Visualization of K-Means

Data Analytics and Visualization (Spring 2019)

Instructor: Debopriya Ghosh

Reference: http://mcube.nju.edu.cn/jokergoo/animation-of-kmeans-clustering.html

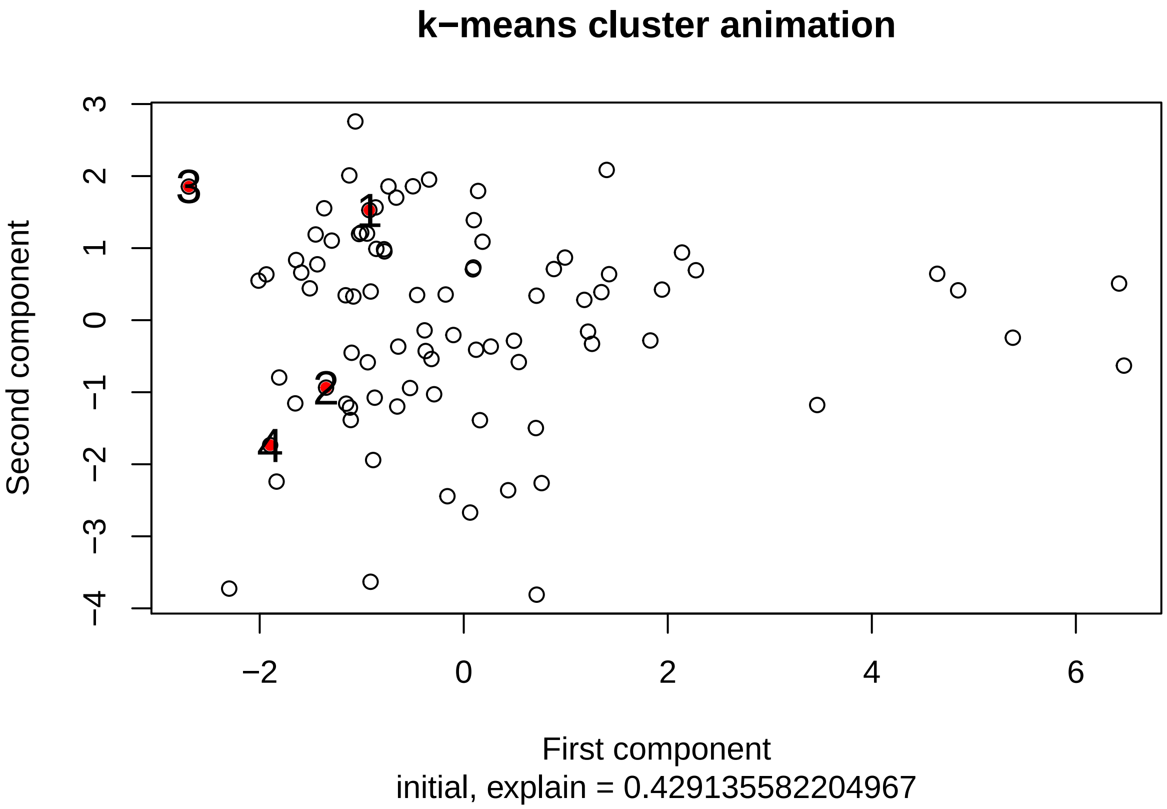
```r
# defining the data
whiskies = read.csv("C:/Users/zhuwe/Desktop/Visualization/Dataset/whiskies.txt")
whiskies = whiskies[,-1]
# generating a subset of the data that included only the 12 flavor variables, rescaled for comparabilit
whiskies_k = scale(whiskies[,2:13])  # rescale selected vars for kmeans
```

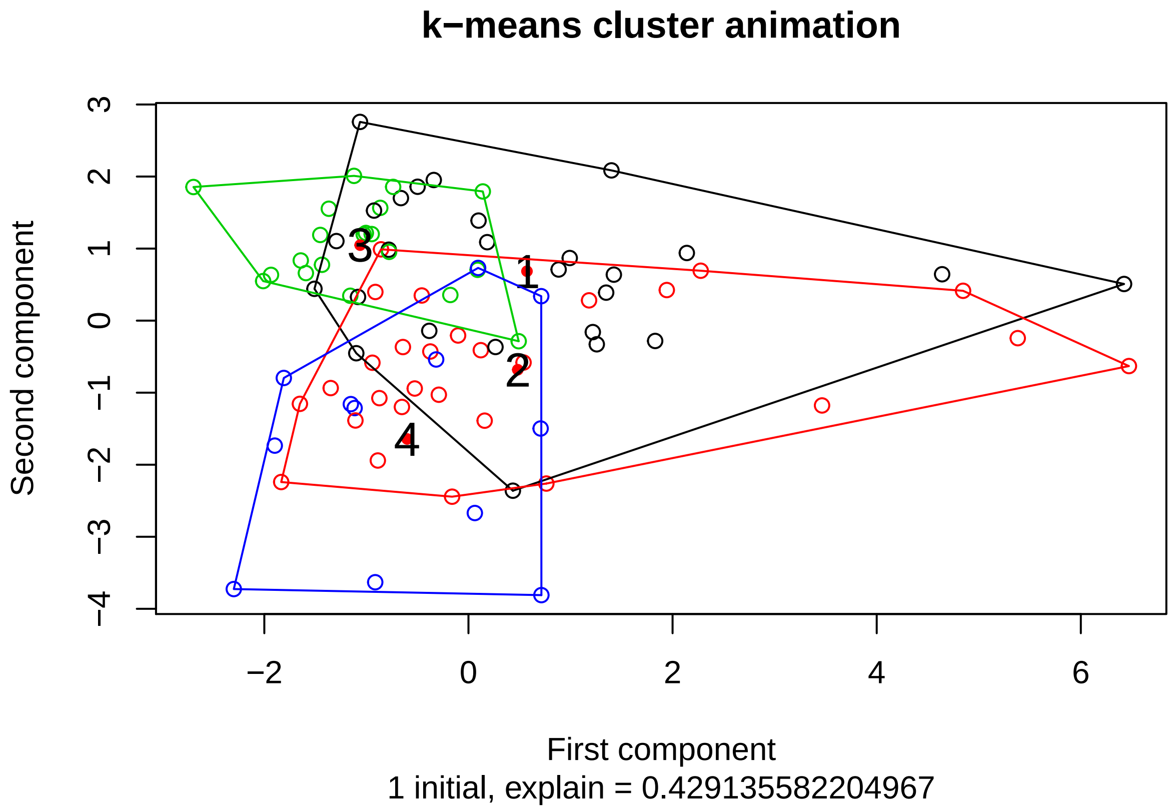
```r
draw.kmeans = function(w, k=2, dimension=2, iteration=500,
                       method="euclidean", initial="random", save=FALSE) {
  require(rgl)
  if(!(dimension ==2 || dimension == 3)) {
    stop("dimension should be 2 or 3\n")
  }
  if(dimension > dim(w)[2]) {
    stop("dimesion should be no smaller than that of data\n")
  }
  if(dim(w)[2] < 2) {
    stop("w must contain at least two dimensions\n")
  }
  if(dim(w)[2] > dimension) {
    pca.res = prcomp(w)
    comp = pca.res$x[,1:dimension]
    explain = sum(pca.res$sdev[1:dimension]^2)/sum(pca.res$sdev^2)
  }
  center = matrix(0, nrow=k, ncol=dim(w)[2])
  if(initial == "quantile") {
    for(i in 1:k) {
      center[i, ] = apply(w, 2, function(x) {quantile(x, i/(k+1))})
    }
  }
  if(initial == "random") {
    center = w[sample(1:dim(w)[1], k, replace=FALSE), ]
  }

  if(save) {
    dir = paste(sample(letters, 26, replace=FALSE), collapse="")
    dir = paste("_", dir, sep="")
    dir.create(dir)
    setwd(dir)
  }
  if(save && dimension == 2) {
    png(file="0.k-means.png")
  }
  if(dimension == 2) {
    if(dim(w)[2] > 2) {
      plot(comp[,1], comp[,2], xlab="First component", ylab="Second component",
           sub=paste("initial, explain =", explain), main="k-means cluster animation")
      cp = pca.trans(center, pca.res, k = 2)
      points(cp[,1], cp[,2], col="red", pch=20)
      text(cp[,1], cp[,2], 1:k, cex=1.5)
    }
    else {
      plot(w[,1], w[,2], xlab="x", ylab="y", sub="initial", main="k-means cluster animation")
      points(center[,1], center[,2], col="red", pch=20)
      text(center[,1], center[,2], 1:k, cex=1.5)
    }
  }
  if(dimension == 3) {
    if(dim(w)[2] > 3) {
      plot3d(comp[,1], comp[,2], comp[, 3], xlab="First component", ylab="Second component",
             zlab="Third component", sub=paste("initial, explain =", explain),
             main="k-means cluster animation")
      cp = pca.trans(center, pca.res, k = 3)
      points3d(cp[,1], cp[,2], cp[,3], col="red", pch=20)
      text3d(cp[,1], cp[,2], cp[,3], 1:k, cex=rep(1.5, dim(cp)[1]))
    }
    else {
      plot3d(w[,1], w[,2], w[,3], xlab="x", ylab="y", zlab="z", sub="initial",
             main="k-means cluster animation")
      points3d(center[,1], center[,2], center[,3], col="red", pch=20)
      text3d(center[,1], center[,2], center[,3], 1:k, cex=rep(1.5, dim(center)[1]))
    }
  }
  if(save && dimension == 2) {
    dev.off()
  }
  if(save && dimension == 3) {
    snapshot3d("0.k-means.png")
  }

  # do kmeans
  belong.flag = NULL
  iter = 1
  while(iter <= iteration) {
    Sys.sleep(1)
    d = matrix(0, nrow=dim(w)[1], ncol=k)
    for (c in 1:k) {
      d[, c] = apply(w, 1, function(x){distance(x, center[c,], method=method)})
    }
    belong = matrix(FALSE, nrow=dim(w)[1], ncol=k)
    for (b in 1:dim(belong)[1]) {
      belong[b, which.min(d[b, ])] = TRUE
    }
    for(i in 1:k) {
      center[i, ] = apply(w, 2, function(x) {sum(x * belong[, i])/sum(belong[, i])})
    }
    # draw
    if(save && dimension == 2) {
      png(file=paste(iter, ".k-means.png", sep=""))
    }
    color = numeric(dim(belong)[1])
    for(i in 1:dim(belong)[2]) {
      color = color + i*belong[,i]
    }
    if(dimension == 2) {
      if(dim(w)[2] > 2) {
        plot(comp[,1], comp[,2], xlab="First component", ylab="Second component",
             sub=paste(iter, "initial, explain =", explain), main="k-means cluster animation", col=colo
        cp = pca.trans(center, pca.res, k = 2)
        points(cp[,1], cp[,2], col="red", pch=20)
        text(cp[,1], cp[,2], 1:k, cex=1.5)
        for(i in 1:k) {
          x = comp[belong[, i], 1]
          y = comp[belong[, i], 2]
          hull = chull(x, y)
          polygon(x[hull], y[hull], border=i)
        }
      }
      else {
        plot(w[,1], w[,2], xlab="x", ylab="y", sub=paste(iter, "iterations"),
             main="k-means cluster animation", col=color)
        points(center[,1], center[,2], col="red", pch=20)
        text(center[,1], center[,2], 1:k, cex=1.5)
        for(i in 1:k) {
          x = w[belong[, i], 1]
          y = w[belong[, i], 2]
          hull = chull(x, y)
          polygon(x[hull], y[hull], border=i)
        }
      }
    }
    if(dimension == 3) {
      if(dim(w)[2] > 3) {
        plot3d(comp[,1], comp[,2], comp[, 3], xlab="First component", ylab="Second component",
               zlab="Third component", sub=paste(iter, "initial, explain =", explain),
               main="k-means cluster animation", col=color)
        cp = pca.trans(center, pca.res, k = 3)
        points3d(cp[,1], cp[,2], cp[,3], col="red")
        text3d(cp[,1], cp[,2], cp[,3], 1:k, cex=rep(1.5, dim(cp)[1]))
      }
      else {
        plot3d(w[,1], w[,2], w[,3], xlab="x", ylab="y", zlab="z", sub=paste(iter, "iterations"),
               main="k-means cluster animation", col=color)
        points3d(center[,1], center[,2], center[,3], col="red")
        text3d(center[,1], center[,2], center[,3], 1:k, cex=rep(1.5, dim(center)[1]))
      }
    }
    if(save && dimension == 2) {
      dev.off()
    }
    if(save && dimension == 3) {
      snapshot3d(paste(iter, ".k-means.png", sep=""))
    }

    cat(paste(iter, "iterations\n"))
    if(identical(belong, belong.flag)) {
      break
    }
    belong.flag = belong
    iter = iter + 1
  }
  if(save) {
    pstr = Sys.getenv()["PATH"]
    path = unlist(strsplit(pstr, ";"))
    image.magick.path = path[grepl("ImageMagick", path)]
    convert.path = file.path(image.magick.path, "convert.exe")
    command = paste("\"", convert.path, "\"", " -delay 80 *.k-means.png ../output.gif")
    #system(command)
    #file.remove(list.files(pattern=".png"))
    setwd("..")
    #file.remove(dir)
    cat(paste("file at ", getwd() , "/", dir, "\n", sep=""))
  }
  rownames(belong) = rownames(w)
  colnames(belong) = paste("C", 1:k, sep="")
  return(invisible(belong))
}

# find the principle component
pca.trans = function(m, pca.res=prcomp(m), k =2) {
  if(k > dim(m)[2]) {
    stop("!")
  }
  w = matrix(0, nrow=dim(m)[1], ncol=k)
  for(i in 1:dim(m)[1]) {
    for ( j in 1:k) {
      w[i,j] = sum((m[i, ] - pca.res$center)*pca.res$rotation[,j])
    }
  }
  return(w)
}

# calculate distance between two vectors
distance = function(x, y, method="euclidean") {
  m = matrix(0, nrow=2, ncol=length(x))
  m[1, ] = x
  m[2, ] = y
  if(method == "pearson" || method == "spearman" || method == "kendall") {
    d = 1 - cor(x, y, method=method)
  }
  else {
    d = as.vector(dist(m, method=method))
  }
  return(d)
}
```

Visualizing k-means

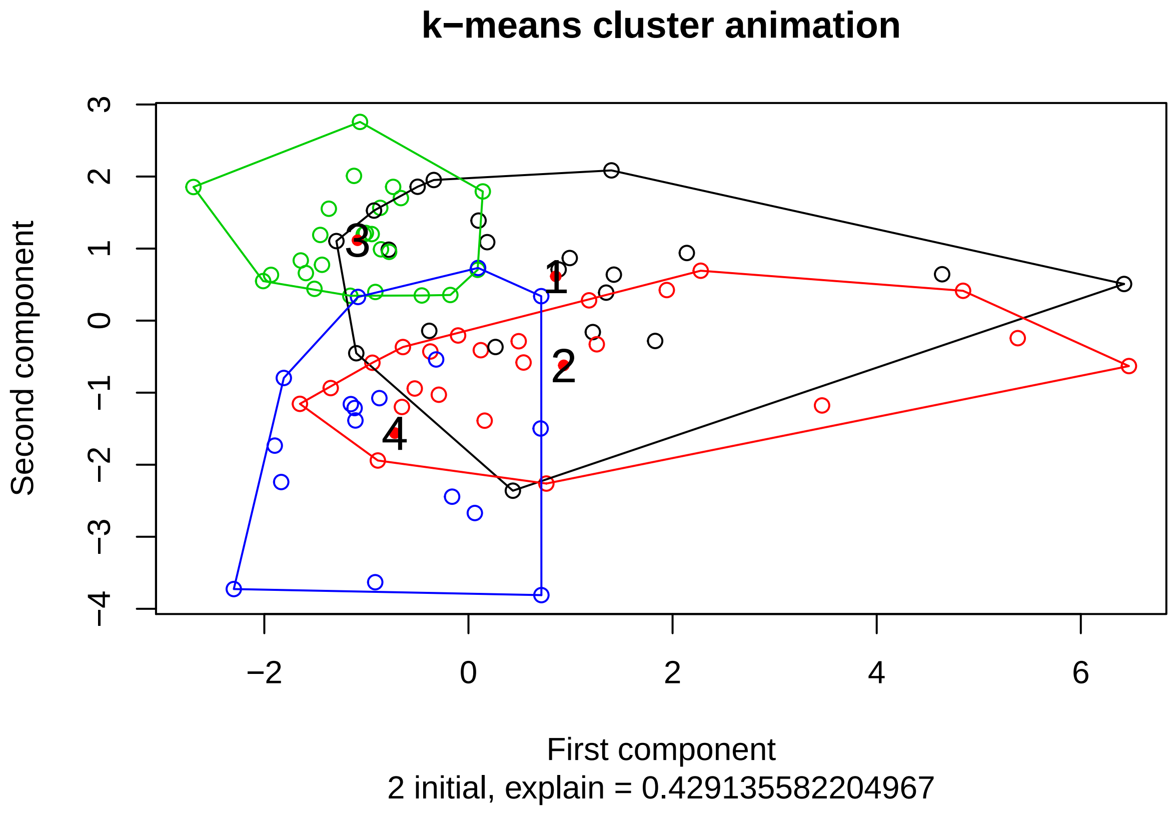
```
draw.kmeans(whiskies_k, k=4)
```

```
## Loading required package: rgl
```

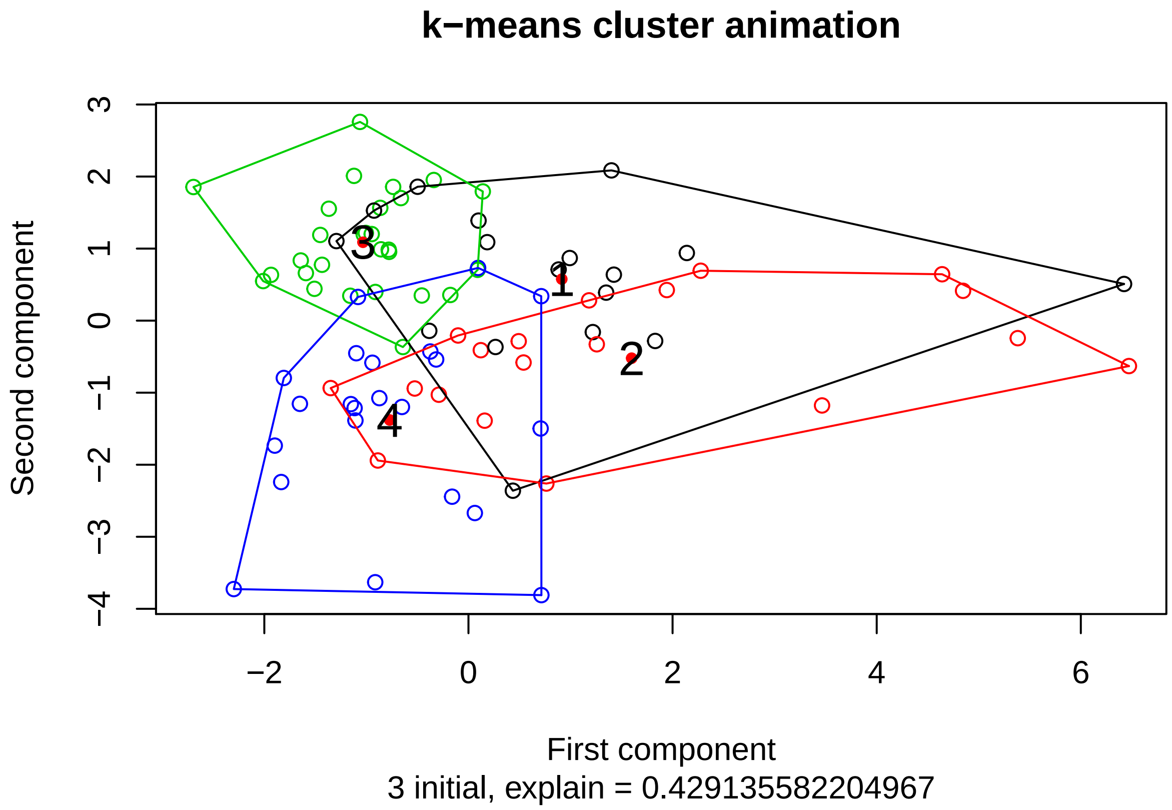




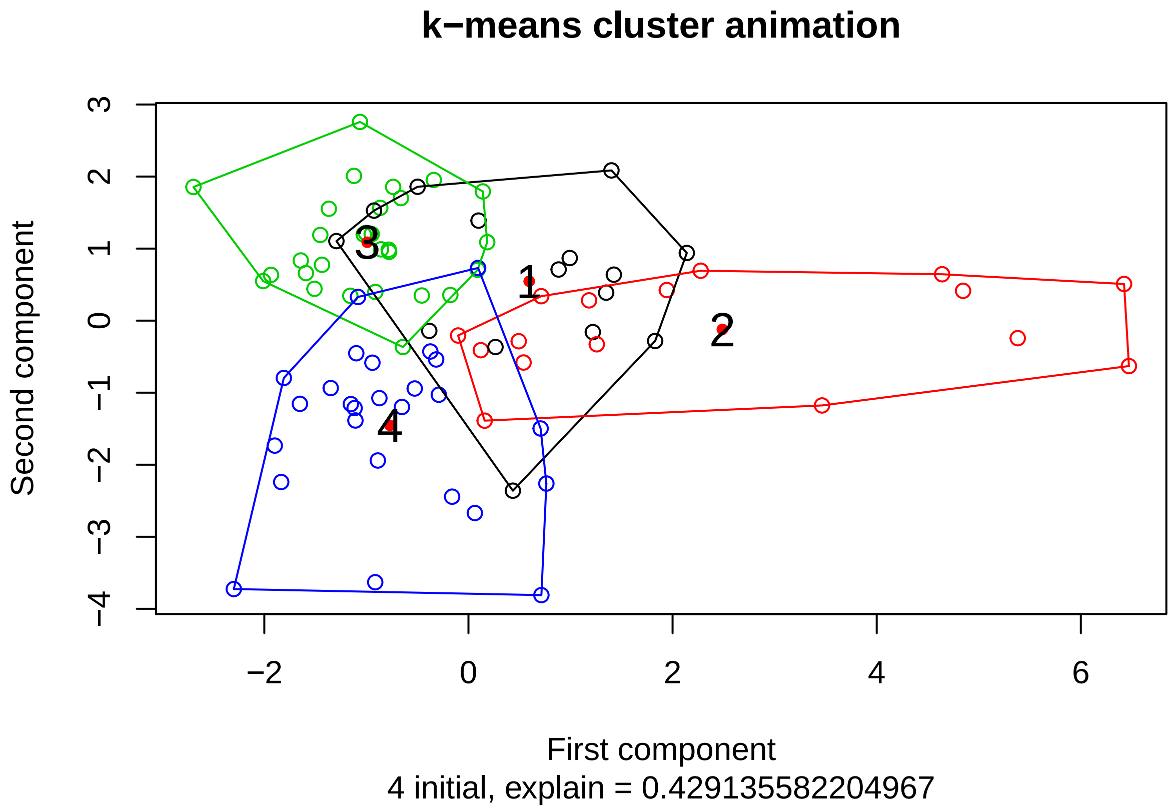
```
## 1 iterations
```



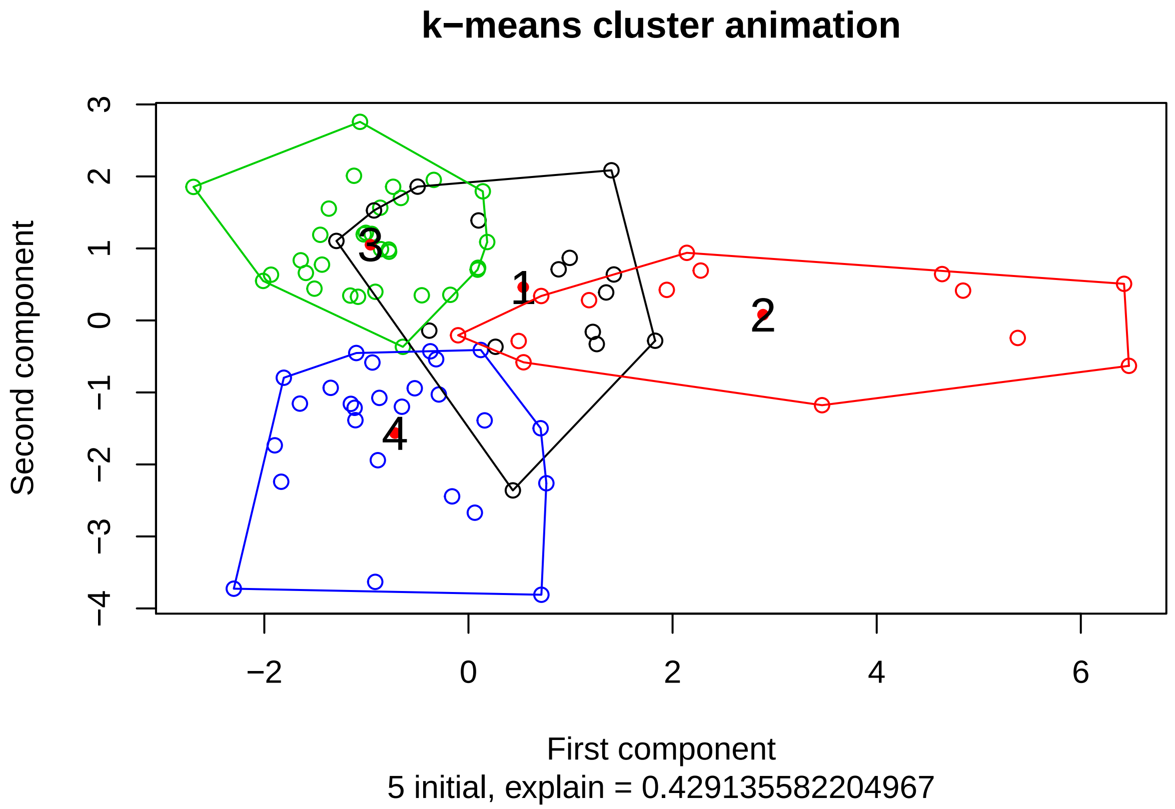
```
## 2 iterations
```



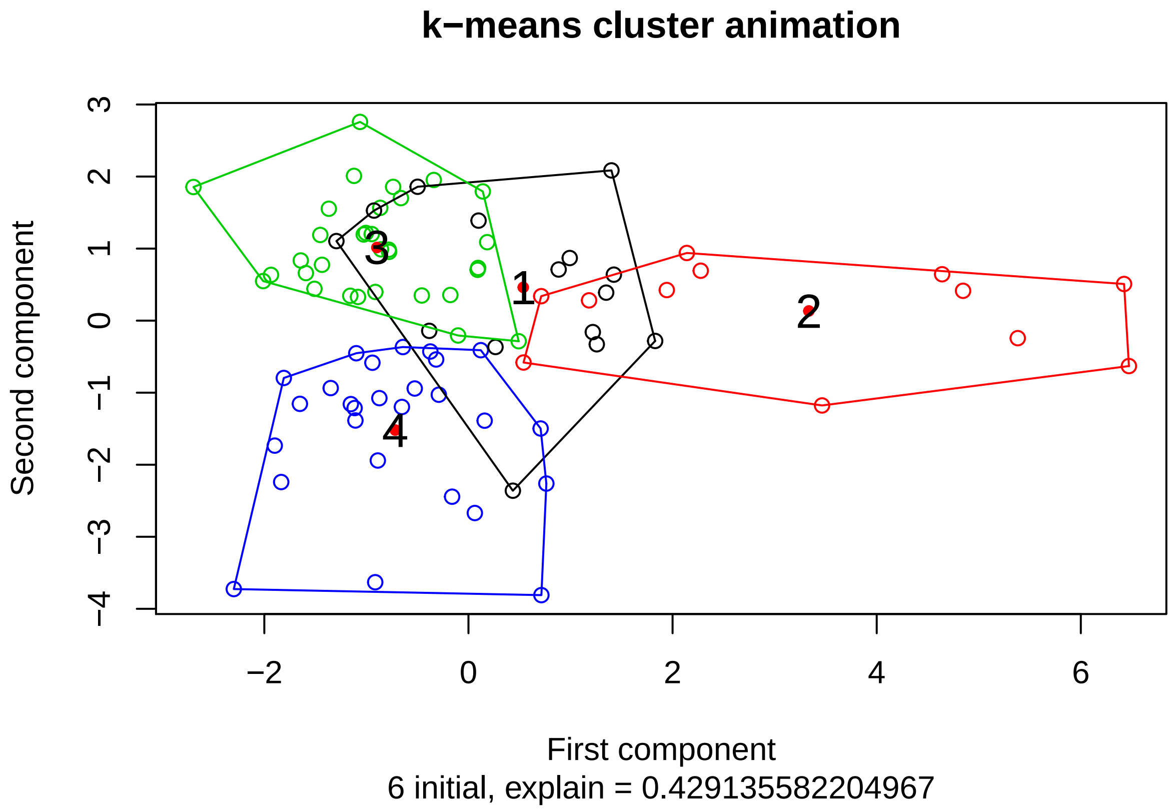
```
## 3 iterations
```



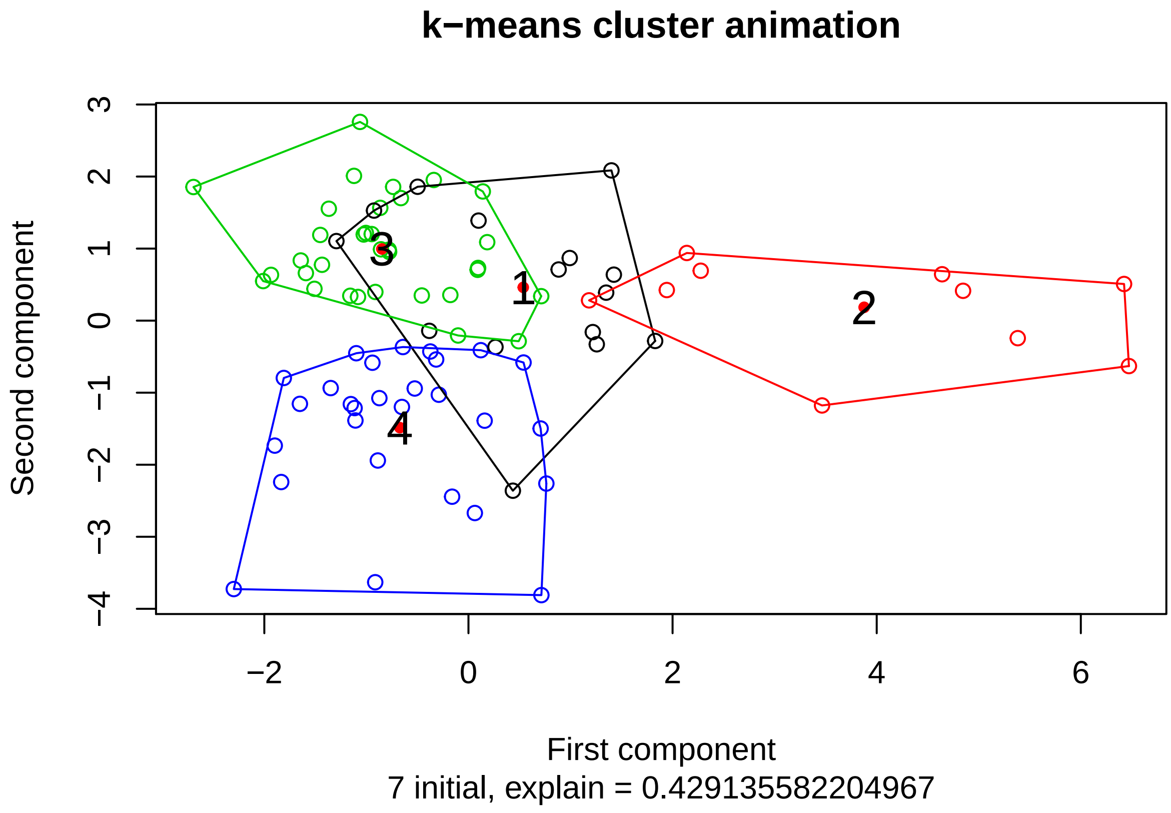
```
## 4 iterations
```



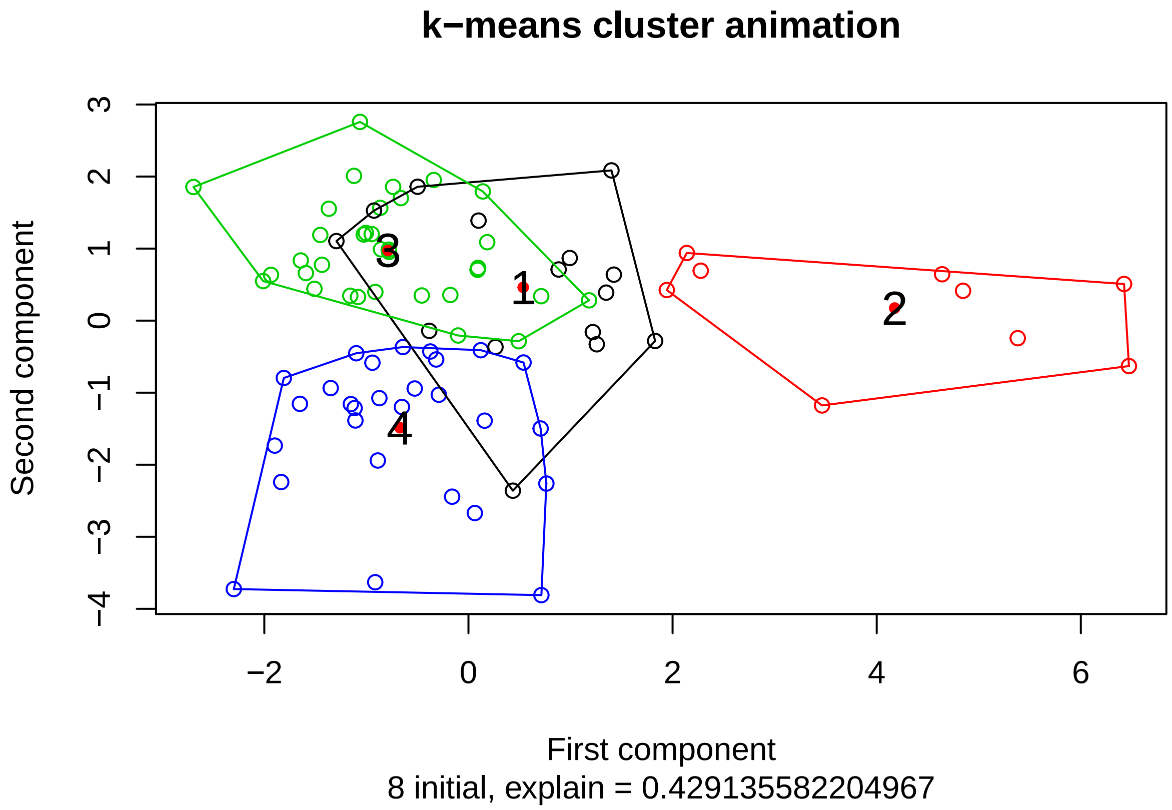
```
## 5 iterations
```



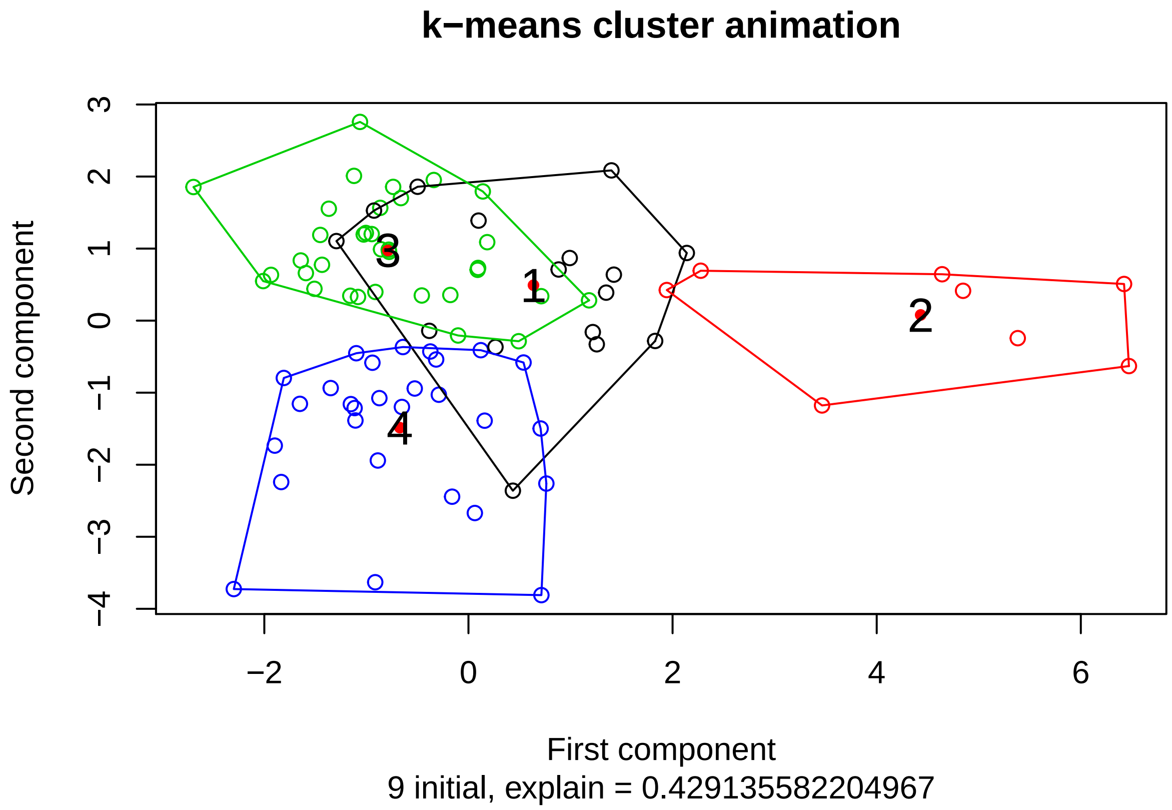
```
## 6 iterations
```



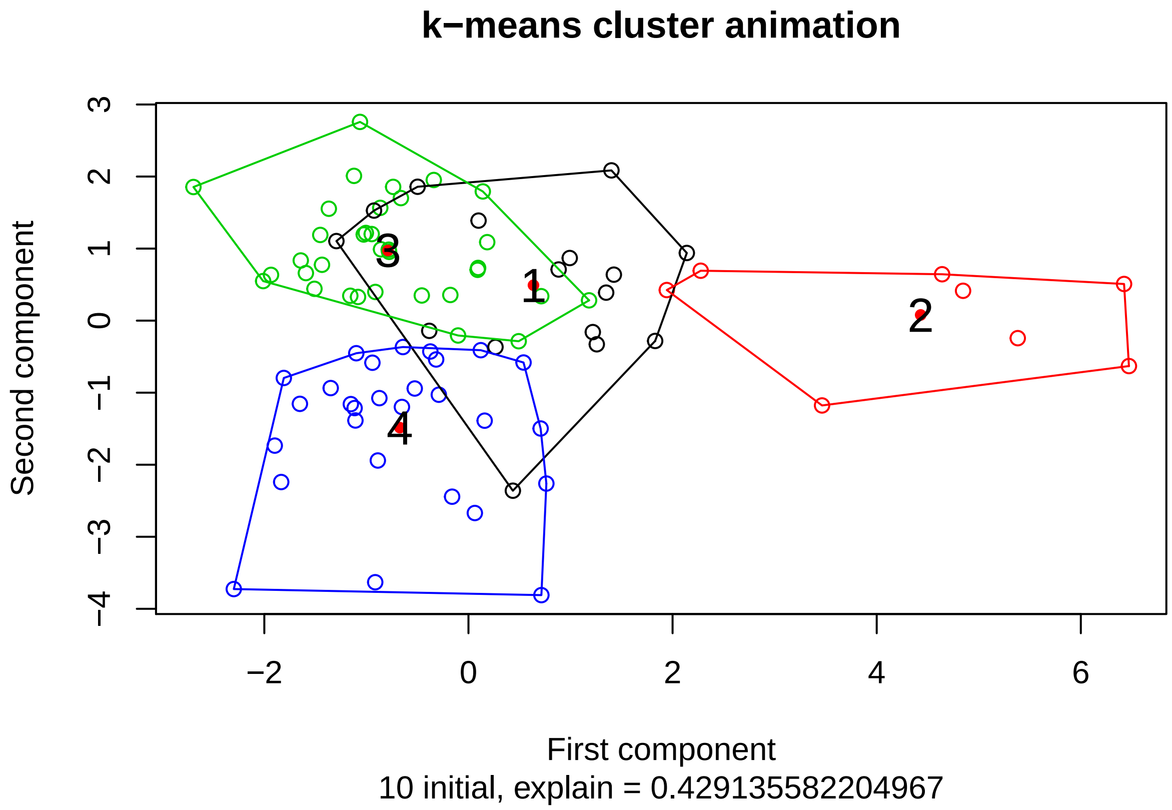
```
## 7 iterations
```



```
## 8 iterations
```



```
## 9 iterations
```



```
## 10 iterations
```