22/23

Machine Learning con R

Master Big Data & Business Analytics

BRUNO URBAN ALFARO

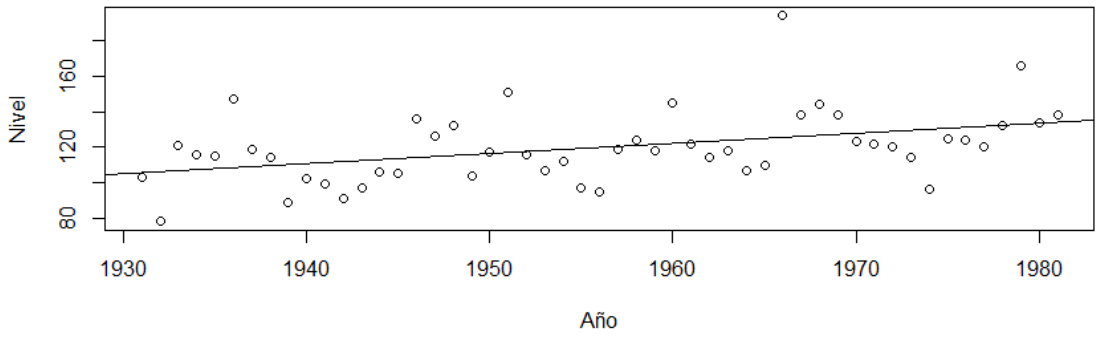
UNED

Contenido

Ejercicio 1: Regresión lineal

a) Dibujar la recta de ajuste del modelo junto con la correspondiente nube de puntos.

```
regresion <- lm(nivel~año, data = venecia)
plot(venecia$año, venecia$nivel, xlab = "Año", ylab = "Nivel")
abline(regresion)
```



b) Obtener un resumen del ajuste.

```
> summary(regresion)

Call:
lm(formula = nivel ~ año, data = venecia)

Residuals:
    Min      1Q  Median      3Q     Max
-33.813 -11.211  -3.309   9.515  68.722

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -989.3822   346.4770  -2.856  0.00628 **
año            0.5670     0.1771   3.201  0.00241 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.62 on 49 degrees of freedom
Multiple R-squared:  0.1729,    Adjusted R-squared:  0.1561
F-statistic: 10.25 on 1 and 49 DF,  p-value: 0.002406
```

c) ¿Cuánto vale el coeficiente de correlación al cuadrado en este caso?

El coeficiente de correlación como podemos ver en el resumen es 0.1729

d) ¿Cuánto valen los estimadores de todos los parámetros del modelo?

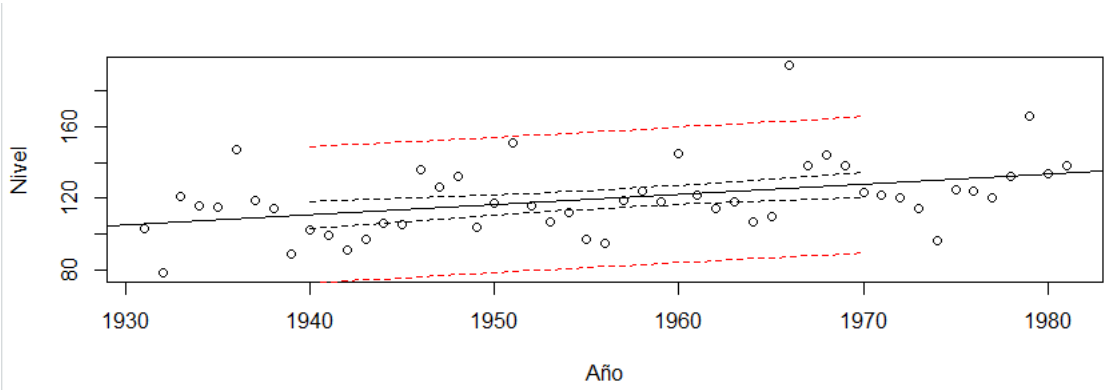
Los estimadores de todos los parámetros del modelo como podemos ver en el resumen son -989.3822 y 0.5670

e) Calcula un intervalo de confianza con un nivel del 90%

```
> confint(regresion, level = 0.9)
                    5 %         95 %
(Intercept) -1570.2684806 -408.4959237
año             0.2700001    0.8639365
```

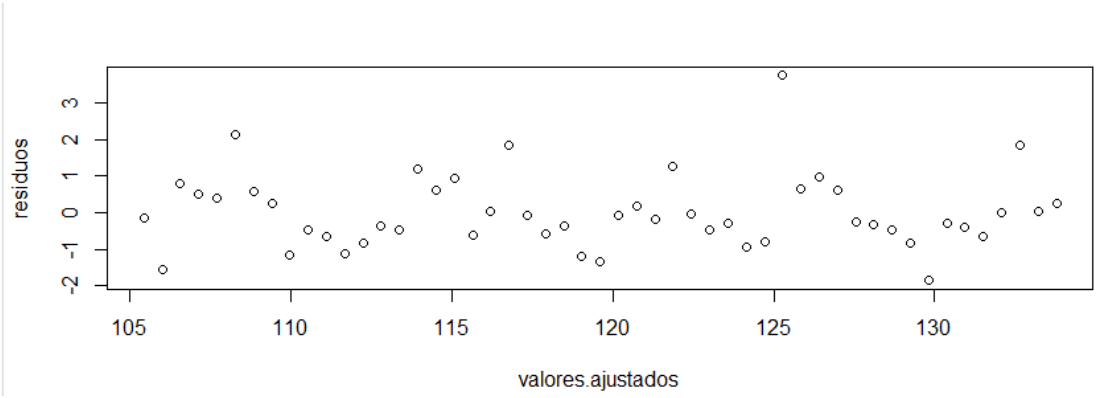
f) Calcular y representar los intervalos de confianza y de predicción al 95% del nivel medio para los años comprendidos entre 1940 y 1970.

```
nuevos.anos <- data.frame(año = seq(1940, 1970))
ic <-predict(regresion, nuevos.anos, interval = "confidence")
lines(nuevos.anos$año, ic[, 2], lty = 2)
lines(nuevos.anos$año, ic[, 3], lty = 2)
ic <-predict(regresion, nuevos.anos, interval = "prediction")
lines(nuevos.anos$año, ic[, 2], lty = 2, col = "red")
lines(nuevos.anos$año, ic[, 3], lty = 2, col = "red")
```

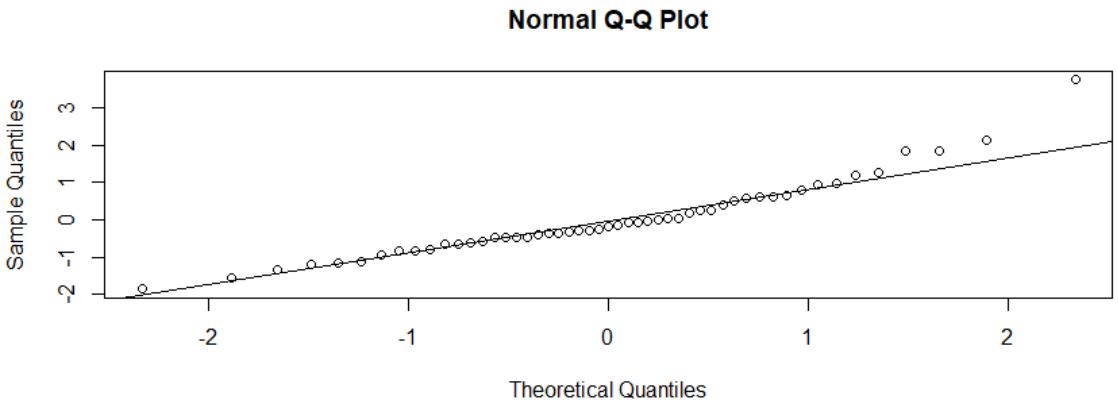


g) Calcular los residuos estandarizados frente a los valores ajustados y verificar la hipótesis de normalidad.

```
residuos <-rstandard(regresion)
valores.ajustados <- fitted(regresion)
plot(valores.ajustados, residuos)
```




```
qqnorm(residuos)
qqline(residuos)
```

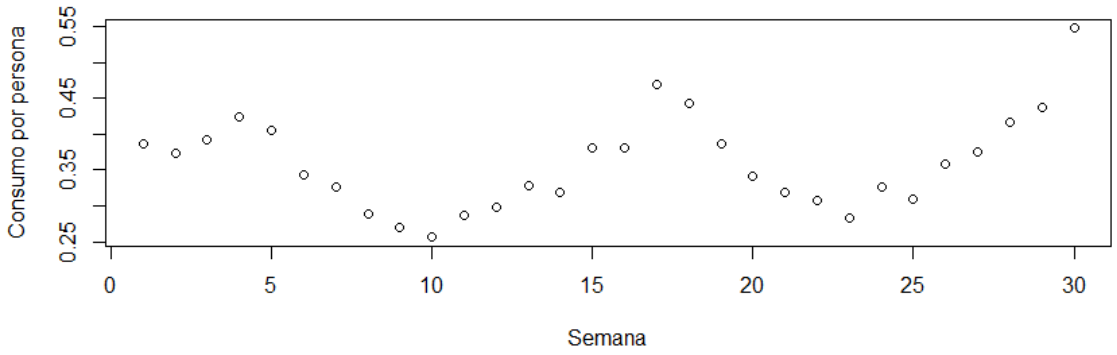


Tanto la homocedasticidad, linealidad y normalidad parecen razonables.

Ejercicio 2:Regresión lineal múltiple

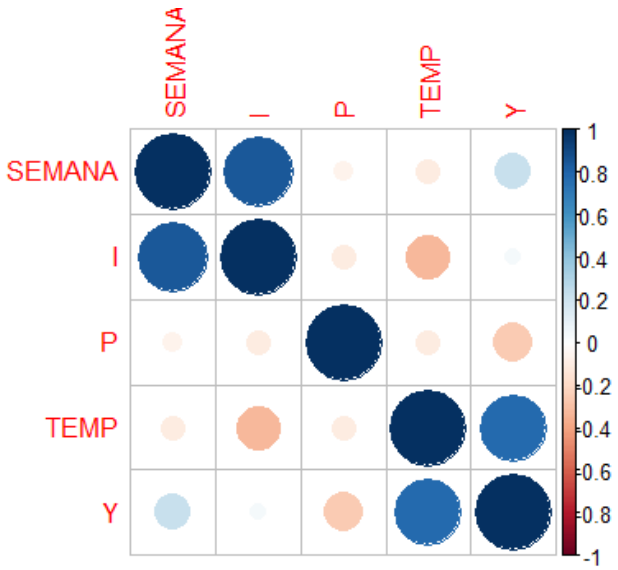
a) Representar gráficamente el consumo de granizados en función de las semanas.

```
plot(SEMANA, Y, xlab = "Semana", ylab = "Consumo por persona")
```



b) Determinar la matriz de correlación de las variables y, p, i y temp.

```
granizado_cor <- cor(granizados, method = "pearson")
library(corrplot)
rrplot 0.92 loaded
corrplot(granizado_cor)
```



c) ¿Cuál es la variable que parece tener más influencia en y?

Gracias a la matriz de correlación podemos definir que la variable que más influencia tiene en Y es la de TEMP.

d) Realizar un ajuste lineal de y sobre p, i y temp. Dibujar la evolución predicha por nuestro modelo. Analizar si todas las variables son igual de significativas en el modelo.

```
> ajuste <- lm(Y ~ P + I + TEMP, data = granizados)
> summary(ajuste)

Call:
lm(formula = Y ~ P + I + TEMP, data = granizados)

Residuals:
      Min        1Q    Median        3Q       Max
-0.065302 -0.011873  0.002737  0.015953  0.078986

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.1973151  0.2702162   0.730  0.47179
P           -1.0444140  0.8343573  -1.252  0.22180
I            0.0033078  0.0011714   2.824  0.00899 **
TEMP         0.0034584  0.0004455   7.762  3.1e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03683 on 26 degrees of freedom
Multiple R-squared:  0.719,     Adjusted R-squared:  0.6866
F-statistic: 22.17 on 3 and 26 DF,  p-value: 2.451e-07
```

No todas las variables son igual de significativas en este modelo, ya que tenemos una diferenciación clara en los coeficientes de regresión.

e) ¿Cuánto vale la varianza residual y R2 ?

Con la función *anova* podemos determinar el análisis de la varianza

```
> anova(ajuste)
Analysis of Variance Table

Response: Y
          Df   Sum Sq  Mean Sq F value  Pr(>F)
P          1 0.008459 0.008459  6.2351 0.01919 *
I          1 0.000051 0.000051  0.0376 0.84781
TEMP       1 0.081741 0.081741 60.2519 3.1e-08 ***
Residuals 26 0.035273 0.001357
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Podemos filtrar R2 del resumen para obtener el dato más claramente, aunque ya estaba en el resumen.

```
> summary(ajuste)$r.square
[1] 0.7189939
```

f) Realizar un ajuste lineal de y sobre i y temp. ¿Cuánto vale en este caso la varianza residual y R2?

Realizamos los mismos pasos anteriores pero para Y sobre I y TEMP:

```
> ajuste2 <- lm(Y ~ I + TEMP, data = granizados)
> summary(ajuste2)

Call:
lm(formula = Y ~ I + TEMP, data = granizados)

Residuals:
      Min        1Q    Median        3Q       Max
-0.065420 -0.022458  0.004026  0.015987  0.091905

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.113195   0.108280  -1.045  0.30511
I            0.003530   0.001170   3.017  0.00551 **
TEMP         0.003543   0.000445   7.963 1.47e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03722 on 27 degrees of freedom
Multiple R-squared:  0.7021,    Adjusted R-squared:   0.68
F-statistic: 31.81 on 2 and 27 DF,  p-value: 7.957e-08

> anova(ajuste2)
Analysis of Variance Table

Response: Y
          Df   Sum Sq  Mean Sq F value   Pr(>F)
I          1 0.000288 0.000288  0.2082   0.6518
TEMP       1 0.087836 0.087836 63.4137 1.47e-08 ***
Residuals 27 0.037399 0.001385
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(ajuste2)$r.square
[1] 0.7020589
```

Ejercicio 3: KNN

a) Cargar los datos y normalizarlos.

Cargamos los datos con la función read y comprobamos que se han cargado correctamente:

```
Data
data                    100 obs. of 10 variables
  $ id                : int  1 2 3 4 5 6 7 8 9 10 ...
  $ diagnosis_result  : chr  "M" "B" "M" "M" ...
  $ radius            : int  23 9 21 14 9 25 16 15 19 25 ...
  $ texture           : int  12 13 27 16 19 25 26 18 24 11 ...
  $ perimeter         : int  151 133 130 78 135 83 120 90 88 84 ...
  $ area              : int  954 1326 1203 386 1297 477 1040 578 520 476 ...
  $ smoothness        : num  0.143 0.143 0.125 0.07 0.141 0.128 0.095 0.119 0.127 0.119 ...
  $ compactness       : num  0.278 0.079 0.16 0.284 0.133 0.17 0.109 0.165 0.193 0.24 ...
  $ symmetry          : num  0.242 0.181 0.207 0.26 0.181 0.209 0.179 0.22 0.235 0.203 ...
  $ fractal_dimension: num  0.079 0.057 0.06 0.097 0.059 0.076 0.057 0.075 0.074 0.082 ...
```

Como el ID no es necesario lo borramos `data <- data[, -1]`

Definimos la función de normalización

```
normalize <- function(x) {
    return ((x - min(x)) / (max(x) - min(x))) }
```

Y, como hemos eliminado el ID, normalizamos desde el segundo valor, ya que diagnosis_result no es numérico

```
tumor_n <- as.data.frame(lapply(data[2:9], normalize))
```

Podemos ver el efecto de la normalización:

```
> summary(data)
 diagnosis_result       radius         texture         perimeter          area          smoothness       compactness
 Length:100         Min.   : 9.00   Min.   :11.00   Min.   : 52.00   Min.   : 202.0   Min.   :0.0700   Min.   :0.0380
 Class :character   1st Qu.:12.00   1st Qu.:14.00   1st Qu.: 82.50   1st Qu.: 476.8   1st Qu.:0.0935   1st Qu.:0.0805
 Mode  :character   Median :17.00   Median :17.50   Median : 94.00   Median : 644.0   Median :0.1020   Median :0.1185
                    Mean   :16.85   Mean   :18.23   Mean   : 96.78   Mean   : 702.9   Mean   :0.1027   Mean   :0.1267
                    3rd Qu.:21.00   3rd Qu.:22.25   3rd Qu.:114.25   3rd Qu.: 917.0   3rd Qu.:0.1120   3rd Qu.:0.1570
                    Max.   :25.00   Max.   :27.00   Max.   :172.00   Max.   :1878.0   Max.   :0.1430   Max.   :0.3450
    symmetry      fractal_dimension
 Min.   :0.1350   Min.   :0.05300
 1st Qu.:0.1720   1st Qu.:0.05900
 Median :0.1900   Median :0.06300
 Mean   :0.1932   Mean   :0.06469
 3rd Qu.:0.2090   3rd Qu.:0.06900
 Max.   :0.3040   Max.   :0.09700
```

b) Crear los conjuntos de datos de entrenamiento y prueba. Dividir para esto los datos en dos partes, de manera que, de las 100 observaciones, los datos del 1 al 65 los tomamos como datos de entrenamiento y los datos del 66 al 100 como datos de prueba.

Como sabemos qué datos usar de observación y cuales de prueba no nos hace falta fijar una semilla. Únicamente definir que los datos del 1-65 son training y el resto test.

```
tumor.training <- tumor_n[1:65,]
tumor.test <- tumor_n[66:100,]
tumor_train_labels <- tumor_n[1:65, 1]
tumor_test_labels <- tumor_n[66:100, 1]
```

c) Construir el clasificador

Clasificamos los datos de prueba

```
tumor_pred <- knn(train = tumor.training, test = tumor.test,cl= tumor_train_labels, k=3)
```

Observamos el resultado:

```
> tumor_pred
 [1] 0.0625 0.0625 0.5625 0.4375 0.625  0.6875 0.1875 0.5625 0.5625 0.5625 0.9375 0.375  0.125  0.3125 0.9375 0.0625
[17] 0.5    0.125  0.5625 0.9375 0.5    0.4375 0.6875 0.0625 0.375  0.5625 0.0625 0.5625 0.0625 0.6875 0.6875 0.5625
[33] 0.875  0.6875 0.4375
Levels: 0 0.0625 0.125 0.1875 0.3125 0.375 0.4375 0.5 0.5625 0.625 0.6875 0.75 0.8125 0.875 0.9375 1
```

d) Evaluar el modelo

Lanzamos la siguiente instrucción:

```
> CrossTable(x=tumor_test_labels, y = tumor_pred, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  35
```

Podemos ver el total de nuestra columna tabla y las predicciones que se han realizado:

tumor_test_labels	tumor_pred 0.0625	0.125	0.1875	0.3125	0.375	0.4375	0.5	0.5625	0.625	0.6875	0.875	0.9375	Row Total
0	0	0	1	0	0	0	0	0	0	0	0	0	1
	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.029
	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.000	0.000	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
0.0625	3	1	0	0	1	0	0	0	0	0	0	0	5
	0.600	0.200	0.000	0.000	0.200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.143
	0.500	0.500	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.086	0.029	0.000	0.000	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
0.125	2	1	0	0	0	0	0	0	0	0	0	0	3
	0.667	0.333	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.086
	0.333	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.057	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
0.1875	1	0	0	0	1	0	0	0	0	0	0	0	2
	0.500	0.000	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.057
	0.167	0.000	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.029	0.000	0.000	0.000	0.029	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
0.3125	0	0	0	0	0	0	1	1	0	0	0	0	2
	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.500	0.000	0.000	0.000	0.000	0.057
	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.125	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.029	0.029	0.000	0.000	0.000	0.000	
0.4375	0	0	0	1	0	2	0	0	0	0	0	0	3
	0.000	0.000	0.000	0.333	0.000	0.667	0.000	0.000	0.000	0.000	0.000	0.000	0.086
	0.000	0.000	0.000	1.000	0.000	0.667	0.000	0.000	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.029	0.000	0.057	0.000	0.000	0.000	0.000	0.000	0.000	
0.5	0	0	0	0	0	0	1	0	1	0	0	0	2
	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.000	0.500	0.000	0.000	0.000	0.057
	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.000	1.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.029	0.000	0.029	0.000	0.000	0.000	
0.5625	0	0	0	0	0	0	0	2	0	0	0	0	2
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.057
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.057	0.000	0.000	0.000	0.000	
0.625	0	0	0	0	0	1	0	0	0	1	1	0	3
	0.000	0.000	0.000	0.000	0.000	0.333	0.000	0.000	0.000	0.333	0.333	0.000	0.086
	0.000	0.000	0.000	0.000	0.000	0.333	0.000	0.000	0.000	0.200	1.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.029	0.000	0.000	0.000	0.029	0.029	0.000	

0.6875	0	0	0	0	0	0	0	1	0	0	0	0	1
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.029
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.125	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.029	0.000	0.000	0.000	0.000	
0.75	0	0	0	0	0	0	0	1	0	2	0	1	4
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.500	0.000	0.250	0.114
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.125	0.000	0.400	0.000	0.333	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.029	0.000	0.057	0.000	0.029	
0.8125	0	0	0	0	0	0	0	2	0	1	0	1	4
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.000	0.250	0.000	0.250	0.114
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.200	0.000	0.333	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.057	0.000	0.029	0.000	0.029	
0.875	0	0	0	0	0	0	0	1	0	1	0	0	2
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.000	0.500	0.000	0.000	0.057
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.125	0.000	0.200	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.029	0.000	0.029	0.000	0.000	
1	0	0	0	0	0	0	0	0	0	0	0	1	1
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.029
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.333	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.029	
Column Total	6	2	1	1	2	3	2	8	1	5	1	3	35
	0.171	0.057	0.029	0.029	0.057	0.086	0.057	0.229	0.029	0.143	0.029	0.086	

Ejercicio 5: Algoritmo Jerárquico Aglomerativo

a) Realizar una clasificación jerárquica de los países en base a su consumo de proteínas según las distintas fuentes de alimentación. Utilizar el método de Ward (D.Ward), y especificar que los casos se etiqueten con la variable Country.

Primero tipificamos los datos con la función dada por el enunciado

```
proteinas.tip=scale(proteinas.dat)
```

A continuación vamos a aplicar el método Ward, que se basa en una medida de la suma de cuadrados entre grupos. Generamos la matriz de distancia entre los puntos.

```
dist(proteinas.dat)
```

```
          1        2        3        4        5        6        7        8        9       10       11       12       13
2  23.176281
3  21.650173  7.868291
4  15.688212 32.304489 32.786125
5  15.154537 10.305338 10.609901 24.005416
6  30.157586 11.956588 11.119802 40.334105 19.420608
7  22.865913 10.742905  8.928606 33.614729 10.613670 15.184532
8  30.990966 17.421251 17.603125 40.335096 24.019992 12.249898 23.827505
9  23.174124 11.010904  6.007495 33.263494 13.435029 12.718490 13.855324 18.181309
10 12.136309 19.529721 18.254862 19.315538 15.025645 24.466099 22.109274 24.107053 18.254588
11 13.157127 16.974982 18.784036 18.399185  9.179869 26.734809 17.517991 29.983829 21.256293 14.933185
12 27.902509 10.039422  9.146584 38.359745 17.582093  8.938121 16.177763 11.569356 10.153325 22.976510 25.019193
13 10.624500 14.688771 13.568714 21.013329  8.705171 21.595833 15.622740 23.755420 15.157836  7.976841 10.702803 20.037215
14 28.302120  6.763875  9.675743 38.526744 16.352676  8.360024 13.268760 14.677534 12.350304 24.062834 23.208619  6.827884 19.872343
15 26.805410 13.684663 10.803703 38.174730 18.727253  6.688049 14.985326 11.688028 13.150285 21.419617 25.669437 10.883014 18.766726
16 17.643979  9.942334 12.201639 24.493877  8.255301 17.813759 14.874811 19.358977 14.013208 12.358803 11.992498 16.008748  9.066973
17 23.111036 22.931201 19.200781 33.293843 19.060430 23.926972 15.156187 31.176273 21.854061 22.156489 22.031568 27.137612 17.866169
18 10.319399 25.261235 25.878563  8.334867 17.311557 33.292191 26.731629 33.398054 27.116969 13.341664 11.638728 31.295687 14.311184
19 17.149927 17.440183 13.916537 28.887367 13.070578 21.187732 11.784312 26.571413 17.215110 16.268682 16.165704 21.806192 10.910545
20 29.987998 13.032268 11.632283 41.480598 20.430370  4.795832 15.584287 11.908820 14.027117 25.157305 27.655198  8.884256 21.877386
```

Le vamos a asignar el nombre *dist.proteinas* para facilitar la escritura.

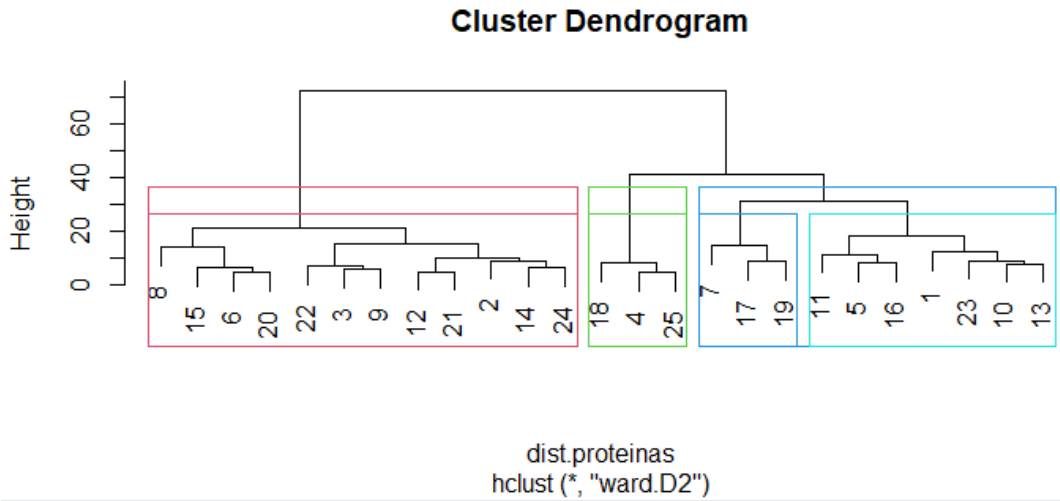
Aplicamos el método Ward:

```
> hclust(dist.proteinas, method = "ward")
The "ward" method has been renamed to "ward.D"; note new "ward.D2"

Call:
hclust(d = dist.proteinas, method = "ward")

Cluster method   : ward.D
Distance         : euclidean
Number of objects: 25
```

Usamos la función *plot* para ver el Cluster Dendrogram y añadimos *rect.hclust* para agrupar:



b) Contestar, examinando el historial de agrupamientos, a las siguientes preguntas: ¿qué dos países se combinan primero? ¿En qué consiste la segunda etapa? ¿Cuándo es la primera vez que se forma un agrupamiento con más de dos países?

```
> proteinas.hcl$merge
      [,1] [,2]
 [1,]   -6  -20
 [2,]   -4  -25
 [3,]  -12  -21
 [4,]   -3   -9
 [5,]  -14  -24
 [6,]  -15    1
 [7,]  -22    4
 [8,]  -10  -13
 [9,]   -5  -16
[10,]  -18    2
[11,]  -17  -19
[12,]  -23    8
[13,]   -2    5
[14,]    3   13
[15,]  -11    9
[16,]   -1   12
[17,]   -8    6
[18,]   -7   11
[19,]    7   14
[20,]   15   16
[21,]   17   19
[22,]   18   20
[23,]   10   22
[24,]   21   23
```

Viendo el historial podemos concluir que los primeros países en combinar son Dinamarca y Suecia.

En la segunda etapa se agrupan Bulgaria y Yugoslavia.

La primera vez que se forma agrupamiento con más de dos países es en la etapa 6, donde se agrupan los países de la primera etapa con Noruega.

c) Examinar el dendograma: si queremos quedarnos con tres grupos, realizar la lista de los países que pertenecen a cada grupo. ¿Y con 4 grupos?

Para agrupar en 3 grupos utilizamos la función

```
groups.proteinas <- cutree(proteinas.hcl, k = 3)
```

Y para visualizar como tabla hacemos:

```
> table(groups.proteinas)
groups.proteinas
 1  2  3
10 12  3
```

Analizando estos datos concluimos que los grupos son:

1-Albania, Checoslovaquia, Alemania este, Grecia, Hungría, Italia, Polonia, Portugal, España y Rusia

2-Austria, Bélgica, Dinamarca, Finlandia, Francia, Irlanda, Holanda, Noruega, Suecia, Suiza, Reino Unido y Alemania oeste

3-Bulgaria, Rumanía y Yugoslavia

Para agrupar en 4 grupos hacemos el mismo proceso pero con

```
groups.proteinas <- cutree(proteinas.hcl, k = 4)
```

```
> table(groups.proteinas)
groups.proteinas
 1  2  3  4
 7 12  3  3
```

Analizando estos datos concluimos que los grupos son:

1-Albania, Checoslovaquia, Grecia, Hungría, Italia, Polonia y Reno Unido

2-Austria, Bélgica, Dinamarca, Finlandia, Francia, Irlanda, Holanda, Noruega, Suecia, Suiza, Reino Unido y Alemania oeste

3- Bulgaria, Rumanía y Yugoslavia

4- Alemania este, Portugal y España

d) Realizar el análisis en componentes principales. Guardar las puntuaciones de los países según la primera componente principal. Ordenar los países por orden creciente de estas puntuaciones. ¿El orden obtenido parece guardar relación con los grupos obtenidos en el apartado anterior? ¿Cómo podemos explicar esta relación?

Primero hacemos una función log para las variables numéricas y llamarlo log.proteinas y llamamos proteínas.pais a lo que definimos como la primera columna que es el país. Aplicamos PCA y lo imprimimos para visualizarlo:

```
> log.proteinas <- log(proteinas[,2:10])
> proteinas.pais <- proteinas[,1]
> proteinas.pca <- prcomp(log.proteinas, center=TRUE, scale=TRUE)
> print(proteinas.pca)
Standard deviations (1, .., p=9):
[1] 2.0999405 1.2549159 1.0452015 0.9126398 0.6427739 0.5281032 0.4699472 0.3461238 0.2395312

Rotation (n x k) = (9 x 9):
                 PC1         PC2         PC3        PC4         PC5          PC6         PC7         PC8         PC9
RedMeat    -0.28452803 -0.24722544  0.4783032  0.4911466  0.18016949 -0.3092580180  0.50011954 -0.02084068 -0.09627561
WhiteMeat  -0.31897429  0.14770203 -0.6104496  0.2843149  0.21606770  0.0513029656  0.03954914  0.11551556 -0.59894120
Eggs       -0.41562972  0.16177881 -0.0868124  0.3107628 -0.27158881 -0.0006063034 -0.26302204 -0.70236950  0.24638441
Milk       -0.36289298 -0.35106599  0.0539184  0.1930579 -0.53684243  0.3451713060 -0.18036894  0.51056060  0.06583851
Fish       -0.30456049  0.28895569  0.4884444 -0.3891523 -0.02040385  0.4507216335  0.07834324 -0.16343022 -0.44420347
Cereals     0.40435571  0.03672802 -0.1970058  0.1832745 -0.42279429  0.3797407693  0.62774209 -0.21050330 -0.03764069
Starch     -0.32935377  0.36957475 -0.1477358 -0.3637824 -0.40392653 -0.5048767728  0.35826218  0.20115429  0.11791985
Nuts        0.38263129  0.21474682  0.2570519  0.2633568 -0.42718835 -0.3617162866 -0.34343289  0.03009093 -0.48975956
Frveg       0.04145582  0.70717604  0.1511390  0.3967010  0.18459567  0.2224671109 -0.01034567  0.34617508  0.33831552
```

Para calcular los las puntuaciones según la primera componente principal hacemos:

```
> puntuaciones <- proteinas.pca$x[,1]
> puntuaciones
 [1]  5.1955201 -1.2600938 -1.5399040  3.0986937 -0.5360529 -2.6415316 -1.5982402 -1.5922517 -1.5106595  1.4325301  1.8528830
[12] -2.1644979  1.1121024 -1.5397701 -1.2136639 -0.2685462  1.8305120  2.5356800  0.9470978 -1.8994155 -0.9274556 -1.4570275
[23]  0.6482326 -2.1254225  3.6212811
```

Y con la función *order* ordenamos por país

```
> proteinas$Country[order(puntuaciones)]
 [1] "Denmark"       "Ireland"       "WGermany"      "Sweden"        "EGermany"      "Finland"       "Belgium"
 [8] "Netherlands"   "France"        "UK"            "Austria"       "Norway"        "Switzerland"   "Czechoslovakia"
[15] "Poland"        "USSR"          "Spain"         "Italy"         "Greece"        "Portugal"      "Hungary"
[22] "Romania"       "Bulgaria"      "Yugoslavia"    "Albania"
```

Claramente hay una relación con los grupos obtenidos previamente, ya que los países que aparecen aquí juntos en su mayoría también estaban agrupados juntos.