# STA5077Z: Assignment 1: Question 1

Natalie Alexander

2023-09-23

## A. Libraries

## B. Objective 1: Cluster the data into 3 clusters:

## 1. Exploratory Data Analysis

### 1.1. Reading in the data

```
##############################Read in Data##################################

#set path to access data files
##CHANGE PATH ACCORDING TO WHERE THE DATA FILE IS STORED ON YOUR COMPUTER!
path = "C:\\Users\\natal\\OneDrive - University of Cape
Town\\UCT\\Unsupervised\\Assignments\\Zazo_Assignment1"


##CHANGE PATH ACCORDING TO WHERE THE TABLES AND FIGURES WILL BE SAVED
##ON YOUR COMPUTER!
path_to_figures_tables = "C:\\Users\\natal\\OneDrive - University of Cape
Town\\UCT\\Unsupervised\\Assignments\\Zazo_Assignment1\\submit\\Question1\\fi
gures_tables"


##read in data
foetal_health_data = read.csv(paste0(path, "\\fetal_health.csv"), sep =";")
```

### 1.2 Inspect data dimensions

```
#check dimensions of data

##dimensions of data set:
dim(foetal_health_data) #2126 row by 21 column data matrix


##number of columns:
ncol(foetal_health_data) #21 columns

##number of rows:
nrow(foetal_health_data) #2126 rows
```

### 1.3. Check for missing data

```
##check for missing data
foetal_health_data[!complete.cases(foetal_health_data),]
```

### 1.4. Assess column names

```
##########################Inspect column names##############################

#column names
colnames(foetal_health_data)

#change column names: Remove spaces and capitalize
colnames(foetal_health_data) <- c('Baseline value', 'Accelerations',
                                  'Fetal movement', 'Uterine contractions',
                                  'Light decelerations','Severe
decelerations',
                                  'Prolongued decelerations',
                                  'Abnormal short term variability',
                                  'Mean value of short term variability',
                                  'Percentage of time with abnormal long term
variability',
                                  'Mean value of long term variability',
                                  'Histogram width', 'Histogram min',
                                  'Histogram max', 'Histogram number of
peaks',
                                  'Histogram number of zeroes',
                                  'Histogram mode', 'Histogram mean',
                                  'Histogram median','Histogram variance',
                                  'Histogram tendency')
```

## 1.5. Head and tail of data

```
##########################Inspect data #####################################

#check head of data
(head_of_foetal_health_data = head(foetal_health_data))

#save table data
repmod::make_word_table(head_of_foetal_health_data[,1:7],
                        paste0(path_to_figures_tables,
                               "\\head_of_foetal_health_data_columns1-7"),
                        info = NULL, use.rownames = TRUE)

repmod::make_word_table(head_of_foetal_health_data[,8:14],
                        paste0(path_to_figures_tables,
                               "\\head_of_foetal_health_data_columns_8-14"),
                        info = NULL, use.rownames = TRUE)

repmod::make_word_table(head_of_foetal_health_data[,15:21],
                        paste0(path_to_figures_tables,
                               "\\head_of_foetal_health_data_columns_15-21"),
                        info = NULL, use.rownames = TRUE)


#check tail of data
(tail_of_foetal_health_data = tail(foetal_health_data))

##save table data
repmod::make_word_table(tail_of_foetal_health_data[,1:7],
                        paste0(path_to_figures_tables,
                               "\\tail_of_foetal_health_data_columns1-7"),
                        info = NULL, use.rownames = TRUE)

repmod::make_word_table(tail_of_foetal_health_data[,8:14],
                        paste0(path_to_figures_tables,
                               "\\tail_of_foetal_health_data_columns_8-14"),
                        info = NULL, use.rownames = TRUE)

repmod::make_word_table(tail_of_foetal_health_data[,15:21],
                        paste0(path_to_figures_tables,
                               "\\tail_of_foetal_health_data_columns_15-21"),
                        info = NULL, use.rownames = TRUE)
```

## 1.6. Check data types

```
################################check data types################################

#check data types
str(foetal_health_data)
```

## 1.7. Summary statistics

```
#############################summary data####################################

#get summary stats
summary_stats = round(t(data.frame(do.call(cbind,
                                           lapply(foetal_health_data,
summary)
                                           ))), 3)

#additional summary statistics
##standard deviation
standard_deviation = lapply(foetal_health_data, sd)
##round off to 3 decimal places
standard_deviation = lapply(standard_deviation,  round, 3)

##variance
variance = lapply(foetal_health_data, var)
##round off to 3 decimal places
variance = lapply(variance,  round, 3)

##append standard deviation and variance to summary statistics data frame
summary_stats = cbind(summary_stats, "Variance" = variance,
                      "Standard deviation" = standard_deviation)

#save table data
repmod::make_word_table(summary_stats,
                        paste0(path_to_figures_tables,
                               "\\summary_stats"),
                        info = NULL,
                        use.rownames = TRUE)

#view summary_stats
head(summary_stats)
```

## 1.8. Box plots of non-histogram independent variables

```
#Boxplot of non-histogram independent variables

##create long format table
df_of_non_hist_variables = foetal_health_data %>%
  dplyr::select(`Baseline value`:`Mean value of long term variability`) %>%
   gather(key = 'Variable', value = 'Value')

##generate boxplots
(boxplot_of_non_hist_variables = df_of_non_hist_variables  %>% ggplot(aes(x =
"",
  y = Value, fill=Variable)) +
   facet_wrap(~ Variable, scales = 'free',labeller = label_wrap_gen(width =
25)) +
    geom_boxplot() +
    labs(x = NULL, y = "Value")+
  theme_linedraw()+
   scale_fill_brewer(palette="Set3")+
  theme(legend.position = "none",strip.text = element_text(size = 7)))

#save image
ggsave(paste0(path_to_figures_tables, "\\boxplot_of_non_hist_variables.png"),
boxplot_of_non_hist_variables, dpi = 300)
```

## 1.9. Box plots of histogram independent variables

```
#Boxplot of histogram independent variables

##create long format table
df_of_hist_variables = foetal_health_data %>%
  dplyr::select(`Histogram width`:`Histogram tendency`) %>%
   gather(key = 'Variable', value = 'Value')

##generate boxplots
(boxplot_of_hist_variables = df_of_hist_variables %>% ggplot(aes(x = "", y =
Value,
fill=Variable)) +
   facet_wrap(~ Variable, scales = 'free',labeller = label_wrap_gen(width =
25)) +
    geom_boxplot() +
    labs(x = NULL, y = "Value")+
  theme_linedraw()+
   scale_fill_brewer(palette="Set3")+
  theme(legend.position = "none",strip.text = element_text(size = 9)))

#save image
ggsave(paste0(path_to_figures_tables, "\\boxplot_of_hist_variables.png"),
       boxplot_of_hist_variables , dpi = 300)
```

## 1.10. Density plot of non-histogram independent variables

```
#Density plot of non-histogram independent variables

##create long format table
df_of_non_hist_variables <- foetal_health_data %>%
  dplyr::select(`Baseline value`:`Mean value of long term variability`) %>%
  gather(key = 'Variable', value = 'Value')

##generate density plot
(density_plot_of_non_hist_variables <- df_of_non_hist_variables %>%
  ggplot(aes(x = Value, fill = Variable)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ Variable, scales = 'free', labeller = label_wrap_gen(width =
25)) +
  labs(x = NULL, y = "Density") +
  scale_fill_brewer(palette = "Set3") +
     theme_linedraw()+
     theme_linedraw()+
  theme(legend.position = "none", strip.text = element_text(size = 9),
        axis.text.x = element_text(angle = 90, vjust = 0.5)))

#save image
ggsave(paste0(path_to_figures_tables,
"\\density_plot_of_non_hist_variables.png"),
       density_plot_of_non_hist_variables, dpi = 300)
```

## 1.11. Density plot of histogram independent variables

```
#Density plot of histogram independent variables

##create long format table
df_of_hist_variables <- foetal_health_data %>%
  dplyr::select(`Histogram width`:`Histogram tendency`) %>%
  gather(key = 'Variable', value = 'Value')

##generate density plot
(density_plot_of_hist_variables <- df_of_hist_variables %>%
  ggplot(aes(x = Value, fill = Variable)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ Variable, scales = 'free', labeller = label_wrap_gen(width =
25)) +
  labs(x = NULL, y = "Density") +
  scale_fill_brewer(palette = "Set3") +
  theme(legend.position = "none", strip.text = element_text(size = 9)))

#save image
ggsave(paste0(path_to_figures_tables,
"\\density_plot_of_hist_variables.png"), density_plot_of_hist_variables, dpi
= 300)
```

## 1.12 Correlation analysis

```
#save image
png(paste0(path_to_figures_tables, "\\cor_plot.png"))

#generate correlation matrix
cor_matrix = cor(foetal_health_data, method="pearson")
corrplot::corrplot(cor_matrix, tl.cex=0.7)

dev.off()

#remove large correlations
foetal_health_data= foetal_health_data %>% select(-c("Histogram mode",
                                                     "Histogram median",
                                                     "Histogram mean",
                                                     "Histogram width",
                                                     "Histogram tendency"))
```

## 2. Principal component analysis (PCA)

### 2.1 Standardize data

```
#standardize #center = True and scale = True
df_scaled_foetal_health = scale(foetal_health_data, center=T, scale=T)
```

### 2.2 Performing PCA

```
#principal component analysis on correlation matrix
## corelation between rows or "observations"
pca.result = princomp(df_scaled_foetal_health, cor=TRUE, scores=TRUE)

#prcomp: loadings
loadings = pca.result$loadings

#prcomp: eigenvalues or variance
eigenvals = pca.result$sdev^2

#PCA scores
scores = pca.result$scores

#proportion of explainable variance by components
var <- eigenvals/sum(eigenvals)

#cumulative proportion of variance
(cumsums = round(cumsum(var), 3))
```

## 2.3. Plot principal components vs cumulative proportion of variance.

```
#save image
png(paste0(path_to_figures_tables, "\\cumsums.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

#plot
plot(x=1:length(cumsums), cumsums, ylab="Cumulative proportion of variance",
     xlab="Principal component", type="b", pch=16,  xaxt="n")
axis(1, at = 1:length(df_scaled_foetal_health), las=2)
abline(h=0.80, lty=2, lwd=3, col="red")
legend("bottomright", legend="Threshold for reliable proportion of
       explainable variance = 0.80 ", title = "Legend", cex=0.9,
title.font=2,
       lty=2, lwd=1, col="red")

dev.off()


#view plot
plot(x=1:length(cumsums), cumsums, ylab="Cumulative proportion of variance",
     xlab="Principal component", type="b", pch=16,  xaxt="n")
axis(1, at = 1:length(df_scaled_foetal_health), las=2)
abline(h=0.80, lty=2, lwd=3, col="red")
legend("bottomright", legend="Threshold for reliable proportion of
       explainable variance = 0.80 ", title = "Legend", cex=0.9,
title.font=2,
       lty=2, lwd=1, col="red")
```

## 2.4. Screeplot of principal components vs eigenvalues

```r
#save plot
png(paste0(path_to_figures_tables, "\\eigenvals.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

#plot
plot(var, xlab = "Principal component",
     ylab = "Eigenvalues",
     type = "b", pch = 20, xaxt="n")
axis(1, at = 1:length(df_scaled_foetal_health), las=2)

dev.off()

#view plot
plot(var, xlab = "Principal component",
     ylab = "Eigenvalues",
     type = "b", pch = 20, xaxt="n")
axis(1, at = 1:length(df_scaled_foetal_health), las=2)
```

## 2.5. Extract principal components

```
#Extract first 2 principal components
two_pcs = scores[, 1:2]
```

## 3. Clustering

### 3.1. Proximity measures

```
#############################Proximity
measurements#############################

#calculate distance matrices

##euclidean distance
euclid_dist = dist(two_pcs, method = "euclidean")

## manhattan distance
manhat_dist = dist(two_pcs, method = "manhattan")

## chebyshev maximum distance
max_dist = dist(two_pcs, method = "maximum")


#observe distances between observations
round(as.matrix(euclid_dist)[1:6, 1:6], 3)
round(as.matrix(manhat_dist)[1:6, 1:6], 3)
round(as.matrix(max_dist)[1:6, 1:6], 3)
```

## 3.2. Model building and model performance

```
#performance metrics data frame
metrics = data.frame(matrix(nrow=0, ncol=12))

#column names
colnames(metrics) = c("Model ID", "Model type", "Distance measure",
                      "Linkage", "Model params",
                      "Cophenetic cor",
                      "Average silhouette width",
                      "Gap stat", "Avg Jaccard similarity",
                      "Avg instability", "Avg within cluster dissim",
                      "Avg between cluster dissim")
```

## 3.3. Hierarchicial agglomerative clustering

### 3.3.1. Complete linkage and Euclidean distance

```
#Hclust- Euclidean distance using complete linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.complete <- hclust(euclid_dist, method="complete")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.complete, k = 3)

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "euclidean"),
method="complete"),
                        k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)
```

```r
#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.complete),euclid_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="euclidean",
                         method ="complete",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#fill in data
metrics[1, 1] = 1
metrics[1, 2] = "hclust"
metrics[1, 3] = "Euclidean"
metrics[1, 4] = "Complete"
metrics[1, 5] = "k=3"
metrics[1, 6] = coph.cor
metrics[1, 7] = avg.sil.width
metrics[1, 8] = avg.gap
metrics[1, 9] = round(overal.mean.Jaccard, 3)
metrics[1, 10] = round(overal.mean.instability, 3)
metrics[1, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[1, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

#############################################################################
####

#cluster plot using plotly
library(plotly)
fig1 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
)%>%
  add_markers(size = 12)
fig1 <- fig1 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))
```

```
############################################################################
####

#dendrogram
dend <- as.dendrogram(model.hclust.complete)
labels_colors(dend) <- "white"
dend1 <- color_branches(dend, k = 3)

############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.2. Complete linkage and Manhattan distance

```r
#Hclust- Manhattan distance using complete linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.complete <- hclust(manhat_dist, method="complete")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.complete, k = 3)

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
method="complete"),
                        k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.complete), manhat_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="manhattan",
                         method ="complete",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)
```

```r
#fill in data
metrics[2, 1] = 2
metrics[2, 2] = "hclust"
metrics[2, 3] = "Manhattan"
metrics[2, 4] = "Complete"
metrics[2, 5] = "k=3"
metrics[2, 6] = coph.cor
metrics[2, 7] = avg.sil.width
metrics[2, 8] = avg.gap
metrics[2, 9] = round(overal.mean.Jaccard, 3)
metrics[2, 10] = round(overal.mean.instability, 3)
metrics[2, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[2, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

#############################################################################
###

#cluster plot using plotly
library(plotly)
fig2 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig2 <- fig2 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

#############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.complete)
labels_colors(dend) <- "white"
dend2 <- color_branches(dend, k = 3)

#############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.3. Complete linkage and Maximum distance

```
#Hclust- Maximum distance using complete linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.complete <- hclust(max_dist, method="complete")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.complete, k = 3)

##silhouette width
sil.width <- silhouette(clusters, max_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "maximum"),
method="complete"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.complete), max_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                          B=1000,
                          clustermethod=hclustCBI,
                          k=k,
                          metric="maximum",
                          method ="complete",
                          count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)
```

```r
#fill in data
metrics[3, 1] = 3
metrics[3, 2] = "hclust"
metrics[3, 3] = "Maximum"
metrics[3, 4] = "Complete"
metrics[3, 5] = "k=3"
metrics[3, 6] = coph.cor
metrics[3, 7] = avg.sil.width
metrics[3, 8] = avg.gap
metrics[3, 9] = round(overal.mean.Jaccard, 3)
metrics[3, 10] = round(overal.mean.instability, 3)
metrics[3, 11] = round(cluster.stats(max_dist, clusters)$average.within, 3)
metrics[3, 12] = round(cluster.stats(max_dist, clusters)$average.between, 3)

###############################################################################
###

#cluster plot
library(plotly)
fig3 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig3 <- fig3 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

###############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.complete)
labels_colors(dend) <- "white"
dend3 <- color_branches(dend, k = 3)

###############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.4. Single linkage and Euclidean distance

```
#Hclust- Euclidean distance using single linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.single <- hclust(euclid_dist, method="single")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.single, k = 3)

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "euclidean"),
method="single"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.single),euclid_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="euclidean",
                         method ="single",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```r
##

#fill in data
metrics[4, 1] = 4
metrics[4, 2] = "hclust"
metrics[4, 3] = "Euclidean"
metrics[4, 4] = "single"
metrics[4, 5] = "k=3"
metrics[4, 6] = coph.cor
metrics[4, 7] = avg.sil.width
metrics[4, 8] = avg.gap
metrics[4, 9] = round(overal.mean.Jaccard, 3)
metrics[4, 10] = round(overal.mean.instability, 3)
metrics[4, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[4, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

#############################################################################
###
#cluster plot using plotly

library(plotly)
fig4 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig4 <- fig4 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

#############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.single)
labels_colors(dend) <- "white"
dend4 <- color_branches(dend, k = 3)

#############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.5. Single linkage and Manhattan distance

```
##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.single <- hclust(manhat_dist, method="single")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.single, k = 3)

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
method="single"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.single), manhat_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="manhattan",
                         method ="single",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
##
```

```r
#fill in data
metrics[5, 1] = 5
metrics[5, 2] = "hclust"
metrics[5, 3] = "Manhattan"
metrics[5, 4] = "single"
metrics[5, 5] = "k=3"
metrics[5, 6] = coph.cor
metrics[5, 7] = avg.sil.width
metrics[5, 8] = avg.gap
metrics[5, 9] = round(overal.mean.Jaccard, 3)
metrics[5, 10] = round(overal.mean.instability, 3)
metrics[5, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[5, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

############################################################################
#####

#cluster plot using plotly
library(plotly)
fig5 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig5 <- fig5 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.single)
labels_colors(dend) <- "white"
dend5 <- color_branches(dend, k = 3)

############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.6. Single linkage and Maximum distance

```r
#Hclust- Maximum distance using single linkage
```

```r
##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.single <- hclust(max_dist, method="single")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.single, k = 3)

##silhouette width
sil.width <- silhouette(clusters, max_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "maximum"), method="single"),
k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.single), max_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="maximum",
                         method ="single",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
##

#fill in data
```

```r
metrics[6, 1] = 6
metrics[6, 2] = "hclust"
metrics[6, 3] = "Maximum"
metrics[6, 4] = "single"
metrics[6, 5] = "k=3"
metrics[6, 6] = coph.cor
metrics[6, 7] = avg.sil.width
metrics[6, 8] = avg.gap
metrics[6, 9] = round(overal.mean.Jaccard, 3)
metrics[6, 10] = round(overal.mean.instability, 3)
metrics[6, 11] = round(cluster.stats(max_dist, clusters)$average.within, 3)
metrics[6, 12] = round(cluster.stats(max_dist, clusters)$average.between, 3)

################################################################################
###

#cluster plot using plotly
library(plotly)
fig6 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig6 <- fig6 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


################################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.single)
labels_colors(dend) <- "white"
dend6 <- color_branches(dend, k = 3)

################################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.7. Average linkage and Euclidean distance

```
#Hclust- Euclidean distance using average linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.average <- hclust(euclid_dist, method="average")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.average, k = 3)

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "euclidean"),
method="average"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.average),euclid_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="euclidean",
                         method ="average",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
```

```
##

#fill in data
metrics[7, 1] = 7
metrics[7, 2] = "hclust"
metrics[7, 3] = "Euclidean"
metrics[7, 4] = "average"
metrics[7, 5] = "k=3"
metrics[7, 6] = coph.cor
metrics[7, 7] = avg.sil.width
metrics[7, 8] = avg.gap
metrics[7, 9] = round(overal.mean.Jaccard, 3)
metrics[7, 10] = round(overal.mean.instability, 3)
metrics[7, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[7, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

############################################################################
###

#cluster plot using plotly
library(plotly)
fig7 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig7 <- fig7 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


#dendrogram
dend <- as.dendrogram(model.hclust.average)
labels_colors(dend) <- "white"
dend7 <- color_branches(dend, k = 3)

############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.8. Average linkage and Manhattan distance

```r
#Hclust- Manhattan distance using average linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.average <- hclust(manhat_dist, method="average")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.average, k = 3)

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
method="average"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.average), manhat_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="manhattan",
                         method ="average",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```r
##

#fill in data
metrics[8, 1] = 8
metrics[8, 2] = "hclust"
metrics[8, 3] = "Manhattan"
metrics[8, 4] = "average"
metrics[8, 5] = "k=3"
metrics[8, 6] = coph.cor
metrics[8, 7] = avg.sil.width
metrics[8, 8] = avg.gap
metrics[8, 9] = round(overal.mean.Jaccard, 3)
metrics[8, 10] = round(overal.mean.instability, 3)
metrics[8, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[8, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

############################################################################
###

#cluster plot using plotly
library(plotly)
fig8 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig8 <- fig8 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


#dendrogram
dend <- as.dendrogram(model.hclust.average)
labels_colors(dend) <- "white"
dend8 <- color_branches(dend, k = 3)

############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.9. Average linkage and Maximum distance

```
#Hclust- Maximum distance using average linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.average <- hclust(max_dist, method="average")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.average, k = 3)

##silhouette width
sil.width <- silhouette(clusters, max_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "maximum"),
method="average"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.average), max_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="maximum",
                         method ="average",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```
##

#fill in data
metrics[9, 1] = 9
metrics[9, 2] = "hclust"
metrics[9, 3] = "Maximum"
metrics[9, 4] = "average"
metrics[9, 5] = "k=3"
metrics[9, 6] = coph.cor
metrics[9, 7] = avg.sil.width
metrics[9, 8] = avg.gap
metrics[9, 9] = round(overal.mean.Jaccard, 3)
metrics[9, 10] = round(overal.mean.instability, 3)
metrics[9, 11] = round(cluster.stats(max_dist, clusters)$average.within, 3)
metrics[9, 12] = round(cluster.stats(max_dist, clusters)$average.between, 3)

############################################################################
#####

#cluster plot using plotly
library(plotly)
fig9 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                color = clusters, colors = c('#EF553B', '#636EFA', '#00CC96')
) %>%
  add_markers(size = 12)
fig9 <- fig9 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


#dendrogram
dend <- as.dendrogram(model.hclust.average)
labels_colors(dend) <- "white"
dend9 <- color_branches(dend, k = 3)


############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.10. Median linkage and Euclidean distance

```r
#Hclust- Euclidean distance using median linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.median <- hclust(euclid_dist, method="median")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.median, k = 3)

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute median silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "euclidean"),
method="median"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.median),euclid_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="euclidean",
                         method ="median",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```
##

#fill in data
metrics[10, 1] = 10
metrics[10, 2] = "hclust"
metrics[10, 3] = "Euclidean"
metrics[10, 4] = "median"
metrics[10, 5] = "k=3"
metrics[10, 6] = coph.cor
metrics[10, 7] = avg.sil.width
metrics[10, 8] = avg.gap
metrics[10, 9] = round(overal.mean.Jaccard, 3)
metrics[10, 10] = round(overal.mean.instability, 3)
metrics[10, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[10, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

##############################################################################
###

#cluster plot using plotly
library(plotly)
fig10 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig10 <- fig10 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.median)
labels_colors(dend) <- "white"
dend10 <- color_branches(dend, k = 3)


##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.11. Median linkage and Manhattan distance

```r
#Hclust- Manhattan distance using median linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.median <- hclust(manhat_dist, method="median")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.median, k = 3)

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute median silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
method="median"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.median), manhat_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="manhattan",
                         method ="median",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
```

```
##

#fill in data
metrics[11, 1] = 11
metrics[11, 2] = "hclust"
metrics[11, 3] = "Manhattan"
metrics[11, 4] = "median"
metrics[11, 5] = "k=3"
metrics[11, 6] = coph.cor
metrics[11, 7] = avg.sil.width
metrics[11, 8] = avg.gap
metrics[11, 9] = round(overal.mean.Jaccard, 3)
metrics[11, 10] = round(overal.mean.instability, 3)
metrics[11, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[11, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

#############################################################################
####

#cluster plot using plotly
library(plotly)
fig11 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig11 <- fig11 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

#############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.median)
labels_colors(dend) <- "white"
dend11 <- color_branches(dend, k = 3)


#############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.12. Median linkage and Maximum distance

```
#Hclust- Maximum distance using median linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.median <- hclust(max_dist, method="median")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.median, k = 3)

##silhouette width
sil.width <- silhouette(clusters, max_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute median silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "maximum"), method="median"),
k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.median), max_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="maximum",
                         method ="median",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
```

```
##

#fill in data
metrics[12, 1] = 12
metrics[12, 2] = "hclust"
metrics[12, 3] = "Maximum"
metrics[12, 4] = "median"
metrics[12, 5] = "k=3"
metrics[12, 6] = coph.cor
metrics[12, 7] = avg.sil.width
metrics[12, 8] = avg.gap
metrics[12, 9] = round(overal.mean.Jaccard, 3)
metrics[12, 10] = round(overal.mean.instability, 3)
metrics[12, 11] = round(cluster.stats(max_dist, clusters)$average.within, 3)
metrics[12, 12] = round(cluster.stats(max_dist, clusters)$average.between, 3)

##############################################################################
###

#cluster plot using plotly
library(plotly)
fig12 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig12 <- fig12 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.median)
labels_colors(dend) <- "white"
dend12 <- color_branches(dend, k = 3)

##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.13. Centroid linkage and Euclidean distance

```
#Hclust- Euclidean distance using centroid linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.centroid <- hclust(euclid_dist, method="centroid")


##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.centroid, k = 3)

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute centroid silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "euclidean"),
method="centroid"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.centroid),euclid_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="euclidean",
                         method ="centroid",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)
```

```
###########################################################################
##

#fill in data
metrics[13, 1] = 13
metrics[13, 2] = "hclust"
metrics[13, 3] = "Euclidean"
metrics[13, 4] = "centroid"
metrics[13, 5] = "k=3"
metrics[13, 6] = coph.cor
metrics[13, 7] = avg.sil.width
metrics[13, 8] = avg.gap
metrics[13, 9] = round(overal.mean.Jaccard, 3)
metrics[13, 10] = round(overal.mean.instability, 3)
metrics[13, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[13, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

###########################################################################
###

#cluster plot using plotly
library(plotly)
fig13 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig13 <- fig13 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

###########################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.centroid)
labels_colors(dend) <- "white"
dend13 <- color_branches(dend, k = 3)


###########################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.14. Centroid linkage and Manhattan distance

```r
#Hclust- Manhattan distance using centroid linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.centroid <- hclust(manhat_dist, method="centroid")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.centroid, k = 3)

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute centroid silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
method="centroid"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.centroid), manhat_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="manhattan",
                         method ="centroid",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
```

```
##

#fill in data
metrics[14, 1] = 14
metrics[14, 2] = "hclust"
metrics[14, 3] = "Manhattan"
metrics[14, 4] = "centroid"
metrics[14, 5] = "k=3"
metrics[14, 6] = coph.cor
metrics[14, 7] = avg.sil.width
metrics[14, 8] = avg.gap
metrics[14, 9] = round(overal.mean.Jaccard, 3)
metrics[14, 10] = round(overal.mean.instability, 3)
metrics[14, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[14, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

##############################################################################
####

#cluster plot using plotly
library(plotly)
fig14 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig14 <- fig14 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.centroid)
labels_colors(dend) <- "white"
dend14 <- color_branches(dend, k = 3)


##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.15. Centroid linkage and Maximum distance

```r
#Hclust- Maximum distance using centroid linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.centroid <- hclust(max_dist, method="centroid")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.centroid, k = 3)

##silhouette width
sil.width <- silhouette(clusters, max_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute centroid silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "maximum"),
method="centroid"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.centroid), max_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                          B=1000,
                          clustermethod=hclustCBI,
                          k=k,
                          metric="maximum",
                          method ="centroid",
                          count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```r
##

#fill in data
metrics[15, 1] = 15
metrics[15, 2] = "hclust"
metrics[15, 3] = "Maximum"
metrics[15, 4] = "centroid"
metrics[15, 5] = "k=3"
metrics[15, 6] = coph.cor
metrics[15, 7] = avg.sil.width
metrics[15, 8] = avg.gap
metrics[15, 9] = round(overal.mean.Jaccard, 3)
metrics[15, 10] = round(overal.mean.instability, 3)
metrics[15, 11] = round(cluster.stats(max_dist, clusters)$average.within, 3)
metrics[15, 12] = round(cluster.stats(max_dist, clusters)$average.between, 3)

###############################################################################
######

#cluster plot using plotly
library(plotly)
fig15 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') )%>%
  add_markers(size = 12)
fig15 <- fig15 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

###############################################################################
####

#dendrogram
dend <- as.dendrogram(model.hclust.centroid)
labels_colors(dend) <- "white"
dend15 <- color_branches(dend, k = 3)


###############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.16. Ward D linkage and Euclidean distance

```
#Hclust- Euclidean distance using ward.D linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.ward.D <- hclust(euclid_dist, method="ward.D")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.ward.D, k = 3)

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute ward.D silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "euclidean"),
method="ward.D"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.ward.D),euclid_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="euclidean",
                         method ="ward.D",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
```

```r
##

#fill in data
metrics[16, 1] = 16
metrics[16, 2] = "hclust"
metrics[16, 3] = "Euclidean"
metrics[16, 4] = "ward.D"
metrics[16, 5] = "k=3"
metrics[16, 6] = coph.cor
metrics[16, 7] = avg.sil.width
metrics[16, 8] = avg.gap
metrics[16, 9] = round(overal.mean.Jaccard, 3)
metrics[16, 10] = round(overal.mean.instability, 3)
metrics[16, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[16, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

##############################################################################
####

#cluster plot using plotly
library(plotly)
fig16 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig16 <- fig16 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
####

#dendrogram
dend <- as.dendrogram(model.hclust.ward.D)
labels_colors(dend) <- "white"
dend16 <- color_branches(dend, k = 3)


##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.17. Ward D linkage and Manhattan distance

```
#Hclust- Manhattan distance using ward.D linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.ward.D <- hclust(manhat_dist, method="ward.D")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.ward.D, k = 3)

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute ward.D silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
method="ward.D"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.ward.D), manhat_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="manhattan",
                         method ="ward.D",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```r
##

#fill in data
metrics[17, 1] = 17
metrics[17, 2] = "hclust"
metrics[17, 3] = "Manhattan"
metrics[17, 4] = "ward.D"
metrics[17, 5] = "k=3"
metrics[17, 6] = coph.cor
metrics[17, 7] = avg.sil.width
metrics[17, 8] = avg.gap
metrics[17, 9] = round(overal.mean.Jaccard, 3)
metrics[17, 10] = round(overal.mean.instability, 3)
metrics[17, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[17, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

##############################################################################
#####

#cluster plot using plotly
library(plotly)
fig17 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig17 <- fig17 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
####

#dendrogram
dend <- as.dendrogram(model.hclust.ward.D)
labels_colors(dend) <- "white"
dend17 <- color_branches(dend, k = 3)


##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.18. Ward D linkage and Maximum distance

```r
#Hclust- Maximum distance using ward.D linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.ward.D <- hclust(max_dist, method="ward.D")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.ward.D, k = 3)

##silhouette width
sil.width <- silhouette(clusters, max_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute ward.D silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "maximum"), method="ward.D"),
k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.ward.D), max_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="maximum",
                         method ="ward.D",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```r
##

#fill in data
metrics[18, 1] = 18
metrics[18, 2] = "hclust"
metrics[18, 3] = "Maximum"
metrics[18, 4] = "ward.D"
metrics[18, 5] = "k=3"
metrics[18, 6] = coph.cor
metrics[18, 7] = avg.sil.width
metrics[18, 8] = avg.gap
metrics[18, 9] = round(overal.mean.Jaccard, 3)
metrics[18, 10] = round(overal.mean.instability, 3)
metrics[18, 11] = round(cluster.stats(max_dist, clusters)$average.within, 3)
metrics[18, 12] = round(cluster.stats(max_dist, clusters)$average.between, 3)

############################################################################
###

#cluster plot using plotly
library(plotly)
fig18 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig18 <- fig18 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

############################################################################
####

#dendrogram
dend <- as.dendrogram(model.hclust.ward.D)
labels_colors(dend) <- "white"
dend18 <- color_branches(dend, k = 3)


############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.19. Ward D2 linkage and Euclidean distance

```r
#Hclust- Euclidean distance using ward.D2 linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.ward.D2 <- hclust(euclid_dist, method="ward.D2")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.ward.D2, k = 3)

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute ward.D2 silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "euclidean"),
method="ward.D2"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.ward.D2),euclid_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="euclidean",
                         method ="ward.D2",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```
##

#fill in data
metrics[19, 1] = 19
metrics[19, 2] = "hclust"
metrics[19, 3] = "Euclidean"
metrics[19, 4] = "ward.D2"
metrics[19, 5] = "k=3"
metrics[19, 6] = coph.cor
metrics[19, 7] = avg.sil.width
metrics[19, 8] = avg.gap
metrics[19, 9] = round(overal.mean.Jaccard, 3)
metrics[19, 10] = round(overal.mean.instability, 3)
metrics[19, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[19, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

##############################################################################
###

#cluster plot using plotly
library(plotly)
fig19  <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                  color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig19 <- fig19 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
####

#dendrogram
dend <- as.dendrogram(model.hclust.ward.D2)
labels_colors(dend) <- "white"
dend19 <- color_branches(dend, k = 3)

##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.20. Ward D2 linkage and Manhattan distance

```
#Hclust- Manhattan distance using ward.D2 linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.ward.D2 <- hclust(manhat_dist, method="ward.D2")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.ward.D2, k = 3)

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute ward.D2 silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
method="ward.D2"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.ward.D2), manhat_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="manhattan",
                         method ="ward.D2",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```
##

#fill in data
metrics[20, 1] = 20
metrics[20, 2] = "hclust"
metrics[20, 3] = "Manhattan"
metrics[20, 4] = "ward.D2"
metrics[20, 5] = "k=3"
metrics[20, 6] = coph.cor
metrics[20, 7] = avg.sil.width
metrics[20, 8] = avg.gap
metrics[20, 9] = round(overal.mean.Jaccard, 3)
metrics[20, 10] = round(overal.mean.instability, 3)
metrics[20, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[20, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

###############################################################################
###

#cluster plot using plotly
library(plotly)
fig20 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig20 <- fig20 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

###############################################################################
###

#dendrogram
dend <- as.dendrogram(model.hclust.ward.D2)
labels_colors(dend) <- "white"
dend20 <- color_branches(dend, k = 3)

###############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.3.21. Ward D2 linkage and Maximum distance

```
#Hclust- Maximum distance using ward.D2 linkage

##number of clusters
k <- 3

##clustering model
set.seed(123)
model.hclust.ward.D2 <- hclust(max_dist, method="ward.D2")

##cut tree at k = 3 clusters
clusters <- cutree(model.hclust.ward.D2, k = 3)

##silhouette width
sil.width <- silhouette(clusters, max_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute ward.D2 silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "maximum"),
method="ward.D2"), k = k))
}

gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 3,
                    B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#cophenetic correlation
coph.cor = round(cor(cophenetic(model.hclust.ward.D2), max_dist), 3)

#bootstrap
bootstrap.clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=hclustCBI,
                         k=k,
                         metric="maximum",
                         method ="ward.D2",
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
```

```
##


#fill in data
metrics[21, 1] = 21
metrics[21, 2] = "hclust"
metrics[21, 3] = "Maximum"
metrics[21, 4] = "ward.D2"
metrics[21, 5] = "k=3"
metrics[21, 6] = coph.cor
metrics[21, 7] = avg.sil.width
metrics[21, 8] = avg.gap
metrics[21, 9] = round(overal.mean.Jaccard, 3)
metrics[21, 10] = round(overal.mean.instability, 3)
metrics[21, 11] = round(cluster.stats(max_dist, clusters)$average.within, 3)
metrics[21, 12] = round(cluster.stats(max_dist, clusters)$average.between, 3)

##############################################################################
###

#cluster plot using plotly
library(plotly)
fig21 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig21 <- fig21 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
####

#dendrogram
dend <- as.dendrogram(model.hclust.ward.D2)
labels_colors(dend) <- "white"
dend21 <- color_branches(dend, k = 3)


##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

## 3.4 Partitioning clustering algorithms

### 3.4.1 K-means

```r
#K-means

##number of clusters
k <- 3

##clustering model
set.seed(123)
kmeans.out = kmeans(two_pcs, centers = k, nstart=100, iter.max=1000)


#clusters
clusters = kmeans.out$cluster

##silhouette width with euclidean distance
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
gap_stat <- clusGap(two_pcs, FUN = kmeans, nstart = 25, K.max = 3, B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#bootstrap
bootstrap_clusters <- clusterboot(two_pcs,
                        B=1000,
                        clustermethod=kmeansCBI,
                        k=3,
                        count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
##

#fill in data
metrics[22, 1] = 22
metrics[22, 2] = "K-means"
metrics[22, 3] = "Euclidean"
metrics[22, 4] = ""
metrics[22, 5] = "centers=3, nstart=100, iter.max=1000"
metrics[22, 6] = ""
metrics[22, 7] = avg.sil.width
```

```
metrics[22, 8] = avg.gap
metrics[22, 9] = round(overal.mean.Jaccard, 3)
metrics[22, 10] = round(overal.mean.instability, 3)
metrics[22, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[22, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

##############################################################################
###

#cluster plot using plotly
library(plotly)
fig22 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig22 <- fig22 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))

##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.4.2 K-mediods- PAM using Euclidean distance

```r
#K-mediods: PAM using Euclidean distance

##number of clusters
k <- 3

##clustering model
set.seed(123)
pam.euclid = pam(two_pcs, k=3, metric="euclidean")

#clusters
clusters = pam.euclid$cluster

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average  silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
gap_stat <- clusGap(two_pcs, FUN = pam, K.max = 3, B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#bootstrap
bootstrap_clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=pamkCBI,
                         metric="euclidean",
                         k=3,
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
##

#fill in data
metrics[23, 1] = 23
metrics[23, 2] = "PAM"
metrics[23, 3] = "Euclidean"
metrics[23, 4] = ""
metrics[23, 5] = "k=3, metric=euclidean"
metrics[23, 6] = ""
metrics[23, 7] = avg.sil.width
```

```
metrics[23, 8] = avg.gap
metrics[23, 9] = round(overal.mean.Jaccard, 3)
metrics[23, 10] = round(overal.mean.instability, 3)
metrics[23, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[23, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

################################################################################
###

#cluster plot using plotly
library(plotly)
fig23 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig23 <- fig23 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


################################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.4.3 K-mediods- PAM using Manhattan distance

```r
#K-mediods: PAM using Manhattan distance

##number of clusters
k <- 3

##clustering model
set.seed(123)
pam.manhat = pam(two_pcs, k=3, metric="manhattan")

#clusters
clusters = pam.manhat$cluster

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average  silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
gap_stat <- clusGap(two_pcs, FUN = pam, K.max = 3, B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#bootstrap
bootstrap_clusters <- clusterboot(two_pcs,
                                  B=1000,
                                  clustermethod=pamkCBI,
                                  metric="manhattan",
                                  k=3,
                                  count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

#############################################################################
##

#fill in data
metrics[24, 1] = 24
metrics[24, 2] = "PAM"
metrics[24, 3] = "Manhattan"
metrics[24, 4] = ""
metrics[24, 5] = "k=3, metric=manhattan"
metrics[24, 6] = ""
metrics[24, 7] = avg.sil.width
```

```r
metrics[24, 8] = avg.gap
metrics[24, 9] = round(overal.mean.Jaccard, 3)
metrics[24, 10] = round(overal.mean.instability, 3)
metrics[24, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[24, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

##############################################################################
##

#cluster plot using plotly
library(plotly)
fig24 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') )%>%
  add_markers(size = 12)
fig24 <- fig24 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


##############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.4.4 K-mediods- CLARA using Euclidean distance

```r
#K-mediods: CLARA using Euclidean distance

##number of clusters
k <- 3

##clustering model
set.seed(123)
clara.euclid = clara(two_pcs, k=3, metric="euclidean")

#clusters
clusters = clara.euclid$cluster

##silhouette width
sil.width <- silhouette(clusters, euclid_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average  silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
gap_stat <- clusGap(two_pcs, FUN = clara, K.max = 3, B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#bootstrap
bootstrap_clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=claraCBI,
                         metric="euclidean",
                         k=3,
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

###############################################################################
##


#fill in data
metrics[25, 1] = 25
metrics[25, 2] = "CLARA"
metrics[25, 3] = "Euclidean"
metrics[25, 4] = ""
metrics[25, 5] = "k=3, metric=euclidean"
metrics[25, 6] = ""
```

```
metrics[25, 7] = avg.sil.width
metrics[25, 8] = avg.gap
metrics[25, 9] = round(overal.mean.Jaccard, 3)
metrics[25, 10] = round(overal.mean.instability, 3)
metrics[25, 11] = round(cluster.stats(euclid_dist, clusters)$average.within,
3)
metrics[25, 12] = round(cluster.stats(euclid_dist, clusters)$average.between,
3)

#############################################################################
###

#cluster plot using plotly
library(plotly)
fig25 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig25 <- fig25 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


#############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

### 3.4.5 K-mediods- CLARA using Manhattan distance

```
#K-mediods: CLARA using Manhattan distance

##number of clusters
k <- 3

##clustering model
set.seed(123)
clara.manhat = clara(two_pcs, k=3, metric="manhattan")

#clusters
clusters = clara.manhat$cluster

##silhouette width
sil.width <- silhouette(clusters, manhat_dist)
sil.width.values <- sil.width[, "sil_width"]

# Compute average  silhouette width
avg.sil.width <- round(mean(sil.width.values), 3)

#gap statistic
gap_stat <- clusGap(two_pcs, FUN = clara, K.max = 3, B = 50)
avg.gap = round((gap_stat$Tab[3,"gap"]), 3)

#bootstrap
bootstrap_clusters <- clusterboot(two_pcs,
                         B=1000,
                         clustermethod=claraCBI,
                         metric="manhattan",
                         k=3,
                         count=FALSE)

#AvgJaccard <0.6 is unstable and AvgJaccard >0.85 is highly stable
mean.Jaccard = bootstrap.clusters$bootmean
overal.mean.Jaccard = mean(mean.Jaccard)
instability = bootstrap.clusters$bootbrd/1000
overal.mean.instability = mean(instability)

##############################################################################
##

#fill in data
metrics[26, 1] = 26
metrics[26, 2] = "CLARA"
metrics[26, 3] = "Manhattan"
metrics[26, 4] = ""
metrics[26, 5] = "k=3, metric=manhattan"
metrics[26, 6] = ""
metrics[26, 7] = avg.sil.width
```

```
metrics[26, 8] = avg.gap
metrics[26, 9] = round(overal.mean.Jaccard, 3)
metrics[26, 10] = round(overal.mean.instability, 3)
metrics[26, 11] = round(cluster.stats(manhat_dist, clusters)$average.within,
3)
metrics[26, 12] = round(cluster.stats(manhat_dist, clusters)$average.between,
3)

###############################################################################
##

#cluster plot using plotly
library(plotly)
fig26 <- plot_ly(as.data.frame(two_pcs), x = ~Comp.1, y = ~Comp.2,
                 color = clusters, colors = c('#EF553B', '#636EFA',
'#00CC96') ) %>%
  add_markers(size = 12)
fig26 <- fig26 %>%
  layout(
    title = "",
    scene = list(bgcolor = "#e5ecf6"))


###############################################################################
####

#cluster plot using clusplot
clusplot(two_pcs, clusters, col.clus=c( '#EF553B','#636EFA','#00CC96'),
         color=T, shade=T, main= "", col.p =c("darkred","steelblue",
"darkgreen"))
```

## 4. Save to RData and save performance metrics to table

```r
# #save r output at rdata
# save.image(file = "Question1_submit.RData")
#
#
# #save metrics data
repmod::make_word_table(metrics,
                        paste0(path_to_figures_tables,
                               "\\metrics"),
                        info = NULL, use.rownames = TRUE)
```

## 5. Dendrograms

```r
#Complete linkage
png(paste0(path_to_figures_tables, "\\dend_complete.png"),
    width = 15, height =8, units="cm", pointsize=9, res=300)
par(mfrow=c(1, 3))
plot(dend1,  ylab="Height", cex.main=0.9, main= "A") #euclid
plot(dend2,  ylab="Height", cex.main=0.9, main= "B") #manhat
plot(dend3,  ylab="Height", cex.main=0.9, main= "C") #max
dev.off()

#Single linkage
png(paste0(path_to_figures_tables, "\\dend_single.png"),
    width = 15, height =8, units="cm", pointsize=9, res=300)
par(mfrow=c(1, 3))
plot(dend4,  ylab="Height", cex.main=0.9, main= "A") #euclid
plot(dend5,  ylab="Height", cex.main=0.9, main= "B") #manhat
plot(dend6,  ylab="Height", cex.main=0.9, main= "C") #max
dev.off()

#average linkage
png(paste0(path_to_figures_tables, "\\dend_average.png"),
    width = 15, height =8, units="cm", pointsize=9, res=300)
par(mfrow=c(1, 3))
plot(dend7,  ylab="Height", cex.main=0.9, main= "A") #euclid
plot(dend8,  ylab="Height", cex.main=0.9, main= "B") #manhat
plot(dend9,  ylab="Height", cex.main=0.9, main= "C") #max
dev.off()

#median linkage
png(paste0(path_to_figures_tables, "\\dend_median.png"),
    width = 15, height =8, units="cm", pointsize=9, res=300)
par(mfrow=c(1, 3))
plot(dend10,  ylab="Height", cex.main=0.9, main= "A") #euclid
plot(dend11,  ylab="Height", cex.main=0.9, main= "B") #manhat
plot(dend12,  ylab="Height", cex.main=0.9, main= "C") #max
dev.off()

#centroid
png(paste0(path_to_figures_tables, "\\dend_centroid.png"),
    width = 15, height =8, units="cm", pointsize=9, res=300)
par(mfrow=c(1, 3))
plot(dend13,  ylab="Height", cex.main=0.9, main= "A") #euclid
plot(dend14,  ylab="Height", cex.main=0.9, main= "B") #manhat
plot(dend15,  ylab="Height", cex.main=0.9, main= "C") #max
dev.off()

#Ward D
png(paste0(path_to_figures_tables, "\\dend_wardD.png"),
    width = 15, height =8, units="cm", pointsize=9, res=300)
```

```
par(mfrow=c(1, 3))
plot(dend16,  ylab="Height", cex.main=0.9, main= "A") #euclid
plot(dend17,  ylab="Height", cex.main=0.9, main= "B") #manhat
plot(dend18,  ylab="Height", cex.main=0.9, main= "C") #max
dev.off()

#Ward D2
png(paste0(path_to_figures_tables, "\\dend_wardD2.png"),
    width = 15, height =8, units="cm", pointsize=9, res=300)
par(mfrow=c(1, 3))
plot(dend19,  ylab="Height", cex.main=0.9, main= "A") #euclid
plot(dend20,  ylab="Height", cex.main=0.9, main= "B") #manhat
plot(dend21,  ylab="Height", cex.main=0.9, main= "C") #max
dev.off()


##############################################################################
###
```

## 6. Clusterplot of best model

```
fig22 # best model
```

## 7. Average silhouette width plot of best model

```
# K-means : model 22

##number of clusters
k <- 3

##clustering model
set.seed(123)
kmeans.out = kmeans(two_pcs, centers = k, nstart=100, iter.max=1000)
clusters = kmeans.out$cluster

# save silhouette plot
png(paste0(path_to_figures_tables, "\\best_model22_sil_plot.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)
plot(silhouette(clusters, euclid_dist),  border = NA, main = "",
     col=c( "lightblue","lightgreen", "pink"))
dev.off()

#observe silhouette plot
png(paste0(path_to_figures_tables, "\\best_model22_sil_plot.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)
plot(silhouette(clusters, euclid_dist),  border = NA, main = "",
     col=c( "lightblue","lightgreen", "pink"))
```

## C. Objective 2: Investigate if three foetal health classes are appropriate.

### 1. NbClust

```
#nbclust library
library(NbClust)

#nbclust model
set.seed(123)
nb_clust_model <- NbClust(two_pcs, distance = "euclidean",min.nc = 2, max.nc
          = 20, method = "kmeans", index ="all")

# save histogram of number of clusters
png(paste0(path_to_figures_tables, "\\nbclust.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

hist(nb_clust_model$Best.nc[1,],breaks=0:21,col="darkgreen",
     xlab="Optimal number of clusters", xaxt="n", main="")
axis(1, at = 0:20, las=2)

dev.off()

#view histogram of number of clusters
hist(nb_clust_model$Best.nc[1,],breaks=0:21,col="darkgreen",
     xlab="Optimal number of clusters", xaxt="n", main="")
axis(1, at = 0:20, las=2)
```

## 2. Elbow method for K-means clustering

### Goodness of fit:

- Increase in number of clusters (k), results in a decrease in within cluster deviation.

```r
#Elbow method for k-means clustering

set.seed(123)
k.max <- 20 # maximum number of clusters
df.out <- two_pcs
#Compute within sum of squares for k = 2 to k = 20
wss <- sapply(2:k.max,
        function(k){kmeans(df.out, k, nstart=500,
iter.max=1000)$tot.withinss})

#save plot number of clusters vs within sum of squares
png(paste0(path_to_figures_tables, "\\k_vs_wss.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

plot(2:k.max, wss,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters (k)",
       ylab="Total within-cluster sum of squares", xaxt="n", main="")
axis(1, at = 0:20, las=2)
abline(v = 3, lty =2, lwd=2, col="red")
abline(v = 6, lty =2, lwd=2, col="red")
legend("topright", lty =2, lwd=2, col="red", title="Legend",
       legend="Optimal number of clusters", title.font=2)

dev.off()

#view plot number of clusters vs within sum of squares
plot(2:k.max, wss,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters (k)",
       ylab="Total within-cluster sum of squares", xaxt="n", main="")
axis(1, at = 0:20, las=2)
abline(v = 3, lty =2, lwd=2, col="red")
abline(v = 6, lty =2, lwd=2, col="red")
legend("topright", lty =2, lwd=2, col="red", title="Legend",
       legend="Optimal number of clusters", title.font=2)
```

## 3.1 Average silhouette width for k using the K-means clustering method

```r
#Calc. average silhouette width for each value of k(no. of clusters = 2 to 
20)
kmax <- 20
asw <- numeric(kmax)

#get average silhouette width for each cluster k
for(k in 2:kmax){
  set.seed(123)
  kmeans.model = kmeans(two_pcs, centers = k, nstart=500, iter.max=1000)
  clusters = kmeans.model$cluster
  sil <- silhouette(clusters, dist(two_pcs, "manhattan"))
  asw[k] <- summary(sil)$avg.width}
k.best <- which.max(asw) #get k with largest silhouette width

#save plot number of clusters vs average silhouette width
png(paste0(path_to_figures_tables, "\\k_vs_silhouette1.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

plot(1:kmax, asw, type="h",
     main = "",
     xlab = "Number of clusters (k)", ylab = "Average silhouette width")
axis(1, k.best, paste("optimum", k.best, sep = "\n"), col = "red", font = 2,
     col.axis = "red")
points(k.best, max(asw), pch = 16, col = "red", cex = 1.5)
legend("topright", pch=16, col="red", title="Legend",
       legend="Optimal number of clusters", title.font=2, cex=0.8)

dev.off()

#view plot number of clusters vs average silhouette width
plot(1:kmax, asw, type="h",
     main = "",
     xlab = "Number of clusters (k)", ylab = "Average silhouette width")
axis(1, k.best, paste("optimum", k.best, sep = "\n"), col = "red", font = 2,
     col.axis = "red")
points(k.best, max(asw), pch = 16, col = "red", cex = 1.5)
legend("topright", pch=16, col="red", title="Legend",
       legend="Optimal number of clusters", title.font=2, cex=0.8)
```

## 3.2 Average silhouette width for k using the hierarchical clustering method

```
#clustering model
set.seed(123)
hclust.model <- hclust(manhat_dist, method="complete")

#Calc. average silhouette width for each value of k(no. of clusters = 2 to
20)
kmax <- 20
asw <- numeric(kmax)

for(k in 2:kmax){
  sil <- silhouette(cutree(hclust.model, k = k), dist(two_pcs, "manhattan"))
  asw[k] <- summary(sil)$avg.width}
k.best <- which.max(asw) #get k with largest silhouette width

# save plot number of clusters vs average silhouette width
png(paste0(path_to_figures_tables, "\\k_vs_silhouette2.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

plot(1:kmax, asw, type="h",
     main = "",
     xlab = "Number of clusters (k)", ylab = "Average silhouette width")
axis(1, k.best, paste("optimum", k.best, sep = "\n"), col = "red", font = 2,
     col.axis = "red")
points(k.best, max(asw), pch = 16, col = "red", cex = 1.5)
points(3, max(asw), pch = 16, col = "red", cex = 1.5)
legend("topright", pch=16, col="red", title="Legend",
       legend="Optimal number of clusters", title.font=2, cex=0.8)

dev.off()


#view plot number of clusters vs average silhouette width
plot(1:kmax, asw, type="h",
     main = "",
     xlab = "Number of clusters (k)", ylab = "Average silhouette width")
axis(1, k.best, paste("optimum", k.best, sep = "\n"), col = "red", font = 2,
     col.axis = "red")
points(k.best, max(asw), pch = 16, col = "red", cex = 1.5)
points(3, max(asw), pch = 16, col = "red", cex = 1.5)
legend("topright", pch=16, col="red", title="Legend",
       legend="Optimal number of clusters", title.font=2, cex=0.8)
```

### 4.1 Gap Statistic for K-means clustering

```
# Compute gap statistic
set.seed(123)
gap_stat <- clusGap(two_pcs, FUN = kmeans, nstart=25, iter.max=100,
                    K.max = 20, B = 50)

# save plot
png(paste0(path_to_figures_tables, "\\gapstat_kmeans.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

plot(gap_stat, frame = FALSE, xlab = "Number of clusters (k)", xaxt="n",
main="",
     ylim=c(0.6, 0.8))
axis(1, at = 1:20, las=2)
abline(v = 3, lty =2, lwd=2, col="darkblue")
legend("topright", lty =2, lwd=2, col="darkblue", title="Legend",
       legend="Optimal number of clusters", title.font=2)

dev.off()

#view plot
plot(gap_stat, frame = FALSE, xlab = "Number of clusters (k)", xaxt="n",
main="",
     ylim=c(0.6, 0.8))
axis(1, at = 1:20, las=2)
abline(v = 3, lty =2, lwd=2, col="darkblue")
legend("topright", lty =2, lwd=2, col="darkblue", title="Legend",
       legend="Optimal number of clusters", title.font=2)
```

### 4.2 Gap Statistic for hierarchical clustering, using complete linkage and the Manhattan distance

```r
# Compute gap statistic
set.seed(123)

##function to compute gap statistic
cluster_fun <- function(x, k, clust_method) {
  list(cluster = cutree(hclust(dist(x, method = "manhattan"),
                               method="complete"), k = k))
}

##gap stat
gap_stat <- clusGap(x=two_pcs, FUN = cluster_fun, K.max = 20,
                    B = 50)

#save plot
png(paste0(path_to_figures_tables, "\\gapstat_completeLinkage.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

plot(gap_stat, frame = FALSE, xlab = "Number of clusters (k)", xaxt="n",
main="",
     ylim=c(0.3, 0.7))
axis(1, at = 1:20, las=2)
abline(v = 3, lty =2, lwd=2, col="darkblue")
legend("topright", lty =2, lwd=2, col="darkblue", title="Legend",
       legend="Optimal number of clusters", title.font=2)

dev.off()

#view plot
png(paste0(path_to_figures_tables, "\\gapstat_completeLinkage.png"),
    width = 15, height =10, units="cm", pointsize=9, res=300)

plot(gap_stat, frame = FALSE, xlab = "Number of clusters (k)", xaxt="n",
main="",
     ylim=c(0.3, 0.7))
axis(1, at = 1:20, las=2)
abline(v = 3, lty =2, lwd=2, col="darkblue")
legend("topright", lty =2, lwd=2, col="darkblue", title="Legend",
       legend="Optimal number of clusters", title.font=2)

dev.off()
```