Problem Set 7

Akira Masuda

2020/3/15

```
Course: MACS30100 Perspectives on Computational Modeling (Winter 2020)
Author: Akira Masuda (ID: alakira)
```

```
knitr::opts_chunk$set(message = FALSE, warning = FALSE)
knitr::opts_chunk$set(fig.width=6,fig.height=3.4,fig.align='center')
```

```
library(tidyverse)
library(tidymodels)
library(knitr)
library(patchwork)
library(rcfss)
library(e1071)
library(caret)
library(pROC)
library(cluster)
library(kernlab)
library(Rtsne)
rm(list=ls())
set.seed(1100)
```

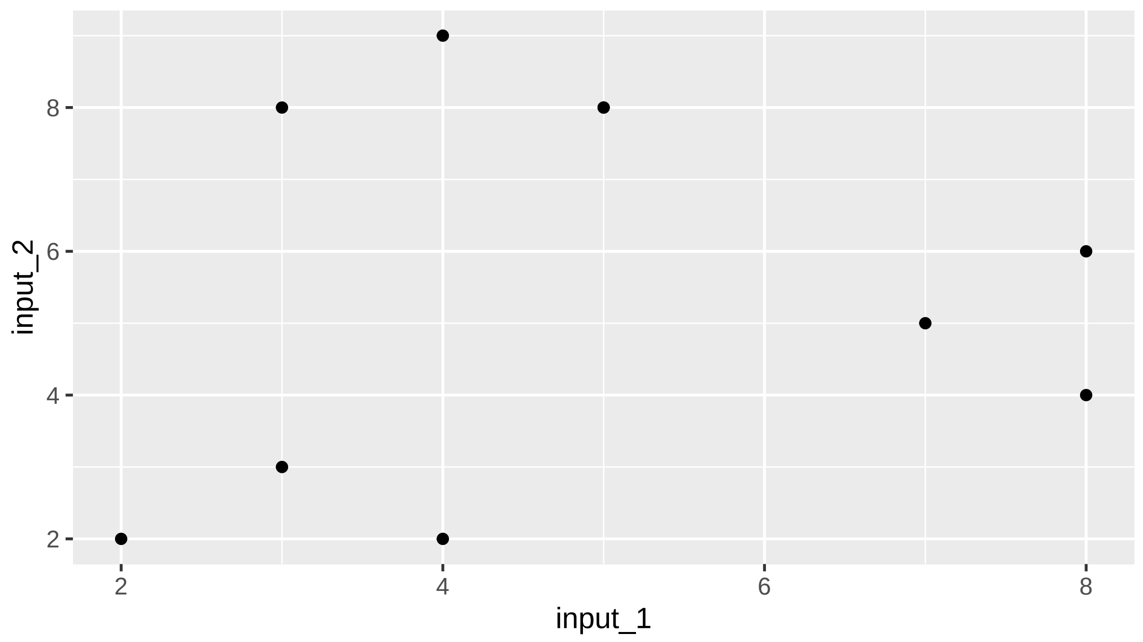
```
# set up parallelization for quicker computation time
library(doParallel)
cl <- makePSOCKcluster(3)
registerDoParallel(cl)

# set seed
set.seed(110)
```

---

k-Means Clustering By Hand

1.

```
# Plot given observations
input_1 <- c(5,8,7,8,3,4,2,3,4,5)
input_2 <- c(8,6,5,4,3,2,2,8,9,8)
df_hand <- data.frame(input_1 = input_1, input_2 = input_2)
df_hand %>%
  ggplot(aes(x=input_1, y=input_2)) +
  geom_point()
```



```
# Assigning each observation to a cluster at random
(df_hand <- df_hand %>%
  mutate(clust = factor(sample(0:2, 10, replace=TRUE))))
```

input_1	input_2	clust
5	8	1
8	6	2
7	5	0
8	4	2
3	3	2
4	2	0
2	2	2
3	8	2
4	9	2
5	8	2

2.

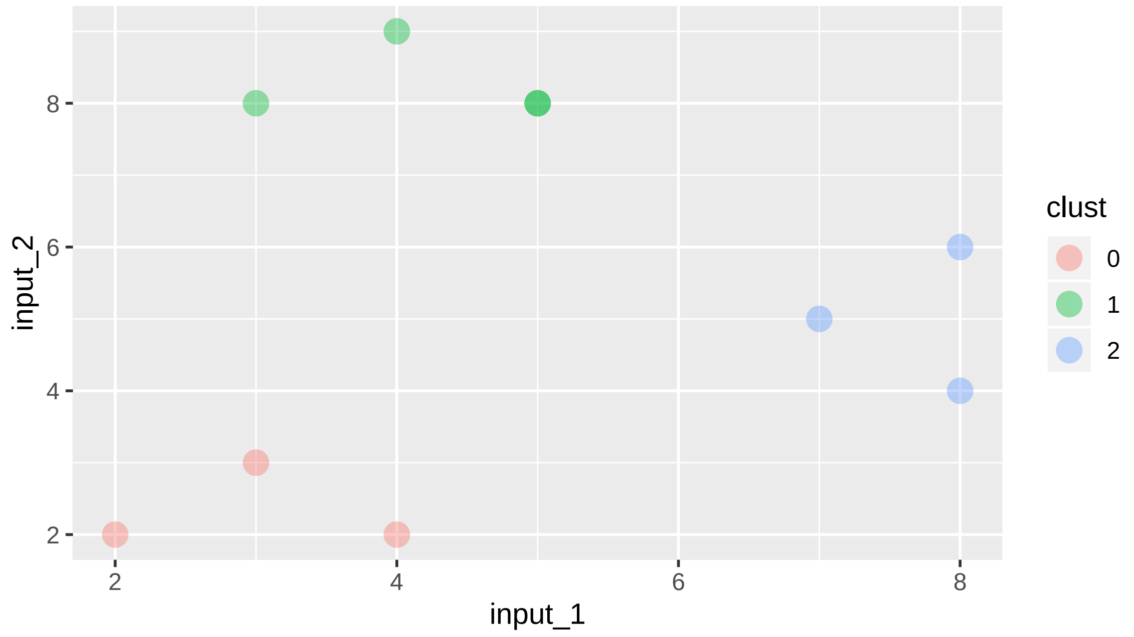
```
# Compute cluster centroids
(df_centroid <- df_hand %>%
  group_by(clust) %>%
  summarise_at(vars(input_1, input_2),funs(mean(., na.rm=TRUE))))
```

clust	input_1	input_2
0	5.500000	3.500000
1	5.000000	8.000000
2	4.714286	5.714286

```
# Loop k-Means algorithm until cluster assignments don't change
while (TRUE) {
    df_centroid <- df_hand %>%
      group_by(clust) %>%
      summarise_at(vars(input_1, input_2),funs(mean(., na.rm=TRUE)))
    df_hand <- df_hand %>%
    mutate(distance_0 = sqrt((input_1 - df_centroid$input_1[1])^2 +
             (input_2 - df_centroid$input_2[1])^2),
           distance_1 = sqrt((input_1 - df_centroid$input_1[2])^2 +
             (input_2 - df_centroid$input_2[2])^2),
           distance_2 = sqrt((input_1 - df_centroid$input_1[3])^2 +
             (input_2 - df_centroid$input_2[3])^2)) %>%
    mutate(new_clust = case_when(
      distance_0 < distance_1 & distance_0 < distance_2 ~ '0',
      distance_1 < distance_0 & distance_1 < distance_2 ~ '1',
      distance_2 < distance_0 & distance_2 < distance_1 ~ '2',
    ))
  if (sum(df_hand$clust == df_hand$new_clust)==10) {
    break
  }
  df_hand <- df_hand %>%
    mutate(clust = new_clust)
}
```

3.

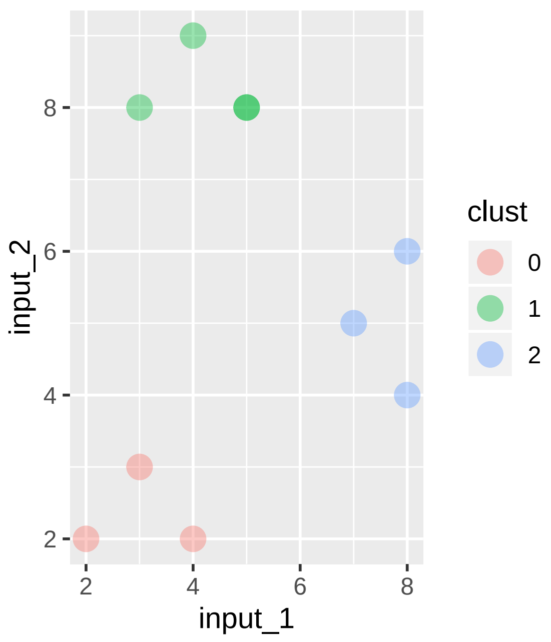
```
# Plot the final cluster assignments
df_hand %>%
  ggplot(aes(x=input_1, y=input_2, color=clust)) +
  geom_point(size=4, alpha=0.4)
```

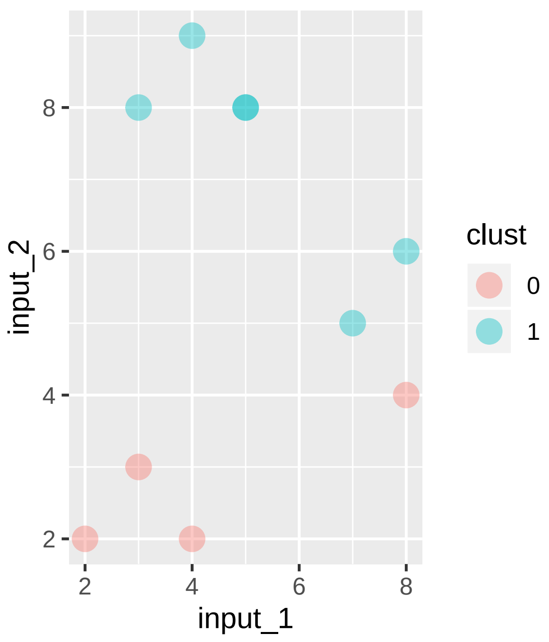


4.

```r
# Repeat the process with 2 clusters
df_hand2 <- data.frame(input_1 = input_1, input_2 = input_2)
df_hand2 <- df_hand2 %>%
  mutate(clust = factor(sample(0:1, 10, replace=TRUE)))
# Loop until each point's cluster doesn't change
while (TRUE) {
    df_centroid2 <- df_hand2 %>%
      group_by(clust) %>%
      summarise_at(vars(input_1, input_2),funs(mean(., na.rm=TRUE)))
    df_hand2 <- df_hand2 %>%
    mutate(distance_0 = sqrt((input_1 - df_centroid$input_1[1])^2 +
             (input_2 - df_centroid$input_2[1])^2),
           distance_1 = sqrt((input_1 - df_centroid$input_1[2])^2 +
             (input_2 - df_centroid$input_2[2])^2)) %>%
    mutate(new_clust = case_when(
      distance_0 < distance_1 ~ '0',
      distance_1 < distance_0 ~ '1',
    ))
  if (sum(df_hand2$clust == df_hand2$new_clust)==10) {
    break
  }
  df_hand2 <- df_hand2 %>%
    mutate(clust = new_clust)
}
```

```r
# Plot the final cluster assignments
p1 <- df_hand %>%
  ggplot(aes(x=input_1, y=input_2, color=clust)) +
  geom_point(size=4, alpha=0.4)
p2 <- df_hand2 %>%
  ggplot(aes(x=input_1, y=input_2, color=clust)) +
  geom_point(size=4, alpha=0.4)
gridExtra::grid.arrange(p1, p2, nrow = 1)
```

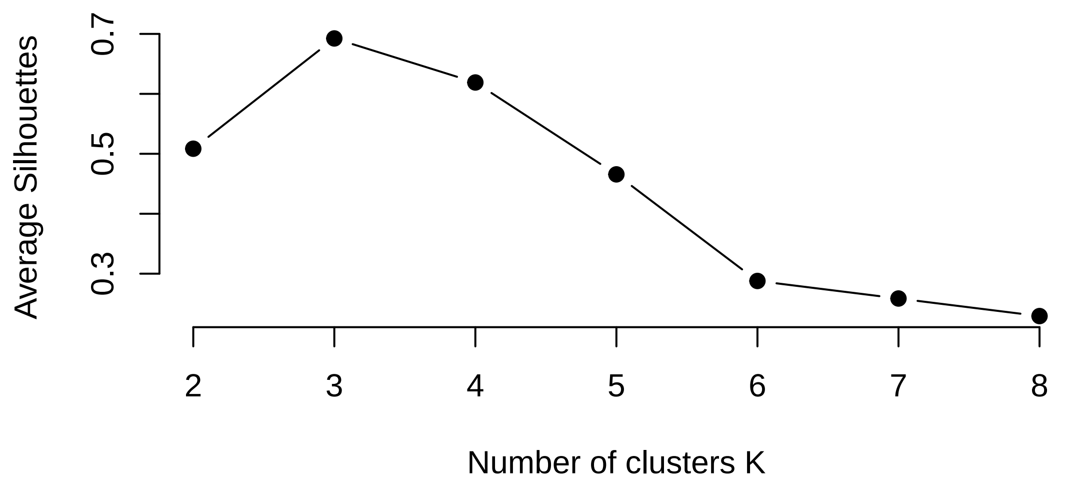




5.

To find the optimal number of clusters based on k-means clustering, I use silhouette analysis.

```r
df_hand3 <- data.frame(input_1 = input_1, input_2 = input_2)
# function to compute average silhouette for k clusters
avg_sil <- function(k) {
  km.res <- kmeans(df_hand3, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(df_hand3))
  mean(ss[, 3])
}
k.values <- 2:8

# extract avg silhouette
avg_sil_values <- map_dbl(k.values, avg_sil)

plot(k.values, avg_sil_values,
       type = "b", pch = 19, frame = FALSE,
       xlab = "Number of clusters K",
       ylab = "Average Silhouettes")
```



From the plot above, the optimal number of cluster is 3, which is also in line with our intuition that there are three clusters in the observations.

Application

Dimension Reduction

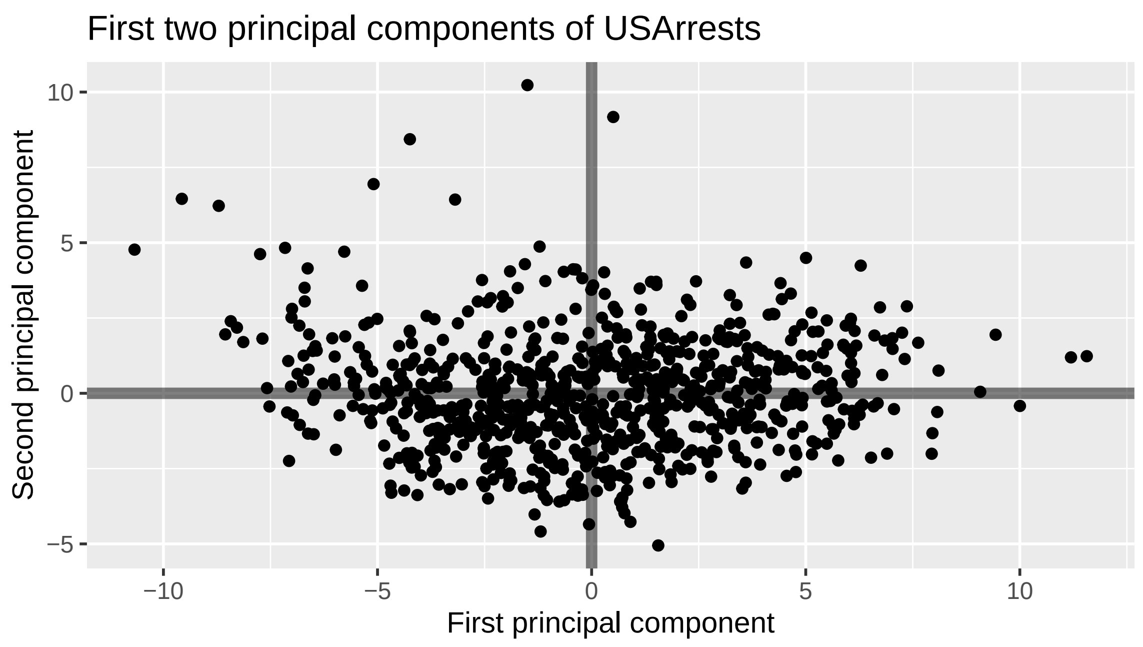
6.

```
# Loading data
df_wiki <- read_csv("data/wiki.csv")
```

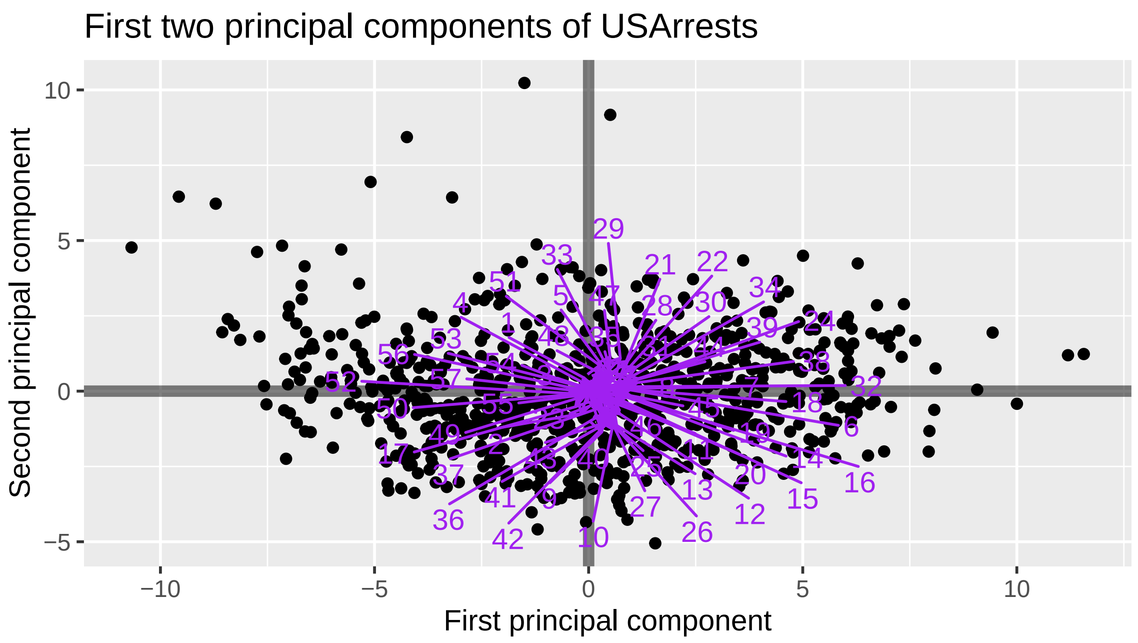
```
# Scaled
scaled_df <- df_wiki %>%
  mutate_at(.vars = vars(colnames(df_wiki)), scale)

# Covariance matrix
wiki_cov <- scaled_df %>%
  cov()

# Eigen vectors
wiki_eigen <- eigen(wiki_cov)

# recover PC loading vectors and extract first two loadings
phi <- wiki_eigen$vectors[, 1:2]
colnames(phi) <- c("PC1", "PC2")
phi
```

```
##                PC1          PC2
##  [1,] -0.021805412  0.088384981
##  [2,] -0.035086317 -0.149461450
##  [3,] -0.030501043  0.030435497
##  [4,] -0.034190490  0.062364714
##  [5,]  0.081363144  0.134387358
##  [6,]  0.192827065  0.008273053
##  [7,]  0.190587716  0.017668791
##  [8,]  0.210862567  0.028776289
##  [9,]  0.061228008 -0.271741292
## [10,]  0.113718709 -0.222367978
## [11,]  0.100218922 -0.068458517
## [12,]  0.145666175 -0.151011732
## [13,]  0.131109826 -0.227602424
## [14,]  0.178057029 -0.038122429
## [15,]  0.163777789 -0.066421876
## [16,]  0.157956174 -0.033472352
## [17,] -0.060796858 -0.103458415
## [18,]  0.183364593  0.010911790
## [19,]  0.171153058 -0.025207626
## [20,]  0.114558913 -0.056217768
## [21,]  0.175351292  0.197634740
## [22,]  0.160432141  0.111106146
## [23,]  0.077810159 -0.059774521
## [24,]  0.160803391  0.044003603
## [25,]  0.121658435 -0.229926005
## [26,]  0.117590405 -0.226760395
## [27,]  0.120376196 -0.242325421
## [28,]  0.181477170  0.197827499
## [29,]  0.147851769  0.218628501
## [30,]  0.218809245  0.155151571
## [31,]  0.214558397  0.160864524
## [32,]  0.206538888  0.029823253
## [33,]  0.102337996  0.114370782
## [34,]  0.103448162  0.018604706
## [35,]  0.109632421  0.094172517
## [36,]  0.080866885 -0.136967544
## [37,]  0.062216127 -0.106296824
## [38,]  0.226193061  0.056374273
## [39,]  0.230923964  0.083430888
## [40,]  0.104666756 -0.245439824
## [41,]  0.095802250 -0.202021404
## [42,]  0.081401727 -0.220985795
## [43,]  0.089707244 -0.202022006
## [44,]  0.208591685  0.070543836
## [45,]  0.195043150 -0.029560476
## [46,]  0.144023257 -0.126416909
## [47,]  0.099872875  0.228494272
## [48,]  0.110628098  0.076095685
## [49,]  0.021982007 -0.014536737
## [50,] -0.017157681 -0.015478496
## [51,]  0.051309109  0.171483803
## [52,] -0.094774659 -0.014887154
## [53,]  0.010922081  0.013134181
## [54,]  0.007123091  0.002311281
## [55,] -0.018040923 -0.023591030
## [56,]  0.004250607 -0.003784534
## [57,] -0.007848555 -0.005301224
```

```r
# calculate PC scores
PC1 <- as.matrix(select_if(scaled_df, is.numeric)) %*% phi[,1]
PC2 <- as.matrix(select_if(scaled_df, is.numeric)) %*% phi[,2]

# create data frame with PC scores
PC <- tibble(
  PC1 = PC1[,1],
  PC2 = PC2[,1]
)

# visualize scores
ggplot(PC, aes(PC1, PC2)) +
  geom_vline(xintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_hline(yintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_point() +
  labs(title = "First two principal components of USArrests",
       x = "First principal component",
       y = "Second principal component")
```



```r
# biplot
phi_df <- (phi * 5) %>%
  as.data.frame %>%
  rownames_to_column(var = "variable")

ggplot(PC, aes(PC1, PC2)) +
  geom_vline(xintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_hline(yintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_point() +
  geom_segment(data = phi_df,
               aes(x = 0, y = 0,
                   xend = PC1,
                   yend = PC2),
               arrow = arrow(length = unit(0.03, "npc")),
                color = "purple") +
  ggrepel::geom_text_repel(data = phi_df,
                           aes(x = PC1, y = PC2, label = variable),
                           color = "purple") +
  labs(title = "First two principal components of USArrests",
       x = "First principal component",
       y = "Second principal component")
```



```
phi_df %>%
  filter(PC1==max(PC1) | PC1==min(PC1) | PC2==max(PC2) | PC2==min(PC2))
```

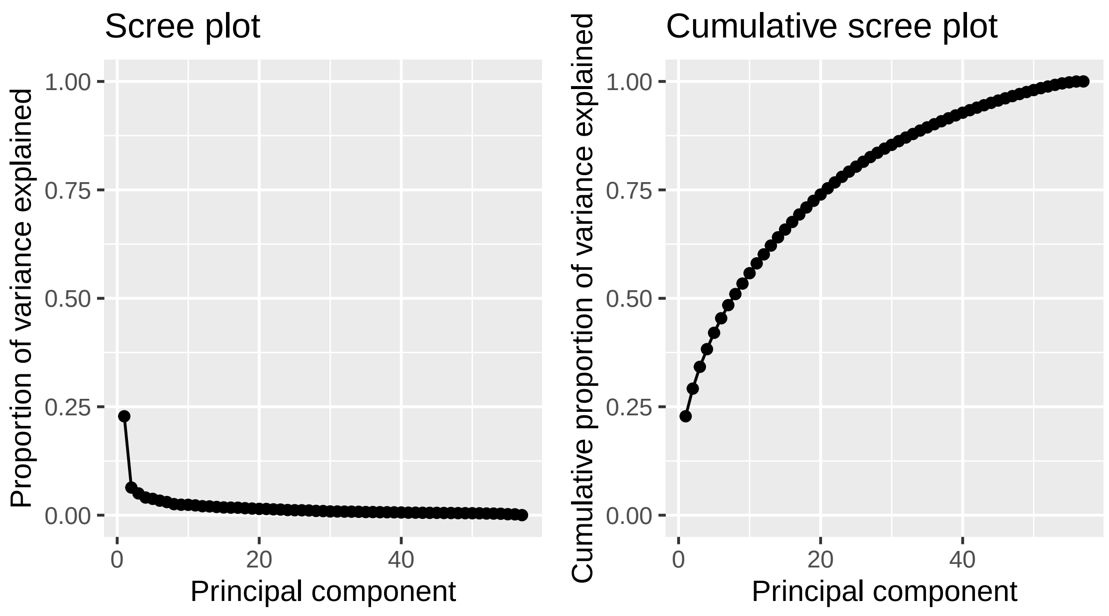
variable	PC1	PC2
9	0.3061400	-1.3587065
39	1.1546198	0.4171544
47	0.4993644	1.1424714
52	-0.4738733	-0.0744358

From the results above, the most correlated variable to PC1 is variable 39 (bi2), and the most correlated variable to PC2 is variable 9 (peu1).

7.

```
# variance explained scree plots
wiki_pve <- tibble(
  var = wiki_eigen$values,
  var_exp = var / sum(var),
  cum_var_exp = cumsum(var_exp)
) %>%
  mutate(pc = row_number())

# PVE plot
```

```r
wiki_pve_p <- ggplot(wiki_pve, aes(pc, var_exp)) +
  geom_line() +
  geom_point() +
  ylim(0, 1) +
  labs(title = "Scree plot",
       x = "Principal component",
       y = "Proportion of variance explained")

# CVE plot
wiki_cve_p <- ggplot(wiki_pve, aes(pc, cum_var_exp)) +
  geom_line() +
  geom_point() +
  ylim(0, 1) +
  labs(title = "Cumulative scree plot",
       x = "Principal component",
       y = "Cumulative proportion of variance explained")

wiki_pve_p + wiki_cve_p
```



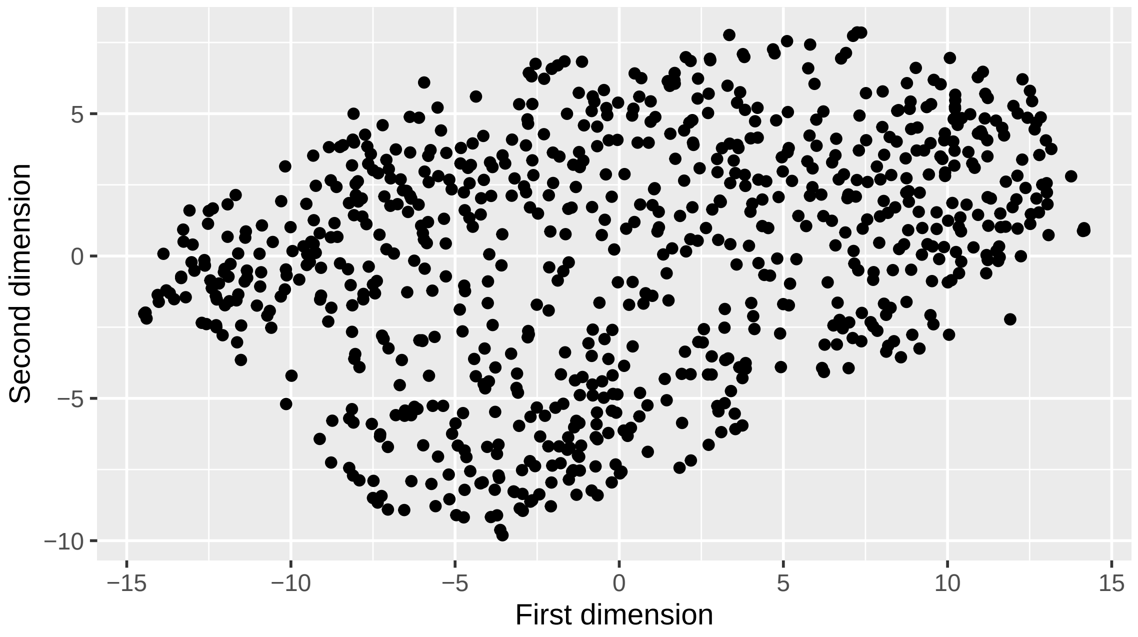
```r
wiki_pve$var_exp[1] + wiki_pve$var_exp[2]
```

```
## [1] 0.291831
```

Approximately, 29% of the variance is explained by the first two principal components.

8.

```r
## t-SNE
wiki_tsne <- Rtsne(as.matrix(df_wiki),
                   perplexity = 65) # can vary; van der Maaten and Hinton suggest this is a good pla

# Plot on the first and second dimensions
wiki_tsne_plot <- df_wiki %>%
  mutate(tsne1 = wiki_tsne$Y[,1],
         tsne2 = wiki_tsne$Y[,2]) %>%
  ggplot(aes(tsne1, tsne2)) +
  geom_point() +
  labs(x = "First dimension",
       y = "Second dimension")
wiki_tsne_plot
```

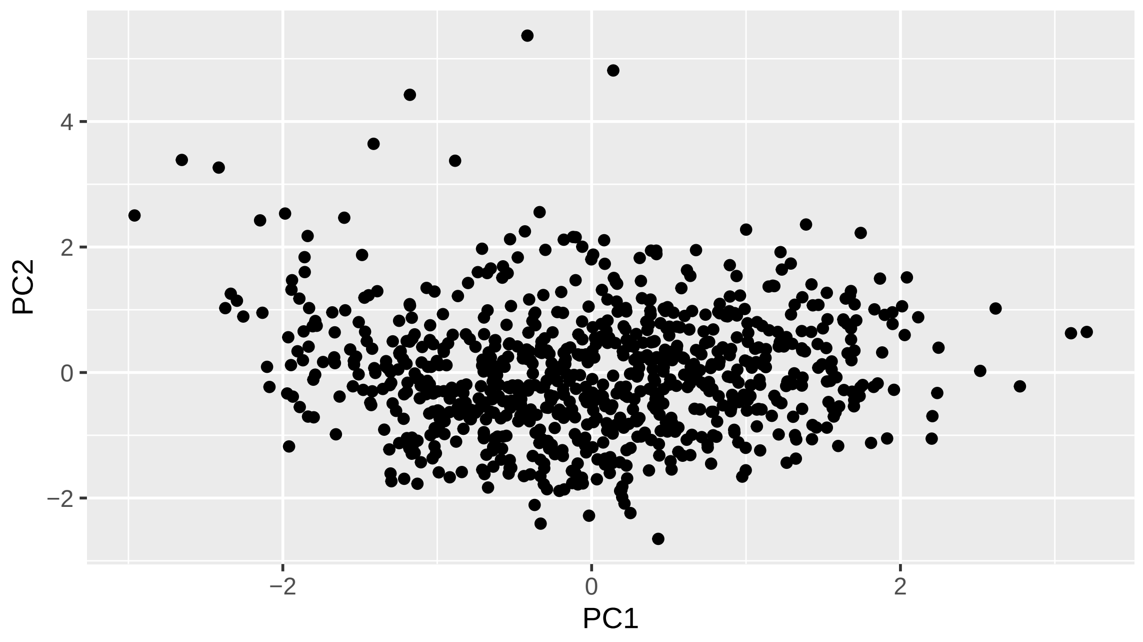


I tried several number of perplexity when generating tSNE. With perplexity 66, it seems the observations scatter into three clusters.

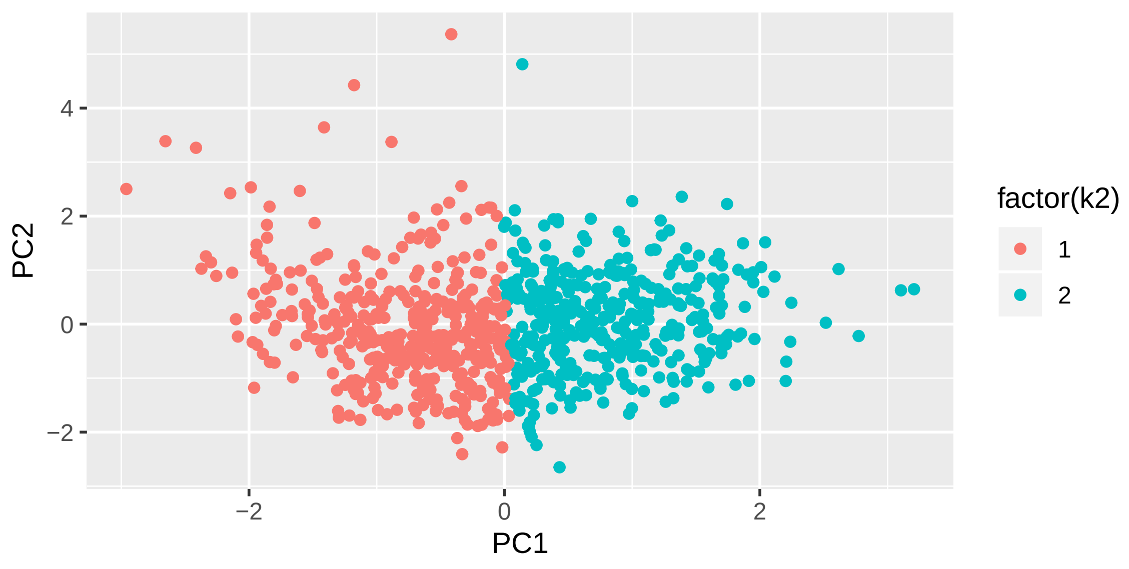
Clustering

9.

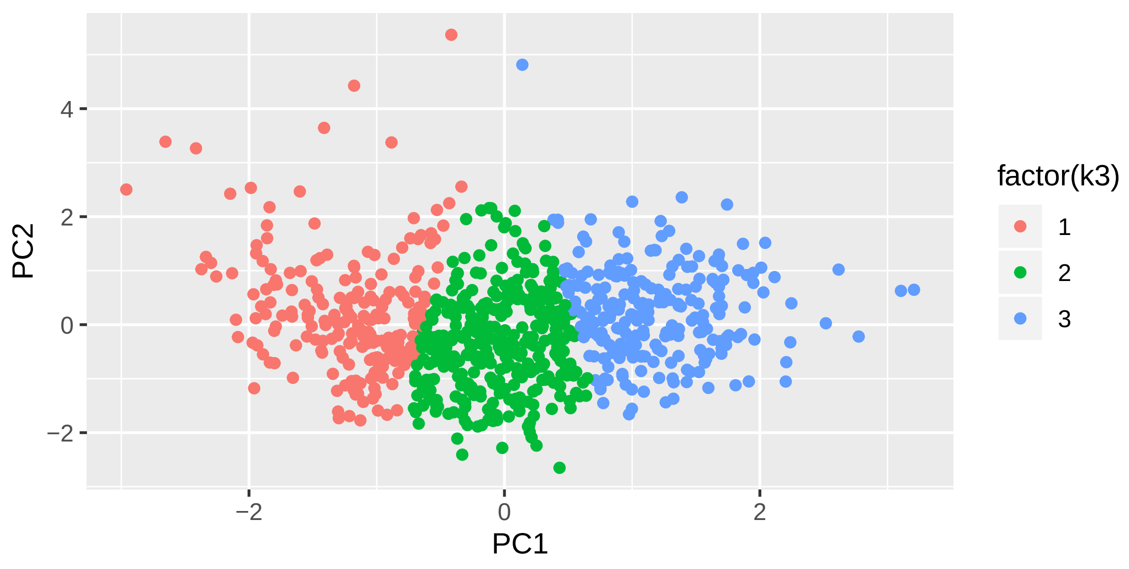
```
# scaled
PC_scale <- PC %>%
  mutate_at(.vars = vars(PC1, PC2), scale)
PC_scale %>%
  ggplot(aes(x=PC1, y=PC2)) +
  geom_point()
```



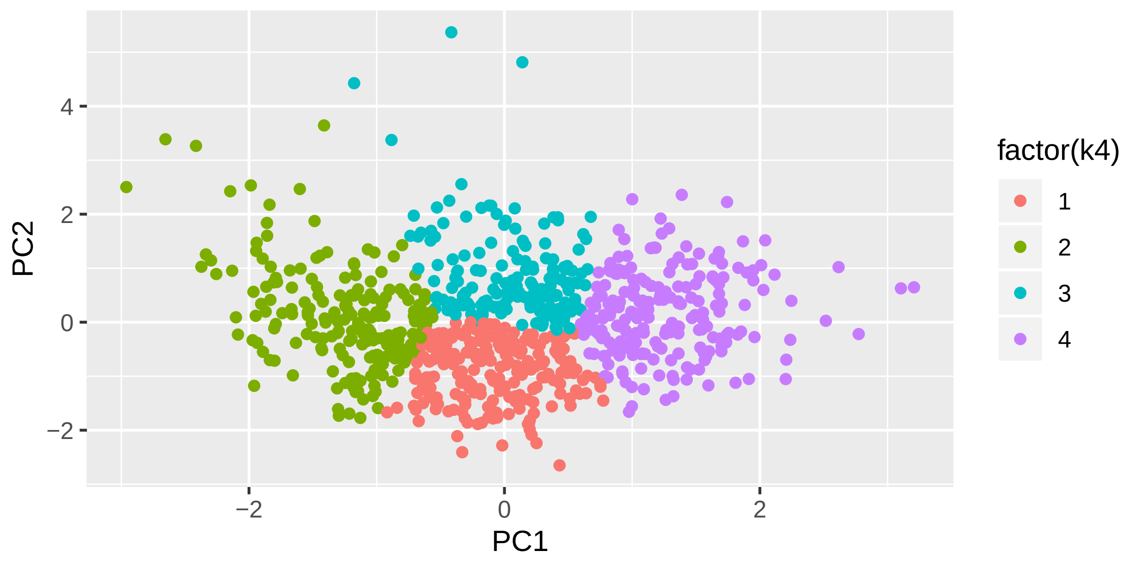
```
PC_km <- PC_scale %>%
  mutate(k2 = kmeans(PC, 2, nstart = 20)$cluster,
         k3 = kmeans(PC, 3, nstart = 20)$cluster,
         k4 = kmeans(PC, 4, nstart = 20)$cluster)
PC_2m <- PC_km %>%
  ggplot(aes(x=PC1, y=PC2, color=factor(k2))) +
  geom_point()
PC_3m <- PC_km %>%
  ggplot(aes(x=PC1, y=PC2, color=factor(k3))) +
  geom_point()
PC_4m <- PC_km %>%
  ggplot(aes(x=PC1, y=PC2, color=factor(k4))) +
  geom_point()

gridExtra::grid.arrange(PC_2m, PC_3m, PC_4m, nrow = 3)
```



PC2
4
2
0
−2
−2
0
2
PC1
factor(k2)
1
2



PC2
4
2
0
−2
−2
0
2
PC1
factor(k3)
1
2
3



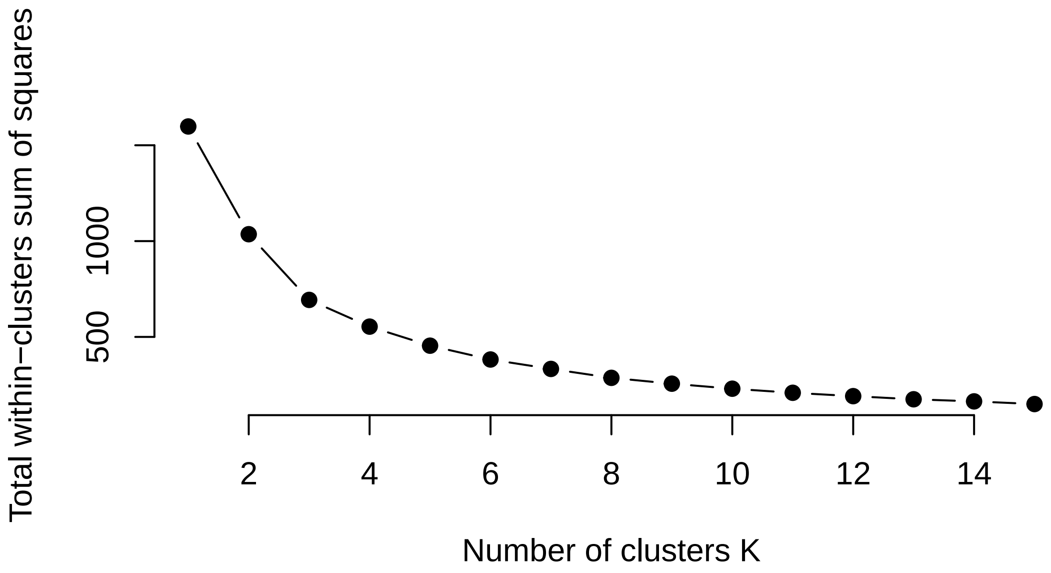
PC2
4
2
0
−2
−2
0
2
PC1
factor(k4)
1
2
3
4

Since the PCA doesn't separate the observations well, there is no clear clusters in the plot.

10.

I use the elbow method, average silhouette, and gap statistic to identify the optimal number of clusters.

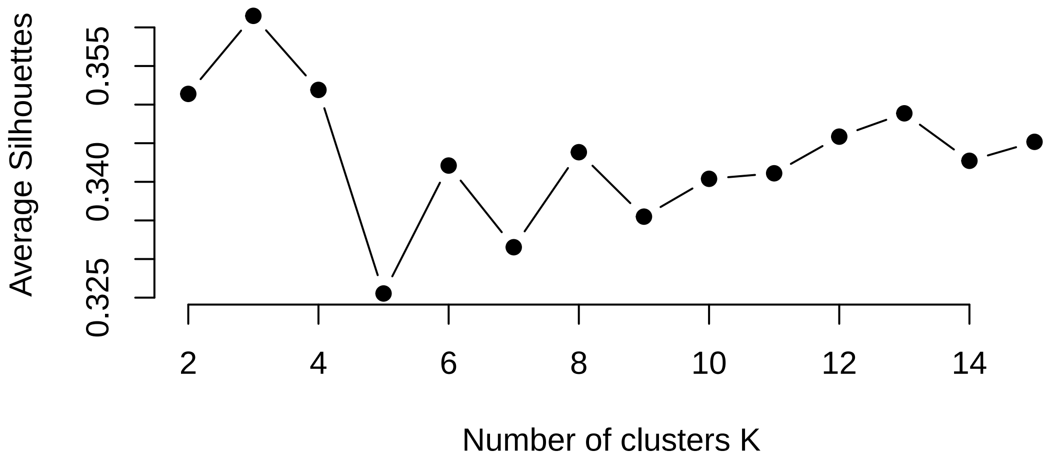
```r
# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(PC_scale, k, nstart = 20)$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)

plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```



From the plot above, the elbow would be number 3.

```r
# function to compute average silhouette for k clusters
avg_sil <- function(k) {
  km.res <- kmeans(PC_scale, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(PC_scale))
  mean(ss[, 3])
}

# Compute and plot wss for k = 2 to k = 15
k.values <- 2:15

# extract avg silhouette for 2-15 clusters
```

```r
avg_sil_values <- map_dbl(k.values, avg_sil)

plot(k.values, avg_sil_values,
       type = "b", pch = 19, frame = FALSE,
       xlab = "Number of clusters K",
       ylab = "Average Silhouettes")
```



From the plot above, the optimal number of clusters is also 3.

```r
gap_stat <- clusGap(PC_scale, FUN = kmeans, nstart = 25,
                    K.max = 10, B = 50)
# Print the result
print(gap_stat, method = "firstmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = PC_scale, FUNcluster = kmeans, K.max = 10, B = 50,     nstart = 25)
## B=50 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'firstmax'): 1
##           logW   E.logW       gap      SE.sim
##  [1,] 5.860785 6.609294 0.7485085 0.010945999
##  [2,] 5.611378 6.283366 0.6719881 0.012774787
##  [3,] 5.419291 6.089869 0.6705781 0.011894862
##  [4,] 5.315933 5.909514 0.5935814 0.011038076
##  [5,] 5.211981 5.794680 0.5826988 0.010459720
##  [6,] 5.125349 5.692407 0.5670578 0.010947878
##  [7,] 5.048559 5.617055 0.5684953 0.010028948
##  [8,] 4.991748 5.546767 0.5550195 0.009568322
##  [9,] 4.925993 5.483477 0.5574843 0.009450485
## [10,] 4.863668 5.425311 0.5616428 0.009729463
```

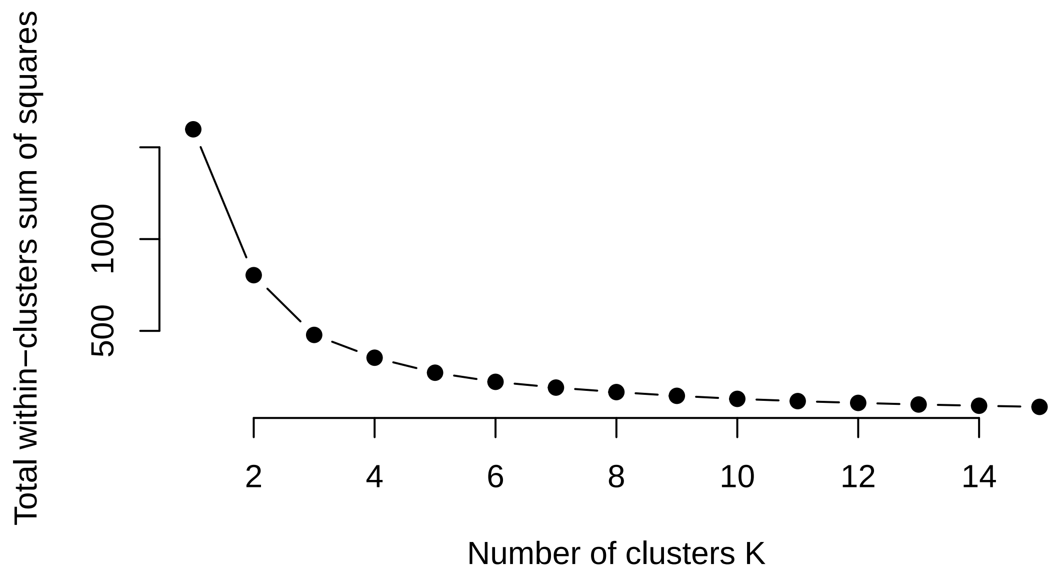
The gap statistic shows that the optimal number of clusters is 1, followed by 2.

From those methods, the optimal number of clusters seems to be 3.

11.

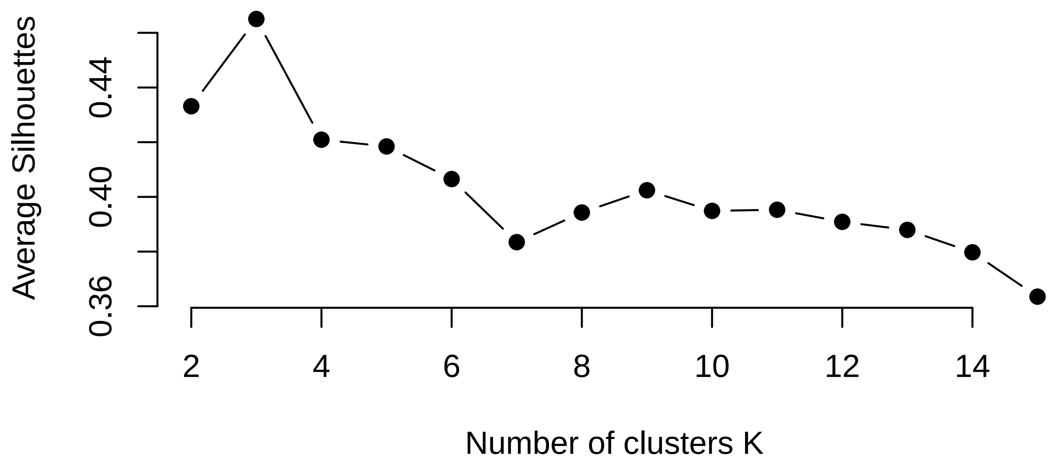
To identify the optimal number of clusters for data from t-SNE, I use three methods again.

```r
df_tsne <- data.frame(tsne1 = wiki_tsne$Y[,1],
                      tsne2 = wiki_tsne$Y[,2])

df_tsne <- df_tsne %>%
  mutate_at(.vars = vars(tsne1, tsne2), scale)

# Elbow method
wss <- function(k) {
  kmeans(df_tsne, k, nstart = 20)$tot.withinss
}

k.values <- 1:15

wss_values <- map_dbl(k.values, wss)

plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```



```r
# Silhouette
avg_sil <- function(k) {
  km.res <- kmeans(df_tsne, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(df_tsne))
  mean(ss[, 3])
}

k.values <- 2:15

avg_sil_values <- map_dbl(k.values, avg_sil)
```

```r
plot(k.values, avg_sil_values,
       type = "b", pch = 19, frame = FALSE,
       xlab = "Number of clusters K",
       ylab = "Average Silhouettes")
```



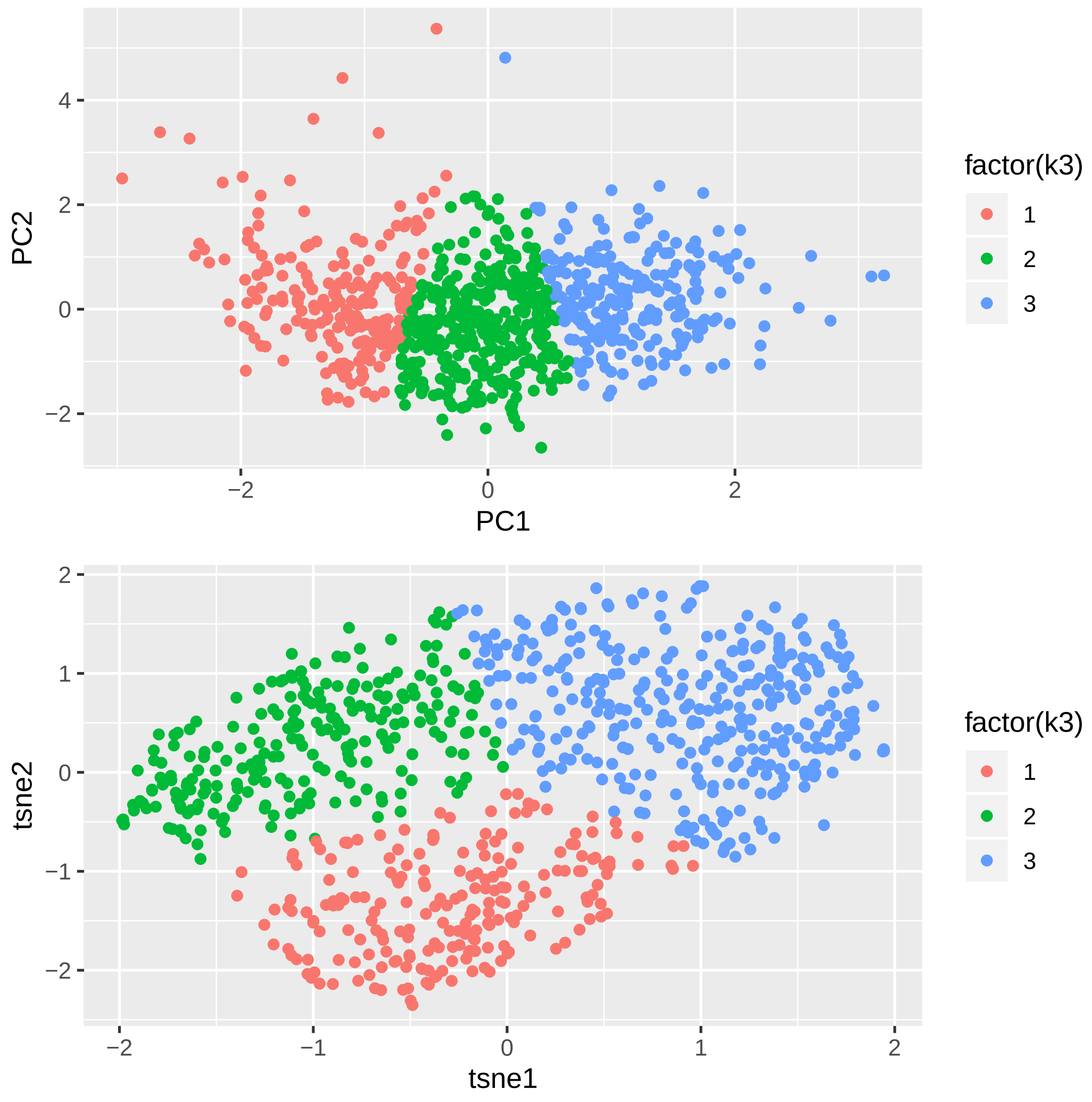
```r
# Gap statistics
gap_stat <- clusGap(df_tsne, FUN = kmeans, nstart = 25,
                    K.max = 10, B = 50)
print(gap_stat, method = "firstmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = df_tsne, FUNcluster = kmeans, K.max = 10, B = 50,     nstart = 25)
## B=50 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'firstmax'): 3
##           logW   E.logW       gap      SE.sim
##  [1,] 5.882118 5.987304 0.1051852 0.011832322
##  [2,] 5.527456 5.641587 0.1141309 0.012601048
##  [3,] 5.268565 5.459331 0.1907670 0.011904051
##  [4,] 5.117370 5.284778 0.1674080 0.010647103
##  [5,] 4.995088 5.164073 0.1689854 0.009099835
##  [6,] 4.890280 5.058096 0.1678160 0.009868100
##  [7,] 4.813536 4.985272 0.1717350 0.008756210
##  [8,] 4.745853 4.915907 0.1700538 0.009637904
##  [9,] 4.678118 4.850955 0.1728368 0.008513087
## [10,] 4.620078 4.790976 0.1708980 0.008435095
```

These results suggest that the optimal number of clusters for tSNE is also 3.

```r
# The optimal number of  cluster for PC is 3.
df_tsne <- df_tsne %>%
  mutate(k3 = kmeans(df_tsne, 3, nstart = 20)$cluster)
tSNE_3m <- df_tsne %>%
  ggplot(aes(x=tsne1, y=tsne2, color=factor(k3))) +
  geom_point()
```

```r
gridExtra::grid.arrange(PC_3m, tSNE_3m, nrow = 2)
```



As I mentioned earlier, PCA does not separate the observations well. On the other hand, when using t-SNE and if we could find certain perplexity, we can find dimensions that separates the observations well for clustering. However, since choosing perplexity is arbitrary and there is no clear way to find optimal perplexity, it makes hard for us to interpret.