Unsupervised Learning Assignment 2

Tiffany Woodley

August 2019

Contents

1 Introduction

1.1 Overview of Problem

As the world leverage's data across multiple industries, with great success, the energy sector holds great potential for this application. With the recent drive for more renewable energy resources, and with battery prices decreasing, there opens up room for applications such as demand management, smart-grids, peak shaving etc.

With these new resources changing our demand chain, larger power suppliers will also need a better understanding of how their power generation would need to be developed in order to accommodate these new demands.

Therefore, there are so many areas in the energy sector where optimization of energy consumption can be useful. However, better understanding of demand requirements are still needed in order to advance this application.

To better understand the demand requirements, the average load profile of a day is needed. Unfortunately, demand requirements can change based on the day. This assignment therefore aims to try find clusters of these days throughout the year to see if meaningful insights may be found from grouping the data.

If appropriate clusters are found, this will be a very useful finding as now when looking ahead to what a day is expected to be, the cluster average can be used rather than the overall average, bringing a greater standard of accuracy.

1.2 Data Set

The data-set used was the load meter data,in kVA, on a 30 minute time interval from an industrial warehouse. The data spans over approximately a year (368 days), and is broken up into observations of each day. The format of the data can be observed in Table 1 below. These days are then clustered to see if there are any patterns between days that can be found.

Some of the days had missing data (maintenance, metering issues etc), and so these days were removed so that they did not skew the cluster averages, leaving 250 days in the data-set.

Day	0:00	0:30	1:00	...	22:30	23:00	23:30
01/01/2016	501	504	498	...	520	505	507
...	...	...	...	...	...	...	...
31/12/2016	432	435	439	...	407	411	414

Table 1: Input Data Format

Each observation plotted will resemble a load profile similar to Figure 1 below

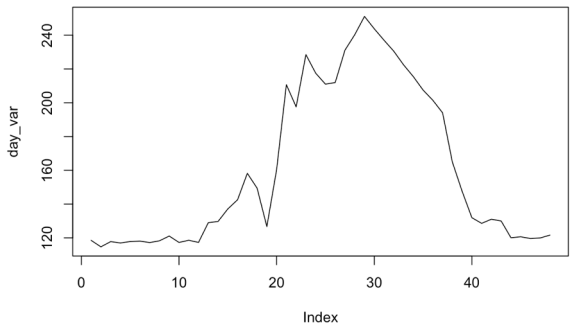


Figure 1: Example of an Observation

2 Self Organizing Maps

Self Organizing Maps (SOMs) were applied to the data for the purpose of visualising the topology of the data, in a lower dimension, which helped visualize/find clusters. The data was not normalised/scaled, as the data was all measured in the same unit.

2.1 Fitting the model

There were 250 observations in the data, a 9 by 9 dimension (81 neuron) SOM was fitted to the data.

Figure 2 below shows how the distance between the observations and their best matching unit decreases, as the training iterations increase. The distance plateaus at around 2500 iterations, and so 3000 iterations were deemed sufficient.

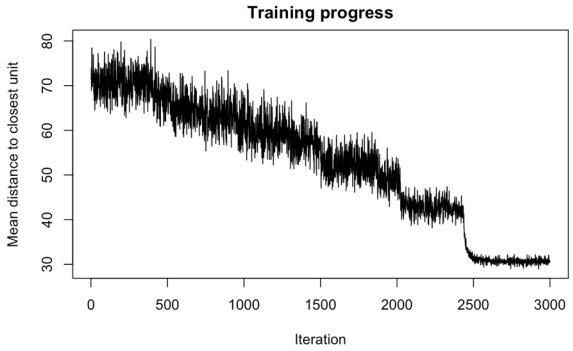


Figure 2: SOM Training Iterations

A counts plot is a useful visualization which helps determine the size your SOM should be. Too many blank spaces would indicate that the map size is too big, whilst too many observations per neuron would point towards the map being too small. As a rule of thumb, 5-10 observations should be mapped to each neuron.

The counts plot shown in Figure 3 shows how the observations we fitted to our model have been mapped. There are 4 neurons without any observations mapped to them, but the rest of the neurons have not been overloaded with observations. With a maximum of 14 observations per a neuron. Most of the neurons have under 10 observations mapped to them. If the grid size is dropped to 8 by 8 neurons, some of the neurons start becoming overcrowded, for this reason the 9 by 9 grid was chosen despite the 4 empty neurons.

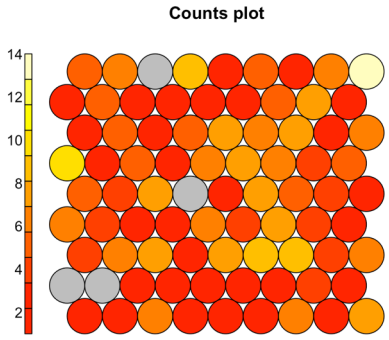


Figure 3: Som Counts Plot

The neighbours distance plot in Figure 4 does not give us too much information in this case, except for the the fact that observations allocated to the bottom left are probably outliers.

The neurons in the bottom left of the neighbours distance plot are quite far from the rest of the neurons, because of this big difference the scale of the distances is thrown off a bit, and its impossible to see the differences between the rest of the neurons as they all lie within the range of the red, which is quite large.

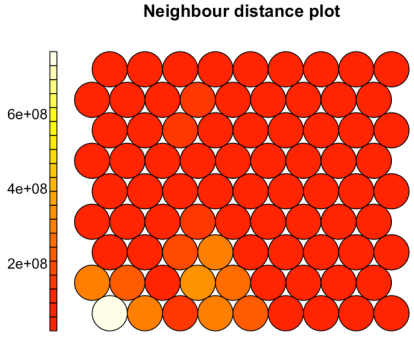


Figure 4: SOM Neighbour Plot

2.2 Visualization of weights in fitted model

The codes plot in Figure 5 shows how weight vectors vary across the grid. Since all the variables were measured in the same unit, and ordered by time, you can see the load profile that would be associated with each neuron and see how the profile changes depending on the area on the map. For instance the load profiles on the left are smaller than on the right, indicating that days mapped to the right are lower demand use days. Days on the bottom of the grid have a sharper dip in the early morning, indicating that something was shut down and restarted at this time on these days.

This plot also gives us more insight into what was happening in the bottom left of the neighbours plot, the load profile associated with this neuron was flat, which is definitely and outlier.

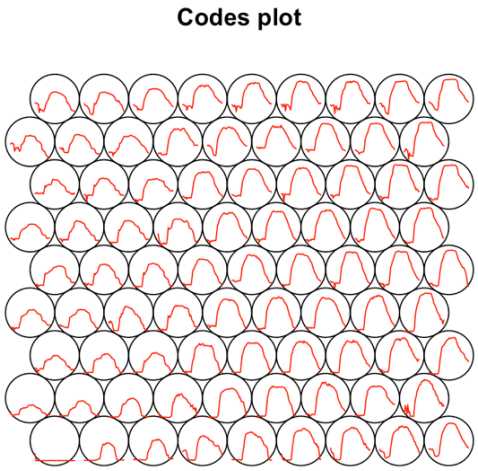


Figure 5: SOM Codes plot

The small maps in the following figures show how each input varies across the neurons in the SOM. Since each variable represents a time instance on the day, it can be seen what time values contribute to the different neurons. Neurons on the top have higher load demands in the early morning. Neurons in the bottom right corner have lower load demands in the middle of the day. These can be very useful once clusters have been found to identify the reason of why they differ from other clusters. As this will allow us to know how to manage the loads in the specific clusters better.

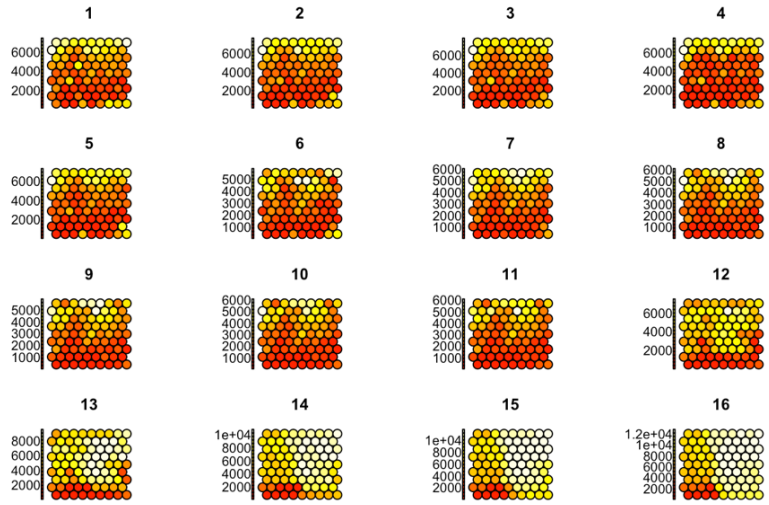


Figure 6: SOM Variable Contribution Plot(1-16)

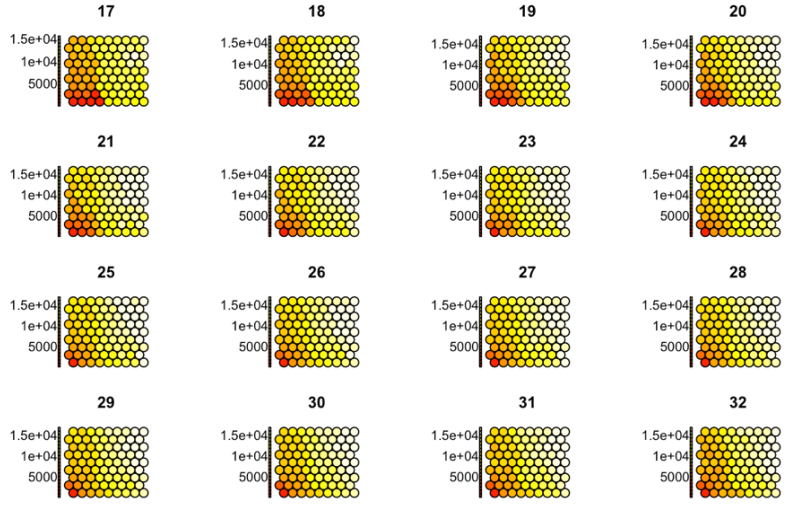


Figure 7: SOM Variable Contribution Plot(17-32)

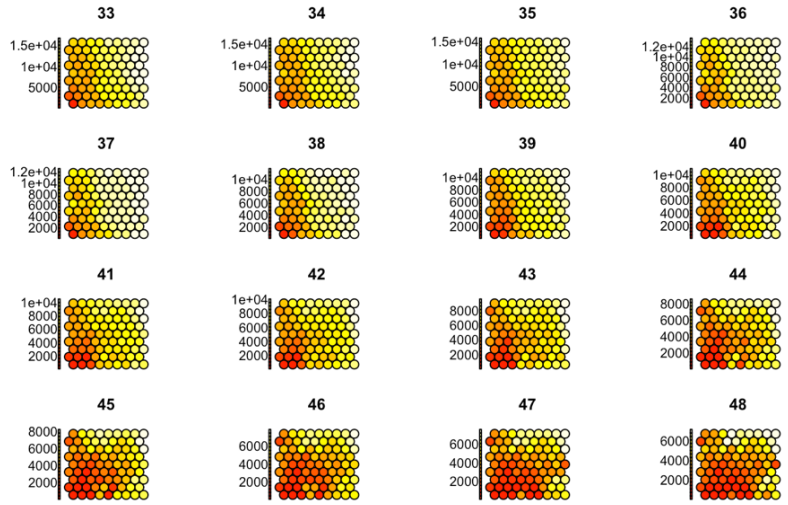


Figure 8: SOM Variable Contribution Plot(33-48)

2.3 Clustering

Now that the SOM has been fitted, these weights are going to be used to find clusters between the neurons and in turn the observations.

2.4 Choice of clusters

K-means was used to help identify the amount of clusters present in the data, by using the weighted sum of squares, this can be seen in Figure 9. From this graph, 5 clusters were chosen as this is where the curve has an "elbow". From this point, hierarchical clustering using complete linkage was used in order to find the 5 clusters.

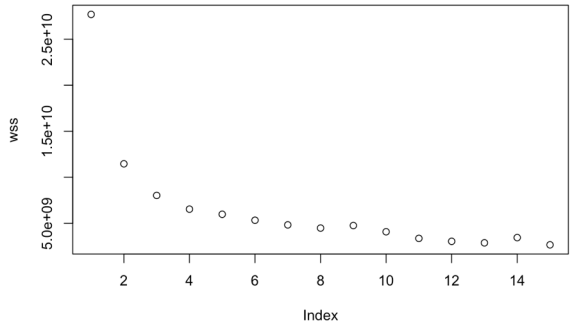


Figure 9: WSS vs number of clusters

The clusters are shown in Figure 10 by color on the grid.

The blue cluster comprised of 1s and 7s, which relate to Sundays and Saturdays and so it has clustered weekends together, which makes sense as there will be less demand at the warehouse on these days. The blue cluster is on the left hand side of the map which makes sense, as from the codes plot seen earlier, the neurons on the left had smaller load profiles.

There are only a few allocated to the orange and red groups, further suggesting that these were anomalies/outliers, and it should be investigated what happened on these days to see if they can be explained.

The general mid week days have been allocated to the yellow and green group. The fact that their are two clusters for the mid week groups allows us to see what differentiates days within the weekdays.

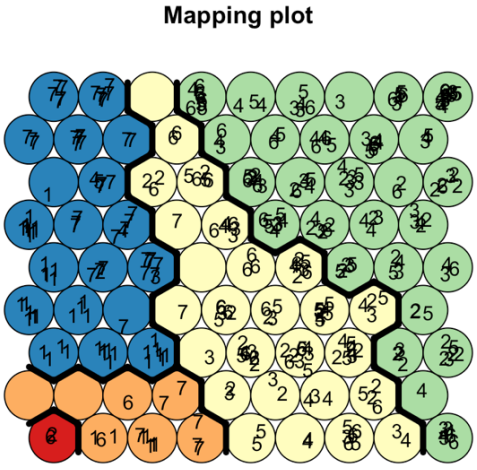


Figure 10: SOM Mapping of Clusters

2.4.1 Viewing clusters over the year

The visualization in Figure 11 below, allows us to see how the clusters are situated over the year. The orange cluster is explained by the Christmas holidays. The weekend days have been clustered into the blue cluster with some inter-week days also in the group, these days are likely to be holidays were the warehouse was shut down.

There are only two days within the red cluster, since there is so few it is likely to be an anomaly and should be looked into.

The weekdays are split into two clusters, yellow and green. When looking at the profiles plotted across the year in Figure 12 we can see the yellow days relate to the days on the profiles where the maximum demand is lower. This suggests towards seasonality in the data which can be further looked into.

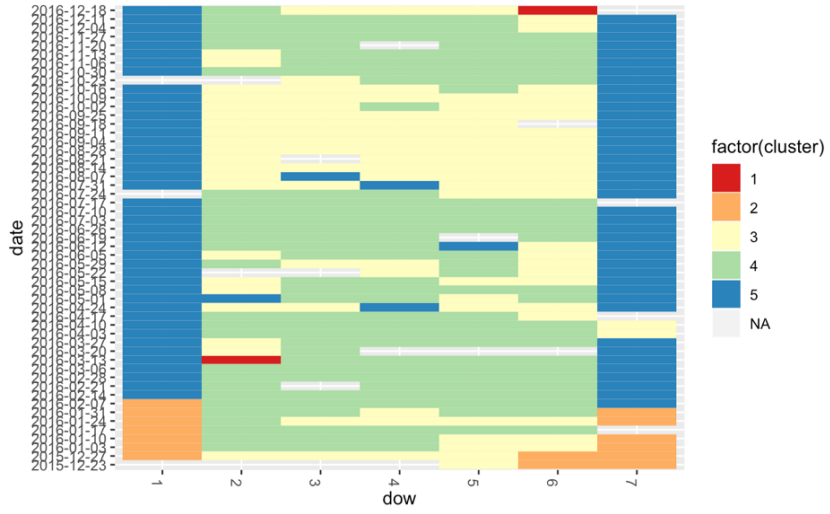


Figure 11: SOM Clusters over Year

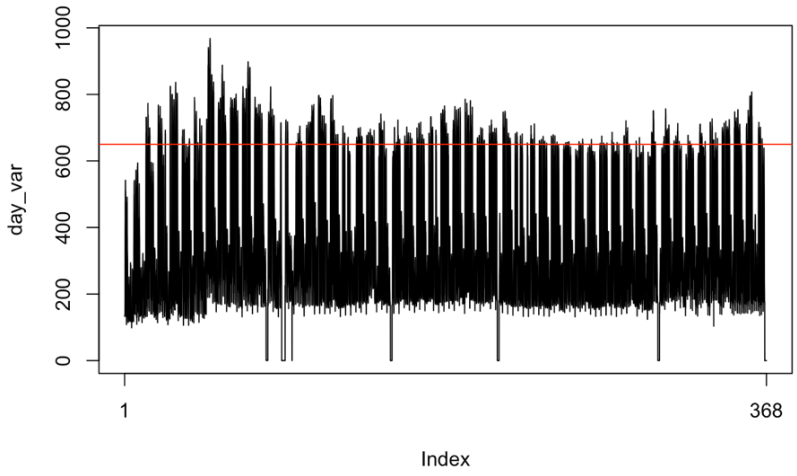


Figure 12: Half hour time interval load from day 1 to 368

2.5 Viewing each cluster

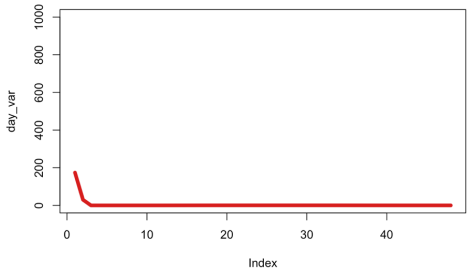
Figure 13 shows the profiles in the different clusters, and their average. From this we can get a better understanding of what is happening in each cluster.

Cluster 1 has a flat load profile which indicates that the data was being stored as zero values instead of missing values, this can become a problem if systems are put in place that follow the load to control systems and so should be looked into so that it can be remedied.

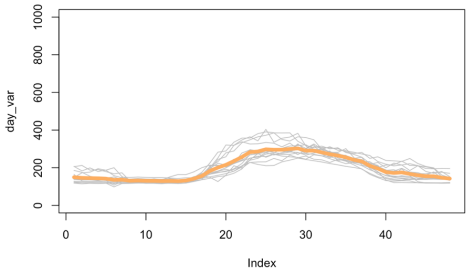
Cluster 2 shows a very small load profile which is expected over the Christmas holiday period.

Cluster 3 and 4 show the general weekday load profiles, with cluster 3 having a lower maximum demand than cluster 4, this makes sense based on what we saw in the seasonality of the data.

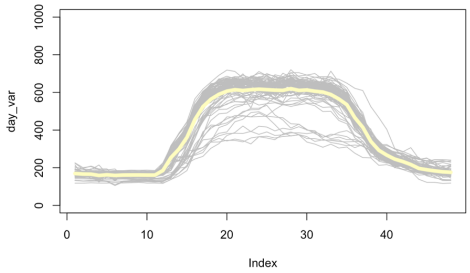
Cluster 5 shows the weekends and holidays, this load profile is a lot smaller than the weekdays, since it differs from the Christmas holiday period it indicates that there is some small activity over general weekends/holidays.



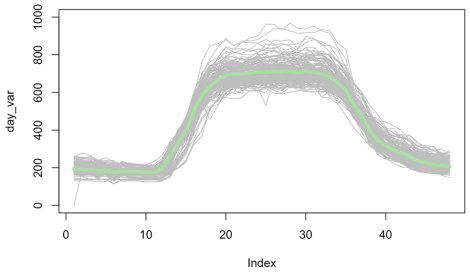
(a) Cluster 1



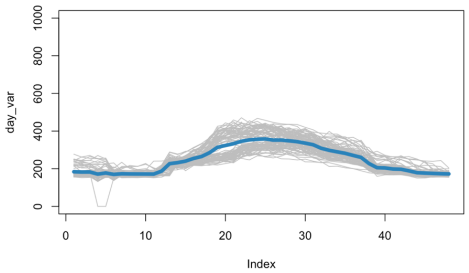
(b) Cluster 2



(c) Cluster 3



(d) Cluster 4



(e) Cluster 5

Figure 13: SOM Visualization of Clusters

3 Clustering Methods

In the previous chapter, clustering was applied to the weights of the neurons of the SOM, in this chapter we look into clustering from the original data. This may be able to pick up information that was lost between the observations and their best matching unit in the SOM.

3.1 Principle Co-ordinate Analysis

Clustering data becomes easier in lower dimensions, and so the data was normalized and PCA was explored to see if the dimensionality of the data could be reduced before the following clustering techniques were preformed. Figure 14 below shows the percentage of variation that can be captured by each component. 90.96 percent of the data is captured by the first four components.

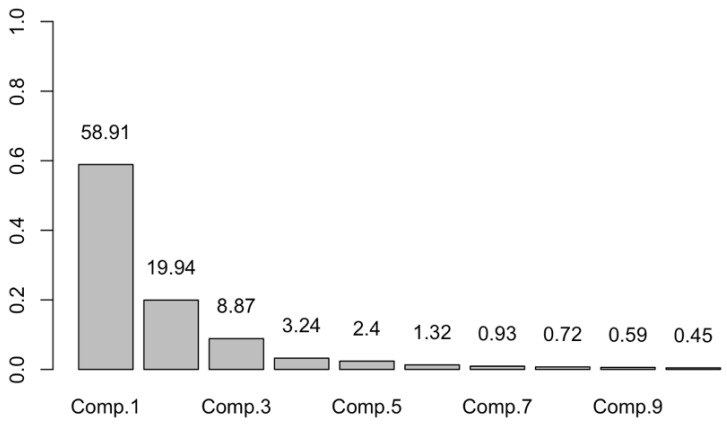


Figure 14: PCA Components % Variation

Figure 15 shows the first two components plotted against each other, from the plot it can be seen that there are 2 distinct clusters, and within these clusters, observations are quite close together. The two points on the left hand side of the plot show outliers that the SOM also picked up. The cluster on the left contains mostly 1s and 7s which are weekends. The other days within this cluster are likely to be public holidays. It is quite hard to visualize the relations of the days on this 2D plot, and so when looking at the techniques we will look at the clusters over a calendar of the year.

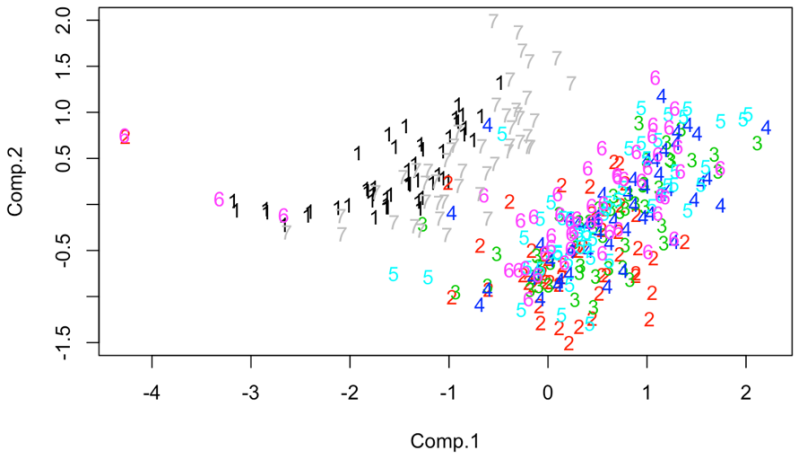
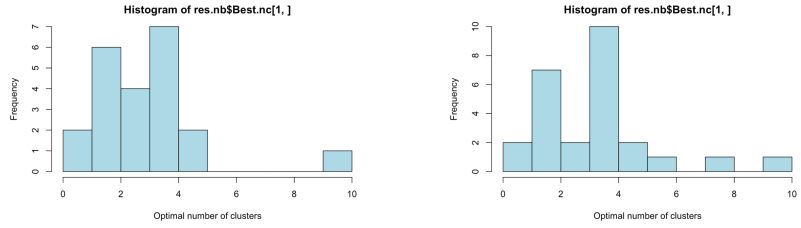


Figure 15: Plot of 2 PCA Components

Using NBclust, which uses a variety of techniques to determine how many clusters were in the data, the following histograms were created. The first one was created using the original data, the second histogram was created using the four principle components.

Both histograms indicate that there are 4 clusters most likely in the data, although the histogram using the principle components has more certainty on the number of clusters. This gives indication to the methods working better on the lower dimensional data.



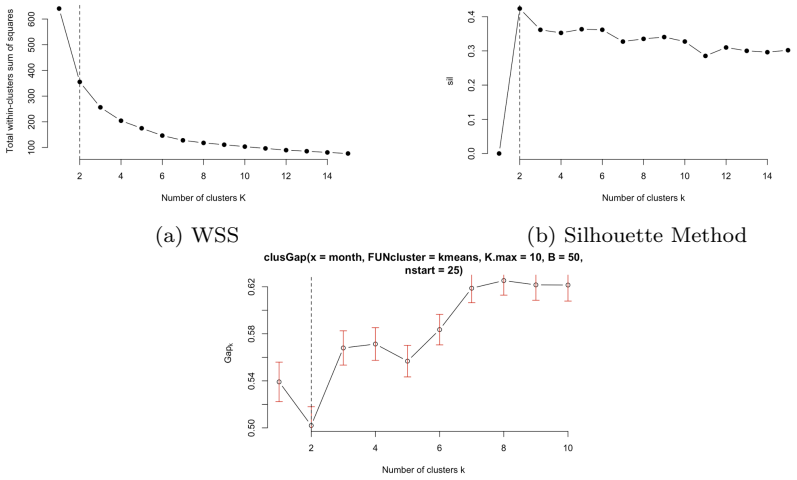
(a) Original Data (b) Four Principle Components

Figure 16: Results from NBClust

3.2 Distance based methods

3.2.1 K-means

The weighted sum of squares, silhouette method and gap statistic where used to find the number of clusters k means would find in the data. From the Figure 17 below we can see all these methods pointed towards there being two clusters. This makes sense as from the plotted data, it can be seen that there are two clear spherical clusters.



(a) WSS

(b) Silhouette Method

(c) Gap Statistic

Figure 17: K-means results for number of clusters

Figure 18 below shows how the K-Means method clustered the days, it clustered the weekdays from the weekends/public holidays. One Saturday was classed as a weekday, indicating to this Saturday having a much higher load than general. It should be investigated to what was happening in the warehouse that day so it can be taken into account in the future.

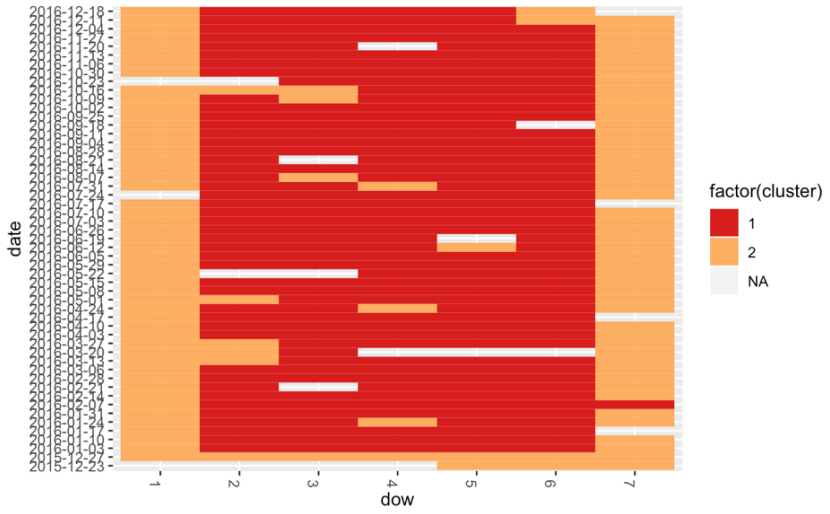
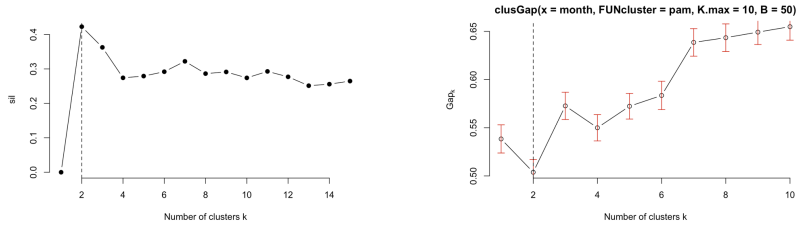


Figure 18: K-means clusters over year

3.2.2 K-Mediods

K-Mediods is similar to the previous K-Means method, it uses actual points as centres instead of sum of squared distances, and is therefore more robust to noise and outliers. The weighted sum of squares and gap statistic was used to find the number of clusters this method should find. Figure 19 shows that they both pointed towards there being two clusters.

PCA helps with removing noise, and so if the original data was used with out PCA, k-Mediods might out perform K-Means.



(a) WSS

(b) Gap Statistic

Figure 19: K-Mediods results for number of clusters

The clusters this method produced can be viewed in Figure 20 below, it produced the same clusters as the K-means method did, indicating towards the data not having huge noise and outlier influences.

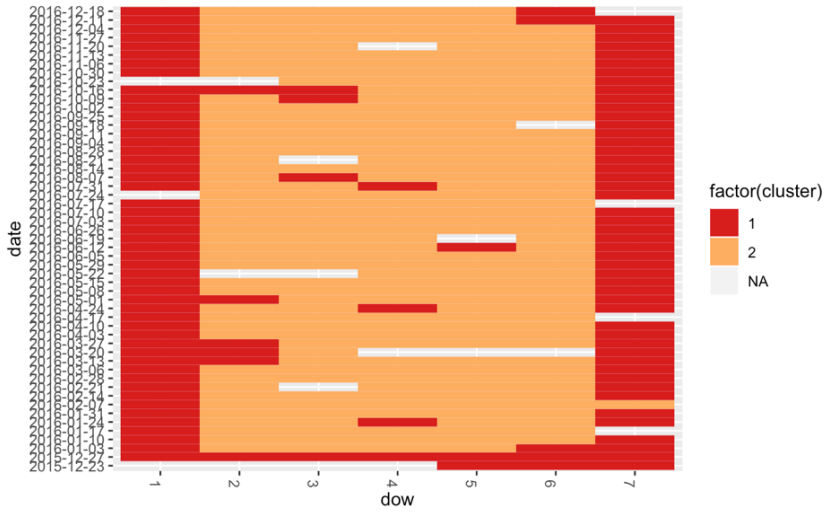
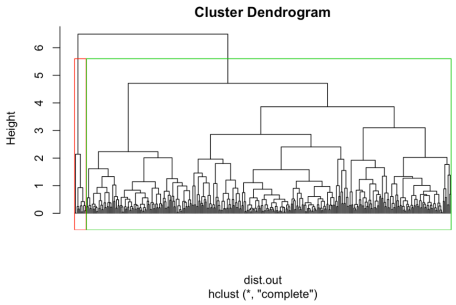


Figure 20: K-Mediods clusters over year

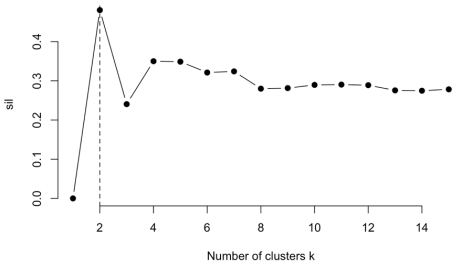
3.2.3 Hierarchical clustering(Agglomerative)

Hierarchical clustering was looked into, an advantage of this method is that the number of clusters does not have to be picked at the get go and can be decided by looking at a dendrogram. Four methods where looked into (complete linkage, single linkage, average linkage and centroid linkage), all of these methods pointed towards there being two clusters in the data according to the silhouette distance.

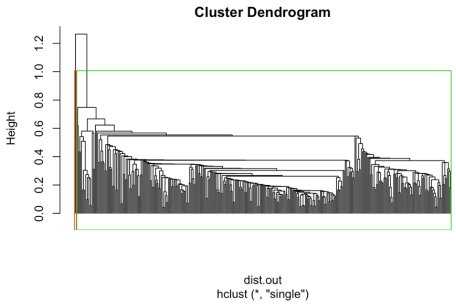
From the silhouette distance it seems that the best methods would be single linkage or centroid linkage as their silhouette distance was around 0.6 which is closer to 1 than the other methods. Although they all point towards 2 clusters when looking at their dendrograms, it looks like the two clusters would be the outliers and the rest of the day which isn't very helpful.



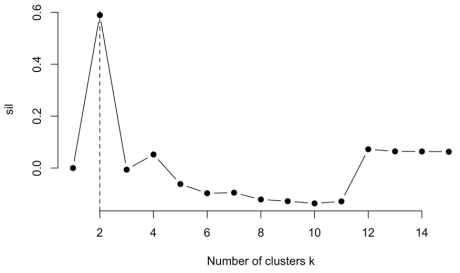
(a) Complete Linkage Dendrogram



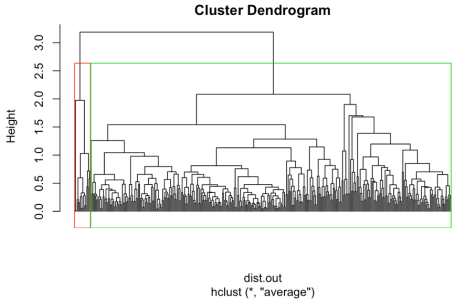
(b) Complete Linkage Silhouette vs k



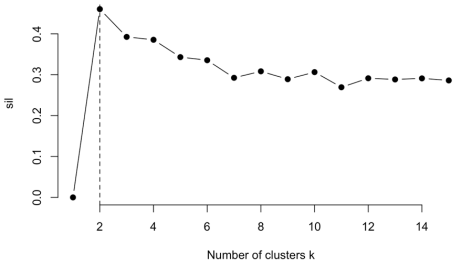
(c) Single Linkage Dendrogram



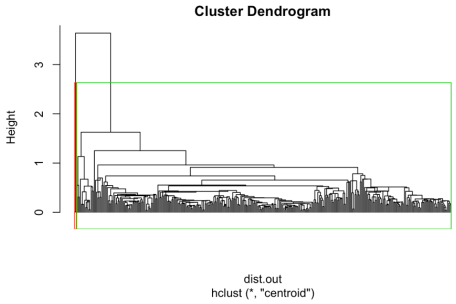
(d) Single Linkage Silhouette vs k



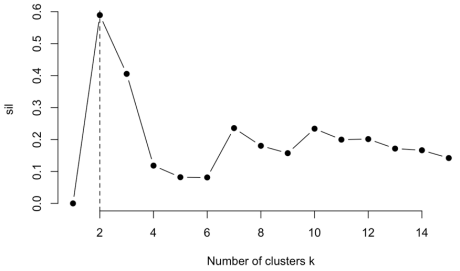
(e) Average Linkage Dendrogram



(f) Average Linkage Silhouette vs k



(g) Centroid Linkage Dendrogram



(h) Centroid Linkage Silhouette vs k

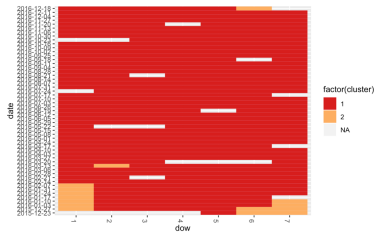
Figure 21: Visualization of clusters

Table 2 below shows the con-phonetic correlation between the distance matrix before and after the method has been applied, from this it shows the best method could be the average linkage method.

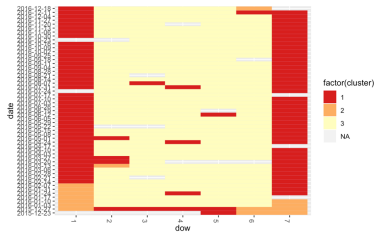
Method	Con-phonetic Correlation
Complete	0.681063
Single	0.7355109
Average	0.7753732
Centroid	0.7476938

Table 2: Input Data Format

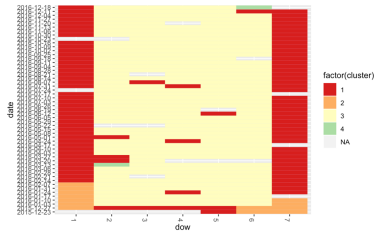
Figure 22 below shows the clusters of the hierarchical clustering using the average linkage method at different number of clusters. With two clusters, as seen before with the dendrogram, it only separates the outliers and Christmas holiday period from the rest of the days. Using three clusters, the weekdays are separated from the weekends. And using four clusters separates the outliers from the Christmas holiday period.



(a) 2 Clusters



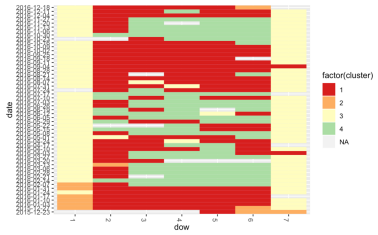
(b) 3 Clusters



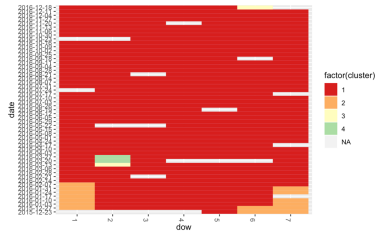
(c) 4 Clusters

Figure 22: Average Linkage Results

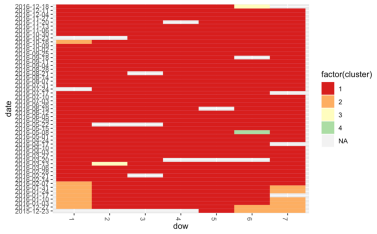
The other methods were looked into visually, from Figure 23 we can see that the single and centroid linkage perform poorly and focus on outliers not picking up the difference between weekends and weekdays in the first four clusters. The complete method is able to pick up the difference between weekends/weekdays, it was also able to pick up the seasonality in data as seen in the SOM. This means it is probably the best method to use in this case, despite the con-phonetic correlation value.



(a) Complete Linkage 4 Clusters



(b) Single Linkage 4 Clusters



(c) Centroid Linkage 4 Clusters

Figure 23: Other Methods 4 Clusters

3.3 Density based methods

3.3.1 DBSCAN

DBSCAN was looked into as it works really well at ignoring outliers and noise. The plot below shows the ranked average of each point to it's 10 closest neighbours. The "elbow" was found as this is the value that was used for eps as this is where the distance between points start to jump up indicating that the distance between points would be around 0.47. And so the parameters used were Minpts = 10 and eps = 0.47.

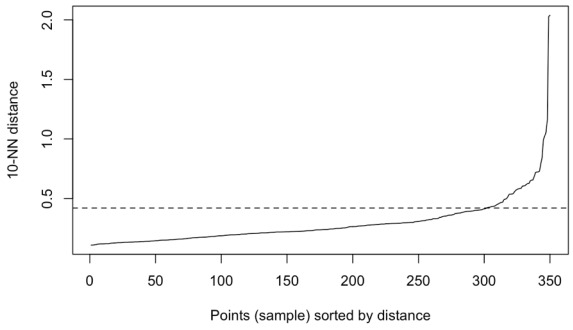


Figure 24: Neighbour average distance vs ranked observations

The cluster plot in Figure 25 below shows that within the first two principle

components, the method was able to find the two main clusters, while ignoring outliers.

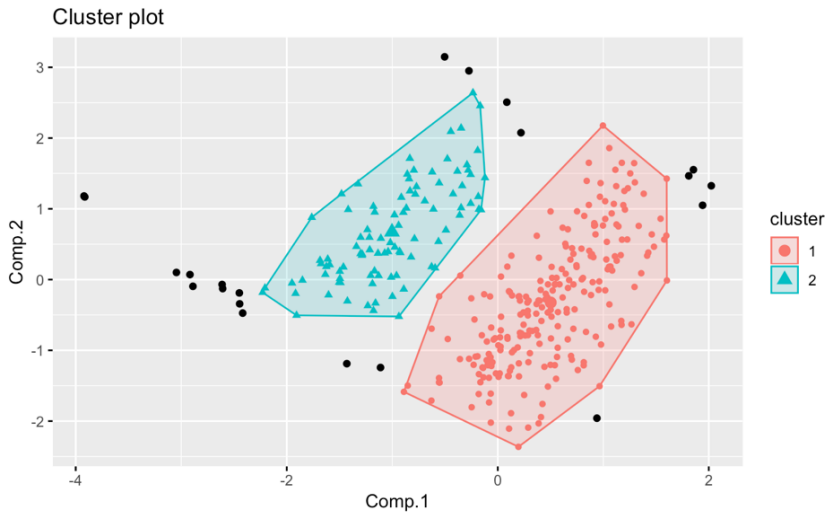


Figure 25: KDBSCAN Clusters

The clusters can be visualised on the calendar below. The method is able to distinguish between weekends and weekdays. The Christmas holidays and outliers are seen as noise by the method, it also sees as the 4 Saturdays above the blue cluster as noise which the previous methods linked to the blue cluster.

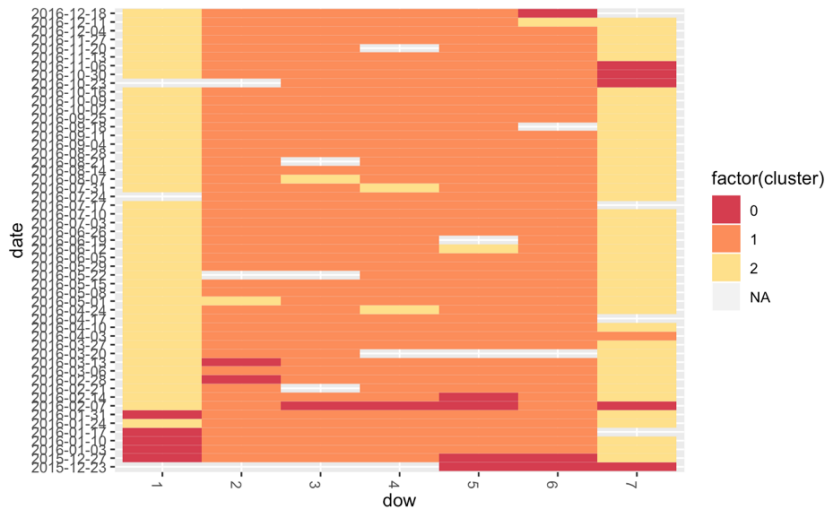


Figure 26: DBSCAN clusters over year

3.4 Probabilistic methods

3.4.1 GMM Clustering

The best number of clusters for the Guassian Mixture Model was found by plotting the BIC score vs the number of clusters, and looking for the "elbow". From Figure 26 below 2 clusters were chosen.

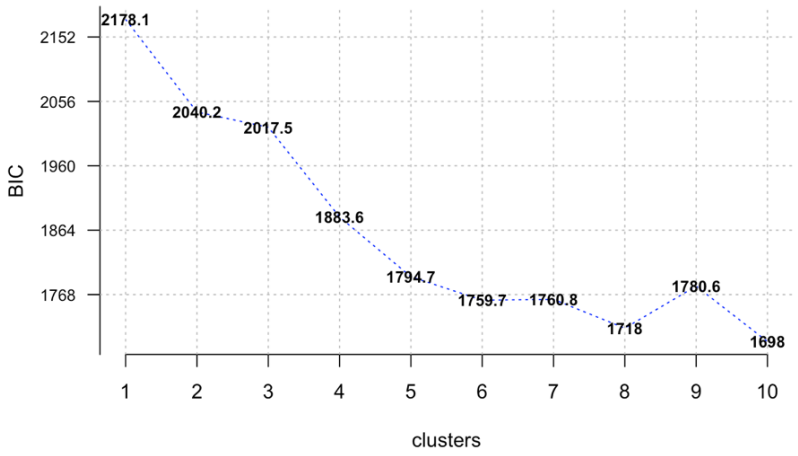


Figure 27: BIC vs no. of clusters

Since GMM soft clusters the data, once the model was fitted the observations were placed into the cluster they were more likely to be in. From the results below it can be seen that the clusters aren't as useful, as they do not pick up weekend/weekday. But looking at the final fit this could be an indication to potential seasonality of the data, as the cluster split seems to be close to a summer/winter split.

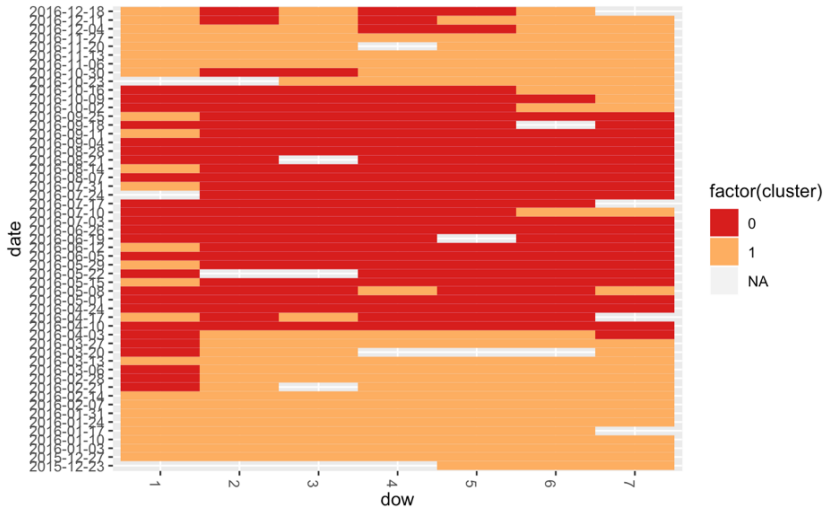
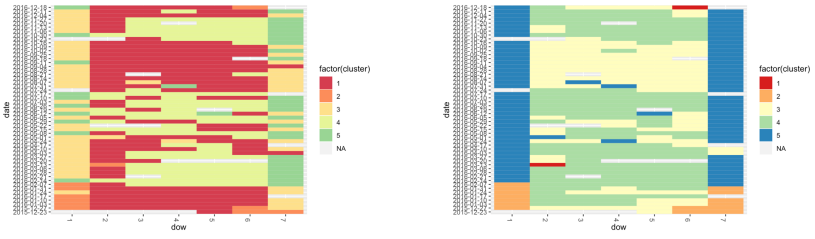


Figure 28: GMM clusters over year

4 Conclusion

Many different insights can be drawn from each of the different clustering techniques, as they all bring insights into the data in different ways. The final method chosen was complete hierarchical clustering applied to the weights of the SOM, this was chosen based on domain knowledge of what clusters could/should be in the data.

Complete Linkage Hierarchical Clustering performed very similarly when looking for four clusters on the SOM weights and the PCA components. But when trying to find 5 clusters, the clusters found from the weights of the SOM were able to pick up the outliers whilst the clusters found in the PCA components grouped the outliers with the Christmas holidays. This can be seen in Figure 29 below.



(a) PCA (b) SOM

Figure 29: 5 Clusters PCA and SOM

5 Appendices

5.1 Plagiarism Declaration

Department of Statistical Sciences Plagiarism Declaration form

A copy of this form, completed and signed, to be attached to all coursework submissions to the Statistical Sciences Department. Submissions without this form will not be marked.

COURSE CODE: **STA5077Z**

COURSE NAME: Unsupervised Learning

STUDENT NAME: Tiffany Woodley

STUDENT NUMBER: WDLTIF001

TUTORS NAME: NA TUT. GROUP #: NA

PLAGIARISM DECLARATION

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed, and has been cited and referenced.
3. This tutorial/report/project is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
5. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Note that agreement to this statement does not exonerate you from the University's plagiarism rules (http://www.uct.ac.za/uct/policies/plagiarism_students.pdf).

Signature:  Date: 01/09/2019

5.2 R Code

5.2.1 Self Organising Maps

```
##CODE ADAPTED FROM CODE GIVEN IN UNSUPERVISED MODULE

# Importing libraries
library(tidyverse)
library(kohonen)
library(dummies)
library(ggplot2)
library(sp)
library(maptools)
library(reshape2)
library(rgeos)
library(RColorBrewer)

#Importing the data for each month
data.one <- as.matrix(read.csv("1.csv", sep = ","))
data.two <- as.matrix(read.csv("2.csv", sep = ","))
data.three <- as.matrix(read.csv("3.csv", sep = ","))
data.four <- as.matrix(read.csv("4.csv", sep = ","))
data.five <- as.matrix(read.csv("5.csv", sep = ","))
data.six <- as.matrix(read.csv("6.csv", sep = ","))
data.seven <- as.matrix(read.csv("7.csv", sep = ","))
data.eight <- as.matrix(read.csv("8.csv", sep = ","))
data.nine <- as.matrix(read.csv("9.csv", sep = ","))
data.ten <- as.matrix(read.csv("10.csv", sep = ","))
data.eleven <- as.matrix(read.csv("11.csv", sep = ","))
data.twelve <- as.matrix(read.csv("12.csv", sep = ","))

#Combining the data into a year
data <- rbind(data.one,data.two,data.three,data.four,data.
    five,data.six,data.seven,data.eight,data.nine,data.ten,
    data.eleven,data.twelve)

#Create Date, Time and Load Column
data <-data[,c(1,2,7)]
colnames(data) <- c("Date", "Time","Load")
data <- data.frame(data)
dates <- unique(data$Date)

#Plotting a day observation
day <- data[data$Date == dates[3],]
day_var = as.numeric(as.character(day$Load))
plot(day_var, type = 'l')

#Plotting the profiles over a Year
day_var = as.numeric(as.character(data$Load))
```

```
plot(day_var, type = 'l', xaxt = 'n')
abline(650,0, col = "red")
axis(1,at = c(1,length(data$Load)), labels = c(1,368))

#Reshape data so that each day is an observation
month <- c()
day_chars <- c()

for (i in 1:(length(dates))){
  if(nrow(data[data$Date == dates[i],]) == 48){
    day <- data[data$Date == dates[i],]
    month <- rbind(month, day$Load)
    day_chars <- c(day_chars, i)
  }
}

times <- data[data$Date == dates[day_chars[1]],]$Time
times <- sort(times)
colnames(month) <- times
row.names(month) <- dates[day_chars]

#Import CSV with day of week values
dow_dic <- read.csv("MyData.csv",sep =";")
dow_dic

#Initialize SOM grid
som_grid <- somgrid(xdim = 9, ydim= 9, topo="hexagonal")

#Fit the SOM to the data
som_model <- som(month,grid=som_grid,
                 rlen=3000,
                 alpha=c(0.05,0.01),
                 keep.data = TRUE
)

#Plots to interpret SOM
plot(som_model, type="changes")
plot(som_model, type="counts")
plot(som_model, type="dist.neighbours")
plot(som_model, type="codes")

a=som_model$codes[[1]]
par(mfrow=c(4,4))
for (i in 1:48){
  plot(som_model, type = "property", property = a[,i], main
      = as.character(i))
}

#Finding number of clusters
par(mfrow=c())
```

```
data_var <- som_model$codes[[1]]
wss <- (nrow(data_var)-1)*sum(apply(data_var,2,var))
for (i in 2:15) {
  wss[i] <- sum(kmeans(data_var, centers=i)$withinss)
}
plot(wss)

#Plotting clusters on SOM grid

pretty_palette <- brewer.pal(5,"Spectral")

som_cluster <- cutree(hclust(dist(som_model$codes[[1]])), 5)
plot(som_model, type="mapping", labels=dow_dic[,2],
     main = "Clusters")
add.cluster.boundaries(som_model, som_cluster)
plot(som_model, type="quality")
plot(som_model, type="mapping", labels=dow_dic[,2])
plot(som_model, type="mapping", labels=dow_dic[,2],
     bgcol = pretty_palette[som_cluster])
som.hc <- cutree(hclust(dist(som_model$codes[[1]])), 5)
add.cluster.boundaries(som_model, som.hc)

#Plotting clusters over calendar year

samp <- data[data$Date %in% dates[day_chars],]
new_dates = sapply(dates,function(x){str_replace_all(x,"-","
    /")})
cluster = som.hc[som_model$unit.classif]
comb <- data.frame(date = unique(samp[,1]), dow = factor(dow
    _dic[,2]), cluster = cluster)
comb.two <-comb

#Add NAs to cluster values of days with missing values
clusters <- c()
for (i in dates)
{
  clusters <- c(clusters,ifelse(i %in% comb.two$date, comb.
      two[comb.two$date == i,]$cluster,NA))
}

#Create dataframe to plot with date, day of week and cluster
     value
combined_test <- data.frame(date = unique(data[,1]), dow =
    factor(c(c(4,5,6,7) , rep(1:7,52))),cluster = clusters)
comb <- combined_test

#Setting dates to the first date of their week
comb$date[2:4] <- comb$date[1]

i =  6
```

```
j =  11
k =  5

for (l in 0:51){
  comb$date[(i+7*l):(j+7*l)] <- comb$date[(k+7*l)]
}

colors <- (brewer.pal(5,"Spectral"))

#Plotting the "heatmap" calendar
ggplot(comb, aes(x = dow, y = date, fill = factor(cluster)))
    +
  geom_tile()+ theme(axis.text.x = element_text(angle = -90,
      hjust = 0)) +
  scale_fill_manual(values=colors)

#To plot profiles in cluster with average
cluster_one <- comb.two%>%filter(cluster == 1)%>%select(date
   )
cluster_one

day <- data[data$Date == cluster_one$date[1],]
day_var = as.numeric(as.character(day$Load))
plot(day_var, type = 'l', col = "grey",ylim = c(0,1000))

total = day_var

for (i in 2:length(cluster_one$date)){
  day <- data[data$Date == cluster_one$date[i],]

  day_var = as.numeric(as.character(day$Load))
  lines(day_var, type = 'l', col = "grey")
  total = total + day_var
}

lines(total/length(cluster_one$date), type = 'l', col =
   colors[1], lwd = 5)
```

5.2.2 Clustering

```
##CODE ADAPTED FROM CODE GIVEN IN UNSUPERVISED MODULE

#Importing libraries
library(cluster)
library(NbClust)
library(fpc)
library(dbscan)
library(factoextra)
library(ClusterR)
library(RColorBrewer)
```

```
#Importing month data
data.one <- as.matrix(read.csv("1.csv", sep = ","))
data.two <- as.matrix(read.csv("2.csv", sep = ","))
data.three <- as.matrix(read.csv("3.csv", sep = ","))
data.four <- as.matrix(read.csv("4.csv", sep = ","))
data.five <- as.matrix(read.csv("5.csv", sep = ","))
data.six <- as.matrix(read.csv("6.csv", sep = ","))
data.seven <- as.matrix(read.csv("7.csv", sep = ","))
data.eight <- as.matrix(read.csv("8.csv", sep = ","))
data.nine <- as.matrix(read.csv("9.csv", sep = ","))
data.ten <- as.matrix(read.csv("10.csv", sep = ","))
data.eleven <- as.matrix(read.csv("11.csv", sep = ","))
data.twelve <- as.matrix(read.csv("12.csv", sep = ","))

#Combining months to become a year
data <- rbind(data.one,data.two,data.three,data.four,data.
    five,data.six,data.seven,data.eight,data.nine,data.ten,
    data.eleven,data.twelve)

#Creating Date,Time and Load Columns
data <-data[,c(1,2,7)]
colnames(data) <- c("Date", "Time","Load")
data <- data.frame(data)
dates <- unique(data$Date)

#Reshaping data so that each day is an observation
month <- c()
day_chars <- c()

for (i in 1:(length(dates))){
  if(nrow(data[data$Date == dates[i],]) == 48){
    day <- data[data$Date == dates[i],]
    month <- rbind(month, day$Load)
    day_chars <- c(day_chars, i)
  }
}

times <- data[data$Date == dates[day_chars[1]],]$Time
times <- sort(times)
colnames(month) <- times
row.names(month) <- dates[day_chars]

#Importing CSV with day of week values
dow_dic <- read.csv("MyData.csv",sep =";")

#Function to normalize variable
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

```
#Normalising each column
for (i in colnames(month)){
  month[,i] <- normalize(month[,i])
}

#Applying PCA
pca.out <- princomp(month)

#Plotting the PCA Variance
Variance<-(pca.out$sdev)^2
bp <-barplot((Variance/sum(Variance))[1:10], ylim = c(0,1))
labels <- as.character(round(Variance/sum(Variance),4)*100)
    [1:10]
text(bp,(Variance/sum(Variance))[1:10]+0.04,labels, cex=1,
    pos=3)

#Plotting first two components
month <- (pca.out$scores[,1:2])
plot(month, pch = dow_dic[,2], col = dow_dic[,2], type = 'n'
    )
text(month[dow_dic[,2] == 1,],label=1,col=1)
text(month[dow_dic[,2] == 2,],label=2,col=2)
text(month[dow_dic[,2] == 3,],label=3,col=3)
text(month[dow_dic[,2] == 4,],label=4,col=4)
text(month[dow_dic[,2] == 5,],label=5,col=5)
text(month[dow_dic[,2] == 6,],label=6,col=6)
text(month[dow_dic[,2] == 7,],label=7,col=8)

#Using the first 4 components
month <- (pca.out$scores[,1:4])

##K-MEANS

#plotting wss
k.max <- 15 # Maximal number of clusters
df.out <- month
wss <- sapply(1:k.max,
              function(k){kmeans(df.out, k, nstart=25 )$tot.
                  withinss})

plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number␣of␣clusters␣K",
     ylab="Total␣within-clusters␣sum␣of␣squares")
abline(v = 2, lty =2)

#plotting silhouette vs k
k.max <- 15
data.out <- month
```

```
sil <- rep(0, k.max)

for(i in 2:k.max){
  km.res <- kmeans(df.out, centers = i, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(df.out))
  sil[i] <- mean(ss[, 3])
}

plot(1:k.max, sil, type = "b", pch = 19,
     frame = FALSE, xlab = "Number of clusters k")
abline(v = which.max(sil), lty = 2)

#Plotting the gap statistic
gap_stat <- clusGap(month, FUN = kmeans, nstart = 25, K.max
    = 10, B = 50)
plot(gap_stat, frame = FALSE, xlab = "Number of clusters k")
abline(v = 2, lty = 2)

#fitting k-means with 2 clusters
km.out <- kmeans(month, 2, nstart = 25)

#function for viewing clusters on calendar - see comments in
     som.R
view_year <- function(clusters, data, day_chars, dow_dic){

  samp <- data[data$Date %in% dates[day_chars],]
  new_dates = sapply(dates,function(x){str_replace_all(x,"-"
      ,"/")})
  comb <- data.frame(date = unique(samp[,1]), dow = factor(
      dow_dic[,2]), cluster = clusters)

  clusters <- c()


  for (i in dates)
  {
    clusters <- c(clusters,ifelse(i %in% comb$date, comb[
        comb$date == i,]$cluster,NA))
  }

  combined_test <- data.frame(date = unique(data[,1]), dow =
       factor(c(c(4,5,6,7) , rep(1:7,52))),cluster = clusters
      )
  comb <- combined_test

  comb$date[2:4] <- comb$date[1]

  i =  6
  j =  11
  k =  5
```

```
  for (l in 0:51){
    comb$date[(i+7*l):(j+7*l)] <- comb$date[(k+7*l)]
  }

  colors <- brewer.pal(6,"Spectral")

  ggplot(comb, aes(x = dow, y = date, fill = factor(cluster)
      )) +
    geom_tile()+ theme(axis.text.x = element_text(angle =
        -90, hjust = 0)) +
    scale_fill_manual(values=colors)
}

#viewing k-means over a year
view_year(km.out$cluster,data, day_chars, dow_dic)

##K_MEDIODS

#plotting the silhouette width
k.max <- 15
data.out <- month
sil <- rep(0, k.max)

for(i in 2:k.max){
  pam.out<- pam(month,i)
  sil[i] <- pam.out$silinfo$avg.width
}

plot(1:k.max, sil, type = "b", pch = 19,
     frame = FALSE, xlab = "Number␣of␣clusters␣k")
abline(v = which.max(sil), lty = 2)

#plotting gap statistic
gap_stat <- clusGap(month, FUN = pam, K.max = 10, B = 50)

plot(gap_stat, frame = FALSE, xlab = "Number␣of␣clusters␣k")
abline(v = 2, lty = 2)

#fitting k-mediods with 2 clusters
pam.out <- pam(month, 2)

#viewing k-mediods over a year
view_year(pam.out$cluster,data, day_chars, dow_dic)

##HIERARCHICAL CLUSTERING
#COMPLETE LINKAGE

#fit and plot dendrogram
dist.out <- dist(month, method = "euclidean")
```

```
hc <- hclust(dist.out, method = "complete")
plot(hc, labels = F,-1)
rect.hclust(hc, k = 2, border = 2:3)

#work out conphenetic correlation
cor(dist.out,cophenetic(hc))

#plot silhouette distance vs k
k.max <- 15
data.out <- month
sil <- rep(0, k.max)

for(i in 2:k.max){
  sil.sum<- summary(silhouette(cutree(hc,i),dist.out))
  sil[i] <- sil.sum$avg.width
}

plot(1:k.max, sil, type = "b", pch = 19,
     frame = FALSE, xlab = "Number␣of␣clusters␣k")
abline(v = which.max(sil), lty = 2)

#Viewing the clusters over the year
clusters <- cutree(hc,4)
view_year(clusters,data, day_chars, dow_dic)

#SINGLE LINKAGE

#fit and plot dendrogram
dist.out <- dist(month, method = "euclidean")
hc <- hclust(dist.out, method = "single")
plot(hc, labels = F,-1)
rect.hclust(hc, k = 2, border = 2:3)

#work out conphenetic correlation
cor(dist.out,cophenetic(hc))

#plot silhouette distance vs k
k.max <- 15
data.out <- month
sil <- rep(0, k.max)

for(i in 2:k.max){
  sil.sum<- summary(silhouette(cutree(hc,i),dist.out))
  sil[i] <- sil.sum$avg.width
}

plot(1:k.max, sil, type = "b", pch = 19,
     frame = FALSE, xlab = "Number␣of␣clusters␣k")
abline(v = which.max(sil), lty = 2)
```

```
#Viewing the clusters over the year
clusters <- cutree(hc,4)
view_year(clusters,data, day_chars, dow_dic)

#AVERAGE LINKAGE

#fit and plot dendrogram
dist.out <- dist(month, method = "euclidean")
hc <- hclust(dist.out, method = "average")
plot(hc, labels = F,-1)
rect.hclust(hc, k = 2, border = 2:3)

#work out conphenetic correlation
cor(dist.out,cophenetic(hc))

#plot silhouette distance vs k
k.max <- 15
data.out <- month
sil <- rep(0, k.max)

for(i in 2:k.max){
  sil.sum<- summary(silhouette(cutree(hc,i),dist.out))
  sil[i] <- sil.sum$avg.width
}

plot(1:k.max, sil, type = "b", pch = 19,
     frame = FALSE, xlab = "Number␣of␣clusters␣k")
abline(v = which.max(sil), lty = 2)

#Viewing the clusters over the year
clusters <- cutree(hc,4)
view_year(clusters,data, day_chars, dow_dic)

#CENTROID LINKAGE

#fit and plot dendrogram
dist.out <- dist(month, method = "euclidean")
hc <- hclust(dist.out, method = "centroid")
plot(hc, labels = F,-1)
rect.hclust(hc, k = 2, border = 2:3)

#work out conphenetic correlation
cor(dist.out,cophenetic(hc))

#plot silhouette distance vs k
k.max <- 15
data.out <- month
sil <- rep(0, k.max)

for(i in 2:k.max){
```

```
  sil.sum<- summary(silhouette(cutree(hc,i),dist.out))
  sil[i] <- sil.sum$avg.width
}

plot(1:k.max, sil, type = "b", pch = 19,
     frame = FALSE, xlab = "Number␣of␣clusters␣k")
abline(v = which.max(sil), lty = 2)

#Viewing the clusters over the year
clusters <- cutree(hc,4)
view_year(clusters,data, day_chars, dow_dic)

##NBCLUST

#Using muliple methods to find the number of clusters in the
     data
res.nb <- NbClust(month, distance = "euclidean",min.nc = 2,
    max.nc
                  = 10, method = "complete", index ="all")

par(mfrow = c(1,1))
hist(res.nb$Best.nc[1,],breaks=0:10,col="lightblue",xlab="
    Optimal␣number␣of␣clusters")

##DBSCAN

#Using neighbour distances to find eps for minpts = 10
dbscan::kNNdistplot(month[,1:2], k =  10)
abline(h = 0.42, lty = 2)

#Fitting DBSCAN
x <- month[,1:2]
db <- fpc::dbscan(x, eps = 0.42, MinPts =10 )

#Viewing DBSCAN clusters
fviz_cluster(db, month[,1:2], geom = "point")

#Viewing DBSCAN clusters over the year
view_year(db$cluster,data, day_chars, dow_dic)

#Plotting BIC vs k for GMM clustering
opt_gmm = Optimal_Clusters_GMM(month, max_clusters = 10,
    criterion = "BIC",

                               dist_mode = "maha_dist", seed
                                   _mode = "random_subset",

                               km_iter = 10, em_iter = 10,
                                   var_floor = 1e-10,
```

```
                                plot_data = T)

#fitting the GMM model
gmm = GMM(month, 2, dist_mode = "maha_dist", seed_mode = "
    random_subset", km_iter = 10,

          em_iter = 10, verbose = F)

pr = predict_GMM(month, gmm$centroids, gmm$covariance_
    matrices, gmm$weights)

#Viewing GMM clusters over the year
view_year(pr$cluster_labels,data, day_chars, dow_dic)
```