Cluster Analysis Lab 2

Craig Alexander

05/04/2023

1. Introduction and aims of the session

In today's lab we will be fitting k-means models and looking at different ways of choosing K, the number of clusters as well as comparing clustering results.

1.1 R commands

The `R` commands introduced today are:

- `kmeans` - fitting k-means clustering from the `stats` library
- `dist` - calculating distances matrices on observation $\times$ variable data
- `pam` - fitting k-medoids (partitioning around medoids) clustering models from `cluster` library
- `clusGap` - calculating the gap statistic for choosing the number of clusters from `cluster` library
- `silhouette` - calculating the silhouette width statistics from the `cluster` library
- `adjustedRand` - calculating the agreement indices between two clustering results on the same data set from the `clues` library

2. K-means clustering

2.1 Life Expectancy in 1960's

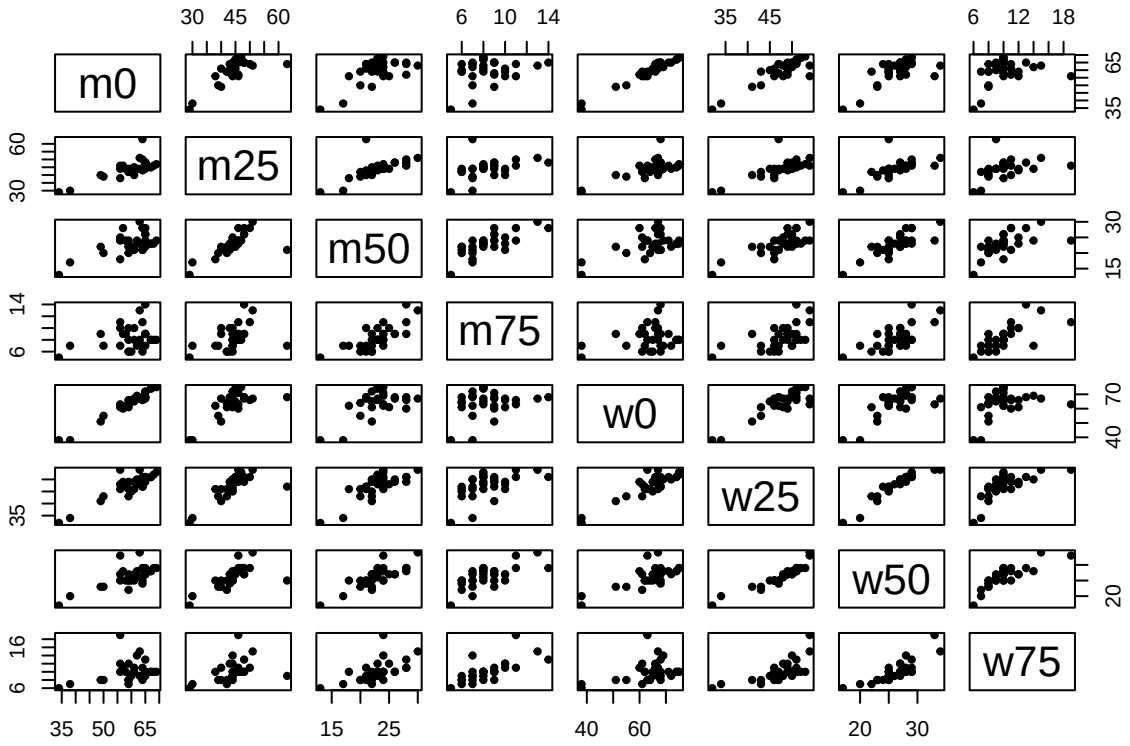
Here, we'll revisit the life expectancy data we looked at last lab with hierarchical clustering.

Data exists that records the life expectancy in the 1960s by country, age and sex.
This data is recorded in years and is accessible to R as object life via the `lifeexp.dat` file on the moodle page.

```
#setwd()
source("lifeexp.dat")
```

As always, we start by doing some initial summaries of the data. We can look at the pairs plot to see if there is any obvious sign of group structure.

```
pairs(life,pch=20)
```



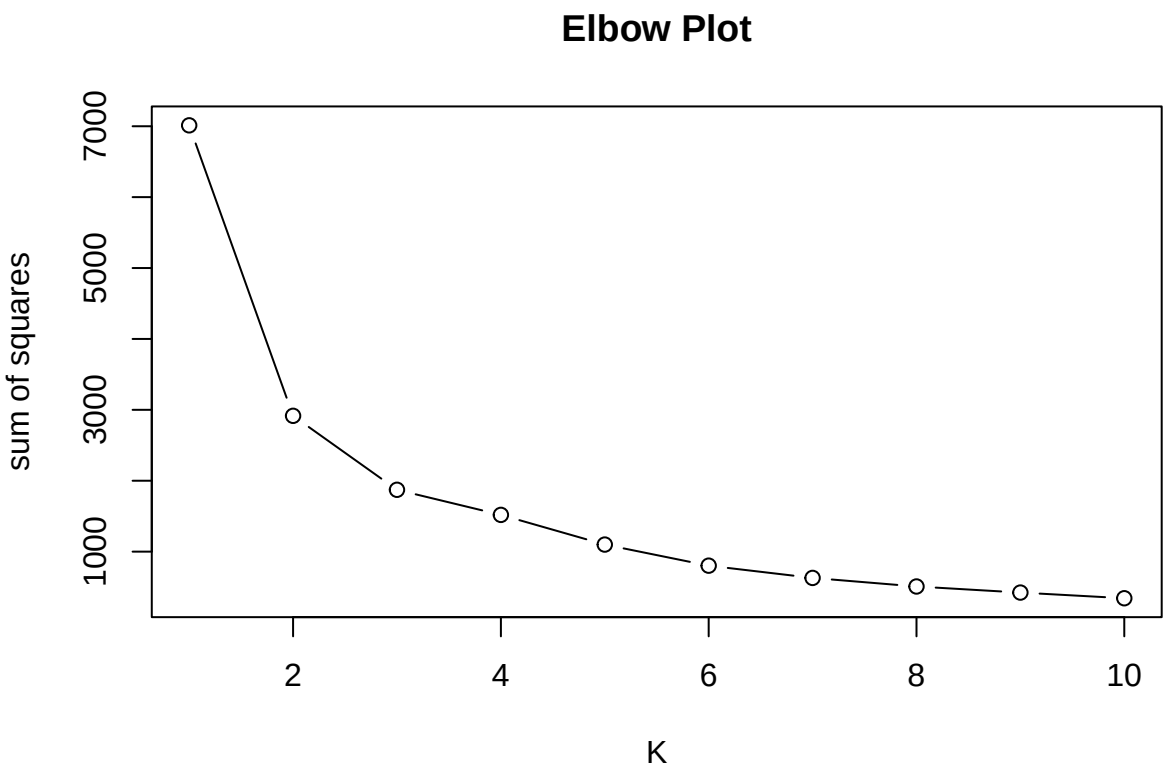
EXERCISE If we wanted to get fancy, we could find a lower dimesional projection (e.g. via principal components) and plot the scores of this, to see if the group structure is any clearer.

We want perform k-means clustering for the data but we have to decide on the number of clusters K. We can use an elbow plot to do so, using the following code.

```
#Home-made elbow plot function, need to give it data (rows=observations, columns=variables)
#and the maximum number of clusters to be fit K.max
plot.elbow<-function(data,K.max)
{
   n <- nrow(data)
   # If K.max is the maximum number of clusters we are looking for in the data
   ss<-rep(0,K.max)
   ss[1] <- (n - 1) * sum(apply(data, 2, var))
   # Now we calculate the sum of squares for each number of clusters from 2 to K.max
   for(i in 2:K.max)
     {
     ss[i] <- sum(kmeans(data,centers = i, nstart=3)$withinss)
   }
   plot(c(1:K.max),ss,type="b",main= "Elbow Plot", xlab="K", ylab="sum of squares")
}
```

Appling this function to the `life` data with a maximum of 10 clusters we get the following plot

```
plot.elbow(life,10)
```



Based on this, we take a closer look at the k=3 k-means solution (your choice of k may be different based on the plot).

```
# First extract the countrynames to use as labels
country <- row.names(life)
```

Fit the k-means model for 3 clusters (with at least 50 random starts)

```
life.k.3<-kmeans(life,3,nstart=50)
names(life.k.3)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

We can see that there are a large number of elements in the list produced by this function. The main ones of interest are `cluster`, which gives a vector of cluster assignments for all observations and `centers`, a matrix of centers for all clusters (each row represents a cluster, each column a variable). Looking at these here we get:

```
life.k.3$cluster
```

```
##              Algeria             Cameroon           Madagascar
##                    2                    1                    1
##            Mauritius              Reunion           Seychelles
##                    3                    3                    2
##      South Africa(C)      South Africa(W)              Tunisia
##                    3                    2                    3
```

```
##               Canada           Costa Rica        Dominican Rep
##                    2                    2                    2
##          El Salvador            Greenland              Grenada
##                    3                    3                    3
##            Guatemala             Honduras              Jamaica
##                    3                    3                    2
##               Mexico            Nicaragua               Panama
##                    3                    2                    2
##         Trinidad(62)        Trinidad (67)   United States (66)
##                    2                    2                    2
## United States (NW66)  United States (W66)   United States (67)
##                    3                    2                    2
##            Argentina                Chile             Columbia
##                    2                    3                    3
##              Ecuador
##                    3
```

```
life.k.3$centers
```

```
##         m0      m25  m50      m75       w0      w25      w50      w75
## 1 36.00000 29.50000 15.0 6.000000 38.00000 33.00000 18.50000  6.50000
## 2 65.06667 47.46667 24.4 8.800000 69.93333 50.33333 27.86667 10.53333
## 3 57.14286 42.64286 22.5 8.285714 61.78571 46.57143 25.71429 10.21429
```

Looking at the three cluster centres, we see that 2 of the clusters have higher life expectancy in general than the other and that these two are mainly different in terms of the first variable (life expectancy at birth).

The resulting clusters of countries can be found using the code:

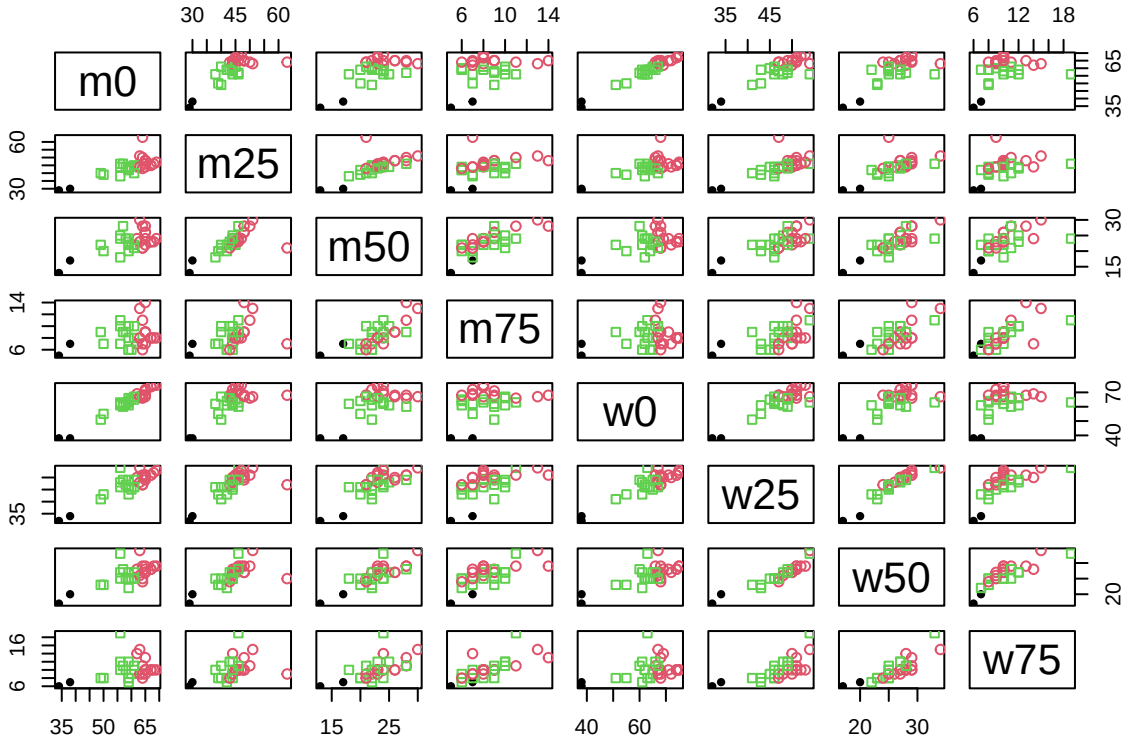
```
# Set K equal to the number of clusters we are looking at
K<-3
country.clus <- lapply(1:K, function(nc) country[life.k.3$cluster==nc])
country.clus
```

```
## [[1]]
## [1] "Cameroon"   "Madagascar"
##
## [[2]]
##  [1] "Algeria"             "Seychelles"          "South Africa(W)"
##  [4] "Canada"              "Costa Rica"          "Dominican Rep"
##  [7] "Jamaica"             "Nicaragua"           "Panama"
## [10] "Trinidad(62)"        "Trinidad (67)"       "United States (66)"
## [13] "United States (W66)" "United States (67)"  "Argentina"
##
## [[3]]
##  [1] "Mauritius"            "Reunion"              "South Africa(C)"
##  [4] "Tunisia"              "El Salvador"          "Greenland"
##  [7] "Grenada"              "Guatemala"            "Honduras"
## [10] "Mexico"               "United States (NW66)" "Chile"
## [13] "Columbia"             "Ecuador"
```

EXERCISE Try looking at the result for a different number of clusters (your choice).

If we want, we can also produce a scatterplot of pairs of the variables with clusters indicated by colour or point type

```
pairs(life,col=life.k.3$cluster,pch=life.k.3$cluster+19)
```

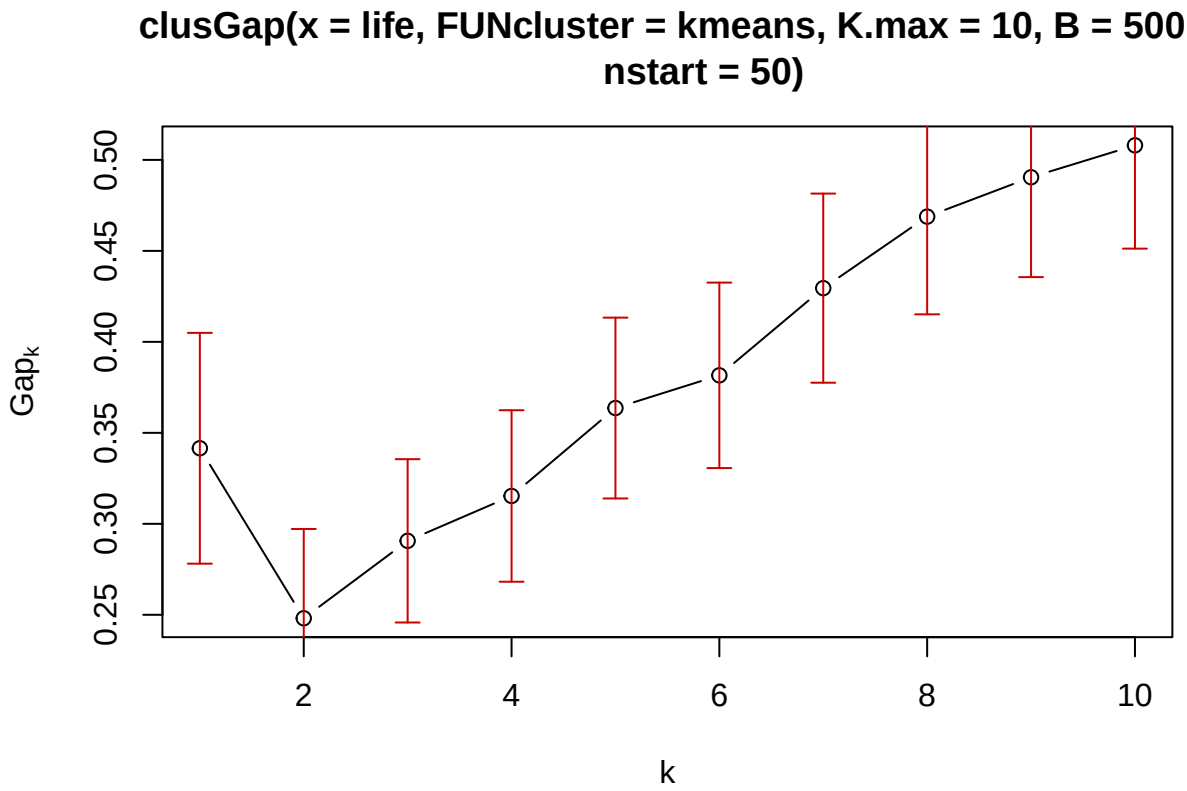


Gap Statistics for the life data

We can also use the gap statistic to try and inform our choice of number of clusters. Here we use 100 bootstrapped datasets to calculate the statistic which is then plotted.

```
library(cluster)
res.gap<-clusGap(life,FUNcluster=kmeans, nstart=50, K.max=10,B=500)
```

```
plot(res.gap)
```

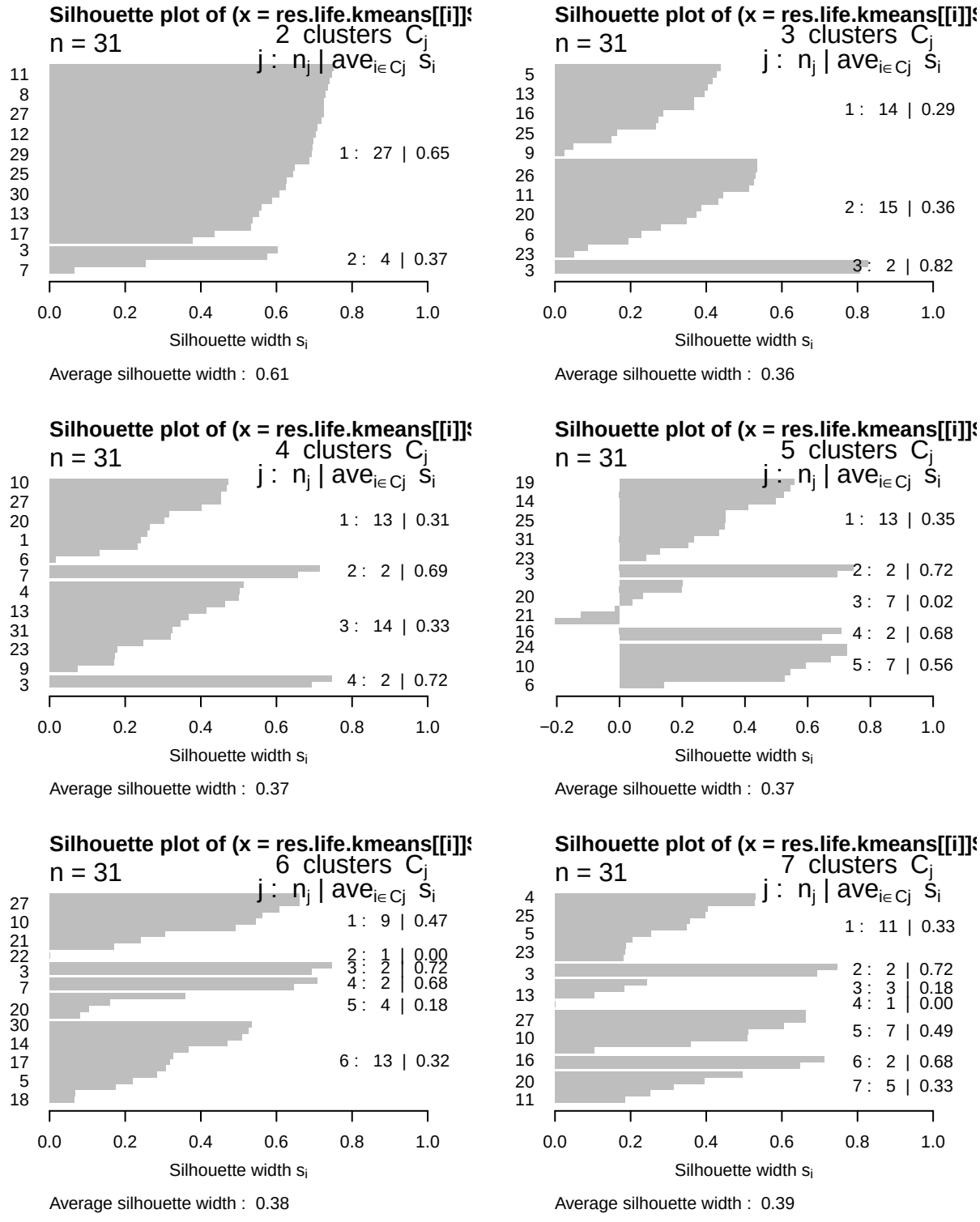


This would suggest no cluster structure in the data (K=1) based on the rule discussed in lectures.

Silhouette plot and statistic for the life data

We can also use the silhouette plot to see how well the clustering is doing and the average silhouette to choose the number of clusters for the data (larger is better).

```
# Here we use a loop to produce the plots and average silhouette widths for k from 2 to 7
par(mfrow=c(3,2))
# First we produce the distance matrix for the life data which is used in the silhouette function
life.dist<-dist(life)
# We create a list to store all the k-means results in case we want to look at them later
# Same for the silhouette width results
res.life.kmeans<-list()
silh.life.kmeans<-list()
for(i in 2:7)
{
  res.life.kmeans[[i]]<-kmeans(life,i,nstart=50)
  silh.life.kmeans[[i]]<-silhouette(res.life.kmeans[[i]]$cluster,life.dist)
  plot(silh.life.kmeans[[i]])
}
```



The average silhouette values strongly suggest 2 as the best value for the number of clusters (note: silhouette width can't be calculated for k=1).

Looking at the result for 2 only:

```
silh.life.kmeans[[2]]
```

```
##       cluster neighbor  sil_width
##  [1,]       1        2 0.62645043
##  [2,]       2        1 0.57512319
##  [3,]       2        1 0.60302361
##  [4,]       1        2 0.58717969
##  [5,]       1        2 0.37815561
##  [6,]       1        2 0.72573941
##  [7,]       2        1 0.06478819
##  [8,]       1        2 0.72952968
##  [9,]       1        2 0.53547469
## [10,]       1        2 0.69678069
## [11,]       1        2 0.74594187
## [12,]       1        2 0.70410228
## [13,]       1        2 0.55393799
## [14,]       1        2 0.64840345
## [15,]       1        2 0.71821163
## [16,]       2        1 0.25477383
## [17,]       1        2 0.43527145
## [18,]       1        2 0.73559292
## [19,]       1        2 0.62484245
## [20,]       1        2 0.69565318
## [21,]       1        2 0.74049572
## [22,]       1        2 0.53142629
## [23,]       1        2 0.68579790
## [24,]       1        2 0.72423191
## [25,]       1        2 0.64303670
## [26,]       1        2 0.70695192
## [27,]       1        2 0.72423191
## [28,]       1        2 0.75109330
## [29,]       1        2 0.69259529
## [30,]       1        2 0.60764867
## [31,]       1        2 0.56032283
## attr(,"Ordered")
## [1] FALSE
## attr(,"call")
## silhouette.default(x = res.life.kmeans[[i]]$cluster, dist = life.dist)
## attr(,"class")
## [1] "silhouette"
```

```
summary(silh.life.kmeans[[2]])
```

```
## Silhouette of 31 units in 2 clusters from silhouette.default(x = res.life.kmeans[[i]]$cluster, dist =
##  Cluster sizes and average silhouette widths:
##        27         4
## 0.6484852 0.3744272
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06479 0.56772 0.64840 0.61312 0.72122 0.75109
```

```
mean(silh.life.kmeans[[2]][,3])
```

```
## [1] 0.6131229
```

K-medoids clustering

This is very similar to k-means clustering except that the "prototypes" representing each cluster in this case are medoids, actual observations rather than means or centroids. We apply it using the `pam` function from the `cluster` package

```
life.p.3<-pam(life, 3)
names(life.p.3)
```

```
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

Again we have a list of objects. We are mainly interested in the `medoids` object which gives the medoids/prototypes/observations chosen to best represent the clusters and the `clustering` vector which gives the cluster assignments for all observations.

```
life.p.3$clustering
```

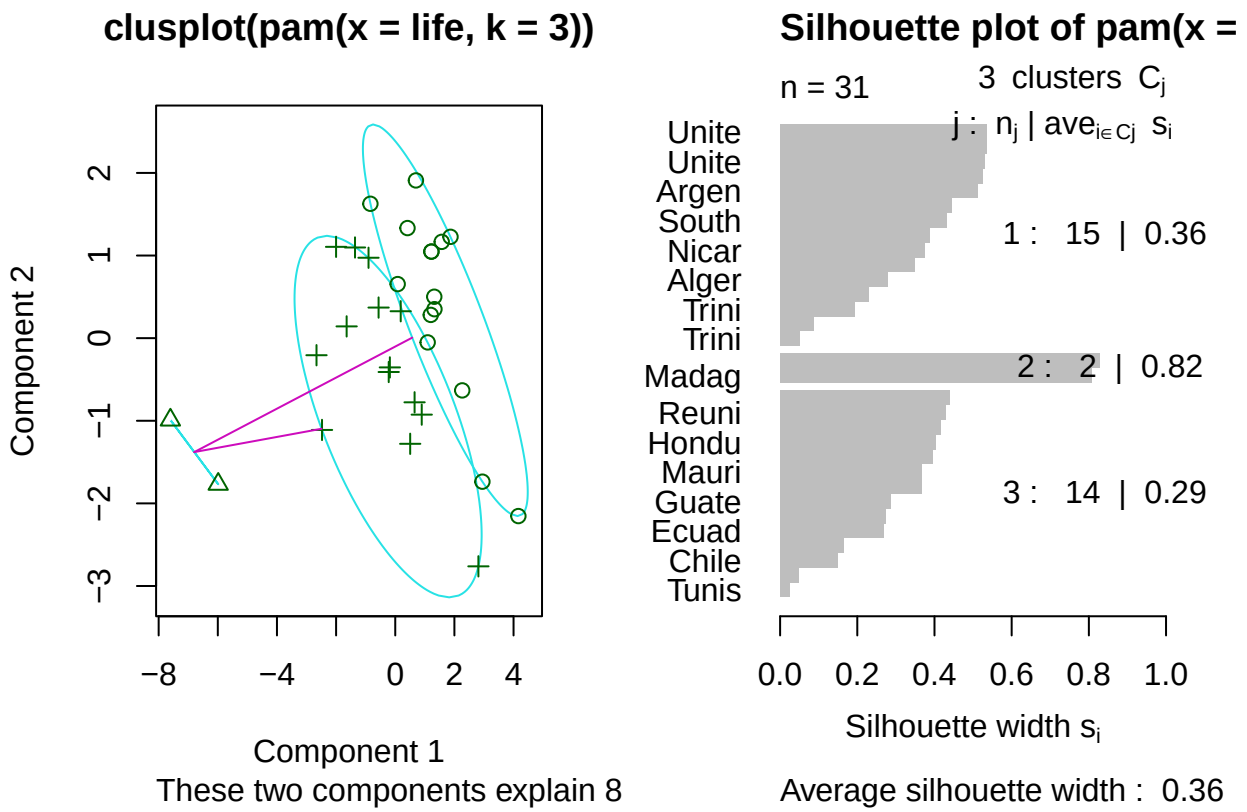
```
##              Algeria             Cameroon           Madagascar
##                    1                    2                    2
##            Mauritius              Reunion           Seychelles
##                    3                    3                    1
##      South Africa(C)      South Africa(W)              Tunisia
##                    3                    1                    3
##               Canada           Costa Rica        Dominican Rep
##                    1                    1                    1
##          El Salvador            Greenland              Grenada
##                    3                    3                    3
##            Guatemala             Honduras              Jamaica
##                    3                    3                    1
##               Mexico            Nicaragua               Panama
##                    3                    1                    1
##         Trinidad(62)        Trinidad (67)   United States (66)
##                    1                    1                    1
## United States (NW66)  United States (W66)   United States (67)
##                    3                    1                    1
##            Argentina                Chile             Columbia
##                    1                    3                    3
##              Ecuador
##                    3
```

```
life.p.3$medoids
```

```
##            m0 m25 m50 m75 w0 w25 w50 w75
## Argentina  65  46  24   9 71  51  28  10
## Madagascar 38  30  17   7 38  34  20   7
## Columbia   58  44  24   9 62  47  25  10
```

We can also plot these using the `plot` command

```
par(mfrow=c(1,2))
plot(life.p.3)
```




```
par(mfrow=c(1,1))
```

This produces a 2-d principal component projection scatterplot of the data with the medoids joined by pink lines and the clusters surronded by cyan ellipses (lines if there are only two points) and also a silhouette width plot.