Práctica en R del tema Clustering

Laura Rodríguez Navas

10/07/2020

Parte introductoria de la práctica

Para la correcta ejecución de esta práctica, el estudiante debe verificar que los paquetes **mlbench** y **caret** están correctamente instalados en la plataforma R. Para instalar estos paquetes en R, solo debe ejecutar:

- **install.packages("mlbench")**
- **install.packages("caret")**

Además, para esta práctica se ha decidido que se necesitan los siguientes paquetes de R:

- **install.packages("lattice")**
- **install.packages("ggplot2")**
- **install.packages("factoextra")**
- **install.packages("ggdendro")**

Luego de haber instalado los paquetes en R, pasamos a cargar en memoria las librerías que usaremos.

```
library(mlbench)
library(lattice)
library(ggplot2)
library(caret)
library(factoextra)
library(ggdendro)
```

El paquete **mlbench** se utiliza para cargar los datasets con los cuales trabajaremos en esta práctica. El paquete **caret** engloba más de 200 modelos de clasificación así como funciones útiles para el preprocesado de datos. A continuación cargamos el dataset BreastCancer de la librería **mlbench**. El objetivo de este dataset es describir si los pacientes tienen un cáncer benigno o maligno.

```
data(BreastCancer)
```

El dataset ya se encuentra disponible en el entorno, ahora solo podrás referenciarlo con la variable BreastCancer.

Ejercicios

A continuación debe resolver los siguientes ejercicios, escribiendo el código en cada sección correspondiente después del comentario *#INSERTAR CÓDIGO AQUÍ.*

Preprocesamiento de datos

Describa brevemente el dataset BreastCancer. Puede apoyarse en las funciones str(..) y summary (..). Como mínimo se espera que aporte la siguiente información:

- Número de ejemplos (observaciones).
- Número de variables.
- Tipo de variables.
- Distribuciones de datos por cada variable.

- Número de valores perdidos.

```
#INSERTAR CÓDIGO AQUÍ
str(BreastCancer, width = 85, strict.width = "cut")
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ Id             : chr  "1000025" "1002945" "1015425" "1016277" ...
##  $ Cl.thickness   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4..
##  $ Cell.size      : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 ..
##  $ Cell.shape     : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 ..
##  $ Marg.adhesion  : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1..
##  $ Epith.c.size   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2..
##  $ Bare.nuclei    : Factor w/ 10 levels "1","2","3","4",..: 1 10 2 4 1 10 10 1 1 1 ..
##  $ Bl.cromatin    : Factor w/ 10 levels "1","2","3","4",..: 3 3 3 3 3 9 3 3 1 2 ...
##  $ Normal.nucleoli: Factor w/ 10 levels "1","2","3","4",..: 1 2 1 7 1 7 1 1 1 1 ...
##  $ Mitoses        : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 5 1 ...
##  $ Class          : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

- Número de ejemplos (observaciones) = 699.
- Número de variables = 11.
- Tipo de variables. Consideramos el dataset BreastCancer definido sobre 10 variables descriptoras (9 discretas, 5 de ellas ordenadas; y una variable de tipo carácter) y una variable clase binaria {benign, malignant}.
- Distribuciones de datos por cada variable:

```
#INSERTAR CÓDIGO AQUÍ
summary(BreastCancer)
```

```
##       Id             Cl.thickness   Cell.size     Cell.shape  Marg.adhesion
##  Length:699         1      :145   1      :384   1      :353   1      :407
##  Class :character   5      :130   10     : 67   2      : 59   2      : 58
##  Mode  :character   3      :108   3      : 52   10     : 58   3      : 58
##                     4      : 80   2      : 45   3      : 56   10     : 55
##                     10     : 69   4      : 40   4      : 44   4      : 33
##                     2      : 50   5      : 30   5      : 34   8      : 25
##                     (Other):117   (Other): 81   (Other): 95   (Other): 63
##   Epith.c.size  Bare.nuclei   Bl.cromatin  Normal.nucleoli    Mitoses
##  2      :386   1      :402   2      :166   1      :443     1      :579
##  3      : 72   10     :132   3      :165   10     : 61     2      : 35
##  4      : 48   2      : 30   1      :152   3      : 44     3      : 33
##  1      : 47   5      : 30   7      : 73   2      : 36     10     : 14
##  6      : 41   3      : 28   4      : 40   8      : 24     4      : 12
##  5      : 39   (Other): 61   5      : 34   6      : 22     7      :  9
##  (Other): 66   NA's   : 16   (Other): 69   (Other): 69     (Other): 17
##        Class
##  benign   :458
##  malignant:241
##
##
##
##
##
```

- Número de valores perdidos.

```
#INSERTAR CÓDIGO AQUÍ
sum(is.na(BreastCancer))
```

```
## [1] 16
```

- En caso de ser necesario, elimine los valores perdidos (R los representa como NA).

```
#INSERTAR CÓDIGO AQUÍ
BreastCancer <- na.omit(BreastCancer)
any(is.na(BreastCancer))
```

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

Nota: en esta práctica, solo con la eliminación de registros es suficiente. Sin embargo, tenga en cuenta que la eliminación de registros que presentan valores perdidos no es una alternativa efectiva en la mayoría de los casos.

- Ahora, transforma todas las variables descriptoras a tipo numérico. Esto es necesario porque muchos métodos de clustering trabajan solamente con variables numéricas.

```
#INSERTAR CÓDIGO AQUÍ
for (i in 1:(ncol(BreastCancer) - 1))
  BreastCancer[, i] <- as.numeric(as.character(BreastCancer[, i]))

str(BreastCancer, width = 85, strict.width = "cut")
```

```
## 'data.frame':    683 obs. of  11 variables:
##  $ Id             : num  1000025 1002945 1015425 1016277 1017023 ...
##  $ Cl.thickness   : num  5 5 3 6 4 8 1 2 2 4 ...
##  $ Cell.size      : num  1 4 1 8 1 10 1 1 1 2 ...
##  $ Cell.shape     : num  1 4 1 8 1 10 1 2 1 1 ...
##  $ Marg.adhesion  : num  1 5 1 1 3 8 1 1 1 1 ...
##  $ Epith.c.size   : num  2 7 2 3 2 7 2 2 2 2 ...
##  $ Bare.nuclei    : num  1 10 2 4 1 10 10 1 1 1 ...
##  $ Bl.cromatin    : num  3 3 3 3 3 9 3 3 1 2 ...
##  $ Normal.nucleoli: num  1 2 1 7 1 7 1 1 1 1 ...
##  $ Mitoses        : num  1 1 1 1 1 1 1 1 5 1 ...
##  $ Class          : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:16] 24 41 140 146 159 165 236 250 276 ..
##   ..- attr(*, "names")= chr [1:16] "24" "41" "140" "146" ...
```

- La variable "ID" representa un identificador que es único para cada paciente y no aporta ninguna información para el aprendizaje automático, por lo que podemos eliminarlo con seguridad.

```
#INSERTAR CÓDIGO AQUÍ
BreastCancer$Id <- NULL
head(BreastCancer, 3)
```

```
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 1            5         1          1             1           2           1
## 2            5         4          4             5           7          10
## 3            3         1          1             1           2           2
##   Bl.cromatin Normal.nucleoli Mitoses  Class
## 1           3               1       1 benign
## 2           3               2       1 benign
## 3           3               1       1 benign
```

- Debido a que los algoritmos de clustering se basan en el cómputo de valores de distancias entre los ejemplos, es ventajoso tener todas las variables en la misma escala para el cálculo de las distancias entre ejemplos. Convierta todos los atributos descriptores a una misma escala.

Nota: se recomienda utilizar la función preProcess(.., method = "scale") de la librería caret.

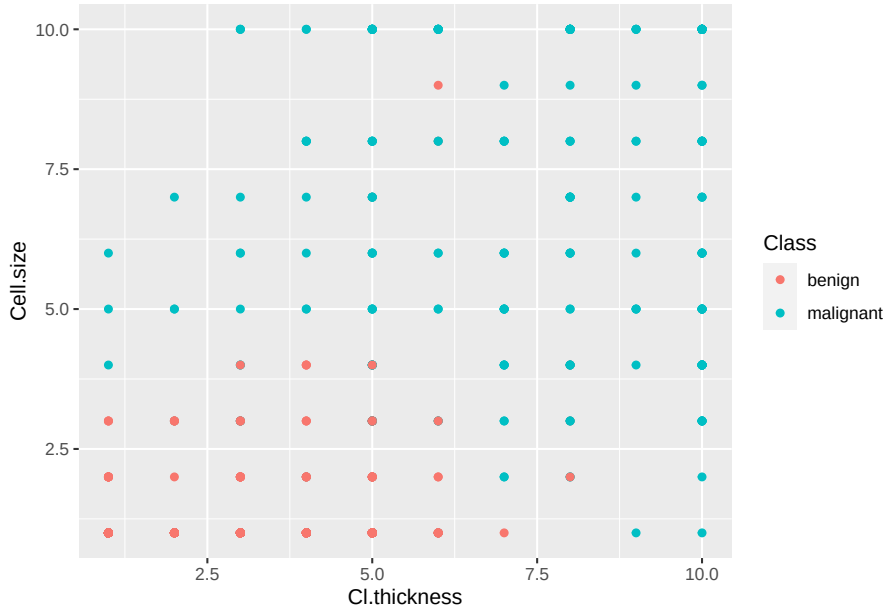
```
# INSERTAR CÓDIGO AQUÍ
BreastCancer.scale <- preProcess(BreastCancer[, 1:9], method=c("scale"))
BreastCancer.features <- predict(BreastCancer.scale, BreastCancer[, 1:9])
str(BreastCancer.features, width = 85, strict.width = "cut")
```

```
## 'data.frame':    683 obs. of  9 variables:
##  $ Cl.thickness   : num  1.77 1.77 1.06 2.13 1.42 ...
##  $ Cell.size      : num  0.326 1.305 0.326 2.61 0.326 ...
##  $ Cell.shape     : num  0.335 1.338 0.335 2.677 0.335 ...
##  $ Marg.adhesion  : num  0.349 1.745 0.349 0.349 1.047 ...
##  $ Epith.c.size   : num  0.9 3.15 0.9 1.35 0.9 ...
##  $ Bare.nuclei    : num  0.274 2.744 0.549 1.098 0.274 ...
##  $ Bl.cromatin    : num  1.22 1.22 1.22 1.22 1.22 ...
##  $ Normal.nucleoli: num  0.328 0.655 0.328 2.293 0.328 ...
##  $ Mitoses        : num  0.577 0.577 0.577 0.577 0.577 ...
```

Clustering con kMeans

Una vez que los datos han sido preprocesados, la librería ggplot2, la cual es automaticamente importada por la librería caret, nos permite visualizar los datos para comprobar cómo las diferentes variables afectan al tipo de cáncer.

```
# En este caso solo estamos mostrando el gráfico para las variables Cl.thickness y Cell.size.
ggplot(BreastCancer, aes(Cl.thickness, Cell.size, color = Class)) + geom_point()
```



Se puede apreciar que considerando estos pares de variables se observan dos grupos claramente definidos.

- Pasemos ahora a ejecutar algoritmos de clustering sobre nuestros datos. Mediante un agrupamiento usando kmeans(..), encuentre el par de variables descriptoras a partir del cual se logra un mejor agrupamiento. Utilice 20 asignaciones aleatorias para inicializar los centroides y además fije el número de cluster igual al número de clases existentes en el dataset.

Nota: recuerde que la variable Class no se debe tener en cuenta a la hora de realizar el clustering, ya que estamos haciendo un aprendizaje no supervisado. El atributo Class es útil para usar una métrica de evaluación externa y así comprobar la calidad del agrupamiento.

```
#INSERTAR CÓDIGO AQUÍ
set.seed(101)
km_clusters <- kmeans(BreastCancer.features, centers = 2, nstart = 20)
km_clusters
```

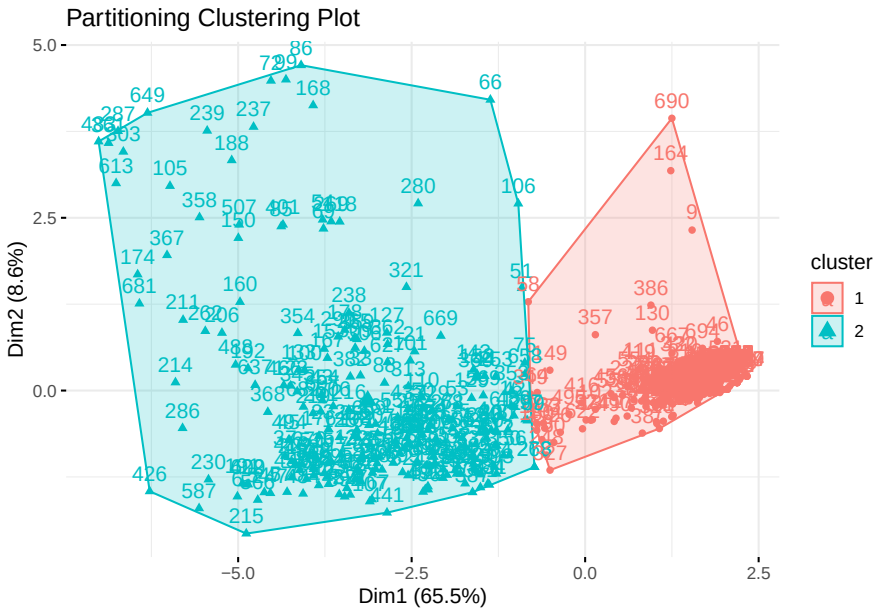
```
## K-means clustering with 2 clusters of sizes 453, 230
##
## Cluster means:
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 1     1.077629 0.4191542  0.4712566     0.4700817    0.9423492   0.3840871
## 2     2.554044 2.2270031  2.2665986     2.0080464    2.4642601   2.1322365
##   Bl.cromatin Normal.nucleoli  Mitoses
## 1   0.8578798       0.4092973 0.617912
## 2   2.4865546       1.9854346 1.530682
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   2   1   2   1   2   1   1   1   1   1   1   1   1   2   2   1   1   2   1
##  21  22  23  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  42
##   2   2   1   1   1   1   1   1   1   1   1   2   1   1   1   2   1   2   2   2
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
##   2   2   2   1   2   1   1   2   2   1   2   2   2   2   2   1   1   1   2   1
##  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   2   1   1   2   1   2   2   1   1   2   1   2   2   1   1   1   1   1   1   1
##  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
##   1   1   2   2   2   2   1   1   1   1   1   1   1   1   1   1   2   2   2   1
## 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
##   1   1   2   2   2   2   1   2   1   2   2   2   1   1   1   2   1   1   1   1
## 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143
##   2   2   2   1   2   1   2   1   1   1   2   1   1   1   1   1   1   1   1   2
## 144 145 147 148 149 150 151 152 153 154 155 156 157 158 160 161 162 163 164 166
##   1   1   1   1   1   2   1   2   2   1   1   2   1   1   2   2   1   1   1   1
## 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186
##   2   2   1   1   1   1   1   2   2   2   1   2   1   1   1   1   1   2   2   1
## 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
##   2   2   2   1   2   2   1   1   1   1   2   1   1   1   2   2   1   1   1   2
## 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226
##   2   1   1   1   2   2   1   2   2   2   1   1   2   1   1   2   1   2   2   1
## 227 228 229 230 231 232 233 234 235 237 238 239 240 241 242 243 244 245 246 247
##   2   2   1   2   2   2   1   2   1   2   2   2   2   1   1   1   1   1   1   2
## 248 249 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268
##   1   1   1   2   2   2   2   2   1   1   1   2   2   2   2   2   2   1   2   2
## 269 270 271 272 273 274 275 277 278 279 280 281 282 283 284 285 286 287 288 289
##   2   1   2   1   2   1   1   1   1   1   2   1   1   2   2   2   2   2   1   2
## 290 291 292 294 296 297 299 300 301 302 303 304 305 306 307 308 309 310 311 312
##   2   1   1   2   2   2   1   2   2   1   2   1   2   2   1   1   2   1   1   1
## 313 314 315 317 318 319 320 321 323 324 325 326 327 328 329 330 331 332 333 334
##   2   1   1   2   2   1   2   2   1   2   1   1   1   1   2   2   2   1   1   2
## 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354
##   2   1   2   1   1   2   2   1   1   1   2   1   1   1   1   2   1   1   2   2
## 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
```

```
##   1   1   1   2   2   2   2   2   1   1   1   1   2   2   1   1   1   1   1   1
## 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394
##   1   1   1   1   1   1   1   2   1   1   1   1   2   1   1   1   1   2   1   1
## 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 413 414 415
##   1   1   1   1   1   1   2   1   1   1   1   1   1   1   1   1   1   2   1   2
## 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
##   1   2   1   1   1   1   2   1   1   1   2   1   2   1   1   1   1   1   1   2
## 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455
##   2   2   1   1   1   2   1   1   1   1   1   1   1   1   2   1   1   1   2   1
## 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475
##   1   2   2   1   1   1   1   1   1   1   2   2   2   1   1   1   1   1   1   1
## 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
##   1   1   1   1   2   1   1   2   2   1   1   1   2   2   1   1   2   1   2   1
## 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   2
## 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
##   2   1   1   1   2   1   1   2   2   1   1   1   1   1   1   2   1   1   1   1
## 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   2   1   1   1   1   1
## 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
##   1   1   1   1   1   1   1   1   1   1   2   1   1   2   2   2   2   1   1   2
## 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595
##   1   1   1   1   1   1   2   2   1   1   1   2   1   2   1   2   2   2   1   2
## 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
##   1   1   1   1   1   1   1   1   2   2   2   1   1   2   1   2   2   2   1   1
## 616 617 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   2   1   1
## 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
##   2   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1
## 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676
##   1   2   2   1   1   1   1   1   1   1   1   1   1   2   2   2   1   1   1   1   1
## 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696
##   1   1   1   1   2   2   1   1   1   1   1   1   1   1   1   2   1   1   1   1
## 697 698 699
##   2   2   2
##
## Within cluster sum of squares by cluster:
## [1]  573.4613 2150.6939
##  (between_SS / total_SS =  55.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

- A partir de los resultados obtenidos, ¿Por qué se puede considerar un buen agrupamiento? Justifica de forma clara y concisa la elección del par de atributos. Puedes ayudarte con la funciones table(..) y ggplot(..) para justificar tu respuesta.

RESPUESTA:

```
#INSERTAR CÓDIGO AQUÍ
fviz_cluster(km_clusters, data = BreastCancer.features,
             ggtheme = theme_minimal(),
             main = "Partitioning Clustering Plot")
```



- Mediante la función preprocess(..) de la librería caret, analice si se obtienen mejores resultados de agrupamiento con el par de atributos seleccionados anteriormente, pero en este caso haciendo inicialmente las siguientes combinaciones de transformaciones a los datos:
  - c("center", "scale")
  - c("center", "scale", "YeoJohnson")
  - c("range")
  - c("range", "YeoJohnson")

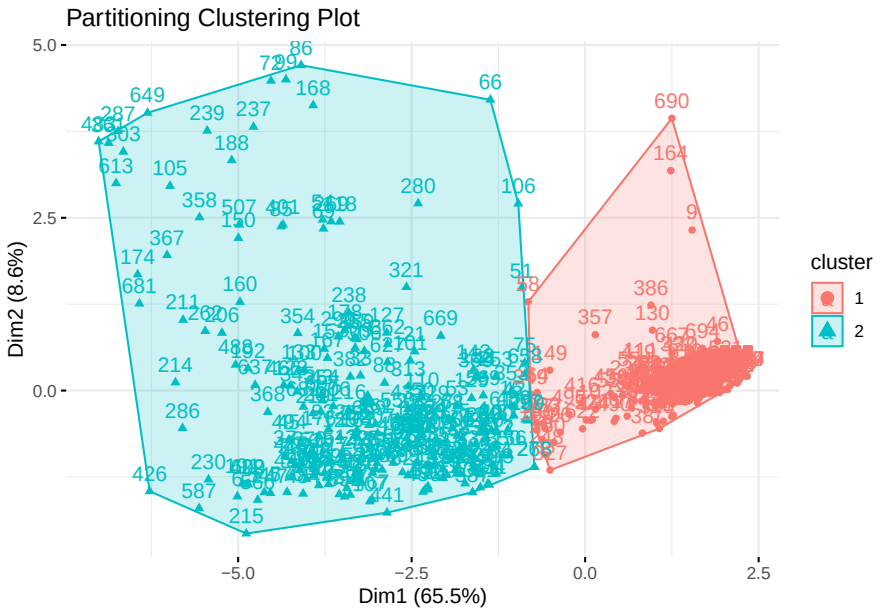
Ayúdate de los resultados obtenidos con table y ggplot.

- c("center", "scale")

```
#INSERTAR CÓDIGO AQUÍ
BreastCancer.scale <- preProcess(BreastCancer[, 1:9], method=c("center", "scale"))
BreastCancer.features <- predict(BreastCancer.scale, BreastCancer[, 1:9])
set.seed(101)
km_clusters <- kmeans(BreastCancer.features, centers = 2, nstart = 20)
km_clusters$centers
```

```
##   Cl.thickness  Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 1   -0.4971820 -0.6087925 -0.6045807    -0.5179091    -0.512503  -0.5886887
## 2    0.9792324  1.1990564  1.1907612     1.0200557     1.009408   1.1594607
##   Bl.cromatin Normal.nucleoli    Mitoses
## 1  -0.5484556      -0.5307637 -0.3073751
## 2   1.0802192       1.0453737  0.6053953
```

```
fviz_cluster(km_clusters, data = BreastCancer.features,
             ggtheme = theme_minimal(),
             main = "Partitioning Clustering Plot")
```

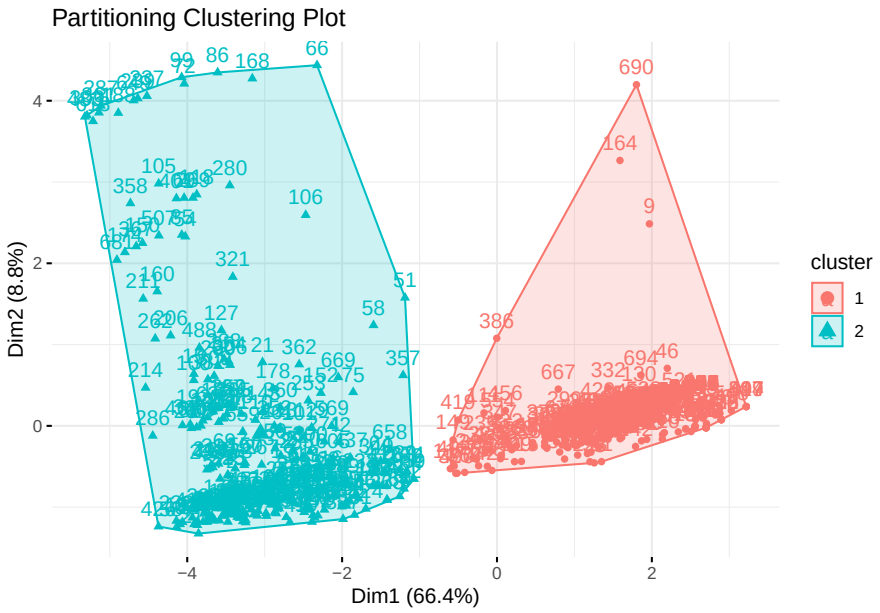


- c("center", "scale", "YeoJohnson")

```
#INSERTAR CÓDIGO AQUÍ
BreastCancer.scale <- preProcess(BreastCancer[, 1:9], method=c("center", "scale", "YeoJohnson"))
BreastCancer.features <- predict(BreastCancer.scale, BreastCancer[, 1:9])
set.seed(101)
km_clusters <- kmeans(BreastCancer.features, centers = 2, nstart = 20)
km_clusters$centers
```

```
##   Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 1   -0.5102508 -0.681466 -0.6688897    -0.5762785   -0.5804375  -0.6403634
## 2    0.8949964  1.195313  1.1732541     1.0108111    1.0181061   1.1232180
##   Bl.cromatin Normal.nucleoli    Mitoses
## 1  -0.5695150      -0.6000646 -0.3123219
## 2   0.9989476       1.0525326  0.5478227
```

```
fviz_cluster(km_clusters, data = BreastCancer.features,
             ggtheme = theme_minimal(),
             main = "Partitioning Clustering Plot")
```

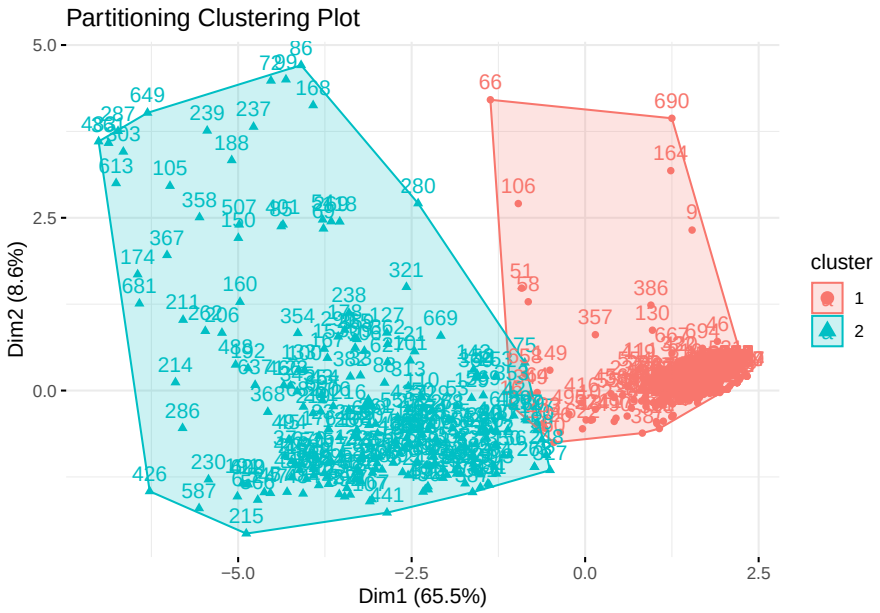


- c("range")

```
#INSERTAR CÓDIGO AQUÍ
BreastCancer.scale <- preProcess(BreastCancer[, 1:9], method=c("range"))
BreastCancer.features <- predict(BreastCancer.scale, BreastCancer[, 1:9])
set.seed(101)
km_clusters <- kmeans(BreastCancer.features, centers = 2, nstart = 20)
km_clusters$centers
```

```
##   Cl.thickness  Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 1    0.2283542 0.03311258 0.04758401    0.03924454    0.1216581  0.03532009
## 2    0.6859903 0.64444444 0.63719807    0.52657005    0.4975845  0.77004831
##   Bl.cromatin Normal.nucleoli   Mitoses
## 1   0.1214128      0.02894285 0.0125092
## 2   0.5676329      0.55990338 0.1743961
```

```
fviz_cluster(km_clusters, data = BreastCancer.features,
             ggtheme = theme_minimal(),
             main = "Partitioning Clustering Plot")
```

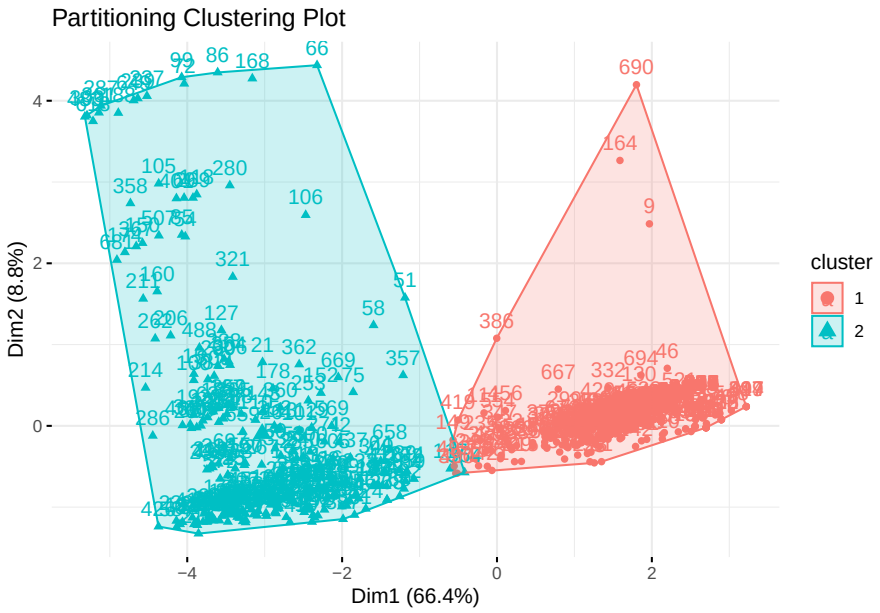


- c("range", "YeoJohnson")

```
#INSERTAR CÓDIGO AQUÍ
BreastCancer.scale <- preProcess(BreastCancer[, 1:9], method=c("range", "YeoJohnson"))
BreastCancer.features <- predict(BreastCancer.scale, BreastCancer[, 1:9])
set.seed(101)
km_clusters <- kmeans(BreastCancer.features, centers = 2, nstart = 20)
km_clusters$centers
```

```
##   Cl.thickness  Cell.size Cell.shape Marg.adhesion Epith.c.size Bare.nuclei
## 1    0.3095026 0.07331498 0.09953216    0.09924841    0.3955499  0.07012383
## 2    0.7631609 0.85523823 0.84947930    0.74680314    0.7835328  0.84905728
##   Bl.cromatin Normal.nucleoli     Mitoses
## 1   0.2634012      0.05855806 0.006928406
## 2   0.7482095      0.75675052 0.171111111
```

```
fviz_cluster(km_clusters, data = BreastCancer.features,
             ggtheme = theme_minimal(),
             main = "Partitioning Clustering Plot")
```



- A partir del par de variables seleccionadas, cree un agrupamiento conformado por tres variables; las dos variables previamente seleccionadas y otra variable seleccionada aleatoriamente entre las variables descriptoras restantes. Este proceso debes repetirlo cinco veces de tal manera que en cada ejecución la variable añadida no necesariamente sea la misma.

```
#INSERTAR CÓDIGO AQUÍ
func <- function (var) {
  new_data <- c("Cell.size", "Cell.shape", var)
  set.seed(101)
  km_clusters <- kmeans(BreastCancer.features[new_data], centers = 2, nstart = 20)
  km_clusters
}

vars <- c("Cl.thickness", "Marg.adhesion", "Epith.c.size", "Bare.nuclei", "Bl.cromatin")

func(vars[1])
```

```
## K-means clustering with 2 clusters of sizes 417, 266
##
## Cluster means:
##    Cell.size Cell.shape Cl.thickness
## 1 0.05035852  0.0825586    0.3021321
## 2 0.84419337  0.8309786    0.7474276
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   2   1   2   1   2   1   1   1   1   1   1   2   1   2   2   1   1   2   1
##  21  22  23  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  42
##   2   2   1   1   2   1   1   1   1   1   1   2   1   1   1   2   1   2   2   2
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
##   2   2   2   1   2   1   1   2   2   2   2   2   2   2   2   2   2   2   2   1
##  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   2   2   1   2   1   2   2   1   1   2   2   2   2   1   1   1   1   1   1   1
##  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
##   1   1   2   2   2   2   1   1   1   1   1   1   1   1   1   1   2   2   2   2
```

```
## 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
##   1   2   2   2   2   2   1   2   1   2   2   2   2   1   2   2   1   1   1   1
## 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143
##   2   2   2   1   2   1   2   1   1   1   2   1   1   1   1   1   1   1   1   2
## 144 145 147 148 149 150 151 152 153 154 155 156 157 158 160 161 162 163 164 166
##   1   1   2   1   1   2   1   2   2   1   1   2   1   1   2   2   1   1   1   1
## 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186
##   2   2   1   1   1   1   1   2   2   2   1   2   1   2   1   1   1   2   2   1
## 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
##   2   2   2   1   2   2   1   1   1   1   2   1   1   1   2   2   1   1   1   2
## 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226
##   2   1   1   1   2   2   1   2   2   2   1   1   2   1   1   2   1   2   2   1
## 227 228 229 230 231 232 233 234 235 237 238 239 240 241 242 243 244 245 246 247
##   2   2   1   2   2   2   2   2   2   2   2   2   2   1   1   1   1   1   1   2
## 248 249 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268
##   2   1   1   2   2   2   2   2   1   1   1   2   2   2   2   2   2   1   2   2
## 269 270 271 272 273 274 275 277 278 279 280 281 282 283 284 285 286 287 288 289
##   2   1   2   1   2   2   1   1   1   1   2   1   1   2   2   2   2   2   1   1
## 290 291 292 294 296 297 299 300 301 302 303 304 305 306 307 308 309 310 311 312
##   2   1   1   2   2   2   1   1   2   1   2   1   2   2   1   1   2   1   1   1
## 313 314 315 317 318 319 320 321 323 324 325 326 327 328 329 330 331 332 333 334
##   2   1   1   2   2   1   2   2   1   2   1   1   1   1   2   2   2   1   1   2
## 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354
##   2   1   2   1   1   2   2   1   1   1   2   1   1   1   2   2   1   1   2   2
## 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
##   1   1   2   2   2   2   2   2   1   2   1   1   2   2   1   1   2   1   1   1
## 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394
##   1   1   1   1   1   2   1   2   1   1   1   2   2   2   1   1   1   2   1   1
## 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 413 414 415
##   1   1   1   1   1   1   2   1   2   1   1   1   1   1   1   1   1   2   1   2
## 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
##   2   2   1   1   1   1   2   2   1   1   2   2   2   1   1   1   1   1   1   2
## 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455
##   2   2   1   1   1   2   1   1   1   1   1   1   1   1   2   1   1   1   2   1
## 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475
##   2   2   2   1   1   1   1   1   1   1   2   2   2   1   1   1   1   1   1   1
## 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
##   1   1   1   1   2   1   2   2   2   1   1   1   2   2   2   1   2   1   2   1
## 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   2
## 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
##   2   1   1   1   2   1   1   2   2   1   1   1   1   1   1   2   1   1   1   1
## 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   2   1   1   1   2   1
## 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
##   1   1   1   1   1   1   1   1   1   1   2   1   1   2   2   2   2   1   1   2
## 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595
##   1   1   1   1   1   1   2   2   1   1   1   2   1   2   1   2   2   2   1   2
## 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
##   1   1   1   1   2   1   1   1   2   2   2   1   1   2   1   2   2   2   1   1
## 616 617 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636
##   1   1   1   1   1   2   1   1   1   1   2   1   1   1   2   1   1   2   1   1
## 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
##   2   2   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1
```

```
## 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676
##   1   2   2   1   1   1   1   1   1   1   1   1   2   2   2   1   1   2   1   1
## 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696
##   1   1   1   1   2   2   1   1   1   1   1   1   1   1   1   2   1   1   1   1
## 697 698 699
##   2   2   2
##
## Within cluster sum of squares by cluster:
## [1] 48.89877 26.95696
##  (between_SS / total_SS =  74.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
func(vars[2])
```

```
## K-means clustering with 2 clusters of sizes 250, 433
##
## Cluster means:
##    Cell.size Cell.shape Marg.adhesion
## 1 0.86067176  0.8467986    0.76567771
## 2 0.07017783  0.1010799    0.08835086
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   1   2   1   2   1   2   2   2   2   2   2   1   2   1   1   2   2   1   2
##  21  22  23  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  42
##   1   1   2   2   1   2   2   2   2   2   2   1   2   2   2   1   2   1   1   1
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
##   1   1   1   2   1   2   2   1   1   1   1   1   1   1   1   2   2   1   1   2
##  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   1   1   2   2   2   1   1   2   2   1   1   1   1   2   2   2   2   2   2   2
##  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
##   2   2   1   1   1   1   2   2   2   2   2   2   2   2   2   2   1   1   1   1
## 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
##   2   2   1   1   1   1   2   1   2   1   1   1   2   2   2   1   2   2   2   2
## 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143
##   1   1   1   2   1   2   1   2   2   2   1   2   2   2   2   2   2   2   2   1
## 144 145 147 148 149 150 151 152 153 154 155 156 157 158 160 161 162 163 164 166
##   2   2   1   2   2   1   2   2   1   2   2   1   2   2   1   1   2   2   2   2
## 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186
##   1   1   2   2   2   2   2   1   1   1   2   1   2   1   2   2   2   1   1   2
## 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
##   1   1   1   2   1   1   2   2   2   2   1   2   2   2   1   1   2   2   2   1
## 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226
##   1   2   2   2   1   1   2   1   1   1   2   2   1   2   2   1   2   1   1   2
## 227 228 229 230 231 232 233 234 235 237 238 239 240 241 242 243 244 245 246 247
##   1   1   2   1   1   1   1   1   2   1   1   1   1   2   2   2   2   2   2   1
## 248 249 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268
##   1   2   2   1   1   1   1   1   2   2   2   1   1   1   1   1   1   2   1   1
## 269 270 271 272 273 274 275 277 278 279 280 281 282 283 284 285 286 287 288 289
##   1   2   1   2   1   2   2   2   2   2   1   2   2   1   1   1   1   1   2   2
## 290 291 292 294 296 297 299 300 301 302 303 304 305 306 307 308 309 310 311 312
```

```
##   1   2   2   1   1   1   2   2   1   2   1   2   1   1   2   2   1   2   2   2
## 313 314 315 317 318 319 320 321 323 324 325 326 327 328 329 330 331 332 333 334
##   1   2   2   1   1   2   1   1   2   1   2   2   2   2   1   1   1   2   2   1
## 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354
##   1   2   1   2   2   1   2   2   2   2   1   2   2   2   1   1   2   2   1   1
## 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
##   2   2   2   1   1   1   1   1   2   1   2   2   1   1   2   2   2   2   2   2
## 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394
##   2   2   2   2   2   1   2   1   2   2   2   1   1   1   2   2   2   1   2   2
## 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 413 414 415
##   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   2   1   2   1
## 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
##   1   1   2   2   2   1   1   2   2   2   1   1   1   2   2   2   2   2   1   1
## 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455
##   1   1   2   2   2   1   1   2   2   2   2   2   2   2   1   2   2   2   1   2
## 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475
##   2   1   1   2   2   2   2   2   2   2   1   1   1   2   2   2   2   2   2   2
## 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
##   2   2   2   2   1   2   1   1   1   2   2   2   1   1   2   2   1   2   1   2
## 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515
##   2   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1
## 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
##   1   2   2   2   1   2   2   1   1   2   2   2   2   2   2   1   2   2   2   2
## 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   1   2   2   2   2   2
## 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
##   2   2   2   2   2   2   2   2   2   2   1   2   2   1   1   1   1   2   2   1
## 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595
##   2   2   2   2   2   2   1   1   2   2   2   1   2   1   2   1   1   1   2   1
## 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
##   2   2   2   2   2   2   2   2   1   1   1   2   2   1   2   1   1   1   2   2
## 616 617 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636
##   2   2   2   2   2   1   2   2   2   2   1   2   2   2   2   2   2   1   2   2
## 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
##   1   1   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2
## 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676
##   2   1   1   2   2   2   2   2   2   2   2   2   1   1   1   2   2   2   2   2
## 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696
##   2   2   2   2   1   1   2   2   2   2   2   2   2   2   2   1   2   2   2   2
## 697 698 699
##   1   1   1
##
## Within cluster sum of squares by cluster:
## [1] 31.69037 53.75504
##  (between_SS / total_SS =  75.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
func(vars[3])
```

```
## K-means clustering with 2 clusters of sizes 414, 269
##
```

```
## Cluster means:
##    Cell.size Cell.shape Epith.c.size
## 1 0.04710711 0.07956919    0.3915786
## 2 0.84034423 0.82723273    0.7622409
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   2   1   2   1   2   1   1   1   1   1   1   2   1   2   2   1   1   2   1
##  21  22  23  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  42
##   2   2   1   1   2   1   1   1   1   1   1   2   1   1   1   2   1   2   2   2
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
##   2   2   2   1   2   1   1   2   2   2   2   2   2   2   2   2   2   2   2   1
##  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   2   2   1   2   1   2   2   1   1   2   2   2   2   1   1   1   1   1   1   1
##  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
##   1   1   2   2   2   2   1   1   1   1   1   1   1   1   1   1   2   2   2   2
## 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
##   1   2   2   2   2   2   1   2   1   2   2   2   2   1   2   2   1   1   1   1
## 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143
##   2   2   2   1   2   1   2   1   1   1   2   1   1   1   1   1   1   1   1   2
## 144 145 147 148 149 150 151 152 153 154 155 156 157 158 160 161 162 163 164 166
##   1   1   2   1   1   2   1   2   2   1   1   2   1   1   2   2   1   1   1   1
## 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186
##   2   2   1   1   1   1   1   2   2   2   1   2   1   2   1   1   1   2   2   1
## 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
##   2   2   2   2   2   2   1   1   1   1   2   1   1   1   2   2   1   1   1   2
## 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226
##   2   1   1   1   2   2   1   2   2   2   1   1   2   1   1   2   1   2   2   1
## 227 228 229 230 231 232 233 234 235 237 238 239 240 241 242 243 244 245 246 247
##   2   2   1   2   2   2   2   2   2   2   2   2   2   1   1   1   1   1   1   2
## 248 249 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268
##   2   1   1   2   2   2   2   2   1   1   1   2   2   2   2   2   2   1   2   2
## 269 270 271 272 273 274 275 277 278 279 280 281 282 283 284 285 286 287 288 289
##   2   1   2   1   2   2   1   1   1   1   2   1   1   2   2   2   2   2   1   1
## 290 291 292 294 296 297 299 300 301 302 303 304 305 306 307 308 309 310 311 312
##   2   1   1   2   2   2   1   1   2   1   2   1   2   2   1   1   2   1   1   1
## 313 314 315 317 318 319 320 321 323 324 325 326 327 328 329 330 331 332 333 334
##   2   1   1   2   2   1   2   2   1   2   1   1   1   1   2   2   2   1   1   2
## 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354
##   2   1   2   1   1   2   2   1   1   1   2   1   1   1   2   2   1   1   2   2
## 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
##   1   1   2   2   2   2   2   2   2   2   1   1   2   2   1   1   2   1   1   1
## 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394
##   1   1   1   1   1   2   1   2   1   1   1   2   2   2   1   1   1   2   1   1
## 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 413 414 415
##   2   1   1   1   1   2   2   1   2   1   1   1   1   1   2   1   1   2   1   2
## 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
##   2   2   1   1   1   1   2   2   1   1   2   2   2   1   1   1   1   1   1   2
## 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455
##   2   2   1   1   1   2   1   1   1   1   1   1   1   1   2   1   1   1   2   1
## 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475
##   1   2   2   1   1   1   1   1   1   1   2   2   2   1   1   1   1   1   1   1
## 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
##   1   1   1   1   2   1   2   2   2   1   1   1   2   2   2   1   2   1   2   1
```

```
## 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1   2
## 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
##   2   1   1   1   2   1   1   2   2   1   1   1   1   1   1   2   1   1   1   1
## 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
##   1   1   1   1   1   1   1   1   1   1   1   2   1   1   2   1   1   1   2   1
## 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
##   1   1   1   1   1   1   1   1   1   1   2   1   1   2   2   2   2   1   1   2
## 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595
##   1   1   1   1   1   1   2   2   1   1   1   2   1   2   1   2   2   2   1   2
## 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
##   1   1   1   1   1   1   1   1   2   2   2   1   1   2   1   2   2   2   1   1
## 616 617 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636
##   1   1   1   1   1   2   1   1   1   1   2   1   1   1   2   1   1   2   1   1
## 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
##   2   2   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1   1
## 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676
##   1   2   2   1   1   1   1   1   1   1   1   1   2   2   2   1   1   2   1   1
## 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696
##   1   1   1   1   2   2   1   1   1   1   1   1   1   1   1   2   1   1   1   1
## 697 698 699
##   2   2   2
##
## Within cluster sum of squares by cluster:
## [1] 32.66440 22.55752
##  (between_SS / total_SS =  79.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
func(vars[4])
```

```
## K-means clustering with 2 clusters of sizes 259, 424
##
## Cluster means:
##    Cell.size Cell.shape Bare.nuclei
## 1 0.84490785 0.83569346  0.84291850
## 2 0.06302785 0.09203454  0.05733974
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   1   2   1   2   1   2   2   2   2   2   2   1   2   1   1   2   2   1   2
##  21  22  23  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  42
##   1   1   2   2   1   2   2   2   2   2   2   1   2   2   2   1   2   1   1   1
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
##   1   1   1   2   1   2   2   1   1   1   1   1   1   1   1   2   1   1   1   2
##  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   1   1   2   1   2   1   1   2   2   1   2   1   1   2   2   2   2   2   2   2
##  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
##   2   2   1   1   1   1   2   2   2   2   2   2   2   2   2   2   1   1   1   1
## 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
##   2   1   1   1   1   1   2   1   2   1   1   1   1   2   1   1   2   2   2   2
## 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143
```

```
##   1   1   1   2   1   2   1   2   2   2   1   2   2   2   2   2   2   2   2   1
## 144 145 147 148 149 150 151 152 153 154 155 156 157 158 160 161 162 163 164 166
##   2   2   1   2   2   1   2   1   1   2   2   1   2   2   1   1   2   2   2   2
## 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186
##   1   1   2   2   2   2   2   1   1   1   2   1   2   1   2   2   2   1   1   2
## 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
##   1   1   1   2   1   1   2   2   2   2   1   2   2   2   1   1   2   2   2   1
## 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226
##   1   2   2   2   1   1   2   1   1   1   2   2   1   2   2   1   2   1   1   2
## 227 228 229 230 231 232 233 234 235 237 238 239 240 241 242 243 244 245 246 247
##   1   1   2   1   1   1   1   1   2   1   1   1   1   2   2   2   2   2   2   1
## 248 249 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268
##   1   2   2   1   1   1   1   1   2   2   2   1   1   1   1   1   1   2   1   1
## 269 270 271 272 273 274 275 277 278 279 280 281 282 283 284 285 286 287 288 289
##   1   2   1   2   1   1   2   2   2   2   1   2   2   1   1   1   1   1   2   1
## 290 291 292 294 296 297 299 300 301 302 303 304 305 306 307 308 309 310 311 312
##   1   2   2   1   1   1   2   1   1   2   1   2   1   1   2   2   1   2   2   2
## 313 314 315 317 318 319 320 321 323 324 325 326 327 328 329 330 331 332 333 334
##   1   2   2   1   1   2   1   1   2   1   2   2   2   2   1   1   1   2   2   1
## 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354
##   1   2   1   2   2   1   1   2   2   2   1   2   2   2   1   1   2   2   1   1
## 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
##   2   2   1   1   1   1   1   1   1   1   2   2   1   1   2   2   2   2   2   2
## 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394
##   2   2   2   2   2   2   2   1   2   2   2   2   1   2   2   2   2   1   2   2
## 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 413 414 415
##   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   2   1   2   1
## 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
##   1   1   2   2   2   1   1   2   2   2   1   1   1   2   2   2   2   2   2   1
## 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455
##   1   1   2   2   2   1   1   2   2   2   2   2   2   2   1   2   2   2   1   2
## 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475
##   1   1   1   2   2   2   2   2   2   2   1   1   1   2   2   2   2   2   2   2
## 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
##   2   2   2   2   1   2   2   1   1   2   2   2   1   1   1   2   1   2   1   2
## 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1
## 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
##   1   2   2   2   1   2   2   1   1   2   2   2   2   2   2   1   2   2   2   2
## 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   1   2   2   2   1   2
## 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
##   2   2   2   2   2   2   2   2   2   2   1   2   2   1   1   1   1   2   2   1
## 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595
##   2   2   2   2   2   2   1   1   2   2   2   1   2   1   2   1   1   1   2   1
## 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
##   2   2   2   2   2   2   2   2   1   1   1   2   2   1   2   1   1   1   2   2
## 616 617 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636
##   2   2   2   2   2   1   2   2   2   2   1   2   2   2   2   2   2   1   2   2
## 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
##   1   1   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2
## 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676
##   2   1   1   2   2   2   2   2   2   2   2   2   1   1   1   2   2   2   2   2
## 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696
```

```
##   2   2   2   2   1   1   2   2   2   2   2   2   2   2   2   1   2   2   2   2
## 697 698 699
##   1   1   1
##
## Within cluster sum of squares by cluster:
## [1] 32.91923 44.40170
##  (between_SS / total_SS =  78.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
func(vars[5])
```

```
## K-means clustering with 2 clusters of sizes 265, 418
##
## Cluster means:
##    Cell.size Cell.shape Bl.cromatin
## 1 0.84420352 0.83213789   0.7298048
## 2 0.05225122 0.08361413   0.2576718
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   1   2   1   2   1   2   2   2   2   2   2   1   2   1   1   2   2   1   2
##  21  22  23  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  42
##   1   1   2   2   1   2   2   2   2   2   2   1   2   2   2   1   2   1   1   1
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
##   1   1   1   2   1   2   2   1   1   1   1   1   1   1   1   1   1   1   1   2
##  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   1   1   2   1   2   1   1   2   2   1   1   1   1   2   2   2   2   2   1   2
##  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
##   2   2   1   1   1   1   2   2   2   2   2   2   2   2   2   2   1   1   1   1
## 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
##   2   1   1   1   1   1   2   1   2   1   1   1   1   2   1   1   2   2   2   2
## 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143
##   1   1   1   2   1   2   1   2   2   2   1   2   2   2   2   2   2   2   2   1
## 144 145 147 148 149 150 151 152 153 154 155 156 157 158 160 161 162 163 164 166
##   2   2   1   2   2   1   2   1   1   2   2   1   2   2   1   1   2   2   2   2
## 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186
##   1   1   2   2   2   2   2   1   1   1   2   1   2   1   2   2   2   1   1   2
## 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
##   1   1   1   1   1   1   2   2   2   2   1   2   2   2   1   1   2   2   2   1
## 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226
##   1   2   2   2   1   1   2   1   1   1   2   2   1   2   2   1   2   1   1   2
## 227 228 229 230 231 232 233 234 235 237 238 239 240 241 242 243 244 245 246 247
##   1   1   2   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   1
## 248 249 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268
##   1   2   2   1   1   1   1   1   2   2   2   1   1   1   1   1   1   2   1   1
## 269 270 271 272 273 274 275 277 278 279 280 281 282 283 284 285 286 287 288 289
##   1   2   1   2   1   1   2   2   2   2   1   2   2   1   1   1   1   1   2   2
## 290 291 292 294 296 297 299 300 301 302 303 304 305 306 307 308 309 310 311 312
##   1   2   2   1   1   1   2   2   1   2   1   2   1   1   2   2   1   2   2   2
## 313 314 315 317 318 319 320 321 323 324 325 326 327 328 329 330 331 332 333 334
##   1   2   2   1   1   2   1   1   2   1   2   2   2   2   1   1   1   2   2   1
```

```
## 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354
##   1   2   1   2   2   1   1   2   2   2   1   2   2   2   1   1   2   2   1   1
## 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
##   2   2   1   1   1   1   1   1   2   1   2   2   1   1   2   2   1   2   2   2
## 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394
##   2   2   2   2   2   1   2   1   2   2   2   2   1   1   2   2   2   1   2   2
## 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 413 414 415
##   1   2   2   2   2   2   1   2   1   2   2   2   2   2   1   2   2   1   2   1
## 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
##   1   1   2   2   2   2   1   1   2   2   1   1   1   2   2   2   2   2   2   1
## 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455
##   1   1   2   2   2   1   2   2   2   2   2   2   2   2   1   2   2   2   1   2
## 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475
##   2   1   1   2   2   2   2   2   2   2   1   1   1   2   2   2   2   2   2   2
## 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
##   2   2   2   2   1   2   2   1   1   2   2   2   1   1   1   2   1   2   1   2
## 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1
## 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
##   1   2   2   2   1   2   2   1   1   2   2   2   2   2   2   1   2   2   2   2
## 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   1   2   2   1   1   2
## 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
##   2   2   2   2   2   2   2   2   2   2   1   2   2   1   1   1   1   2   2   1
## 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595
##   2   2   2   2   2   2   1   1   2   2   2   1   2   1   2   1   1   1   2   1
## 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
##   2   2   2   2   2   2   2   2   1   1   1   2   2   1   2   1   1   1   2   2
## 616 617 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636
##   2   2   2   2   2   1   2   2   2   2   1   2   2   2   2   2   2   1   2   2
## 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
##   1   1   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2
## 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676
##   2   1   1   2   2   2   2   2   2   2   2   2   1   1   1   2   2   2   2   2
## 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696
##   2   2   2   2   1   1   2   2   2   2   2   2   2   2   2   1   2   2   2   2
## 697 698 699
##   1   1   1
##
## Within cluster sum of squares by cluster:
## [1] 23.96439 43.65250
##  (between_SS / total_SS =  77.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

- A partir de los resultados obtenidos responda lo siguiente, ¿Tiene alguna ventaja o desventaja incluir más variables? ¿Qué posible explicación puedes darle a los resultados?

Clustering jerárquico

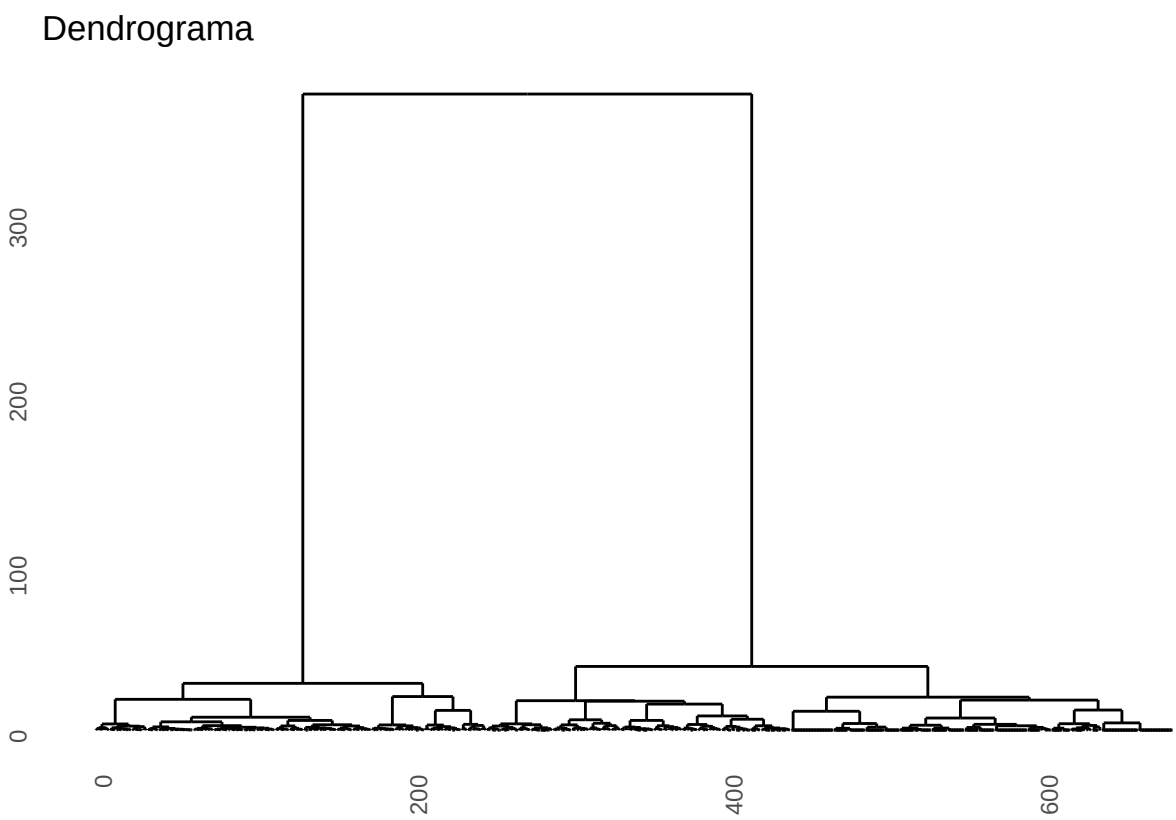
Como hemos visto, con kmeans debemos de especificar a priori el número de clusters que queremos obtener. Dado que el clustering es una tarea de aprendizaje no supervisado, en algunos casos puede que no tengamos

esa información a priori, por lo que necesitaremos de otro tipo de técnicas, como por ejemplo el agrupamiento jerárquico. Un agrupamiento jerárquico en R se puede realizar mediante la función hclust(..), para ello debemos especificar el tipo de método de aglomeración a usar.

- Crea un agrupamiento jerárquico usando hclust(..), para ello debe encontrar el método de aglomeración que mejor agrupa los tipos de cáncer. Una vez encontrado el mejor método, gráfica un dendograma que muestre información sobre el agrupamiento.

Nota: Consulta la ayuda de la función hclust(..) para ver los tipos de métodos de aglomeración que soporta. También puedes ayudarte de las funciones table(..) y plot(..) para realizar tu elección final. Además, recuerda que no debes tener en cuenta la variable Class cuando ejecutas la función hclust(..).

```
#INSERTAR CÓDIGO AQUÍ
dendrogram <- hclust(dist(BreastCancer.features, method = 'euclidean'), method = 'ward.D')
ggdendrogram(dendrogram, rotate = FALSE, labels = FALSE, theme_dendro = TRUE) + labs(title = "Dendrogra
```



- ¿Por qué el dendograma tendrá tantos grupos en el nivel más bajo? ¿Coincide este número con algún otro? Justifica.

RESPUESTA:

- ¿Por qué es necesario usar la función dist(..) al llamar a hclust(..) ?

RESPUESTA:

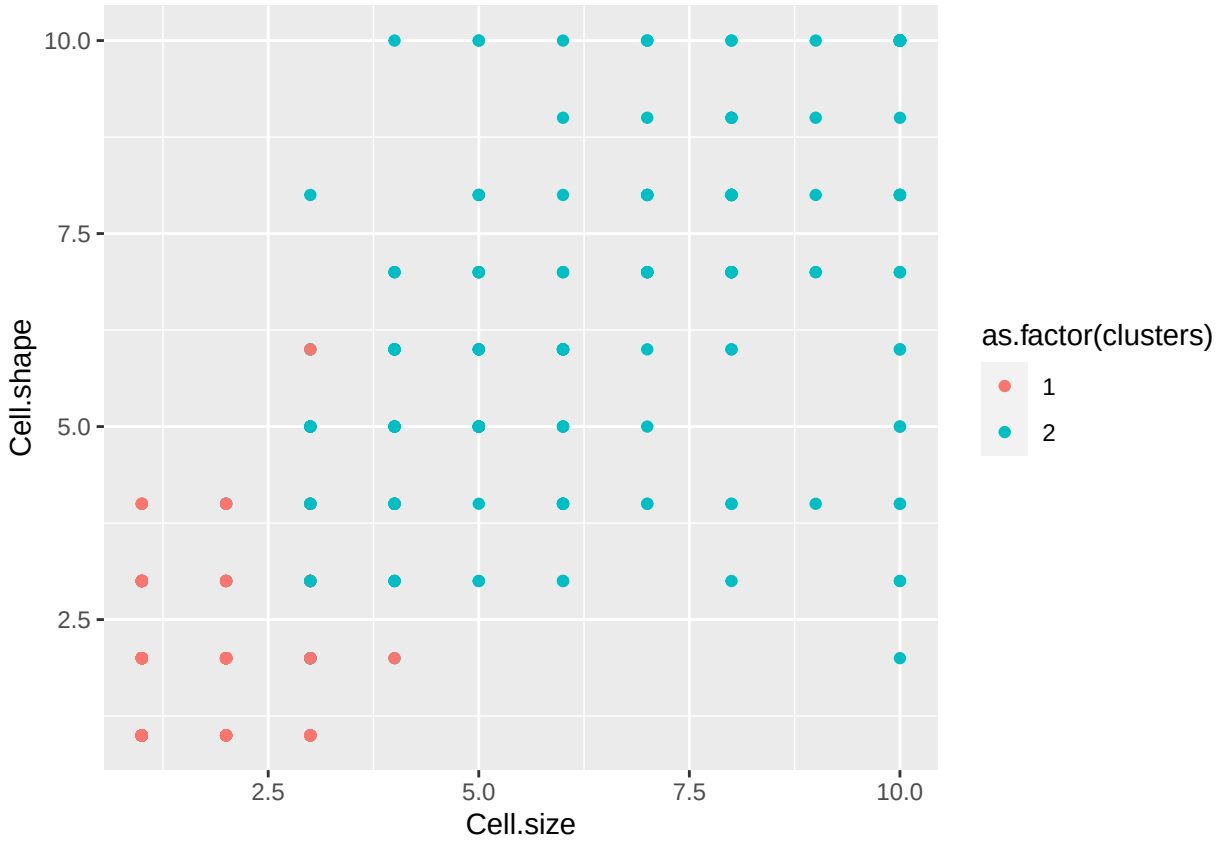
- ¿Por qué no es necesario usar set.seed(..) antes de llamar a hclust(..) ?

RESPUESTA:

- Debido a que hclust puede crear un número elevado de clústers, utilice la función cutree(..) para fijar el número de clúster igual al número de clases existentes en el dataset.

```
#INSERTAR CÓDIGO AQUÍ
agrupamientoJ <- hclust(dist(BreastCancer.features, method = 'euclidean'), method = 'ward.D')
clases_aj <- cutree(agrupamientoJ, k = 2)
BreastCancer.features$cluster <- clases_aj
clusters <- BreastCancer.features$cluster

ggplot(BreastCancer, aes(Cell.size, Cell.shape, color = as.factor(clusters))) + geom_point()
```




```
table(BreastCancer$Class, clusters)
```

```
##            clusters
##               1   2
##   benign    430  14
##   malignant   5 234
```

-¿Cuando hacemos esta última operación con la función cutree(..), el agrupamiento que obtenemos por hclust(..) es mejor al obtenido con kMeans en el ejercicio anterior?

RESPUESTA: