Práctica 4

Ejercicio 1. Construye las siguientes tablas de contingencia:

(a) Hay varios algoritmos que permiten ordenar una lista. En el fichero 'tiempos.csv' se encuentran datos sobre el tiempo que han tardado tres algoritmos en ordenar diferentes listas de números. Para cada experimento se ha almacenado el algoritmo usado (entre 0 y 2), el máximo número de veces que se ha repetido algún elemento de la lista, la longitud de la lista y el tiempo necesitado para ordenarla. Se pide:

- Crea una nueva variable llamada "Categoria" que parta los tiempos en tres categorías: "Rapido","Normal" y "Lento". En la categoría 'Rapido' se deben incluir el 25 % de los tiempos más rápidos, en "Lento" el 25 % de los tiempos más lentos y en "Normal" las demás.
- Añade esta variable al data.frame.
- Construye las tablas de contingencia de "Categoria" respecto a "Algoritmo" para las frecuencias conjuntas absolutas y relativas.

```
datos=read.csv("tiempos.csv")
attach(datos)
str(datos)

## 'data.frame': 999 obs. of  4 variables:
##  $ Algoritmo       : int  1 0 2 1 0 2 0 1 2 1 ...
##  $ Max_repeticiones: int  5 5 6 5 4 6 7 6 6 6 ...
##  $ Longitud        : int  1022 1529 1495 1059 1153 1656 1771 1646 1709 1
##  $ Tiempo          : num  0.00327 0.16185 0.06938 0.00349 0.08813 ...

summary(datos)

##    Algoritmo     Max_repeticiones    Longitud        Tiempo
##  Min.   :0.000   Min.   :3.000    Min.   :1000   Min.   :0.003205
##  1st Qu.:0.000   1st Qu.:4.000    1st Qu.:1200   1st Qu.:0.005487
##  Median :1.000   Median :5.000    Median :1413   Median :0.058914
##  Mean   :1.016   Mean   :4.953    Mean   :1405   Mean   :0.066984
##  3rd Qu.:2.000   3rd Qu.:5.000    3rd Qu.:1609   3rd Qu.:0.096247
##  Max.   :2.000   Max.   :8.000    Max.   :1800   Max.   :0.221847

Q1<-as.numeric(quantile(Tiempo,0.25))
Q3<-as.numeric(quantile(Tiempo,0.75))

Cat<-cut(Tiempo,breaks=c(0,Q1,Q3,1),labels<-c("Rapido","Normal","Lento"), i
nuevosdatos<-data.frame(datos,Cat)
```

OTRO MODO:

```
Categoria<-{}
for(i in 1:length(Tiempo)){
  if(Tiempo[i]<Q1){Categoria[i]<-"Lento"}
  if(Tiempo[i]>Q3){Categoria[i]<-"Rapido"}
```

```
  if(Q1<=Tiempo[i] & Tiempo[i]<=Q3){Categoria[i]<-"Normal"}
}
nuevosdatos<-data.frame(datos,Categoria)
str(nuevosdatos)

## 'data.frame': 999 obs. of  5 variables:
##  $ Algoritmo       : int  1 0 2 1 0 2 0 1 2 1 ...
##  $ Max_repeticiones: int  5 5 6 5 4 6 7 6 6 6 ...
##  $ Longitud        : int  1022 1529 1495 1059 1153 1656 1771 1646 1709 1
##  $ Tiempo          : num  0.00327 0.16185 0.06938 0.00349 0.08813 ...
##  $ Categoria       : chr  "Lento" "Rapido" "Normal" "Lento" ...
```

Construimos ahora las tablas de contingencia:

```
n_ij <- table(Categoria,Algoritmo)
n_ij

##          Algoritmo
## Categoria   0   1   2
##    Lento    0 250   0
##    Normal  83  85 331
##    Rapido 241   0   9

f_ij<-prop.table(n_ij) # frecuencias relativas conjuntas (f_ij) de Genero
f_ij # tambien se obtiene como:

##          Algoritmo
## Categoria           0           1           2
##    Lento  0.000000000 0.250250250 0.000000000
##    Normal 0.083083083 0.085085085 0.331331331
##    Rapido 0.241241241 0.000000000 0.009009009

f_ij<-n_ij/sum(n_ij)
f_ij

##          Algoritmo
## Categoria           0           1           2
##    Lento  0.000000000 0.250250250 0.000000000
##    Normal 0.083083083 0.085085085 0.331331331
##    Rapido 0.241241241 0.000000000 0.009009009

N_ij<-addmargins(n_ij)
N_ij

##          Algoritmo
## Categoria   0   1   2 Sum
##    Lento    0 250   0 250
##    Normal  83  85 331 499
##    Rapido 241   0   9 250
##    Sum    324 335 340 999
```

```
F_ij<-addmargins(f_ij)
F_ij

##           Algoritmo
## Categoria           0           1           2         Sum
##    Lento  0.000000000 0.250250250 0.000000000 0.250250250
##    Normal 0.083083083 0.085085085 0.331331331 0.499499499
##    Rapido 0.241241241 0.000000000 0.009009009 0.250250250
##    Sum    0.324324324 0.335335335 0.340340340 1.000000000
```

(b) Construye la siguiente tabla de contingencia:

	Castaño	Moreno	Rubio
Hombre	12	8	5
Mujer	20	10	25

Leemos las casillas de la tabla de izquierda a derecha y de arriba abajo indicando el valor de la fila, columna y la frecuencia correspondiente:

```
#       Primera casilla: Hombre Castaño 12
#       Segunda casilla: Hombre Moreno 8
#       Tercera casilla: Hombre Rubio 5
#       Cuarta casilla:  Mujer Castaño 20
#       Quinta casilla:  Mujer Moreno 10
#       Sexta casilla:   Mujer Rubio 25
```

Creamos tres variables para almacenar los nombres de fila, columna y datos de cada casilla:

```
gen <- c(rep("Hombre",3),rep("Mujer",3))
gen

## [1] "Hombre" "Hombre" "Hombre" "Mujer"  "Mujer"  "Mujer"

# Esto representa la columna de observaciones del género
# que hemos puesto arriba.

pelo<-rep(c("Castaño","Moreno","Rubio"),2)
pelo

## [1] "Castaño" "Moreno"  "Rubio"   "Castaño" "Moreno"  "Rubio"

# Esto representa la columna de observaciones del pelo
# que hemos puesto arriba.

frec <- c(12,8,5,20,10,25)
frec

## [1] 12  8  5 20 10 25
```

```
# Aquí ponemos la columna de frecuencias absolutas observadas

xtabs(frec~gen+pelo)

##          pelo
## gen       Castaño Moreno Rubio
##   Hombre      12      8     5
##   Mujer       20     10    25

# Esta orden construye la tabla de contingencia
```

Tambien podría hacerse como sigue:

```
 genero<-rep(gen,frec)
 genero

##  [1] "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hom
##  [9] "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hom
## [17] "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hombre" "Hom
## [25] "Hombre" "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Muj
## [33] "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Muj
## [41] "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Muj
## [49] "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Muj
## [57] "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Muj
## [65] "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Muj
## [73] "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Mujer"  "Muj

 color.pelo<-rep(pelo,frec)
 color.pelo

##  [1] "Castaño" "Castaño" "Castaño" "Castaño" "Castaño" "Castaño" "Castañ
##  [8] "Castaño" "Castaño" "Castaño" "Castaño" "Castaño" "Moreno"  "Moreno
## [15] "Moreno"  "Moreno"  "Moreno"  "Moreno"  "Moreno"  "Moreno"  "Rubio"
## [22] "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Castaño" "Castaño" "Castañ
## [29] "Castaño" "Castaño" "Castaño" "Castaño" "Castaño" "Castaño" "Castañ
## [36] "Castaño" "Castaño" "Castaño" "Castaño" "Castaño" "Castaño" "Castañ
## [43] "Castaño" "Castaño" "Castaño" "Moreno"  "Moreno"  "Moreno"  "Moreno
## [50] "Moreno"  "Moreno"  "Moreno"  "Moreno"  "Moreno"  "Moreno"  "Rubio"
## [57] "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"
## [64] "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"
## [71] "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"   "Rubio"
## [78] "Rubio"   "Rubio"   "Rubio"

 table(genero,color.pelo)

##          color.pelo
## genero    Castaño Moreno Rubio
##   Hombre      12      8     5
##   Mujer       20     10    25
```

Otra forma de hacer dicha tabla es hacer primero un data.frame con las variables genero y color.pelo:

```
datos<-data.frame(genero,color.pelo)
table(datos) # Tabla de contingencia genero/color.pelo
```

```
##          color.pelo
## genero    Castaño Moreno Rubio
##    Hombre      12      8     5
##    Mujer       20     10    25
```

Ejercicio 2 Vamos a usar la tabla de contingencia creada en el ejercicio 1, apartado (a).

(a) Dibuja el diagrama de barras de esa tabla.

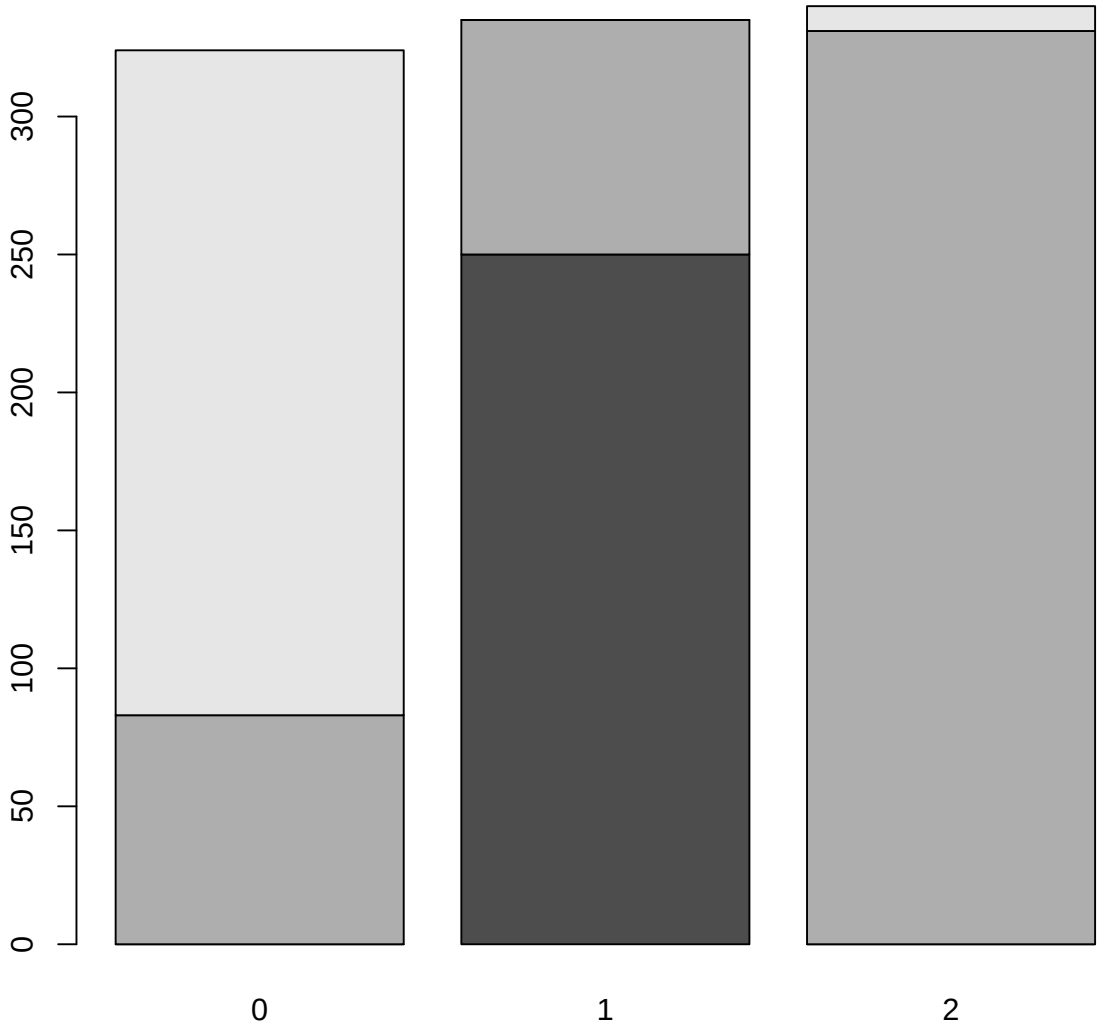
Recordemos que habíamos almacenado esa tabla de contingencia como n_ij:

```
n_ij
```

```
##           Algoritmo
## Categoria   0   1   2
##    Lento    0 250   0
##    Normal  83  85 331
##    Rapido 241   0   9
```

Ahora dibujamos el diagrama de barras de la tabla de contingencia:

```
barplot(n_ij)
```



Obsérvese que en el dibujo aparecen tres barras, una por algoritmo. En cada barra la altura representa

el número total de veces que se ha usado el algoritmo asociado a la barra, y se usan tonos de gris distintos para las categorías en las que su ejecución se ha colocado (lento, rápido, normal)

(b) Construye la tabla de frecuencias marginales.

```
rowSums(n_ij) # tabla de frecuencias marginales absolutas (ni_) (Las de X)

##  Lento Normal Rapido
##    250    499    250

colSums(n_ij) # tabla de frecuencias marginales absolutas (n_j) (Las de Y)

##   0   1   2
## 324 335 340
```

(c) Construye las tablas de frecuencias condicionadas.

```
CondColumnas<-prop.table(n_ij,2)
# tabla de frecuencias condicionadas de
# Categoría condicionada a Algoritmo:

CondColumnas

##          Algoritmo
## Categoria          0          1          2
##    Lento  0.00000000 0.74626866 0.00000000
##    Normal 0.25617284 0.25373134 0.97352941
##    Rapido 0.74382716 0.00000000 0.02647059

CondFilas<-prop.table(n_ij,1)
# tabla de frecuencias condicionadas de
# Algoritmo condicionada a Categoría:
CondFilas

##          Algoritmo
## Categoria         0         1         2
##    Lento  0.0000000 1.0000000 0.0000000
##    Normal 0.1663327 0.1703407 0.6633267
##    Rapido 0.9640000 0.0000000 0.0360000
```

(d) Contesta a las siguientes preguntas indicando en qué tabla aparece cada dato pedido:

- Número y proporción de veces que se ha ejecutado cada algoritmo.

Recordemos la tabla de frecuencias absolutas con marginales incluidas:

```
N_ij
```

```
##           Algoritmo
## Categoria   0   1   2 Sum
##    Lento    0 250   0 250
##    Normal  83  85 331 499
##    Rapido 241   0   9 250
##    Sum    324 335 340 999
```

```
# El algoritmo 0 se ha ejecutado:
N_ij[4,1] # veces
```

```
## [1] 324
```

```
#Proporción (en tanto por ciento):
F_ij[4,1]*100
```

```
## [1] 32.43243
```

```
# Análogamente, con el algoritmo 1:
# Se ha ejecutado:
N_ij[4,2] # veces
```

```
## [1] 335
```

```
#Proporción (en tanto por ciento):
F_ij[4,2]*100
```

```
## [1] 33.53353
```

Análogamente, con el algoritmo 2:

```
# Se ha ejecutado:
N_ij[4,3] # veces
```

```
## [1] 340
```

```
# Proporción (en tanto por ciento):
F_ij[4,3]*100
```

```
## [1] 34.03403
```

- Número y proporción de tiempos en cada categoría.

Recordemos de nuevo la tabla de frecuencias absolutas con marginales incluidas:

```
N_ij
```

```
##           Algoritmo
## Categoria   0   1   2 Sum
##    Lento    0 250   0 250
##    Normal  83  85 331 499
##    Rapido 241   0   9 250
##    Sum    324 335 340 999
```

```
# La categoría Lento se ha obtenido en :
N_ij[1,4] # veces
```

```
## [1] 250
```

```
#Proporción (en tanto por ciento):
F_ij[1,4]*100
```

```
## [1] 25.02503
```

```
# Análogamente, la categoría Normal:
# Se ha obtenido en:
N_ij[2,4] # veces
```

```
## [1] 499
```

```
#Proporción (en tanto por ciento):
F_ij[2,4]*100
```

```
## [1] 49.94995
```

```
# Análogamente, con la categoría Rápido:
# Se ha obtenido en:
N_ij[4,3] # veces
```

```
## [1] 340
```

```
#Proporción (en tanto por ciento):
F_ij[4,3]*100
```

```
## [1] 34.03403
```

- Proporción de tiempos lentos dentro del algoritmo 0.

Recordemos las tablas de frecuencias condicionadas absolutas:

```
CondColumnas
```

```
##          Algoritmo
## Categoria          0          1          2
##    Lento  0.00000000 0.74626866 0.00000000
##    Normal 0.25617284 0.25373134 0.97352941
##    Rapido 0.74382716 0.00000000 0.02647059
```

```
CondFilas
```

```
##          Algoritmo
## Categoria         0         1         2
##    Lento  0.0000000 1.0000000 0.0000000
##    Normal 0.1663327 0.1703407 0.6633267
##    Rapido 0.9640000 0.0000000 0.0360000
```

```
CondColumnas[1,1]
```

```
## [1] 0
```

- Proporción de algoritmo 0 dentro de tiempos lentos.

```
CondFilas[1,1]
```

```
## [1] 0
```

- Proporción de veces que se ha usado cada algoritmo.

```
c(F_ij[4,1],F_ij[4,2],F_ij[4,3])
```

```
## [1] 0.3243243 0.3353353 0.3403403
```

Análogamente:

```
colSums(f_ij)
```

```
##         0         1         2
## 0.3243243 0.3353353 0.3403403
```

- ¿Son independientes las variables 'Categoría' y 'Algoritmo'?

No lo son. Razón: Si sabemos que estamos en la categoría "Lento" entonces el algoritmo es 1 necesariamente:

$$P(\text{Algoritmo } 1|\text{Lento}) = 1$$

Sin embargo

$$P(\text{Algoritmo } 1) = 335/999 < 1.$$

Basta observar la tabla de contingencias:

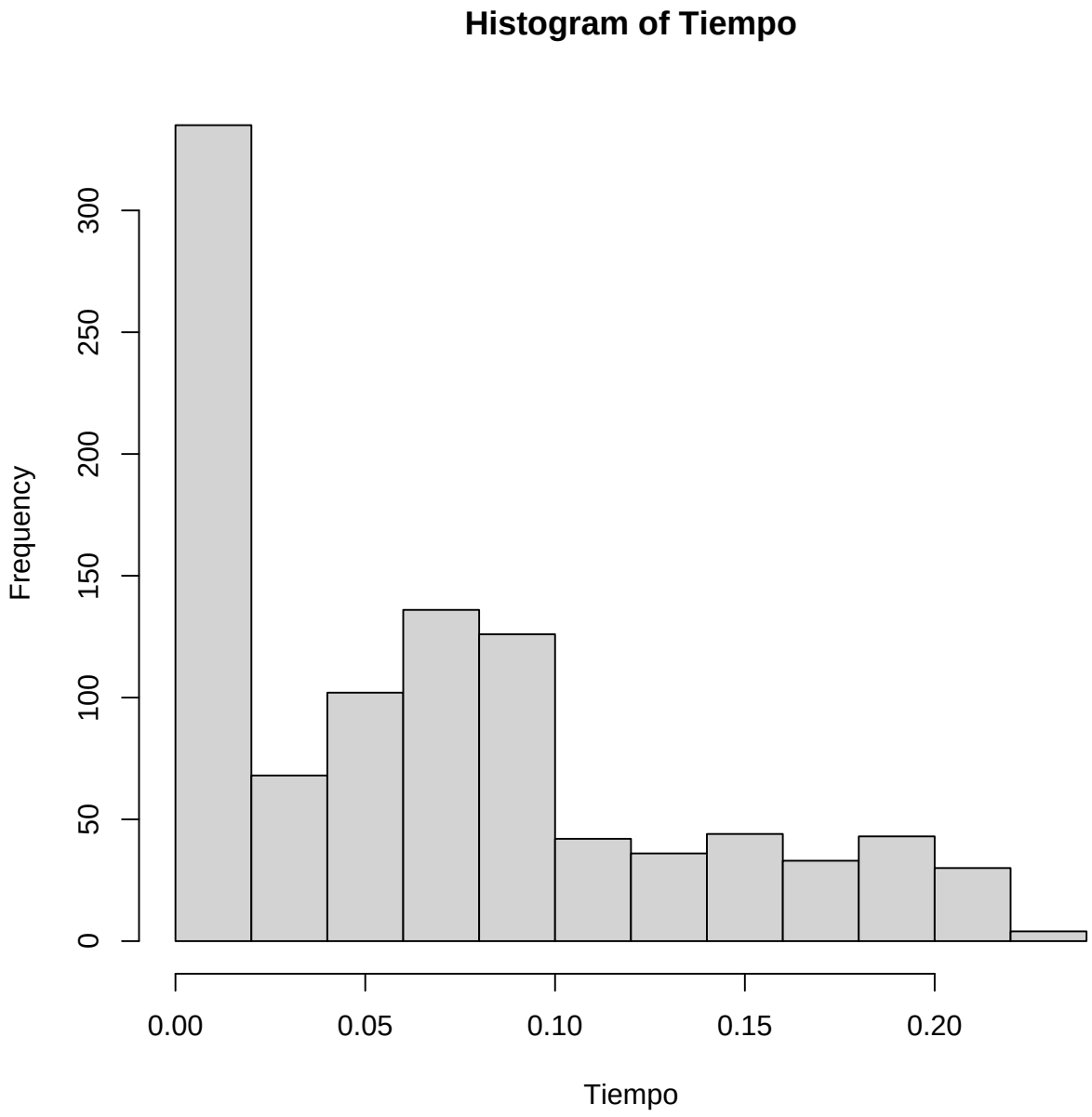
```
addmargins(n_ij)
```

```
##           Algoritmo
## Categoria   0   1   2 Sum
##    Lento    0 250   0 250
##    Normal  83  85 331 499
##    Rapido 241   0   9 250
##    Sum    324 335 340 999
```

Ejercicio 3. Siguiendo con los datos usados en el ejercicio 1 (a), se pide:

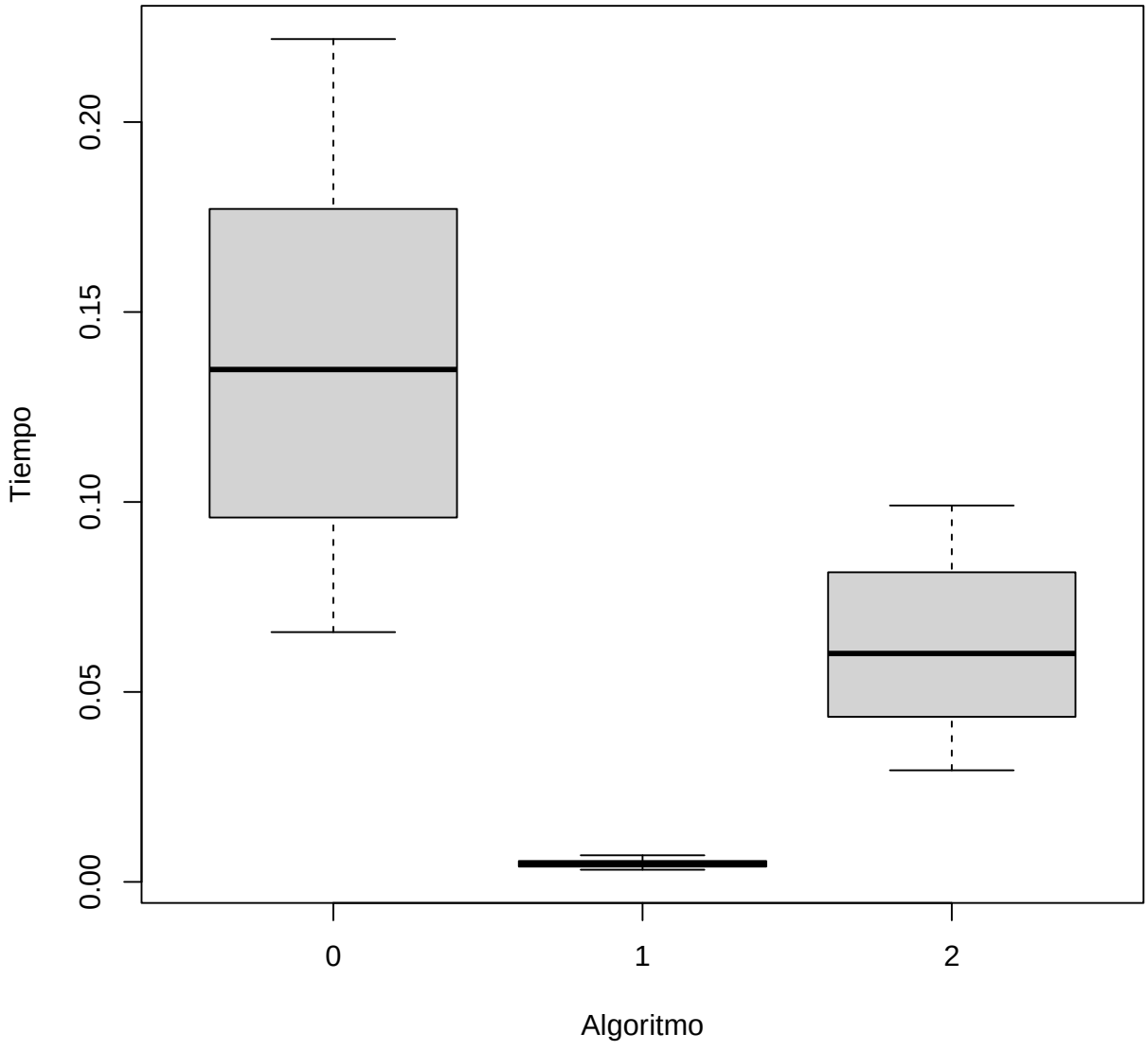
- Haz un sumario de Tiempo y dibuja su histograma.

```
summary(Tiempo)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.003205 0.005487 0.058914 0.066984 0.096247 0.221847

hist(Tiempo)
```



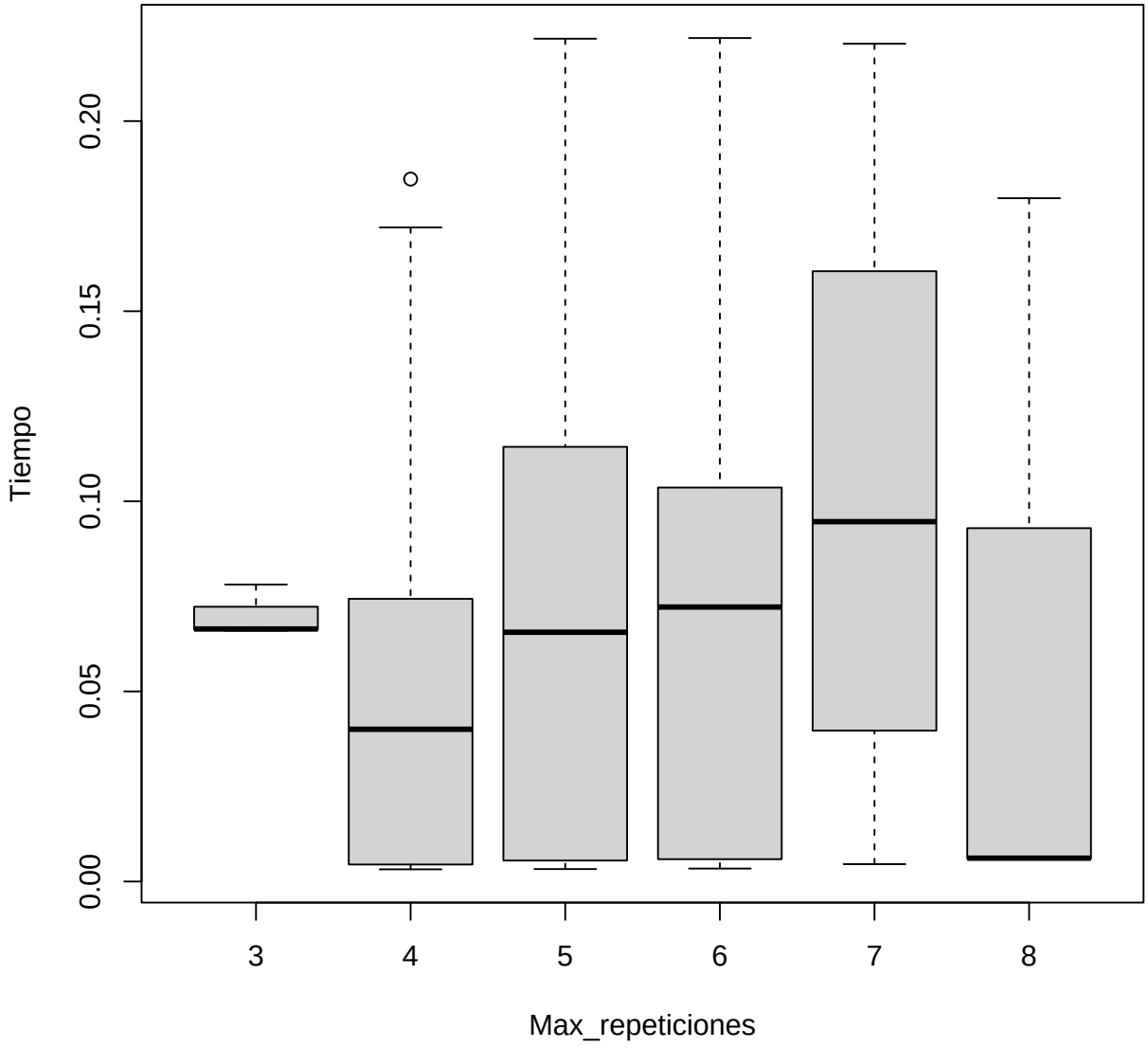
- Haz un sumario, histograma y boxplot de Tiempo dependiendo del algoritmo. ¿Qué conclusiones crees que se pueden sacar?

```
by(Tiempo,Algoritmo,summary)

## Algoritmo: 0
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06575 0.09592 0.13487 0.13672 0.17712 0.22185
## ------------------------------------------------------------
## Algoritmo: 1
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003205 0.004029 0.004789 0.004795 0.005497 0.007004
## ------------------------------------------------------------
## Algoritmo: 2
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02938 0.04349 0.06012 0.06181 0.08146 0.09907

boxplot(Tiempo~Algoritmo)
```



- Haz un sumario, histograma y boxplot de Tiempo dependiendo del máximo número de repeticiones. ¿Qué conclusiones crees que se pueden sacar?

```
by(Tiempo,Max_repeticiones,summary)

## Max_repeticiones: 3
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06598 0.06620 0.06642 0.07017 0.07227 0.07812
## ------------------------------------------------------------
## Max_repeticiones: 4
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003205 0.004474 0.040043 0.048574 0.074357 0.184754
## ------------------------------------------------------------
## Max_repeticiones: 5
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003272 0.005521 0.065562 0.072195 0.114324 0.221673
## ------------------------------------------------------------
## Max_repeticiones: 6
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003408 0.005878 0.072194 0.075050 0.101345 0.221847
## ------------------------------------------------------------
## Max_repeticiones: 7
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.004573 0.039691 0.094645 0.099310 0.160518 0.220347
## ------------------------------------------------------------
## Max_repeticiones: 8
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.005903 0.006022 0.006141 0.063924 0.092935 0.179729

boxplot(Tiempo~Max_repeticiones)
```

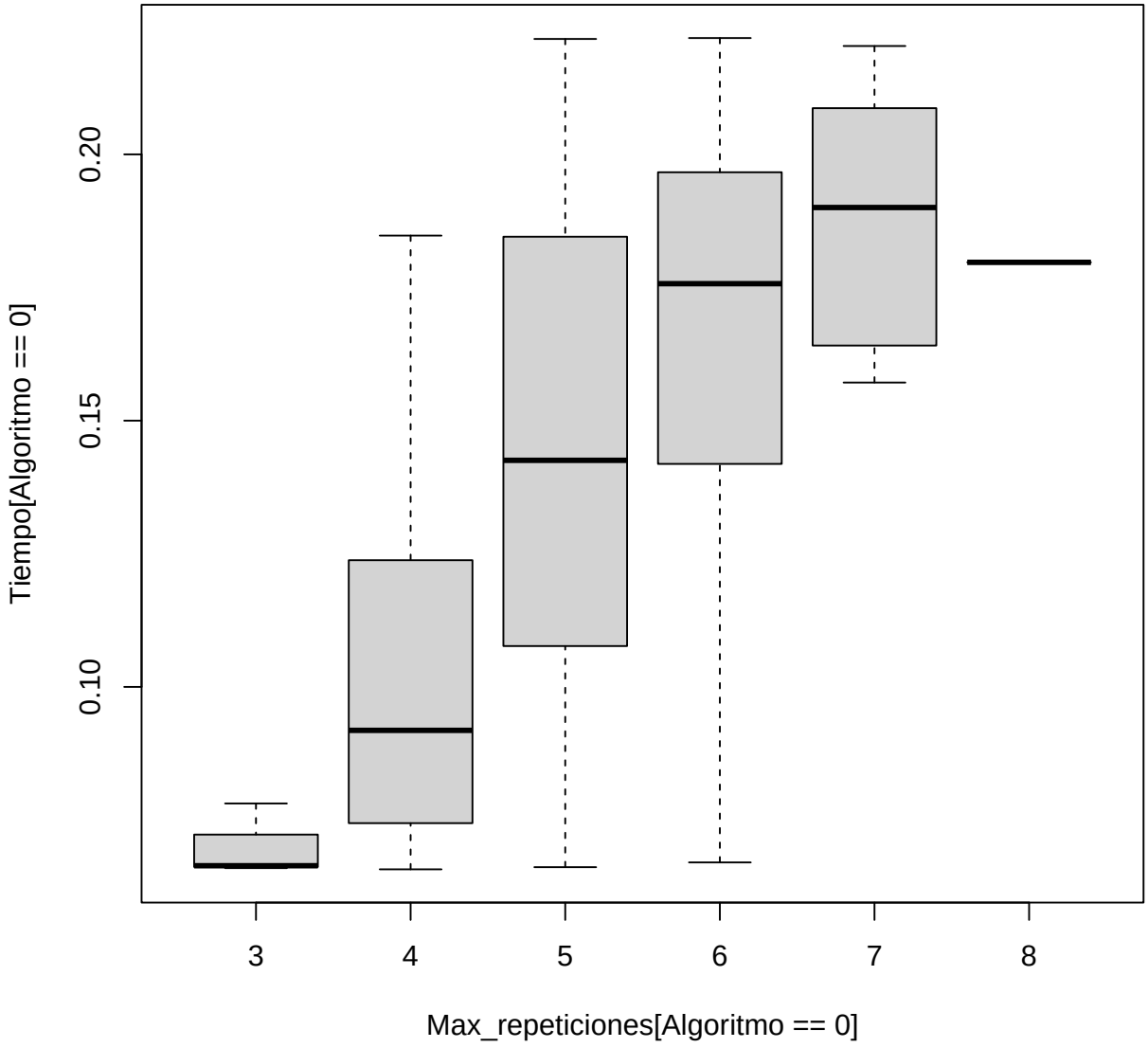


Aparentemente, no hay un patrón claro que relacione ambas variables cuando tomamos todos los algoritmos a la vez.

- Repite el apartado anterior para cada algoritmo por separado.

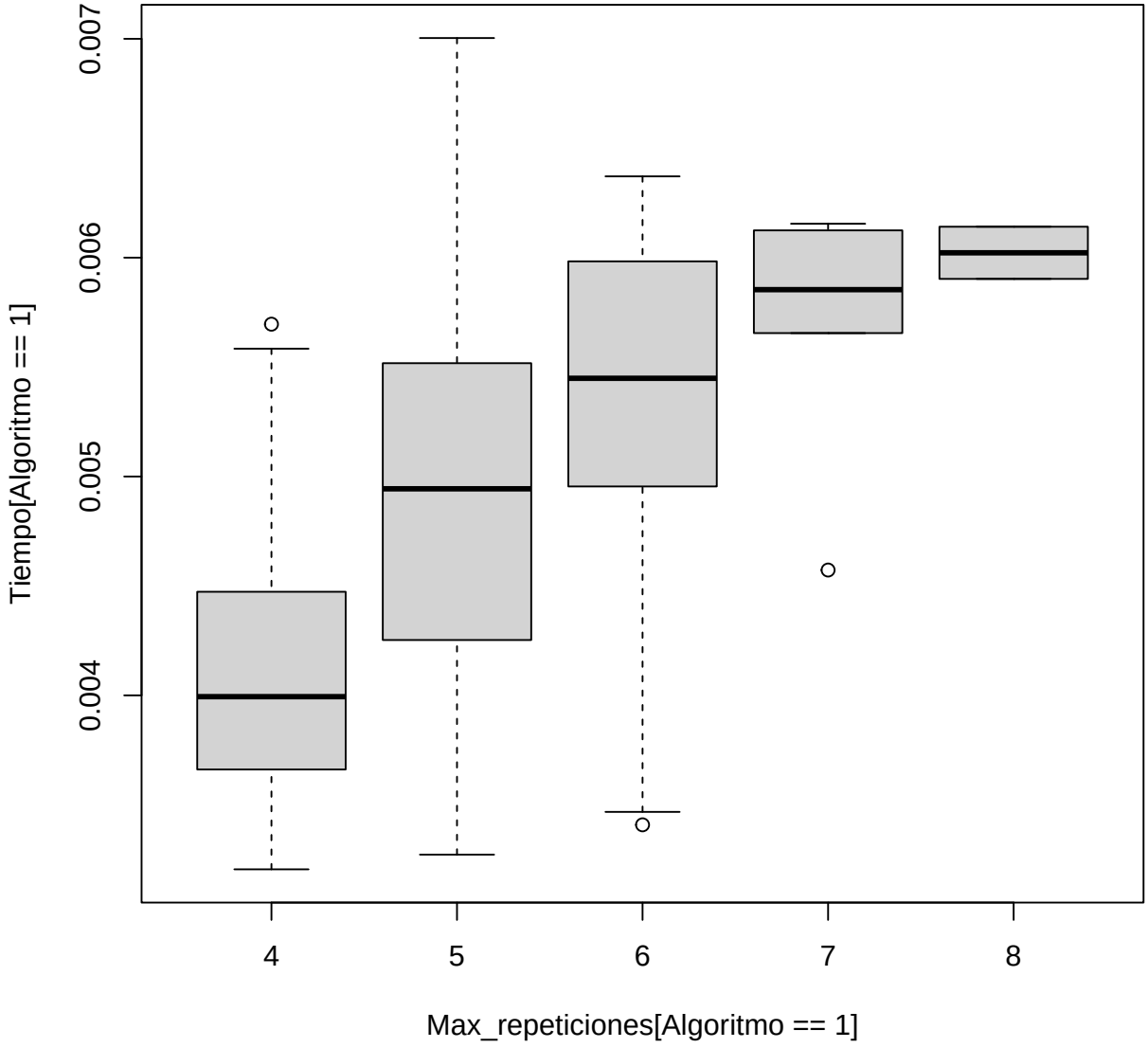
Comenzamos con el Algoritmo 0:

```
by(Tiempo[Algoritmo==0],Max_repeticiones[Algoritmo==0],summary)

## Max_repeticiones[Algoritmo == 0]: 3
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06598 0.06620 0.06642 0.07017 0.07227 0.07812
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 0]: 4
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06575 0.07444 0.09184 0.10074 0.12357 0.18475
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 0]: 5
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06615 0.10838 0.14255 0.14485 0.18449 0.22167
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 0]: 6
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06704 0.14188 0.17572 0.16527 0.19665 0.22185
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 0]: 7
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1572  0.1641  0.1900  0.1876  0.2087  0.2203
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 0]: 8
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1797  0.1797  0.1797  0.1797  0.1797  0.1797

boxplot(Tiempo[Algoritmo==0]~Max_repeticiones[Algoritmo==0])
```


Tiempo[Algoritmo == 0]
0.10
0.15
0.20
3
4
5
6
7
8
Max_repeticiones[Algoritmo == 0]

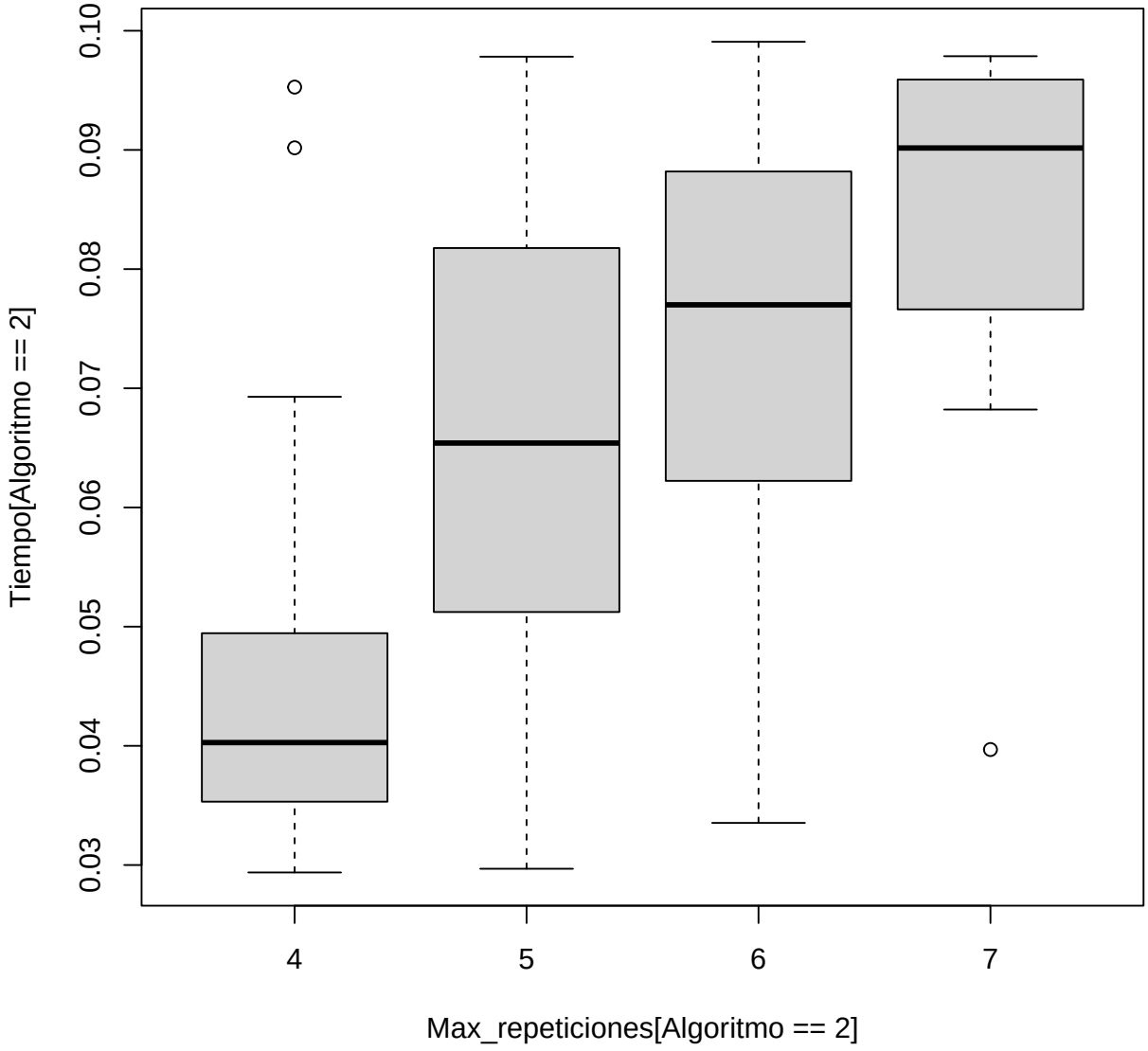
Ahora el Algoritmo 1:

```
by(Tiempo[Algoritmo==1],Max_repeticiones[Algoritmo==1],summary)

## Max_repeticiones[Algoritmo == 1]: 4
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003205 0.003669 0.003994 0.004096 0.004472 0.005696
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 1]: 5
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003272 0.004253 0.004944 0.004913 0.005518 0.007004
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 1]: 6
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003408 0.004955 0.005449 0.005346 0.005983 0.006371
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 1]: 7
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.004573 0.005655 0.005854 0.005672 0.006125 0.006155
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 1]: 8
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.005903 0.005963 0.006022 0.006022 0.006082 0.006141

boxplot(Tiempo[Algoritmo==1]~Max_repeticiones[Algoritmo==1])
```


0.004
0.005
0.006
0.007
Tiempo[Algoritmo == 1]
4
5
6
7
8
Max_repeticiones[Algoritmo == 1]

Finalmente, el Algoritmo 2:

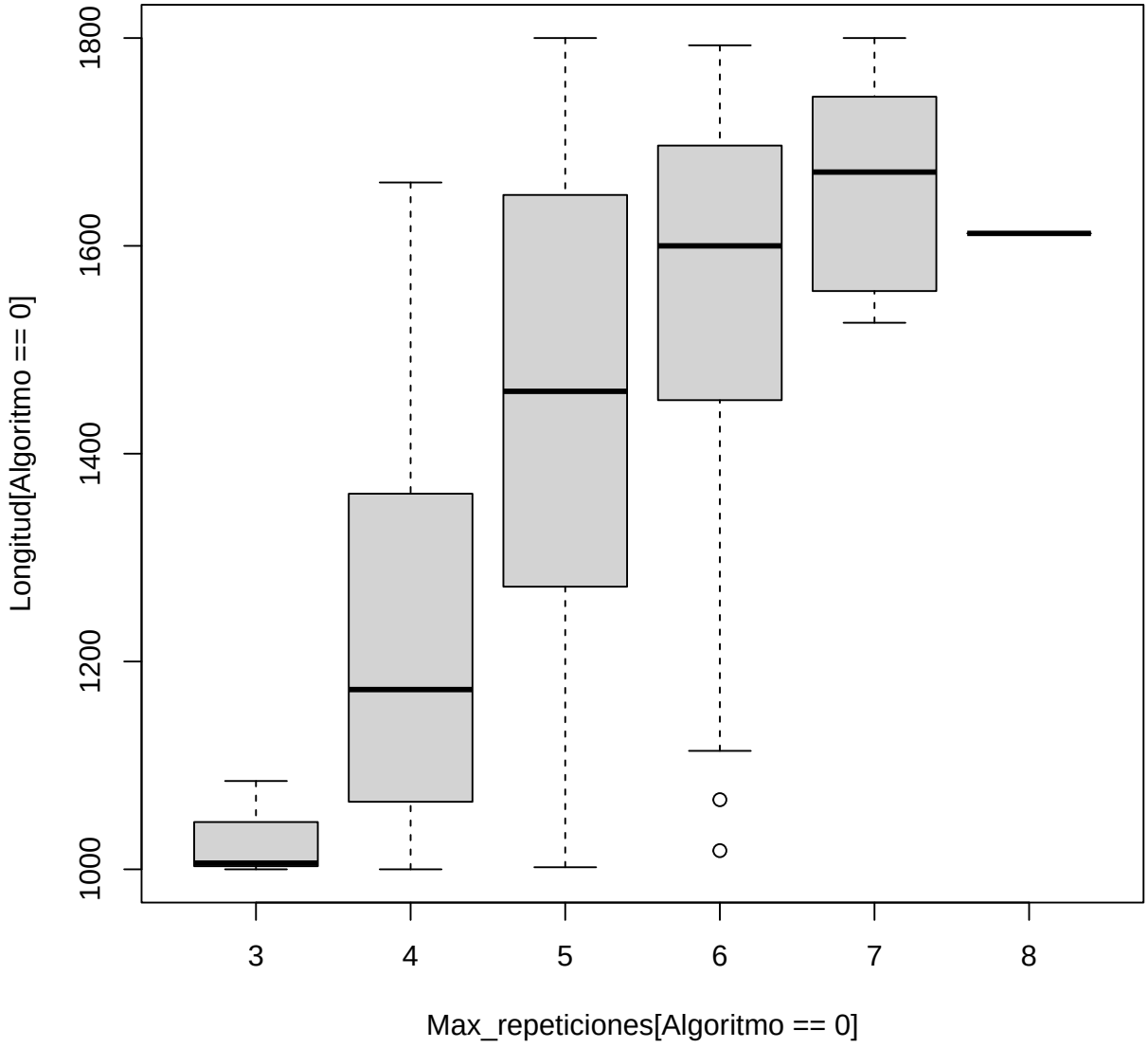
```
by(Tiempo[Algoritmo==2],Max_repeticiones[Algoritmo==2],summary)

## Max_repeticiones[Algoritmo == 2]: 4
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02938 0.03534 0.04027 0.04349 0.04927 0.09527
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 2]: 5
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02969 0.05123 0.06541 0.06545 0.08177 0.09782
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 2]: 6
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03353 0.06227 0.07700 0.07523 0.08792 0.09907
## ------------------------------------------------------------
## Max_repeticiones[Algoritmo == 2]: 7
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03969 0.07662 0.09016 0.08267 0.09590 0.09786

boxplot(Tiempo[Algoritmo==2]~Max_repeticiones[Algoritmo==2])
```



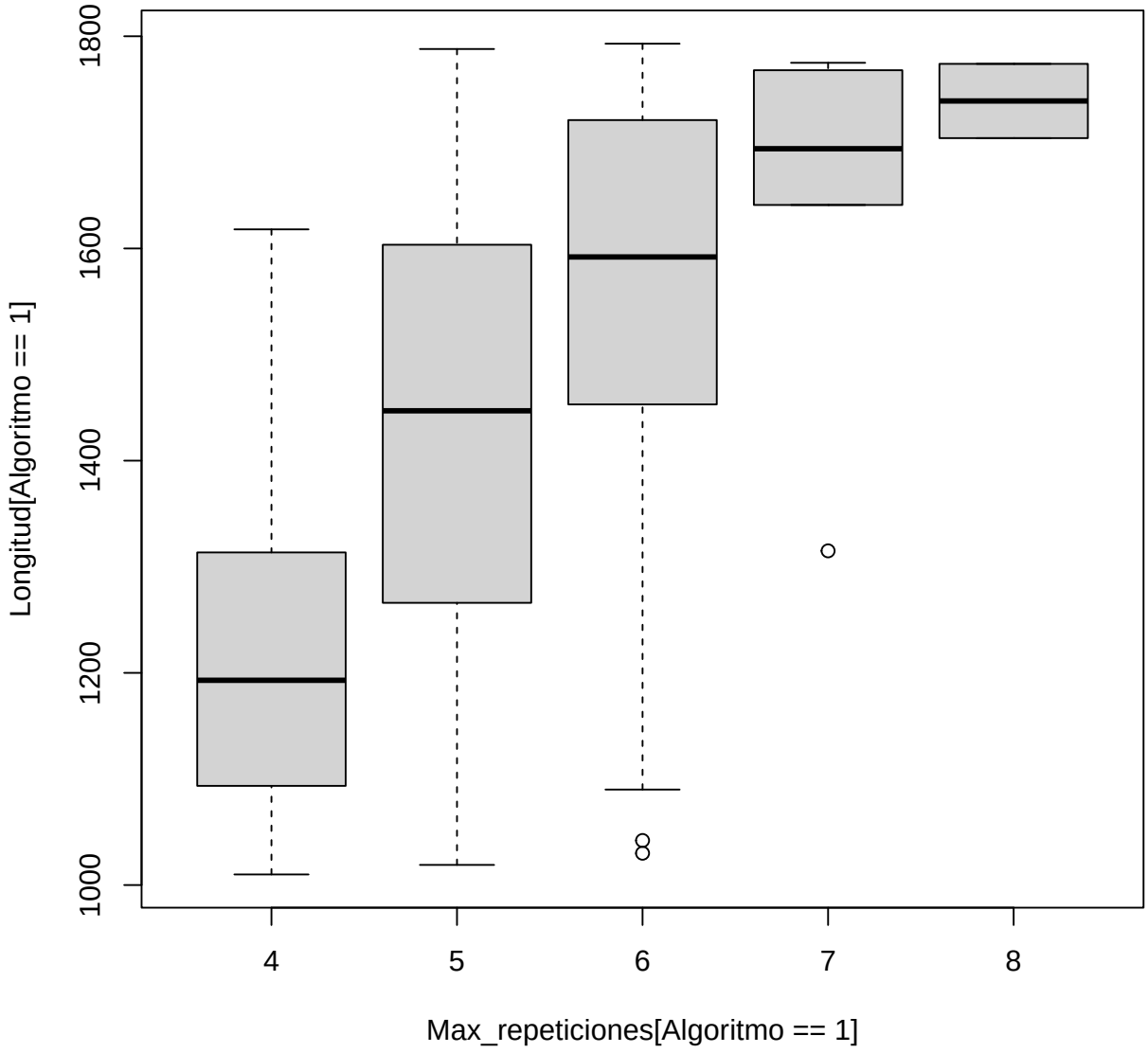
En todos los casos, da la impresión de que las variables están relacionadas y un aumento del número máximo de reperticiones implica un aumento del tiempo de ejecución. Sin embargo, esto en realidad carece de fundamento. No tiene sentido, porque el algoritmo va realizar todas sus comprobaciones (comparaciones) independientemente de que existan repeticiones en las listas con las que trabaja. El algoritmo es ciego.

Bien. Entonces, ¿qué está sucediendo?. Pues que cuando observamos las listas, es más probable encontrar elementos que se repiten mucho en listas más grandes que en listas más pequeñas. Lo que estamos viendo es, por tanto, que la variable Max_repeticiones aumenta con la longitud (y viceversa). Y obviamente, el tiempo de ejecución del algoritmo aumenta al incrementar el tamaño de la lista a la que se aplica. Lo podemos comprobar para cada algoritmo:

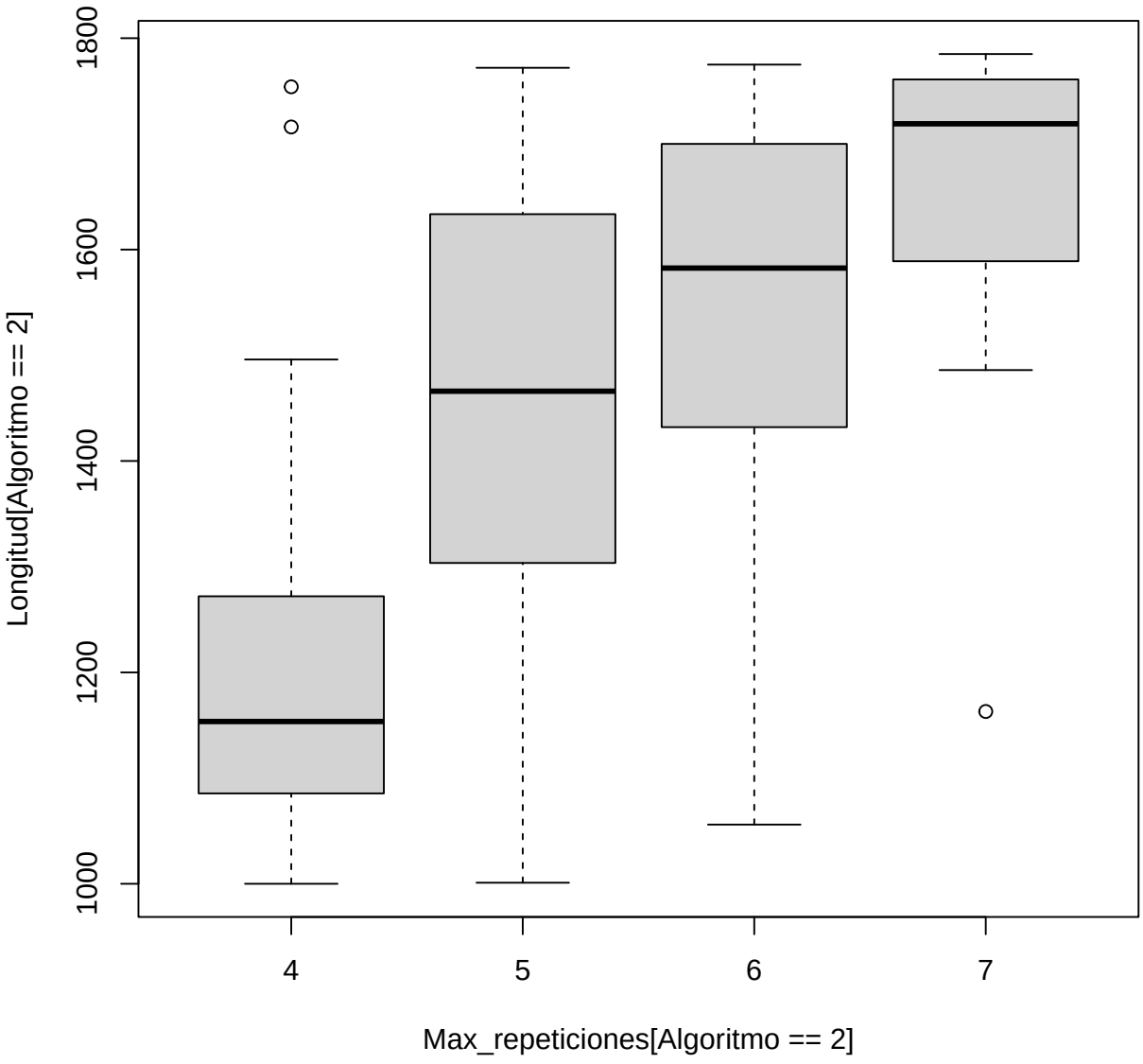
```
boxplot(Longitud[Algoritmo==0]~Max_repeticiones[Algoritmo==0])
```




```
boxplot(Longitud[Algoritmo==1]~Max_repeticiones[Algoritmo==1])
```




```
boxplot(Longitud[Algoritmo==2]~Max_repeticiones[Algoritmo==2])
```


Longitud[Algoritmo == 2]
1000
1200
1400
1600
1800
4
5
6
7
Max_repeticiones[Algoritmo == 2]