Unsupervised Project

Federico Matteucci - 955753

September 2020

Contents

1 Abstract

In this report I will present a brief clustering analysis of the water treatment dataset [2]. The variables are data gathered from sensors in a water treatment plant. The goal of my work is to group the data in order to classify the operational status of the plant itself. To do so, I first reduced the dimensionality of the dataset with PCA, and then applied three different clustering approches: hyerarchical (with both single and complete linkages) and kmeans.
I found that the data is quite well separated even in the projected dataset.

2 Dataset

The dataset consists of 527 observations of 38 numeric variables, with 591 missing values that I replaced with the mean of the corresponding attribute. I then centered and scaled the data.
Fifteen pairs of variables are highly correlated (Pearson correlation greater than 0.80). Even though multicollinearity does not affect clustering analysis, dimensionality does, so I want to reduce the data to the minimum significative number of dimensions. This can be achieved with a PCA.
From the boxplot I deduce there are many outliers in te data, spanning even 15 standard deviations from the mean. It is acceptable as this analysis tries to identify extreme operating conditions of the plant.

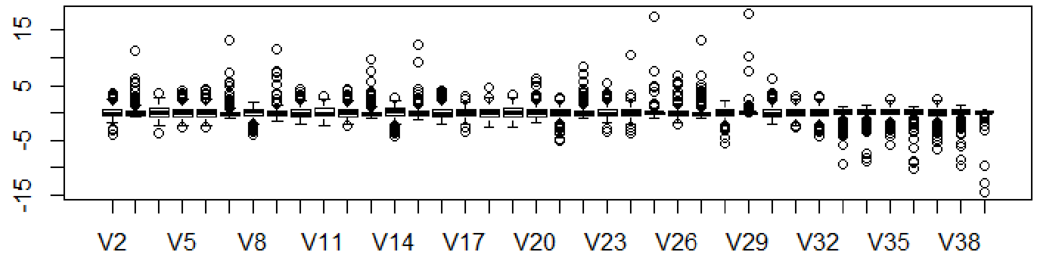

Figure 1: boxplot of water treatment. Lots of outliers are visible.

3 PCA

I selected the number of principal components according to the updated Kaiser rule descripted in [1]: for each eigenvalue λ I estimated its 95% confidence interval

$$\left(\lambda - 0.95\left(\sqrt{\frac{2}{527}}\lambda\right), \lambda + 0.95\left(\sqrt{\frac{2}{527}}\lambda\right)\right)$$

and kept only $\lambda : \lambda + 0.95\left(\sqrt{\frac{2}{527}}\lambda\right) \geq 1$.
This ensures that every principal direction will explain more variance than one initial variable, but it is a softer rule than the standard Kaiser, which instead would have chosen 9 components.

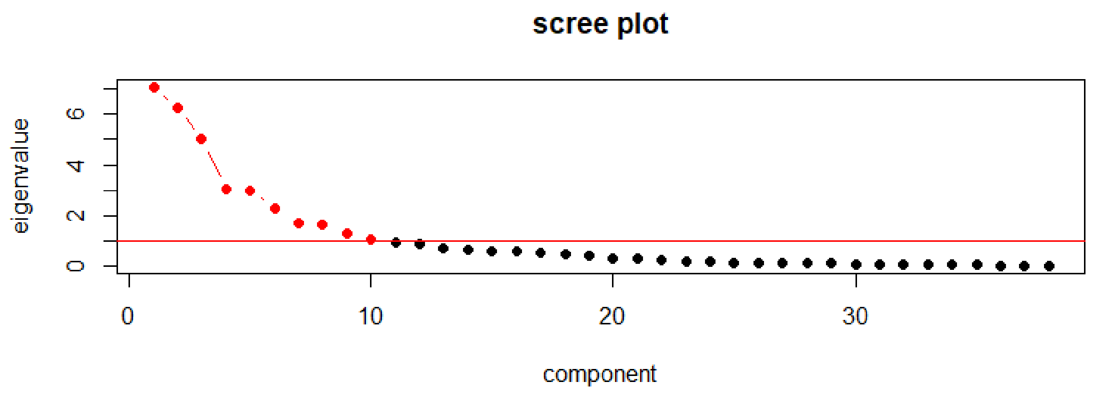

Figure 2: scree plot of $\lambda + 0.95\left(\sqrt{\frac{2}{527}}\lambda\right)$

The number of principal components I will use is 10, that explain 85% of the total variance.

	eigenvalue	expl var	*CI expl var*	cum var	***CI cum var***
1	6.679	0.176	*0.186*	**0.176**	***0.186***
2	5.918	0.156	*0.165*	**0.332**	***0.351***
3	4.737	0.125	*0.132*	**0.456**	***0.483***
4	2.880	0.076	*0.080*	**0.532**	***0.563***
5	2.843	0.075	*0.079*	**0.607**	***0.642***
6	2.134	0.056	*0.059*	**0.663**	***0.702***
7	1.589	0.042	*0.044*	**0.705**	***0.746***
8	1.569	0.041	*0.044*	**0.746**	***0.790***
9	1.231	0.032	*0.034*	**0.778**	***0.824***
10	0.988	0.026	*0.028*	**0.804**	***0.851***
11	0.883	0.023	*0.025*	**0.828**	***0.876***
12	0.836	0.022	*0.023*	**0.850**	***0.899***
13	0.653	0.017	*0.018*	**0.867**	***0.918***
14	0.634	0.017	*0.018*	**0.884**	***0.935***
15	0.577	0.015	*0.016*	**0.899**	***0.951***
16	0.541	0.014	*0.015*	**0.913**	***0.966***
17	0.503	0.013	*0.014*	**0.926**	***0.980***
18	0.425	0.011	*0.012*	**0.937**	***0.992***
19	0.388	0.010	*0.011*	**0.948**	***1.003***
20	0.278	0.007	*0.008*	**0.955**	***1.011***
21	0.275	0.007	*0.008*	**0.962**	***1.018***
22	0.209	0.006	*0.006*	**0.968**	***1.024***
23	0.180	0.005	*0.005*	**0.972**	***1.029***
24	0.156	0.004	*0.004*	**0.976**	***1.034***
25	0.125	0.003	*0.003*	**0.980**	***1.037***
26	0.116	0.003	*0.003*	**0.983**	***1.040***
27	0.105	0.003	*0.003*	**0.986**	***1.043***
28	0.094	0.002	*0.003*	**0.988**	***1.046***
29	0.088	0.002	*0.002*	**0.990**	***1.048***
30	0.076	0.002	*0.002*	**0.992**	***1.050***
31	0.068	0.002	*0.002*	**0.994**	***1.052***
32	0.049	0.001	*0.001*	**0.995**	***1.054***
33	0.040	0.001	*0.001*	**0.996**	***1.055***
34	0.036	0.001	*0.001*	**0.997**	***1.056***
35	0.035	0.001	*0.001*	**0.998**	***1.057***
36	0.025	0.001	*0.001*	**0.999**	***1.057***
37	0.023	0.001	*0.001*	**1.000**	***1.058***
38	0.014	0.000	*0.000*	**1.000**	***1.059***

Table 1: Components and explained variance. CI indicates the use of $\lambda + 0.95\left(\sqrt{\frac{2}{527}}\lambda\right)$

4 Clustering

In order to assess the optimal number of clusters, I plotted wss against number of clusters and used the elbow method.
For hyerarchical clustering, this means to find a proper local minimum of wss as a function of number of clusters and balancing the tradeoff between a small number of clusters and a low wss. This process is highly subjective and no number is proven to be better than others.

4.1 Single linkage hyerarchical clustering

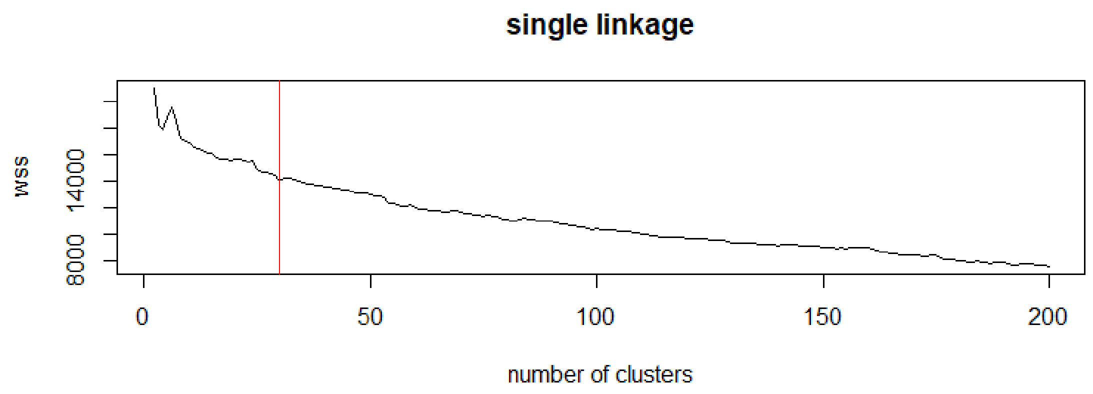

Figure 3: Single linkage. 30 is my choice for the best number of clusters.

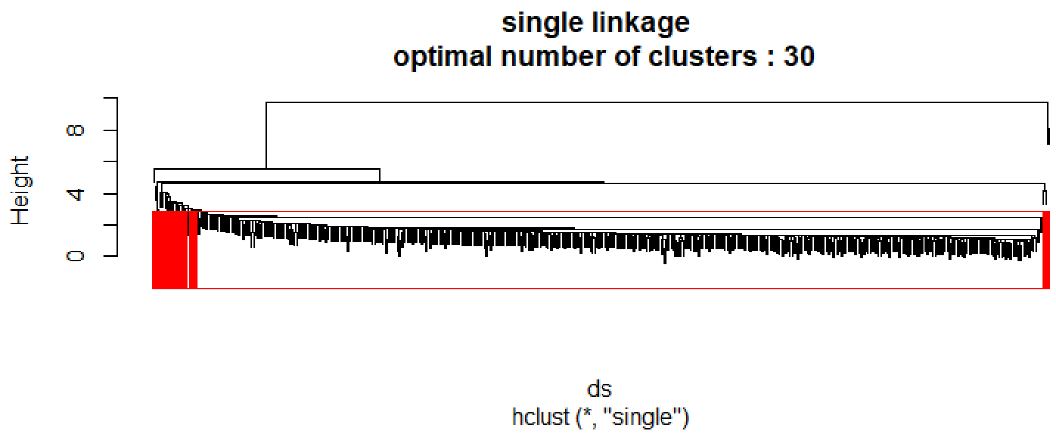

Figure 4: Single linkage produces 30 clusters, of which 28 singletons and one pair.

single linkage

cluster	cardinality
1	497
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	2
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1

Table 2: Single linkage produces 30 clusters, of which 28 singletons and one pair.

4.2 Complete linkage hyerarchical clustering

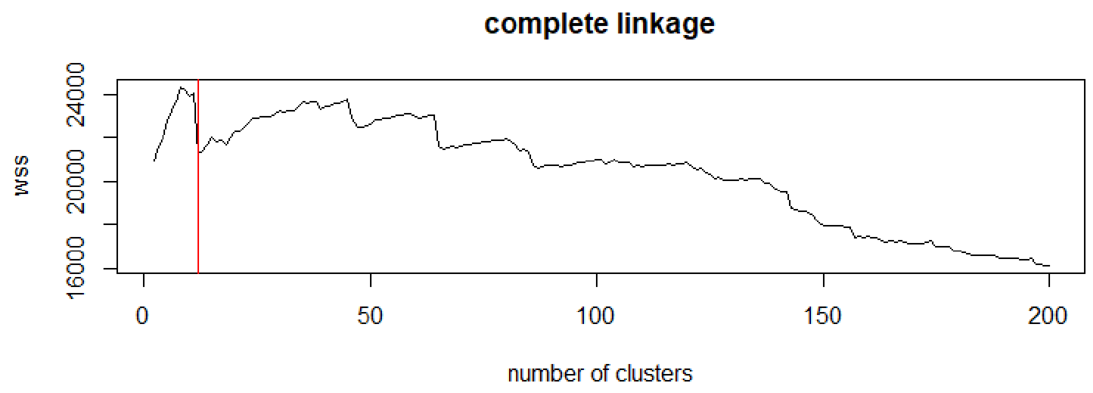

Figure 5: Complete linkage. 12 is my choice as optimal number of clusters.

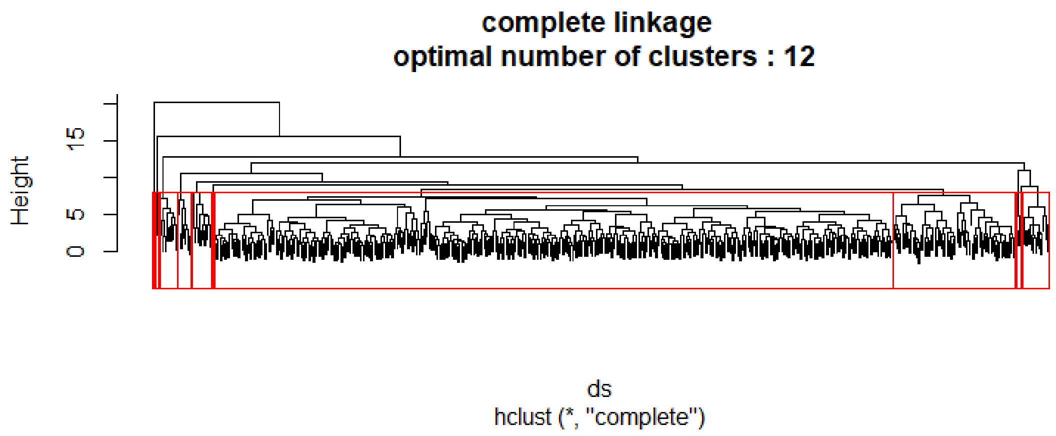

Figure 6: Complete linkage produces 12 clusters, one of which is massive.

complete linkage

cluster	cardinality
1	399
2	1
3	1
4	16
5	11
6	72
7	12
8	2
9	8
10	1
11	1
12	3

Table 3: Complete linkage produces 12 clusters, one of which is massive.

4.3 Comment on hyerarchical clustering

Using a different metric to measure the distance of clusters leads to very different results: the complete linkage clusters are much more balanced in size than the single linkage ones.
Recall that, if X and Y are clusters and d is the euclidean distance,

$$dist_{single}(X,Y) = \inf_{x \in X, y \in Y} d(x,y)$$
$$dist_{complete}(X,Y) = \sup_{x \in X, y \in Y} d(x,y)$$

Therefore, $dist_{complete}$ considers similar two clusters if their furthest points are close, while $dist_{single}$ (that uses the canonical distance for subsets of $\mathbb{R}^p$) groups clusters according to their closest points, regardless of the furthest ones.
The two method use opposite approaches, that's why I included both of them.

4.4 k-means clustering

The optimal number of clusters is chosen with a different, more objective, elbow method:

$$elbow := min\{x : wss[x] - wss_{std}[x] < wss[x+1] + wss_{std}[x+1]\}$$

where x is the number of clusters, wss the vector of total within sum of squares, wss_{std} its standard deviation.
Please note that wss_{std} and $wss = wss_{mean}$ were calculated on a sample of 20 runs of the algorithm with random centroids.

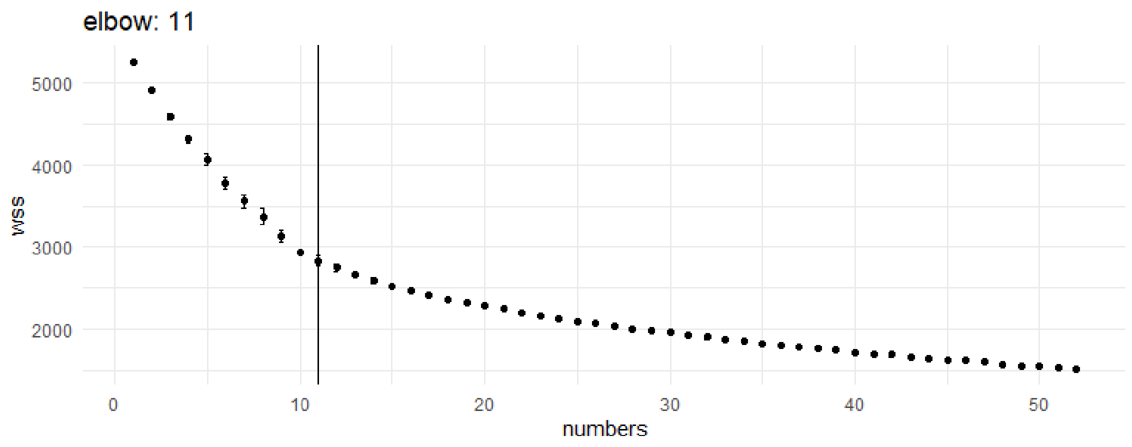

Figure 7: The elbow is the cutoff integer between fast and slow decrease in wss

This choice of the elbow was repeated 40 times, each time different centroids, in order to have a better grasp of what the best k could be. There was an ambiguity as sometimes the method would suggest 10 and sometimes 11.
I chose $k = 11$.

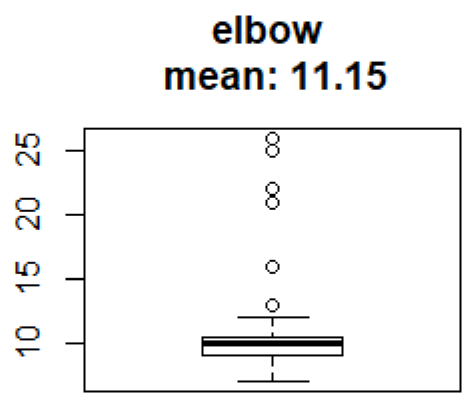

Figure 8: boxplot of the values of elbow.

11-means clustering

cluster	cardinality
1	2
2	74
3	13
4	63
5	68
6	23
7	66
8	6
9	103
10	23
11	86

Table 4: 11-means clustering produces clusters that are much more balanced than its hyerarchical counterparts.

4.5 Cluster visualization

We can visualize what the clusters look like on the two principal components of the original dataset. Results will not be very neat, as the two principal components together explain only 33.2% of the variance and thus are not a good representation of the actual dataset.

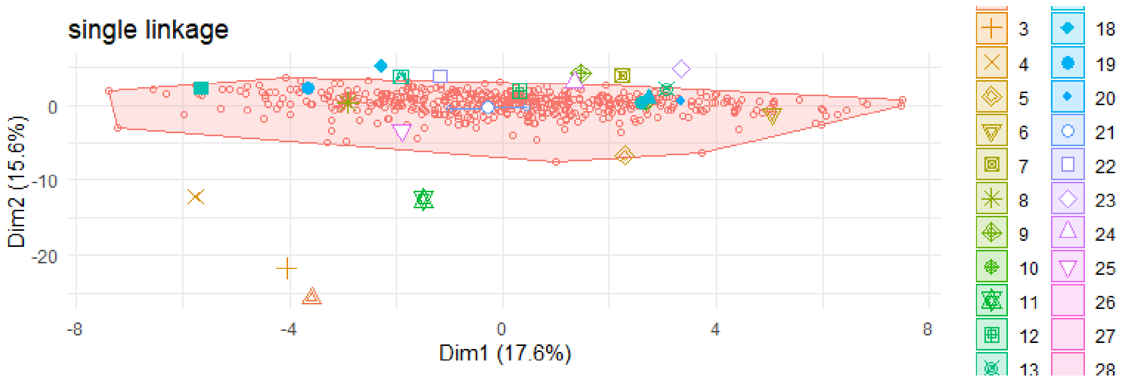

Figure 9: Single linkage.

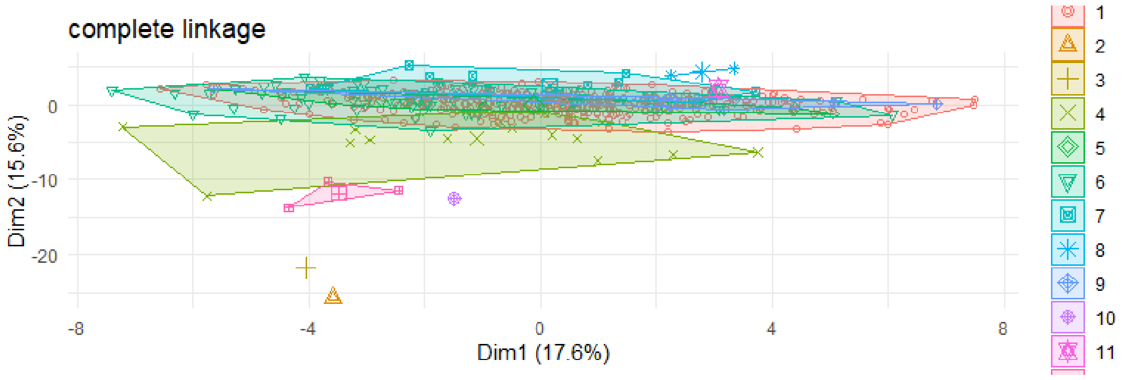

Figure 10: Complete linkage.

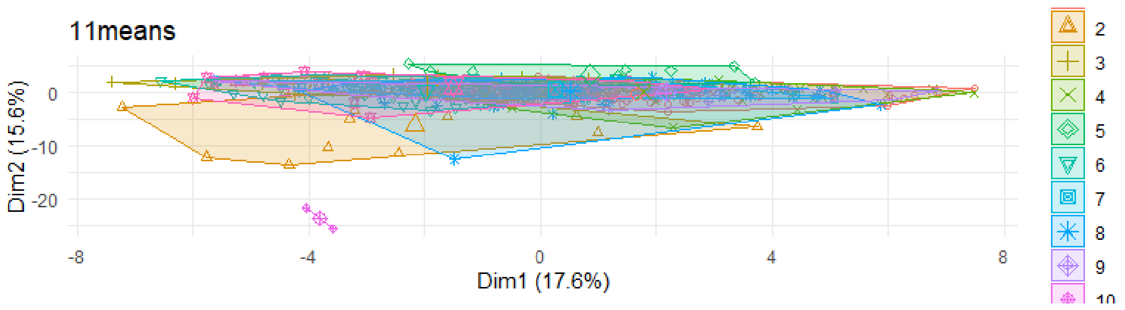

Figure 11: 11-means.

4.6 R squared

There are two different R^2 one can consider to evaluate the goodness of a clustering algorithm, measuring:

- cluster on attributes: how well the position of a point predicts its cluster - how well separated the dataset is.
- attributes on cluster: how much being in a certain cluster affects the attributes - how predictive the clustering is.

R^2 is intended to be result of a linear regression. Attributes refer to the components of the reduced dataset (after PCA).

variable	single linkage	complete linkage	11-means
X1	0	0.05	0.16
X2	0.02	0	0.09
X3	0.17	0.11	0.03
X4	0.01	0.04	0.08
X5	0.01	0.06	0.01
X6	0.01	0	0.08
X7	0	0	0
X8	0.08	0.06	0
X9	0.01	0.03	0
X10	0.03	0.01	0.03
mean	0.034	0.036	0.048

Table 5: attributes on cluster: the average R^2 is extremely low.

single linkage	complete linkage	11-means
0.52	0.43	0.5

Table 6: cluster on attribute (adjusted R^2): methods 'see' data to be quite separated, but single clustering outperforms the other two (I recon do to the many singletons that it produces).

5 Conclusion

- The data is well separated into clusters
- 11-means is the best one: it has the relatively high R^2 of the single linkage yet having clusters of similar cardinality
- The predictive power of suh clusters is very limited

References

[1] Ross Larsen and Russell T Warne. Estimating confidence intervals for eigenvalues in exploratory factor analysis. *Behavior research methods*, 42(3):871–876, 2010.

[2] Manel Poch. Water Treatment Plant Data Set. https://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant

6 Appendix

```
library(readr)
library(imputeTS)
library(haven)
library(gtools)
library(Hmisc)
library(magrittr)
library(ggplot2)
library(ggcorrplot)
library(PerformanceAnalytics)
library(factoextra)
library(cluster)

# ---- Data Loading ----


data <- read.csv("C:/Users/Io/Desktop/SML_Unsupervised/water_treatment.csv", sep=',', header=FALSE, stringsAsFactors=FALSE)
length(data[data=='?'])
data[data=='?'] <- NA
data <- data %>% subset(select=-c(1)) %>% data.frame

for(i in 1:ncol(data)){
        data[[i]] <- as.numeric(data[[i]])
        data_mean <- mean(data[[i]], na.rm=TRUE)
        data[[i]] <- data[[i]] %>% na.replace(data_mean)
}
data <- scale(data) %>% data.frame()

p <- ncol(data)
n <- nrow(data)

# ---- Correlation, number of highly correlated pairs ----

cor_matrix <- cor(data)
length(cor_matrix[cor_matrix != 1 & (cor_matrix >= 0.8 | cor_matrix <= -0.8) ])/2

# chart.Correlation(data, pch=19, subset=cor_matrix[cor_matrix != 1 & (cor_matrix >= 0.8 | cor_matrix <= -0.8) ])

# ---- PCA ----

eig_val <- eigen(cor_matrix)$values
eig_vec <- eigen(cor_matrix)$vectors

eig_val_CI <- 0.95 * sqrt(2/n) * eig_val + eig_val

expl_var <- eig_val/p
CI_var <- eig_val_CI/p

cumexpl_var <- cumsum(expl_var)
cumCI_var <- cumsum(CI_var)
tab <- cbind(eig_val, expl_var, CI_var, cumexpl_var, cumCI_var) %>% round(3)
colnames(tab) <- c('eigenvalue', 'variance', 'variance_CI', 'cumulative_variance', 'cumulative_variance_CI')
tab

COL <- (eig_val_CI >= 1)
COL[COL==TRUE] <- 'red'
COL[COL==FALSE] <- 'black'

plot(eig_val_CI, type='b', pch=19, main='scree_plot', xlab='component', ylab='eigenvalue', col=COL)
abline(h=1, col="red")

# 10 components selected with updated Kaiser Rule, corresponding to 0.851 of total variance
# 13 components selected with fraction of explained variance > 0.9

P <- eig_vec[,1:10]
PCA_data <- as.matrix(data) %*% P %>% scale

# ---- H Clustering ----

ds <- dist(PCA_data)

clusters <- function(H){
        data.frame( row.names = paste0('Cluster', seq_along(H$height)),
                                height = H$height,
                                components = ifelse(H$merge<0, H$labels[abs(H$merge)], paste0('Cluster', H$merge)),
                                stringsAsFactors = FALSE
                )
}

sin_linkage <- hclust(ds, method='single')
#plot(sin_linkage, main='single')
#rect.hclust(sin_linkage, k=5)

cpl_linkage <- hclust(ds, method='complete')
#plot(cpl_linkage, main='complete')

# ---- Optimal Number of clusters ----

optimal_number <- function(ds, method, max_K, target='wss', plot=FALSE){
        valid_target <- list('within_variance', 'wss')

        out <- list()
        number_clust <- c()
        obj <- c()


        clust <- hclust(ds, method=method)
        numbers <- 2:max_K

        for(K in numbers){
                data$cluster <- cutree(clust, k=K)
                temp <- rep(NA, K)

                # minimize target
                for(k in 1:K) {
                        clusterk <- data[data$cluster == k, ]
                        clusterk <- clusterk[, -c(length(clusterk))]     # drop the clusters column

                        if(target=='within_variance'){
                                if(nrow(clusterk) > 1)
                                        temp[k] <- sum((clusterk - colMeans(clusterk))^2) /(nrow(clusterk)-1)
                                else
                                        temp[k] <- 0
                        }
                        if(target=='wss')
                                temp[k] <- sum((clusterk - colMeans(clusterk))^2)
                }

                obj <- obj %>% append(sum(temp))
        }

        opt <- min(numbers[obj == min(as.numeric(obj))])

        if(plot){
                plot(numbers, type='l', obj,
                        xlab='number_of_clusters', ylab=target,
                        main=paste0(method, '_linkage', '\n',
                                                'optimal_number_of_clusters:_', opt, '\n',
                                                target, ':_', obj[numbers == opt] %>% round(2)
                        )
                )
                abline(v=opt, col='red')
        }

        out$k <- opt

        if(target=='within_variance')
                out$summary <- data.frame('K' = numbers, 'within_variance' = obj)
        if(target=='wss')
                out$summary <- data.frame('K' = numbers, 'wss' = obj)

        return(out)
}

sin_opt <- optimal_number(ds, 'single', 200, 'wss', plot=FALSE)
cpl_opt <- optimal_number(ds, 'complete', 200, 'wss', plot=FALSE)

plot(sin_opt$summary$K, sin_opt$summary$wss, type='l',
        main='single_linkage', xlab='number_of_clusters', ylab='wss'
        ); abline(v=30, col='red')
plot(cpl_opt$summary$K, cpl_opt$summary$wss, type='l',
        main='complete_linkage', xlab='number_of_clusters', ylab='wss'
        ); abline(v=12, col='red')

sin_opt$k <- 30
cpl_opt$k <- 12

plot(sin_linkage, main=paste0('single_linkage_\n_optimal_number_of_clusters_:_', sin_opt$k), labels=FALSE);
rect.hclust(sin_linkage, k=sin_opt$k)
plot(cpl_linkage, main=paste0('complete_linkage_\n_optimal_number_of_clusters_:_', cpl_opt$k), labels=FALSE);
rect.hclust(cpl_linkage, k=cpl_opt$k)

# ---- Optimal dataset ----

sin_top <- cutree(sin_linkage, k=sin_opt$k)
cpl_top <- cutree(cpl_linkage, k=cpl_opt$k)

data_sin <- data.frame(PCA_data, 'cluster'=sin_top)
data_cpl <- data.frame(PCA_data, 'cluster'=cpl_top)

aggregate(data_sin$cluster, by=list(data_sin$cluster), FUN=length)
aggregate(data_cpl$cluster, by=list(data_cpl$cluster), FUN=length)

# ---- Cluster Visualization ----

data_single <- list('data'=data, 'cluster'=sin_top)
data_complete <- list('data'=data, 'cluster'=cpl_top)

fviz_cluster(data_single,
                                main = 'single_linkage',
                             frame.type = 'convex',
                             geom = 'point',
                             labelsize = 1,
) + theme_minimal()

fviz_cluster(data_complete,
                             main = 'complete_linkage',
                             frame.type = 'convex',
                             geom = 'point',
                             labelsize = 1,
) + theme_minimal()

# ---- elbow kmeans----

n <- nrow(PCA_data)
times <- 20
TIMES <- 40
N <- n%/%10
wss <- rep(NA, N)
wss_std <- rep(NA, N)
wssK <- rep(NA, times)
numbers <- 1:N
elbow <- rep(NA, TIMES)


for(T in 1:TIMES){
        for(K in numbers){
                for(t in 1:times)
                        wssK[t] <- kmeans(PCA_data, centers=K)$tot.withinss
                wss[K] <- mean(wssK)
                wss_std[K] <- sd(wssK)
        }

        for(x in 1:(length(wss)-1)){
                if(wss[x]-wss_std[x] < wss[x+1] + wss_std[x+1]){
                        elbow[T] <- x
                        break
                }
        }
}
boxplot(elbow, main=paste0('elbow_\n_mean:_', mean(elbow)))

qplot(numbers, wss, main=paste0('elbow:_', elbow[TIMES])) +
        geom_errorbar(aes(x=numbers, ymin=wss-wss_std, ymax=wss+wss_std), width=0.25) +
        theme_minimal() +
        geom_vline(xintercept=elbow[TIMES])

#---- 11means ----

algo <- kmeans(PCA_data, centers=11)

aggregate(algo$cluster, by=list(algo$cluster), FUN=length)
data_kmeans <- list('data'=data, 'cluster'=algo$cluster)

fviz_cluster(data_kmeans,
                             main = '11means',
                             frame.type = 'convex',
                             geom = 'point',
                             labelsize = 1,
) + theme_minimal()

# ---- R squared ----

rsq <- function(data, cluster_label, r=2, target='data'){

        if(target=='cluster'){

                data <- as.matrix(data)
                n <- nrow(data)
                p <- ncol(data)


                for(i in 1:(ncol(data)-1)){
                        RSS <- anova( aov( cluster_label ~ data ))[2,2] # RSS from linear regression
                        TSS <- sum((cluster_label-mean(cluster_label))^2)
                        rsq <- 1 - RSS/TSS * (n-1)/(n-p-1)
                }

                return(round(rsq, r))
        }


        if(target=='data'){

                data <- data.frame(data)
                data$cluster <- cluster_label
                rsq <- rep(NA, ncol(data)-1)

                for(i in 1:(ncol(data)-1)){
                        RSS <- anova( aov( data[,i] ~ data$cluster))[2,2] # RSS from linear regression
                        TSS <- sum((data[[i]])^2)
                        rsq[i] <- 1 - RSS/TSS
```

```
                    }

                    data <- data[, -ncol(data)]

            }

            rsq <- round(rsq, r)
            RSQ <- cbind(colnames(data), rsq) %>% data.frame(stringsAsFactors=FALSE)
            colnames(RSQ) <- c('variable', 'r_squared')
            #RSQ <- RSQ[order(RSQ$r_squared), ]

            RSQ[dim(RSQ)[1]+1, ] <- c('mean', mean(as.numeric(RSQ[[2]])) %>% round(n)) # Add mean rsq
            return(RSQ)
}

comp_data <- PCA_data

rsq_comparison <- merge(rsq(comp_data, sin_top), rsq(comp_data, cpl_top),
                                                        by.x='variable', by.y='variable') %>%
                                    merge(rsq(comp_data, algo$cluster),
                                                        by.x='variable', by.y='variable')
colnames(rsq_comparison) <- c('variable', 'rsq_single_linkage', 'rsq_complete_linkage', 'rsq_11means')
rsq_comparison

# ---- in R squared ----

comp_data <- PCA_data
rsq_inv_comp <- list('rsq_single_linkage' = rsq(comp_data, sin_top, target='cluster'),
                                            'rsq_complete_linkage' = rsq(comp_data, cpl_top, target='cluster'),
                                            'rsq_11means' = rsq(comp_data, algo$cluster, target='cluster'))
rsq_inv_comp


comp_data <- data
rsq_inv_comp <- list('rsq_single_linkage' = rsq(comp_data, sin_top, target='cluster'),
                                            'rsq_complete_linkage' = rsq(comp_data, cpl_top, target='cluster'),
                                            'rsq_11means' = rsq(comp_data, algo$cluster, target='cluster'))
rsq_inv_comp
```