Clustering

Rebecca C. Steorts

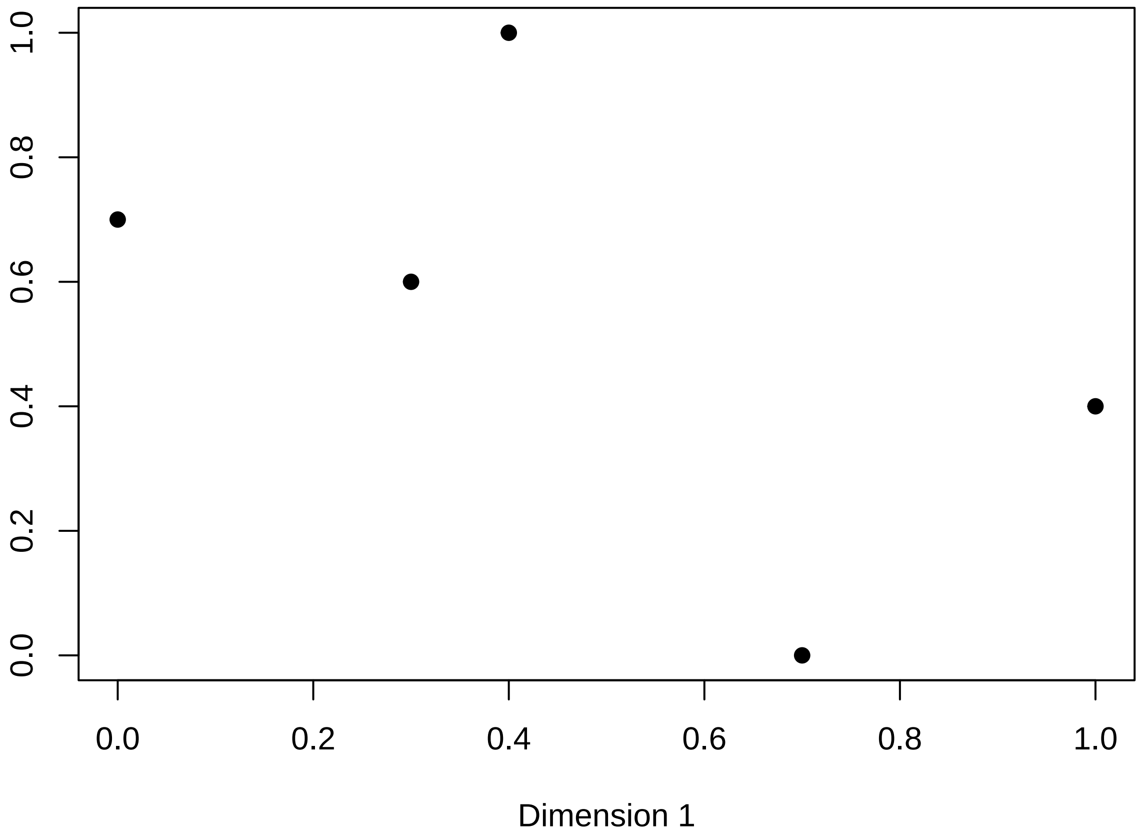
September 13, 2015

Clustering 1: K-means and K-mediods

These first set of notes correspond to Clustering 1: K-means and mediods.

We first give the replicable code for the Simple example that we went over in class.

```
# define a matrix x
x = cbind(c(0.7,1,0,0.3,0.4),c(0,0.4,0.7,0.6,1))
# We then plot it in two dimensions so that possible
# clusterings can be seen
par(mar=c(4,4,1,1))
plot(x,xlab="Dimension 1",ylab="",pch=19)
```



We then for the red and blue clusterings, calculate the within-cluster scatter, W.

```
# m is the matrix in the simple example, where we have
# taken the euclidean distance, squared
```

```
m = as.matrix(dist(x,diag=TRUE,upper=TRUE)^2)
for (i in 1:5) {
  cat(sprintf("%i ",i))
for (j in 1:5) {
  if (m[i,j]!=0)
  cat(sprintf("& %0.2f ", m[i,j]))
  else
    cat(sprintf("& 0 "))
}
cat(sprintf("\\\\\n\\hline\n"))
}
```

```
## 1 & 0 & 0.25 & 0.98 & 0.52 & 1.09 \\
## \hline
## 2 & 0.25 & 0 & 1.09 & 0.53 & 0.72 \\
## \hline
## 3 & 0.98 & 1.09 & 0 & 0.10 & 0.25 \\
## \hline
## 4 & 0.52 & 0.53 & 0.10 & 0 & 0.17 \\
## \hline
## 5 & 1.09 & 0.72 & 0.25 & 0.17 & 0 \\
## \hline
```

```
  print(m)
```

```
##      1    2    3    4    5
## 1 0.00 0.25 0.98 0.52 1.09
## 2 0.25 0.00 1.09 0.53 0.72
## 3 0.98 1.09 0.00 0.10 0.25
## 4 0.52 0.53 0.10 0.00 0.17
## 5 1.09 0.72 0.25 0.17 0.00
```

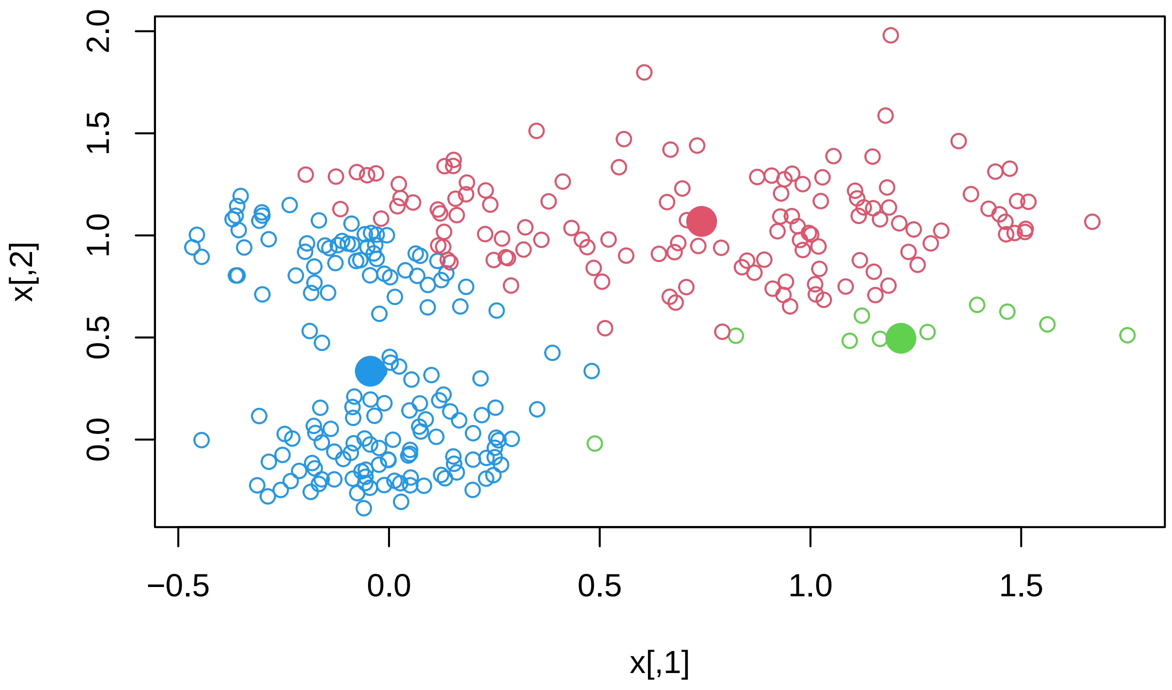
```
## Inputs: N,K (numbers of data points N
## Assumed number of clusters K)
## Output: total number of possible representation based
## on N data points and K clusters
totalPossibleRepresentation = function(N,K) {
  k = 1:K
  return(1/factorial(K) * sum((-1)^(K-k) * choose(K,k) * k^N))
}
```

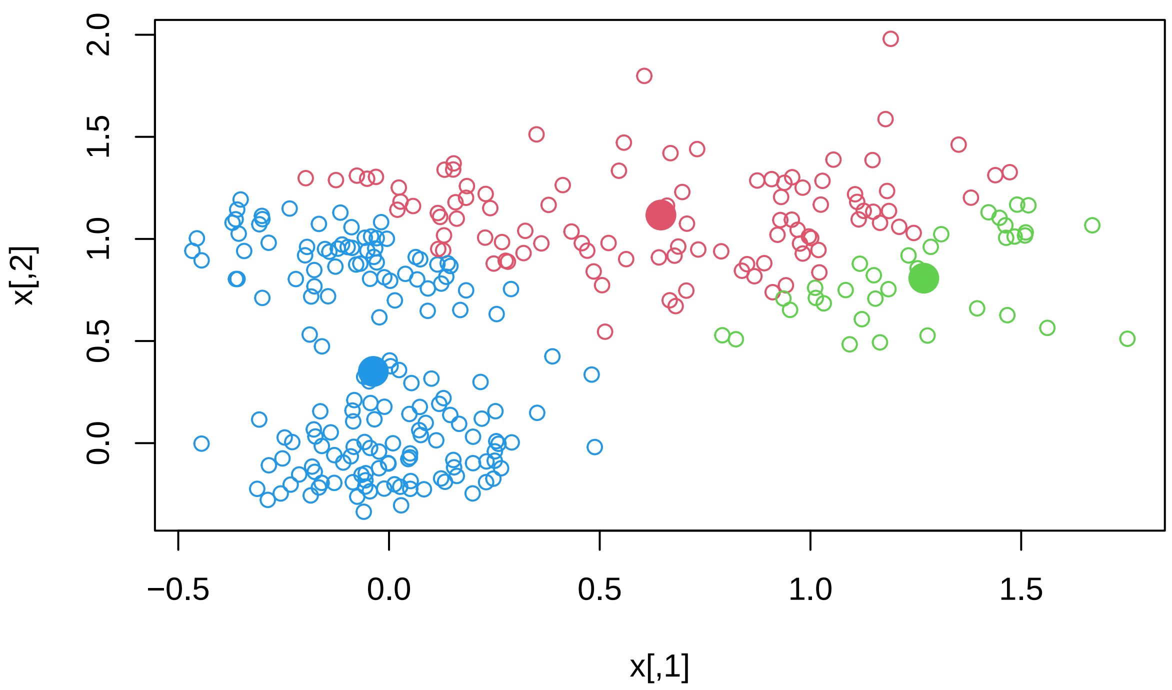
Add some text here

```
## kclust.R performs k-means or k-mediods clustering
## It is well documented.
source("kclust.R")
# We set a seed such that the results will always be the same.
set.seed(0)
# Create some random normal data (matrix form)
x = rbind(matrix(rnorm(100*2,sd=0.2),ncol=2),
scale(matrix(rnorm(100*2,sd=0.3),ncol=2),cent=-c(1,1),scal=F),
scale(matrix(rnorm(100*2,sd=0.2),ncol=2),cent=-c(0,1),scal=F))
```

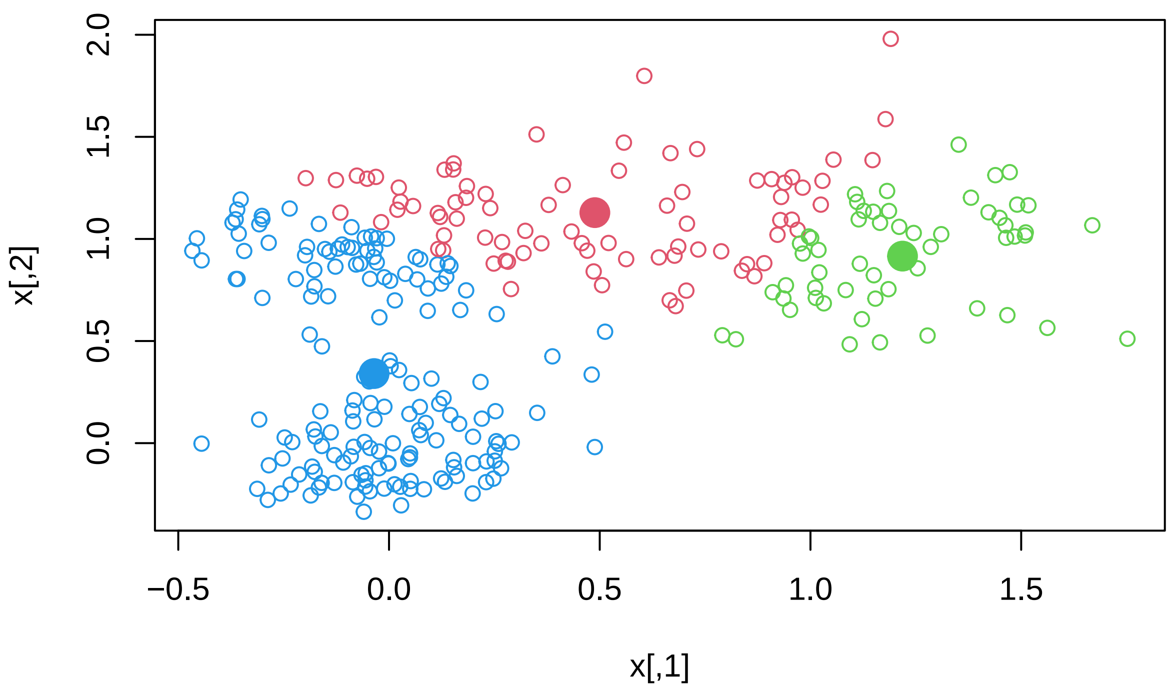
We first write some code that produces the k-means algorithm (visually) for every iteration. Notice how many plots this produces.

```r
# Apply kclust and plot the results until "convergence"
k = 3
cent.init = rbind(c(0.5,1),c(1,0),c(0,0.5))
km = kclust(x,centers=cent.init,alg="kmeans")
for (i in 1:km$iter) {
plot(x,col=km$cluster.history[i,]+1)
  points(avgfromclust(x,km$cluster.history[i,],k),col=2:4,pch=19,cex=2)
}
```

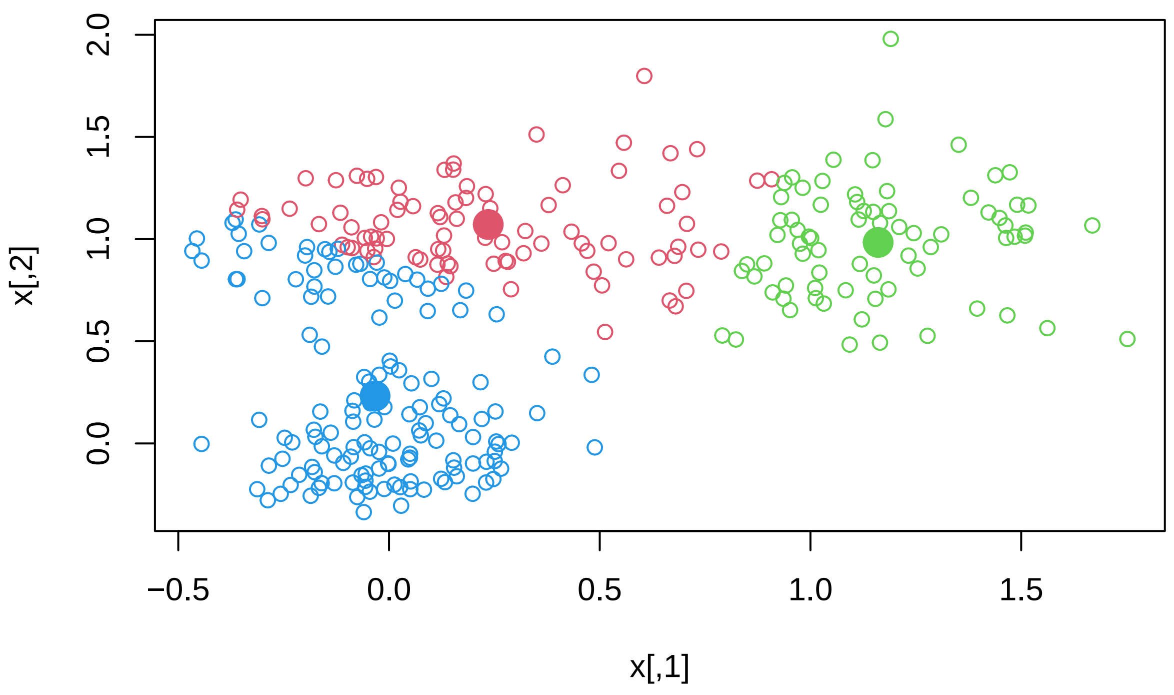


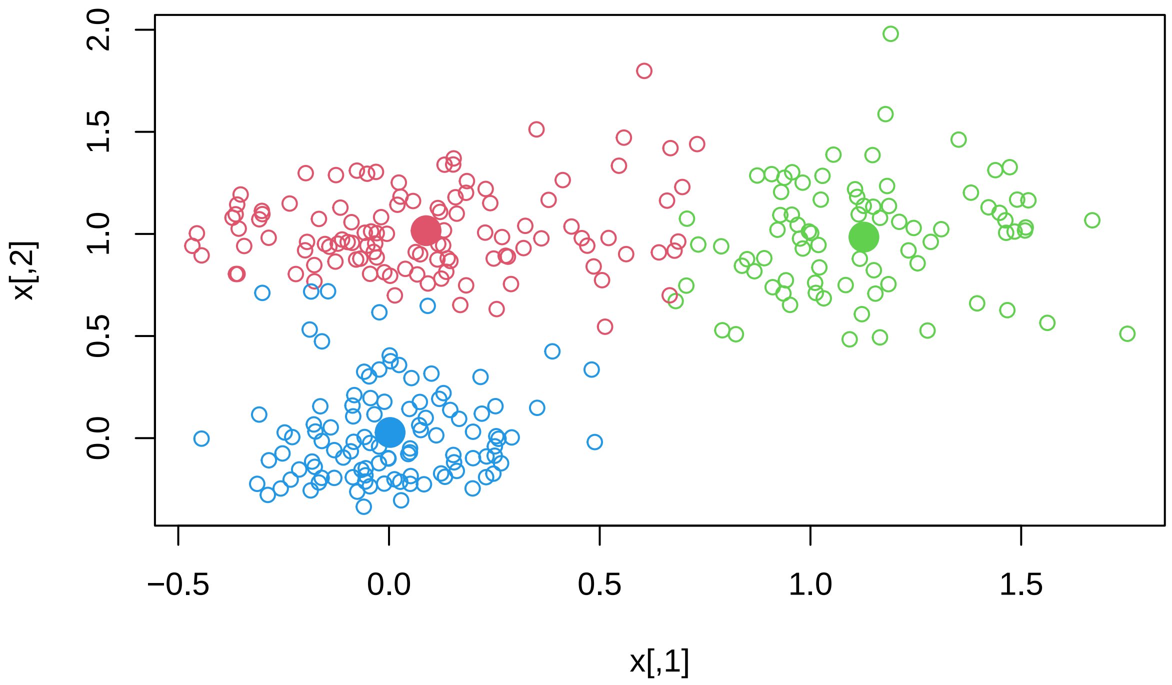


2.0
1.5
1.0
0.5
0.0
x[,2]
−0.5
0.0
0.5
1.0
1.5
x[,1]

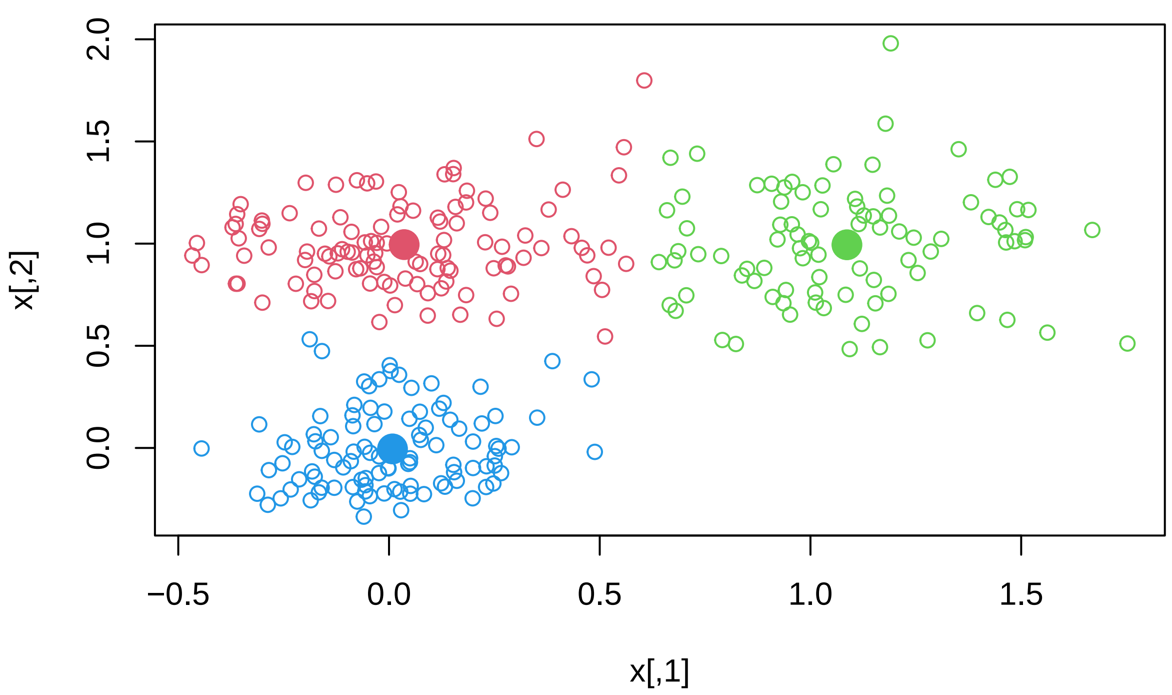


2.0
1.5
1.0
0.5
0.0
x[,2]
−0.5
0.0
0.5
1.0
1.5
x[,1]

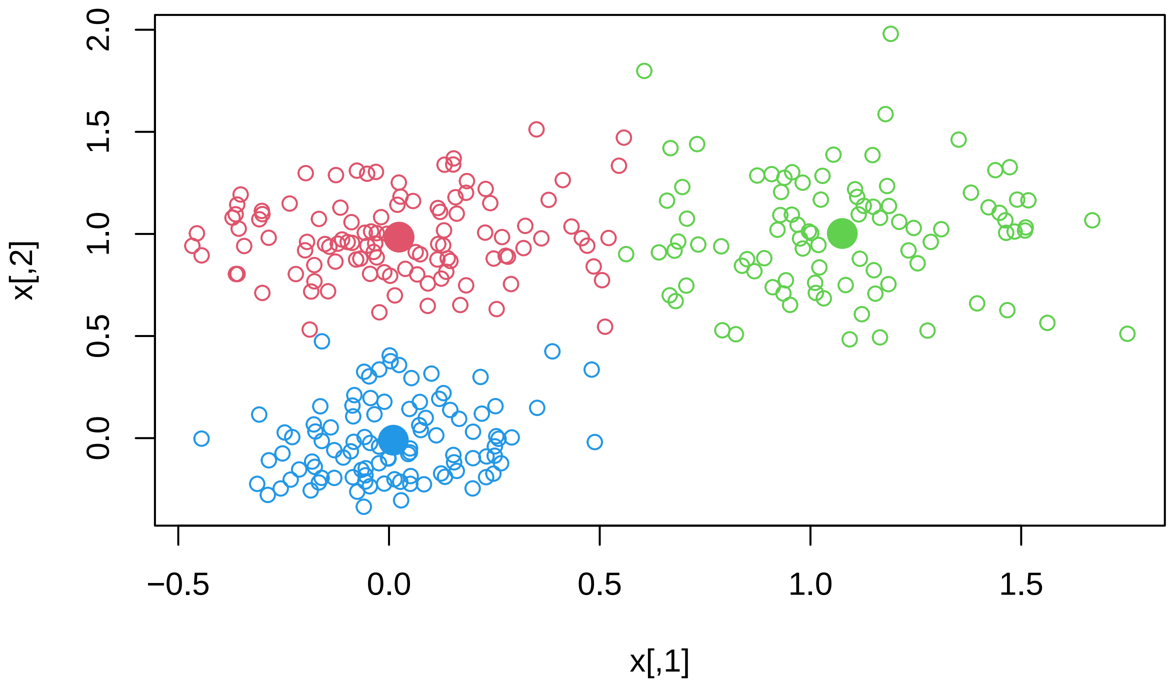

x[,2]
x[,1]
−0.5
0.0
0.5
1.0
1.5
2.0



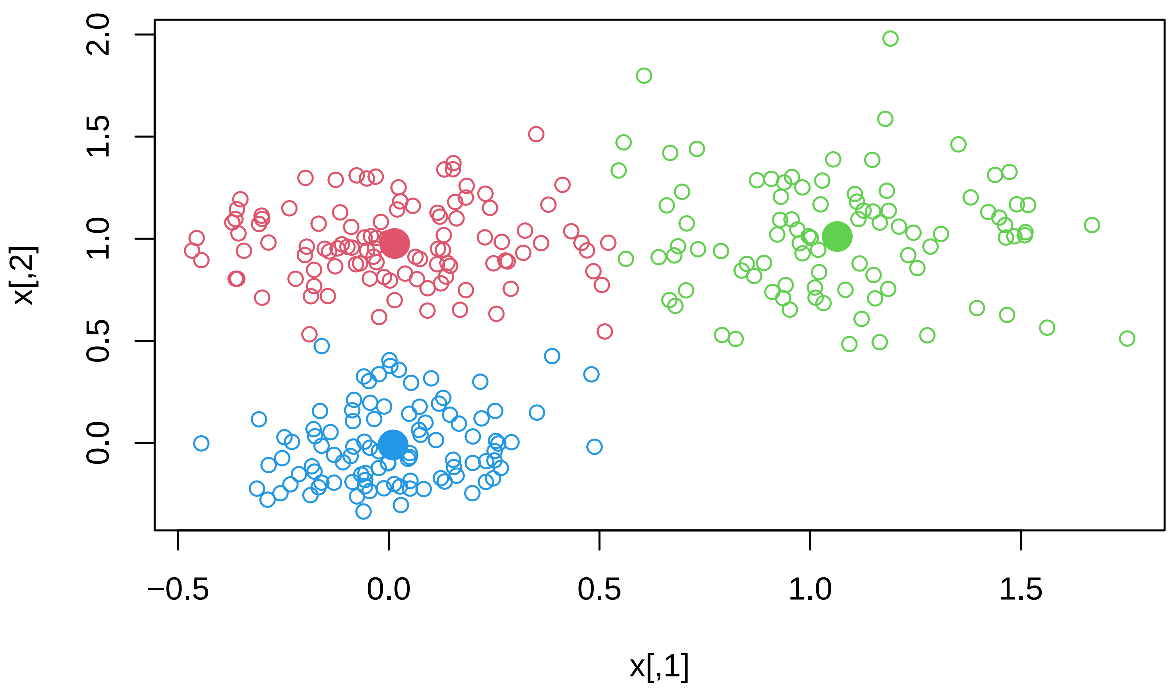
2.0
1.5
1.0
0.5
0.0
x[,2]
−0.5
0.0
0.5
1.0
1.5
x[,1]



x[,2]
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5
x[,1]

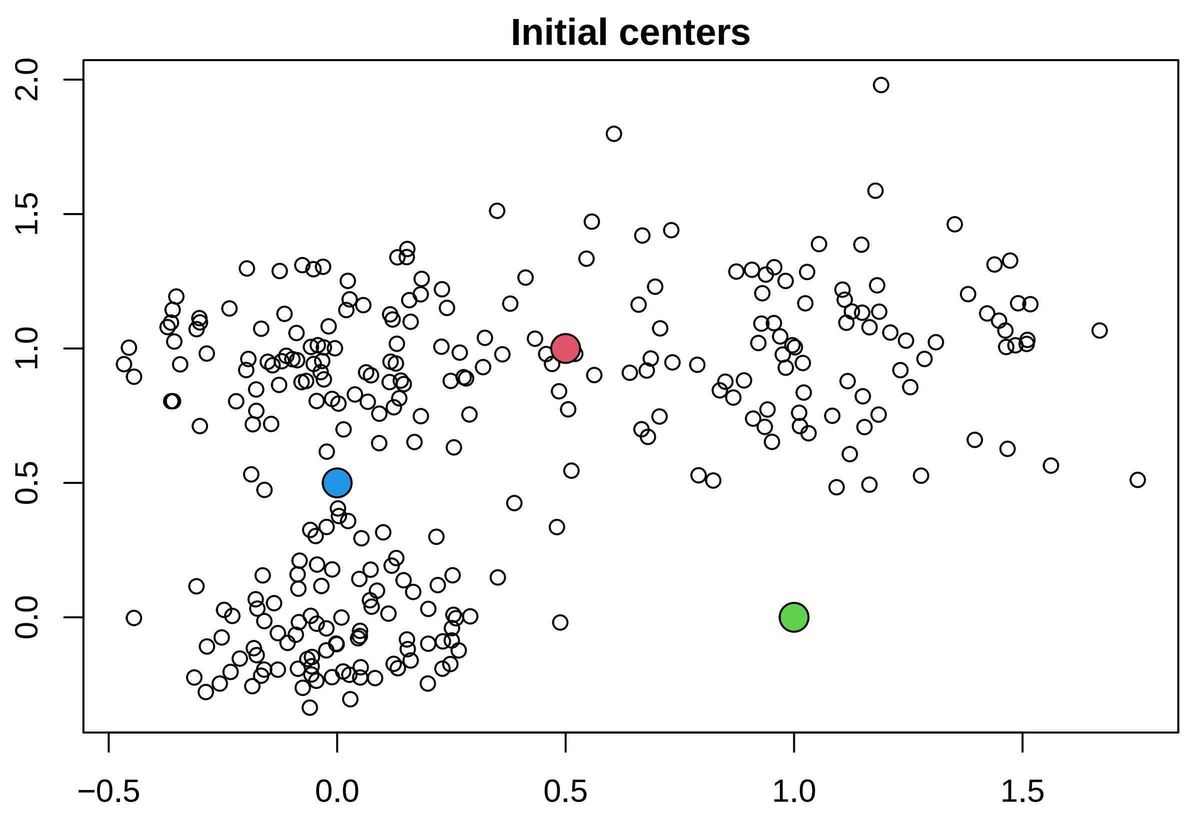


2.0
1.5
1.0
0.5
0.0
x[,2]
−0.5
0.0
0.5
1.0
1.5
x[,1]

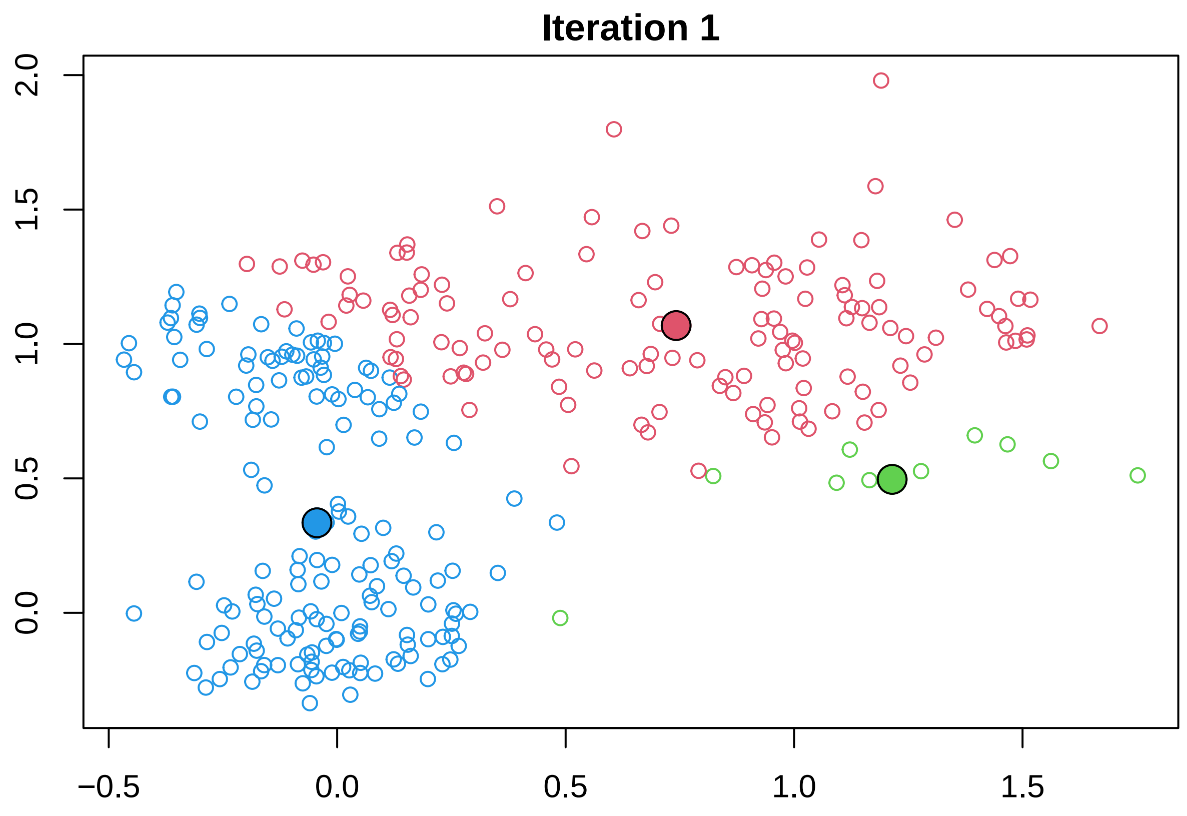


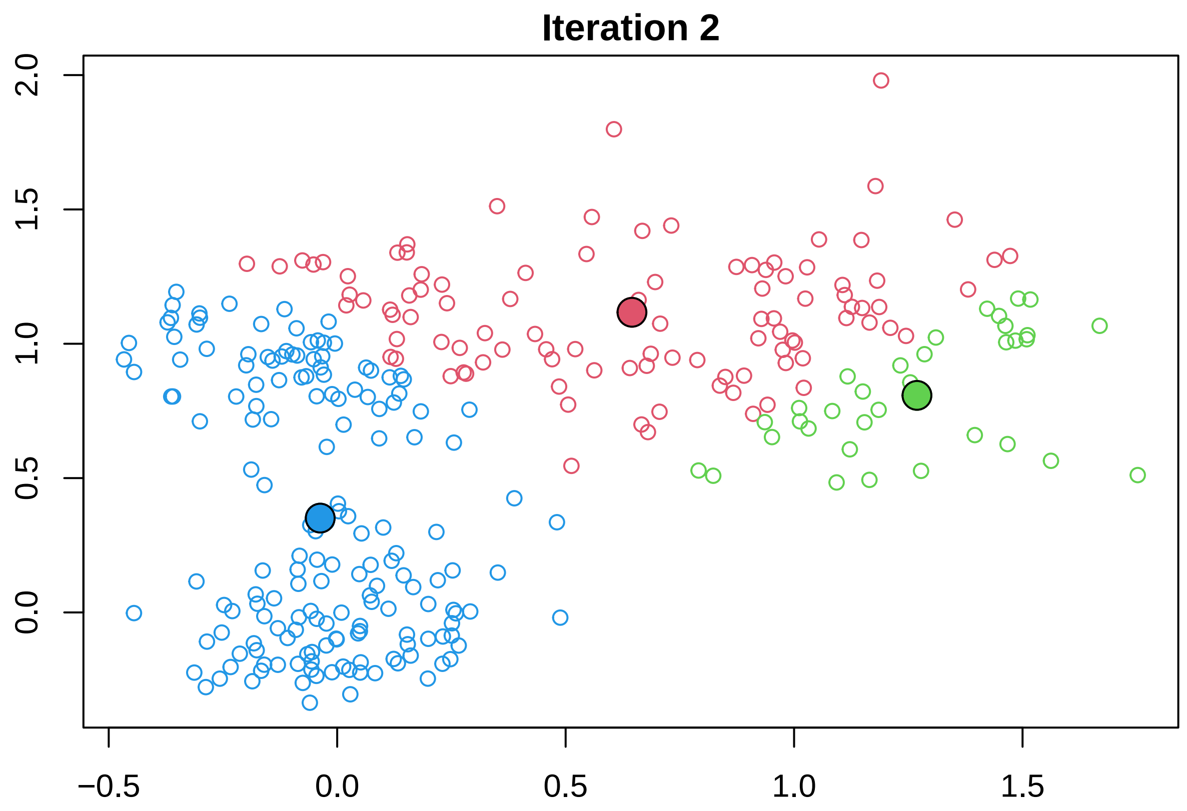
Now, we plot the initial centers along with iterations 1,2,3, and 9, after applying k-means.

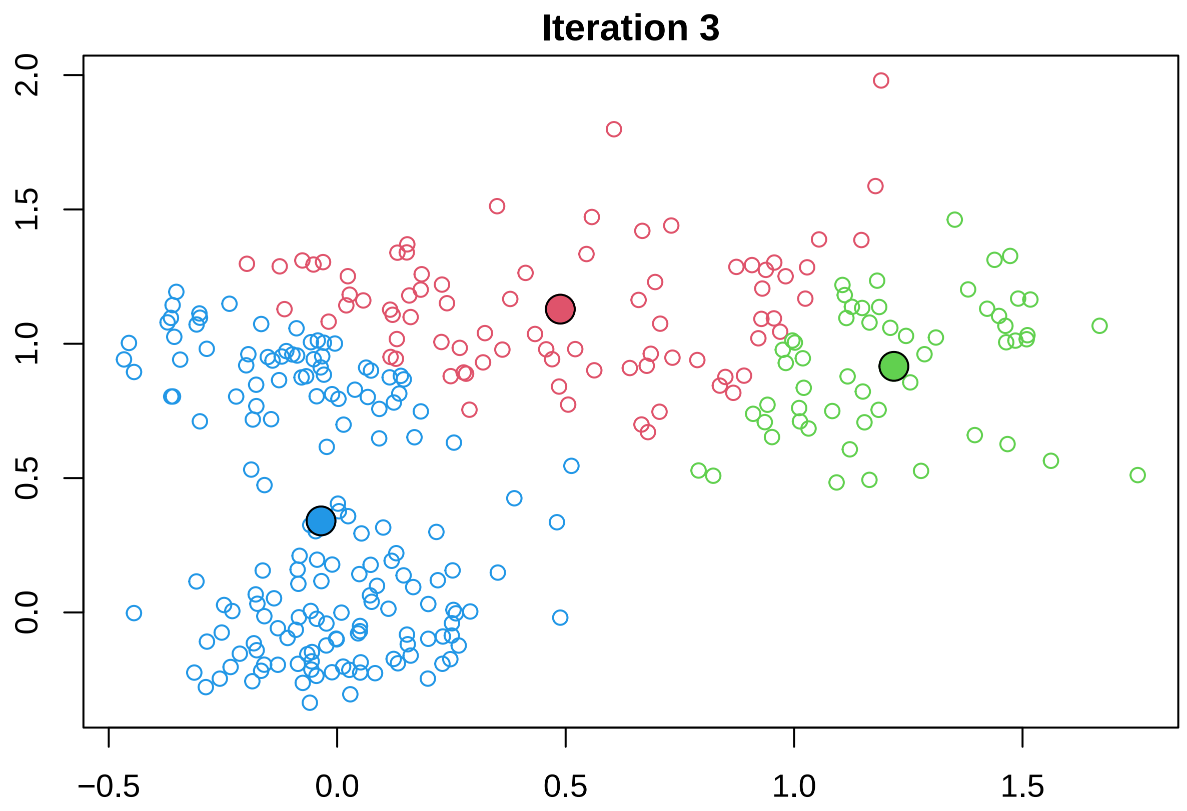
```
par(mar=c(3.5,3.5,1.5,0.5))
plot(x,xlab="",ylab="",main="Initial centers")
points(cent.init,pch=19,cex=2,col=2:4)
points(cent.init,pch=21,cex=2)
```

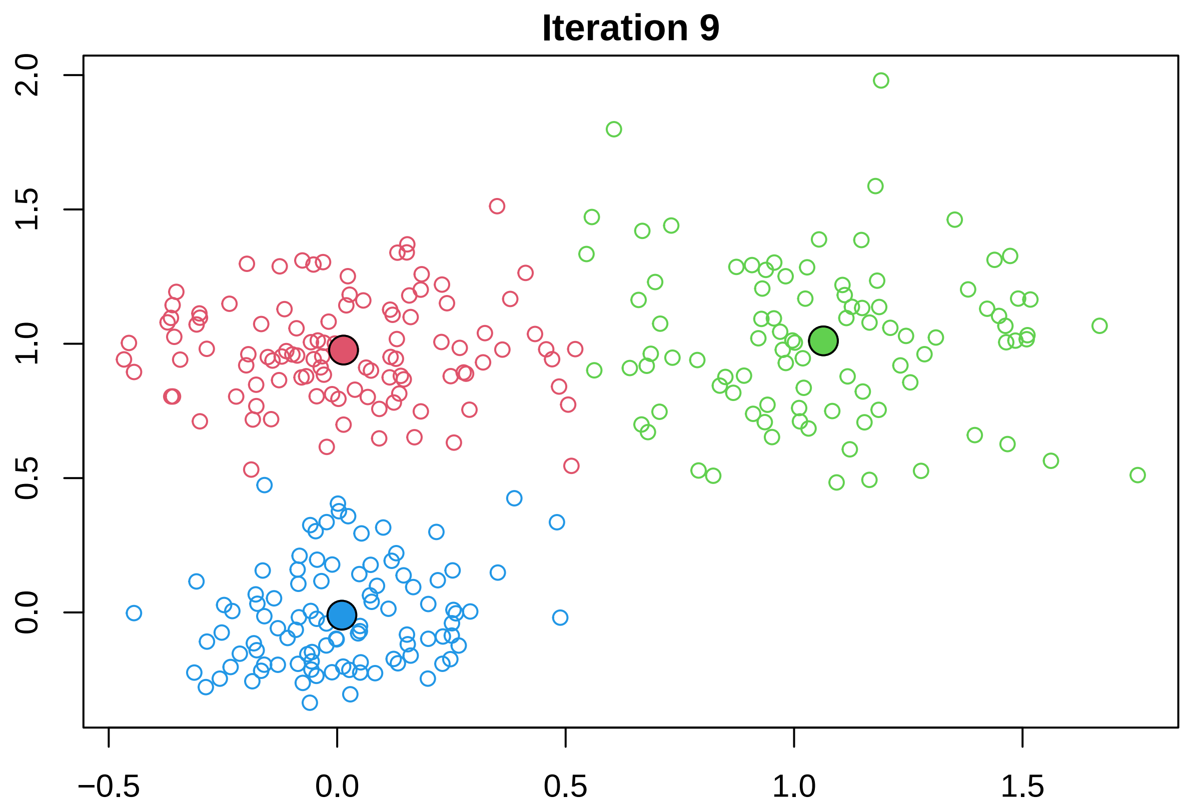


```
for (i in c(1,2,3,9)) {
par(mar=c(3.5,3.5,1.5,0.5))
plot(x,col=km$cluster.history[i,]+1,xlab="",ylab="",
main=sprintf("Iteration %i",i))
points(avgfromclust(x,km$cluster.history[i,],k),col=2:4,pch=19,cex=2)
points(avgfromclust(x,km$cluster.history[i,],k),pch=21,cex=2)
}
```


Iteration 1
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5


Iteration 2
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5


Iteration 3
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5



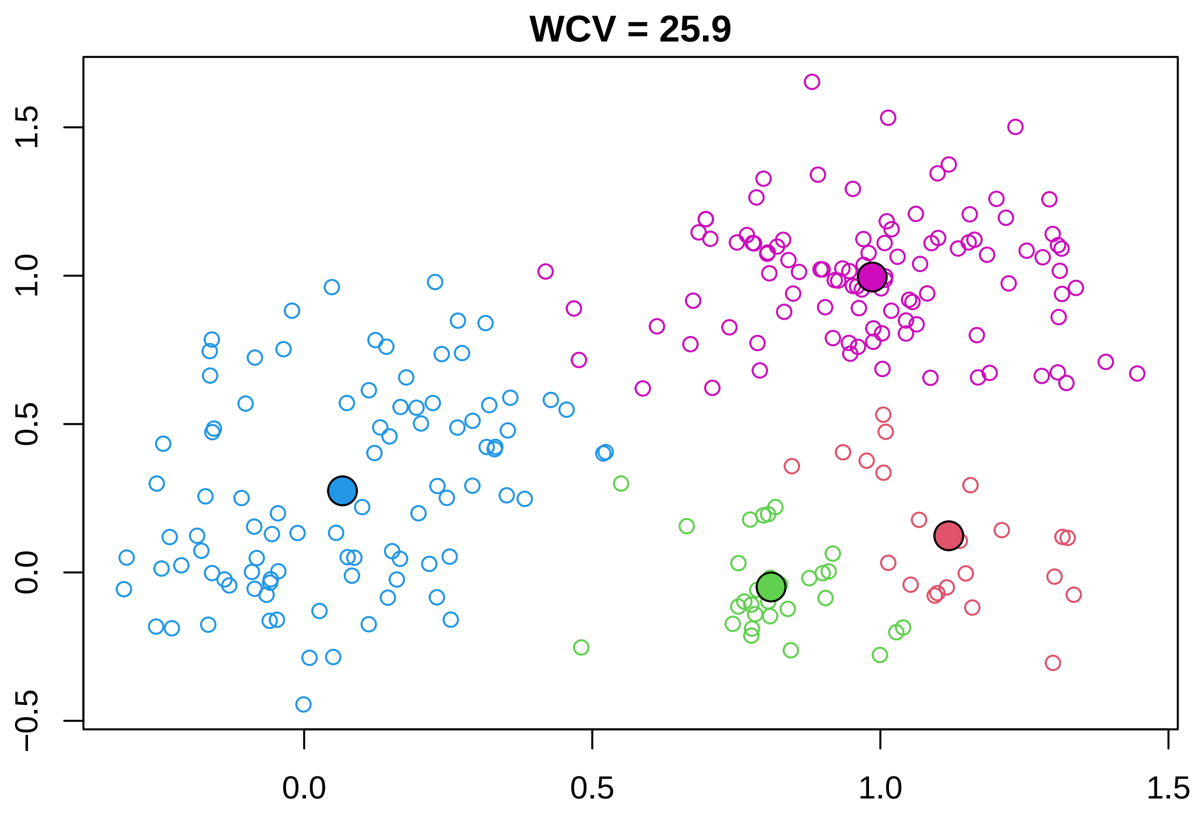
Our example that we gave was fairly simple. We could try different centers, but this doesn't change things much since the clusters were so well defined. Instead, we look at a less obvious case. To see this,

```
km2 = kmeans(x,centers=cent.init,alg="Lloyd")
sum(km$cluster!=km2$cluster)
```

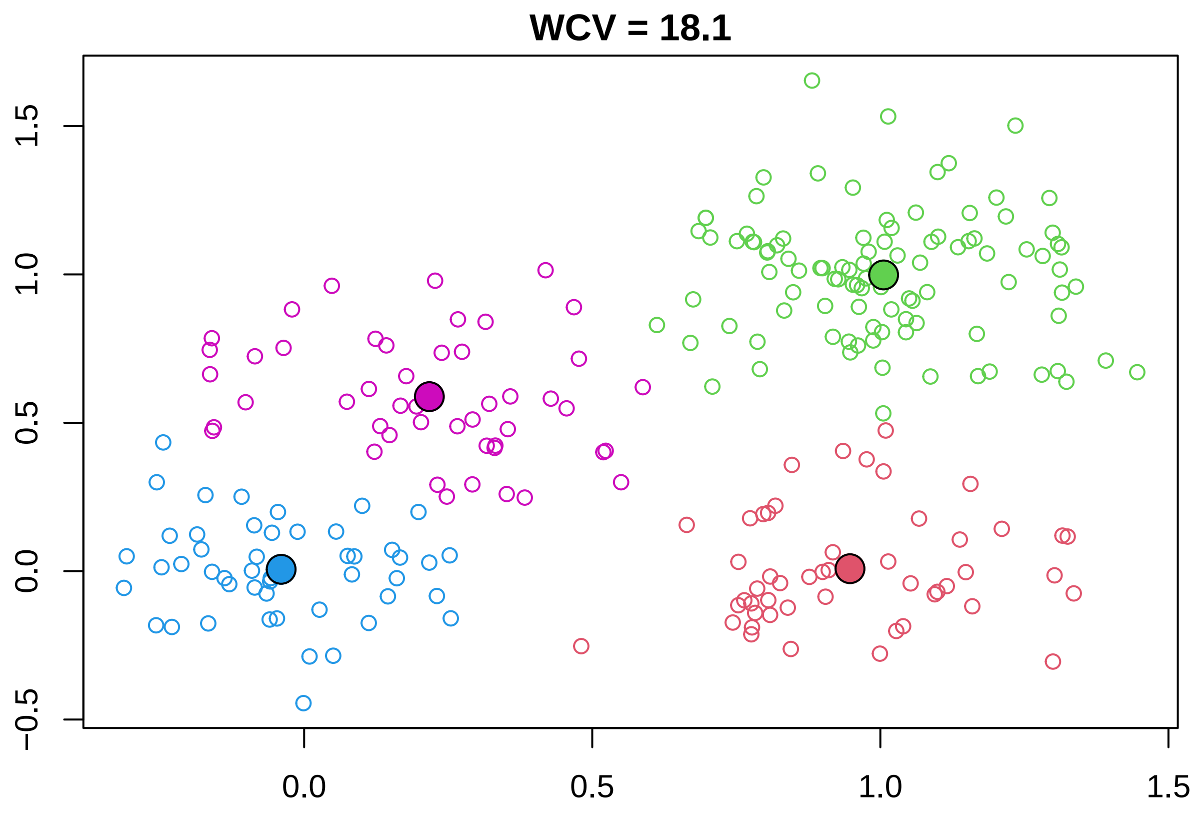
```
## [1] 0
```

```
set.seed(0)
x = rbind(matrix(rnorm(50*2,sd=0.2),ncol=2),
scale(matrix(rnorm(50*2,sd=0.2),ncol=2),cent=-c(1,0),scale=F),
scale(matrix(rnorm(50*2,sd=0.2),ncol=2),cent=-c(1,1),scale=F),
scale(matrix(rnorm(50*2,sd=0.2),ncol=2),cent=-c(1,1),scale=F),
scale(matrix(rnorm(50*2,sd=0.2),ncol=2),cent=-c(0.2,0.6),scale=F))

k = 4
R = 10
cols = c(2,3,4,6)
set.seed(0)
for (r in 1:3) {
  cat(sprintf("r=%i\n",r))
  cent.init = matrix(runif(k*2,min(x),max(x)),ncol=2)
  km3 = kclust(x,centers=cent.init,alg="kmeans")
  if (r==2) kmv=km3
  w = wcv(x,km3$cluster,km3$centers)
  par(mar=c(3.5,3.5,1.5,0.5))
  plot(x,col=cols[km3$cluster],xlab="",ylab="",
  main=sprintf("WCV = %0.1f",w))
  points(km3$centers,col=cols,pch=19,cex=2)
  points(km3$centers,pch=21,cex=2)
}
```

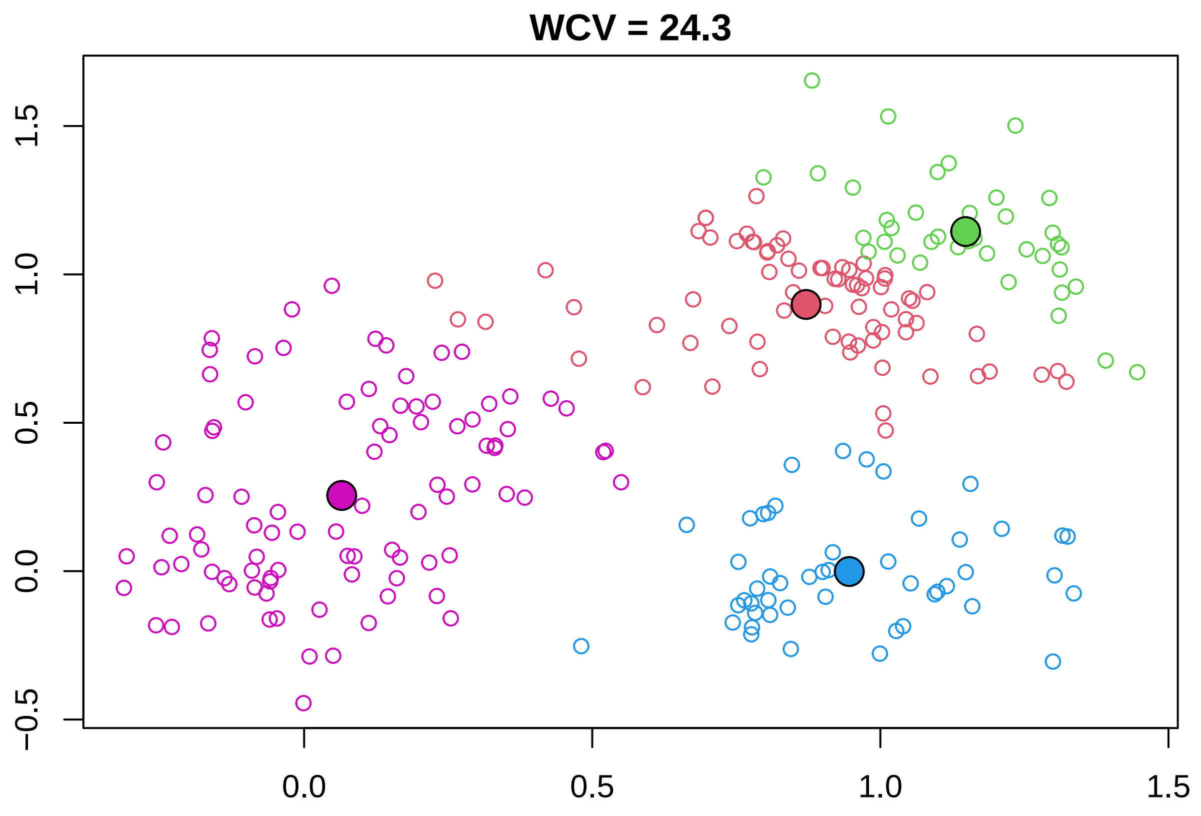
```
## r=1
```



```
## r=2
```

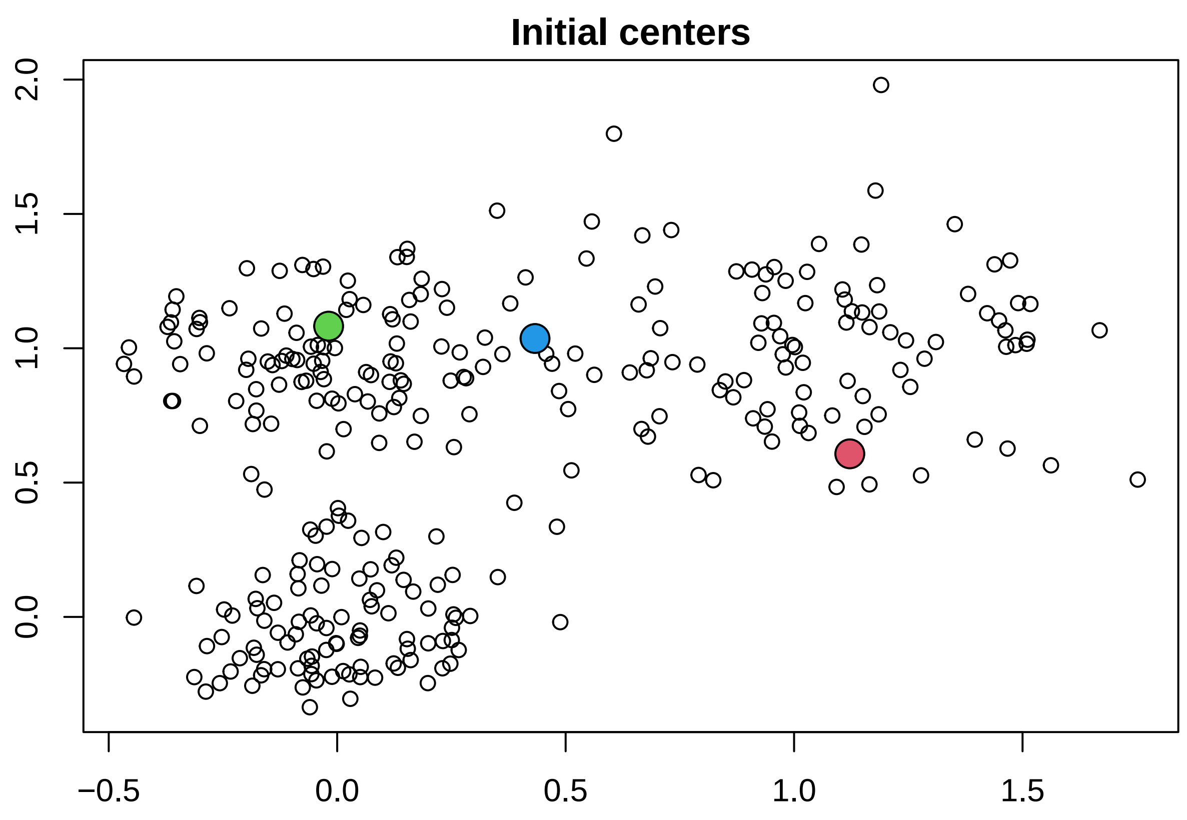


```
## r=3
```

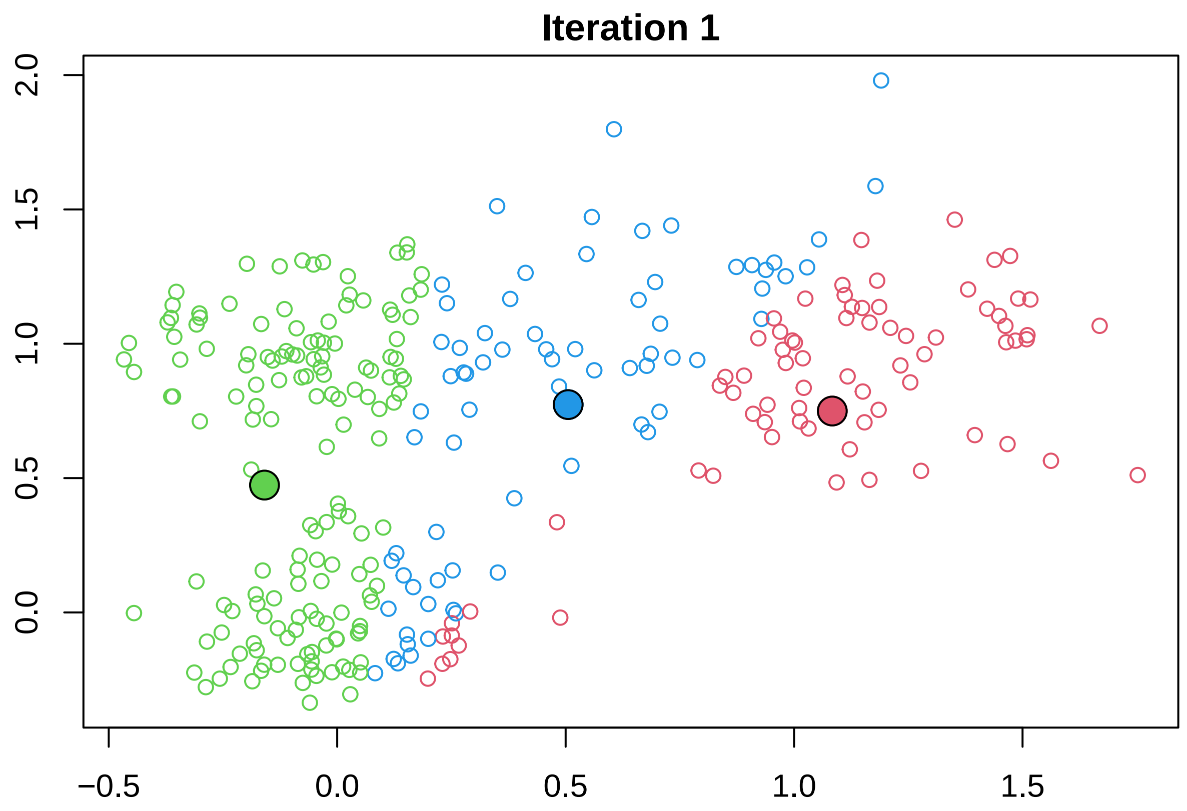


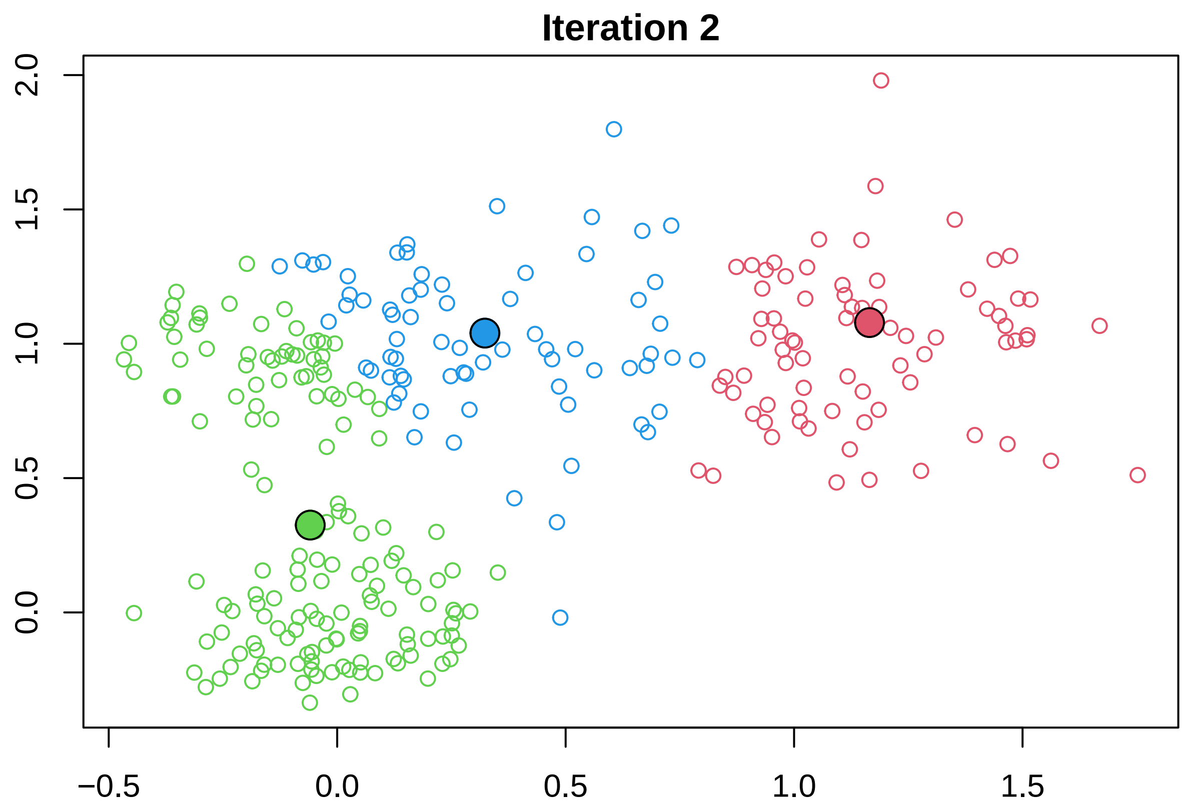
We now look at the k-medoids algorithm.

```
source("kclust.R")

set.seed(0)
x = rbind(matrix(rnorm(200,sd=0.2),ncol=2),
scale(matrix(rnorm(200,sd=0.3),ncol=2),cent=-c(1,1),scal=F),
scale(matrix(rnorm(200,sd=0.2),ncol=2),cent=-c(0,1),scal=F))

n = nrow(x)
k = 3
# we must set a different seed here and initialize
# the algorithm
set.seed(1)
cent.init = x[sample(n,k),]
cent.init = cent.init[c(2,3,1),]
kmed = kclust(x,centers=cent.init,alg="kmedoids")

par(mar=c(3.5,3.5,1.5,0.5))
plot(x,xlab="",ylab="",main="Initial centers")
points(cent.init,pch=19,cex=2,col=2:4)
points(cent.init,pch=21,cex=2)
```

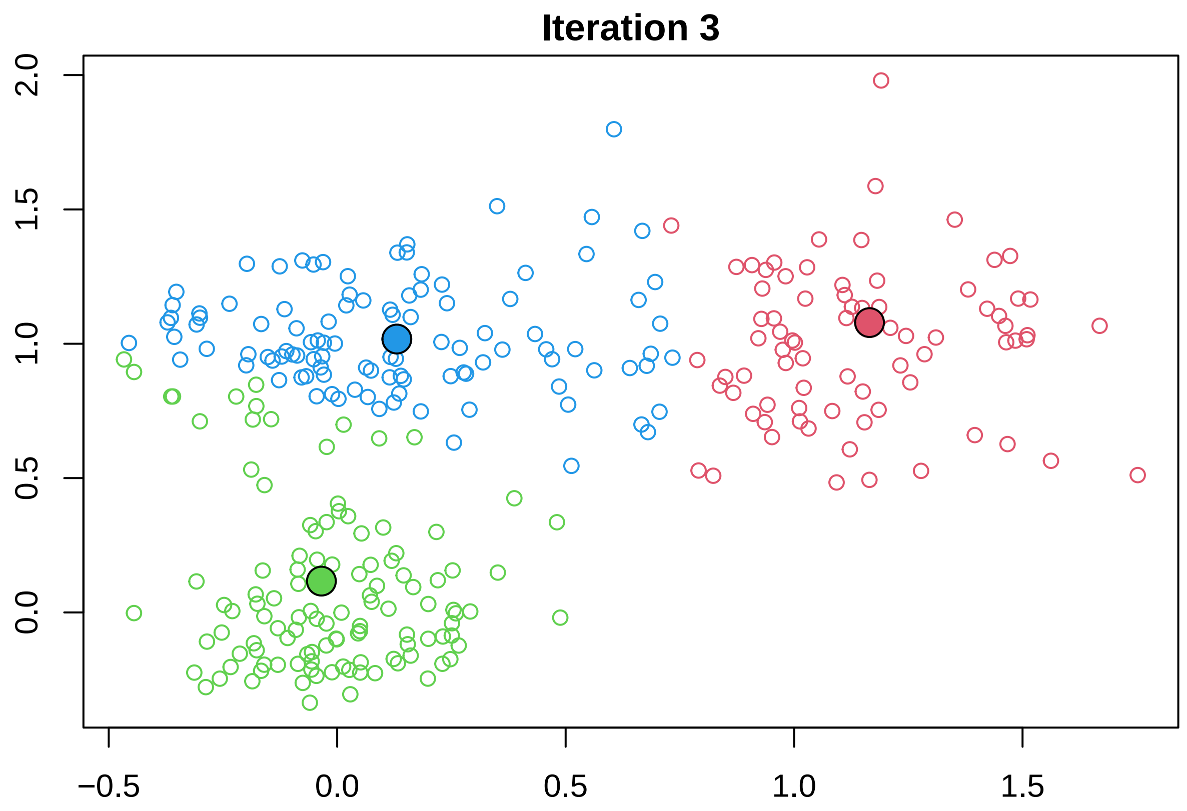


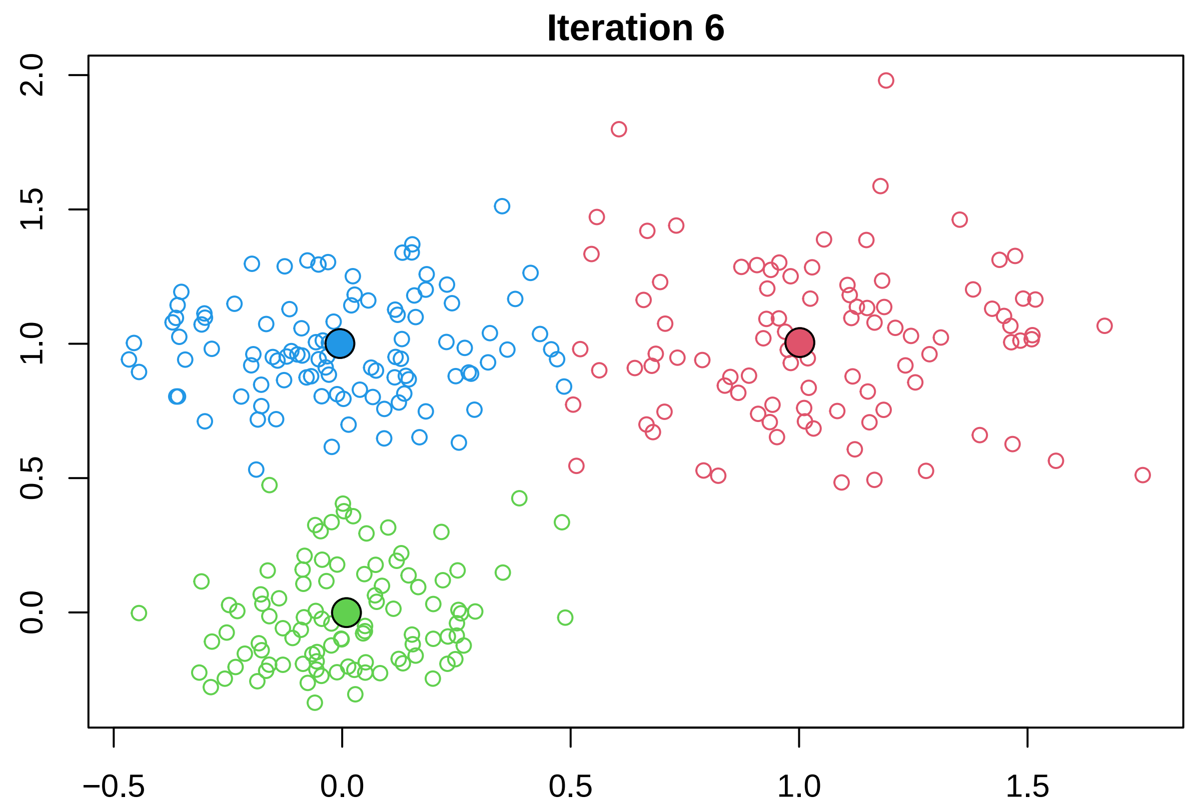
```r
for (i in c(1,2,3,6)) {
  par(mar=c(3.5,3.5,1.5,0.5))
  plot(x,col=kmed$cluster.history[i,]+1,xlab="",ylab="",
  main=sprintf("Iteration %i",i))
  points(medfromclust(x,kmed$cluster.history[i,],k),col=2:4,pch=19,cex=2)
  points(medfromclust(x,kmed$cluster.history[i,],k),pch=21,cex=2)
}
```


Iteration 1
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5



Iteration 2
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5


Iteration 3
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
1.0
1.5



Hierarchical Clustering