hw7

Hengle Li

3/13/2020

```
library(tidyverse)
library(tidymodels)
library(patchwork)
library(cluster)
library(e1071)
library(Rtsne)

set.seed(6758)
```

k-Means Clustering "By Hand"

```
#set up data
df <- tibble(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))
```

```
#randomly assign clusters
ran_clus <- sample(1:3, 10, replace = TRUE)
df1 <- cbind(df, ran_clus)
colnames(df1) <- c("X", "Y", "cluster")
```

Trial run

```
#centroid 1 based on cluster 1
cen1 <- df1 %>%
  filter(cluster == 1)

df1$centroid1_X <- mean(cen1$X)
df1$centroid1_Y <- mean(cen1$Y)

#centroid 2 based on cluster 2
cen2 <- df1 %>%
  filter(cluster == 2)

df1$centroid2_X <- mean(cen2$X)
df1$centroid2_Y <- mean(cen2$Y)
```

```r
#centroid 3 based on cluster 3
cen3 <- df1 %>%
  filter(cluster == 3)

df1$centroid3_X <- mean(cen3$X)
df1$centroid3_Y <- mean(cen3$Y)
```

```r
df1 <- df1 %>%
  mutate(dist1 = sqrt((X - centroid1_X)^2 + (Y - centroid1_Y)^2),
         dist2 = sqrt((X - centroid2_X)^2 + (Y - centroid2_Y)^2),
         dist3 = sqrt((X - centroid3_X)^2 + (Y - centroid3_Y)^2)) %>%
  #find the min distance for each observation
  transform(min = pmin(dist1, dist2, dist3)) %>%
  #reassign clusters
  mutate(cluster = ifelse(min == dist1, 1,
                          ifelse(min == dist2, 2, 3)))

df1
```

```
##    X Y cluster centroid1_X centroid1_Y centroid2_X centroid2_Y centroid3_X
## 1  5 8       2           5           4         4.8         6.2           5
## 2  8 6       3           5           4         4.8         6.2           5
## 3  7 5       3           5           4         4.8         6.2           5
## 4  8 4       1           5           4         4.8         6.2           5
## 5  3 3       1           5           4         4.8         6.2           5
## 6  4 2       1           5           4         4.8         6.2           5
## 7  2 2       1           5           4         4.8         6.2           5
## 8  3 8       2           5           4         4.8         6.2           5
## 9  4 9       2           5           4         4.8         6.2           5
## 10 5 8       2           5           4         4.8         6.2           5
##    centroid3_Y    dist1    dist2    dist3      min
## 1     5.333333 4.000000 1.811077 2.666667 1.811077
## 2     5.333333 3.605551 3.206244 3.073181 3.073181
## 3     5.333333 2.236068 2.505993 2.027588 2.027588
## 4     5.333333 3.000000 3.883298 3.282953 3.000000
## 5     5.333333 2.236068 3.671512 3.073181 2.236068
## 6     5.333333 2.236068 4.275512 3.480102 2.236068
## 7     5.333333 3.605551 5.047772 4.484541 3.605551
## 8     5.333333 4.472136 2.545584 3.333333 2.545584
## 9     5.333333 5.099020 2.912044 3.800585 2.912044
## 10    5.333333 4.000000 1.811077 2.666667 1.811077
```

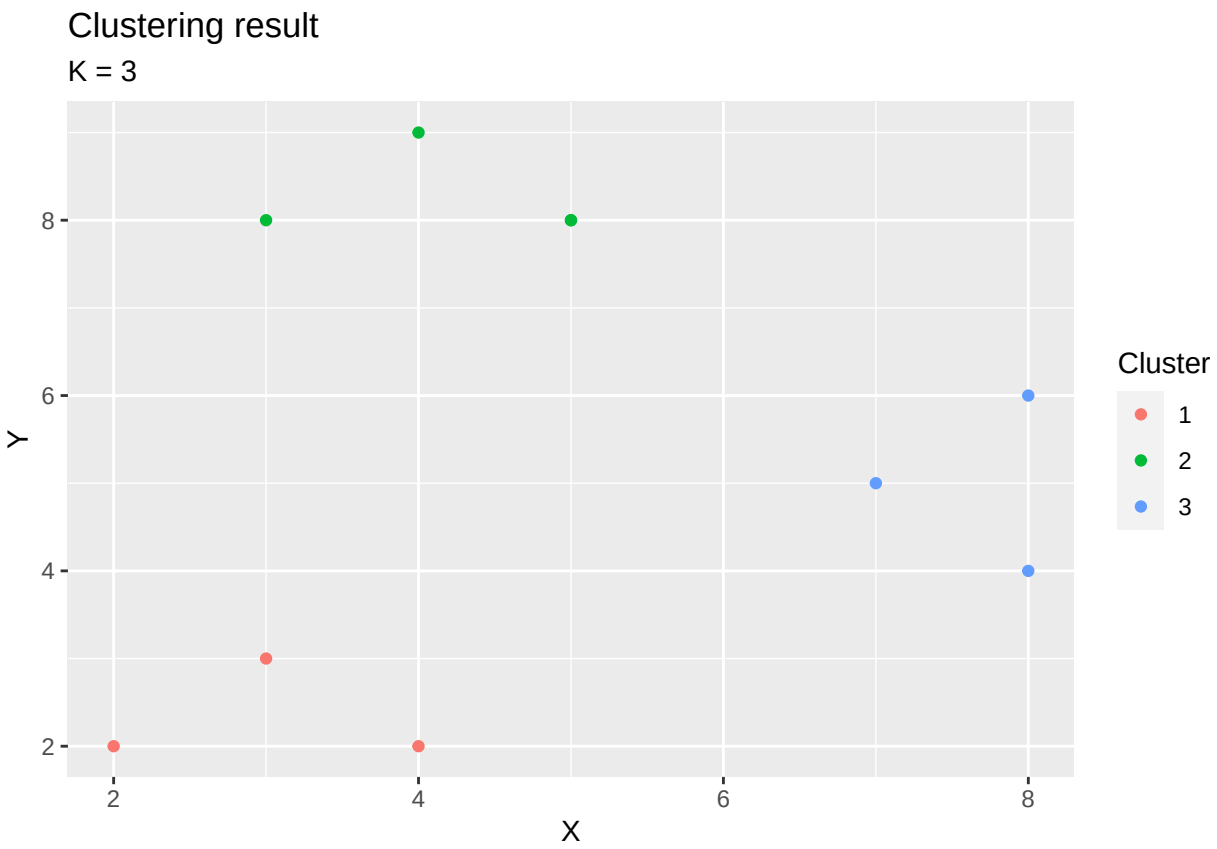
- Repeated run (k = 3)

```r
repeat{
  #centroid 1 based on cluster 1
  cen1 <- df1 %>%
    filter(cluster == 1)

  df1$centroid1_X <- mean(cen1$X)
  df1$centroid1_Y <- mean(cen1$Y)
```

```
  #centroid 2 based on cluster 2
  cen2 <- df1 %>%
    filter(cluster == 2)

  df1$centroid2_X <- mean(cen2$X)
  df1$centroid2_Y <- mean(cen2$Y)

  #centroid 3 based on cluster 3
  cen3 <- df1 %>%
    filter(cluster == 3)

  df1$centroid3_X <- mean(cen3$X)
  df1$centroid3_Y <- mean(cen3$Y)

  df1 <- df1 %>%
    mutate(cluster_old = cluster) %>%
    mutate(dist1 = sqrt((X - centroid1_X)^2 + (Y - centroid1_Y)^2),
           dist2 = sqrt((X - centroid2_X)^2 + (Y - centroid2_Y)^2),
           dist3 = sqrt((X - centroid3_X)^2 + (Y - centroid3_Y)^2)) %>%
    #find the min distance for each observation
    transform(min = pmin(dist1, dist2, dist3)) %>%
    #reassign clusters
    mutate(cluster = ifelse(min == dist1, 1,
                            ifelse(min == dist2, 2, 3)))
  #if only takes length 1 conditions
  #use a number to signify whether the two lists are equal
  gogo <- ifelse(df1$cluster == df1$cluster_old, 1, 0) %>%
    sum()
  #when gogo is 10, all 10 observations converge
  if(gogo == 10){
    break
  }
}
```

Result

```
#plot the clustered result
k3 <- df1 %>%
  ggplot(aes(X, Y, color = as.factor(cluster))) +
  geom_point() +
  scale_color_discrete(name = "Cluster") +
  labs(title = "Clustering result",
       subtitle = "K = 3")

k3
```



- When k = 2

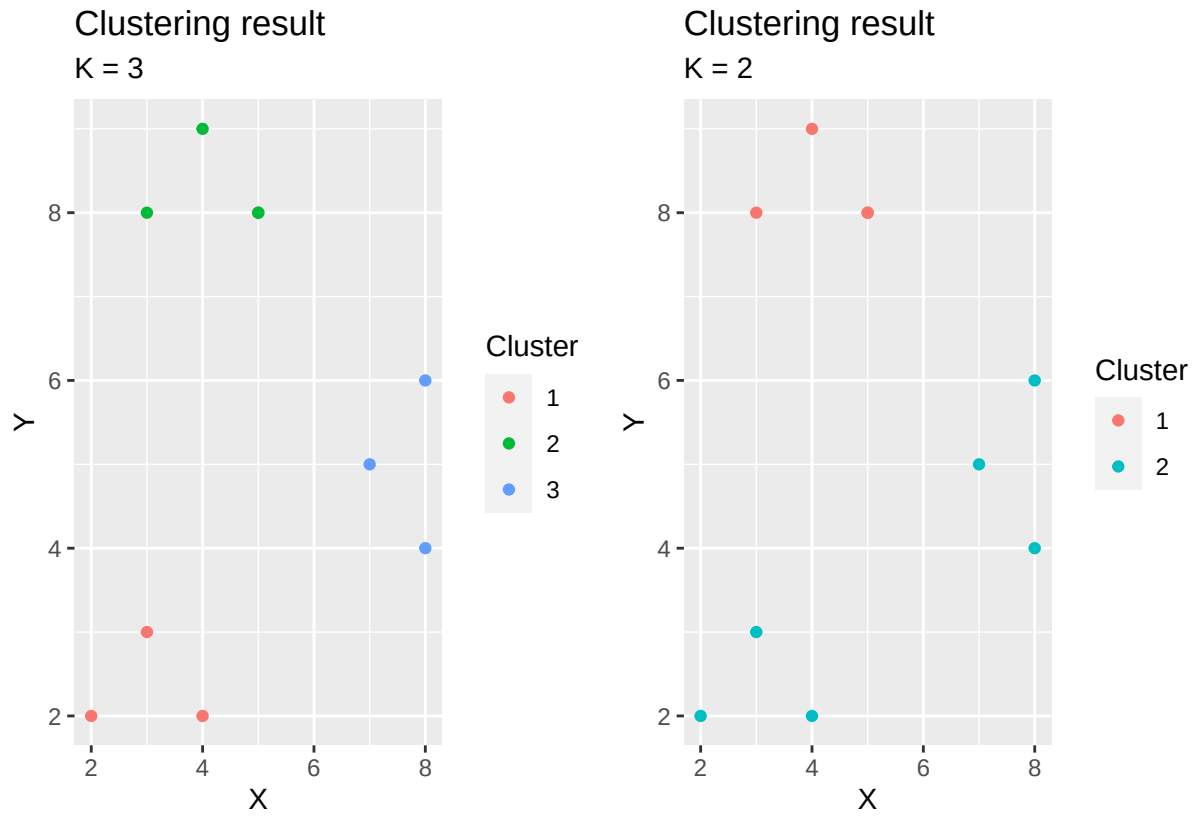
```
#set up before repetition
ran_clus <- sample(1:2, 10, replace = TRUE)
df2 <- cbind(df, ran_clus)
colnames(df2) <- c("X", "Y", "cluster")
```

```
repeat{
  #centroid 1 based on cluster 1
  cen1 <- df2 %>%
    filter(cluster == 1)

  df2$centroid1_X <- mean(cen1$X)
  df2$centroid1_Y <- mean(cen1$Y)

  #centroid 2 based on cluster 2
  cen2 <- df2 %>%
    filter(cluster == 2)

  df2$centroid2_X <- mean(cen2$X)
  df2$centroid2_Y <- mean(cen2$Y)

  df2 <- df2 %>%
    mutate(cluster_old = cluster) %>%
    mutate(dist1 = sqrt((X - centroid1_X)^2 + (Y - centroid1_Y)^2),
```

```
            dist2 = sqrt((X - centroid2_X)^2 + (Y - centroid2_Y)^2)) %>%
    #find the min distance for each observation
    transform(min = pmin(dist1, dist2)) %>%
    #reassign clusters
    mutate(cluster = ifelse(min == dist1, 1, 2))
  #if only takes length 1 conditions
  #use a number to signify whether the two lists are equal
  gogo <- ifelse(df2$cluster == df2$cluster_old, 1, 0) %>%
    sum()
  #when gogo is 10, all 10 observations converge
  if(gogo == 10){
    break
  }
}
```

```
#plot the results
k2 <- df2 %>%
  ggplot(aes(X, Y, color = as.factor(cluster))) +
  geom_point() +
  scale_color_discrete(name = "Cluster") +
  labs(title = "Clustering result",
       subtitle = "K = 2")
```

```
#combine both plots
k3 + k2
```



Comparing the outcomes for k = 3 and for k = 2, we can tell that k = 3 is a better fit for this dataset. As the graph shows, there are in fact three groups of points. The three dots on the bottom left corner clearly form a cluster separate from those in the middle on the right. When k = 2, due to the limited number of clusters, the two clusters have to be combined. When k = 3, the number of clusters is just enough to capture the grouping of the data.

Application

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

Dimension reduction

PCA

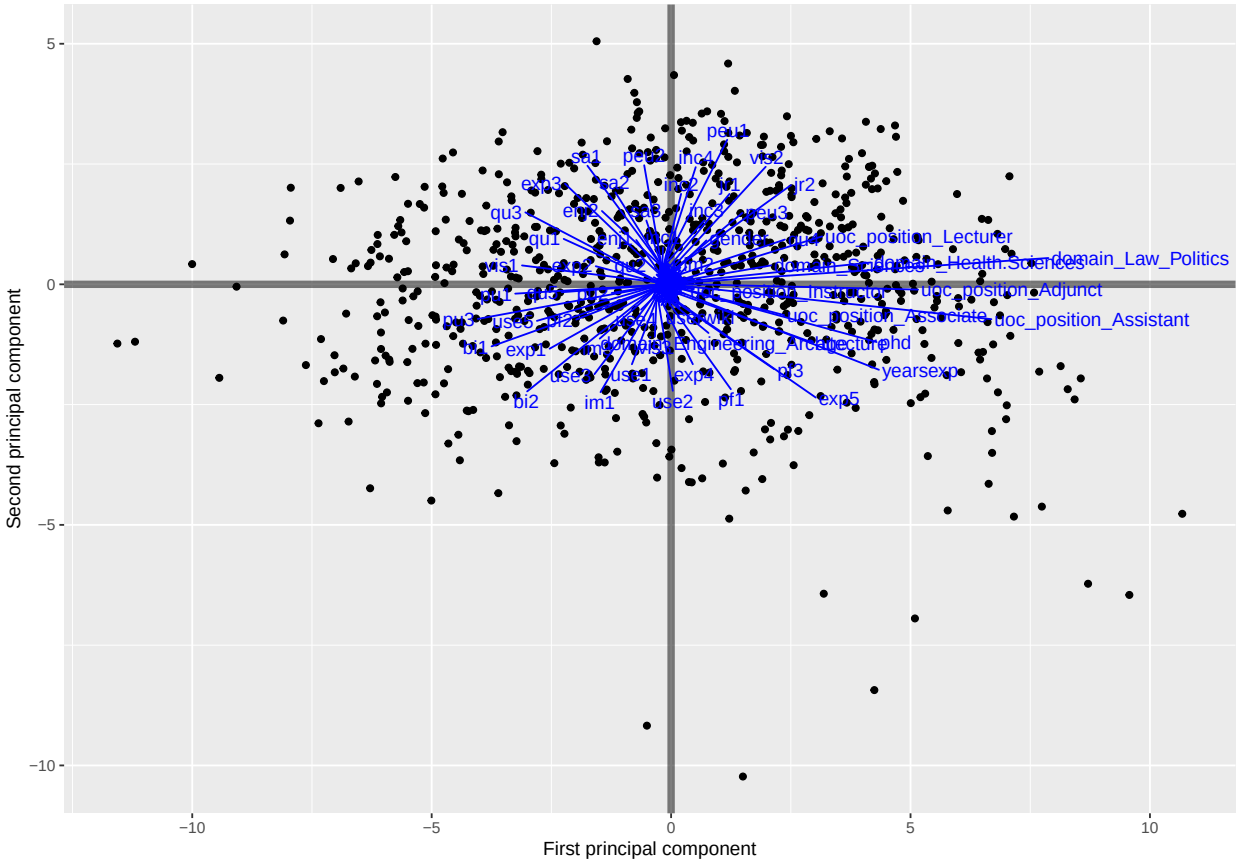
```
#scale data for PCA and t-SNE
wiki_scaled <- wiki %>%
  mutate_all(scale)

#calculate covariance among the features
wiki_cov <- wiki_scaled %>% cov()

#calculate eigen values for the covariance matrix
wiki_eigen <- eigen(wiki_cov)
```

```
#extract values from eigen vectors
#by definition, they are the first and second PC
phi <- wiki_eigen$vectors[, 1:2]

#join PC1 and PC2 with corresponding features
phi <- -phi
row.names(phi) <- names(wiki_scaled)
colnames(phi) <- c("PC1", "PC2")
```

```
#calcualtion of PC for observations by hand
PC1 <- as.matrix(select_if(wiki_scaled, is.numeric)) %*% phi[,1]
PC2 <- as.matrix(select_if(wiki_scaled, is.numeric)) %*% phi[,2]

#combine the calculation outcomes for plotting
PC <- tibble(
  PC1 = PC1[,1],
  PC2 = PC2[,1]
)

#a simpler way to calculate PC for observations
#wiki_prc <- prcomp(wiki, scale. = TRUE)
#PC <- as.data.frame(wiki_prc$x)

#prepare the features for plotting
```

```
phi_df <- (phi * 1.75) %>%
  as.data.frame %>%
  rownames_to_column(var = "variable")
```

```
#plot observations and features on first and second PC
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 = "blue") +
  ggrepel::geom_text_repel(data = phi_df,
                           aes(x = PC1, y = PC2, label = variable),
                           color = "blue") +
  labs(x = "First principal component",
       y = "Second principal component")
```



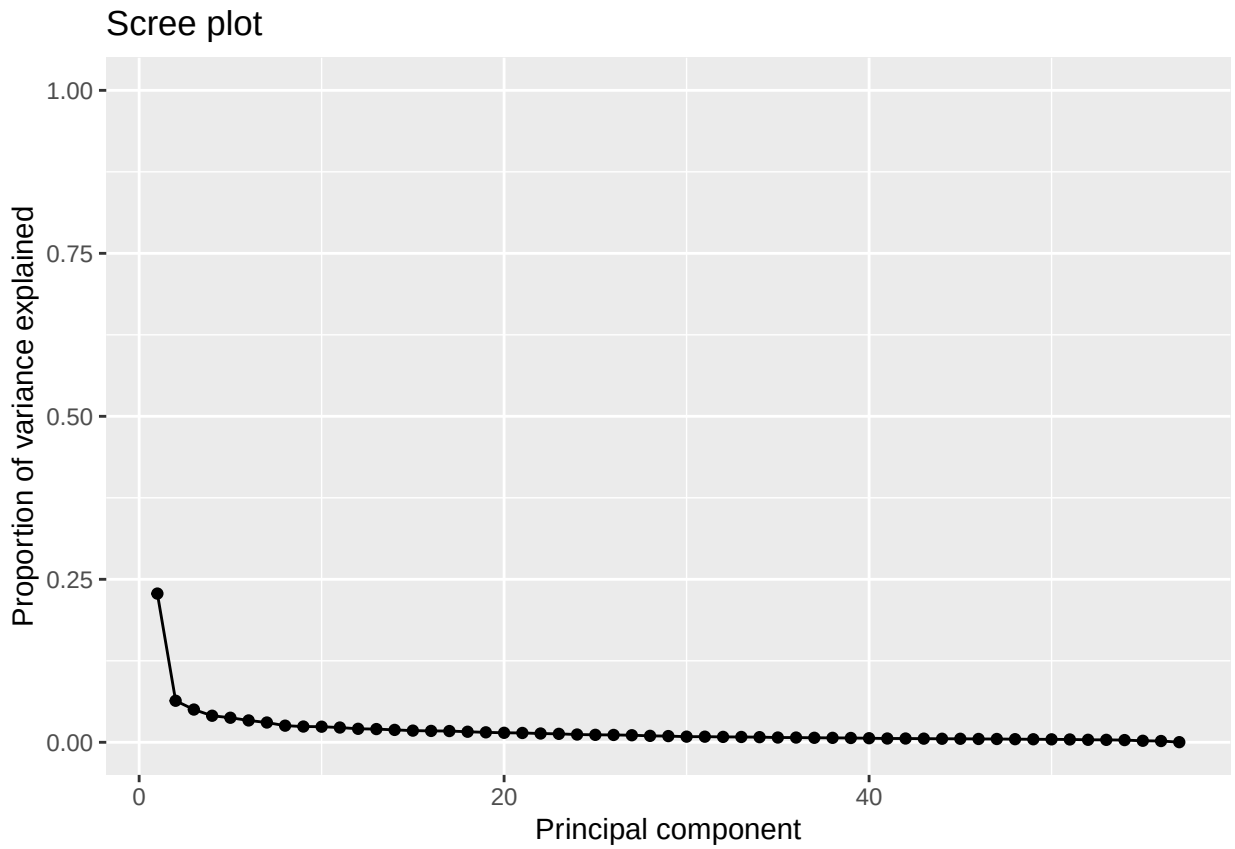
- The plot shows variables including `domain_Law_Politics`, `qu4`, `domain_Sciences`, and `uoc_position_Adjunct` are strongly positively correlated with the first principal component; variables including `pu1`, `pu2`, `pu3` and `exp2` are strongly negatively correlated with the first principal component.

- Variables including `inc2`, `peu 1`, `peu 2` and `inc1` are strongly positively correlated with the second component; variables including `pf3`, `exp4`, `pf1`, and `domain_Engineering_Architecture` are strongly negatively correlated with the second principal component.

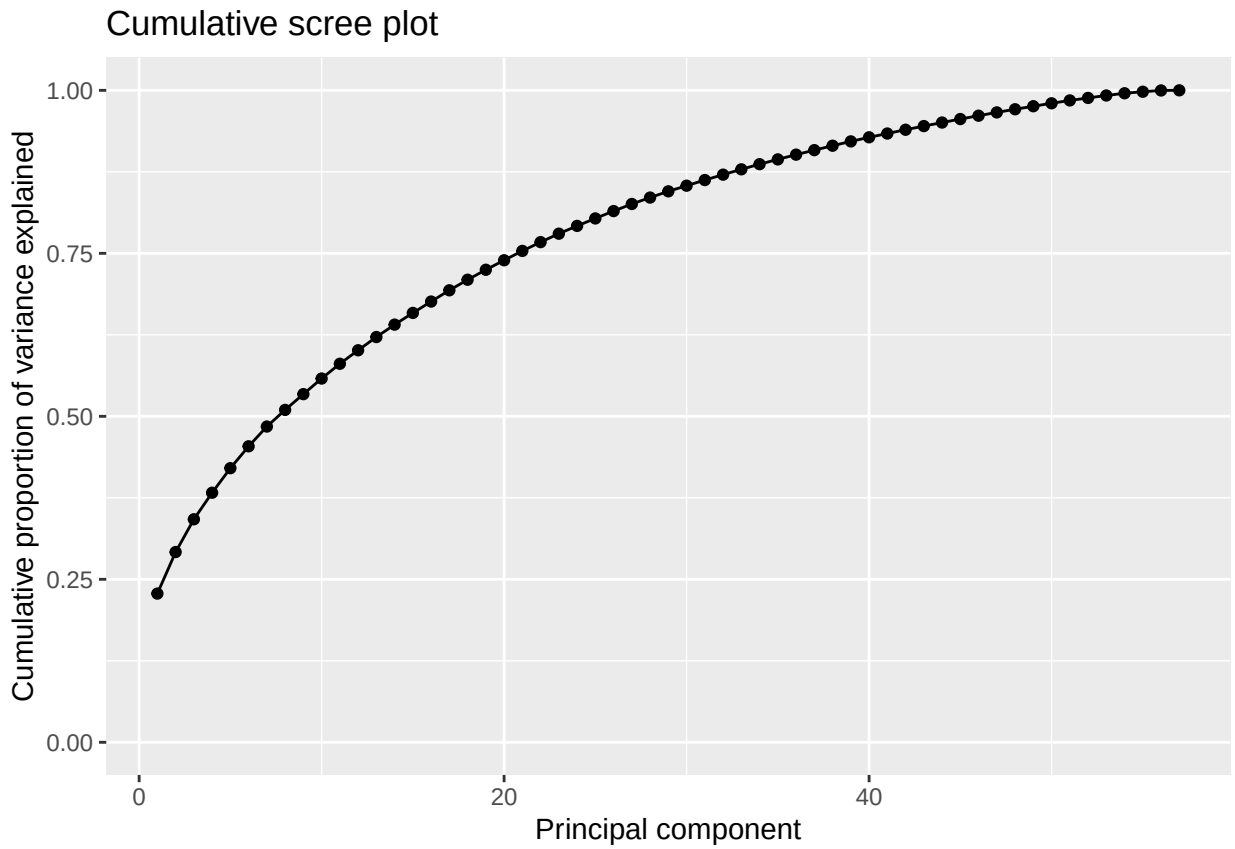
```
#extract values from the eigen matrix
wiki_pve <- tibble(
  var = wiki_eigen$values,
  #calculated PVE by hand
  var_exp = var / sum(var),
  cum_var_exp = cumsum(var_exp)
) %>%
  mutate(pc = row_number())

head(wiki_pve, n = 10)
```

```
## # A tibble: 10 x 4
##      var var_exp cum_var_exp    pc
##    <dbl>   <dbl>       <dbl> <int>
##  1 13.0   0.228        0.228     1
##  2  3.63  0.0637       0.292     2
##  3  2.86  0.0502       0.342     3
##  4  2.32  0.0407       0.383     4
##  5  2.15  0.0377       0.420     5
##  6  1.91  0.0335       0.454     6
##  7  1.73  0.0303       0.484     7
##  8  1.45  0.0255       0.510     8
##  9  1.38  0.0242       0.534     9
## 10  1.36  0.0239       0.558    10
```

```
#PVE for each principal component
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")
```



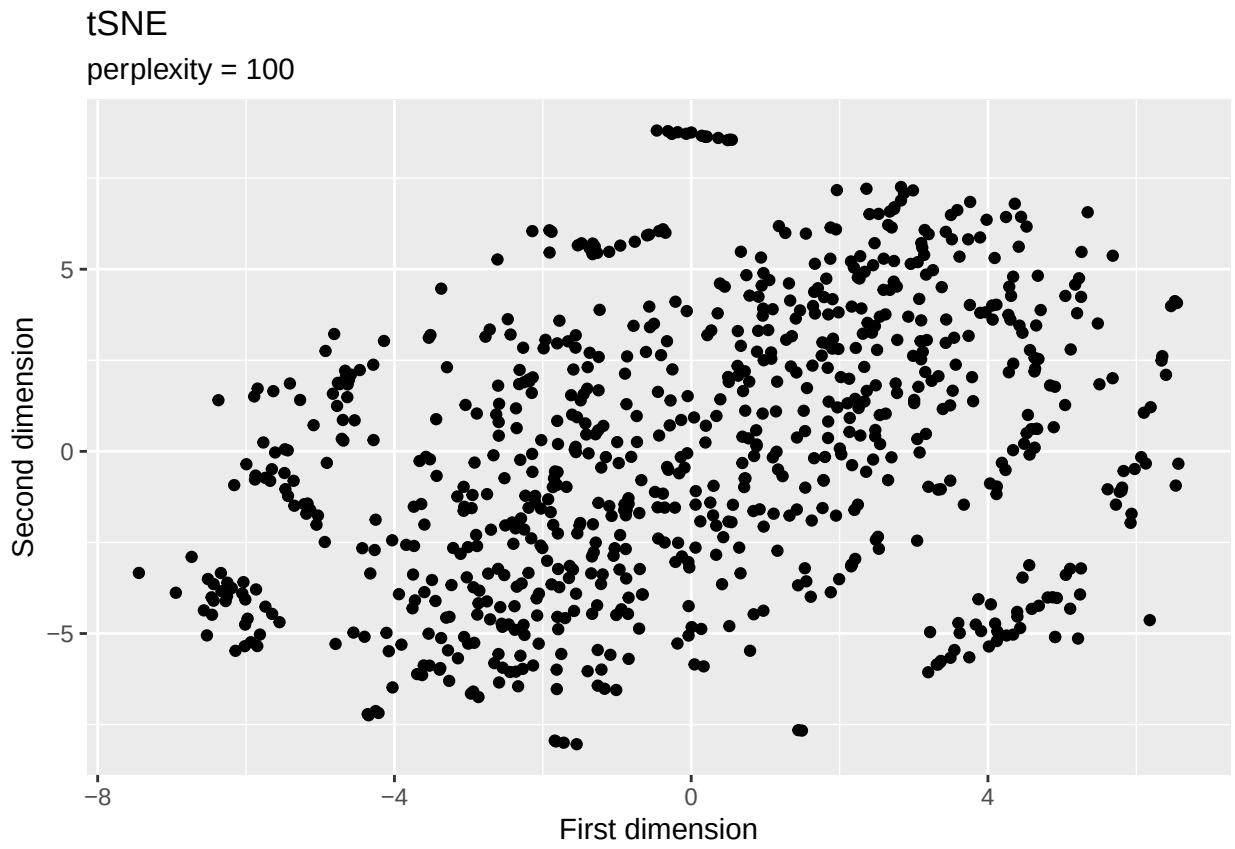

```
#Cumulative PVE
ggplot(wiki_pve, aes(pc, cum_var_exp)) +
  geom_line() +
  geom_point() +
  ylim(0, 1) +
  labs(title = "Cumulative scree plot",
       x = "Principal component",
       y = "Cumulative proportion of variance explained")
```



Approximately 30% of the variance is explained by the first two components.

t-SNE

```
#use scaled dataset for t-SNE
wiki_tsne <- Rtsne(as.matrix(wiki_scaled), perplexity = 100)

#plot the results
wiki_scaled %>%
  mutate(tsne1 = wiki_tsne$Y[,1],
         tsne2 = wiki_tsne$Y[,2]) %>%
  ggplot(aes(tsne1, tsne2, color = )) +
  geom_point() +
  labs(title = "tSNE",
       subtitle = "perplexity = 100",
       x = "First dimension",
       y = "Second dimension")
```



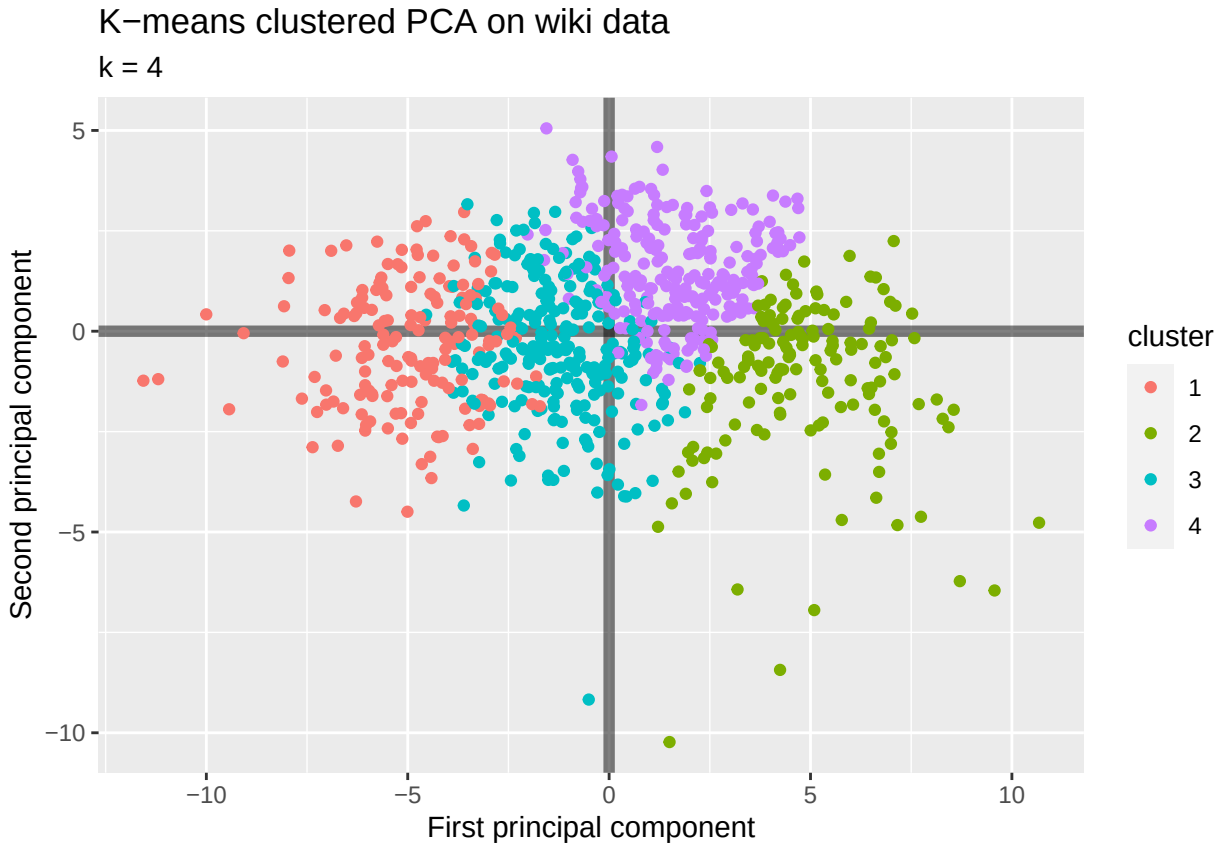
- The t-SNE plot alone here doesn't tell much about the intrinsic relations among the data points. Here it looks like there is one major cluster, and two to six smaller clusters. But making judgment by appearances is not wise.
- In a t-SNE plot, the grouping of points and the sizes of groups can be misleading, because the size of groups is generally meaningless, and as is the grouping itself. Moreover, since we don't know much about the dataset, it may very well be mixed with noises, which can influence the result of t-SNE.

Clustering

```
km4 <- kmeans(wiki_scaled, centers = 4, nstart = 4, iter.max = 100)

PC4 <- tibble(
  PC1 = PC1[,1],
  PC2 = PC2[,1],
  cluster = as.factor(km4$cluster)
)
```

```
ggplot(PC4, aes(PC1, PC2, color = cluster)) +
  geom_vline(xintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_hline(yintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_point() +
```

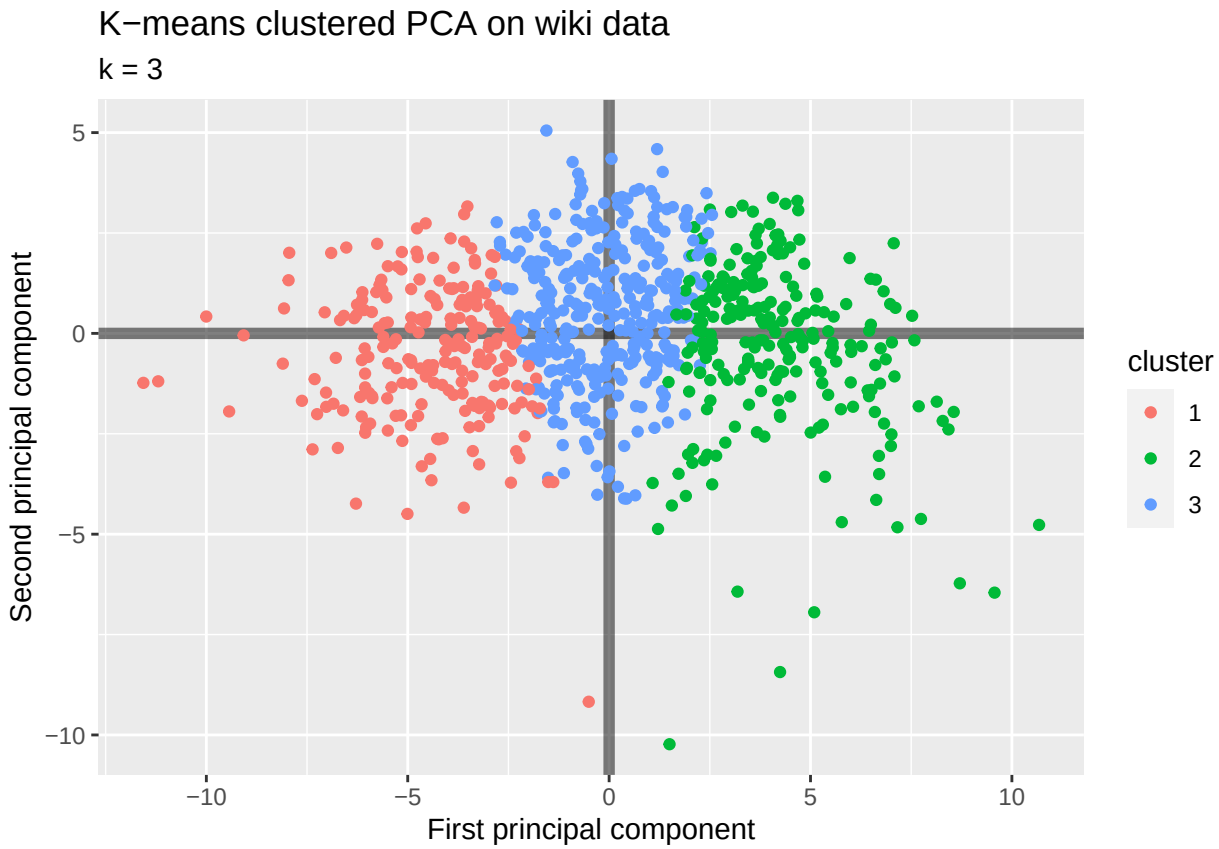
```
  labs(title = "K-means clustered PCA on wiki data",
       subtitle = "k = 4",
       x = "First principal component",
       y = "Second principal component")
```



$\mathbf{k = 4}$

```
km3 <- kmeans(wiki_scaled, centers = 3, nstart = 3, iter.max = 100)

PC3 <- tibble(
  PC1 = PC1[,1],
  PC2 = PC2[,1],
  cluster = as.factor(km3$cluster)
)
```

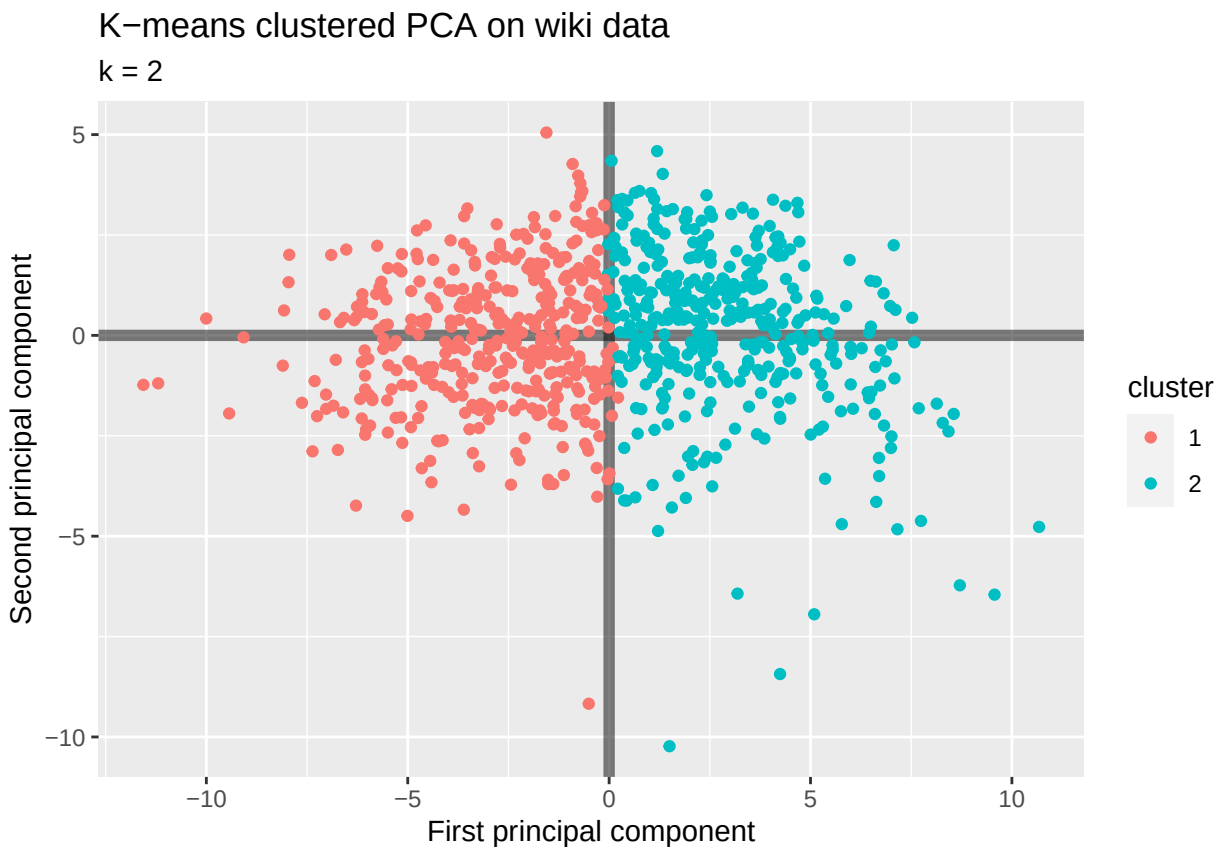
```
ggplot(PC3, aes(PC1, PC2, color = cluster)) +
  geom_vline(xintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_hline(yintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_point() +
  labs(title = "K-means clustered PCA on wiki data",
       subtitle = "k = 3",
       x = "First principal component",
       y = "Second principal component")
```



k = 3

```
km2 <- kmeans(wiki_scaled, centers = 2, nstart = 2, iter.max = 100)

PC2 <- tibble(
  PC1 = PC1[,1],
  PC2 = PC2[,1],
  cluster = as.factor(km2$cluster)
)
```

```
ggplot(PC2, aes(PC1, PC2, color = cluster)) +
  geom_vline(xintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_hline(yintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_point() +
  labs(title = "K-means clustered PCA on wiki data",
       subtitle = "k = 2",
       x = "First principal component",
       y = "Second principal component")
```



k = 2

- As we can see by comparing corresponding plots for the three values of k, k = 2 appears to be the optimal value here.
- When k = 4, cluster 1 has points mixed with both cluster 2 and cluster 3, which means points in this cluster can not be very well distinguished from points from the other two clusters. When k = 3, between cluster 2 and 3, there are also points mixed with those from the other cluster. Although point mixure is much less compared with k = 4, the result of k = 3 is not ideal either.
- When k = 2, the two clusters can be clearly distinguished from each other. There is an almost linear boundary along the y-axis(the second principal component) between the two clusters. That makes k = 2 the best among the three values of k.

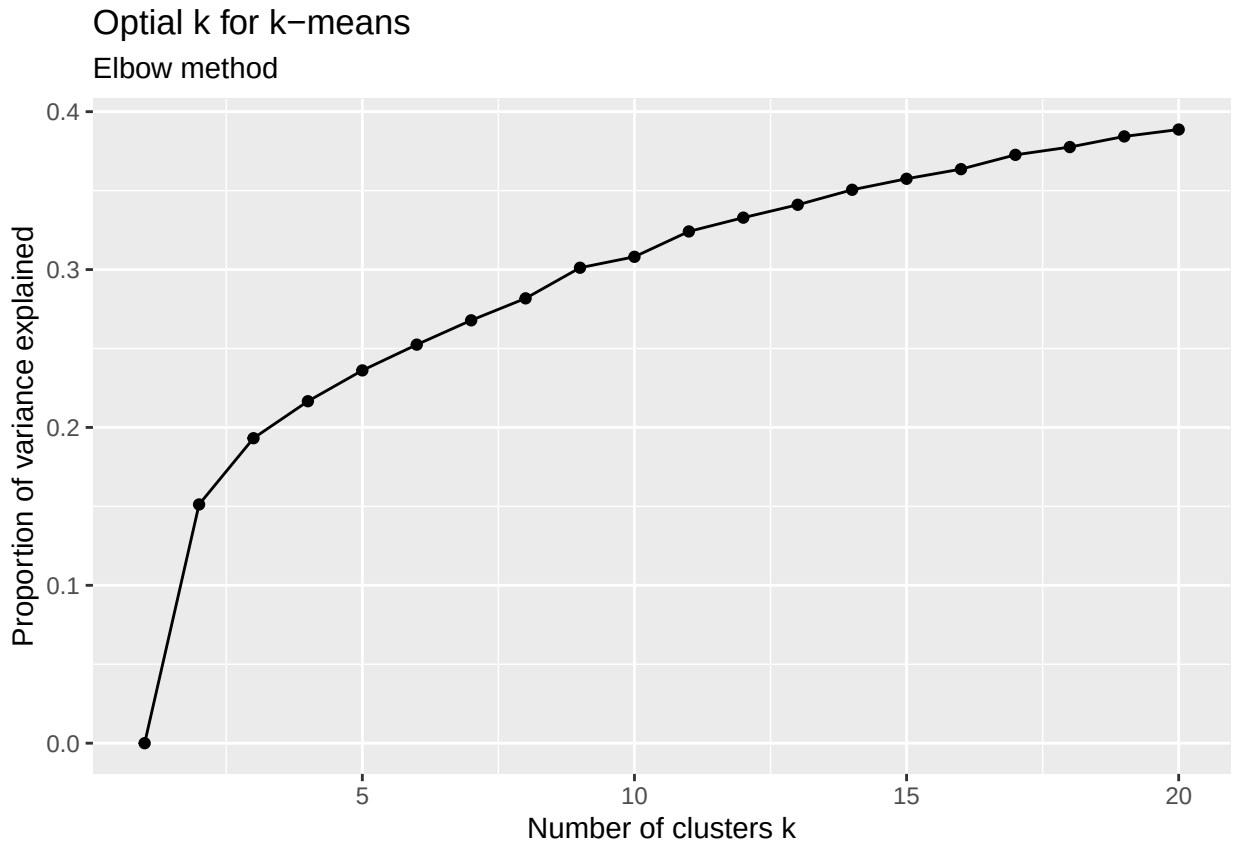
tech issue

- Note: there's an issue with `ggplot2`. When I plot the kmeans graphs from k = 2 to k = 4, it fails at k = 3; when I plot from k = 4 to k = 2, it works, but only once unless I clean the environment.
- attempts to install CRAN version of `ggplot2` with `install_github('cran/ggplot2')` fails with error message: cannot remove prior installation of package 'glue'

Elbow method

```
#kmean() calculates the between cluster sum of squares(betweenss)
#and the total sum of sqaures(totss)
#the proportion of variance explained is betweenss/totss
wss <- function(k) {
```

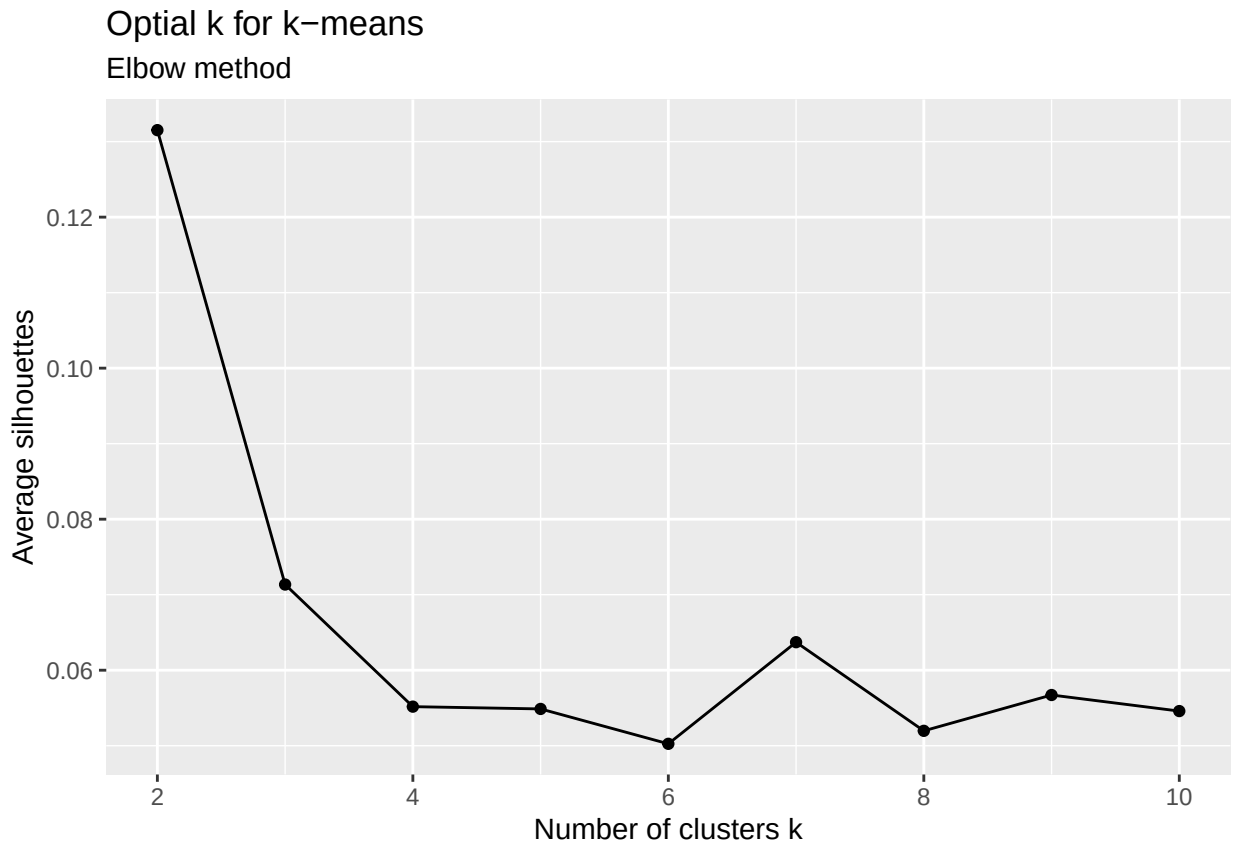
```
  tempk <- kmeans(wiki_scaled, k, nstart = 20)
  tempk$betweenss / tempk$totss
}

#plot the change of PVE
tibble(
  k = 1:20
) %>%
  mutate(wss = map_dbl(k, wss)) %>%
  ggplot(aes(k, wss)) +
  geom_line() +
  geom_point() +
  labs(title = "Optial k for k-means",
       subtitle = "Elbow method",
       x = "Number of clusters k",
       y = "Proportion of variance explained")
```



Average silhouette

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

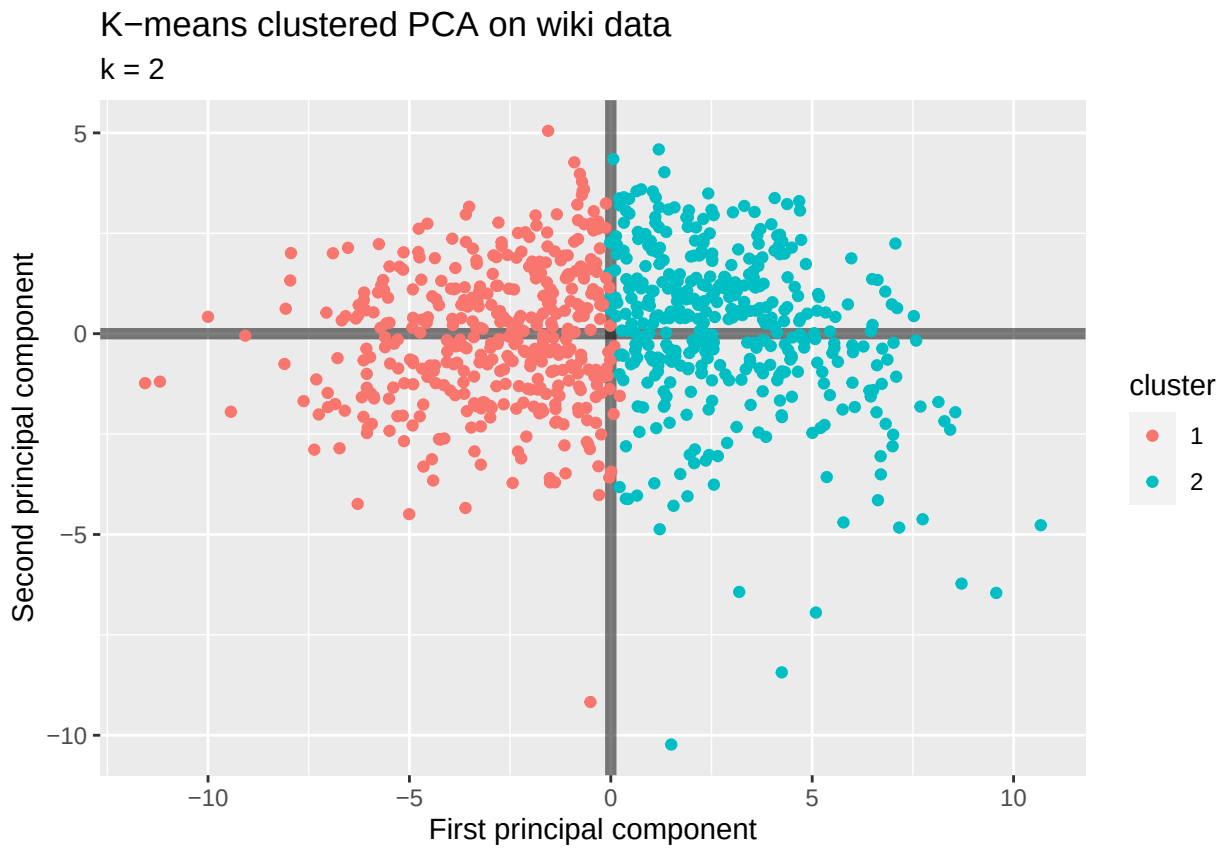
```
}

tibble(
  k = 2:10
) %>%
  mutate(avg_sil = map_dbl(k, avg_sil)) %>%
  ggplot(aes(k, avg_sil)) +
  geom_line() +
  geom_point() +
  labs(title = "Optial k for k-means",
       subtitle = "Elbow method",
       x = "Number of clusters k",
       y = "Average silhouettes")
```



- It appears the optimal k is 2.
- The elbow method cannot very show why 2 is the optimal value for k, since there is no clear "elbow" in the plot. Nonetheless, we can see that at k = 2, PVE experiences the most drastic increase. After k = 2, the increase of PVE gradually decreases. That suggests k = 2 could be a good fit.
- The average silhouette has the highest value at k = 2, which clearly suggests 2 is the optimal value for this data set.

```
wiki_prc <- prcomp(wiki, scale. = TRUE)
PC_op <- as.data.frame(wiki_prc$x)

km_op <- kmeans(wiki_scaled, centers = 2, nstart = 3, iter.max = 100)
```

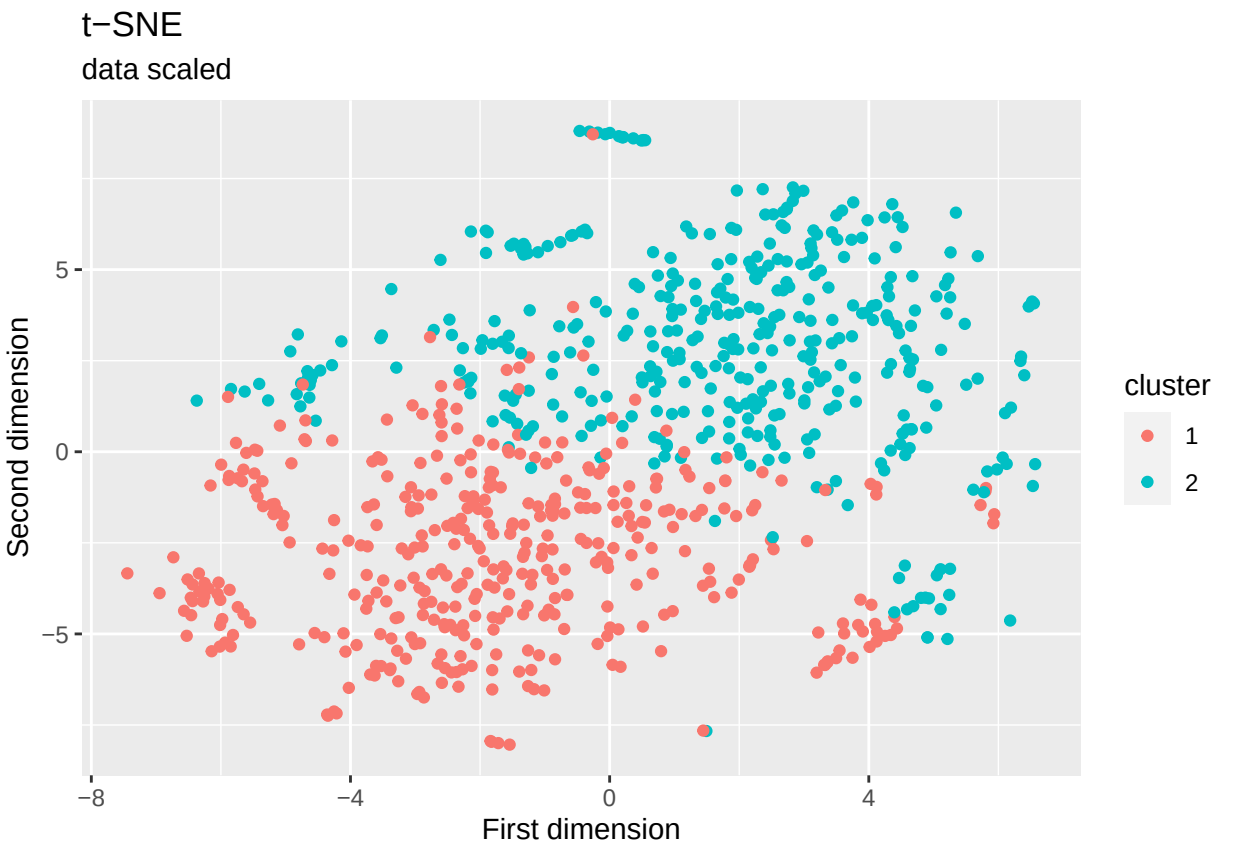
```
PC_final <- tibble(
  PC1 = PC_op$PC1,
  PC2 = PC_op$PC2,
  cluster = as.factor(km_op$cluster)
)

ggplot(PC_final, aes(PC1, PC2, color = cluster)) +
  geom_vline(xintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_hline(yintercept = 0, size = 2, color = "black", alpha = .5) +
  geom_point() +
  labs(title = "K-means clustered PCA on wiki data",
       subtitle = "k = 2",
       x = "First principal component",
       y = "Second principal component")
```



Since k = 2 is the optimal setting for the dataset, this graph is the same as the one previously shown in comparing k = 2, 3, 4. There is a clear boundary between the two clusters along the axis of Second principal component.

```
wiki_scaled %>%
  cbind(cluster = PC_final$cluster) %>%
  mutate(tsne1 = wiki_tsne$Y[,1],
         tsne2 = wiki_tsne$Y[,2]) %>%
  ggplot(aes(tsne1, tsne2, color = cluster)) +
  geom_point() +
```

```
labs(title = "t-SNE",
     subtitle = "data scaled",
     x = "First dimension",
     y = "Second dimension")
```



- On one hand, with the data points colored according to their clusters, it appears that 100 is not a bad value for using t-SNE on this dataset. Here the data points are roughly divided into two groups.
- However, the shapes of these two groups are very different from what one may imagine by looking at the uncolored t-SNE plot. In particular, what one takes for a major cluster appears to be divided by two k-mean clusters. The smaller outlying clusters are grouped to the major k-mean cluster closer to them.
- On the other hand, this plot suggests that using clustering after t-SNE can be misleading. As is mentioned above, cluster sizes and cluster distances in a t-SNE plot may not mean anything. Indeed, here neither mean anything. Although the points are roughly grouped according to their k-mean clusters, there are still points mixed in the other cluster. Moreover, the way the k-mean clusters are divided suggests k-mean clustering has little to do with how the data points are ploted in the t-SNE plot.
- In general, we have to be cautious in interpreting t-SNE plots.