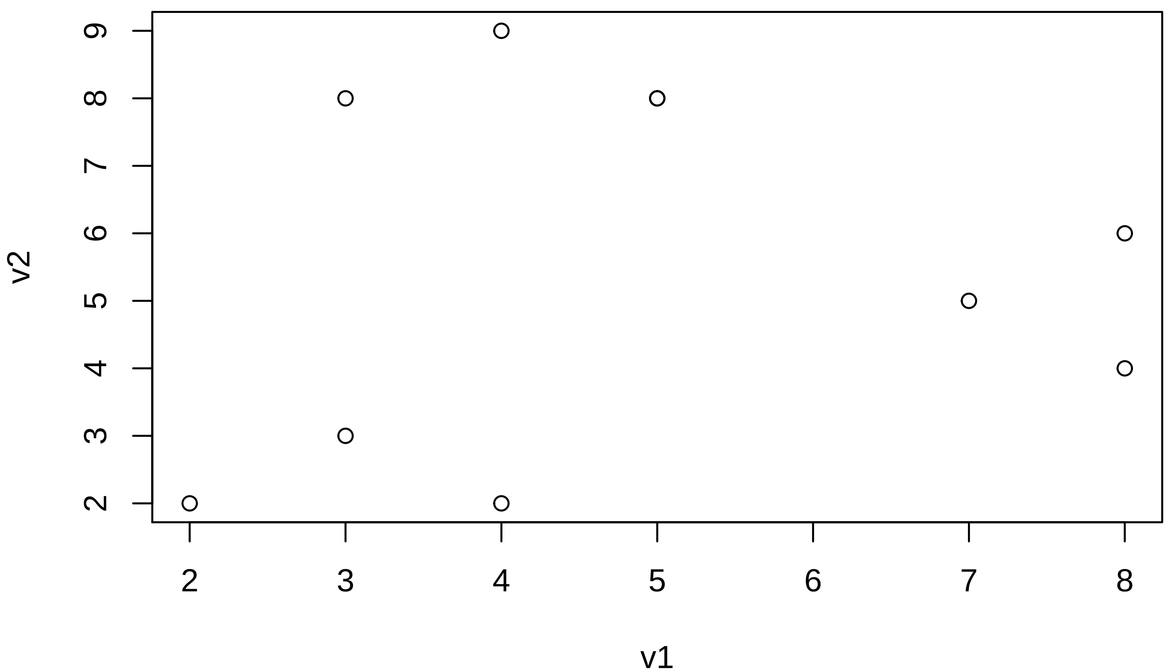
Mingtao_Gao_HW7

Mingtao Gao

3/14/2020

k-Means Clustering "By Hand" 0.

```
library(cluster)
library(ggplot2)
set.seed(1234)
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)
input <- data.frame(v1=input_1, v2=input_2)
plot(input)
```



1.

```
# Assigning each observation to a cluster at random
cluster <- sample(c(1,2,3), size=nrow(input), replace=TRUE)
```

2.

```
# Compute the cluster centroid and update cluster assignments
# based on spatial similarity.
centroids <- input[sample.int(nrow(input),3),]
stop_crit <- 1000
converged <- FALSE
it=1
```

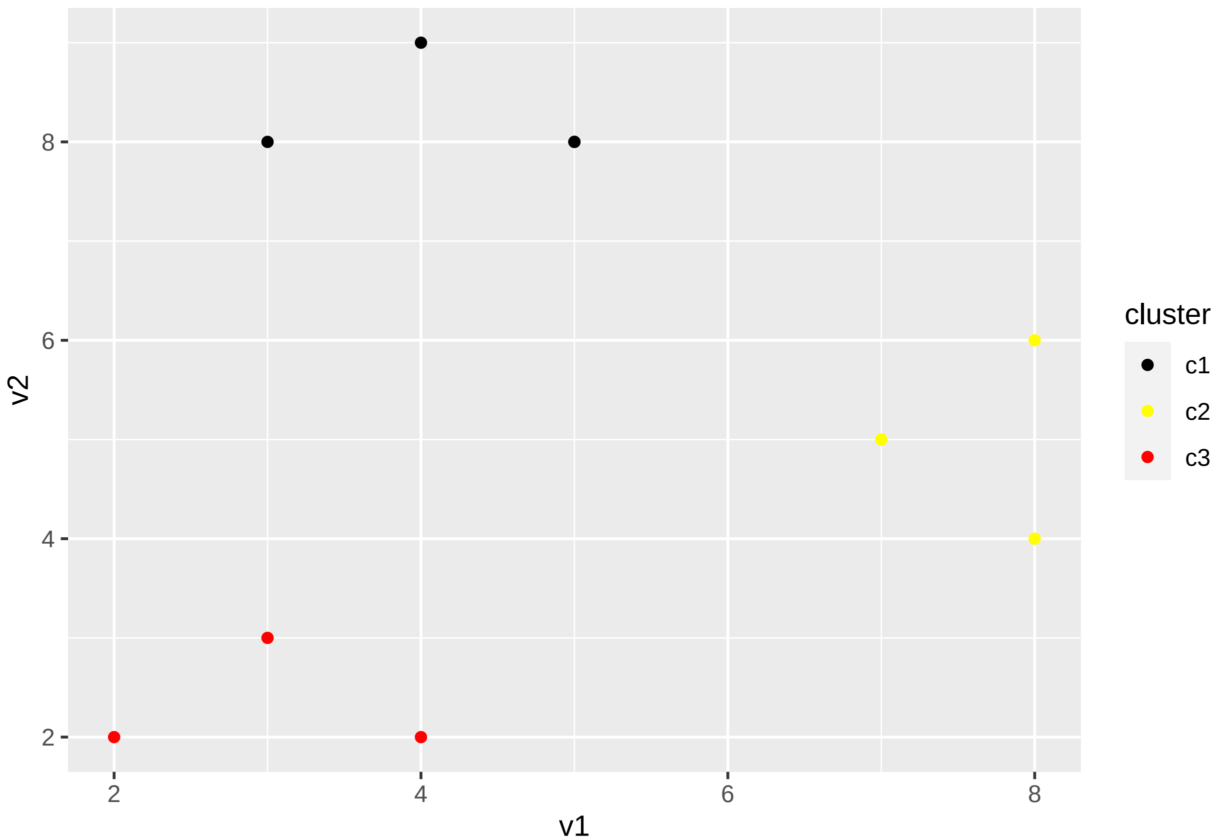
```r
while (stop_crit >= 0.001 & converged==F) {
  it=it+1
  if (stop_crit <= 0.001) {
    converged=T
  }

  old_centroids=centroids

  ##Assigning each point to a centroid
  for (i in 1:nrow(input)) {
    min_dist=10e10
    for (centroid in 1:nrow(centroids)) {
      distance_to_centroid <- sum((centroids[centroid,] - input[i,])^2)
      if (distance_to_centroid<=min_dist) {
        cluster[i]=centroid
        min_dist=distance_to_centroid
      }
    }
  }

  for (i in 1:nrow(centroids)) {
    centroids[i,]=apply(input[cluster==i,],2,mean)
  }
  stop_crit <- mean(colMeans((old_centroids-centroids)^2))
}
k3.out <- list(data=data.frame(input,cluster),centroids=centroids)
k3.out
```

```
## $data
##    v1 v2 cluster
## 1   5  8       1
## 2   8  6       2
## 3   7  5       2
## 4   8  4       2
## 5   3  3       3
## 6   4  2       3
## 7   2  2       3
## 8   3  8       1
## 9   4  9       1
## 10  5  8       1
##
## $centroids
##          v1       v2
## 10 4.250000 8.250000
## 6  7.666667 5.000000
## 7  3.000000 2.333333
```

3.

```r
# Visual description of the final cluster assignments for K=3
ggplot() +
  geom_point(data=k3.out$data,
                     mapping=aes(x=v1,
                                 y=v2,
```

```
                                colour=cut(cluster, c(0, 1, 2, 3)))) +
  scale_color_manual(name = "cluster",
                     values = c("(0,1]" = "black",
                                "(1,2]" = "yellow",
                                "(2,3]" = "red"),
                     labels = c("c1", "c2", "c3"))
```



From the graph above, we can see the data was classified into three clusters.

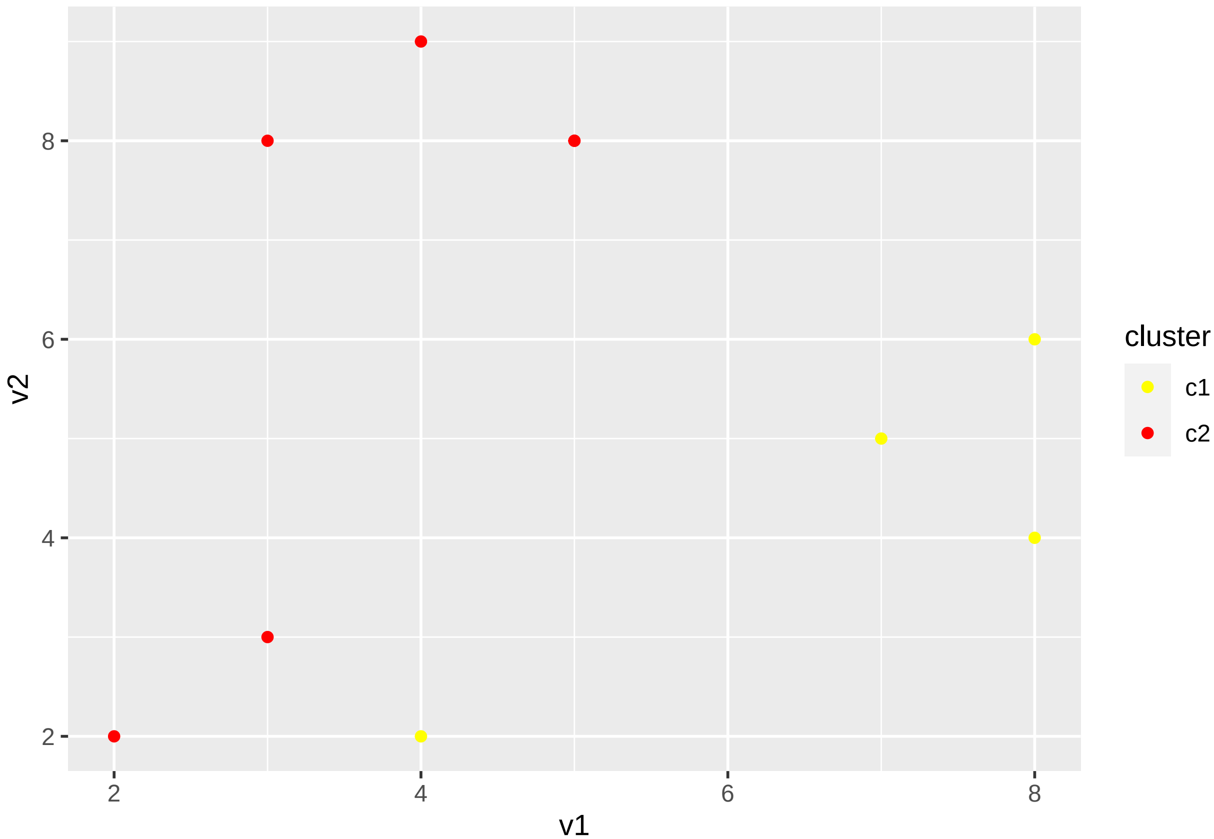
4.

```
# Repeat the same process with K = 2

# Assigning each observation to a cluster at random
cluster <- sample(c(0,1), size=nrow(input), replace=TRUE)

# Compute the cluster centroid and update cluster assignments
# based on spatial similarity.
centroids <- input[sample.int(nrow(input),2),]
stop_crit <- 1000
converged <- FALSE
it=1

while (stop_crit >= 0.001 & converged==F) {
  it=it+1
  if (stop_crit <= 0.001) {
    converged=T
  }
```

```r
  old_centroids=centroids

  ##Assigning each point to a centroid
  for (i in 1:nrow(input)) {
    min_dist=10e10
    for (centroid in 1:nrow(centroids)) {
      distance_to_centroid <- sum((centroids[centroid,] - input[i,])^2)
      if (distance_to_centroid<=min_dist) {
        cluster[i]=centroid
        min_dist=distance_to_centroid
      }
    }
  }

  for (i in 1:nrow(centroids)) {
    centroids[i,]=apply(input[cluster==i,],2,mean)
  }
  stop_crit <- mean(colMeans((old_centroids-centroids)^2))
}
k2.out <- list(data=data.frame(input,cluster),centroids=centroids)
k2.out
```

```
## $data
##    v1 v2 cluster
## 1   5  8       2
## 2   8  6       1
## 3   7  5       1
## 4   8  4       1
## 5   3  3       2
## 6   4  2       1
## 7   2  2       2
## 8   3  8       2
## 9   4  9       2
## 10  5  8       2
##
## $centroids
##         v1       v2
## 4 6.750000 4.250000
## 8 3.666667 6.333333
```

```r
# Visual description of the final cluster assignments for K=2
ggplot() +
  geom_point(data=k2.out$data,
                    mapping=aes(x=v1,
                                y=v2,
                                colour=cut(cluster, c(0, 1, 2)))) +
  scale_color_manual(name = "cluster",
                    values = c("(0,1]" = "yellow",
                               "(1,2]" = "red"),
                    labels = c("c1", "c2"))
```



5. Based on above analysis, we can see the initial 3 clusters fit these data better. The basic idea behind partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation is minimized. Looking at the graph with 3 cluters, we can see the variances is much smaller compared to the graph with 2 cluters. The points within the same cluster are more spread out in K=2 graph, and thus K=3 performs better.

Dimension Reduction 6.

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

# Perform scaled PCA
wiki.pca <- prcomp(wiki, center=TRUE, scale=TRUE)
# Inspect model output
summary(wiki.pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6058 1.90586 1.69219 1.52365 1.46547 1.38228 1.31487
## Proportion of Variance 0.2281 0.06372 0.05024 0.04073 0.03768 0.03352 0.03033
## Cumulative Proportion  0.2281 0.29183 0.34207 0.38280 0.42047 0.45399 0.48433
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.20613 1.17385 1.16779 1.13641 1.08654 1.07515 1.04158
## Proportion of Variance 0.02552 0.02417 0.02393 0.02266 0.02071 0.02028 0.01903
## Cumulative Proportion  0.50985 0.53402 0.55795 0.58060 0.60132 0.62160 0.64063
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     1.01080 0.99808 0.99197 0.96071 0.93339 0.91156 0.90379
## Proportion of Variance 0.01792 0.01748 0.01726 0.01619 0.01528 0.01458 0.01433
## Cumulative Proportion  0.65855 0.67603 0.69329 0.70949 0.72477 0.73935 0.75368
```

```
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.8771 0.85951 0.82448 0.80853 0.80231 0.78661 0.75050
## Proportion of Variance 0.0135 0.01296 0.01193 0.01147 0.01129 0.01086 0.00988
## Cumulative Proportion  0.7672 0.78014 0.79206 0.80353 0.81482 0.82568 0.83556
##                           PC29    PC30    PC31   PC32    PC33   PC34    PC35
## Standard deviation     0.73659 0.70268 0.70157 0.6878 0.68203 0.6708 0.64653
## Proportion of Variance 0.00952 0.00866 0.00864 0.0083 0.00816 0.0079 0.00733
## Cumulative Proportion  0.84508 0.85374 0.86238 0.8707 0.87884 0.8867 0.89407
##                           PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.64385 0.62790 0.62332 0.61367 0.59686 0.57617 0.57549
## Proportion of Variance 0.00727 0.00692 0.00682 0.00661 0.00625 0.00582 0.00581
## Cumulative Proportion  0.90134 0.90826 0.91507 0.92168 0.92793 0.93375 0.93956
##                          PC43    PC44    PC45    PC46    PC47   PC48    PC49
## Standard deviation     0.5650 0.55613 0.55423 0.54026 0.53701 0.5231 0.51546
## Proportion of Variance 0.0056 0.00543 0.00539 0.00512 0.00506 0.0048 0.00466
## Cumulative Proportion  0.9452 0.95059 0.95598 0.96110 0.96616 0.9710 0.97562
##                           PC50    PC51    PC52    PC53    PC54    PC55    PC56
## Standard deviation     0.50816 0.49831 0.46804 0.46300 0.43911 0.36615 0.33486
## Proportion of Variance 0.00453 0.00436 0.00384 0.00376 0.00338 0.00235 0.00197
## Cumulative Proportion  0.98015 0.98451 0.98835 0.99211 0.99549 0.99784 0.99981
##                           PC57
## Standard deviation     0.10351
## Proportion of Variance 0.00019
## Cumulative Proportion  1.00000
```

```
str(wiki.pca)
```

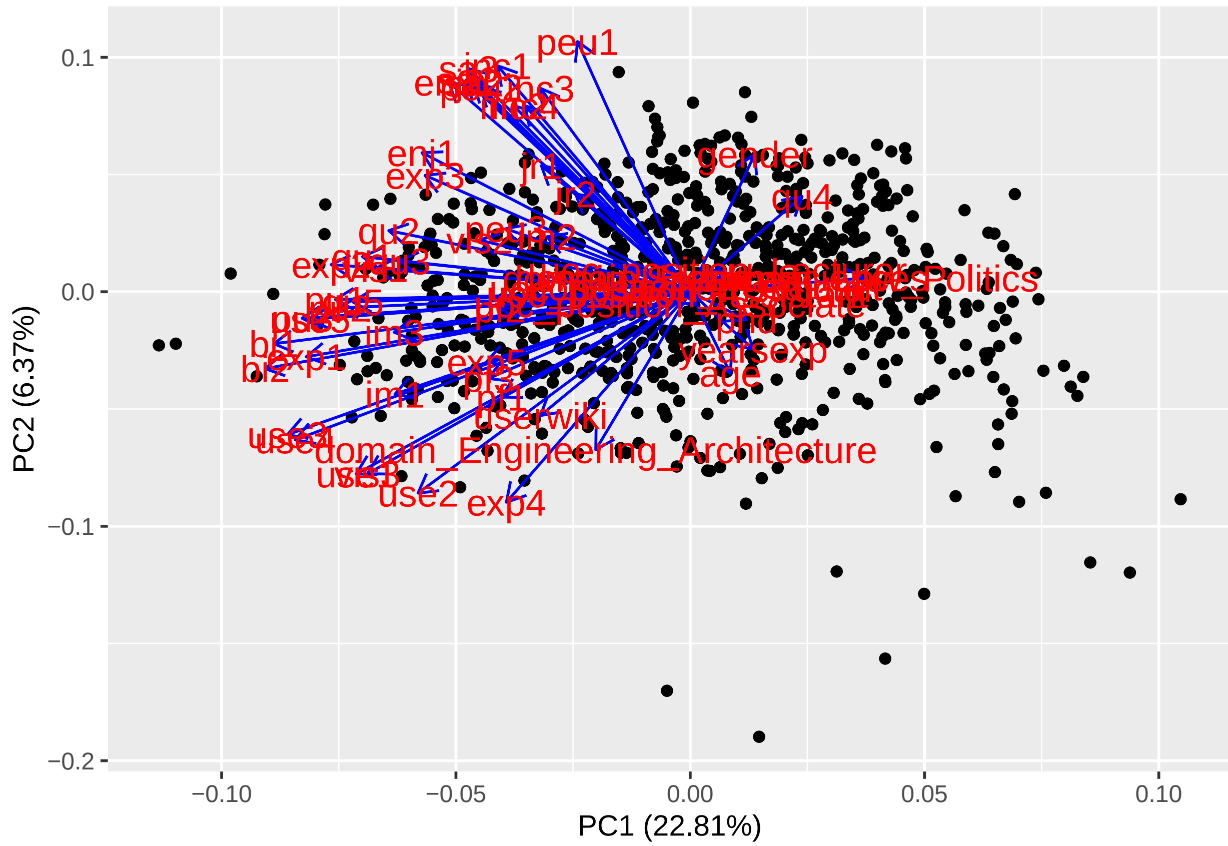
```
## List of 5
##  $ sdev    : num [1:57] 3.61 1.91 1.69 1.52 1.47 ...
##  $ rotation: num [1:57, 1:57] 0.0218 0.0351 0.0305 0.0342 -0.0814 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:57] "age" "gender" "phd" "yearsexp" ...
##   .. ..$ : chr [1:57] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:57] 42.166 0.427 0.434 10.409 0.136 ...
##   ..- attr(*, "names")= chr [1:57] "age" "gender" "phd" "yearsexp" ...
##  $ scale   : Named num [1:57] 7.548 0.495 0.496 6.757 0.343 ...
##   ..- attr(*, "names")= chr [1:57] "age" "gender" "phd" "yearsexp" ...
##  $ x       : num [1:800, 1:57] 0.15 3.31 4.68 -1.77 -7.25 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:57] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```
wiki.pca$rotation[,1:2]
```

```
##                                  PC1          PC2
## age                      0.021805412 -0.088384981
## gender                   0.035086317  0.149461450
## phd                      0.030501043 -0.030435497
## yearsexp                 0.034190490 -0.062364714
## userwiki                -0.081363144 -0.134387358
## pu1                     -0.192827065 -0.008273053
## pu2                     -0.190587716 -0.017668791
## pu3                     -0.210862567 -0.028776289
## peu1                    -0.061228008  0.271741292
```

```
## peu2                            -0.113718709  0.222367978
## peu3                            -0.100218922  0.068458517
## enj1                            -0.145666175  0.151011732
## enj2                            -0.131109826  0.227602424
## qu1                             -0.178057029  0.038122429
## qu2                             -0.163777789  0.066421876
## qu3                             -0.157956174  0.033472352
## qu4                              0.060796858  0.103458415
## qu5                             -0.183364593 -0.010911790
## vis1                            -0.171153058  0.025207626
## vis2                            -0.114558913  0.056217768
## vis3                            -0.175351292 -0.197634740
## im1                             -0.160432141 -0.111106146
## im2                             -0.077810159  0.059774521
## im3                             -0.160803391 -0.044003603
## sa1                             -0.121658435  0.229926005
## sa2                             -0.117590405  0.226760395
## sa3                             -0.120376196  0.242325421
## use1                            -0.181477170 -0.197827499
## use2                            -0.147851769 -0.218628501
## use3                            -0.218809245 -0.155151571
## use4                            -0.214558397 -0.160864524
## use5                            -0.206538888 -0.029823253
## pf1                             -0.102337996 -0.114370782
## pf2                             -0.103448162 -0.018604706
## pf3                             -0.109632421 -0.094172517
## jr1                             -0.080866885  0.136967544
## jr2                             -0.062216127  0.106296824
## bi1                             -0.226193061 -0.056374273
## bi2                             -0.230923964 -0.083430888
## inc1                            -0.104666756  0.245439824
## inc2                            -0.095802250  0.202021404
## inc3                            -0.081401727  0.220985795
## inc4                            -0.089707244  0.202022006
## exp1                            -0.208591685 -0.070543836
## exp2                            -0.195043150  0.029560476
## exp3                            -0.144023257  0.126416909
## exp4                            -0.099872875 -0.228494272
## exp5                            -0.110628098 -0.076095685
## domain_Sciences                 -0.021982007  0.014536737
## domain_Health.Sciences           0.017157681  0.015478496
## domain_Engineering_Architecture -0.051309109 -0.171483803
## domain_Law_Politics              0.094774659  0.014887154
## uoc_position_Associate          -0.010922081 -0.013134181
## uoc_position_Assistant          -0.007123091 -0.002311281
## uoc_position_Lecturer            0.018040923  0.023591030
## uoc_position_Instructor         -0.004250607  0.003784534
## uoc_position_Adjunct             0.007848555  0.005301224
```

```r
library(ggfortify); library(ggplot2)

autoplot(wiki.pca, data=wiki,loadings = TRUE, loadings.colour = 'blue',
         loadings.label=TRUE, loadings.label.size=5)
```



From the graph and rotation matrix displayed above, we can see, in this case, high values of PC1 are strongly associated with low values of bi2 (loading: -0.23), bi1 (loading: -0.22), pu3 use3, and use4 values (loading: -0.21). Other varialbes that also associate with PC1 are use5 and exp1. PC1 explains 22.8% of the total variance, which means that more than one-fifth of the information in the dataset can be encapsulated by the first one Principal Component.

PC2 on the other hand (explaining 6.4% of the variance), is largely influenced by peu1 (the associated loading is 0.27), sa3, inc1 (both loading: 0.24), and enj2 (loading: 0.23).

7.

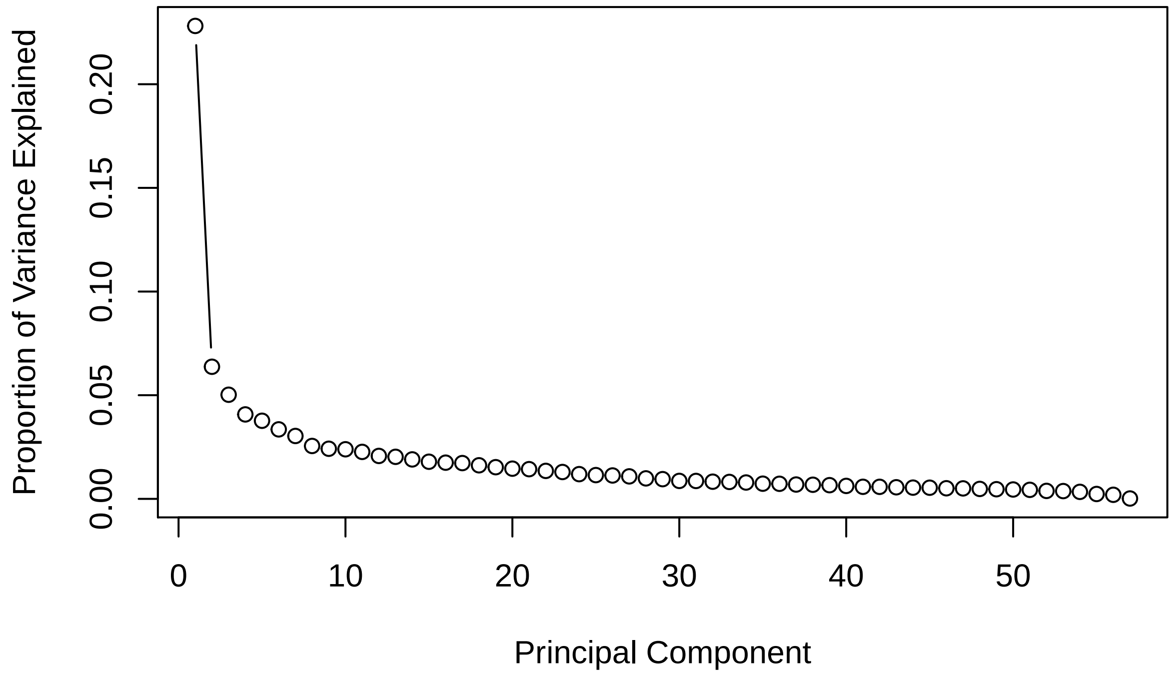
```r
# Variability of each principal component
wikipca.var <- wiki.pca$sdev^2

# Variance explained by each principal component: pve
pve <- wikipca.var / sum(wikipca.var)
round(pve, 3)
```

```
##  [1] 0.228 0.064 0.050 0.041 0.038 0.034 0.030 0.026 0.024 0.024 0.023 0.021
## [13] 0.020 0.019 0.018 0.017 0.017 0.016 0.015 0.015 0.014 0.013 0.013 0.012
## [25] 0.011 0.011 0.011 0.010 0.010 0.009 0.009 0.008 0.008 0.008 0.007 0.007
## [37] 0.007 0.007 0.007 0.006 0.006 0.006 0.006 0.005 0.005 0.005 0.005 0.005
## [49] 0.005 0.005 0.004 0.004 0.004 0.003 0.002 0.002 0.000
```

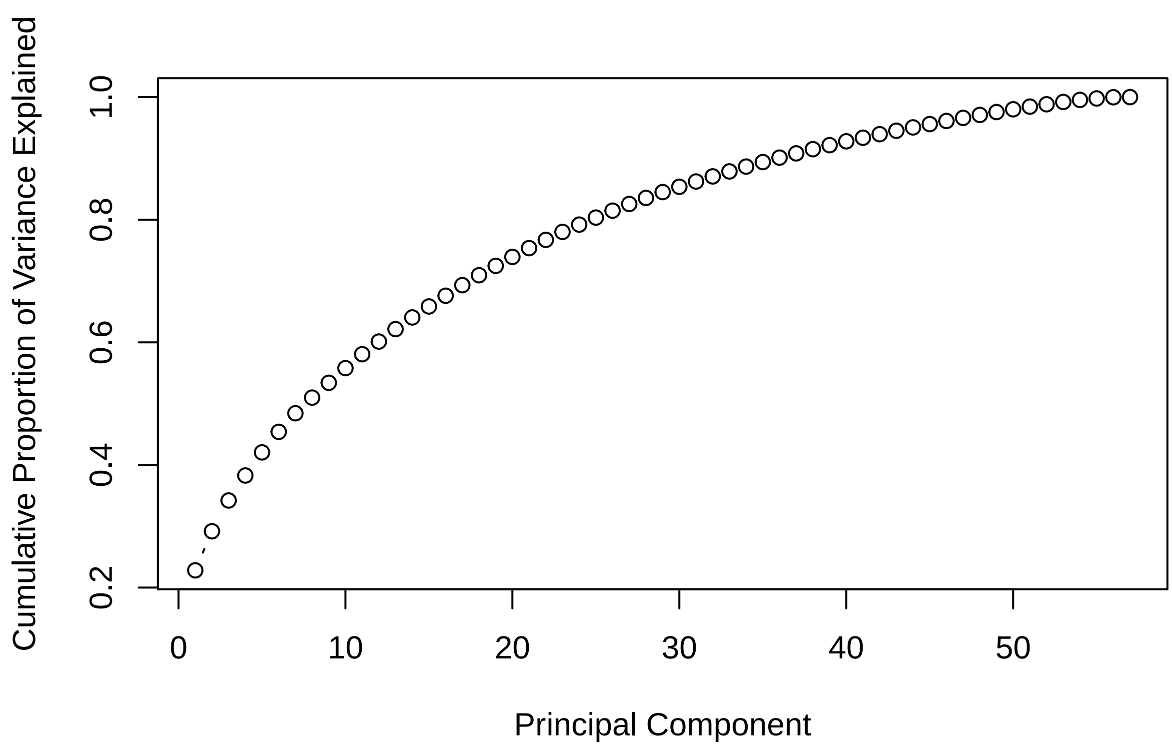
```r
# Cumulative PVE for all the principal components
sum(pve)
```

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

```r
plot(pve, xlab = "Principal Component",
          ylab = "Proportion of Variance Explained",
          type = "b")
```



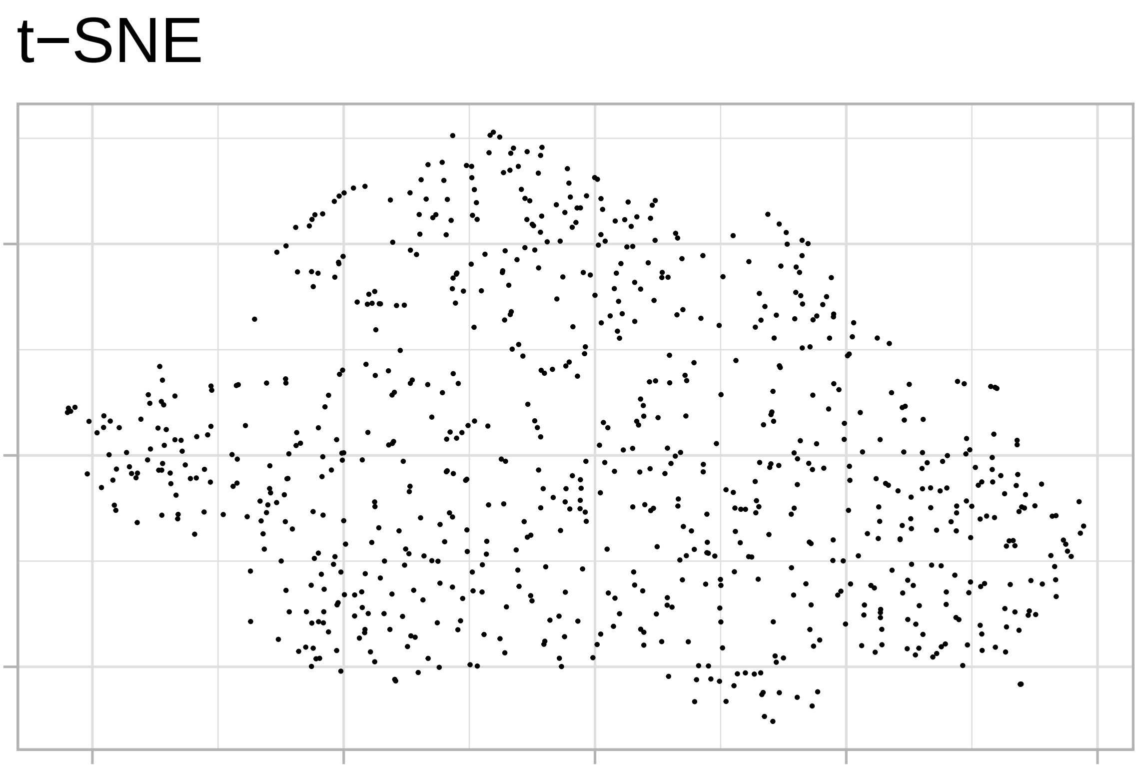

```r
plot(cumsum(pve), xlab = "Principal Component",
           ylab = "Cumulative Proportion of Variance Explained",
           type = "b")
```



As we see above, PC1 explains 22.8% of variances and PC2 explains 6.4% of variances. Using both the first and second principle component, only 29.2% of variances can be explained. The cumulative PVE is 1 which means all 57 principal components explain all variances.

8.

```
# tSNE and plot observations on the first two dimensions
library(Rtsne)
wiki.tsne <- Rtsne(wiki, dims=2, pca=TRUE, perplexity=30, max_iter=500)
d.tsne <- as.data.frame(wiki.tsne$Y)

ggplot(d.tsne, aes(x=V1, y=V2)) +
  geom_point(size=0.25) +
  guides(colour=guide_legend(override.aes=list(size=6))) +
  xlab("") + ylab("") +
  ggtitle("t-SNE") +
  theme_light(base_size=20) +
  theme(axis.text.x=element_blank(),
        axis.text.y=element_blank()) +
  scale_colour_brewer(palette = "Set2")
```



The graph above displays the 2-D dataset reduced by t-SNE. Because t-SNE dimension reduction method tries to minimize the difference between these conditional probabilities in higher-dimensional and lower-dimensional space. One of the limitations with linear dimensionality reduction algorithms, like PCA, is that they place dissimilar data points far apart in a lower dimension representation. As we can see on PCA graph, points are more spreading out than points displayed using t-SNE. Besides, as t-SNE measures the similarity of data points with multiple features, it naturally identify observed clusters in data and the data points on graph above are more clustered together.

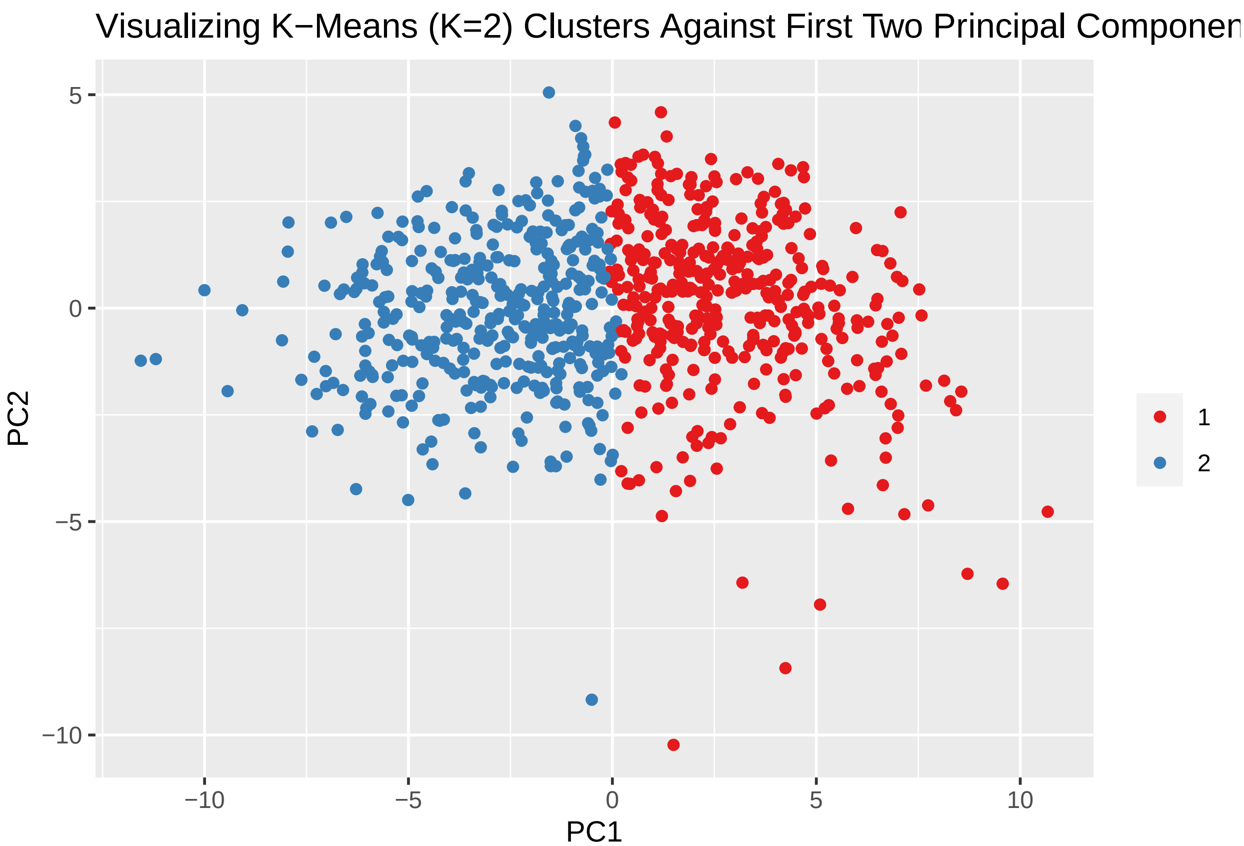
Clustering 9.

```
# Fortify() gets pca into usable format
library(ggfortify)
```

```r
pca.fortify <- fortify(wiki.pca)

# Scale the data
wiki.scaled <- scale(wiki)
```

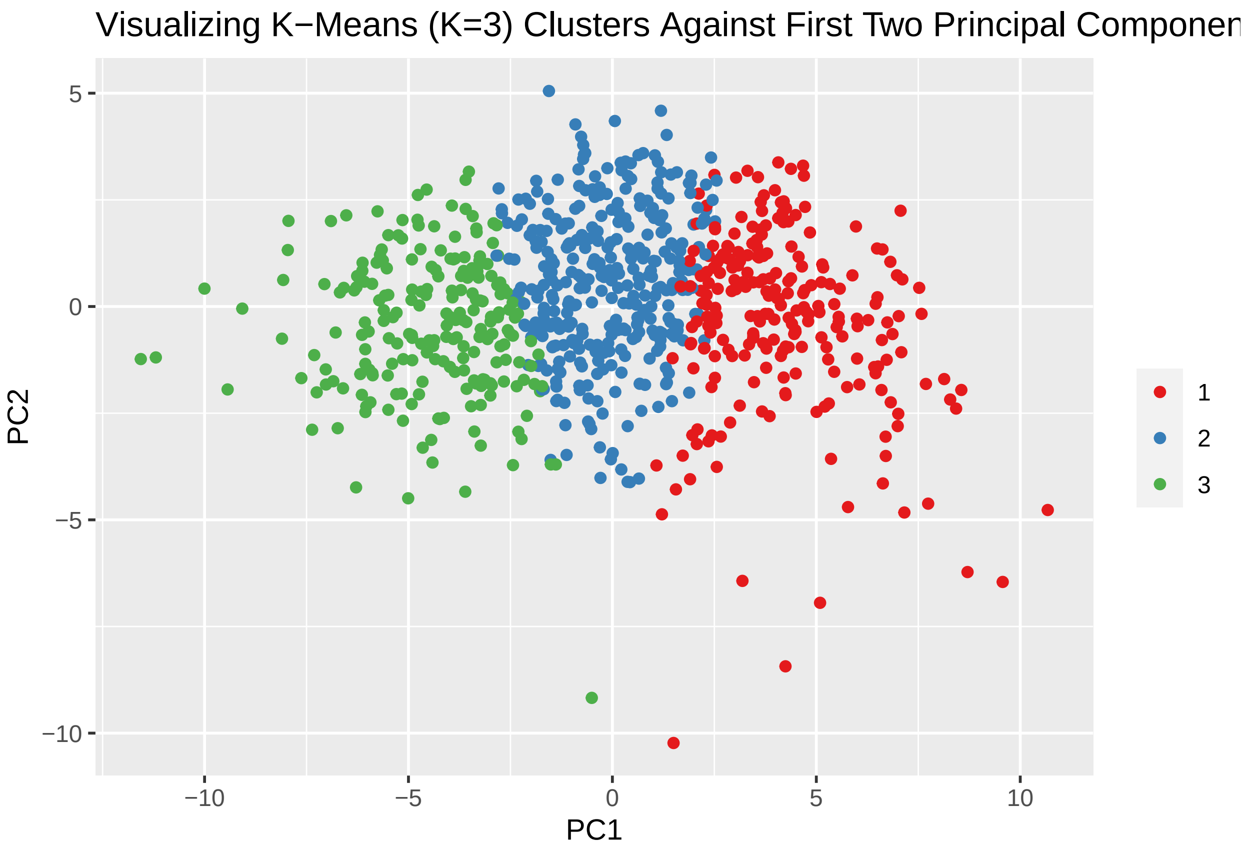
```r
# K-means clustering with K=2
wiki.k2 <- kmeans(wiki.scaled, 2, nstart=25, iter.max=50)
pca2.dat <- cbind(pca.fortify, group=wiki.k2$cluster)

# Script for plotting K=2
ggplot(pca2.dat) +
  geom_point(aes(x=PC1, y=PC2, col=factor(group), text=rownames(pca2.dat))) +
  labs(title = "Visualizing K-Means (K=2) Clusters Against First Two Principal Components") +
  scale_color_brewer(name="", palette = "Set1")
```

```
## Warning: Ignoring unknown aesthetics: text
```



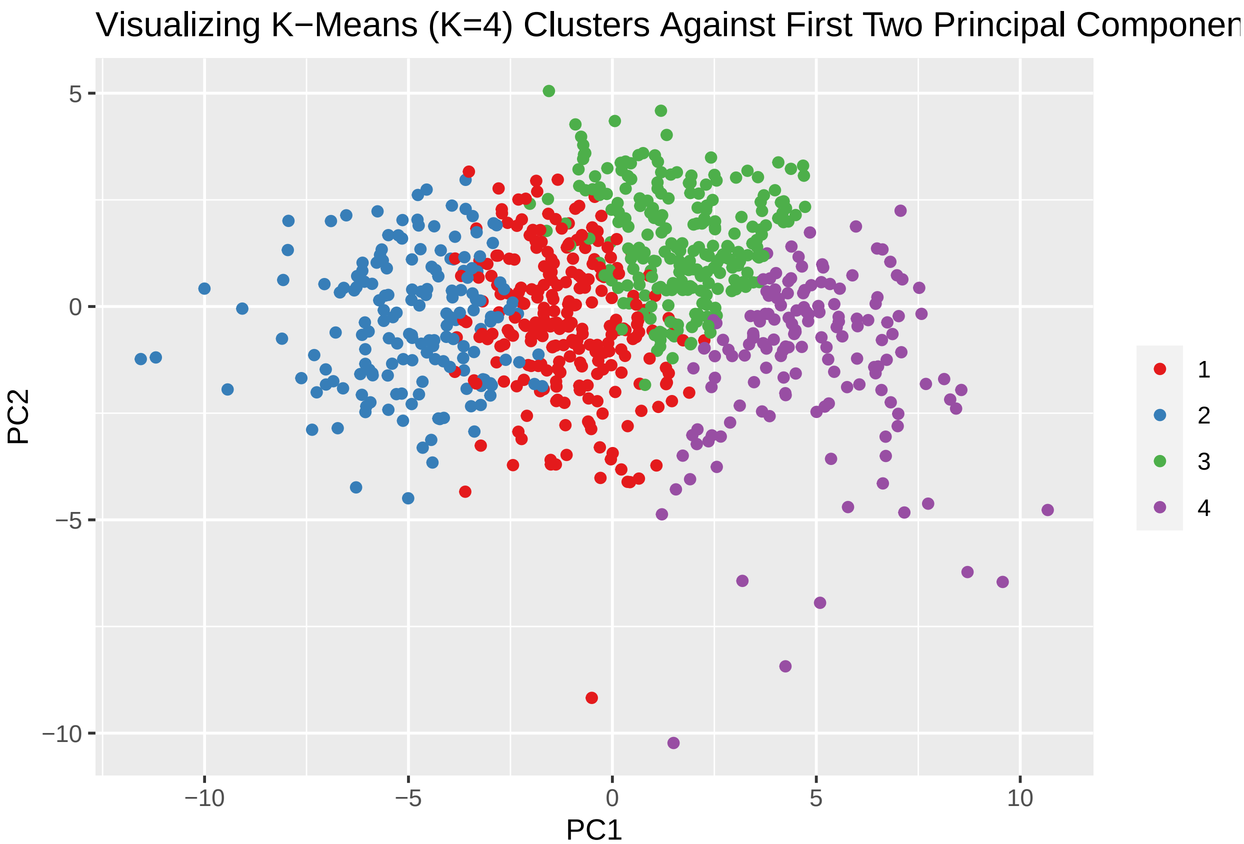

```r
# K-means clustering with K=3
wiki.k3 <- kmeans(wiki.scaled, 3, nstart=25, iter.max=50)
pca3.dat <- cbind(pca.fortify, group=wiki.k3$cluster)

# Script for plotting k=2
ggplot(pca3.dat) +
  geom_point(aes(x=PC1, y=PC2, col=factor(group), text=rownames(pca3.dat))) +
  labs(title = "Visualizing K-Means (K=3) Clusters Against First Two Principal Components") +
  scale_color_brewer(name="", palette = "Set1")
```

```
## Warning: Ignoring unknown aesthetics: text
```




```
# K-means clustering with K=4
wiki.k4 <- kmeans(wiki.scaled, 4, nstart=25, iter.max=50)
pca4.dat <- cbind(pca.fortify, group=wiki.k4$cluster)

# Script for plotting k=2
ggplot(pca4.dat) +
  geom_point(aes(x=PC1, y=PC2, col=factor(group), text=rownames(pca4.dat))) +
  labs(title = "Visualizing K-Means (K=4) Clusters Against First Two Principal Components") +
  scale_color_brewer(name="", palette = "Set1")
```

```
## Warning: Ignoring unknown aesthetics: text
```



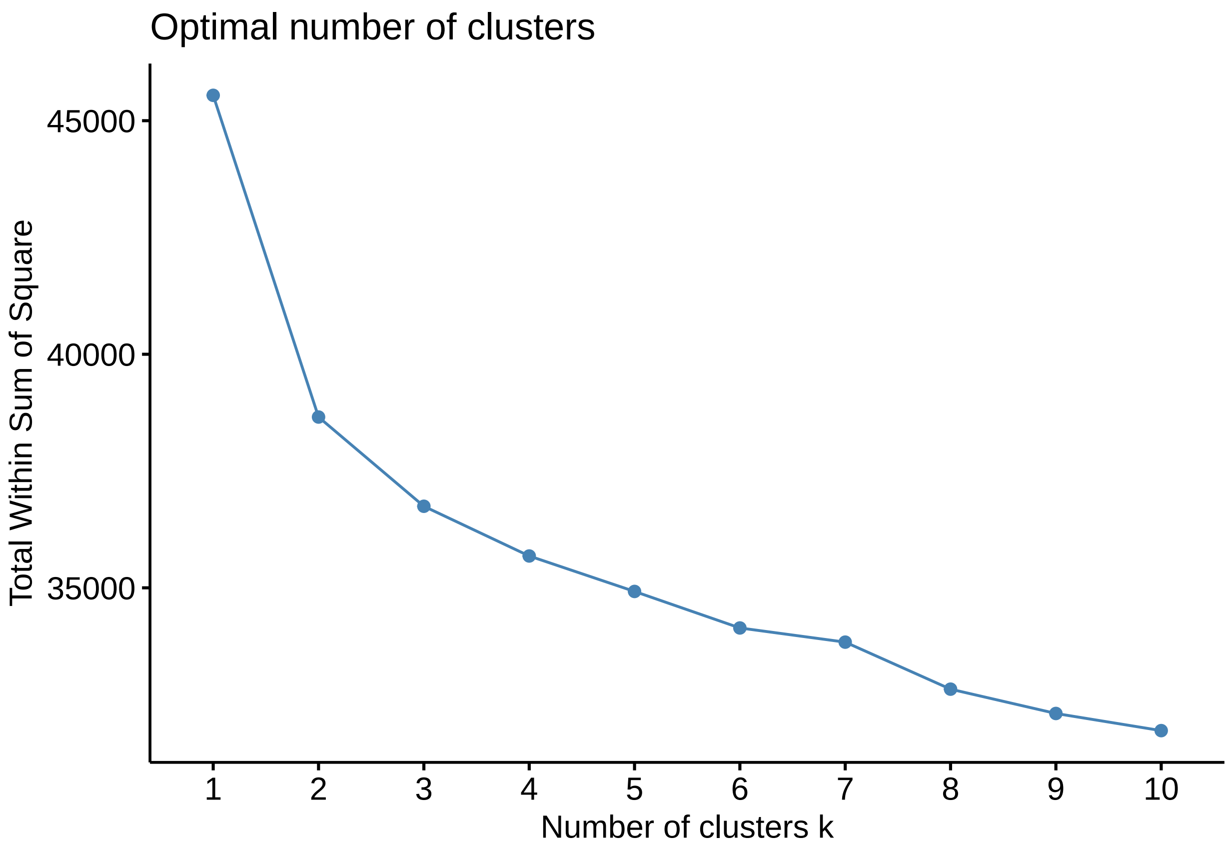
From graphs of k-means clustering data with reduced dimensions using PCA, we can observe a linear relation for all clusters boundaries. Because PCA isa type of linear transformation for dimentionally reduction, the clusters are projected and separated linearly. Comparing all three graphs, we can see the two clusters overlaps the least compared to the other two, which may indicate when K=2, the data was fitted the best.

10.

```
# Elbow Method
library(factoextra)
```

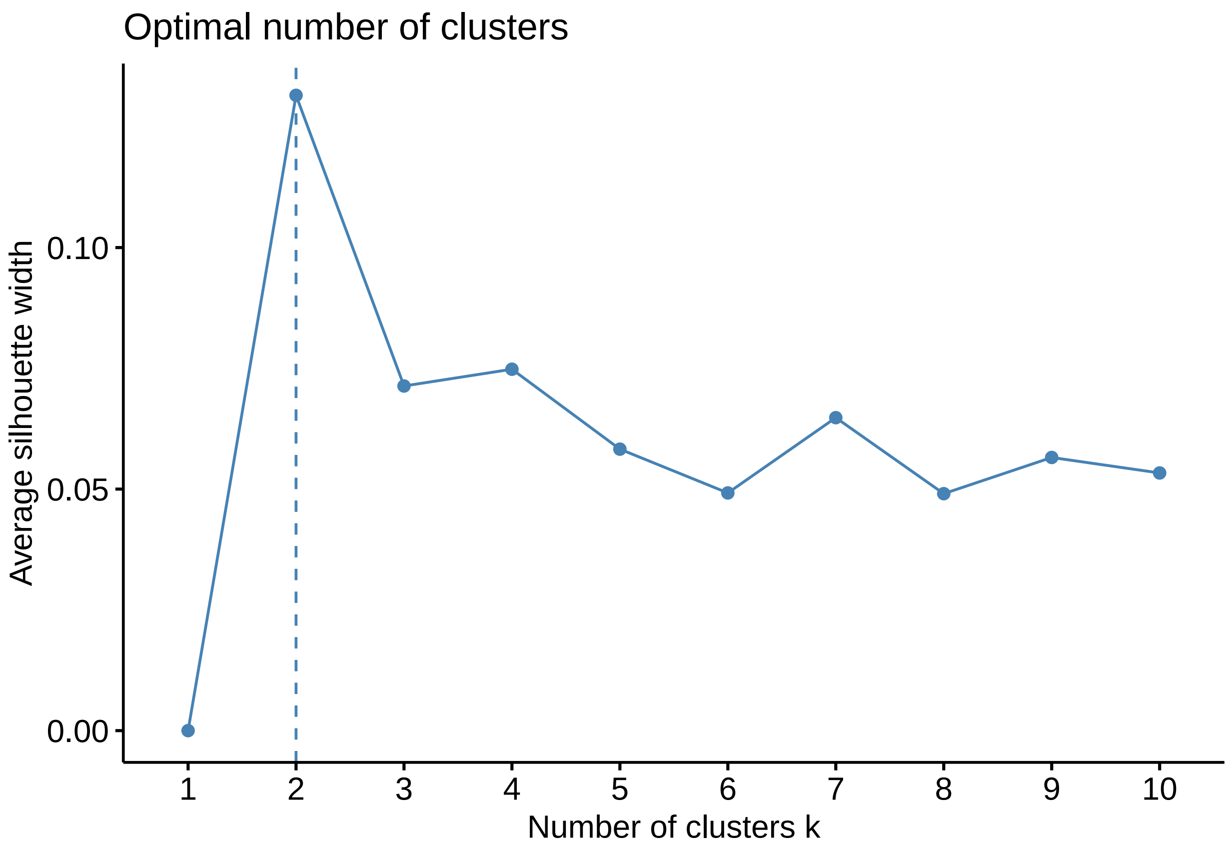
```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_nbclust(wiki.scaled, kmeans, method="wss")
```



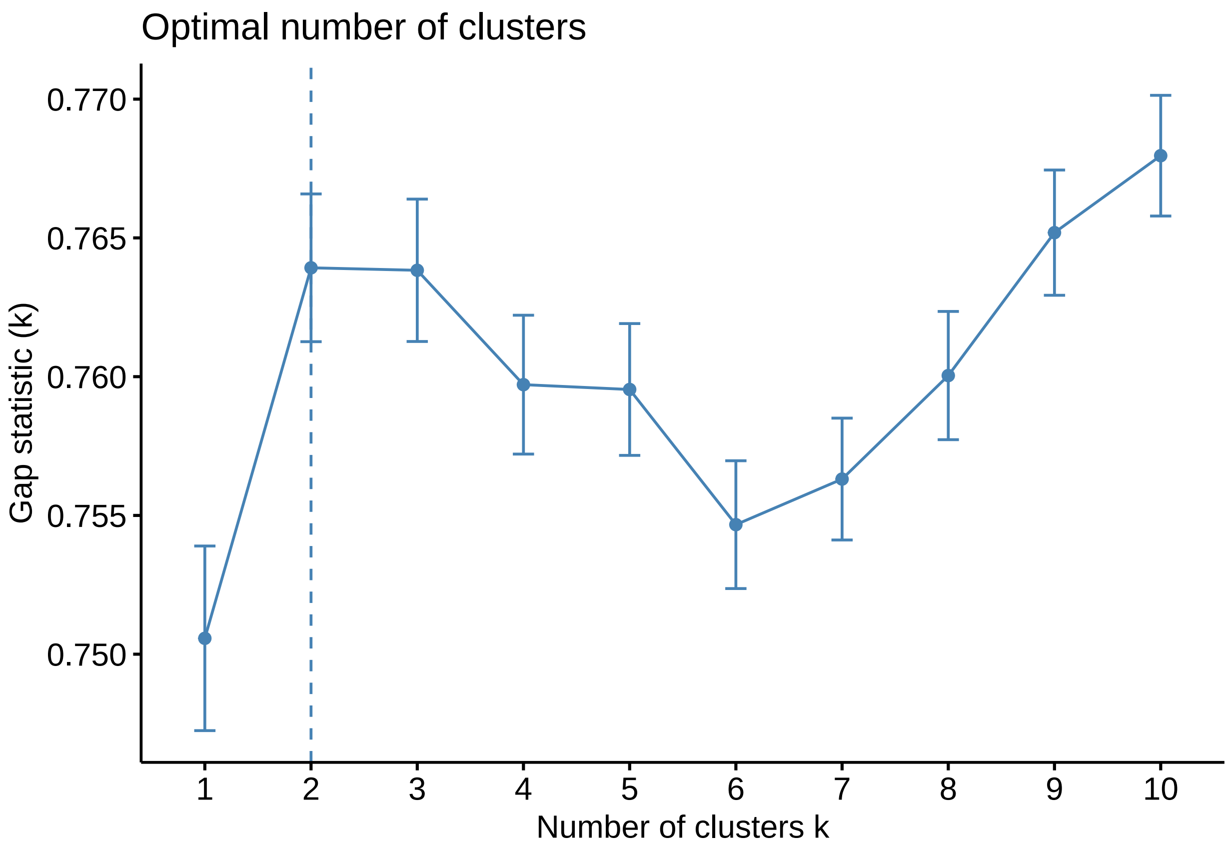
With the Elbow Method, we know the location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters. According the plot generated above, 2 or 3 seems to be the optimal number of clusters because it appears to be the bend in the knee.

```
# Silhouette Method
fviz_nbclust(wiki.scaled, kmeans, method="silhouette")
```



With Silhouette method, the optimal number of clusters k is the one that maximizes the average silhouette over a range of possible values for k. Thus, the graph show that 2 clusters maximize the average silhouette values with 4 clusters coming in as second optimal number of clusters.

```
# Gap Statistic Method
gap_stat <- clusGap(wiki.scaled, FUN=kmeans, nstart=25, iter.max=50,
                    K.max=10, B=50)
fviz_gap_stat(gap_stat)
```



Using Gap Statistic Method, the estimate of the optimal clusters will be the value that maximizes Gap(k). This means that the clustering structure is far away from the uniform distribution of points. From the graph above, 2 or 3 are the optimal number of clusters.

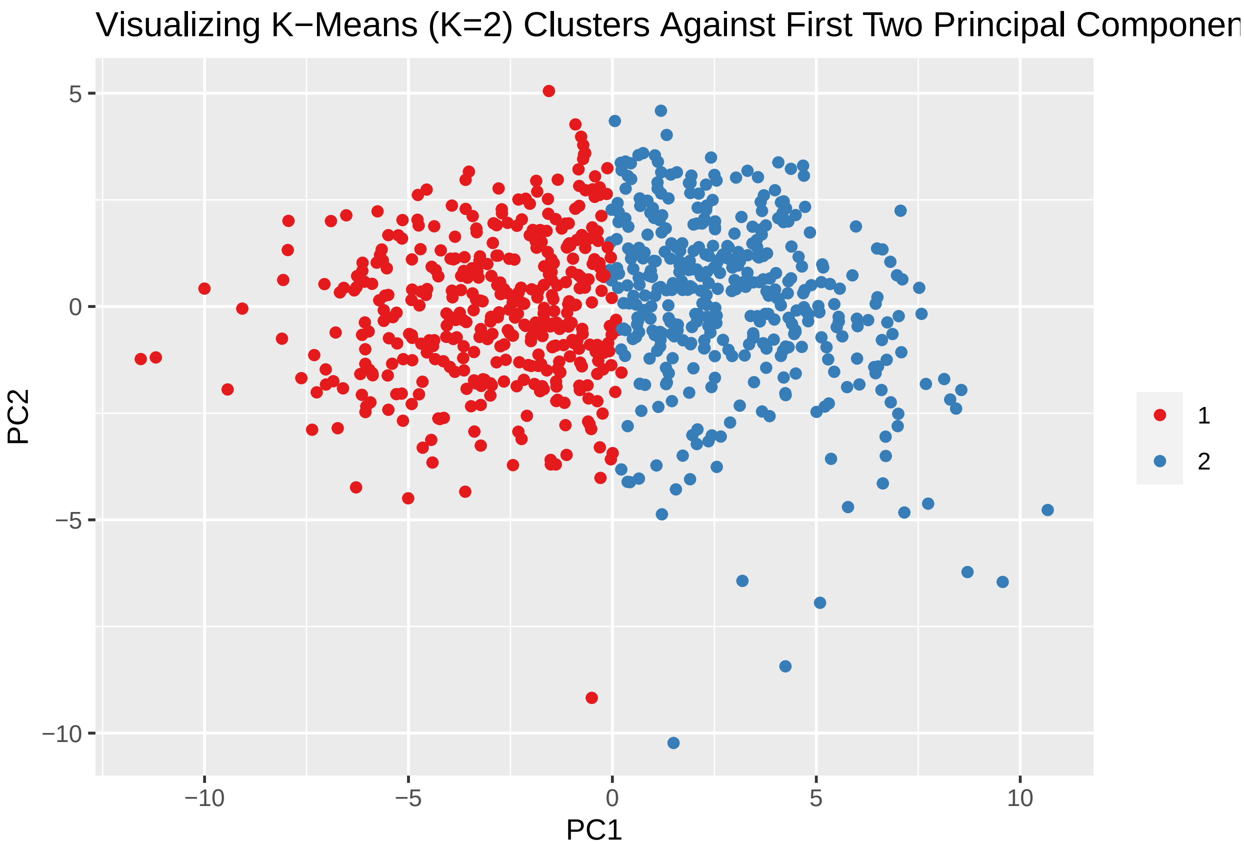
According to all three methods, I chose 2 as the optimal number of clusters.

11.

```
# K-means clustering with K=2
wiki.k2 <- kmeans(wiki.scaled, 2, nstart=25, iter.max=50)
pca.dat <- cbind(pca.fortify, group=wiki.k2$cluster)
tsne.dat <- cbind(wiki.tsne$Y, group=wiki.k2$cluster)
```

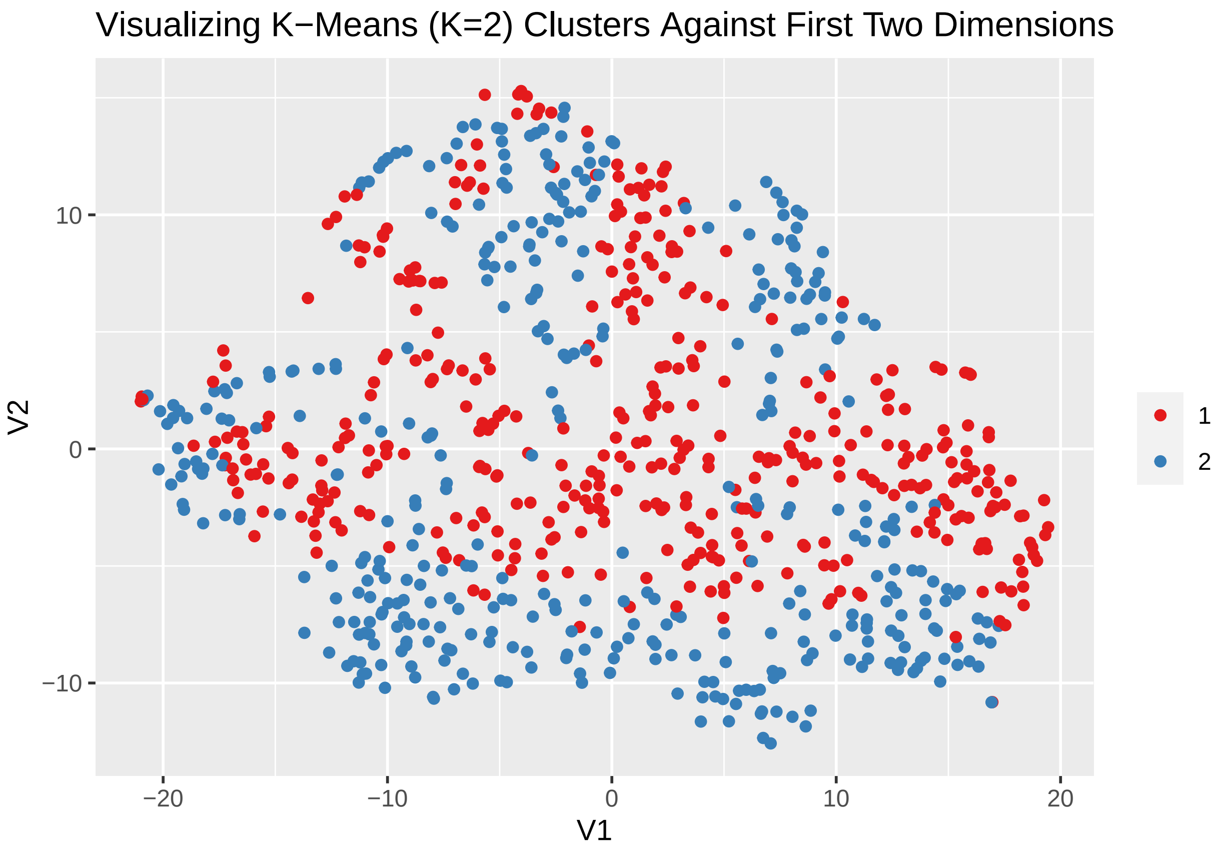
```
# Script for plotting K=2 using PCA
ggplot(pca.dat) +
  geom_point(aes(x=PC1, y=PC2, col=factor(group), text=rownames(pca.dat))) +
  labs(title = "Visualizing K-Means (K=2) Clusters Against First Two Principal Components") +
  scale_color_brewer(name="", palette = "Set1")
```

```
## Warning: Ignoring unknown aesthetics: text
```




```r
# Script for plotting K=2 using tSNE
ggplot(tsne.dat) +
  geom_point(aes(x=V1, y=V2, col=factor(group), text=rownames(tsne.dat))) +
  labs(title = "Visualizing K-Means (K=2) Clusters Against First Two Dimensions") +
  scale_color_brewer(name="", palette = "Set1")
```

```
## Warning: Ignoring unknown aesthetics: text
```



PCA and t-SNE are two different approaches for dimentional reduction. Because PCA is a type of linear transformation, PCA is limited to capture only the linear structure of the dataset. As we can see above, the two clusters are separated with a linear boundary. However, the t-SNE algorithm works in a very different way and focuses to preserve the local distances of the high-dimensional data in some mapping to low-dimensional data. By looking at the plot generated using t-SNE, we can see t-SNE helps make the cluster more accurate because it converts data into a 2-dimension space where dots are in a circular shape. Such a non-linear relation after performing k-means clustering is not captured using PCA.