



# Gráficos avanzados con R

CNE/ISCIII

# Estructura del curso

1. Gráficos básicos
2. Gráficos avanzados
3. Mapas
4. Informes automatizados

# ¿Porqué usar gráficos?

“Una imagen vale más que mil palabras” (Napoleón)

## Motivos:

- Potente herramienta de síntesis de la información estadística
- Agiliza la exploración, modelización y comunicación en el análisis de datos

## Objetivos del curso:

- Manejar el paquete `ggplot2` de R que implementa una “gramática” de diseño de los gráficos
- Controlar el proceso de elaboración de gráficos, que va desde la preparación de los datos hasta la publicación de los resultados del análisis

# Gráficos básicos

# Una función si no hay tiempo para pensar: `qplot`

Instalación del paquete `ggplot2` (“gg” para “Grammar of Graphics”).

```
#install.packages("ggplot2")
library(ggplot2) #carga la librería ggplot2
```

Empezaremos con la función básica `qplot` (“quick plot”) de este paquete:

```
qplot(x, y=NULL, data, geom="auto")
```

- `x` : valores en el eje de abscisas.
- `y` : valores en el eje de ordenadas (opcional).
- `data` : `data.frame` de donde salen los datos (opcional).
- `geom` : elementos gráficos o geometrías (“point”, “line”, “bar”, ..). Por defecto, “point” si `y` viene especificado, e “histogram” si sólo se especifica `x`.
- ... y otros argumentos relacionados con los ejes del gráfico (`xlab`, `ylab`: etiquetas de los ejes; `xlim`, `ylim`: límites en los ejes de `x` e `y`; `log`: eje en escala log, “`x`”, “`y`” o bien “`xy`”).

# Distribución de una variable continua

## Un ejemplo

Para ilustrar la descripción gráfica de la distribución de una variable numérica, se utiliza la base de datos `msleep` que contiene información sobre el tiempo de sueño (en horas) de mamíferos:

```
str(msleep) # ?msleep para más detalles
```

```
## # tibble [83 x 11] (S3: tbl_df/tbl/data.frame)
## $ name      : chr [1:83] "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-tailed shrew" ...
## $ genus     : chr [1:83] "Acinonyx" "Aotus" "Aplopontia" "Blarina" ...
## $ vore       : chr [1:83] "carni" "omni" "herbi" "omni" ...
## $ order      : chr [1:83] "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...
## $ conservation: chr [1:83] "lc" NA "nt" "lc" ...
## $ sleep_total : num [1:83] 12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...
## $ sleep_rem   : num [1:83] NA 1.8 2.4 2.3 0.7 2.2 1.4 NA 2.9 NA ...
## $ sleep_cycle : num [1:83] NA NA NA 0.133 0.667 ...
## $ awake       : num [1:83] 11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...
## $ brainwt     : num [1:83] NA 0.0155 NA 0.00029 0.423 NA NA NA 0.07 0.0982 ...
## $ bodywt      : num [1:83] 50 0.48 1.35 0.019 600 ...
```

# Histograma

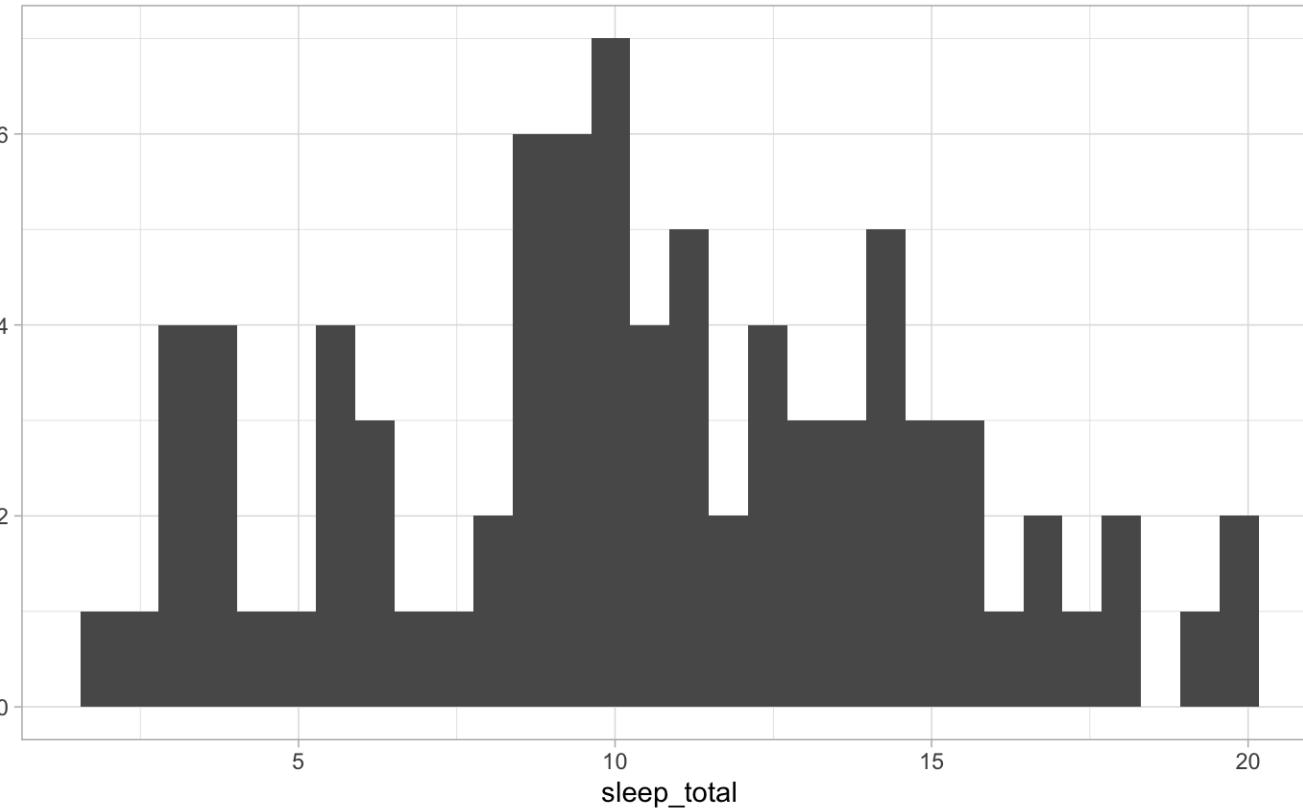
A mano

```
#summary(msleep$sleep_total) #tiempo total de sueño (en horas)
stem(msleep$sleep_total)
```

```
##
##      The decimal point is at the |
##
##      0 | 9
##      2 | 79013589
##      4 | 0423346
##      6 | 23307
##      8 | 03446779114456788
##     10 | 01113346900135
##     12 | 15555880578
##     14 | 234456996889
##     16 | 604
##     18 | 01479
```

... y con qplot

```
qplot(sleep_total, data=msleep) #histograma
```

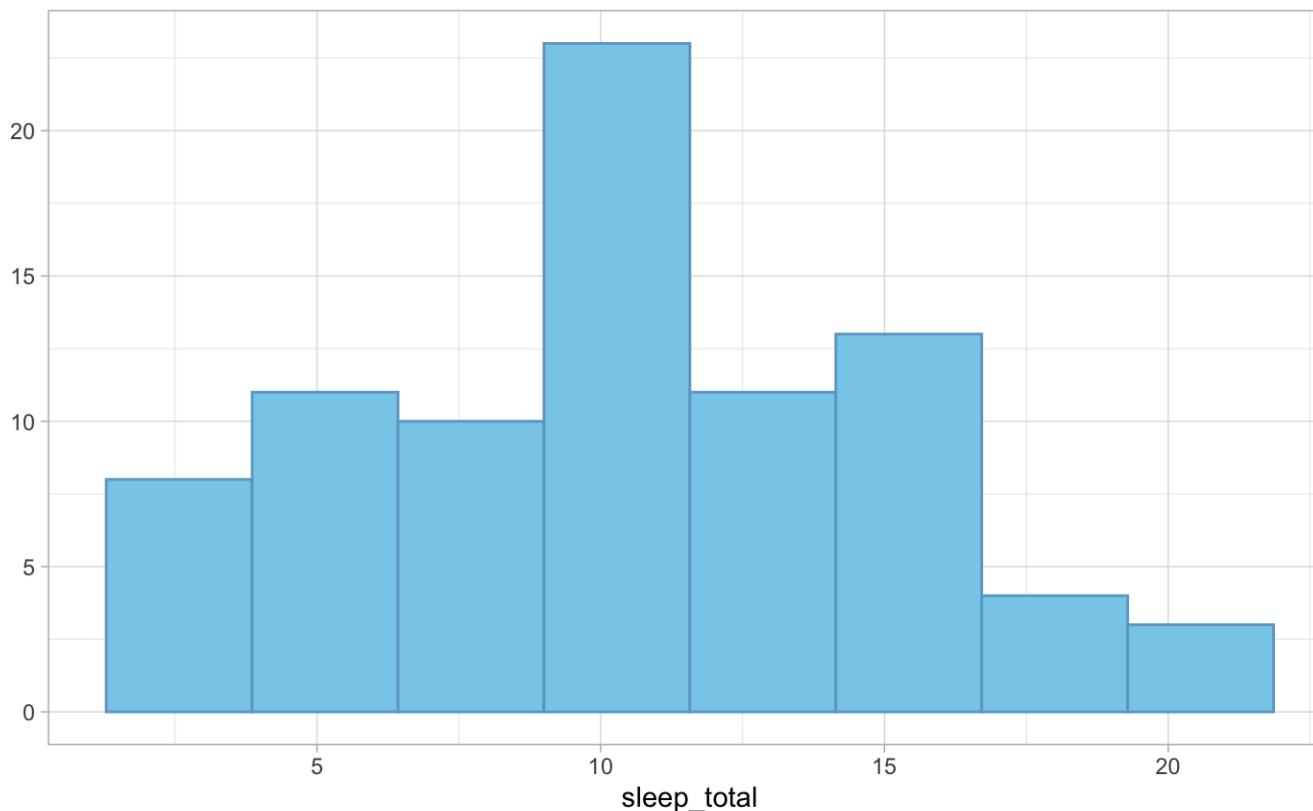


La altura de cada barra en el histograma es proporcional a la frecuencia de datos que caen en el intervalo correspondiente. Por defecto, el número de barras es igual al valor arbitrario `bins=30`.

## Número de intervalos

Otra alternativa consiste en elegir un número  $k$  de barras en función del tamaño muestral  $n$ , como por ejemplo, el criterio de Sturges ( $k = 1 + \log_2(n)$ ) o el criterio de Rule ( $k = 2n^{1/3}$ ).

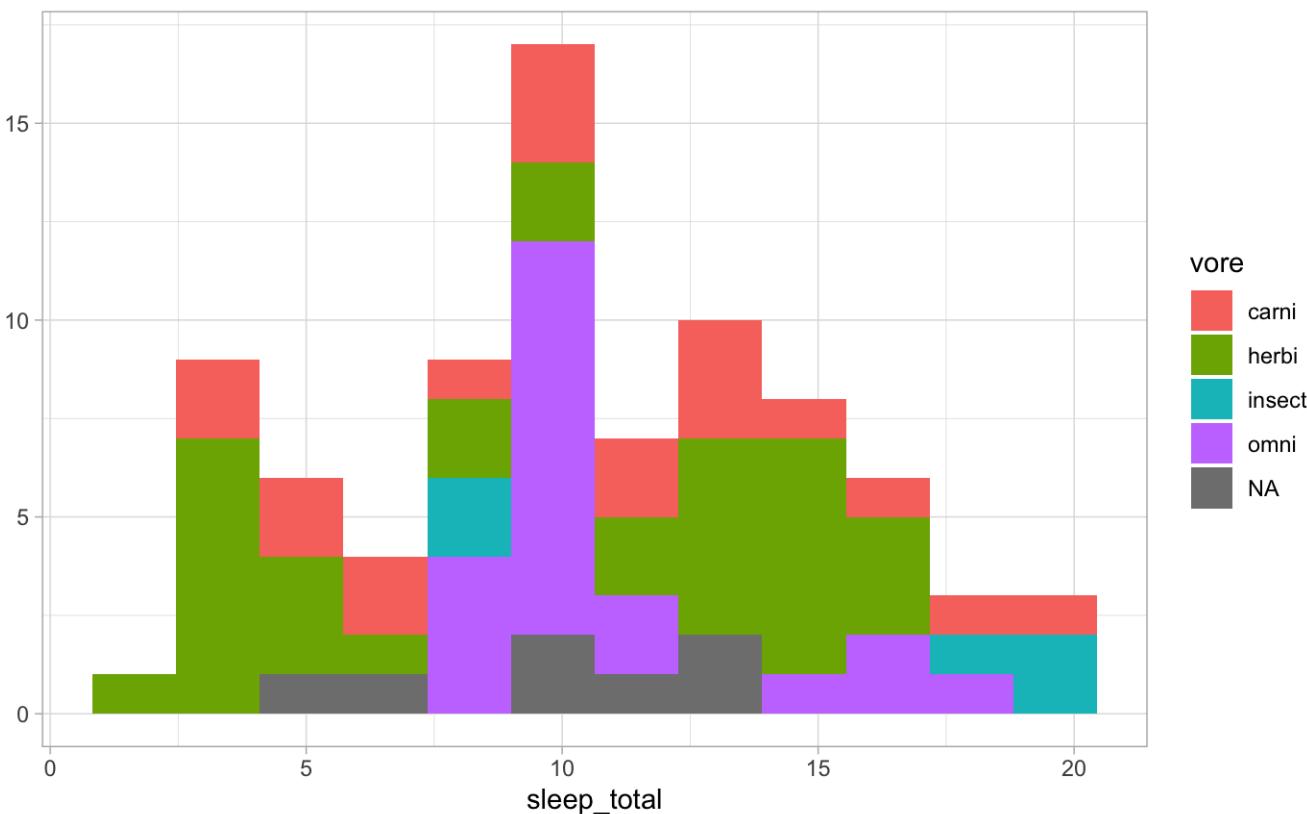
```
qplot(sleep_total, data=msleep, bins=8, color=I("skyblue3"), fill=I("skyblue")) #con criterio de Rule
```



## Una dimensión más

El argumento `fill` controla el color de relleno de las barras y el argumento `color` el color del borde. Para especificar un color concreto se utiliza la función `I()`. Si el color varía con otra variable `z`, se especifica esta dependencia escribiendo `fill=z`.

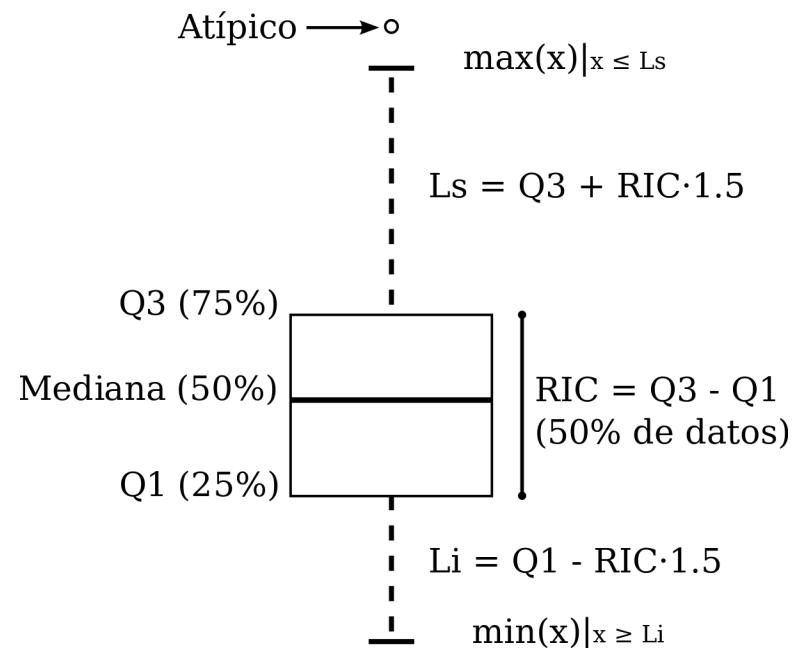
```
qplot(sleep_total, data=msleep, bins=12, fill=vore) #distribución del tiempo de sueño según dieta del mamífero.
```



# Diagrama de caja (boxplot)

Describe la distribución de una variable numérica mediante una caja y unos segmentos que acotan las regiones donde tiene el grueso de sus valores.

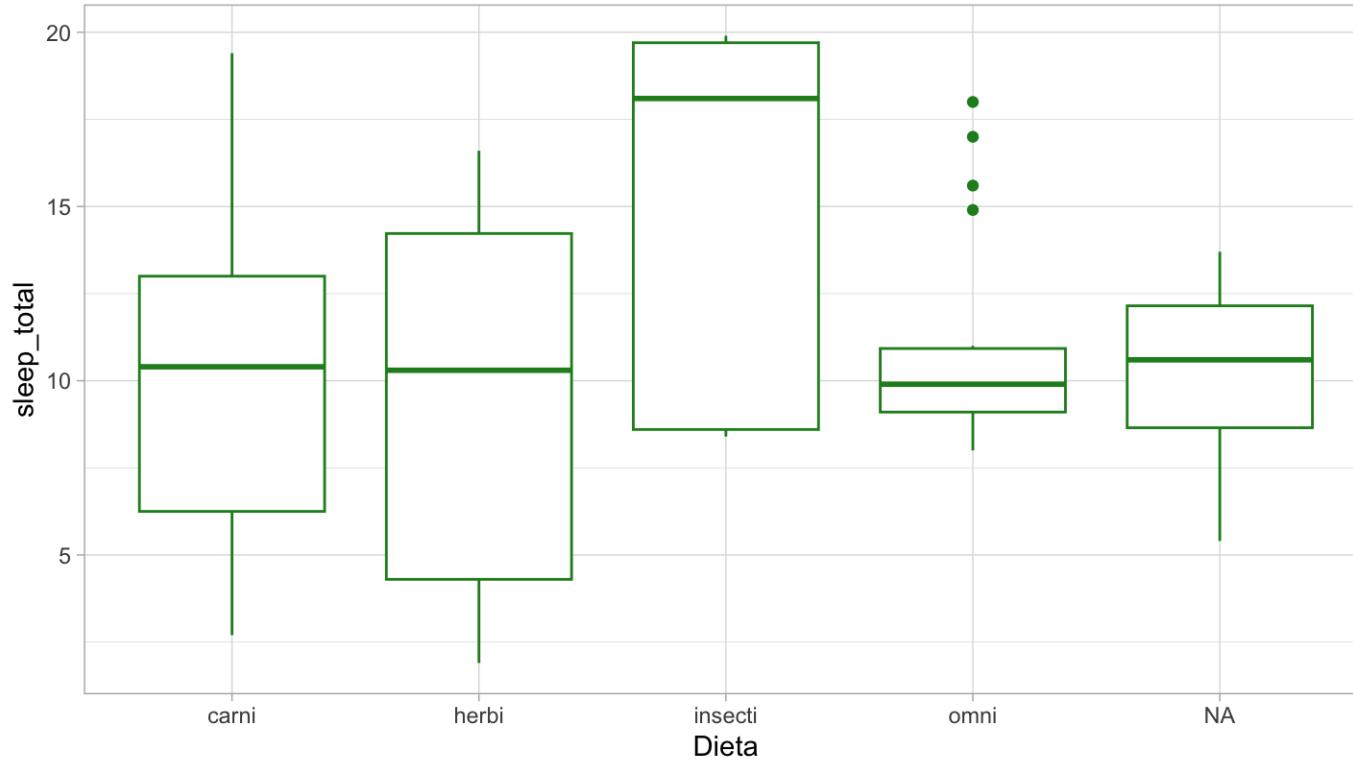
- Menos fina que el histograma pero más robusta.
- Adecuada para representar dependencia con otra variable (categórica).



# Una dimensión más

```
qplot(vore,sleep_total,data=msleep,geom="boxplot",xlab="Dieta",color=I("forestgreen"))
```

Distribución del tiempo de sueño según la dieta del mamífero

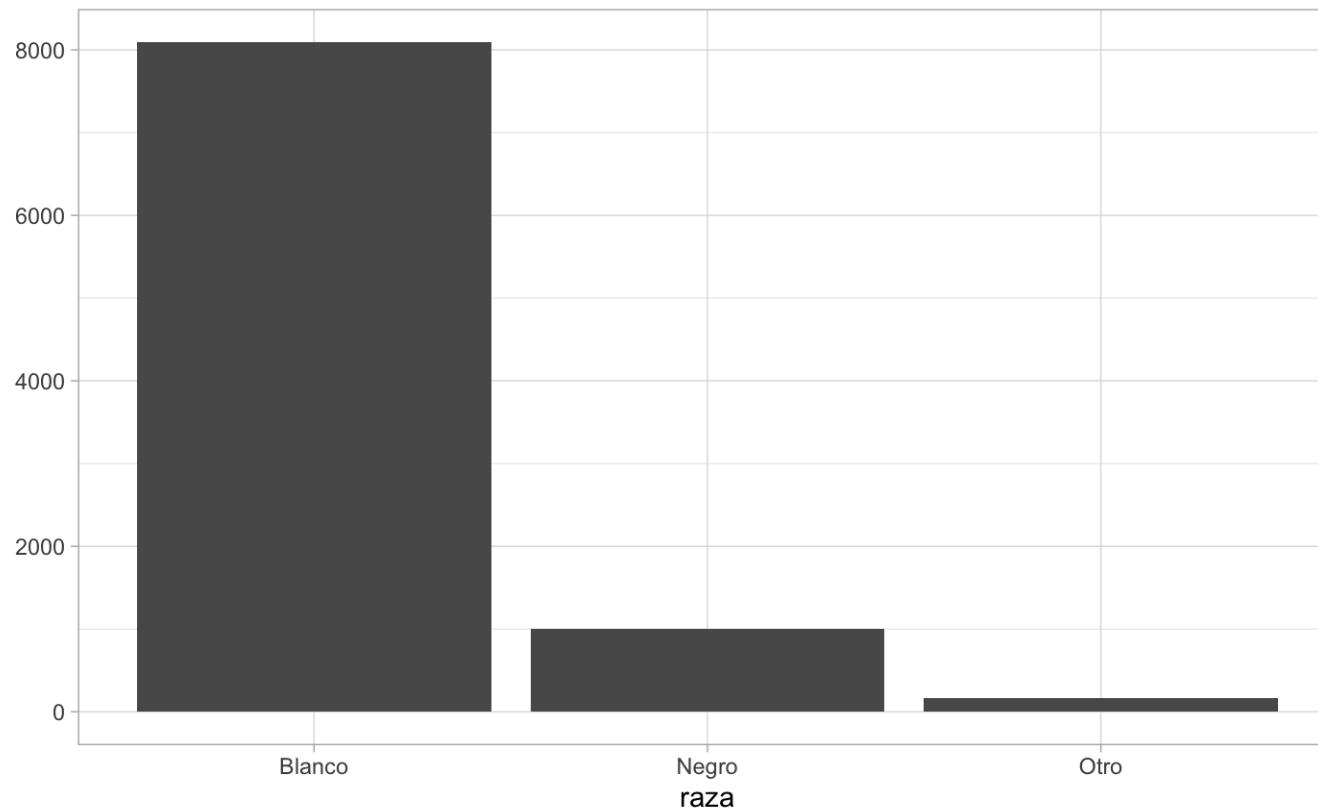


**Ejercicio:** Cargar la base de datos de la encuesta nacional americana nhs y representar la distribución del índice de masa corporal (imc) según el sexo y la raza.

# Diagrama de barras

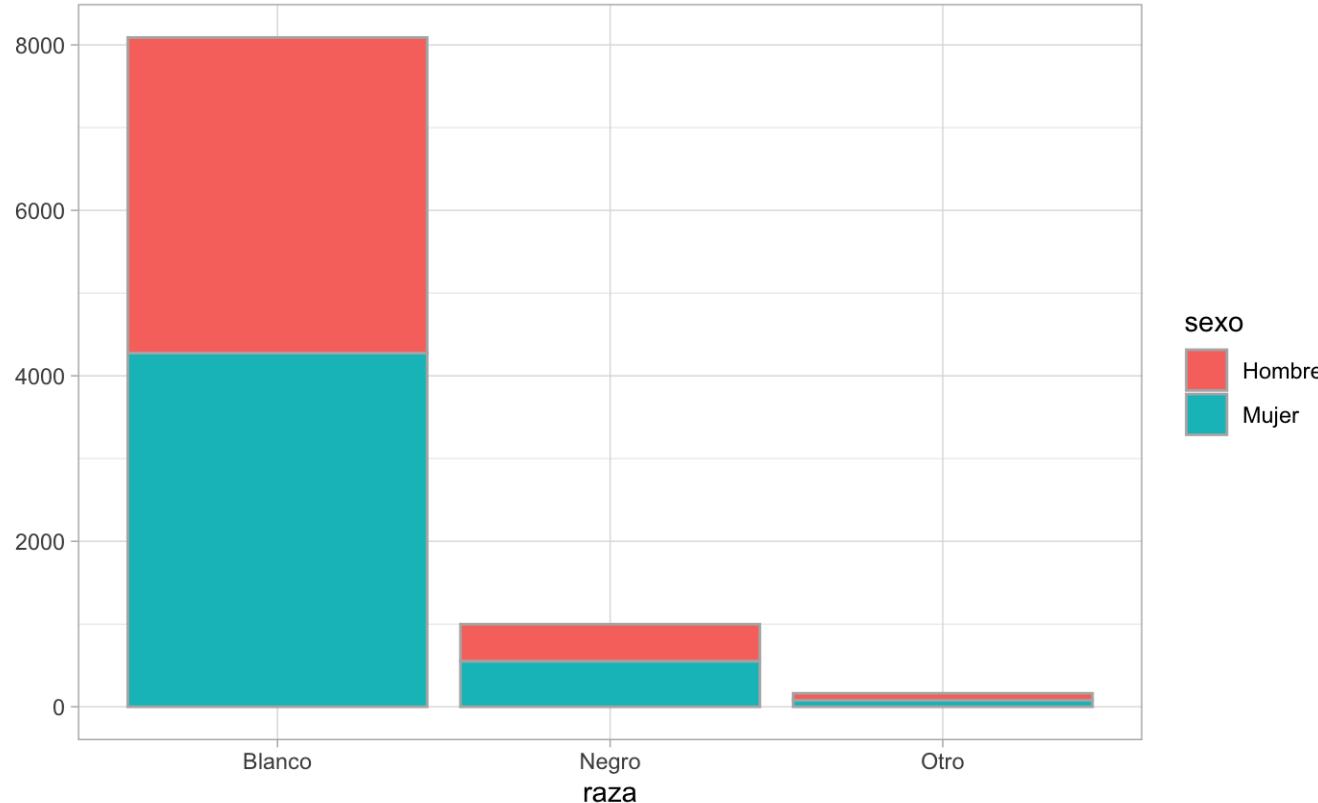
Los diagramas de barras permiten representar la distribución de una variable categórica. En esta representación, cada categoría viene representada por una barra cuya altura es proporcional a su frecuencia en la muestra.

```
qplot(raza,data=nhs) #Distribución de las razas en la muestra de la encuesta americana
```



## Una dimensión más

```
qplot(raza,data=nhs,fill=sexo,color=I("grey70")) #Distribución del sexo segun la raza
```

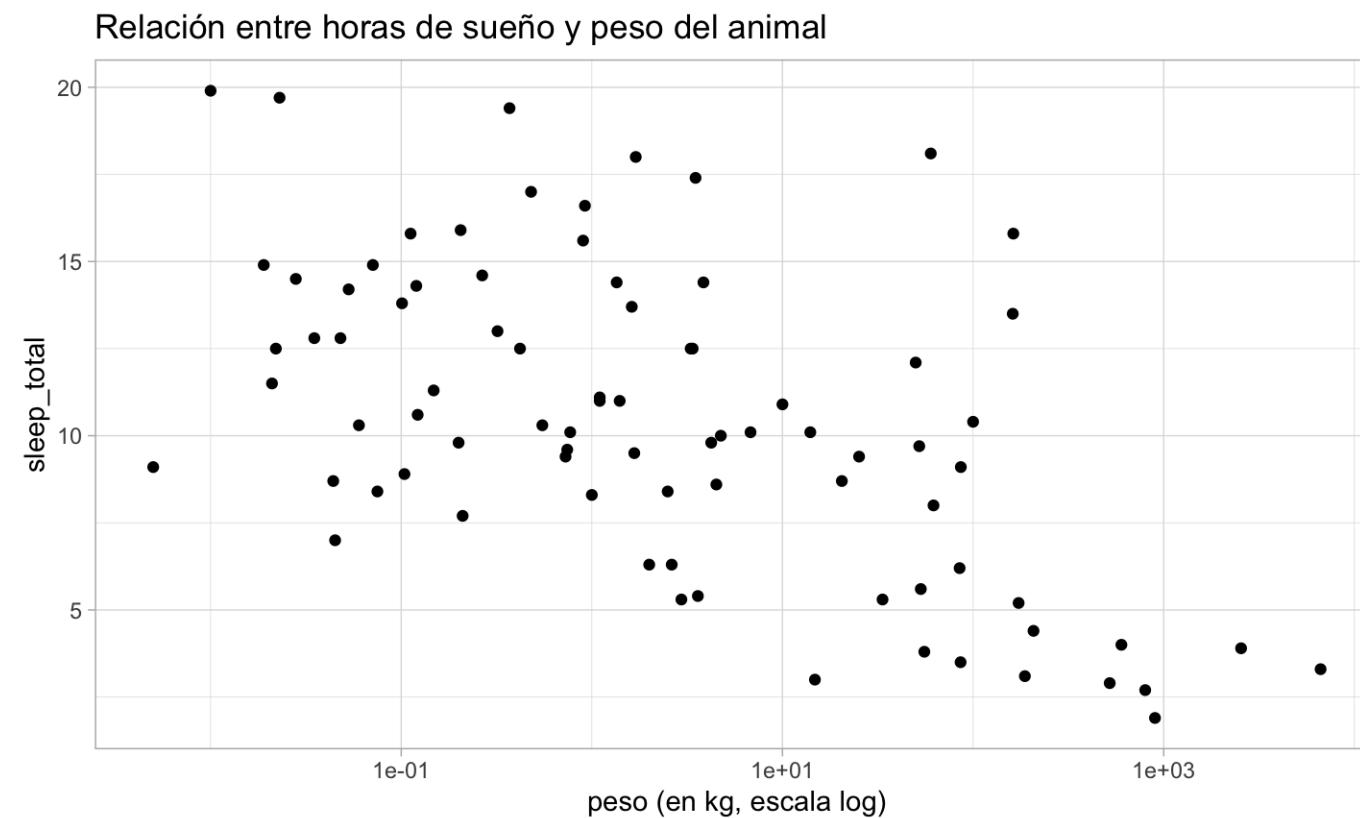


Con el argumento `fill` se puede ver como varia la distribución de una variable respecto a otra (aquí el sexo según la raza). El gráfico obtenido resulta poco claro y veremos más adelante como mejorarlo.

# Relación entre dos variables cuantitativas

## Diagrama de dispersión

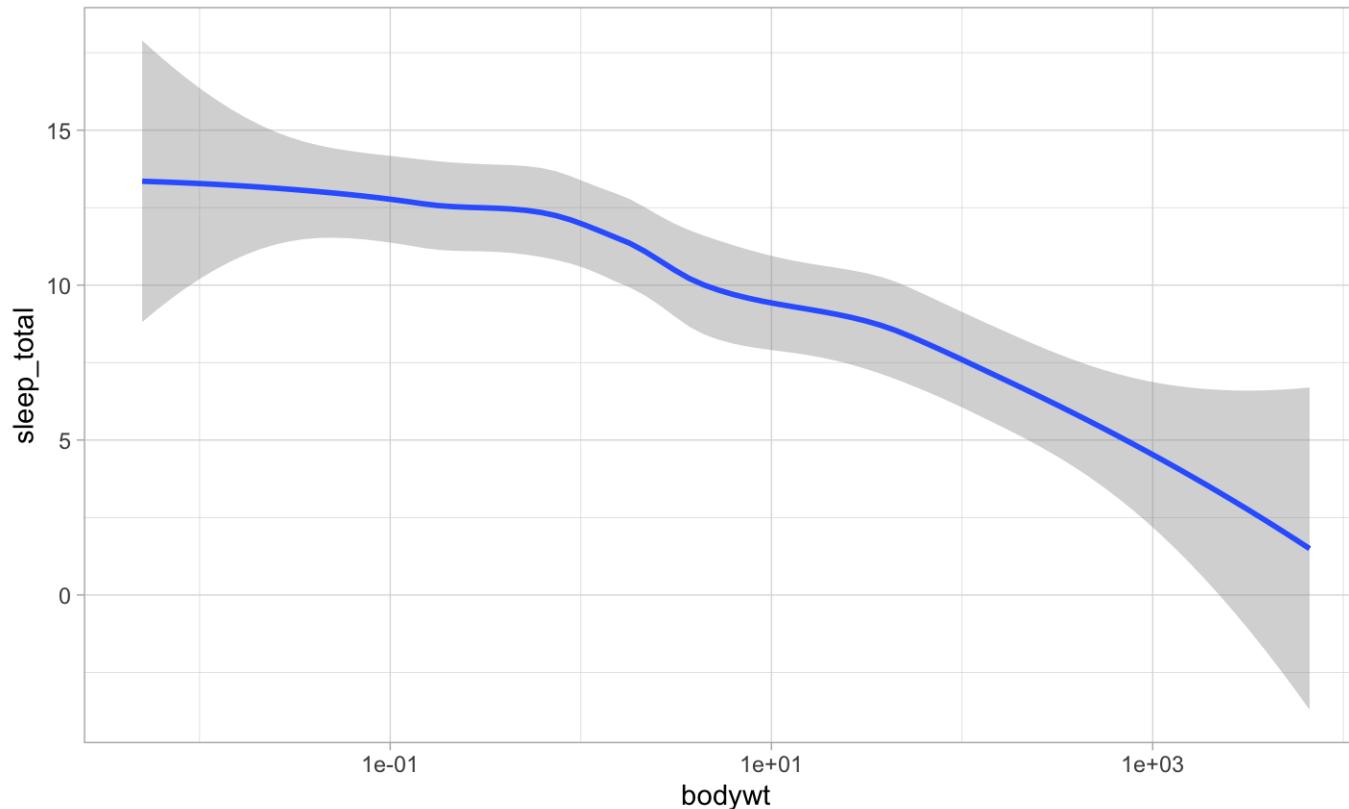
```
qplot(bodywt,sleep_total,data=msleep,xlab="peso (en kg, escala log)",log="x")
```



## Ajuste

Para apreciar mejor la tendencia en la nube de puntos, se puede ajustar una curva (“smooth”):

```
qplot(bodywt, sleep_total, data=msleep, log="x", geom="smooth")
# utilizar el argumento method="lm" para ajustar una recta
```

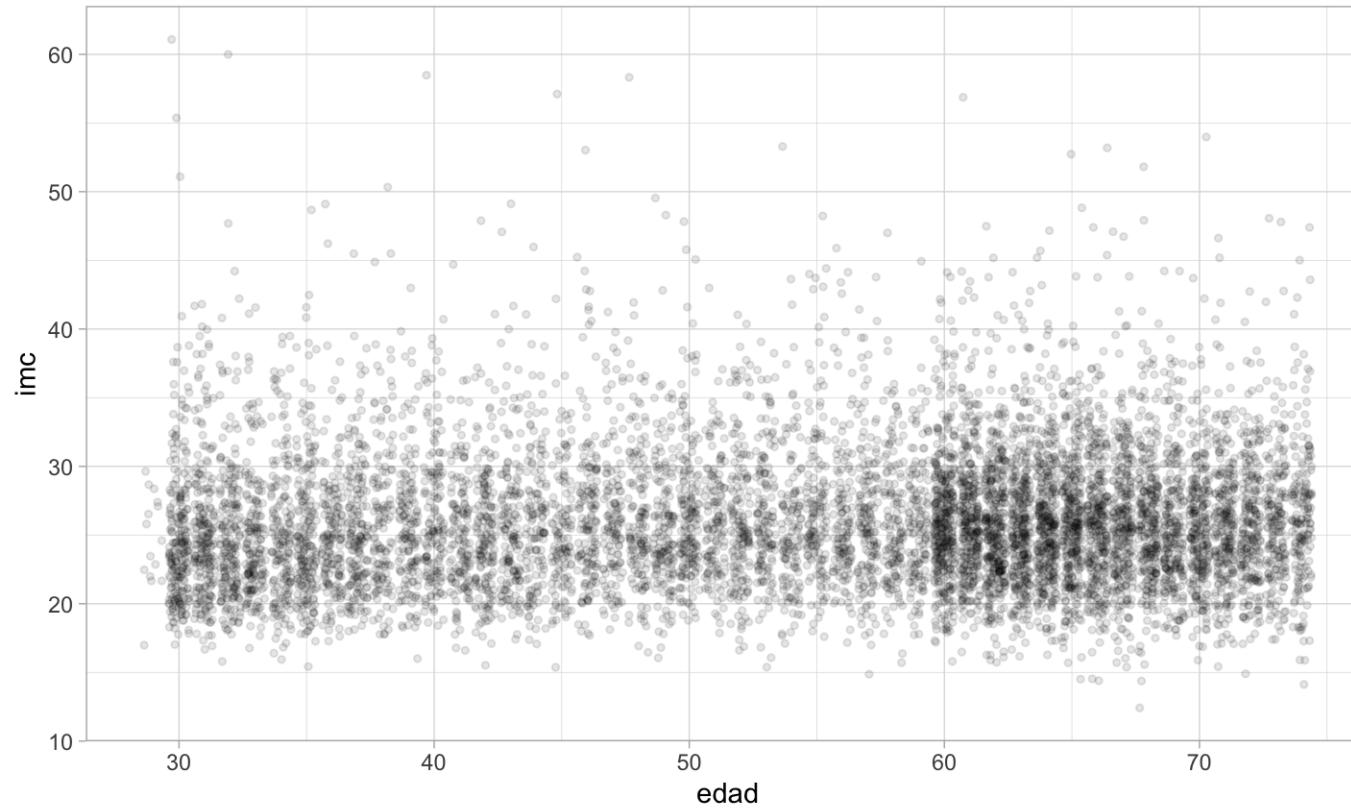


**Ejercicio:** Representar la tendencia de la esperanza de vida por continentes y en España, Grecia y Polonia, utilizando la base de datos gapminder.

# Problema de solapamiento

El solapamiento de puntos puede ser minimizado insertando algo de ruido en los datos (`geom="jitter"`), reduciendo el tamaño de los puntos (`size`) o recurriendo a la transparencia (`alpha`):

```
qplot(edad, imc, data=nhs, alpha=I(.1), size=I(1), geom="jitter")
```



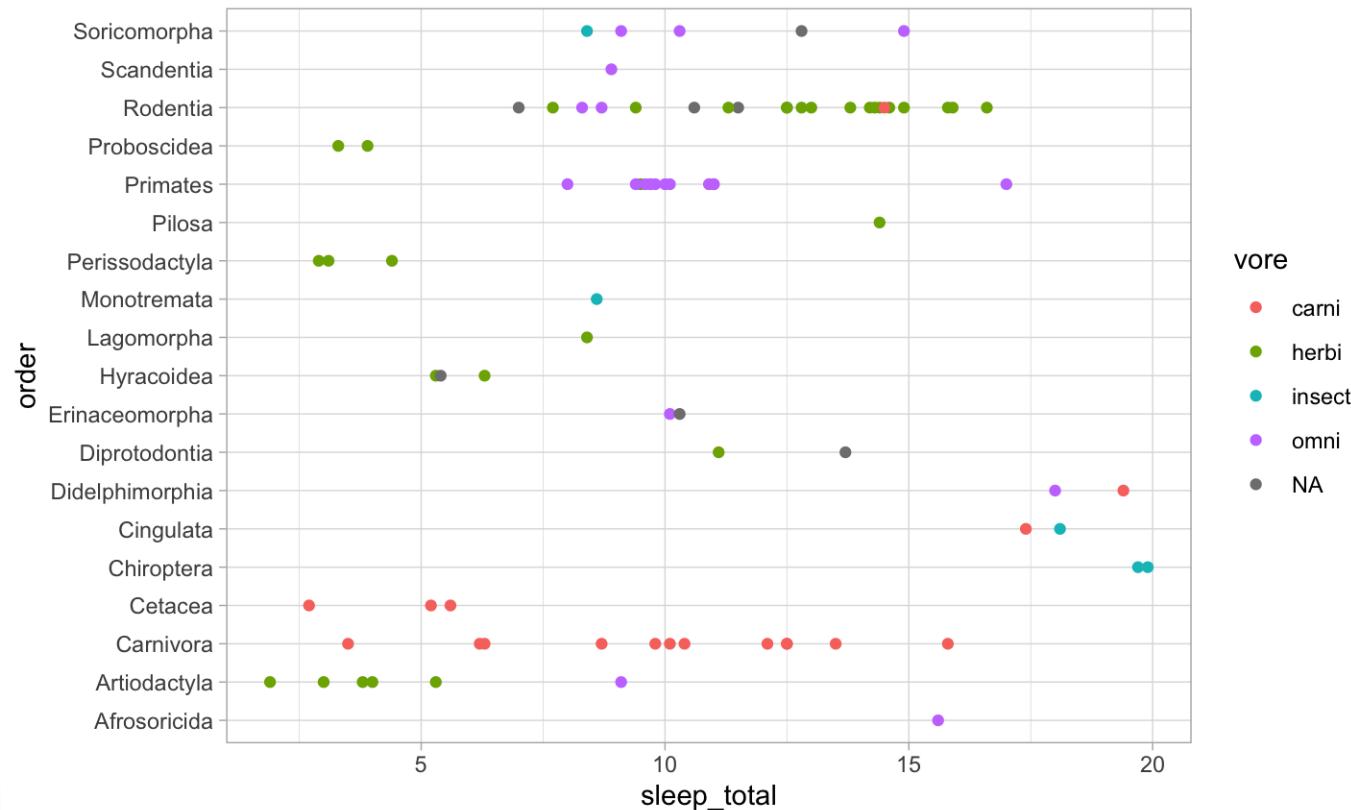
Ejercicio: Describir la relación entre edad y presión arterial sistólica, y como esta relación cambia con el sexo.

# Dotchart

## Relación con una variable categorica

Si una de las variables es categórica y tiene muchas categorías, el gráfico de dispersión puede ser también apropiado:

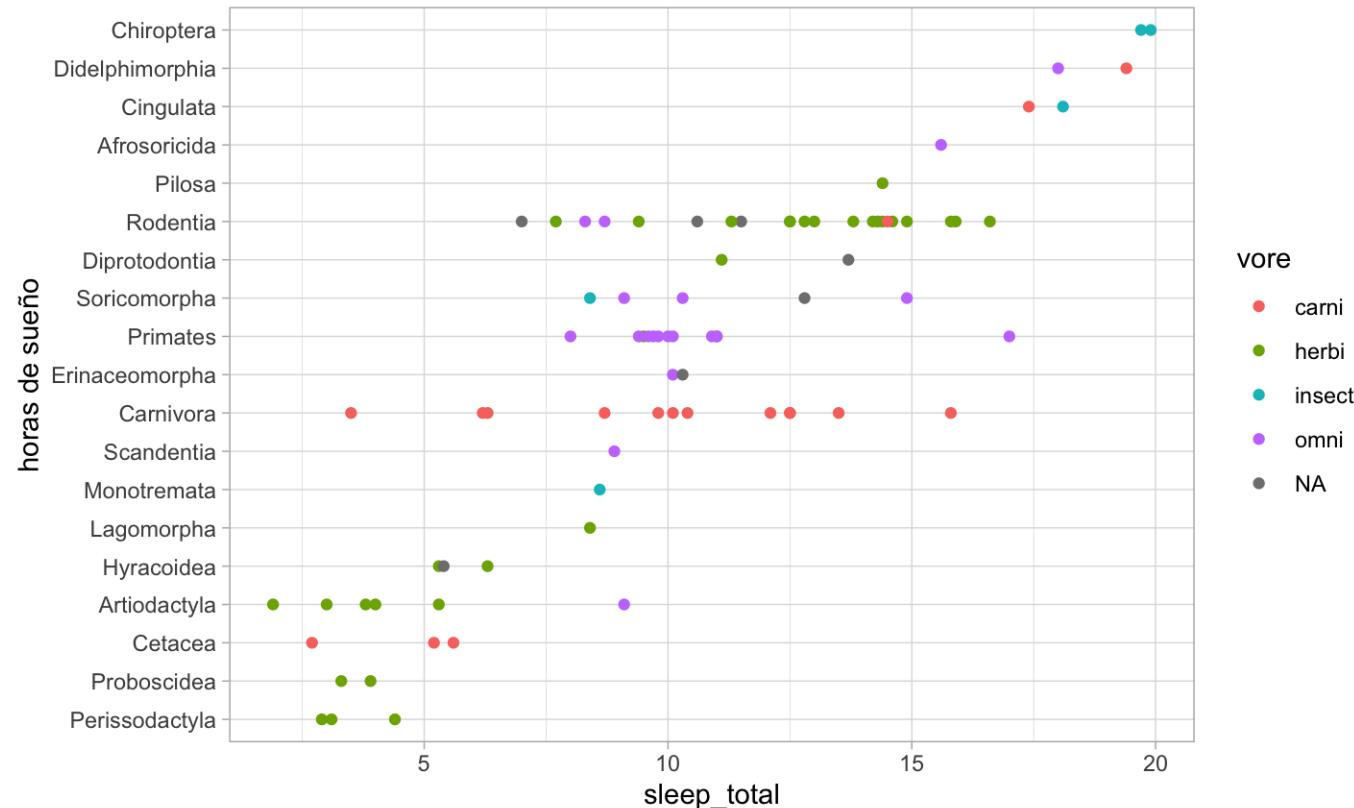
```
qplot(sleep_total, order, data=msleep, col=vore)
```



# Poniendo orden

Para mayor claridad, es recomendable ordenar la variable categórica de acuerdo a la variable numérica:

```
qplot(sleep_total,reorder(order,sleep_total),data=msleep,col=vore,ylab="horas de sueño")
```



Ejercicio: Describir con un gráfico similar al anterior, los datos de la base de datos *islands* sobre superficies de islas (es aconsejable escoger una escala log).

# Gráficos avanzados

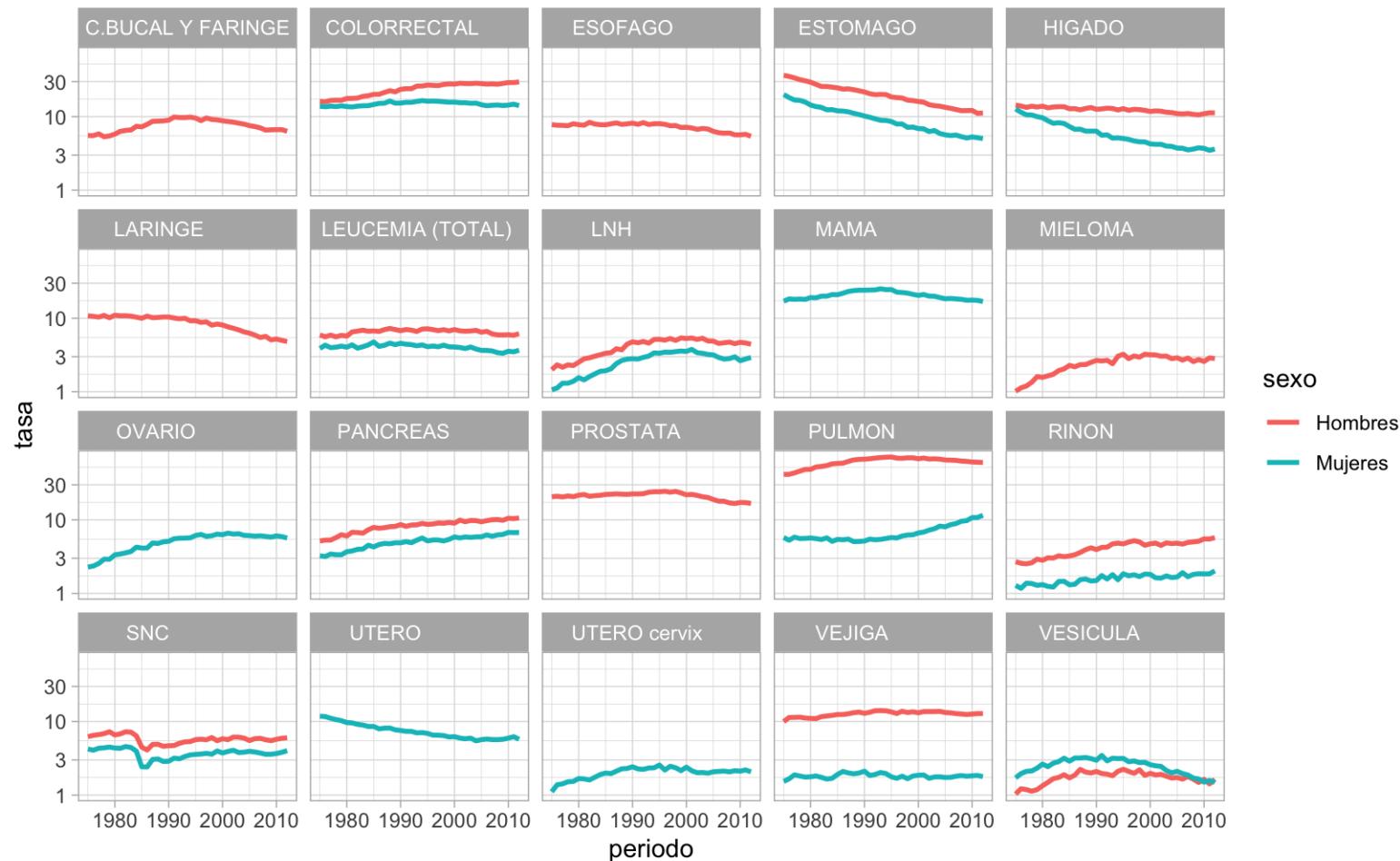
# Gramática de los graficos

Introducción a *The Grammar of Graphics* de Leland Wilkinson (2005) tal y como viene implementada en el paquete `ggplot2` de Hadley Wickham (2009).



# Un ejemplo

Evolución de la mortalidad por cáncer según su localización (España, 1975-2012)



# Un ejemplo

## El código

El código de `ggplot2` para crear el gráfico anterior fue:

```
ggplot(cancer) + #carga la base de datos cancer
  aes(x = periodo, y = tasa, col = sexo) +
  geom_line(size=1) +
  scale_y_log10() +
  facet_wrap(~ tumor)
```

Esta expresión combina varios elementos:

- **Datos:** siempre un “`data.frame`”
- **Estéticas:** columnas del `data.frame` representables gráficamente (coordenadas `x` e `y`, el color, ...)
- **Geometrías (o capas):** puntos, rectas, áreas, histogramas, ..., que pueden superponerse.
- **Facetas:** parten un gráfico en sublienzo preservando el diseño original

# Elementos de un gráfico

## Datos

La base del gráficos son los datos:

- El primer argumento de la función `ggplot` es un `data.frame`.
- **Formato alargado:** una columna para cada dimensión y una fila para cada observación.

```
load("data/cancer.RData") #carga los datos  
cancer
```

```
##          sexo periodo         tumor      tasa
## 1: Hombres    1975 C.BUCAL Y FARINGE 5.541879
## 2: Hombres    1976 C.BUCAL Y FARINGE 5.512168
## 3: Hombres    1977 C.BUCAL Y FARINGE 5.827874
## 4: Hombres    1978 C.BUCAL Y FARINGE 5.323451
## 5: Hombres    1979 C.BUCAL Y FARINGE 5.461059
## ...
## 1174: Mujeres 2008 LEUCEMIA (TOTAL) 3.402265
## 1175: Mujeres 2009 LEUCEMIA (TOTAL) 3.337342
## 1176: Mujeres 2010 LEUCEMIA (TOTAL) 3.558769
## 1177: Mujeres 2011 LEUCEMIA (TOTAL) 3.494782
## 1178: Mujeres 2012 LEUCEMIA (TOTAL) 3.675811
```

# Formato alargado

La siguiente base de datos no viene en un formato alargado:

```
VADeaths # mortalidad (por 1000 p.a) según grupos socio-demográficos y de edad (Virginia, 1940):
```

```
##          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
```

Conversión al formato alargado (función `melt`):

```
temp=data.table(VADeaths,keep.rownames=TRUE) #require(data.table)
mortalidad=melt(temp,id.vars="rn") #formato alargado
names(mortalidad) <- c("edad","grupo","tasa")
str(mortalidad)
```

```
## Classes 'data.table' and 'data.frame': 20 obs. of 3 variables:
## $ edad : chr "50-54" "55-59" "60-64" "65-69" ...
## $ grupo: Factor w/ 4 levels "Rural Male","Rural Female",...: 1 1 1 1 1 2 2 2 2 ...
## $ tasa : num 11.7 18.1 26.9 41 66 8.7 11.7 20.3 30.9 54.3 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

# Estéticas (aes)

El siguiente código crea un “protográfico” p que contiene los datos que vamos a utilizar:

```
p <- ggplot(mortalidad)
```

Pero, este código es insuficiente para dibujar un gráfico ya que no hemos indicado que dimensión de la base de datos se va a representar.

Para ello, se añade al objeto p información sobre las “estéticas” (las coordenadas, el color, la forma o el tamaño de un punto, ...) y su relación con las variables de la base de datos:

```
p <- p + aes(x = edad, y = tasa, colour = grupo)  
p$mapping # relación (o mapeo) entre estéticas y columnas de la base de datos
```

```
## Aesthetic mapping:  
## * `x`      -> `edad`  
## * `y`      -> `tasa`  
## * `colour` -> `grupo`
```

# ¿Cuántas estéticas existen?

Alrededor de una docena, aunque se utilizan, generalmente, menos:

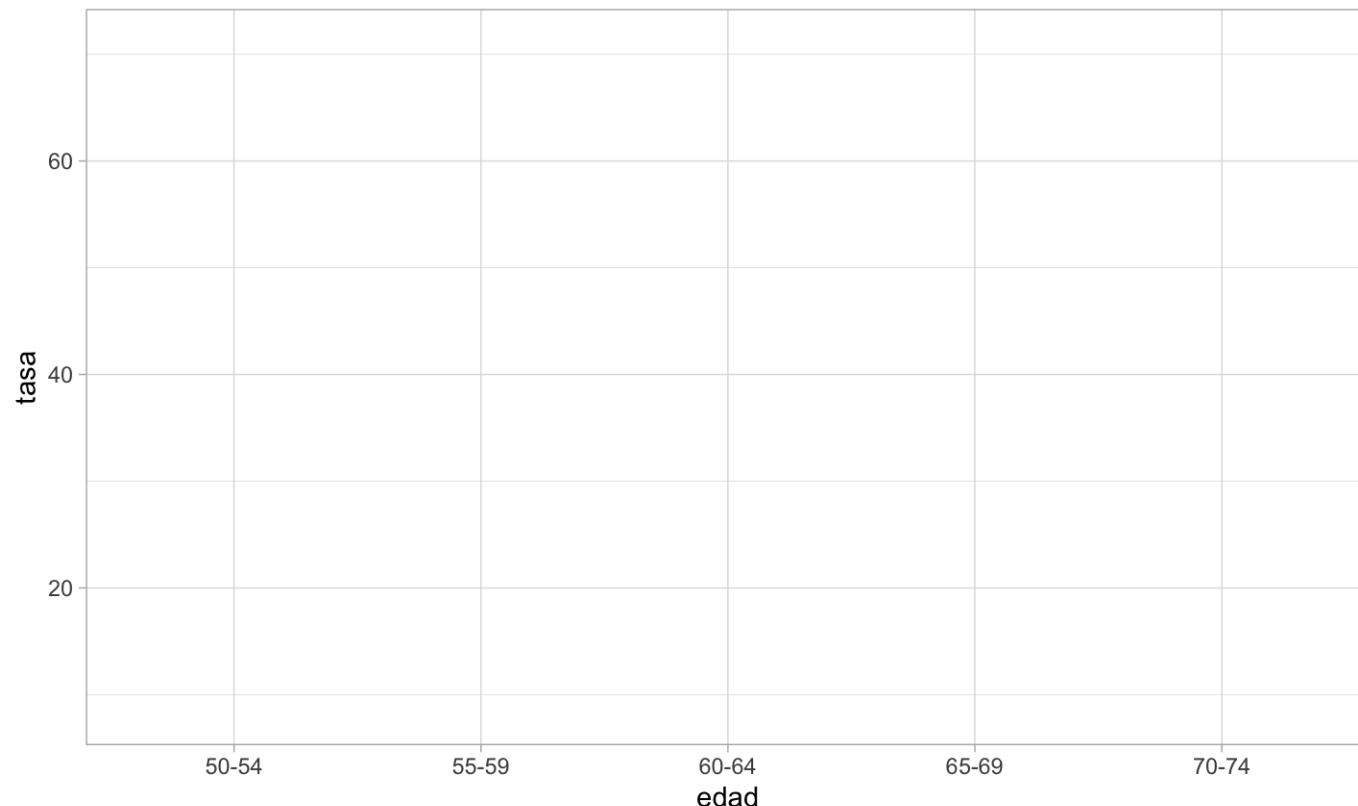
- `x` e `y`, coordenadas horizontal y vertical.
- `colour`, color de líneas y bordes.
- `size`, para el tamaño.
- `shape`, que indica la forma de los puntos (cuadrados, triángulos, etc.) de los puntos o del trazo (continuo, punteado) de las líneas.
- `alpha` para la transparencia: los valores más altos tendrían formas opacas y los más bajos, casi transparentes. También muy útil para el solapamiento de puntos.
- `fill`, para el color de relleno de las formas sólidas (barras, etc.).

No todas las *estéticas* tienen la misma potencia en un gráfico. El ojo humano percibe fácilmente colores y longitudes, pero tiene problemas para comparar áreas. Se recomienda usar las estéticas más potentes para representar las variables más importantes.

## Un lienzo

El objeto `p` resultante aún no permite representar los datos (le falta capas):

`p`



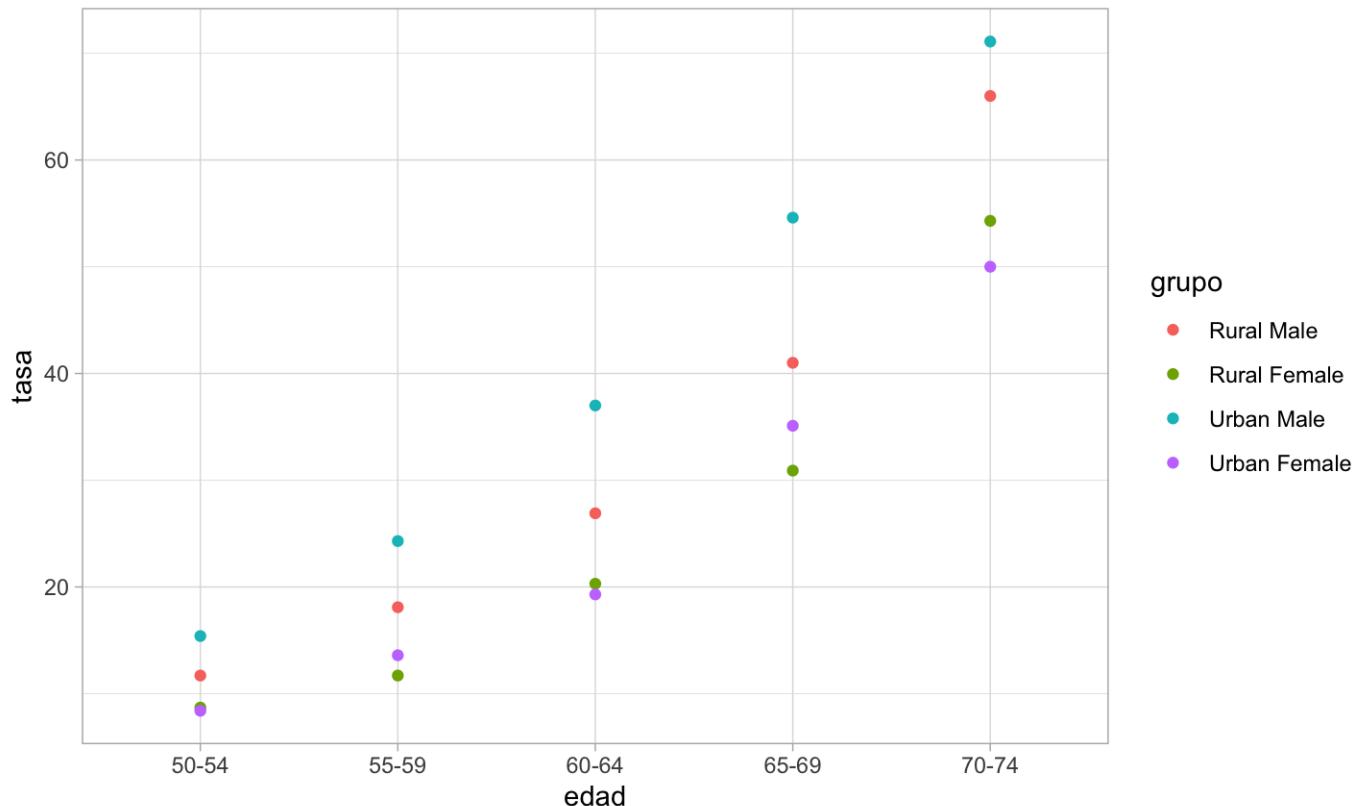
puede apreciar los ejes.

... no obstante, ya se

# Capas (geoms)

Las capas (o geoms para `ggplot2`) son los verbos del lenguaje de los gráficos. Indican como representar los datos mediante las estéticas en un lienzo. Una vez añadida una capa al gráfico, este puede pintarse:

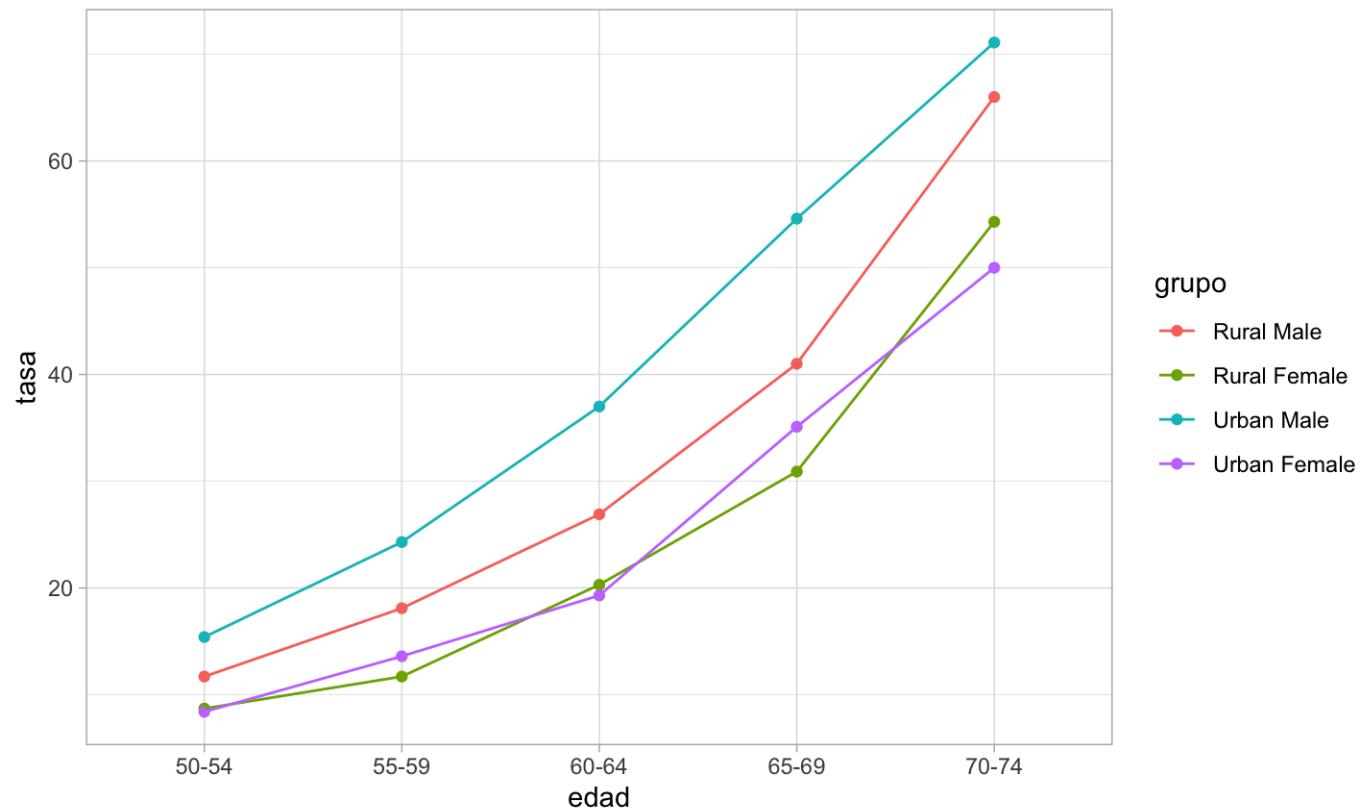
```
p <- p + geom_point()  
p #ggplot(mortalidad, aes(x = edad, y = tasa, colour = grupo)) + geom_point()
```



## Varias capas

Una característica de las capas, y de ahí su nombre, es que pueden superponerse:

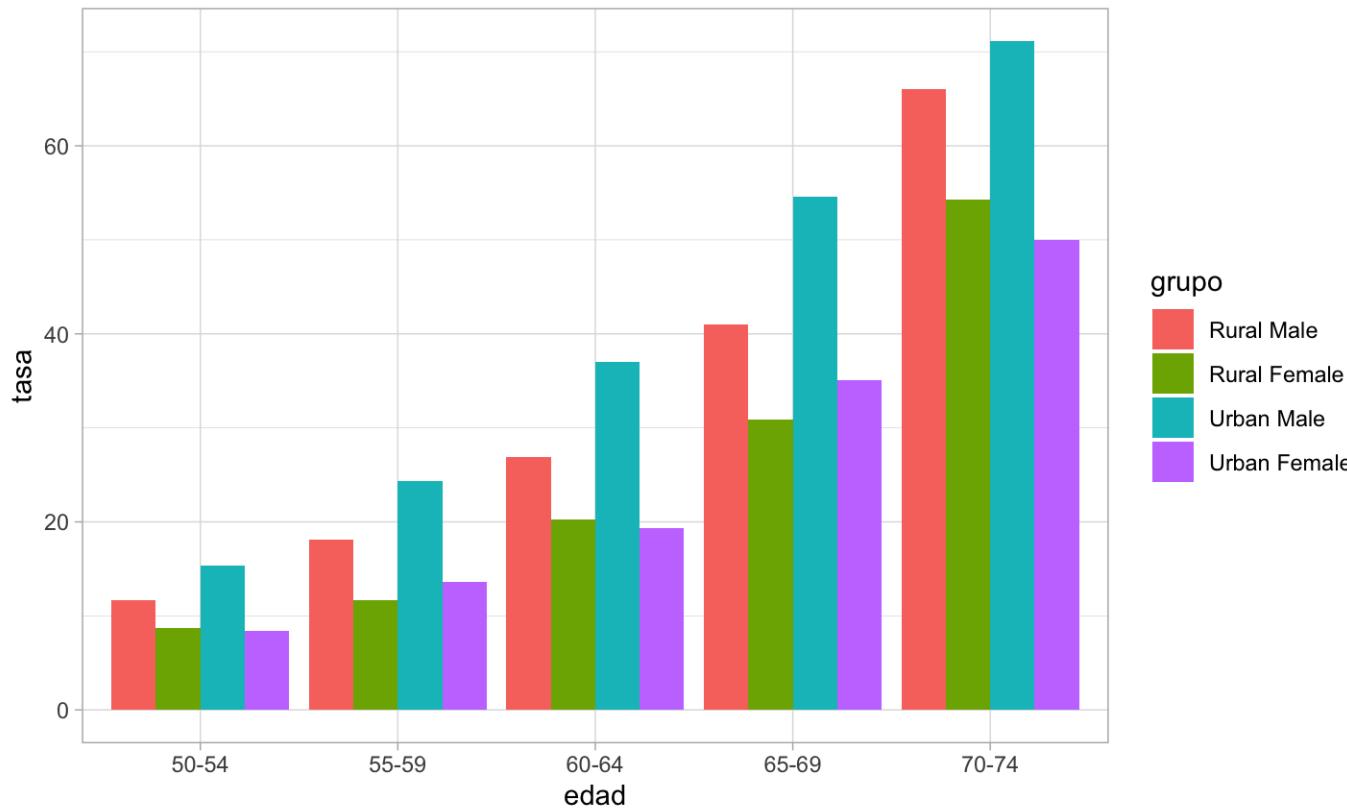
```
# Se requiere la estética `group` para conectar los puntos de una linea  
# cuando la variable en abscisa es un factor.  
ggplot(mortalidad, aes(x = edad, y = tasa, colour = grupo, group= grupo)) +  
  geom_point() +  
  geom_line()
```



## Más capas

Abajo una representación mediante un diagrama de barra de los datos anteriores:

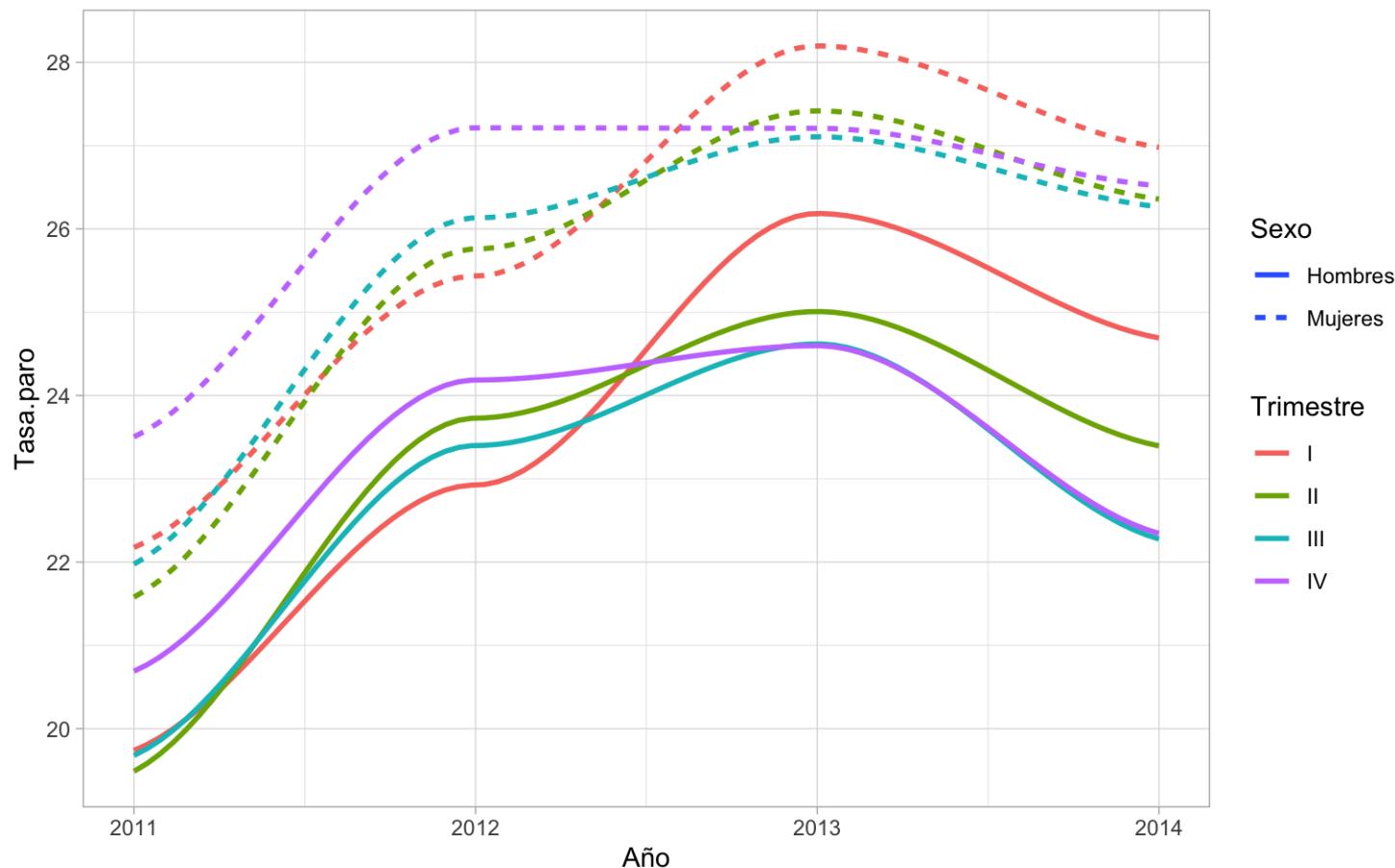
```
ggplot(mortalidad, aes(x = edad, y = tasa, fill = grupo)) +  
  geom_bar(stat="identity", position="dodge")
```



Existen muchos tipos de capas. Los más usuales son `geom_point`, `geom_line`, `geom_histogram`, `geom_bar` y `geom_boxplot` (ver <https://ggplot2.tidyverse.org/reference/>) para una lista actualizada.

# Ejercicio

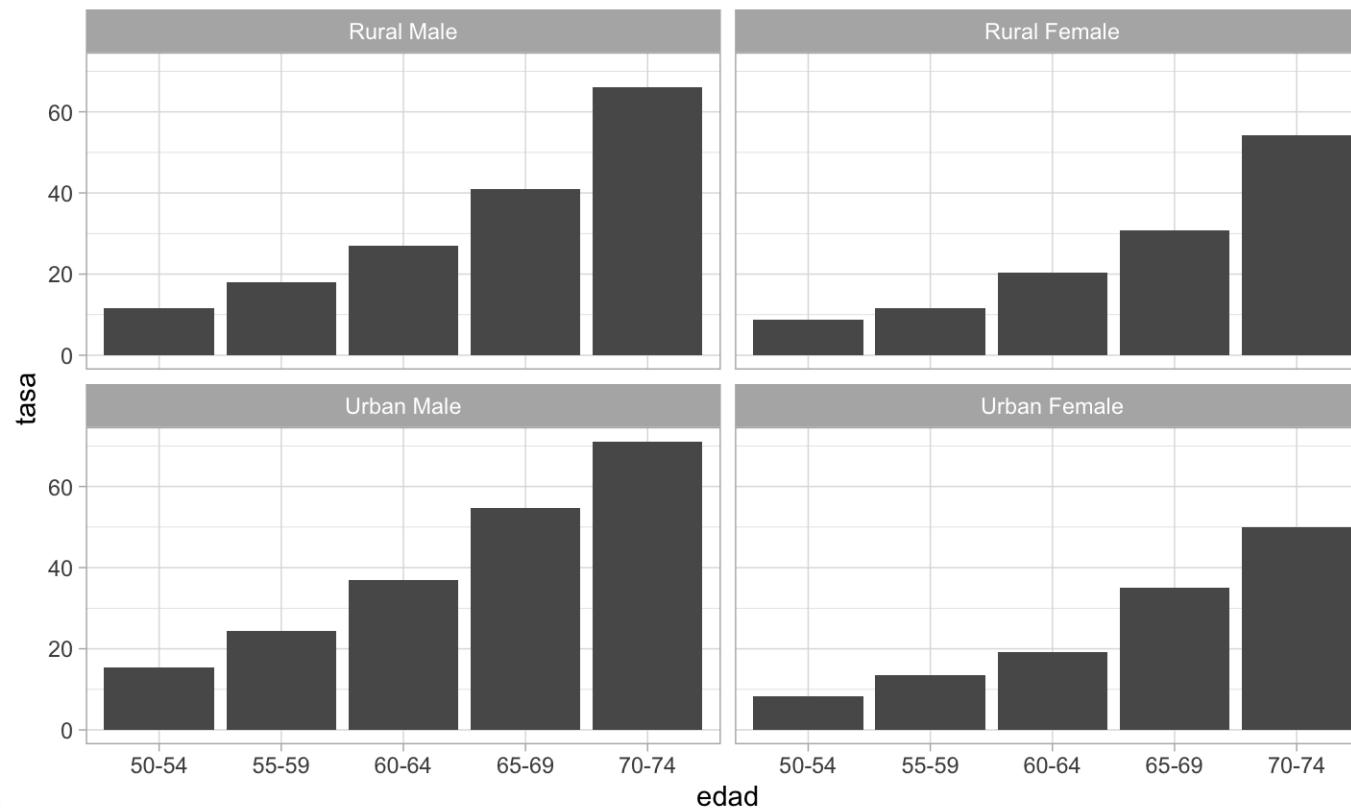
Elaborar el siguientes gráfico sobre la evolución del paro en España. Utilizar la capa `geom_smooth` para suavizar la tendencia y la estética `linetype` para distintos tipos de curvas.



# Facetas

Las facetas permiten subdividir un gráfico. Suele ser un recurso muy eficiente para añadir otra dimensión al gráfico:

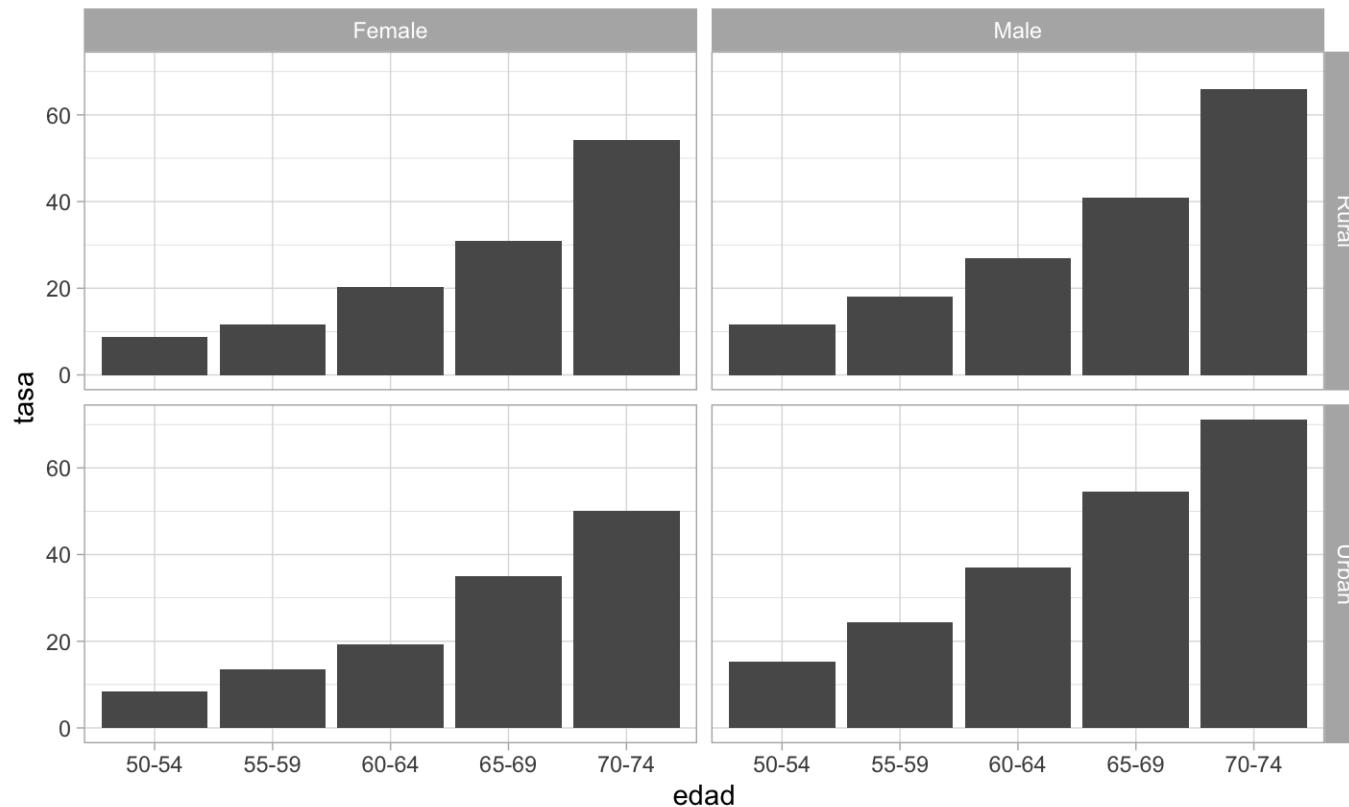
```
ggplot(mortalidad, aes(x = edad, y = tasa)) +  
  geom_bar(stat="identity") +  
  facet_wrap(~grupo)
```



## facetas cruzadas (facet\_grid)

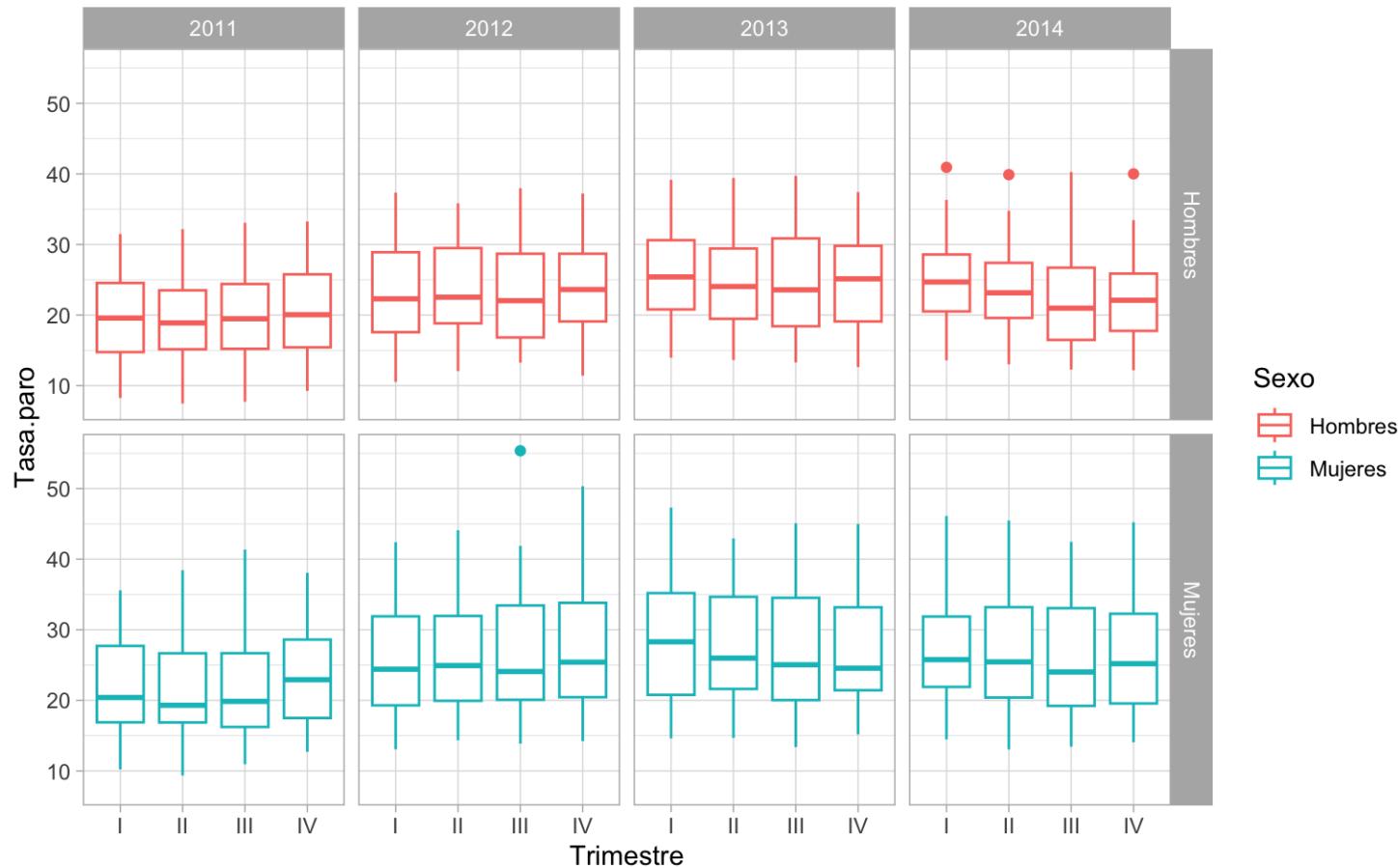
Se puede subdividir el lienzo de acuerdo a dos (¡o más!) variables:

```
mortalidad[,c("zone", "sex")]:= tstrsplit(grupo, " ")  
ggplot(mortalidad, aes(x = edad, y = tasa)) +  
  geom_bar(stat="identity") +  
  facet_grid(zone ~ sex)
```



# Ejercicio

Elaborar el siguiente gráficos sobre la evolución del paro



## Otro ejercicio más

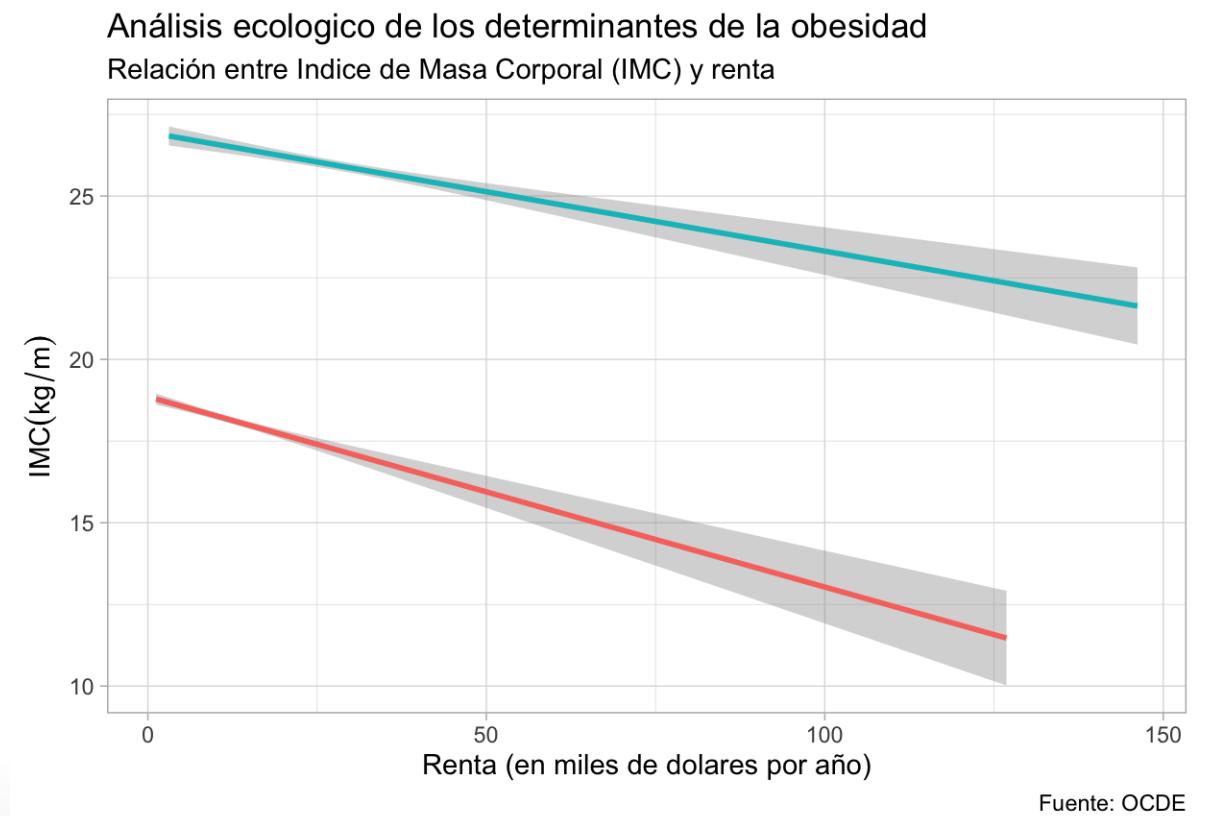
Para este segundo gráfico, limitar la base a las provincias de Zaragoza, Huesca y Teruel.



# Etiquetas

Las estéticas se pueden etiquetar con la función `labs`. Esta misma función se puede usar para añadir un título, un subtítulo o una nota al pie del gráfico:

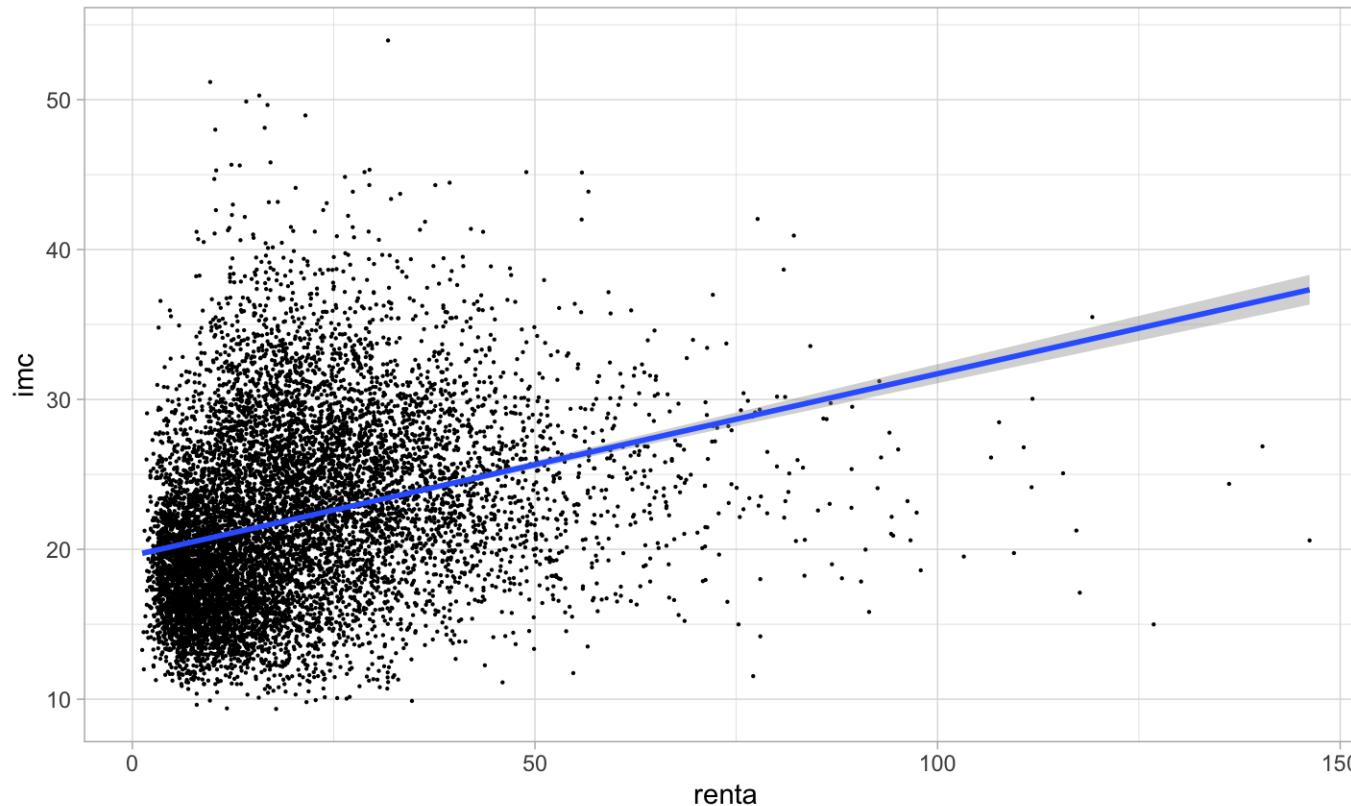
```
obesidad<-fread("data/obesidad.csv")
p<-ggplot(obesidad,aes(x=renta,y=imc,color=region))+geom_smooth(method="lm")
p + labs(x = "Renta (en miles de dolares por año)", y = quote(IMC (kg/m)), color = "Continente",
         title="Análisis ecologico de los determinantes de la obesidad",
         subtitle="Relación entre Indice de Masa Corporal (IMC) y renta", caption = "Fuente: OCDE")
```



# Escalas

La escala por defecto de una estética no es siempre la adecuada para una buena representación:

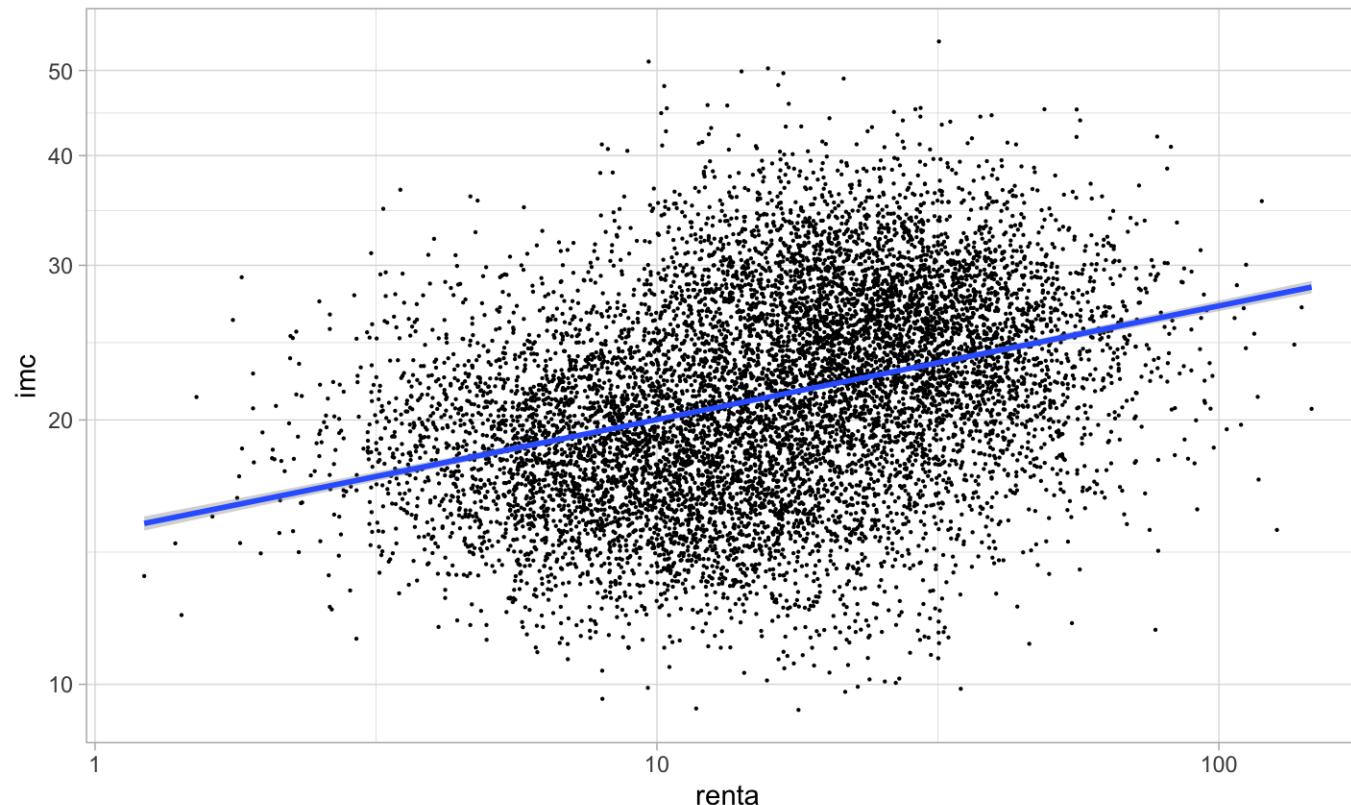
```
p<-ggplot(obesidad,aes(x=renta,y=imc))+geom_point(size=.1)+geom_smooth(method="lm")  
p
```



# Transformación

La escala de una estética puede ser modificada para mejorar la claridad de la representación:

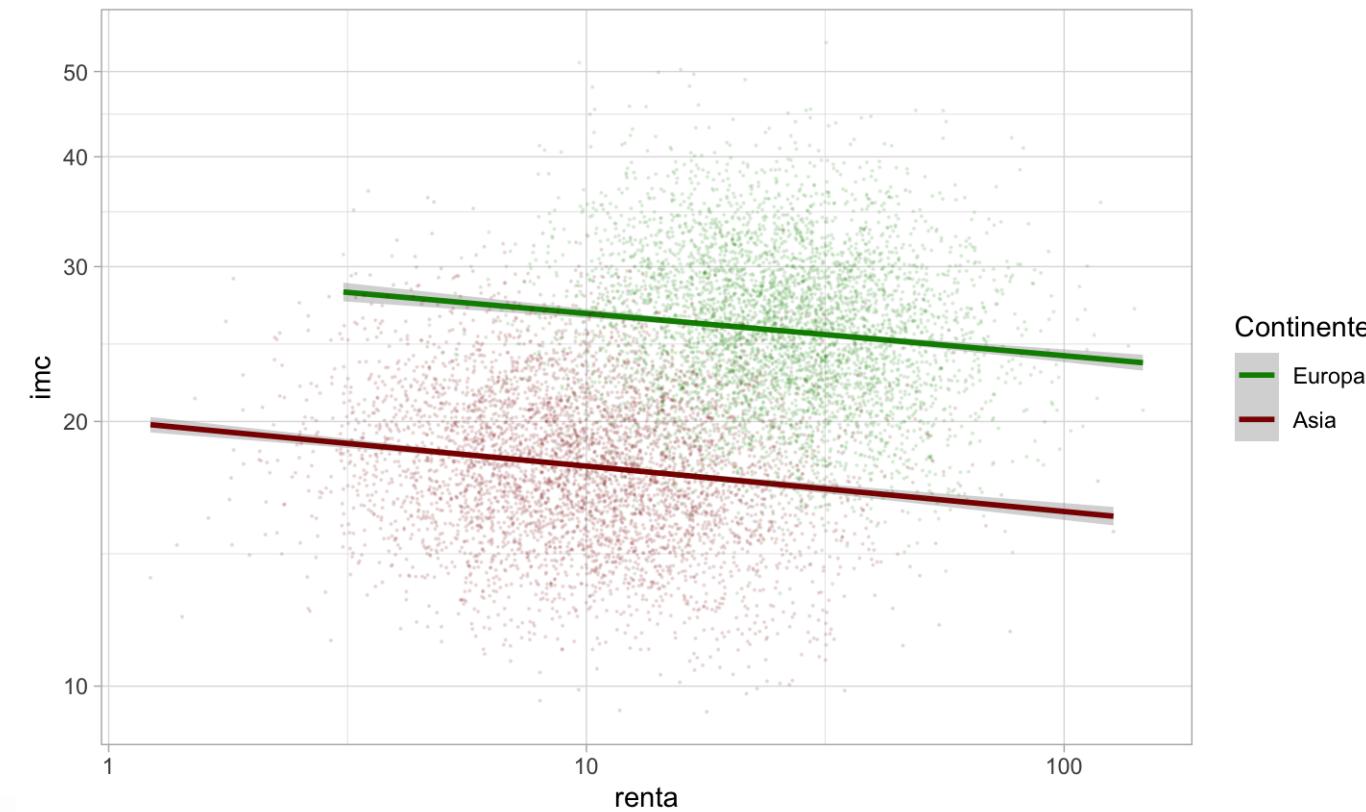
```
p+scale_x_log10()+scale_y_continuous(breaks=seq(10,50,10),trans="log")
```



# Transformación

Cada una de las estéticas (incluido el color, la transparencia, ...) tiene escala que puede ser configurada:

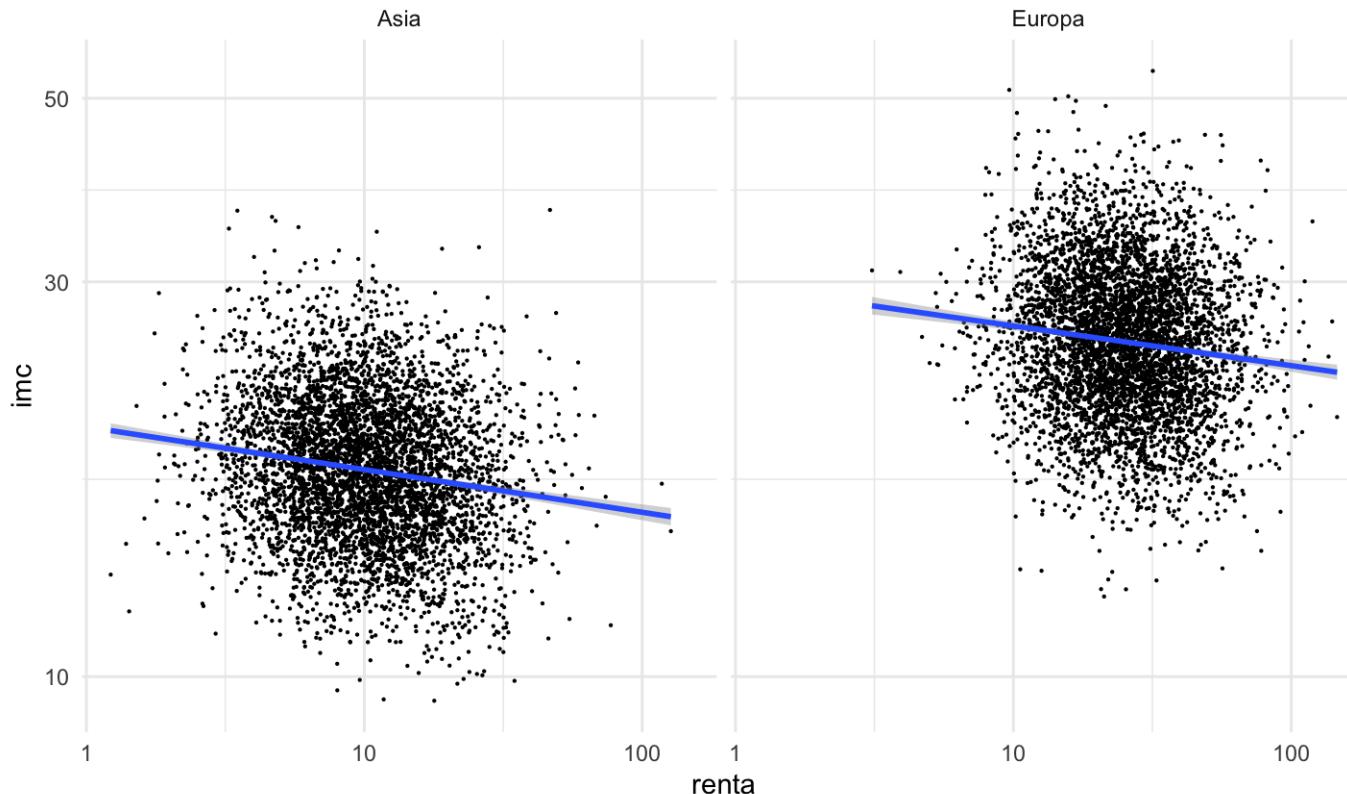
```
ggplot(obesidad, aes(x=renta, y=imc, color=region)) + geom_smooth(method="lm") +  
  geom_point(size=.1, alpha=.1) +  
  scale_x_log10() +  
  scale_y_continuous(breaks=seq(10, 50, 10), trans="log") +  
  scale_color_manual("Continente", values=c("green4", "red4"), limits=c("Europa", "Asia"))
```



# Temas

- Los *temas* de `ggplot2` permiten modificar aspectos estéticos del gráfico (ejes, colores de fondo, tamaño de los caracteres, ...) para adecuarse a criterios de estilo de publicación.
- Existen muchos temas predefinidos (ver <https://ggplot2.tidyverse.org/reference/ggtheme.html>) y el paquete `ggthemes` para temas que se ajustan al estilo de reconocidas revistas científicas.
- El tema que usa `ggplot2` por defecto es `theme_grey`, aquí otro más minimalista:

```
p + facet_grid(~region) + scale_y_log10() + scale_x_log10() + theme_minimal()
```



# Exportación de los graficos

Una vez creado un gráfico, es posible exportarlo en diversos formatos:

- Imagen tipo bitmap (jpeg, png, bmp, tiff,...)
- Imagen vectorial (pdf, svg,...)

La función `ggsave` guarda en un fichero el último gráfico generado con `ggplot2` con el formato indicado en la extensión del nombre del fichero que se quiere generar:

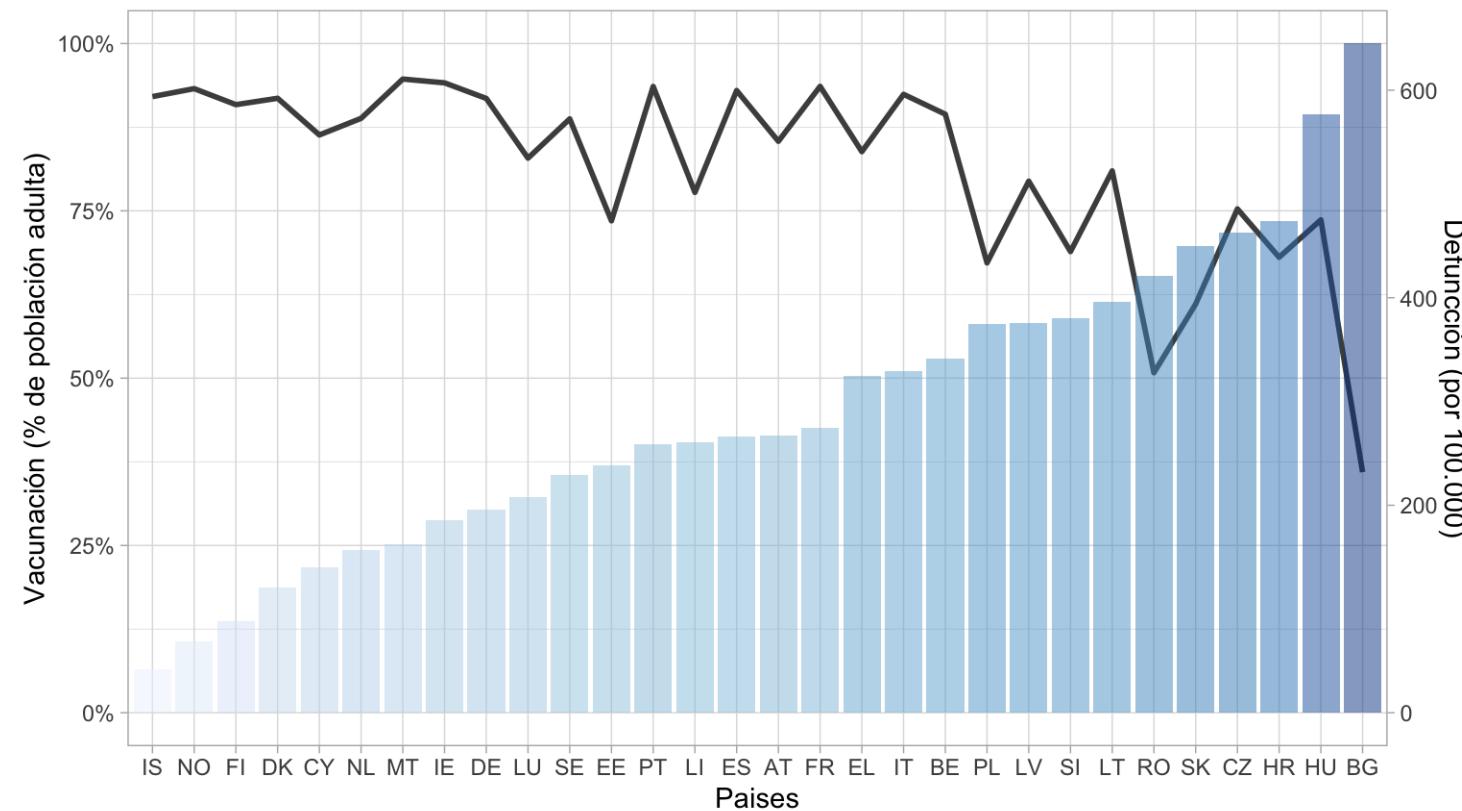
```
ggplot(obesidad,aes(x=renta,y=imc,color=region)) +geom_smooth(method="lm")
ggsave("obesidad.pdf")
#ggsave("mortalidad.pdf", width = 20, height = 20, units = "cm")
ggsave("obesidad.png")
```

Las imágenes vectoriales tienen una resolución “infinita” y suelen ocupar poca memoria. Sin embargo, no todos los editores de texto admiten este tipo de formato.

# Unos graficos destacados

## Dos ejes

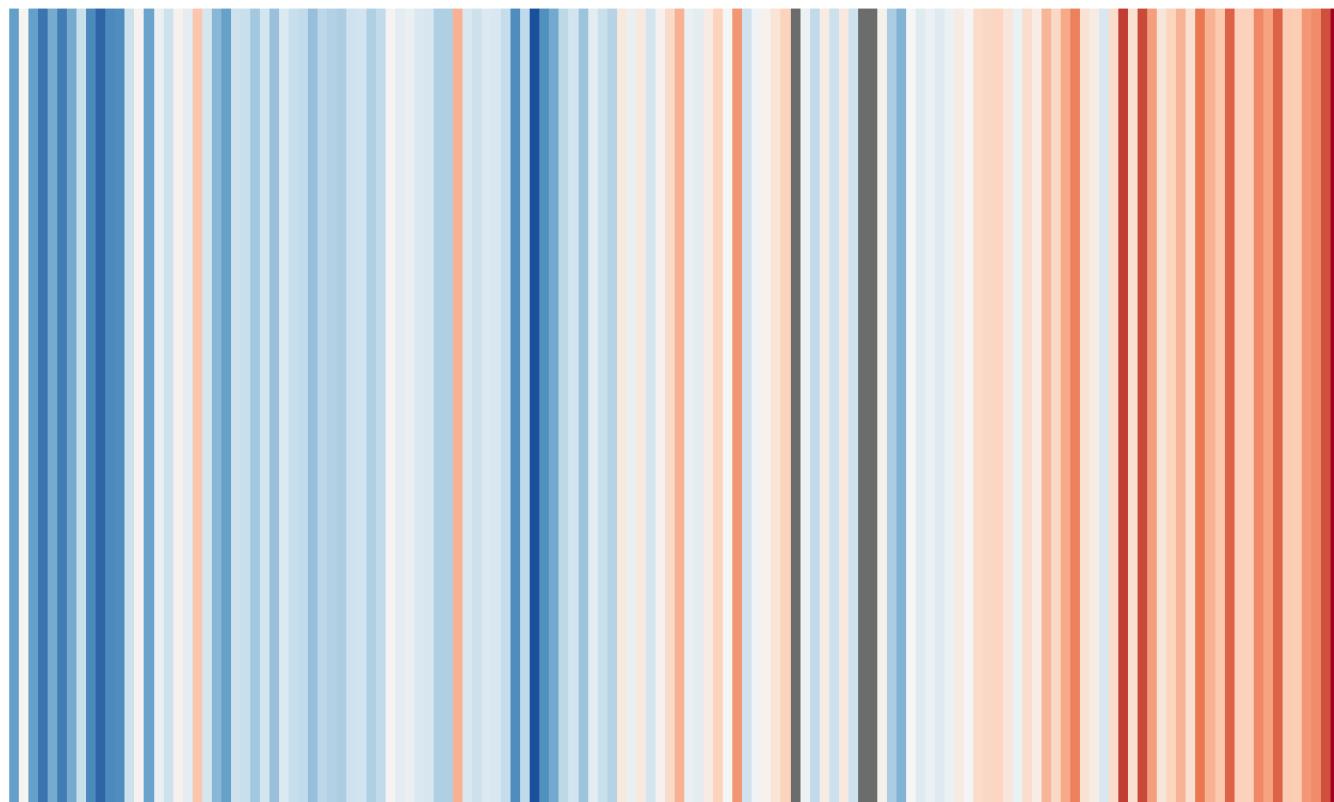
Representar en un sólo gráfico las variaciones de las tasas de mortalidad por COVID y vacunación entre países de la UE (ver `?sec_axis`).



# Warming stripes

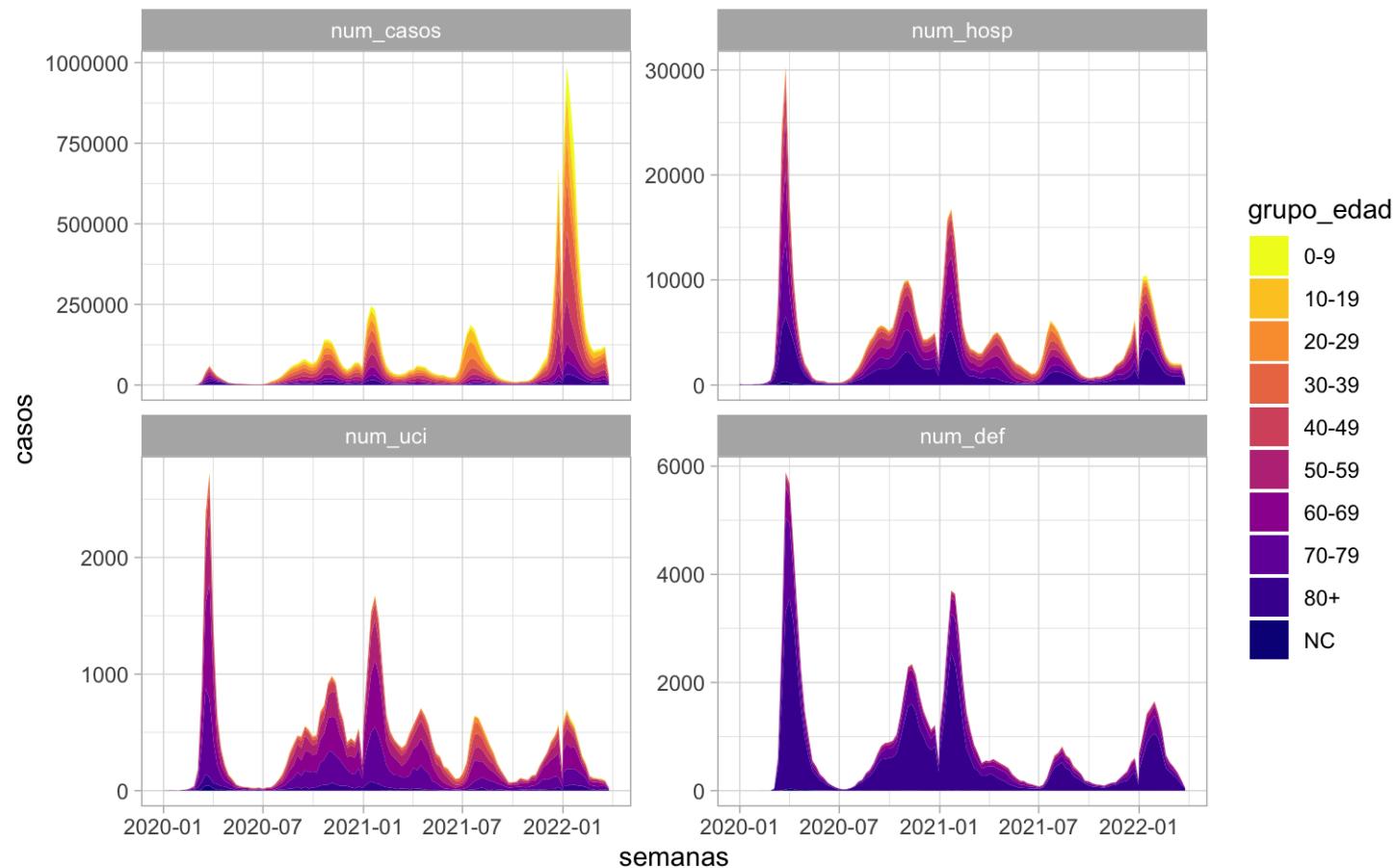
A partir de los datos de temperaturas anuales en Lisboa de 1880 a 2008 (base de datos `temp_lisboa.csv`), crear y exportar el siguiente gráfico, utilizando la capa `geom_tile`, el tema vacío `theme_void()` y la paleta de colores "RdBu" `scale_fill_distiller(palette = 'RdBu')`.

LISBOA 1880-2018



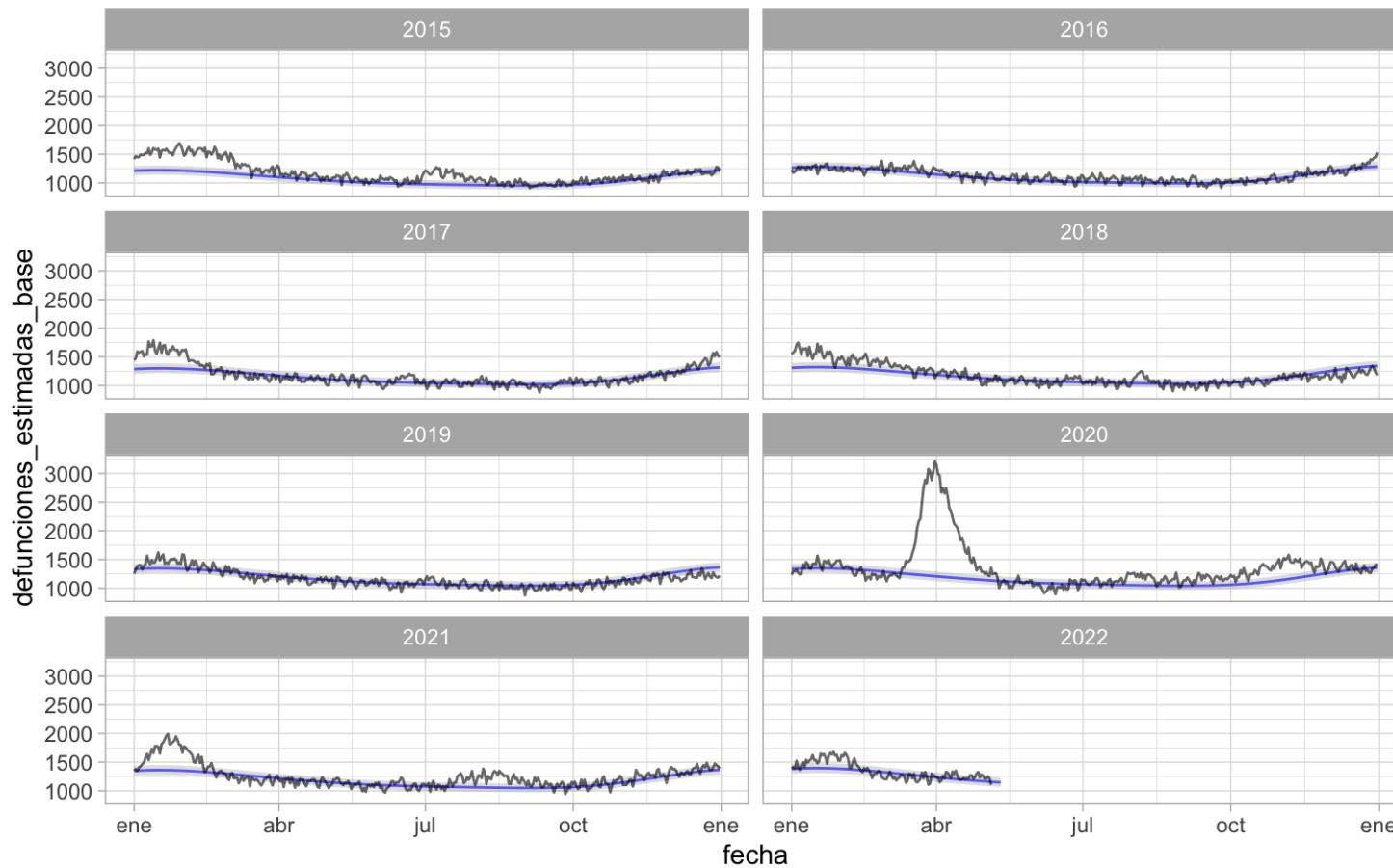
# Representación por áreas

Utilizando la base de COVID, describir en un único gráfico la evolución de esta pandemia en España de acuerdo a la edad y gravedad (se sugiere utilizar la geometría `geom_area`).



# Comparación anual de series

Describir en un único gráfico la evolución de las defunciones en España por año respecto a las defunciones esperadas (base de datos del MOMO).



# Mapas

# Representación con `ggmap`

- Con `ggplot2` se puede también construir representaciones gráficas con información geográfica (puntos, segmentos, etc.): basta con que las estéticas `x` e `y` se correspondan con la longitud y la latitud de los datos.
- Lo que permite hacer el paquete `ggmap` es añadir a una representación grafica de datos georeferenciados una capa cartográfica adicional.
- Para eso usa recursos disponibles en la web a través de APIs (de Google, OSM y otros).

```
require(ggmap) #carga la librería ggmap
```

# Geolocalización

Existen varios proveedores que proporcionan APIs de geolocalización. Uno de ellos es Google: dado el nombre más o menos normalizado de un lugar, la API de Google devuelve sus coordenadas.

Hoy en día, este servicio requiere una licencia. La función `geocode` encapsula la consulta a dicha API y devuelve un objeto (un `data.frame`) que contiene las coordenadas del lugar de interés:

```
#cne<- geocode('Calle Monforte de Lemos 5, madrid') #sólo funciona con licencia de Google
cne <-c(lat= 40.47767,lon=-3.691096)

### Una alternativa gratuita sin límites para direcciones en España
#require(cartocciudad)
#cne <- cartocciudad_geocode('Calle Monforte de Lemos 5, madrid')

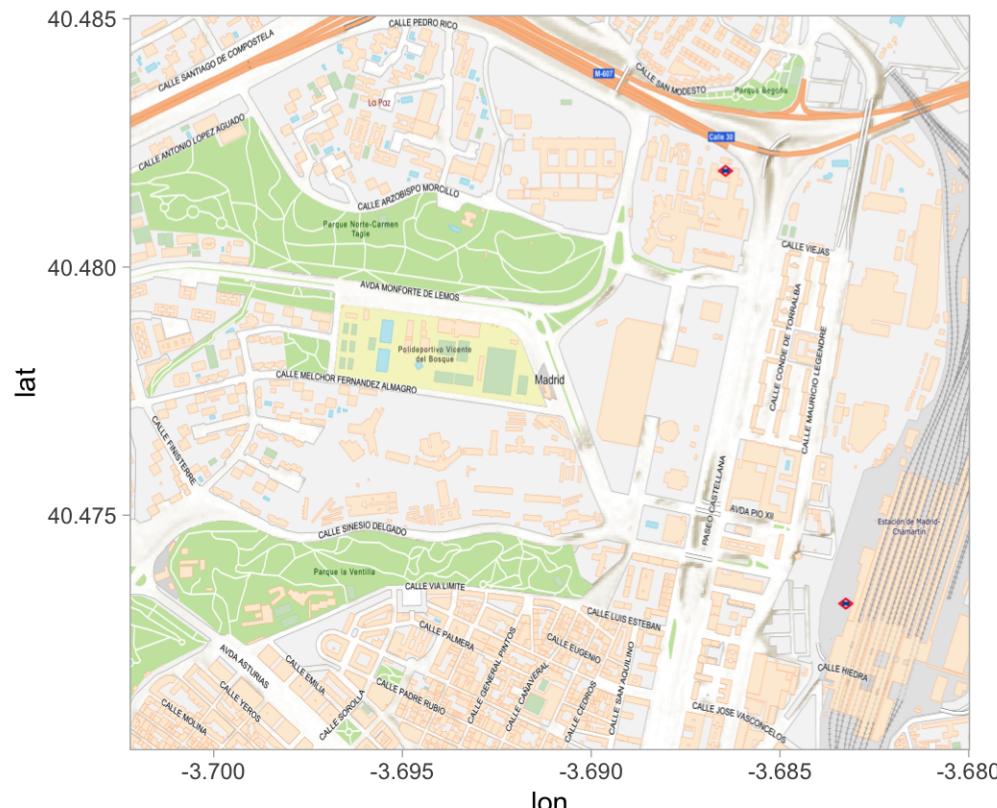
### Otra alternativa Open Street Map (OSM)
# require(tmaptools)
# cne=geocode_OSM('Monforte de Lemos 5, madrid')
```

# Capa cartográfica

La función `get_map` consulta un servicio de información cartográfica y descarga un mapa (una imagen):

```
require(cartocciudad)
mapa=cartocciudad_get_map(cne, radius=2)
#cne<-c(left=-3.7, bottom=40.47, right=-3.68, top=40.485)
#mapa <- get_stamenmap(cne, zoom = 16, maptype="toner-lite")

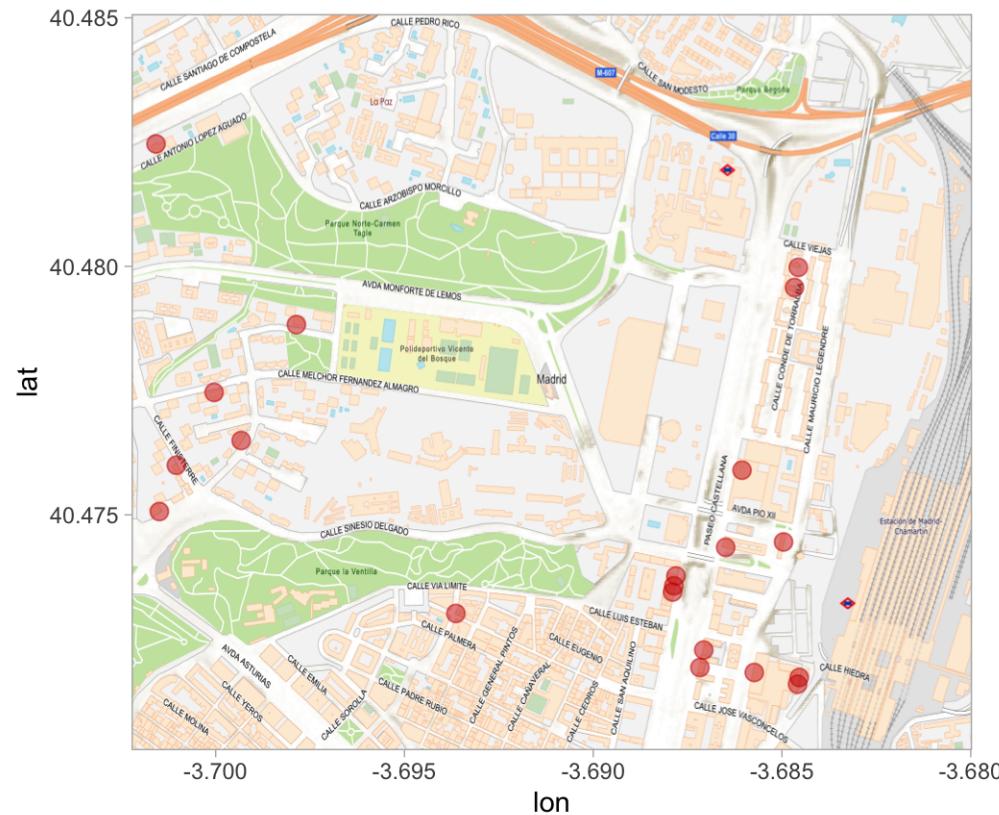
ggmap(mapa) #representación de la capa cartográfica
```



# Representación de los datos georeferenciados

En el ejemplo siguiente se representa mediante puntos las terrazas alrededor del CNE:

```
require(data.table)
terrazas=fread("data/terrazas.csv") # localización de los bares con terrazas de Madrid
ggmap(mapa) + geom_point(aes(x=lon, y=lat), data = terrazas, colour = 'red3', size = 3, alpha=.5)
```

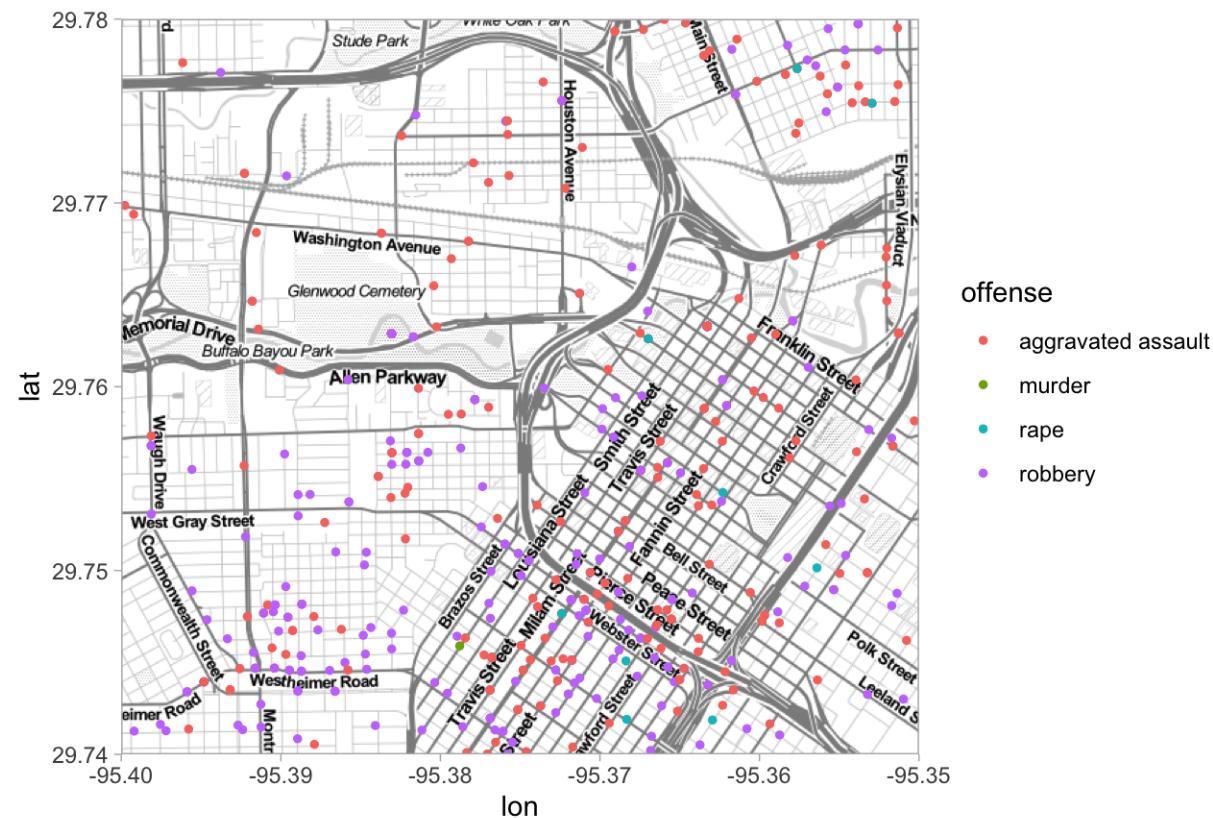


# más ejemplos de mapas con puntos

El conjunto de datos `crime` incluye información geolocalizada de delitos cometidos en Houston:

```
delitos.houston <- subset(crime,!crime$offense %in% c("auto theft", "theft", "burglary")) #delitos violentos  
#houston<-geocode_OSM('Houston') #require(tmaptools) para geolocalizar Houston  
houston=c(left = -95.4, bottom = 29.74, right = -95.35, top = 29.78) #barrio alrededor de Louisiana Street  
houston.callejero <- get_stamenmap(houston,zoom=14, maptype="toner-lite")
```

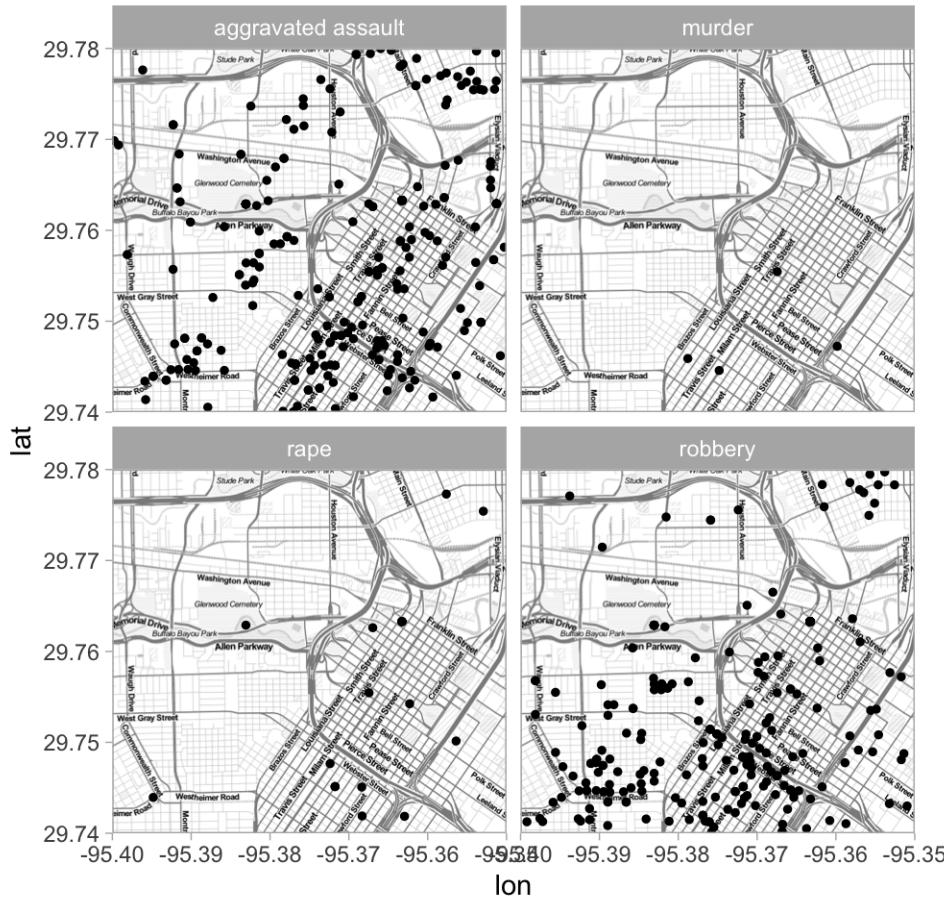
```
p <- ggmap(houston.callejero)  
p + geom_point(aes(x = lon, y = lat, colour = offense), data = delitos.houston, size = 1)
```



# Facetas

El recurso a facetas de `ggplot2` está también disponible en `ggmap`:

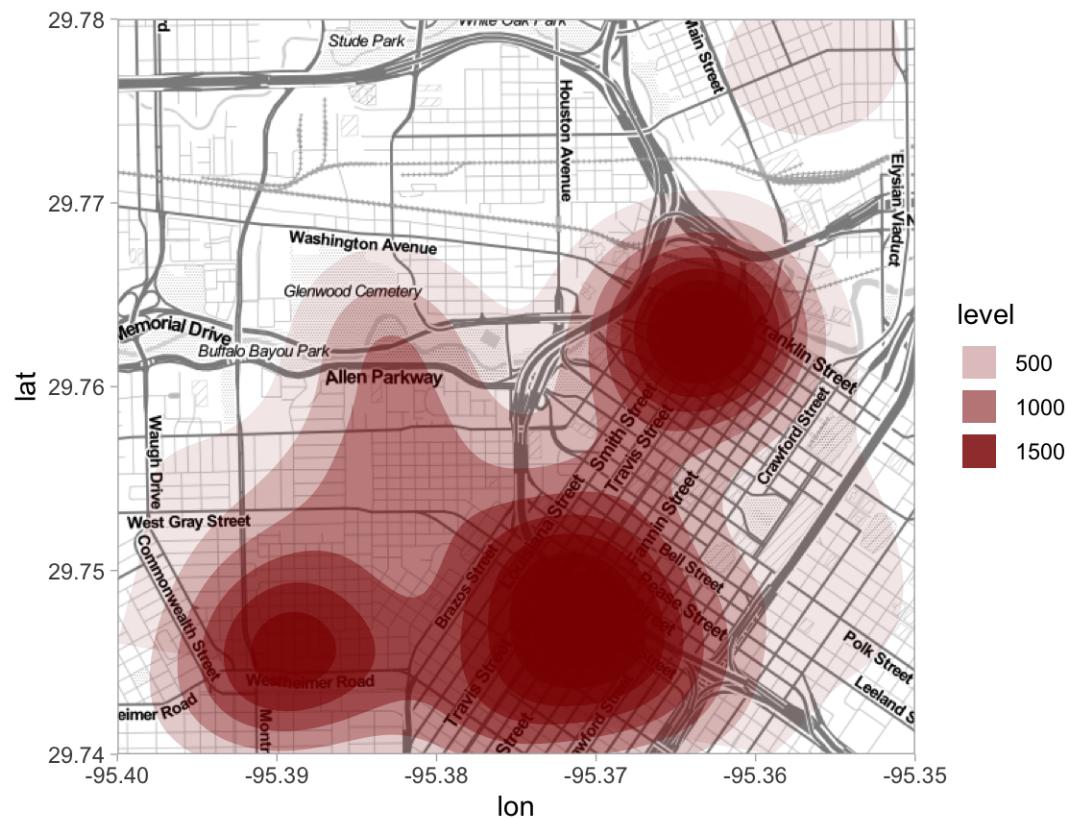
```
p + geom_point(aes(x = lon, y = lat), data = delitos.houston, size = 1) + facet_wrap(~ offense)
```



## Más allá de los puntos: densidades y polígonos

Además de `geom_point`, están disponibles otros tipos de capas como `stat_bin2d`, que cuenta el número de eventos (aquí atracos) que suceden en regiones cuadradas de un tamaño predefinido. Se puede también utilizar `stat_density2d`, más flexible, para identificar las zonas de mayor criminalidad.

```
p + stat_density2d(aes(x = lon, y = lat, alpha = ..level..), fill="red4",
                     size = 2, data = subset(delitos.houston, offense=="robbery"), geom = "polygon")
```



# Información agregada y Shape

Por otra parte, la información estadística puede ser proporcionada de manera agregada en unidades espaciales (a nivel provincial, municipal, ...), como para los datos del paro.

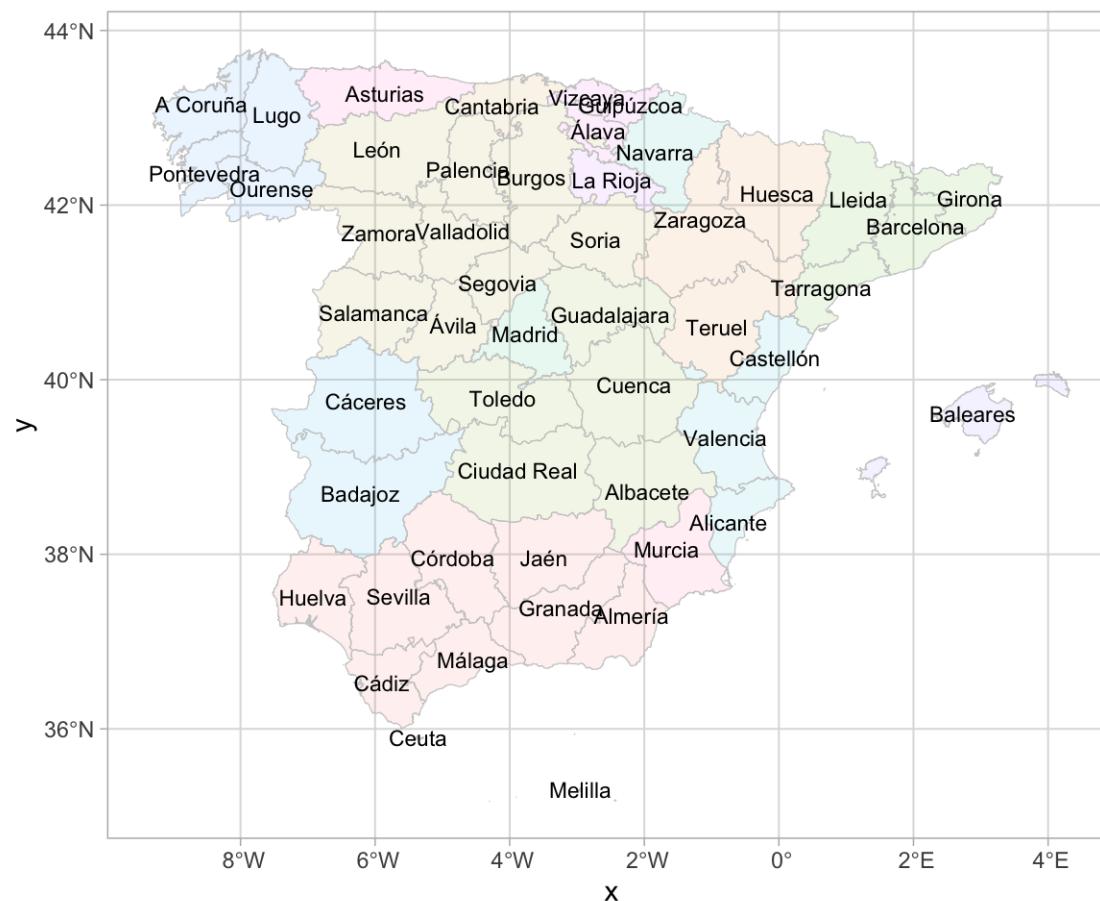
Para representar estos datos, se necesita el conjunto de polígonos (o “shape”) que definen las secciones geográficas. Esta información se puede descargar desde el servidor GADM mediante el paquete `raster`:

```
require(raster); require(sf)
shape <- getData("GADM", country= "Spain", level = 2, type="sf") #mapa administrativo a nivel provincial
```

```
## Simple feature collection with 52 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension: XY
## Bounding box: xmin: -18.16153 ymin: 27.63736 xmax: 4.328195 ymax: 43.79153
## Geodetic CRS: WGS 84
## First 10 features:
##   GID_0 NAME_0    NAME_1    NAME_2    TYPE_2 CC_2               geometry
## 1   ESP Spain Andalucía Almería Provincia    04 MULTIPOLYGON ((((-3.030417 3...
## 2   ESP Spain Andalucía Cádiz Provincia    11 MULTIPOLYGON ((((-6.219583 3...
## 3   ESP Spain Andalucía Córdoba Provincia   14 MULTIPOLYGON ((((-5.048538 3...
## 4   ESP Spain Andalucía Granada Provincia   18 MULTIPOLYGON ((((-3.35014 36...
## 5   ESP Spain Andalucía Huelva Provincia   21 MULTIPOLYGON ((((-6.836479 3...
## 6   ESP Spain Andalucía Jaén Provincia    23 MULTIPOLYGON ((((-3.008117 3...
## 7   ESP Spain Andalucía Málaga Provincia   29 MULTIPOLYGON ((((-4.000826 3...
## 8   ESP Spain Andalucía Sevilla Provincia  41 MULTIPOLYGON ((((-5.941178 3...
## 27  ESP Spain Aragón Huesca Provincia   22 MULTIPOLYGON (((0.347473 41...
## 28  ESP Spain Aragón Teruel Provincia   44 MULTIPOLYGON (((0.026484 40...
```

# Representar un shape

```
peninsula <- subset(shape,!NAME_1=="Islas Canarias") #mapa sin las islas canarias  
ggplot(peninsula) +  
  geom_sf(aes(fill=NAME_1),alpha=.1,col="grey80",show.legend = FALSE) +  
  geom_sf_text(aes(label=NAME_2),size=3) + theme_bw()
```



# Mapa Coropleto

Para pintar el mapa con los datos del paro, juntamos los datos al shape:

```
paro=fread("data/paro.csv",encoding="UTF-8")
paro[,id:=sub(" ","0",format(Prov.id,width=2))]
Paro <- subset(paro,Año==2011 & Trimestre=="I") #Mujeres, 2011, primer trimestre

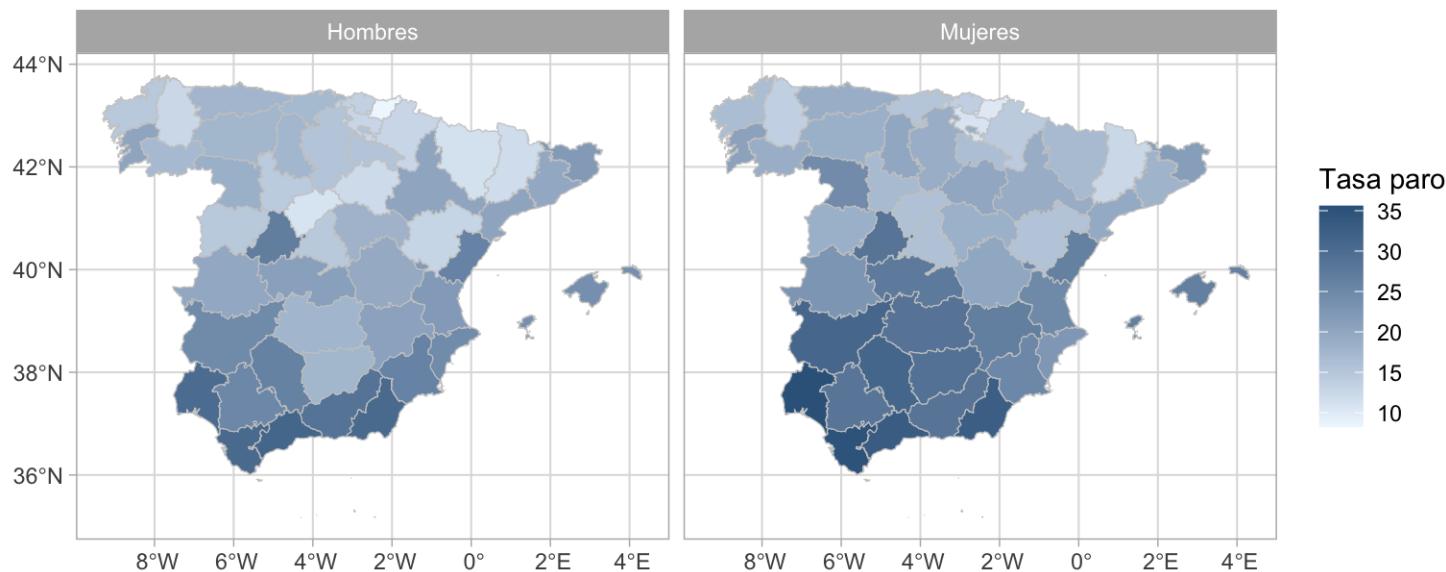
peninsula$id=peninsula$CC_2
peninsula.paro=merge(peninsula,Paro,by="id") #juntamos las dos bases
```

```
## Simple feature collection with 100 features and 5 fields
## Geometry type: MULTIPOLYGON
## Dimension: XY
## Bounding box: xmin: -9.301806 ymin: 35.17058 xmax: 4.328195 ymax: 43.79153
## Geodetic CRS: WGS 84
## First 10 features:
##   Año Trimestre Tasa.paro Prov.id Provincia           geometry
## 1 2011          I 10.79235      1  Araba/Álava MULTIPOLYGON ((((-3.131158 4...
## 2 2011          I 12.85866      1  Araba/Álava MULTIPOLYGON ((((-3.131158 4...
## 3 2011          I 26.60443      2    Albacete MULTIPOLYGON ((((-2.761982 3...
## 4 2011          I 20.73733      2    Albacete MULTIPOLYGON ((((-2.761982 3...
## 5 2011          I 24.52791      3 Alicante/Alacant MULTIPOLYGON ((((-0.457083 3...
## 6 2011          I 22.41865      3 Alicante/Alacant MULTIPOLYGON ((((-0.457083 3...
## 7 2011          I 32.68507      4   Almería MULTIPOLYGON ((((-3.030417 3...
## 8 2011          I 30.58938      4   Almería MULTIPOLYGON ((((-3.030417 3...
## 9 2011          I 26.78571      5     Ávila MULTIPOLYGON ((((-4.45362 40...
## 10 2011         I 28.47896      5     Ávila MULTIPOLYGON ((((-4.45362 40...
```

# Mapa Coropleto

Ahora podemos pintar el mapa con los datos de paro (Mujeres, 2011, primer trimestre):

```
ggplot(peninsula.paro) +  
  geom_sf(aes(fill=Tasa.paro), colour = "grey80", size = .1) +  
  facet_grid(~ Sexo) +  
  scale_fill_gradient("Tasa paro", low="aliceblue", high="steelblue4") +  
  theme_bw()
```

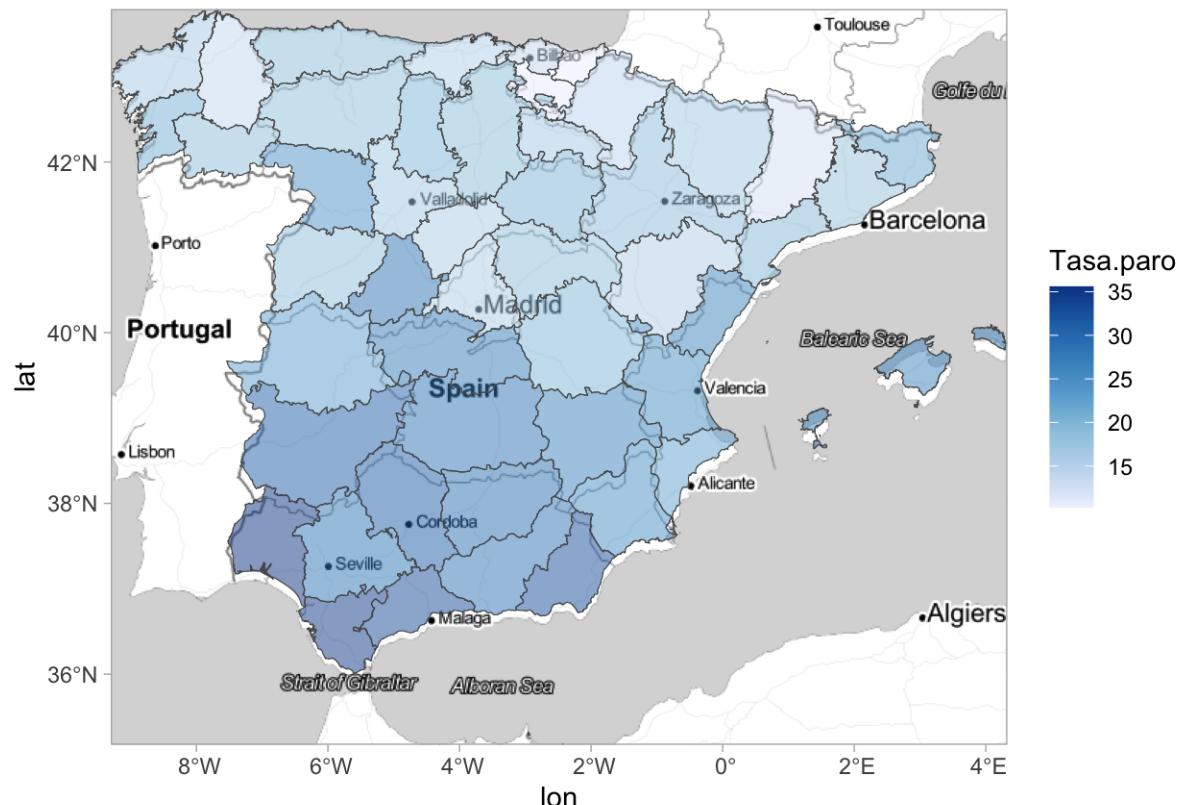


# Problema de proyección

Podemos incluso dibujar este mapa, sobre un lienzo obtenido mediante `get_map`:

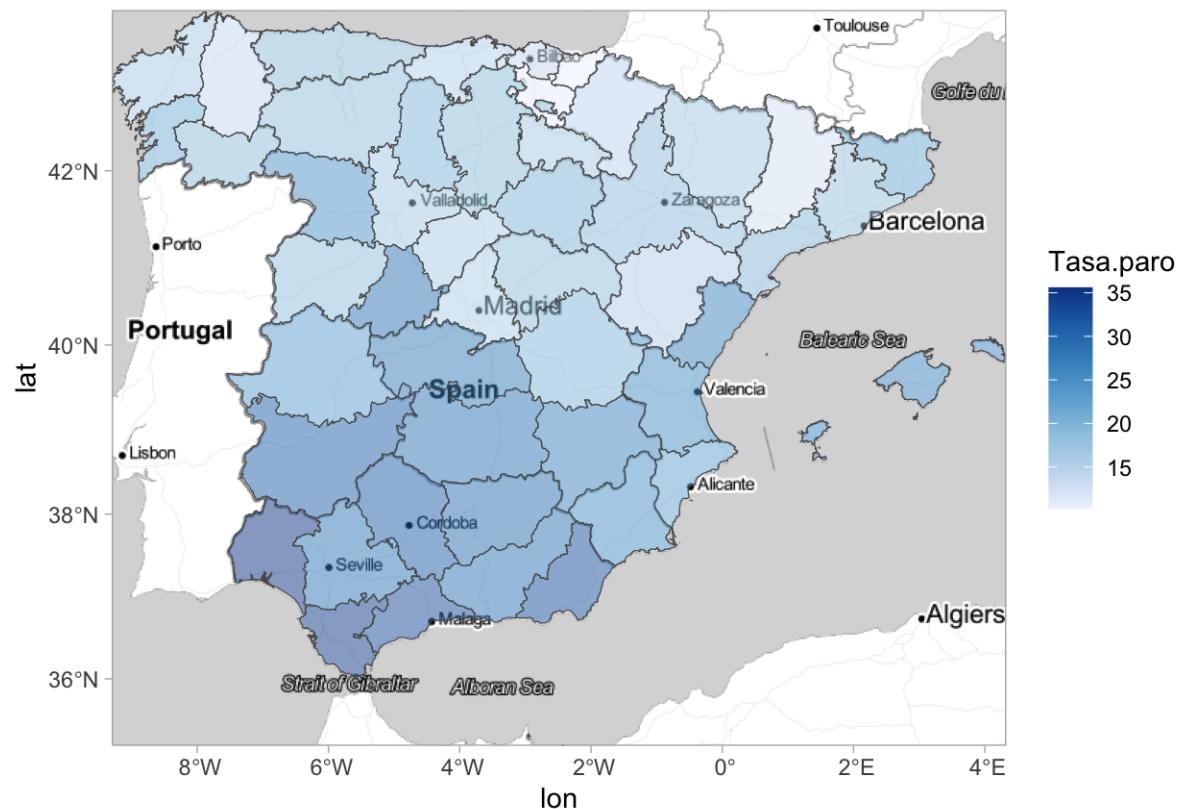
```
bb=st_bbox(peninsula)
españa <- get_stamenmap(unname(bb), zoom=6, maptype="toner-lite")
mujeres=subset(peninsula.paro, Sexo=="Mujeres")

p <- ggmap(españa) +
  geom_sf(data = mujeres, aes(fill=Tasa.paro), alpha=.5, inherit.aes=FALSE) +
  scale_fill_distiller(direction=1)
p #No muy alineado!
```



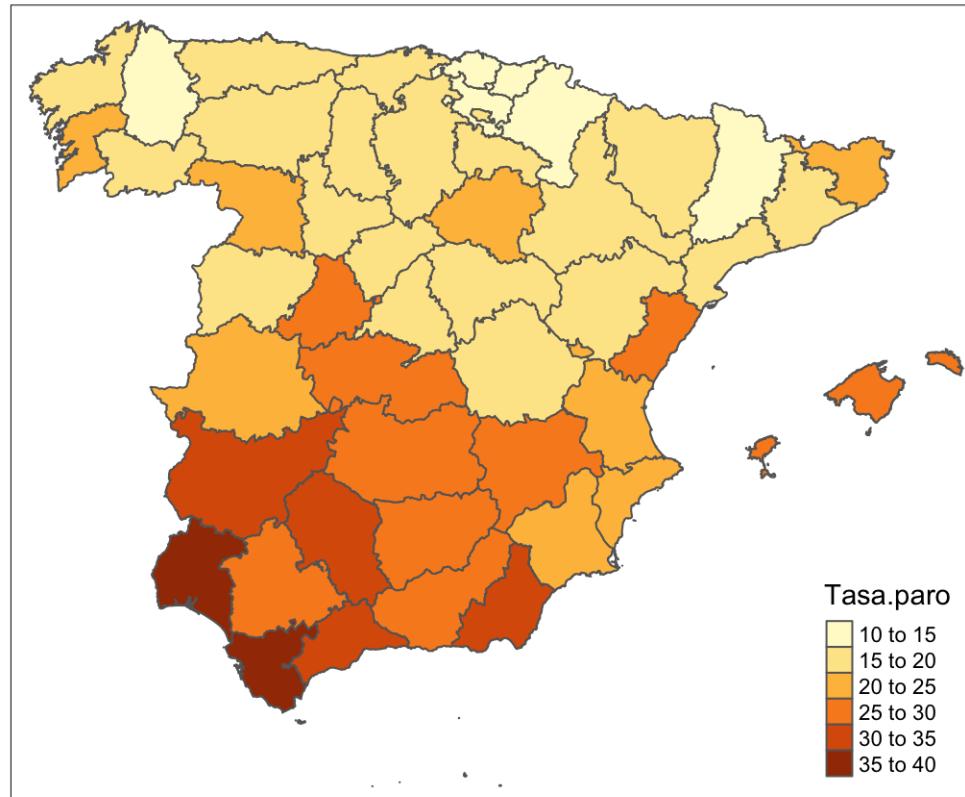
# Problema de proyección solucionado

```
p + coord_sf(  
  crs = st_crs(3857), #coordenadas en la superficie de la esfera (proyección de lienzo de google o osm)  
  default_crs = st_crs(peninsula), # coordenadas en la esfera (lon, lat) WGS84 (EPSG:4326)  
) # ahora si
```



# Lienzo dinamico

```
#lienzo dinamico con tmap  
require(tmap)  
tmap_mode("plot") #tmap_mode("view") para versión dinamica  
tm_shape(mujeres)+tm_polygons("Tasa.paro")
```



# Informes automatizados

# Rmarkdown

- El paquete `rmarkdown` es una extensión de R que permite crear documentos automatizados que combinan texto con código, tablas y gráficos generados directamente por R.
- Rmarkdown permite generar documentos dinámicos al mezclar texto formateado y resultados generados por R. Los documentos generados pueden estar en HTML, PDF, Word y muchos otros formatos.

Las ventajas de esta herramienta son numerosas:

- El código y sus resultados no están separados de los comentarios asociados a ellos
- El documento final es reproducible
- El documento se puede actualizar fácilmente, por ejemplo, si los datos de origen se han modificado.

Por lo tanto, es una herramienta muy práctica para exportar, comunicar y difundir resultados estadísticos.

# Estructura

- Markdown, un *formato* para escribir documentos en modo texto, fácil de leer y procesable para volcarlos a otros formatos
- La integración entre R y markdown

Aquí, un documento de R Markdown básico:

```
---
```

```
title: "Ejemplo de R Markdown"
output:
  html_document:
    df_print: paged
---
```

\*R Markdown\* permite mezclar :

- texto libre puesto en formato
- bloques de código de R

Los bloques de código se pueden ejecutar para incluir sus resultados en el documento,  
así por ejemplo :

```
```{r}
mean(airquality$Ozone, na.rm=TRUE)
```
```

# Compilación

Al “compilar” el documento, el texto se formatea, los bloques de código se ejecutan, sus resultados se agregan al documento y todo se transforma en uno de los formatos disponibles (html, pdf, word).

Aquí, la representación del documento anterior en formato HTML

---

## Ejemplo de R Markdown

*R Markdown* permite mezclar :

- texto libre puesto en formato
- bloques de código de R

Los bloques de código se pueden ejecutar para incluir sus resultados en el documento, así por ejemplo :

```
mean(airquality$Ozone,na.rm=TRUE)
```

```
## [1] 42.12931
```

# Rmarkdown en 15 mn

Para aprender Markdown, se recomiendan los dos siguientes ejercicios :

Crear un fichero `.Rmd` usando `File > New File > R Markdown`.

Al crear un nuevo fichero de tipo `R Markdown`, RStudio proporciona, en lugar de uno vacío, una plantilla que muestra algunas de las opciones disponibles en este formato. Eso facilita el siguiente ejercicio:

Modificar el fichero de ejemplo creado en el ejercicio anterior añadiéndole títulos de varios niveles, párrafos de texto, cursivas, negritas, enlaces, listas (numeradas y sin numerar), etc. usando como guía el Cheat Sheet del paquete.

*Compilar* el documento (p.e., pulsando el botón con la etiqueta `Knit HTML` situado encima del panel de edición de RStudio) para inspeccionar el resultado final.

Se puede también generar documentos en formato Word y PDF. Para estos formato, es necesario tener instalados los programas : MS Word, LibreOffice o similar para el primero y LaTeX para el segundo.

# Elementos de un documento Rmarkdown

## Encabezado (préambulo)

La primera parte del documento es su *encabezado*. Se encuentra al principio del documento y está delimitado por tres guiones (---) antes y después:

```
---
title: "Titulo"
author: "Nombre Apellido"
date: "2 de mayo de 2018"
output: html_document
---
```

Este encabezado contiene los metadatos del documento, como su título, autor, fecha, más una serie de opciones posibles que permiten configurar o personalizar todo el documento y su representación. Aquí, por ejemplo, la línea `output: html_document` indica que el documento generado tendrá un formato HTML.

# Texto del documento

## Listas

El cuerpo del documento consiste en texto con la sintaxis de Markdown: un marcado ligero que permite establecer niveles de títulos o formatear texto. Por ejemplo, el siguiente texto:

Este es un texto *en cursiva* y **en negrita**.

Se puede definir una lista así:

- primer elemento
- segundo elemento

Que dará la siguiente salida

Este es un texto *en cursiva* y **en negrita**.

Se puede definir una lista así:

- primer elemento
- segundo elemento

# Titulos y vinculos

Los títulos de diferentes niveles se pueden definir comenzando una línea con uno o más caracteres #:

```
# Titulo de nivel 1  
## Titulo de nivel 2  
### Titulo de nivel 3
```

Cuando se han definido los títulos, al hacer clic en el ícono *Outline* en el extremo derecho de la barra de herramientas asociada al archivo, se muestra una tabla dinámica de contenidos que permite navegar fácilmente en el documento.

La sintaxis de Markdown permite también insertar enlaces o imágenes. Por ejemplo, la siguiente sintaxis:

```
[ISCIII] (http://www.isciii.es/)
```

Dará el siguiente vínculo:

[ISCIII](http://www.isciii.es/)

# Bloques de código

Además del texto libre en formato Markdown, un documento R Markdown contiene, como su nombre indica, código R. Este código se incluye en fragmentos definidos por la siguiente sintaxis:

Como esta cadena de caracteres no es muy fácil de escribir, se puede usar *R* en el menú Insertar de RStudio, o teclear el atajo `Ctrl+Alt+i`.

Se puede dar un nombre al bloque y se indica directamente después de `r`:

```
{r nombre_del_bloque}
```

No es obligatorio, pero puede ser útil en caso de error de compilación, para identificar el bloque que causó el problema. Atención, no podemos tener dos bloques con el mismo nombre.

# Opción de salida del código echo

Además de un nombre, se puede pasar a un bloque una serie de opciones para modificar su comportamiento.

```
```{r echo = FALSE, warning = FALSE}
x <- 1:5
````
```

Una de las opciones más útiles es la opción `echo`. Por defecto `echo=TRUE`, y el bloque de código R se inserta en el documento generado:

```
x <- 1:5
print(x)
```

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

Pero, si la opción `echo=FALSE`, entonces el código R ya no se inserta en el documento, y solo se el resultado será visible:

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

# Otras opciones

Aquí, una lista de algunas de las opciones más comunes:

| opción  | valores      | descripción   |
|---------|--------------|---|
| echo    | TRUE / FALSE | Mostrar o no el código R en el documento              |
| eval    | TRUE / FALSE | Ejecutar o no el código R en tiempo de compilación    |
| warning | TRUE / FALSE | Mostrar o no las advertencias generadas por el bloque |
| message | TRUE / FALSE | Mostrar o no los mensajes generados por el bloque     |

Hay muchas otras opciones descritas en la [Guía de referencia de R Markdown](#).

# Tablas con Rmarkdown

## Tablas cruzadas

Las tablas generadas por la función `table` se muestran tal y como aparecen en la consola de R:

```
titanic<-apply(Titanic,c(1,4),sum) #Supervivencia al Titanic según clase  
titanic
```

```
##          Survived  
## Class    No Yes  
##   1st    122 203  
##   2nd    167 118  
##   3rd    528 178  
##   Crew   673 212
```

## Tablas cruzadas con kable

Su presentación se puede mejorar utilizando la función `tbl` de la extensión `kableExtra`:

```
require(kableExtra)
tbl(titanic,caption="Supervivencia a la catastrofe del Titanic según la clase") %>%
  kable_paper("striped")
```

Supervivencia a la catastrofe del Titanic según la clase

|      | No  | Yes |
|------|-----|-----|
| 1st  | 122 | 203 |
| 2nd  | 167 | 118 |
| 3rd  | 528 | 178 |
| Crew | 673 | 212 |

## Base de datos

La representación HTML de bases de datos (`data.frame`) por defecto es el contenido que aparece en consola. Este formato puede ser poco adecuado si la tabla excede una cierta dimensión. Una alternativa es usar la función `datatable` del paquete `DT`, que ofrece además interactividad:

```
require(DT)
datatable(gapminder)
```

# Resumen de base de datos

El paquete `gtsummary` permite generar automáticamente resúmenes de base de datos:

```
require(gtsummary)
temp <- subset(nhs, select=c(edad:raza, imc:pas, dth))
tbl_summary(temp, by=sexo)
```

| Characteristic | Hombre, N = 4,349 <sup>1</sup> | Mujer, N = 4,901 <sup>1</sup> |
|----------------|--------------------------------|-------------------------------|
| edad           | 58 (42, 65)                    | 58 (42, 66)                   |
| raza           |                                |                               |
| Blanco         | 3,815 (88%)                    | 4,274 (87%)                   |
| Negro          | 449 (10%)                      | 549 (11%)                     |
| Otro           | 85 (2.0%)                      | 78 (1.6%)                     |
| imc            | 25.6 (23.2, 28.2)              | 25.2 (22.2, 29.1)             |
| pulso          | 78 (68, 84)                    | 80 (72, 88)                   |
| Unknown        | 16                             | 28                            |
| pas            | 132 (121, 147)                 | 130 (116, 149)                |
| Unknown        | 6                              | 15                            |
| dth            | 1,258 (29%)                    | 887 (18%)                     |

<sup>1</sup> Median (IQR); n (%)

# Tabla de resultados de análisis de regresión

Este mismo paquete permite también elaborar de manera eficiente tablas de resultados de regresión:

```
fit = lm(lifeExp ~ continent, data=gapminder)
T1 = tbl_regression(fit)
T1
```

| Characteristic | Beta | 95% CI <sup>1</sup> | p-value |
|----------------|------|---------------------|---------|
| continent      |      |                     |         |
| Africa         | —    | —                   |         |
| Americas       | 16   | 15, 17              | <0.001  |
| Asia           | 11   | 10, 12              | <0.001  |
| Europe         | 23   | 22, 24              | <0.001  |
| Oceania        | 25   | 22, 29              | <0.001  |

<sup>1</sup> CI = Confidence Interval

# Tabla de resultados de análisis de regresión

Este mismo paquete permite también elaborar de manera eficiente tablas de resultados de regresión:

```
fit2 = lm(lifeExp ~ log(gdpPercap,2) + continent, data=gapminder)
T2 = tbl_regression(fit2)
tbl_merge(list(T1,T2),c("Efectos crudos","Efectos ajustados"))
```

| Characteristic    | Efectos crudos |                     |         | Efectos ajustados |                     |         |
|-------------------|----------------|---------------------|---------|-------------------|---------------------|---------|
|                   | Beta           | 95% CI <sup>1</sup> | p-value | Beta              | 95% CI <sup>1</sup> | p-value |
| continent         |                |                     |         |                   |                     |         |
| Africa            | —              | —                   |         | —                 | —                   |         |
| Americas          | 16             | 15, 17              | <0.001  | 7.0               | 5.9, 8.1            | <0.001  |
| Asia              | 11             | 10, 12              | <0.001  | 5.9               | 5.0, 6.8            | <0.001  |
| Europe            | 23             | 22, 24              | <0.001  | 9.6               | 8.4, 11             | <0.001  |
| Oceania           | 25             | 22, 29              | <0.001  | 9.2               | 6.2, 12             | <0.001  |
| log(gdpPercap, 2) |                |                     |         | 4.5               | 4.2, 4.7            | <0.001  |

<sup>1</sup> CI = Confidence Interval

# Referencias

Este curso está basado en las siguientes referencias:

- *R para profesionales de los datos*, Carlos Bellosta, 2017,  
[https://datanalytics.com/libro\\_r](https://datanalytics.com/libro_r)
- El libro [R for data science](#) disponible en línea, que contiene un capítulo dedicado a R Markdown .

Y también muy inspirado de estos otros libros:

- *Data Visualisation with R*, Thomas Rahlf, 2014,  
<http://www.datavisualisation-r.com/>
- *R Graph Cookbook* , Hrishi Mittal, 2011