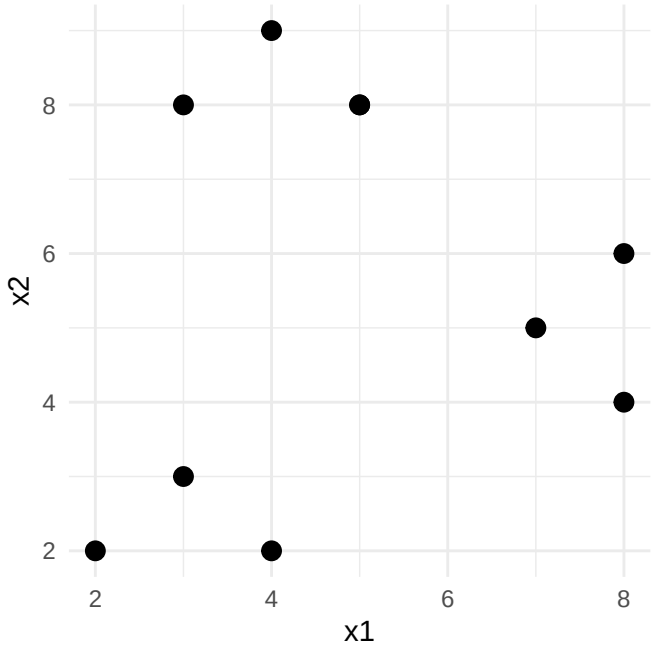
Homework 7: Unsupervised Learning

Wanitchaya Poonpatanapricha

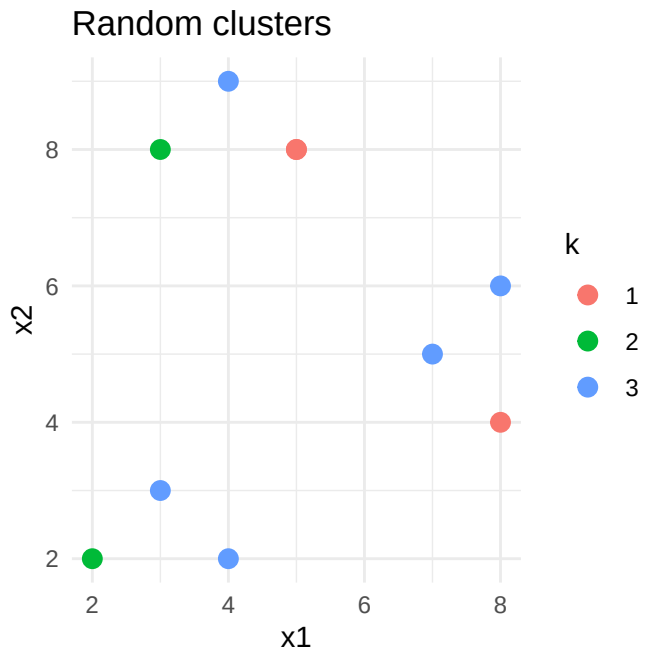
k-Means Clustering "By Hand"

```
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 <- data.frame(x1 = input_1, x2 = input_2)
df %>% ggplot(aes(x1, x2)) + geom_point(size = 3)
```



(1)

```
# randomly assigned k
k_df <- df %>% mutate(k = as.factor(sample.int(3, 10, replace = TRUE)), id = 1:10)

# plot random k
k_df %>% ggplot(aes(x1, x2, color = k)) + geom_point(size = 3) + ggtitle("Random clusters")
```

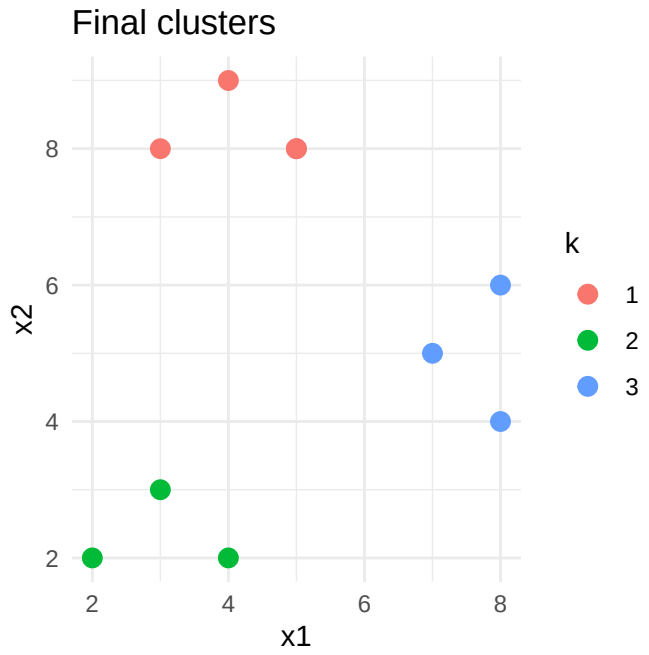


(2)

```
# function for euclidian distance
eu_dist <- function(x1, x2, cen1, cen2){
  return ((x1-cen1)^2 + (x2-cen2)^2)
}

# iteratively update k
new_k_df <- k_df
for (i in 1:25){
  # calculate centroids for each k
  cent <- new_k_df %>%
    mutate(cen_k = k) %>%
    group_by(cen_k) %>%
    dplyr::summarise(x1_c = mean(x1), x2_c = mean(x2))
  # update each observation to be in k with closest centroid
  newer_k_df <- merge(cent, k_df) %>%
    mutate(distance = eu_dist(x1, x2, x1_c, x2_c)) %>%
    group_by(id) %>%
    slice(which.min(distance)) %>% # pick closest centroid
    mutate(k = cen_k) %>% # change to k with closest centroid
    select(x1, x2, k)
  if (nrow(full_join(new_k_df, newer_k_df)) == 10) {
    break() # stop when no change in k assignment
  }
  new_k_df <- newer_k_df
}
```

(3)

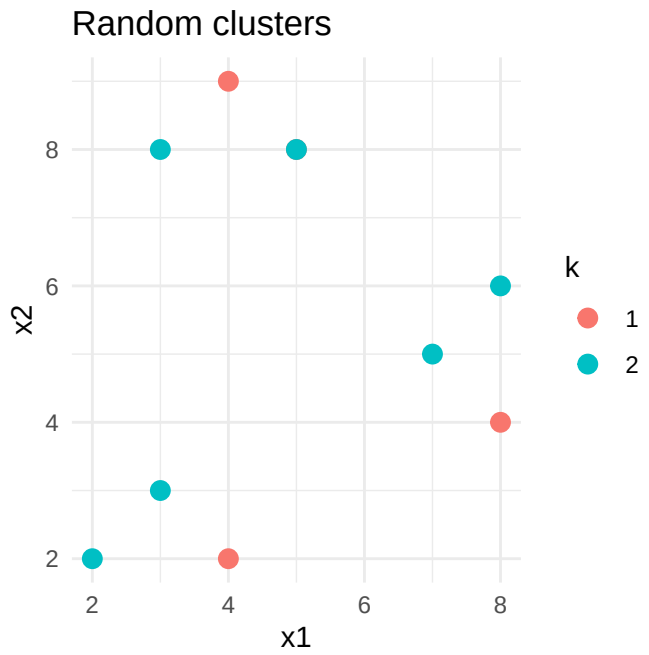
```
# plot the final cluster
p <- new_k_df %>% ggplot(aes(x1, x2, color = k)) + geom_point(size = 3) + ggtitle("Final clusters")
p
```



The 3 clusters look right as we expected them to be. That is observations in the same cluster are closer to each other than to observations outside the cluster.

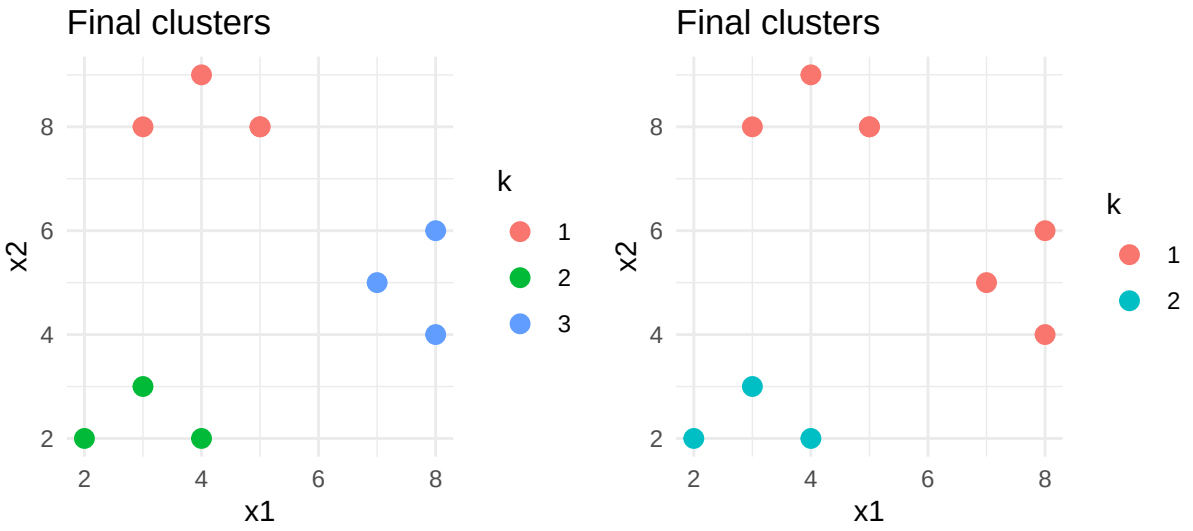
(4)

```
# random k = 2
k_df <- k_df %>% mutate(k = as.factor(sample.int(2, 10, replace = TRUE)), id = 1:10)

# plot random k = 2
k_df %>% ggplot(aes(x1, x2, color = k)) + geom_point(size = 3) + ggtitle("Random clusters")
```




```
# iteratively update k as above
new_k_df <- k_df
for (i in 1:25) {
    cent <- new_k_df %>% mutate(cen_k = k) %>% group_by(cen_k) %>% dplyr::summarise(x1_c = mean(x1),
        x2_c = mean(x2))
    newer_k_df <- merge(cent, k_df) %>% mutate(distance = eu_dist(x1, x2, x1_c, x2_c)) %>%
        group_by(id) %>% slice(which.min(distance)) %>% mutate(k = cen_k) %>% select(x1,
        x2, k)
    if (nrow(full_join(new_k_df, newer_k_df)) == 10) {
        (break)()
    }
    new_k_df <- newer_k_df
}

# plot the final cluster with k = 2
p2 <- new_k_df %>% ggplot(aes(x1, x2, color = k)) + geom_point(size = 3) + ggtitle("Final clusters")
```

```
(p + p2)
```



The clusters seem off because the the three points on the right of the plot are as close to observations from the same cluster (three points on top of the plot) as to observations from different cluster (three points at the bottom of the plot).

(5)

k = 3 fits the data better because when k = 2, there aren't much different between distances within a cluster and between different clusters (as explain in 4). Hence, the clusters don't make sense when k = 2. On the other hand, when k = 3, we can see clearly that distances within a cluster are a lot smaller than distances between clusters, which is the goal of clustering after all.

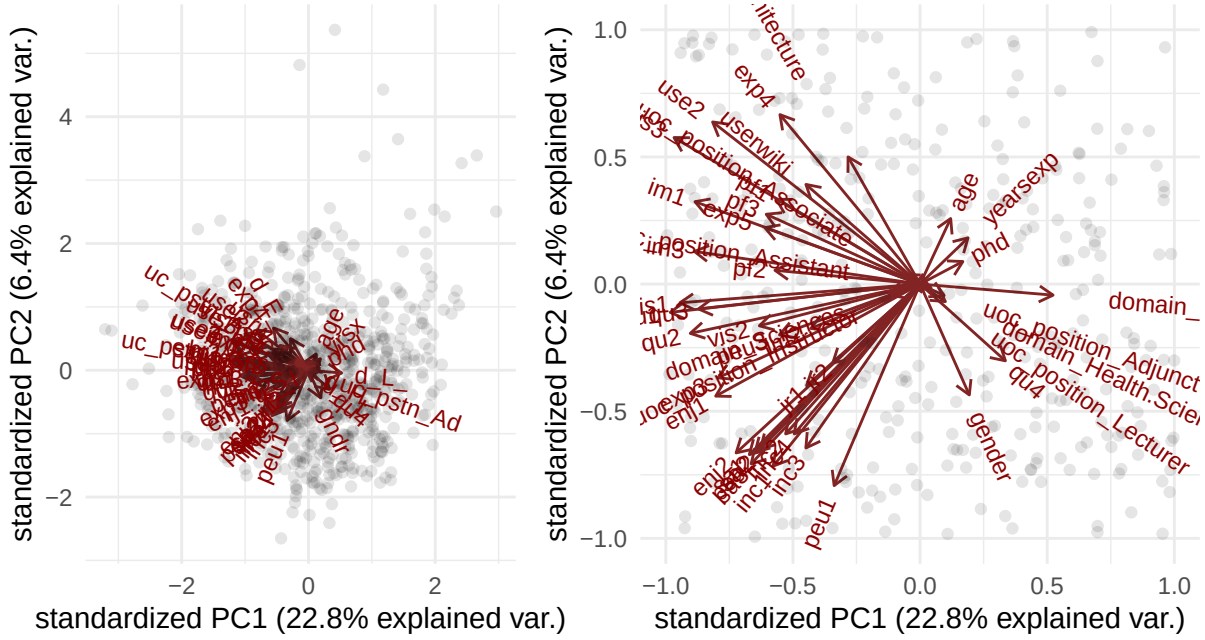
Application

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

# standardize the data
scaled_df <- wiki %>% mutate_if(is.numeric, scale)
```

Dimension reduction

(6)

```
# pca from standardized data
pc <- princomp(scaled_df)

# biplot
p3 <- ggbiplot(pc, varname.size = 3, alpha = 0.1, varname.abbrev = T)
p4 <- ggbiplot(pc, varname.size = 3, alpha = 0.1, varname.abbrev = F) + ylim(-1,
    1) + xlim(-1, 1)
(p3 + p4)
```




```
# pca loadings on PC1 and PC2
load <- as.data.frame(pc$loadings[, 1:2])
load$var <- rownames(pc$loadings)

# highest load on PC1
load %>% select(var, Comp.1, Comp.2) %>% arrange(desc(abs(Comp.1))) %>% head(5) %>%
    kable(caption = "Top 5 variables strongly correlated on PC1")
```

Table 1: Top 5 variables strongly correlated on PC1

var	Comp.1	Comp.2
bi2	-0.2309	0.0834
bi1	-0.2262	0.0564
use3	-0.2188	0.1552
use4	-0.2146	0.1609
pu3	-0.2109	0.0288

```
# highest load on PC2
load %>% select(var, Comp.1, Comp.2) %>% arrange(desc(abs(Comp.2))) %>% head(5) %>%
    kable(caption = "Top 5 variables strongly correlated on PC2")
```

Table 2: Top 5 variables strongly correlated on PC2

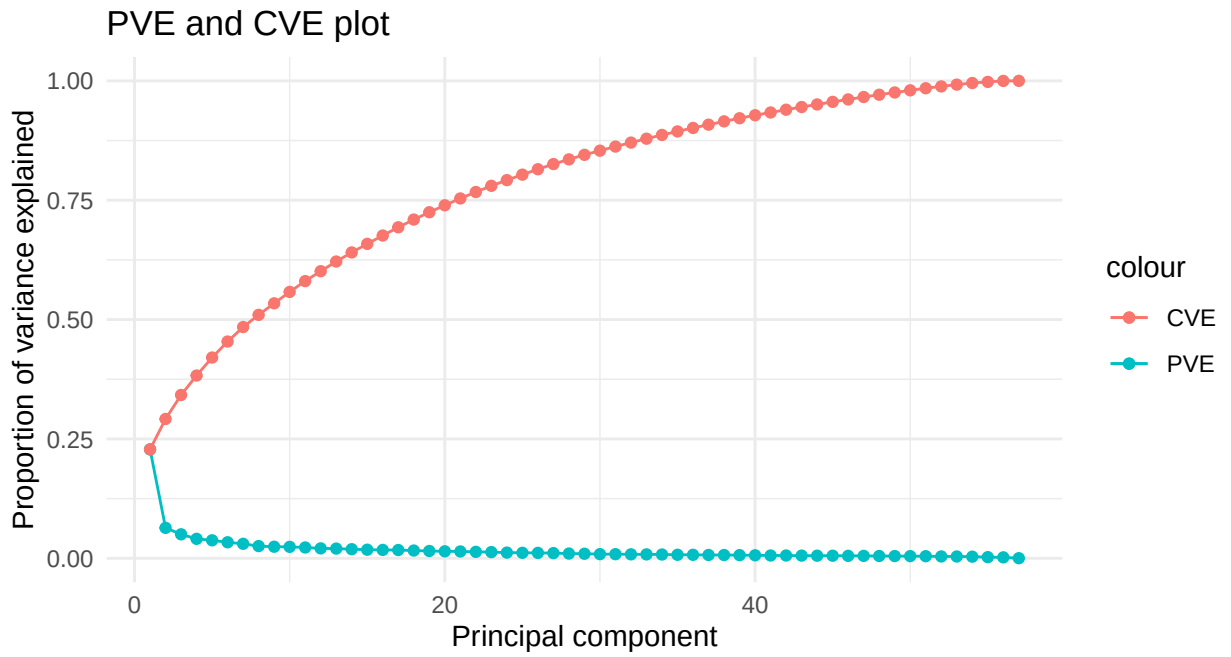
var	Comp.1	Comp.2
peu1	-0.0612	-0.2717
inc1	-0.1047	-0.2454
sa3	-0.1204	-0.2423
sa1	-0.1217	-0.2299
exp4	-0.0999	0.2285

From the loadings, `bi2` is most correlated with PC1. The direction of correlation between `bi2` and PC1 is

negative. `peu1` is most correlated with PC2. The direction of correlation between `peu1` and PC2 is negative. From the 2 tables, the top variables for each PC suggest that PC1 and PC2 are orthogonal; top variables for PC1 have low correlations to PC2 and vice versa.

(7)

```
# calculate PVE and CVE
wiki_pve <- tibble(var = pc$sdev^2, var_exp = var/sum(var), cum_var_exp = cumsum(var_exp),
    pc = c(1:length(var)))

# PVE and CVE plot
ggplot(wiki_pve) + geom_line(aes(pc, var_exp, color = "PVE")) + geom_point(aes(pc,
    var_exp, color = "PVE")) + geom_line(aes(pc, cum_var_exp, color = "CVE")) + geom_point(aes(pc,
    cum_var_exp, color = "CVE")) + ylim(0, 1) + labs(title = "PVE and CVE plot",
    x = "Principal component", y = "Proportion of variance explained")
```



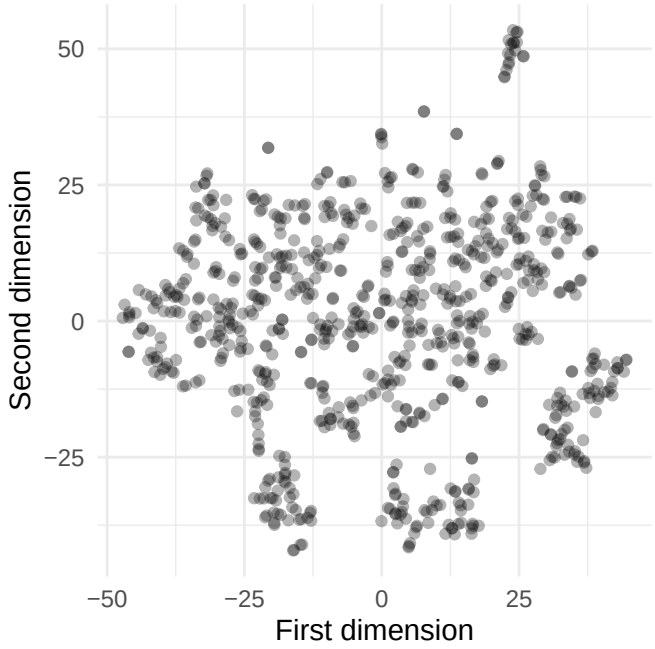

```
# exact values
wiki_pve %>% filter(pc <= 2) %>% select(pc, var_exp, cum_var_exp) %>% kable()
```

pc	var_exp	cum_var_exp
1	0.2281	0.2281
2	0.0637	0.2918

PC1 explains 22.81% of the variance. PC2 explains 6.37% of the variance. Combining both PC1 and PC2, they explain 29.18% of the variance. This suggests that some other PCs beyond the 1st and 2nd are quite important too. From the graph, if we want at least 3 quarters of the variance explained, we need to include until around the 20th PC.

(8)

```
wiki_tsne <- Rtsne(as.matrix(scaled_df), perplexity = 5)

wiki_tsne_plot <- scaled_df %>% mutate(tsne1 = wiki_tsne$Y[, 1], tsne2 = wiki_tsne$Y[,
    2]) %>% ggplot(aes(tsne1, tsne2)) + geom_point(alpha = 0.3) + labs(x = "First dimension",
    y = "Second dimension")
wiki_tsne_plot
```

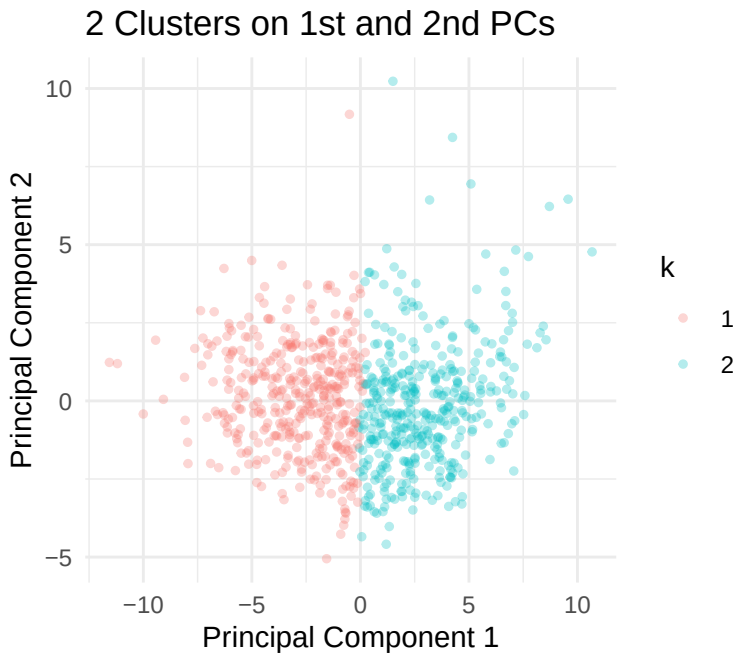


There doesn't seem to be any obvious cluster in the t-SNE plot. However, there are 4 tiny, outlier-ish clusters. It is, however, unclear what they are or why they cluster like that.

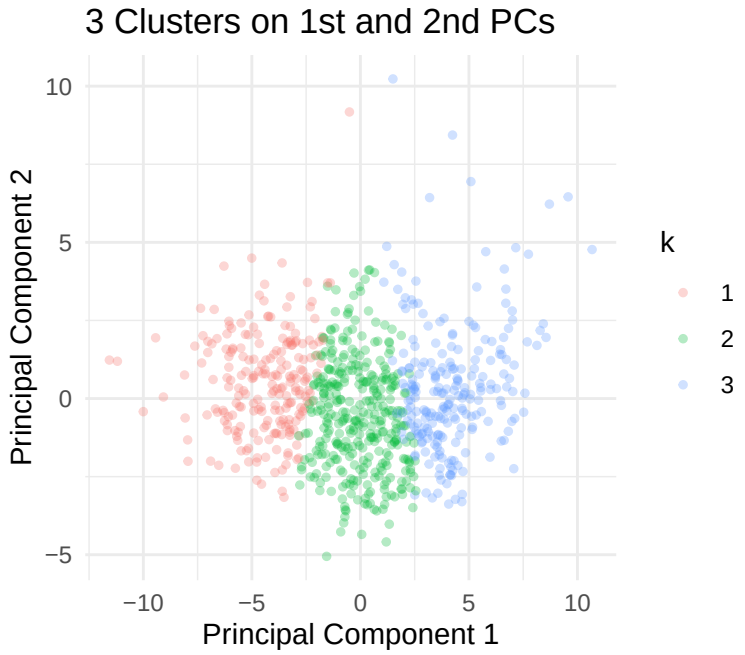
Clustering

(9)

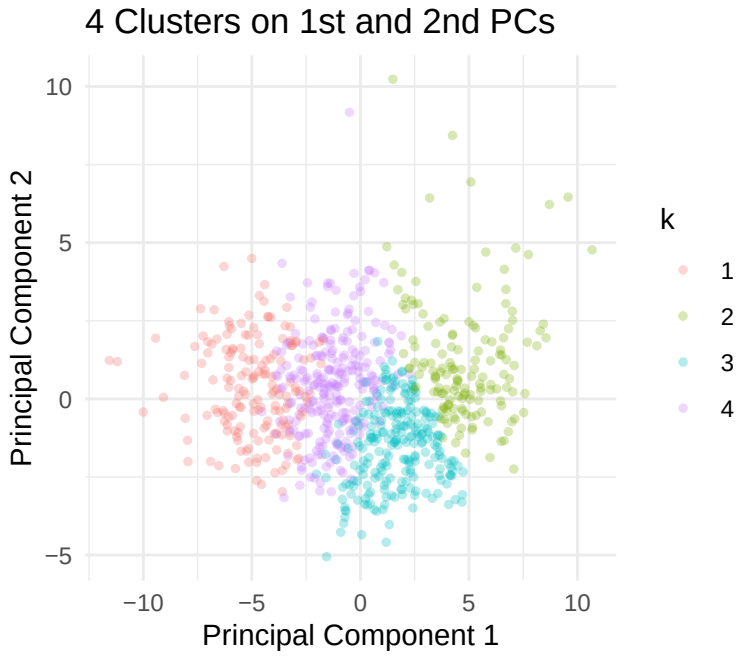
```
# function to plot kmeans clustering using PC1 and PC2
k_plot <- function(i) {
    pck <- princomp(scaled_df)
    cl <- kmeans(scaled_df, i, iter.max = 50)
    stdData1 <- cbind(scaled_df, as.factor(cl$cluster))
    p <- ggplot(as.data.frame(pck$scores), aes(x = Comp.1, y = Comp.2, color = as.factor(stdData1[,
        length(stdData1)]))) + geom_point(size = 1, alpha = 0.3) + labs(title = paste0(as.character(i),
        " Clusters on 1st and 2nd PCs"), x = "Principal Component 1", y = "Principal Component 2",
        color = "k")
    return(p)
}

k_plot(2)
```




```
k_plot(3)
```



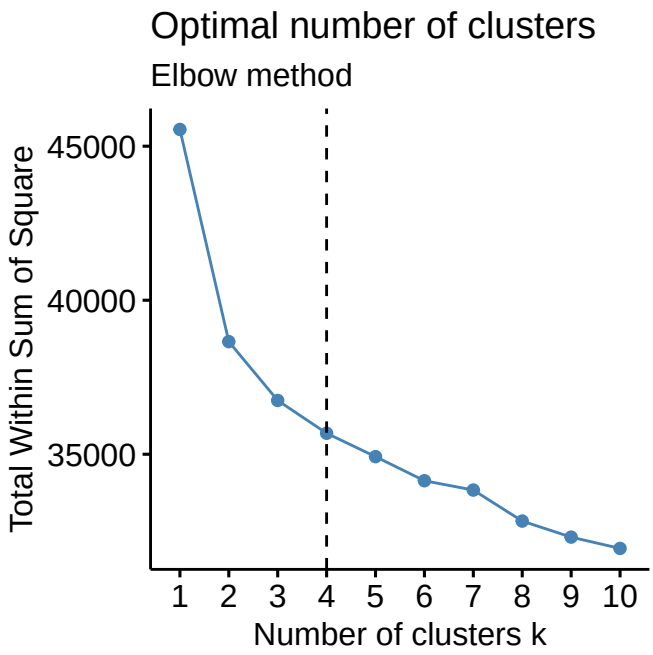

```
k_plot(4)
```



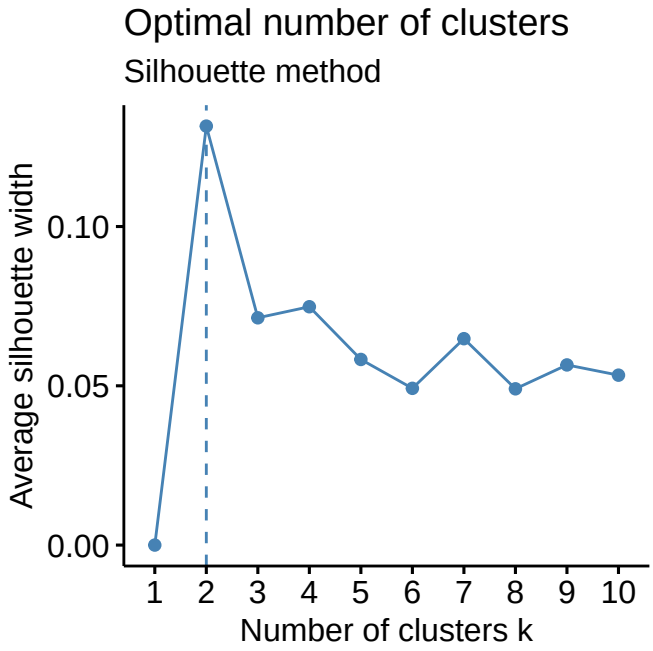
All clusters with k from 2 to 4 seem to be primarily divided by the 1st PC. When k = 2, the clusters are almost literally defined by whether a observation has PC1 >= 0 or < 0. When k = 3, the boundaries are a bit deviated from a vertical line, hence PC2 has a bit of influence for clustering here. It is clearer when k = 4 to see that PC2 has some influences on the clustering. In addition, in both k = 3 and k = 4, we can see that other PCs beyond 1 and 2 have some influences since the boundaries are not clean-cut given only PC1 and PC2.

(10)

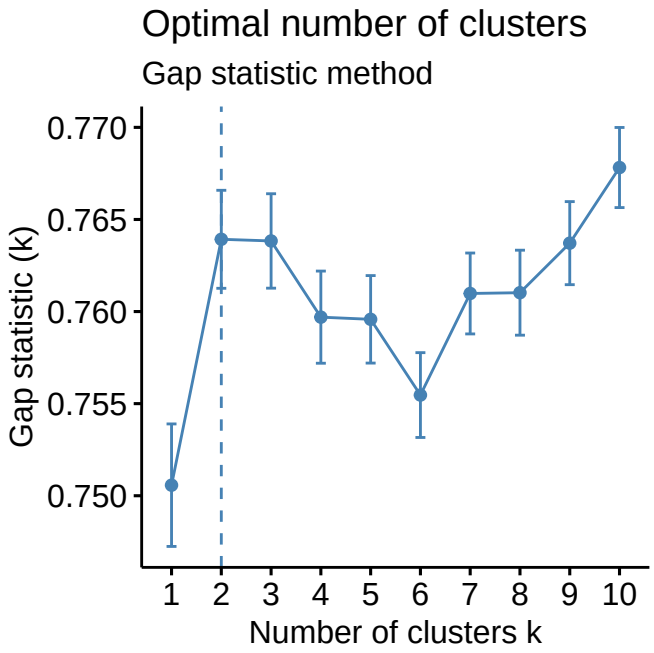
```
# Elbow method
fviz_nbclust(scaled_df, kmeans, method = "wss") + geom_vline(xintercept = 4, linetype = 2) +
    labs(subtitle = "Elbow method")
```




```r
# Silhouette method
fviz_nbclust(scaled_df, kmeans, method = "silhouette") + labs(subtitle = "Silhouette method")
```



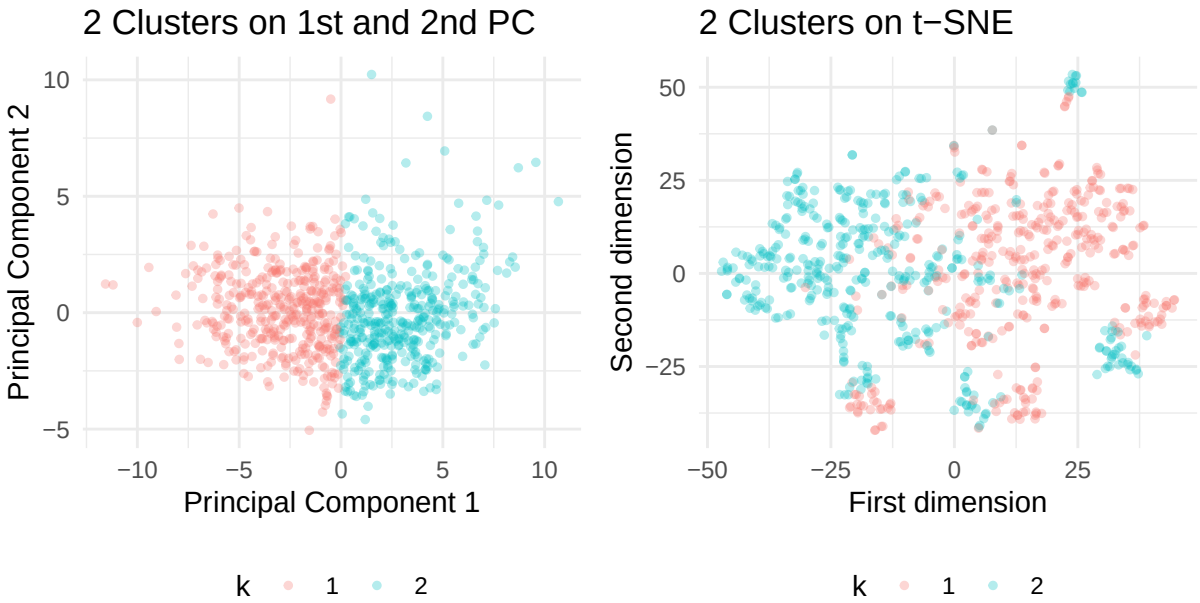

```r
# Gap statistic
fviz_nbclust(scaled_df, kmeans, nstart = 25, method = "gap_stat", nboot = 50) + labs(subtitle = "Gap sta
```



Since 2 out of the 3 methods suggest k = 2, we will say that the optimal number of k is 2.

(11)

```
pck <- princomp(scaled_df)
cl <- kmeans(scaled_df, 2, iter.max = 50)
stdData1 <- cbind(scaled_df, as.factor(cl$cluster))
p5 <- ggplot(as.data.frame(pck$scores), aes(x = Comp.1, y = Comp.2, color = as.factor(stdData1[,
    length(stdData1)]))) + geom_point(size = 1, alpha = 0.3) + labs(title = "2 Clusters on 1st and 2nd
    x = "Principal Component 1", y = "Principal Component 2", color = "k") + theme(legend.position = "b

p6 <- ggplot(scaled_df %>% mutate(tsne1 = wiki_tsne$Y[, 1], tsne2 = wiki_tsne$Y[,
    2]), aes(x = tsne1, y = tsne2, color = as.factor(stdData1[, length(stdData1)]))) +
    geom_point(size = 1, alpha = 0.3) + labs(title = "2 Clusters on t-SNE", x = "First dimension",
    y = "Second dimension", color = "k") + theme(legend.position = "bottom")

(p5 + p6)
```



From the plots, we can see that PCA has a much cleaner separation between clusters than t-SNE has. This is likely because t-SNE is a probabilistic technique that its goal is to place neighbors close to each other. Hence, t-SNE focuses more on local structure and less on global structure than PCA. In terms of clustering, the global structure is very important. Hence, PCA is better at capturing the clusters than t-SNE. From the t-SNE plot, we can clearly see that there are 1 main t-SNE cluster and 4 tiny ones. Each of the 5 has both clusters by kmeans. This confirms the fact that t-SNE prioritizes locality over globality.