)
```

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

Residuals:
    Min      1Q  Median      3Q     Max
-1.3539 -0.8070 -0.1857  0.8144  2.4875

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                3.40573    0.19278  17.667  < 2e-16 ***
poor_physical_health_days -0.42072    0.05183  -8.118 1.51e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.9288 on 923 degrees of freedom
Multiple R-squared:  0.06664,   Adjusted R-squared:  0.06563
F-statistic:  65.9 on 1 and 923 DF,  p-value: 1.509e-15
```

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.