```r
fichero = read.csv("distancia_universitarios.csv")
fichero
```

```
##    Distancia
## 1       16.5
## 2       34.8
## 3       20.7
## 4        6.2
## 5        4.4
## 6        3.4
## 7       24.0
## 8       24.0
## 9       32.0
## 10      30.0
## 11      33.0
## 12      27.0
## 13      15.0
## 14       9.4
## 15       2.1
## 16      34.0
## 17      24.0
## 18      12.0
## 19       4.4
## 20      28.0
## 21      31.4
## 22      21.6
## 23       3.1
## 24       4.5
## 25       5.1
## 26       4.0
## 27       3.2
## 28      25.0
## 29       4.5
## 30      20.0
## 31      34.0
## 32      12.0
## 33      12.0
## 34      12.0
## 35      12.0
## 36       5.0
## 37      19.0
## 38      30.0
## 39       5.5
## 40      38.0
## 41      25.0
```

```
## 42       3.7
## 43       9.0
## 44      30.0
## 45      13.0
## 46      30.0
## 47      30.0
## 48      26.0
## 49      30.0
## 50      30.0
## 51       1.0
## 52      26.0
## 53      22.0
## 54      10.0
## 55       9.7
## 56      11.0
## 57      24.1
## 58      33.0
## 59      17.2
## 60      27.0
## 61      24.0
## 62      27.0
## 63      21.0
## 64      28.0
## 65      30.0
## 66       4.0
## 67      46.0
## 68      29.0
## 69       3.7
## 70       2.7
## 71       8.1
## 72      19.0
## 73      16.0
```

```
len = function(list){
        count = 0
        for (element in list){
                count = count + 1
        }
        count
}

distancias = fichero$Distancia

longitud = len(distancias)
longitud
```

```
## [1] 73
```

```
bubble = function(list, asc = TRUE){
        n = len(list)
        if(asc){
                for (i in 2:n){
                        for (j in 1:(n-1)){
                                if (list[j] > list[j+1]){
                                        temp = list[j]
                                        list[j] = list[j+1]
                                        list[j+1] = temp
                                }
                        }
                }
        }
        else {
                for (i in 2:n){
                        for (j in 1:(n-1)){
                                if (list[j] < list[j+1]){
                                        temp = list[j]
                                        list[j] = list[j+1]
                                        list[j+1] = temp
                                }
                        }
                }
        }
        list
}
distanciasordenadas = bubble(distancias, FALSE)
distanciasordenadas
```

```
##  [1] 46.0 38.0 34.8 34.0 34.0 33.0 33.0 32.0 31.4 30.0 30.0 30.0 30.0 30.0 30.0
## [16] 30.0 30.0 29.0 28.0 28.0 27.0 27.0 27.0 26.0 26.0 25.0 25.0 24.1 24.0 24.0
## [31] 24.0 24.0 22.0 21.6 21.0 20.7 20.0 19.0 19.0 17.2 16.5 16.0 15.0 13.0 12.0
## [46] 12.0 12.0 12.0 12.0 11.0 10.0  9.7  9.4  9.0  8.1  6.2  5.5  5.1  5.0  4.5
## [61]  4.5  4.4  4.4  4.0  4.0  3.7  3.7  3.4  3.2  3.1  2.7  2.1  1.0
```

```
rank = function(list){
        ordered_list = bubble(list)
        ordered_list[len(ordered_list)] - ordered_list[1]
}

rango = rank(distanciasordenadas)
rango
```

```
## [1] 45
```

```
absolute_freq = function(list){
        ordered_list = bubble(list)
        n = len(ordered_list)
        elements = vector()
        frequencies = vector()
        i = 1
        while (i <= n){
                actual_element = ordered_list[i]
                elements = append(elements, actual_element)
                actual_freq = 0
                j = i
                while(j <= n & actual_element == ordered_list[j]){
                        actual_freq = actual_freq + 1
                        j = j+1
                }
                frequencies = append(frequencies, actual_freq)
                i = j
        }
        rbind(elements, frequencies)
}
```

PARTE 2

```
for (i in 2:kMax) { #itera dimensiones k

for (j in 1:len(split)) { #itera elementos de dim k

for (k in 1:(i-1)) { #itera posibilidades que haya en el lado izquierdo; -1 es para qu

## Error: <text>:8:0: unexpected end of input
## 6: for (k in 1:(i-1)) { #itera posibilidades que haya en el lado izquierdo;
-1 es para quitar los conjuntos vacios; numero de elementos que hay a la izq de
la implic
## 7:
##   ^
```

```
len = function(list){
count = 0
for (element in list){
count = count + 1
}
count
}
```

```
union = function(c1, c2){
if (len(c1) == 0){
c2
}
else if (is.element(c1[1], c2)){
union(c1[-1], c2)
}
else{
union(c1[-1], append(c2, c1[1]))
}
}

intersect = function(c1, c2){
if (len(c1) == 0){
c()
}
else if (is.element(c1[1], c2)){
append(intersect(c1[-1], c2), c1[1])
}
else{
intersect(c1[-1], c2)
}
}

dif = function(c1, c2) {
res = c()
for (element in c1) {
if (!(element %in% c2)) {
res = append(res, element)
}
}
res
}

tabla <- matrix(c(1,1,0,1,1, 1,1,1,1,0, 1,1,0,1,0, 1,0,1,1,0, 1,1,0,0,0, 0,0,0,1,0),6,5,

count_appearance = function(table, elements){
count = 0
for (i in 1:len(table[,1])){
acum = 1
for (element in elements){
acum = (table[i,element]) & acum
}
count = count + acum
```

```
}
count
}

support = function(table, elements) {
count_appearance(table, elements) / len(table[,1])
}

support_clasif = function(table, ocurrences, s){
valid_ocurrences = c()
for (ocurrence in ocurrences){
support_oc = support(table, ocurrence)
if (support_oc >= s){
valid_ocurrences = append(valid_ocurrences, ocurrence)
}
}
valid_ocurrences
}

create_comb = function(table, clasif, s) {
lista = c()
dim = 2
next_dim = TRUE
while (dim <= len(clasif) & next_dim == TRUE) {
next_dim = FALSE
comb = unlist(lapply(dim, function(m) {combn(clasif, m=m, simplify=TRUE)}), recursive=FA

for (j in seq(1, len(comb), by=dim)) {
add = c()
for (k in j:(j+dim-1)) {
add = append(add, comb[k])
}
if (support(table, add) >= s) {
next_dim = TRUE
lista = append(lista, list(add))
}
}
dim = dim+1
}
lista
}

confidence = function(table, left, right) {
count_appearance(table, union(left, right)) / count_appearance(table, left)
```

```
}

get_asotiations = function(table, comb, c) {
kMax = len(comb[len(comb)][[1]])
listLeft = list()
listRight = list()

for (i in 2:kMax) {

split = Filter(function(x) length(x)==i, comb)

for (j in 1:len(split)) {

for (k in 1:(i-1)) {

leftSides = lapply(k, function(m) {combn(split[j][[1]], m=m, simplify=TRUE)})

df = do.call(rbind.data.frame, leftSides)

for (n in 1:len(df[1,])) {

all = split[j][[1]]

left = df[,n]

right = dif(split[j][[1]], df[,n])

if (confidence(table, left, right) >= c) {

listLeft = append(listLeft , list(left))
listRight = append(listRight , list(right))

} else {
print("Mejora posible")
}
}
}
}
}
asoc = data.frame(left = I(listLeft), right = I(listRight))
asoc
}

apriori = function(table, s, c) {
```

```
soporte_clasif = support_clasif(tabla, c(c("P"), c("A") ,c("L"), c("C"), c("N")), s)

combinaciones = create_comb(tabla, soporte_clasif, s)

conf = get_asotiations(tabla, combinaciones, c)

print(conf)

}

apriori(tabla, 0.5, 0.8)

## [1] "Mejora posible"
## [1] "Mejora posible"
## [1] "Mejora posible"
## [1] "Mejora posible"
## [1] "Mejora posible"
## [1] "Mejora posible"
## [1] "Mejora posible"
##   left right
## 1    P     A
## 2    A     P
## 3    P     L
## 4    L     P
## 5 A, L     P
```

40 de soporte y 90 de confianza