# PUBPOL542: Computational Thinking for Governance Analytics

## Professor Jose Manuel Magallanes


### Hanna Peterson, MPA Candidate at the Evans School of Public Policy of Governance


## Introduction to R

The following code reads your team's final dataset from the link provided. It will also define the final data as CSV.

```
linkcsv="https://github.com/ComputationalThinkingGroup5/Merge/raw/master/MergedData.csv"
FinalData=read.csv(linkcsv)
```

You can view the data types in your dataset using the following str() code:

```
str(FinalData)
```

```
## 'data.frame':    162 obs. of  4 variables:
##  $ country           : chr  "Cambodia" "Niger" "Laos" "Malta" ...
##  $ percentunemployment: num  0.3 0.3 0.7 0.78 0.8 0.99 1 1.7 1.8 2 ...
##  $ pct_GDP_exp       : num  2.2 3.5 2.9 4.8 4.8 4.1 2.9 4.5 2.8 2.7 ...
##  $ percentbirthrate  : num  1.34 3.65 1.46 0.75 -0.29 0.26 3.36 1.67 2.31 0.79 ...
```

The following code will show you the names of the variables in your dataset.

```
names(FinalData)
```

```
## [1] "country"             "percentunemployment" "pct_GDP_exp"
## [4] "percentbirthrate"
```

Determine the value of a cell by typing the indexes of where it is located:

```
FinalData[2,3]
```

```
## [1] 3.5
```

The following code will display a row:

```
FinalData[2,]
```

```
##   country percentunemployment pct_GDP_exp percentbirthrate
## 2   Niger                 0.3         3.5             3.65
```

The following code will display the specified columns, and the c() command prepares a vector of indexes.

```
FinalData[2,c("country", "pct_GDP_exp","percentunemployment" )]
```

```
##   country pct_GDP_exp percentunemployment
## 2   Niger         3.5                 0.3
```

The following command defines the first condition as the country with the highest chosen variable, this case being the highest GDP expenditure.

```
condition1=FinalData$pct_GDP_exp==max(FinalData$pct_GDP_exp)
FinalData[condition1,]
```

```
##    country percentunemployment pct_GDP_exp percentbirthrate
## 19    Cuba                 2.6        12.8            -0.23
```

This command defines which country has the highest GDP.

```
FinalData[condition1, "country"]
```

```
## [1] "Cuba"
```

```
FinalData[FinalData$popgrowth.rate<0, 'country']
```

```
## character(0)
```

The next command defines a new dataset, "shrinking population", as the set of countries which have a negative value of the "popgrowth.rate" variable.

```
shrinkingpop=FinalData[FinalData$popgrowth.rate<0,]
```

```
shrinkingpop[shrinkingpop$percentunemployment==max(shrinkingpop$percentunemployment),]
```

```
## Warning in max(shrinkingpop$percentunemployment): no non-missing arguments to
## max; returning -Inf
```

```
## [1] country             percentunemployment pct_GDP_exp
## [4] percentbirthrate
## <0 rows> (or 0-length row.names)
```

The following condition defines condition2 as a new dataset from the shrinking population dataset. Condition2 is defined as the country which has the maximum percentage of unemployment within all countries that have a shrinking population size.

```
condition2=shrinkingpop$percentunemployment==max(shrinkingpop$percentunemployment)
```

```
## Warning in max(shrinkingpop$percentunemployment): no non-missing arguments to
## max; returning -Inf
```

```
shrinkingpop[condition2,]
```

```
## [1] country             percentunemployment pct_GDP_exp
## [4] percentbirthrate
## <0 rows> (or 0-length row.names)
```

```
condition2
```

```
## logical(0)
```

Install pipes and dplyr.

```
library(magrittr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Define dfClus as the final dataset, including the specified columns. Explore the variables you will use for clustering.

```
dfClus=FinalData[,c("percentunemployment", "percentbirthrate", "pct_GDP_exp")]
summary(dfClus)
```

```
##  percentunemployment percentbirthrate   pct_GDP_exp
##  Min.   : 0.300      Min.   :-2.4600   Min.   : 1.200
##  1st Qu.: 3.705      1st Qu.: 0.3050   1st Qu.: 3.200
```

```
##  Median : 6.525      Median : 0.8300   Median : 4.350
##  Mean   : 9.609      Mean   : 0.9519   Mean   : 4.474
##  3rd Qu.:10.725      3rd Qu.: 1.5825   3rd Qu.: 5.375
##  Max.   :77.000      Max.   : 3.6500   Max.   :12.800
```

Rescale units if needed into new variable:

```
dfClus=scale(dfClus)
summary(dfClus)
```

```
##  percentunemployment percentbirthrate   pct_GDP_exp
##  Min.   :-0.9083     Min.   :-3.3149   Min.   :-1.85867
##  1st Qu.:-0.5761     1st Qu.:-0.6285   1st Qu.:-0.72328
##  Median :-0.3009     Median :-0.1184   Median :-0.07044
##  Mean   : 0.0000     Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.1089     3rd Qu.: 0.6127   3rd Qu.: 0.51145
##  Max.   : 6.5756     Max.   : 2.6215   Max.   : 4.72658
```

## R for Clustering

```
link='https://github.com/ComputationalThinkingGroup5/Merge/raw/master/MergedData.csv'
myFile=url(link)
fromPy=read.csv(file=myFile)
row.names(fromPy)=NULL
```

Rename subset indexes and verify what your input is:

```
row.names(dfClus)=fromPy$country
head(dfClus)
```

```
##          percentunemployment percentbirthrate pct_GDP_exp
## Cambodia          -0.9082903        0.3771237  -1.2909781
## Niger             -0.9082903        2.6215132  -0.5529760
## Laos              -0.8692609        0.4937153  -0.8935924
## Malta             -0.8614550       -0.1961187   0.1850262
## Belarus           -0.8595035       -1.2065798   0.1850262
## Thailand          -0.8409645       -0.6722013  -0.2123596
```

Set random seed: this number must correspond with any group members' seeds. This ensures replicability of results.

```
set.seed(999)
```

Decide distance method and compute distance matrix:

```
library(cluster)
dfClus_D=cluster::daisy(x=dfClus)
```

Partitioning Technique 1. Apply function: you need to indicate the amount of clusters required

```
NumCluster=4
res.pam=pam(x=dfClus_D,
            k= NumCluster,
            cluster.only = F)
```

2. Clustering results: 2.1 Add results to original data frame:

```
fromPy$pam=as.factor(res.pam$clustering)
```

2.2 Query data frame as needed: Example 1:

```r
fromPy[fromPy$pam==1, 'country']
```

```
##  [1] "Cambodia"                 "Niger"
##  [3] "Laos"                     "Benin"
##  [5] "Vanuatu"                  "Madagascar"
##  [7] "Macau"                    "Monaco"
##  [9] "Singapore"                "Guatemala"
## [11] "Liechtenstein"            "Papua New Guinea"
## [13] "Guinea"                   "Rwanda"
## [15] "Liberia"                  "British Virgin Islands"
## [17] "Seychelles"               "Vietnam"
## [19] "Malaysia"                 "Mexico"
## [21] "Bahrain"                  "Burma"
## [23] "Cameroon"                 "Bangladesh"
## [25] "Saint Kitts and Nevis"    "Kazakhstan"
## [27] "Sri Lanka"                "Ireland"
## [29] "Azerbaijan"               "Samoa"
## [31] "Indonesia"                "Luxembourg"
## [33] "Paraguay"                 "Pakistan"
## [35] "Panama"                   "Nicaragua"
## [37] "Comoros"                  "Peru"
## [39] "Angola"                   "Brunei"
## [41] "Central African Republic" "Bermuda"
## [43] "El Salvador"              "Mali"
## [45] "Mongolia"                 "San Marino"
## [47] "India"                    "Qatar"
## [49] "Cote d'Ivoire"            "Uganda"
## [51] "Lebanon"                  "Turks and Caicos Islands"
## [53] "Mauritania"               "Tanzania"
## [55] "Turkmenistan"             "Iran"
## [57] "Ghana"                    "Turkey"
## [59] "Jordan"                   "Saint Lucia"
## [61] "Grenada"
```

Example 2:

```r
fromPy[fromPy$country=="Peru", 'pam']
```

```
## [1] 1
## Levels: 1 2 3 4
```

2.2 Report: table of clusters

```r
table(fromPy$pam)
```

```
##
##  1  2  3  4
## 61 41 42 18
```

3. Evaluate Results 3.1 Report: average silhouettes

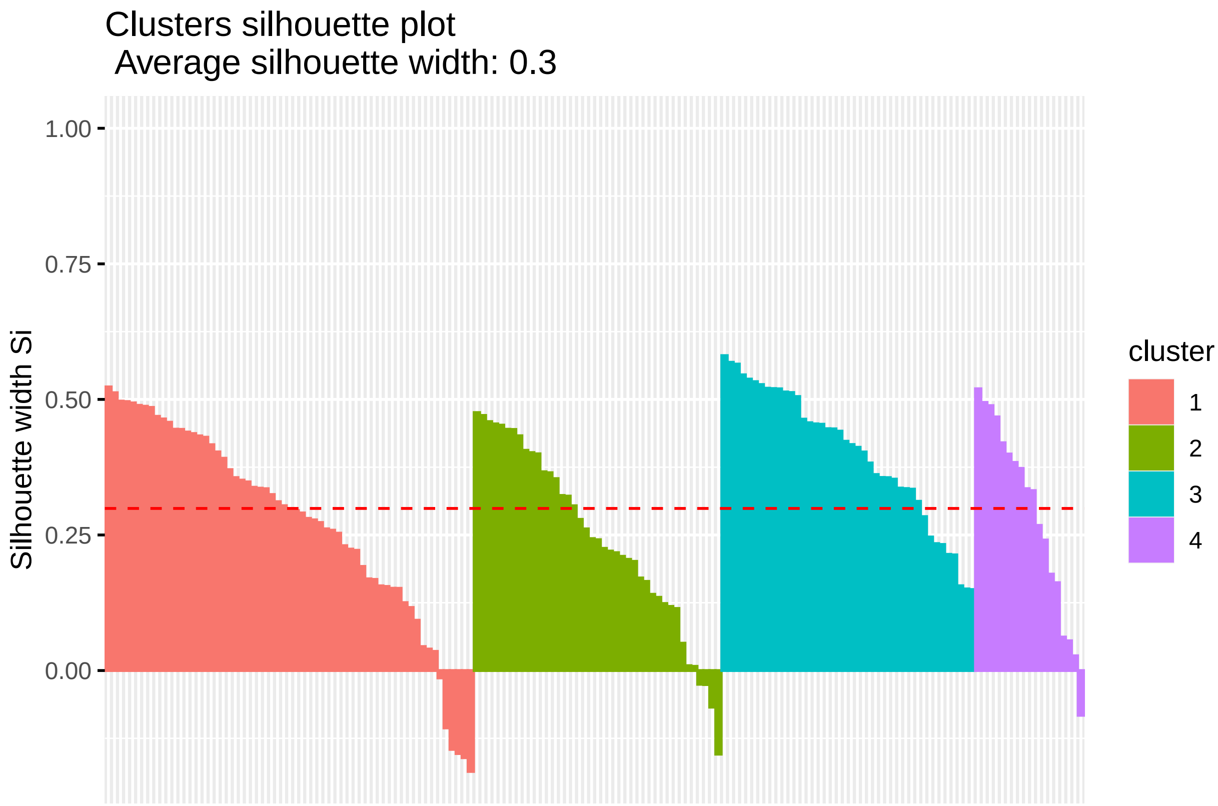
```r
library(factoextra)
```

```
## Loading required package: ggplot2

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

```r
fviz_silhouette(res.pam)
```

```
##   cluster size ave.sil.width
## 1       1   61          0.27
## 2       2   41          0.24
## 3       3   42          0.40
## 4       4   18          0.28
```




3.2 Detecting Anomalies: a. Save individual silhouettes:

```r
pamEval=data.frame(res.pam$silinfo$widths)
head(pamEval)
```

```
##                          cluster neighbor sil_width
## British Virgin Islands         1        3 0.5230789
## Pakistan                       1        2 0.5123383
## Rwanda                         1        2 0.4969379
## Comoros                        1        3 0.4960402
## Turks and Caicos Islands       1        2 0.4935898
## Guatemala                      1        3 0.4890850
```

b. Request negative silhouettes: these are the ones which are poorly clustered.

```r
pamEval[pamEval$sil_width<0,]
```

```
##            cluster neighbor   sil_width
## San Marino       1        3 -0.01356122
## Grenada          1        3 -0.10578269
## Turkey           1        3 -0.14540630
```

```
## Saint Lucia          1        3 -0.15309505
## Seychelles           1        3 -0.16069755
## Samoa                1        3 -0.18609387
## United Kingdom       2        3 -0.02530168
## Malta                2        3 -0.02585179
## Colombia             2        1 -0.06746643
## France               2        3 -0.15413921
## Zambia               4        1 -0.08251775
```

Hierarchizing: agglomerative 1. Apply function:

```
library(factoextra)

res.agnes=hcut(dfClus_D,
              k= NumCluster, isdiss=T,
              hc_func='agnes',
              hc_method = "ward.D2")
```

2. Clustering Results: 2.1 Add results to original data frame:

```
fromPy$agn=as.factor(res.agnes$cluster)
```

2.2 Query data frame as needed: Example 1:

```
fromPy[fromPy$agn==1, 'country']
```

```
##  [1] "Cambodia"                 "Niger"
##  [3] "Laos"                     "Benin"
##  [5] "Vanuatu"                  "Madagascar"
##  [7] "Guatemala"                "Papua New Guinea"
##  [9] "Guinea"                   "Rwanda"
## [11] "Liberia"                  "British Virgin Islands"
## [13] "Cameroon"                 "Luxembourg"
## [15] "Pakistan"                 "Angola"
## [17] "Brunei"                   "Central African Republic"
## [19] "Togo"                     "Mali"
## [21] "Cote d'Ivoire"            "Uganda"
## [23] "Turks and Caicos Islands" "Mauritania"
## [25] "Tanzania"                 "Ghana"
## [27] "Zambia"
```

Example 2:

```
fromPy[fromPy$country=="Peru",'agn']
```

```
## [1] 3
## Levels: 1 2 3 4
```

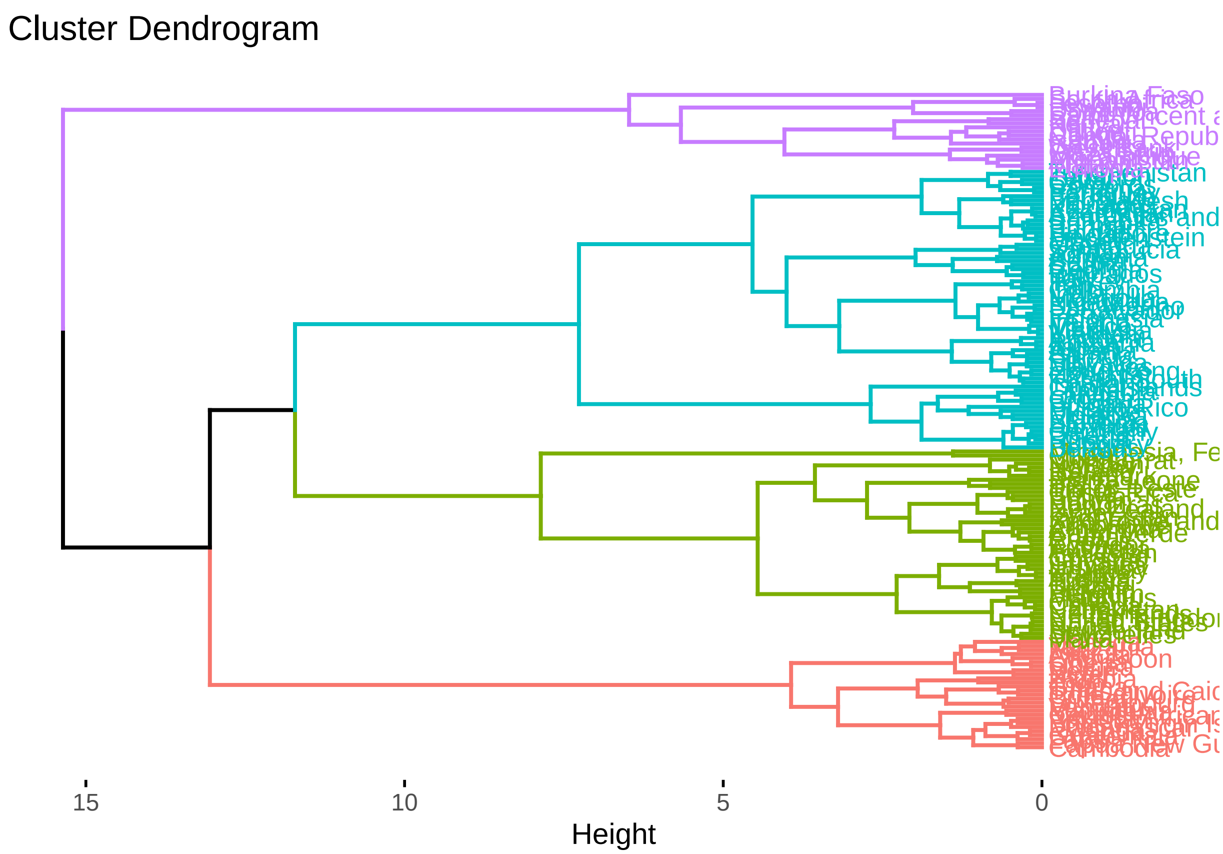
2.2 Report: Table of Clusters Reporting results:

```
table(fromPy$agn)
```

```
##
##  1  2  3  4
## 27 47 69 19
```

3. Evaluate Results 3.1a Report: Dendogram

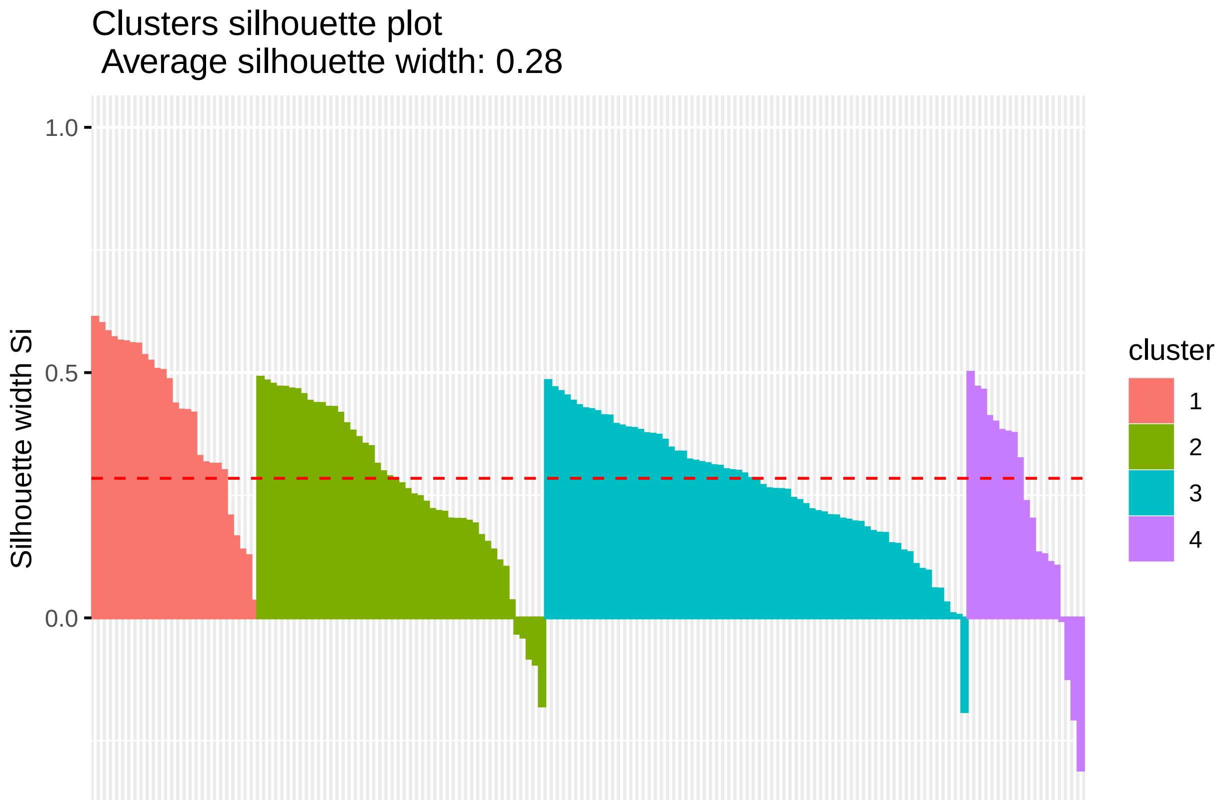
```
fviz_dend(res.agnes,k=NumCluster, cex = 0.7, horiz= T)
```




3.1b Report: Average silhouettes

```
library(factoextra)
fviz_silhouette(res.agnes)
```

```
##   cluster size ave.sil.width
## 1       1   27          0.41
## 2       2   47          0.27
## 3       3   69          0.27
## 4       4   19          0.21
```




3.2 Report: Detecting Anomalies a. Saving silhouettes:

```
agnEval=data.frame(res.agnes$silinfo$widths)
head(agnEval)
```

```
##            cluster neighbor sil_width
## Cameroon         1        3 0.6128457
## Liberia          1        3 0.6001858
## Guinea           1        3 0.5836463
## Tanzania         1        3 0.5713340
## Madagascar       1        3 0.5645315
## Angola           1        2 0.5629379
```

b. Request negative silhouettes:

```
agnEval[agnEval$sil_width<0,]
```

```
##                                  cluster neighbor   sil_width
## Jamaica                                2        3 -0.03132904
## Uruguay                                2        3 -0.03924979
## Curacao                                2        3 -0.08221345
## Mauritius                              2        3 -0.09456060
## Seychelles                             2        3 -0.17948812
## Comoros                                3        1 -0.19108640
## Malawi                                 4        1 -0.00562688
## Dominica                               4        2 -0.12398589
## Ethiopia                               4        1 -0.20616070
## Saint Vincent and the Grenadines       4        2 -0.31033947
```

Hierarchizing: divisive 1. Apply function: you need to indicate amount of clusters required. Install factoextra.

```
library(factoextra)

res.diana= hcut(dfClus_D, k = NumCluster,
                hc_func='diana',
                hc_method = "ward.D")
```

2. Clustering Results: 2.1 Adding results to original data frame:

```
fromPy$dia=as.factor(res.diana$cluster)
```

2.2 Query data frame as needed: Example 1:

```
fromPy[fromPy$dia==1, 'country']
```

```
##  [1] "Cambodia"                 "Niger"
##  [3] "Laos"                     "Benin"
##  [5] "Vanuatu"                  "Madagascar"
##  [7] "Macau"                    "Monaco"
##  [9] "Singapore"                "Guatemala"
## [11] "Japan"                    "Liechtenstein"
## [13] "Papua New Guinea"         "Guinea"
## [15] "Rwanda"                   "Liberia"
## [17] "British Virgin Islands"   "Hong Kong"
## [19] "Romania"                  "Vietnam"
## [21] "Malaysia"                 "Mexico"
## [23] "Bahrain"                  "Andorra"
## [25] "Burma"                    "Cameroon"
## [27] "Bangladesh"               "Fiji"
## [29] "Saint Kitts and Nevis"    "Kazakhstan"
## [31] "Sri Lanka"                "Ireland"
## [33] "Azerbaijan"               "Samoa"
## [35] "Indonesia"                "Luxembourg"
## [37] "Paraguay"                 "Albania"
## [39] "Pakistan"                 "Panama"
## [41] "Nicaragua"                "Comoros"
## [43] "Peru"                     "Angola"
## [45] "Brunei"                   "Central African Republic"
## [47] "Bermuda"                  "El Salvador"
## [49] "Mali"                     "Mongolia"
## [51] "San Marino"               "India"
## [53] "Qatar"                    "Cote d'Ivoire"
## [55] "Uganda"                   "Lebanon"
## [57] "Turks and Caicos Islands" "Mauritania"
## [59] "Tanzania"                 "Colombia"
## [61] "Turkmenistan"             "Georgia"
## [63] "Iran"                     "Ghana"
## [65] "Turkey"                   "Armenia"
## [67] "Jordan"                   "Saint Lucia"
## [69] "Grenada"
```

Example 2:

```
fromPy[fromPy$country=="Peru", 'dia']
```

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

```
## Levels: 1 2 3 4
```

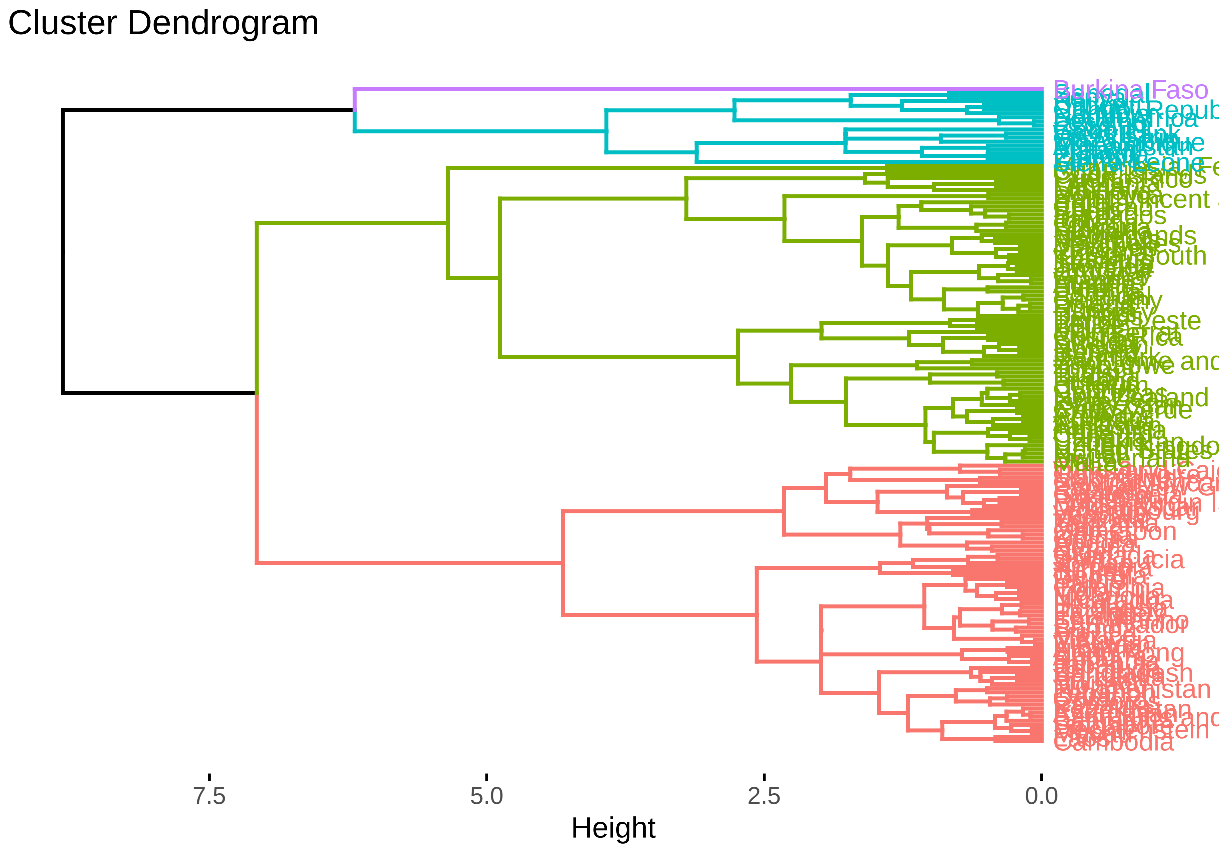
2.3 Report: Table of Clusters Reporting results:

```r
table(fromPy$dia)
```

```
##
##  1  2  3  4
## 69 74 18  1
```

Evaluating Results: 3.1a Report: Dendogram

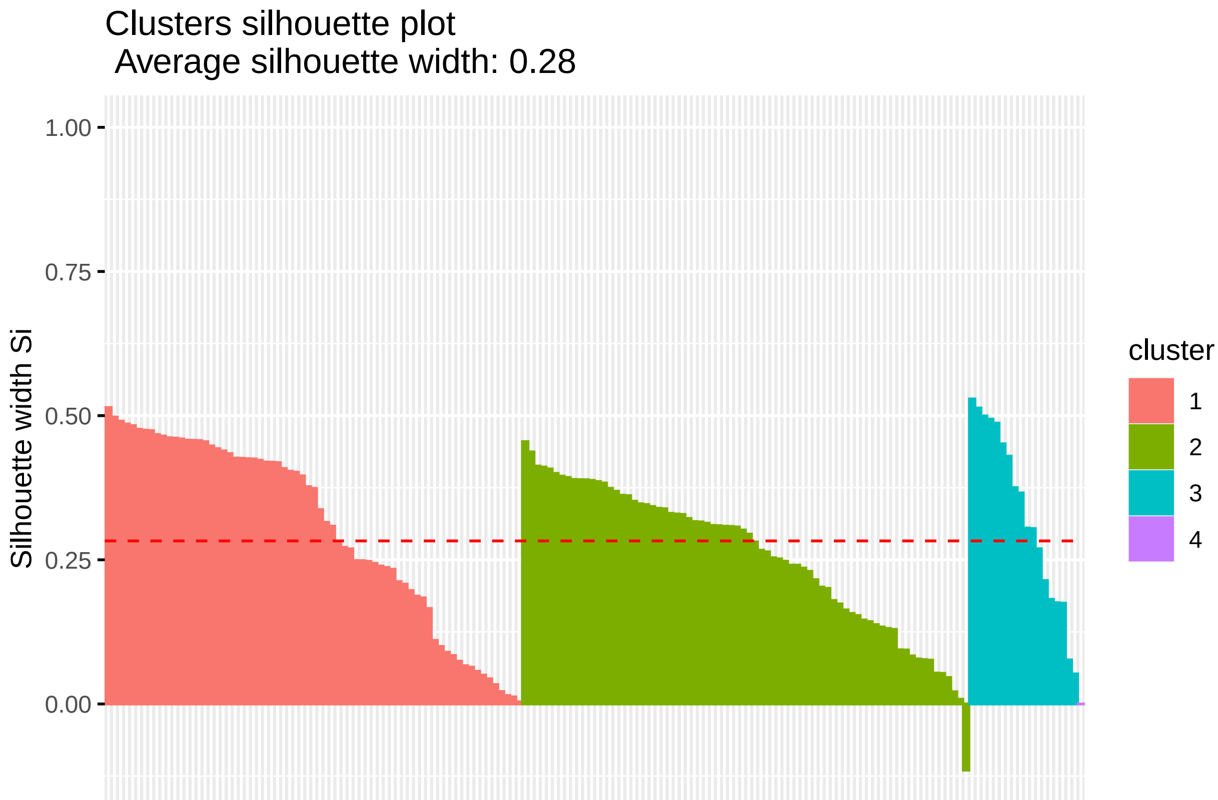
```r
fviz_dend(res.diana, k=NumCluster, cex = 0.7, horiz = T)
```




3.1b Report: Average silhouettes. Install factoextra.

```r
library(factoextra)
fviz_silhouette(res.diana)
```

```
##   cluster size ave.sil.width
## 1       1   69          0.30
## 2       2   74          0.26
## 3       3   18          0.33
## 4       4    1          0.00
```




3.2 Report: Detecting anomolies Saving silhouettes:

```
diaEval=data.frame(res.diana$silinfo$widths)
head(diaEval)
```

```
##                          cluster neighbor sil_width
## Comoros                        1        2 0.5140884
## Papua New Guinea               1        2 0.4977189
## British Virgin Islands         1        2 0.4905131
## Qatar                          1        2 0.4855992
## Pakistan                       1        2 0.4827332
## Cambodia                       1        2 0.4764699
```

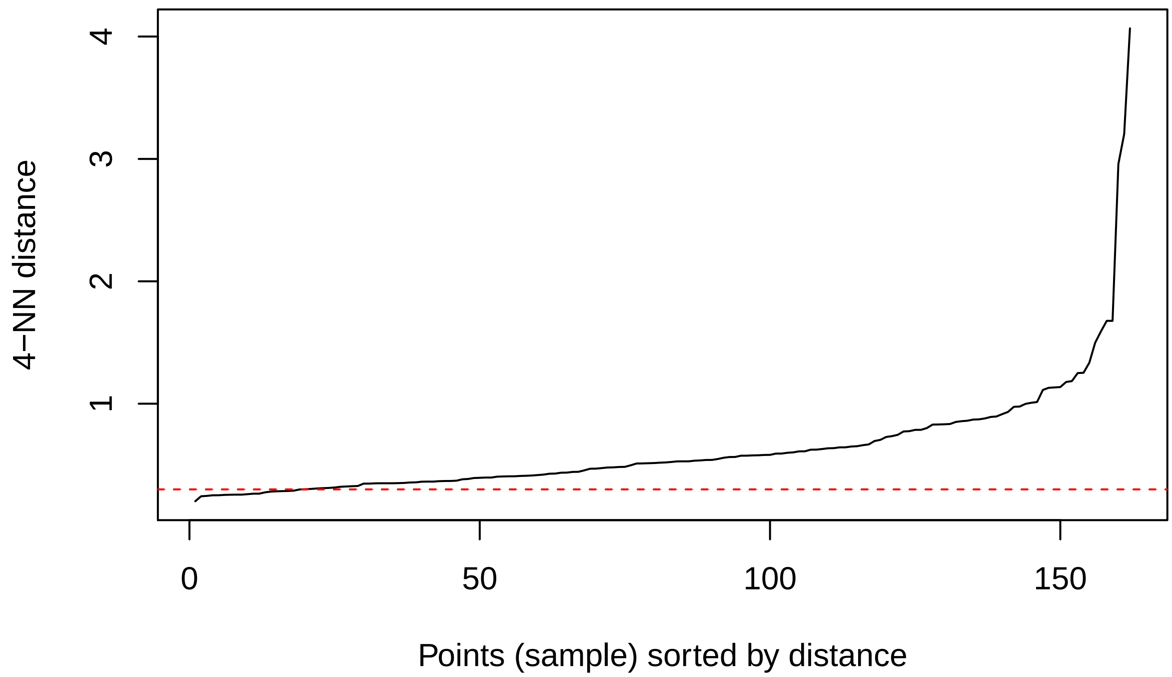
Request negative silhouettes:

```
diaEval[diaEval$sil_width<0,]
```

```
##      cluster neighbor sil_width
## Togo       2        1  -0.11473
```

Density-based clustering Input the distance and the minimal number of neighbors that form a cluster.

```
library(dbscan)
minNeighs=4
kNNdistplot(dfClus_D, k = minNeighs)
abline(h=0.3, col = "red", lty=2)
```




Format the table

```
distance=0.3
res.db=dbscan::dbscan(dfClus_D,
                      eps=distance,
                      minPts=minNeighs)
```

Report: How many clusters were produced, and how many outliers are there?

```
res.db
```

```
## DBSCAN clustering for 162 objects.
## Parameters: eps = 0.3, minPts = 4
## The clustering contains 7 cluster(s) and 113 noise points.
##
##   0   1   2   3   4   5   6   7
## 113   6   9   9   7  10   4   4
##
## Available fields: cluster, eps, minPts
```

Save results:

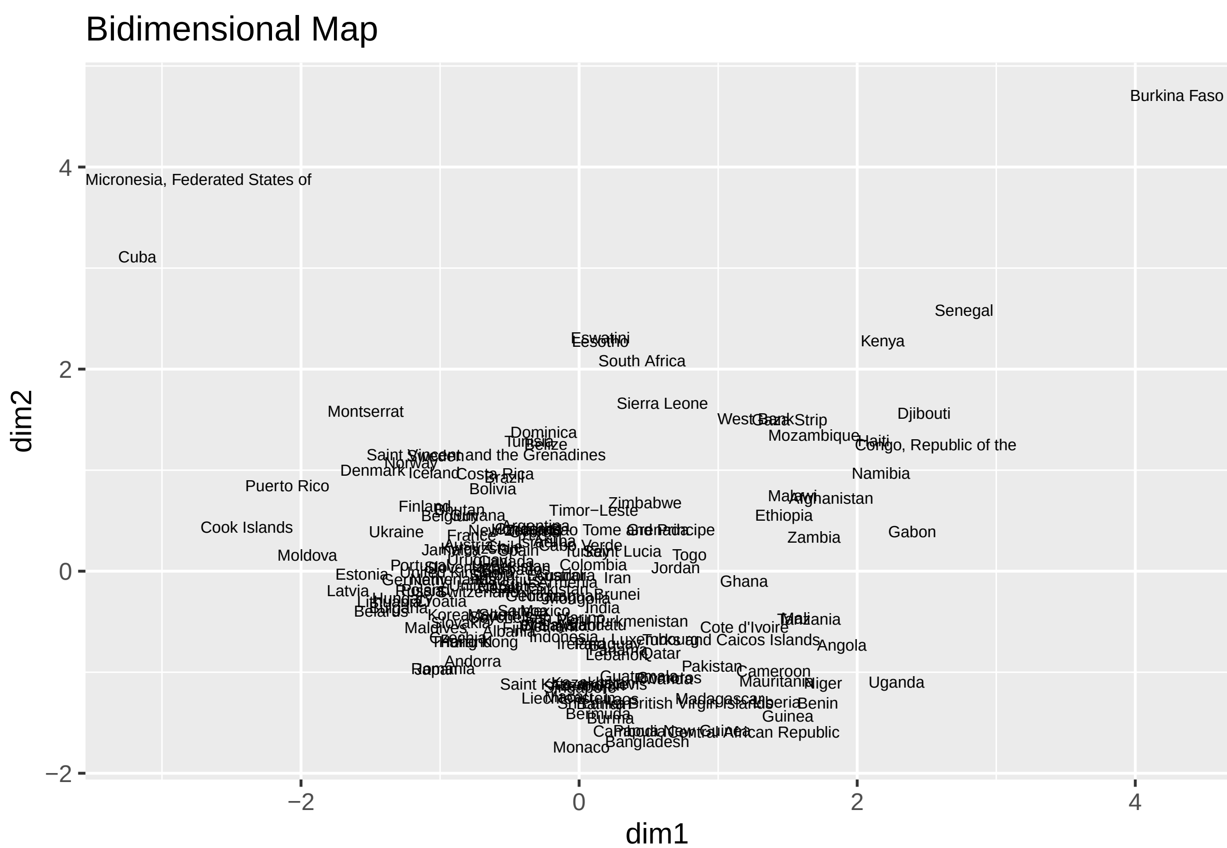
```
fromPy$db=as.factor(res.db$cluster)
```

Comparing clustering Prepare a bidimensional map:

```
projectedData=cmdscale(dfClus_D, k=2)
fromPy$dim1 = projectedData[,1]
fromPy$dim2 = projectedData[,2]
```

See the data plotted/mapped:

```r
base= ggplot(data=fromPy,
             aes(x=dim1, y=dim2,
                 label=country))
base + labs(title= "Bidimensional Map") + geom_text(size=2)
```




Plot results from PAM:

```r
Cluster1=base + labs(title = "Cluster1") + geom_point(size = 2,
                                                      aes(color=agn),
                                                      show.legend = F)
```

Plot results from hierarchial AGNES:

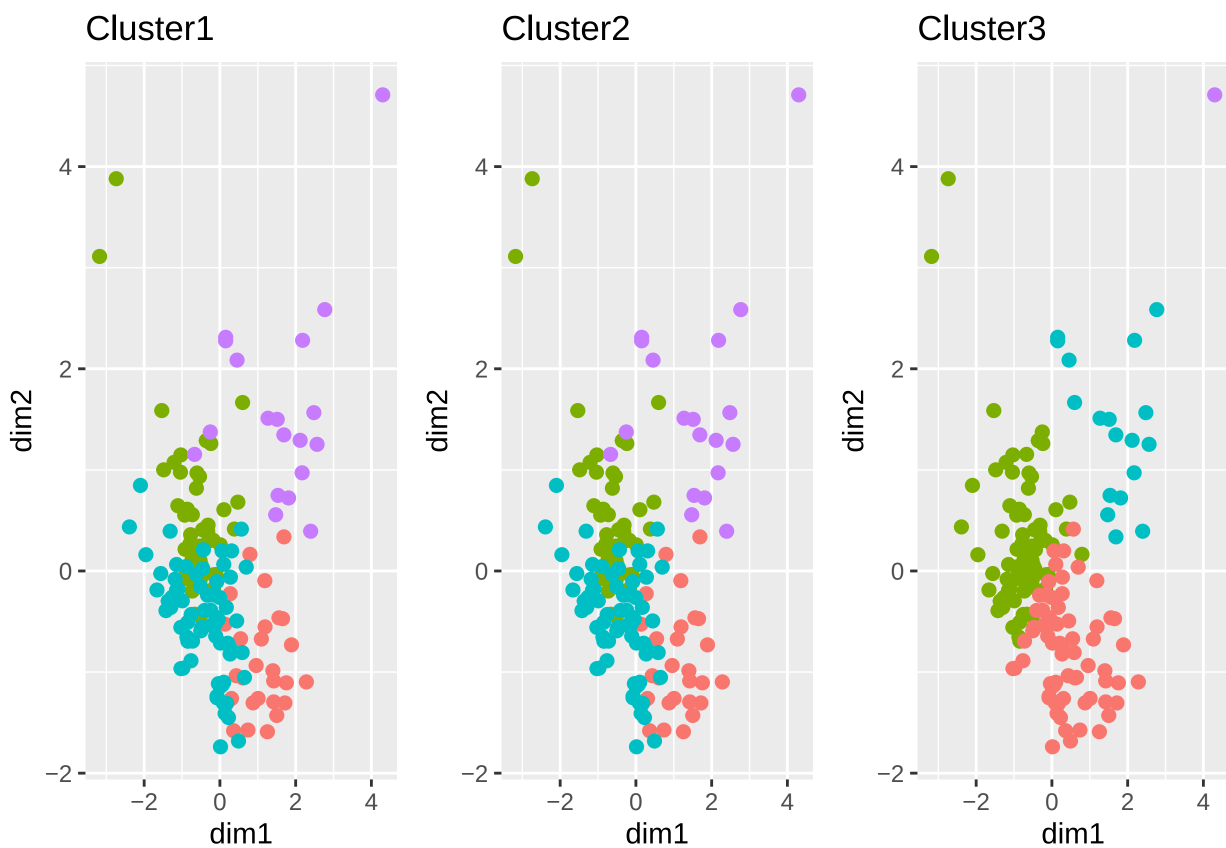
```r
Cluster2=base + labs(title = "Cluster2") + geom_point(size=2,
                                                        aes(color=agn),
                                                        show.legend = F)
```

Plot results from hierarchical DIANA:

```r
Cluster3=base + labs(title = "Cluster3") + geom_point(size=2,
                                                        aes(color=dia),
                                                        show.legend = F)
```

Visually compare the data.

```r
library(ggpubr)
ggarrange(Cluster1, Cluster2, Cluster3, ncol = 3)
```



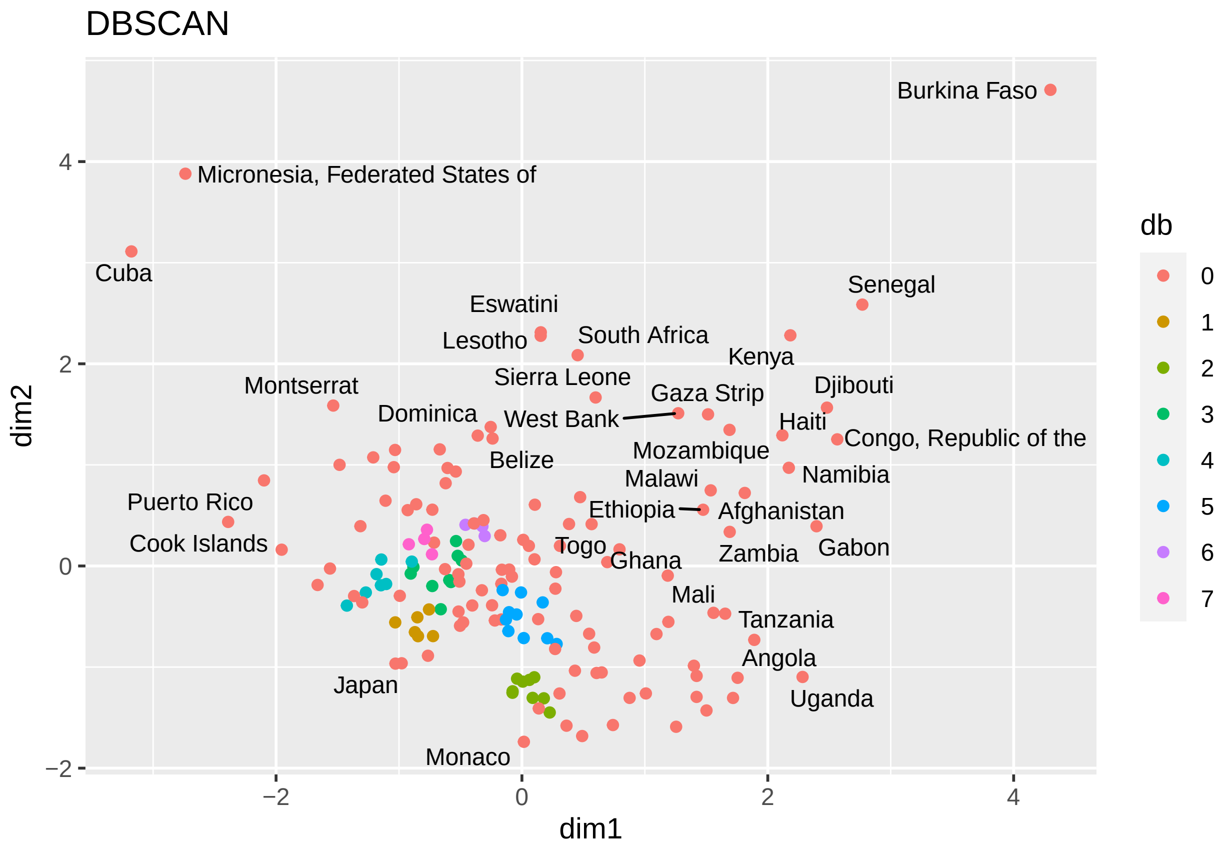

Plot results:

```
dbPlot = base + labs(title = "DBSCAN") + geom_point(aes(color=db),
                                                   show.legend = T)
```

Annotate data:

```
library(ggrepel)
dbPlot + geom_text_repel(size=3, aes(label=country))
```

```
## Warning: ggrepel: 128 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

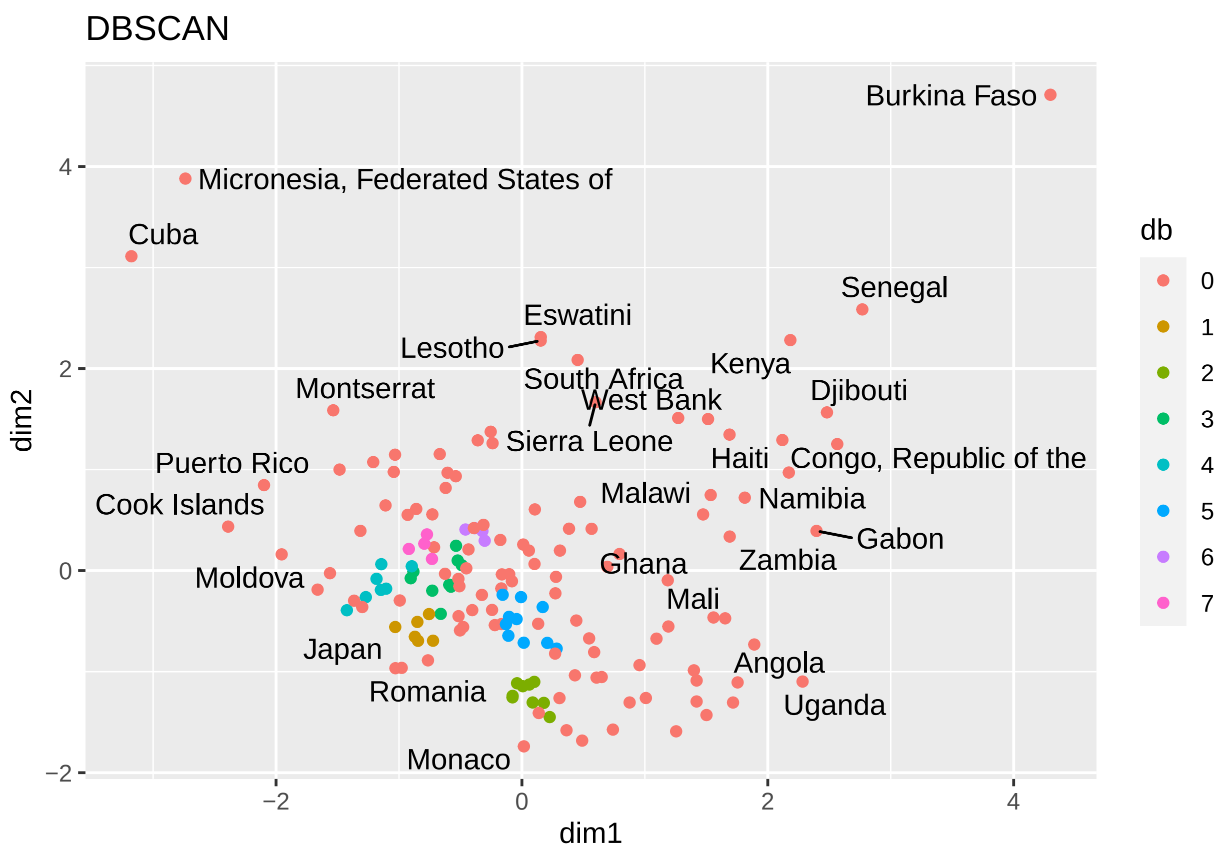



Annotate outliers of the dataset:

```
LABEL=ifelse(fromPy$db==0, fromPy$country, "")

dbPlot + geom_text_repel(aes(label=LABEL))
```

```
## Warning: ggrepel: 85 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```


DBSCAN
Burkina Faso
Micronesia, Federated States of
Cuba
Senegal
Eswatini
Lesotho
South Africa
Kenya
Montserrat
West Bank
Djibouti
Sierra Leone
Puerto Rico
Haiti
Congo, Republic of the
Cook Islands
Malawi
Namibia
Gabon
Moldova
Ghana
Zambia
Mali
Japan
Angola
Romania
Uganda
Monaco
dim1
dim2
db
0
1
2
3
4
5
6
7
4
2
0
−2
−2
0
2
4