Klasteryzacja

Maciej Urbaniak 200842

May 7, 2018

1 Wstęp

K Means Clustering to algorytm uczenia nienadzorowanego który dzieli dane bazując na ich podobieństwie. Znajduje pewne zależności w danych nie zawsze przydatne. Działa on według poniższych kroków.

1. Określ k czyli ile ma zostać stworzonych klastrów.
2. Wybierz losowo k obiektów z zbioru danych jako początkowe centra klastrów.
3. Przyporządkuj punkt do najbliższego centroidu wykorzystując odległość euklidesową.
4. Dla każdego klastru ponownie oblicz centroid biorąc średnią ze wszystkich punktów w tym klastrze.
5. Powtarzaj kroki 3 i 4 aż centroidy nie będą się zmieniać albo osiągnięta zostanie maksymalna ilość iteracji. (Dla R standardowa wartość to 10)

Metoda PAM (Partitioning Around Medoids) wykorzystuje natomiast medoidy czyli obiekty występujące w zbiorze danych. Jest mniej wrażliwy na szum i elementy odstające. Wymaga podobnie jak poprzedni algorytm zdefiniowania na ile klastrów mają zostać podzielone dane. Poniżej przedstawiono kolejne kroki tego algorytmu.

1. Wybierz k obiektów które zostaną medoidami albo jeśli zostały wcześniej podane nie ma potrzeby ich losowania.
2. Oblicz dissimilarity matrix w przypadku gdy nie została podana.
3. Przyporządkuj każdy obiekt do jego najbliższego medoidu.
4. Dla każdego klastra poszukaj czy jakikolwiek obiekt zmniejsza średni dissimilarity coefficient, jeśli tak wybierz ten o największym wpływie jako medioid dla tego klastra.
5. Jeśli chociaż jeden medioid się zmienił wróć do kroku 3 w przeciwnym wypadku zakończ algorytm.

2 Parametry

Opis czterech wybranych miar określających jak przebiegł proces klasteryzacji.
Average.between: średni dystans pomiędzy klastrami, im większy tym lepiej.
Average.within: średni dystans między punktami wewnątrz klastrów, im mniejszy tym lepiej.
Silhouette Coefficient obliczany jest według poniższego wzoru:

$$s = \frac{b - a}{max(a, b)}$$

Gdzie a to średni dystans między próbką a innymi punktami w tej samej klasie. A b średnia odległość od wszystkich innych punktów w najbliższym klastrze. Wynik zawiera się między -1 a +1. Wyniki w pobliżu zera świadczą o nachodzących na siebie polach (w przypadku dwuwymiarowym) klasteryzacji. Wartość tego wskaźnika oznacza jak obiekt jest podobny do własnego klastra w porównaniu do innych, przy czym im większa tym bardziej podobny do obiektów z własnego klastra a mniej do innych.
Purity to proporcja między dominującą klasą w klastrze a jego wielkością.

Dunn ratio between the minimal inter-cluster (inter wewnątrz grup i pomiędzy nimi) distance to maximal intra-cluster distance (intra w środku grupy). Indeks Dunn to stosunek najmniejszej odległości pomiędzy obiektami w różnych klastrach do największej wewnątrz klastra. Im większy tym lepsza klasteryzacja według tego algorytmu.
Davies-Bouldin stosunek rozproszenia punktów wewnątrz klastra do separacji między poszczególnymi klastrami. Im mniejsza wartość tym lepsza klasteryzacja według tego indeksu, w przeciwieństwie do poprzednich przyjmuje wartości od zera do nieskończoności.

Wine			
Klastry	Davies-Bouldi	Dunn	Silhouette
2	2,1075	0,145	0,1619
3	2,0001	0,1592	0,2038
4	1,7878	0,172	0,1938
5	1,7749	0,1856	0,1884
6	1,6018	0,1892	0,1755
7	1,5631	0,2031	0,1886
8	1,465	0,2108	0,1906
9	1,5056	0,2162	0,1783
10	1,5642	0,2228	0,1726
11	1,4618	0,2415	0,1647
12	1,4608	0,246	0,156
13	1,5604	0,2465	0,1358
14	1,5208	0,2682	0,1419
15	1,5634	0,2706	0,1283

Wine 3 grupy wybrany wskaźnik: Silhouette
Problem jaki należy rozwiązać w tym zbiorze danych to określenie z ilu różnych winiarni pochodzą dane wina. Wybrano wskaźnik Silhouette z powodu dużego prawdopodobieństwa że decydującym czynnikiem odróżniającm będzie znacznie odmienny skład chemiczny pomiędzy grupami.

Glass

Klastry	Davies-Bouldi	Dunn	Silhouette
2	1,5591	0,1095	0,3886
3	1,3901	0,1443	0,4068
4	1,0841	0,1478	0,4032
5	0,9374	0,1551	0,4095
6	0,8198	0,1573	0,3894
7	0,9658	0,1592	0,2923
8	0,9437	0,1621	0,2813
9	0,9691	0,1112	0,2803
10	0,9714	0,1282	0,3825
11	0,9216	0,1317	0,381
12	0,9993	0,1339	0,3619
13	0,9569	0,1351	0,3648
14	0,9636	0,1183	0,3585
15	0,9024	0,1199	0,365

Glass 5 lub 6 grup wybrany wskaźnik: Silhouette lub Davies-Bouldin
Zbiór tych danych to różny rozkład składników w badanych próbkach szkła. Wybrano wskaźniki Silhouette oraz Davies-Bouldin. W celu uzyskania odmiennych zbiorów pod względem rozkładu atrybutów w przypadku Silhouette, bądź podobieństwa wewnątrz klastrów w odniesieniu do innych grup w przypadku wskaźnika Davies-Bouldin.

Diabetes

Klastry	Davies-Bouldi	Dunn	Silhouette
2	2,1209	0,0604	0,191
3	1,8903	0,067	0,1626
4	2,0809	0,0732	0,1302
5	1,85	0,0774	0,1085
6	1,6092	0,0818	0,1365
7	1,7091	0,0844	0,1414
8	1,5888	0,085	0,1415

Diabetes 2 grupy wybrany wskaźnik: Silhouette
Biorąc pod uwagę że jest to zbiór danych medycznych najbardziej prawdopodobnym wyborem jest podział na dwie grupy: osoby zdrowe oraz chore. Wybrano miarę Silhouette z powodu tego iż rozkład atrybutów osób chorych będzie się różnił od osób zdrowych z dużym prawdopodobieństwem.

Seeds

Klastry	Davies-Bouldi	Dunn	Silhouette
2	0,8437	0,0955	0,452
3	1,031	0,1102	0,3502
4	1,2766	0,1037	0,3149
5	1,2332	0,0907	0,2937
6	1,5034	0,1005	0,2174
7	1,3515	0,1006	0,2192
8	1,5685	0,1012	0,1574

Seeds 3 grupy wybrany wskaźnik: Dunn
Odległości między obiektami wewnątrz klastra są małe w porównaniu do odstępu od innych obiektów z sąsiednich skupień. Biorąc pod uwagę że porównywane są ziarna z różnych gatunków zakładam iż jest to najlepsza miara dla tego przypadku. Uzasadnieniem tego stwierdzenia jest duże podobieństwo w rozmiarze ziaren z tego samego gatunku.

2.1 Iris

program1.R

m23

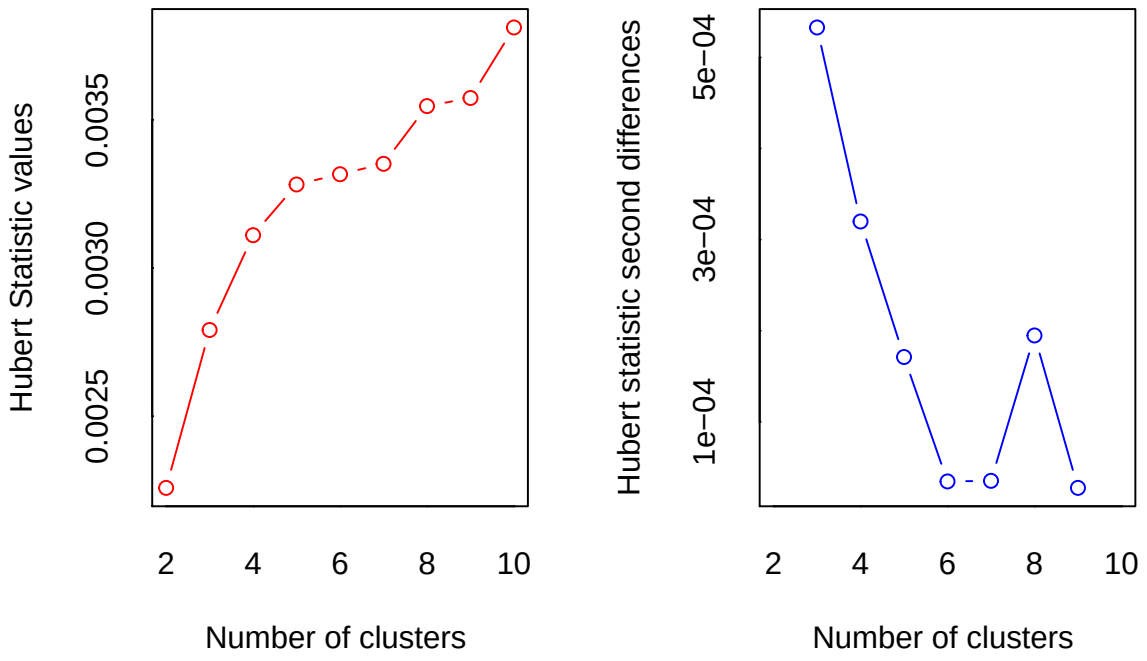
Sun Apr 22 19:10:20 2018

```
library(factoextra)
```

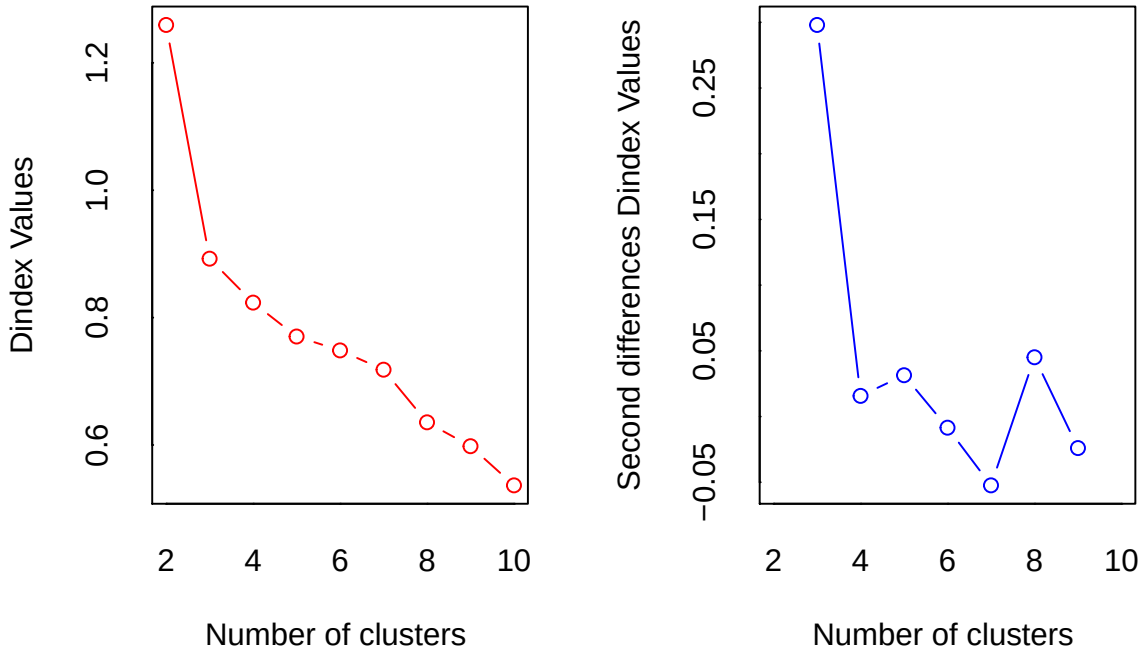
```
## Loading required package: ggplot2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(cluster)
library(NbClust)
library(datasets)
library(ggplot2)


# iris ------------------------------------------------------------------

data(iris)
iris.scaled <- scale(iris[, -5]) #usunięcie kolumny 5 (klasa) i skalowanie
NbClust(iris.scaled, distance = "euclidean", min.nc = 2, max.nc = 10, method = "complete", index ="all")
```




```
## *** : 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.
##
```

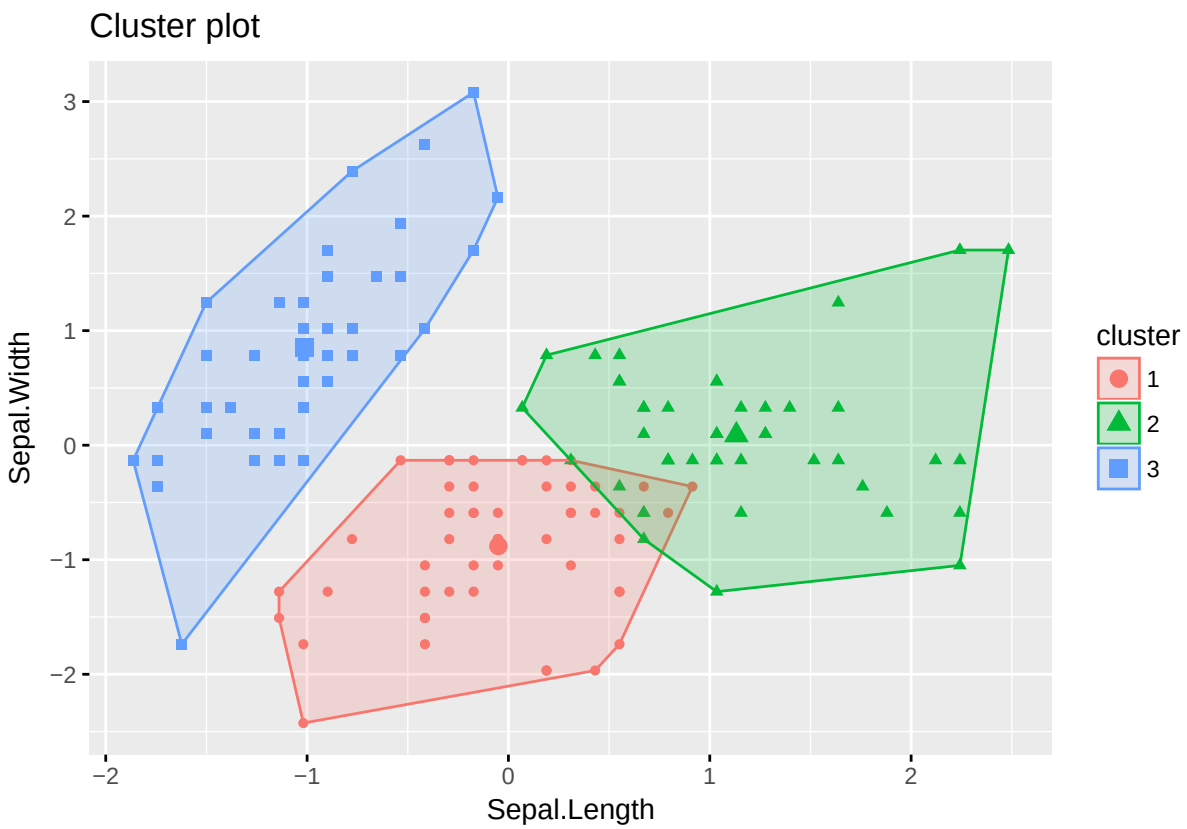



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 2 proposed 2 as the best number of clusters
## * 18 proposed 3 as the best number of clusters
## * 3 proposed 10 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************

## $All.index
##        KL       CH Hartigan     CCC    Scott   Marriot     TrCovW   TraceW
## 2  1.1854 151.6332 137.0327 -1.7090 206.2459 4042610.9 11446.7289 294.3866
## 3  9.8471 213.0817  33.2185  3.3712 444.9163 1852776.5  1065.2914 152.8569
## 4  1.3651 183.9682  24.9638  2.0682 543.6783 1705121.4   900.8342 124.6818
## 5  2.2951 166.6596  10.6281  0.4146 592.5766 1923092.4   562.5608 106.4760
## 6  2.7125 144.2243  12.7058 -0.7426 633.8818 2102673.6   558.6742  99.2046
## 7  0.0462 131.9862  39.5776 -1.2574 662.7020 2361686.4   432.8320  91.1610
## 8  2.7803 149.0463  19.9304  1.3079 781.5413 1396790.2   430.6392  71.3999
## 9  0.5276 150.1464  32.3094  2.0036 848.5792 1130685.4   285.0416  62.6120
## 10 2.6428 166.4470  16.6788  4.1139 922.7090  851586.6   199.1249  50.9395
##    Friedman   Rubin Cindex     DB Silhouette   Duda Pseudot2  Beale
## 2   38.3959  2.0245 0.2978 0.9735     0.4408 0.2934 170.9649 5.7326
## 3   53.5029  3.8991 0.3247 0.8581     0.4496 0.7005  32.0629 1.0185
## 4   63.1994  4.7802 0.3085 0.9581     0.4106 0.5462  39.0434 1.9637
```

```
## 5   64.5874  5.5975 0.3486 0.9262     0.3521 0.6102  14.0540 1.4752
## 6   71.9579  6.0078 0.3922 0.9569     0.3107 0.3089  20.1316 4.8602
## 7   72.7720  6.5379 0.4288 0.8597     0.3076 0.6359  36.6526 1.3613
## 8   80.5472  8.3474 0.3937 0.9451     0.3303 0.6792  19.8416 1.1140
## 9   85.6861  9.5189 0.3828 0.9766     0.3421 0.4147  56.4663 3.3249
## 10  89.4399 11.7002 0.3895 0.9858     0.3266 0.5931  18.5213 1.5969
##    Ratkowsky     Ball Ptbiserial   Frey McClain   Dunn Hubert SDindex
## 2     0.4606 147.1933     0.5970 0.0616  0.5265 0.0412 0.0023  2.0088
## 3     0.4958  50.9523     0.7169 0.7401  0.6296 0.0580 0.0028  1.5641
## 4     0.4431  31.1704     0.7030 0.8948  0.7655 0.0623 0.0031  1.8915
## 5     0.4047  21.2952     0.6892 1.3879  0.8368 0.0740 0.0033  1.5738
## 6     0.3722  16.5341     0.6817 0.4503  0.8664 0.0842 0.0033  1.8071
## 7     0.3476  13.0230     0.6815 1.1435  0.8708 0.0927 0.0034  1.6904
## 8     0.3314   8.9250     0.5977 1.0644  1.2698 0.0990 0.0035  1.9843
## 9     0.3151   6.9569     0.5491 0.8307  1.5619 0.1044 0.0036  2.4947
## 10    0.3023   5.0939     0.4938 0.5605  1.9793 0.1185 0.0038  2.3764
##    Dindex   SDbw
## 2  1.2595 1.1052
## 3  0.8925 0.3975
## 4  0.8236 0.4585
## 5  0.7703 0.3046
## 6  0.7485 0.2503
## 7  0.7183 0.1602
## 8  0.6356 0.1434
## 9  0.5981 0.1335
## 10 0.5367 0.1031
##
## $All.CriticalValues
##    CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2          0.6044            46.4793       0.0002
## 3          0.6106            47.8328       0.3979
## 4          0.5522            38.1140       0.1017
## 5          0.4284            29.3527       0.2166
## 6          0.2316            29.8538       0.0031
## 7          0.5921            44.0831       0.2478
## 8          0.5362            36.3229       0.3517
## 9          0.5291            35.6039       0.0120
## 10         0.4656            30.9841       0.1804
##
## $Best.nc
##                       KL       CH Hartigan     CCC    Scott Marriot   TrCovW
## Number_clusters 3.0000   3.0000   3.0000 10.0000   3.0000       3     3.00
## Value_Index     9.8471 213.0817 103.8142  4.1139 238.6703 2042179 10381.44
##                     TraceW Friedman   Rubin Cindex     DB Silhouette   Duda
## Number_clusters   3.0000    3.000  3.0000 2.0000 3.0000     3.0000 3.0000
## Value_Index     113.3546   15.107 -0.9934 0.2978 0.8581     0.4496 0.7005
##                   PseudoT2  Beale Ratkowsky   Ball PtBiserial Frey McClain
## Number_clusters   3.0000 3.0000    3.0000  3.000     3.0000    1  2.0000
## Value_Index      32.0629 1.0185    0.4958 96.241     0.7169   NA  0.5265
##                     Dunn Hubert SDindex Dindex    SDbw
## Number_clusters 10.0000      0  3.0000      0 10.0000
## Value_Index      0.1185      0  1.5641      0  0.1031
##
## $Best.partition
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 3 3 2 3 2 3 2 3 2 2 3 2 3 3 3 3 2 2 2
##  [71] 3 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 2 3 2 2 3 2 2 2 3 3 3 2 2 3 3 3 3 3
## [106] 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3
```

```
irisCluster <- kmeans(iris.scaled, centers=3, nstart = 20) #klasteryzacja iris | centers= ilość klastró


fviz_cluster(object=irisCluster, data = iris.scaled, geom = "point",stand = FALSE, choose.vars = c("Sep
```



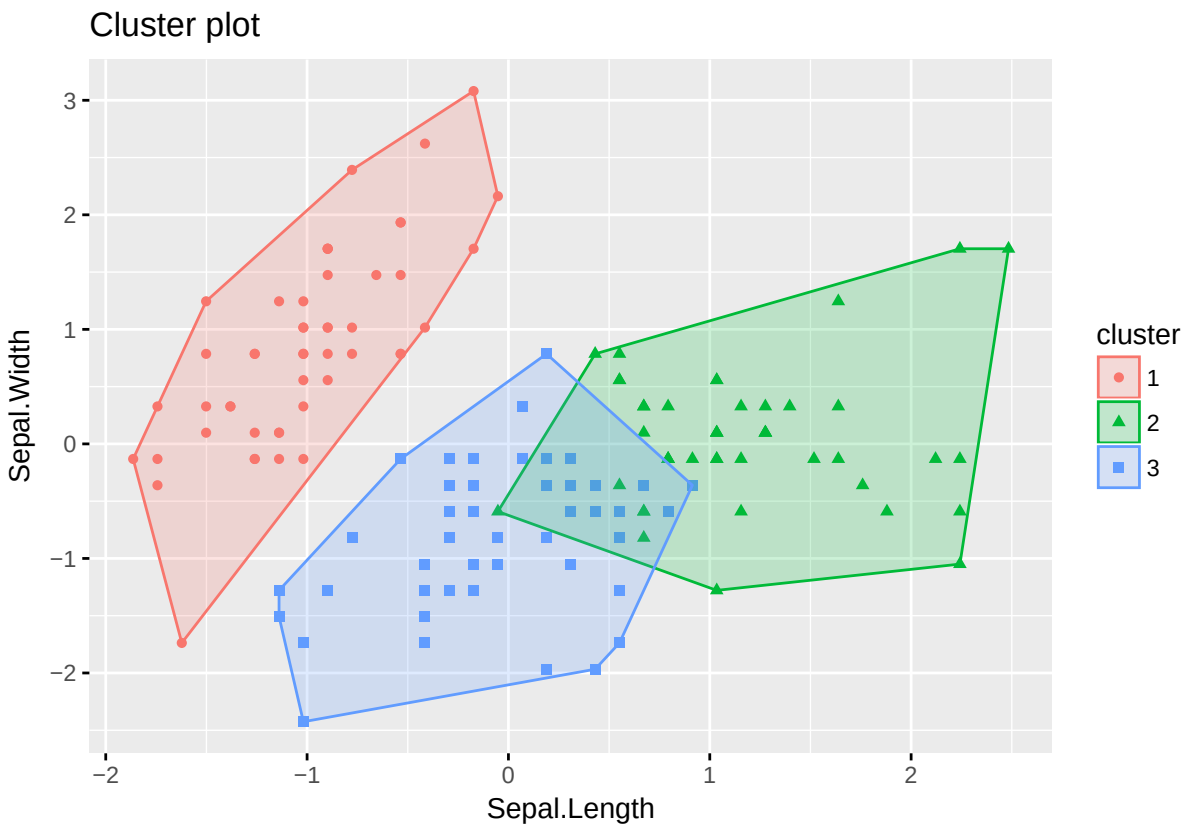

```
table(iris$Species, irisCluster$cluster)# macierz błędów porównanie czy klastry odpowiadają klasom
```

```
##
##                1  2  3
##   setosa       0  0 50
##   versicolor 39 11  0
##   virginica  14 36  0
```

```
# klasteryzacja PAM (Partitioning Around Medoids)
library("cluster")
pam.res <- pam(iris.scaled, 3)#drugi argument na ile klastrów podzielić
pam.res$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 3 3 3 3 3 3 3 2 3 3 3 3
##  [71] 3 3 3 3 3 2 2 2 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2
## [106] 2 3 2 2 2 2 2 2 3 2 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 3 2 2 2 3 2
## [141] 2 2 3 2 2 2 3 2 2 3
```

```
fviz_cluster(object=pam.res, data=iris.scaled, stand = FALSE, geom = "point", choose.vars = c("Sepal.Le
```




```
table(pam.res$clustering, iris$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0          9        36
##   3      0         41        14
```

```
# porównanie klasteryzacji
library(fpc)
#cluster.stats(d=dist(iris.scaled), irisCluster$cluster, pam.res$clustering)
species <- as.numeric(iris$Species)#przekodowanie na wartości numeryczne bo się wysypuje
cluster.stats(d=dist(iris.scaled), species, pam.res$clustering)
```

```
## $n
## [1] 150
##
## $cluster.number
## [1] 3
##
## $cluster.size
## [1] 50 50 50
##
## $min.cluster.size
## [1] 50
##
```

```
## $noisen
## [1] 0
##
## $diameter
## [1] 5.034198 3.761749 4.830573
##
## $average.distance
## [1] 1.175155 1.293268 1.472444
##
## $median.distance
## [1] 0.9884177 1.1996947 1.3334024
##
## $separation
## [1] 1.5533592 0.3722352 0.3722352
##
## $average.toother
## [1] 3.647912 2.536273 3.066746
##
## $separation.matrix
##          [,1]      [,2]      [,3]
## [1,] 0.000000 1.5533592 2.6652052
## [2,] 1.553359 0.0000000 0.3722352
## [3,] 2.665205 0.3722352 0.0000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 3.117440 4.178385
## [2,] 3.117440 0.000000 1.955107
## [3,] 4.178385 1.955107 0.000000
##
## $average.between
## [1] 3.083644
##
## $average.within
## [1] 1.313622
##
## $n.between
## [1] 7500
##
## $n.within
## [1] 3675
##
## $max.diameter
## [1] 5.034198
##
## $min.separation
## [1] 0.3722352
##
## $within.cluster.ss
## [1] 165.4283
##
## $clus.avg.silwidths
##         1         2         3
## 0.6254096 0.3066578 0.2113110
```

```
##
## $avg.silwidth
## [1] 0.3811262
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.6299982
##
## $dunn
## [1] 0.0739413
##
## $dunn2
## [1] 1.327797
##
## $entropy
## [1] 1.098612
##
## $wb.ratio
## [1] 0.4259967
##
## $ch
## [1] 191.3036
##
## $cwidegap
## [1] 1.3892251 0.6975467 1.0809668
##
## $widestgap
## [1] 1.389225
##
## $sindex
## [1] 0.4270882
##
## $corrected.rand
## [1] 0.6416027
##
## $vi
## [1] 0.7129034
```

```
ClusterPurity <- function(clusters, classes) { #https://stackoverflow.com/questions/9253843/r-clusterin
  sum(apply(table(classes, clusters), 2, max)) / length(clusters)
}

ClusterPurity(pam.res$clustering, species)
```

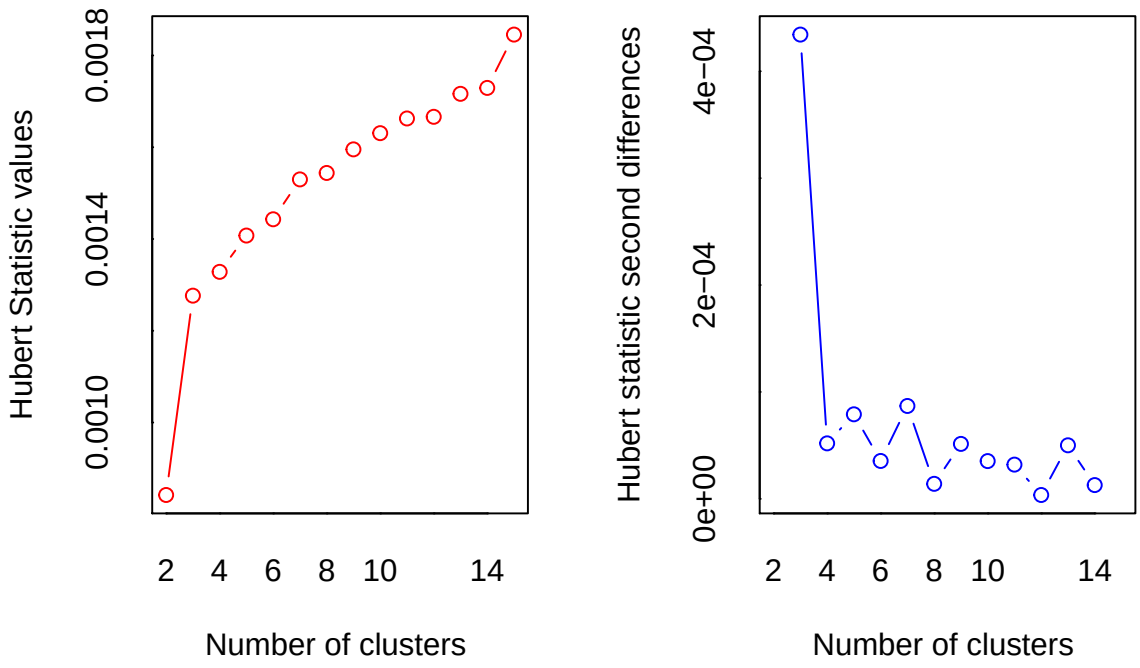
```
## [1] 0.8466667
```

```
ClusterPurity(irisCluster$cluster, species)
```

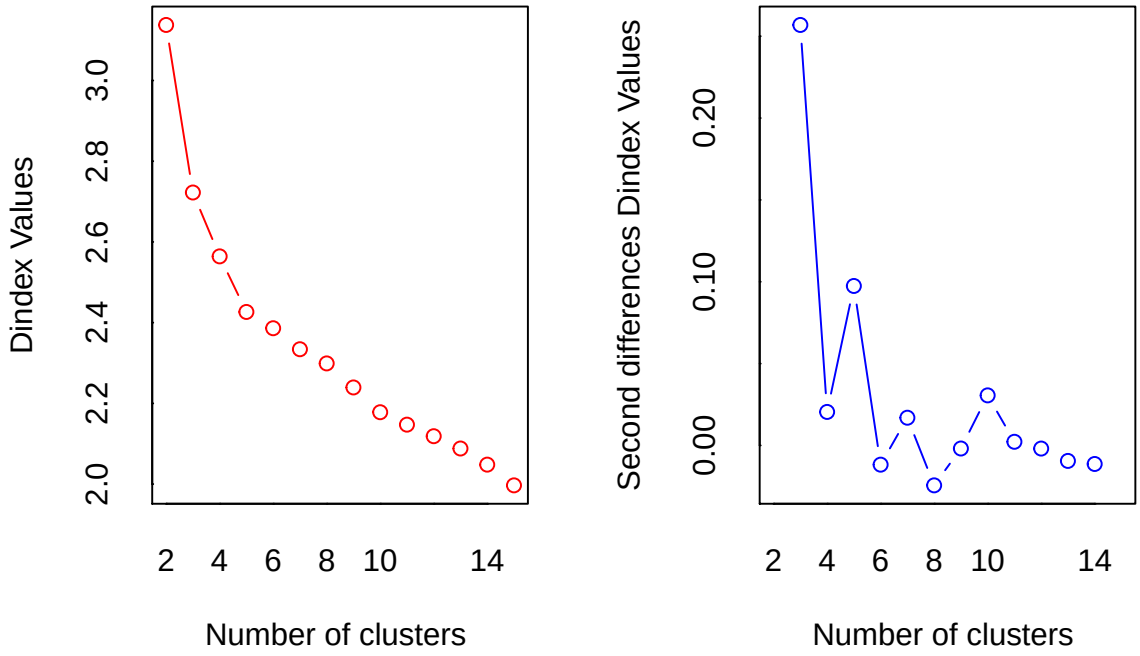
```
## [1] 0.8333333
```

2.2 Wine

```r
# wine ----------------------------------------------------------------------
wine = read.table("wine.data", header = FALSE, sep = ",")
names(wine) <- c("Class","Alcohol","Malic Acid","Ash","Alcalinity of ash","Magnesium","Total phenols","
wine$`Class` <- factor(wine$`Class`)

wine.scaled <- scale(wine[, -1 ])
NbClust(wine.scaled, distance = "euclidean", min.nc = 2, max.nc = 15, method = "complete", index ="all"
```




```
## *** : 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.
##
```

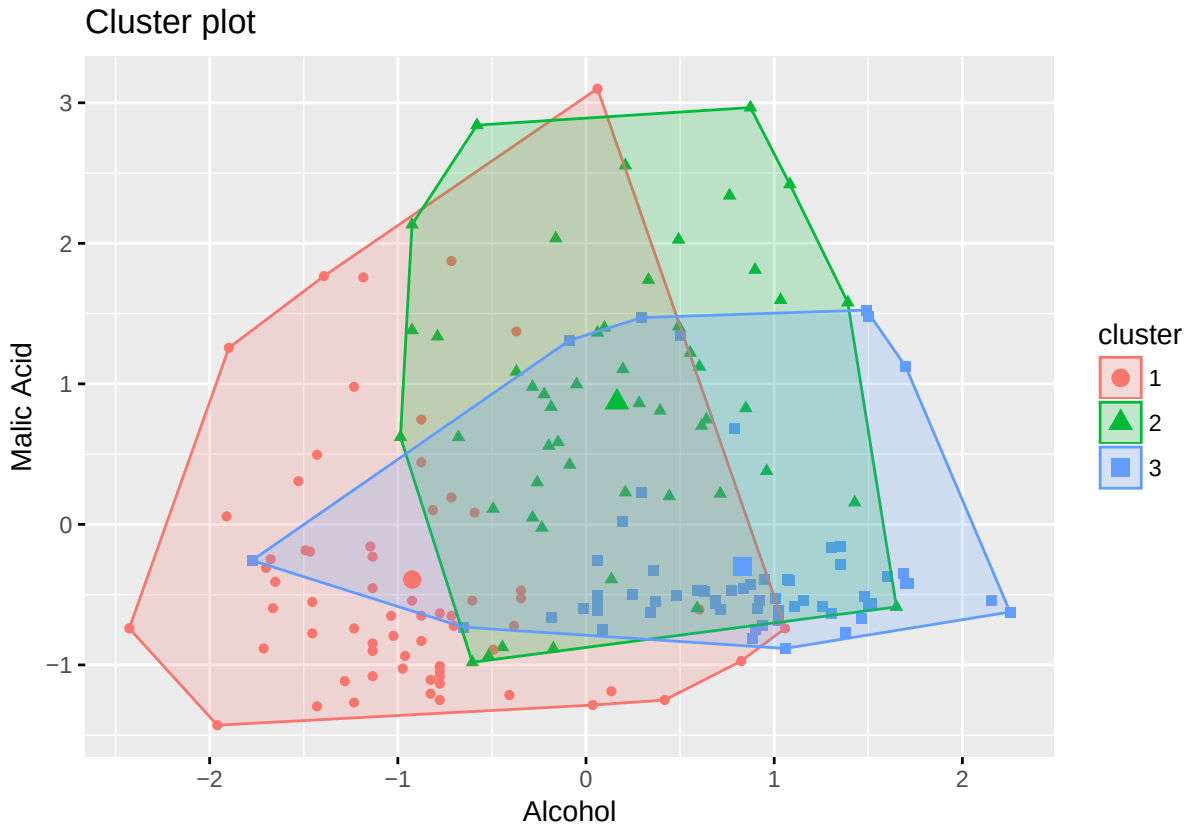



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 1 proposed 2 as the best number of clusters
## * 11 proposed 3 as the best number of clusters
## * 5 proposed 5 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 1 proposed 9 as the best number of clusters
## * 1 proposed 12 as the best number of clusters
## * 3 proposed 15 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************

## $All.index
##         KL      CH Hartigan     CCC     Scott      Marriot    TrCovW
## 2   0.4765 36.3108  51.5859 -5.9723  173.4093 1.184550e+26 38733.044
## 3   2.7077 48.9898  23.6889 -1.9993  514.3727 3.924951e+25 16343.217
## 4   1.0996 44.7202  21.3971 -0.8479  672.0734 2.877032e+25 12254.140
## 5  14.2227 42.7667   5.9919  1.1480  815.5341 2.007895e+25 10265.337
## 6   0.1566 36.3852  11.1684  0.1102  891.2764 1.889310e+25  9706.379
## 7   3.3508 33.9527   5.6567  0.7031 1021.1316 1.239849e+25  8653.215
## 8   0.3283 30.6925   9.7189  0.0608 1095.1712 1.068333e+25  8122.049
## 9   0.9925 29.4320   9.6347  0.9150 1246.5193 5.777553e+24  7187.825
## 10  3.5697 28.5539   4.4142  1.8457 1334.2475 4.357272e+24  6143.166
```

```
## 11  0.7665 26.6555   4.7915  1.4832 1423.6751 3.190138e+24  5792.563
## 12  0.9245 25.2113   4.8310  1.3300 1468.8363 2.945777e+24  5405.046
## 13  0.4757 24.0398   7.8917  1.2710 1529.8839 2.453446e+24  5251.322
## 14  0.9415 23.7143   8.3300  2.0453 1598.5598 1.934580e+24  4673.155
## 15  1.6399 23.5892   5.7106  2.9587 1678.7581 1.415275e+24  4233.020
##       TraceW Friedman  Rubin Cindex     DB Silhouette   Duda Pseudot2
## 2  1907.4674   7.2073 1.2063 0.3859 2.1075     0.1619 0.6648  53.9534
## 3  1475.1099  17.9014 1.5599 0.3551 2.0001     0.2038 0.7153  26.6719
## 4  1299.2384  22.6969 1.7710 0.3567 1.7878     0.1938 0.7259  21.1503
## 5  1156.9645  25.9819 1.9888 0.3591 1.7749     0.1884 0.8856   6.3314
## 6  1118.2341  27.8842 2.0577 0.3611 1.6018     0.1755 0.8139  12.5720
## 7  1050.0515  30.3304 2.1913 0.3683 1.5631     0.1886 0.4700   4.5107
## 8  1016.4279  31.4942 2.2638 0.3841 1.4650     0.1906 0.8156  10.6230
## 9   961.4610  35.9936 2.3932 0.3821 1.5056     0.1783 0.8351   9.0857
## 10  909.6045  37.7840 2.5297 0.3750 1.5642     0.1726 1.0585  -0.4421
## 11  886.3165  42.5269 2.5961 0.4319 1.4618     0.1647 0.6718   4.8854
## 12  861.5960  43.3219 2.6706 0.4393 1.4608     0.1560 0.9024   4.1119
## 13  837.2306  45.6132 2.7483 0.4335 1.5604     0.1358 0.7962   7.9342
## 14  799.0150  46.7022 2.8798 0.4651 1.5208     0.1419 0.8355   9.6445
## 15  760.3928  48.6933 3.0261 0.4533 1.5634     0.1283 0.7968   6.6299
##      Beale Ratkowsky     Ball Ptbiserial    Frey McClain   Dunn Hubert
## 2   4.4394    0.2699 953.7337     0.2943 -0.0894  0.7687 0.1450 0.0008
## 3   3.4856    0.3318 491.7033     0.4847 -0.0054  1.4426 0.1592 0.0013
## 4   3.2974    0.3204 324.8096     0.5376 -0.0036  1.6648 0.1720 0.0013
## 5   1.1253    0.3112 231.3929     0.5811  0.0623  1.8335 0.1856 0.0014
## 6   1.9950    0.2891 186.3723     0.5877  0.0363  1.8736 0.1892 0.0014
## 7   8.0168    0.2759 150.0074     0.6128 -0.0352  1.9968 0.2031 0.0015
## 8   1.9667    0.2617 127.0535     0.6141  0.5233  1.9996 0.2108 0.0015
## 9   1.7178    0.2528 106.8290     0.6024  0.5105  2.2454 0.2162 0.0016
## 10 -0.4365    0.2446  90.9605     0.5815 -0.0147  2.6755 0.2228 0.0016
## 11  3.9467    0.2352  80.5742     0.5827  0.3013  2.6794 0.2415 0.0017
## 12  0.9369    0.2273  71.7997     0.5820  1.2003  2.7121 0.2460 0.0017
## 13  2.2033    0.2204  64.4024     0.5565  0.1688  3.0675 0.2465 0.0017
## 14  1.7141    0.2155  57.0725     0.5573  1.1191  3.2185 0.2682 0.0017
## 15  2.1821    0.2108  50.6929     0.4955  0.3248  4.3472 0.2706 0.0018
##    SDindex Dindex   SDbw
## 2   1.5258 3.1375 0.8444
## 3   1.4353 2.7225 0.6980
## 4   1.2810 2.5643 0.6228
## 5   1.2187 2.4265 0.5918
## 6   1.1806 2.3862 0.5031
## 7   1.2750 2.3341 0.6042
## 8   1.1785 2.2990 0.4723
## 9   1.1521 2.2395 0.4575
## 10  1.2747 2.1781 0.4513
## 11  1.1796 2.1473 0.3951
## 12  1.2258 2.1187 0.3763
## 13  1.3948 2.0882 0.3738
## 14  1.3910 2.0483 0.3671
## 15  1.4038 1.9970 0.3618
##
## $All.CriticalValues
##    CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2          0.8346            21.2026       0.0000
```

```
## 3          0.8047            16.2594       0.0000
## 4          0.7914            14.7570       0.0001
## 5          0.7808            13.7528       0.3338
## 6          0.7900            14.6160       0.0188
## 7          0.4549             4.7941       0.0000
## 8          0.7774            13.4577       0.0213
## 9          0.7756            13.3086       0.0534
## 10         0.5667             6.1170       1.0000
## 11         0.6002             6.6617       0.0000
## 12         0.7589            12.0751       0.5141
## 13         0.7395            10.9226       0.0089
## 14         0.7808            13.7528       0.0540
## 15         0.7213            10.0439       0.0100
##
## $Best.nc
##                       KL      CH Hartigan     CCC    Scott      Marriot
## Number_clusters  5.0000  3.0000   3.0000 15.0000   3.0000 3.000000e+00
## Value_Index     14.2227 48.9898  27.8971  2.9587 340.9634 6.872632e+25
##                    TrCovW   TraceW Friedman   Rubin Cindex      DB
## Number_clusters     3.00   3.0000   3.0000  5.0000 3.0000 12.0000
## Value_Index     22389.83 256.4861  10.6941 -0.1489 0.3551  1.4608
##                 Silhouette   Duda PseudoT2  Beale Ratkowsky     Ball
## Number_clusters     3.0000 5.0000   5.0000 5.0000    3.0000   3.0000
## Value_Index         0.2038 0.8856   6.3314 1.1253    0.3318 462.0304
##                 PtBiserial Frey McClain    Dunn Hubert SDindex Dindex
## Number_clusters     8.0000    1  2.0000 15.0000      0  9.0000      0
## Value_Index         0.6141   NA  0.7687  0.2706      0  1.1521      0
##                    SDbw
## Number_clusters 15.0000
## Value_Index      0.3618
##
## $Best.partition
##   [1] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1
##  [71] 1 2 2 2 2 1 1 1 1 2 2 2 2 3 2 2 1 2 2 2 2 2 2 2 2 1 3 2 2 2 1 2 2 2 2
## [106] 2 2 2 2 2 1 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3 3
```

```r
wineCluster <- kmeans(wine.scaled, centers=3, nstart = 20)

fviz_cluster(object=wineCluster, data = wine.scaled, geom = "point",stand = FALSE, choose.vars = c("Alc
```



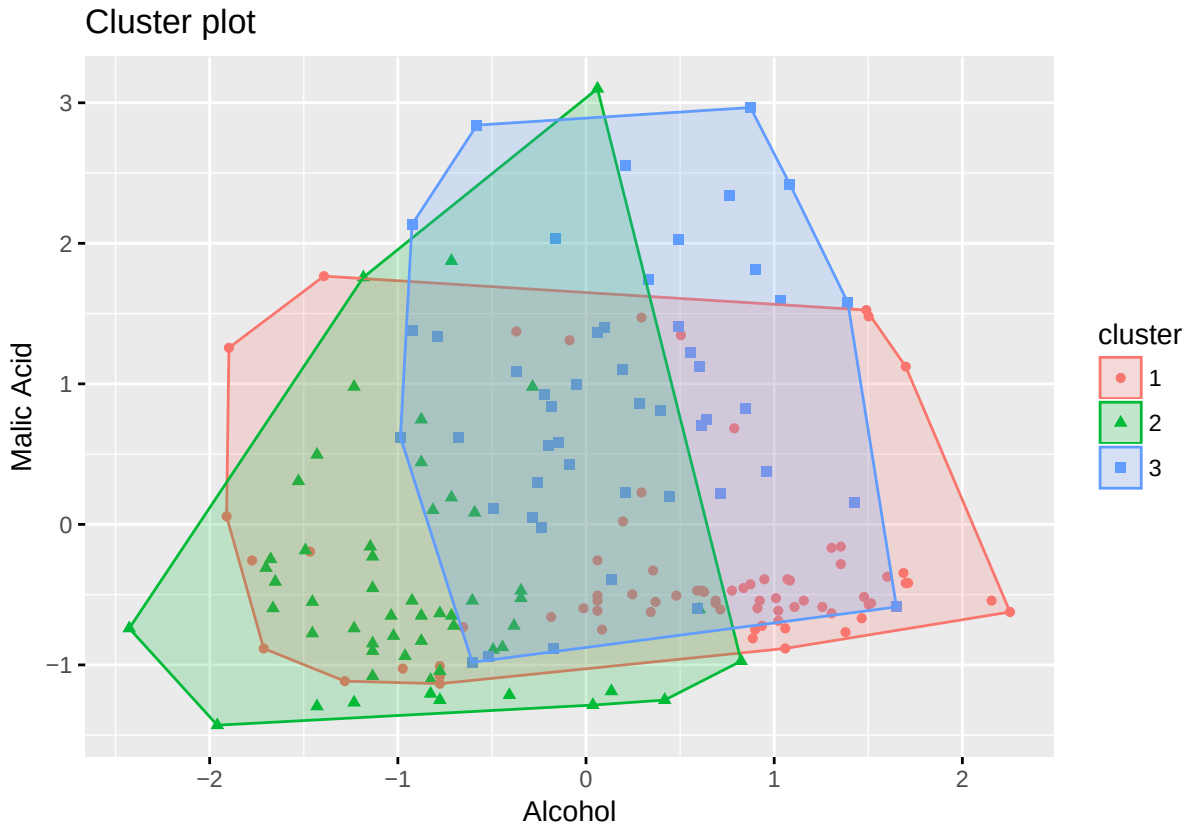

```
table(wine$Class, wineCluster$cluster)# macierz błędów porównanie czy klastry odpowiadają klasom
```

```
##
##      1  2  3
##   1  0  0 59
##   2 65  3  3
##   3  0 48  0
```

```
# klasteryzacja PAM (Partitioning Around Medoids)
library("cluster")
pam.res <- pam(wine.scaled, 3)#drugi argument na ile klastrów podzielić
pam.res$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 2 2 2 1
##  [71] 2 1 2 1 1 2 2 2 2 1 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 2 2
## [106] 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 1 2 2 1 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3 3
```

```
fviz_cluster(object=pam.res, data=wine.scaled, stand = FALSE, geom = "point", choose.vars = c("Alcohol"
```




```
table(pam.res$clustering, wine$Class)
```

```
##
##      1  2  3
##   1 59 15  0
##   2  0 55  0
##   3  0  1 48
```

```
# porównanie klasteryzacji
library(fpc)
#cluster.stats(d=dist(iris.scaled), irisCluster$cluster, pam.res$clustering)
species <- as.numeric(wine$Class)#przekodowanie na wartości numeryczne bo się wysypuje
cluster.stats(d=dist(wine.scaled), species, pam.res$clustering)
```

```
## $n
## [1] 178
##
## $cluster.number
## [1] 3
##
## $cluster.size
## [1] 59 71 48
##
## $min.cluster.size
## [1] 48
##
## $noisen
## [1] 0
```

```
##
## $diameter
## [1]  7.116800 11.179959  7.258158
##
## $average.distance
## [1] 3.083077 4.255366 3.423865
##
## $median.distance
## [1] 3.001971 4.052318 3.337195
##
## $separation
## [1] 1.977703 1.977703 2.148839
##
## $average.toother
## [1] 5.490766 5.311337 5.740092
##
## $separation.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 1.977703 3.375771
## [2,] 1.977703 0.000000 2.148839
## [3,] 3.375771 2.148839 0.000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 5.142364 6.006110
## [2,] 5.142364 0.000000 5.519034
## [3,] 6.006110 5.519034 0.000000
##
## $average.between
## [1] 5.500003
##
## $average.within
## [1] 3.702451
##
## $n.between
## [1] 10429
##
## $n.within
## [1] 5324
##
## $max.diameter
## [1] 11.17996
##
## $min.separation
## [1] 1.977703
##
## $within.cluster.ss
## [1] 1292.681
##
## $clus.avg.silwidths
##         1         2         3
## 0.3930113 0.1231148 0.3723332
##
## $avg.silwidth
```

```
## [1] 0.2797798
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.5919041
##
## $dunn
## [1] 0.1768972
##
## $dunn2
## [1] 1.208442
##
## $entropy
## [1] 1.086038
##
## $wb.ratio
## [1] 0.6731725
##
## $ch
## [1] 68.25193
##
## $cwidegap
## [1] 2.594685 4.782409 3.000725
##
## $widestgap
## [1] 4.782409
##
## $sindex
## [1] 2.229628
##
## $corrected.rand
## [1] 0.7411365
##
## $vi
## [1] 0.4708601
```

```
cluster.stats(d=dist(wine.scaled), species, wineCluster$cluster)
```

```
## $n
## [1] 178
##
## $cluster.number
## [1] 3
##
## $cluster.size
## [1] 59 71 48
##
## $min.cluster.size
## [1] 48
##
```

```
## $noisen
## [1] 0
##
## $diameter
## [1]  7.116800 11.179959  7.258158
##
## $average.distance
## [1] 3.083077 4.255366 3.423865
##
## $median.distance
## [1] 3.001971 4.052318 3.337195
##
## $separation
## [1] 1.977703 1.977703 2.148839
##
## $average.toother
## [1] 5.490766 5.311337 5.740092
##
## $separation.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 1.977703 3.375771
## [2,] 1.977703 0.000000 2.148839
## [3,] 3.375771 2.148839 0.000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 5.142364 6.006110
## [2,] 5.142364 0.000000 5.519034
## [3,] 6.006110 5.519034 0.000000
##
## $average.between
## [1] 5.500003
##
## $average.within
## [1] 3.702451
##
## $n.between
## [1] 10429
##
## $n.within
## [1] 5324
##
## $max.diameter
## [1] 11.17996
##
## $min.separation
## [1] 1.977703
##
## $within.cluster.ss
## [1] 1292.681
##
## $clus.avg.silwidths
##         1         2         3
## 0.3930113 0.1231148 0.3723332
```

```
##
## $avg.silwidth
## [1] 0.2797798
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.5919041
##
## $dunn
## [1] 0.1768972
##
## $dunn2
## [1] 1.208442
##
## $entropy
## [1] 1.086038
##
## $wb.ratio
## [1] 0.6731725
##
## $ch
## [1] 68.25193
##
## $cwidegap
## [1] 2.594685 4.782409 3.000725
##
## $widestgap
## [1] 4.782409
##
## $sindex
## [1] 2.229628
##
## $corrected.rand
## [1] 0.897495
##
## $vi
## [1] 0.2704766
```

```
ClusterPurity <- function(clusters, classes) { #https://stackoverflow.com/questions/9253843/r-clusterin
  sum(apply(table(classes, clusters), 2, max)) / length(clusters)
}

ClusterPurity(pam.res$clustering, species)
```

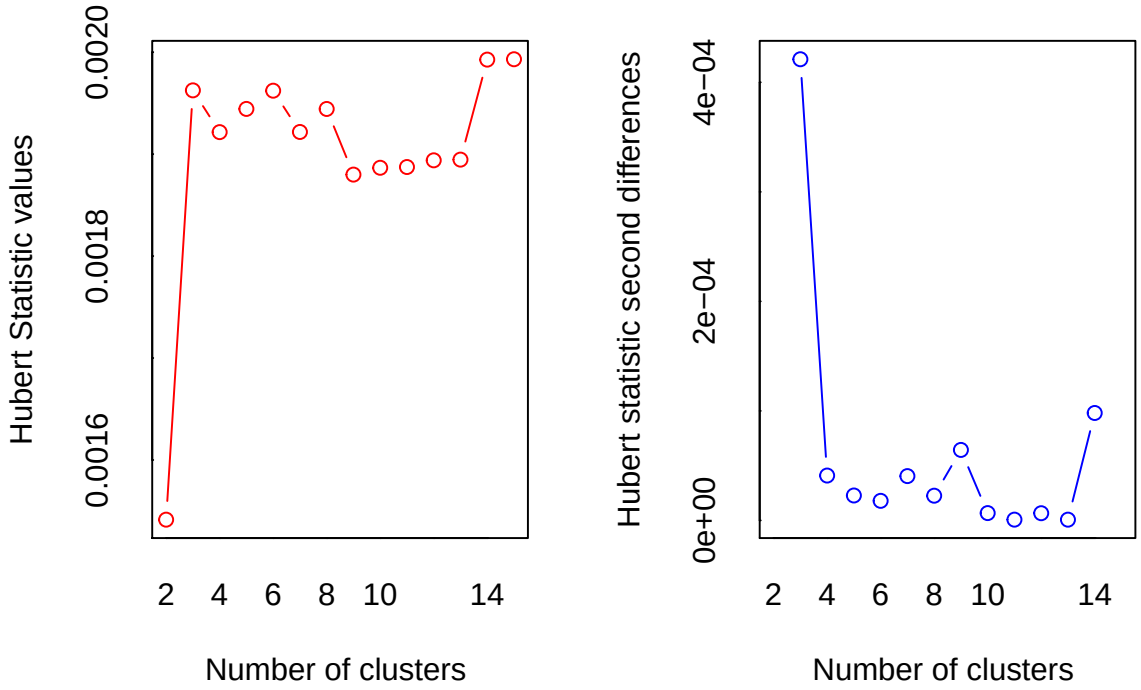
```
## [1] 0.9101124
```

```
ClusterPurity(wineCluster$cluster, species)
```

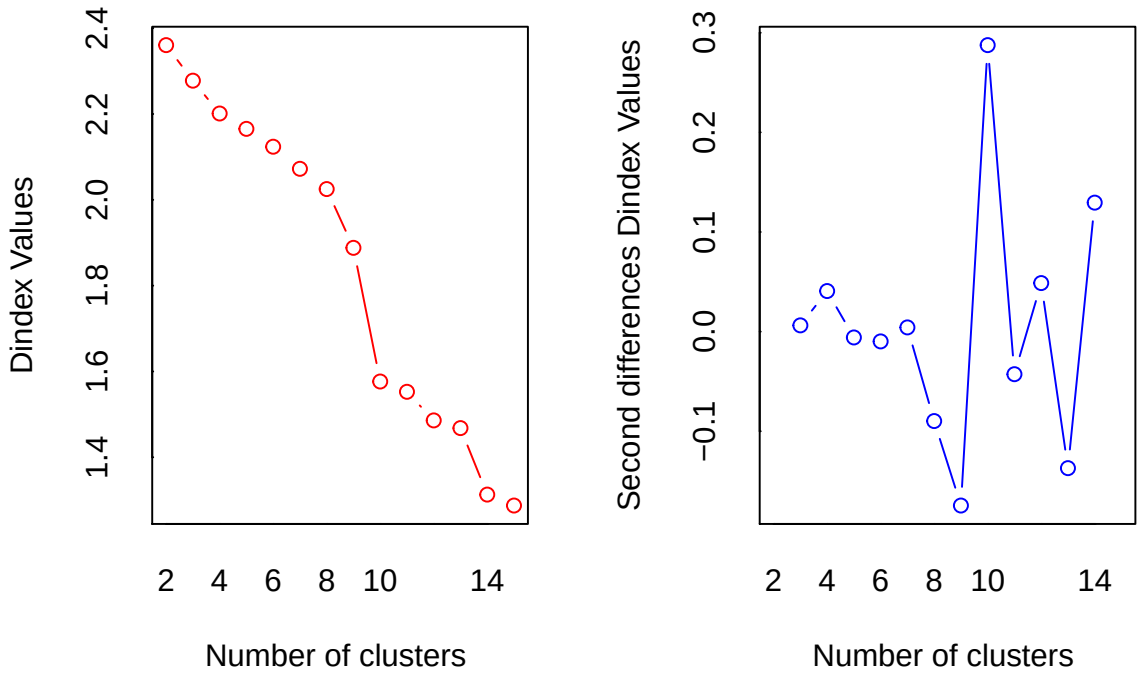
```
## [1] 0.9662921
```

2.3 Glass

```r
# glass ----------------------------------------------------------------------

glass = read.csv("glass.data", header = FALSE, sep = ",")
names(glass) <- c("Id number","RI", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe", "Type_of_glass")
glass = glass[-1] ##usuwam pierwszą kolumnę ID
glass$`Type_of_glass` <- factor(glass$`Type_of_glass`)


glass.scaled <- scale(glass[, -10 ])
NbClust(glass.scaled, distance = "euclidean", min.nc = 2, max.nc = 15, method = "complete", index ="all
```




```
## *** : 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.
##
```

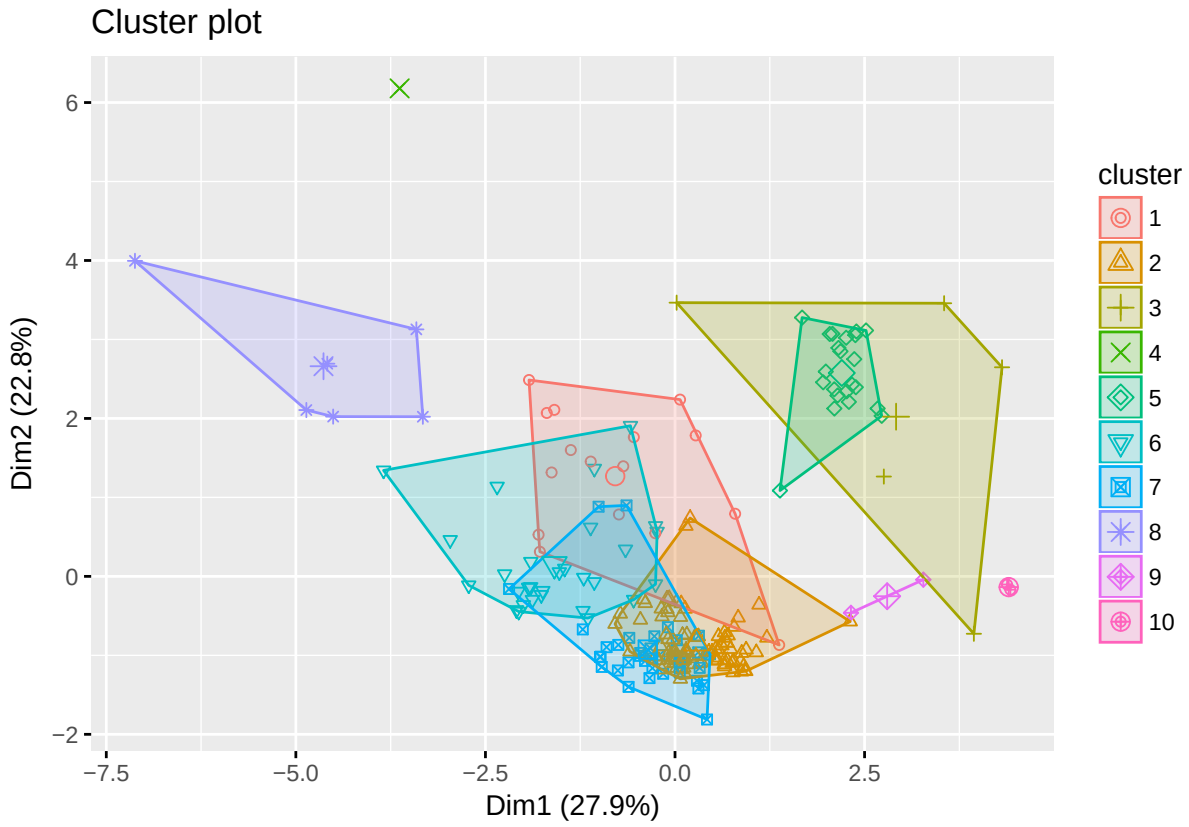



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 5 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 2 proposed 6 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 7 proposed 10 as the best number of clusters
## * 1 proposed 13 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
## * 2 proposed 15 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  10
##
##
## *******************************************************************

## $All.index
##         KL      CH Hartigan      CCC     Scott      Marriot    TrCovW
## 2   0.4005 30.7805  31.0913  -6.9247  184.4420 2.333689e+17 62877.657
## 3   6.1495 33.0364  15.6938  -8.0891  433.4537 1.640128e+17 49203.931
## 4   0.7395 28.7567   8.3598 -10.2531  735.0407 7.123779e+16 45889.336
## 5   1.1827 24.3993  10.5904 -12.9843  815.7360 7.634250e+16 43275.026
## 6   0.5785 22.5183  10.6326 -14.3800  967.4634 5.410175e+16 41481.709
```

```
## 7   0.7769 21.3933  10.6384 -15.3893 1102.6915 3.914452e+16 39326.281
## 8   0.2057 20.6988  29.0661 -16.4152 1203.0296 3.199097e+16 36708.694
## 9   0.3648 24.1822  97.3598 -11.1787 1343.1422 2.103712e+16 28754.737
## 10 23.7867 42.3143   8.3010   6.7396 1773.7238 3.472746e+15 10271.912
## 11  0.2555 40.2643  20.4788   6.7080 1825.4634 3.299569e+15  9420.119
## 12  9.6867 41.9506   5.3332   9.4858 2000.2726 1.734900e+15  8146.234
## 13  0.0515 39.7168  42.6758   8.9770 2045.4504 1.648595e+15  7794.020
## 14  7.9923 47.4909   8.1027  16.1862 2263.2940 6.908580e+14  5249.629
## 15  0.6066 46.2317  11.4375  16.3511 2304.9384 6.528322e+14  4723.159
##       TraceW Friedman  Rubin Cindex     DB Silhouette   Duda Pseudot2
## 2  1673.9565 151.7633 1.1452 0.2421 1.5591     0.3886 0.8541  33.4887
## 3  1459.8582 212.4082 1.3131 0.2930 1.3901     0.4068 0.2423  12.5088
## 4  1358.7935 243.3411 1.4108 0.2999 1.0841     0.4032 1.0016  -0.3066
## 5  1306.7730 265.6465 1.4670 0.3098 0.9374     0.4095 1.1319  -1.6310
## 6  1243.7501 277.4509 1.5413 0.3136 0.8198     0.3894 0.9433  11.3506
## 7  1183.2634 319.5498 1.6201 0.3661 0.9658     0.2923 0.5977   8.7494
## 8  1125.4241 347.9153 1.7034 0.3718 0.9437     0.2813 0.8572  30.3082
## 9   986.2647 385.3807 1.9437 0.3914 0.9691     0.2803 0.5953 113.5107
## 10  668.6877 558.9464 2.8668 0.3388 0.9714     0.3825 0.5899   6.2576
## 11  642.5420 567.7194 2.9835 0.3473 0.9216     0.3810 0.5743  21.4958
## 12  583.6616 590.4520 3.2844 0.3467 0.9993     0.3619 2.2387  -1.1066
## 13  568.6482 595.1181 3.3712 0.3498 0.9569     0.3648 0.6972  59.0616
## 14  469.0588 604.6685 4.0869 0.3058 0.9636     0.3585 3.3720  -1.4069
## 15  450.7956 610.1865 4.2525 0.3096 0.9024     0.3650 0.5181  12.0910
##      Beale Ratkowsky     Ball Ptbiserial    Frey McClain   Dunn Hubert
## 2   1.0208    0.1969 836.9783     0.3626 -2.0538  0.0960 0.1095 0.0015
## 3  15.0230    0.2641 486.6194     0.5614 -2.0841  0.1161 0.1443 0.0020
## 4  -0.0096    0.2504 339.6984     0.5633 -1.2588  0.1161 0.1478 0.0019
## 5  -0.6529    0.2395 261.3546     0.5922 -1.7652  0.1195 0.1551 0.0019
## 6   0.3587    0.2314 207.2917     0.5954  5.0661  0.1196 0.1573 0.0020
## 7   3.7528    0.2250 169.0376     0.5771  0.4336  0.1729 0.1592 0.0019
## 8   0.9945    0.2193 140.6780     0.5793  2.5795  0.1744 0.1621 0.0019
## 9   4.0573    0.2253 109.5850     0.5682  0.4814  0.3004 0.1112 0.0019
## 10  3.7577    0.2504  66.8688     0.6652  0.0551  0.5164 0.1282 0.0019
## 11  4.3027    0.2426  58.4129     0.6662  0.5594  0.5169 0.1317 0.0019
## 12 -2.2151    0.2376  48.6385     0.6682 -0.0152  0.5293 0.1339 0.0019
## 13  2.5888    0.2297  43.7422     0.6684  1.0532  0.5293 0.1351 0.0019
## 14 -2.8161    0.2317  33.5042     0.6286 -0.0636  0.7600 0.1183 0.0020
## 15  5.1861    0.2254  30.0530     0.6289  0.0383  0.7598 0.1199 0.0020
##    SDindex Dindex   SDbw
## 2   1.7401 2.3605 1.2600
## 3   2.6792 2.2777 2.1842
## 4   1.3926 2.2012 0.9006
## 5   1.2265 2.1654 0.7144
## 6   1.0807 2.1238 0.5434
## 7   1.2899 2.0723 0.5595
## 8   1.2976 2.0251 0.5667
## 9   1.3231 1.8881 0.5457
## 10  1.3226 1.5766 0.5434
## 11  1.3416 1.5526 0.5097
## 12  1.4976 1.4860 0.4838
## 13  1.3980 1.4680 0.4216
## 14  1.4405 1.3131 0.3974
## 15  1.3262 1.2876 0.3248
```

```
##
## $All.CriticalValues
##    CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2          0.8270            41.0007       0.4207
## 3          0.3418             7.7024       0.0000
## 4          0.8254            40.1869       1.0000
## 5          0.5695            10.5819       1.0000
## 6          0.8251            40.0507       0.9544
## 7          0.5577            10.3089       0.0004
## 8          0.8232            39.0931       0.4423
## 9          0.8186            37.0130       0.0000
## 10         0.4954             9.1671       0.0005
## 11         0.6708            14.2307       0.0000
## 12         0.2098             7.5336       1.0000
## 13         0.8068            32.5731       0.0059
## 14         0.2098             7.5336       1.0000
## 15         0.5577            10.3089       0.0000
##
## $Best.nc
##                       KL      CH Hartigan     CCC    Scott      Marriot
## Number_clusters 10.0000 14.0000  10.0000 15.0000  10.0000 4.000000e+00
## Value_Index     23.7867 47.4909  89.0588 16.3511 430.5816 9.787967e+16
##                   TrCovW   TraceW Friedman   Rubin Cindex     DB
## Number_clusters    10.00  10.0000  10.0000 10.0000 2.0000 6.0000
## Value_Index     18482.83 291.4313 173.5657 -0.8065 0.2421 0.8198
##                 Silhouette   Duda PseudoT2  Beale Ratkowsky     Ball
## Number_clusters     5.0000 2.0000   2.0000 2.0000    3.0000   3.0000
## Value_Index         0.4095 0.8541  33.4887 1.0208    0.2641 350.3589
##                 PtBiserial Frey McClain   Dunn Hubert SDindex Dindex
## Number_clusters    13.0000    1   2.000 8.0000      0  6.0000      0
## Value_Index         0.6684   NA   0.096 0.1621      0  1.0807      0
##                    SDbw
## Number_clusters 15.0000
## Value_Index      0.3248
##
## $Best.partition
##   [1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1  1
##  [24]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  1  1  1  2  1  1
##  [47]  1  2  2  1  2  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1  1
##  [70]  2  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [93]  1  1  1  1  1  1  1  1  1  1  1  2  2  3  4  5  3  6  5  5  5  1  1
## [116]  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  1  1  1  1  1  1
## [139]  1  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1  1  2  1  1  1
## [162]  1  1  7  8  8  8  8  8  8  3  9  9  3  3  3  6  6  6  6  6  6  6  6
## [185] 10  7  7  1  2  2  6  6  6  6  6  6  6  6  6  6  6  8  6  6  6  6  6
## [208]  7  6  6  6  6  6  6
```

```r
glassCluster <- kmeans(glass.scaled, centers=10, nstart = 20)

fviz_cluster(object=glassCluster, data = glass.scaled, geom = "point",stand = FALSE)#rysowanie
```



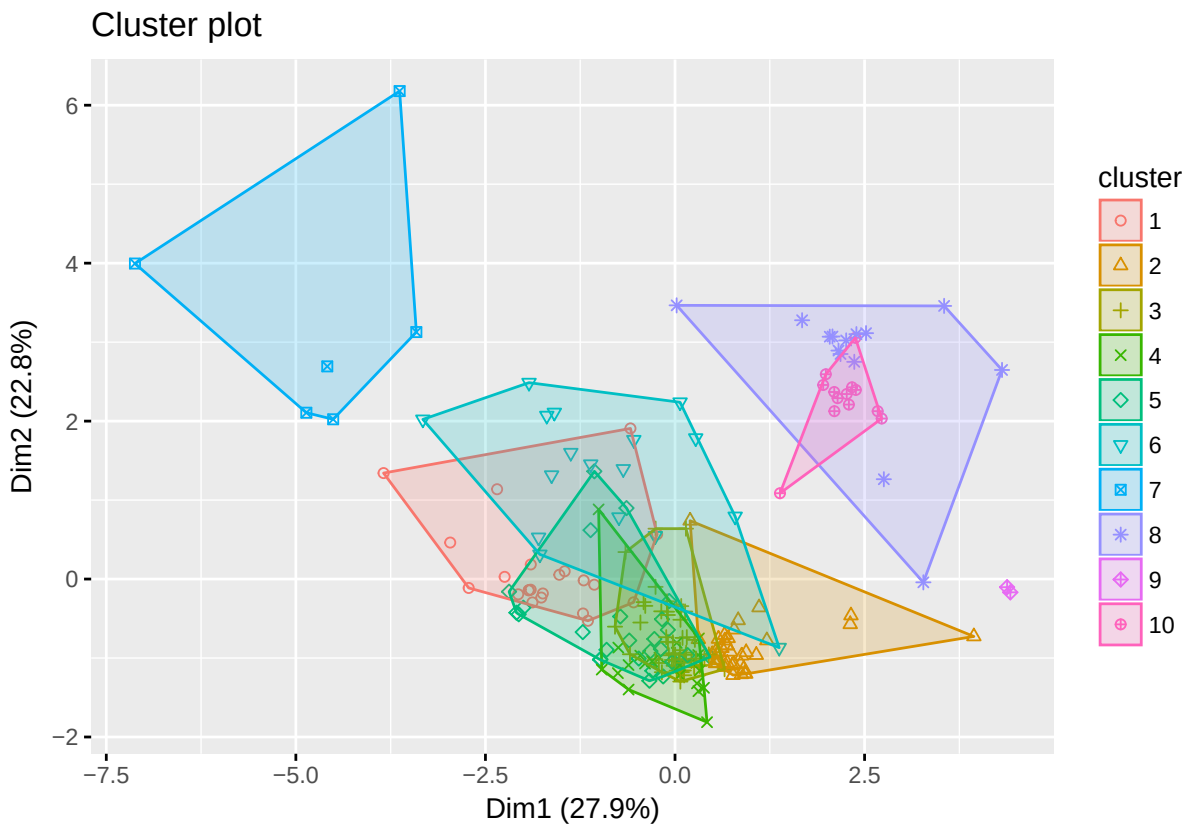

```
table(glass$Type_of_glass, glassCluster$cluster)# macierz błędów porównanie czy klastry odpowiadają kla
```

```
##
##      1  2  3  4  5  6  7  8  9 10
##   1  0 39  0  0  0 20 11  0  0  0
##   2  3 42  0  1  0  4 20  6  0  0
##   3  0 12  0  0  0  2  3  0  0  0
##   5  9  0  1  0  0  0  1  0  0  2
##   6  3  2  0  0  0  2  0  0  2  0
##   7  1  0  4  0 22  2  0  0  0  0
```

```
# klasteryzacja PAM (Partitioning Around Medoids)
library("cluster")
pam.res <- pam(glass.scaled, 10)#drugi argument na ile klastrów podzielić
pam.res$cluster
```

```
##   [1]  1  2  2  3  3  4  3  3  3  5  4  3  4  5  3  3  3  1  1  2  4  1  3
##  [24]  3  3  3  3  3  3  3  5  3  4  3  3  3  3  3  1  1  3  3  3  1  4  3
##  [47]  5  1  1  3  1  5  3  3  5  4  4  3  3  5  3  1  1  1  1  1  5  5  5
##  [70]  1  2  4  2  2  2  2  2  2  5  2  2  2  2  2  2  2  2  2  2  2  5  2
##  [93]  4  2  3  3  5  4  2  3  4  3  5  1  1  6  7  7  6  6  7  7  7  5  3
## [116]  3  5  2  4  2  3  4  2  2  3  5  3  5  5  5  6  7  3  5  3  4  5  2
## [139]  2  2  2  5  4  2  4  4  3  2  5  3  4  1  3  3  3  3  3  1  3  5  3
## [162]  5  5  8  6  6  6  6  6  6  6  9  9  6  4  6  3  3  2  3  2  6  6  6
## [185]  8  2  8  1  1  8 10 10 10  8  8 10 10 10 10 10 10  6 10  8 10  8  8
## [208]  8 10 10  8  8  8  8
```

```
fviz_cluster(object=pam.res, data=glass.scaled, stand = FALSE, geom = "point")
```




```
table(pam.res$clustering, glass$Type_of_glass)
```

```
##
##       1  2  3  5  6  7
##    1 16  2  2  0  0  2
##    2  3 30  1  0  2  1
##    3 33 11  9  0  3  0
##    4  8 10  1  1  0  0
##    5 10 13  4  0  0  0
##    6  0  4  0  9  3  1
##    7  0  6  0  0  0  0
##    8  0  0  0  1  1 12
##    9  0  0  0  2  0  0
##    10 0  0  0  0  0 13
```

```
# porównanie klasteryzacji
library(fpc)
#cluster.stats(d=dist(iris.scaled), irisCluster$cluster, pam.res$clustering)
species <- as.numeric(glass$Type_of_glass)#przekodowanie na wartości numeryczne bo się wysypuje
cluster.stats(d=dist(glass.scaled), species, pam.res$clustering)
```

```
## $n
## [1] 214
##
## $cluster.number
## [1] 6
```

```
##
## $cluster.size
## [1] 70 76 17 13  9 29
##
## $min.cluster.size
## [1] 9
##
## $noisen
## [1] 0
##
## $diameter
## [1]  5.910316 11.549908  5.409601 11.614249  7.456907  9.580278
##
## $average.distance
## [1] 2.174511 3.217294 2.226808 5.996042 3.355318 3.421395
##
## $median.distance
## [1] 2.178928 2.597019 2.077453 5.538649 3.252564 2.758891
##
## $separation
## [1] 0.2900630 0.2904070 0.2900630 1.2674185 1.1697734 0.8113059
##
## $average.toother
## [1] 3.469146 3.634728 3.122056 5.564185 4.125843 5.059579
##
## $separation.matrix
##           [,1]      [,2]     [,3]     [,4]     [,5]      [,6]
## [1,] 0.0000000 0.3215468 0.290063 1.643444 1.169773 1.3206600
## [2,] 0.3215468 0.0000000 0.290407 1.267418 1.523180 0.8113059
## [3,] 0.2900630 0.2904070 0.000000 1.788254 1.347812 1.7451753
## [4,] 1.6434435 1.2674185 1.788254 0.000000 1.555392 2.4686443
## [5,] 1.1697734 1.5231798 1.347812 1.555392 0.000000 1.9179258
## [6,] 1.3206600 0.8113059 1.745175 2.468644 1.917926 0.0000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 0.000000 2.815856 2.221533 5.397817 3.757769 4.958429
## [2,] 2.815856 0.000000 2.817819 5.504971 4.165287 5.087152
## [3,] 2.221533 2.817819 0.000000 5.456911 3.671318 4.875918
## [4,] 5.397817 5.504971 5.456911 0.000000 5.782949 6.115938
## [5,] 3.757769 4.165287 3.671318 5.782949 0.000000 4.434531
## [6,] 4.958429 5.087152 4.875918 6.115938 4.434531 0.000000
##
## $average.between
## [1] 3.937221
##
## $average.within
## [1] 2.820663
##
## $n.between
## [1] 16870
##
## $n.within
## [1] 5921
```

```
##
## $max.diameter
## [1] 11.61425
##
## $min.separation
## [1] 0.290063
##
## $within.cluster.ss
## [1] 1413.027
##
## $clus.avg.silwidths
##           1           2           3           4           5
##  0.020655635 -0.171701586  0.003345915 -0.167502031  0.045688927
##           6
##  0.210786173
##
## $avg.silwidth
## [1] -0.0336452
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.2258918
##
## $dunn
## [1] 0.02497476
##
## $dunn2
## [1] 0.3704999
##
## $entropy
## [1] 1.508658
##
## $wb.ratio
## [1] 0.7164098
##
## $ch
## [1] 14.83713
##
## $cwidegap
## [1] 1.603803 7.379597 2.470477 8.670881 4.752407 5.940563
##
## $widestgap
## [1] 8.670881
##
## $sindex
## [1] 0.3699863
##
## $corrected.rand
## [1] 0.1955628
```

```
##
## $vi
## [1] 2.24701
```

```
cluster.stats(d=dist(glass.scaled), species, glassCluster$cluster)
```

```
## $n
## [1] 214
##
## $cluster.number
## [1] 6
##
## $cluster.size
## [1] 70 76 17 13  9 29
##
## $min.cluster.size
## [1] 9
##
## $noisen
## [1] 0
##
## $diameter
## [1]  5.910316 11.549908  5.409601 11.614249  7.456907  9.580278
##
## $average.distance
## [1] 2.174511 3.217294 2.226808 5.996042 3.355318 3.421395
##
## $median.distance
## [1] 2.178928 2.597019 2.077453 5.538649 3.252564 2.758891
##
## $separation
## [1] 0.2900630 0.2904070 0.2900630 1.2674185 1.1697734 0.8113059
##
## $average.toother
## [1] 3.469146 3.634728 3.122056 5.564185 4.125843 5.059579
##
## $separation.matrix
##           [,1]      [,2]     [,3]     [,4]     [,5]      [,6]
## [1,] 0.0000000 0.3215468 0.290063 1.643444 1.169773 1.3206600
## [2,] 0.3215468 0.0000000 0.290407 1.267418 1.523180 0.8113059
## [3,] 0.2900630 0.2904070 0.000000 1.788254 1.347812 1.7451753
## [4,] 1.6434435 1.2674185 1.788254 0.000000 1.555392 2.4686443
## [5,] 1.1697734 1.5231798 1.347812 1.555392 0.000000 1.9179258
## [6,] 1.3206600 0.8113059 1.745175 2.468644 1.917926 0.0000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 0.000000 2.815856 2.221533 5.397817 3.757769 4.958429
## [2,] 2.815856 0.000000 2.817819 5.504971 4.165287 5.087152
## [3,] 2.221533 2.817819 0.000000 5.456911 3.671318 4.875918
## [4,] 5.397817 5.504971 5.456911 0.000000 5.782949 6.115938
## [5,] 3.757769 4.165287 3.671318 5.782949 0.000000 4.434531
## [6,] 4.958429 5.087152 4.875918 6.115938 4.434531 0.000000
##
## $average.between
```

```
## [1] 3.937221
##
## $average.within
## [1] 2.820663
##
## $n.between
## [1] 16870
##
## $n.within
## [1] 5921
##
## $max.diameter
## [1] 11.61425
##
## $min.separation
## [1] 0.290063
##
## $within.cluster.ss
## [1] 1413.027
##
## $clus.avg.silwidths
##           1           2           3           4           5
##  0.020655635 -0.171701586  0.003345915 -0.167502031  0.045688927
##           6
##  0.210786173
##
## $avg.silwidth
## [1] -0.0336452
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.2258918
##
## $dunn
## [1] 0.02497476
##
## $dunn2
## [1] 0.3704999
##
## $entropy
## [1] 1.508658
##
## $wb.ratio
## [1] 0.7164098
##
## $ch
## [1] 14.83713
##
## $cwidegap
```

```
## [1] 1.603803 7.379597 2.470477 8.670881 4.752407 5.940563
##
## $widestgap
## [1] 8.670881
##
## $sindex
## [1] 0.3699863
##
## $corrected.rand
## [1] 0.2027399
##
## $vi
## [1] 1.920896
```

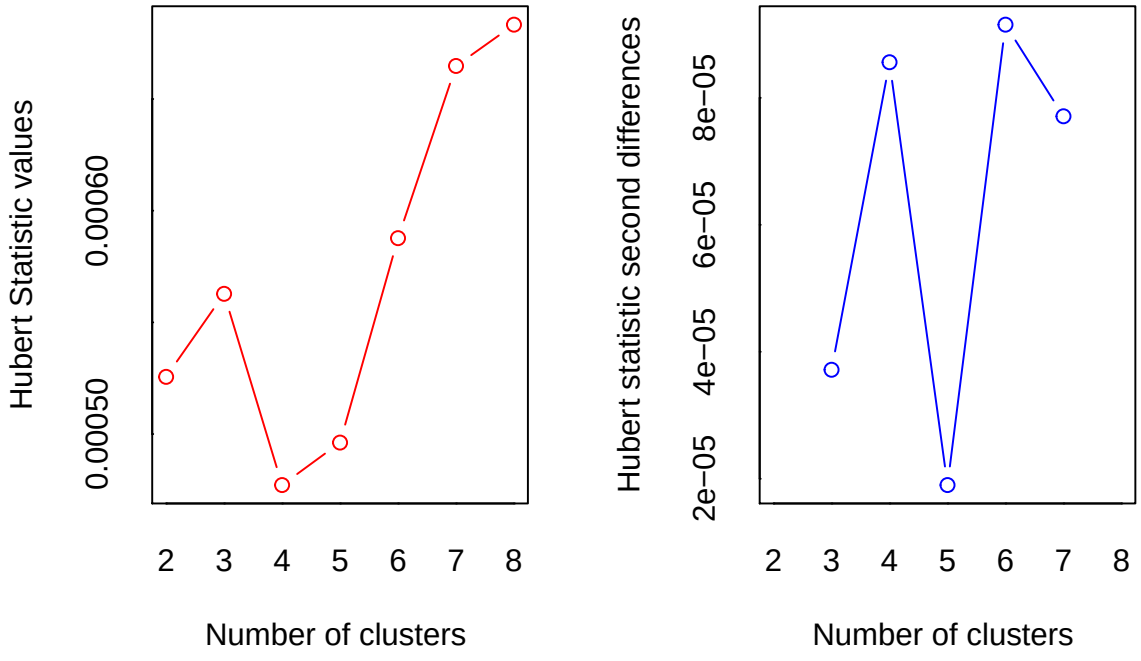
```
ClusterPurity <- function(clusters, classes) { #https://stackoverflow.com/questions/9253843/r-clusterin
  sum(apply(table(classes, clusters), 2, max)) / length(clusters)
}

ClusterPurity(pam.res$clustering, species)
```

```
## [1] 0.6728972
```

```
ClusterPurity(glassCluster$cluster, species)
```

```
## [1] 0.5981308
```

```
# diabetes ----------------------------------------------------------------

diabetes = read.table("pima_indians_diabetes.txt", header = FALSE, sep = ",")
names(diabetes) <- c("No_pregnant", "Plasma_glucose", "Blood_pres", "Skin_thick", "Serum_insu", "BMI",
diabetes$`Class` <- factor(diabetes$`Class`)

diabetes.scaled <- scale(diabetes[, -9 ])
NbClust(diabetes.scaled, distance = "euclidean", min.nc = 2, max.nc = 8, method = "complete", index ="al
```



2.4 Diabetes

```
## [1] 1.603803 7.379597 2.470477 8.670881 4.752407 5.940563
##
## $widestgap
## [1] 8.670881
##
## $sindex
## [1] 0.3699863
##
## $corrected.rand
## [1] 0.2027399
##
## $vi
## [1] 1.920896
```

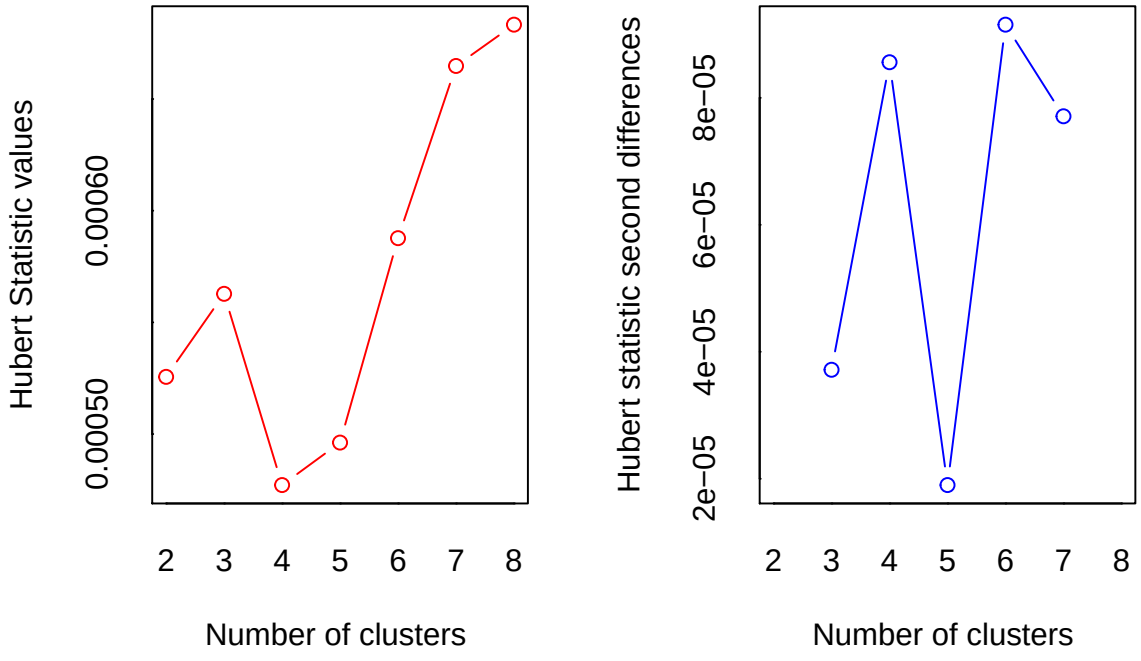
```
ClusterPurity <- function(clusters, classes) { #https://stackoverflow.com/questions/9253843/r-clusterin
  sum(apply(table(classes, clusters), 2, max)) / length(clusters)
}

ClusterPurity(pam.res$clustering, species)
```

```
## [1] 0.6728972
```

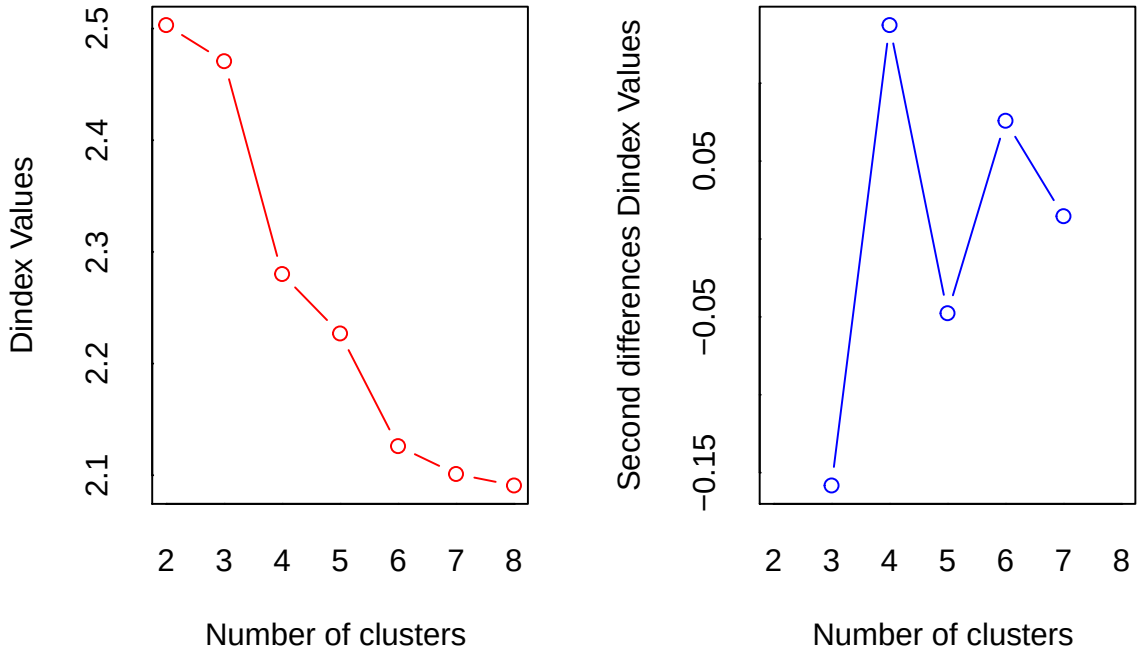
```
ClusterPurity(glassCluster$cluster, species)
```

```
## [1] 0.5981308
```

```
# diabetes ----------------------------------------------------------------

diabetes = read.table("pima_indians_diabetes.txt", header = FALSE, sep = ",")
names(diabetes) <- c("No_pregnant", "Plasma_glucose", "Blood_pres", "Skin_thick", "Serum_insu", "BMI",
diabetes$`Class` <- factor(diabetes$`Class`)

diabetes.scaled <- scale(diabetes[, -9 ])
NbClust(diabetes.scaled, distance = "euclidean", min.nc = 2, max.nc = 8, method = "complete", index ="al
```



```
## *** : 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.
##
```

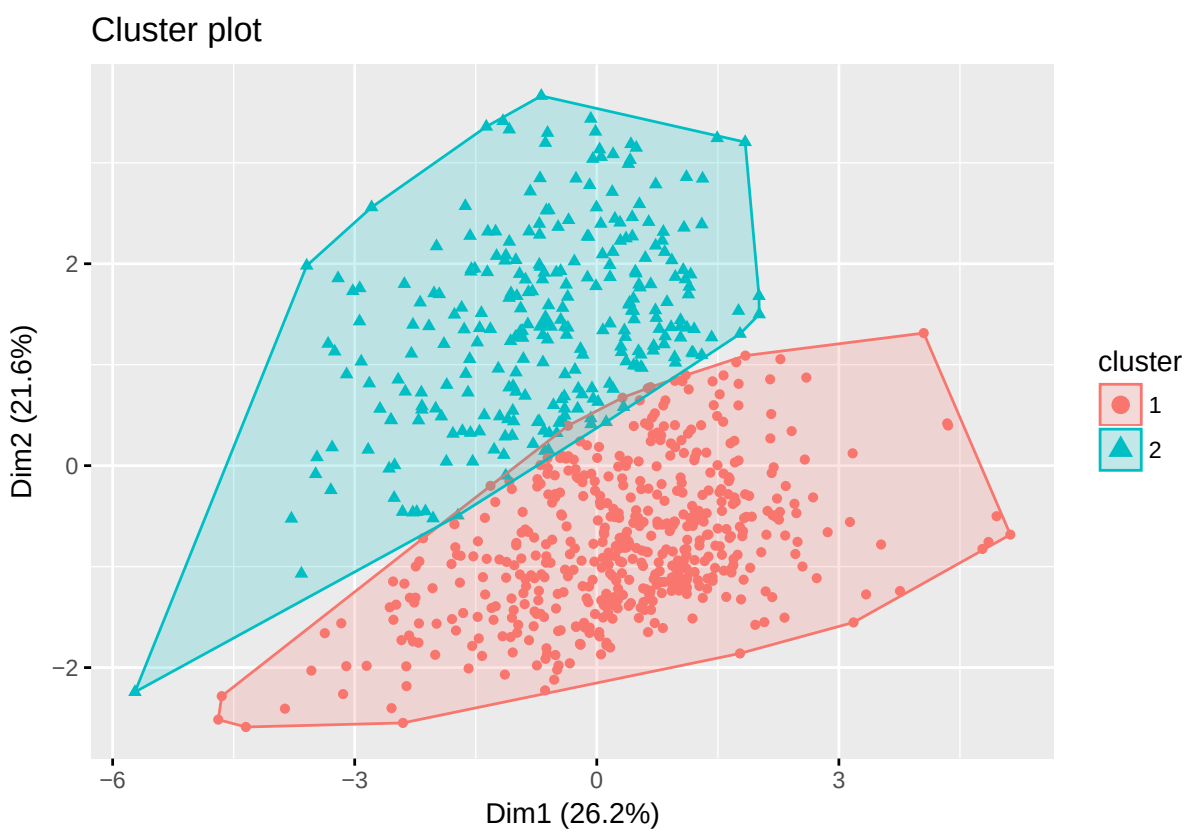



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 8 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 8 proposed 6 as the best number of clusters
## * 3 proposed 8 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************

## $All.index
##        KL      CH Hartigan      CCC     Scott      Marriot   TrCovW
## 2  0.7819 93.2743  24.3959 -14.3692  485.4034 7.354663e+22 417231.5
## 3  1.1246 60.2416 109.6337 -26.0912  692.4726 1.263718e+23 394654.7
## 4  6.7472 82.3534  36.2443 -21.4110 1262.0080 1.070194e+23 319388.9
## 5  0.1394 73.6598  93.2574 -24.2971 1610.6214 1.062052e+23 291431.5
## 6 68.4855 84.6706  30.7749 -18.0861 2311.2495 6.142030e+22 249683.2
```

```
## 7  0.0507 78.4346   9.7913 -19.3052 2546.6467 6.153062e+22 227007.8
## 8  0.8384 69.4021  42.2202 -21.2329 2636.7471 7.147012e+22 222310.7
##     TraceW Friedman  Rubin Cindex     DB Silhouette   Duda Pseudot2  Beale
## 2 5469.937   1.3164 1.1218 0.3008 2.1209     0.1910 0.8255  24.1029 1.1077
## 3 5301.105   1.6527 1.1575 0.3339 1.8903     0.1626 0.8524 112.5335 0.9136
## 4 4636.622   3.1816 1.3234 0.3324 2.0809     0.1302 0.6727  46.7076 2.5449
## 5 4426.622   4.0236 1.3862 0.3503 1.8500     0.1085 0.8024  80.7618 1.2974
## 6 3944.506   5.6857 1.5556 0.3481 1.6092     0.1365 0.8905  39.3622 0.6481
## 7 3791.383   6.2749 1.6184 0.3433 1.7091     0.1414 0.6930   7.0864 2.2031
## 8 3743.222   6.5495 1.6392 0.3459 1.5888     0.1415 0.8932  35.8621 0.6297
##   Ratkowsky      Ball Ptbiserial    Frey McClain   Dunn Hubert SDindex
## 2    0.2095 2734.9685     0.2942 -0.9011  0.2737 0.0604  5e-04  1.9189
## 3    0.1949 1767.0351     0.3031  1.2871  0.2809 0.0670  6e-04  1.8600
## 4    0.2385 1159.1554     0.3043  0.0224  1.3070 0.0732  5e-04  1.9490
## 5    0.2266  885.3243     0.3091 -0.3013  1.3445 0.0774  5e-04  1.7022
## 6    0.2411  657.4177     0.3840 -0.1284  1.4204 0.0818  6e-04  1.6820
## 7    0.2314  541.6262     0.4221 -0.2662  1.5182 0.0844  7e-04  1.8013
## 8    0.2186  467.9027     0.4225  0.2605  1.5183 0.0850  7e-04  1.9226
##   Dindex   SDbw
## 2 2.5031 2.4416
## 3 2.4708 2.2377
## 4 2.2802 1.1362
## 5 2.2270 0.8262
## 6 2.1261 0.7679
## 7 2.1012 0.9007
## 8 2.0909 0.8023
##
## $All.CriticalValues
##   CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2         0.7796            32.2301       0.3553
## 3         0.8610           104.9178       0.5038
## 4         0.7672            29.1296       0.0097
## 5         0.8369            63.9101       0.2400
## 6         0.8359            62.8229       0.7376
## 7         0.5629            12.4235       0.0312
## 8         0.8331            60.0828       0.7534
##
## $Best.nc
##                       KL      CH Hartigan      CCC    Scott      Marriot
## Number_clusters  6.0000  2.0000   3.0000   2.0000   6.0000 6.000000e+00
## Value_Index     68.4855 93.2743  85.2378 -14.3692 700.6281 4.489526e+22
##                    TrCovW   TraceW Friedman   Rubin Cindex     DB
## Number_clusters     4.00   4.0000   6.0000  6.0000 2.0000 8.0000
## Value_Index     75265.77 454.4837   1.6621 -0.1066 0.3008 1.5888
##                 Silhouette   Duda PseudoT2  Beale Ratkowsky     Ball
## Number_clusters      2.000 2.0000   2.0000 2.0000    6.0000   3.0000
## Value_Index          0.191 0.8255  24.1029 1.1077    0.2411 967.9334
##                 PtBiserial Frey McClain  Dunn Hubert SDindex Dindex   SDbw
## Number_clusters     8.0000    1  2.0000 8.000      0   6.000      0 6.0000
## Value_Index         0.4225   NA  0.2737 0.085      0   1.682      0 0.7679
##
## $Best.partition
##   [1] 1 1 1 1 2 1 1 1 2 1 1 1 1 2 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1
##  [36] 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1
```

```
##  [71] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1
## [106] 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 2
## [141] 1 1 1 1 2 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1
## [176] 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1
## [211] 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1
## [246] 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1
## [281] 1 1 1 1 1 1 1 2 2 1 1 1 1 2 2 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1
## [316] 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [351] 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1
## [386] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 1 2 2 1 1 1
## [421] 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1
## [456] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 2 1 1 1 1 2 2 2 1
## [491] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [526] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [561] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 2 1
## [596] 2 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1
## [631] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 2 1 1 1 2 1 1
## [666] 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 2 1 1 2 1 1 2 1 1
## [701] 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 2 2 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 1
## [736] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1
```

```
diabetesCluster <- kmeans(diabetes.scaled, centers=2, nstart = 20)

fviz_cluster(object=diabetesCluster, data = diabetes.scaled, geom = "point",stand = FALSE)#rysowanie
```



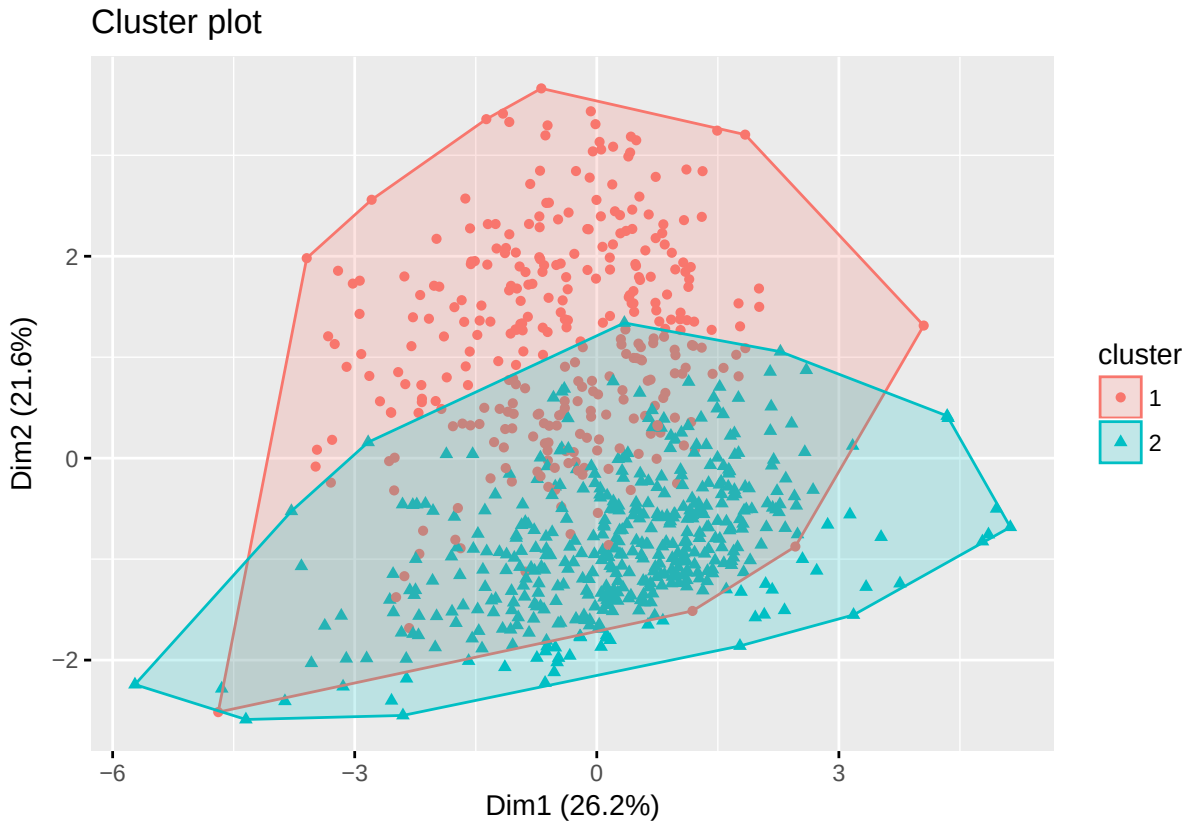

```
table(diabetes$Class, diabetesCluster$cluster)# macierz błędów porównanie czy klastry odpowiadają klaso

##
```

```
##       1   2
##   0 373 127
##   1 123 145
```

```
# klasteryzacja PAM (Partitioning Around Medoids)
library("cluster")
pam.res <- pam(diabetes.scaled, 2)#drugi argument na ile klastrów podzielić
pam.res$cluster
```

```
##   [1] 1 2 1 2 2 1 2 2 1 1 2 1 1 2 1 2 2 1 2 2 2 1 1 1 1 1 2 1 1 1 2 2 1 1
##  [36] 1 1 1 2 1 2 1 1 1 1 1 2 2 1 2 2 2 1 1 1 2 1 2 1 2 2 1 2 2 1 1 1 1 2 1
##  [71] 2 2 1 2 2 2 1 2 2 2 2 2 1 2 1 2 1 2 1 2 2 2 1 1 2 1 2 2 2 2 1 2 2 2 2
## [106] 2 1 1 2 2 2 1 2 2 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 1 1 1 2 1 2 2 2 2 2 2
## [141] 1 1 2 1 2 1 1 1 1 2 2 1 1 2 1 1 2 2 2 1 1 1 2 2 2 1 2 1 2 2 1 2 2 2 2
## [176] 1 1 2 1 1 1 2 2 2 1 1 1 1 1 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2 1 1 1 1 2 1
## [211] 2 2 1 2 1 1 2 1 1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 1 2
## [246] 1 1 2 2 2 1 2 2 2 1 2 2 2 2 1 1 2 2 1 2 1 2 2 2 2 1 2 1 2 1 2 1 2 1 2
## [281] 2 1 1 1 1 1 2 2 2 1 2 2 2 2 1 1 2 2 1 1 2 2 1 2 1 2 1 2 2 2 1 2 2 2 1
## [316] 2 2 1 2 1 2 2 2 1 2 2 2 1 2 1 1 2 2 1 2 2 2 1 1 1 2 2 2 1 1 1 2 2 2 2
## [351] 2 1 1 2 2 1 2 1 1 2 2 1 1 1 2 2 1 2 2 1 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2
## [386] 2 1 1 1 2 2 2 2 1 1 2 1 2 2 1 2 1 1 1 1 2 1 2 1 2 1 2 2 2 2 2 2 1 2 2
## [421] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 1 1 2 1 1 2 2 1 2 1 2 2 2 2 2 2 2 1 2
## [456] 1 1 2 1 1 1 2 1 1 1 2 2 2 2 2 2 2 2 1 2 1 2 1 1 1 2 2 2 2 2 2 2 1 2 1
## [491] 2 1 2 1 2 1 2 2 1 1 2 2 1 1 1 1 1 2 2 1 1 2 1 2 2 2 1 1 1 1 2 2 2 1 2
## [526] 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 1 2 1 2 2 1 1 2 1 1 2 2 1 2 2 1 2 1 1 1
## [561] 1 2 2 1 2 2 2 1 1 2 2 2 2 2 2 2 1 2 1 1 2 1 1 1 1 2 1 1 1 2 1 2 1 2 1
## [596] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 1 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2
## [631] 1 2 2 2 1 1 1 2 1 2 2 2 1 2 2 2 1 2 1 2 2 2 2 2 2 2 2 1 1 2 1 1 1 1 1
## [666] 2 1 1 1 1 1 2 1 2 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 1 1 2 1 2 1 2 2 2 2
## [701] 2 1 1 2 2 2 1 2 1 2 2 1 1 2 2 1 1 1 2 1 1 2 1 1 1 1 2 2 1 2 1 1 2 2 1
## [736] 2 2 1 2 2 1 2 2 1 1 1 2 2 2 1 2 2 2 2 2 1 1 1 1 2 1 2 1 1 1 2 1 1 2
```

```
fviz_cluster(object=pam.res, data=diabetes.scaled, stand = FALSE, geom = "point")
```




```
table(pam.res$clustering, diabetes$Class)
```

```
##
##       0   1
##   1 165 157
##   2 335 111
```

```
# porównanie klasteryzacji
library(fpc)
#cluster.stats(d=dist(iris.scaled), irisCluster$cluster, pam.res$clustering)
species <- as.numeric(diabetes$Class)#przekodowanie na wartości numeryczne bo się wysypuje
cluster.stats(d=dist(diabetes.scaled), species, pam.res$clustering)
```

```
## $n
## [1] 768
##
## $cluster.number
## [1] 2
##
## $cluster.size
## [1] 500 268
##
## $min.cluster.size
## [1] 268
##
## $noisen
## [1] 0
##
```

```
## $diameter
## [1] 11.97153 12.00814
##
## $average.distance
## [1] 3.410204 4.049018
##
## $median.distance
## [1] 3.242192 3.896962
##
## $separation
## [1] 0.3453488 0.3453488
##
## $average.toother
## [1] 4.003447 4.003447
##
## $separation.matrix
##           [,1]      [,2]
## [1,] 0.0000000 0.3453488
## [2,] 0.3453488 0.0000000
##
## $ave.between.matrix
##          [,1]     [,2]
## [1,] 0.000000 4.003447
## [2,] 4.003447 0.000000
##
## $average.between
## [1] 4.003447
##
## $average.within
## [1] 3.552581
##
## $n.between
## [1] 134000
##
## $n.within
## [1] 160528
##
## $max.diameter
## [1] 12.00814
##
## $min.separation
## [1] 0.3453488
##
## $within.cluster.ss
## [1] 5778.189
##
## $clus.avg.silwidths
##          1          2
##  0.1510519 -0.0179445
##
## $avg.silwidth
## [1] 0.09207918
##
## $g2
```

```
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.1637623
##
## $dunn
## [1] 0.02875956
##
## $dunn2
## [1] 0.9887453
##
## $entropy
## [1] 0.6467994
##
## $wb.ratio
## [1] 0.8873805
##
## $ch
## [1] 47.43409
##
## $cwidegap
## [1] 5.420375 4.554952
##
## $widestgap
## [1] 5.420375
##
## $sindex
## [1] 0.661966
##
## $corrected.rand
## [1] 0.07583667
##
## $vi
## [1] 1.265925
```

```
cluster.stats(d=dist(diabetes.scaled), species, diabetesCluster$cluster)
```

```
## $n
## [1] 768
##
## $cluster.number
## [1] 2
##
## $cluster.size
## [1] 500 268
##
## $min.cluster.size
## [1] 268
##
## $noisen
## [1] 0
##
```

```
## $diameter
## [1] 11.97153 12.00814
##
## $average.distance
## [1] 3.410204 4.049018
##
## $median.distance
## [1] 3.242192 3.896962
##
## $separation
## [1] 0.3453488 0.3453488
##
## $average.toother
## [1] 4.003447 4.003447
##
## $separation.matrix
##           [,1]      [,2]
## [1,] 0.0000000 0.3453488
## [2,] 0.3453488 0.0000000
##
## $ave.between.matrix
##          [,1]     [,2]
## [1,] 0.000000 4.003447
## [2,] 4.003447 0.000000
##
## $average.between
## [1] 4.003447
##
## $average.within
## [1] 3.552581
##
## $n.between
## [1] 134000
##
## $n.within
## [1] 160528
##
## $max.diameter
## [1] 12.00814
##
## $min.separation
## [1] 0.3453488
##
## $within.cluster.ss
## [1] 5778.189
##
## $clus.avg.silwidths
##          1          2
##  0.1510519 -0.0179445
##
## $avg.silwidth
## [1] 0.09207918
##
## $g2
```

```
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.1637623
##
## $dunn
## [1] 0.02875956
##
## $dunn2
## [1] 0.9887453
##
## $entropy
## [1] 0.6467994
##
## $wb.ratio
## [1] 0.8873805
##
## $ch
## [1] 47.43409
##
## $cwidegap
## [1] 5.420375 4.554952
##
## $widestgap
## [1] 5.420375
##
## $sindex
## [1] 0.661966
##
## $corrected.rand
## [1] 0.1139329
##
## $vi
## [1] 1.21609
```

```
ClusterPurity <- function(clusters, classes) { #https://stackoverflow.com/questions/9253843/r-clusterin
  sum(apply(table(classes, clusters), 2, max)) / length(clusters)
}

ClusterPurity(pam.res$clustering, species)
```

```
## [1] 0.6510417
```

```
ClusterPurity(diabetesCluster$cluster, species)
```

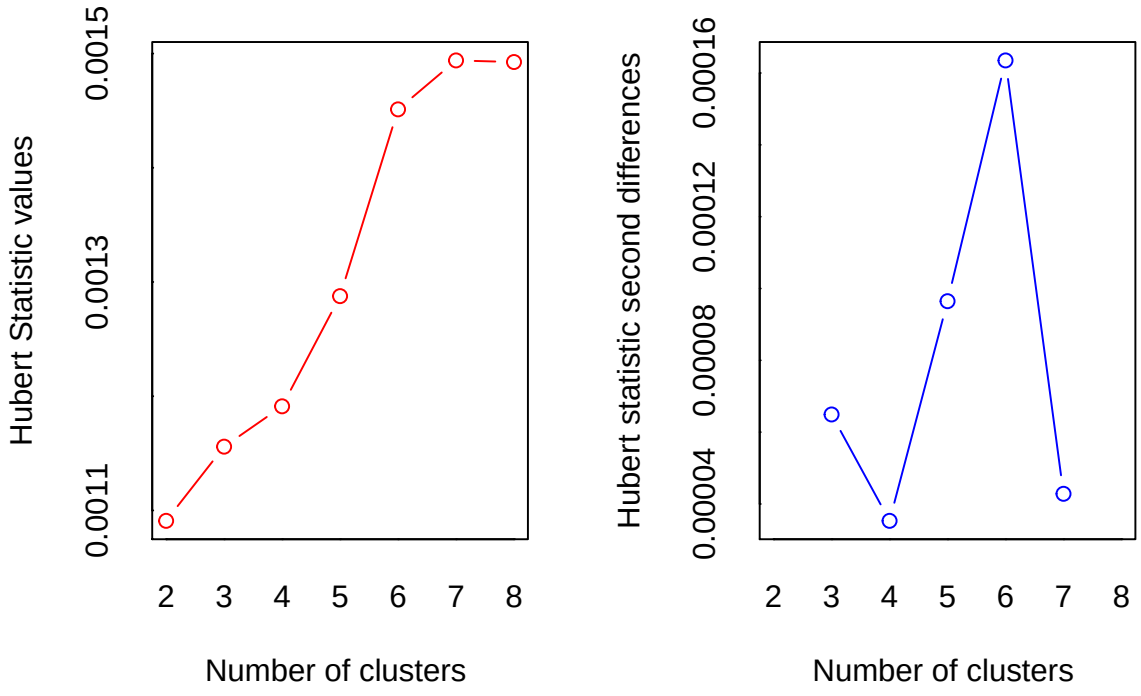
```
## [1] 0.6744792
```

```
# Cluster_data ------------------------------------------------------------


seeds = read.table("seeds_dataset.txt", header = FALSE, sep = "\t")
names(seeds) <- c("area A", "perimeter P", "compactness C", "length of kernel", "width of kernel", "asy
```

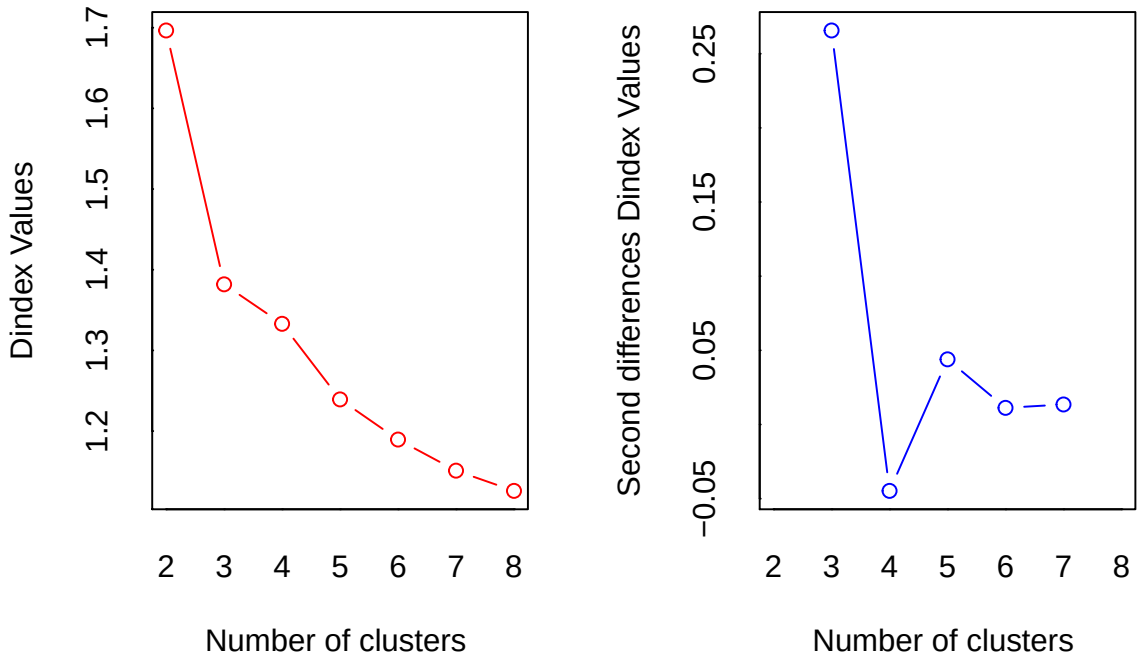
2.5 Seeds

```
# Cluster_data --------------------------------------------------------------
seeds = read.table("seeds_dataset.txt", header = FALSE, sep = "\t")
names(seeds) <- c("area A", "perimeter P", "compactness C", "length of kernel", "width of kernel",
```

```
seeds.scaled <- scale(seeds[,-8])
NbClust(seeds.scaled, distance = "euclidean", min.nc = 2, max.nc = 8, method = "complete", index ="all"
```




```
## *** : 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.
##
```

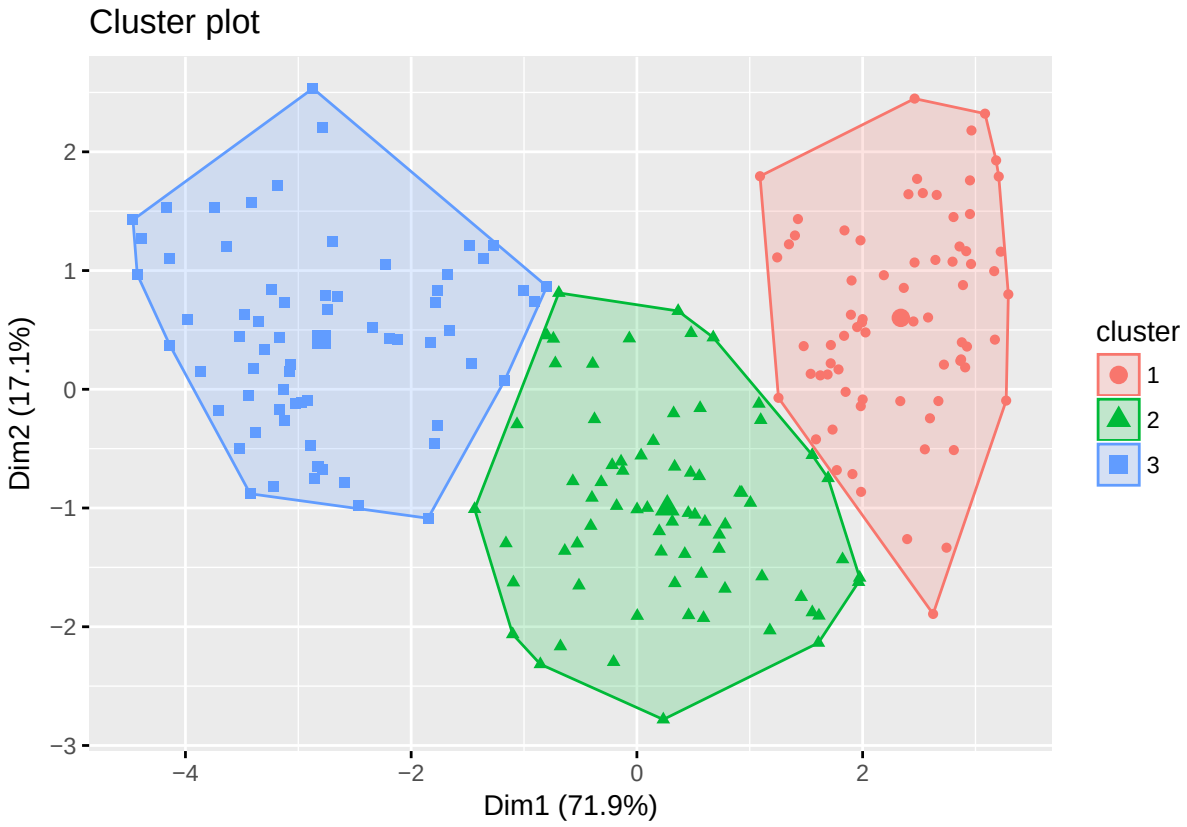



```
## *** : The D index is a graphical method of determining the number of clusters.
```

```
##                 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:
## * 7 proposed 2 as the best number of clusters
## * 16 proposed 3 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
##
##
## *****************************************************************
## $All.index
##        KL       CH Hartigan     CCC     Scott   Marriot    TrCovW   TraceW
## 2  3.1198 230.3102  96.7168 -0.7577  371.0497 269515259 22890.498 694.2663
## 3 18.9130 216.0153  21.4134  1.0174  604.6831 199340620  8573.466 473.9069
## 4  0.2086 165.2432  32.2670 -1.4760  699.5577 225562095  7840.757 429.4789
## 5  2.1491 150.6764  19.8768 -8.6338  811.6064 206710080  4863.020 371.3172
## 6  1.9457 135.5397  13.8064 -8.5984  915.9796 181081108  4546.658 338.4967
## 7  3.6594 122.2927   9.1511 -8.9359  981.5402 180377719  3999.012 317.0398
## 8  0.1881 110.3088  13.7494 -9.5680 1022.4632 193880978  3763.283 303.3644
##   Friedman  Rubin Cindex     DB Silhouette   Duda Pseudot2  Beale
## 2 1064.239 2.1073 0.3804 0.8437     0.4520 0.5935  95.8851 3.1050
## 3 1531.880 3.0871 0.3566 1.0310     0.3502 0.8207  19.2289 0.9865
## 4 1608.988 3.4065 0.3642 1.2766     0.3149 0.6497  39.9004 2.4291
## 5 1798.662 3.9400 0.3583 1.2332     0.2937 0.7843  18.1505 1.2369
## 6 2057.273 4.3221 0.4186 1.5034     0.2174 0.5759  17.6758 3.2283
## 7 2106.664 4.6146 0.4118 1.3515     0.2192 0.8152  11.3364 1.0149
## 8 2128.176 4.8226 0.4084 1.5685     0.1574 0.4816  12.9173 4.5369
##   Ratkowsky     Ball Ptbiserial   Frey McClain   Dunn Hubert SDindex
## 2    0.4627 347.1331     0.6485 1.0795  0.4200 0.0955 0.0011  1.5276
## 3    0.4650 157.9690     0.6118 0.7444  0.9234 0.1102 0.0012  1.6023
## 4    0.4150 107.3697     0.5970 1.2306  1.1071 0.1037 0.0012  1.8128
## 5    0.3839  74.2634     0.5559 0.8488  1.4135 0.0907 0.0013  1.7688
## 6    0.3556  56.4161     0.5169 0.4477  1.8110 0.1005 0.0015  2.1046
## 7    0.3329  45.2914     0.5136 3.4012  1.8662 0.1006 0.0015  1.8107
## 8    0.3133  37.9206     0.4677 0.1280  2.3213 0.1012 0.0015  2.5437
##   Dindex   SDbw
## 2 1.6965 1.1356
## 3 1.3819 0.6493
## 4 1.3330 0.9597
## 5 1.2393 0.3551
## 6 1.1895 0.3357
## 7 1.1510 0.3090
## 8 1.1259 0.2507
##
## $All.CriticalValues
##   CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2         0.7741            40.8596       0.0030
```

```
## 3         0.7395            30.9969       0.4399
## 4         0.7246            28.1311       0.0187
## 5         0.7140            26.4359       0.2807
## 6         0.5936            16.4297       0.0031
## 7         0.6860            22.8852       0.4203
## 8         0.4792            13.0421       0.0002
##
## $Best.nc
##                     KL       CH Hartigan    CCC    Scott  Marriot   TrCovW
## Number_clusters  3.000   2.0000   3.0000 3.0000   3.0000        3     3.00
## Value_Index     18.913 230.3102  75.3034 1.0174 233.6334 96396113 14317.03
##                   TraceW Friedman   Rubin Cindex     DB Silhouette   Duda
## Number_clusters   3.0000   3.0000  3.0000 3.0000 2.0000      2.000 3.0000
## Value_Index     175.9313 467.6405 -0.6605 0.3566 0.8437      0.452 0.8207
##                 PseudoT2  Beale Ratkowsky     Ball PtBiserial   Frey
## Number_clusters   3.0000 3.0000     3.000   3.0000     2.0000 2.0000
## Value_Index      19.2289 0.9865     0.465 189.1642     0.6485 1.0795
##                 McClain   Dunn Hubert SDindex Dindex   SDbw
## Number_clusters    2.00 3.0000      0  2.0000      0 8.0000
## Value_Index        0.42 0.1102      0  1.5276      0 0.2507
##
## $Best.partition
##   [1] 1 1 1 1 1 1 1 2 1 1 1 3 1 1 3 3 1 1 3 1 1 1 3 1 1 3 3 1 3 3 1 1 1 1
##  [36] 1 1 1 1 3 1 1 3 2 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 3
##  [71] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [106] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
seedsCluster <- kmeans(seeds.scaled, centers=3, nstart = 20)

fviz_cluster(object=seedsCluster, data = seeds.scaled, geom = "point",stand = FALSE)#rysowanie
```



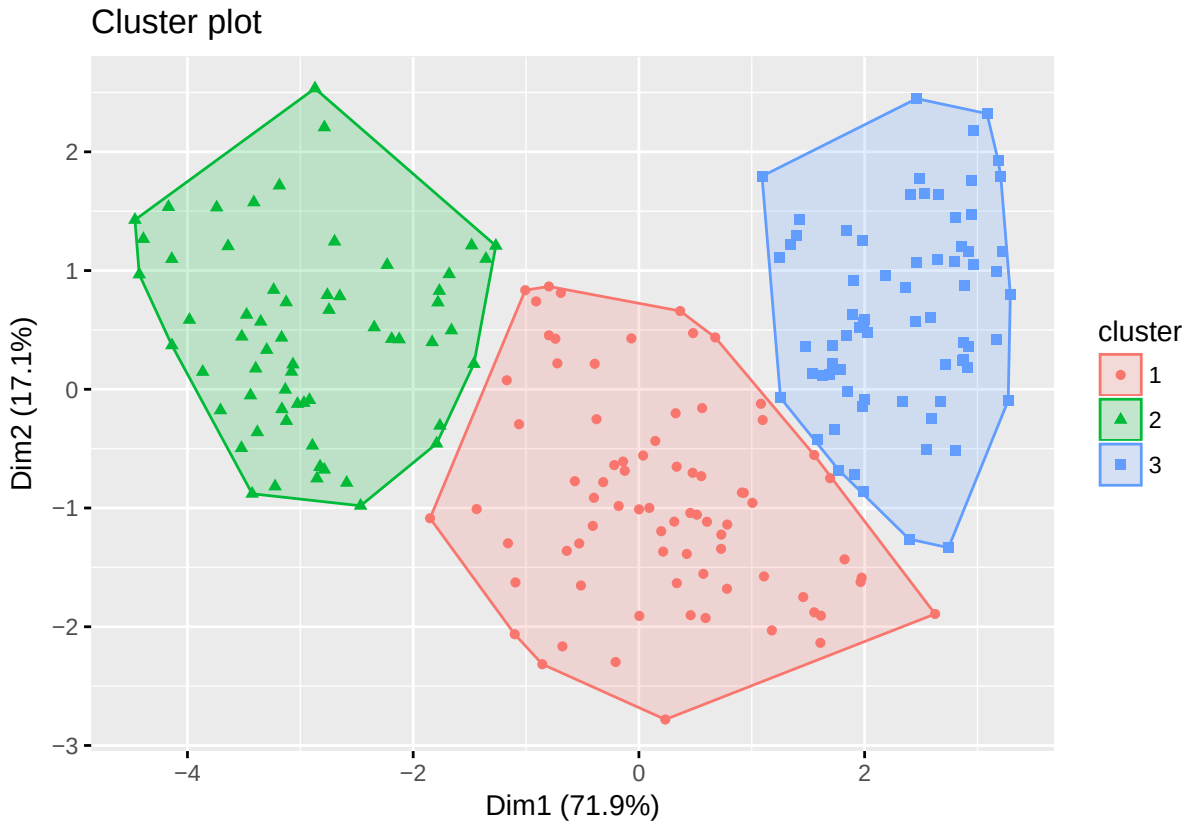

```
table(seeds$class, seedsCluster$cluster)# macierz błędów porównanie czy klastry odpowiadają klasom
```

```
##
##      1  2  3
##   1  6 62  2
##   2  0  5 65
##   3 66  4  0
```

```
# klasteryzacja PAM (Partitioning Around Medoids)
library("cluster")
pam.res <- pam(seeds.scaled, 3)#drugi argument na ile klastrów podzielić
pam.res$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1 1 3 1 1 1 1 1 3
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
## [106] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 1 1 1 2 1 1 2
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3
## [176] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 3 1 3 3 3 3 3 3 3 3
```

```
fviz_cluster(object=pam.res, data=seeds.scaled, stand = FALSE, geom = "point")
```




```
table(pam.res$clustering, seeds$class)
```

```
##
##      1  2  3
##   1 64  9  4
##   2  1 61  0
##   3  5  0 66
```

```
# porównanie klasteryzacji
library(fpc)
#cluster.stats(d=dist(iris.scaled), irisCluster$cluster, pam.res$clustering)
species <- as.numeric(seeds$class)#przekodowanie na wartości numeryczne bo się wysypuje
cluster.stats(d=dist(seeds.scaled), species, pam.res$clustering)
```

```
## $n
## [1] 210
##
## $cluster.number
## [1] 3
##
## $cluster.size
## [1] 70 70 70
##
## $min.cluster.size
## [1] 70
##
## $noisen
## [1] 0
```

```
##
## $diameter
## [1] 4.855676 5.080527 4.922123
##
## $average.distance
## [1] 1.951704 2.027024 1.882386
##
## $median.distance
## [1] 1.861343 1.995320 1.769879
##
## $separation
## [1] 0.3933331 0.6683238 0.3933331
##
## $average.toother
## [1] 3.455563 4.541432 4.210428
##
## $separation.matrix
##           [,1]      [,2]      [,3]
## [1,] 0.0000000 0.6683238 0.3933331
## [2,] 0.6683238 0.0000000 1.7935192
## [3,] 0.3933331 1.7935192 0.0000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 3.786567 3.124559
## [2,] 3.786567 0.000000 5.296298
## [3,] 3.124559 5.296298 0.000000
##
## $average.between
## [1] 4.069141
##
## $average.within
## [1] 1.953704
##
## $n.between
## [1] 14700
##
## $n.within
## [1] 7245
##
## $max.diameter
## [1] 5.080527
##
## $min.separation
## [1] 0.3933331
##
## $within.cluster.ss
## [1] 465.566
##
## $clus.avg.silwidths
##         1         2         3
## 0.2953166 0.4325643 0.3747741
##
## $avg.silwidth
```

```
## [1] 0.3675517
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.612488
##
## $dunn
## [1] 0.07741973
##
## $dunn2
## [1] 1.541452
##
## $entropy
## [1] 1.098612
##
## $wb.ratio
## [1] 0.4801269
##
## $ch
## [1] 221.7396
##
## $cwidegap
## [1] 1.369539 1.208413 1.676707
##
## $widestgap
## [1] 1.676707
##
## $sindex
## [1] 0.5743573
##
## $corrected.rand
## [1] 0.7469561
##
## $vi
## [1] 0.6264608
```

```
cluster.stats(d=dist(seeds.scaled), species, seedsCluster$cluster)
```

```
## $n
## [1] 210
##
## $cluster.number
## [1] 3
##
## $cluster.size
## [1] 70 70 70
##
## $min.cluster.size
## [1] 70
##
```

```
## $noisen
## [1] 0
##
## $diameter
## [1] 4.855676 5.080527 4.922123
##
## $average.distance
## [1] 1.951704 2.027024 1.882386
##
## $median.distance
## [1] 1.861343 1.995320 1.769879
##
## $separation
## [1] 0.3933331 0.6683238 0.3933331
##
## $average.toother
## [1] 3.455563 4.541432 4.210428
##
## $separation.matrix
##           [,1]      [,2]      [,3]
## [1,] 0.0000000 0.6683238 0.3933331
## [2,] 0.6683238 0.0000000 1.7935192
## [3,] 0.3933331 1.7935192 0.0000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 3.786567 3.124559
## [2,] 3.786567 0.000000 5.296298
## [3,] 3.124559 5.296298 0.000000
##
## $average.between
## [1] 4.069141
##
## $average.within
## [1] 1.953704
##
## $n.between
## [1] 14700
##
## $n.within
## [1] 7245
##
## $max.diameter
## [1] 5.080527
##
## $min.separation
## [1] 0.3933331
##
## $within.cluster.ss
## [1] 465.566
##
## $clus.avg.silwidths
##         1         2         3
## 0.2953166 0.4325643 0.3747741
```

```
##
## $avg.silwidth
## [1] 0.3675517
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.612488
##
## $dunn
## [1] 0.07741973
##
## $dunn2
## [1] 1.541452
##
## $entropy
## [1] 1.098612
##
## $wb.ratio
## [1] 0.4801269
##
## $ch
## [1] 221.7396
##
## $cwidegap
## [1] 1.369539 1.208413 1.676707
##
## $widestgap
## [1] 1.676707
##
## $sindex
## [1] 0.5743573
##
## $corrected.rand
## [1] 0.7732937
##
## $vi
## [1] 0.5978132
```

```
ClusterPurity <- function(clusters, classes) { #https://stackoverflow.com/questions/9253843/r-clusterin
  sum(apply(table(classes, clusters), 2, max)) / length(clusters)
}

ClusterPurity(pam.res$clustering, species)
```

```
## [1] 0.9095238
```

```
ClusterPurity(seedsCluster$cluster, species)
```

```
## [1] 0.9190476
```

3 Wnioski

1. Czy dane muszą być dyskretyzowane i/lub normalizowane?
Normalizacja sprawi, że każdy atrybut będzie miał takie samo znaczenie przy użyciu euklidesowej miary dystansu, bez względu na to czy wartości są rozpięte na dużym czy małym przedziale.
2. Czy kroswalidacja jest potrzebna?
Przy miarach wewnętrznych klastrów nie. Przy zewnętrznych takich jak F-score tak. 3. Czym różnią się oba algorytmy?
Centrami klastrów 4. Który z algorytmów jest podatniejszy na szum w danych i „outliery"?
K-means 5. Czy istnieje potrzeba powtarzania uzyskanych wyników?
Klasteryzacja w powyższych przypadkach polega na wybraniu losowych punktów. Losowość tego procesu wpływa na to że powtarzając obliczenia możemy uzyskać inne wyniki.
6. Czy sposób mierzenia odległości (miar) wpływa na skuteczność algorytmów?
Tak jeśli są to miary euklidesowa i manhattan jeśli chodzi o miary typu purity czy średnia odległość między punktami wtedy osoba analizująca podział danych określa jak bardzo te wyliczenia do czegokolwiek się nadają.
7. Co mierzą wskazane miary jakości klasyfikacji i jakie są wartości „optymalne". Np. jakie wartości może przyjąć miara ABC gdy mamy tylko jeden klaster, a jaką wartość jeśli mamy tyle klastrów co instancji (danych)?
Wszystkie te miary są dyskusyjne np. podzielenie klastrów na tyle ile występuje obiektów gwarantuje najwyższy wskaźnik Purity ale uzyskany podział jest całkowicie bezużyteczny. Miary wewnętrzne określają w jaki sposób dane zostały podzielone z czego możemy następnie wnioskować.