Ejercicio en clase y Práctico 2

Ingrid Vanessa Daza Perilla y Gonzalo Zigarán

Ejercicio de algoritmo KNN

Haciendo uso del caso visto en clase pasamos a analizar como resulta el método de KNN cuando se aplica una normalización de tipo *z-score*, como éste reacciona a distintas elecciones de K vecinos y como varia el método si se elige de manera aleatoria las muestras de validación.

Diagnosticando Cáncer:

Normalizaciń de tipo z-score

```
data <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc
data <- data[-1]
data_escalada <- scale(data[2:31], center = TRUE, scale = TRUE)
summary(data)
```

```
##  V2            V3               V4              V5
##  B:357   Min.   : 6.981   Min.   : 9.71   Min.   : 43.79
##  M:212   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17
##          Median :13.370   Median :18.84   Median : 86.24
##          Mean   :14.127   Mean   :19.29   Mean   : 91.97
##          3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10
##          Max.   :28.110   Max.   :39.28   Max.   :188.50
##        V6               V7                V8                V9
##  Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000
##  1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956
##  Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154
##  Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880
##  3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070
##  Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680
##       V10               V11              V12               V13
##  Min.   :0.00000   Min.   :0.1060   Min.   :0.04996   Min.   :0.1115
##  1st Qu.:0.02031   1st Qu.:0.1619   1st Qu.:0.05770   1st Qu.:0.2324
##  Median :0.03350   Median :0.1792   Median :0.06154   Median :0.3242
##  Mean   :0.04892   Mean   :0.1812   Mean   :0.06280   Mean   :0.4052
##  3rd Qu.:0.07400   3rd Qu.:0.1957   3rd Qu.:0.06612   3rd Qu.:0.4789
##  Max.   :0.20120   Max.   :0.3040   Max.   :0.09744   Max.   :2.8730
##       V14              V15               V16               V17
##  Min.   :0.3602   Min.   : 0.757   Min.   :  6.802   Min.   :0.001713
##  1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169
##  Median :1.1080   Median : 2.287   Median : 24.530   Median :0.006380
##  Mean   :1.2169   Mean   : 2.866   Mean   : 40.337   Mean   :0.007041
##  3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146
##  Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130
##       V18               V19               V20
##  Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
##  Median :0.020450   Median :0.02589   Median :0.010930
##  Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
```

```
##  3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
##  Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
##       V21                V22                 V23             V24
##  Min.   :0.007882   Min.   :0.0008948   Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480   1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870   Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949   Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580   3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400   Max.   :36.04   Max.   :49.54
##       V25              V26              V27               V28
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
##  Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##       V29              V30               V31              V32
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1145   1st Qu.:0.06493   1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993   Median :0.2822   Median :0.08004
##  Mean   :0.2722   Mean   :0.11461   Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140   3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100   Max.   :0.6638   Max.   :0.20750
```

División de los datos en muestra de Validación, en muestra de Entrenamiento y en Variable Objetivo.

- Muestra de Validación y Muestra de entrenamiento:

```
data_train <- data_escalada[1:469, ]
data_test  <- data_escalada[470:569, ]
```

- Salida anotada:

```
data_train_labels <- data[1:469, 1]
data_test_labels <- data[470:569, 1]
```

Ejecución del algoritmo KNN

Fijamos un número de vecinos K = 21

```
library(class)
data_test_pred <- knn(train = data_train, test = data_test, cl=data_train_labels, k=21)
```

Validación cruzada

```
library(gmodels)
CrossTable(x=data_test_labels, y=data_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
```

```
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        77 |         0 |        77 |
##                  |     1.000 |     0.000 |     0.770 |
##                  |     0.975 |     0.000 |           |
##                  |     0.770 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##                M |         2 |        21 |        23 |
##                  |     0.087 |     0.913 |     0.230 |
##                  |     0.025 |     1.000 |           |
##                  |     0.020 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        79 |        21 |       100 |
##                  |     0.790 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

Podemos concluir en este paso que hemos bajado el costo computacional usando la normalización de tipo z-score a diferencia de la *minimización minmax* sin influir en un cambio en la precisión.

Distintas predicciones en función de los distintos números de K vecinos

```
k <- c(1, 5, 11, 15, 21)
fraccion_total_v  <- c()

for (j in 1:5)
{
    data_test_pred <- knn(train = data_train, test = data_test, cl=data_train_labels,
                          k= k[j])
    validacion_cruzada = CrossTable(x=data_test_labels, y=data_test_pred,
                                    prop.chisq = FALSE)
    fraccion_total = validacion_cruzada$prop.tbl['M','M'] +
                     validacion_cruzada$prop.tbl['B','B']

    fraccion_total_v[j] <- fraccion_total
}
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
```

```
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        73 |         4 |        77 |
##                  |     0.948 |     0.052 |     0.770 |
##                  |     0.973 |     0.160 |           |
##                  |     0.730 |     0.040 |           |
## -----------------|-----------|-----------|-----------|
##                M |         2 |        21 |        23 |
##                  |     0.087 |     0.913 |     0.230 |
##                  |     0.027 |     0.840 |           |
##                  |     0.020 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        75 |        25 |       100 |
##                  |     0.750 |     0.250 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        73 |         4 |        77 |
##                  |     0.948 |     0.052 |     0.770 |
##                  |     1.000 |     0.148 |           |
##                  |     0.730 |     0.040 |           |
## -----------------|-----------|-----------|-----------|
##                M |         0 |        23 |        23 |
##                  |     0.000 |     1.000 |     0.230 |
##                  |     0.000 |     0.852 |           |
##                  |     0.000 |     0.230 |           |
## -----------------|-----------|-----------|-----------|
```

```
##     Column Total |        73 |        27 |       100 |
##                  |     0.730 |     0.270 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        76 |         1 |        77 |
##                  |     0.987 |     0.013 |     0.770 |
##                  |     0.987 |     0.043 |           |
##                  |     0.760 |     0.010 |           |
## -----------------|-----------|-----------|-----------|
##                M |         1 |        22 |        23 |
##                  |     0.043 |     0.957 |     0.230 |
##                  |     0.013 |     0.957 |           |
##                  |     0.010 |     0.220 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        77 |        23 |       100 |
##                  |     0.770 |     0.230 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
```

```
##                B |        77 |         0 |        77 |
##                  |     1.000 |     0.000 |     0.770 |
##                  |     0.975 |     0.000 |           |
##                  |     0.770 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##                M |         2 |        21 |        23 |
##                  |     0.087 |     0.913 |     0.230 |
##                  |     0.025 |     1.000 |           |
##                  |     0.020 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        79 |        21 |       100 |
##                  |     0.790 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        77 |         0 |        77 |
##                  |     1.000 |     0.000 |     0.770 |
##                  |     0.975 |     0.000 |           |
##                  |     0.770 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##                M |         2 |        21 |        23 |
##                  |     0.087 |     0.913 |     0.230 |
##                  |     0.025 |     1.000 |           |
##                  |     0.020 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        79 |        21 |       100 |
##                  |     0.790 |     0.210 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```
cat('La precisión correspondiente a k  = [1, 5, 11, 15, 21] es:' , fraccion_total_v, 'respectivamente')
```

```
## La precisión correspondiente a k  = [1, 5, 11, 15, 21] es: 0.94 0.96 0.98 0.98 0.98 respectivamente
```

Por lo tanto podemos concluir que con k = 11 se logra la misma aproximación que con valores de k mayores, lo que complejiza el problema.

Pacientes elegidos aleatoriamente para el conjunto de validación.

Realizamos el mismo procedimiento anterior a excepción de la aleatoriedad en las muestras a usar.

```
data <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc
data <- data[-1]

#####################################################################
set.seed(8)
ind = sample (x = nrow(data), size = nrow(data), replace = FALSE )
data <- data[ind,]
#####################################################################

data_escalada <- scale(data[2:31], center = TRUE, scale = TRUE)

data_train <- data_escalada[1:469, ]
data_test  <- data_escalada[470:569, ]

data_train_labels <- data[1:469, 1]
data_test_labels <- data[470:569, 1]

k <- c(1, 5, 11, 15, 21)
fraccion_total_v  <- c()

for (j in 1:5)
{
    data_test_pred <- knn(train = data_train, test = data_test, cl=data_train_labels, k= k[j])
    validacion_cruzada = CrossTable(x=data_test_labels, y=data_test_pred, prop.chisq = FALSE)
    fraccion_total = validacion_cruzada$prop.tbl['M','M'] + validacion_cruzada$prop.tbl['B','B']
    fraccion_total_v[j] <- fraccion_total
}
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        62 |         2 |        64 |
##                  |     0.969 |     0.031 |     0.640 |
##                  |     0.984 |     0.054 |           |
##                  |     0.620 |     0.020 |           |
## -----------------|-----------|-----------|-----------|
##                M |         1 |        35 |        36 |
```

```
##                  |     0.028 |     0.972 |     0.360 |
##                  |     0.016 |     0.946 |           |
##                  |     0.010 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        63 |        37 |       100 |
##                  |     0.630 |     0.370 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        64 |         0 |        64 |
##                  |     1.000 |     0.000 |     0.640 |
##                  |     0.985 |     0.000 |           |
##                  |     0.640 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##                M |         1 |        35 |        36 |
##                  |     0.028 |     0.972 |     0.360 |
##                  |     0.015 |     1.000 |           |
##                  |     0.010 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        65 |        35 |       100 |
##                  |     0.650 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
```

```
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        64 |         0 |        64 |
##                  |     1.000 |     0.000 |     0.640 |
##                  |     0.985 |     0.000 |           |
##                  |     0.640 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##                M |         1 |        35 |        36 |
##                  |     0.028 |     0.972 |     0.360 |
##                  |     0.015 |     1.000 |           |
##                  |     0.010 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        65 |        35 |       100 |
##                  |     0.650 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        63 |         1 |        64 |
##                  |     0.984 |     0.016 |     0.640 |
##                  |     0.984 |     0.028 |           |
##                  |     0.630 |     0.010 |           |
## -----------------|-----------|-----------|-----------|
##                M |         1 |        35 |        36 |
##                  |     0.028 |     0.972 |     0.360 |
##                  |     0.016 |     0.972 |           |
##                  |     0.010 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        64 |        36 |       100 |
##                  |     0.640 |     0.360 |           |
## -----------------|-----------|-----------|-----------|
##
##
##
##
##    Cell Contents
```

```
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                   | data_test_pred
## data_test_labels |         B |         M | Row Total |
## -----------------|-----------|-----------|-----------|
##                B |        64 |         0 |        64 |
##                  |     1.000 |     0.000 |     0.640 |
##                  |     0.985 |     0.000 |           |
##                  |     0.640 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##                M |         1 |        35 |        36 |
##                  |     0.028 |     0.972 |     0.360 |
##                  |     0.015 |     1.000 |           |
##                  |     0.010 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        65 |        35 |       100 |
##                  |     0.650 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```
cat('La precisión correspondiente a k  = [1, 5, 11, 15, 21] es:' , fraccion_total_v, 'respectivamente')
```

```
## La precisión correspondiente a k  = [1, 5, 11, 15, 21] es: 0.97 0.99 0.99 0.98 0.99 respectivamente
```

Observamos una mejora general en el procedimiento, en comparación con no elegir aleatoriamente. Los valores parecen no seguir un patron (antes observabamos que, en general, mejoraba mientras se agrandaba el k), sino que todos tienen una buena fracción.

Observamos que el valor de la semilla interfiere en la precisión del algoritmo KNN. Además el valor del número de vecinos en este caso ya no es necesariamente k = 11, se puede ahorrar costo computacional eligiendo un k = 5, pero esto es solo por que hemos fijado el valor de la semilla igual a 8, al variar el valor de semilla se debería elegir un valor distinto del k 'optimo'.

Podemos concluir que el algoritmo está sujeto a como se ejecutó la ''elección aleatoria" de las filas del data set. Por lo tanto se debe ser conciente de este hecho en la implementación del método a la hora de fijar un criterio en la elección del K óptimo. Aunque también puede notarse que los resultados son muy buenos para cualquier valor de k al tomar las muestras de forma aleatoria.

Práctico 2

Datos

El data set a usar es una identificación de grupos compactos **GC**. Haremos uso solo de algunos *features* del dataset de entrada para la implementación de los métodos de **K-means** y **Mixture Models**. Las entradas son coordenadas angulares *ascensión recta* **ar** y *declinación* **dec**.

```
library(VIM)
```

```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

```
## Loading required package: data.table
```

```
## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues
```

```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
read.table("2masscgs.dat") -> D

GC <- D[c(2,3)]
ra              <- GC$V2
dec             <- GC$V3
dim(GC)
```

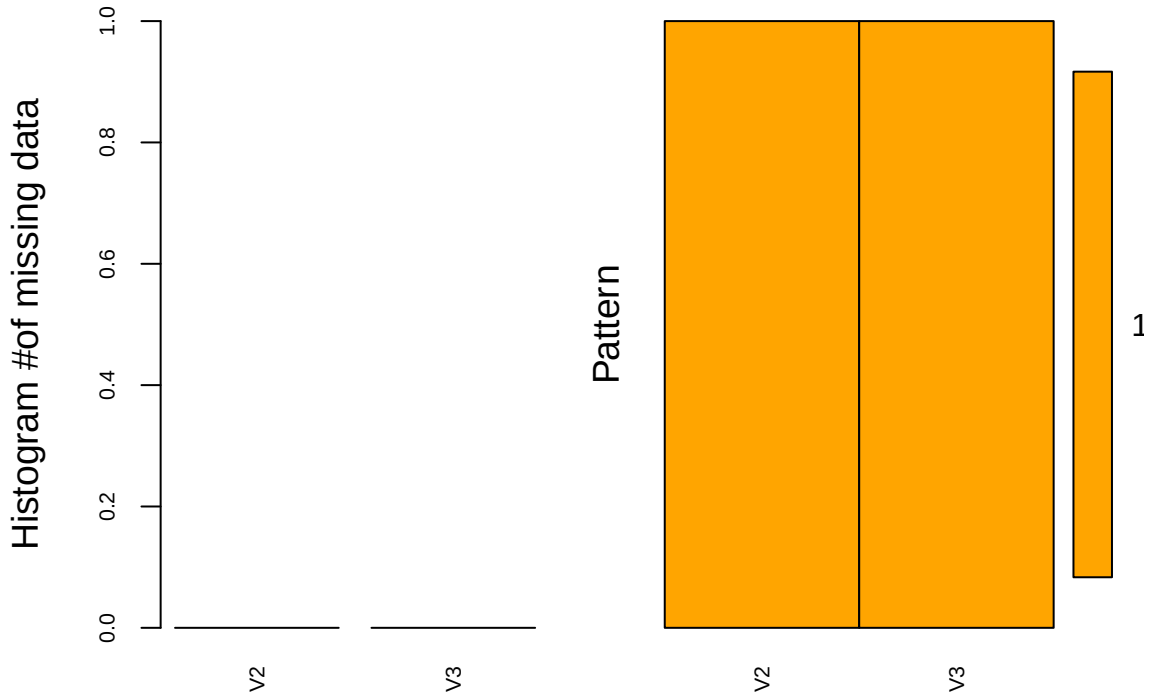
```
## [1] 85  2
```

```
summary(GC)
```

```
##        V2                V3
##  Min.   :  0.1799   Min.   :-74.2461
##  1st Qu.: 56.6316   1st Qu.:-24.3098
##  Median :173.7964   Median :  0.3924
##  Mean   :169.0966   Mean   : -0.8636
##  3rd Qu.:234.0931   3rd Qu.: 25.8928
##  Max.   :358.3984   Max.   : 85.6283
```

Iniciamos con el análisis de valores perdidos en el data set y descartamos algún orden en el data set generando un nuevo data set de manera que las filas sean aleatorias.

```
aggr(GC, col=c('orange','red','navyblue'), numbers=TRUE, sortVars=TRUE,
     cex.axis=.7, gap=3, ylab=c("Histogram #of missing data","Pattern"))
```




```
##
##  Variables sorted by number of missings:
##  Variable Count
##        V2     0
##        V3     0
```

```
set.seed(8)
ind = sample (x = nrow(D),size = nrow(D), replace = FALSE )
GC <- GC[ind,]
```

No es necesario aplicar ninguna corrección puesto que no tienen ningún valor perdido el data set. La distribución de los **GC** la podemos observar en la siguiente figura donde veremos que los grupos compactos de galaxias están repartidos en tres regiones.

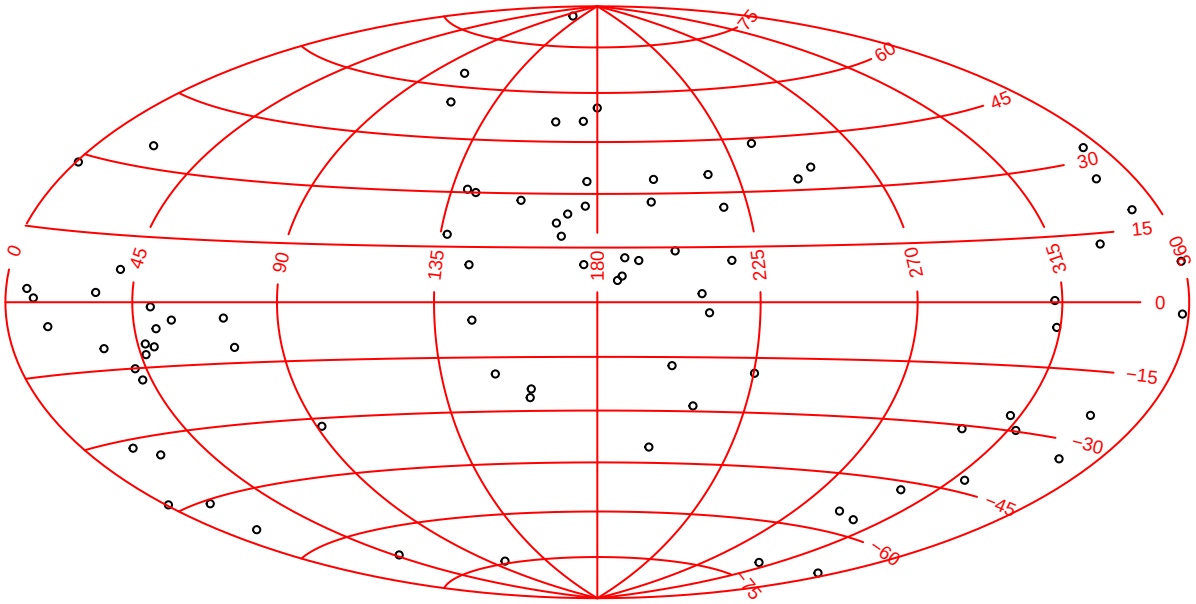
```
library(SPADAR)
```

```
## Loading required package: mapproj
```

```
## Loading required package: maps
```

```
## Loading required package: RCEIM
```

```
createAllSkyScatterPlotChart(ra, dec, pointcol = "black",
dataCoordSys = "equatorial", mainGrid = "equatorial", eqCol = "red",
eqLty = 1, eqLwd = 1,
eqDraw = TRUE, eclDraw = FALSE, galDraq = FALSE, projname = "aitoff", projparam = NULL,
projorient = NULL, nGridpoints = 100, addLab=TRUE, label.cex=0.6,
main =  'Posición Proyectada de los GCs')
```



Dado a que la **ra** es una variable angular que toma valores desde 0° a 360°, objetos que se encuentre cerca de los 0° y 360° estarán realmente cerca en proyección, lo cual el método de k-means no lo percibe de esta manera, por esto se decidió para tener mejores particiones dividir la muestra en dos grupos. El primer grupo abarcara todos los **GC** que están en la zona de **ra < 180°** (GC_1) y el segundo grupo abarca los **GC** que tengan una **ra > 180°** (GC_2).

Método K-means

Nosotros decidimos iniciar con el Método K-means e implementar el *Método del Codo* para elegir el número de centros K "optimo" para cada grupo (GC_1 y GC_2). Y en cada caso se observo como este método varia al emplear la normalización por z-score.

- **Método del codo**

Este método utiliza los valores de la inercia obtenidos tras aplicar K-means (desde m a N Clusters), siendo la inercia la suma de las distancias al cuadrado de cada objeto del Cluster a su centroide

$$inercia = \sum_{i=m}^{N} \|x_i - \mu\|^2$$

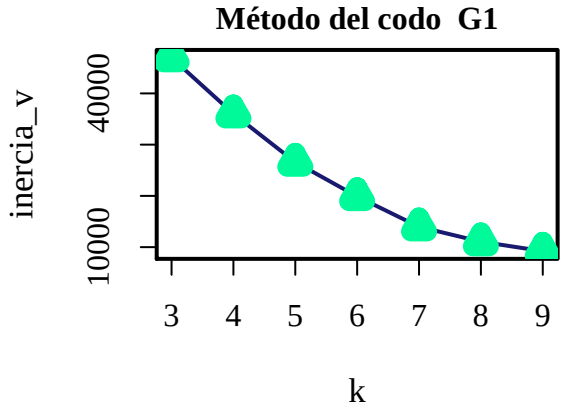
Una vez obtenidos los valores de la inercia tras aplicar el K-means de m a N Clusters, representamos en una gráfica lineal la inercia respecto del número de Clusters. En esta gráfica se debería de apreciar un cambio brusco en la evolución de la inercia, teniendo la línea representada una forma similar a la de un brazo y su codo. El punto en el que se observa ese cambio brusco en la inercia nos dirá el número óptimo de Clusters a seleccionar para ese data set; o dicho de otra manera: el punto que representaría al codo del brazo será el número óptimo de Clusters para ese data set.

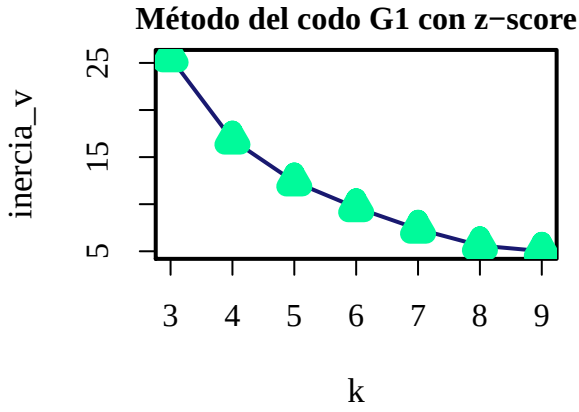
En particular nosotros tomamos m = 3 y N = 9.

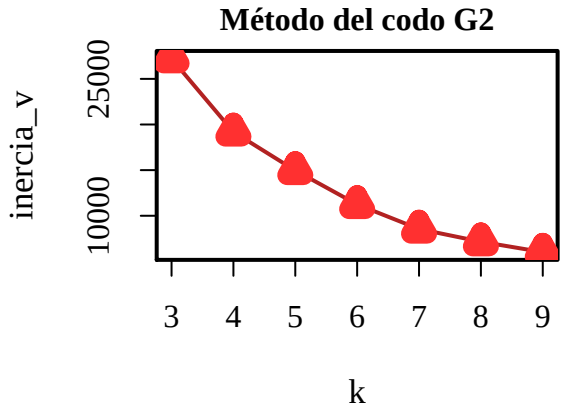
```
par(lwd=2)
par(mar=c(5,5,2,2))
```

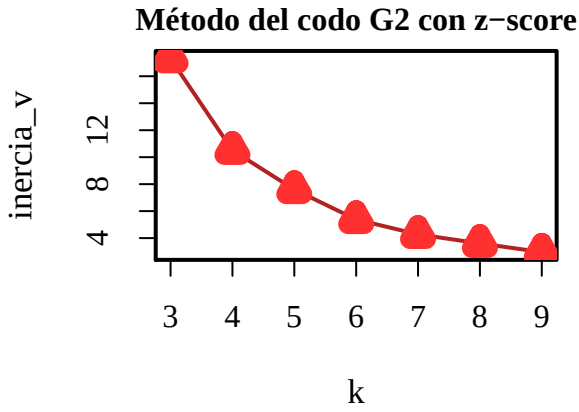
```r
par(mfrow=c(2,2))
par(mgp=c(3.7,1.3,0))
par(cex.axis=1.2,cex.lab=1.3)
par(family="serif")

library(VIM)
read.table("2masscgs.dat") -> D
GC <- D[c(2,3)]
ra              <- GC$V2
dec             <- GC$V3

################################## Met.Codo GC_1 ######################################
k <- c(3,4,5,6,7,8,9)                                                                  #
inercia_v <- c()                                                                       #
GC_1 <- subset(GC,ra < 180)                                                            #
                                                                                       #
for (j in 1:7)                                                                         #
    {                                                                                  #
    set.seed(20)                                                                       #
    GC_1_Cluster <- kmeans(GC_1, k[j], nstart = 20)                                    #
    inercia <- GC_1_Cluster$tot.withinss                                               #
    inercia_v[j] <- inercia                                                            #
    }                                                                                  #
plot(k,inercia_v, type = "l",col='midnightblue',
     main= 'Método del codo  G1')                                                      #
points(k,inercia_v,col= 'mediumspringgreen',pch=2,lwd = 10)                            #
                                                                                       #
          ################# con escalado z-score ################                    #
                                                                                       #
k <- c(3,4,5,6,7,8,9)                                                                  #
inercia_v <- c()                                                                       #
GC_1_escalado <- scale(GC_1, center = TRUE, scale = TRUE)                              #
                                                                                       #
for (j in 1:7)                                                                         #
    {                                                                                  #
    set.seed(20)                                                                       #
    GC_1_Cluster_esc <- kmeans(GC_1_escalado, k[j], nstart = 20)                       #
    inercia <- GC_1_Cluster_esc$tot.withinss                                           #
    inercia_v[j] <- inercia                                                            #
    }                                                                                  #
plot(k,inercia_v, type = "l",col='midnightblue',                                       #
     main= 'Método del codo G1 con z-score')
points(k,inercia_v,col= 'mediumspringgreen',pch=2,lwd = 10)                            #
########################################################################################



################################## Met.Codo GC_2 ######################################
                                                                                       #
k <- c(3,4,5,6,7,8,9)                                                                  #
inercia_v <- c()                                                                       #
GC_2 <- subset(GC,ra > 180)                                                            #
                                                                                       #
```

```r
                                                                                #
for (j in 1:7)                                                                  #
    {                                                                           #
    set.seed(20)                                                                #
    GC_2_Cluster <- kmeans(GC_2, k[j], nstart = 20)                             #
    inercia <- GC_2_Cluster$tot.withinss                                        #
    inercia_v[j] <- inercia                                                     #
    }                                                                           #
plot(k,inercia_v, type = "l",col='firebrick',                                   #
     main= 'Método del codo G2')                                                #
points(k,inercia_v,col= 'firebrick1',pch=2,lwd = 10)                            #
                                                                                #
            ################## con escalado z-score ############################ #
k <- c(3,4,5,6,7,8,9)                                                           #
inercia_v <- c()                                                                #
                                                                                #
GC_2_escalado <- scale(GC_2, center = TRUE, scale = TRUE)                       #
for (j in 1:7)                                                                  #
    {                                                                           #
    set.seed(20)                                                                #
    GC_2_Cluster_esc <- kmeans(GC_2_escalado, k[j], nstart = 20)                #
    inercia <- GC_2_Cluster_esc$tot.withinss                                    #
    inercia_v[j] <- inercia                                                     #
    }                                                                           #
plot(k,inercia_v, type = "l",col='firebrick',                                   #
     main= 'Método del codo G2 con z-score')                                    #
points(k,inercia_v,col= 'firebrick1',pch=2,lwd = 10)                            #
```









```
#########################################################################################
```

Como se puede apreciar en los resultados obtenidos, en el conjunto de datos sin normalizar no se aprecie "el codo" a diferencia de el conjunto de datos normalizado donde se ve claramente un cambio brusco en $\mathbf{k = 4}$ tanto para GC_1 como para GC_2.

Tomando $\mathbf{k = 4}$ mostramos las particiones al implementar K-means.

```
par(lwd=2)
par(mar=c(5,5,2,2))
par(mfrow=c(2,2))
par(mgp=c(3.7,1.3,0))
par(cex.axis=1.2,cex.lab=1.3)
par(family="serif")

library(VIM)
read.table("2masscgs.dat") -> D
GC <- D[c(2,3)]
ra              <- GC$V2
dec             <- GC$V3

################################### K-means GC_1 ########################################
GC_1 <- subset(GC,ra < 180)                                                             #
dim(GC_1)                                                                               #
```

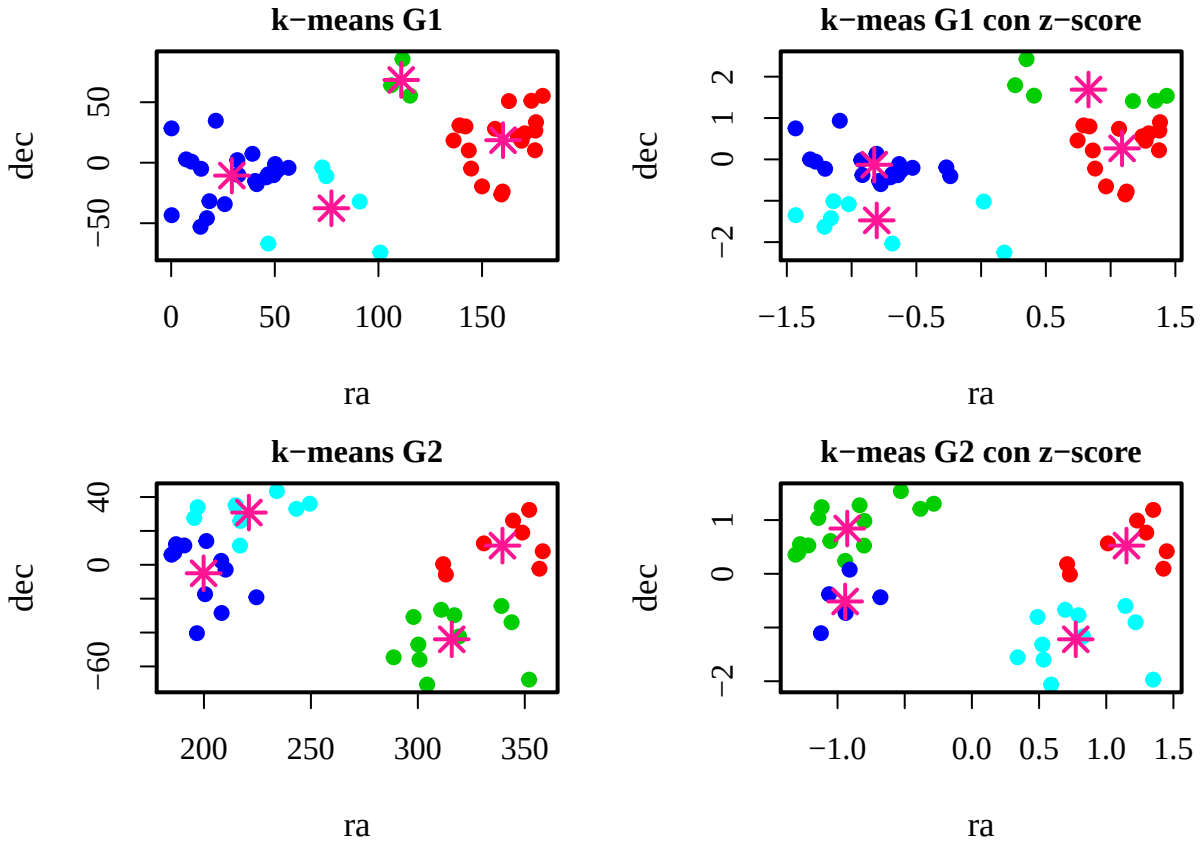
```
## [1] 47  2
```

```
                                                                                        #
set.seed(20)                                                                            #
GC_1_Cluster <- kmeans(GC_1, 4, nstart = 20)                                            #
                                                                                        #
plot(GC_1, col = GC_1_Cluster$cluster+1, xlab='ra', ylab='dec', pch=19,                 #
     main= 'k-means G1')                                                                #
points(GC_1_Cluster$centers, pch=8, col='deeppink', cex=2)                              #
                                                                                        #
         ################# con escalado z-score ############################          #
                                                                                        #
GC_1_escalado <- scale(GC_1, center = TRUE, scale = TRUE)                               #
                                                                                        #
set.seed(20)                                                                            #
GC_1_Cluster_esc <- kmeans(GC_1_escalado, 4, nstart = 20)                               #
                                                                                        #
plot(GC_1_escalado, col = GC_1_Cluster_esc$cluster+1, xlab='ra',ylab='dec',pch=19,      #
     main= 'k-meas G1 con z-score')                                                     #
points(GC_1_Cluster_esc$centers, pch=8, col='deeppink', cex=2)                          #
#########################################################################################


################################### K-means GC_2 ########################################
GC_2 <- subset(GC,ra > 180)                                                             #
dim(GC_2)                                                                               #
```

```
## [1] 38  2
```

```
                                                                                        #
set.seed(20)                                                                            #
GC_2_Cluster <- kmeans(GC_2, 4, nstart = 20)                                            #
```

```
                                                                                #
plot(GC_2, col = GC_2_Cluster$cluster+1, xlab='ra', ylab='dec', pch=19,         #
     main= 'k-means G2')                                                        #
points(GC_2_Cluster$centers, pch=8, col='deeppink', cex=2)                      #
                                                                                #
         ################# con escalado z-score ############################    #
                                                                                #
GC_2_escalado <- scale(GC_2, center = TRUE, scale = TRUE)                       #
                                                                                #
set.seed(20)                                                                    #
GC_2_Cluster_esc <- kmeans(GC_2_escalado, 4, nstart = 20)                       #
                                                                                #
plot(GC_2_escalado, col = GC_2_Cluster_esc$cluster+1, xlab='ra',ylab='dec',pch=19,  #
     main= 'k-meas G2 con z-score')                                             #
points(GC_2_Cluster_esc$centers, pch=8, col='deeppink', cex=2)                  #
```




```
################################################################################
```

En la figura se muestran las cuatro particiones en distintos colores y con asterisco los centros de los mismos.

Lo primero que podemos comentar en este resultado es que se distingue una división entre dos zonas dada por la posición intrínseca de los grupos compactos como se mostró en la figura de “Posición proyectada de GCs”.

También es notable como la normalización influye en el resultado de las particiones tanto en el tamaño como en la posición de los centros. Respecto a las particiones observados éstas son posibles agrupaciones en el espacio, para saber si realmente estas agrupaciones son reales se debe analizar con la distancia de cada Grupo Compacto.

Método de Mixtura de Gaussianas.

Este método se implemento con un K fijo dado por el valor resultante en la elección del número de centros en k-meas, la razón es que la instauración de criterios para la elección del número de gaussianas en este método no son fáciles y están poco documentados.

```
par(lwd=2)
par(mar=c(5,5,2,2))
par(mfrow=c(2,2))
par(mgp=c(3.7,1.3,0))
par(cex.axis=1.2,cex.lab=1.3)
par(family="serif")

library(mclust)
```

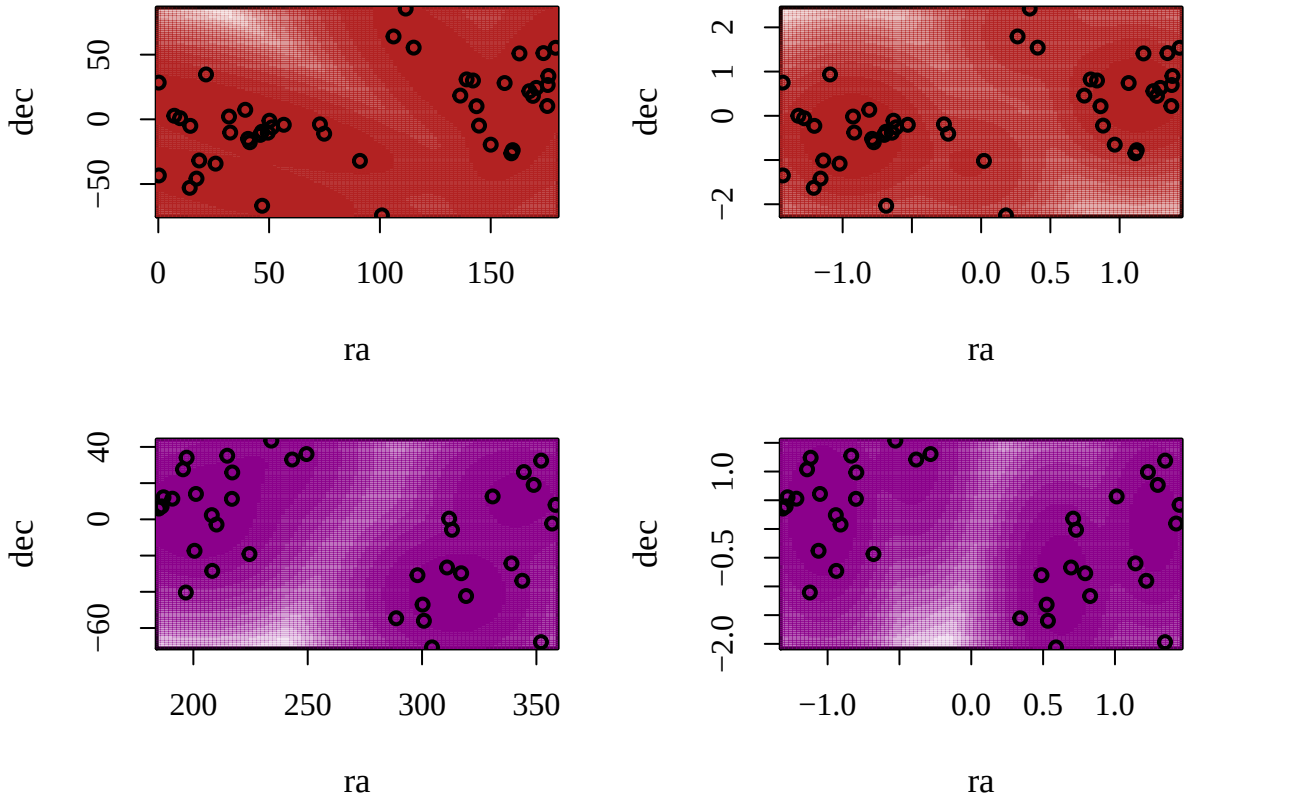
```
## Package 'mclust' version 5.4
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

```
## The following object is masked from 'package:maps':
##
##     map
```

```
library(VIM)
read.table("2masscgs.dat") -> D
GC <- D[c(2,3)]
ra              <- GC$V2
dec             <- GC$V3

################################## Mix.Gaussiana GC_1 ######################################
GC_1 <- subset(GC,ra < 180)                                                                  #
mcl.model <- Mclust(GC_1, 4)                                                                 #
plot(mcl.model, what = "density", type = "image", xlab='ra', ylab='dec',                     #
     main= 'Mix.Gaussiana G1', col='firebrick')                                              #
points(GC_1)                                                                                 #
                                                                                             #
                       ########  ESCALADO CON  Z.SCORE #########                           #
                                                                                             #
GC_1 <- subset(GC, ra < 180)                                                                 #
GC_1_escalado <- scale(GC_1, center = TRUE, scale = TRUE)                                    #
mcl.model <- Mclust(GC_1_escalado, 4)                                                        #
plot(mcl.model, what = "density", type = "image", xlab='ra', ylab='dec', col='firebrick',    #
     main= 'Mix.Gaussiana G1 con z-score')                                                   #
points(GC_1_escalado)                                                                        #
##############################################################################################



################################## Mix.Gaussiana GC_2 ######################################
                                                                                             #
GC_2 <- subset(GC, ra > 180)                                                                 #
mcl.model <- Mclust(GC_2, 4)                                                                 #
plot(mcl.model, what = "density", type = "image", col ='darkmagenta', xlab='ra',ylab='dec', #
     main= 'Mix.Gaussiana G2')                                                               #
points(GC_2)                                                                                 #
```

```
                        ########  ESCALADO CON  Z.SCORE #########                                  #
GC_2 <- subset(GC,ra > 180)                                                                       #
GC_2_escalado <- scale(GC_2, center = TRUE, scale = TRUE)                                         #
mcl.model <- Mclust(GC_2_escalado, 4)                                                             #
plot(mcl.model, what = "density", type = "image", col ='darkmagenta', xlab='ra',ylab='dec',      #
     main= 'Mix.Gaussiana G1 con z-score')                                                        #
points(GC_2_escalado)                                                                             #
```




```
#############################################################################################
```

El gráfico muestra los resultados de aplicar Mixture Model a los grupos GC_1 y GC_2 en primera y segunda fila respectivamente, acompañados de los resultados al usar los datos normalizados por z-score.

En el gráfico anterior al igual que en el gráfico resultante del método de k-means se ve claramente dos zonas de sobredensidades y como se justificó anteriormente es por que el data set usado es un catálogo de las posiciones en ciertas zonas del cielo.

Conclusión:

- **Método KNN** Podemos concluir que el algoritmo está sujeto a como se ejecutó la ''elección aleatoria" de las filas del data set (i.e que las muestras de validación y entrenamiento no siguen una real aleatoriedad). Por lo tanto se debe ser consciente de este hecho en la implementación del método a la hora de fijar un criterio en la elección del K óptimo. El aprendizaje de KNN depende fuertemente del orden del data set.
- **k-means y Mixture Model** Los resultados obtenidos, en el conjunto de datos sin normalizar no se aprecie "el codo" a diferencia del conjunto de datos normalizado donde se ve claramente un cambio brusco en $\mathbf{k = 4}$ tanto para GC_1 como para GC_2.

Lo primero que podemos comentar en este resultado es que se distingue una división entre dos zonas dada por la posición intrínseca de los grupos compactos como se mostró en la figura de “Posición proyectada de GCs”.

También es notable como la normalización influye en el resultado de las particiones tanto en el tamaño como en la posición de los centros. Respecto a las particiones observados éstas son posibles agrupaciones en el espacio, para saber si realmente estas agrupaciones son reales se debe analizar con la distancia de cada Grupo Compacto. Para nosotros sacar conclusiones de la forma de los grupos y sus detalles fue mas fácil al implementar el método de K-means