Clustering

Erick Marroquín

2026-01-26

Ejemplo de clustering

En este documento se presenta un ejemplo completo de **agrupamiento (clustering)** utilizando técnicas de aprendizaje no supervisado.
El objetivo del clustering es **identificar grupos naturales en los datos**, sin utilizar una variable respuesta.

```r
library(cluster)
library(e1071)
library(mclust)
library(fpc)
library(NbClust)
library(factoextra)
library(hopkins)
library(GGally)
library(FeatureImpCluster)
library(pheatmap)
```

Descripción de los Datos

Vamos a trabajar con la base de datos **iris**, la cual contiene 150 observaciones y 5 variables.

```r
datos <- iris
set.seed(123)
datos <- datos[complete.cases(iris),]
summary(datos)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##                 
##                 
##                 
```

```r
datos[,1:4] <- scale(datos[,1:4])
```

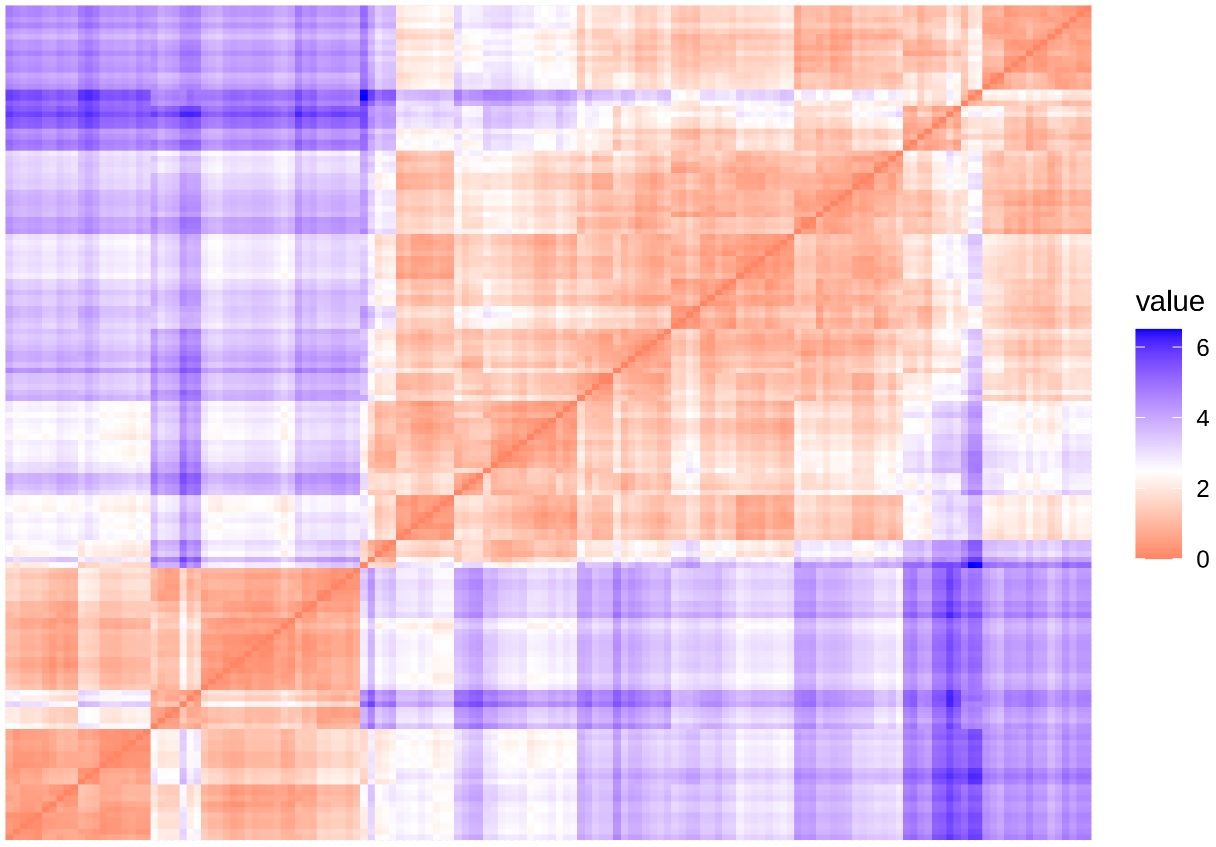
¿Hacemos Agrupamiento?

```
set.seed(123)
hopkins(datos[,1:4])
```

```
## [1] 0.9980312
```

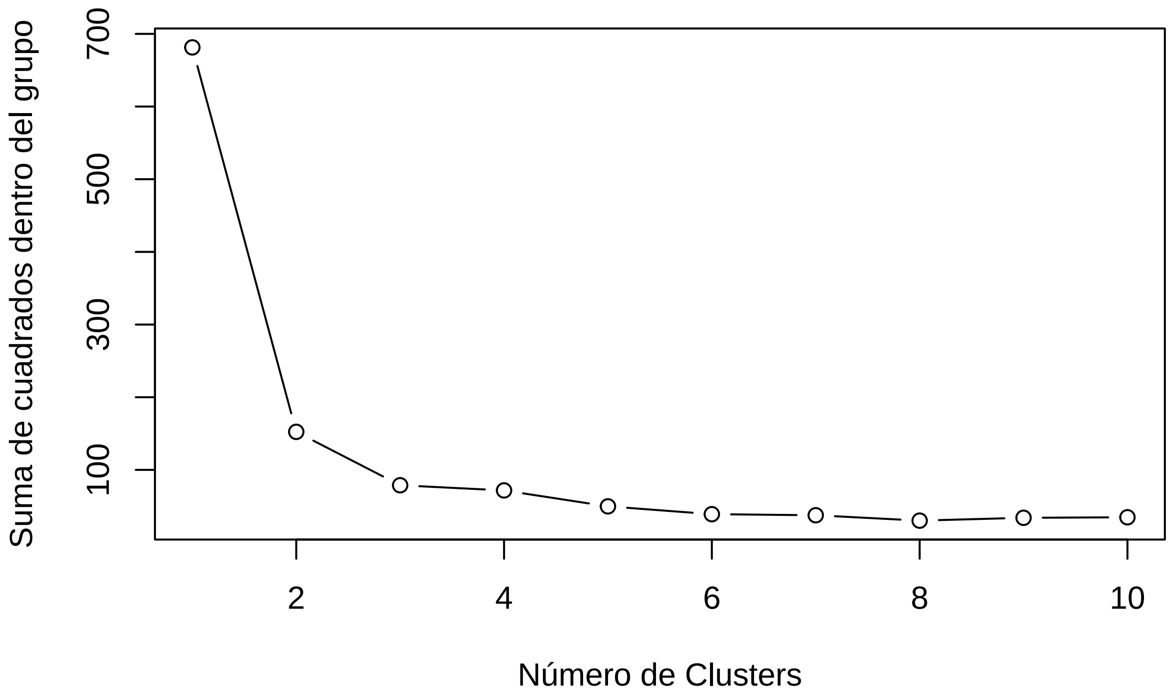
```
datos_dist <- dist(datos[,1:4])
fviz_dist(datos_dist, show_labels = FALSE)
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## i The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

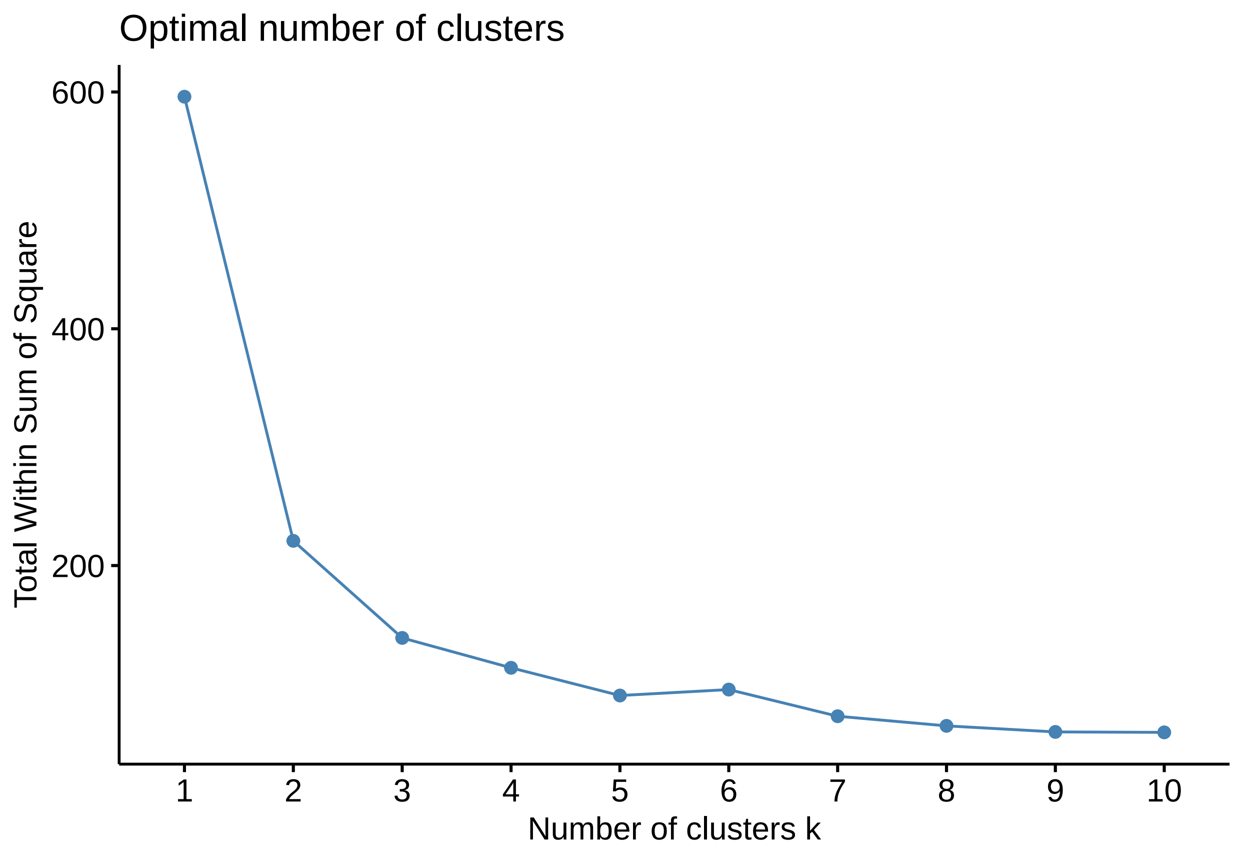


¿Cuántos grupos debemos hacer?

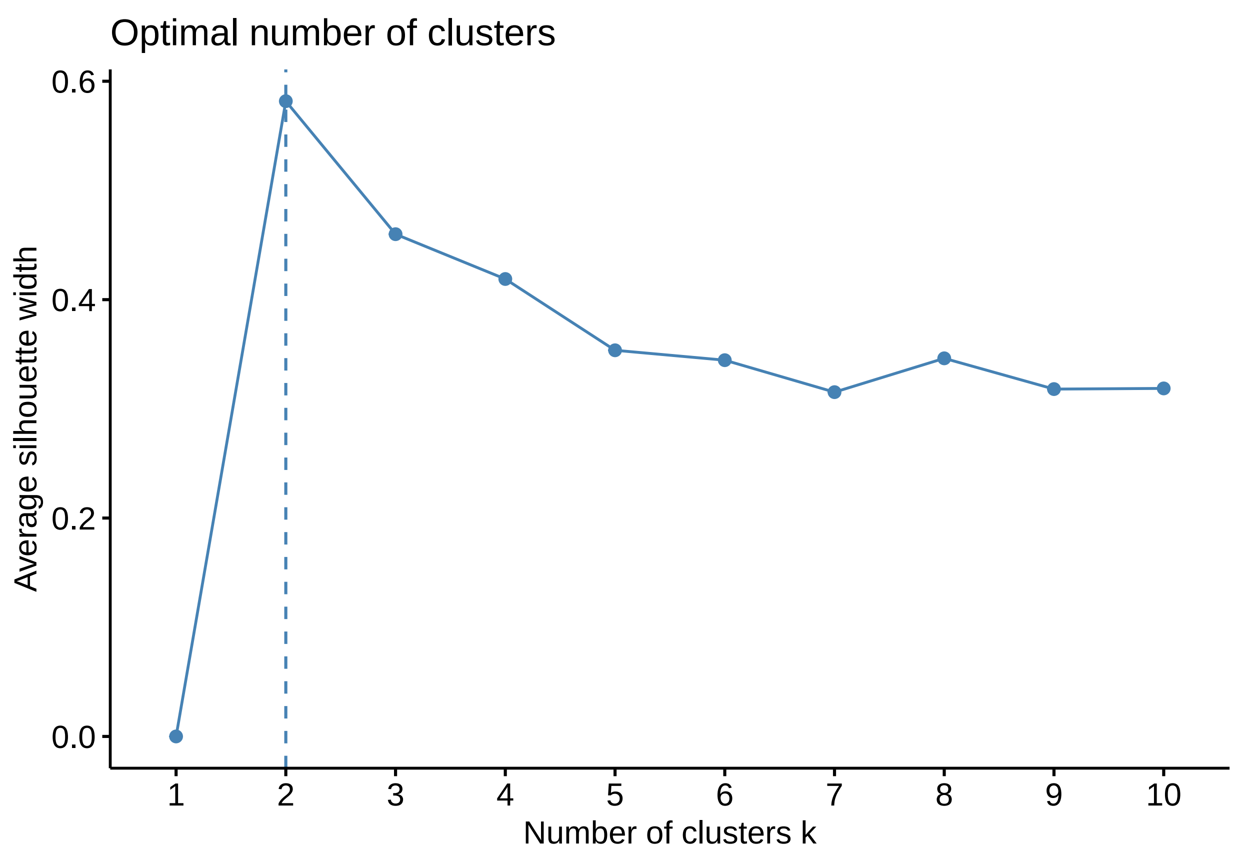
```
wss = 0
for (i in 1:10)
  wss[i] <- sum(kmeans(iris[,1:4], centers=i)$withinss)

plot(1:10, wss, type="b",
     xlab="Número de Clusters",
     ylab="Suma de cuadrados dentro del grupo")
```



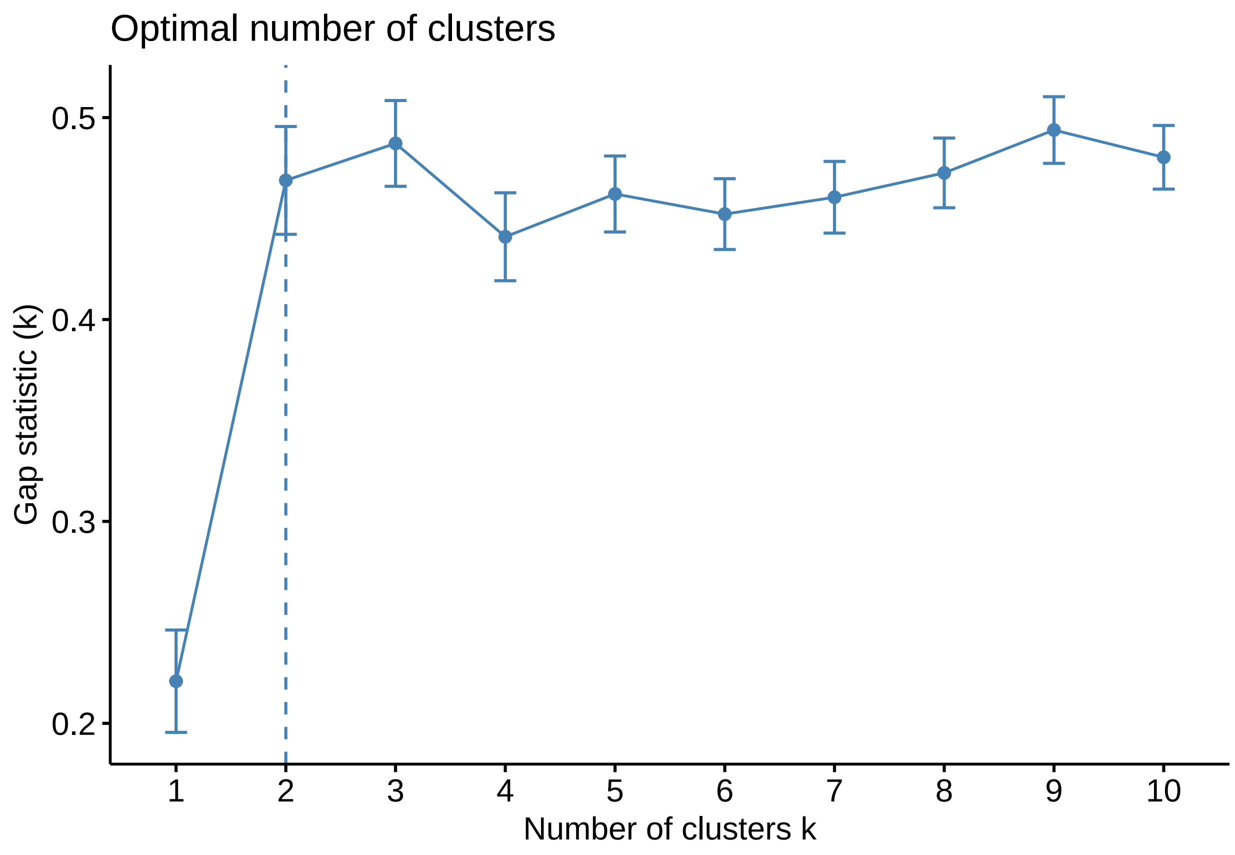
```
fviz_nbclust(datos[,1:4], kmeans, method = "wss")
```



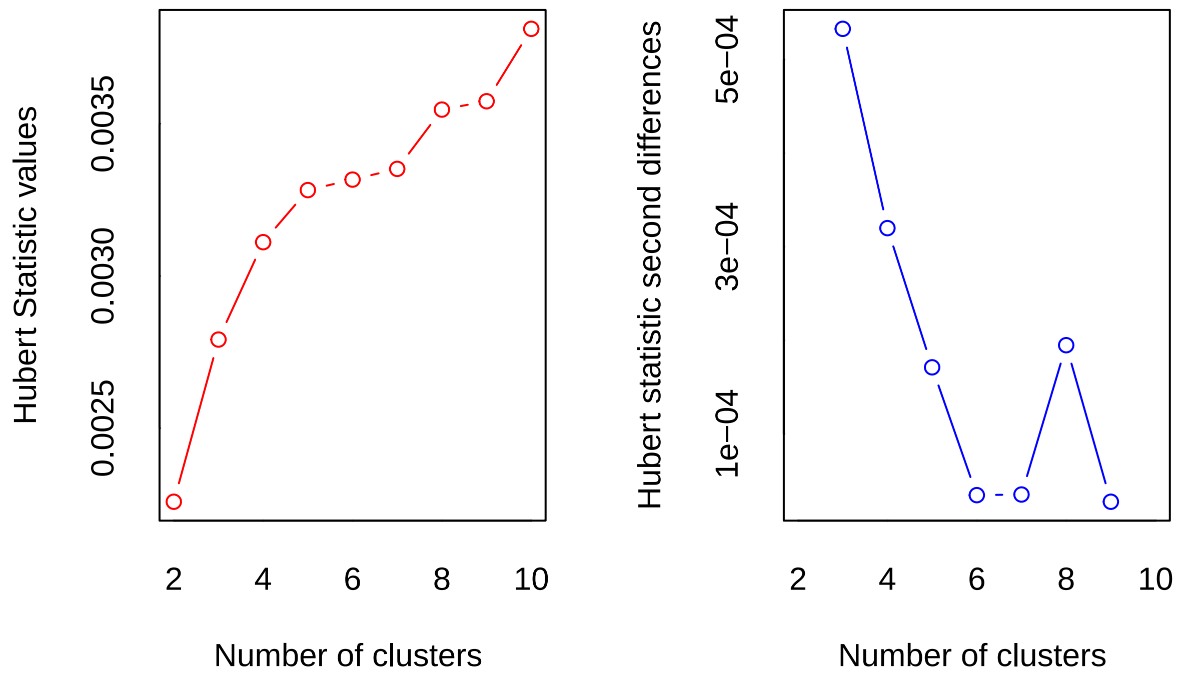
```
fviz_nbclust(datos[,1:4], kmeans, method = "silhouette")
```



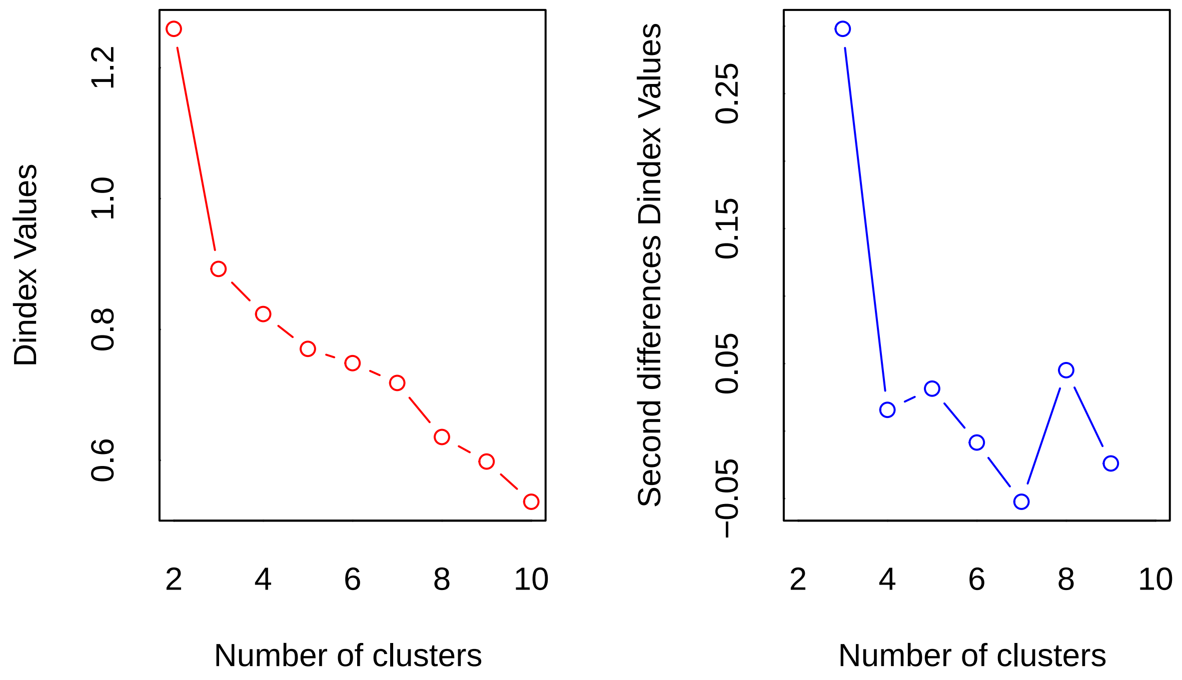
```r
fviz_nbclust(datos[,1:4], kmeans,
             nstart = 25,
             method = "gap_stat",
             nboot = 50,
             verbose = FALSE)
```



```r
NbClust(datos[,1:4],
        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 Ratkowsky
## 2   38.3959  2.0245 0.2978 0.9735     0.4408 0.2934 170.9649 5.7326    0.4606
## 3   53.5029  3.8991 0.3247 0.8581     0.4496 0.7005  32.0629 1.0185    0.4958
## 4   63.1994  4.7802 0.3085 0.9581     0.4106 0.5462  39.0434 1.9637    0.4431
## 5   64.5874  5.5975 0.3486 0.9262     0.3521 0.6102  14.0540 1.4752    0.4047
## 6   71.9579  6.0078 0.3922 0.9569     0.3107 0.3089  20.1316 4.8602    0.3722
## 7   72.7720  6.5379 0.4288 0.8597     0.3076 0.6359  36.6526 1.3613    0.3476
## 8   80.5472  8.3474 0.3937 0.9451     0.3303 0.6792  19.8416 1.1140    0.3314
## 9   85.6861  9.5189 0.3828 0.9766     0.3421 0.4147  56.4663 3.3249    0.3151
## 10  89.4399 11.7002 0.3895 0.9858     0.3266 0.5931  18.5213 1.5969    0.3023
##        Ball Ptbiserial   Frey McClain   Dunn Hubert SDindex Dindex   SDbw
## 2  147.1933     0.5970 0.0616  0.5265 0.0412 0.0023  2.0088 1.2595 1.1052
## 3   50.9523     0.7169 0.7401  0.6296 0.0580 0.0028  1.5641 0.8925 0.3975
## 4   31.1704     0.7030 0.8948  0.7655 0.0623 0.0031  1.8915 0.8236 0.4585
## 5   21.2952     0.6892 1.3879  0.8368 0.0740 0.0033  1.5738 0.7703 0.3046
## 6   16.5341     0.6817 0.4503  0.8664 0.0842 0.0033  1.8071 0.7485 0.2503
## 7   13.0230     0.6815 1.1435  0.8708 0.0927 0.0034  1.6904 0.7183 0.1602
## 8    8.9250     0.5977 1.0644  1.2698 0.0990 0.0035  1.9843 0.6356 0.1434
## 9    6.9569     0.5491 0.8307  1.5619 0.1044 0.0036  2.4947 0.5981 0.1335
## 10   5.0939     0.4938 0.5605  1.9793 0.1185 0.0038  2.3764 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   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   2   1   1   1   1   1   1   1   1   3   3   3   2   3   2   3   2   3   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   3   2   3   3   3   3   2   2   2   3   3   3   3   3   3   3   3   3   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   2   3   3   3   3   2   3   2   2   3   2   2   2   3   3   3   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 141 142 143 144 145 146 147 148 149 150
##   3   3   3   3   3   3   3   3   3   3
```

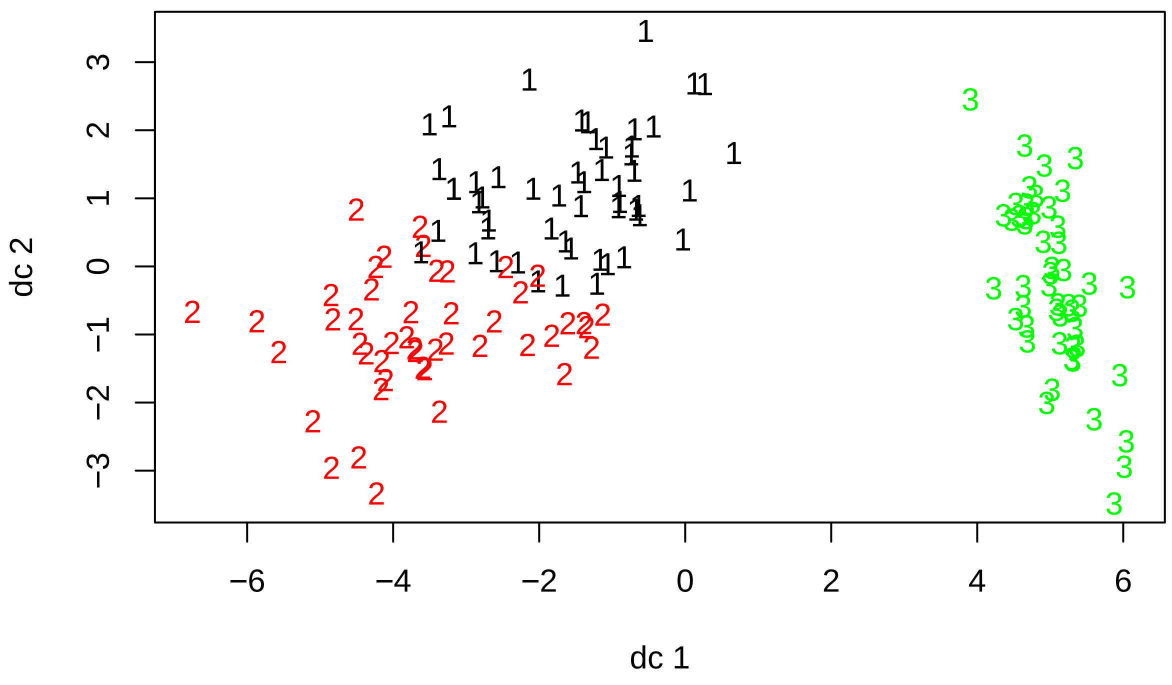
Algoritmos de Agrupamiento

```
km <- kmeans(datos[,1:4], 3, iter.max = 100)
datos$grupo <- km$cluster
```

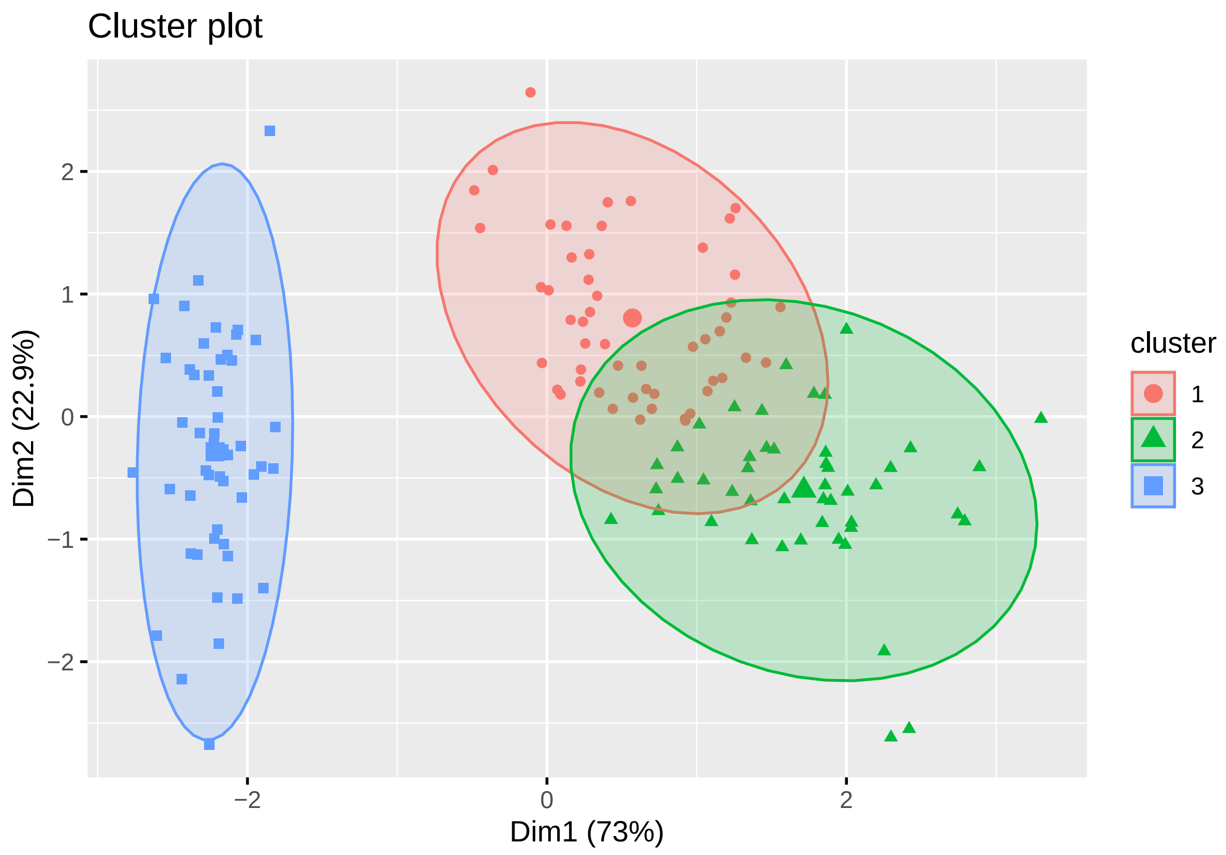
```
km
```

```
## K-means clustering with 3 clusters of sizes 53, 47, 50
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -0.05005221 -0.88042696    0.3465767   0.2805873
## 2   1.13217737  0.08812645    0.9928284   1.0141287
## 3  -1.01119138  0.85041372   -1.3006301  -1.2507035
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   3   3   3   3   3   3   3   3   3   3   2   2   2   1   1   1   2   1   1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   1   1   1   1   2   1   1   1   1   2   1   1   1   1   2   2   2   1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   1   1   1   2   2   1   1   1   1   1   1   1   1   1   1   1   1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   1   2   2   2   2   1   2   2   2   2   2   2   1   1   2   2   2   2   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   2   1   2   1   2   2   1   2   2   2   2   2   2   1   1   2   2   2   1   2
## 141 142 143 144 145 146 147 148 149 150
##   2   2   1   2   2   2   1   2   2   1
##
## Within cluster sum of squares by cluster:
## [1] 44.08754 47.45019 47.35062
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
plotcluster(datos[,1:4], km$cluster)
```



```r
fviz_cluster(km, data = datos[,1:4], geom = "point", ellipse.type = "norm")
```


Cluster plot
Dim2 (22.9%)
Dim1 (73%)
2
1
0
−1
−2
−2
0
2
cluster
1
2
3