SANTIAGO_ROYUELA_SAMIT_PEC3_SAD

Santiago Royuela Samit

10/1/2022

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Software para el Análisis de Datos. PEC 3

Estudio de la pandemia del COVID en España y Simulaciones de Montecarlo.

1.Presentando el trabajo de estudio:

En el presente trabajo tomaremos datos oficiales del INE, basados en el año 2021 y a los que iremos haciendo referencia en concreto, sobre los casos de COVID en España, sus CCAA y provincias, así como grupos de edades y sexo. Iremos aplicando diferentes técnicas estadísticas aprendidas en el curso para analizar y graficar los datos, así como las conclusiones pertinentes. Posteriormente haremos una simulación de la pandemia con Métodos de Montecarlo.

Primero cargamos librerías y ficheros de datos que vamos a necesitar:

```
library(dplyr)
#install.packages ("sqldf")
library(sqldf) #librería para sql
library(ggplot2)
############FUENTE DE DATOS DE ESPAÑA COVID: https://cnecovid.isciii.es/covid19/#documentaci%C3%B3n-y-da
datosprov <- read.csv("casos_tecnica_provincia.csv")
datosprov <- na.omit(datosprov) #Eliminamos los valores nulos.
#Hemos leído los casos de COVID por provincia en España, del año 2021. Hemos omitido valores nulos.
#Ahora cargaremos un fichero con el código ISO de las provincias de España.
codigoprov <- read.csv("provinces_es.csv")
#Ahora unimos ambos dataframes por la columna que comparte, la de provincia_iso
datprov <- merge(datosprov, codigoprov, by = "provincia_iso", all = TRUE)


#Cargamos el censo por provincias que he extraído de WIKIPEDIA según el INE, del año 2021.
censoprov <- read.csv("censoprovincia21.csv",sep=";")
head(censoprov,4) #Hacemos un head para ver qué tipo de datos almacena "censoprov".
```

```
##   Provincias  Sexo Periodo   Total
## 1   Albacete Total    2021  386464
## 2    Alacant Total    2021 1881762
## 3    Almeria Total    2021  731792
## 4      Alava Total    2021  333626
```

```
#Unimos los datos de ambos dataframes por la columna Provincias
datprov <- merge(datprov, censoprov, by = "Provincias", all = TRUE)
#Omitimos valores nulos.
```

```r
datprov <- na.omit(datprov)
str(datprov)
```

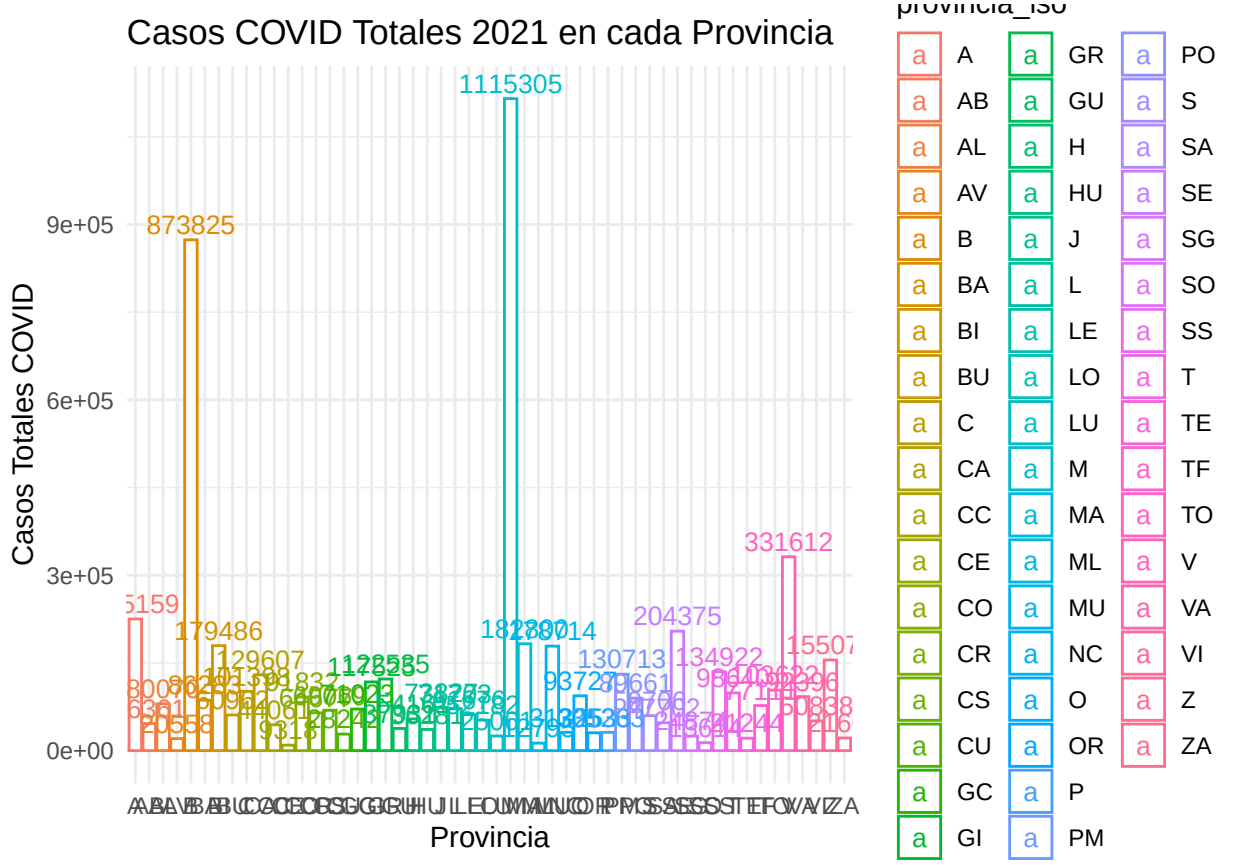
```
## 'data.frame':    35770 obs. of  15 variables:
##  $ Provincias                 : chr  "Alacant" "Alacant" "Alacant" "Alacant" ...
##  $ provincia_iso              : chr  "A" "A" "A" "A" ...
##  $ fecha                      : chr  "2020-01-22" "2020-01-04" "2021-06-15" "2021-04-03" ...
##  $ num_casos                  : int  0 0 55 38 532 2824 4 158 11 545 ...
##  $ num_casos_prueba_pcr       : int  0 0 20 27 295 1095 1 67 8 156 ...
##  $ num_casos_prueba_test_ac   : int  0 0 0 0 0 0 0 0 2 0 ...
##  $ num_casos_prueba_ag        : int  0 0 19 8 65 583 0 0 0 21 ...
##  $ num_casos_prueba_elisa     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_casos_prueba_desconocida: int  0 0 16 3 172 1146 3 91 1 368 ...
##  $ postal_code                : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ phone_code                 : int  950 950 950 950 950 950 950 950 950 950 ...
##  $ iso2                       : chr  "ES" "ES" "ES" "ES" ...
##  $ Sexo                       : chr  "Total" "Total" "Total" "Total" ...
##  $ Periodo                    : int  2021 2021 2021 2021 2021 2021 2021 2021 2021 2021 ...
##  $ Total                      : int  1881762 1881762 1881762 1881762 1881762 1881762 1881762 188176
##  - attr(*, "na.action")= 'omit' Named int [1:2193] 13141 13142 13143 13144 13145 13146 13147 13148 1
##   ..- attr(*, "names")= chr [1:2193] "13141" "13142" "13143" "13144" ...
```

```r
#Ahora creamos una columna nueva en donde indicamos el número de casos por el
#total de habitantes en la población.
datprov$casospercent <- (datprov$num_casos/datprov$Total)*100
head(datprov,4) #Habrá muchos campos que no vamos a utilizar, como los tipos de
```

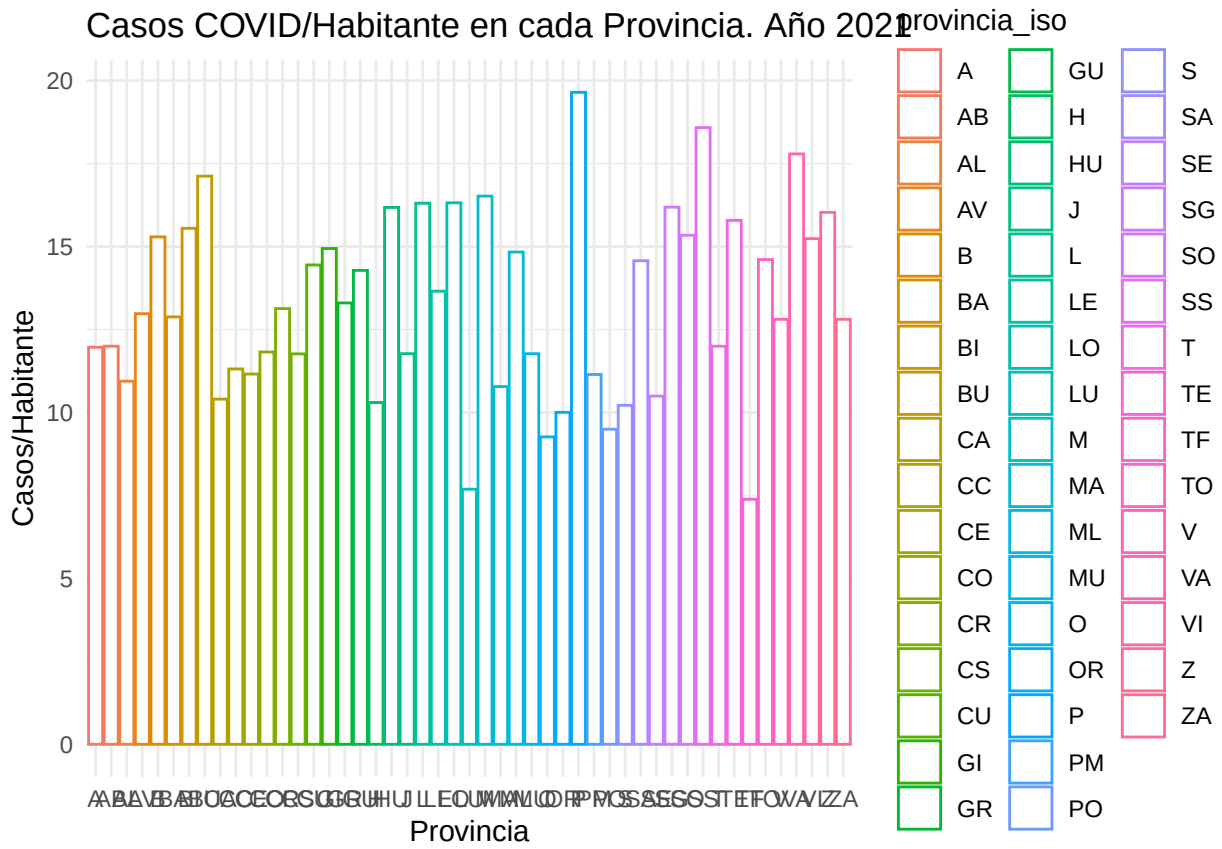
```
##   Provincias provincia_iso      fecha num_casos num_casos_prueba_pcr
## 1    Alacant             A 2020-01-22         0                    0
## 2    Alacant             A 2020-01-04         0                    0
## 3    Alacant             A 2021-06-15        55                   20
## 4    Alacant             A 2021-04-03        38                   27
##   num_casos_prueba_test_ac num_casos_prueba_ag num_casos_prueba_elisa
## 1                        0                   0                     0
## 2                        0                   0                     0
## 3                        0                  19                     0
## 4                        0                   8                     0
##   num_casos_prueba_desconocida postal_code phone_code iso2  Sexo Periodo
## 1                            0           3        950   ES Total    2021
## 2                            0           3        950   ES Total    2021
## 3                           16           3        950   ES Total    2021
## 4                            3           3        950   ES Total    2021
##     Total casospercent
## 1 1881762  0.000000000
## 2 1881762  0.000000000
## 3 1881762  0.002922793
## 4 1881762  0.002019384
```

```r
#test de COVID. No atenderemos a ellos.

#Sumamos los casos por cada provincia (no dividiendo por el nº de habitantes):
A = datosprov %>% group_by(provincia_iso)
A = A %>% summarise(n=sum(num_casos))

####Graficamos los casos por provincia
```

```
ggplot(data=A, aes(x=provincia_iso, y=n,color=provincia_iso)) +
  geom_bar(stat="identity", fill="white")+
  geom_text(aes(label=n), vjust=-0.3, size=3.5)+

  ggtitle("Casos COVID Totales 2021 en cada Provincia") +
  xlab("Provincia") + ylab("Casos Totales COVID") +
  theme_minimal()
```




```
### Hacemos un gráfico con las casos/habitante de las provincias:
######## Sacar datos del censo o filtrarlo para que sea más útil ******************
A2 = datprov %>% group_by(provincia_iso)
A2 = A2 %>% summarise(n=sum(casospercent))
ggplot(data=A2, aes(x=provincia_iso, y=n,color=provincia_iso)) +
  geom_bar(stat="identity", fill="white") +
  #geom_text(aes(label=n), vjust=-0.3, size=3.5) +
  ggtitle("Casos COVID/Habitante en cada Provincia. Año 2021") +
  xlab("Provincia") + ylab("Casos/Habitante") +
  theme_minimal()
```



Observamos una mayor uniformidad en el gráfico de casos/habitante en cada población, como era de esperar, pues el número absoluto de casos dependerá siempre del número de habitantes en la población.

Veamos el número de casos por provincia que se han dado hasta día de hoy:

```
# Enlace población provincias españa fichero csv: https://www.ine.es/jaxiT3/Tabla.htm?t=2852&L=0

#Ahora vamos a ver los casos totales de positivos por provinciaque se han dado:

prov = c()
totcasos = c()
provincias <- unique(datprov$provincia_iso)
provincias
```

```
##  [1] "A"  "VI" "AB" "AL" "O"  "AV" "BA" "PM" "B"  "BI" "BU" "CC" "CA" "S"  "CS"
## [16] "CE" "CR" "CO" "CU" "SS" "GI" "GR" "GU" "H"  "HU" "J"  "LE" "L"  "LU" "M"
## [31] "MA" "ML" "MU" "OR" "P"  "PO" "LO" "SA" "SG" "SE" "SO" "TF" "T"  "TE" "TO"
## [46] "V"  "VA" "ZA" "Z"
```

```
for (i in (1:length(provincias))) {
  print(provincias[i])
  tot <- sum(datprov$num_casos[datprov$provincia_iso == provincias[i]])
  print(tot)
  prov <- rbind(prov, provincias[i])
  totcasos <- rbind(totcasos,tot)
}
```

```
## [1] "A"
## [1] 225159
```

```
## [1] "VI"
## [1] 50838
## [1] "AB"
## [1] 46361
## [1] "AL"
## [1] 80070
## [1] "O"
## [1] 93727
## [1] "AV"
## [1] 20558
## [1] "BA"
## [1] 86283
## [1] "PM"
## [1] 130713
## [1] "B"
## [1] 873825
## [1] "BI"
## [1] 179486
## [1] "BU"
## [1] 60962
## [1] "CC"
## [1] 44061
## [1] "CA"
## [1] 129607
## [1] "S"
## [1] 59706
## [1] "CS"
## [1] 69069
## [1] "CE"
## [1] 9318
## [1] "CR"
## [1] 64677
## [1] "CO"
## [1] 91832
## [1] "CU"
## [1] 28242
## [1] "SS"
## [1] 134922
## [1] "GI"
## [1] 117525
## [1] "GR"
## [1] 122535
## [1] "GU"
## [1] 37931
## [1] "H"
## [1] 54161
## [1] "HU"
## [1] 36281
## [1] "J"
## [1] 73827
## [1] "LE"
## [1] 61676
## [1] "L"
## [1] 71703
```

```
## [1] "LU"
## [1] 25061
## [1] "M"
## [1] 1115305
## [1] "MA"
## [1] 182800
## [1] "ML"
## [1] 12795
## [1] "MU"
## [1] 178714
## [1] "OR"
## [1] 30533
## [1] "P"
## [1] 31263
## [1] "PO"
## [1] 89661
## [1] "LO"
## [1] 52182
## [1] "SA"
## [1] 47702
## [1] "SG"
## [1] 24874
## [1] "SE"
## [1] 204375
## [1] "SO"
## [1] 13614
## [1] "TF"
## [1] 77111
## [1] "T"
## [1] 98640
## [1] "TE"
## [1] 21244
## [1] "TO"
## [1] 103622
## [1] "V"
## [1] 331612
## [1] "VA"
## [1] 92396
## [1] "ZA"
## [1] 21607
## [1] "Z"
## [1] 155075
```

```
prov <- as.data.frame(prov)
totcasos <- as.data.frame(totcasos)
totcasosprov = cbind(totcasos,prov)
colnames(totcasosprov)[1] <- "Casos"
colnames(totcasosprov)[2] <- "Provincia"
head(totcasosprov,51)
```

```
##          Casos Provincia
## tot     225159         A
## tot.1    50838        VI
## tot.2    46361        AB
## tot.3    80070        AL
```

```
## tot.4    93727         O
## tot.5    20558        AV
## tot.6    86283        BA
## tot.7   130713        PM
## tot.8   873825         B
## tot.9   179486        BI
## tot.10   60962        BU
## tot.11   44061        CC
## tot.12  129607        CA
## tot.13   59706         S
## tot.14   69069        CS
## tot.15    9318        CE
## tot.16   64677        CR
## tot.17   91832        CO
## tot.18   28242        CU
## tot.19  134922        SS
## tot.20  117525        GI
## tot.21  122535        GR
## tot.22   37931        GU
## tot.23   54161         H
## tot.24   36281        HU
## tot.25   73827         J
## tot.26   61676        LE
## tot.27   71703         L
## tot.28   25061        LU
## tot.29 1115305         M
## tot.30  182800        MA
## tot.31   12795        ML
## tot.32  178714        MU
## tot.33   30533        OR
## tot.34   31263         P
## tot.35   89661        PO
## tot.36   52182        LO
## tot.37   47702        SA
## tot.38   24874        SG
## tot.39  204375        SE
## tot.40   13614        SO
## tot.41   77111        TF
## tot.42   98640         T
## tot.43   21244        TE
## tot.44  103622        TO
## tot.45  331612         V
## tot.46   92396        VA
## tot.47   21607        ZA
## tot.48  155075         Z
```

```
head(totcasos,51)
```

```
##             V1
## tot     225159
## tot.1    50838
## tot.2    46361
## tot.3    80070
## tot.4    93727
## tot.5    20558
```

```
## tot.6    86283
## tot.7   130713
## tot.8   873825
## tot.9   179486
## tot.10   60962
## tot.11   44061
## tot.12  129607
## tot.13   59706
## tot.14   69069
## tot.15    9318
## tot.16   64677
## tot.17   91832
## tot.18   28242
## tot.19  134922
## tot.20  117525
## tot.21  122535
## tot.22   37931
## tot.23   54161
## tot.24   36281
## tot.25   73827
## tot.26   61676
## tot.27   71703
## tot.28   25061
## tot.29 1115305
## tot.30  182800
## tot.31   12795
## tot.32  178714
## tot.33   30533
## tot.34   31263
## tot.35   89661
## tot.36   52182
## tot.37   47702
## tot.38   24874
## tot.39  204375
## tot.40   13614
## tot.41   77111
## tot.42   98640
## tot.43   21244
## tot.44  103622
## tot.45  331612
## tot.46   92396
## tot.47   21607
## tot.48  155075
```

Vemos un pqueño resumen estadístico de los casos Totales de Covid por provincias en 2021:

```
summary(totcasosprov)
```

```
##      Casos          Provincia
##  Min.   :   9318   Length:49
##  1st Qu.:  37931   Class :character
##  Median :  71703   Mode  :character
##  Mean   : 121740
##  3rd Qu.: 122535
##  Max.   :1115305
```

```r
maxi <- max(totcasosprov$Casos)
mini <- min(totcasosprov$Casos)
promaxi <- totcasosprov[totcasosprov$Casos == maxi,]
print("La Provincia con más casos de COVID en el año 2021 es:")
```

```
## [1] "La Provincia con más casos de COVID en el año 2021 es:"
```

```r
print(promaxi$Provincia)
```

```
## [1] "M"
```

```r
print(promaxi$Casos)
```

```
## [1] 1115305
```

```r
promini <- totcasosprov[totcasosprov$Casos == mini,]
print("La Provincia con menos casos de COVID en el año 2021 es:")
```

```
## [1] "La Provincia con menos casos de COVID en el año 2021 es:"
```

```r
print(promini$Provincia)
```

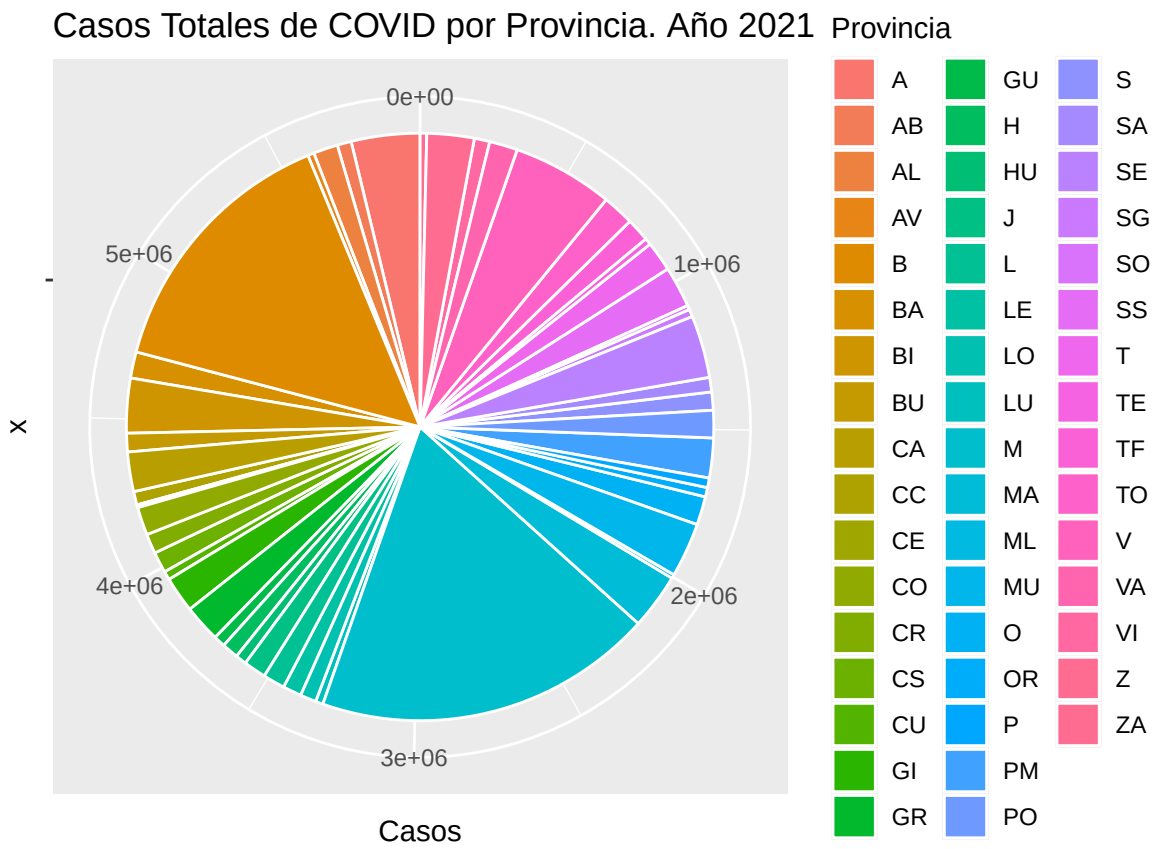
```
## [1] "CE"
```

```r
print(promini$Casos)
```

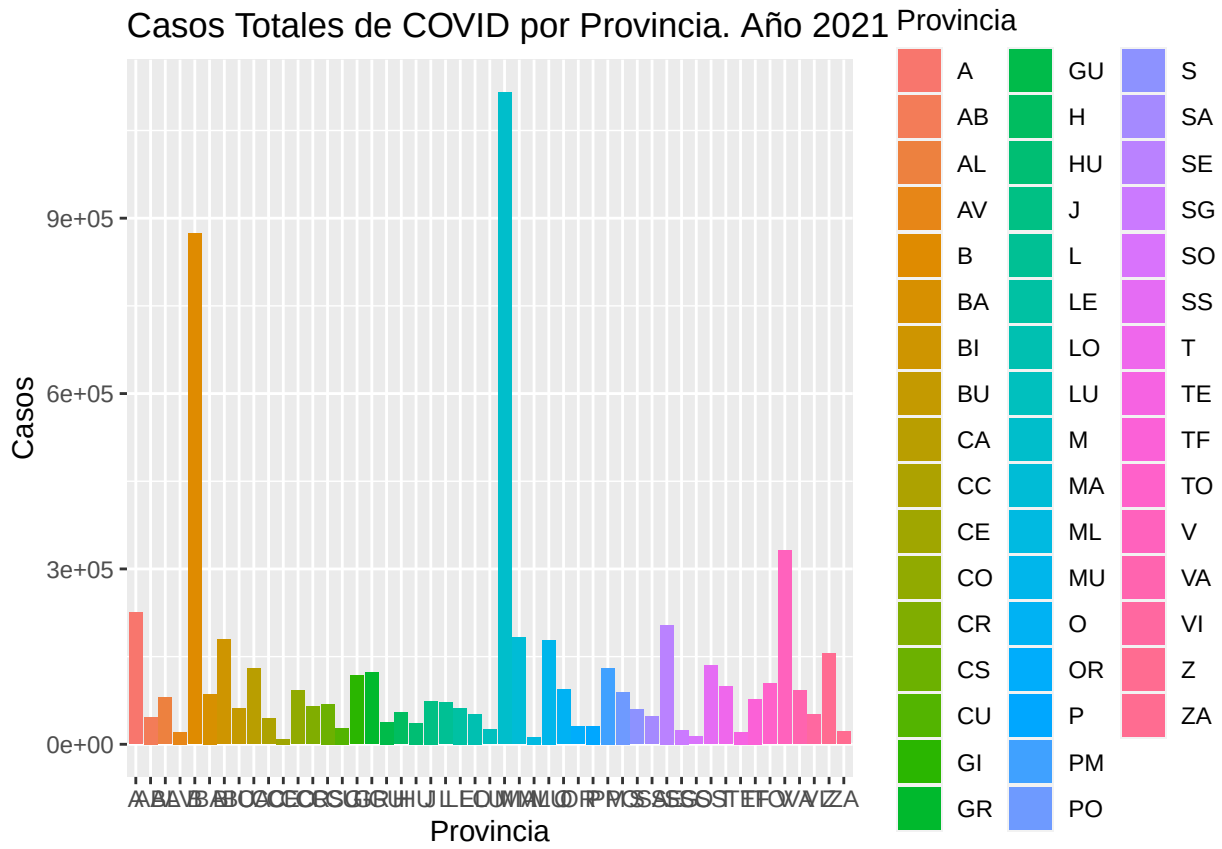
```
## [1] 9318
```

Ahora realizamos algunas gráficas de los casos de COVID por provincias:

```r
#library(ggplot2)
ggplot(totcasosprov,aes(x="",y=Casos, fill=Provincia))+
  geom_bar(stat = "identity",color="white")+
  coord_polar(theta="y") +
  ggtitle("Casos Totales de COVID por Provincia. Año 2021")
```



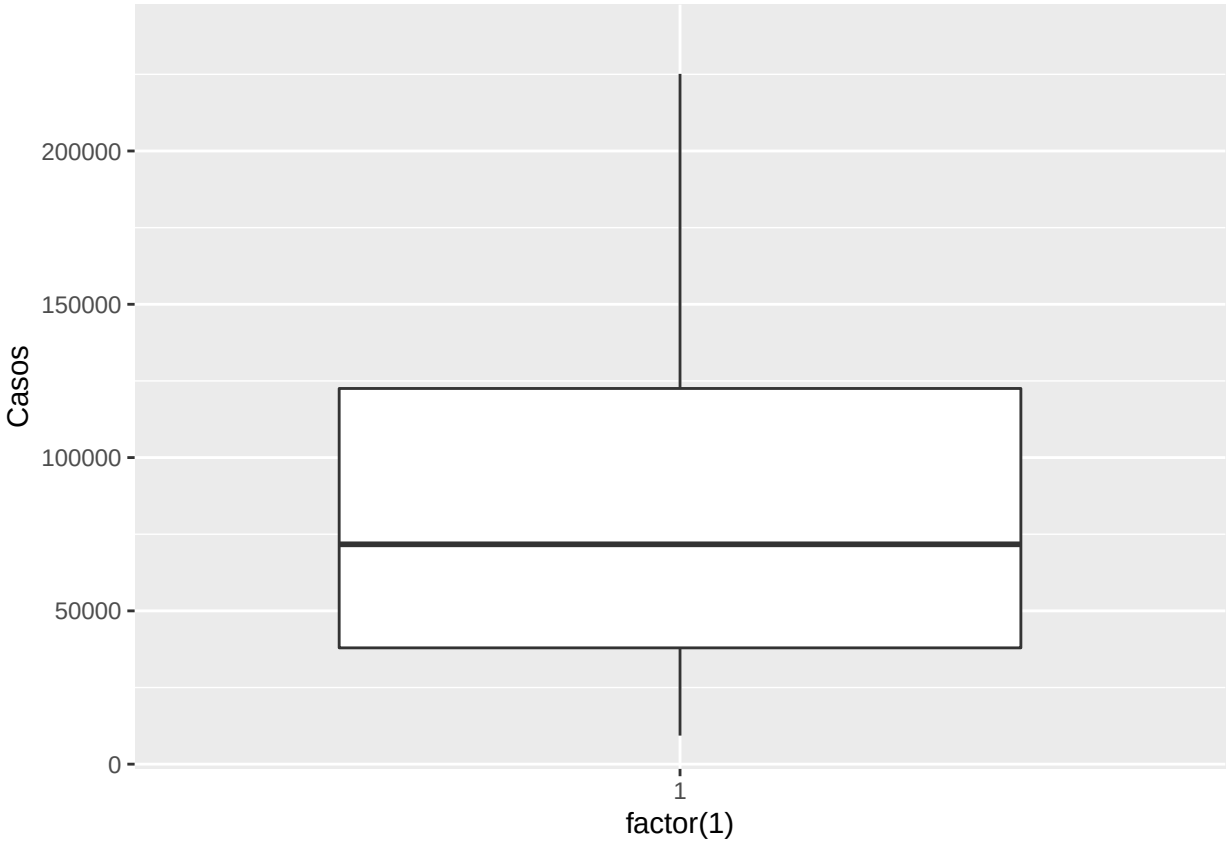

```
### Gráfico Barplot:
ggplot(totcasosprov,aes(x=Provincia,y=Casos, fill=Provincia))+
  geom_bar(stat = "identity") +
  ggtitle("Casos Totales de COVID por Provincia. Año 2021")
```



En este caso, al ser números totales y no relativos al numero de habitantes en la población, la distribución es menos homogénea que cuando graficábamos los casos/nºhabitantes en cada población.

Ahora realizamos un gráfico Boxplot de los casos totales de COVID:

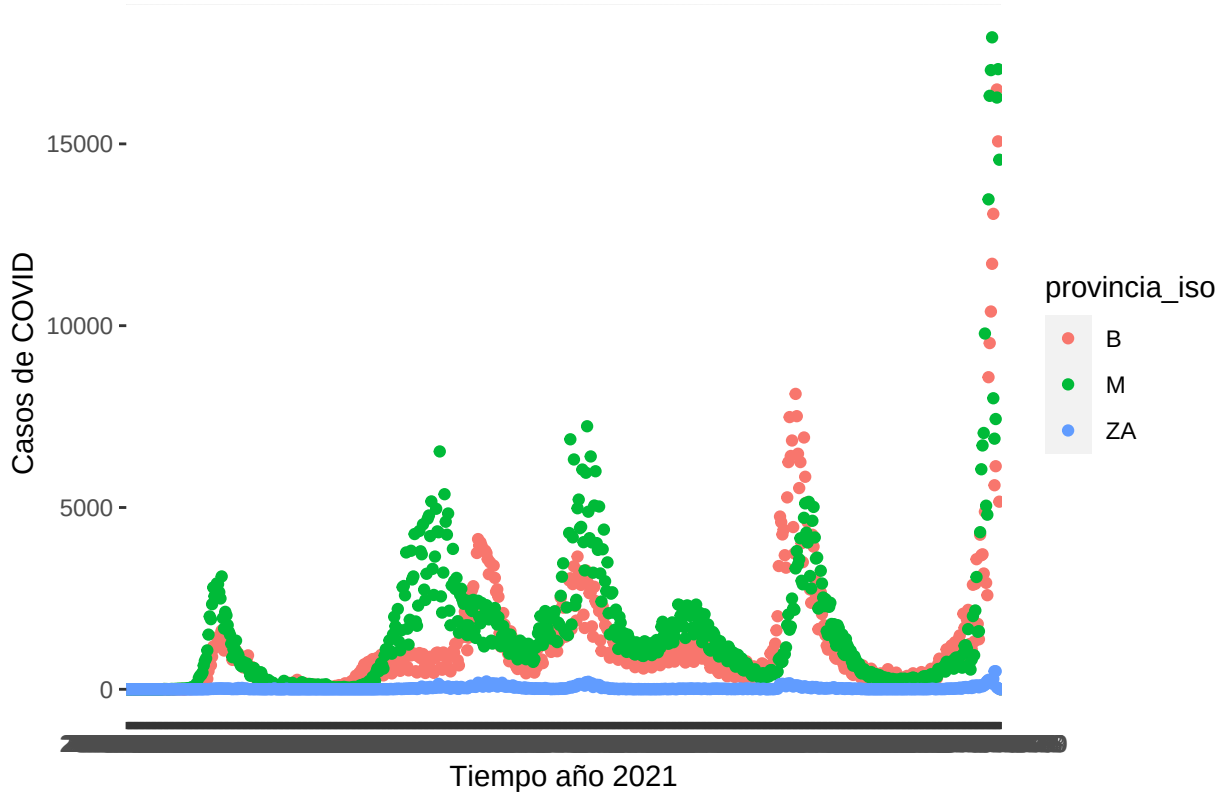
```
###### hacemos bien el boxplot. Utilizo la información de un enlace que he encontrado: #https://www.it-
# compute lower and upper whiskers
ylim1 = boxplot.stats(totcasosprov$Casos)$stats[c(1, 5)]
p0 = ggplot(totcasosprov, aes(y = Casos )) + geom_boxplot(aes(x = factor(1)))
# scale y limits based on ylim1
p1 = p0 + coord_cartesian(ylim = ylim1*1.05)
p1
```




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

Graficamos los casos de COVID Totales en el año 2021 para las provincias de Madrid, Barcelona y Zaragoza:

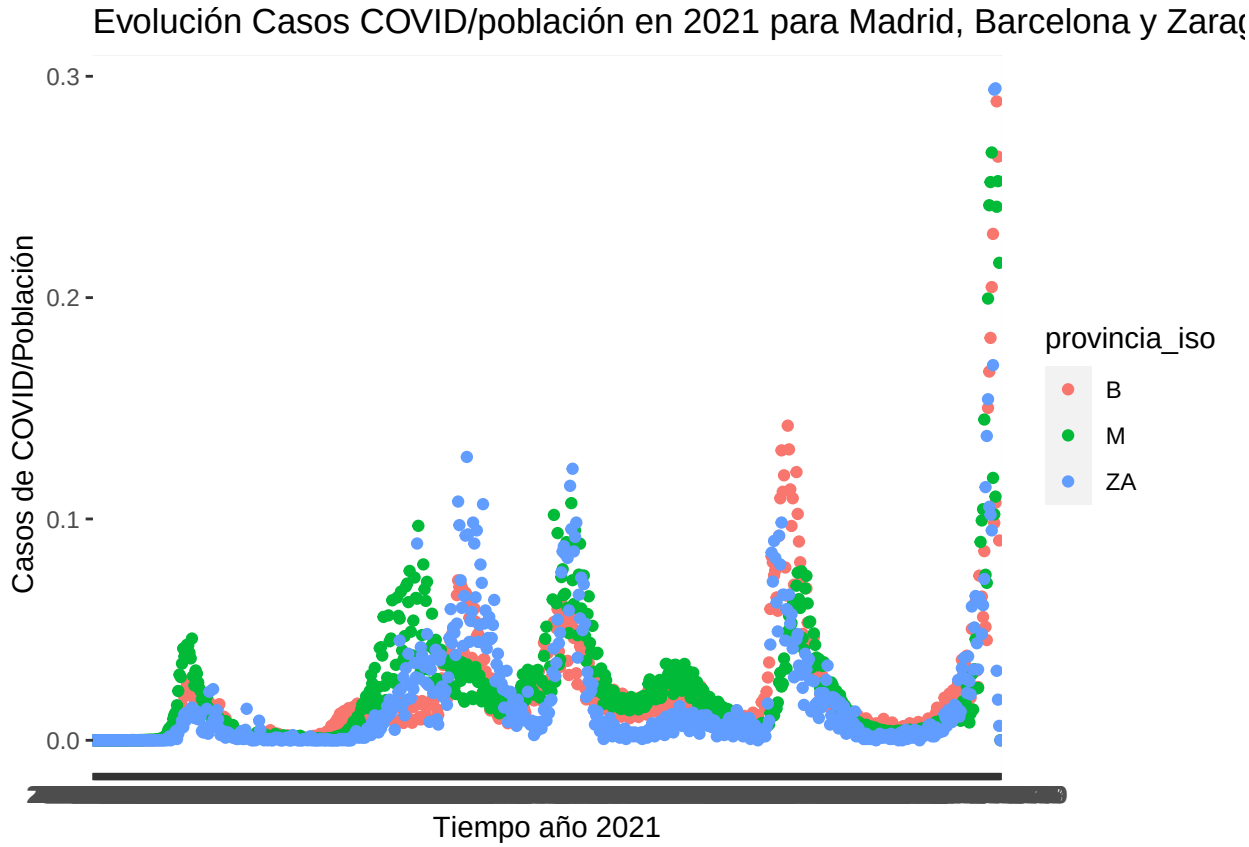
```
ggplot(datprov[(datprov$provincia_iso == 'B') | (datprov$provincia_iso == 'M') |(datprov$provincia_iso =
       aes(x=fecha, y=num_casos,  color=provincia_iso)) +
  geom_point() +
  ggtitle("Evolución Casos COVID en 2021 para Madrid, Barcelona y Zaragoza") +
  xlab("Tiempo año 2021") + ylab("Casos de COVID")
```



En el gráfico observamos que las 3 provincias guardan un patrón muy similar en los casos de COVID, mostrando los mismos picos, aunque quizá algunos de ellos algo desfasados, pero mostrando el mismo comportamiento. El segundo pico o brote de COVID detectado en 2021 se produjo en Madrid, la capital, antes de que se diera en Barcelona, por ejemplo.

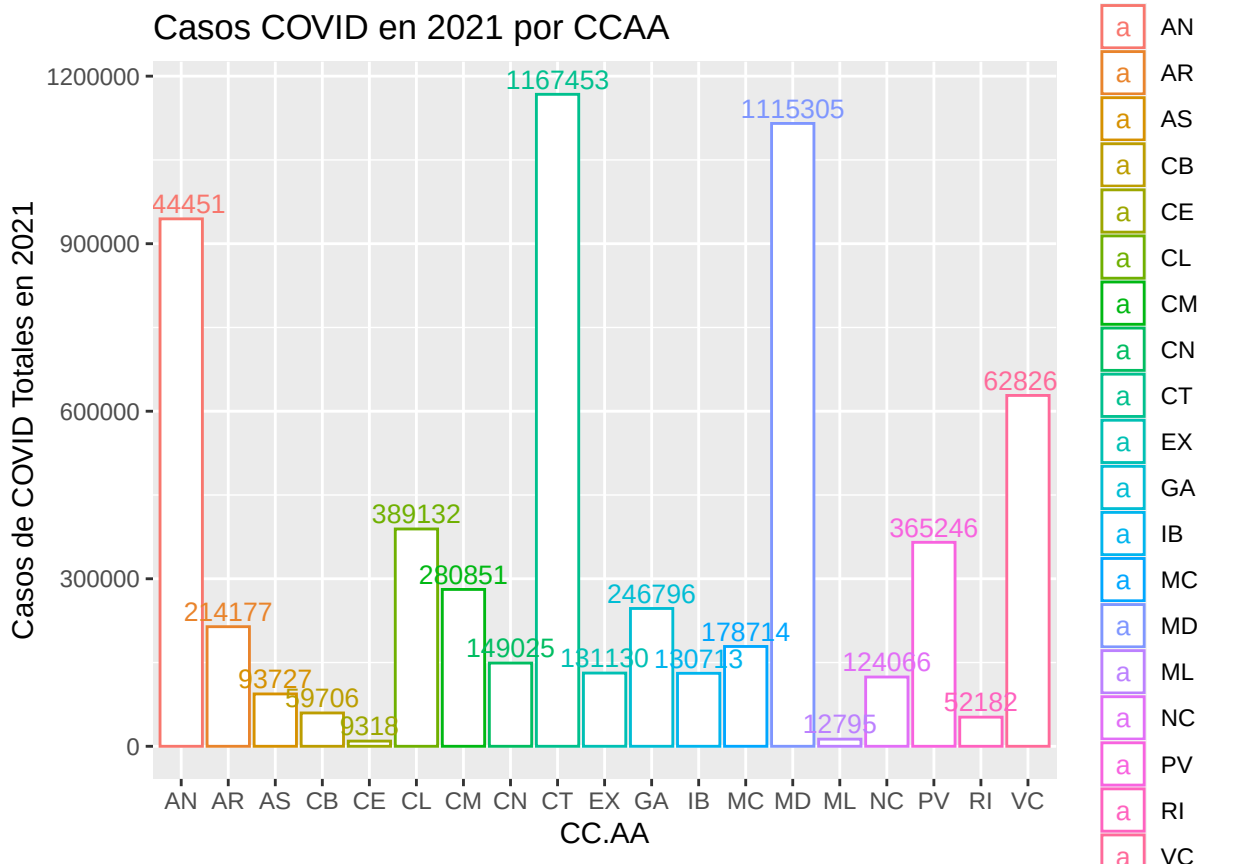
Ahora graficaremos para las mismas provincias (M,B,ZA) en el 2021 pero con los Casos/Nº de Habitantes, donde veremos que Zaragoza recobra importancia, puesto que en el gráfico anterior quedaba "disimulada" la pandemia debido a ser comparada con los casos en provincias con mucha mayor población. Incluso se observa que Zaragoza presenta un pico más que las otras dos provincias.

```
ggplot(datprov[(datprov$provincia_iso == 'B') | (datprov$provincia_iso == 'M') |(datprov$provincia_iso =
       aes(x=fecha, y=casospercent,  color=provincia_iso)) +
  geom_point() +
  ggtitle("Evolución Casos COVID/población en 2021 para Madrid, Barcelona y Zaragoza") +
  xlab("Tiempo año 2021") + ylab("Casos de COVID/Población")
```



Hacemos un pequeño estudio por CCAA. Para ello cargamos el fichero pertinente de la misma fuente de datos, extraída a su vez del INE. Vamos a graficar los casos totales de COVID en 2021 por CCAA:

```
###################### COMUNIDADES AUTÓNOMAS *********
datoscom <- read.csv("casos_tecnica_ccaa_res.csv")
datoscom <- na.omit(datoscom)
#View(datoscom)
B = datoscom %>% group_by(ccaa_iso)
B = B %>% summarise(n=sum(num_casos))

####MODIFICAR ESTE GRÁFICO DIVIDIENDO POR EL NÚMERO TOTAL DE HABITANTES/CCAA***************
ggplot(data=B, aes(x=ccaa_iso, y=n,color=ccaa_iso)) +
  geom_bar(stat="identity", fill="white")+
  geom_text(aes(label=n), vjust=-0.3, size=3.5)+
  ggtitle("Casos COVID en 2021 por CCAA") +
  xlab("CC.AA") + ylab("Casos de COVID Totales en 2021")
```




```
  theme_minimal()
```

```
## List of 93
##  $ line                        :List of 6
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                        :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                        :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
```

```
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                     : NULL
##  $ aspect.ratio              : NULL
##  $ axis.title                : NULL
##  $ axis.title.x              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top          :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom       : NULL
##  $ axis.title.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left         : NULL
##  $ axis.title.y.right        :List of 11
```

```
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
```

```
##  $ axis.text.x.bottom        : NULL
##  $ axis.text.y               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left          : NULL
##  $ axis.text.y.right         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x              : NULL
##  $ axis.ticks.x.top          : NULL
##  $ axis.ticks.x.bottom       : NULL
##  $ axis.ticks.y              : NULL
##  $ axis.ticks.y.left         : NULL
##  $ axis.ticks.y.right        : NULL
##  $ axis.ticks.length         : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x       : NULL
##  $ axis.ticks.length.x.top   : NULL
##  $ axis.ticks.length.x.bottom: NULL
##  $ axis.ticks.length.y       : NULL
##  $ axis.ticks.length.y.left  : NULL
##  $ axis.ticks.length.y.right : NULL
##  $ axis.line                 : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x               : NULL
##  $ axis.line.x.top           : NULL
##  $ axis.line.x.bottom        : NULL
##  $ axis.line.y               : NULL
##  $ axis.line.y.left          : NULL
##  $ axis.line.y.right         : NULL
```

```
##  $ legend.background         : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin             : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing            : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x          : NULL
##  $ legend.spacing.y          : NULL
##  $ legend.key                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size           : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height         : NULL
##  $ legend.key.width          : NULL
##  $ legend.text               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.align         : NULL
##  $ legend.title              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.align        : NULL
##  $ legend.position           : chr "right"
##  $ legend.direction          : NULL
##  $ legend.justification      : chr "center"
##  $ legend.box                : NULL
##  $ legend.box.just           : NULL
##  $ legend.box.margin         : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing        : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ panel.background          : list()
```

```
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.border              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.spacing             : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ panel.spacing.x           : NULL
##  $ panel.spacing.y           : NULL
##  $ panel.grid                :List of 6
##   ..$ colour       : chr "grey92"
##   ..$ size         : NULL
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major          : NULL
##  $ panel.grid.minor          :List of 6
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.5
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major.x        : NULL
##  $ panel.grid.major.y        : NULL
##  $ panel.grid.minor.x        : NULL
##  $ panel.grid.minor.y        : NULL
##  $ panel.ontop               : logi FALSE
##  $ plot.background           : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ plot.title                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.title.position       : chr "panel"
##  $ plot.subtitle             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
```

```
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption.position     : chr "panel"
##  $ plot.tag                  :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ strip.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ strip.background.x        : NULL
##  $ strip.background.y        : NULL
##  $ strip.placement           : chr "inside"
##  $ strip.text                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey10"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 4.4points 4.4points 4.4points 4.4points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
```

```
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.text.x              : NULL
##  $ strip.text.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.switch.pad.grid     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.switch.pad.wrap     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.text.y.left         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

Vemos que los casos más altos corresponden a las comunidades de Cataluña, Madrid y Andalucía. Veamos ah
Para ello cargamos un fichero con el censo de 2021 de las CCAA de España obtenido de WIKIPEDIA y que ha

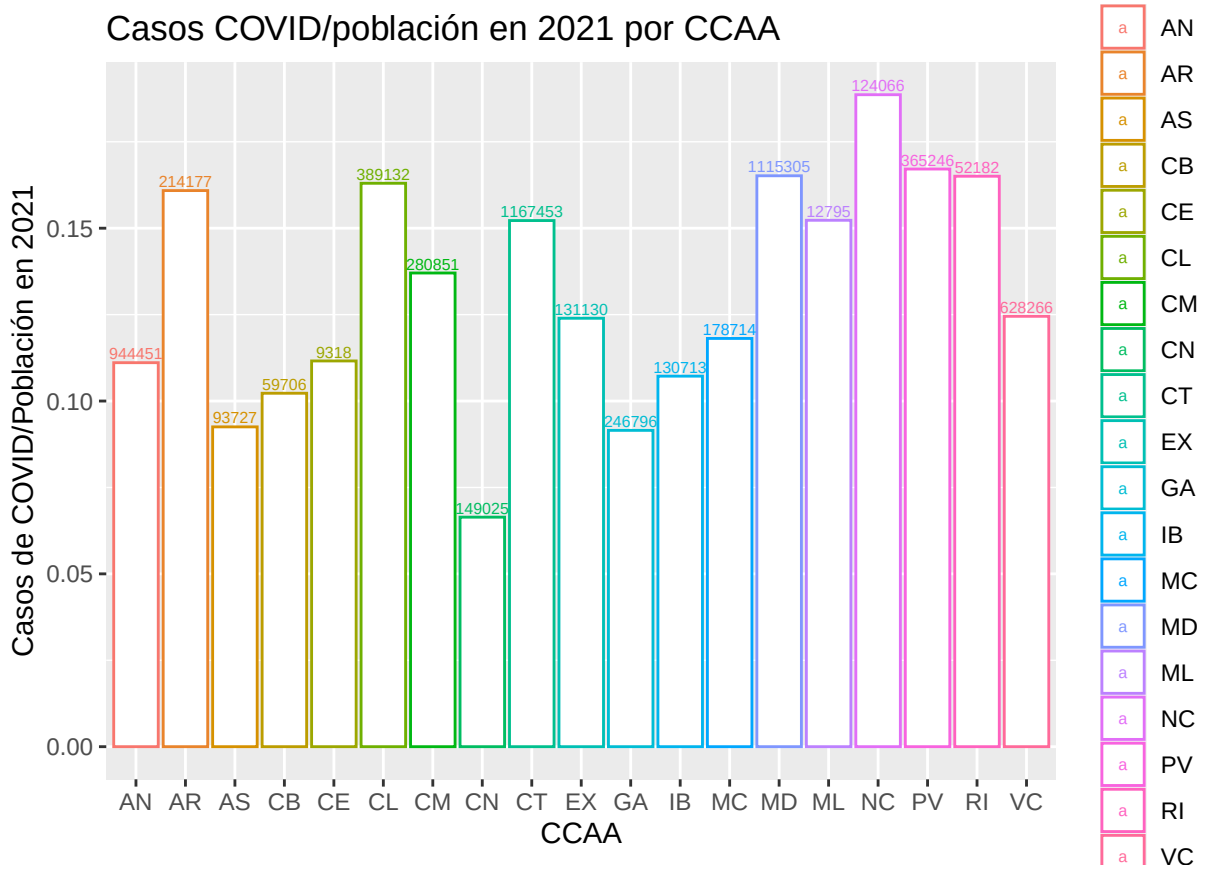
```
######### Estudio por densidad de población en CC.AA: *****
censowiki <- read.csv("censowiki.csv",header = TRUE, sep="")
censowiki$X2021
```

```
##   [1] 338786 363341 382059 408287 431472 443170 450016 445284 469239 486596
##  [11] 491380 501426 527716 510018 507740 498747 495260 488794 475565 479813
##  [21] 484847 479183 470774 479101 478719 481718 490899 509736 525781 523378
##  [31] 531524 539644 549056 555864 569230 588611 611266 631276 667423 697388
##  [41] 731529 753631 787828 798467 812938 806728 800884 781296 777575 765164
##  [51] 753516 745521 738061 741354 721854 715168 722272 686102 661337 644202
##  [61] 644806 624692 607662 590685 549081 530169 503637 500877 495402 460630
##  [71] 446989 459503 475082 429268 404434 416011 388845 374888 317167 284752
##  [81] 338743 217288 236104 252791 262547 237765 218866 204540 182691 153104
##  [91] 134152 107330  88281  66962  53675  40002  29854  21407  15093   9851
## [101]  13043
```

```
tot20 <- sum(censowiki$X2020)
tot21 <- sum(censowiki$X2021)

tot21
```

```
## [1] 47394223
```

```
censoccaa21 <- read.csv("censoccaa21.csv",header=TRUE)


ccaa <- unique(censoccaa21$ccaa)

B <- as.data.frame((B))


#Uno la información de los Dataframes censoccaa21 y B en un Dataframe C:
C <- merge(censoccaa21, B, by = "ccaa_iso", all = TRUE)
C$porcencasos <- C$n/C$poblacion
#Ahora, en la columna "porcencasos", tenemos los casos por numero de habitantes en cada CCAA.
ggplot(data=C, aes(x=ccaa_iso, y=porcencasos,color=ccaa_iso)) +
  geom_bar(stat="identity", fill="white")+
  geom_text(aes(label=n), vjust=-0.3, size=2)+
  ggtitle("Casos COVID/población en 2021 por CCAA") +
  xlab("CCAA") + ylab("Casos de COVID/Población en 2021")
```




```
  theme_minimal()
```

```
## List of 93
##  $ line                       :List of 6
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                       :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                       :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                      : NULL
##  $ aspect.ratio               : NULL
##  $ axis.title                 : NULL
##  $ axis.title.x               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
```

```
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom       : NULL
##  $ axis.title.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left         : NULL
##  $ axis.title.y.right        :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
```

```
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom        : NULL
##  $ axis.text.y               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left          : NULL
##  $ axis.text.y.right         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
```

```
##  $ axis.ticks               : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x             : NULL
##  $ axis.ticks.x.top         : NULL
##  $ axis.ticks.x.bottom      : NULL
##  $ axis.ticks.y             : NULL
##  $ axis.ticks.y.left        : NULL
##  $ axis.ticks.y.right       : NULL
##  $ axis.ticks.length        : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x      : NULL
##  $ axis.ticks.length.x.top  : NULL
##  $ axis.ticks.length.x.bottom: NULL
##  $ axis.ticks.length.y      : NULL
##  $ axis.ticks.length.y.left  : NULL
##  $ axis.ticks.length.y.right : NULL
##  $ axis.line                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x              : NULL
##  $ axis.line.x.top          : NULL
##  $ axis.line.x.bottom       : NULL
##  $ axis.line.y              : NULL
##  $ axis.line.y.left         : NULL
##  $ axis.line.y.right        : NULL
##  $ legend.background        : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin            : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing           : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x         : NULL
##  $ legend.spacing.y         : NULL
##  $ legend.key               : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size          : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height        : NULL
##  $ legend.key.width         : NULL
##  $ legend.text              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.align        : NULL
##  $ legend.title             :List of 11
##   ..$ family       : NULL
```

```
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.align        : NULL
##  $ legend.position           : chr "right"
##  $ legend.direction          : NULL
##  $ legend.justification      : chr "center"
##  $ legend.box                : NULL
##  $ legend.box.just           : NULL
##  $ legend.box.margin         : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing        : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ panel.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.border              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.spacing             : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ panel.spacing.x           : NULL
##  $ panel.spacing.y           : NULL
##  $ panel.grid                :List of 6
##   ..$ colour       : chr "grey92"
##   ..$ size         : NULL
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major          : NULL
##  $ panel.grid.minor          :List of 6
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.5
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major.x        : NULL
##  $ panel.grid.major.y        : NULL
##  $ panel.grid.minor.x        : NULL
##  $ panel.grid.minor.y        : NULL
##  $ panel.ontop               : logi FALSE
##  $ plot.background           : list()
```

```
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ plot.title               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.title.position      : chr "panel"
##  $ plot.subtitle            :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption.position    : chr "panel"
##  $ plot.tag                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
```

```
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ strip.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ strip.background.x        : NULL
##  $ strip.background.y        : NULL
##  $ strip.placement           : chr "inside"
##  $ strip.text                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey10"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 4.4points 4.4points 4.4points 4.4points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.text.x              : NULL
##  $ strip.text.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.switch.pad.grid     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.switch.pad.wrap     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.text.y.left         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
```

```
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

En este caso, es la Comunidad Foral de Navarra la que ha tenido un mayor impacto de casos de COVID por número de habitantes. Luego podremos hacer análisis con las densidades poblacionales, puesto que uno de los ficheros cargados nos proporciona la densidad de habitantes por KM2, cosa que podría ser representativa.

Hacemos sacamos unos estadísticos por CCAA:

```
summary(C) #n es el número de casos totales.
```

```
##    ccaa_iso            densidad         poblacion             n
##  Length:19          Min.   :  25.34   Min.   :  83502   Min.   :   9318
##  Class :character   1st Qu.:  62.98   1st Qu.: 835397   1st Qu.: 108897
##  Mode  :character   Median : 109.74   Median :1513161   Median : 178714
##                     Mean   : 741.03   Mean   :2494433   Mean   : 331213
##                     3rd Qu.: 272.83   3rd Qu.:2542182   3rd Qu.: 377189
##                     Max.   :7001.58   Max.   :8501450   Max.   :1167453
##   porcencasos
##  Min.   :0.0664
##  1st Qu.:0.1091
##  Median :0.1245
##  Mean   :0.1316
##  3rd Qu.:0.1619
##  Max.   :0.1886
```

```
maxi <- max(C$n)
mini <- min(C$n)
promaxi <- C[C$n == maxi,]
print("La CC.AA con más casos de COVID en el año 2021 es:")
```

```
## [1] "La CC.AA con más casos de COVID en el año 2021 es:"
```

```
print(promaxi$ccaa_iso)
```

```
## [1] "CT"
```

```
print(promaxi$n)
```

```
## [1] 1167453
```

```
promini <- C[C$n == mini,]
print("La CC.AA con menos casos de COVID en el año 2021 es:")
```

```
## [1] "La CC.AA con menos casos de COVID en el año 2021 es:"
```

```
print(promini$ccaa_iso)
```

```
## [1] "CE"
```

```
print(promini$n)
```

```
## [1] 9318
```

Sacamos unos estadísticos por CCAA para los Casos Covid/Densidad de población:

```
summary(C) #n es el número de casos totales.
```

```
##    ccaa_iso            densidad         poblacion             n
##  Length:19          Min.   :  25.34   Min.   :  83502   Min.   :   9318
##  Class :character   1st Qu.:  62.98   1st Qu.: 835397   1st Qu.: 108897
##  Mode  :character   Median : 109.74   Median :1513161   Median : 178714
##                     Mean   : 741.03   Mean   :2494433   Mean   : 331213
##                     3rd Qu.: 272.83   3rd Qu.:2542182   3rd Qu.: 377189
##                     Max.   :7001.58   Max.   :8501450   Max.   :1167453
##   porcencasos
##  Min.   :0.0664
##  1st Qu.:0.1091
##  Median :0.1245
##  Mean   :0.1316
##  3rd Qu.:0.1619
##  Max.   :0.1886
```

```
maxi <- max(C$porcencasos)
mini <- min(C$porcencasos)
promaxi <- C[C$porcencasos == maxi,]
print("La CC.AA con más casos en % de COVID en el año 2021 es:")
```

```
## [1] "La CC.AA con más casos en % de COVID en el año 2021 es:"
```

```
print(promaxi$ccaa_iso)
```

```
## [1] "NC"
```

```
print(promaxi$porcencasos)
```

```
## [1] 0.1886144
```

```
promini <- C[C$porcencasos == mini,]
print("La CC.AA con menos casos en % COVID en el año 2021 es:")
```

```
## [1] "La CC.AA con menos casos en % COVID en el año 2021 es:"
```

```
print(promini$ccaa_iso)
```

```
## [1] "CN"
```

```
print(promini$porcencasos)
```

```
## [1] 0.06639791
```

La mayor incidencia de COVID en CC.AA por densidad de población se da en la Comunidad Foral de Nvarra, mientras que la de menor incidencia es la Comunidad de Canarias, casi una tercera parte.

2.Análisis de Regesión y Clustering

Ahora que tenemos la incidencia por porcentaje de habitantes por CCAA, pasamos a hacer un análisis de regresión y de clustering en base a las densidades de población de cada CCAA. Queda más que justificado que no debemos tomar el total de casos por CC.AA, si no su incidencia por numero de habitantes; esta será la variable a predecir, mientras que la densidad de cada CC.AA será la predictora, puesto que el aglomeramiento de personas bien pudiera tener un efecto de transmisión en la pandemia.

3.Regresión Lineal de porcentaje de casos frente a densidad de población en CC.AA.

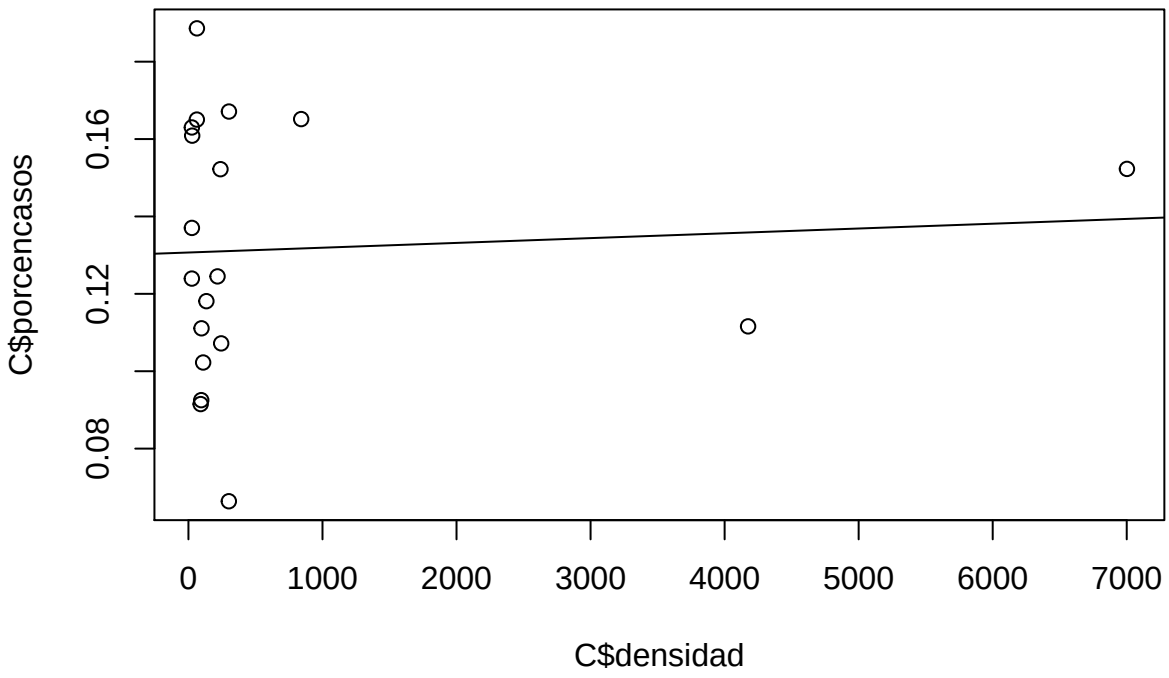
```
lmgrupoenfermo <- lm(C$porcencasos~C$densidad)
summary(lmgrupoenfermo)
```

```
##
## Call:
## lm(formula = C$porcencasos ~ C$densidad)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.064660 -0.024032 -0.006442  0.031222  0.057852
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.307e-01  8.417e-03  15.527 1.79e-11 ***
## C$densidad  1.240e-06  4.463e-06   0.278    0.785
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03374 on 17 degrees of freedom
## Multiple R-squared:  0.004519,   Adjusted R-squared:  -0.05404
## F-statistic: 0.07717 on 1 and 17 DF,  p-value: 0.7845
```

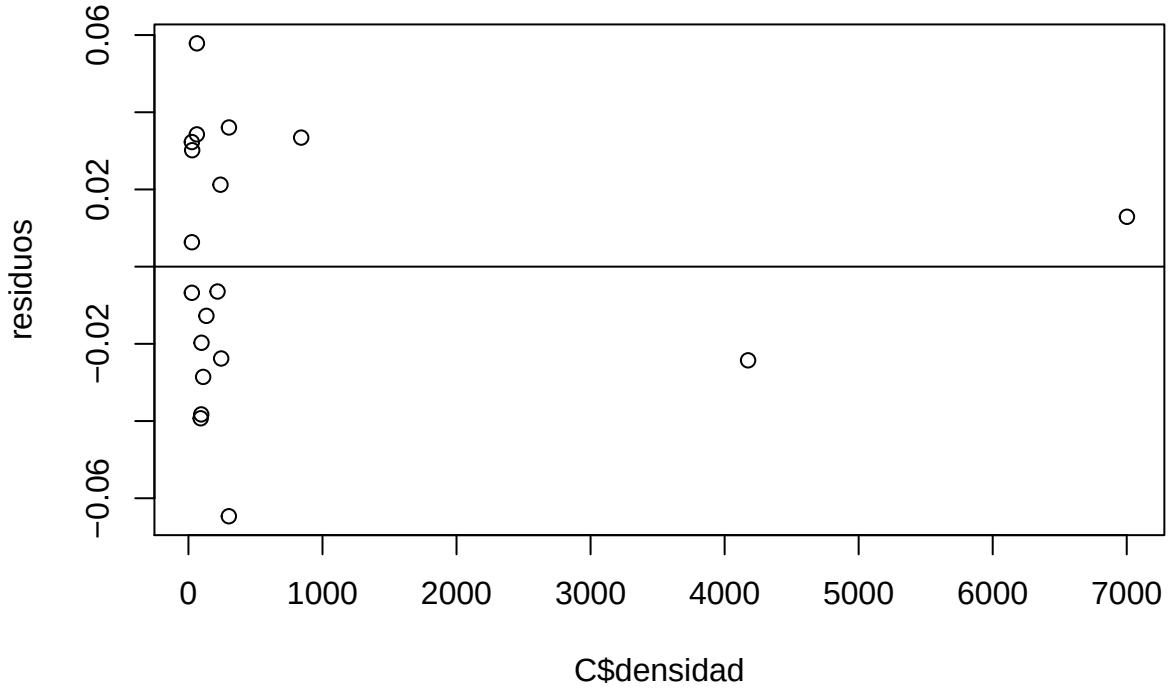
```
anova(lmgrupoenfermo)
```

```
## Analysis of Variance Table
##
## Response: C$porcencasos
##            Df    Sum Sq    Mean Sq F value Pr(>F)
## C$densidad  1 0.0000878 0.00008783  0.0772 0.7845
## Residuals  17 0.0193481 0.00113812
```

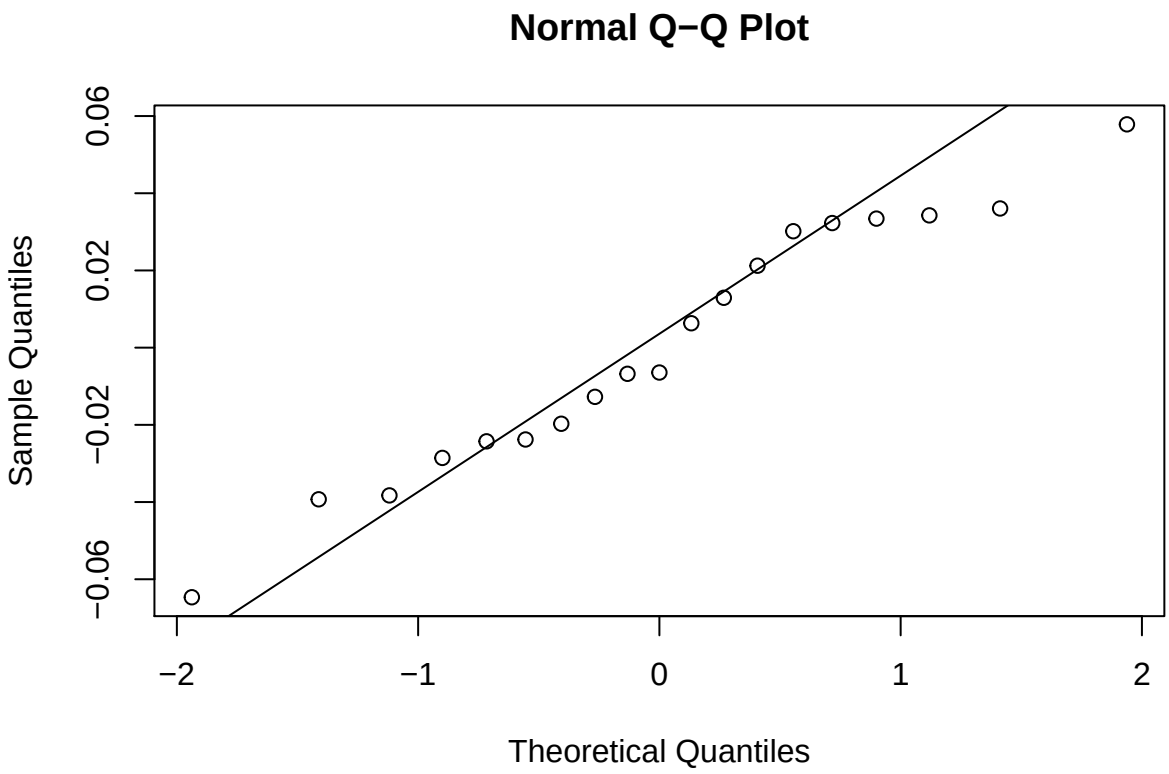
```
plot(C$densidad,C$porcencasos)
abline(lmgrupoenfermo)
```



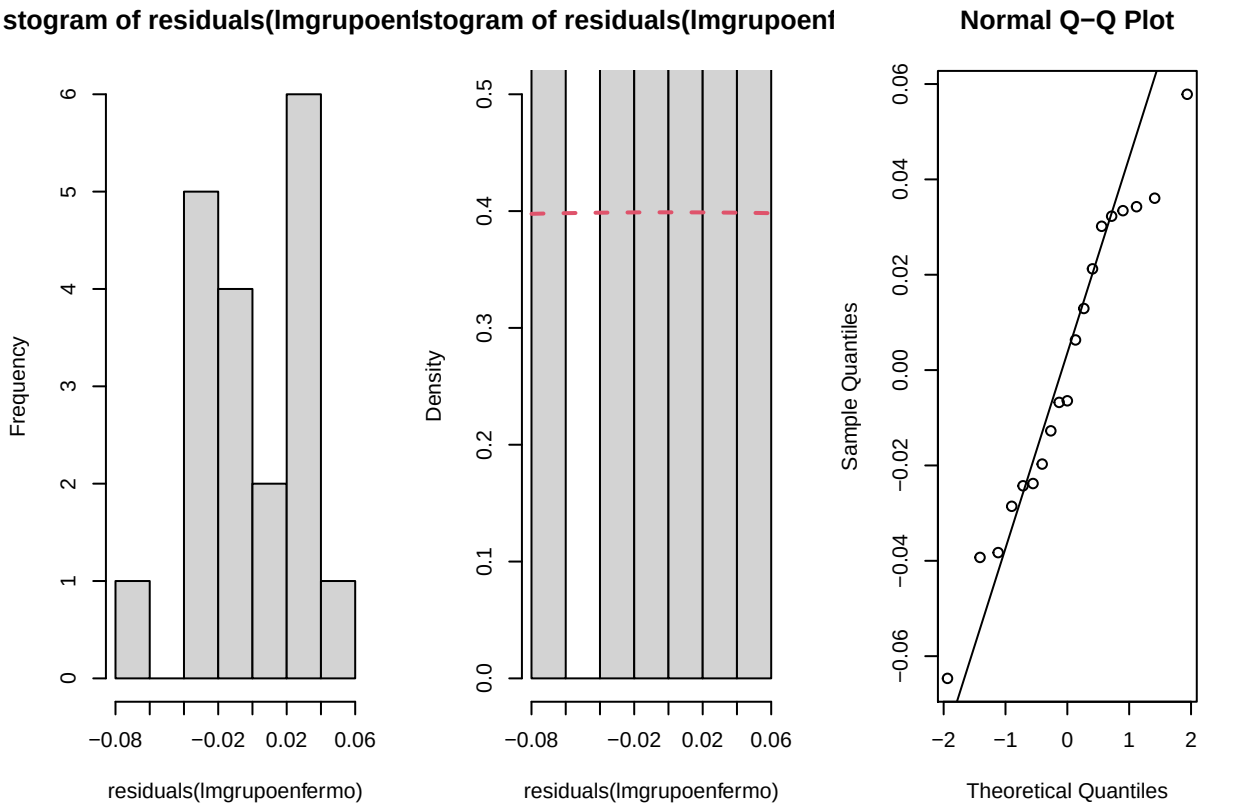

```
plot(C$densidad, residuals(lmgrupoenfermo), ylab = "residuos");abline(h = 0)
```




```
df_residuos <- data.frame(C$densidad, residuals(lmgrupoenfermo))
#View(df_residuos)
###pdf de web biblioteca
qqnorm(residuals(lmgrupoenfermo));qqline(residuals(lmgrupoenfermo))
```




```
par(mfcol=c(1,3))
hist(residuals(lmgrupoenfermo))
hist(residuals(lmgrupoenfermo),freq = F, ylim=c(0,0.5))
x <- rnorm(100,0,1)
curve(dnorm(x, 0,1), col = 2, lty = 2, lwd = 2, add = TRUE)
qqnorm(residuals(lmgrupoenfermo));qqline(residuals(lmgrupoenfermo))
```




```
dev.off()
```

```
## null device
##           1
```

```
ggplot(C,aes(x=densidad, y=porcencasos,  color=ccaa_iso)) +
  geom_point() +
  ggtitle("Evolución Casos COVID en 2021 para las CC.AA") +
  xlab("Densidad población") + ylab("Porcentaje de Casos de COVID por población")
```

4.Estudio de Normalidad: Test de Shapiro:

Vamos a hacer unas test de Shapiro para comprobar la normalidad de los datos. Se plantea como hipótesis nula que una muestra

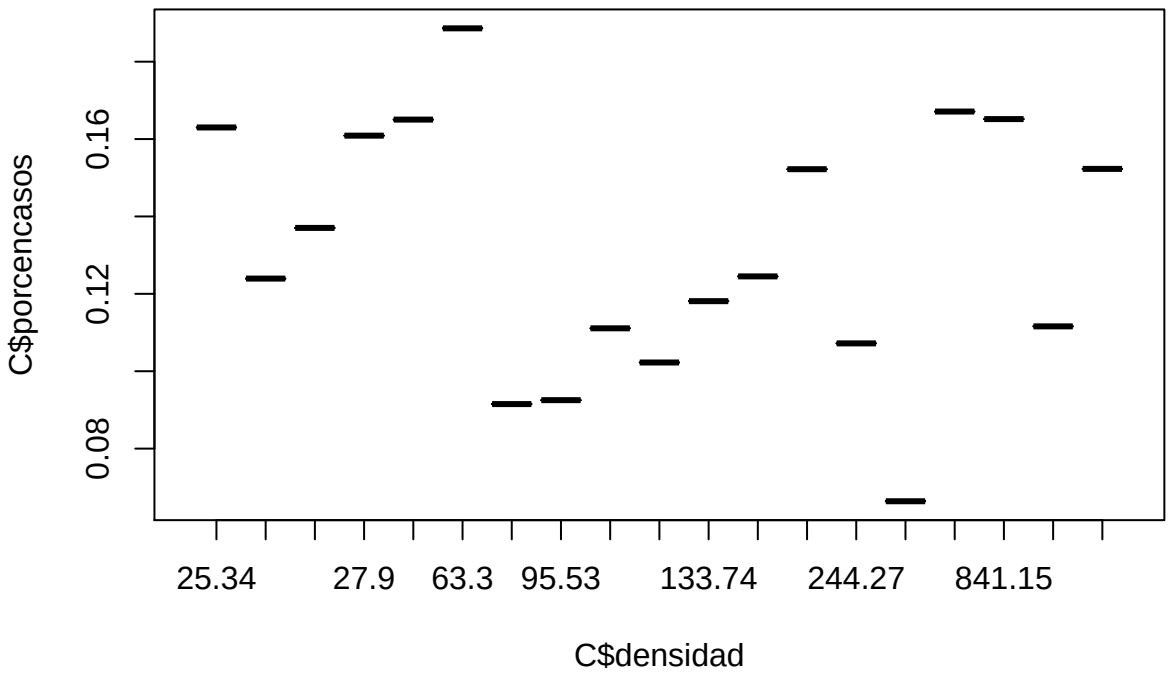
$$x_1, ..., x_n$$

proviene de una población normalmente distribuida.

```
shapiro.test(residuals(lmgrupoenfermo))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lmgrupoenfermo)
## W = 0.96319, p-value = 0.6367
```

```
boxplot(C$porcencasos~C$densidad)
```



Shapiro-Wilk normality test: La hipótesis nula se rechazará si W es demasiado pequeño. El valor de W puede oscilar entre 0 y 1. Podemos decir que existe normalidad en los datos o casos/población, tanto observando la W como el p-valor, que es mayor al nivel de significación de 0.05.

W = 0.96319, p-value = 0.6367

A pesar de que existe normalidad, no podemos decir que hay una relación lineal entre las variables de porcentajes de casos de COVID y densidad de población en CC.AA.

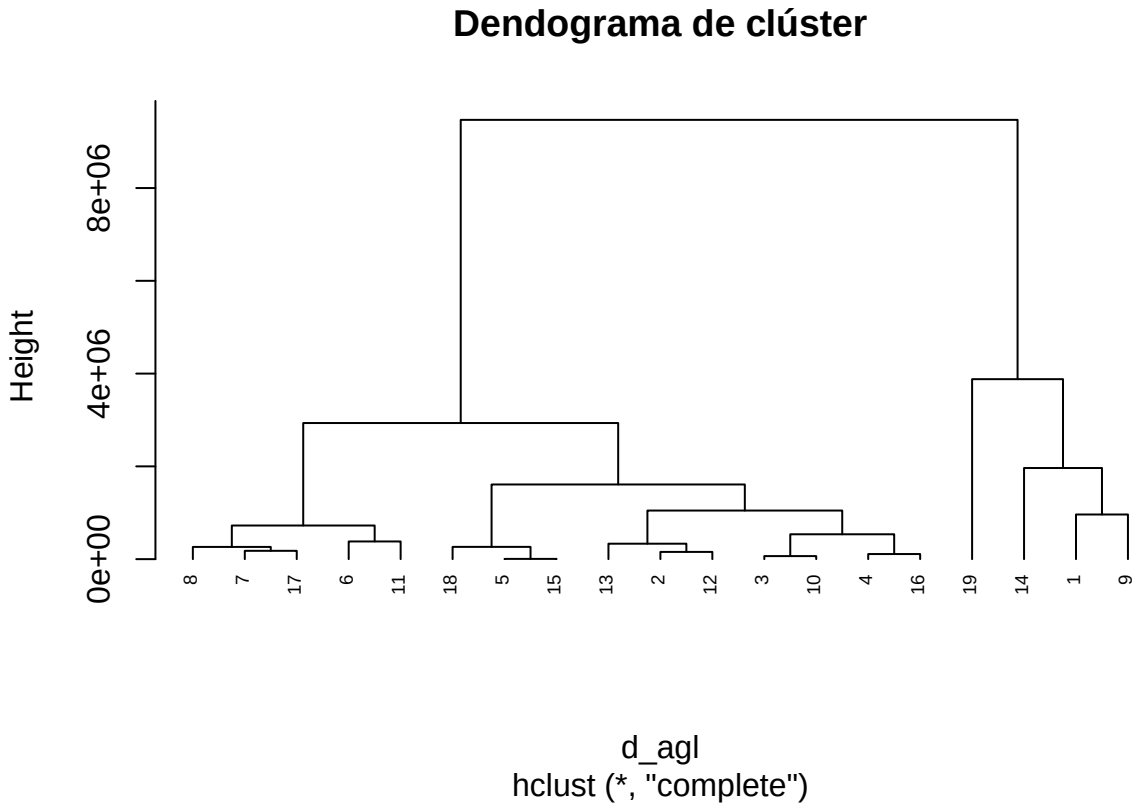
5.Estudio de Clustering:

A continuación veamos un ejemplo de agrupamiento jerárquico aglorematívo:

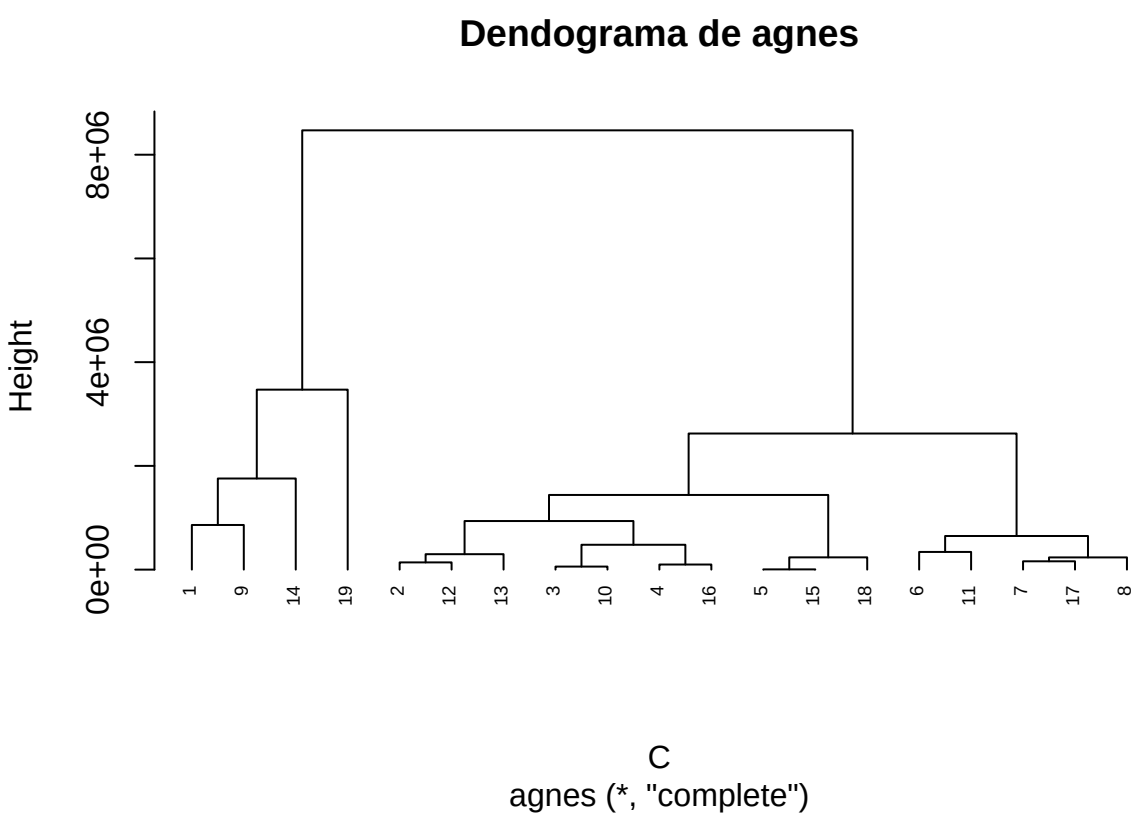
```
#Test de clusters**********************************************************************************
par(mfrow=c(1,1))
library("cluster")
library("factoextra")
d_agl<-dist(C,method="euclidean") ##especificamos los valores de distancia
hc_agl<-hclust(d_agl,method="complete") ##calculamos el clúster jerárquico
hc_agl
```

```
##
## Call:
## hclust(d = d_agl, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 19
```

```
plot(hc_agl,cex=0.6,hang=-1, main ="Dendograma de clúster")
```




```
##representamos el dendograma

#Función agnes y coeficiente de aglomeración del dendograma (mejor cuanto más
#se aproxima a 1)   que indica el grado de fortaleza del agrupamiento.
##Agrupamiento divisional jerárquico.
hc_ag<-agnes(C,method="complete")
##calculamos los clústeres jerárquicos
pltree(hc_ag,cex=0.6,hang=-1,main="Dendograma de agnes") ##representamos dendograma
```




```
#Coeficiente de aglomeración:
hc_ag$ac
```

```
## [1] 0.9420764
```

```
#El valor es: 0.9420764 lo que indica que la agrupación es fuerte.
```

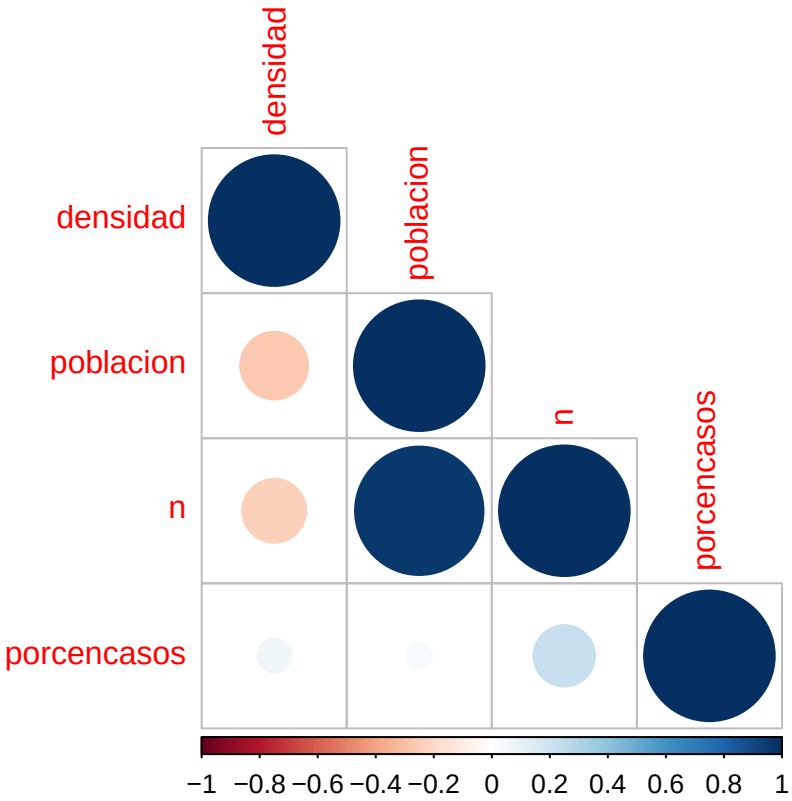
El Coeficiente de aglomeración sale de: 0.9420764, que es bastante alto. Observando los gráficos, el de Agnes sobretodo, podemos ver que cada CC.AA está agrupada correctamente en base a su población y número de casos, puesto que es lógico que por volumen de población se agrupen y, también, es de esperar que haya más casos cuanta más población exista en una comunidad autónoma. (La numeración en los dendogramas se corresponden con las líneas de la tabla C, con cada CC.AA y sus respectivos datos. Es una buena aglomeración o clustering, aunque era algo lógico y de esperar.)

Vamos a ver ahora un estudio de correlaciones:

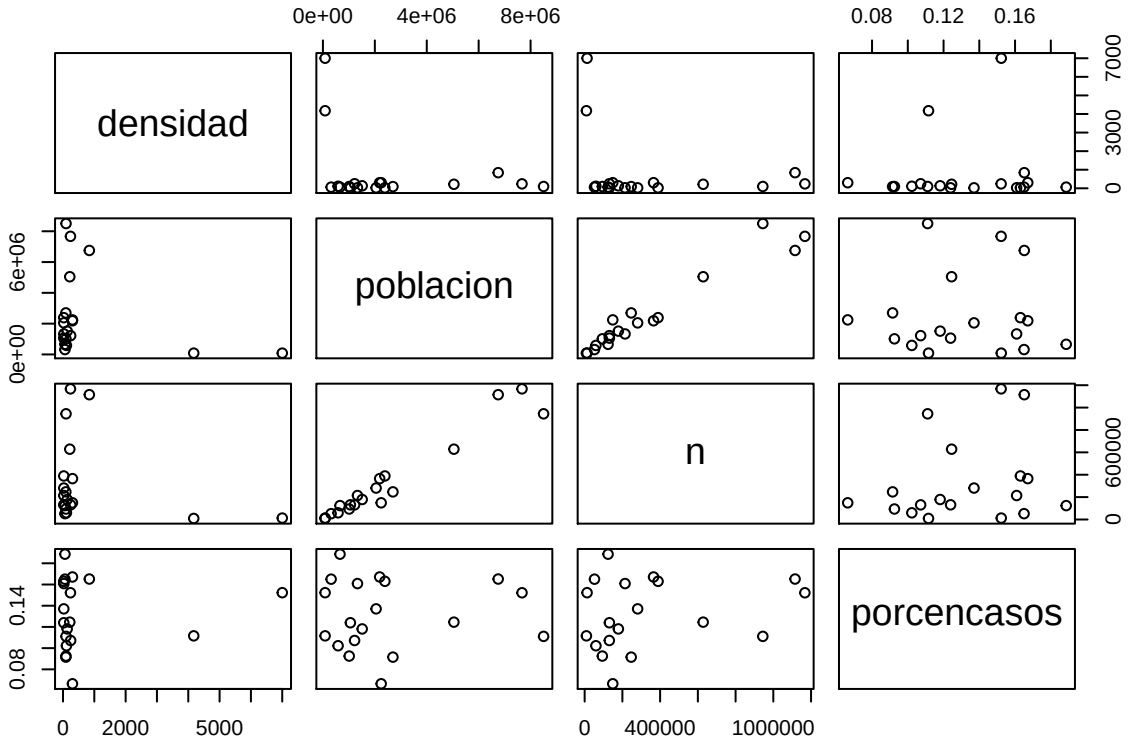
```
#Estudio de correlaciones entre casos y muertes:
require(corrplot)
round(cor(subset(C[,2:5]), method = "pearson"), digits = 3)
```

```
##             densidad poblacion      n porcencasos
## densidad       1.000    -0.269 -0.239       0.067
## poblacion     -0.269     1.000  0.967       0.037
## n             -0.239     0.967  1.000       0.222
## porcencasos    0.067     0.037  0.222       1.000
```

```
corrplot(round(cor(subset(C[,2:5])), digits = 3), type = "lower")
```


densidad
poblacion
n
porcencasos
densidad
poblacion
n
porcencasos
−1 −0.8 −0.6 −0.4 −0.2 0 0.2 0.4 0.6 0.8 1

```
pairs(C[,2:5])
```



Según el gráfico, la correlación más significativa es la de casos totales por población total en cada CC.AA, como es de esperar. El resto de correlaciones no son significativas.

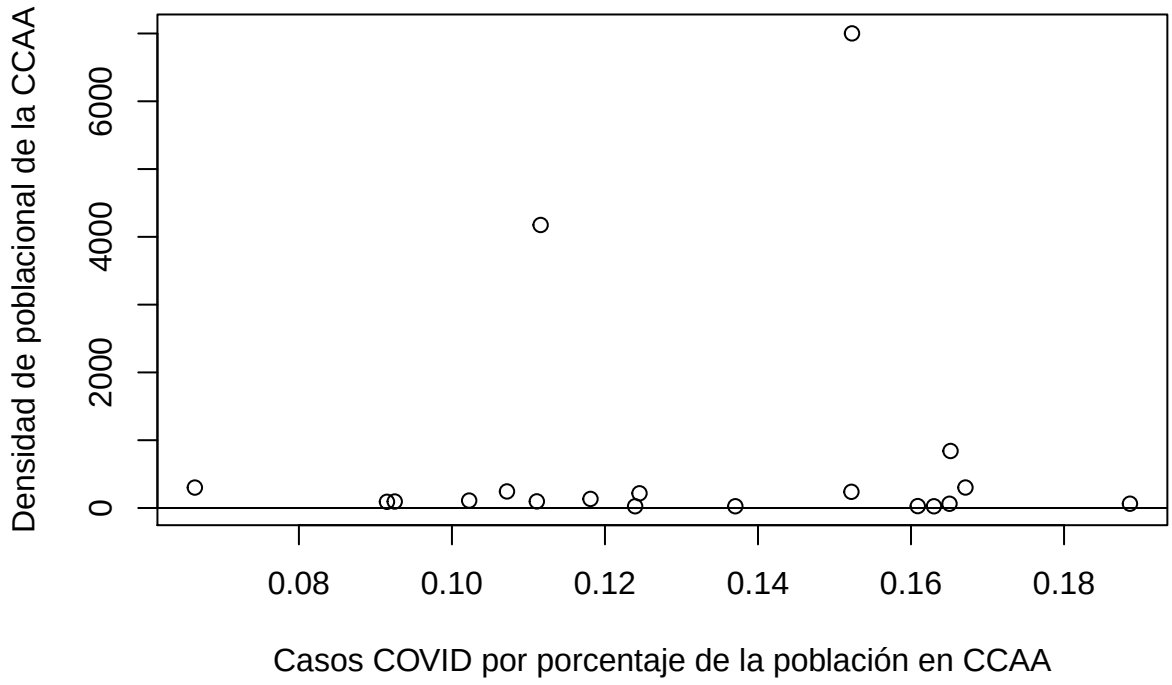
Volvemos de nuevo con la Regresiones Lineales, esta vez comparando los casos de COVID frente a la densidad de población y frente al número total de la población. De antemano, ya hemos visto anteriormente que con la densidad de población no hay relación lineal, pero veremos que sí la hay con el total de habitantes en cada CC.AA, como es de esperar:

```
#Modelo de regresión lineal
regl<-lm(C$porcencasos ~ C$densidad,data=C)
summary(regl)
```

```
##
## Call:
## lm(formula = C$porcencasos ~ C$densidad, data = C)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.064660 -0.024032 -0.006442  0.031222  0.057852
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.307e-01  8.417e-03  15.527 1.79e-11 ***
## C$densidad  1.240e-06  4.463e-06   0.278    0.785
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03374 on 17 degrees of freedom
```

```
## Multiple R-squared:  0.004519,   Adjusted R-squared:  -0.05404
## F-statistic: 0.07717 on 1 and 17 DF,  p-value: 0.7845
```

```
plot(C$porcencasos,C$densidad, xlab="Casos COVID por porcentaje de la población en CCAA"
     , ylab="Densidad de poblacional de la CCAA")
abline(regl)
```

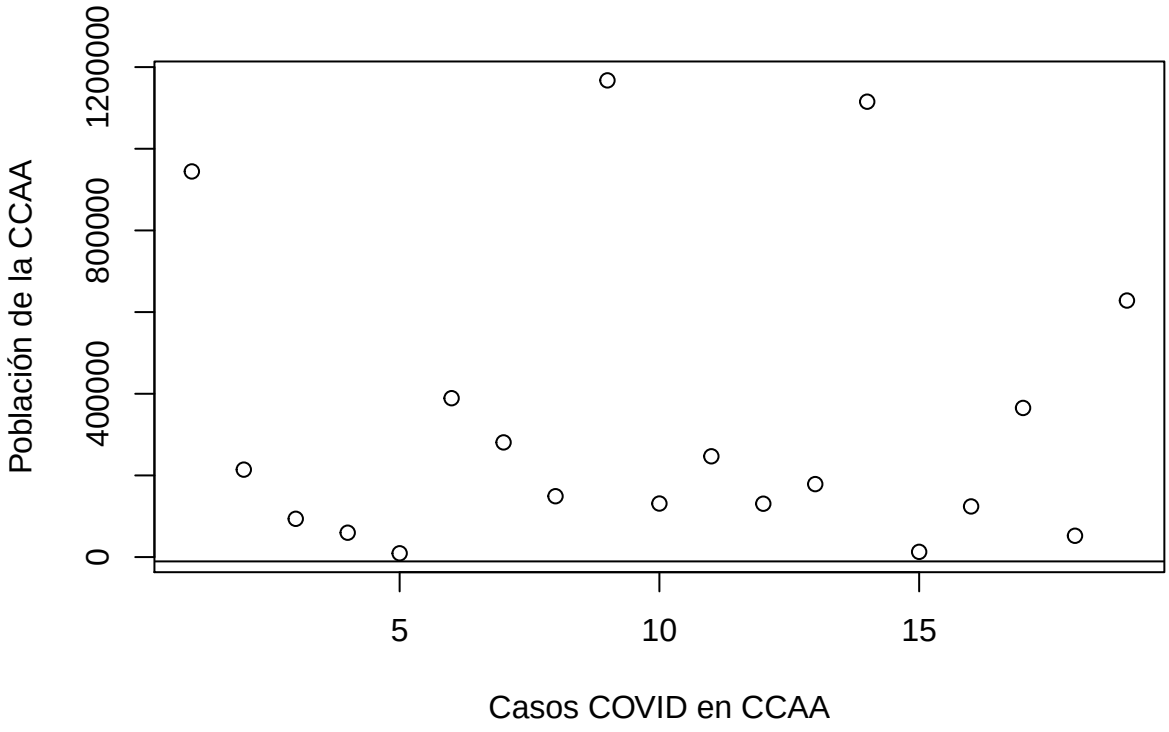



```
# Otra regresión n contra población:
regl<-lm(C$n ~ C$poblacion,data=C)
summary(regl)
```

```
##
## Call:
## lm(formula = C$n ~ C$poblacion, data = C)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -209759  -30170    8416   43404  200676
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.054e+04  3.095e+04  -0.341    0.738
## C$poblacion  1.370e-01  8.749e-03  15.659 1.56e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 95650 on 17 degrees of freedom
```

```
## Multiple R-squared:  0.9352, Adjusted R-squared:  0.9314
## F-statistic: 245.2 on 1 and 17 DF,  p-value: 1.561e-11
```

```
plot(C$n,C$población, xlab="Casos COVID en CCAA"
     , ylab="Población de la CCAA")
abline(regl)
```



Vemos que el modelo de casos para la población total de una CC.AA tiene un ajuste de

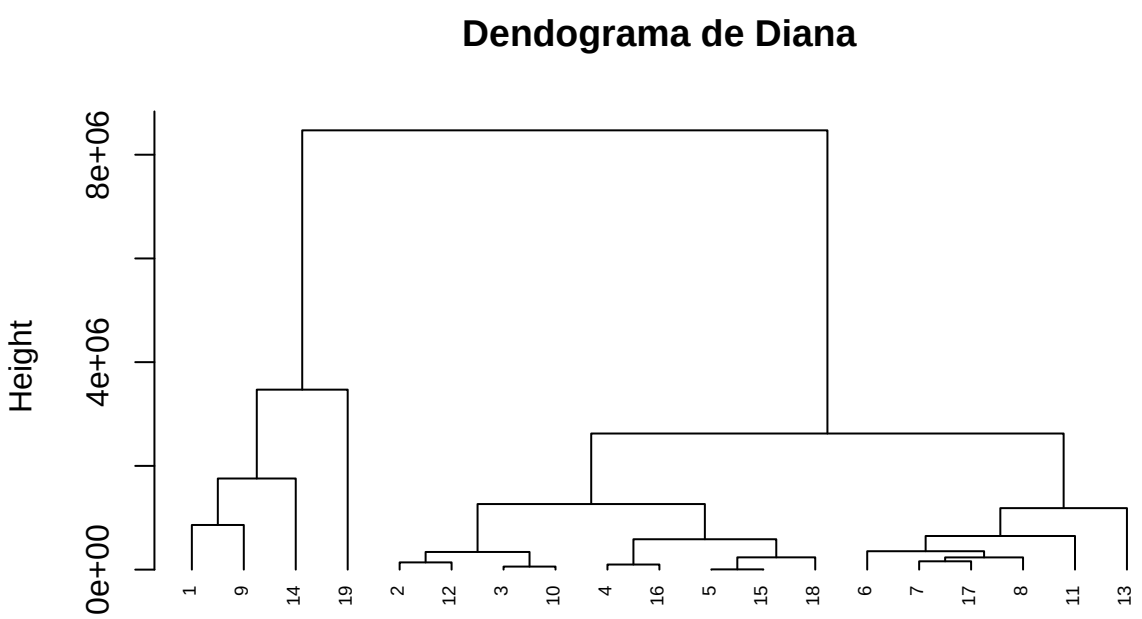
$$R^2 = 0.9314$$

, que es muy bueno. En el gráfico, salvo 4 CC.AA, el resto parecen alinearse bastante bien, pero estas 4 harán que la recta de regresión descienda sobre las que si se ajustan a ella.

6.Agrupamiento Divisional Jerárquico.

Veamos ahora un ejemplo de agrupamiento divisional jerárquico utilizando la función diana.

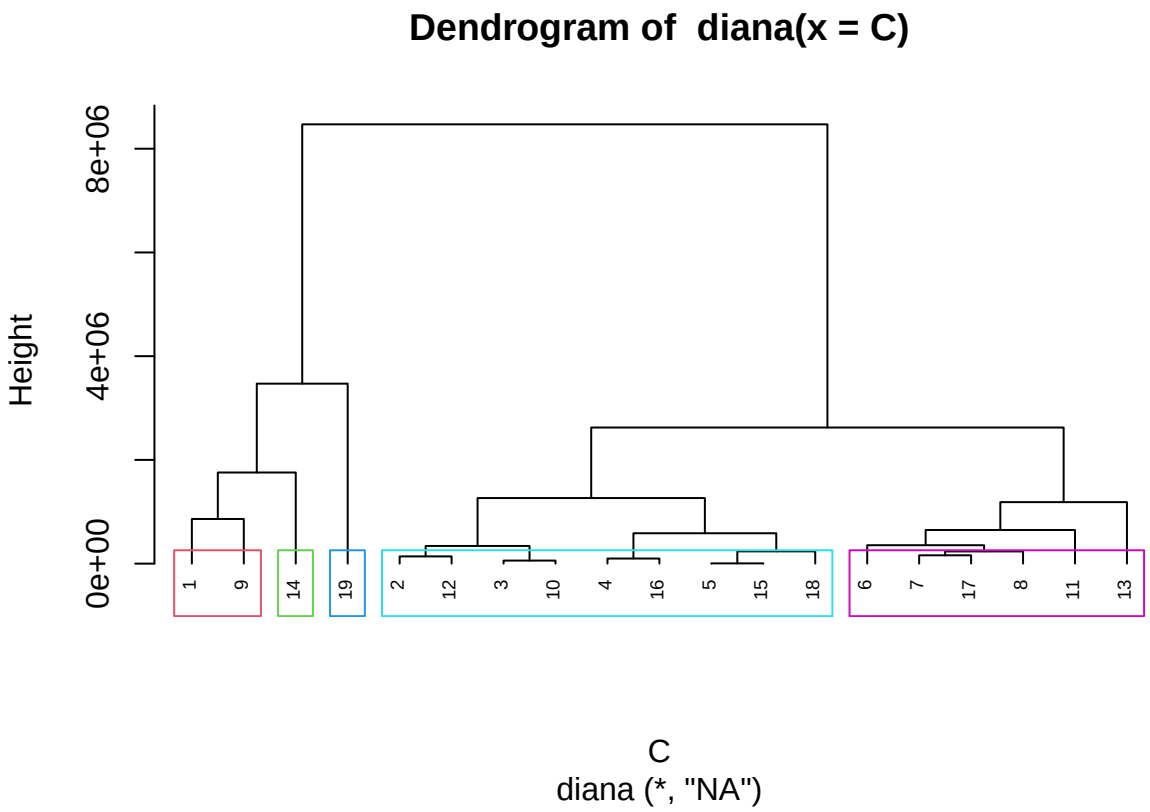
```
##Agrupamiento divisional jerárquico
par(mfrow=c(1,1))

hc_div<-diana(C) ##calculamos los clústeres jerárquicos
pltree(hc_div,cex=0.6,hang=-1,main="Dendograma de Diana") ##representamos el dendograma
```




```
hc_div$dc ##calculamos el coeficiente de división
```

```
## [1] 0.9345581
```

```
#El valor es:0.9345581 lo que indica que la agrupación es algo fuerte.
#Una vez creados los agrupamientos, asociamos los clústeres a las observaciones
#de datos.

pltree(hc_div,hang=-1,cex=0.6)
rect.hclust(hc_div,k=5,border=2:10)
```

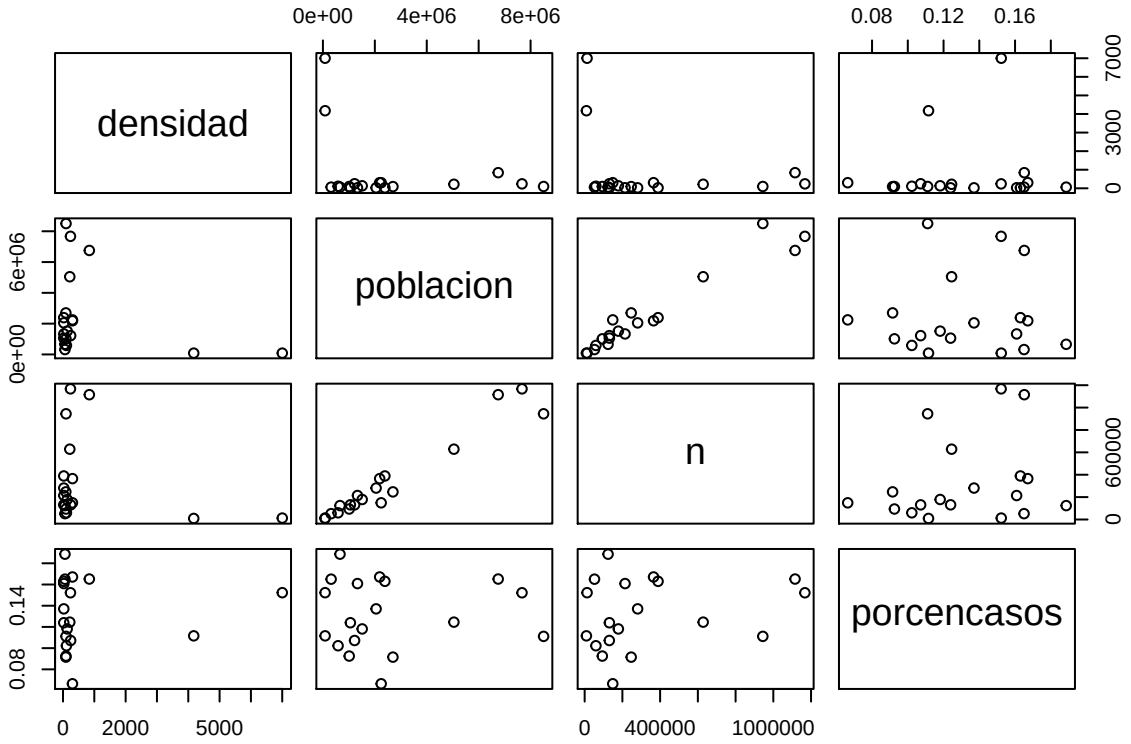


Tenemos un coeficiente de agrupamiento de 0.9345581 que es bastante alto.

7.Tests clustering jerárquicos.

Veremos cómo los métodos no jerárquicos organizan los elementos en grupos según un número de clústeres definido de antemano, creando grupos homogéneos y teniendo en cuenta que ningún elemento puede pertenecer a más de un grupo. Estos además serán heterogéneos entre ellos. Una de las técnicas más utilizadas de este tipo de método es la k-means, cuya idea es agrupar las observaciones en k clústeres distintos determinados de antemano. En nuestro caso y vistos los análisis anteriores de clustering, tomaremos 5 grupos:

```
##consideramos y nuestra variable objetivo ##realizamos un gráfico de
#dispersión según la variable predictora
pairs(C[,2:5])
```




```
corr=cor(C[-1])
corr
```

```
##                densidad   poblacion          n porcencasos
## densidad     1.00000000 -0.26942959 -0.2393978  0.06722405
## poblacion   -0.26942959  1.00000000  0.9670413  0.03664755
## n           -0.23939779  0.96704128  1.0000000  0.22186997
## porcencasos  0.06722405  0.03664755  0.2218700  1.00000000
```

```
##Construimos los grupos de clústeres, crearemos 5 grupos, puesto que en
#el caso anterior hemos visto que se agrupaban bastante bien
grupos_km=kmeans(C[,2:5],5) ##
grupos_km
```

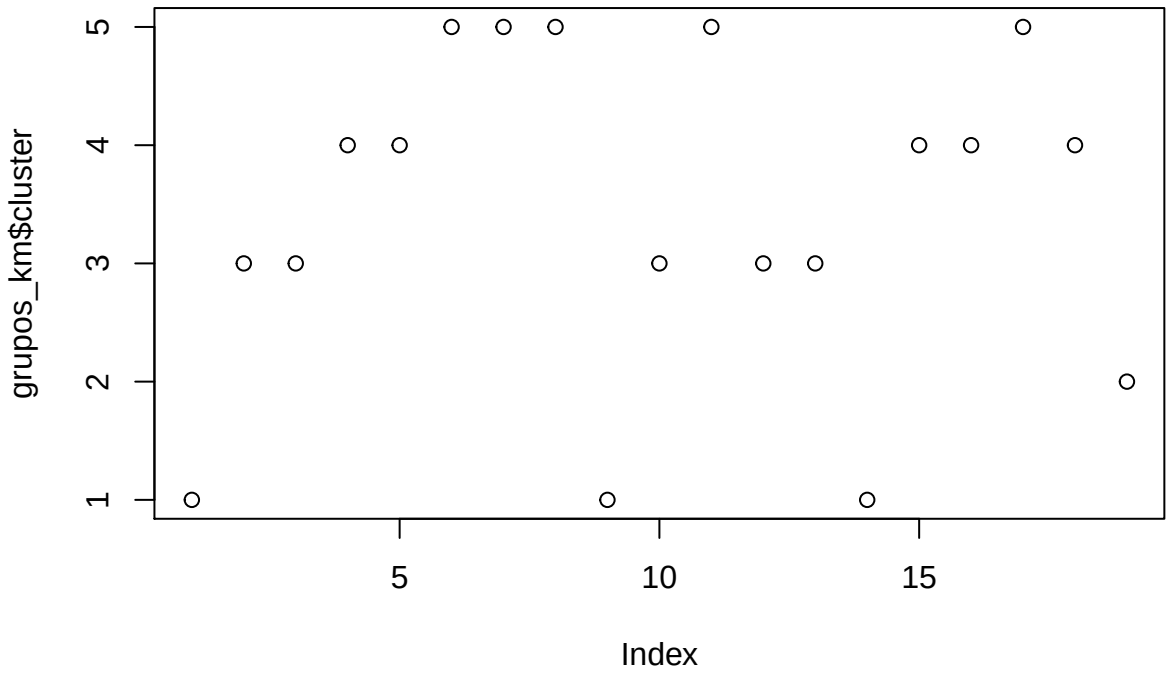
```
## K-means clustering with 5 clusters of sizes 3, 1, 5, 5, 5
##
## Cluster means:
##    densidad poblacion         n porcencasos
## 1  392.4967 7641404.0 1075736.3   0.1428220
## 2  216.9800 5045885.0  628266.0   0.1245106
## 3  105.3700 1226976.2  149692.2   0.1205287
## 4 2282.4780  345079.6   51613.4   0.1439549
## 5  149.1680 2312769.6  286210.0   0.1250106
##
## Clustering vector:
##  [1] 1 3 3 4 4 5 5 5 1 3 5 3 3 1 4 4 5 4 2
##
```

```
## Within cluster sum of squares by cluster:
## [1] 1.557393e+12 0.000000e+00 1.760069e+11 3.008772e+11 2.806127e+11
##  (between_SS / total_SS =  98.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
grupos_km$cluster ##grupo al cual pertenecen los datos
```

```
##  [1] 1 3 3 4 4 5 5 5 1 3 5 3 3 1 4 4 5 4 2
```

```
plot(grupos_km$cluster)
```



Hechos estos análisis previos sobre CC.AA, vamos ahora a proceder a un análisis sobre las provincias.

8.Análisis de casos de COVID por provincias y Grupos de EDAD.

```
# Enlace población provincias españa fichero csv: https://www.ine.es/jaxiT3/Tabla.htm?t=2852&L=0

#Ahora vamos a ver los casos totales de positivos por provincia a fecha de ¡Hoy!:

prov = c()
totcasos = c()
#Creamos un vector de provincias.
provincias <- unique(datosprov$provincia_iso)
```

```
for (i in (1:length(provincias))) {
  #print(provincias[i])
  # Sumamos para la provincia en todo el Dataframe que está listado por días.
  tot <- sum(datosprov$num_casos[datosprov$provincia_iso == provincias[i]])
  prov <- rbind(prov, provincias[i])
  totcasos <- rbind(totcasos,tot)
}
totcasos <- cbind(totcasos,prov)
totcasos <- as.data.frame(totcasos)
totcasos$V1 <- as.numeric(totcasos$V1)
m = max(totcasos$V1)
n = min(totcasos$V1)

p <- totcasos$V2[totcasos$V1 == m]
print('La provincia de España con más casos de COVID es:')
```

```
## [1] "La provincia de España con más casos de COVID es:"
```

```
print(p)
```

```
## [1] "M"
```

```
print(m)
```

```
## [1] 1115305
```

```
q <- totcasos$V2[totcasos$V1 == n]
print('La provincia de España con menos casos de COVID es:')
```

```
## [1] "La provincia de España con menos casos de COVID es:"
```

```
print(q)
```

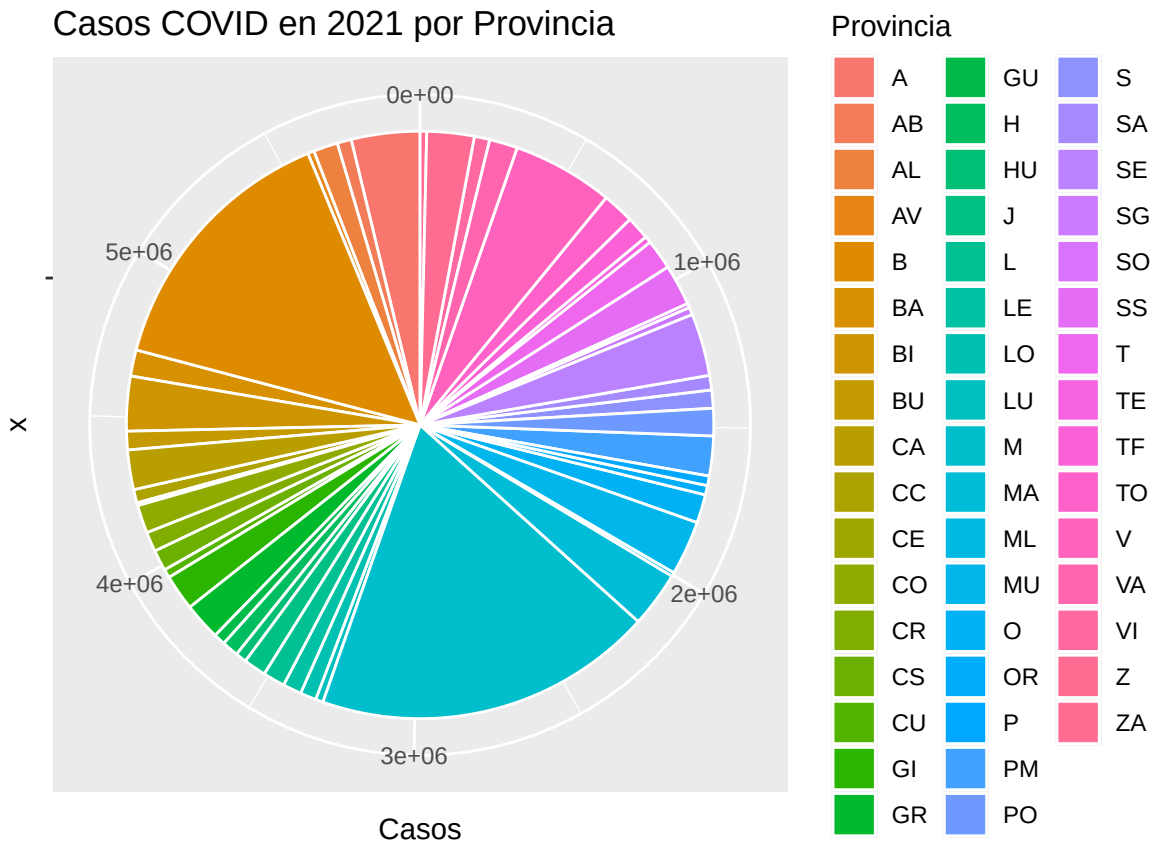
```
## [1] "CE"
```

```
print(n)
```

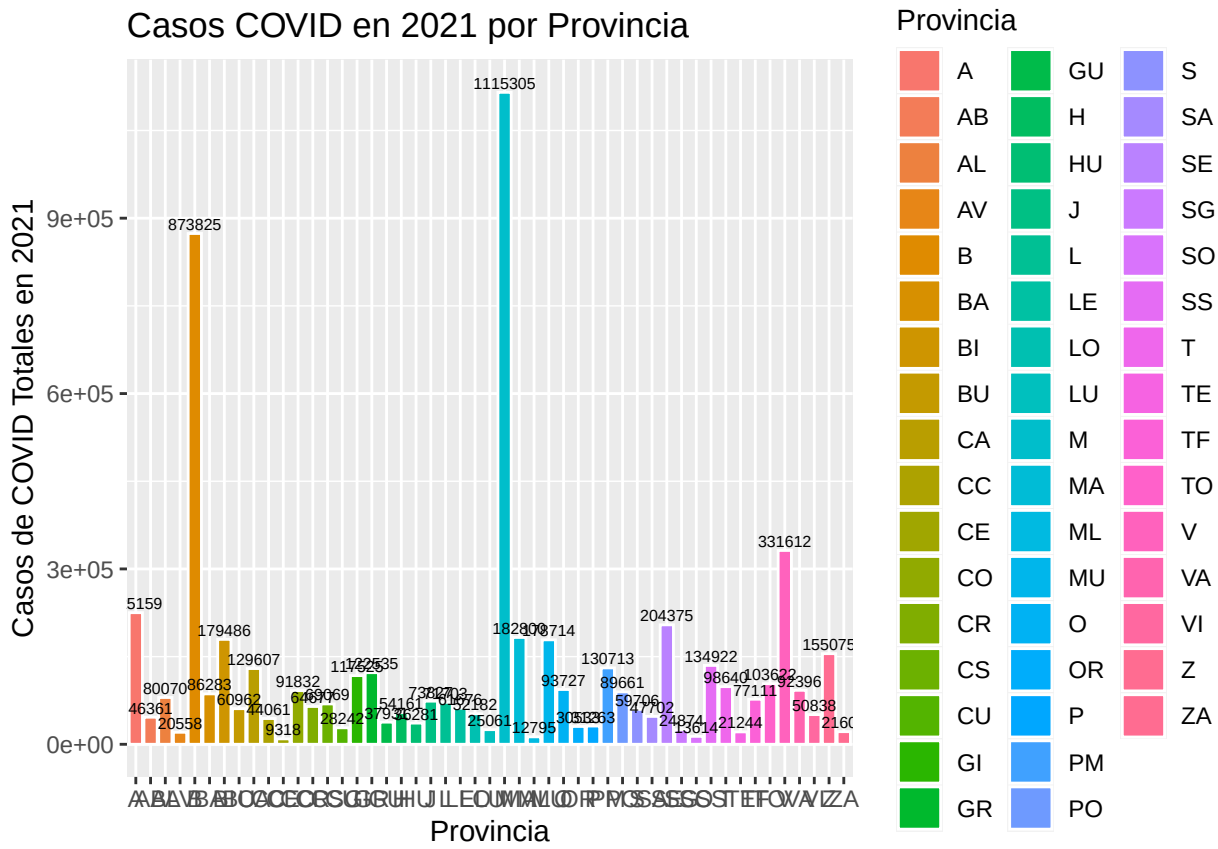
```
## [1] 9318
```

Vamos a realizar unos gráficos de los casos de COVID por provincias:

```
#library(ggplot2)
ggplot(totcasosprov,aes(x="",y=Casos, fill=Provincia))+
  geom_bar(stat = "identity",color="white")+
  coord_polar(theta="y")+
  ggtitle("Casos COVID en 2021 por Provincia")
```

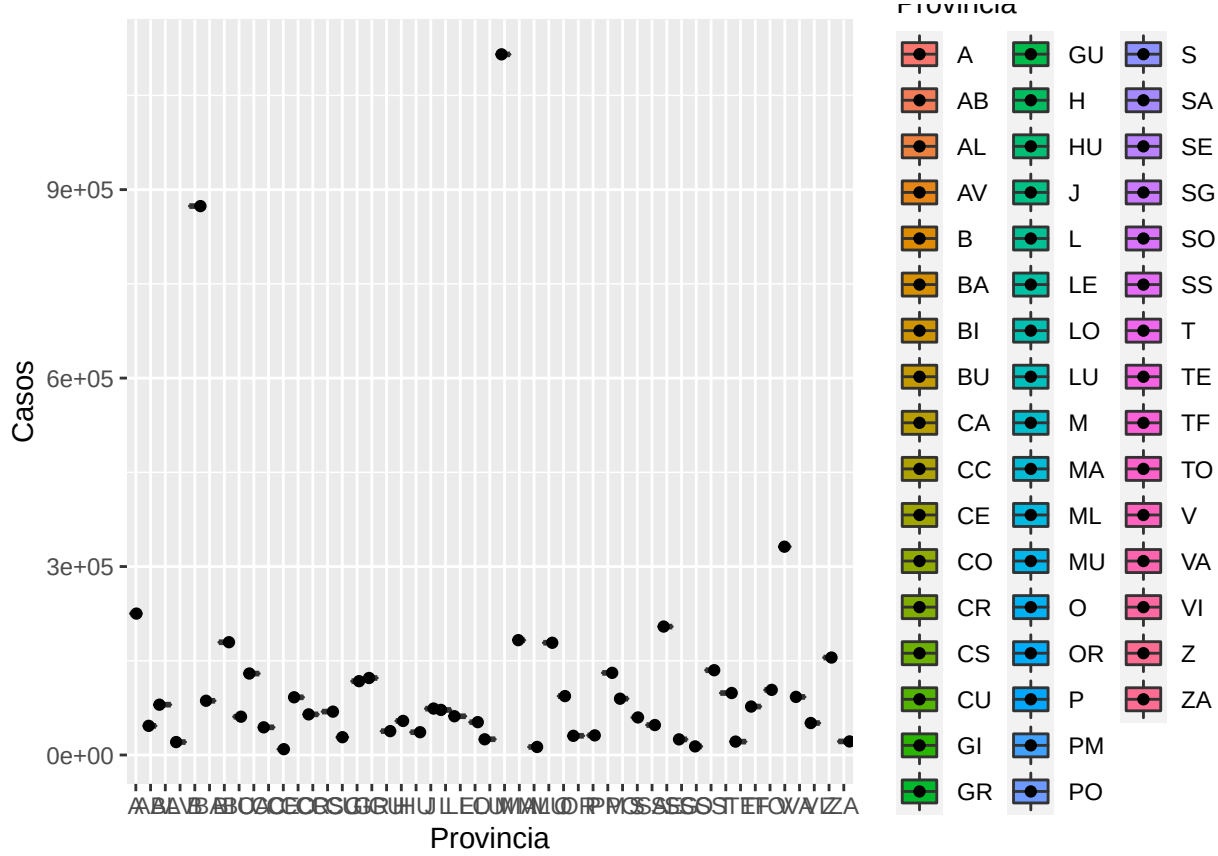



```
### Gráfico Barplot:
ggplot(totcasosprov,aes(x=Provincia,y=Casos, fill=Provincia)) +
  geom_bar(stat = "identity",
           color="white") +
  ggtitle("Casos COVID en 2021 por Provincia") +
  geom_text(aes(label=Casos), vjust=-0.3, size=2) +
  xlab("Provincia") + ylab("Casos de COVID Totales en 2021")
```

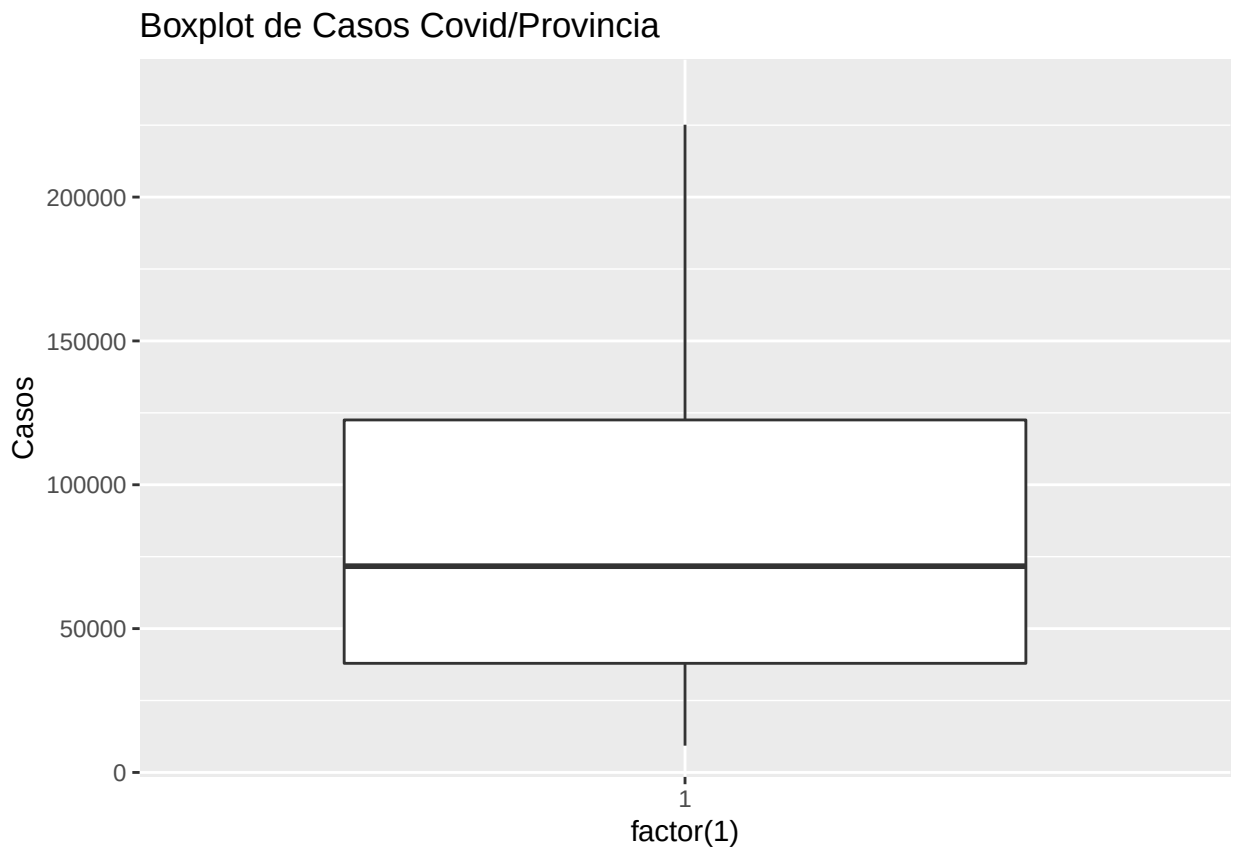


Vamos a hacer un gráfico Boxplot para ver cómo se distribuyen los casos totales por provincia:

```
#### Gráfico Boxplot:
# boxplot con puntos
ggplot(data = totcasosprov,
       mapping = aes(x = Provincia,
                     y = Casos,
                     fill = Provincia)) +
  geom_boxplot() +
  geom_jitter()
```




```
###### hacemos bien el boxplot #https://www.it-swarm-es.com/es/r/ignorar-los-valores-atipicos-en-ggplot
# compute lower and upper whiskers
ylim1 = boxplot.stats(totcasosprov$Casos)$stats[c(1, 5)]
p0 = ggplot(totcasosprov, aes(y = Casos )) + geom_boxplot(aes(x = factor(1))) +
  ggtitle("Boxplot de Casos Covid/Provincia")
# scale y limits based on ylim1
p1 = p0 + coord_cartesian(ylim = ylim1*1.05)
p1
```



En el Boxplot podemos ver dos bigotes, el de arriba bastante más largo y alejado de la media, que correspondería a Madrid.

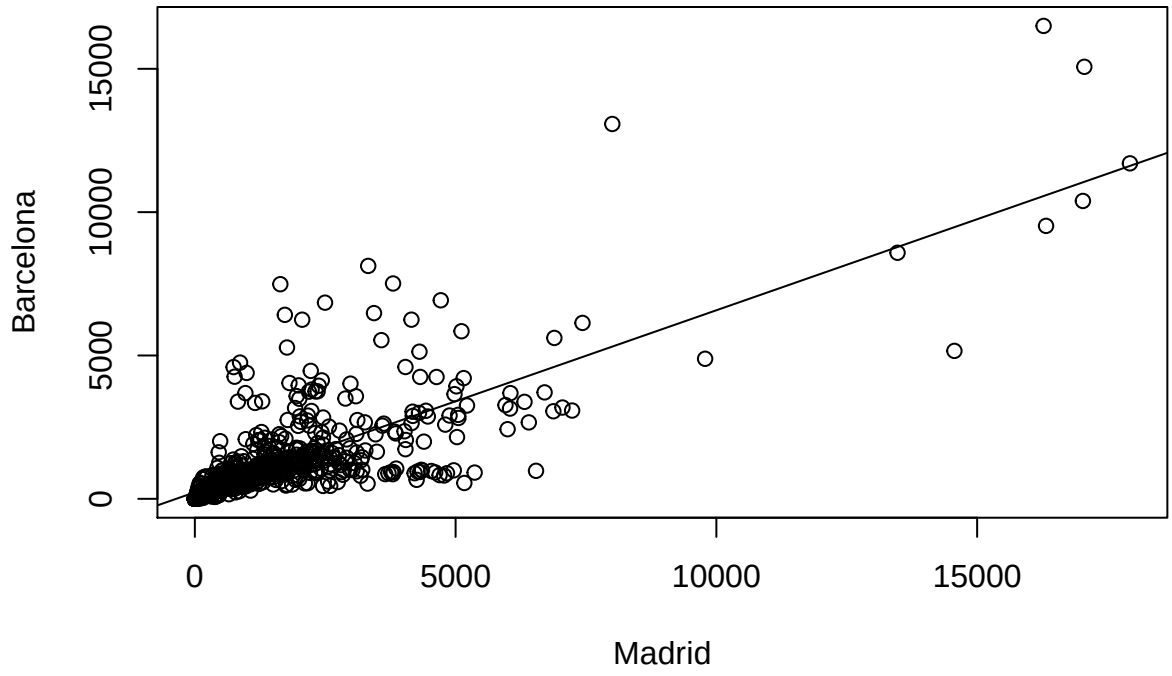
Vamos a estudiar algunas poblaciones concretas y a compraralas (Barcelona, Madrid y Zaragoza) haciendo algunas regresiones lineales entre los casos, por fecha, de las mencionadas provincias:

```
# Cargamos en dataframes laz ciudades de Barcelona, Madrid y zaragoza
Barcelona <- datosprov[datosprov$provincia_iso=='B',]
Madrid <- datosprov[datosprov$provincia_iso=='M',]
Zaragoza<- datosprov[datosprov$provincia_iso=='Z',]


tabla = data.frame(Barcelona = Barcelona$num_casos,Madrid = Madrid$num_casos)
RegMode <-lm(tabla$Barcelona~tabla$Madrid, data=tabla)
summary(RegMode)
```

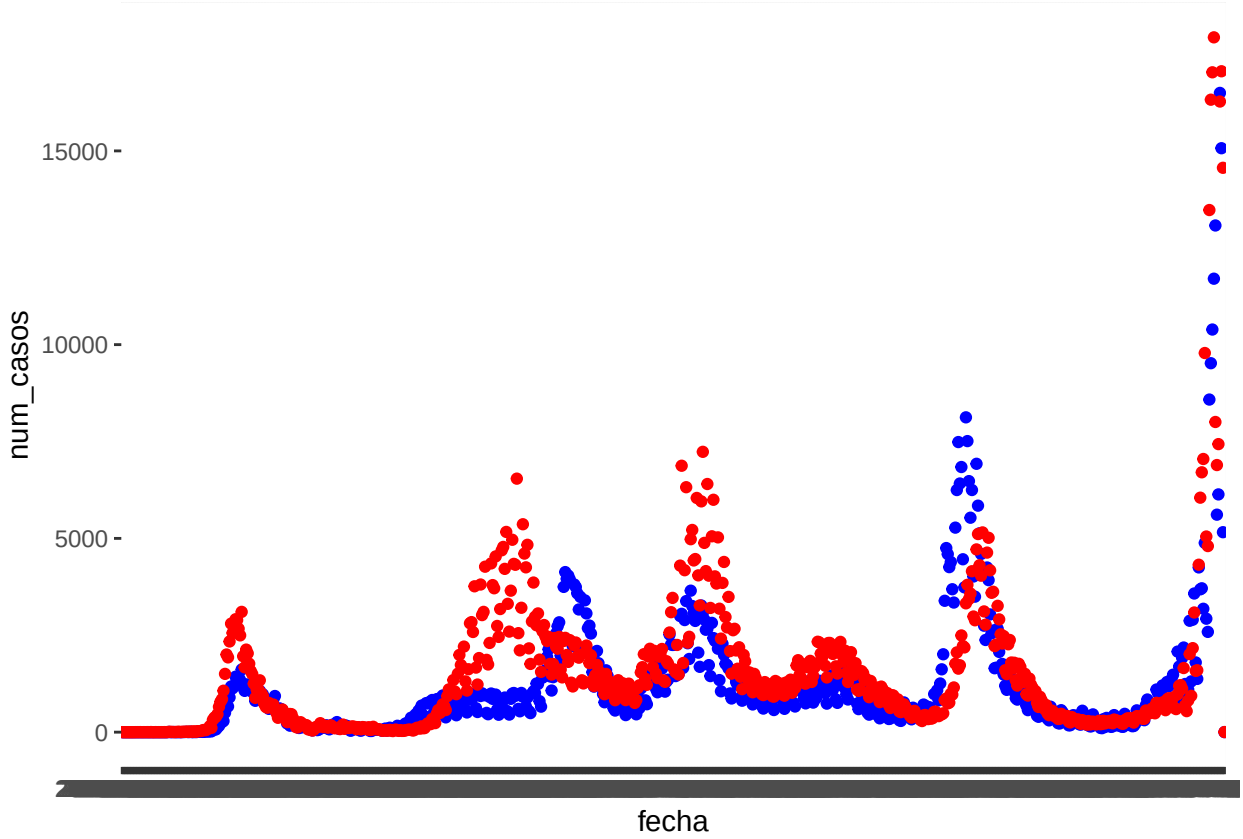
```
##
## Call:
## lm(formula = tabla$Barcelona ~ tabla$Madrid, data = tabla)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4315.7  -251.8  -156.8    68.0  7765.2
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  226.77405   47.47341   4.777 2.15e-06 ***
```

```
## tabla$Madrid   0.63505    0.01859  34.157  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1028 on 728 degrees of freedom
## Multiple R-squared:  0.6158, Adjusted R-squared:  0.6152
## F-statistic:  1167 on 1 and 728 DF,  p-value: < 2.2e-16
```

```
plot(tabla$Madrid,tabla$Barcelona, xlab="Madrid", ylab="Barcelona")
abline(RegMode)
```




```
# Estudiar las correlaciones ***** otras provincias y/o métodos de test ***********


######
ggplot() +
        geom_point(data=datprov[datprov$provincia_iso == 'B',],aes(x=fecha, y=num_casos),color='Blue')
  geom_point(data=datprov[datprov$provincia_iso == 'M',],aes(x=fecha, y=num_casos),color='Red')
```



Atendiendo a

$$R^2$$

y al p-valor, podemos ver que no existe una relación lineal entre los casos de COVID por día entre Barcelona y Madrid.

9.Análisis más detallado de casos de COVID por provincia, grupos de edad y sexo

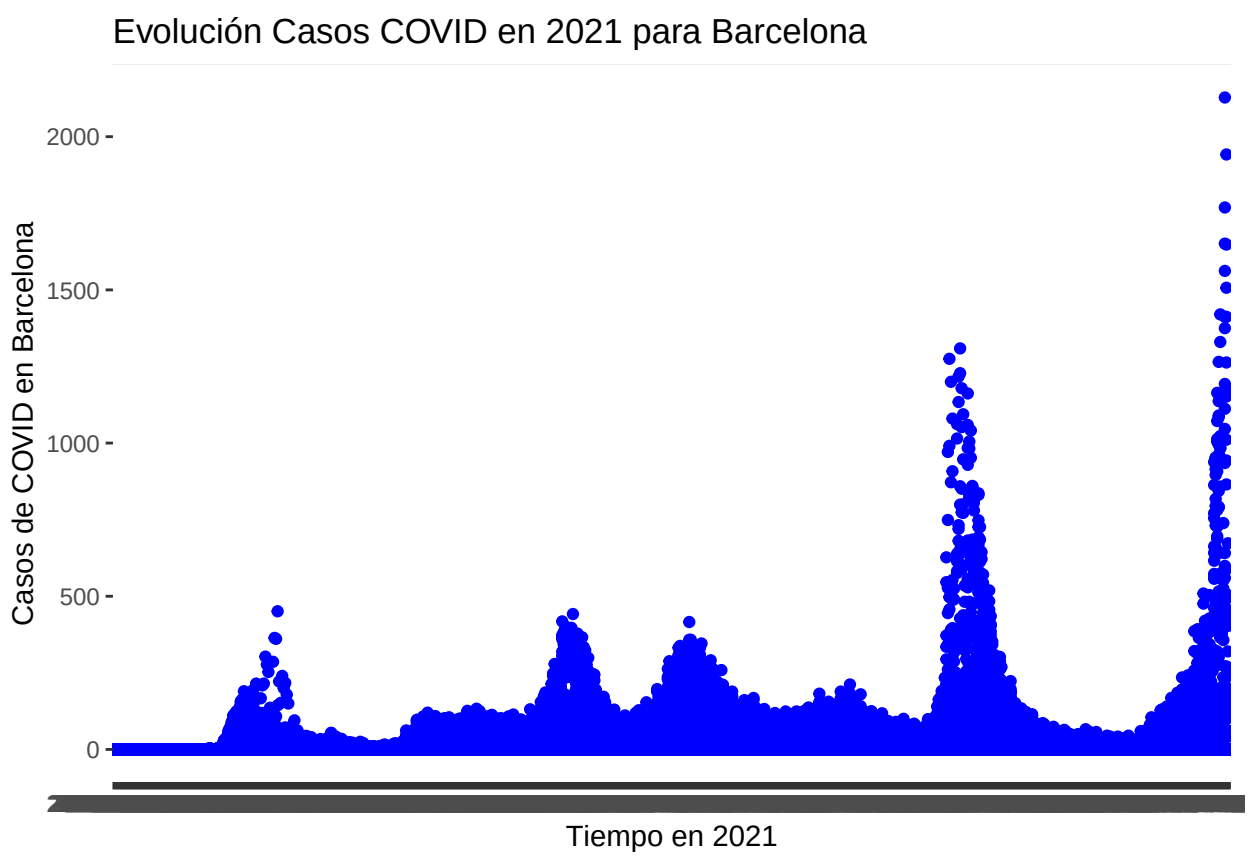
Ahora vamos a realizar un estudio de la pandemia, con datos del 2021 y de la misma fuente (INE), en donde tendremos en cuenta el sexo y el grupo de edad, de 10 en 10, partiendo de 0 hasta 79 y +80 años. Para ellos cargaremos el archivo csv con los datos pertinentes:

```
######################################################################
# Cargamos el fichero detallado de covid provincias de España:
datoscovid <- read.csv("casossexouci.csv")
datoscovid <- na.omit(datoscovid)

str(datoscovid)
```

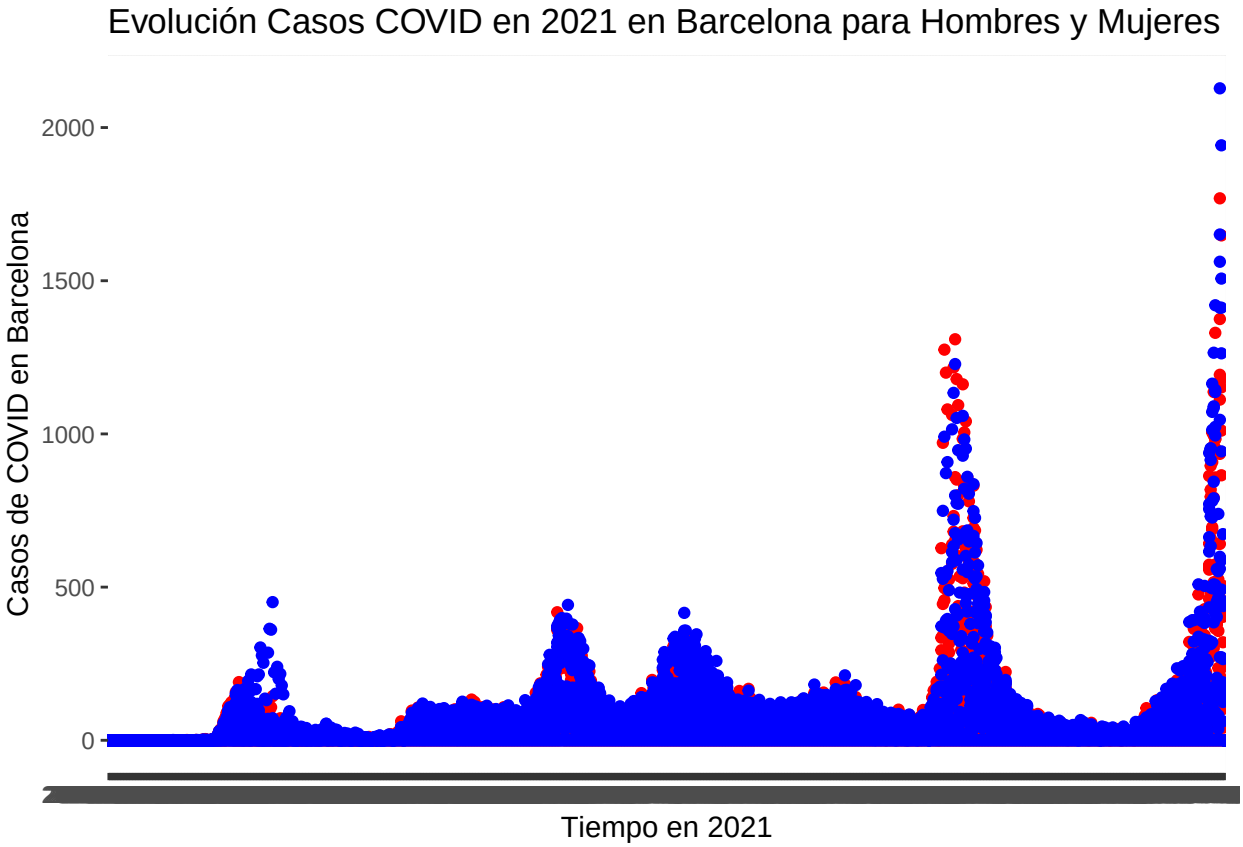
```
## 'data.frame':    1138800 obs. of  8 variables:
##  $ provincia_iso: chr  "A" "A" "A" "A" ...
##  $ sexo         : chr  "H" "H" "H" "H" ...
##  $ grupo_edad   : chr  "0-9" "10-19" "20-29" "30-39" ...
##  $ fecha        : chr  "2020-01-01" "2020-01-01" "2020-01-01" "2020-01-01" ...
##  $ num_casos    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_hosp     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_uci      : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ num_def      : int  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "na.action")= 'omit' Named int [1:21900] 931 932 933 934 935 936 937 938 939 940 ...
##   ..- attr(*, "names")= chr [1:21900] "931" "932" "933" "934" ...
```

```
#Grafico los casos en el tiempo para Barcelona:
ggplot() +
  geom_point(data=datoscovid[datoscovid$provincia_iso == 'B',],aes(x=fecha, y=num_casos),color='Blue') +
  ggtitle("Evolución Casos COVID en 2021 para Barcelona") +
  xlab("Tiempo en 2021") + ylab("Casos de COVID en Barcelona")
```




```
# Ahora graficamos los casos de Barcelona para Hombres y Mujeres:
ggplot() +
  geom_point(data=datoscovid[datoscovid$provincia_iso == 'B' & datoscovid$sexo == 'H',],aes(x=fecha, y=
  geom_point(data=datoscovid[datoscovid$provincia_iso == 'B' & datoscovid$sexo == 'M',],aes(x=fecha, y=
  ggtitle("Evolución Casos COVID en 2021 en Barcelona para Hombres y Mujeres") +
  xlab("Tiempo en 2021") + ylab("Casos de COVID en Barcelona")
```



Vemos que tanto mujeres como hombres siguen la misma gráfica, habiendo más casos de mujeres (azul). Ahora vamos a contar los casos totales de COVID en España por sexo:

```
xx <- sum(datoscovid$num_casos[datoscovid$sexo == 'M'])
xy <- sum(datoscovid$num_casos[datoscovid$sexo == 'H'])
print('El número de casos y % de COVID en 2021 para mujeres fue de:')
```

```
## [1] "El número de casos y % de COVID en 2021 para mujeres fue de:"
```

```
print(xx)
```

```
## [1] 3199373
```

```
print(100*xx/(xx+xy))
```

```
## [1] 51.93078
```

```
print('El número de casos y % de COVID en 2021 para hombres fue de:')
```

```
## [1] "El número de casos y % de COVID en 2021 para hombres fue de:"
```

```
print(xy)
```

```
## [1] 2961468
```

```
print(100*xy/(xx+xy))
```

```
## [1] 48.06922
```

```
print('Hay una diferencia entre mujeres y hombres de:')
```

```
## [1] "Hay una diferencia entre mujeres y hombres de:"
```

```
print(xx-xy)
```

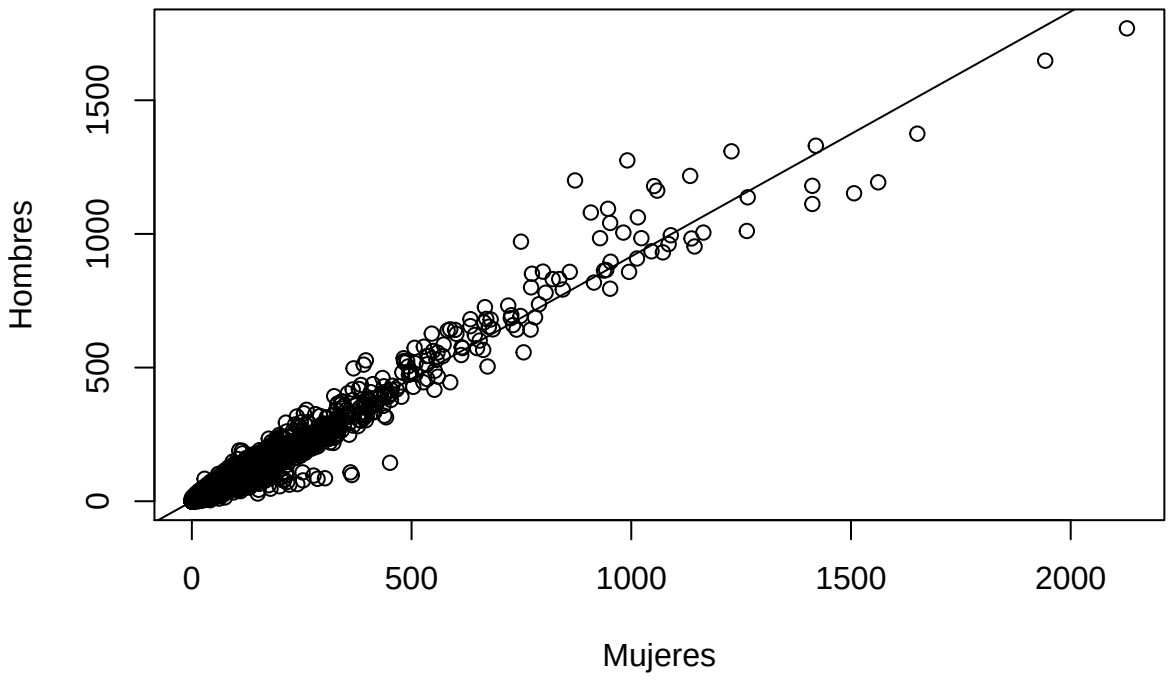
```
## [1] 237905
```

Ahora vamos a hacer unas regresiones lineales de casos de COVID sobre 2 provincias concretas (Barcelona y Madrid) para hombres y mujeres:

```
Barcelona <- datoscovid[datoscovid$provincia_iso=='B',]
Madrid <- datoscovid[datoscovid$provincia_iso=='M',]
tabla1 = data.frame(Hombres= Barcelona[Barcelona$sexo=='H',]$num_casos,Mujeres=Barcelona[Barcelona$sexo=
RegMode <-lm(tabla1$Hombres~tabla1$Mujeres, data=tabla1)
summary(RegMode)
```

```
##
## Call:
## lm(formula = tabla1$Hombres ~ tabla1$Mujeres, data = tabla1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -268.69   -4.23    0.56    4.25  401.54
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.563656   0.277812  -2.029   0.0425 *
## tabla1$Mujeres  0.916313   0.002033 450.623   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.08 on 7298 degrees of freedom
## Multiple R-squared:  0.9653, Adjusted R-squared:  0.9653
## F-statistic: 2.031e+05 on 1 and 7298 DF,  p-value: < 2.2e-16
```

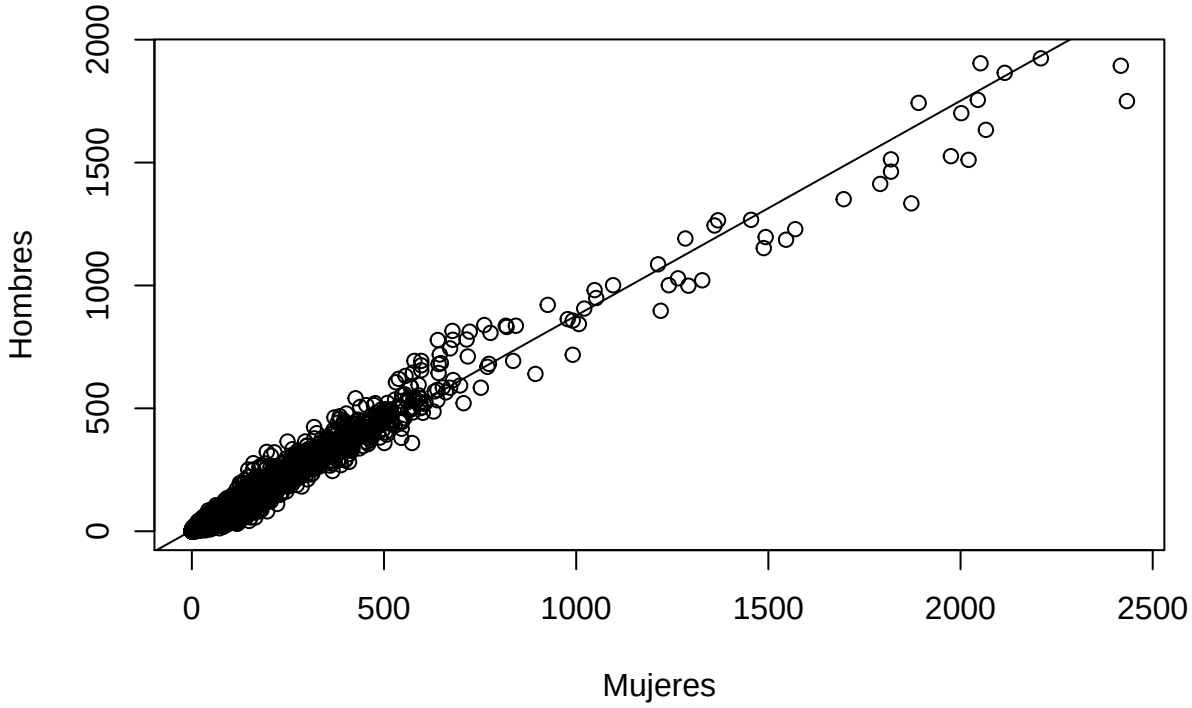
```
#### ¡¡¡¡Buena correlación hombres mujeres!!!!!!!!!!!!!
plot(tabla1$Mujeres,tabla1$Hombres, xlab="Mujeres", ylab="Hombres")
abline(RegMode)
```




```
##### ********** Calcular desviación estándar y demás.
tabla2 = data.frame(Hombres= Madrid[Madrid$sexo=='H',]$num_casos,Mujeres=Madrid[Madrid$sexo=='M',]$num_
RegMode <-lm(tabla2$Hombres~tabla2$Mujeres, data=tabla2)
summary(RegMode)
```

```
##
## Call:
## lm(formula = tabla2$Hombres ~ tabla2$Mujeres, data = tabla2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -379.83   -5.56   -3.32    3.95  219.09
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.31515    0.29193   11.36   <2e-16 ***
## tabla2$Mujeres  0.87403    0.00166  526.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.23 on 7298 degrees of freedom
## Multiple R-squared:  0.9743, Adjusted R-squared:  0.9743
## F-statistic: 2.771e+05 on 1 and 7298 DF,  p-value: < 2.2e-16
```

```
## ¡¡¡En madrid es mejor el ajuste lineal.
plot(tabla2$Mujeres,tabla2$Hombres, xlab="Mujeres", ylab="Hombres")
abline(RegMode)
```



Observamos un ajuste para Barcelona entre casos de hombres y mujeres de: 0.9653, lo que confirma un buen ajuste lineal, como es de esperar, puesto que la pandemia no discrimina entre sexos. Para Madrid el ajuste, por azar presumiblemente, es algo mayor: 0.9743.

10.Análisis de casos de COVID en España en 2021 por Grupos de EDAD

- Grupo 1: 0-9
- Grupo 2: 10-19
- Grupo 3: 20-29
- Grupo 4: 30-39
- Grupo 5: 40-49
- Grupo 6: 50-59
- Grupo 7: 60-69
- Grupo 8: 70-79
- Grupo 9: +80

#Creo una función que, pasándole dos dataframes, uno con los grupos de edad y el otro con los campos que queremos sumar para cada grupo de edad. Funcion "cuentagrupedad":

```
# Saco los grupos de edades que hay:
gruposedad <- unique(datoscovid$grupo_edad)
gruposedad
```

```
##  [1] "0-9"   "10-19" "20-29" "30-39" "40-49" "50-59" "60-69" "70-79" "80+"
## [10] "NC"
```

```
############ ¿Poner nombres a los grupos?????? *****************


### Función que cuenta lo que sea del dataframe por grupos de edad definidos.
cuentagrupedad <- function(x,y) {

    grup <-c(0,0,0,0,0,0,0,0,0,0)
    for (i in 1:length(x)){
      if (x[i]=='0-9'){
        grup[1] = grup[1] + y[i]
      }   else if (x[i]=='10-19'){
        grup[2] = grup[2] + y[i]
      }   else if (x[i]=='20-29'){
        grup[3] = grup[3] + y[i]
      }   else if (x[i]=='30-39'){
        grup[4] = grup[4] + y[i]
      }   else if (x[i]=='40-49'){
        grup[5] = grup[5] + y[i]
      }   else if (x[i]=='50-59'){
        grup[6] = grup[6] + y[i]
      }   else if (x[i]=='60-69'){
        grup[7] = grup[7] + y[i]
      }   else if (x[i]=='70-79'){
        grup[8] = grup[8] + y[i]
      }   else if (x[i]=='80+'){
        grup[9] = grup[9] + y[i]
      }   else {
        grup[10] = grup[10] + y[i]
      }

}
 return(grup)
}
##########  Final de la Función ####################
```

El grupo NC (No Contesta, más adelante será distribuído proporcionalmente entre el resto de grupos según sus porcentajes, es decir, asumiendo que los NC se distribuyen uniformemente sobre el resto de grupos.)

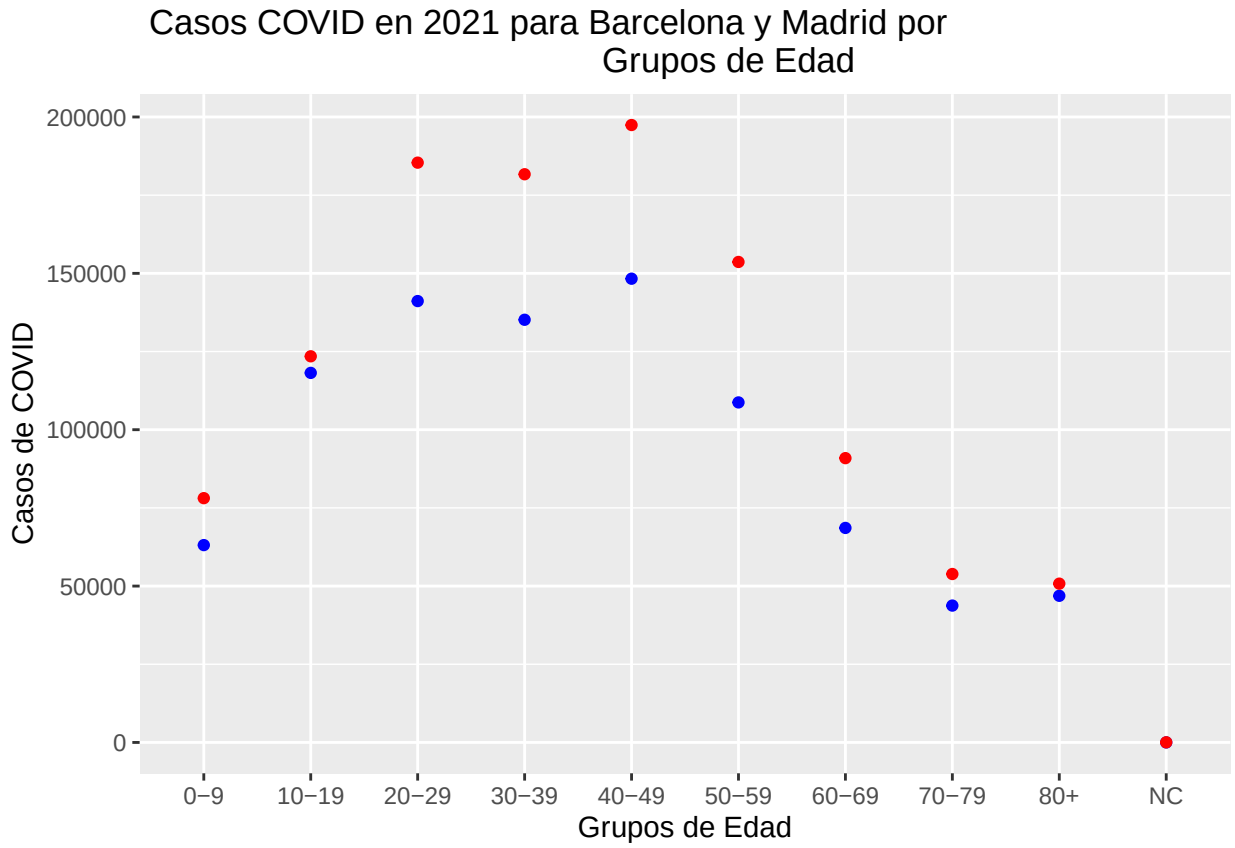
Estudiamos dos provincias, Barcelona y Madrid, para los casos de COVID según los grupos de edad definidos:

```
casosedadbcn = cuentagrupedad(Barcelona$grupo_edad,Barcelona$num_casos)
casosedadbcn <- data.frame(casosedadbcn)
colnames(casosedadbcn)[1] <- "grupo"

casosedadmadrid = cuentagrupedad(Madrid$grupo_edad,Madrid$num_casos)
casosedadmadrid <- data.frame(casosedadmadrid)
colnames(casosedadmadrid)[1] <- "grupo"
```

Graficamos los casos totales de COVID en 2021 por Grupos de Edades para las provincias de Madrid y barcelona:

```
####Terminar de matizar ************
ggplot() +
  geom_point(data=casosedadbcn, aes(x=as.integer(rownames(casosedadbcn)),y=grupo),color='Blue') +
  geom_point(data=casosedadmadrid, aes(x=as.integer(rownames(casosedadmadrid)),y=grupo),color='Red') +
  scale_x_discrete(name ="Grupos de Edad",
```

```
                limits=gruposedad) + ggtitle(" Casos COVID en 2021 para Barcelona y Madrid por
                                              Grupos de Edad")  + ylab("Casos de COVID")
```



Se observan ligeras diferencias algo acusadas en determinados grupos de edad para las dos provincias, cosa que puede ser debido al volumen de población. Sigamos analizando el asunto.

Ahora cargaremos los datos de las provincias en listas y, llamando a la función "cuentagrupedad" definida anteriormente, calcularemos los totales por grupos de edades:

```
#Creamos el dataframe donde cargaremos los datos:
dataset <- data.frame(G0=character(),G1=character(),G2=character(),G3=character(),G4=character(),G5=cha
length(dataset)
```

```
## [1] 11
```

```
for (i in provincias){
  #Cargo los datos de la provincia en un dataframe.
  a <- datoscovid[datoscovid$provincia_iso==i,]
  #Calculamos por grupos:
  casosedada <- as.numeric(cuentagrupedad(a$grupo_edad,a$num_casos))
  casosedada[11] <- as.character(i)
  #Ya están todos los datos en casosedada
  dataset <- rbind(dataset,casosedada)
}
#Renombramos columnas.
colnames(dataset)[1] <- "G0"
colnames(dataset)[2] <- "G1"
colnames(dataset)[3] <- "G2"
```

```r
colnames(dataset)[4] <- "G3"
colnames(dataset)[5] <- "G4"
colnames(dataset)[6] <- "G5"
colnames(dataset)[7] <- "G6"
colnames(dataset)[8] <- "G7"
colnames(dataset)[9] <- "G8"
colnames(dataset)[10] <- "G9"
colnames(dataset)[11] <- "Provincia"

#Transformamos las columnas pertinentes a numéricas.
dataset <- transform(dataset, G0 = as.numeric(G0),G1 = as.numeric(G1),G2 = as.numeric(G2),
          G3 = as.numeric(G3),G4 = as.numeric(G4),G5 = as.numeric(G5),G6 = as.numeric(G6),
          G7 = as.numeric(G7),G8 = as.numeric(G8),G9 = as.numeric(G9))
head(dataset,52)
```

```
##       G0     G1     G2     G3     G4     G5    G6    G7    G8    G9 Provincia
## 1  18627  29110  33260  31979  37428  30977 20791 12964 10008    15         A
## 2   3466   5933   7025   6319   7372   6866  4233  2502  2644     1        AB
## 3   8069  11850  13444  12723  13105   9753  5813  3112  2195     6        AL
## 4   1441   2400   2632   2589   3064   2965  2138  1356  1970     3        AV
## 5  63105 118168 141132 135156 148284 108726 68597 43762 46890     5         B
## 6   6993  11403  14429  12552  13093  11832  7205  4254  4253   269        BA
## 7  15431  23822  26330  24022  30091  25446 15652  8868  9813    11        BI
## 8   4616   7599   8700   7897   9444   8744  5958  3485  4519     0        BU
## 9   7482  12789  16330  14547  16950  12913  8804  6060  5520     3         C
## 10  9828  17765  20809  19914  21531  17748 11393  6115  4487    17        CA
## 11  3222   5738   6809   5578   6222   6285  4044  2494  3608    61        CC
## 12   752   1448   1530   1619   1573   1168   695   339   191     3        CE
## 13  6516  12623  15026  12860  14364  13391  8035  4478  4538     1        CO
## 14  4646   8019   9120   8531   9834   9525  6355  4069  4572     6        CR
## 15  6379   9321   9895   9405  11723   9468  6172  3537  3164     5        CS
## 16  1934   3461   3925   3606   3916   4304  2847  1857  2391     1        CU
## 17  5391   8770  13076  12507  12116   9461  5418  2779  1497     8        GC
## 18  9860  17438  17398  17094  19888  14691  9821  5524  5811     0        GI
## 19  8995  16789  20257  17530  19543  17208 10444  5861  5900     8        GR
## 20  3458   5042   5310   5188   6472   5361  3075  1784  2241     0        GU
## 21  4512   8076   8656   8239   9255   7201  4307  2312  1602     1         H
## 22  2716   4161   4641   4242   5088   4377  3083  1778  2517  3678        HU
## 23  5570   9976  11988  10190  10893  10700  6922  3644  3943     1         J
## 24  6157   9649   9680   9847  12427   9753  6244  3751  4195     0         L
## 25  3770   6775   7774   7791   9679   9276  6625  4117  5864     5        LE
## 26  3783   6291   6990   7298   8741   7400  4924  3092  3640    23        LO
## 27  1842   3186   4206   3349   3694   3115  2226  1544  1899     0        LU
## 28 78110 123470 185392 181686 197427 153640 90906 53856 50773    45         M
## 29 13165  24738  30916  30598  31079  23314 14594  8233  5979   184        MA
## 30  1451   1819   2146   2151   1895   1571  1014   412   310    26        ML
## 31 15075  23167  28985  26551  29362  22877 13104  6909  5923  6761        MU
## 32  1802   3969   4954   3269   3218   3025  2433  1533  1066  6056        NC
## 33  6044  10716  12550  11986  15677  13249 10219  6113  7131    42         O
## 34  2157   3156   4338   4055   4684   3992  3107  2320  2724     0        OR
## 35  2321   3734   4010   3827   4851   4627  3367  2034  2491     1         P
## 36 10275  16604  23155  23515  22879  15835  9561  4852  4037     0        PM
## 37  7349  11475  14028  12916  15808  11514  7528  4832  4210     1        PO
## 38  4649   8297   9138   7948   9606   7759  5673  3263  3373     0         S
```

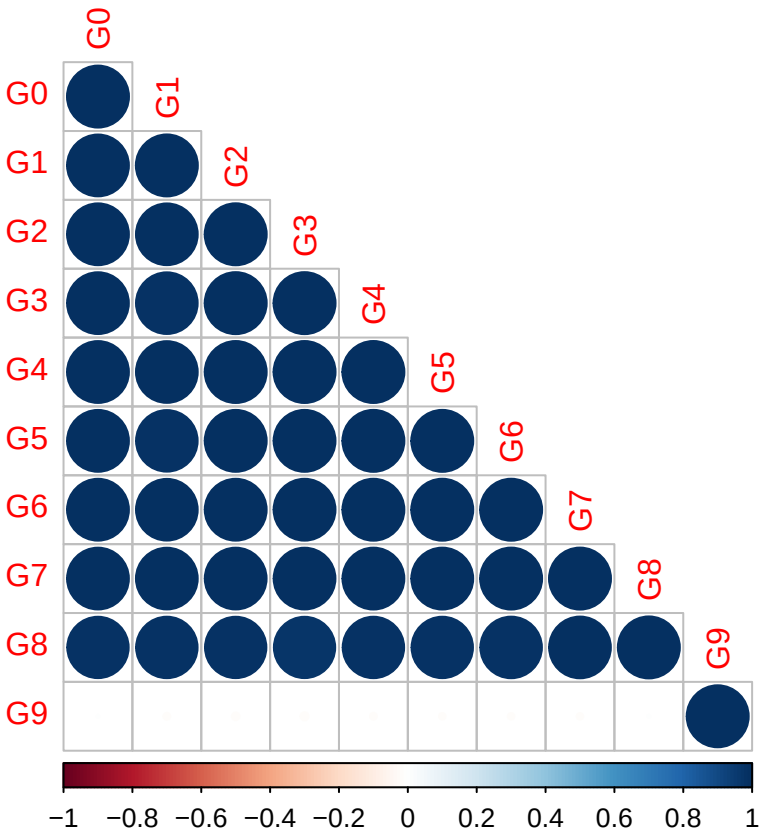
```
## 39  2925   5707   8066   5821   6581   6922  4567  2937  4173     3        SA
## 40 15204  28743  32657  31405  35960  27815 15988  9468  7127     8        SE
## 41  2015   2988   3434   3106   3808   3769  2325  1407  2022     0        SG
## 42  1091   1624   1904   1717   2086   1916  1225   728  1323     0        SO
## 43 13419  20124  19537  16728  22252  18374 11540  6991  5951     6        SS
## 44  7750  14699  15176  13819  16492  12257  8039  5225  5183     0         T
## 45  1653   2550   2722   2376   2936   2911  1944  1212  1624  1316        TE
## 46  5428   9385  14008  13548  13684   9979  5848  3025  2205     1        TF
## 47  8701  13483  14057  13169  16243  14206  9573  6112  8074     4        TO
## 48 26925  42961  51897  47656  55813  44453 29169 17187 15530    21         V
## 49  6653  10847  12860  11960  15783  13916  8496  5507  6373     1        VA
## 50  5218   6945   6715   6778   8793   7053  4238  2571  2524     3        VI
## 51 11484  17343  19850  18964  23548  19293 12561  7845  9356 14831         Z
## 52  1329   2404   2931   2576   3121   3214  2253  1487  2288     4        ZA
```

Estudiamos las correlaciones del dataset de casos por provincia y grupos de edad:

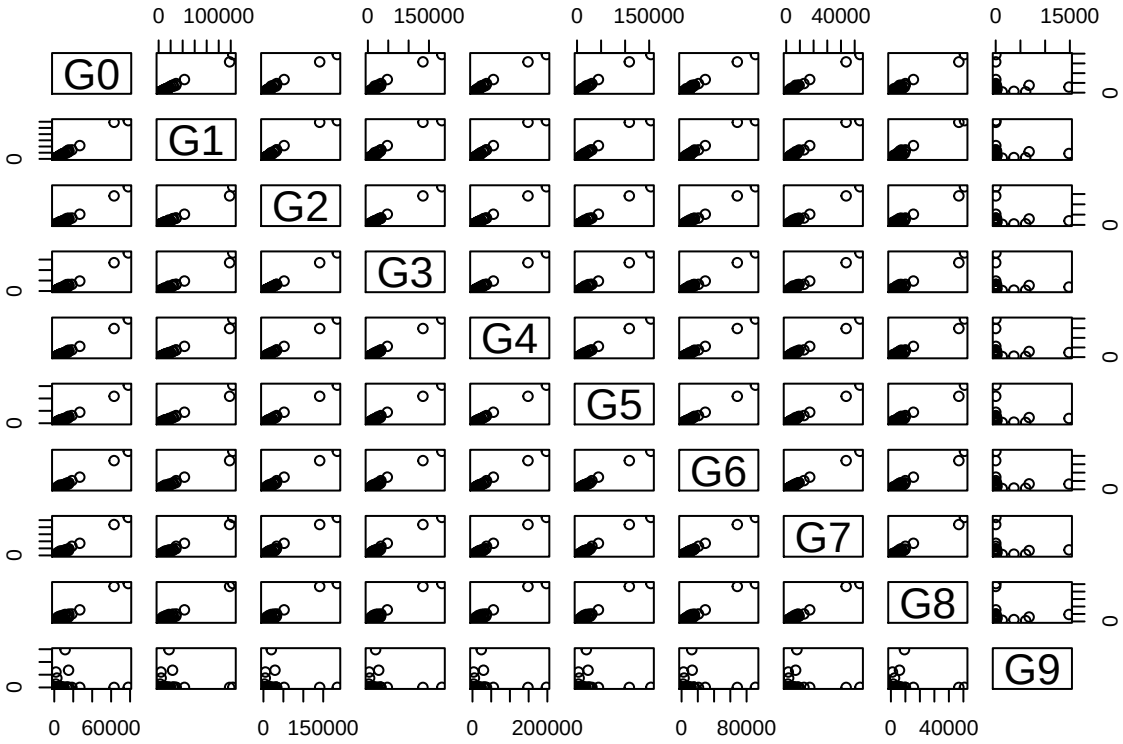
```
#Miramos las correlaciones:
cor(dataset[,1:10])
```

```
##             G0          G1          G2          G3          G4         G5
## G0 1.000000000  0.99475553  0.99492701  0.99346528  0.99619891  0.9947879
## G1 0.994755526  1.00000000  0.99085694  0.98858659  0.99072511  0.9864073
## G2 0.994927013  0.99085694  1.00000000  0.99959927  0.99919597  0.9976352
## G3 0.993465275  0.98858659  0.99959927  1.00000000  0.99913299  0.9973792
## G4 0.996198909  0.99072511  0.99919597  0.99913299  1.00000000  0.9988446
## G5 0.994787934  0.98640727  0.99763516  0.99737924  0.99884463  1.0000000
## G6 0.995874900  0.98983708  0.99647813  0.99571988  0.99809425  0.9987365
## G7 0.994262263  0.99153598  0.99411264  0.99300013  0.99602906  0.9956385
## G8 0.979873984  0.98283991  0.97976507  0.97805866  0.98216118  0.9808182
## G9 0.001813362 -0.01170871 -0.01488759 -0.01728652 -0.01192305 -0.0107582
##             G6          G7         G8           G9
## G0  0.99587490  0.99426226 0.97987398  0.001813362
## G1  0.98983708  0.99153598 0.98283991 -0.011708714
## G2  0.99647813  0.99411264 0.97976507 -0.014887592
## G3  0.99571988  0.99300013 0.97805866 -0.017286520
## G4  0.99809425  0.99602906 0.98216118 -0.011923047
## G5  0.99873646  0.99563847 0.98081816 -0.010758198
## G6  1.00000000  0.99852147 0.98608368 -0.010622036
## G7  0.99852147  1.00000000 0.99179484 -0.011393149
## G8  0.98608368  0.99179484 1.00000000  0.002327610
## G9 -0.01062204 -0.01139315 0.00232761  1.000000000
```

```
corrplot(round(cor(subset(dataset[,1:10])), digits = 3), type = "lower")
```



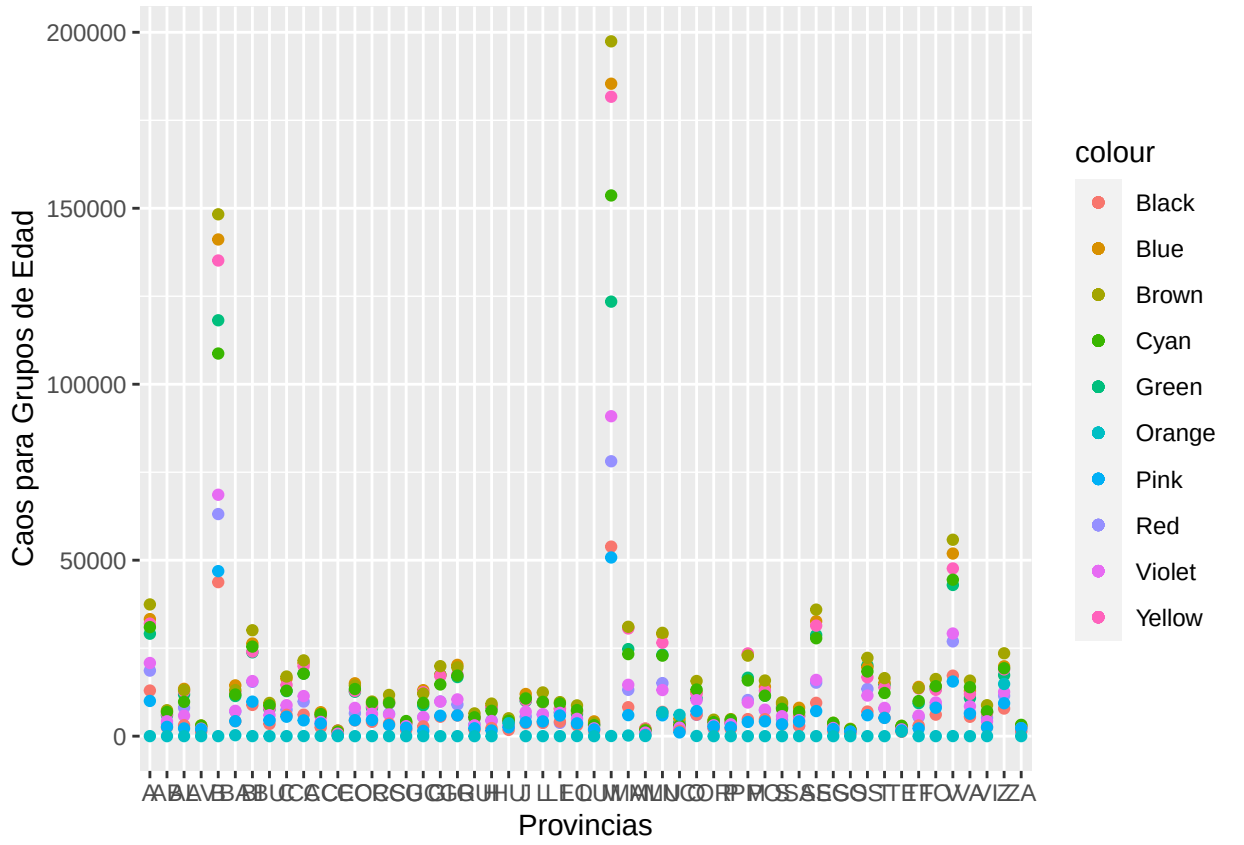

```
pairs(dataset[,1:10])
```




```
#Estudiar las correlaciones ************************
```

Como podemos ver, las correlaciones entre los grupos de edad son bastante altas entre todos los grupos de edad. Pasamos a efectuar un gráfico de los casos por grupo de edad para las 52 provincias.

```
#Pruebo un gráfico:
 ggplot(data=dataset) +
   geom_point( aes(x=dataset[1:52,11],y=G0,color="Red"))  +
   geom_point( aes(x=dataset[1:52,11],y=G1,color="Green"))  +
   geom_point( aes(x=dataset[1:52,11],y=G2,color="Blue"))  +
   geom_point( aes(x=dataset[1:52,11],y=G3,color="Yellow"))  +
   geom_point( aes(x=dataset[1:52,11],y=G4,color="Brown"))  +
   geom_point( aes(x=dataset[1:52,11],y=G5,color="Cyan"))  +
   geom_point( aes(x=dataset[1:52,11],y=G6,color="Violet"))  +
   geom_point( aes(x=dataset[1:52,11],y=G7,color="Black"))  +
   geom_point( aes(x=dataset[1:52,11],y=G8,color="Pink"))  +
   geom_point( aes(x=dataset[1:52,11],y=G9,color="Orange"))  +

   scale_x_discrete(name ="Provincias",
                    limits=dataset[1:52,11]) + ylab("Caos para Grupos de Edad")
```



En el gráfico, en las distintas provincias, podemos observar casi un mismo patrón de colores, más o menos, lo que indica una uniformidad del grado de incidencia en las provincias por grupos de edad, es decir, que la provincia no es un factor determinate en cuanto a la edad o a enfermar de COVID.

Ahora, con la funcion "cuentagrupedad", vamos a calcular los casos totales y las defunciones totales de COVID en España en 2021 por nuestros grupos de edad definidos y efectuaremos un pequeño estudio:

```
# ESTUDIO anova GRUPOS EDAD/DEFUNCIONES Y CASOS TOTALES.

deftot <- cuentagrupedad(datoscovid$grupo_edad,datoscovid$num_def)
deftot
```

```
##  [1]    19    22   128   294  1002  3395  8805 19130 54965   347
```

```
casostot <- cuentagrupedad(datoscovid$grupo_edad,datoscovid$num_casos)
casostot <- as.data.frame(casostot)
casostot
```

```
##    casostot
## 1    470754
## 2    788550
## 3    965798
## 4    910697
## 5   1029376
## 6    826135
## 7    521093
## 8    311497
## 9    311642
```

```
## 10    33445
```

```
dataset2 <- data.frame(Casos=casostot,Muertes=deftot)

summary(dataset2)
```

```
##     casostot          Muertes
##  Min.   :  33445   Min.   :   19.0
##  1st Qu.: 351420   1st Qu.:  169.5
##  Median : 654822   Median :  674.5
##  Mean   : 616899   Mean   : 8810.7
##  3rd Qu.: 889557   3rd Qu.: 7452.5
##  Max.   :1029376   Max.   :54965.0
```

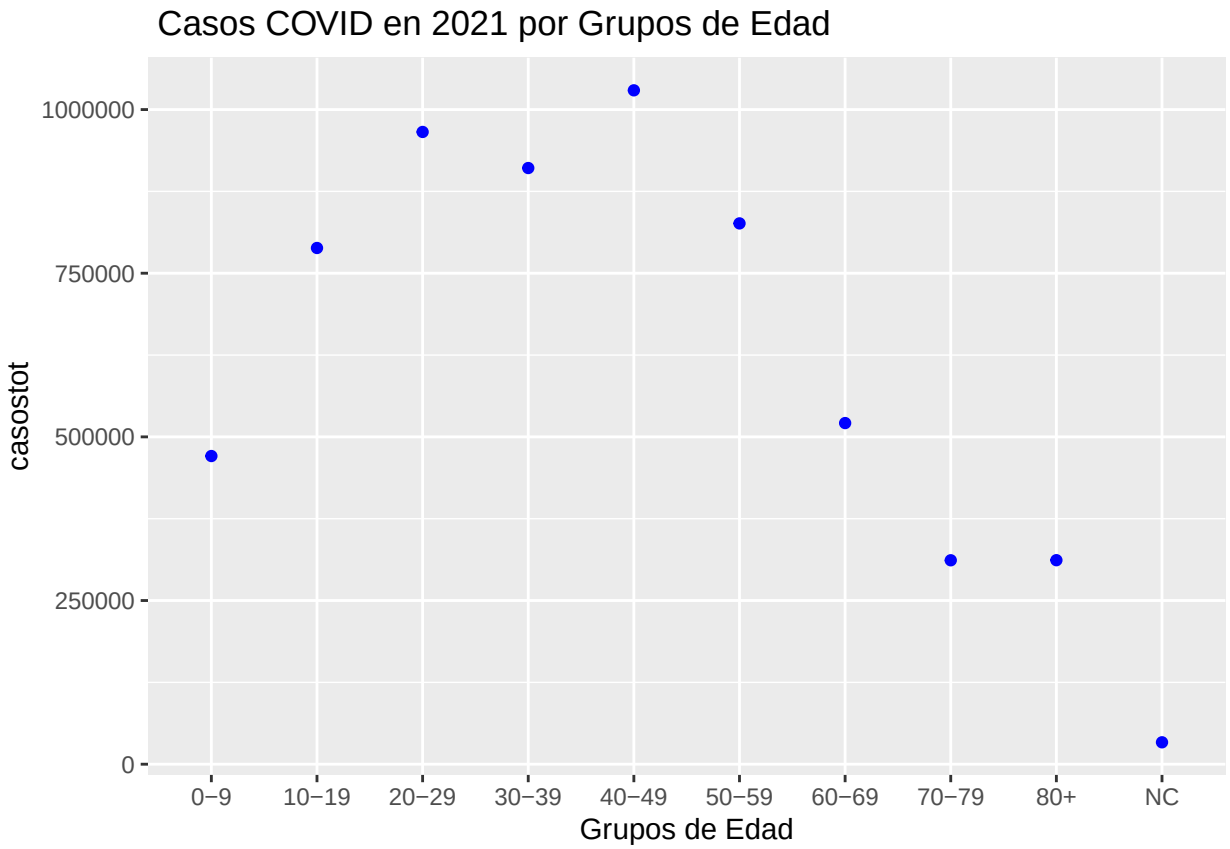
```
cor(dataset2)
```

```
##            casostot    Muertes
## casostot  1.0000000 -0.4293208
## Muertes  -0.4293208  1.0000000
```

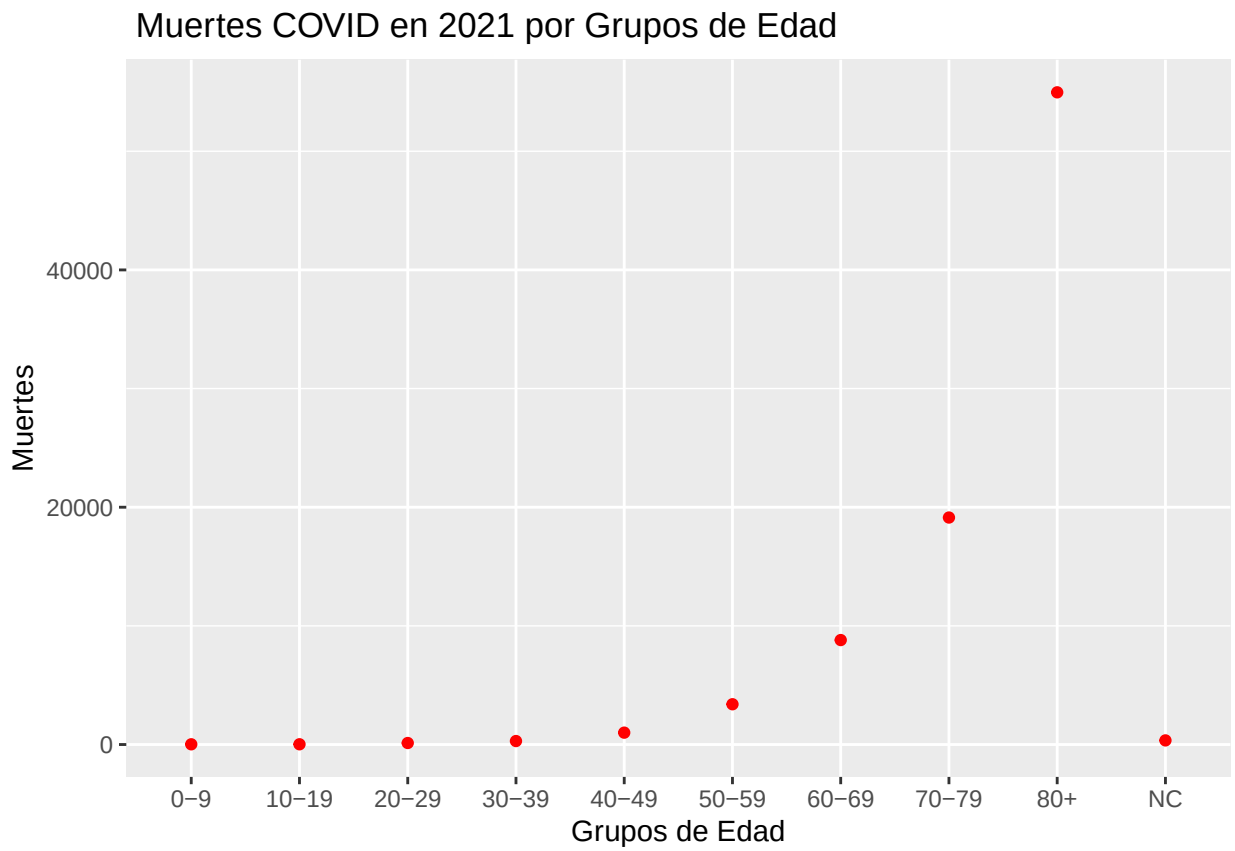
No se observa una correlación significativa, dentro de los grupos de edad, entre los casos de COVID y fallecimiento por COVID

Vamos a efectuar un gráfico de los casos y muertes de COVID para los diferentes grupos de edad:

```
ggplot() +
  geom_point(data=dataset2, aes(x=gruposedad,y=casostot),color='Blue') +
  ggtitle(" Casos COVID en 2021 por Grupos de Edad")   +
  #geom_point(data=dataset2, aes(x=gruposedad,y=Muertes),color='Red') + #Poner etiqueta
  scale_x_discrete(name ="Grupos de Edad")
```



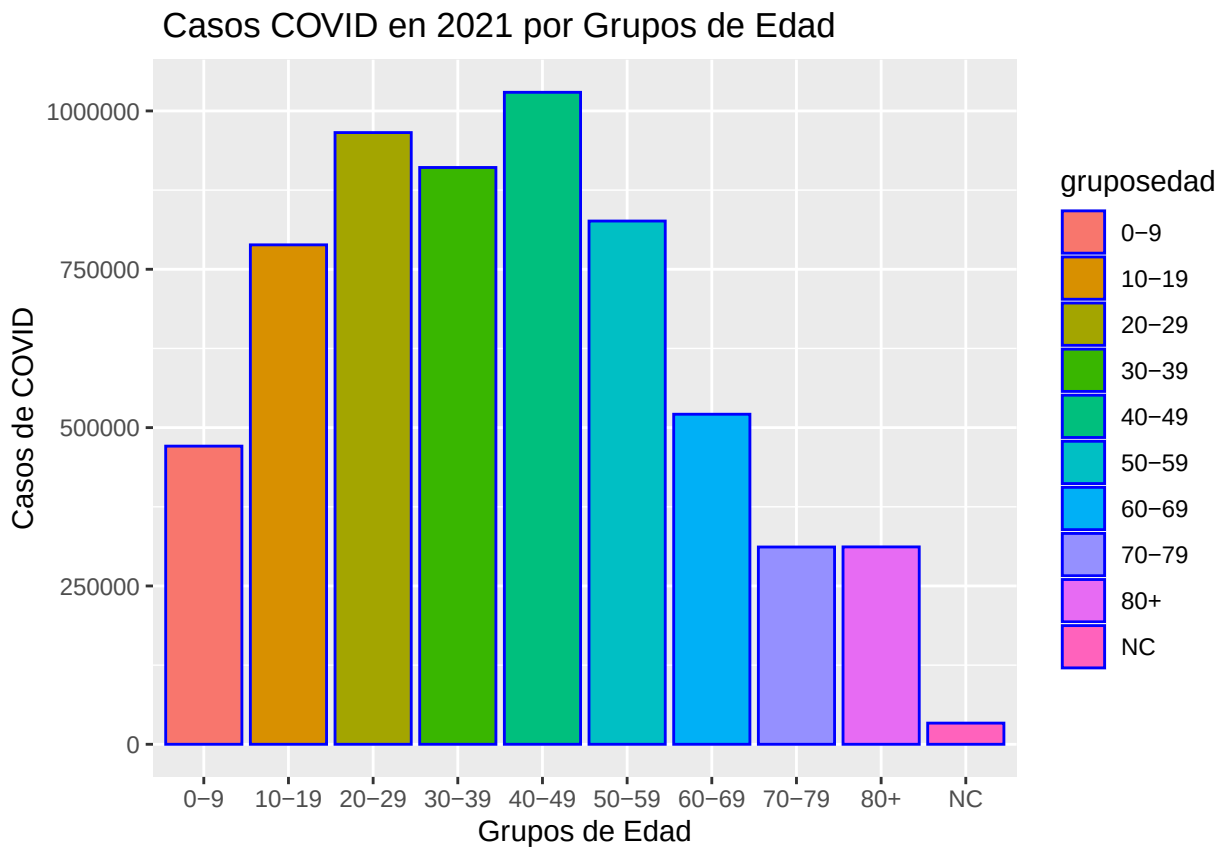

```
ggplot() +
  geom_point(data=dataset2, aes(x=gruposedad,y=Muertes),color='Red') +
  ggtitle(" Muertes COVID en 2021 por Grupos de Edad")   +#Poner etiqueta
  scale_x_discrete(name ="Grupos de Edad")
```




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

La muerte por COVID parece crecer en los grupos de edad. Sin embargo, los casos de infección por COVID hacen una especie de parábola invertida con un máximo en el grupo de 40-49 años.

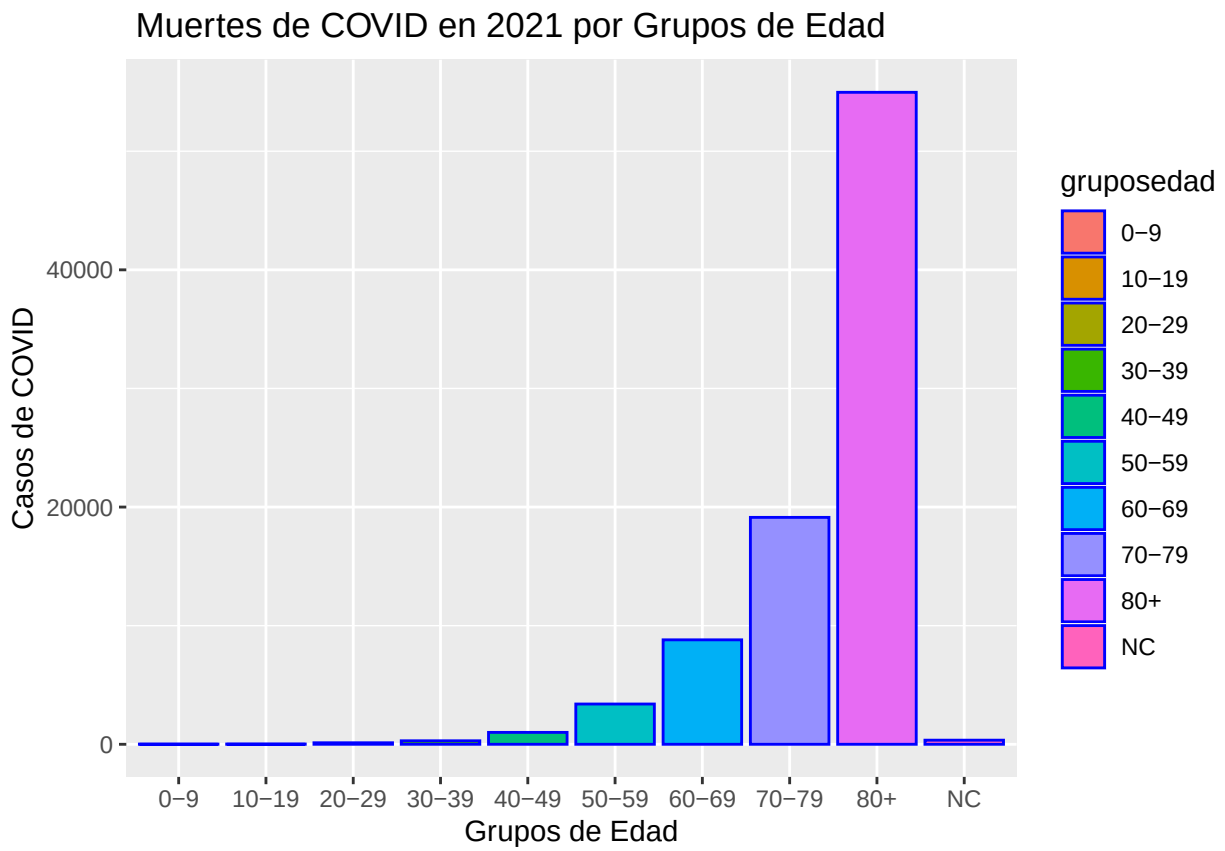
```
ggplot(dataset2,aes(x=gruposedad,y=casostot,fill=gruposedad))+
  geom_bar(stat = "identity",
           color="Blue") + ggtitle(" Casos COVID en 2021 por Grupos de Edad")  +
  ylab("Casos de COVID")  +
  xlab("Grupos de Edad")
```




```
gruposedad
```

```
##  [1] "0-9"   "10-19" "20-29" "30-39" "40-49" "50-59" "60-69" "70-79" "80+"
## [10] "NC"
```

```
ggplot(dataset2,aes(x=gruposedad,y=Muertes,fill=gruposedad))+
  geom_bar(stat = "identity",
           color="Blue")+ ggtitle(" Muertes de COVID en 2021 por Grupos de Edad")  +
  ylab("Casos de COVID")  +
  xlab("Grupos de Edad")
```



11.Simulación de la Pandemia de Covid en España 2021. Métodos de Montecarlo.

Como solo tenemos datos absolutos por provincia o por comunidad, vamos a generar o simular una muestra de 50 mil personas de una población para poder hacer una estadística. Para ello nos valdremos de un censo del INE que nos dice cuántas personas hay en España de cada edad, de 0 a 100, y de 100 o más. Este censo lo he sacado de Wikipedia y corresponde al año 2021. De esta manera, mediante una simulación de Montecarlo generaremos una muestra de 50 mil individuos distribuídos por los grupos de edades que hemos definido antes y según los porcentajes del INE.

De otro lado, como tenemos los casos de COVID que se han dado por grupos de edad de manera empírica, podremos simular, vía Montecarlo, un proceso de enfermar a la población antes generada, siempre según el modelo estadístico que nos marca la experiencia del año 2021, según se haya distribuído la pandemia por los grupos de edad definidos.

Cargamos el censo y distribuimos por los grupos de edades:

```
# DEMOGRAFÍA DATOS: https://es.wikipedia.org/wiki/Demograf%C3%ADa_de_Espa%C3%B1a

#Cargo el censo de Wikipedia que tengo en el fichero censowiki: Wikipedia lo saca del INE.
censowiki <- read.csv("censowiki.csv",header = TRUE, sep="")
tot20 <- sum(censowiki$X2020)
tot21 <- sum(censowiki$X2021)
tot20
```

```
## [1] 47332613
```

```
tot21
```

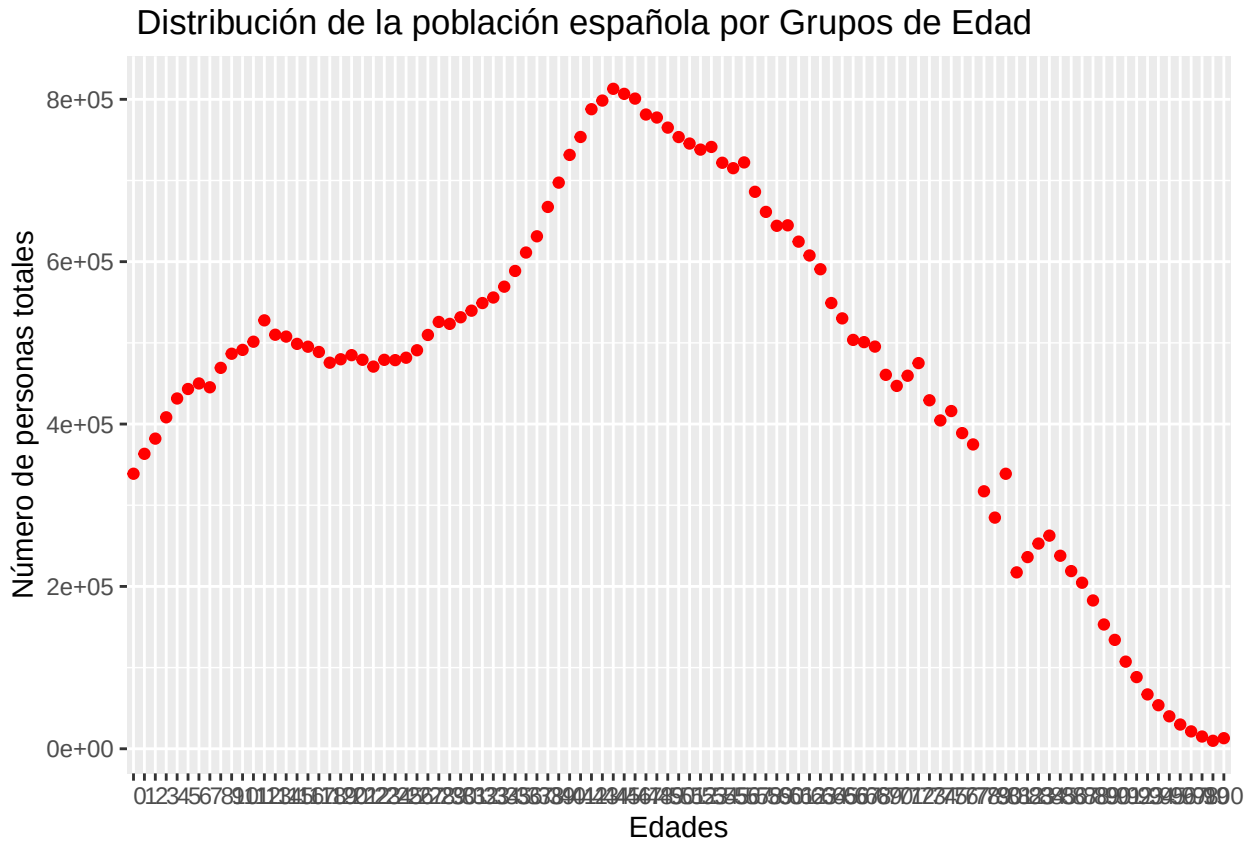
```
## [1] 47394223
```

Hacemos un gráfico del número de personas por grupo de edad:

```
nrow(censowiki)
```

```
## [1] 101
```

```
edades =c(0:100) #Creo un "vector de edades" en unidades de año. El número 100
#indica 100 años o más.
ggplot() +
  geom_point(data=censowiki, aes(x=edades,y=X2021),color='Red') + #Poner etiqueta
  scale_x_discrete(name ="Edades",limits=edades) +
  ggtitle(" Distribución de la población española por Grupos de Edad")  +
  ylab("Número de personas totales")  +
  xlab("Grupos de Edad")
```



Ahora calculamos los totales y los porccentajes de los grupos de edades:

```
#Calculamos porcentajes según los grupos de edad anterior:
grupedad <-c(0,0,0,0,0,0,0,0,0,0)
for (i in 1:101){
  if (i <=9){
      grupedad[1] = grupedad[1] + censowiki$X2021[i]

  } else if ((i>9) & (i<=19)){
      grupedad[2] = grupedad[2] + censowiki$X2021[i]
```

```
  } else if ((i>19) & (i<=29)){
      grupedad[3] = grupedad[3] + censowiki$X2021[i]
  } else if ((i>29) & (i<=39)){
      grupedad[4] = grupedad[4] + censowiki$X2021[i]
  } else if ((i>39) & (i<=49)){
      grupedad[5] = grupedad[5] + censowiki$X2021[i]
  } else if ((i>49) & (i<=59)){
      grupedad[6] = grupedad[6] + censowiki$X2021[i]
  } else if ((i>59) & (i<=69)){
      grupedad[7] = grupedad[7] + censowiki$X2021[i]
  } else if ((i>69) & (i<=79)){
      grupedad[8] = grupedad[8] + censowiki$X2021[i]
  } else {
      grupedad[9] = grupedad[9] + censowiki$X2021[i]
  }
}
# La variable grupedad es una lista o vector con los números de personas por grupos
#en España
print('Distribución por grupos de edad')
```

```
## [1] "Distribución por grupos de edad"
```

```
grupedad
```

```
## [1] 3731654 4983242 4880571 5767272 7748264 7250349 5691213 4172817 3168841
```

```
print('Total Población España 2021:')
```

```
## [1] "Total Población España 2021:"
```

```
sum(grupedad)
```

```
## [1] 47394223
```

```
grupedad <- (grupedad/tot21)
print('Tanto por Uno por grupos de edad')
```

```
## [1] "Tanto por Uno por grupos de edad"
```

```
grupedad
```

```
## [1] 0.07873647 0.10514450 0.10297818 0.12168724 0.16348541 0.15297959 0.12008242
## [8] 0.08804484 0.06686133
```

```
print('Suma de los porcentajes en tanto por uno de los grupos de edades:')
```

```
## [1] "Suma de los porcentajes en tanto por uno de los grupos de edades:"
```

```
sum(grupedad)
```

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

```
# "grupedad" contiene los porcentajes de la población en cada grupo.
```

Ahora vamos a generar por el Método de Montecarlo una población española distribuída por grupos de edades según el año 2021:

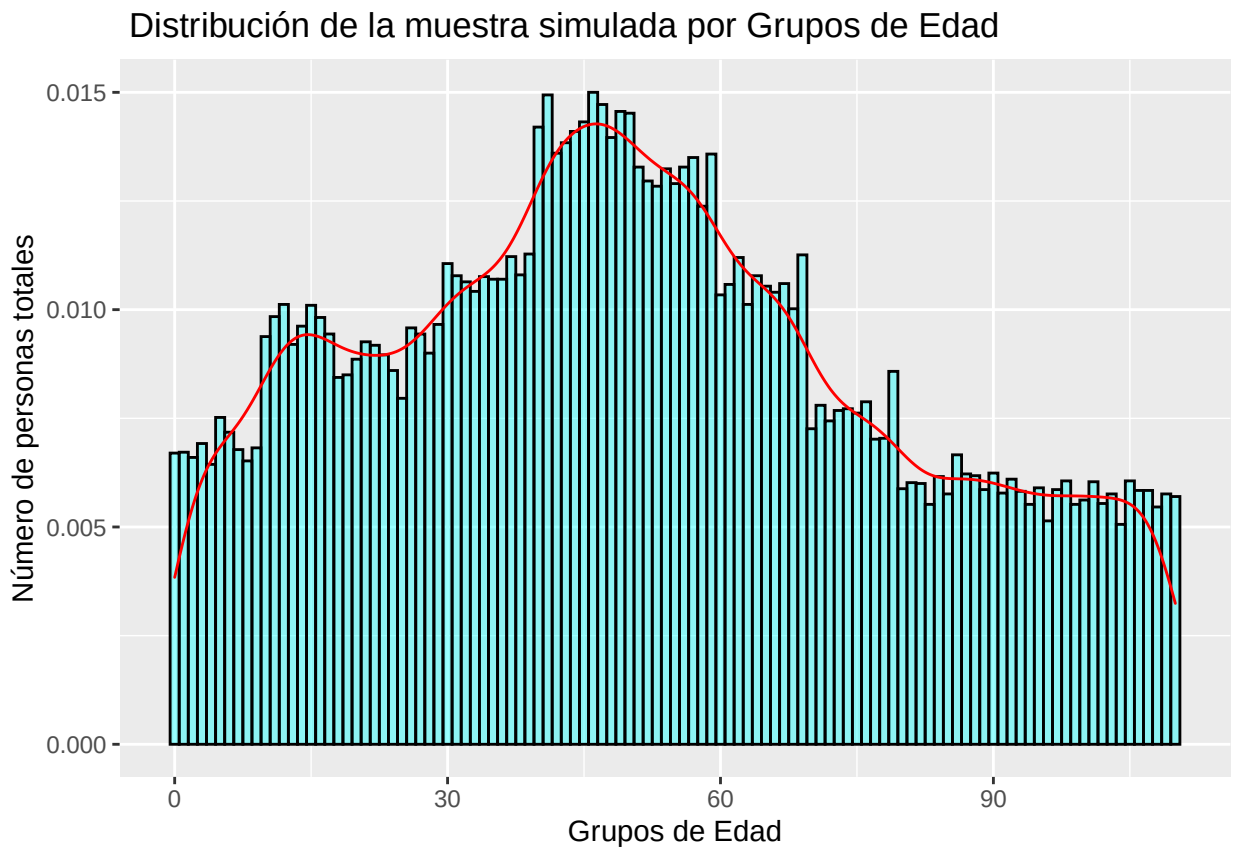
```
#### Generamos por Montecarlo una muestra simulada de edades para 50.000 sujetos
#de la población española, según los párametros estadísticos del INE.
muestra <- data.frame(Edad = numeric())
```

```
i=0
while (i<50000){
   a <- sample(0:110,1,replace=F) #Genero una edad
   #Ahora vemos si la aceptamos o no:
   if (a <=9){
     b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
     #Vemos si aceptamos o no la edad i:
     if (b <= grupedad[1]){
       muestra <- rbind(muestra,a)
       i <- i + 1
     } else {
       next
     }



   } else if ((a>9) & (a<=19)){
     b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
     #Vemos si aceptamos o no la edad i:
     if (b <= grupedad[2]){
       muestra <- rbind(muestra,a)
       i <- i + 1
     } else {
       next
     }
   } else if ((a>19) & (a<=29)){
     b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
     #Vemos si aceptamos o no la edad i:
     if (b <= grupedad[3]){
       muestra <- rbind(muestra,a)
       i <- i + 1
     } else {
       next
     }
   } else if ((a>29) & (a<=39)){
     b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
     #Vemos si aceptamos o no la edad i:
     if (b <= grupedad[4]){
       muestra <- rbind(muestra,a)
       i <- i + 1
     } else {
       next
     }
   } else if ((a>39) & (a<=49)){
     b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
     #Vemos si aceptamos o no la edad i:
     if (b <= grupedad[5]){
       muestra <- rbind(muestra,a)
       i <- i + 1
     } else {
       next
     }
   } else if ((a>49) & (a<=59)){
```

```
    b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
    #Vemos si aceptamos o no la edad i:
    if (b <= grupedad[6]){
      muestra <- rbind(muestra,a)
      i <- i + 1
    } else {
      next
    }
  } else if ((a>59) & (a<=69)){
    b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
    #Vemos si aceptamos o no la edad i:
    if (b <= grupedad[7]){
      muestra <- rbind(muestra,a)
      i <- i + 1
    } else {
      next
    }
  } else if ((a>69) & (a<=79)){
    b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
    #Vemos si aceptamos o no la edad i:
    if (b <= grupedad[8]){
      muestra <- rbind(muestra,a)
      i <- i + 1
    } else {
      next
    }
  } else {
    b <- runif(1,0,1) #Genero un número aleatorio uniforme entre 0 y 1
    #Vemos si aceptamos o no la edad i:
    if (b <= grupedad[9]){
      muestra <- rbind(muestra,a)
      i <- i + 1
    } else {
      next
    }
  }

}
#muestra # Tenemos una muestra de 50.000 personas simuladas según la distribución
#proporcionada por el INE de la población española en 2021.
colnames(muestra) <- 'Edad'
```

Ahora pasaremos a analizar los estadísticos de esta muestra y luego los pasaremos por una simulación en donde los individuos enfermarán o no en base a los datos estadísticos del INE sobre los casos de COVID-19.

```
ggplot(muestra, aes(x=Edad )) + geom_histogram(aes(y=..density..), binwidth = 1, col='black', fill='cya
  ylab("Número de personas totales")  +
  xlab("Grupos de Edad")
```



Vemos que la función de densidad inferida, y el histograma de frecuencias, se asemeja bastante al histograma que hemos visto anteriormente de la población española oficial y real por edades. La variación puede residir en el tamaño muestral y en la agrupación por grupos de edad. El histograma anterior tenía un ajuste más "fino" 'para las edades que no éste, que lo es por grupos de 10 en 10.

Sacamos algunos estadísiticos básicos de la muestra poblacional generada vía Montecarlo:

```
#View(muestra)
#Algunos estadísticos:
print('Media de edad:')
```

```
## [1] "Media de edad:"
```

```
mean(muestra$Edad)
```

```
## [1] 50.8652
```

```
print('Desviación estándar de la distribución en grupos de edades:')
```

```
## [1] "Desviación estándar de la distribución en grupos de edades:"
```

```
sd(muestra$Edad)
```

```
## [1] 28.35371
```

```
print('Resumen estadístico de la distribución de edades de la muestra generada:')
```

```
## [1] "Resumen estadístico de la distribución de edades de la muestra generada:"
```

```
summary(muestra)
```

```
##       Edad
```

```
##  Min.   :  0.00
##  1st Qu.: 29.00
##  Median : 49.00
##  Mean   : 50.87
##  3rd Qu.: 70.00
##  Max.   :110.00
```

Ya tenemos nuestra muestra de sujetos con una edad de la población española en base a las estadísticas de censo del año 2021. Ahora vamos a proceder a simular un proceso de para enfermarlos o no de COVID. No atenderemos a los tipos de pruebas de detectar COVID, sólo al hecho de haber sido diagnosticado como positivo. De otro lado,también simularemos defunciones por COVID en base a los parámetros estadísticosobtenidos de las estadísticas de COVID 19 del INE, solo en base a lo 9 grupos de edades que han vienen definidos.

Antes de enfermarlos haremos unos estudios previos sobre los datos reales de casos y muertes por grupos de edad. Todavía no entramos en la simulación de Montecarlo, así que guardamos la "muestra" generada para más tarde.

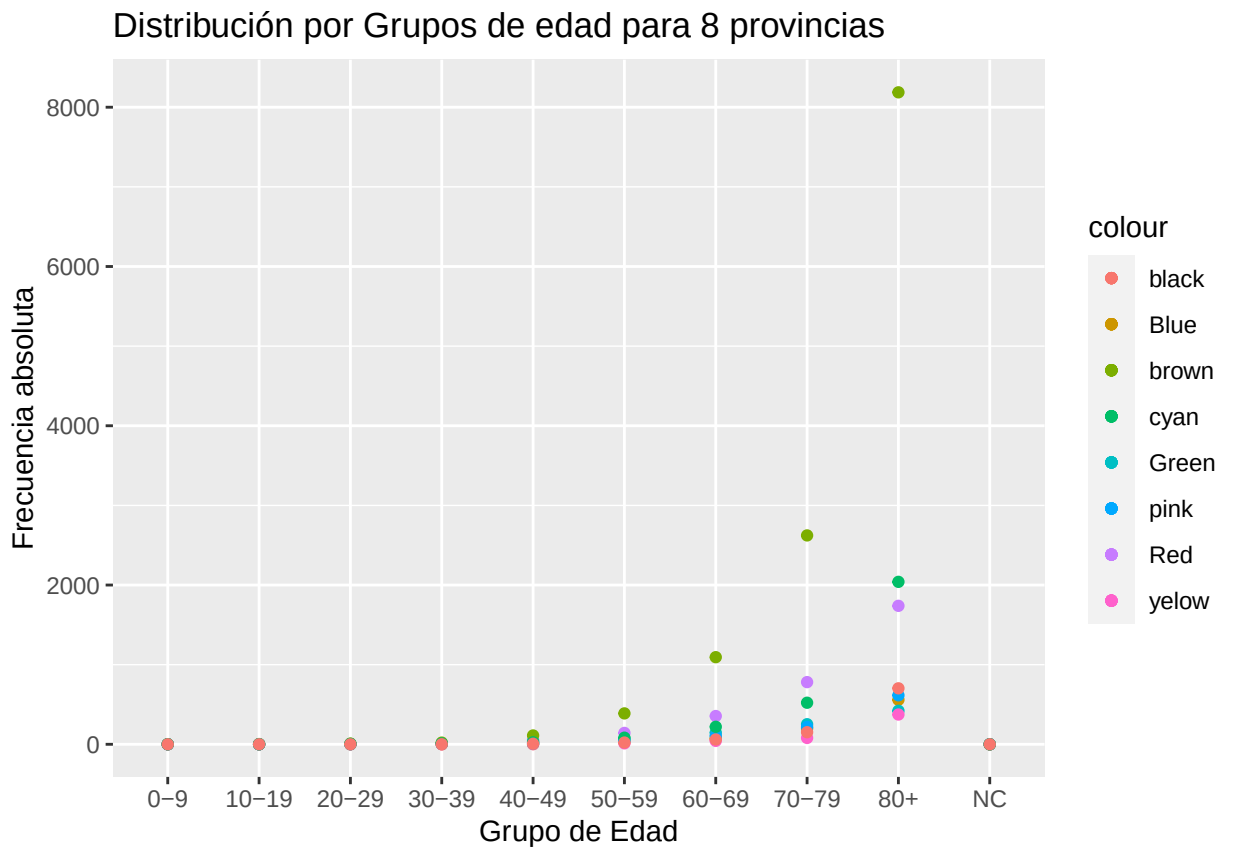
Una forma de agrupar en los grupos de edades los datos concretos del censo, por provincias, sería la siguiente. Además, haremos algunos gráficos de la distribución real según el censo del INE de las edades en grupos para algunas provincias.

```
########## Analizamos los casos de COVID y fallecidos en base a las estadísticas
# de los datos con los que trabajamos:
totcasoscovid <- sum(datoscovid$num_casos)
totmuertes <- sum(datoscovid$num_def)

#install.packages("reshape2")
censo_spain_sexo_df = read.csv("casossexouci.csv")
myDatacasos = aggregate(censo_spain_sexo_df$num_casos, list(censo_spain_sexo_df$grupo_edad, censo_spain
library("reshape2")
#View(myDatacasos)
#Dcast es un pivot table (eje: 2 columnas)
resultado = reshape2::dcast(data= myDatacasos, formula = Group.1 ~ Group.2, fun.aggregate = sum, value.
#View(resultado)
resultado_tr = reshape2::dcast(data= myDatacasos, formula = Group.2 ~ Group.1, fun.aggregate = sum, valu
#View(resultado_tr)

# Fallecidos por edad y provincia. 2 maneras:
# CASOS POR EDAD Y PROVINCIA DE 2 FORMAS.

#censo_spain_sexo_df = read.csv("casossexouci.csv")
myDatadef = aggregate(censo_spain_sexo_df$num_def, list(censo_spain_sexo_df$grupo_edad, censo_spain_sexo

#View(myDatadef)
#Dcast es un pivot table (eje: 2 columnas)
resultado2 = reshape2::dcast(data= myDatadef, formula = Group.1 ~ Group.2, fun.aggregate = sum, value.va
#View(resultado2)
resultado_tr2 = reshape2::dcast(data= myDatadef, formula = Group.2 ~ Group.1, fun.aggregate = sum, valu
#View(resultado_tr2)




#dataset <- transform(resultado2, G0 = as.numeric(G0),G1 = as.numeric(G1),G2 = as.numeric(G2),
 #                    G3 = as.numeric(G3),G4 = as.numeric(G4),G5 = as.numeric(G5),G6 = as.numeric(G6),
```

```
  #                   G7 = as.numeric(G7),G8 = as.numeric(G8),G9 = as.numeric(G9))

#transform(resultado2,colnames(resultado2)=as.numeric(resultado2[,1:52]))

res <- lapply(resultado2[, 2:53],as.numeric)

res$grupos = resultado2[, 1]
#View(res)
#Miramos las correlaciones:
cor(dataset[,1:10])
```

```
##             G0          G1          G2          G3          G4         G5
## G0 1.000000000  0.99475553  0.99492701  0.99346528  0.99619891  0.9947879
## G1 0.994755526  1.00000000  0.99085694  0.98858659  0.99072511  0.9864073
## G2 0.994927013  0.99085694  1.00000000  0.99959927  0.99919597  0.9976352
## G3 0.993465275  0.98858659  0.99959927  1.00000000  0.99913299  0.9973792
## G4 0.996198909  0.99072511  0.99919597  0.99913299  1.00000000  0.9988446
## G5 0.994787934  0.98640727  0.99763516  0.99737924  0.99884463  1.0000000
## G6 0.995874900  0.98983708  0.99647813  0.99571988  0.99809425  0.9987365
## G7 0.994262263  0.99153598  0.99411264  0.99300013  0.99602906  0.9956385
## G8 0.979873984  0.98283991  0.97976507  0.97805866  0.98216118  0.9808182
## G9 0.001813362 -0.01170871 -0.01488759 -0.01728652 -0.01192305 -0.0107582
##             G6          G7         G8           G9
## G0  0.99587490  0.99426226 0.97987398  0.001813362
## G1  0.98983708  0.99153598 0.98283991 -0.011708714
## G2  0.99647813  0.99411264 0.97976507 -0.014887592
## G3  0.99571988  0.99300013 0.97805866 -0.017286520
## G4  0.99809425  0.99602906 0.98216118 -0.011923047
## G5  0.99873646  0.99563847 0.98081816 -0.010758198
## G6  1.00000000  0.99852147 0.98608368 -0.010622036
## G7  0.99852147  1.00000000 0.99179484 -0.011393149
## G8  0.98608368  0.99179484 1.00000000  0.002327610
## G9 -0.01062204 -0.01139315 0.00232761  1.000000000
```

```
#View(dataset)
#Estudiar las correlaciones **********************
#Pruebo un gráfico:
p=c()
res = as.data.frame(res)
ggplot(data=res) + geom_point(aes(x=grupos,y=res[,1],color="Red")) +
   geom_point(aes(x=grupos,y=res[,2],color="Blue")) +
   geom_point(aes(x=grupos,y=res[,3],color="Green")) +
  geom_point(aes(x=grupos,y=res[,4],color="yelow")) +
  geom_point(aes(x=grupos,y=res[,5],color="brown")) +
  geom_point(aes(x=grupos,y=res[,6],color="pink")) +
  geom_point(aes(x=grupos,y=res[,7],color="cyan")) +
  geom_point(aes(x=grupos,y=res[,8],color="black")) +

  labs (title= "Distribución por Grupos de edad para 8 provincias",
      x="Grupo de Edad",
      y="Frecuencia absoluta")
```



No he graficado para todas las provincias por grupos de edad puesto que sale muy engorroso.

Ahora, sobre la estadística real y antes de pasar a la simulación, vamos a agrupar los casos y fallecimientos de COVID por grupos de edad. Además, los casos NC (No Contesta a la edad) los distribuiremos uniformemente por los grupos de edades según sus porcentajes del total.

```
casosgrupos <- c(0,0,0,0,0,0,0,0,0,0) #Casos por grupo de edad.
muertosgrupos <- c(0,0,0,0,0,0,0,0,0,0) #Muertos por grupo de edad.

for (i in 1:10){
  casosgrupos[i] = sum(resultado[i,2:53])
  muertosgrupos[i] <- sum(resultado2[i,2:53])
 }

#casosgrupos
#muertosgrupos

#Vamos a corregir los casos que pertenecen al grupo de NC (No contesta a la edad).
#Para ello, tenemos en cuenta las proporciones de cada grupo y, el total de NC lo
#distribuiremos ponderadamente en los grupos de edades, en base a sus % de representación.
casosnc <- casosgrupos[10]
muertosnc <- muertosgrupos[10]

sumacasnc = sum(casosgrupos[1:9])
sumamuernc = sum(muertosgrupos[1:9])
porcentajescasos <- casosgrupos[1:9]/sumacasnc
porcentajesmuertes <- muertosgrupos[1:9]/sumamuernc
```

```
porcentajescasos <- porcentajescasos * casosnc
porcentajescasos <- round(porcentajescasos)
porcentajesmuertes <- porcentajesmuertes * muertosnc
porcentajesmuertes <- round(porcentajesmuertes)

muertosgrupos <- muertosgrupos[1:9] + porcentajesmuertes
casosgrupos <- casosgrupos[1:9] + porcentajescasos


dfcovidgrupos <- data.frame(Grupo = gruposedad[1:9], Casos = casosgrupos[1:9], Muertes = muertosgrupos[

head(dfcovidgrupos,9)
```

```
##   Grupo   Casos Muertes
## 1   0-9  473320      19
## 2 10-19  792848      22
## 3 20-29  971063     129
## 4 30-39  915661     295
## 5 40-49 1034987    1006
## 6 50-59  830638    3408
## 7 60-69  523933    8840
## 8 70-79  313195   19206
## 9   80+  313341   55182
```

```
sum(casosgrupos)
```

```
## [1] 6168986
```

```
sum(muertosgrupos)
```

```
## [1] 88107
```

Pasamos a realizar un ver los porcentajes de casos/muertes por grupos de edad.

```
# CMiramos los casos y Muertes por Grupos.
###M
ccasos <- aggregate(Casos~Grupo,data=dfcovidgrupos,FUN=mean)
# ##Media por grupo
mmuertes <- aggregate(Muertes~Grupo,data=dfcovidgrupos,FUN=mean)

totca <- sum(casosgrupos)
totmu <- sum(muertosgrupos)
#################################
#En porcentajes
casgrup <- casosgrupos
casmuer <- muertosgrupos
casosgrupos <- casosgrupos/totca
muertosgrupos <- muertosgrupos/totmu
```

12.Simulación de montecarlo para enfermar de COVID la muestra generada

Pasamos a simular un proceso de enfermar a los sujetos de nuestra muestra generada anteriormente mediante un método de Montecarlo y según la edad del sujeto.

```
################
N <- nrow(muestra)
simulacion <- data.frame(Edad = numeric(),Enfermo = numeric(),Fallecido=numeric())
#View(simulacion)
Enfermos <- c(0,0,0,0,0,0,0,0,0)
Fallecidos <- c(0,0,0,0,0,0,0,0,0)

for (i in 1:N){
  #Miramos la edad del individuo y a qué grupo pertenece:
  if (muestra$Edad[i] <=9){
    simulacion[i,1] <- muestra$Edad[i]
    #Miramos si va a enfermar o no:
    enferma <- runif(1,0,1)
    if (enferma <= casosgrupos[1]){
      simulacion[i,2] <-  1
      # Ahora miramos si fallece o no:
      muere <- runif(1,0,1)
      if (muere <= muertosgrupos[1]){
        simulacion[i,3] <-  1

      } else {
        simulacion[i,2] <- 1
        simulacion[i,3] <- 0
      }

    } else {
      simulacion[i,2] <- 0
      simulacion[i,3] <- 0
    }
  }

  else if ((muestra$Edad[i]>9) & (muestra$Edad[i]<=19)){
    simulacion[i,1] <- muestra$Edad[i]
    #Miramos si va a enfermar o no:
    enferma <- runif(1,0,1)
    if (enferma <= casosgrupos[2]){
      simulacion[i,2] <-  1
      # Ahora miramos si fallece o no:
      muere <- runif(1,0,1)
      if (muere <= muertosgrupos[2]){
        simulacion[i,3] <-  1

      } else {
        simulacion[i,2] <- 1
        simulacion[i,3] <- 0
      }

    } else {
      simulacion[i,2] <- 0
      simulacion[i,3] <- 0

  }}
  else if ((muestra$Edad[i]>19) & (muestra$Edad[i]<=29)){
```

```
  simulacion[i,1] <- muestra$Edad[i]
  #Miramos si va a enfermar o no:
  enferma <- runif(1,0,1)
  if (enferma <= casosgrupos[3]){
    simulacion[i,2] <-  1
    # Ahora miramos si fallece o no:
    muere <- runif(1,0,1)
    if (muere <= muertosgrupos[3]){
      simulacion[i,3] <-  1

    } else {
      simulacion[i,2] <- 1
      simulacion[i,3] <- 0
    }

  } else {
    simulacion[i,2] <- 0
    simulacion[i,3] <- 0

}}
else if ((muestra$Edad[i]>29) & (muestra$Edad[i]<=39)){
  simulacion[i,1] <- muestra$Edad[i]
  #Miramos si va a enfermar o no:
  enferma <- runif(1,0,1)
  if (enferma <= casosgrupos[4]){
    simulacion[i,2] <-  1
    # Ahora miramos si fallece o no:
    muere <- runif(1,0,1)
    if (muere <= muertosgrupos[4]){
      simulacion[i,3] <-  1

    } else {
      simulacion[i,2] <- 1
      simulacion[i,3] <- 0
    }

  } else {
    simulacion[i,2] <- 0
    simulacion[i,3] <- 0

}}
else if ((muestra$Edad[i]>39) & (muestra$Edad[i]<=49)){
  simulacion[i,1] <- muestra$Edad[i]
  #Miramos si va a enfermar o no:
  enferma <- runif(1,0,1)
  if (enferma <= casosgrupos[5]){
    simulacion[i,2] <-  1
    # Ahora miramos si fallece o no:
    muere <- runif(1,0,1)
    if (muere <= muertosgrupos[5]){
      simulacion[i,3] <-  1

    } else {
```

```
      simulacion[i,2] <- 1
      simulacion[i,3] <- 0
    }

  } else {
    simulacion[i,2] <- 0
    simulacion[i,3] <- 0

}}
else if ((muestra$Edad[i]>49) & (muestra$Edad[i]<=59)){
  simulacion[i,1] <- muestra$Edad[i]
  #Miramos si va a enfermar o no:
  enferma <- runif(1,0,1)
  if (enferma <= casosgrupos[6]){
    simulacion[i,2] <-  1
    # Ahora miramos si fallece o no:
    muere <- runif(1,0,1)
    if (muere <= muertosgrupos[6]){
      simulacion[i,3] <-  1

    } else {
      simulacion[i,2] <- 1
      simulacion[i,3] <- 0
    }

  } else {
    simulacion[i,2] <- 0
    simulacion[i,3] <- 0

}}
else if ((muestra$Edad[i]>59) & (muestra$Edad[i]<=69)){
  simulacion[i,1] <- muestra$Edad[i]
  #Miramos si va a enfermar o no:
  enferma <- runif(1,0,1)
  if (enferma <= casosgrupos[7]){
    simulacion[i,2] <-  1
    # Ahora miramos si fallece o no:
    muere <- runif(1,0,1)
    if (muere <= muertosgrupos[7]){
      simulacion[i,3] <-  1

    } else {
      simulacion[i,2] <- 1
      simulacion[i,3] <- 0
    }

  } else {
    simulacion[i,2] <- 0
    simulacion[i,3] <- 0

}}
else if ((muestra$Edad[i]>69) & (muestra$Edad[i]<=79)){
  simulacion[i,1] <- muestra$Edad[i]
```

```
    #Miramos si va a enfermar o no:
    enferma <- runif(1,0,1)
    if (enferma <= casosgrupos[8]){
      simulacion[i,2] <-  1
      # Ahora miramos si fallece o no:
      muere <- runif(1,0,1)
      if (muere <= muertosgrupos[8]){
        simulacion[i,3] <-  1

      } else {
        simulacion[i,2] <- 1
        simulacion[i,3] <- 0
      }


    } else {
      simulacion[i,2] <- 0
      simulacion[i,3] <- 0

  }}
  else if (muestra$Edad[i]>79){
    simulacion[i,1] <- muestra$Edad[i]
    #Miramos si va a enfermar o no:
    enferma <- runif(1,0,1)
    if (enferma <= casosgrupos[9]){
      simulacion[i,2] <-  1
      # Ahora miramos si fallece o no:
      muere <- runif(1,0,1)
      if (muere <= muertosgrupos[9]){
        simulacion[i,3] <-  1

      } else {
        simulacion[i,2] <- 1
        simulacion[i,3] <- 0
      }

    } else {
      simulacion[i,2] <- 0
      simulacion[i,3] <- 0

    }



  }}
```

Ya tenemos la pandemia simulada para la muestra de edades anterior.

```
print('Casos totales de COVID en la muestra simulada:')
```

```
## [1] "Casos totales de COVID en la muestra simulada:"
```

```
print(sum(simulacion$Fallecido))
```

```
## [1] 436
```

```
print('Fallecidos de COVID totales en la muestra simulada:')
```

```
## [1] "Fallecidos de COVID totales en la muestra simulada:"
```

```
print(sum(simulacion$Enfermo))
```

```
## [1] 5626
```

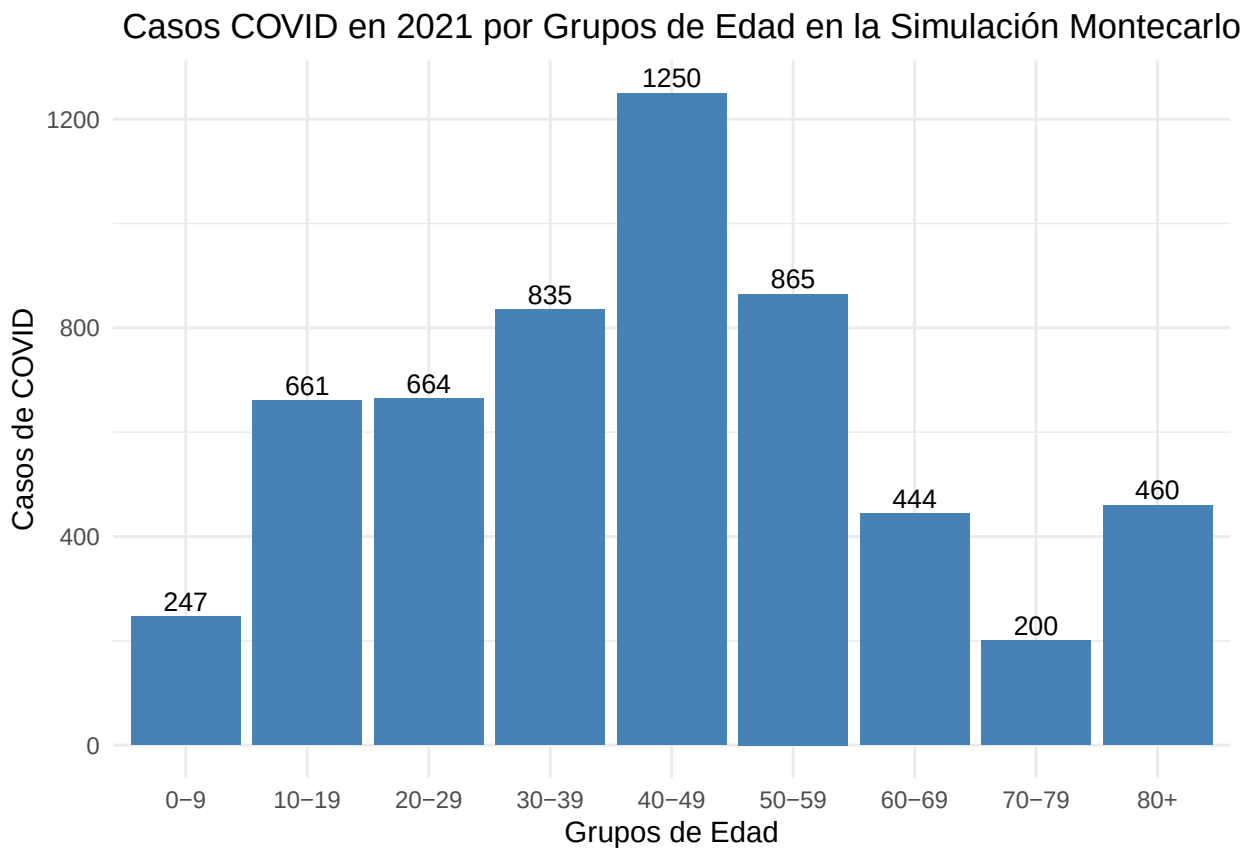
Añadimos una columna para indicar a qué grupo de edades pertenece cada individuo

```
#Añadimos una columna para indicar a qué grupo de edades pertenece cada individuo.
for(i in 1:50000){

  if (simulacion$Edad[i] <=9){
    simulacion$Grupo[i] ='0-9'
     } else if ((simulacion$Edad[i]>9) & (simulacion$Edad[i]<=19)){
    simulacion$Grupo[i] ='10-19'
    }
    else if ((simulacion$Edad[i]>19) & (simulacion$Edad[i]<=29)){
            simulacion$Grupo[i] ='20-29'}
    else if ((simulacion$Edad[i]>29) & (simulacion$Edad[i]<=39)){
      simulacion$Grupo[i] ='30-39'
    }

    else if ((simulacion$Edad[i]>39) & (simulacion$Edad[i]<=49)){
      simulacion$Grupo[i] ='40-49'
    }

    else if ((simulacion$Edad[i]>49) & (simulacion$Edad[i]<=59)){
      simulacion$Grupo[i] ='50-59'
    }

    else if ((simulacion$Edad[i]>59) & (simulacion$Edad[i]<=69)){
      simulacion$Grupo[i] ='60-69'
    }

    else if ((simulacion$Edad[i]>69) & (simulacion$Edad[i]<=79)){
      simulacion$Grupo[i] ='70-79'
    }


  else if (simulacion$Edad[i]>79){
      simulacion$Grupo[i] ='80+'}
      }
```

Vamos a Graficar los casos de COVID para la simulación por Grupos de Edad.

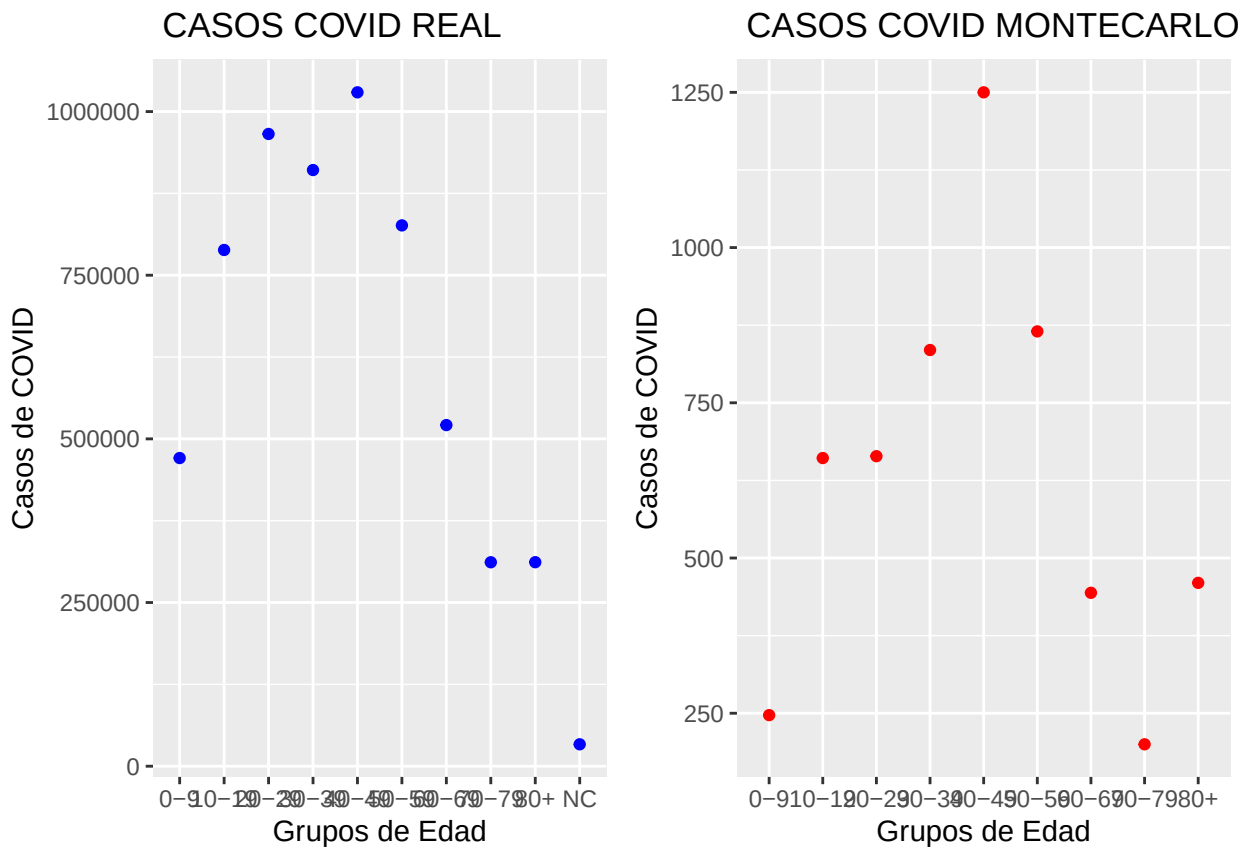
```
#Gráfico de casos por grupos y luego hacer muertes ********
#View(simulacion)
A = simulacion %>%  group_by(Grupo)
A = A %>% summarise(n=sum(Enfermo))
#A = A %>% summarise(n=sum(n))

ggplot(data=A, aes(x=Grupo, y=n)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=n), vjust=-0.3, size=3.5)+
```

```
  ggtitle(" Casos COVID en 2021 por Grupos de Edad en la Simulación Montecarlo")  +
  ylab("Casos de COVID") + xlab("Grupos de Edad") +
  theme_minimal()
```



Rescatamos el gráfico real hecho anteriormente

```
library(ggplot2)
library(cowplot)


graf1 <- ggplot() +
  geom_point(data=dataset2, aes(x=gruposedad,y=casostot),color='Blue') +
  ggtitle(" CASOS COVID REAL")  +
  ylab("Casos de COVID") + xlab("Grupos de Edad")

graf2 <- ggplot() +
  geom_point(data=A,aes(x=Grupo, y=n), color='Red') +
  ggtitle(" CASOS COVID MONTECARLO")  +
  ylab("Casos de COVID") + xlab("Grupos de Edad")
plot_grid(graf1,graf2)
```

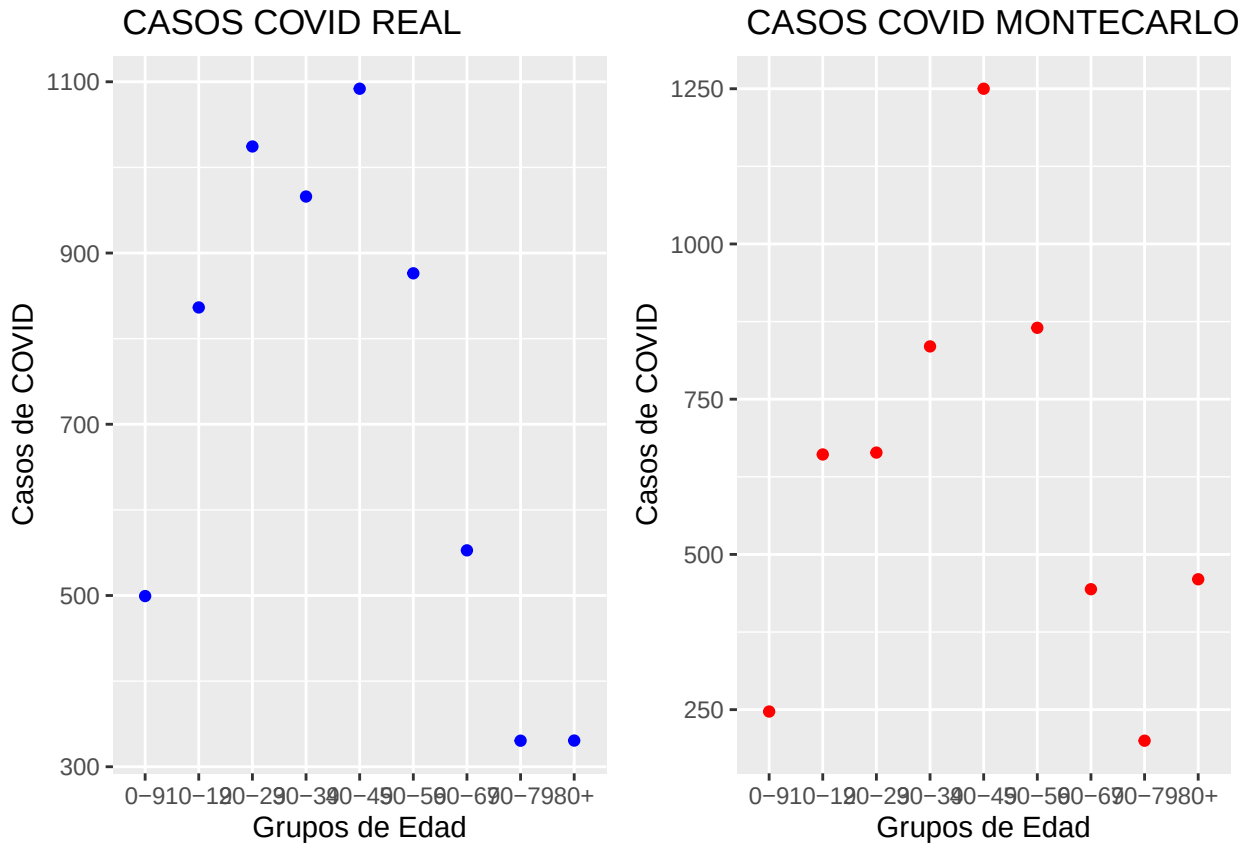


vemos que las gráficas guardan un patrón bastante similar, teniendo en cuenta que son casos totales sobre poblaciones de distinto tamaño y que, en el caso real, tenemos el NC. Vemos que en los dos casos hay un pico en el grupo de edades de 40-49 años, y que el caso inferior es el grupo de 70-79. hay que tener en cuenta que en la simulación de montecarlo he puesto simulaciones de edades hasta 110 años, cosa que, quizá, podría corregirse con datos empíricos.

Ahora volvemos a rescatar la gráfica de casos reales y la escalamos a 50 mil personas, es decir, dividimos por el porcentaje del grupo de edad -sin tener en cuenta el NC- y multiplicamos por 50 mil. Una vez hecho esto, volvemos a comparar el gráfico inferido de la realidad con el de la simulación para 50 mil sujetos:

```
#Gráfico de casos por grupos dividiendo por el porcentaje de cada grupo ********

library(ggplot2)
library(cowplot)

datasete <- dataset2[1:9,1:2]
#Introducimos la corrección de los casos reales en España a una muestra de 50 mil
#personas. "grupedad" tiene los tanto por uno de cada grupo de edad.
datasete$casostot <- (casosgrupos)*(50000/tot21)*totca
datasete$Muertes <- (muertosgrupos)*(50000/tot21)*totmu
graf1 <- ggplot() +
  geom_point(data=datasete, aes(x=gruposedad[-10],y=casostot),color='Blue') +
  ggtitle(" CASOS COVID REAL")  +
  ylab("Casos de COVID") + xlab("Grupos de Edad")

graf2 <- ggplot() +
  geom_point(data=A,aes(x=Grupo, y=n), color='Red') +
  ggtitle(" CASOS COVID MONTECARLO")  +
```

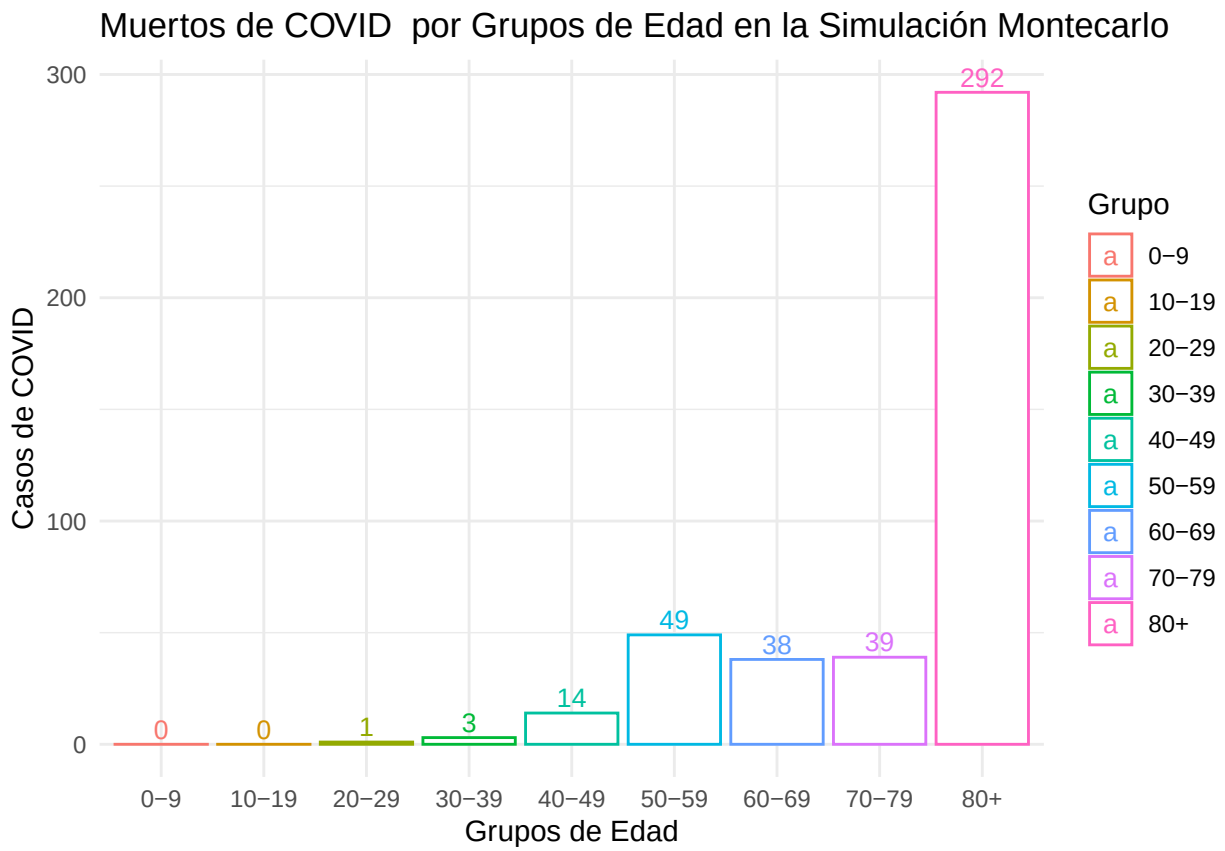
```
  ylab("Casos de COVID") + xlab("Grupos de Edad")
plot_grid(graf1,graf2)
```



Vemos que ambos gráficos son muy similares, tanto en su curva como en sus magnitudes para cada grupo, salvo el grupo +80, que por motivos de extender hasta 110 años las edades en la simulación, han podido generar personas más longevas de lo normal, pues el límite que el censo nos daba era hasta 100 y luego +100.

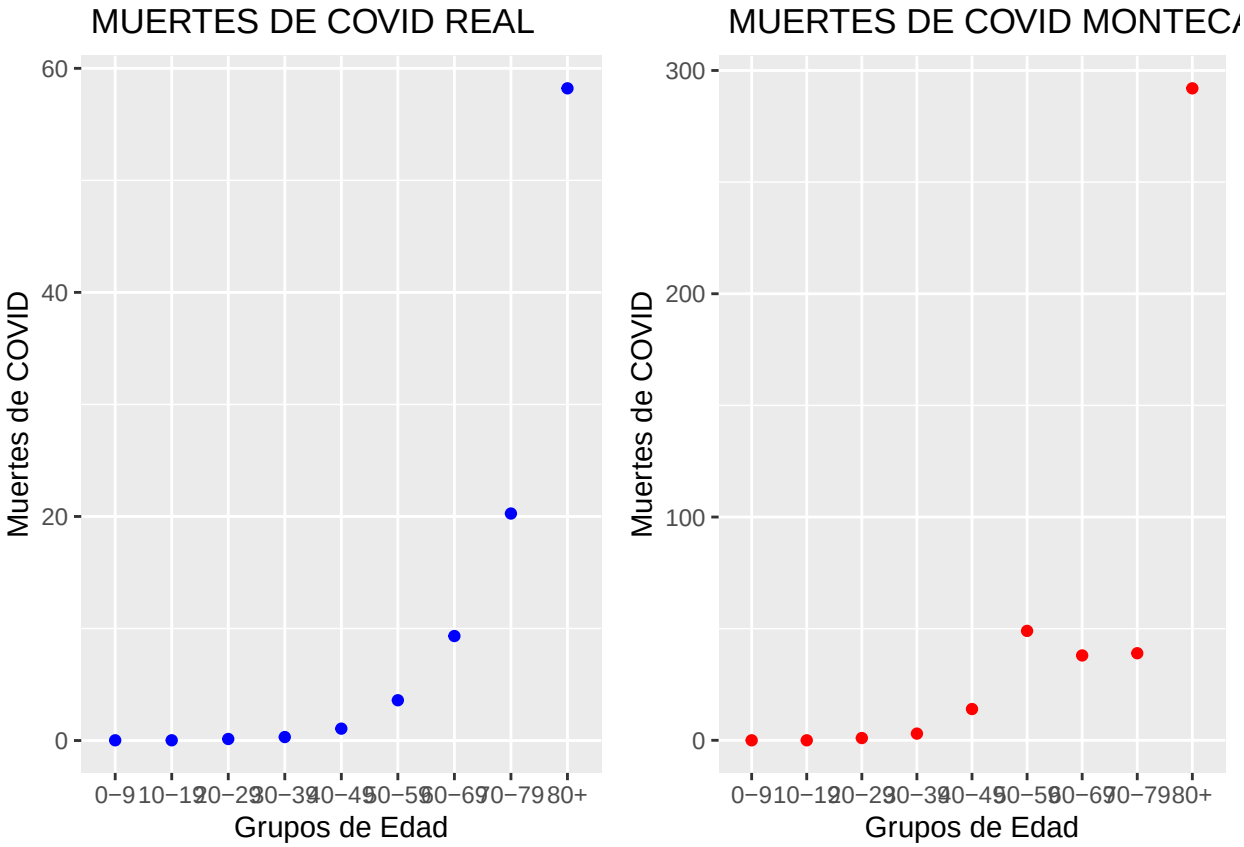
Pasamos a analizar los casos de muerte por COVID en la simulación de montecarlo:

```
B = simulacion %>% group_by(Grupo)
B = B %>% summarise(n=sum(Fallecido))
#datasete <- dataset2[1:9,1:2]
ggplot(data=B, aes(x=Grupo, y=n,color=Grupo)) +
  geom_bar(stat="identity", fill="white")+
  geom_text(aes(label=n), vjust=-0.3, size=3.5)+
  ggtitle("Muertos de COVID  por Grupos de Edad en la Simulación Montecarlo")  +
  ylab("Casos de COVID") + xlab("Grupos de Edad") +
  theme_minimal()
```



Graficamos el total de casos de muerte para la simulación y los datos del INE sin escalar los reales a 50 mil casos. Solo compararemos los aspectos del comportamiento de los grupos de edad.:

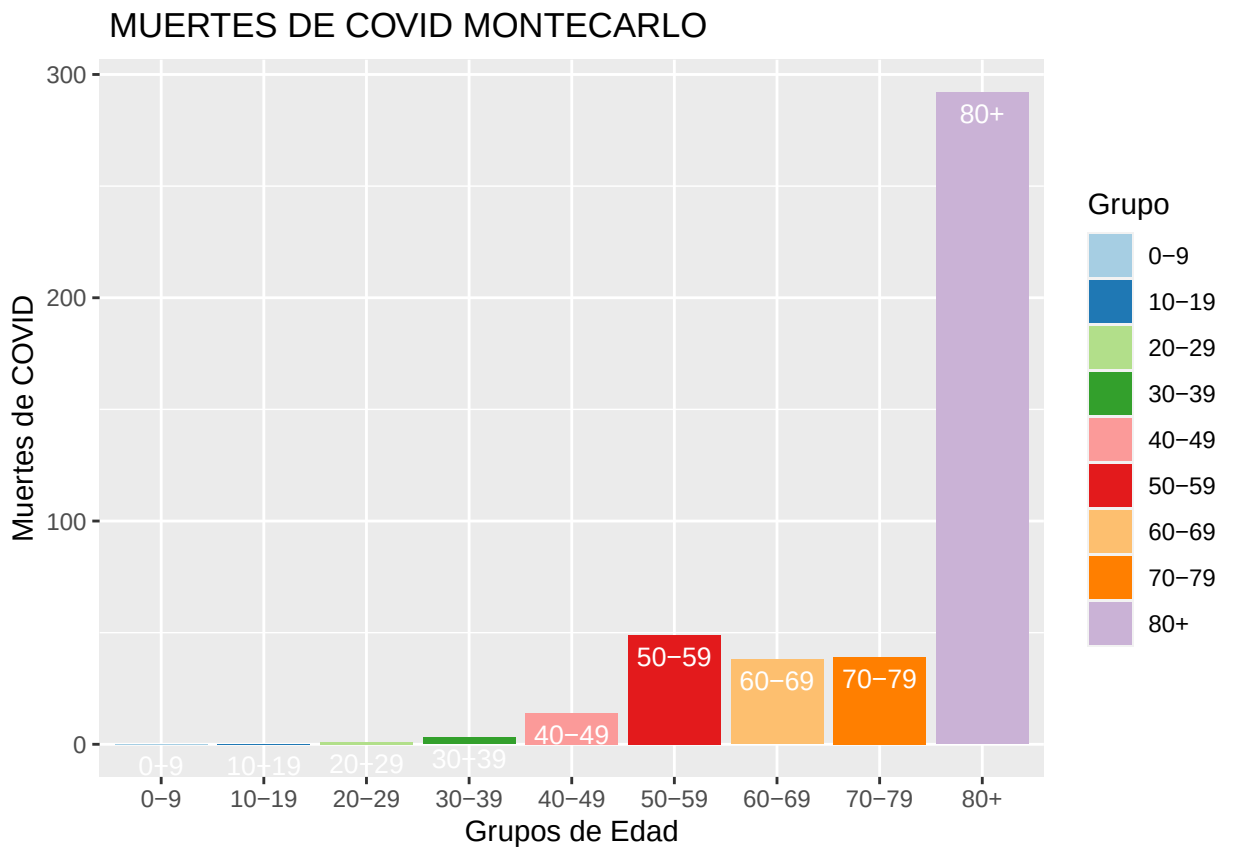
```
library(ggplot2)
library(cowplot)


graf3 <- ggplot() +
  geom_point(data=datasete, aes(x=gruposedad[-10],y=datasete$Muertes),color='Blue') +
  ggtitle(" MUERTES DE COVID REAL")  +
  ylab("Muertes de COVID") + xlab("Grupos de Edad")

graf4 <- ggplot() +
  geom_point(data=B,aes(x=Grupo, y=n), color='Red') +
  ggtitle(" MUERTES DE COVID MONTECARLO")  +
  ylab("Muertes de COVID") + xlab("Grupos de Edad")
plot_grid(graf3,graf4)
```



En cuanto a los casos de muerte, la simulación tambien sale bien, salvo errores propios de la deriva o la estocasticidad en el tamaño muestral. También hay que tener en cuenta que en la simulación de montecarlo he considerado que podía generar sujetos de hasta 110 años, tomándolos en el grupo +80. De otro lado, el agrupamiento en edades de 10 en 10, en la simulación, hace menos fino el ajuste que un muestreo real. Salvo el grupo +80 que se dispara en la simulación, el resto están por debajo de 50 y entre la zona del 25, que más o menos coincide con el 20 máximo del grupo 70-79 inferido de datos reales. En este caso también el grupo +80 se dispara, si bien no tanto como en la simulación.

En los dataframes A y B tenemos los casos y muertes de COVID de la simulación Montecarlo por grupos de edades. Vamos a graficarlos juntos:

```
ByGrupos <- merge.data.frame(A, B, by.x = "Grupo", by.y = "Grupo")
#View(ByGrupos)
ggplot(data=ByGrupos, aes(x=Grupo, y=n.y, fill=Grupo)) +
  geom_bar(stat="identity", position=position_dodge())+
  geom_text(aes(label=Grupo), vjust=1.6, color="white",
            position = position_dodge(0.9), size=3.5)+
  scale_fill_brewer(palette="Paired")+  ggtitle(" MUERTES DE COVID MONTECARLO")  +
  ylab("Muertes de COVID") + xlab("Grupos de Edad")
```




```
  theme_minimal()
```

```
## List of 93
##  $ line                        :List of 6
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                        :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                        :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
```

```
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                     : NULL
##  $ aspect.ratio              : NULL
##  $ axis.title                : NULL
##  $ axis.title.x              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top          :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom       : NULL
##  $ axis.title.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left         : NULL
##  $ axis.title.y.right        :List of 11
```

```
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top          :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
```

```
##  $ axis.text.x.bottom        : NULL
##  $ axis.text.y               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left          : NULL
##  $ axis.text.y.right         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x              : NULL
##  $ axis.ticks.x.top          : NULL
##  $ axis.ticks.x.bottom       : NULL
##  $ axis.ticks.y              : NULL
##  $ axis.ticks.y.left         : NULL
##  $ axis.ticks.y.right        : NULL
##  $ axis.ticks.length         : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x       : NULL
##  $ axis.ticks.length.x.top   : NULL
##  $ axis.ticks.length.x.bottom: NULL
##  $ axis.ticks.length.y       : NULL
##  $ axis.ticks.length.y.left  : NULL
##  $ axis.ticks.length.y.right : NULL
##  $ axis.line                 : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x               : NULL
##  $ axis.line.x.top           : NULL
##  $ axis.line.x.bottom        : NULL
##  $ axis.line.y               : NULL
##  $ axis.line.y.left          : NULL
##  $ axis.line.y.right         : NULL
```

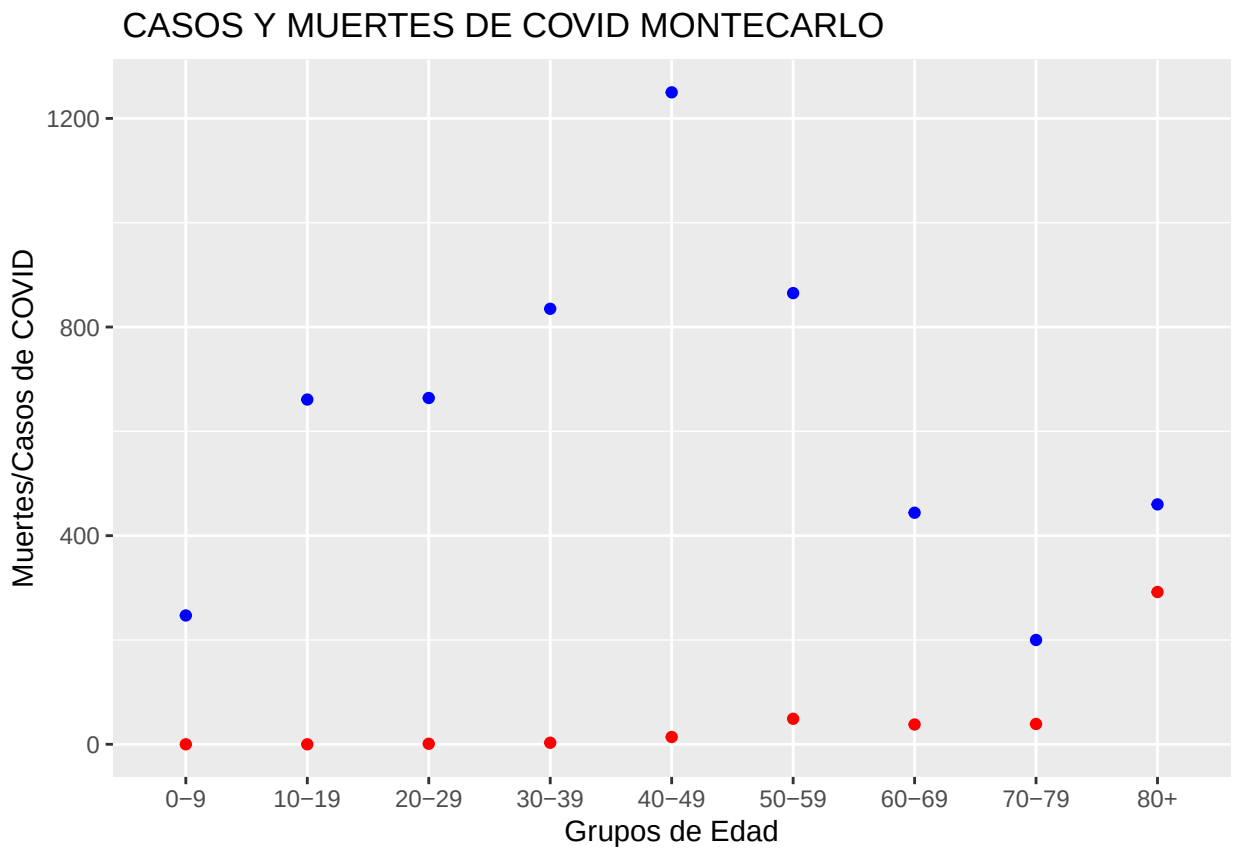
```
##  $ legend.background         : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin             : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing            : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x          : NULL
##  $ legend.spacing.y          : NULL
##  $ legend.key                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size           : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height         : NULL
##  $ legend.key.width          : NULL
##  $ legend.text               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.align         : NULL
##  $ legend.title              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.align        : NULL
##  $ legend.position           : chr "right"
##  $ legend.direction          : NULL
##  $ legend.justification      : chr "center"
##  $ legend.box                : NULL
##  $ legend.box.just           : NULL
##  $ legend.box.margin         : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing        : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ panel.background          : list()
```

```
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.border              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.spacing             : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ panel.spacing.x           : NULL
##  $ panel.spacing.y           : NULL
##  $ panel.grid                :List of 6
##   ..$ colour       : chr "grey92"
##   ..$ size         : NULL
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major          : NULL
##  $ panel.grid.minor          :List of 6
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.5
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major.x        : NULL
##  $ panel.grid.major.y        : NULL
##  $ panel.grid.minor.x        : NULL
##  $ panel.grid.minor.y        : NULL
##  $ panel.ontop               : logi FALSE
##  $ plot.background           : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ plot.title                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.title.position       : chr "panel"
##  $ plot.subtitle             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
```

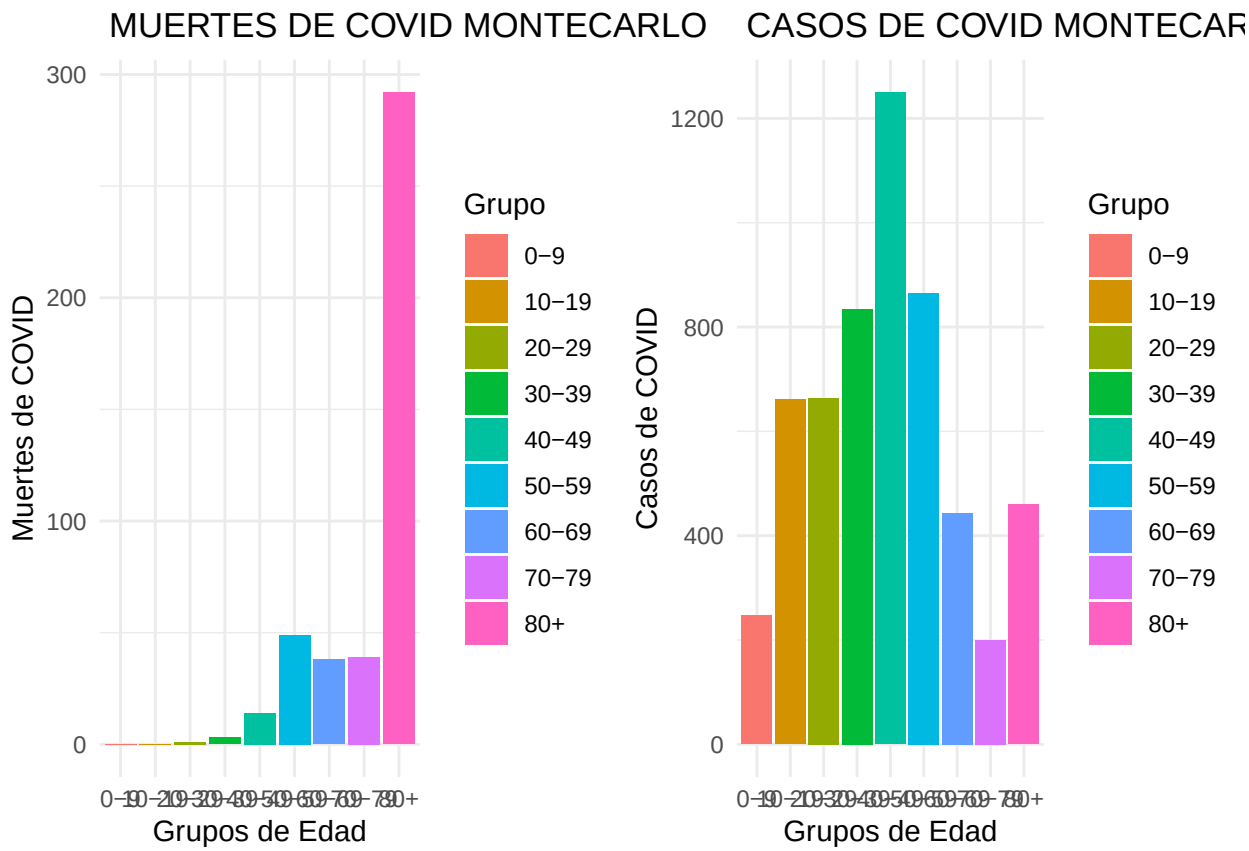
```
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption.position     : chr "panel"
##  $ plot.tag                  :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ strip.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ strip.background.x        : NULL
##  $ strip.background.y        : NULL
##  $ strip.placement           : chr "inside"
##  $ strip.text                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey10"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 4.4points 4.4points 4.4points 4.4points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
```

```
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.text.x              : NULL
##  $ strip.text.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.switch.pad.grid     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.switch.pad.wrap     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.text.y.left         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

```
ggplot() +
  geom_point(data=ByGrupos,aes(x=Grupo, y=n.x),color='Blue') +
  geom_point(data=ByGrupos,aes(x=Grupo, y=n.y),color='Red')+
  ggtitle(" CASOS Y MUERTES DE COVID MONTECARLO")  +
  ylab("Muertes/Casos de COVID") + xlab("Grupos de Edad")
```




```
graf5 <- ggplot() + geom_bar(data=ByGrupos,aes(x=Grupo, y=n.y, fill=Grupo) ,stat="identity")+
  theme_minimal() + ggtitle(" MUERTES DE COVID MONTECARLO")  +
  ylab("Muertes de COVID") + xlab("Grupos de Edad")
graf6 <- ggplot() + geom_bar(data=ByGrupos,aes(x=Grupo, y=n.x, fill=Grupo) ,stat="identity") +
   ggtitle(" CASOS DE COVID MONTECARLO")  +
  ylab("Casos de COVID") + xlab("Grupos de Edad") +
  theme_minimal()
plot_grid(graf5,graf6)
```



Pasamos a realizar un análisis ANOVA de los casos y muertes por grupos de edad:

```
# LISTO PARA UN ANOVA:
# Calculemos ahora la media cada grupo.
##
aggregate(Enfermo~Grupo,data=simulacion,FUN=mean)
```

```
##   Grupo    Enfermo
## 1   0-9 0.07243402
## 2 10-19 0.13995342
## 3 20-29 0.14674033
## 4 30-39 0.15411591
## 5 40-49 0.17453225
## 6 50-59 0.13058575
## 7 60-69 0.08390023
## 8 70-79 0.05260389
## 9   80+ 0.05086245
```

```
# ##Media por grupo
aggregate(Fallecido~Grupo,data=simulacion,FUN=mean)
```

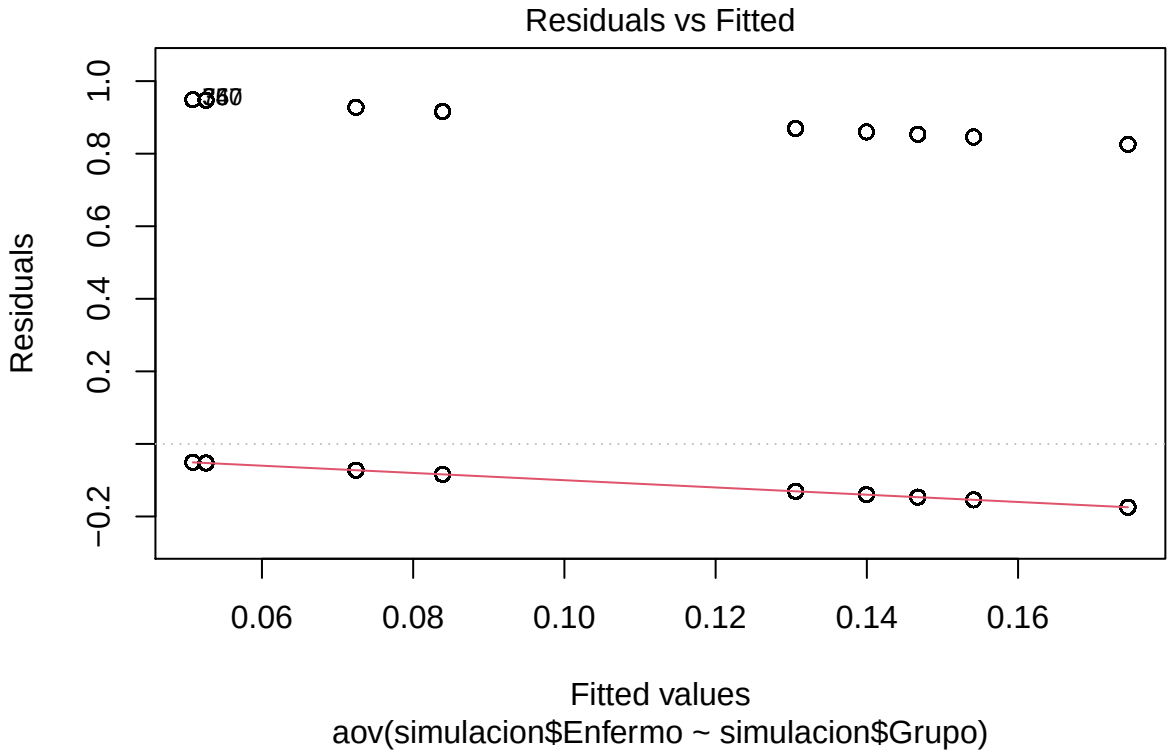
```
##   Grupo    Fallecido
## 1   0-9 0.0000000000
## 2 10-19 0.0000000000
## 3 20-29 0.0002209945
## 4 30-39 0.0005537099
## 5 40-49 0.0019547612
## 6 50-59 0.0073973430
```

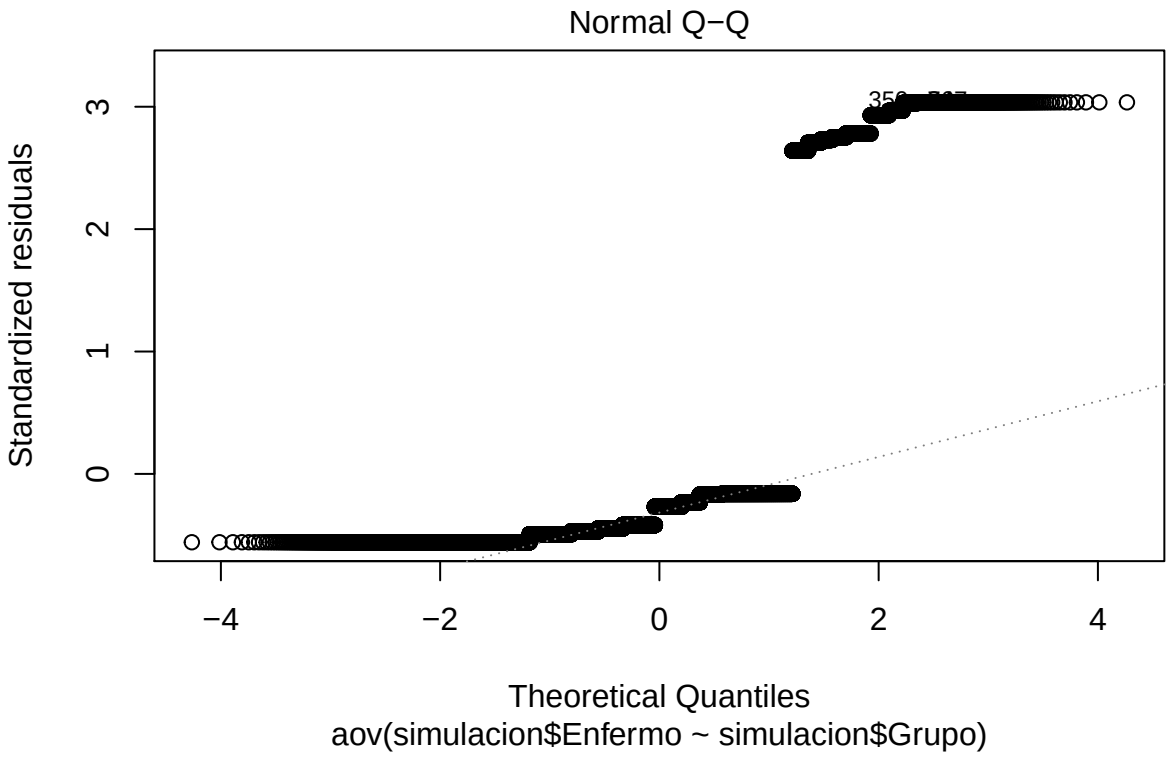
```
## 7 60-69 0.0071806500
## 8 70-79 0.0102577591
## 9   80+ 0.0322865989
```

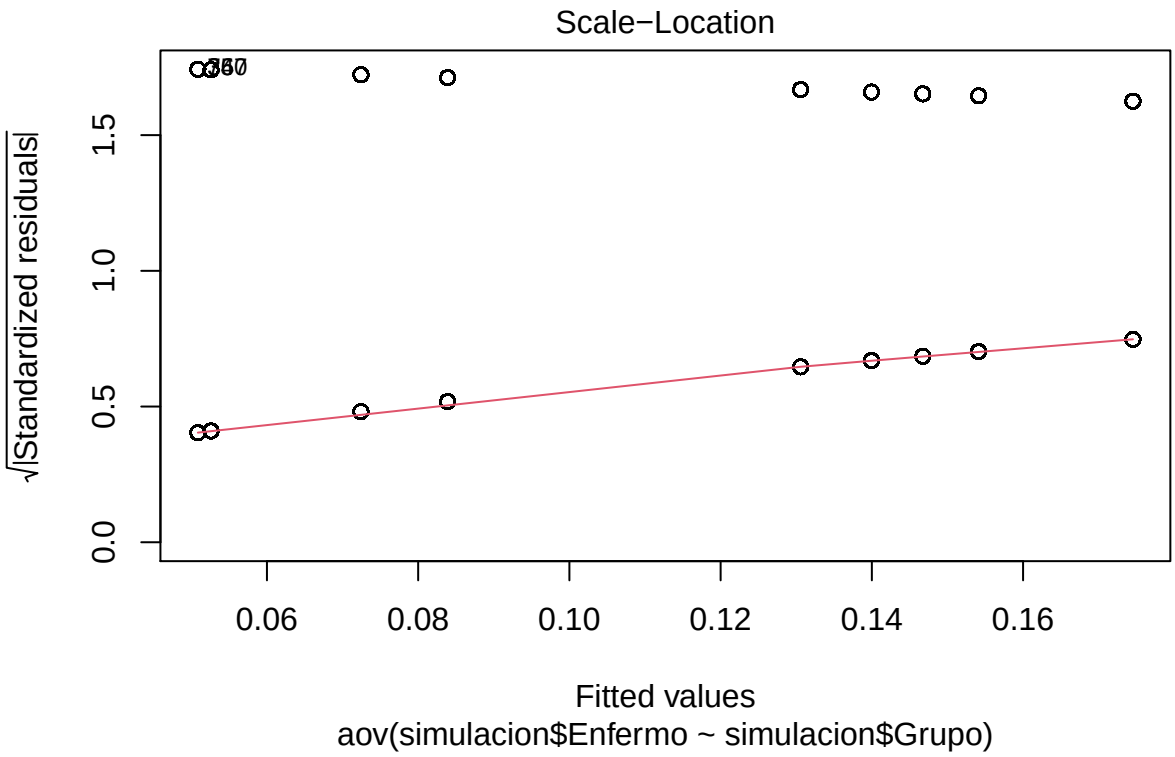
```
anova <- aov(simulacion$Enfermo~simulacion$Grupo,data=simulacion)
summary(anova)
```

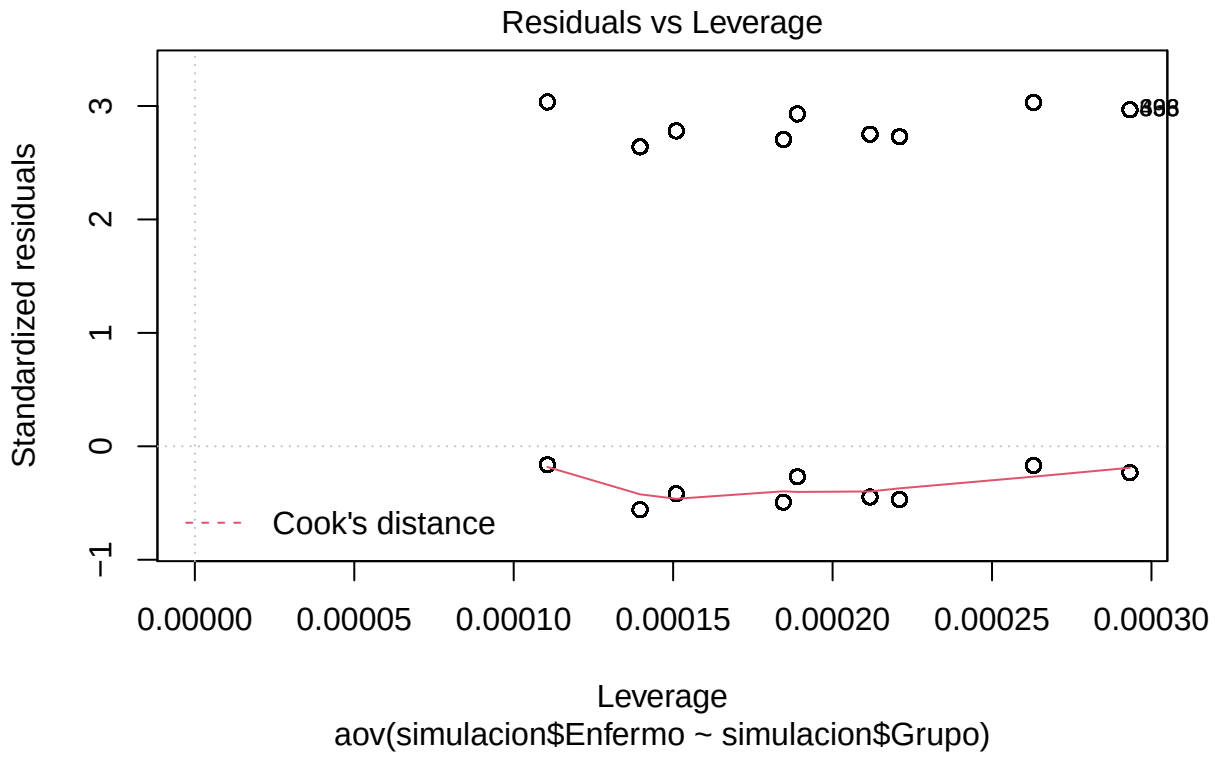
```
##                     Df Sum Sq Mean Sq F value Pr(>F)
## simulacion$Grupo     8    106  13.222   135.2 <2e-16 ***
## Residuals        49991   4887   0.098
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(anova)
```




Normal Q−Q
Standardized residuals
0
1
2
3
−4
−2
0
2
4
Theoretical Quantiles
aov(simulacion$Enfermo ~ simulacion$Grupo)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.06
0.08
0.10
0.12
0.14
0.16
Fitted values
aov(simulacion$Enfermo ~ simulacion$Grupo)



Podemos concluir que cada grupo de personas recibe un impacto diferente a contagiarse de COVID.

```
lmgrupoenfermo <- lm(simulacion$Enfermo~simulacion$Grupo)
summary(lmgrupoenfermo)
```

```
##
## Call:
## lm(formula = simulacion$Enfermo ~ simulacion$Grupo)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17453 -0.14674 -0.08390 -0.05086  0.94914
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.072434   0.005354  13.528  < 2e-16 ***
## simulacion$Grupo10-19    0.067519   0.007026   9.610  < 2e-16 ***
## simulacion$Grupo20-29    0.074306   0.007090  10.480  < 2e-16 ***
## simulacion$Grupo30-39    0.081682   0.006835  11.951  < 2e-16 ***
## simulacion$Grupo40-49    0.102098   0.006505  15.695  < 2e-16 ***
## simulacion$Grupo50-59    0.058152   0.006590   8.824  < 2e-16 ***
## simulacion$Grupo60-69    0.011466   0.006866   1.670 0.094928 .
## simulacion$Grupo70-79   -0.019830   0.007374  -2.689 0.007168 **
## simulacion$Grupo80+     -0.021572   0.006283  -3.433 0.000597 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.3127 on 49991 degrees of freedom
## Multiple R-squared:  0.02119,    Adjusted R-squared:  0.02103
## F-statistic: 135.2 on 8 and 49991 DF,  p-value: < 2.2e-16
```

```
anova(lmgrupoenfermo)
```

```
## Analysis of Variance Table
##
## Response: simulacion$Enfermo
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## simulacion$Grupo      8  105.8 13.2221  135.25 < 2.2e-16 ***
## Residuals         49991 4887.2  0.0978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#plot(simulacion$Grupo, residuals(lmgrupoenfermo), ylab = "residuos");abline(h = 0)
df_residuos <- data.frame(simulacion$Grupo, residuals(lmgrupoenfermo))
#View(df_residuos)
```

No existe una relación lineal entre los grupos de edad y los casos de COVID, aunque sí una diferencia en su varianza que hace que cada grupo reciba un impacto diferente a la pandemia.

realizamos un test de Barlett y un oneway.test. La prueba de Bartlett se utiliza para probar si k muestras provienen de poblaciones con la misma varianza

```
bartlett.test(simulacion$Enfermo~simulacion$Grupo)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  simulacion$Enfermo by simulacion$Grupo
## Bartlett's K-squared = 4055.6, df = 8, p-value < 2.2e-16
```

```
oneway.test(simulacion$Enfermo~simulacion$Grupo)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  simulacion$Enfermo and simulacion$Grupo
## F = 155.35, num df = 8, denom df = 18628, p-value < 2.2e-16
```

```
#UNED pag. 286. Sale un p-valor muy bajo.

## En los casos en los que se detectan diferencias significativas entre
#las medias de los grupos, se utilizarían métodos de comparaciones múltiples
#y correcciones. Las más recomendadas son Corrección de Holm y TukeyHSD.
#En el test que exponemos el tamaño muestral de cada tratamiento debe ser el
#mismo. Pueden haber tratamiento, de entre los 3, que tengan en mismo efecto.
```

Vistos los resultados de las pruebas, no existe igualdad de varianza en los grupos de edades frente a los casos de COVID.

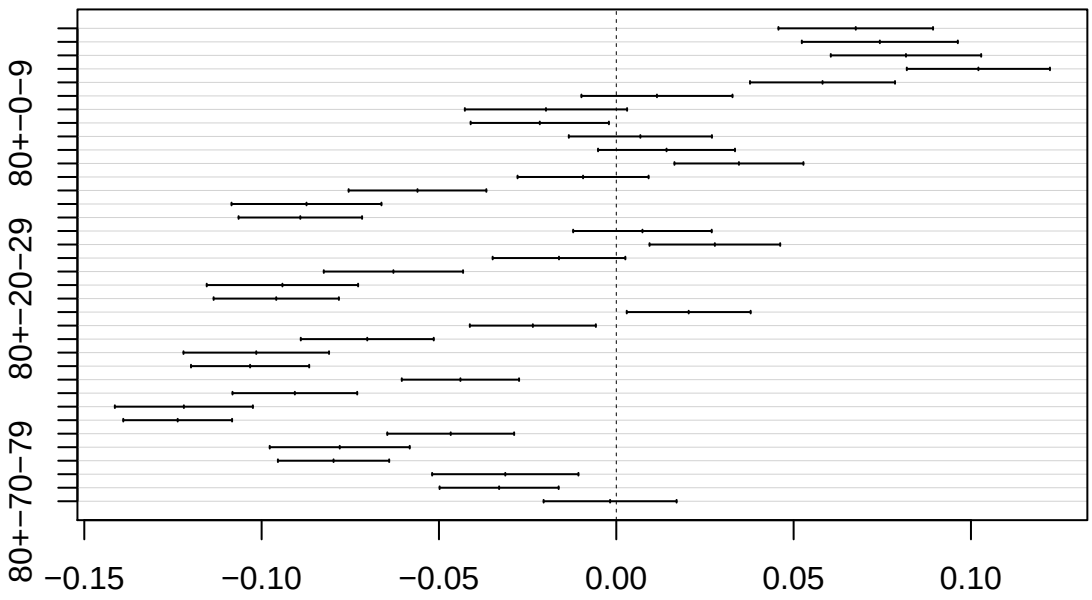
Mediante una prueba de Tukey HSD comprobamos que las medias no cumplen la hipótesis nula.

```
TukeyHSD(anova)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
```

```
## Fit: aov(formula = simulacion$Enfermo ~ simulacion$Grupo, data = simulacion)
##
## $`simulacion$Grupo`
##                       diff          lwr          upr     p adj
## 10-19-0-9     0.067519402  0.045725911  0.089312893 0.0000000
## 20-29-0-9     0.074306314  0.052313816  0.096298812 0.0000000
## 30-39-0-9     0.081681892  0.060482566  0.102881218 0.0000000
## 40-49-0-9     0.102098236  0.081920529  0.122275943 0.0000000
## 50-59-0-9     0.058151731  0.037711431  0.078592032 0.0000000
## 60-69-0-9     0.011466209 -0.009830378  0.032762796 0.7650580
## 70-79-0-9    -0.019830125 -0.042703602  0.003043352 0.1515476
## 80+-0-9      -0.021571567 -0.041060342 -0.002082792 0.0173448
## 20-29-10-19   0.006786912 -0.013387144  0.026960968 0.9816089
## 30-39-10-19   0.014162490 -0.005143829  0.033468810 0.3571445
## 40-49-10-19   0.034578834  0.016400209  0.052757459 0.0000001
## 50-59-10-19  -0.009367671 -0.027837334  0.009101992 0.8196434
## 60-69-10-19  -0.056053193 -0.075466260 -0.036640125 0.0000000
## 70-79-10-19  -0.087349527 -0.108480512 -0.066218542 0.0000000
## 80+-10-19    -0.089090969 -0.106501739 -0.071680199 0.0000000
## 30-39-20-29   0.007375578 -0.012155108  0.026906265 0.9625111
## 40-49-20-29   0.027791922  0.009375187  0.046208658 0.0000996
## 50-59-20-29  -0.016154583 -0.034858651  0.002549486 0.1552358
## 60-69-20-29  -0.062840105 -0.082476319 -0.043203890 0.0000000
## 70-79-20-29  -0.094136439 -0.115472611 -0.072800266 0.0000000
## 80+-20-29    -0.095877881 -0.113537118 -0.078218644 0.0000000
## 40-49-30-39   0.020416344  0.002954454  0.037878233 0.0087710
## 50-59-30-39  -0.023530161 -0.041294835 -0.005765488 0.0013208
## 60-69-30-39  -0.070215683 -0.088959278 -0.051472088 0.0000000
## 70-79-30-39  -0.101512017 -0.122029661 -0.080994374 0.0000000
## 80+-30-39    -0.103253460 -0.119914495 -0.086592425 0.0000000
## 50-59-40-49  -0.043946505 -0.060478657 -0.027414353 0.0000000
## 60-69-40-49  -0.090632027 -0.108211867 -0.073052186 0.0000000
## 70-79-40-49  -0.121928361 -0.141388632 -0.102468090 0.0000000
## 80+-40-49    -0.123669803 -0.139009899 -0.108329708 0.0000000
## 60-69-50-59  -0.046685522 -0.064566149 -0.028804895 0.0000000
## 70-79-50-59  -0.077981856 -0.097714270 -0.058249442 0.0000000
## 80+-50-59    -0.079723299 -0.095407193 -0.064039404 0.0000000
## 70-79-60-69  -0.031296334 -0.051914455 -0.010678213 0.0000872
## 80+-60-69    -0.033037777 -0.049822391 -0.016253162 0.0000000
## 80+-70-79    -0.001741442 -0.020486432  0.017003547 0.9999986
```

```
plot(TukeyHSD(anova)) ####MIRAR: https://rpubs.com/Joaquin_AR/219148
```




```
#En el test de Tukey observamos que solo contiene al "0" el intervalo del efecto
#de los tratamientos/hospitales 2 y 3. {2,3}
pairwise.t.test(x=simulacion$Enfermo,g=simulacion$Grupo,p.adjust.method ="holm",pool.sd=TRUE,
                paired=FALSE,alternative="two.sided")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  simulacion$Enfermo and simulacion$Grupo
##
##       0-9     10-19   20-29   30-39   40-49   50-59   60-69   70-79
## 10-19 < 2e-16 -       -       -       -       -       -       -
## 20-29 < 2e-16 0.72440 -       -       -       -       -       -
## 30-39 < 2e-16 0.13734 0.72440 -       -       -       -       -
## 40-49 < 2e-16 5.1e-08 3.4e-05 0.00288 -       -       -       -
## 50-59 < 2e-16 0.47464 0.05735 0.00044 2.9e-15 -       -       -
## 60-69 0.47464 < 2e-16 < 2e-16 < 2e-16 < 2e-16 9.1e-15 -       -
## 70-79 0.05735 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 3.3e-05 -
## 80+   0.00537 < 2e-16 < 2e-16 < 2e-16 < 2e-16 < 2e-16 1.6e-08 0.77323
##
## P value adjustment method: holm
```

```
#Parece indicar significación en hospital 2 y 3.
```

Hacemos una prueba de Levene:

```
####Prueba de Levene. La prueba de Levene utiliza la desviación absoluta de las
#observaciones respecto de la mediana de cada. Y evalúa mediante un ANOVA si la
```

```r
#media de estas desviaciones absolutas es similar para todos los tratamientos.

med <- tapply(simulacion$Enfermo, simulacion$Grupo, median)
aresid <- abs(simulacion$Enfermo - med[simulacion$Grupo])
anova(lm(aresid ~ simulacion$Grupo))
```

```
## Analysis of Variance Table
##
## Response: aresid
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## simulacion$Grupo      8  105.8 13.2221  135.25 < 2.2e-16 ***
## Residuals         49991 4887.2  0.0978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

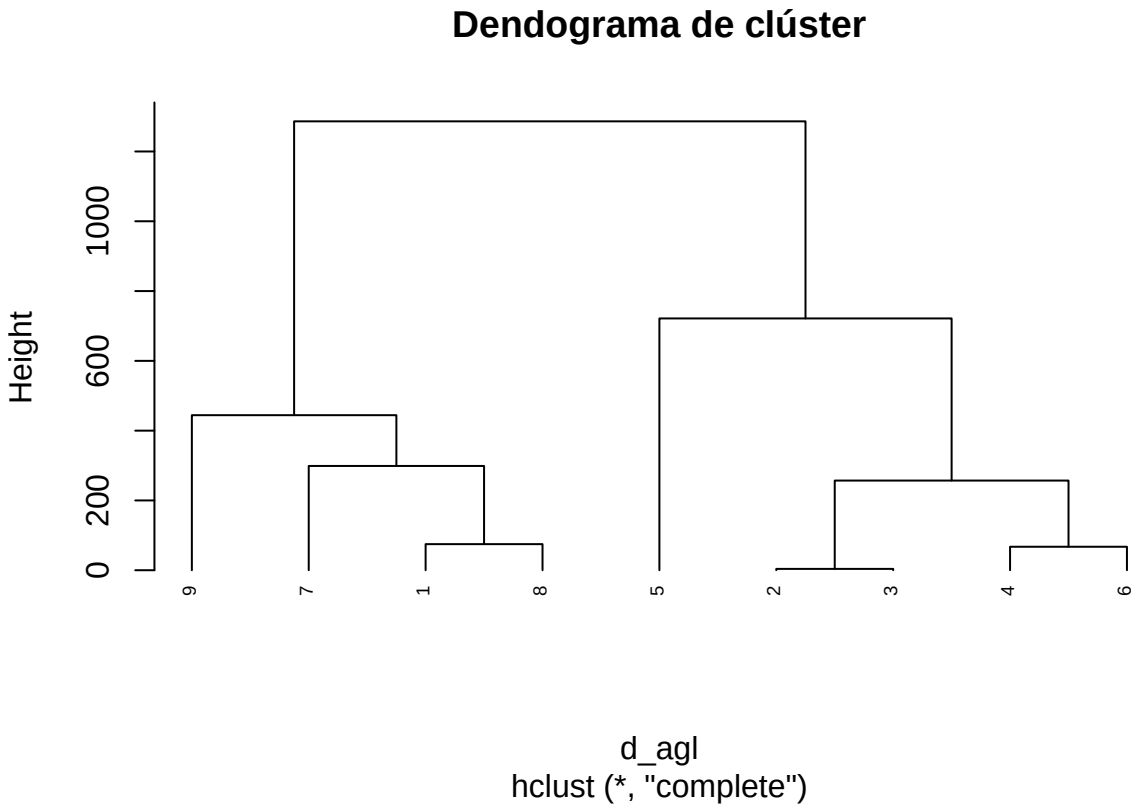
Sale un p-valor inferior a 0.05, lo que hace rechazar H0 y que se cumpla el supuesto de homocedasticidad.

13.Test de Clustering sobre la simulación de Montecarlo

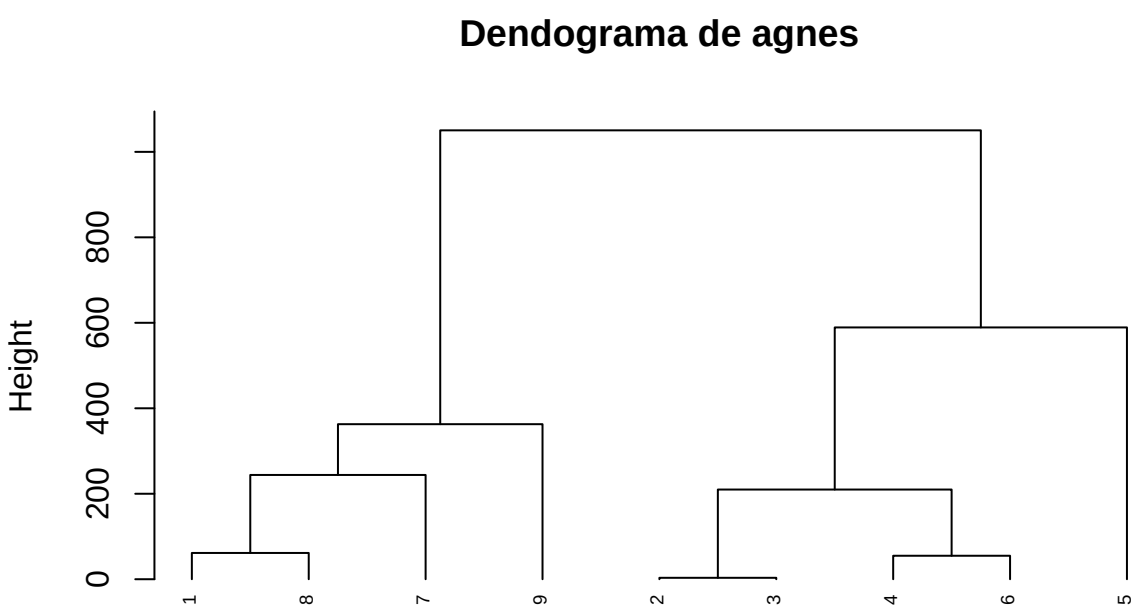
```r
#Test de clusters**************************************************************************************************
par(mfrow=c(1,1))
library("cluster")
library("factoextra")
d_agl<-dist(ByGrupos,method="euclidean") ##especificamos los valores de distancia
hc_agl<-hclust(d_agl,method="complete") ##calculamos el clúster jerárquico
hc_agl
```

```
##
## Call:
## hclust(d = d_agl, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 9
```

```r
plot(hc_agl,cex=0.6,hang=-1, main ="Dendograma de clúster")
```




```
##representamos el dendograma

#Función agnes y coeficiente de aglomeración del dendograma (mejor cuanto más
#se aproxima a 1)   que indica el grado de fortaleza del agrupamiento.
##Agrupamiento divisional jerárquico.
hc_ag<-agnes(ByGrupos,method="complete")
##calculamos los clústeres jerárquicos
pltree(hc_ag,cex=0.6,hang=-1,main="Dendograma de agnes") ##representamos dendograma
```




```
#*************** COMENTAR EL DENDOGRAMA. FÍJATE EN CASOS Y NO MUERTES, SALE BIEN.

#Coeficiente de aglomeración:
hc_ag$ac
```

```
## [1] 0.8481443
```

```
#El valor es: 0.8229438 lo que indica que la agrupación es fuerte.
```

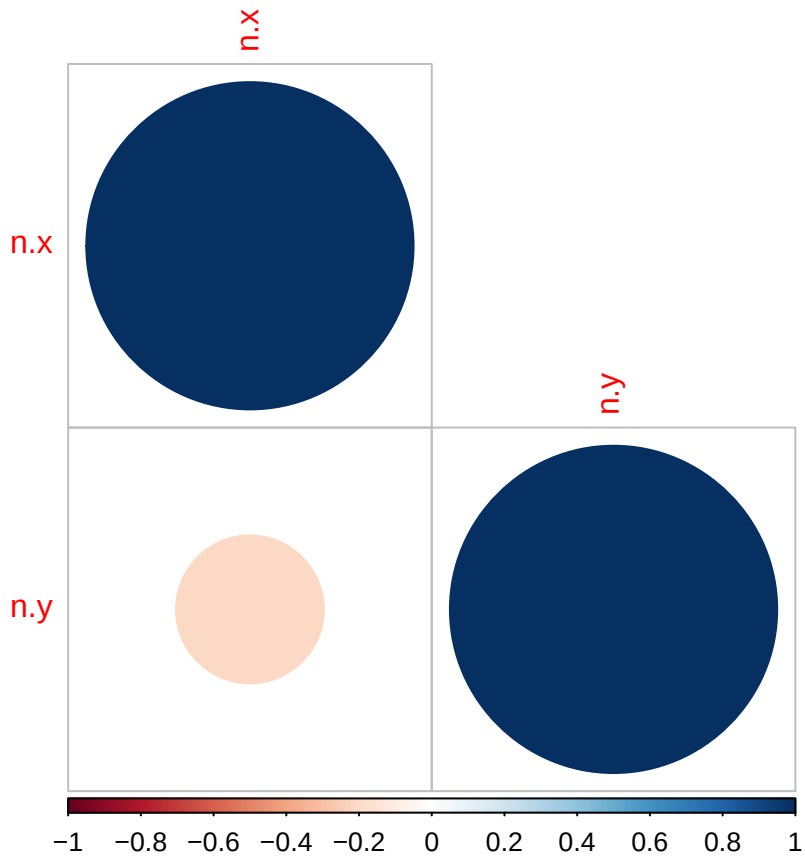
El valor es: 0.8229438 lo que indica que la agrupación es aceptable. En los dendogramas podemos ver los "linajes" de cada grupo o sus relaciones.

Hacemos un estudio de regresión entre casos y muertes en la simulación:

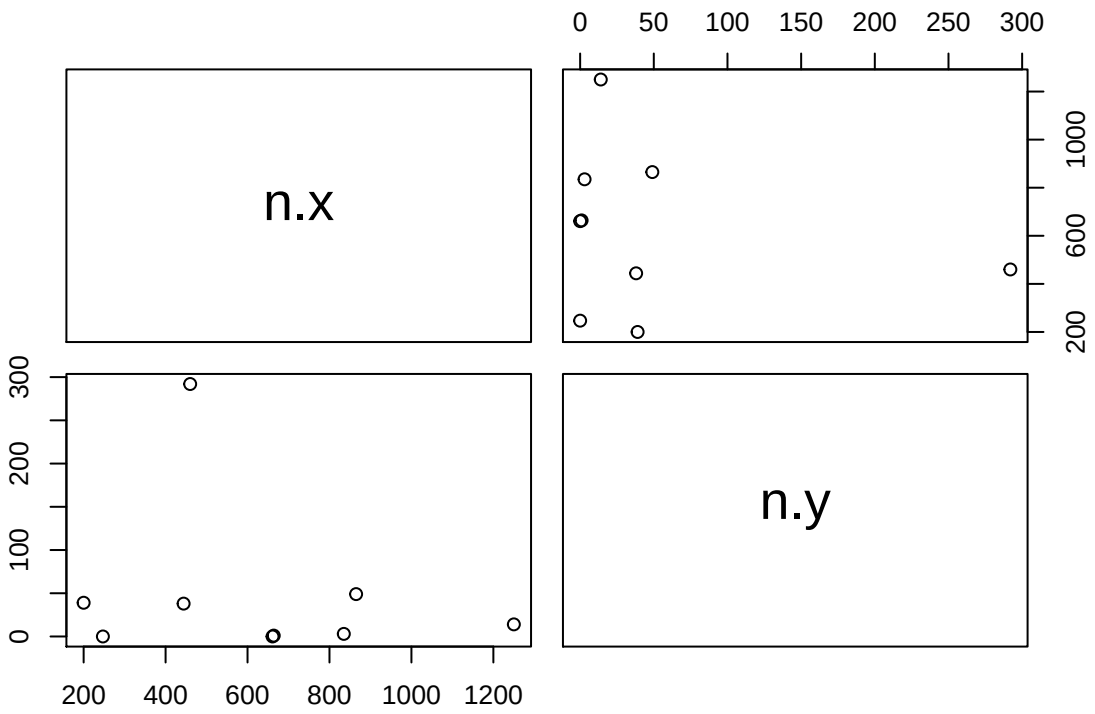
```
#Estudio de correlaciones entre casos y muertes:
require(corrplot)
round(cor(subset(ByGrupos[,2:3]), method = "pearson"), digits = 3)
```

```
##        n.x    n.y
## n.x  1.000 -0.204
## n.y -0.204  1.000
```

```
corrplot(round(cor(subset(ByGrupos[,2:3])), digits = 3), type = "lower")
```



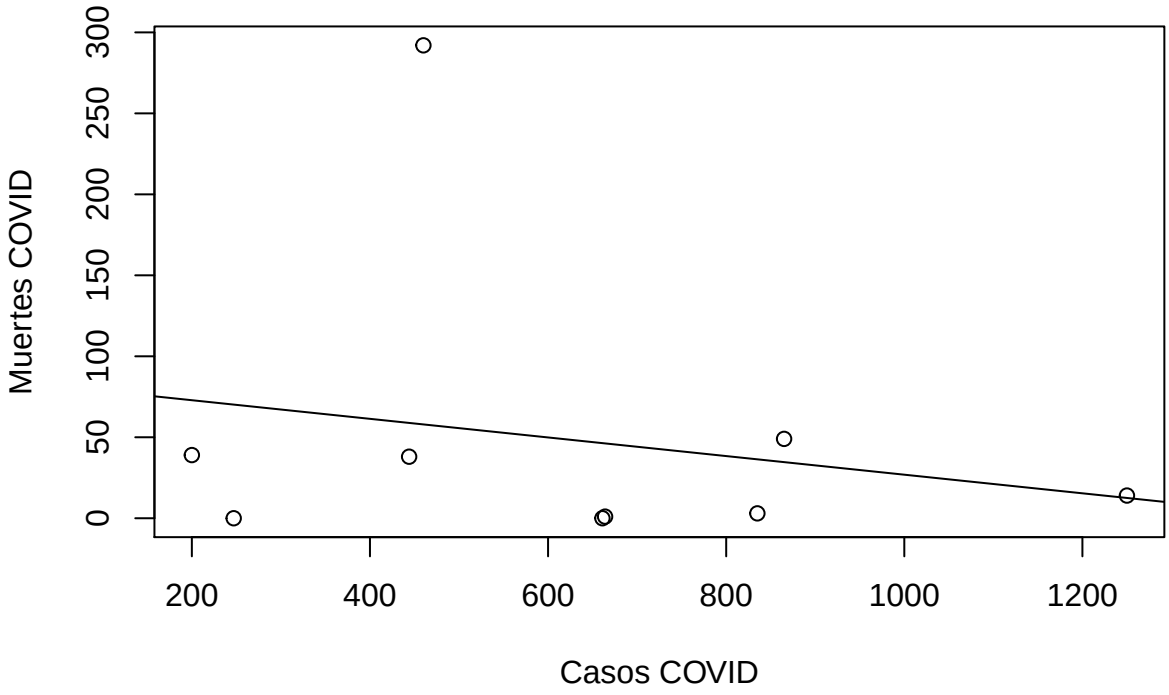

```
pairs(ByGrupos[,2:3])
```




```
#MODELO DE REGRESIÓN MÚLTIPLE.
#Modelo de regresión múltiple
regl<-lm(ByGrupos$n.y ~ByGrupos$n.x,data=ByGrupos)
summary(regl)
```

```
##
## Call:
## lm(formula = ByGrupos$n.y ~ ByGrupos$n.x, data = ByGrupos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.178 -45.209 -33.380   1.473 234.065
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  84.37469   72.83752   1.158    0.285
## ByGrupos$n.x -0.05748    0.10422  -0.551    0.598
##
## Residual standard error: 97.7 on 7 degrees of freedom
## Multiple R-squared:  0.04164,    Adjusted R-squared:  -0.09527
## F-statistic: 0.3041 on 1 and 7 DF,  p-value: 0.5985
```

```
plot(ByGrupos$n.x,ByGrupos$n.y, xlab="Casos COVID", ylab="Muertes COVID")
abline(regl)
```

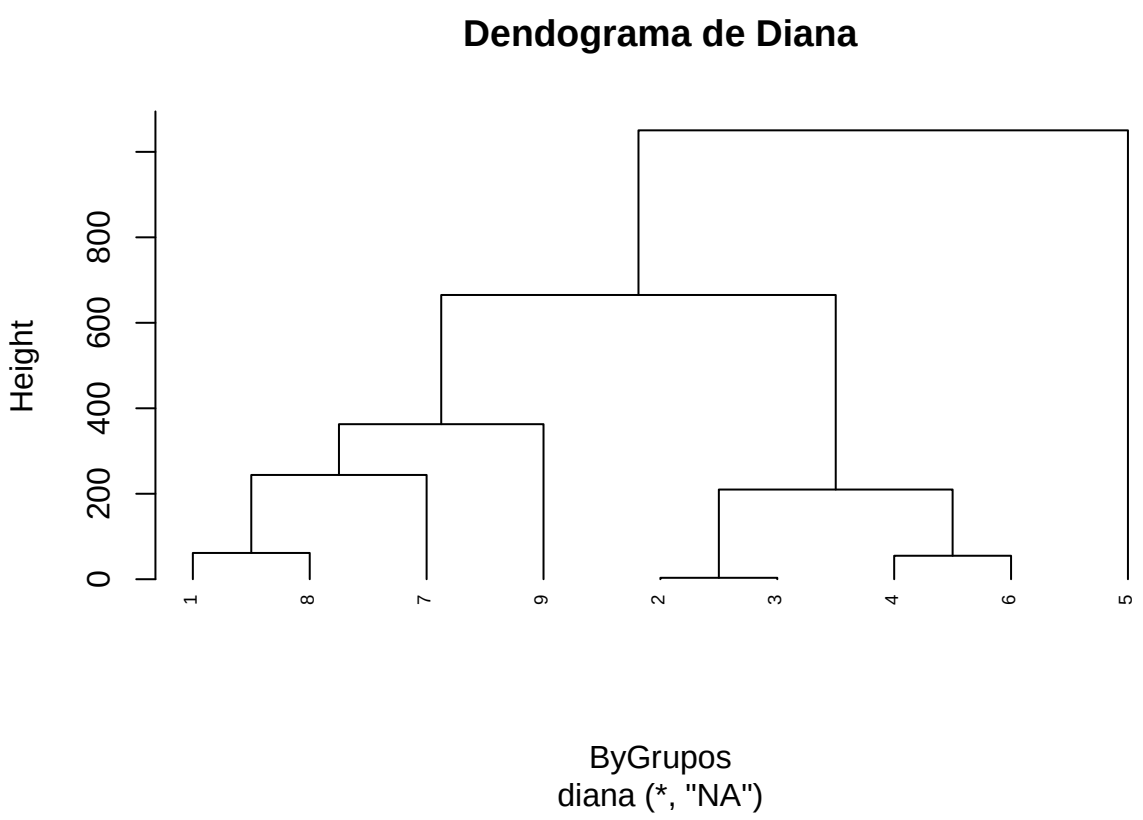


De los datos y gráficos vemos que existe una ligera relación lineal entre casos y muertes, pero no muy clara.

14.Estudio de clustering de la Simulación Montecarlo.

Pasamos a realizar un estudio de clustering sobre la simulación realizada utilizando una función diana:

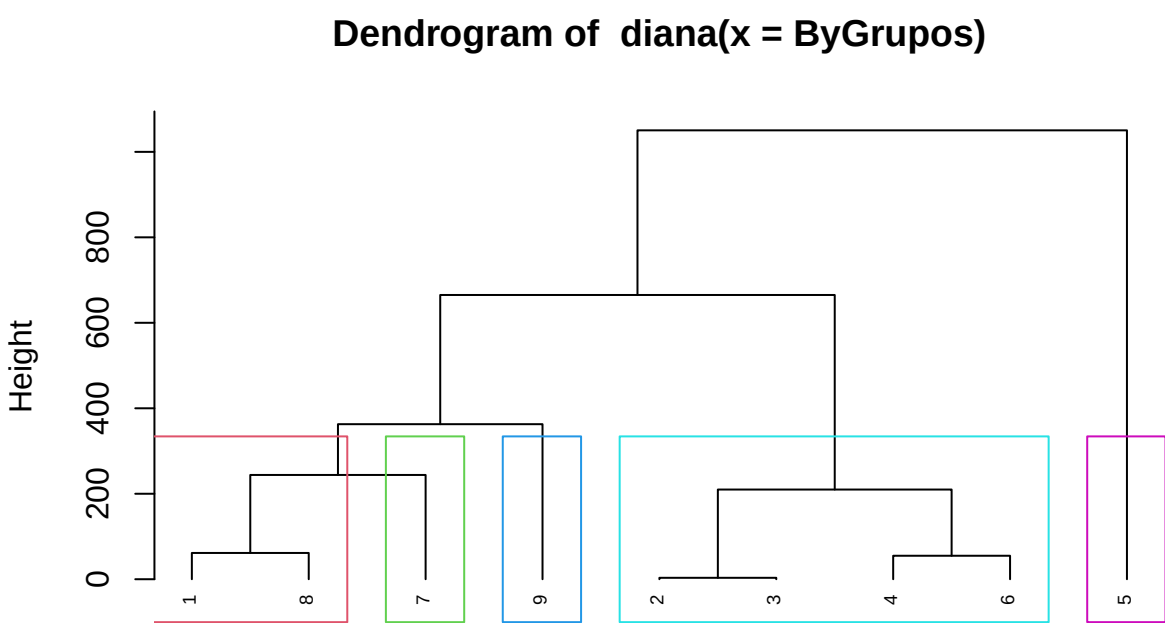
```
#Veamos ahora un ejemplo de agrupamiento divisional jerárquico utilizando la
#función diana.
##Agrupamiento divisional jerárquico
par(mfrow=c(1,1))

hc_div<-diana(ByGrupos) ##calculamos los clústeres jerárquicos
pltree(hc_div,cex=0.6,hang=-1,main="Dendograma de Diana") ##representamos el dendograma
```




```
hc_div$dc ##calculamos el coeficiente de división
```

```
## [1] 0.7993618
```

```
#El valor es: 0.8105296 lo que indica que la agrupación es algo fuerte.
#Una vez creados los agrupamientos, asociamos los clústeres a las observaciones
#de datos.

pltree(hc_div,hang=-1,cex=0.6)
rect.hclust(hc_div,k=5,border=2:10)
```



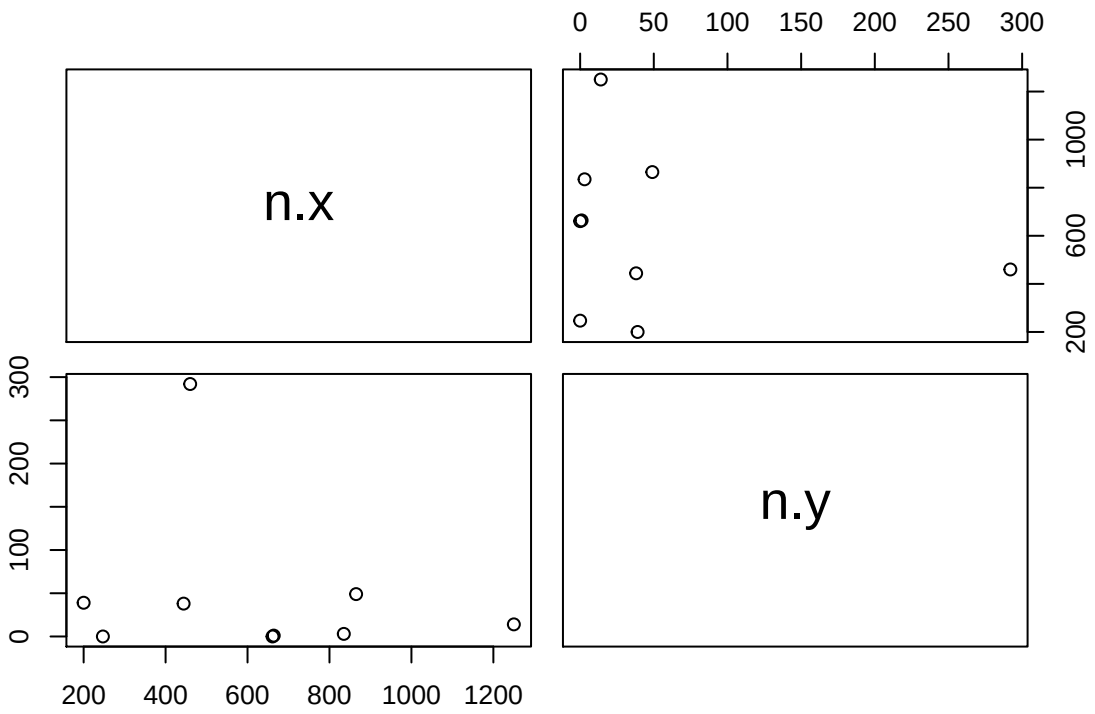
Tenemos un coeficiente de agrupamiento de. 0.8144578, que es bastante aceptable y fuerte.

Hacemos un test de Clusterin Jerárquico y consideraré 3 grupos (K=3) dado lo visto y analizado hasta ahora.

```
## Tests clustering jerárquicos.
#Veremos cómo los métodos no jerárquicos organizan los elementos en grupos según
#un número de clústeres definido de antemano, creando grupos homogéneos y
#teniendo en cuenta que ningún elemento puede pertenecer a más de un grupo.
#Estos además serán heterogéneos entre ellos. Una de las técnicas más utilizadas
#de este tipo de método es la k-means, cuya idea es agrupar las observaciones
#en k clústeres distintos determinados de antemano.

str(ByGrupos)
```

```
## 'data.frame':    9 obs. of  3 variables:
##  $ Grupo: chr  "0-9" "10-19" "20-29" "30-39" ...
##  $ n.x  : num  247 661 664 835 1250 865 444 200 460
##  $ n.y  : num  0 0 1 3 14 49 38 39 292
```

```
##consideramos y nuestra variable objetivo ##realizamos un gráfico de
#dispersión según la variable predictora
pairs(ByGrupos[,2:3])
```




```
corr=cor(ByGrupos[-1])
corr
```

```
##            n.x        n.y
## n.x  1.0000000 -0.2040549
## n.y -0.2040549  1.0000000
```

```
##Construimos los grupos de clústeres, crearemos 3 grupos para los grupo de edad
grupos_km=kmeans(ByGrupos[,2:3],3) ##10 variables
grupos_km
```

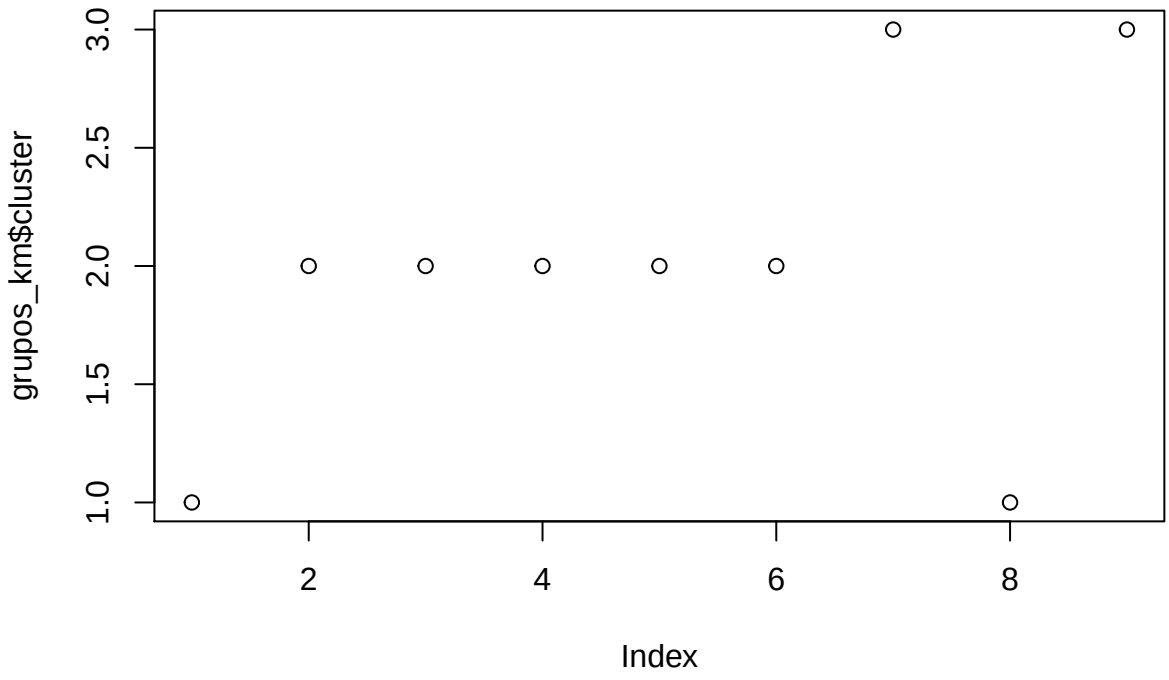
```
## K-means clustering with 3 clusters of sizes 2, 5, 2
##
## Cluster means:
##     n.x   n.y
## 1 223.5  19.5
## 2 855.0  13.4
## 3 452.0 165.0
##
## Clustering vector:
## [1] 1 2 2 2 2 2 3 1 3
##
## Within cluster sum of squares by cluster:
## [1]   1865.0 232351.2  32386.0
##  (between_SS / total_SS =  71.9 %)
##
## Available components:
```

```
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
grupos_km$cluster ##grupo al cual pertenecen los datos
```

```
## [1] 1 2 2 2 2 2 3 1 3
```

```
plot(grupos_km$cluster)
```



15.Conclusiones

Hemos visto que podemos simular por Montecarlo la pandemia y, salvo matices y datos que faltan la simulación reporduce bastante bien la realidad. De otro lado, hemos visto que los grupos de edad afectan en la incidencia de casos y muertes de COVID. ha sido una buena experiencia para trabajar métodos de montecarlo y de análisis estadísticos vistos en la asignatura.

NOTA: ¡No dispongo de más tiempo! Pero he aprendido mucho en el manejo de R y sus posibilidades para el análisis de datos que, debo completar con más conocimientos de estadística y datamining.