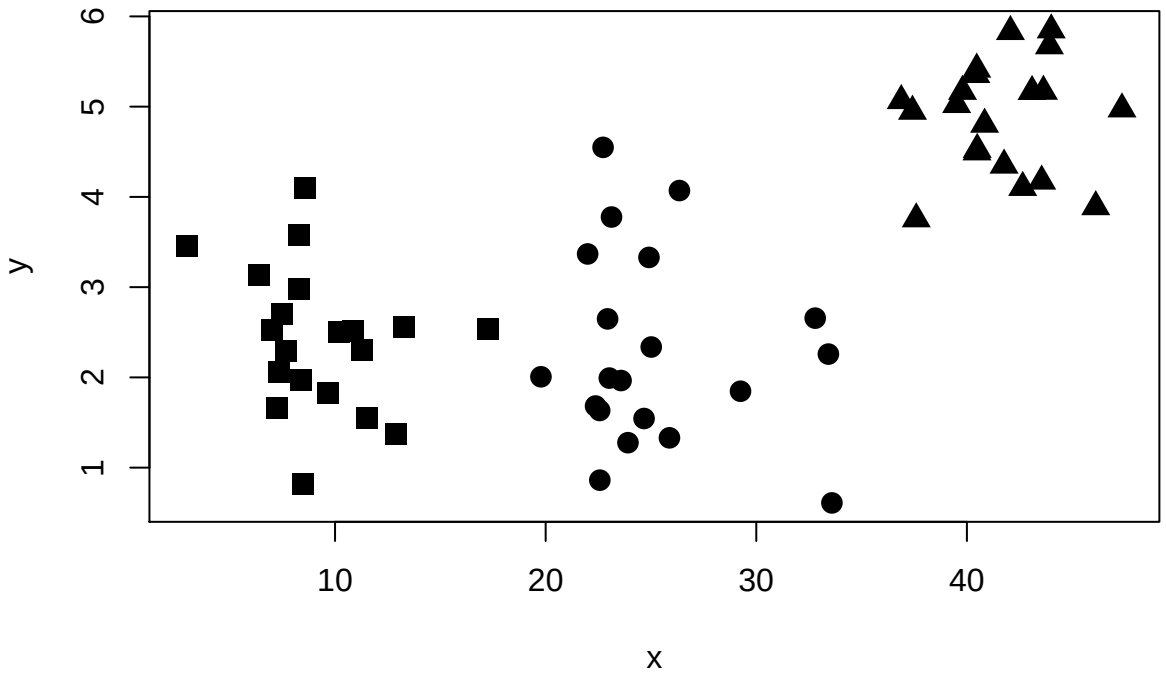
Exploring the Number of Clusters

Karen Mazidi

Create synthetic data

First we create some synthetic data using the rnorm() function. We create 3 distributions with centers (10, 3), (27, 2) and (41, 5). These are the "true" clusters but the regions overlap a little. We plot the unclustered data with different shapes for each distribution.

```
set.seed(1234)
x <- rep(0, 60)
y <- rep(0, 60)
x[1:20] <- rnorm(20, mean=10, sd=3)
y[1:20] <- rnorm(20, mean=3, sd=1)
x[21:40] <- rnorm(20, mean=27, sd=4)
y[21:40] <- rnorm(20, mean=2, sd=1)
x[41:60] <- rnorm(20, mean=41, sd=3)
y[41:60] <- rnorm(20, mean=5, sd=1)
# uncomment the next two lines to see what happens
# with a more uniform distribution
#x <- rnorm(60, mean=30, sd=10)
#y <- rnorm(60, mean=3, sd=2)
true <- c(rep(1,20), rep(2,20), rep(3,20))
plot(x, y, cex=1.5, pch=c(15, 16, 17)[true])
```



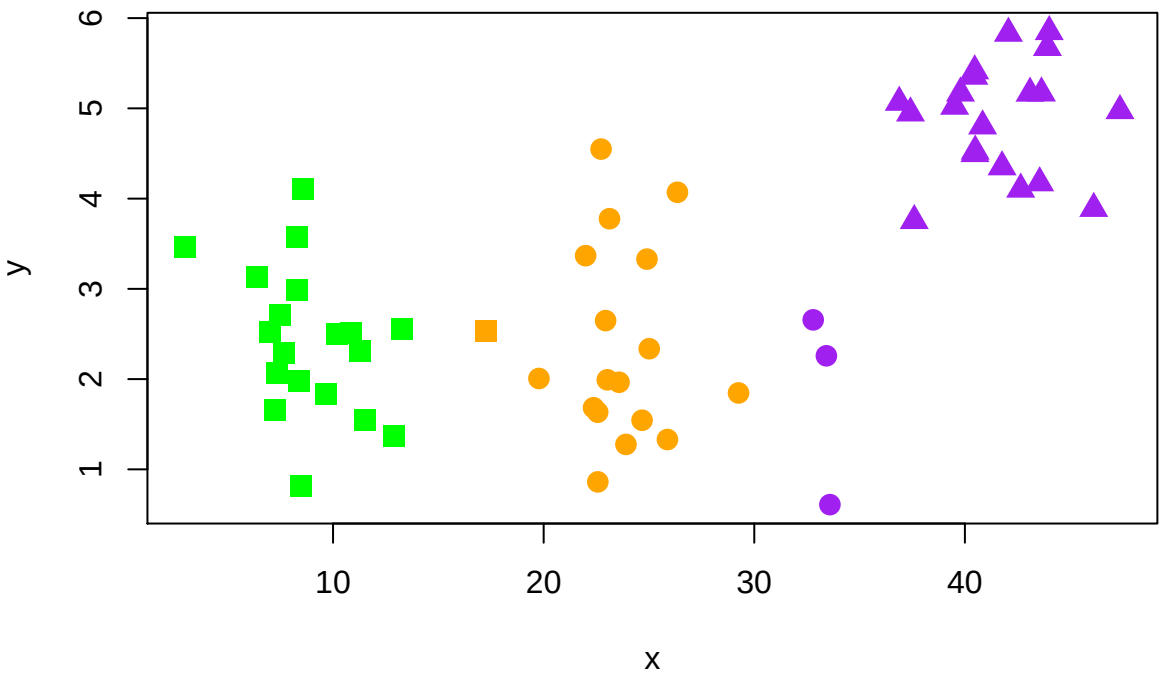
k-means: one iteration

Apply the k-means algoirthm with only one iteration and one start.

```
set.seed(1234)
df <- data.frame(cbind(x, y))
res <- kmeans(df, 3, iter.max=1, nstart=1 )
```

```
## Warning: did not converge in 1 iteration
```

```
plot(x, y, col=c("orange", "green", "purple")[res$cluster], cex=1.5, pch=c(15, 16, 17)[true])
```

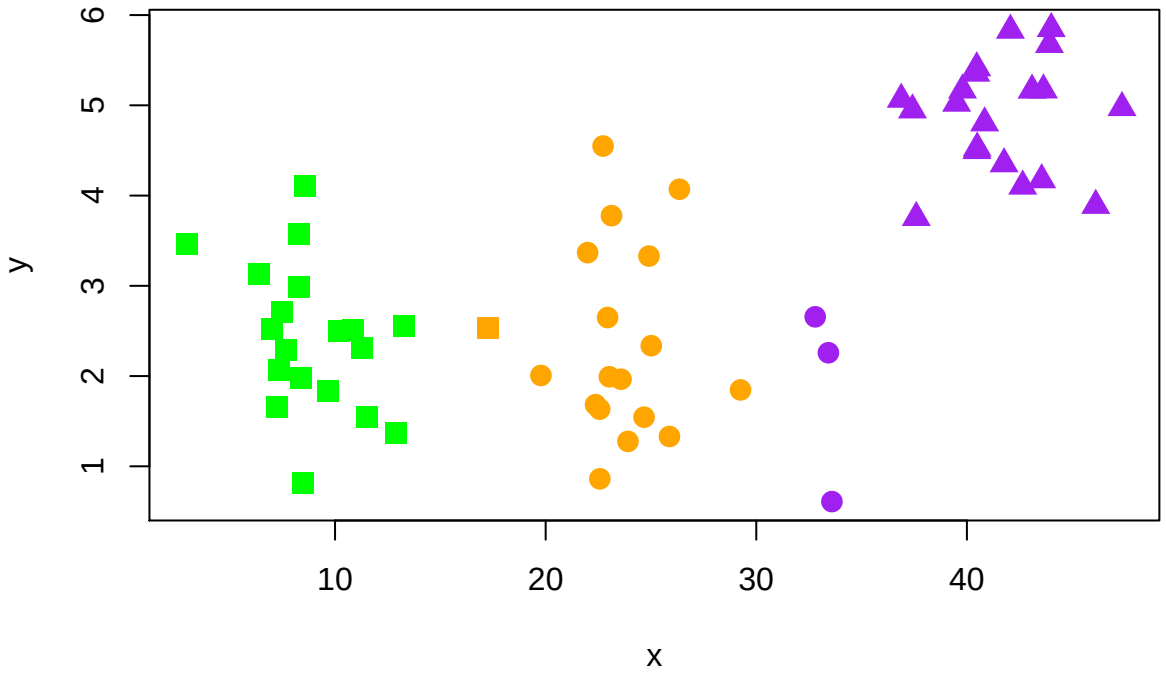


###

k-means: unlimited iterations

Although when we ran one iteration we got a warning message that it did not converge, we se no change when we let it run as many iterations as needed. Typing res2$iter at the console shows that it only ran 2 iterations.

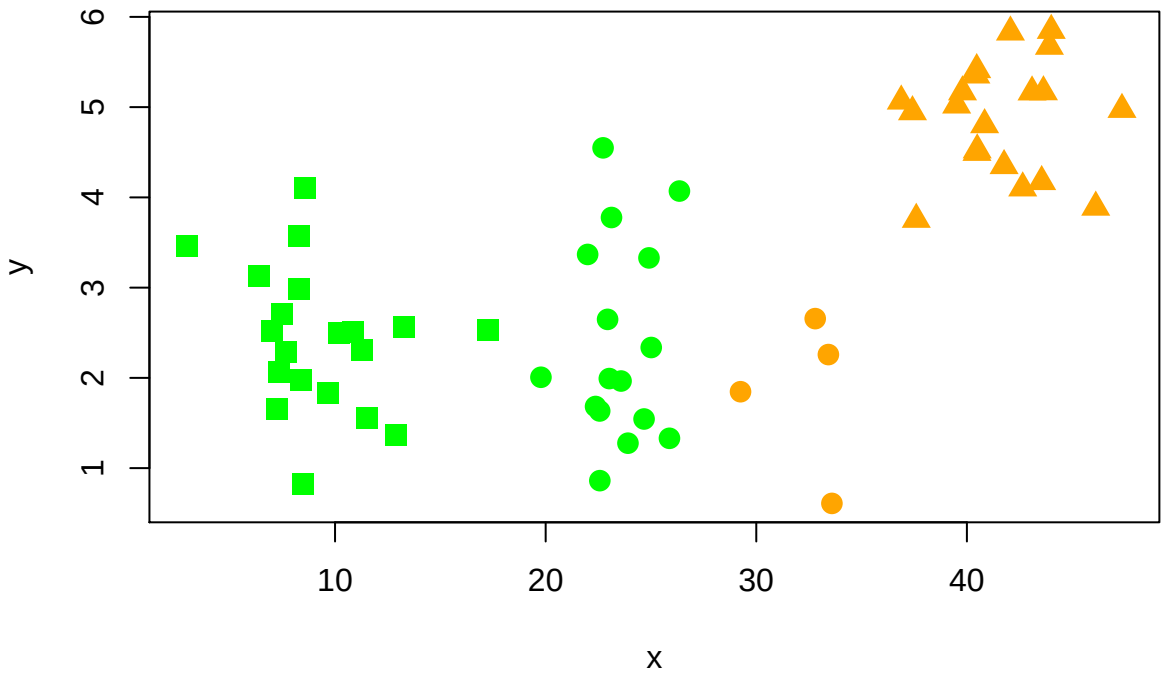
```
set.seed(1234)
res3 <- kmeans(df, 3,  nstart=1 )
plot(x, y, col=c("orange", "green", "purple")[res3$cluster], cex=1.5, pch=c(15, 16, 17)[true])
```



###

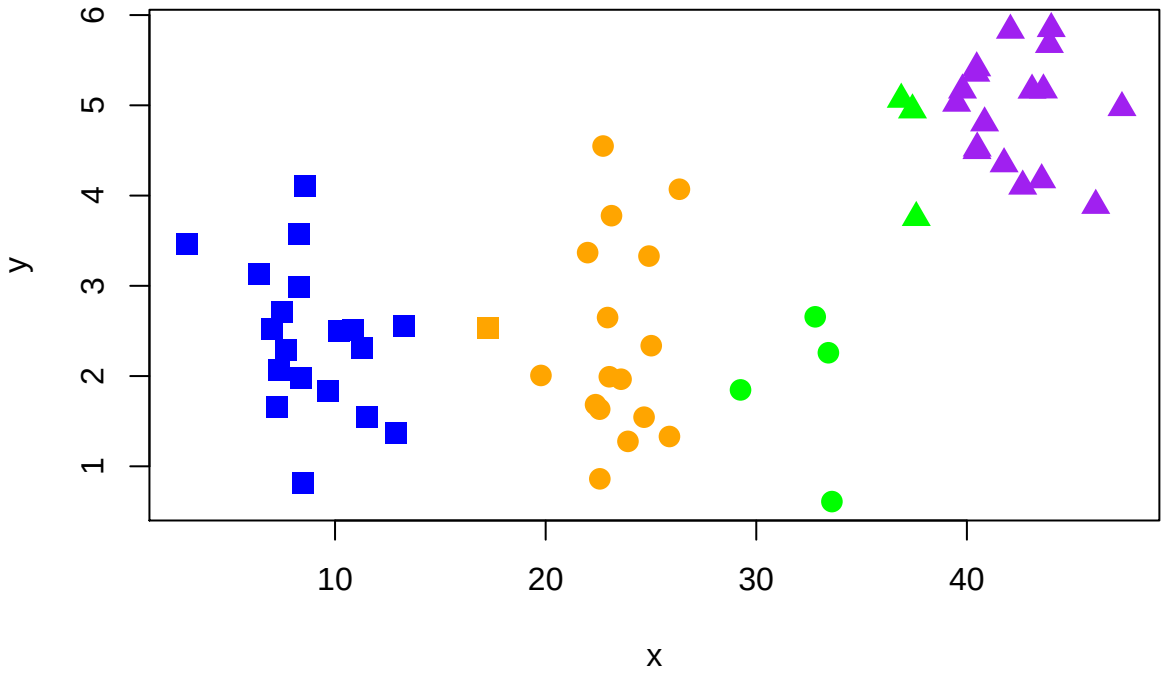
Try k=2

```
set.seed(1234)
res2 <- kmeans(df, 2,  nstart= 5)
plot(x, y, col=c("orange", "green", "purple", "blue")[res2$cluster], cex=1.5, pch=c(15, 16, 17)[true])
```



Try k=4

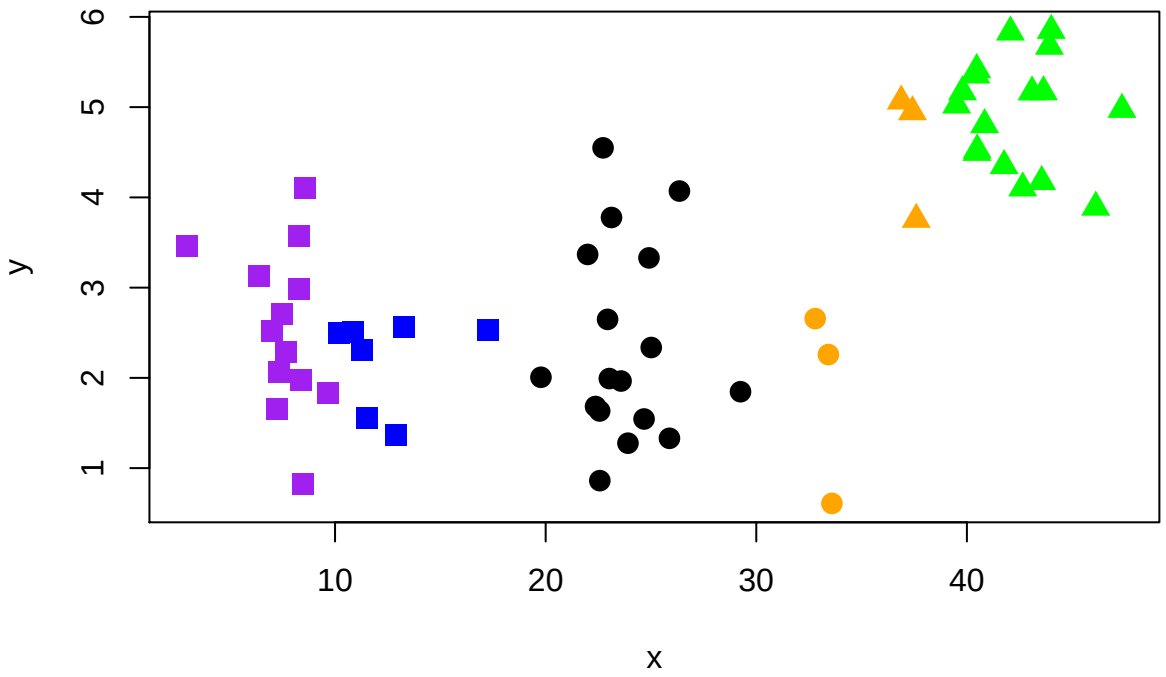
```
set.seed(1234)
res4 <- kmeans(df, 4,  nstart= 5)
plot(x, y, col=c("orange", "green", "purple", "blue")[res4$cluster], cex=1.5, pch=c(15, 16, 17)[true])
```



###

Try 5 clusters

```
set.seed(1234)
res5 <- kmeans(df, 5,  nstart= 5)
plot(x, y, col=c("orange", "green", "purple", "blue", "black")[res5$cluster], cex=1.5, pch=c(15, 16, 17
```



withinss

Our goal is to reduce within sum of squares, this means our clusters are more homogenous. Let's compare the withinss for k=2, k=4 and k=5.

It seems there is a dramatic drop from k=2 to k=3 then it gradually decreases. It makes sense that the larger the number of clusters, the smaller the withinss. After all, if k=n then withinss would be 0.

```
print(paste("k=2: ", sum(res2$withinss)))
```

```
## [1] "k=2:  2530.74905628694"
```

```
print(paste("k=3: ", sum(res3$withinss)))
```

```
## [1] "k=3:  611.695448018061"
```

```
print(paste("k=4: ", sum(res4$withinss)))
```

```
## [1] "k=4:  373.0421592289"
```

```
print(paste("k=5: ", sum(res5$withinss)))
```

```
## [1] "k=5:  291.501191384612"
```

```
library(flexclust)
```

```
## Loading required package: grid
```

```
## Loading required package: lattice
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
km2_table <- table(true, res2$cluster)
print(paste('rand index for k=2', randIndex(km2_table, correct=FALSE)))
```

```
## [1] "rand index for k=2 0.737853107344633"
```

```
km3_table <- table(true, res3$cluster)
print(paste('rand index for k=3', randIndex(km3_table, correct=FALSE)))
```

```
## [1] "rand index for k=3 0.916949152542373"
```

```
km4_table <- table(true, res4$cluster)
print(paste('rand index for k=4', randIndex(km4_table, correct=FALSE)))
```

```
## [1] "rand index for k=4 0.908474576271186"
```

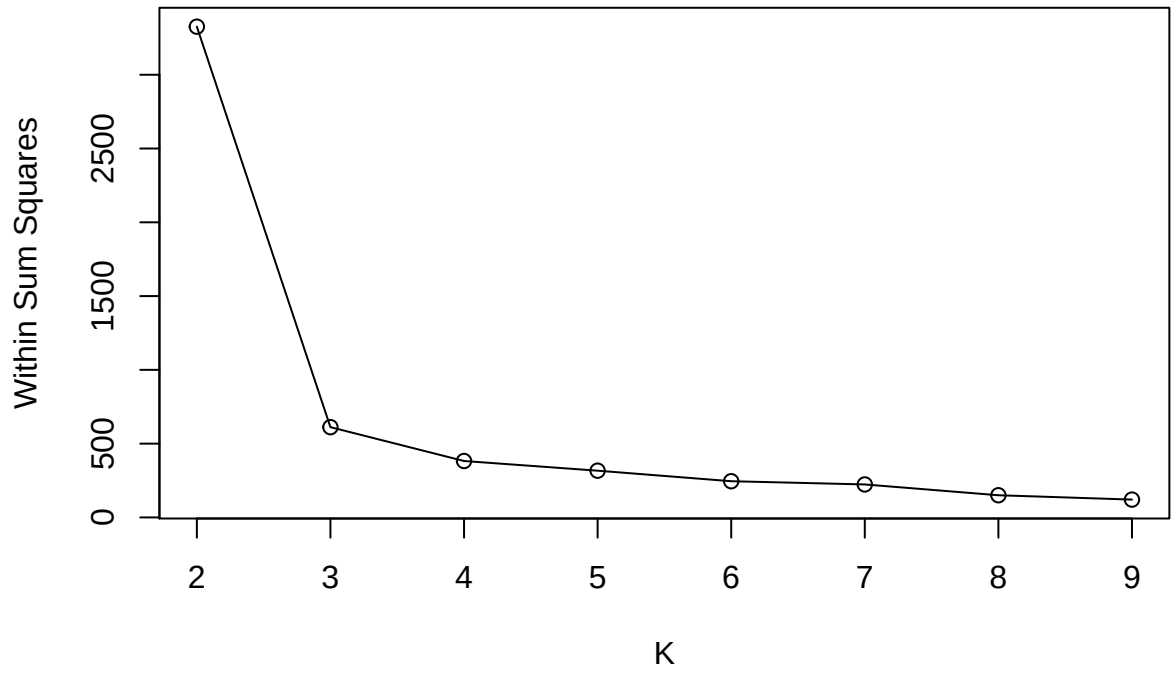
```
km5_table <- table(true, res5$cluster)
print(paste('rand index for k=5', randIndex(km5_table, correct=FALSE)))
```

```
## [1] "rand index for k=5 0.885875706214689"
```

Finding k with a function

We can write a function to randomly try different k values and plot the within sum of squares.

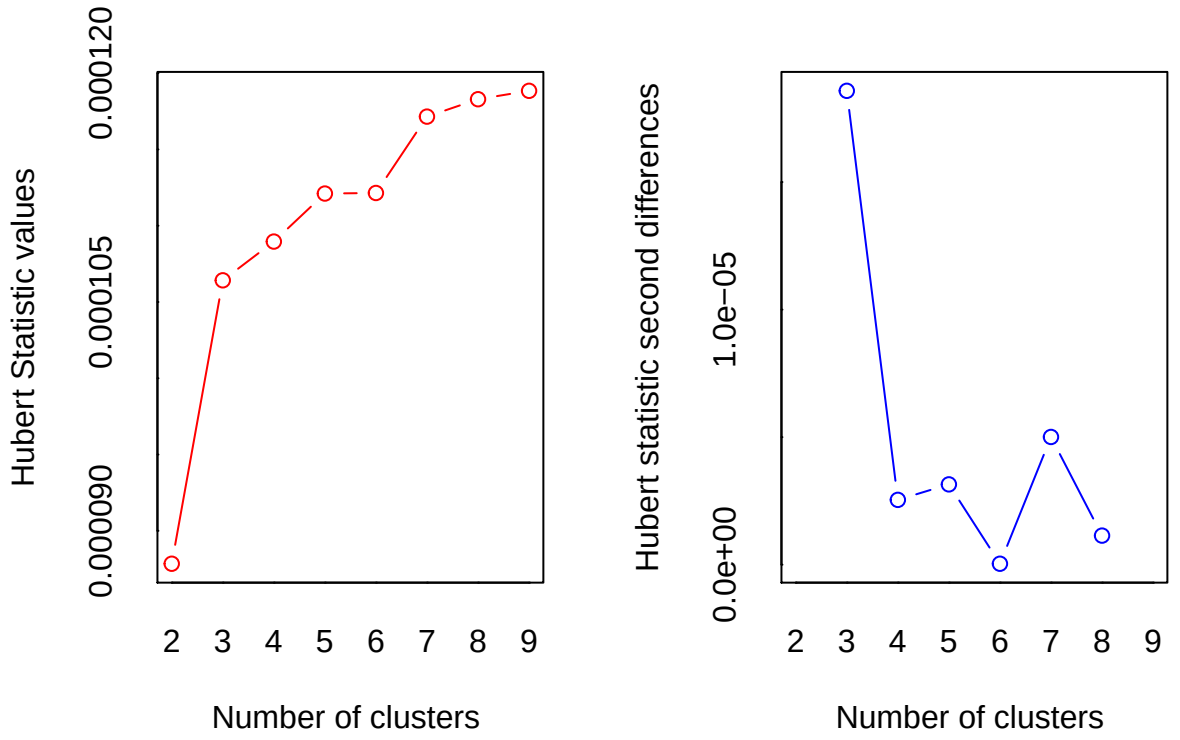
```
plot_withinss <- function(df, max_clusters){
  withinss <- rep(0, max_clusters-1)
  for (i in 2:max_clusters){
    set.seed(1234)
    withinss[i] <- sum(kmeans(df, i)$withinss)
  }
  plot(2:max_clusters, withinss[2:max_clusters], type="o", xlab="K", ylab="Within Sum Squares")
}
plot_withinss(df, 9)
```



NbClust()

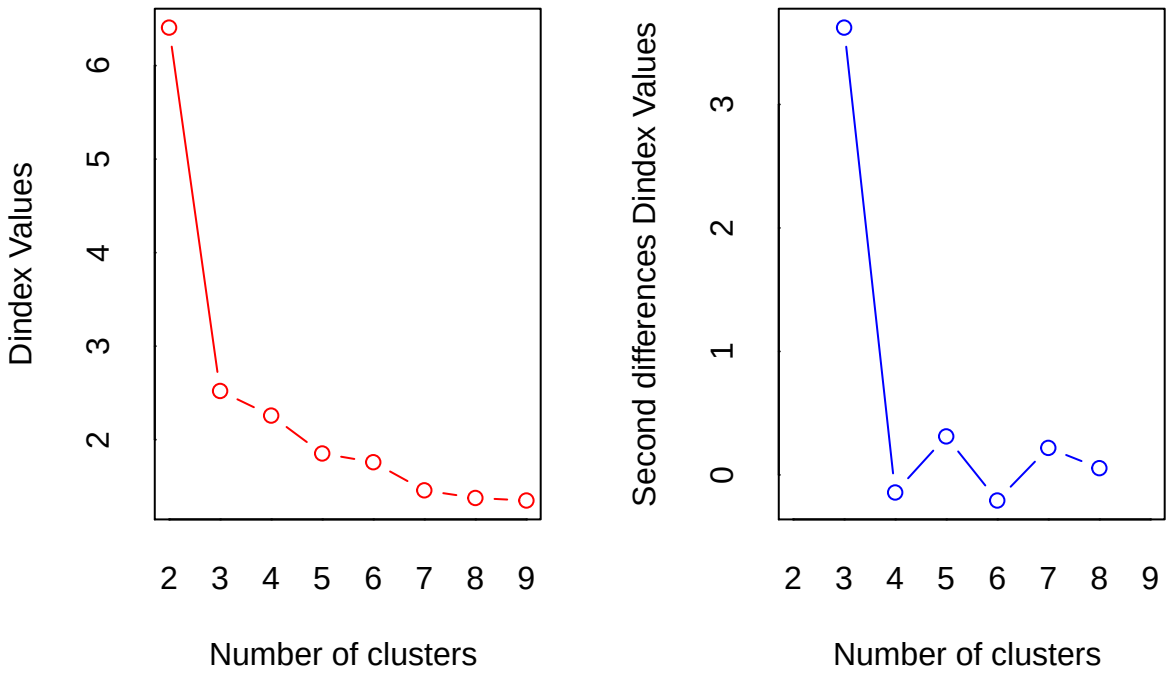
Next we try the NbClust() function to find the best number of clusters.

```
library(NbClust)
set.seed(1234)
nc <- NbClust(df, min.nc=2, max.nc=9, method="kmeans")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
```

```
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

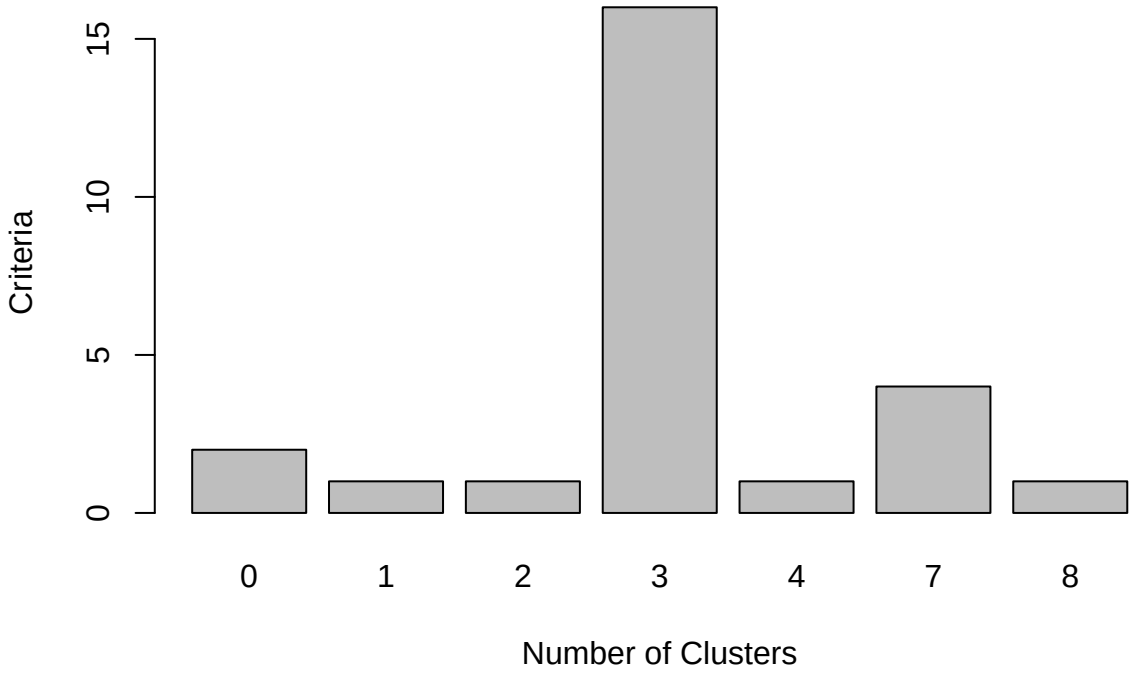



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 1 proposed 2 as the best number of clusters
## * 16 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 4 proposed 7 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
t <- table(nc$Best.n[1,])
t
```

```
##
##  0  1  2  3  4  7  8
##  2  1  1 16  1  4  1
```

```
barplot(t, xlab="Number of Clusters", ylab = "Criteria")
```



Plot for the book

```
par(mfrow=c(2,1))
plot(x, y, col=c("orange", "green", "purple", "blue")[res3$cluster], cex=1.5, pch=c(15, 16, 17)[true])
plot(x, y, col=c("orange", "green", "purple", "blue")[res4$cluster], cex=1.5, pch=c(15, 16, 17)[true])
```