GRADO EN ESTADÍSTICA (UNIVERSIDAD DE GRANADA)
ESTADÍSTICA COMPUTACIONAL I
TAREA 2

Alumna: López Moreno María
Profesores: Román Montoya Yolanda
González Carmona Andrés

Índice

Nota previa: el directorio de trabajo, como en la tarea anterior, lo tengo en una carpeta (*<<Nueva carpeta>>*) que uso sólo para este ejercicio:

```
> getwd()
[1] "C:/Users/Usuario/Desktop/Matemáticas/Cuarto/Primer cuatrimestre/Estadística computacional 1/programas r/Nueva carpeta"
```

EJERCICIO 1

En este ejercicio nos indican que tenemos que trabajar con los datos *<<swiss>>* que están en el libro básico de R. Así que el primer paso en la resolución del ejercicio será cargar los datos y ver qué variables tiene y qué indican:

```
> data(swiss)
> head(swiss)
             Fertility Agriculture Examination Education Catholic
Courtelary        80.2        17.0          15        12     9.96
Delemont          83.1        45.1           6         9    84.84
Franches-Mnt      92.5        39.7           5         5    93.40
Moutier           85.8        36.5          12         7    33.77
Neuveville        76.9        43.5          17        15     5.16
Porrentruy        76.1        35.3           9         7    90.57
             Infant.Mortality
Courtelary               22.2
Delemont                 22.2
Franches-Mnt             20.2
Moutier                  20.3
Neuveville               20.6
Porrentruy               26.6
```

Para entenderlo mejor, recurrimos a:

```
?swiss
```

R: Swiss Fertility and Socioeconomic Indicators (1888) Data

Y vemos que se trata de un data frame que recoge datos de la fertilidad y de algunos indicadores socioeconómicos de 47 provincias de Suiza en el año 1888.

Además, como se nos pide trabajar con ciertas variables de *swiss*, lo que haremos será usar la función *attach()* para facilitar el manejo de las variables:

```
attach(swiss)
```

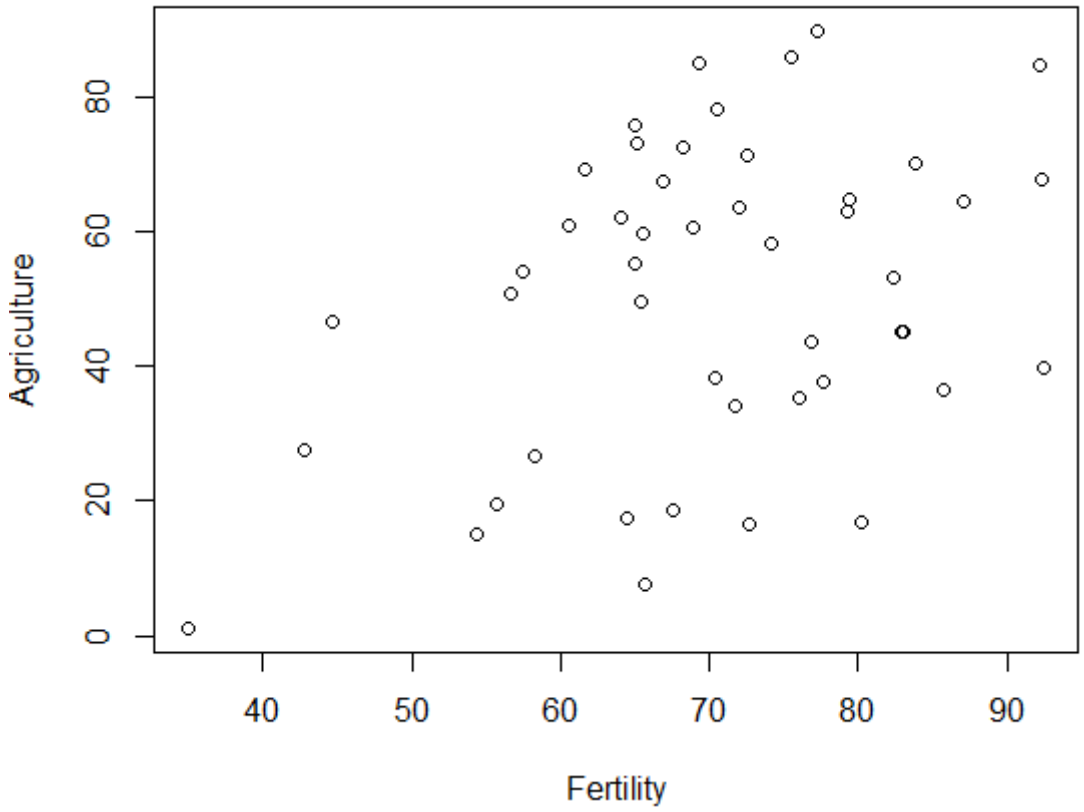
I.Gráficos de dispersión

Apartado (a)

Se nos pide hacer un gráfico de dispersión de las variables *Fertility* frente *Agriculture*. Para ello usaré la función *plot()*:

```
plot(Fertility, Agriculture)
```

Este comando con ese argumento devuelve una dispersión sencilla:

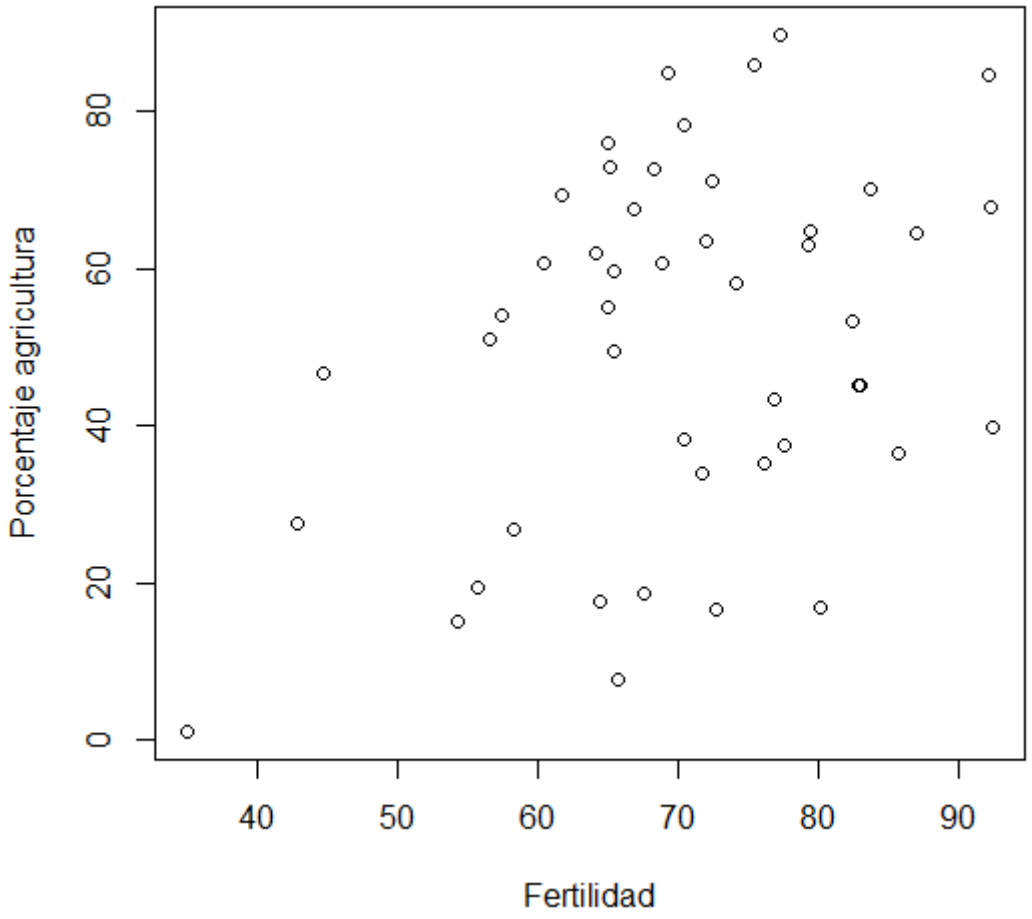


Pero esta presentación se puede mejorar, por ejemplo, poniendo los nombres en español o ajustando los límites de los ejes. Estas son algunas mejoras que se proponen en los apartados siguientes.

Apartado (b)

Se pide que cambiemos las etiquetas de los ejes por unas más apropiadas. Entiendo que lo que se pretende es tener unas etiquetas más explicativas en español. Para ello cambié los nombres de los ejes. Se podrían añadir un título y un subtítulo con la finalidad de que el gráfico sea auto explicativo, pero se nos requerirá en apartados posteriores, por lo tanto, de momento no lo añado.

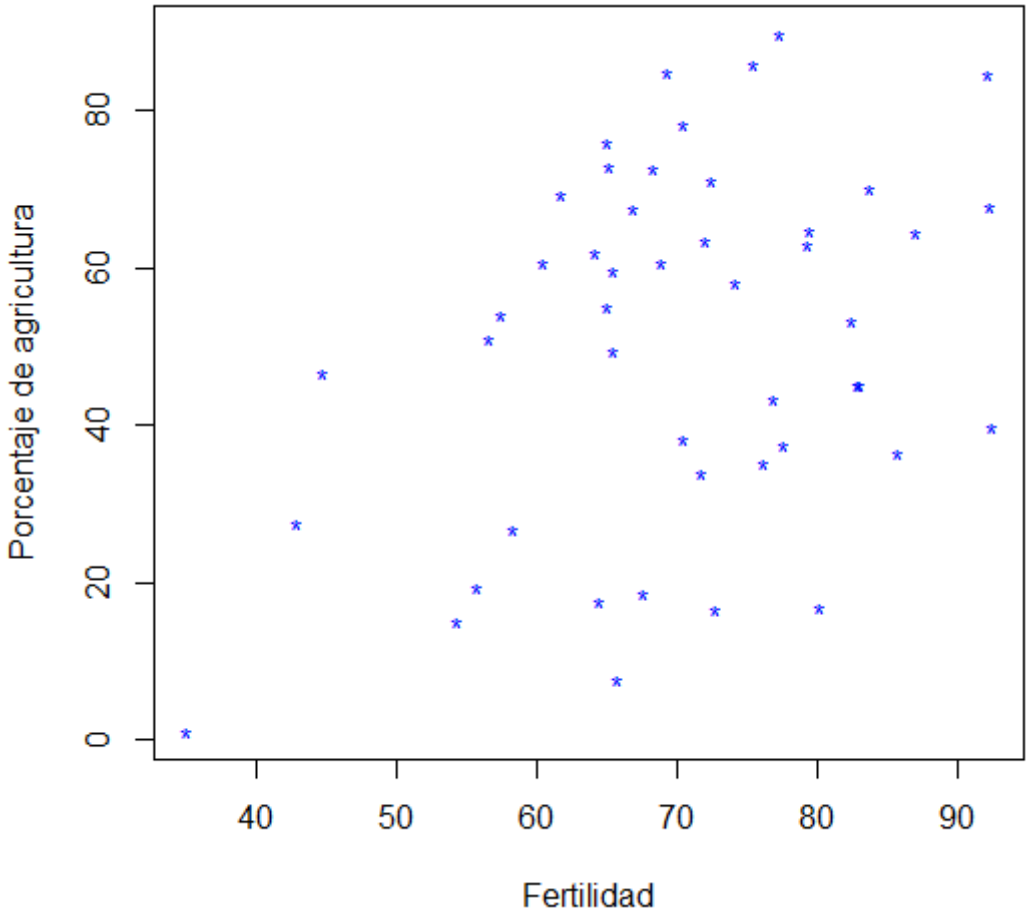
```
plot(Fertility, Agriculture, xlab="Fertilidad",
ylab="Porcentaje agricultura")
```



Apartado (c)

Se pide que los puntos aparezcan como "*" y en azul. Para ello, se busca qué número en ASCII corresponde al símbolo "*" (42) y se iguala el parámetro *pch* a 42. Para que sean azules se iguala el parámetro *col* a blue (azul en inglés).

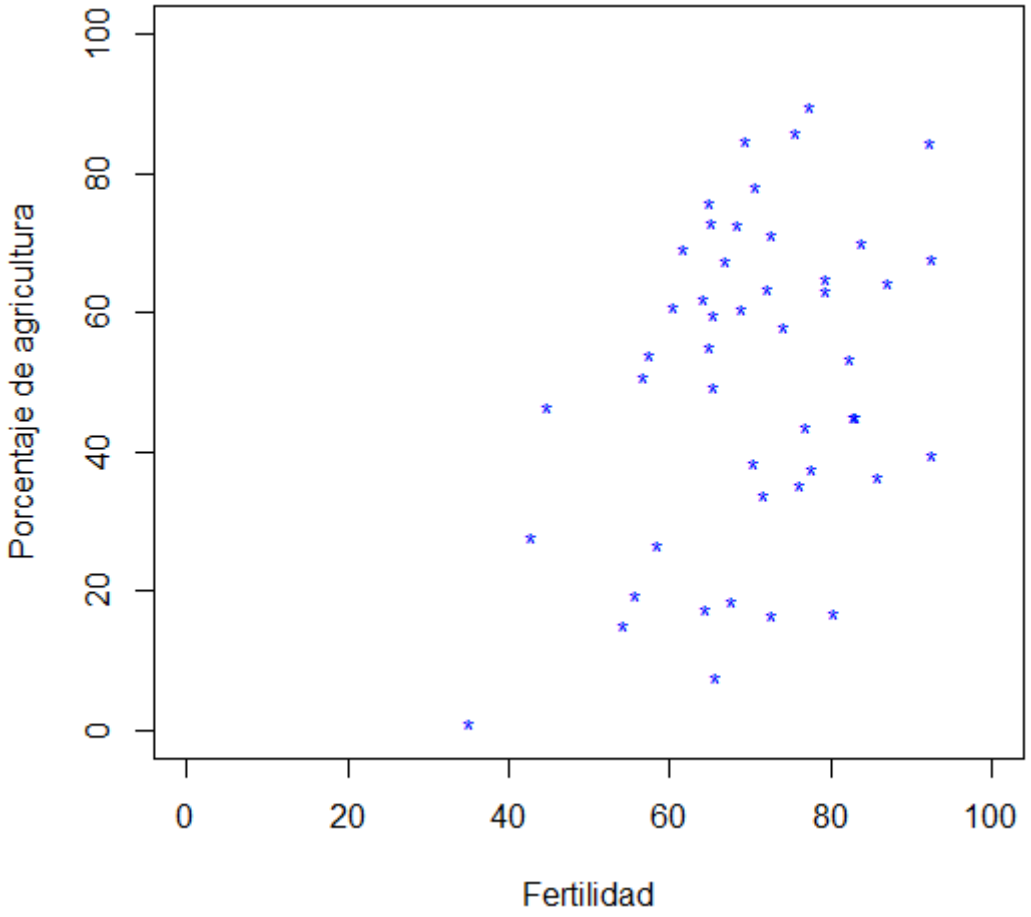
```
plot(Fertility, Agriculture, xlab="Fertilidad",
ylab="Porcentaje de agricultura", pch=42, col="blue")
```



Apartado (d)

Se nos pide que los ejes estén limitados por 0 y 100. Utilizamos los parámetros *xlim* e *ylim.*

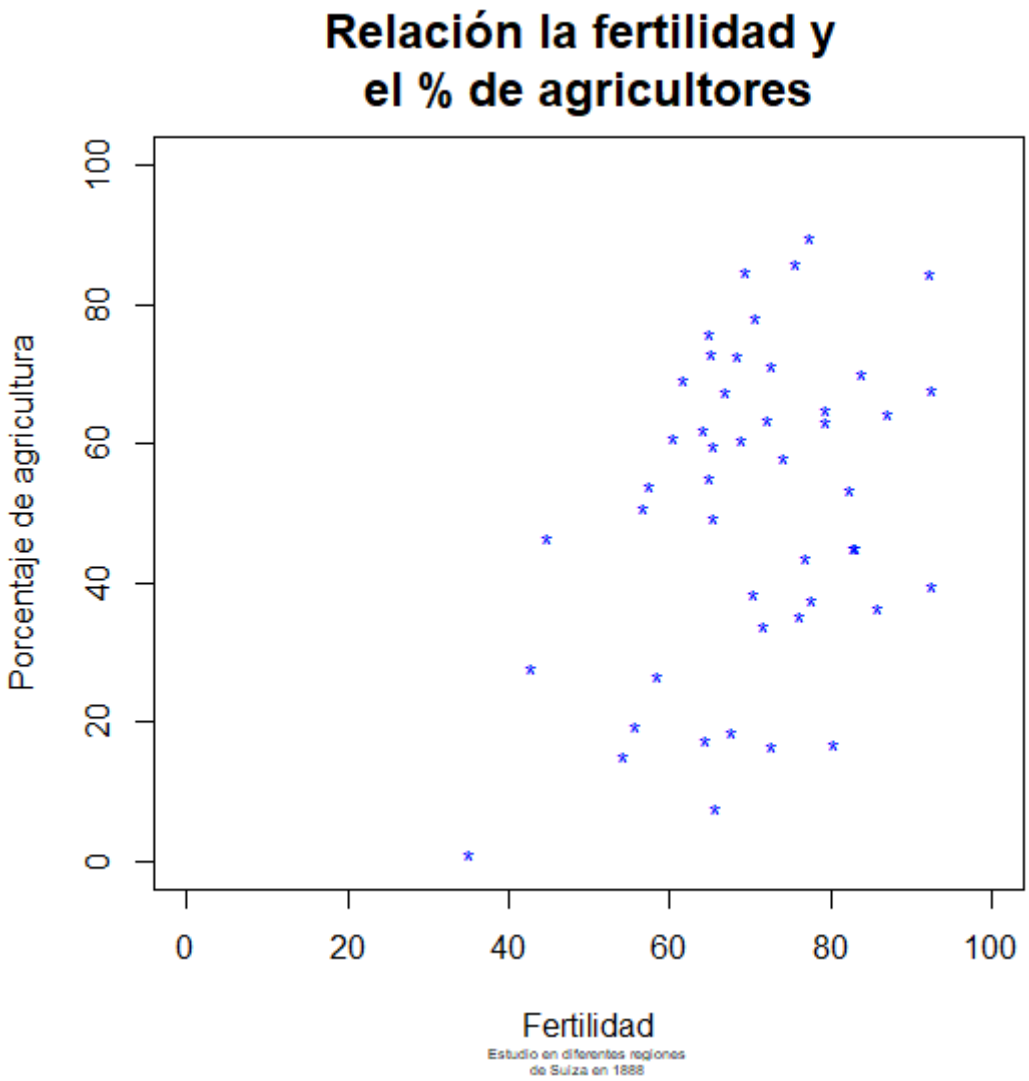
```
plot(Fertility, Agriculture, xlab="Fertilidad",
ylab="Porcentaje de agricultura", pch=42, col="blue", xlim=c(0,100),
ylim=c(0,100))
```



Apartado (e)

Aquí, se solicita dotar de título y subtítulo al gráfico. Lo hago con los parámetros *main* y *sub*.

```
plot(Fertility, Agriculture, xlab="Fertilidad",
ylab="Porcentaje de agricultura", main="Relación la fertilidad y
el % de agricultores", cex.main=1.5, sub="Estudio en diferentes regiones
de Suiza en 1888",cex.sub=0.4, pch=42, col="blue", xlim=c(0,100),
ylim=c(0,100))
```



Conclusiones

Por el gráfico no podemos intuir ninguna relación entre ambas variables. La linealidad parece fallar. Reafirmémoslo:

Para ello, podemos usar la función *lm()* que nos dará la ecuación de la recta de regresión.

```
> lm(Agriculture~Fertility)

Call:
lm(formula = Agriculture ~ Fertility)

Coefficients:
(Intercept)    Fertility
     5.6326        0.6419
```

Agriculture = 5.6326 + Fertility*0.6419

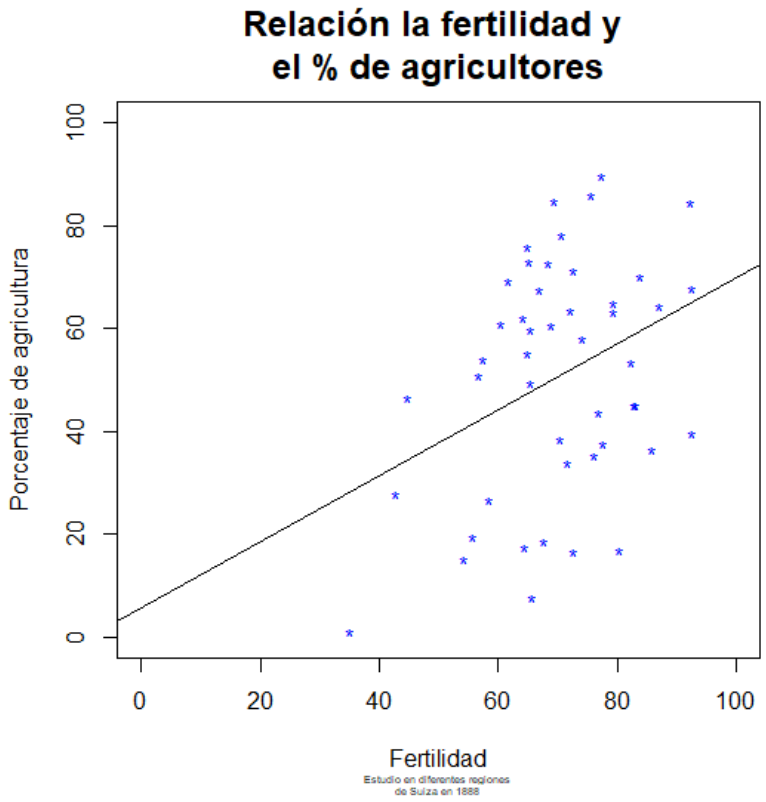
Pero, queremos ver cómo de acertado es. Usamos la función *summary()*

```
> summary(lm(Agriculture~Fertility))

Call:
lm(formula = Agriculture ~ Fertility)

Residuals:
    Min      1Q  Median      3Q     Max
-40.116 -17.753   4.836  15.775  34.781

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.6326    18.0601   0.312   0.7566
Fertility     0.6419     0.2536   2.532   0.0149 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.48 on 45 degrees of freedom
Multiple R-squared:  0.1247,    Adjusted R-squared:  0.1052
F-statistic: 6.409 on 1 and 45 DF,  p-value: 0.01492
```

Se nos da información sobre dicha recta de regresión. Nos fijamos en que el R-cuadrado en nuestro caso es 0.1247, por lo que, confirmamos que la recta de regresión no es un modelo explicativo de la posible relación de estas variables. El modelo sólo explica el 12,47% de una variable respecto a la otra: es una mala modelización.

Vemos, en la siguiente figura, el gráfico de dispersión y la recta de regresión, para ver más explícitamente la casi nula relación lineal.

```
> r_plot <- lm(Agriculture~Fertility)

> abline(r plot)
```



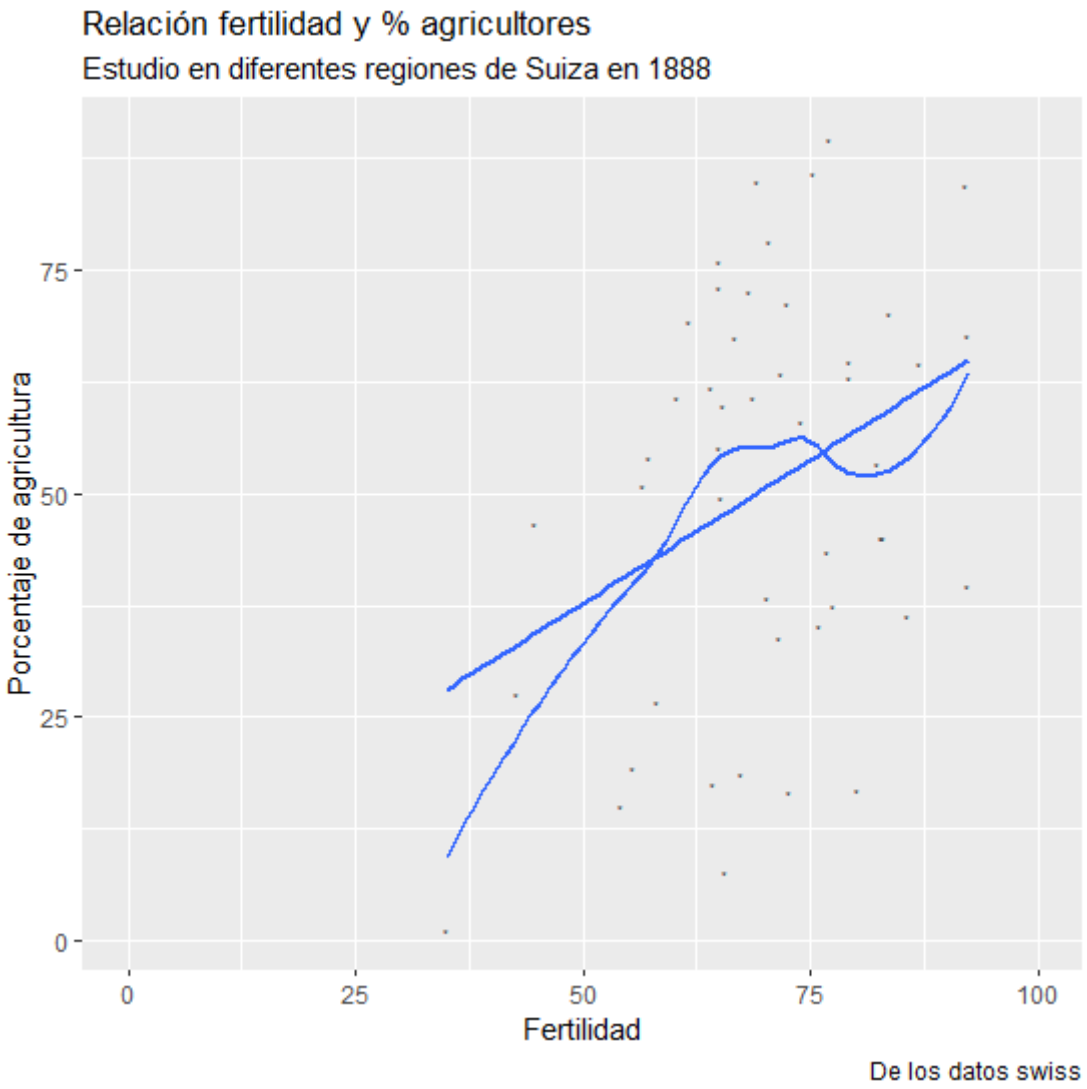
Otro gráfico

En R existen otras funciones para graficar, por ejemplo, la función *ggplot()* que permite multitud de opciones. Para ello, hacemos uso de dos librerías:

```
> library(tidyverse)
-- Attaching packages -----------------
v ggplot2 3.3.2     v purrr   0.3.4
v tibble  3.0.3     v dplyr   1.0.1
v tidyr   1.1.1     v stringr 1.4.0
v readr   1.3.1     v forcats 0.5.0
-- Conflicts --------------------------
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
> library(ggplot2)

> ggplot(swiss, aes(Fertility, Agriculture)) +
+ geom_point(shape = 42) +
+ geom_smooth(se = FALSE) +
+ geom_smooth(method = 'lm', se=FALSE) +
+ labs(title = "Relación fertilidad y % agricultores",
+      subtitle = "Estudio en diferentes regiones de Suiza en 1888",
+      caption = "De los datos swiss",
+      x = "Fertilidad",
+      y = "Porcentaje de agricultura") +
+ scale_x_continuous(limits = c(0,100))
```

Con *ggplot()* especificamos qué datos usamos y qué ejes usaremos. Con *geom_point()* le indicamos que cada observación la indique con un punto con la forma 42. Nos ayudamos con

labs() para poner el nombre a los ejes, el título, el subtítulo y una pequeña aclaración que dicta "De los datos swiss". Con *scale_x_continuous()* conseguimos que el eje x tenga límites en 0 y 100.



Aquí, además, hemos indicado que nos muestre el modelaje predeterminado que hace R con *geom_smooth()*. El método predeterminado en R para esta función depende del tamaño de los datos: cuando hay menos de 1000 observaciones, usa el método *'loess' (local regression*), cuando es mayor usa el método *'lm'*. Como observamos sale una especie de polinomio muy alejado de la recta de regresión. Al ser un método local, nos arriesgamos a que, de manera global, no sea un modelo deseable. Aquí, al haber casi 50 observaciones, lo más probable es que, globalmente, no sea un modelo óptimo ya que las últimas observaciones están alejadas de las primeras y, por tanto, el modelo que se ajusta a las últimas podría no ajustarse a las primeras observaciones. Sin embargo, quería hacer hincapié en los métodos que sigue R con *geom_smooth()* ya que es una función muy interesante.

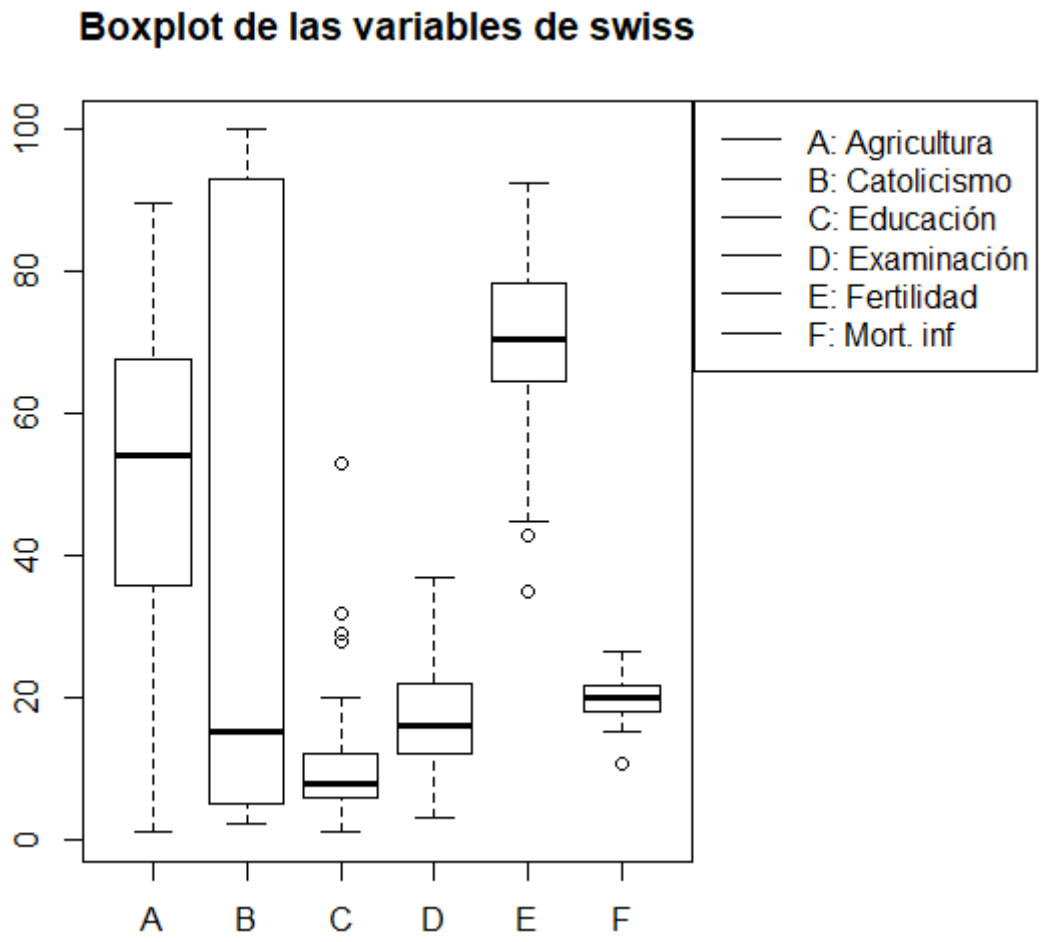
II.Boxplot

Apartado (a)

En este apartado se pide hacer un *boxplot* que reúna todas las variables del *data frame swiss*. Para que el gráfico fuese autoexplicativo, añadí una leyenda. Como las posiciones disponibles tapaban de uno u otro modo parte del gráfico, decidí poner la leyenda fuera. Para ello recurro a la función *par()* y uso los argumentos *mar* y *xpd*. *mar* indica en qué posición de la ventana gráfica queremos que aparezca el gráfico. En este caso, *mar= c(5,4,4,9)* que dista de la posición predeterminada (que es (5,4,4,2)). Además le indicamos *xpd=TRUE* para que la leyenda pueda habitar fuera del gráfico.

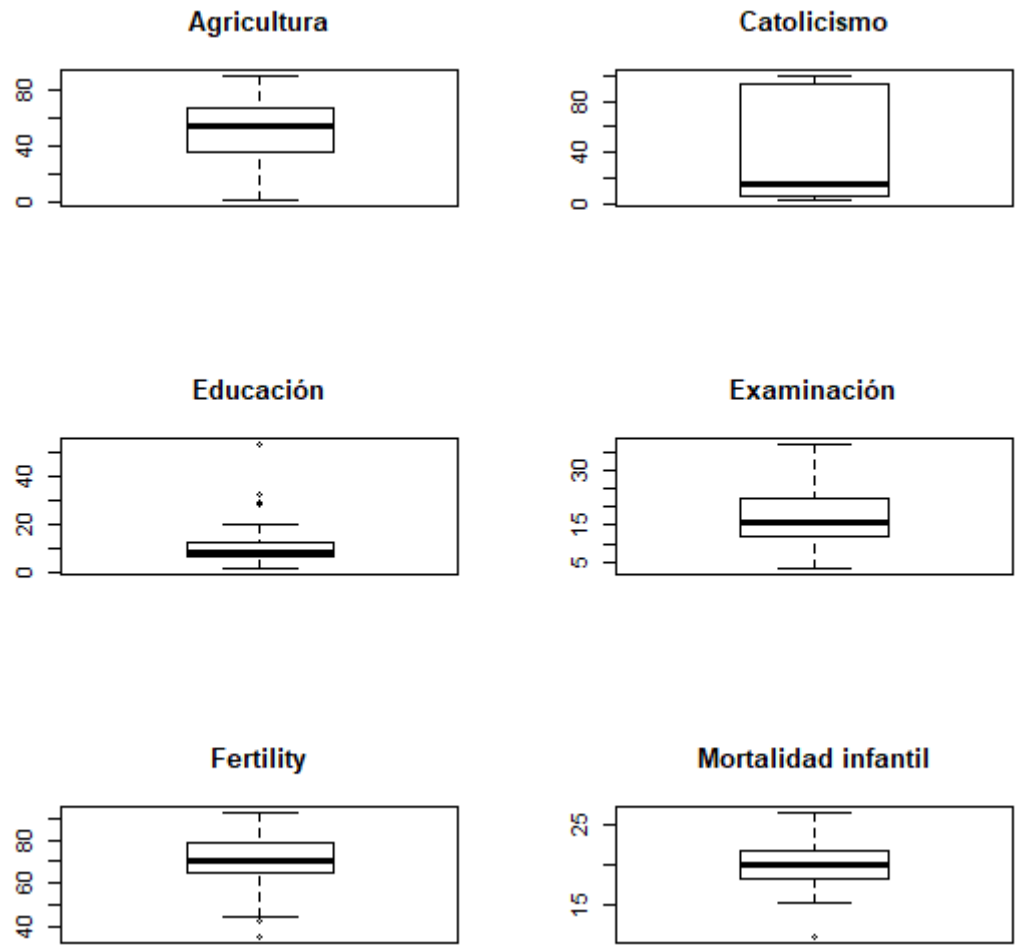
Asimismo, en la función *legend()* le especificamos *inset=c(-0.58,0)* para desplazar la posición *"topright"* hacia donde queremos.

```
par(mar=c(5, 4, 4, 9), xpd=TRUE) ## Cambiamos mar para hacer hueco a la leyenda
boxplot(Agriculture, Catholic, Education, Examination, Fertility, Infant.Mortality,
names=c("A","B","C","D","E","F"),main="Boxplot de las variables de swiss")

legend("topright", lwd=0.5, inset=c(-0.58,0),
legend=c("A: Agricultura", "B: Catolicismo", "C: Educación",
"D: Examinación", "E: Fertilidad", "F: Mort. inf"))
```



También podría haberse contemplado la posibilidad de hacer 6 boxplots diferentes y ponerlos en una misma ventana gráfica de la siguiente forma:

```
par(mfrow=c(3,2)) ##La ventana gráfica tendrá 3 filas y 2 columnas
boxplot(Agriculture, main="Agricultura")
boxplot(Catholic, main="Catolicismo")
boxplot(Education, main="Educación")
boxplot(Examination, main="Examinación")
boxplot(Fertility, main="Fertility")
boxplot(Infant.Mortality, main="Mortalidad infantil")
```



Pero en el enunciado se explicita la creación de **un** –solo- boxplot que reúna todas las variables. Por lo tanto, esta opción queda descartada, pero creo que merecía ser mencionada.

Nota: También podríamos haber puesto las etiquetas de las letras de la siguiente forma:

```
> letters[1:7]
[1] "a" "b" "c" "d" "e" "f" "g"
```

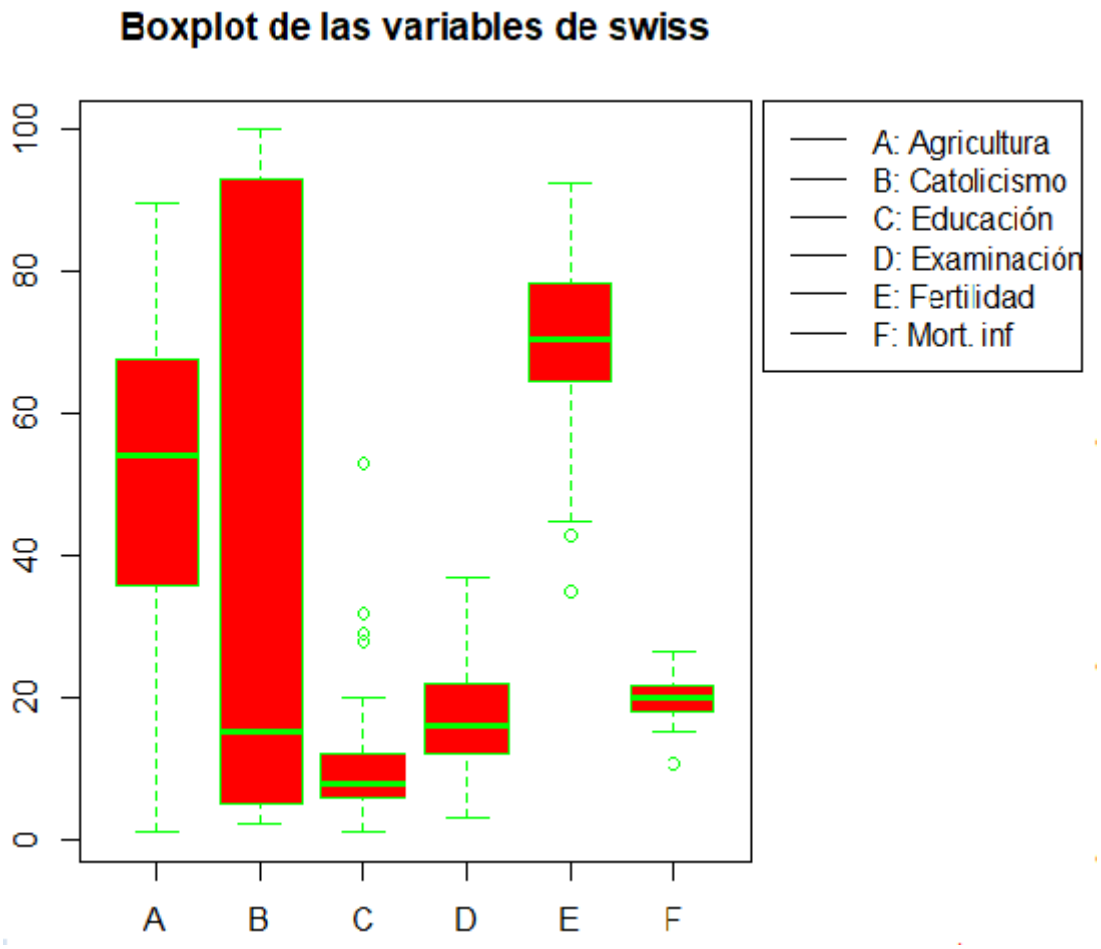
Apartado (b)

En este apartado se pretende cambiar el color de las cajas al rojo y los bordes de las mismas a verde. Para ello usamos dos argumentos de *boxplot(): col*, que se utiliza para dotar de color al interior de las cajas y *border*, que se usa para cambiar el color de los bordes de las cajas.

```
boxplot(Agriculture, Catholic, Education, Examination, Fertility, Infant.Mortality,
names=c("A","B","C","D","E","F"), main="Boxplot de las variables de swiss",
 border="green", col="red")
```

Además, ponemos, de nuevo, la leyenda:

```
legend("topright", lwd=0.5, inset=c(-0.58,0),
legend=c("A: Agricultura", "B: Catolicismo", "C: Educación",
"D: Examinación", "E: Fertilidad", "F: Mort. inf"))
```



Apartado (c)

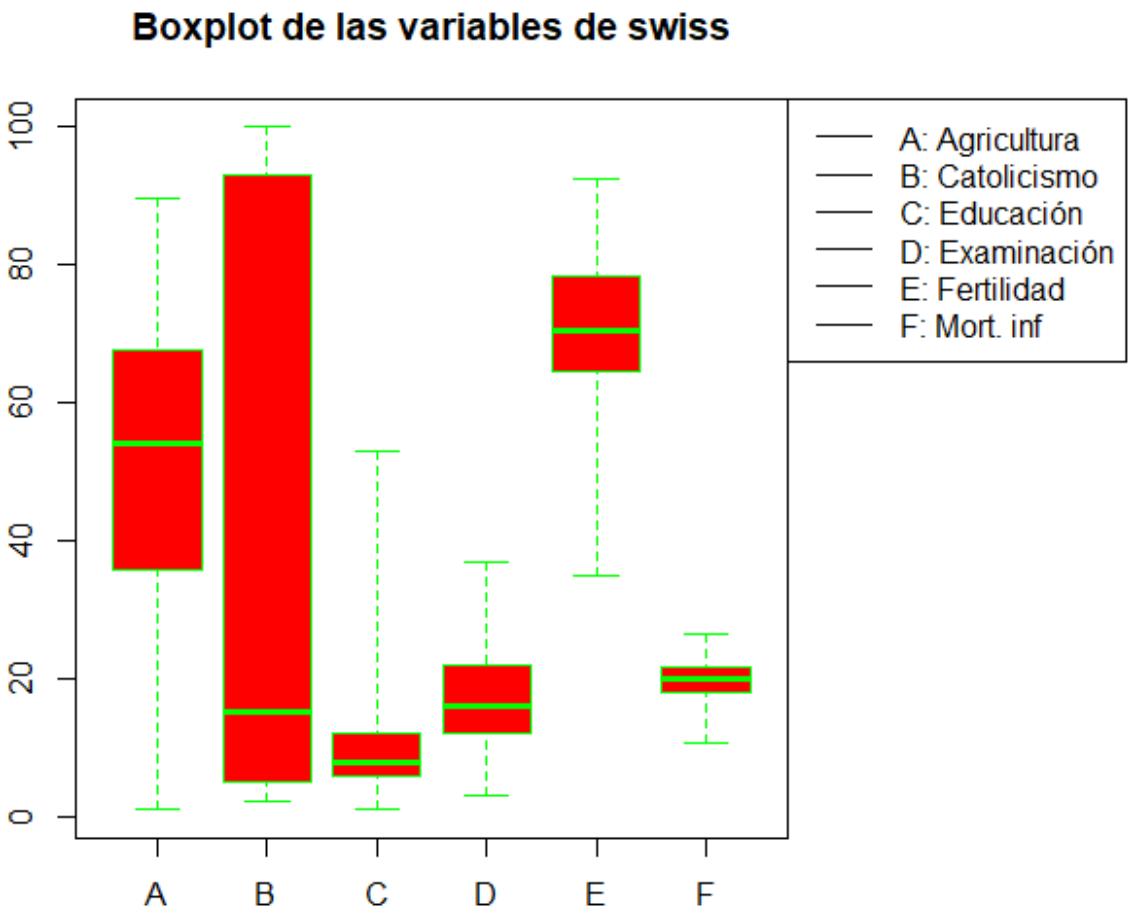
En este apartado se nos pide que hagamos las modificaciones necesarias para que los datos extremos estén también en el gráfico. Por lo tanto, lo que hacemos es igualar *range* a 0, con ello lo que R entiende es, precisamente, que los bigotes se han de extender hasta los valores extremos.

```
boxplot(Agriculture, Catholic, Education, Examination, Fertility, Infant.Mortality,
names=c("A","B","C","D","E","F"),main="Boxplot de las variables de swiss",
border="green", col="red",range=0)
```

De nuevo, añadimos la leyenda:

Nota: En este gráfico y los que siguen, la leyenda está menos desplazada ya que la ventana gráfica era de menor tamaño respecto a los gráficos anteriores.

```
legend("topright", lwd=0.5, inset=c(-0.48,0),
legend=c("A: Agricultura", "B: Catolicismo", "C: Educación",
"D: Examinación", "E: Fertilidad", "F: Mort. inf"))
```



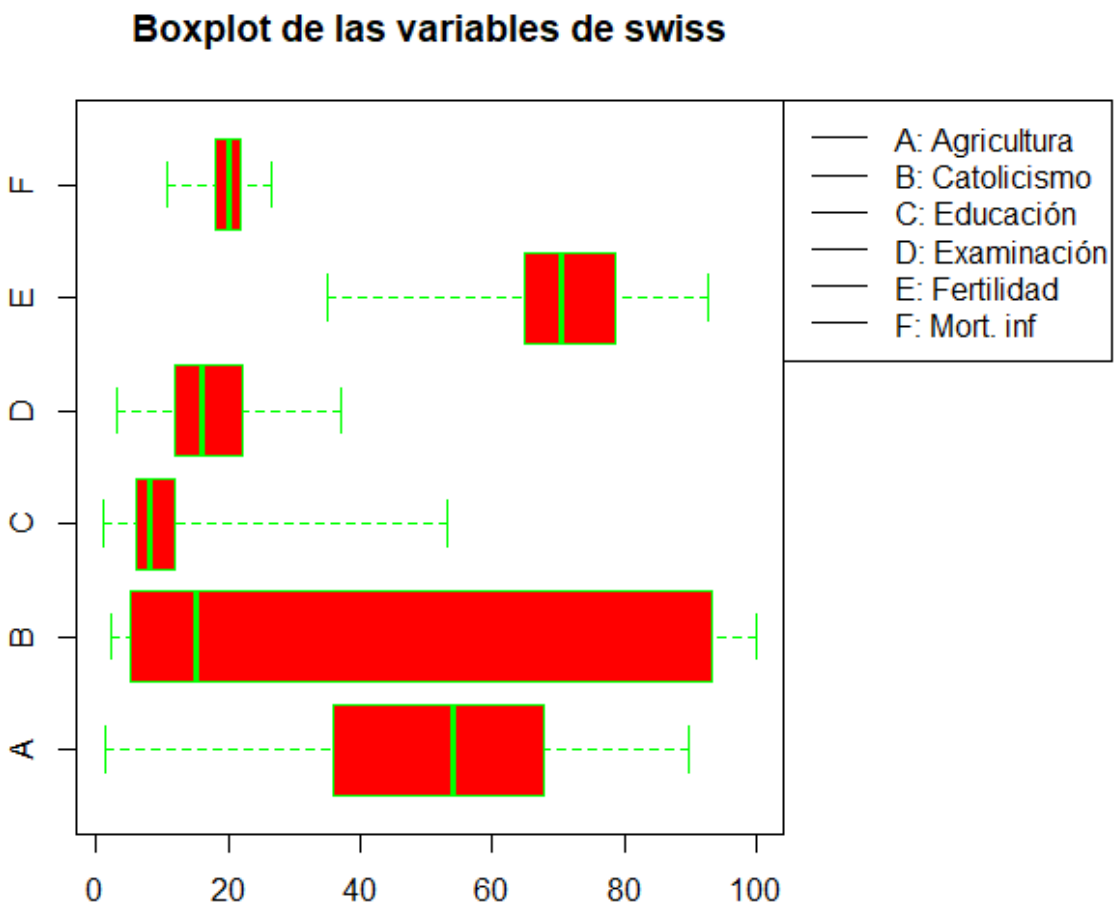
Apartado (d)

Se nos pide que el *boxplot*, a partir de ahora, sea horizontal. Para ello añadimos un argumento que reza *horizontal=TRUE*, cuya finalidad es precisamente lo que se requiere.

```
boxplot(Agriculture, Catholic, Education, Examination, Fertility, Infant.Mortality,
names=c("A","B","C","D","E","F"),main="Boxplot de las variables de swiss",
border="green", col="red",range=0, horizontal=TRUE)
```

Leyenda:

```
legend("topright", lwd=0.5, inset=c(-0.48,0),
legend=c("A: Agricultura", "B: Catolicismo", "C: Educación",
"D: Examinación", "E: Fertilidad", "F: Mort. inf"))
```



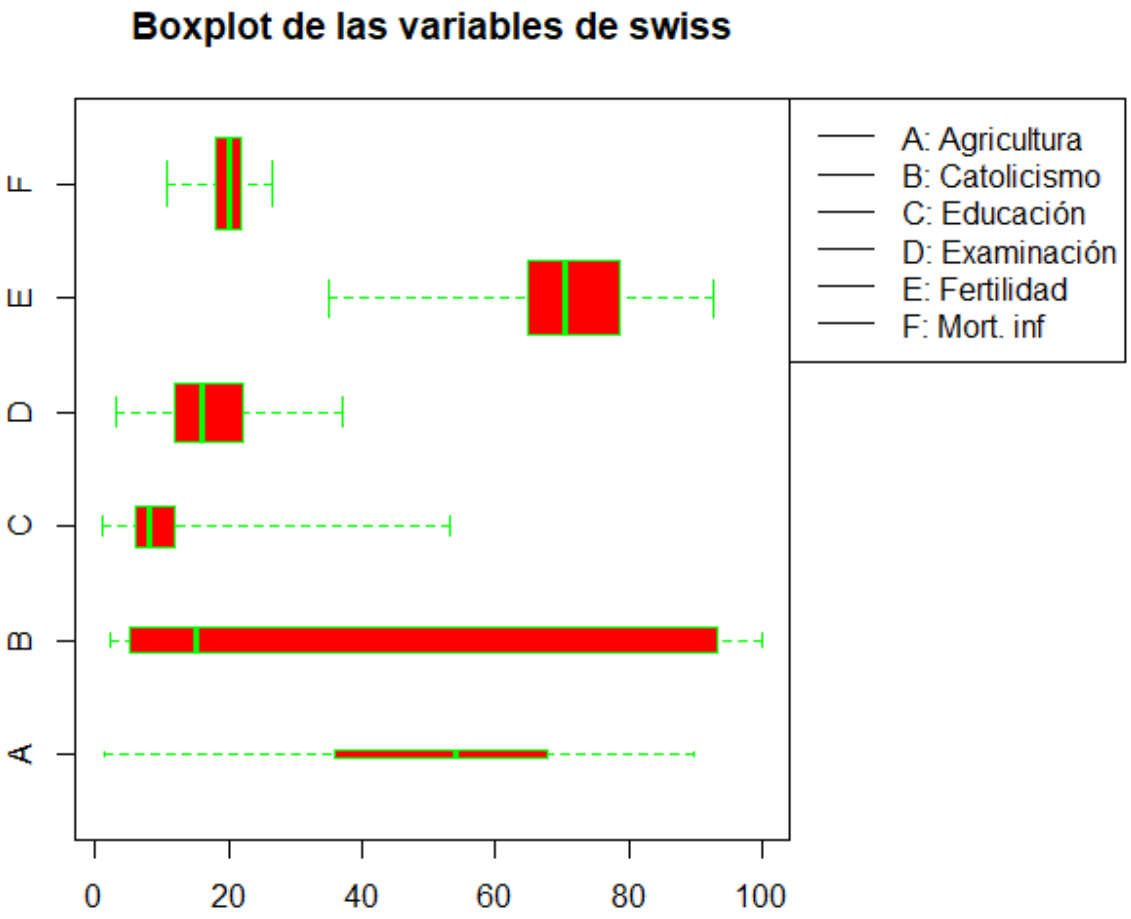
Apartado (e)

Aquí se demanda cambiar el ancho de las variables. Para ello igualo *width* a un vector numérico de 6 componentes, todas diferentes, para que haya 6 anchuras diferentes.

```
boxplot(Agriculture, Catholic, Education, Examination, Fertility, Infant.Mortality,
names=c("A","B","C","D","E","F"),main="Boxplot de las variables de swiss",
border="green", col="red",range=0, horizontal=TRUE, width = c(1,3,5,7,9,11))
```

Leyenda:

```
legend("topright", lwd=0.5, inset=c(-0.48,0),
legend=c("A: Agricultura", "B: Catolicismo", "C: Educación",
"D: Examinación", "E: Fertilidad", "F: Mort. inf"))
```



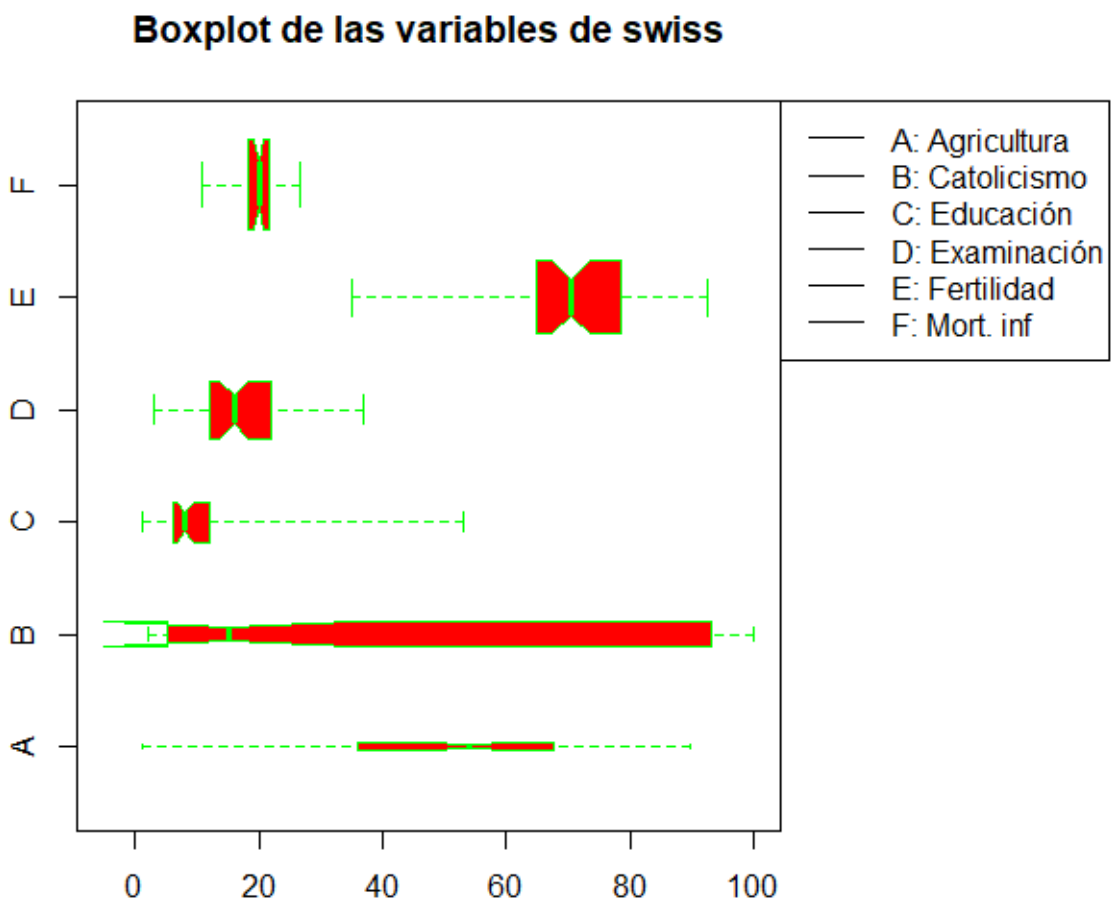
Apartado (f)

Se desea que el *boxplot* sea de tipo *notch*. Para ello se iguala *notch* a *TRUE* y lo que se realiza con esta orden es crear una "muesca" en cada diagrama de caja.

```
boxplot(Agriculture, Catholic, Education, Examination, Fertility, Infant.Mortality,
names=c("A","B","C","D","E","F"),main="Boxplot de las variables de swiss",
border="green", col="red",range=0, horizontal=TRUE, width = c(1,3,5,7,9,11),
notch=TRUE)
```

Leyenda:

```
legend("topright", lwd=0.5, inset=c(-0.48,0),
legend=c("A: Agricultura", "B: Catolicismo", "C: Educación",
"D: Examinación", "E: Fertilidad", "F: Mort. inf"))
```



Conclusiones

Este tipo de diagrama lo que realiza es una caja donde los lados mayores muestran el recorrido intercuartílico. Es muy sencillo de ver, a través de este tipo de gráficos, dónde se hallan los cuartiles y cómo se distribuye la población en base a ello. Se pueden ver si los valores

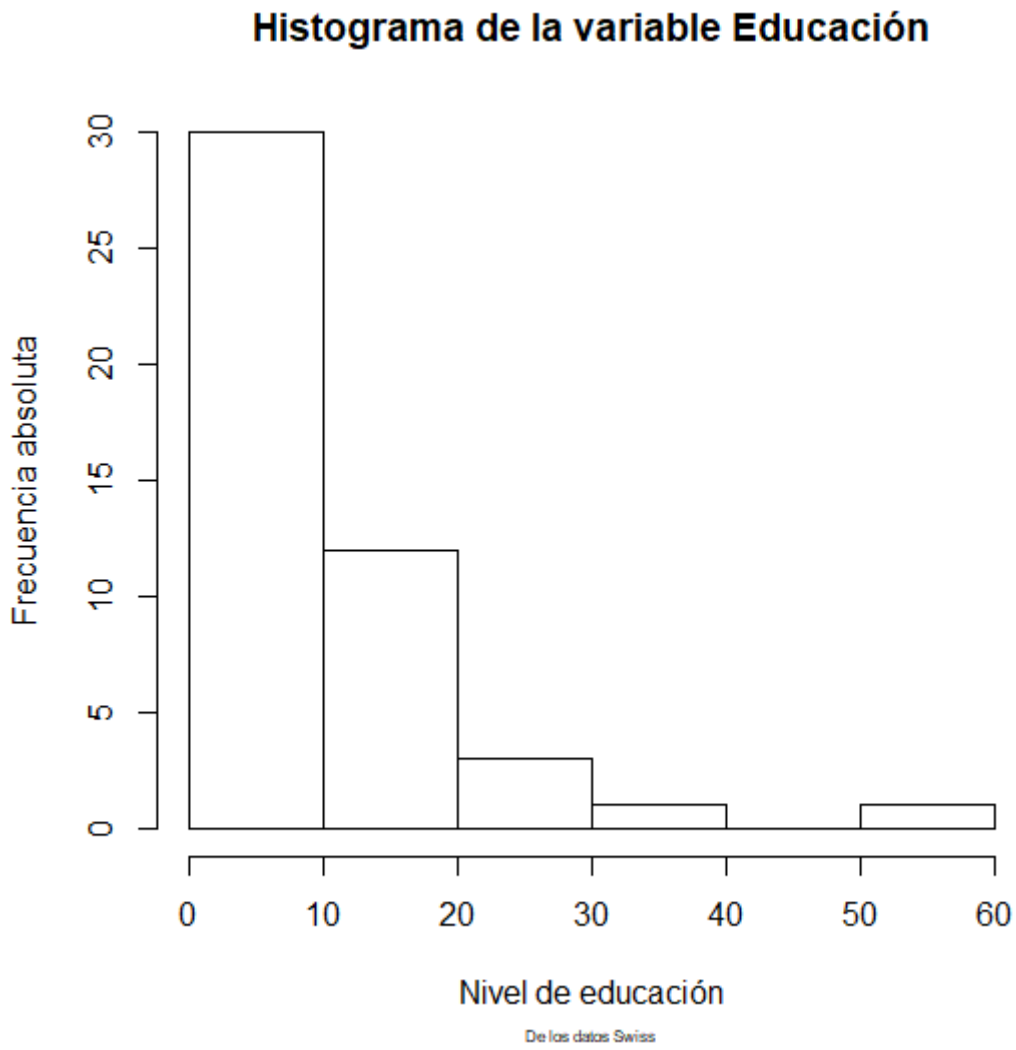
extremos están muy apartados (como es el caso de la variable “agricultura”). También se puede observar en qué momento se alcanza el 50% de los datos (la línea verde intermedia dentro de la caja).

III. Histograma

Apartado (a)

En este apartado se desea crear un histograma. Como no se pide en el resto de apartados, tomé la libertad de dotar de nombres explicativos a los ejes, un título y un subtítulo (que reza "de los datos swiss", que es donde estamos trabajando). Para ello recurro a la función *hist()*

```
hist(Education, xlab="Nivel de educación", ylab="Frecuencia absoluta",
main="Histograma de la variable Educación", sub="De los datos Swiss",
cex.sub=0.5)
```



Apartado (b)

En este apartado se solicita proporcionar distintos colores a cada una de las barras del histograma. Hay dos formas de hacerla, la primera es definir cada uno de los colores de este modo:

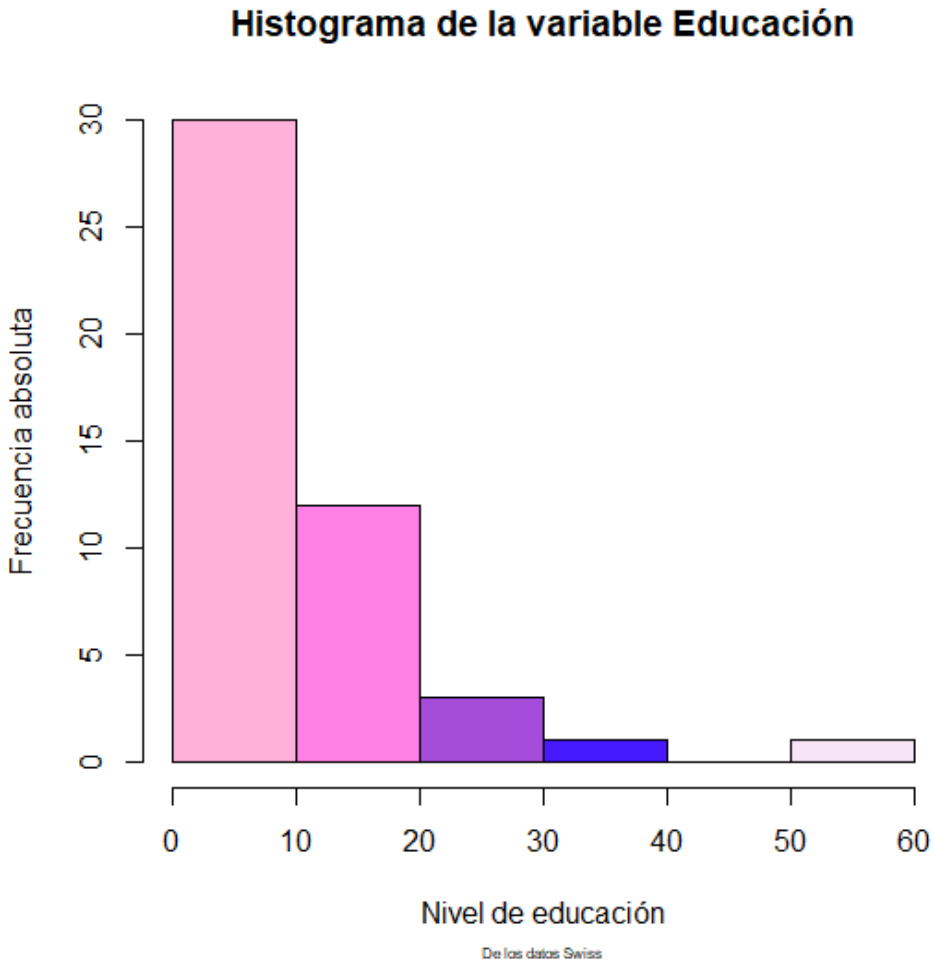
```
colores <- c(rgb(1,0,0.5,0.3),rgb(1,0,0.8,0.5),rgb(0.5,0,0.8,0.7),
rgb(0.2,0,1,0.9),rgb(0.2,0.5,1), rgb(0.7,0,0.7,0.1))
```

Donde rgb corresponde a las palabras *red, green* y *blue* en inglés, rojo, verde y azul en español. Estos son los colores bases que se mezclan para obtener el resto de colores. Así que cada una de los valores numéricos de *rgb()* indica qué *"intensidad"* de cada uno de los colores se requiere para el color buscado. El último digito pertenece al nivel de trasparencia, siendo 0 la trasparencia total y 1 la nula trasparencia.

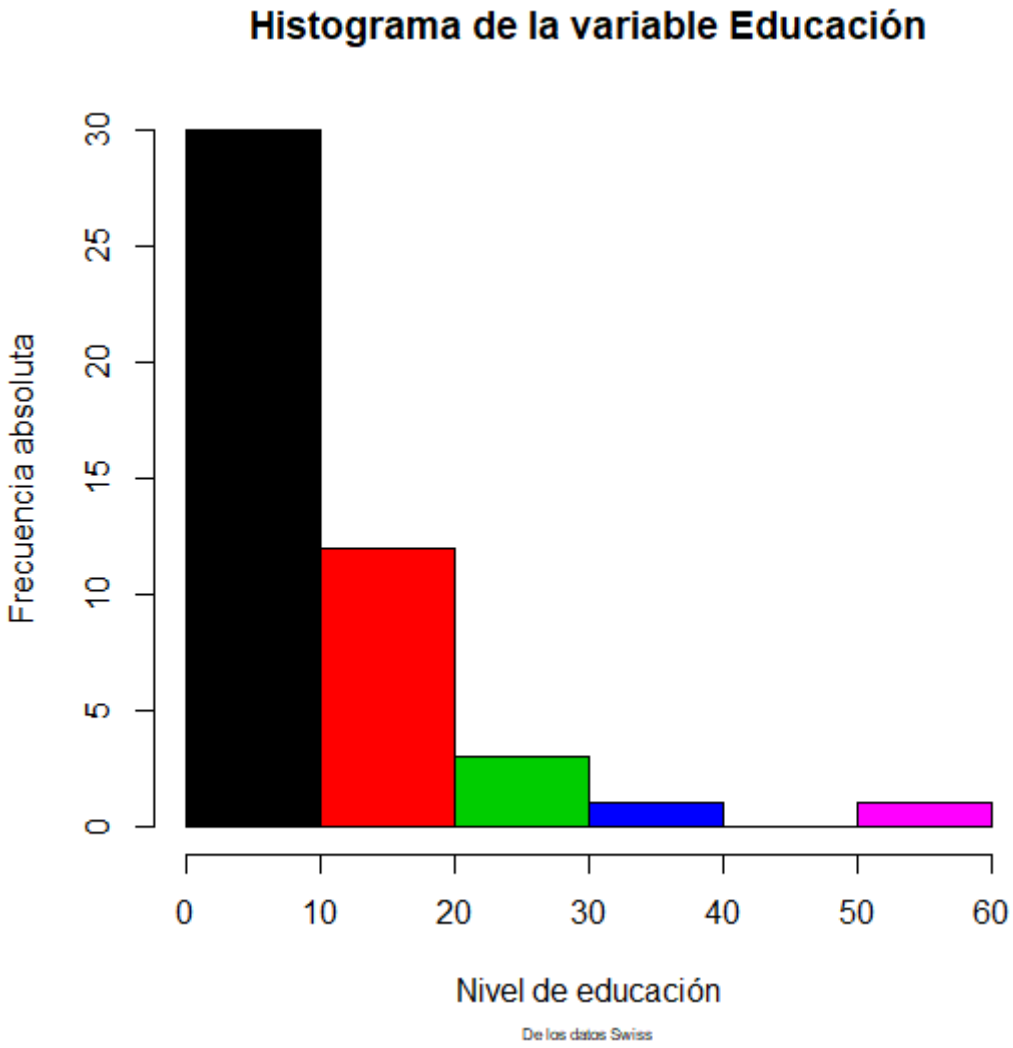
La segunda forma es, simplemente, indicar en el argumento col un número de colores. Cada uno de los colores tiene un número asignado, así que se nota de la siguiente forma:

```
col=1:6
```

```
hist(Education, xlab="Nivel de educación", ylab="Frecuencia absoluta",
main="Histograma de la variable Educación", sub="De los datos Swiss",
cex.sub=0.5, col=colores)
```



```
hist(Education, xlab="Nivel de educación", ylab="Frecuencia absoluta",
main="Histograma de la variable Educación", sub="De los datos Swiss",
cex.sub=0.5, col=1:6)
```



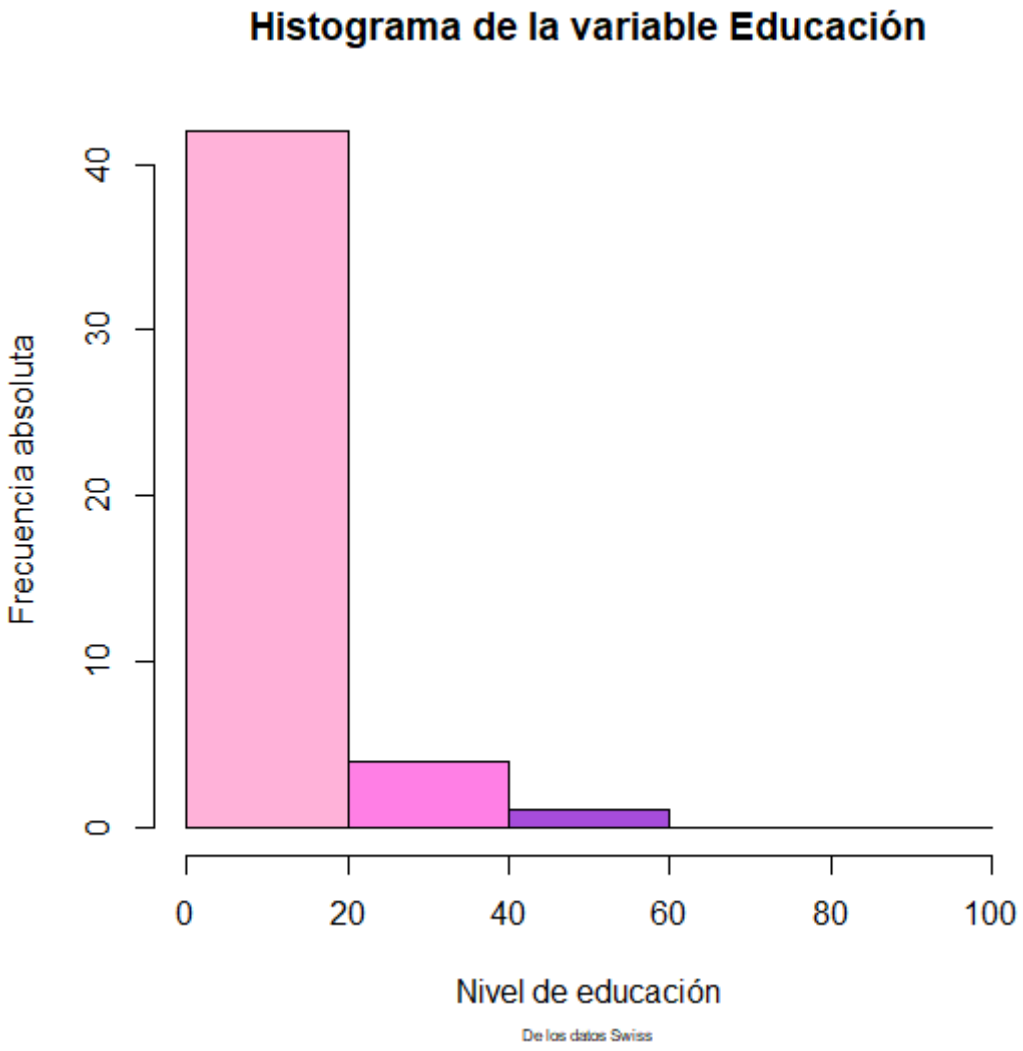
Apartado (c)

Se requiere que los datos se agrupen de la siguiente forma:

$$[0, 20], (20, 40], (40, 60], (60, 80], (80, 100]$$

Para ello usamos *breaks*, que permite delimitar cada una de las barras del histograma.

```
hist(Education, xlab="Nivel de educación", ylab="Frecuencia absoluta",
main="Histograma de la variable Educación", sub="De los datos Swiss",
cex.sub=0.5, col=colores, breaks=c(0,20,40,60,80,100))
```



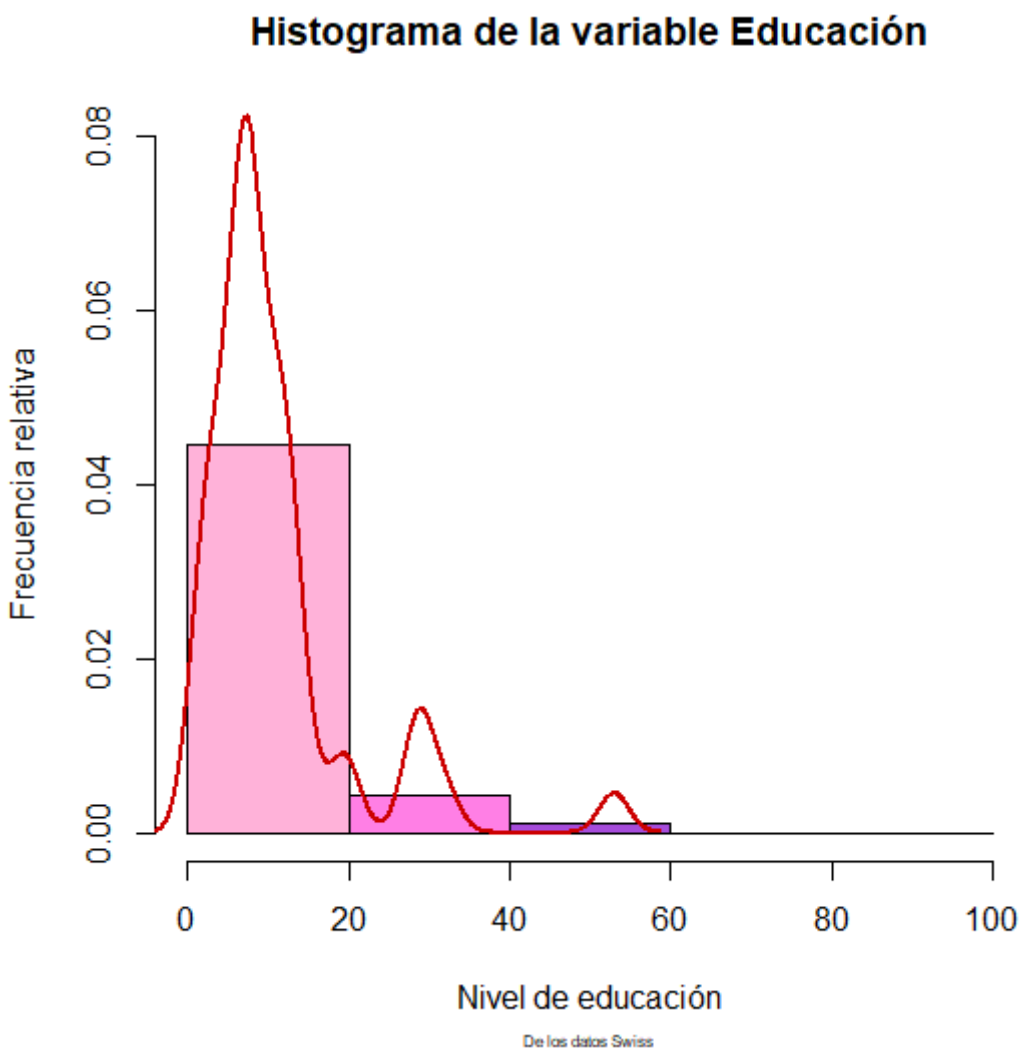
Apartado (d)

El modo *"probabilidad"* del histograma se consigue igualando *freq* a *FALSE*. Esto nos dará la frecuencia relativa, que tomamos como probabilidad. Para la densidad, usamos la función *density()* que estima a la misma.

Para superponerla al histograma, usamos una función gráfica de bajo nivel (i.e, funciones que nos permiten añadir contenidos gráficos a gráficos pre existentes)

Nota: ylim se explicita como sigue ya que, de lo contrario, la densidad obtenida salía de los límites predeterminados del gráfico.

```
hist(Education, xlab="Nivel de educación", ylab="Frecuencia relativa",
main="Histograma de la variable Educación", sub="De los datos Swiss",
cex.sub=0.5, col=colores, ylim=c(0,0.08),breaks=c(0,20,40,60,80,100),
freq=FALSE)
densidad_educacion <- density(Education)
lines(densidad_educacion,col=rgb(0.8,0,0),lwd=2)
```



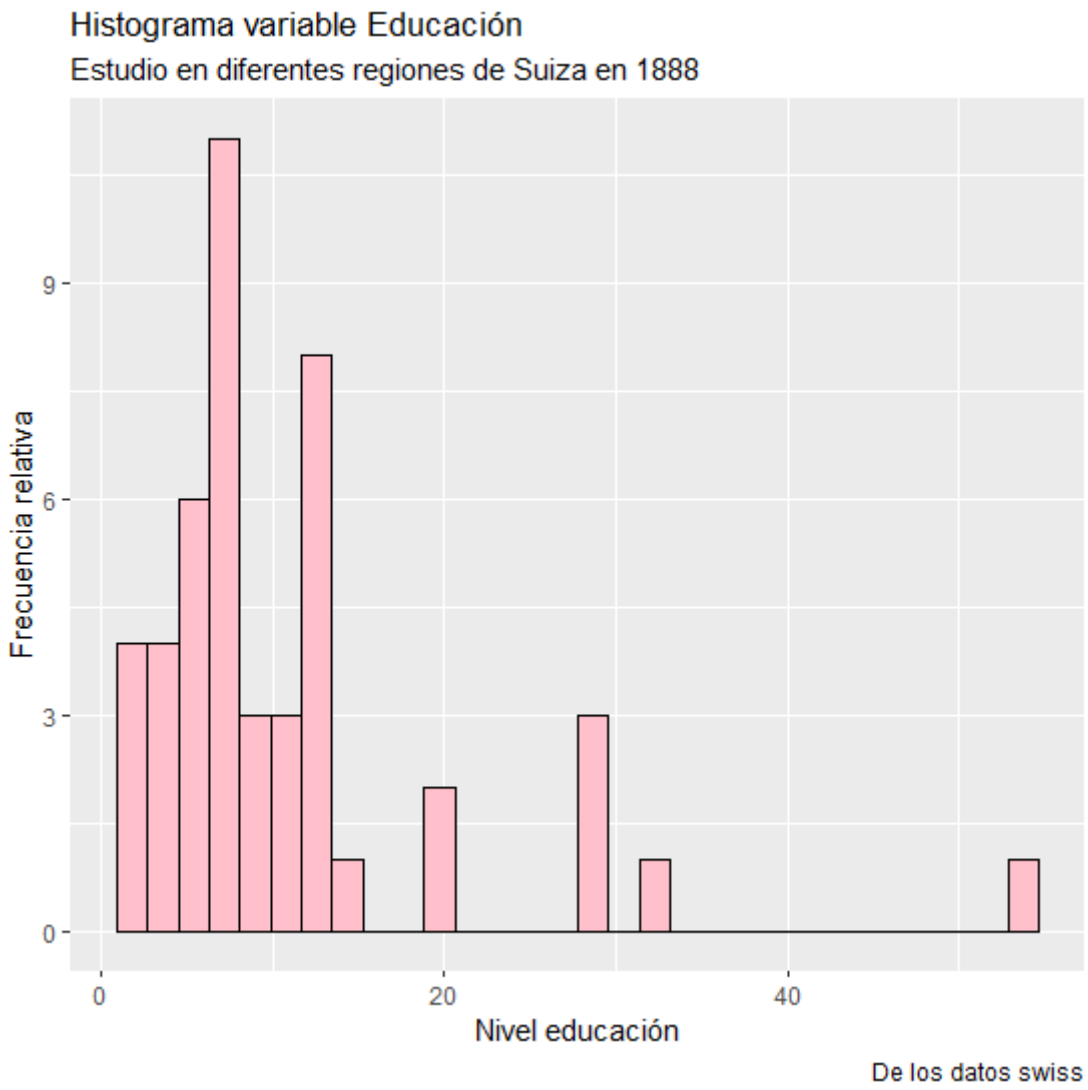
Conclusiones

Se puede observar que cuanto mayor nivel de educación, menos porcentaje poblacional hay. Además se puede intuir que sigue, más o menos, una distribución multimodal. Aunque esto, son sólo intuiciones a partir del histograma. Para conclusiones más profundas, habría que estudiar si nuestras hipótesis son ciertas.

Otro gráfico

Análogamente al apartado (a) R permite hacer histogramas mediante la función *ggplot().*

```
> ggplot(swiss, aes(Education)) +
+ geom_histogram(col='black', fill='pink') +
+ labs(title = "Histograma variable Educación",
+       subtitle = "Estudio en diferentes regiones de Suiza en 1888",
+       caption = "De los datos swiss",
+       x = "Nivel educación",
+       y = "Frecuencia relativa")
```



IV. Otros gráficos

Apartado (a)

Se pretende discretizar dos variables en cuatro grupos de tamaño similar.

Si no nos dijeran en cuántos grupos se han de dividir podríamos haber recurrido a la recomendación que R nos ofrece.

```
> nclass.Sturges(Catholic)
[1] 7
> nclass.Sturges(Agriculture)
[1] 7
```

Como se nos pide que lo hagamos con la función *cut(),* tenemos el siguiente código:

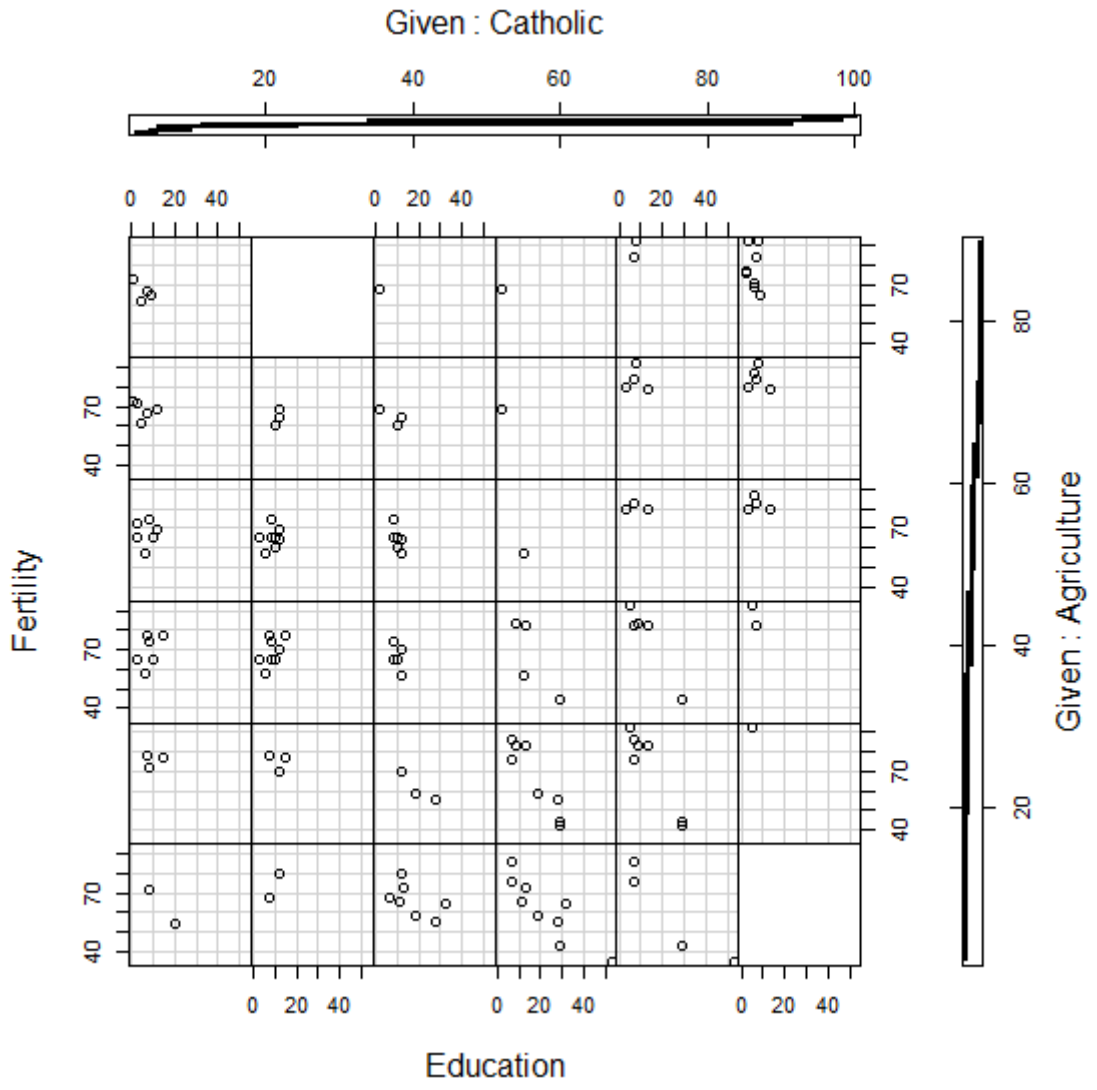
```
> int <- max(Catholic) - min(Catholic)
> division <- int/4
> division
[1] 24.4625

> int <- max(Agriculture) - min(Agriculture)
> division2 <- int/4
> division2
[1] 22.125
```

Primero vemos cuánto hay que sumar en cada intervalo para que recorra todos los datos (desde el mínimo hasta el máximo) y haya exactamente 4 intervalos.

```
> m1 <- min(Catholic)
> cut(Catholic, breaks=c(m1, m1+division, m1+2*division,
+ m1+3*division, m1+4*division))
 [1] (2.15,26.6] (75.5,100]   (75.5,100]   (26.6,51.1] (2.15,26.6] (75.5,100]
 [7] (75.5,100]  (75.5,100]   (75.5,100]   (75.5,100]  (75.5,100]  (2.15,26.6]
[13] (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6]
[19] <NA>        (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6]
[25] (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6]
[31] (75.5,100]  (75.5,100]  (75.5,100]  (75.5,100]  (75.5,100]  (75.5,100]
[37] (75.5,100]  (75.5,100]  (2.15,26.6] (2.15,26.6] (2.15,26.6] (2.15,26.6]
[43] (2.15,26.6] (2.15,26.6] (26.6,51.1] (26.6,51.1] (51.1,75.5]
Levels: (2.15,26.6] (26.6,51.1] (51.1,75.5] (75.5,100]

> m2 <- min(Agriculture)
> cut(Agriculture, breaks=c(m2, m2+division2, m2+2*division2,
+  m2+3*division2, m2 + 4*division2))
 [1] (1.2,23.3]   (23.3,45.5] (23.3,45.5] (23.3,45.5] (23.3,45.5] (23.3,45.5]
 [7] (67.6,89.7] (67.6,89.7] (45.5,67.6] (23.3,45.5] (45.5,67.6] (45.5,67.6]
[13] (45.5,67.6] (45.5,67.6] (67.6,89.7] (67.6,89.7] (23.3,45.5] (1.2,23.3]
[19] (1.2,23.3]  (67.6,89.7] (45.5,67.6] (45.5,67.6] (45.5,67.6] (45.5,67.6]
[25] (67.6,89.7] (45.5,67.6] (45.5,67.6] (45.5,67.6] (23.3,45.5] (45.5,67.6]
[31] (67.6,89.7] (67.6,89.7] (67.6,89.7] (67.6,89.7] (45.5,67.6] (67.6,89.7]
[37] (67.6,89.7] (45.5,67.6] (23.3,45.5] (1.2,23.3]  (1.2,23.3]  (1.2,23.3]
[43] (23.3,45.5] (1.2,23.3]  <NA>        (45.5,67.6] (23.3,45.5]
Levels: (1.2,23.3] (23.3,45.5] (45.5,67.6] (67.6,89.7]
```

Primeramente, hice el coplot con las variables Catholic y Agriculture sin haberlas sometido a *cut()*

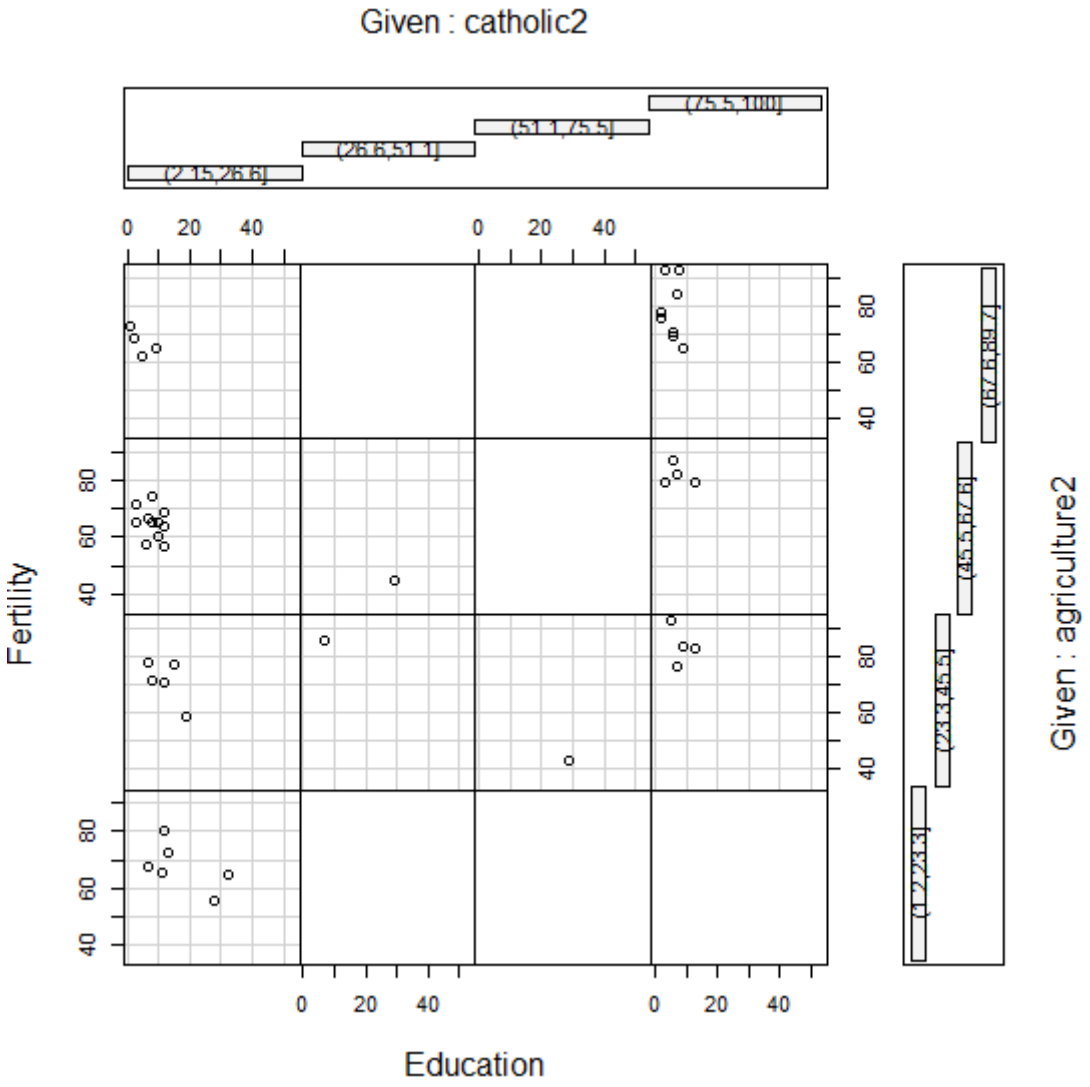
```
coplot(Fertility ~ Education| Catholic * Agriculture)
```



La nulidad de datos en ciertas regiones da lugar a que salgan celdas en blanco.

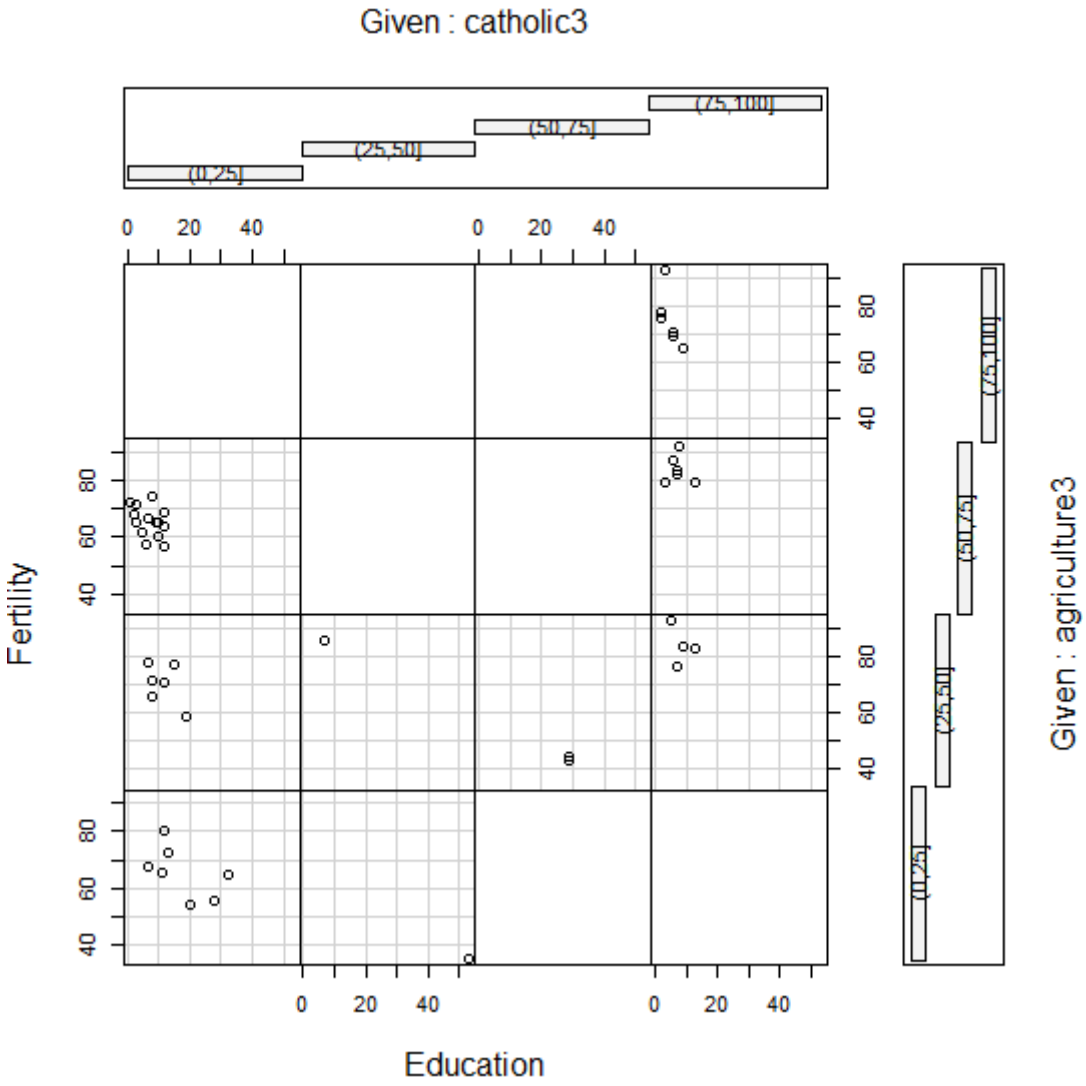
Veamos, ahora, *coplot()* con las variables discretizadas:

```
> catholic2 <-cut(Catholic, breaks=c(m1, m1+division, m1+2*division,
+ m1+3*division, m1+4*division))
>
> agriculture2 <- cut(Agriculture, breaks=c(m2, m2+division2, m2+2*division2,
+   m2+3*division2, m2 + 4*division2))
> coplot(Fertility ~ Education| catholic2 * agriculture2)
```



Nota: También podríamos haber dividido las variables en intervalos [0,25], (25,50], (50,75] y (75,100]. Los he dividido de la anterior forma ya que salvas el caso extremo de que no hubiese ningún dato en el primer intervalo o en el último. Veamos esta división:

```
> catholic3 <-cut(Catholic, breaks=seq(0,100,by=25))
> agriculture3 <- cut(Agriculture, breaks=seq(0,100,by=25))
```


Given : catholic3
(0,25]
(25,50]
(50,75]
(75,100]
Fertility
Education
Given : agriculture3
(0,25]
(25,50]
(50,75]
(75,100]

Apartado (b)

Se pide practicar con otros gráficos. Hay algunos que se podrían haber usado en otros apartados, sin embargo, para no extender demasiado el trabajo, he preferido usarlos exclusivamente en este apartado.

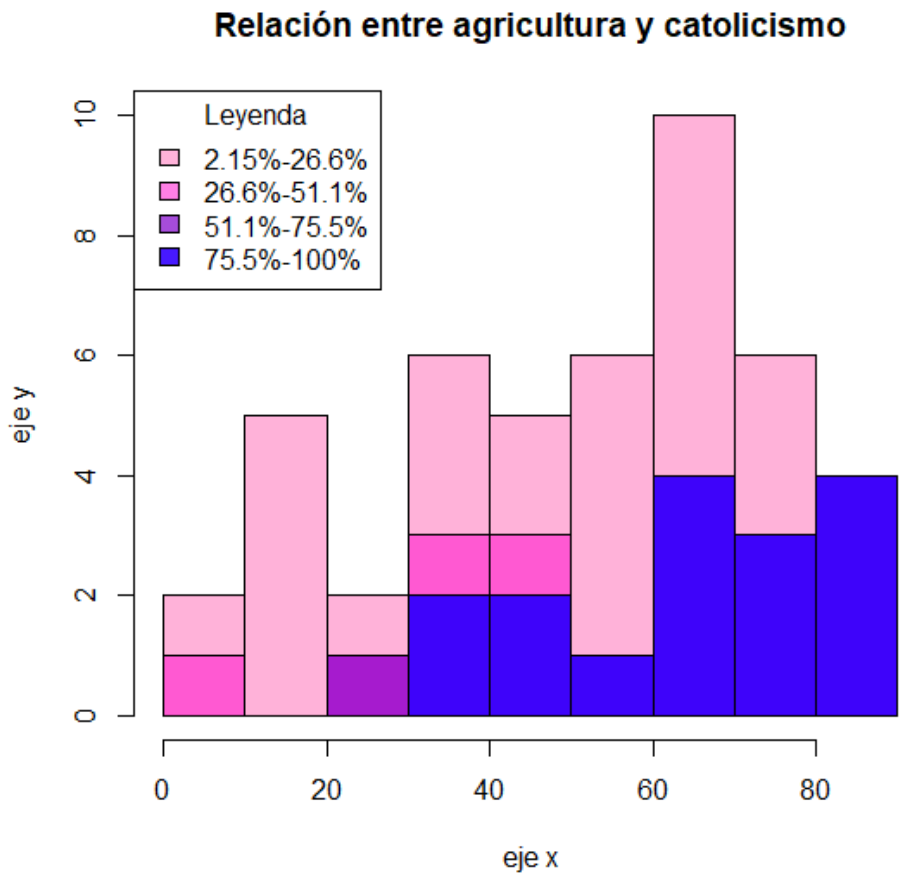
Para usar algunas de las funciones que veremos a continuación, es necesario instalar y cargar el libro *"plotrix"*.

```
> install.packages("plotrix")

> library(plotrix)
```

histStack()

Empezamos con la función *histStack(),* que permite crear histogramas de variables cuantitativas mediante los valores de la variable de un factor.

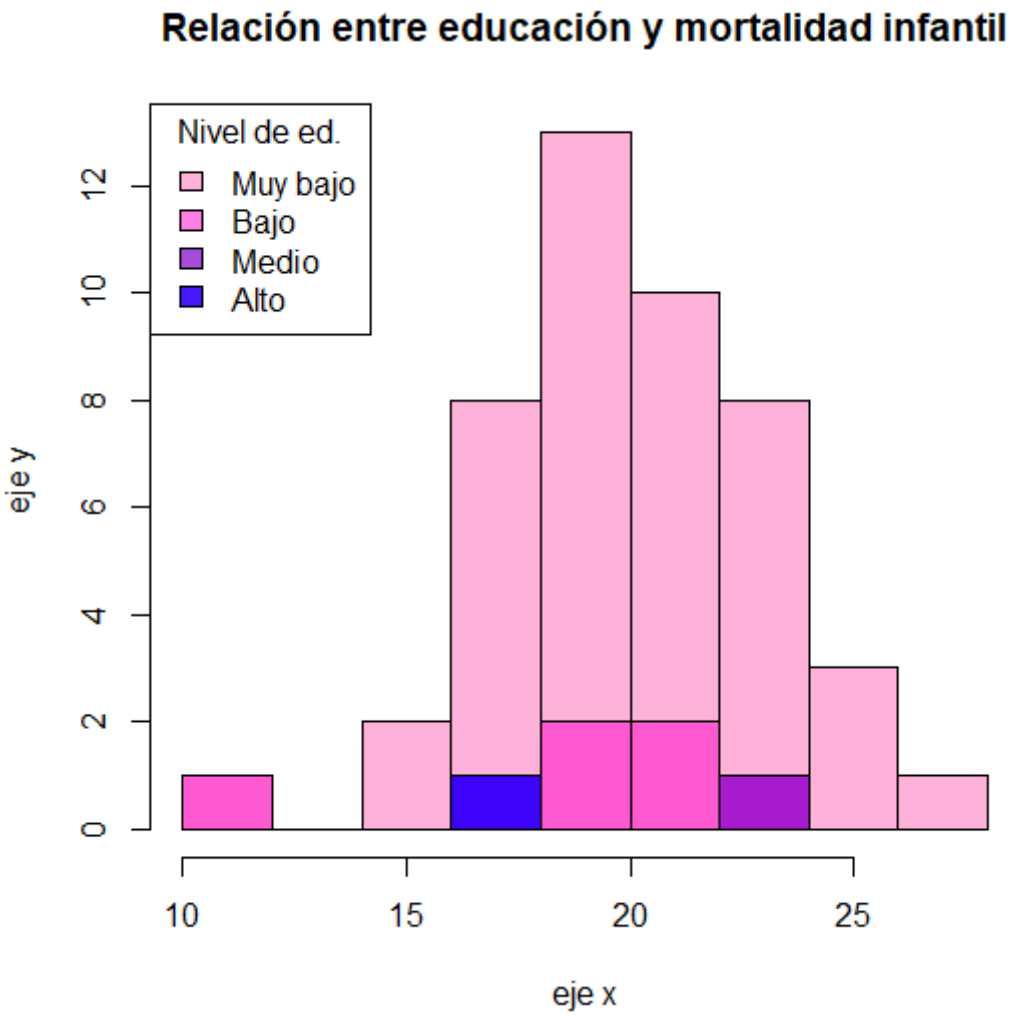
```
> factor_c <- cut(Catholic, breaks=c(ml, ml+division, ml+2*division,
+ ml+3*division, ml+4*division))
> str(factor_c)
 Factor w/ 4 levels "(2.15,26.6]",..: 1 4 4 2 1 4 4 4 4 4 ...

> levels(factor_c)
[1] "(2.15,26.6]" "(26.6,51.1]" "(51.1,75.5]" "(75.5,100]"
```



Nota: Lo que podemos observar, es que cuánto mayor nivel de catolicismo menos es la altura de las barras. Quizá haya una relación directa, cuánto más catolicismo hay en la región menos agricultores. Aunque dicha afirmación, habría que ver cómo de correcta es a través de modelizaciones y verificaciones.

```
> range(Education)
[1]  1 53
> eje_ed <- c(1,15,30,45,60)
> cut(Education, breaks=eje_ed)
 [1] (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]
[11] (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (15,30] (15,30] (1,15]
[21] (1,15]  (1,15]  (1,15]  (1,15]  <NA>    (1,15]  (1,15]  (1,15]  (15,30] (1,15]
[31] (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]  (1,15]
[41] (1,15]  (30,45] (1,15]  (1,15]  (45,60] (15,30] (15,30]
Levels: (1,15] (15,30] (30,45] (45,60]
```

```
> c <- cut(Education, breaks=eje_ed)
> histStack(Infant.Mortality ~ c, swiss,
+ col=colores, xlab="eje x", ylab="eje y",
+ main="Relación entre educación y mortalidad infantil")

legend("topleft", fill = colores, title="Nivel de ed.",
legend=c("Muy bajo","Bajo","Medio","Alto"))
```

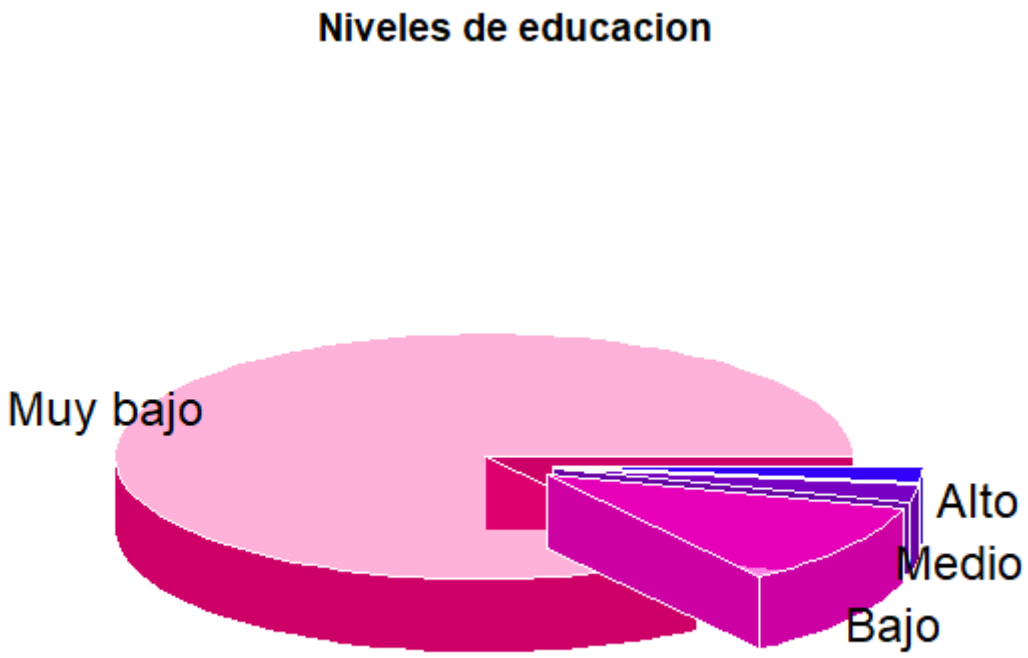


Observamos que la mortalidad más alta (la zona del 20% al 25%) está casi enteramente ocupada por el color más claro que corresponde a la menor educación. Sin embargo, no parece que haya una relación lineal entre ambas variables.

pie3D()

Aquí se pretende ver, a través de la función pie3D(), cómo se divide la población en relación al nivel de educación.

```
> etiquetas <- c("Muy bajo","Bajo","Medio","Alto")
> c2 <- table(c)
> pie3D(c2,labels=etiquetas,main="Niveles de educacion",col=colores,
+ explode=0.1,labelcol="black",border="white")
```



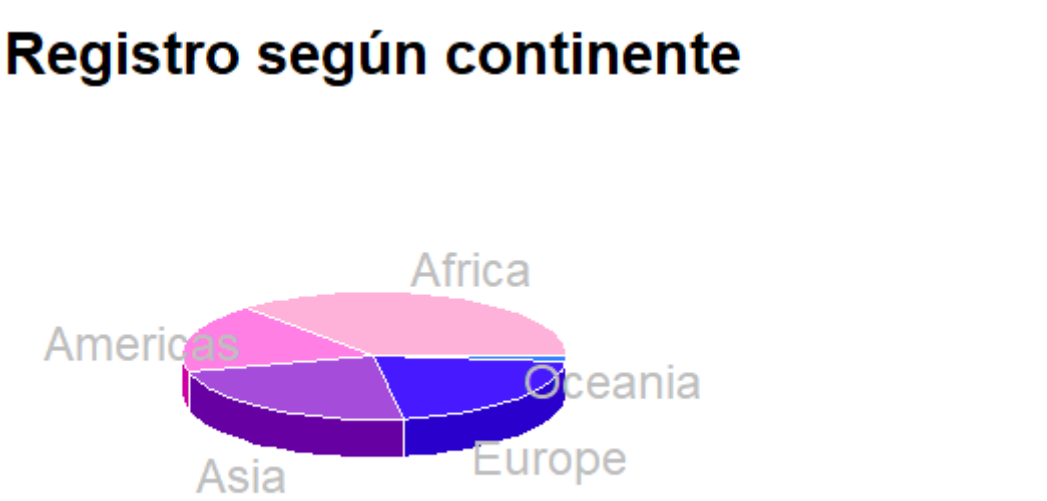
Para el segundo ejemplo, vamos a recurrir a unos datos ya utilizados anteriormente en la asignatura. Usamos los datos *"2Sesion_ECI_datos"* y estudiaremos, de nuevo con la función *pie3D(),* cómo se distribuyen los datos según la variable *"continent"*.

Cargamos los datos:

```
> datos <- read.csv("2Sesion_ECI_datos.txt")
> head(datos)
      country year      pop continent lifeExp gdpPercap
1 Afghanistan 1952  8425333      Asia  28.801  779.4453
2 Afghanistan 1957  9240934      Asia  30.332  820.8530
3 Afghanistan 1962 10267083      Asia  31.997  853.1007
4 Afghanistan 1967 11537966      Asia  34.020  836.1971
5 Afghanistan 1972 13079460      Asia  36.088  739.9811
6 Afghanistan 1977 14880372      Asia  38.438  786.1134
> str(datos)
'data.frame':   1704 obs. of  6 variables:
 $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
 $ pop      : num  8425333 9240934 10267083 11537966 13079460 ...
 $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
 $ gdpPercap: num  779 821 853 836 740 ...
```

Vamos a hacer el gráfico con los datos de la variable continent.

```
> attach(datos)
The following objects are masked from datos (pos = 3):

    continent, country, gdpPercap, lifeExp, pop, year

> c2 <- table(continent)
>
> etiquetas2 <- c("Africa", "Americas", "Asia", "Europe", "Oceania")
>
> colores <- c(rgb(1,0,0.5,0.3),rgb(1,0,0.8,0.5),rgb(0.5,0,0.8,0.7),
+ rgb(0.2,0,1,0.9),rgb(0.2,0.5,1), rgb(0.7,0,0.7,0.1))
>
> pie3D(c2,labels=etiquetas2,main="Registro según continente",col=colores,
+ labelcol="grey",border="white", cex.main=1.2, mar=c(4,4,4,8))
>
>
> detach(datos)
```

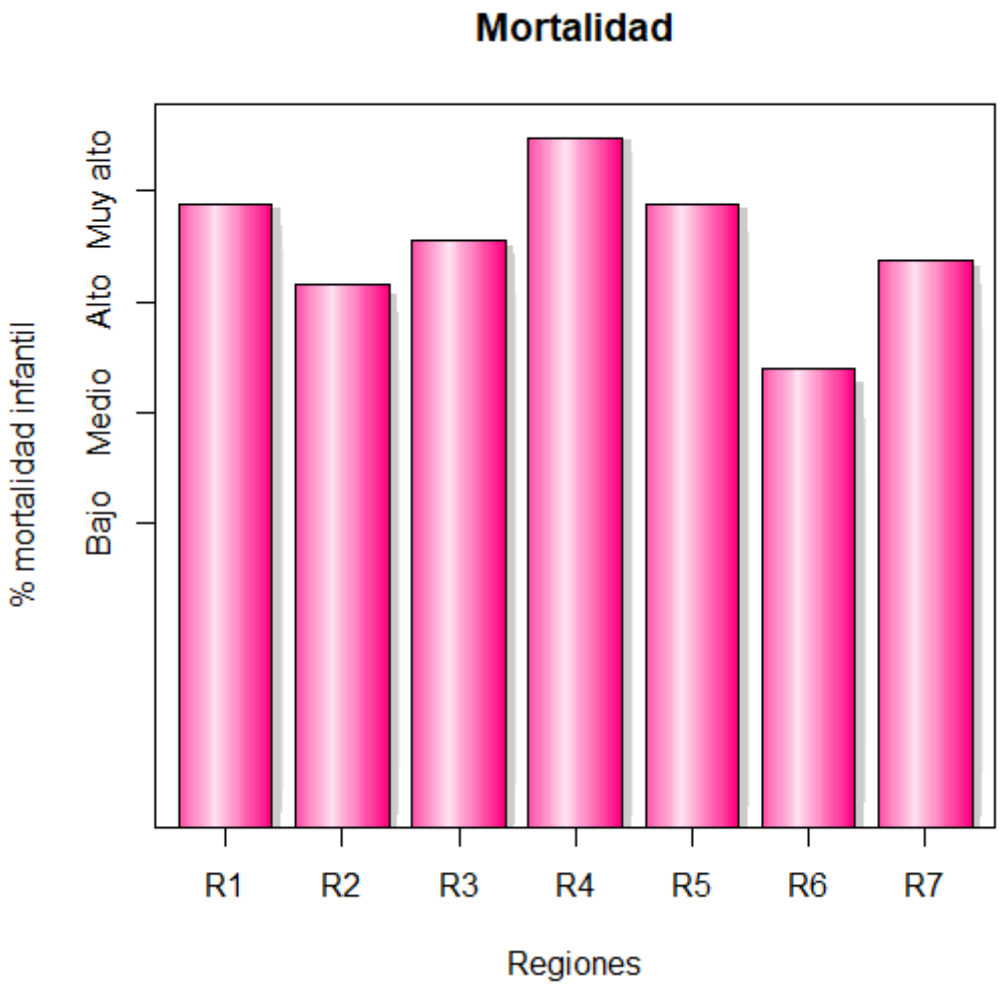


barp()

Para esta función uso, de nuevo, la variable de la mortalidad infantil. He hecho un *sampleo* de los datos para que el diagrama de barras sea de 7 muestras aleatorias de los mismos.

Además, he puesto unas etiquetas basándome en el mínimo y el máximo de los datos observados.

```
> range(Infant.Mortality)
[1] 10.8 26.6
> eje_mort <- c(10.8,14.75,18.7,22.65,26.6)
> nombres <- c("Bajo","Medio","Alto","Muy alto","Extremo")
> nombres2 <- c("R1","R2","R3","R4","R5","R6","R7")

> n <- length(Infant.Mortality)
> s <- sample(1:n, 7, replace=FALSE)
> regiones <- Infant.Mortality[s]

> barp(regiones, names.arg=nombres2, col=colores, xlab="Regiones",
+  main="Mortalidad", height.lab=nombres, height.at=eje_mort,
+ ylab="% mortalidad infantil",shadow=TRUE,cylindrical=TRUE)
```

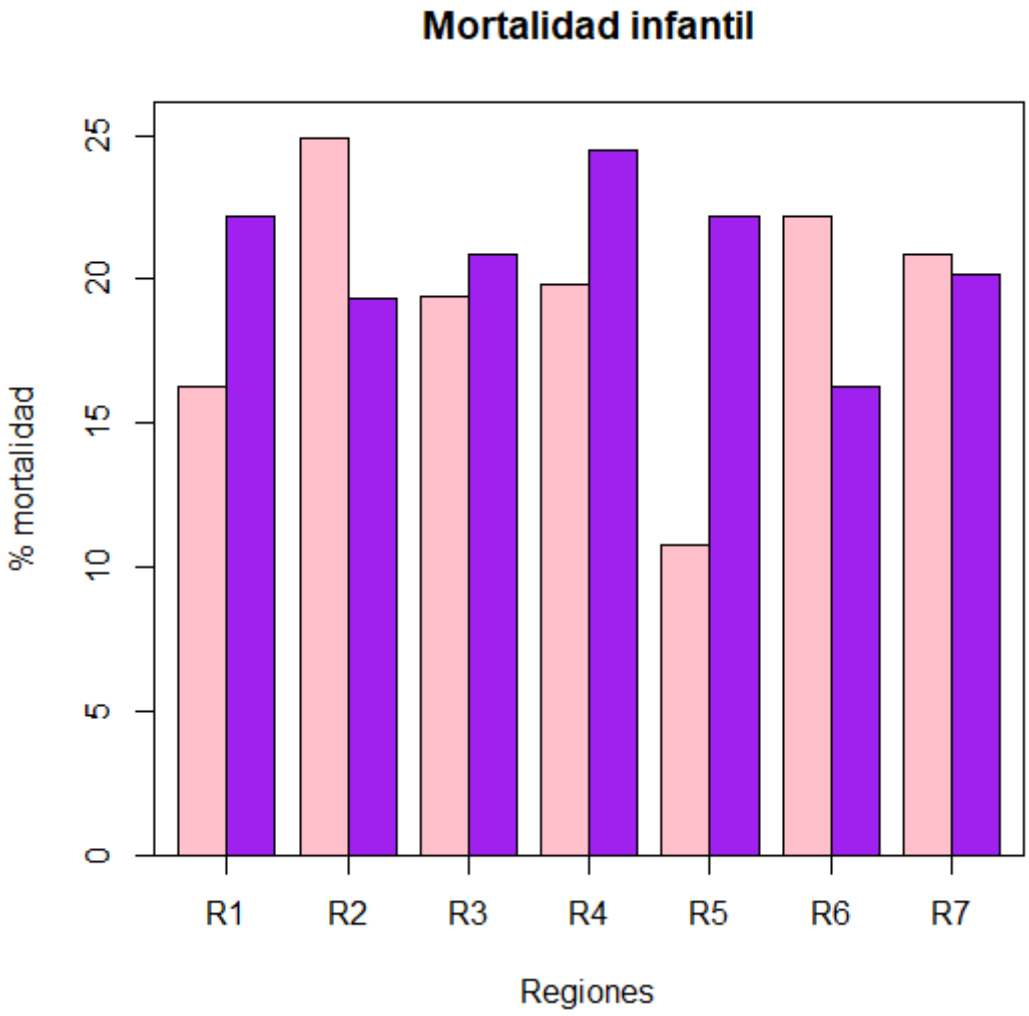


Ahora, hago un segundo *sampleo* con, de nuevo, 7 muestras aleatorias.

```
> s2 <- sample(1:n, 7, replace=FALSE)
> regiones2 <- Infant.Mortality[s2]

> regiones3 <- rbind(regiones1, regiones2)
```

Llamo regiones3 a la variable que contendrá ambos registros y, con la función *barp(),* hago el diagrama:

```
> barp(regiones3, names.arg=nombres2, xlab="Regiones",
+  main="Mortalidad infantil", col=c("pink","purple"),
+ ylab="% mortalidad")
```




```
> detach(swiss)
```

Llamo a la función *detach(),* ya que de aquí en adelante, no vuelvo a usar las variables de los datos *swiss*.

En este ejemplo, integraré un ejercicio práctico de una asignatura del grado de matemáticas para dotar de más transversalidad a la creación de gráficos. En concreto, en este ejemplo se estudia el comportamiento (respecto a los parámetros α y β) de una determinada población. Se usa, para ello, la tasa neta de reproducción de la población.

α y β las denominamos como x e y para simplificar el manejo de datos.

$$f(x) = \frac{96-8\pi^2-96sin(x)}{3\pi^2}$$

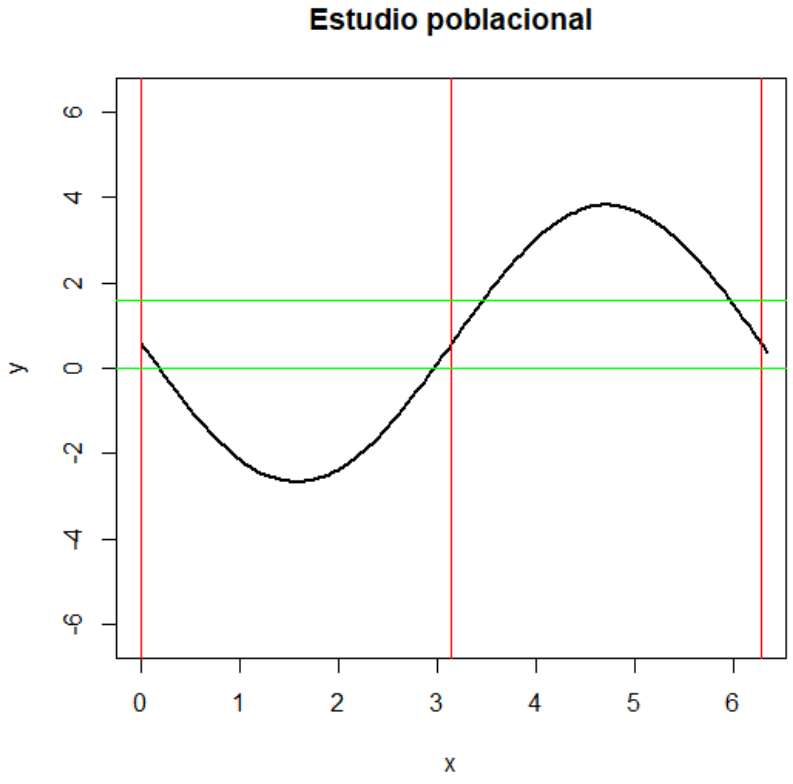
Esta función es la tasa neta de reproducción que nos ayudará a determinar en qué condiciones la población converge, crece ilimitadamente o se extingue.

```
> x <- seq(0, 2*pi+0.1, 0.05)
> y <- (96-8*(pi^2)-96*sin(x))/(3*(pi^2))
```

Definimos varios puntos de x y con ellos, construimos y (i.e, f(x)).

```
> plot(x,y, xlim=c(0,2*pi), ylim=c(-2*pi,2*pi), type="l", lwd=2,
+ main="Estudio poblacional")
> abline(v=0, col="red")
> abline(v=2*pi, col="red")
> abline(v=pi, col="red")
>
> abline(h=0, col="green")
> abline(h=pi/2, col="green")
```

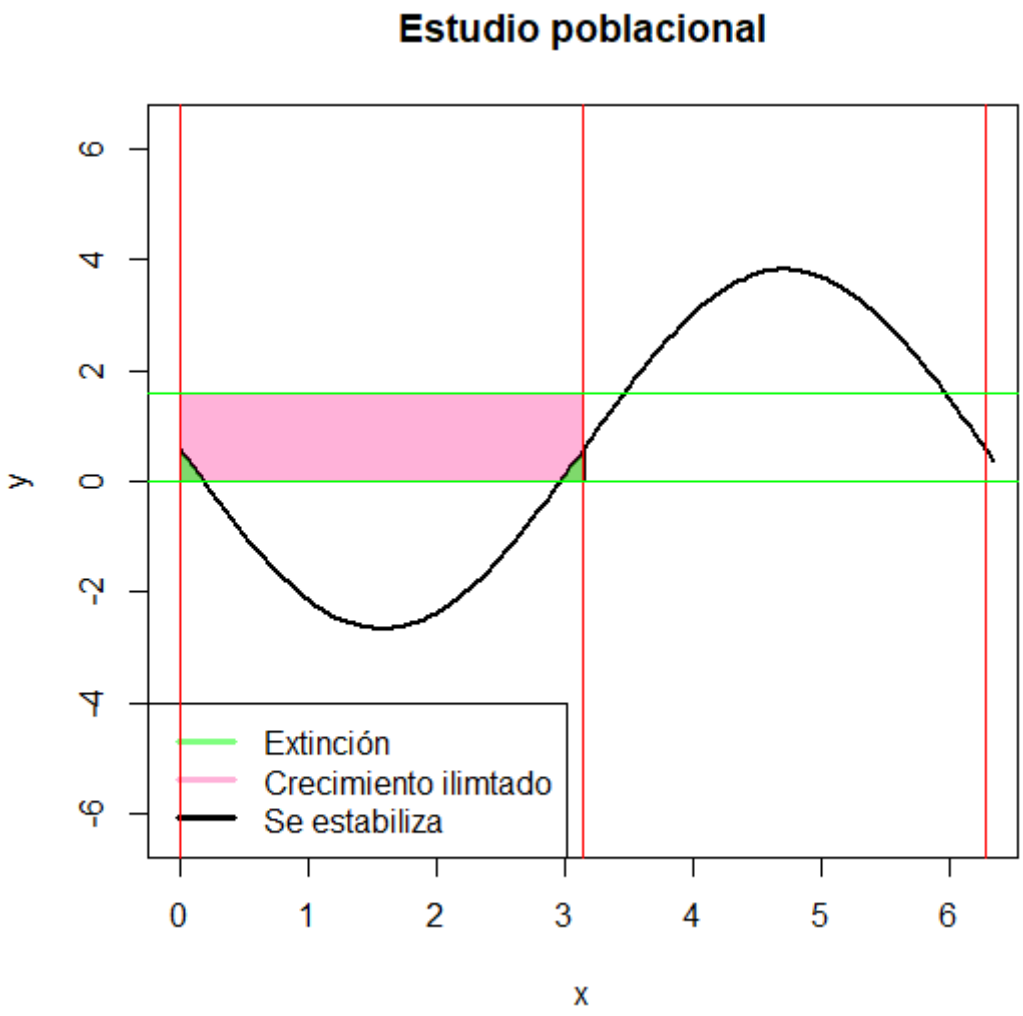
Hacemos el gráfico con la ayuda de la función plot() de x e y. Además añadimos una serie de líneas horizontales y verticales:



Por unos estudios previos, sabemos que en las regiones donde *sen(x)<0* no tiene sentido estudiar el comportamiento de la población. Y, además, también sabemos que *y* está entre 0 y pi/2. Por ello, precisamente, añadimos las líneas ya definidas.

Sabemos que las regiones que se encuentran por debajo de la gráfica, se extinguen, las que se encuentran por encima, crecen ilimitadamente y las que coinciden con la función, convergen.

Veamos dichas áreas gráficamente.



Para ello usé la función *polygon()* que nos permite colorear ciertas partes de las gráficas.

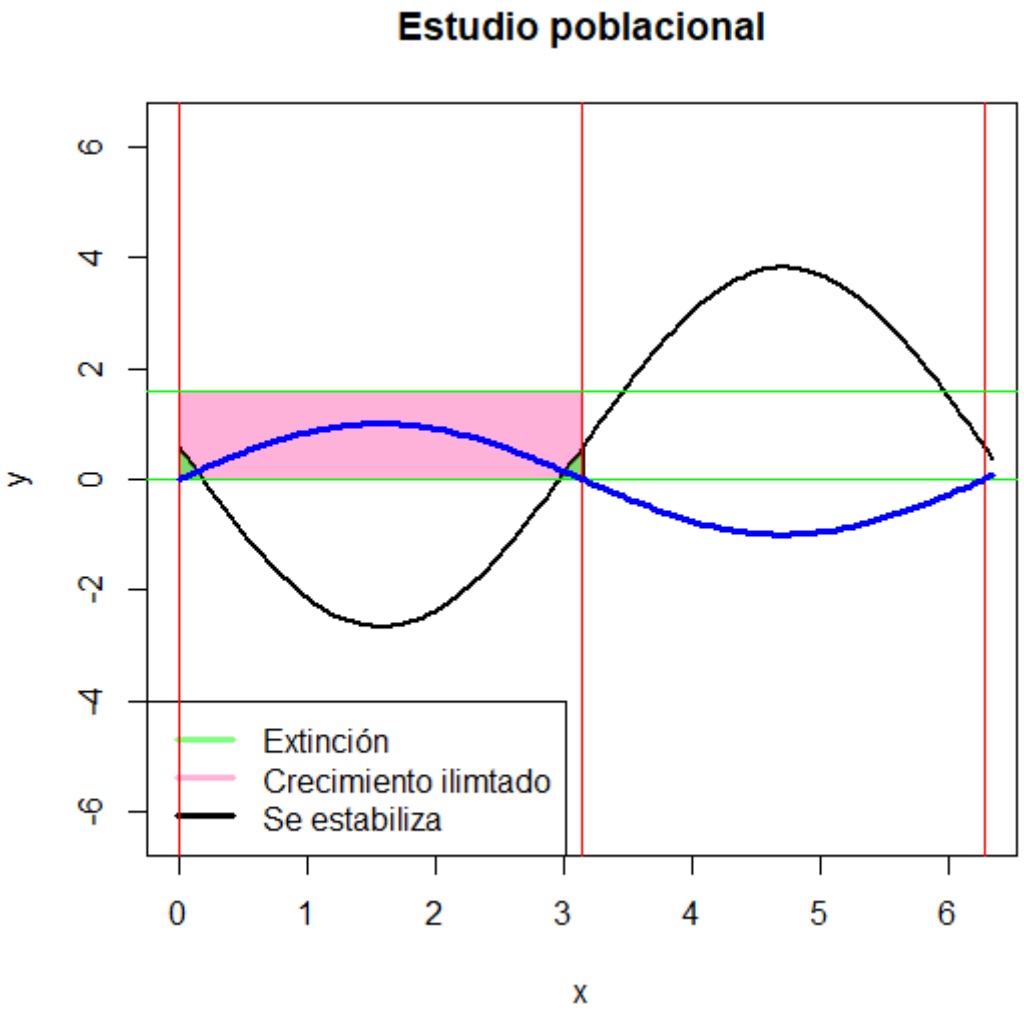
```
> polygon(x=c(0,pi,pi,0), y=c(0,0,pi/2,pi/2), col=rgb(1,0,0.5,0.3))
>
> minimox <- min(which(x >=0))
> maximox <- max(which(x <=0.2))
> polygon(x = c(x[c(minimox, minimox:maximox, maximox)]), y = c(0,
+     y[minimox:maximox], 0), col = rgb(0, 1, 0, 0.5))
>
> minimox2 <- min(which(x>=pi-0.2))
> maximox2 <- max(which(x<=pi+0.05))
> polygon(x = c(x[c(minimox2, minimox2:maximox2, maximox2)]), y = c(0,
+     y[minimox2:maximox2], 0), col = rgb(0, 1, 0, 0.5))
```

Y, además, el código para la leyenda:

```
> colores <- c(rgb(0, 1, 0, 0.5),rgb(1,0,0.5,0.3), "black")
>
> legend("bottomleft", col=colores, legend=c("Extinción", "Crecimiento ilimtado$
+ "Se estabiliza"), lwd=3)
```

Asimismo, al gráfico, añadí la función seno para comprobar, que, efectivamente, es negativo, desde pi a 2*pi (y, debido a unas discusiones matemáticas previas, no es necesario estudiar qué hace la función descrita en esa zona):

```
> lines(x,sin(x),col="blue",lwd=3)
```



Podríamos, también, identificar dónde están algunos puntos que nos podrían ser útiles (como el punto de la gráfica que intersecta con las líneas verticales x=pi y x=2*pi)

```
> locator(2)
$x
[1] 3.146284 6.289324

$y
[1] 0.5206527 0.5568091
```

ggplot

Para este apartado haré uso de varias bibliotecas:

```
> library(tidyverse)
-- Attaching packages ------------------
v ggplot2 3.3.2     v purrr   0.3.4
v tibble  3.0.3     v dplyr   1.0.1
v tidyr   1.1.1     v stringr 1.4.0
v readr   1.3.1     v forcats 0.5.0
-- Conflicts ---------------------------
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

tidyverse nos aporta algunas librerías que son muy útiles en algunos contextos. Por ejemplo *tibble* es una especie de *data frame* mejorado donde se hacen los mínimos cambios posibles respecto a los datos. Además la forma de presentación de los datos está optimizada. También es importante la función *filter()* que nos facilita la declaración de subconjuntos dentro de los *data frame (o tibbles)*.

```
> data(package = .packages(all.available = TRUE))
```

Como quería trabajar con otros datos, miré qué conjuntos de datos hay en R.

Usaré los datos *death_prob*, que contienen las probabilidades de muerte según la edad y el sexo.

```
> library(dslabs)
> data(death_prob)
> head(death_prob)
  age  sex     prob
1   0 Male 0.006383
2   1 Male 0.000453
3   2 Male 0.000282
4   3 Male 0.000230
5   4 Male 0.000169
6   5 Male 0.000155
> attach(death_prob)
```

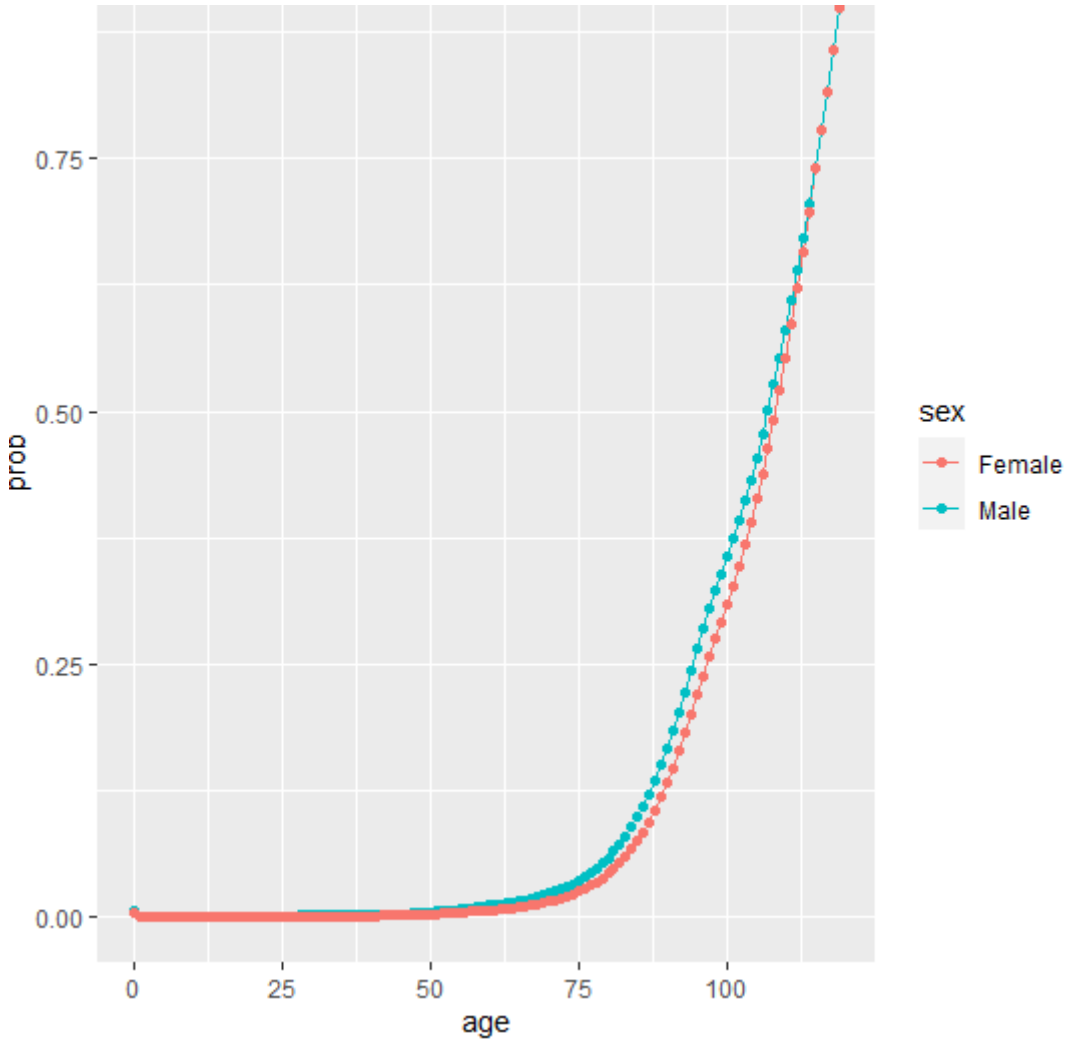
Miro qué niveles constituyen la variable sexo (podría ser, por ejemplo, que se considerara sexo intersexual):

```
> levels(sex) ##ver
[1] "Female" "Male"
```

Y, ahora, declaro lo siguiente:

```
> ggplot(data=death_prob, aes(x=age, y=prob, color=sex)) +
+ geom_line() +
+ geom_point()
```

Se considera que el eje x será la edad, el eje y es la probabilidad y se pide que se distinga según el sexo. Además se pide que cada (x,y) se represente con un punto y que los puntos se unan por líneas.



Veamos otro ejemplo de *ggplot()* con el siguiente conjunto de datos:

```
> library(Lahman)
> data(Teams)
> attach(Teams)
> ?Teams
starting httpd help server ... done
```

Description

Yearly statistics and standings for teams

Son las estadísticas de los equipos de baseball americano durante los años 1871 y 2019. Las variables que utilizaré son:

WSWin

World Series Winner (Y or N)

Si son ganadores de la serie mundial o no.

SB

Stolen bases

Las bases robadas.

G

Games played

Los juegos totales.

R

Runs scored

Y las carreras anotadas.

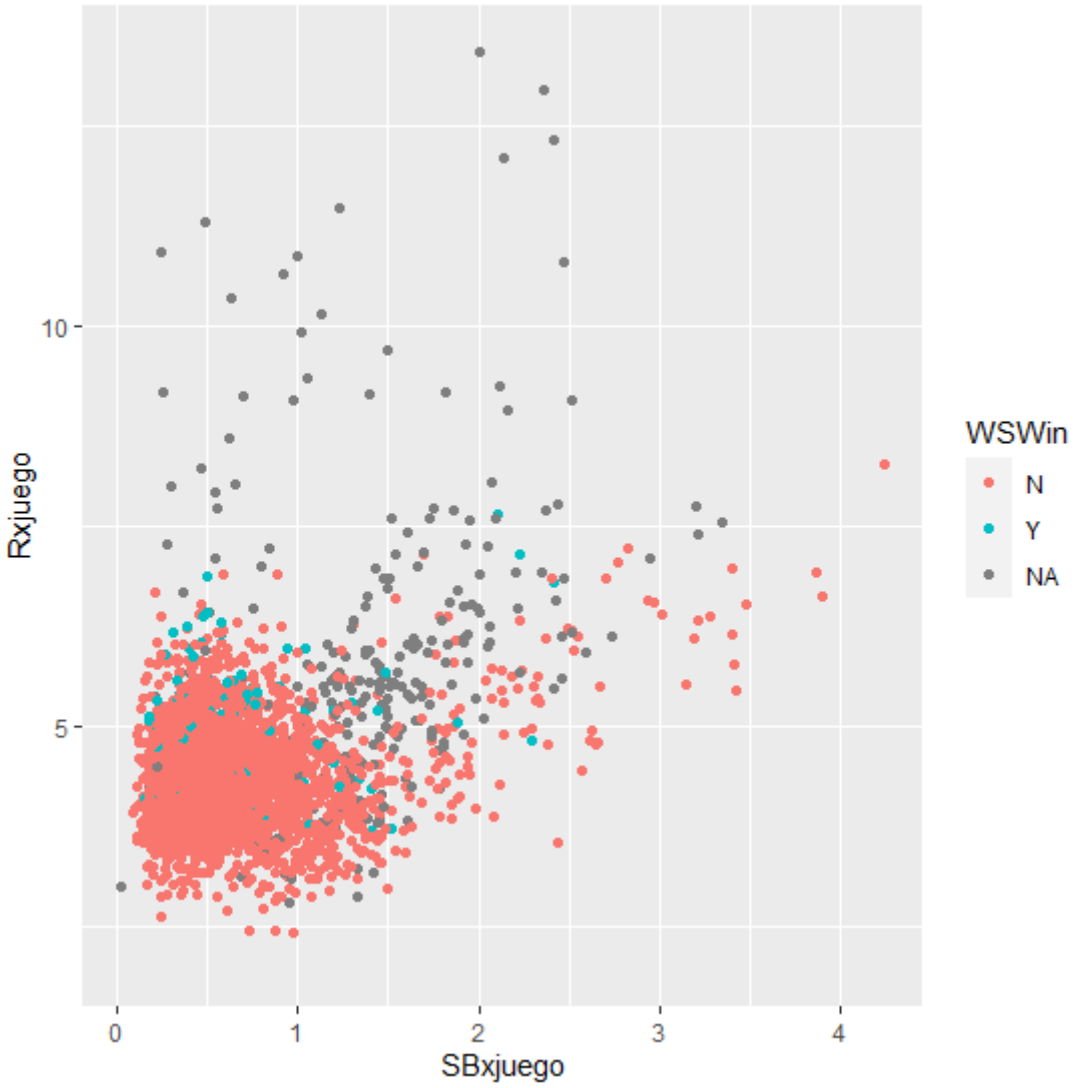
```
> equipos <- Teams[Teams$yearID>=1980 && Teams$yearID<=2001]
> SBxjuego <- SB / G
> Rxjuego <- R / G
```

Hago un subconjunto de datos para manejar únicamente los que se comprendan entre el año 1980 y 2001. Y, además, declaro dos variables que serán: las bases robadas por juego y las carreras anotadas por juego.

Vemos si tienen alguna relación mediante el siguiente gráfico:

```
> ggplot(equipos, aes(SBxjuego, Rxjuego, color=WSWin)) +
+   geom_point()
```

Comprobamos, por lo menos, que no existe ninguna relación lineal entre las dos variables que elegimos para estudiar:



Vemos un último ejemplo donde se pretende modelizar la relación entre dos variables: la concentración en sangre y RLU (unidades en las que se mide la quimioluminiscencia).

Declaramos, ahora dos vectores y declaramos un *data frame* que contenga a ambos:

```
> RLU <- c(0.11, 0.22, 0.40, 0.79, 1.59, 2.84, 9.98)
> conc <- c(0.16, 0.32, 0.63, 1.25, 2.5, 5, 10)
> datos <- data.frame(RLU, conc)
```

Describimos tres modelos diferentes: un modelo lineal y dos polinómicos que se ajusten a los datos con los que estamos trabajando:

```
> modl <- lm(conc ~ RLU)
> modl

Call:
lm(formula = conc ~ RLU)

Coefficients:
(Intercept)          RLU
     0.5821       0.9909
```

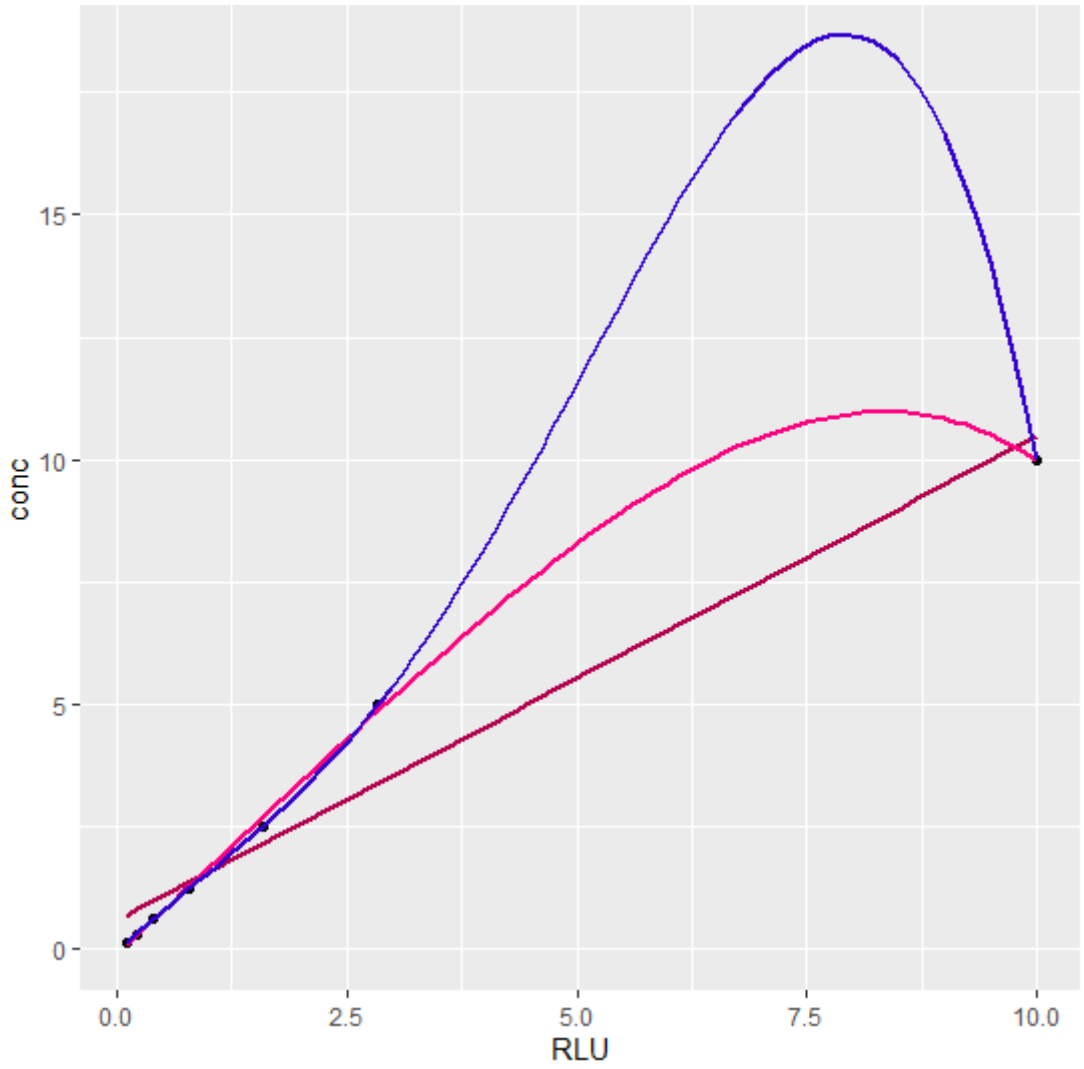
Para los modelos polinómicos usamos la función *I()* para incluir términos específicos en el modelo (x^i, i=2,3,…,n):

```
> mod2 <- lm(conc ~ RLU + I(RLU^4))
> mod2

Call:
lm(formula = conc ~ RLU + I(RLU^4))

Coefficients:
(Intercept)          RLU      I(RLU^4)
  -0.104910     1.772051    -0.000764

>
> mod3 <- lm(conc ~ RLU + I(RLU^2) + I(RLU^3) + I(RLU^4))
> mod3

Call:
lm(formula = conc ~ RLU + I(RLU^2) + I(RLU^3) + I(RLU^4))

Coefficients:
(Intercept)          RLU      I(RLU^2)      I(RLU^3)      I(RLU^4)
   -0.04831      1.81421      -0.33985       0.15058      -0.01249
```

Para el gráfico, nos ayudamos de la función *geom_smooth()* que nos permite sobreponer ciertos trazados para entender mejor el comportamiento de los datos:

**se=FALSE*, si fuese *TRUE* nos sombrea el intervalo de confianza y, por defecto, es *TRUE*.

```
> ggplot(datos, aes(x=RLU, y=conc)) +
+   geom_point() +
+   geom_smooth(method='lm', formula=y~x, se=F, col=rgb(0.7,0,0.3)) +
+   geom_smooth(method='lm', formula=y~x+I(x^4), se=F, col=rgb(1,0,0.5)) +
+   geom_smooth(method='lm', formula=y~x+I(x^2)+I(x^3)+I(x^4), se=F,
+ col=rgb(0.2,0,0.8))
```



Se puede estudiar, de hecho, cómo se ajustan los modelajes a los datos mediante la función *summary()*.

EJERCICIO 2

Apartado (a)

VADeaths contiene datos sobre índices de mortalidad (por 1000) en Virginia en 1940. Más específicamente, *VADeaths* es una tabla de contingencia que contiene datos clasificados por grupos de edad y de zonas.

```
> data(VADeaths)
> head(VADeaths)
      Rural Male Rural Female Urban Male Urban Female
50-54       11.7          8.7       15.4          8.4
55-59       18.1         11.7       24.3         13.6
60-64       26.9         20.3       37.0         19.3
65-69       41.0         30.9       54.6         35.1
70-74       66.0         54.3       71.1         50.0
```

Vemos, efectivamente, que se trata de una tabla de contingencia.

Nota: head() en este caso nos muestra la tabla entera, ya que, en este caso, coincide que las primeras seis líneas contienen la tabla entera con la que trabajaremos.

```
> ?VADeaths
```

Description

Death rates per 1000 in Virginia in 1940.

Details

The death rates are measured per 1000 population per year. They are cross-classified by age group (rows) and population group (columns). The age groups are: 50–54, 55–59, 60–64, 65–69, 70–74 and the population groups are Rural/Male, Rural/Female, Urban/Male and Urban/Female.

Apartado (b)

Para poder conocer un poco más los datos con los que estamos tratando, usamos *str():*

```
> str(VADeaths)
 num [1:5, 1:4] 11.7 18.1 26.9 41 66 8.7 11.7 20.3 30.9 54.3 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:5] "50-54" "55-59" "60-64" "65-69" ...
  ..$ : chr [1:4] "Rural Male" "Rural Female" "Urban Male" "Urban Female"
```

Vemos, entonces, que tenemos cinco grupos de edad [50-54],[55,59],[60-64],[65,69],[70-74] y cuatro grupos demográficos “Hombre rural”, “Mujer rural”, “Hombre urbano”, “Mujer urbana”.

Se nos pide construir una tabla con los datos recogidos pero los datos ya son en sí mismos una tabla. Podríamos tratarlos como un *data frame,* que nos pueda dar algo de más información sobre cómo son las diferentes variables:

```
> data_frame <- as.data.frame.table(VADeaths)
> head(data_frame)
   Var1         Var2 Freq
1 50-54   Rural Male 11.7
2 55-59   Rural Male 18.1
3 60-64   Rural Male 26.9
4 65-69   Rural Male 41.0
5 70-74   Rural Male 66.0
6 50-54 Rural Female  8.7
> str(data_frame)
'data.frame':   20 obs. of  3 variables:
 $ Var1: Factor w/ 5 levels "50-54","55-59",..: 1 2 3 4 5 1 2 3 4 5 ...
 $ Var2: Factor w/ 4 levels "Rural Male","Rural Female",..: 1 1 1 1 1 2
 $ Freq: num  11.7 18.1 26.9 41 66 8.7 11.7 20.3 30.9 54.3 ...
```

Vemos que tenemos dos variables de tipo factor, con 5 y 4 niveles. Estos niveles constituirán las diferentes variables que, más tarde, estudiaremos. Por último tenemos *Freq* que, como ya sabíamos, indica la frecuencia absoluta de cada uno de los sucesos que podemos estudiar (p.e., que sea hombre rural entre 50 y 54 años, que sea mujer urbana entre 60 y 64 años…). Sabemos que esta frecuencia es sobre 1000 habitantes. Es importante tener esto último en cuenta para saber con qué datos estamos trabajando.

Apartado (c)

Hemos de determinar las frecuencias marginales de las variables. Para ello, usamos la función *margin.table().*

```
> margin.table(VADeaths)
[1] 618.4
> v1 <- margin.table(VADeaths,1)    ##marginal grupos de edad
> v1
50-54 55-59 60-64 65-69 70-74
 44.2  67.7 103.5 161.6 241.4
> v2 <- margin.table(VADeaths,2)  ##marginal grupos de zonas
> v2
  Rural Male Rural Female   Urban Male Urban Female
       163.7        125.9        202.4        126.4
```

Con los números 1 y 2, indicamos que se nos muestre las frecuencias marginales de las filas y las columnas, respectivamente. Estas frecuencias son absolutas, para obtener las frecuencias relativas marginales, hemos de dividir entre el número total de observaciones (dado por *margin.table(VADeaths)).* Notamos que el total de observaciones no es 1000. Las marginales relativas que nos saldrán tienen un significado condicionado al total de observaciones. Es decir, las probabilidades que nos saldrán no son en relación a toda la población, son en relación a esos grupos de edad y de zona:

```
> margin.table(VADeaths,1)/margin.table(VADeaths) #frec. marginal relativa
     50-54      55-59      60-64      65-69      70-74
0.07147477 0.10947607 0.16736740 0.26131953 0.39036223
> margin.table(VADeaths,2)/margin.table(VADeaths)
  Rural Male Rural Female   Urban Male Urban Female
   0.2647154    0.2035899    0.3272962    0.2043984
```

Si queremos la probabilidad de mortalidad de dichos grupos de zona y de edad respecto a toda la población, tendríamos que dividir entre 1000:

```
> margin.table(VADeaths,1)/1000
  50-54   55-59   60-64   65-69   70-74
0.0442 0.0677 0.1035 0.1616 0.2414
```

Teniendo en cuenta la población total, las personas que tienen entre 60 y 64 años, p.e., tienen p=0.1 probabilidades de morir. Respecto a los grupos de zona no podemos decir mucho más, porque podemos intuir que hay personas que tienen una edad que no está entre 50 y 74 pero no podemos saber si los grupos de zona se mantienen constantes en toda la población o hay más. Si se mantienen constantes, entonces, habría que ver cuál es la frecuencia de cada una de las zonas en cada uno de los grupos de edad que faltan en la tabla. Por lo tanto, a falta de más información sobre la población general, no podemos extrapolar los datos de zona a datos de la población total.

También podemos extraer las frecuencias condicionadas usando la función *prop.table()*. Análogamente:

```
> prop.table(VADeaths,1)  #condicionada edad
      Rural Male Rural Female Urban Male Urban Female
50-54  0.2647059    0.1968326  0.3484163    0.1900452
55-59  0.2673560    0.1728213  0.3589365    0.2008863
60-64  0.2599034    0.1961353  0.3574879    0.1864734
65-69  0.2537129    0.1912129  0.3378713    0.2172030
70-74  0.2734051    0.2249379  0.2945319    0.2071251
```

Condicionando a cada grupo de edad. Es decir, la frecuencia relativa que tiene cada zona si condicionamos a cada uno de los grupos de edad.

```
> prop.table(VADeaths,2)  #condicionada zona
      Rural Male Rural Female Urban Male Urban Female
50-54 0.07147221   0.06910246 0.07608696    0.0664557
55-59 0.11056811   0.09293090 0.12005929    0.1075949
60-64 0.16432498   0.16123908 0.18280632    0.1526899
65-69 0.25045816   0.24543288 0.26976285    0.2776899
70-74 0.40317654   0.43129468 0.35128458    0.3955696
```

Condicionando a cada grupo de zona. Es decir, la frecuencia relativa que tiene cada grupo de edad si condicionamos a cada uno de los grupos de zona.

De hecho, podemos extraer algún ejemplo de frecuencia condicionada. Declaramos:

```
> v3 <- prop.table(VADeaths,1)
> v4 <- prop.table(VADeaths,2)
```

Como v3 y v4 son tablas, se pueden ver como matrices. Por tanto, podríamos extraer alguna fila o alguna columna. Por ejemplo si queremos saber la frecuencia condicionada a ser mujer urbana podemos ejecutar el siguiente código:

```
> v4[,2]
     50-54      55-59      60-64      65-69      70-74
0.06910246 0.09293090 0.16123908 0.24543288 0.43129468
```

Una comprobación sencilla que podríamos hacer para ver si tiene sentido es sumar las probabilidades y ver si el sumatorio es igual a 1.

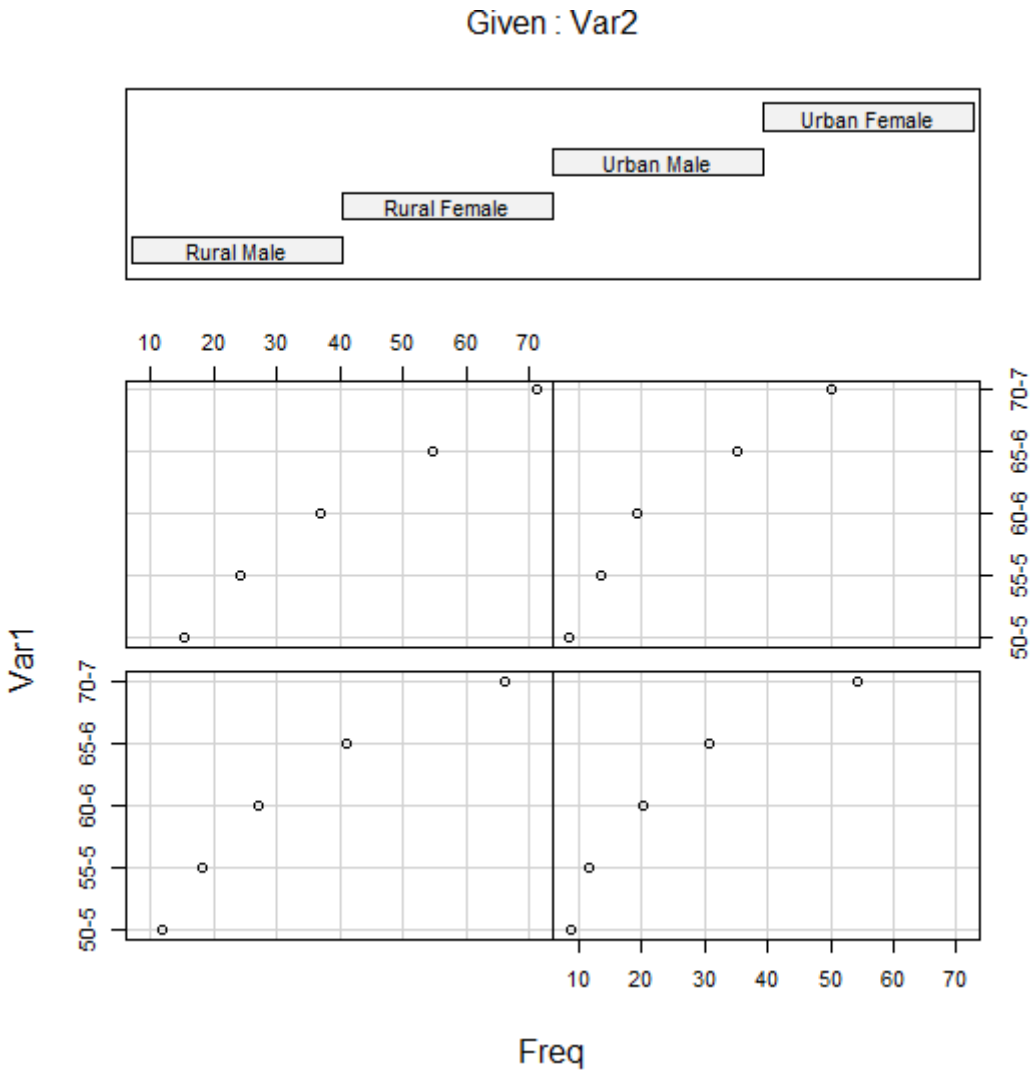
Otro ejemplo, podríamos ver las probabilidades condicionadas al grupo de edad 60-64:

```
> v3[3,]
  Rural Male Rural Female   Urban Male Urban Female
   0.2599034    0.1961353    0.3574879    0.1864734
```

Apartado (d)

Para hacer los histogramas, primero vamos a ver cómo se distribuyen los datos según las tres variables (grupos de edad, los de zona y la frecuencia):

```
> attach(data_frame)
The following objects are masked from data_frame (pos = 3):

    Freq, Var1, Var2

> coplot(Var1 ~ Freq | Var2, data = data_frame,
+          main = "Gráfico de las 3 variables",
+          ylab = "Var1")
> detach(data_frame)
```

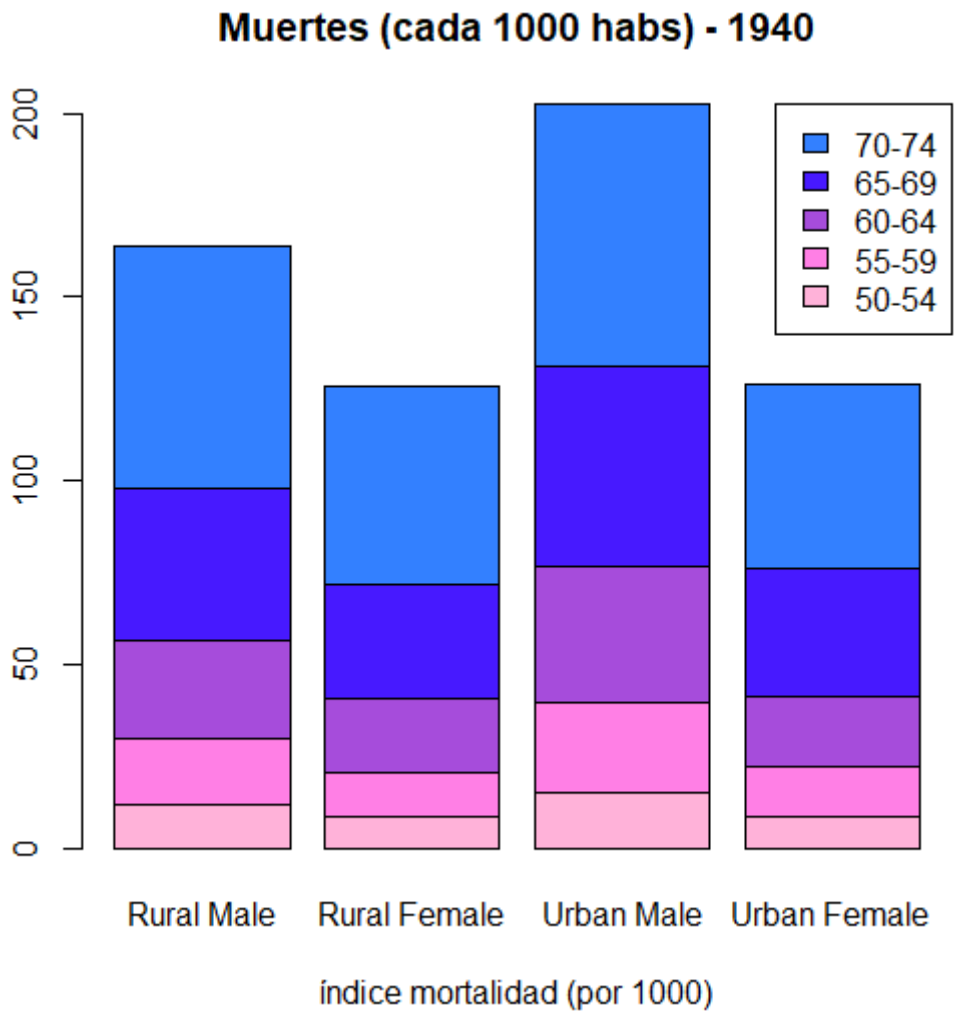


De izquierda a derecha. Los de arriba corresponden a *Urban Male* y *Urban female.* Los de abajo corresponden a *Rural male* y *Rural female*.

Veamos ahora, los gráficos de barras correspondientes a las marginales. Para ello usamos *barplot()* que, precisamente, trabaja con tablas.

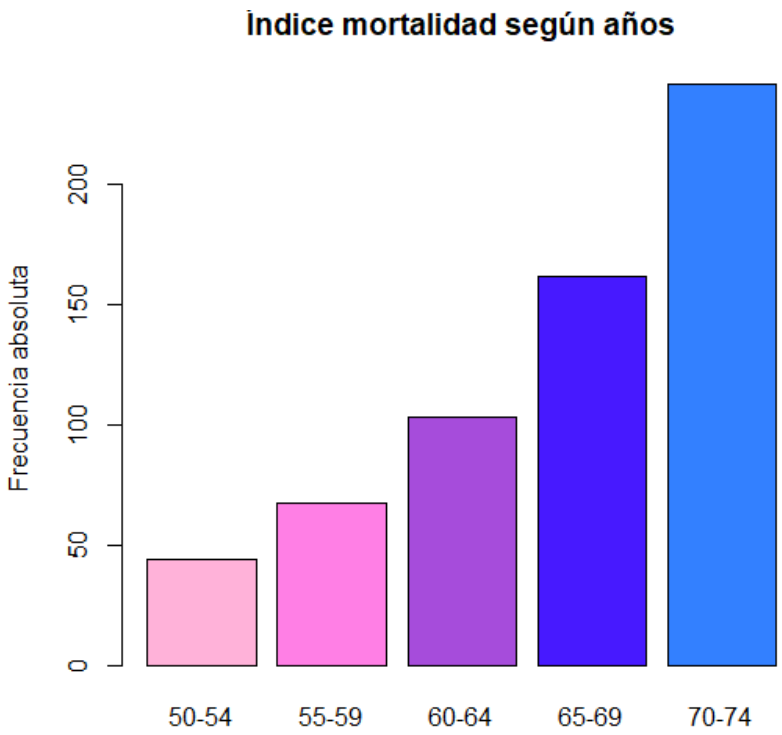
De nuevo, hacemos uso del *data frame* para hacer más fácil el comando. Para la leyenda, indicamos *legend.text=levels(Var1)* para indicarle que en la leyenda habrá que explicitar los grupos de edad.

```
> barplot(VADeaths, main="Muertes (cada 1000 habs) - 1940",
+          xlab="índice mortalidad (por 1000)", col=colores,
+ legend.text=levels(Var1),args.legend=list(x="topright"))
```

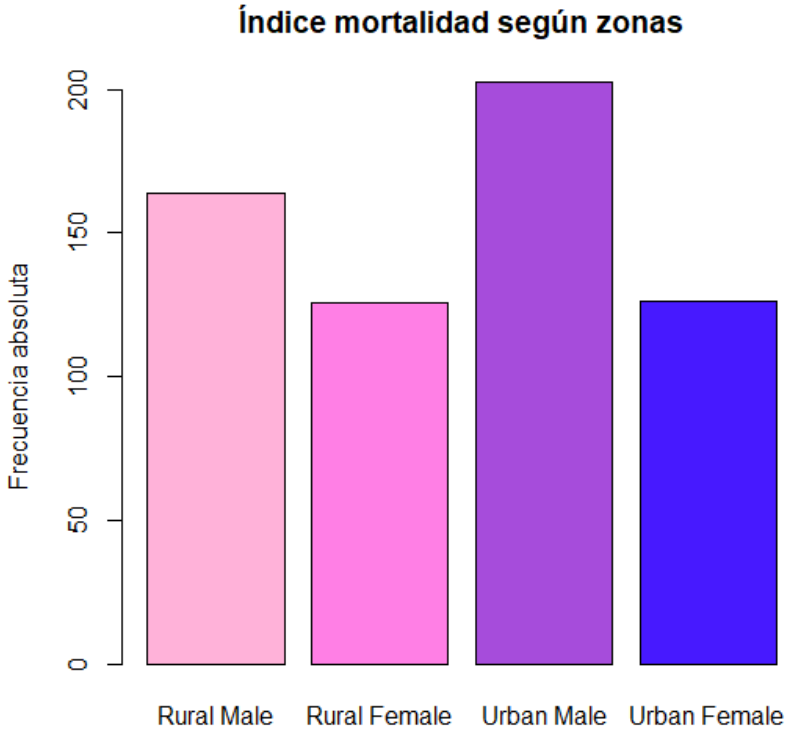


Hacemos ahora los diagramas de barras para las frecuencias marginales. Estos gráficos nos van a aportar una información muy valiosa y que se puede entender a la perfección gráficamente: cuál es el índice de mortalidad (por 1000) según los grupos de edad y según las zonas. Como podremos observar, la mortalidad crece cuando nos encontramos entre personas de mayor edad (que no contradice nuestra intuición social adquirida). Por zonas, podemos observar que los hombres de ciudad tienen mayor mortalidad que los de zonas rurales. Sin embargo, entre mujeres no podemos observar una diferencia notoria. Con las frecuencias marginales hemos podido intuir a simple vista la mortalidad en relación a la edad y la mortalidad en relación a las zonas y al género de las personas (ya que, al hacer la división entre hombres y mujeres, esta información también puede ser extraída).

```
> barplot(vl, col=colores, ylab="Frecuencia absoluta",
+ main="Índice mortalidad según años")
```




```
> barplot(v2, col=colores, ylab="Frecuencia absoluta",
+ main="Índice mortalidad según zonas")
```



Estos gráficos podríamos, también, exportarlos a un PDF desde R de la siguiente forma:

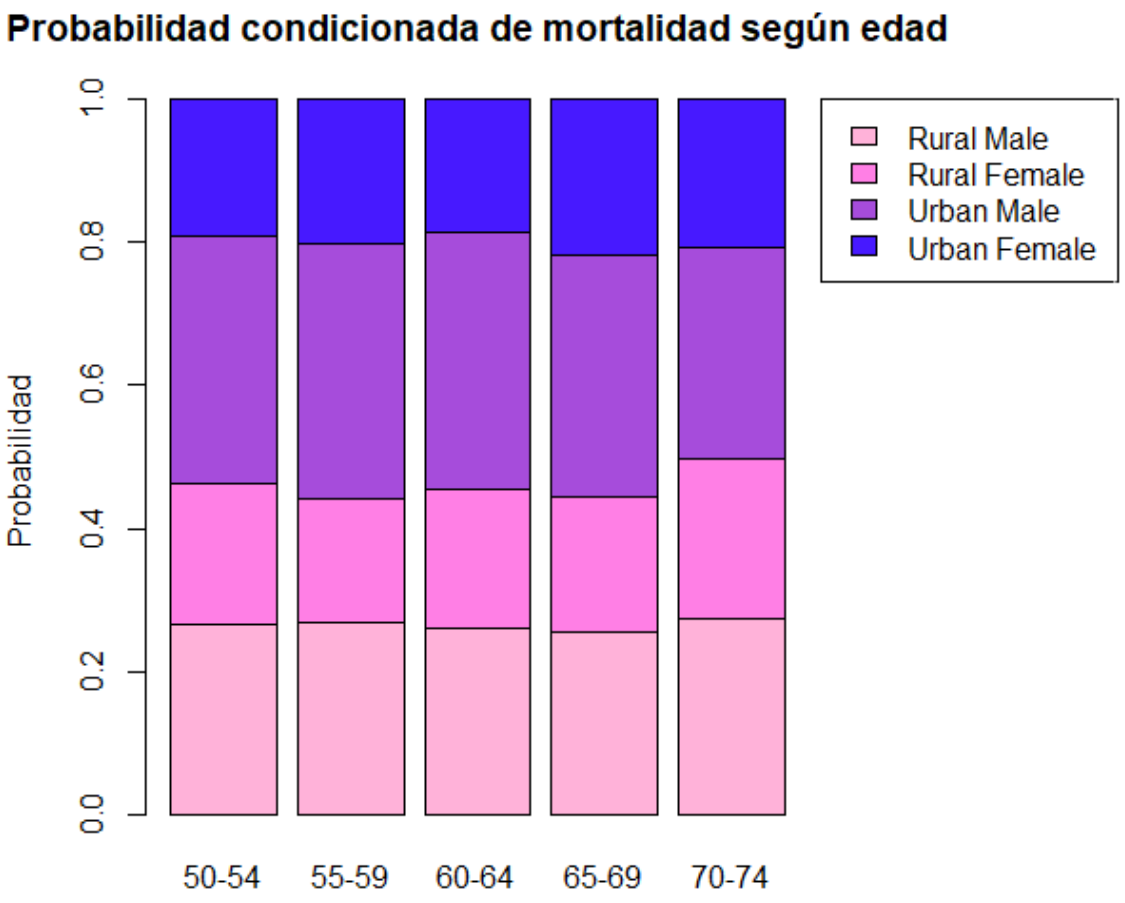
```
> pdf(file="GraficoTarea2.pdf")
> par(mfrow=c(2,1))
> barplot(v1, col=colores, ylab="Frecuencia absoluta",
+ main="Índice mortalidad según años")
> barplot(v2, col=colores, ylab="Frecuencia absoluta",
+ main="Índice mortalidad según zonas")
> dev.off()
windows
      2
```

Nota: he añadido el PDF resultante en la carpeta de la tarea.

Además los gráficos de las probabilidades condicionadas son:

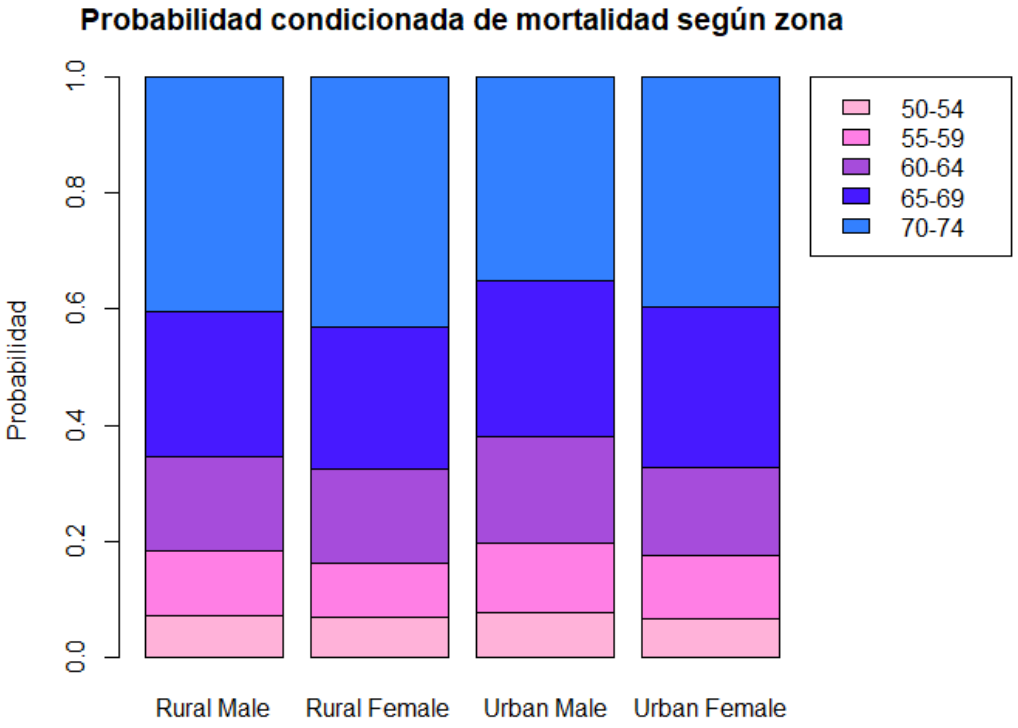
```
> par(mar=c(5,4,4,9),xpd=TRUE)
> barplot(t(v3), col=colores, ylab="Probabilidad",
+ main="Probabilidad condicionada de mortalidad según edad")
>
> legend("topright", inset=c(-0.5,0),legend=levels(Var2), fill=colores)
```

Usamos la transpuesta de v3 ya que el gráfico se hace por columnas y queríamos que fuese por filas. Así que usamos la transpuesta.



Podemos ver la probabilidad de que sea de cada una de las zonas y los sexos, condicionada a cada grupo de edad.

Análogamente, condicionando a grupos de zona y sexo:




```
> par(mar=c(5,4,4,9),xpd=TRUE)
> barplot(v4, col=colores, ylab="Probabilidad",
+ main="Probabilidad condicionada de mortalidad según zona")
>
> legend("topright", inset=c(-0.3,0), fill=colores, legend=levels(Var1))
```

EJERCICIO 3

Apartado (a)

En este ejercicio trabajaremos con los datos *thuesen* del libro *ISwR*. Estos datos son un *data frame* que contiene varios registros de dos variables: glucosa en sangre y velocidad ventricular. Contiene 24 registros y son datos de pacientes con diabetes tipo 1.

```
> library("ISwR")
> data(thuesen)
> head(thuesen)
  blood.glucose short.velocity
1          15.3           1.76
2          10.8           1.34
3           8.1           1.27
4          19.5           1.47
5           7.2           1.27
6           5.3           1.49
> str(thuesen)
'data.frame':   24 obs. of  2 variables:
 $ blood.glucose : num  15.3 10.8 8.1 19.5 7.2 5.3 9.3 11.1 7.5 12.2 ...
 $ short.velocity: num  1.76 1.34 1.27 1.47 1.27 1.49 1.31 1.09 1.18 1.22 ...
```

Podemos observar, efectivamente, que el *data frame* está constituido por dos variables. Usamos la función *attach()* para que sea más rápido el manejo de datos.

Description

The thuesen data frame has 24 rows and 2 columns. It contains ventricular shortening velocity and blood glucose for type 1 diabetic patients.

Apartado (b)

Vemos la relación lineal entre las dos variables. Para ello usamos la función *lm().*

```
> r <- lm(short.velocity ~ blood.glucose) ##Recta
> r

Call:
lm(formula = short.velocity ~ blood.glucose)

Coefficients:
  (Intercept)  blood.glucose
      1.09781        0.02196
```

short.velocity= 1.09781 + 0.02196*blood.glucose

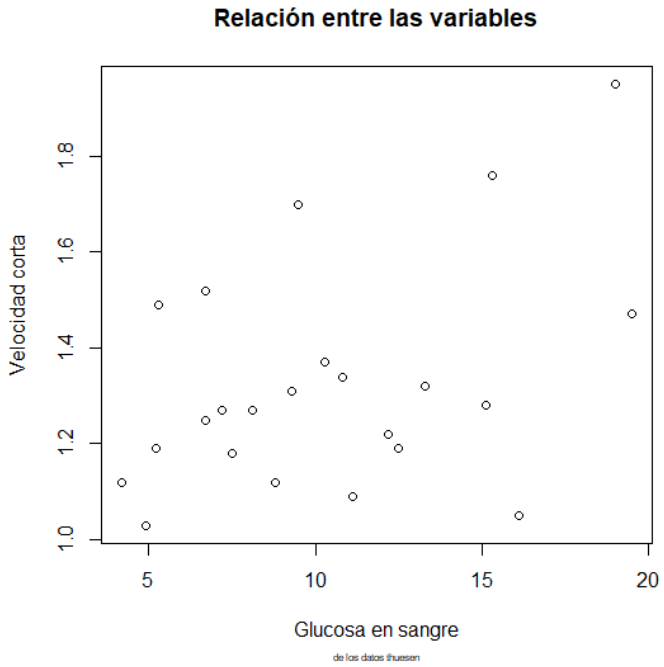
i.e, por cada unidad de glucosa en sangre, se puede esperar que la velocidad corta aumente un promedio de 0.02 unidades.

Apartado (c)

Para determinar la bondad del ajuste, usaremos la función *summary()* que nos aporta algunos datos estadísticos importantes de la recta de regresión r.

```
> summary(r)

Call:
lm(formula = short.velocity ~ blood.glucose)

Residuals:
     Min       1Q   Median       3Q      Max
-0.40141 -0.14760 -0.02202  0.03001  0.43490

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.09781    0.11748   9.345 6.26e-09 ***
blood.glucose  0.02196    0.01045   2.101   0.0479 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 

Residual standard error: 0.2167 on 21 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.1737,    Adjusted R-squared:  0.1343
F-statistic: 4.414 on 1 and 21 DF,  p-value: 0.0479
```

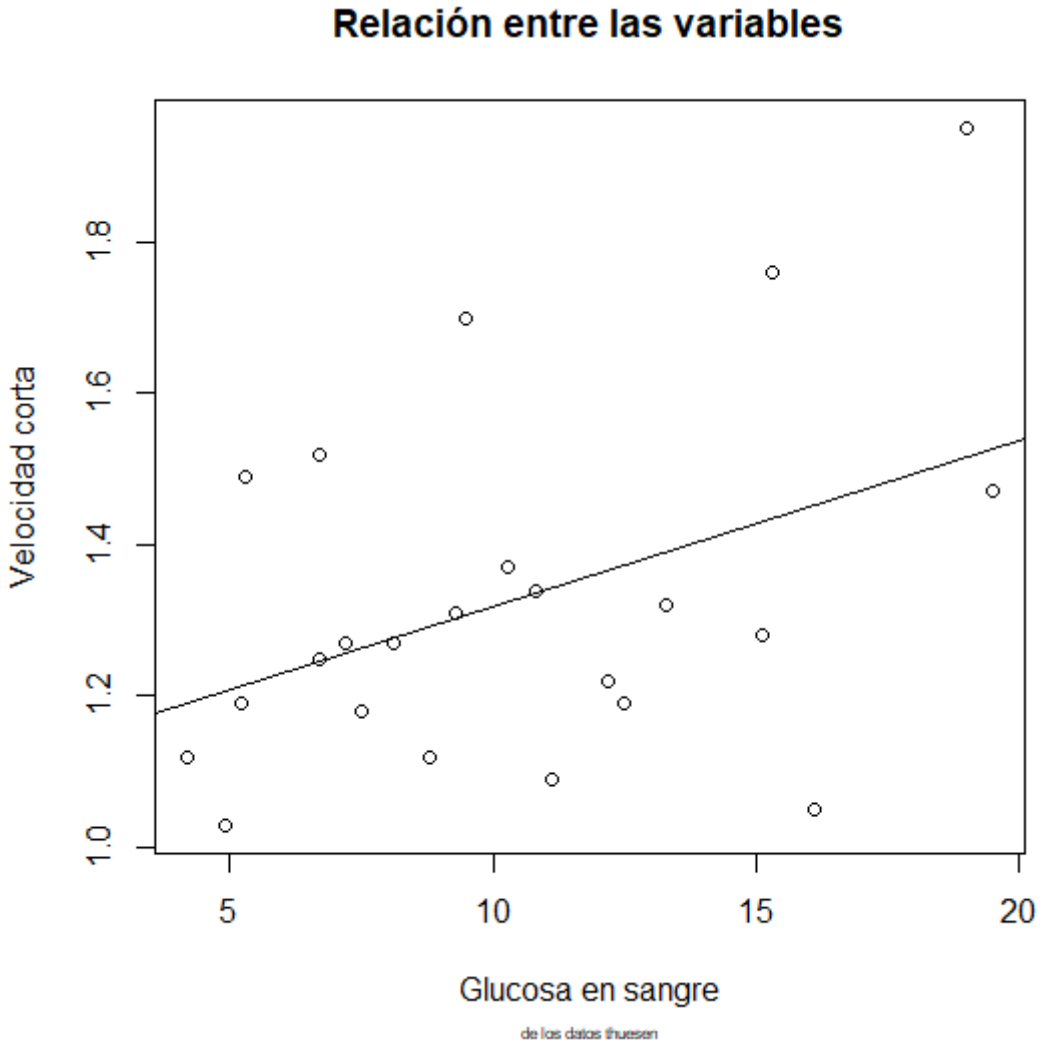
Vemos que R-cuadrado es 0.1737 por lo que podemos afirmar que no existe relación lineal entre las dos variables. Para verlo un poco mejor, veamos el gráfico de dispersión:



Aquí vemos que, efectivamente, falla la linealidad. En el gráfico de dispersión ya se observa que no existe ninguna relación lineal entre ambas variables. Reafirmamos, por lo tanto, el rechazo al modelo estimado. Podríamos seguir estudiando la regresión lineal y ver cómo se distribuyen los residuos (si se distribuyen según una distribución normal con media cero o no), si la varianza de los residuos es constante, cómo son los valores atípicos… Sin embargo, ya sabemos que no existe ningún atisbo de relación lineal entre la glucosa en sangre y la velocidad corta, por lo tanto, no tiene sentido estudiar las propiedades mencionadas más allá de saber cómo se comportan los datos en relación a la recta.

Vemos el gráfico con la recta de regresión:

```
> abline(r)
```




```
> detach(thuesen)
```

Bibliografía

[1] Román M. Y. Gráficos en el entorno de computación estadística de R.

[2] Román M. Y. ECI_sesion5_vid.mp4 (2020) Universidad de granada. GRANADA.

[3] R documentation https://www.rdocumentation.org/

[4] H. Wickham, G. Grolemund. R for Data science (2017). Data visualisation. https://r4ds.had.co.nz/data-visualisation.html

[5] H. Wickham, G. Grolemund. R for Data science (2017). Graphics for communication. https://r4ds.had.co.nz/graphics-for-communication.html

[6] S. Prabhakaran. R statistics (2016-2017). http://r-statistics.co/Loess-Regression-With-R.html#:~:text=r%2Dstatistics.co%20by%20Selva%20Prabhakaran&text=Loess%20Regression%20is%20the%20most,for%20smoothing%20any%20numerical%20vector

[7] Gil M. C. Métodos de regresión no lineal (2018). Regresión no lineal. https://rpubs.com/Cristina_Gil/Regr_no_lineal

[8] Hernández F., Mazo M. Modelos de regresión con R (2020). Modelos polinomiales. https://fhernanb.github.io/libro_regresion/mod-poli.html

[9] Coll V., Pérez J. P. Importar (y exportar) datos en R (2017). Intro. https://www.uv.es/pjperez/curso_R/tt_3_cargar_datos_v4.html#1_intro