Visualización de datos en R

Ernesto Mislej

December 12, 2016

Previously...

Para los que vienen hoy :) este taller está basado en R, si no lo tienen instalado, descarguenlo desde acá: http://cran.r-project.org

También les recomiendo instalar RStudio, el sitio de descarga es: http://rstudio.com

Continuaremos con el dataset *breast-cancer* que estuvimos trabajando la semana pasada. Recordemos que estuvimos realizando algunas transformaciones sobre el dataset crudo, así como lo descargamos, para poder trabajarlo en R. Lo fuimos trabajando y lo guardamos en el file *datos_limpios_1.RDS*. Lo levantamos para verlo...

```
df1 <- readRDS("datos_limpios_1.RDS")

dim(df1)
```

```
## [1] 286  10
```

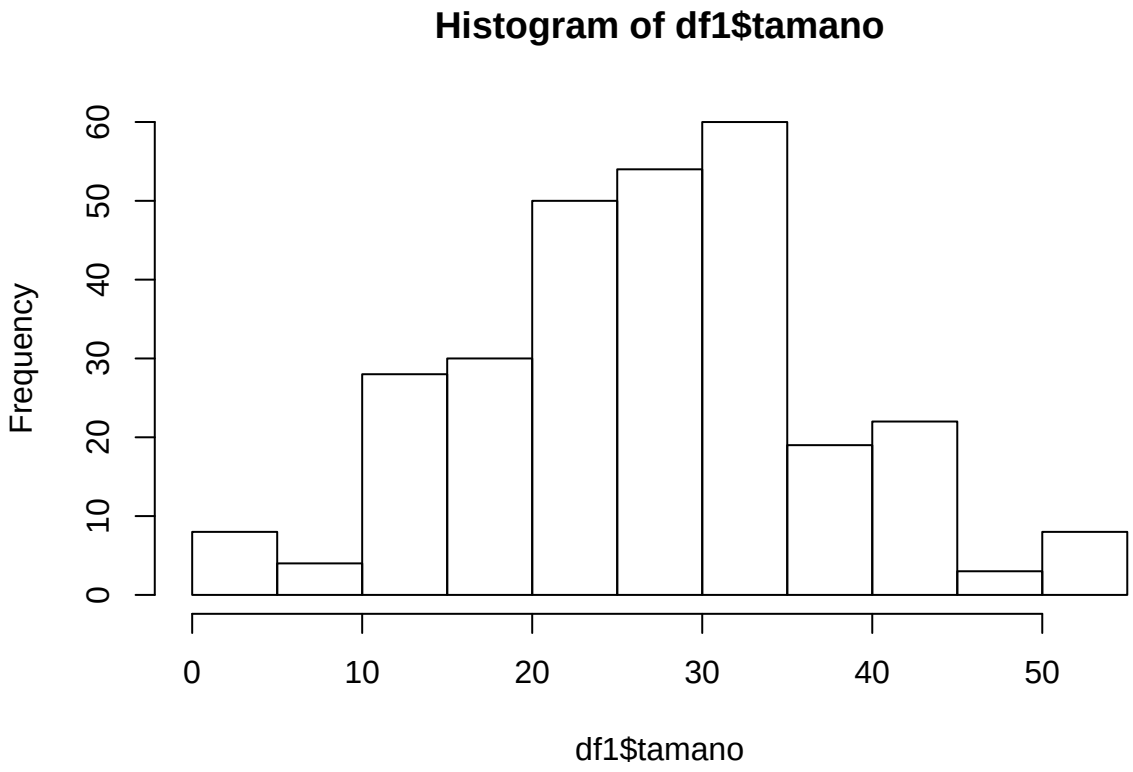
```
str(df1)
```

```
## 'data.frame':    286 obs. of  10 variables:
##  $ edad       : num  3 4 4 3 3 4 4 3 3 3 ...
##  $ menopausia : Factor w/ 3 levels "ge40","lt40",..: 3 1 1 3 3 3 1 3 3 1 ...
##  $ tamano     : num  17 17 37 37 32 27 42 12 2 42 ...
##  $ nodulos    : num  1 1 1 1 4 4 1 1 1 16 ...
##  $ capsula    : logi  TRUE FALSE FALSE TRUE TRUE FALSE ...
##  $ grado      : int  3 1 2 3 2 2 3 2 2 2 ...
##  $ mama       : Factor w/ 2 levels "left","right": 2 2 1 2 1 2 1 1 2 2 ...
##  $ cuadrante  : Factor w/ 5 levels "central","left_low",..: 3 1 2 2 5 3 3 3 4 3 ...
##  $ radiot     : logi  FALSE FALSE FALSE TRUE FALSE TRUE ...
##  $ recurrencia: logi  TRUE FALSE TRUE FALSE TRUE FALSE ...
```

Gráficos básicos

Veamos cómo hacer un histograma de una varible numérica como es el tamaño.

```
hist(df1$tamano)
```



Flojito...

Llegaremos?

Mi intención es lograr reproducir gráficas complejas (o no tanto) como la siguiente

Sobre ggplot2

Nos centraremos en el uso de la librería *ggplot2*.

```
install.packages("ggplot2")
```

ggplot2 es una librería gráfica montada sobre R.

Toma como referencia una metodología de visualización de datos llamada `The Grammar of Graphics, (Wilkinson, 2005)`, vamos a verla a medida que presentemos los ejemplos. La idea es describir los mapeos visuales para poder armar visualizaciones complejas sin preocuparnos por la parte dificil.

Algunas ventajas:

- consistente con la metodología `grammar of graphics`
- permite especificar los gráficos con un nivel alto de abstracción
- sumamente flexible
- mantiene una estética elegante y profesional
- cuenta con una gran comunidad de usuarios y muy activa

Por otro lado, con *ggplot2* no podremos

- realizar gráficas en 3D (odio los gráficos en 3D, me parecen horribles y muy fácilmente caés en problemas de ocultamiento)
- grafos, metáfora de nodos y aristas

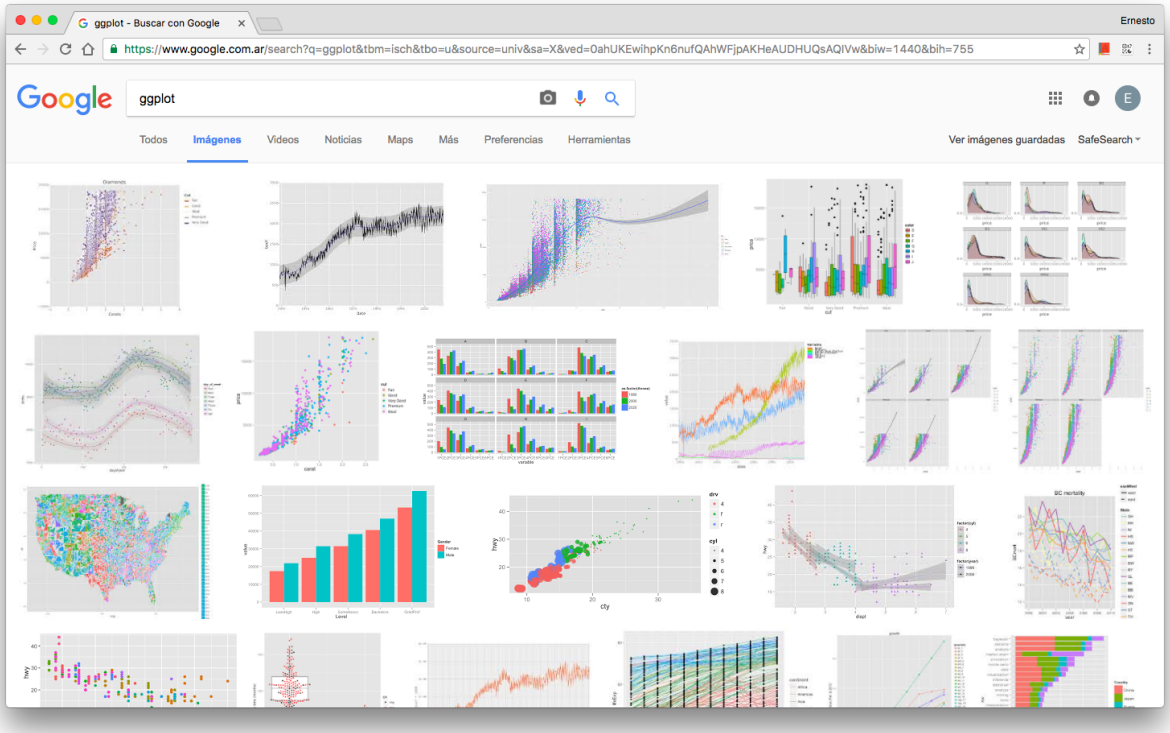


Figure 1:

- gráficos interactivos

En qué consiste The Grammar Of Graphics?

La idea básica consiste en especificar de manera independiente las componentes del gráfico, como si fuesen bloques y luego combinarlas. Las componenetes que define *The Grammar Of Graphics* son:

- los datos (obvio! sin ellos no somos nada)
- los mapeos estéticos (ejes, posición, colores, tamaños, etc.)
- objetos (como el caso de los boxplots...)
- transformaciones (log, sqrt, exp)
- escalas
- sistema de coordenadas (cartesianas, polares)
- facetado (muy útil para sumar variables al análisis)

La estructura del `ggplot`

La función `ggplot()` se usa para inicializar la estructura básica de los gráficos. Se ve algo así:

```
ggplot(data = <default data set>,
       aes(x = <default x axis variable>,
           y = <default y axis variable>,
           ... <other default aesthetic mappings>),
       ... <other plot defaults>) +
```

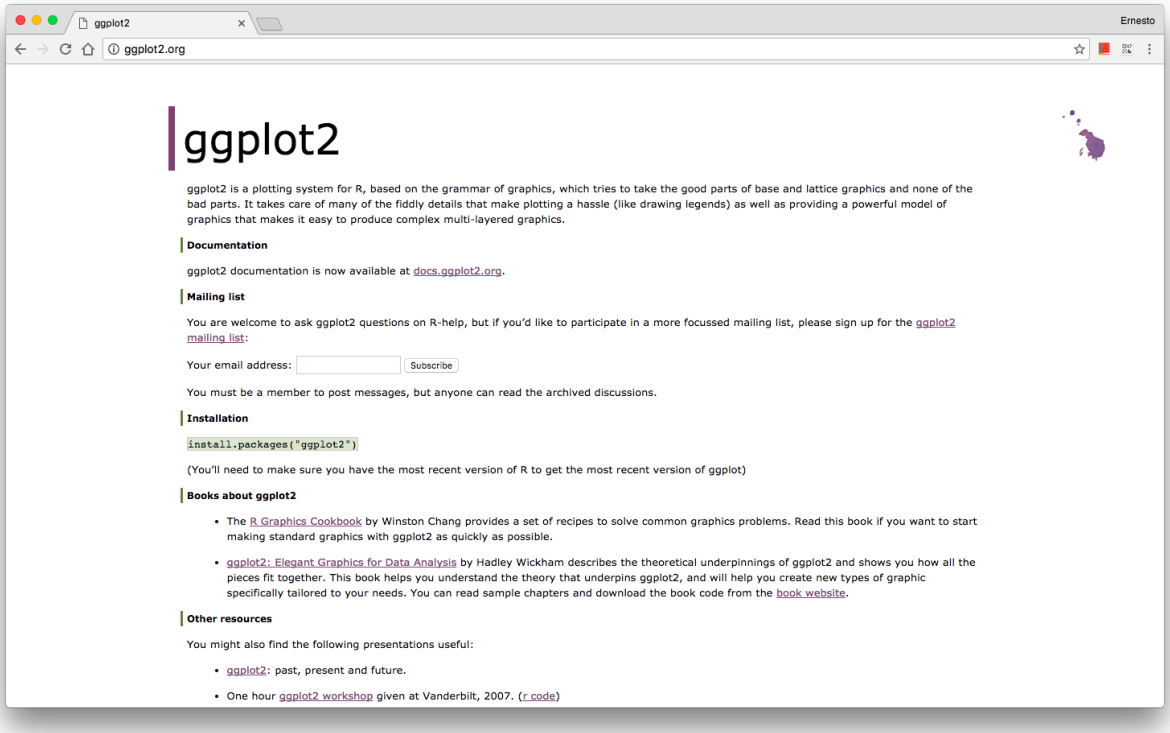


Figure 2:

```
      geom_<geom type>(aes(size = <size variable for this geom>,
                           ... <other aesthetic mappings>),
                       data = <data for this point geom>,
                       stat = <statistic string or function>,
                       position = <position string or function>,
                       color = <"fixed color specification">,
                       <other arguments, possibly passed to the _stat_ function) +

  scale_<aesthetic>_<type>(name = <"scale label">,
                           breaks = <where to put tick marks>,
                           labels = <labels for tick marks>,
                           ... <other options for the scale>) +

  theme(plot.background = element_rect(fill = "gray"),
        ... <other theme elements>)
```

No teman!

(Espero) Se va a entender a medida que vayamos transcurriendo el taller. Además existe una manera más cómoda de especificar las componentes de los gráficos y es utilizando el operador `+`.

`ggplot2` vs. la librería básica de R

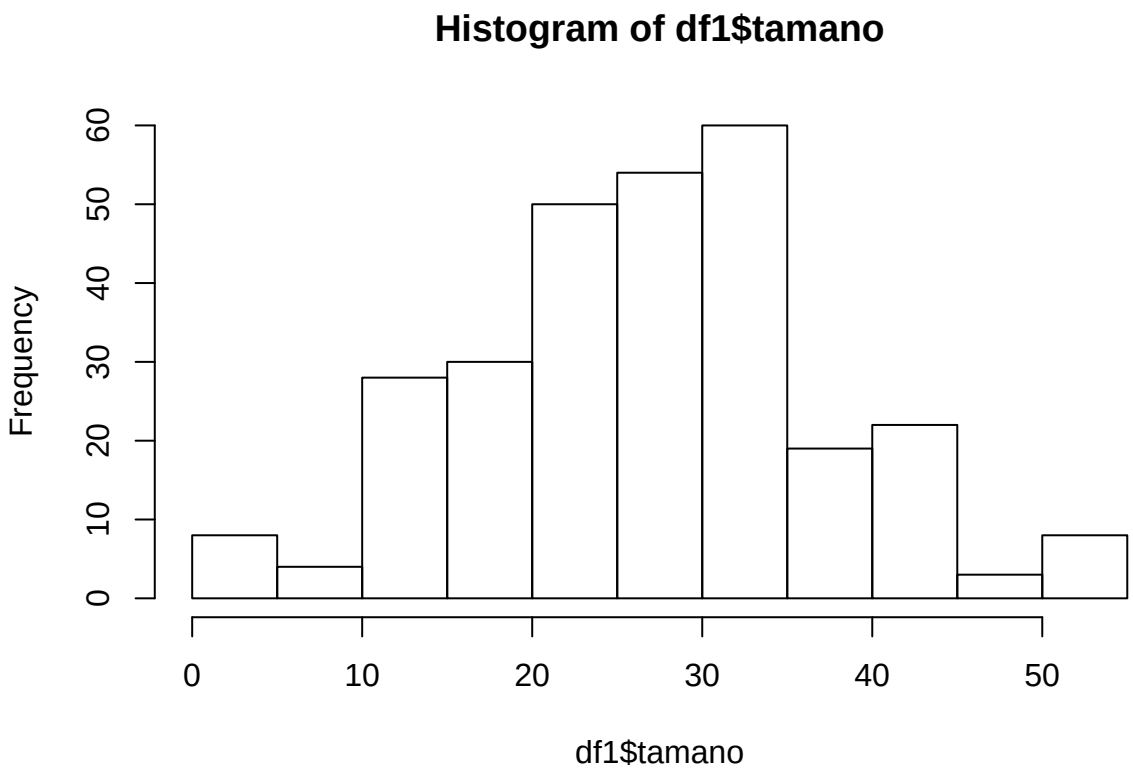
Comparada con la librería básica de R, el `ggplot2`:

- es más dificil para gráficas enlatadas

- es más simple para gráficas complejas
- los datos tienen que estar siempre en dataframes

Versión R:

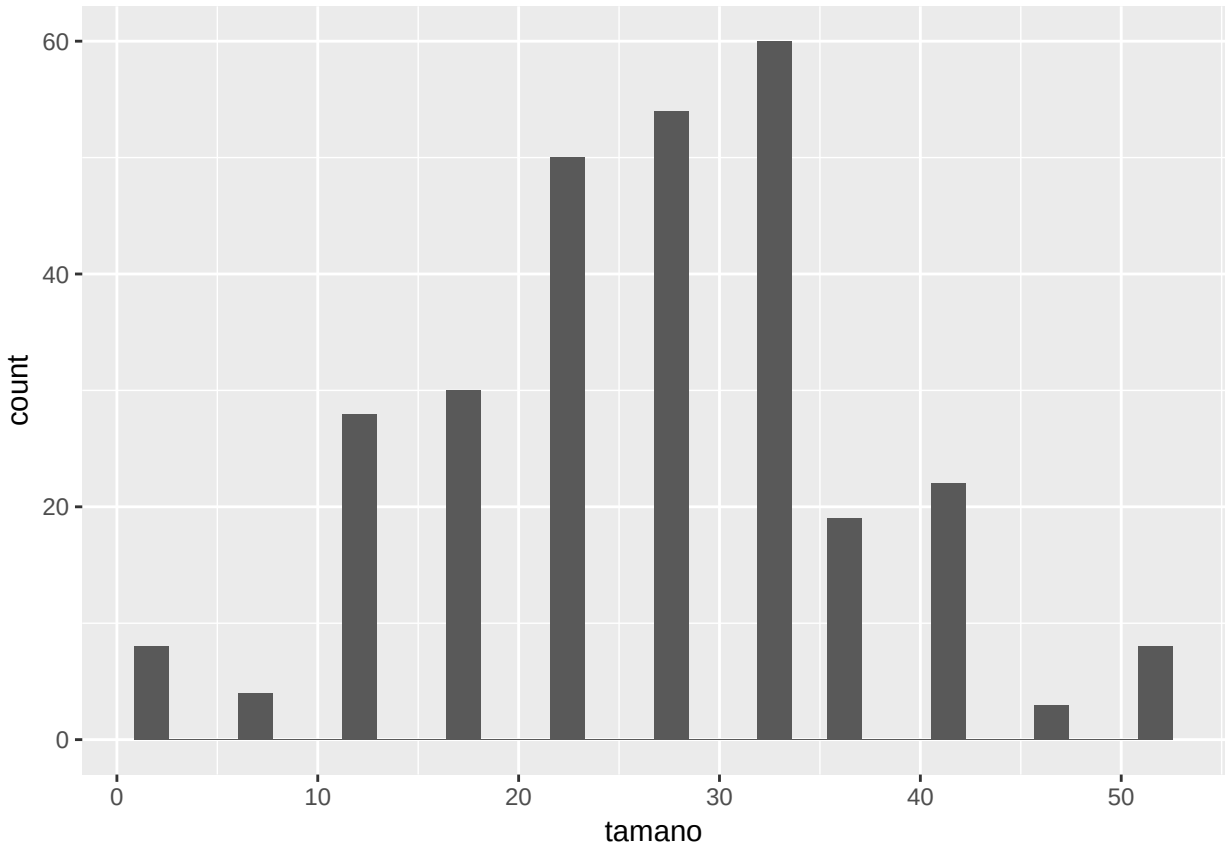
```
hist(df1$tamano)
```



Versión ggplot2:

```
library(ggplot2)
ggplot(df1, aes(x = tamano)) +
  geom_histogram()
```

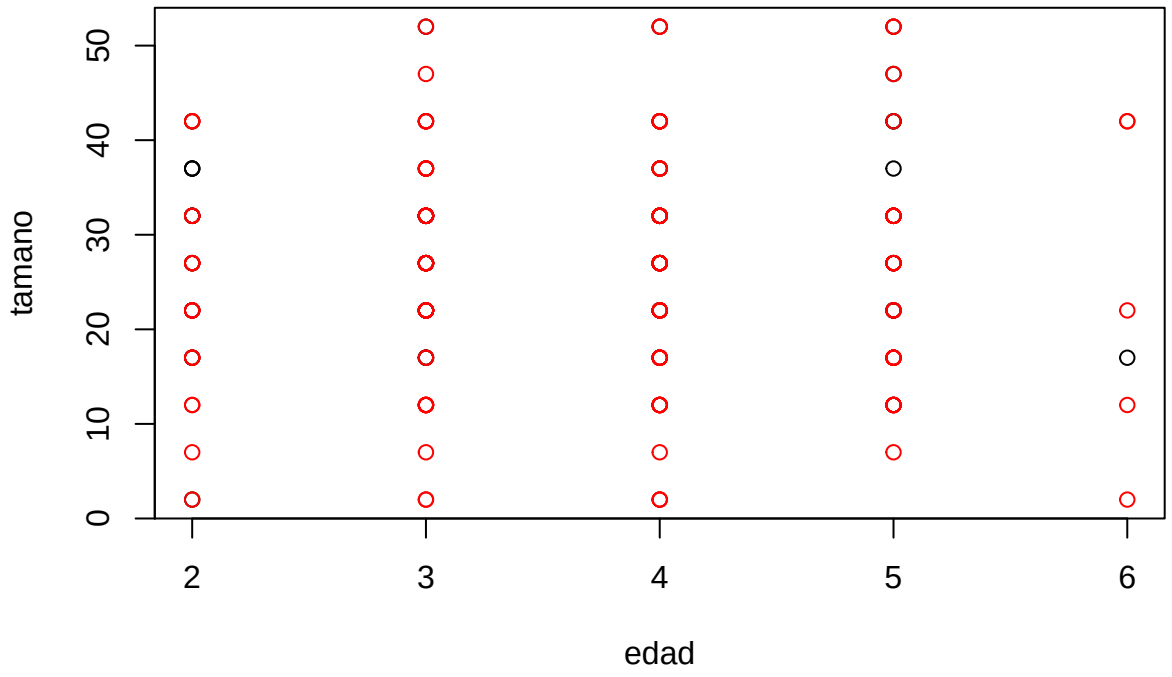
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



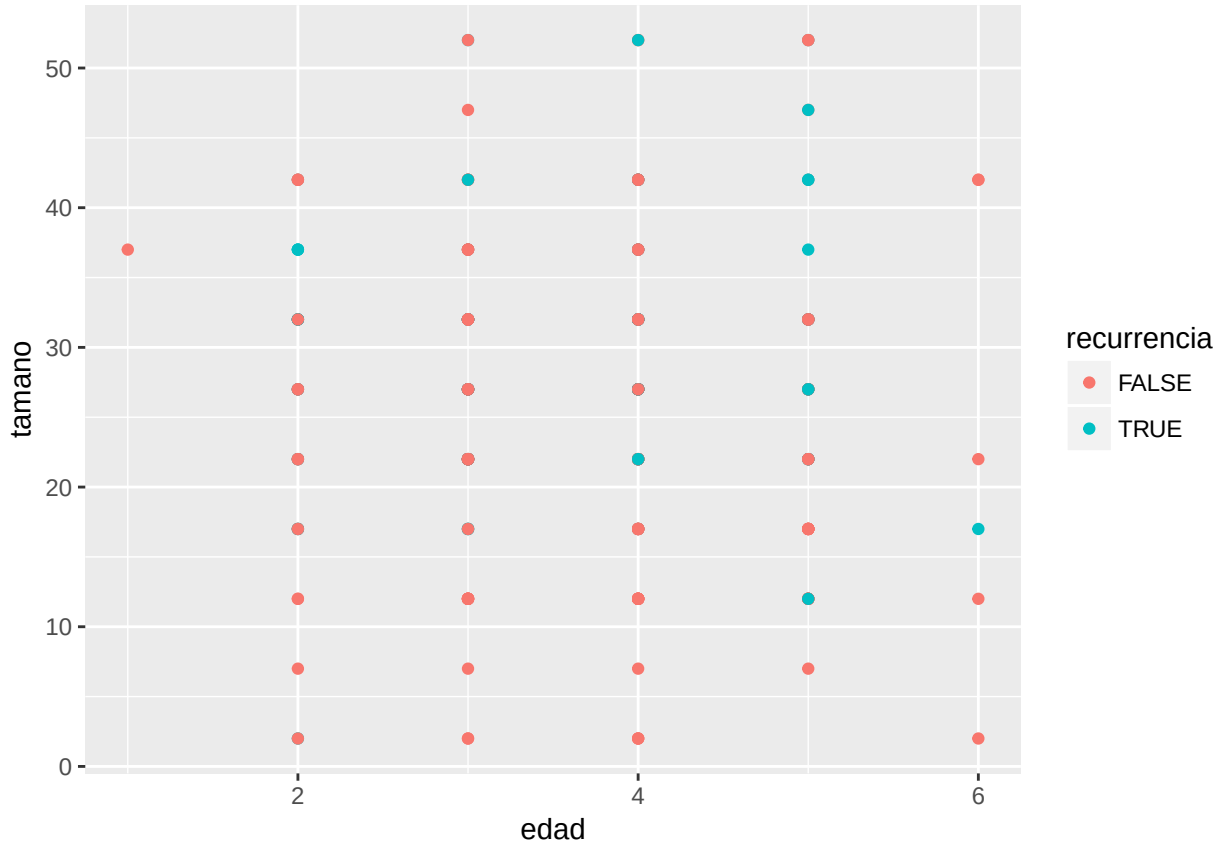
La librería básica es un poco más simple (aunque más fea)

Veamos un gráfico más complejo:

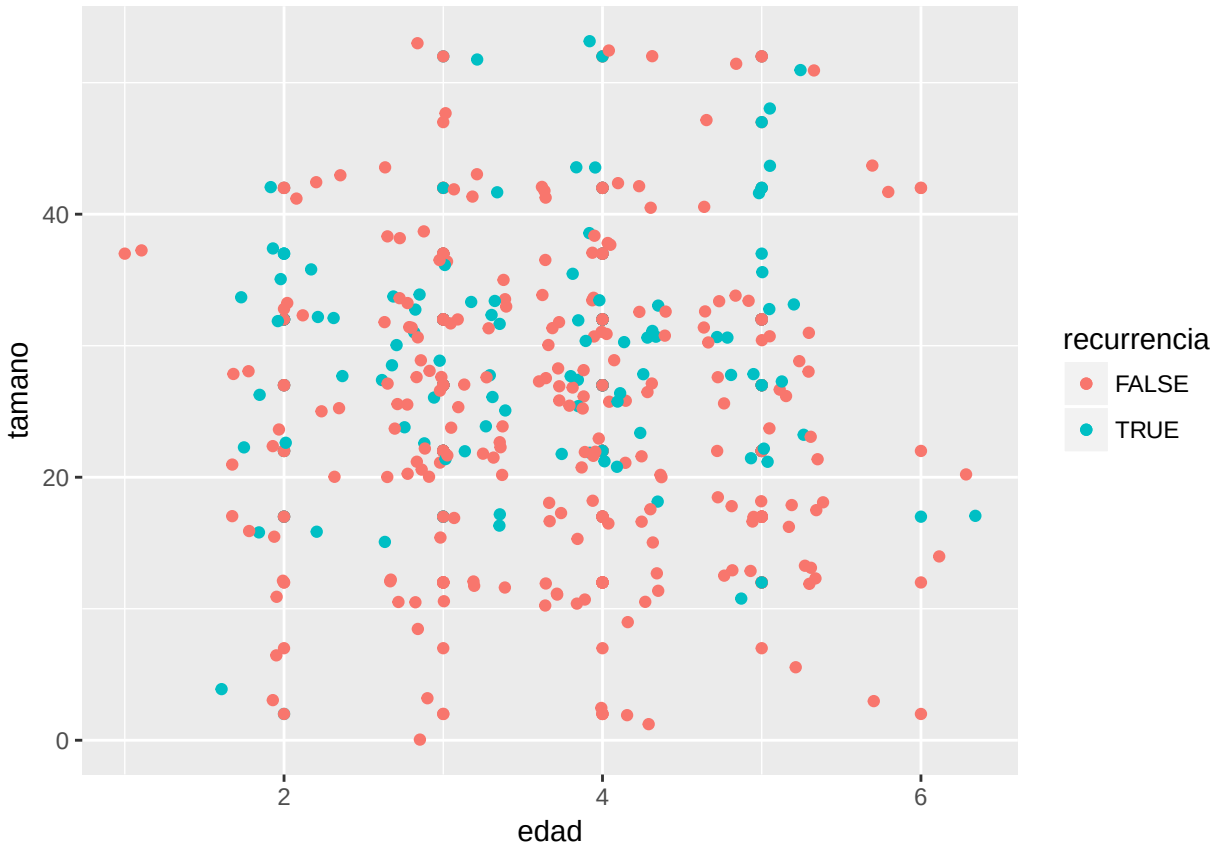
```
plot(tamano ~ edad,
  col="black",
  data=subset(df1,recurrencia))

points(tamano ~ edad,
  col="red",
  data=subset(df1,!recurrencia))
```




```
ggplot(df1, aes(x=edad, y=tamano, color=recurrencia)) +
  geom_point()
```




```
ggplot(df1, aes(x=edad, y=tamano, color=recurrencia)) +
  geom_point() +
  geom_jitter()
```



Fijensé qué fácil fue incluir la categoría recurrencia como color. Y un *jitter* para evitar el solapamiento

`ggplot2` wins!

Aes & Geom

Las variables visuales que maneja `ggplot2` son las siguientes:

- posición espacial
- color
- relleno
- forma del objeto
- tamaño
- tipo de linea

No todas las variables visuales tienen sentido en todas las visualizaciones. Los mapeos visuales se describen con la función `aes()`.

Los objetos se definen con el grupo de funciones `geom_()`. Los más clásicos son los siguientes:

- puntos `geom_point`, usados en los *scatter plots*.
- líneas `geom_line`, para series de tiempo.
- barras `geom_bar`, para gráficos de barras.
- boxplot `geom_boxplot`, para idem!
- ...

Para ver la lista completa de objetos se puede ir a la documentación:

```
help.search("geom_", package = "ggplot2")
```

También comenzar a tipear `geom_<tab>` en el *Rstudio* y ver el listado.

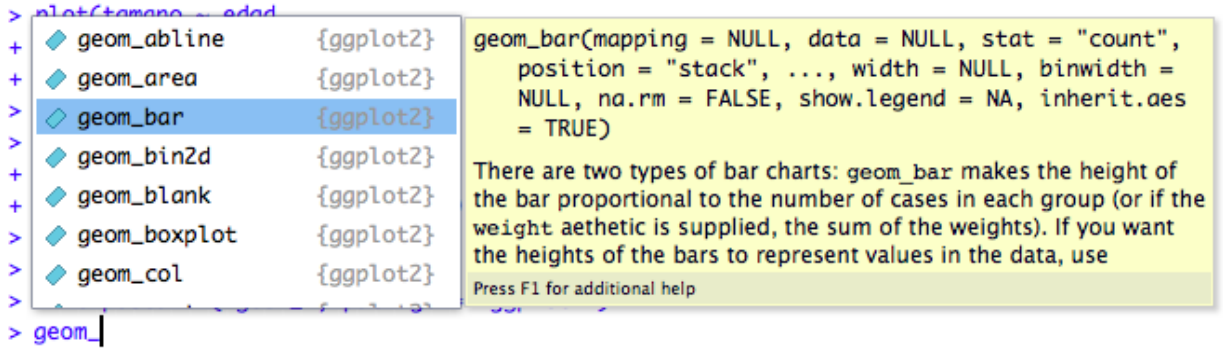
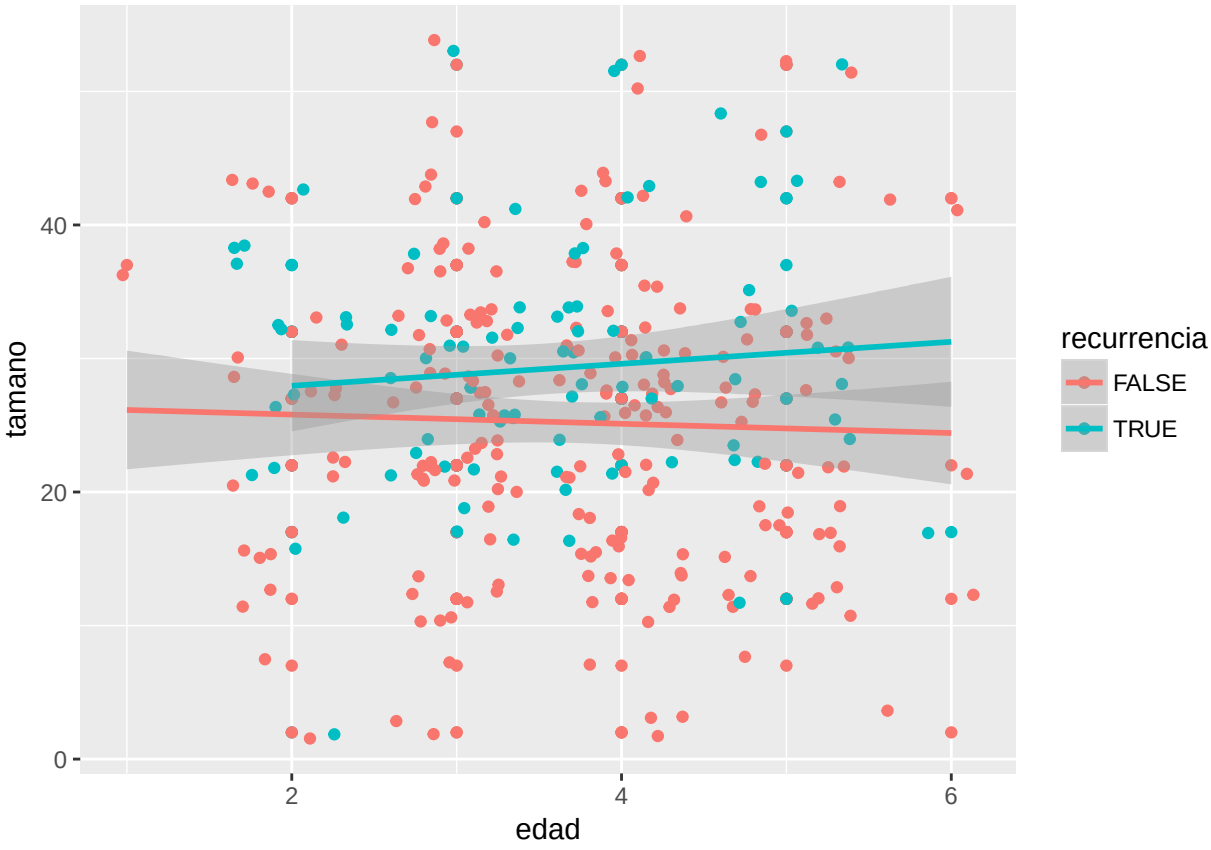


Figure 3:

Las visualizaciones pueden tener más de una *geom*. Se pueden ir agregando fácilmente usando el operador `+`.

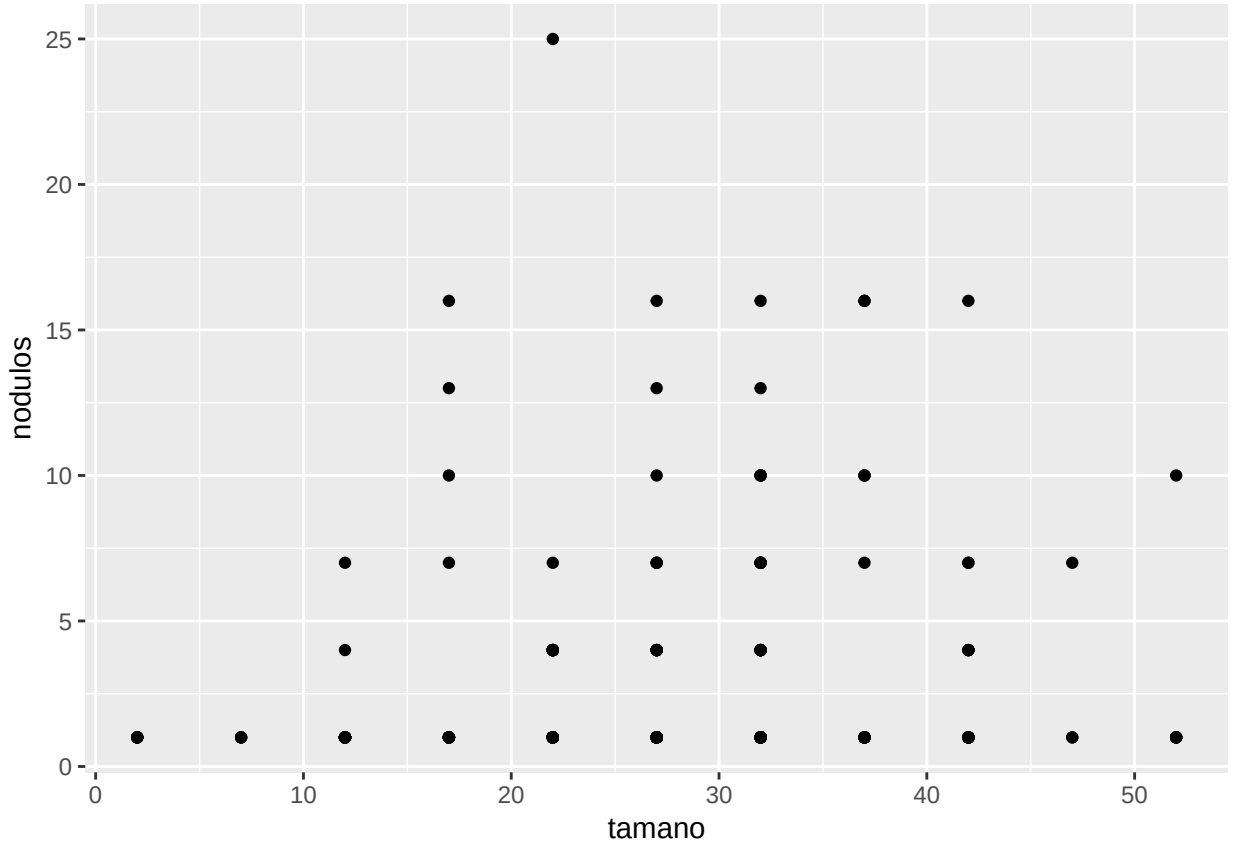
```
ggplot(df1, aes(x=edad, y=tamano, color=recurrencia)) +
  geom_point() +
  geom_jitter() +
  geom_smooth(method=lm)
```



Scatterplots

Los scatterplots son tipos de gráficas donde objetos serán puntos que se disponen en un diagrama de ejes. Mayormente utilizamos coordenadas cartesianas y una escala lineal, pero sabemos que podemos usar ejes radiales y escala logaritmicas, entre otros. La manera de invocar estos gráficos es mediante la función `geom_point`; lo que obligatoriamente necesitamos es indicar cuáles son las variables que irán a los ejes x e y.

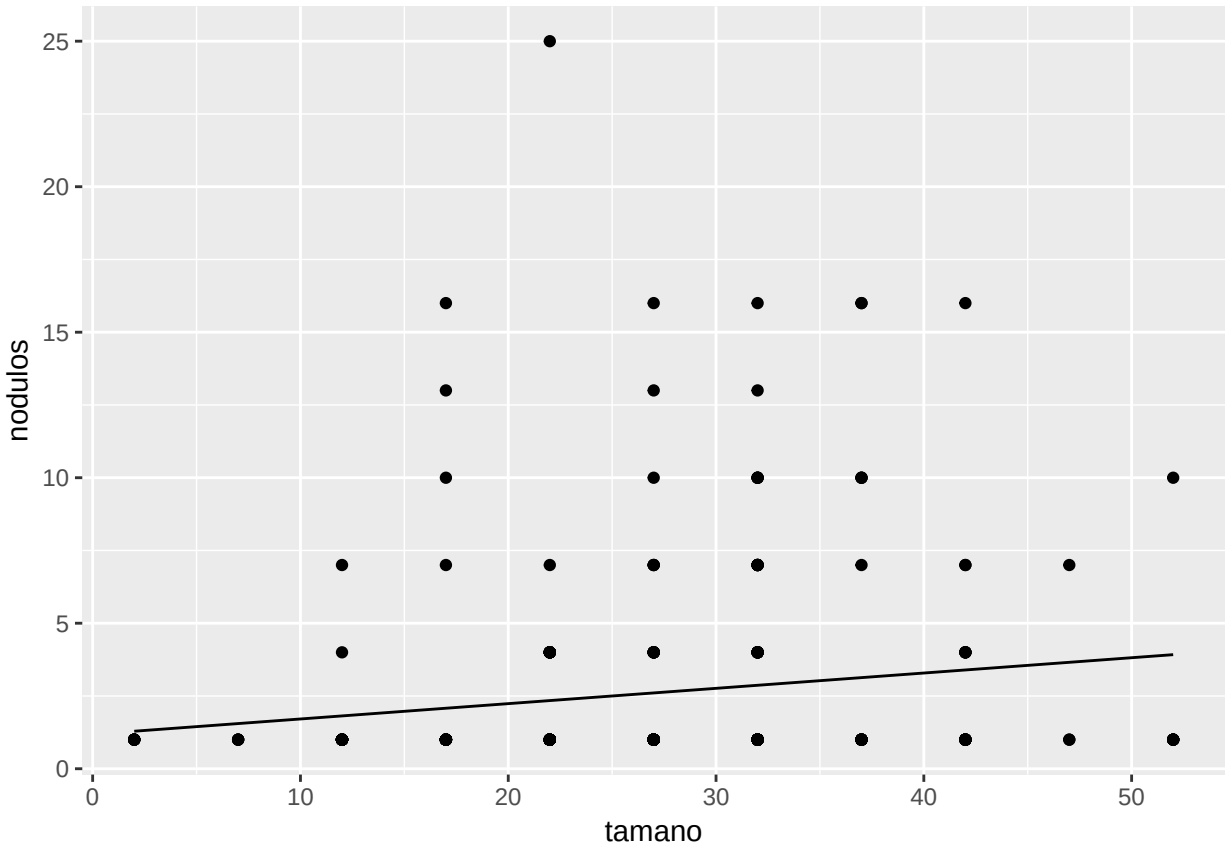
```
ggplot(df1, aes(x=tamano, y=nodulos)) +
  geom_point()
```



Líneas

`ggplot` nos permite sumar elementos al gráfico, como por ejemplo la línea de regresión:

```
df1$reg_line <- predict(lm(nodulos ~ tamano, data=df1))

p1 <- ggplot(df1, aes(x=tamano, y=nodulos)) +
  geom_point() +
  geom_line(aes(y=reg_line))

p1
```

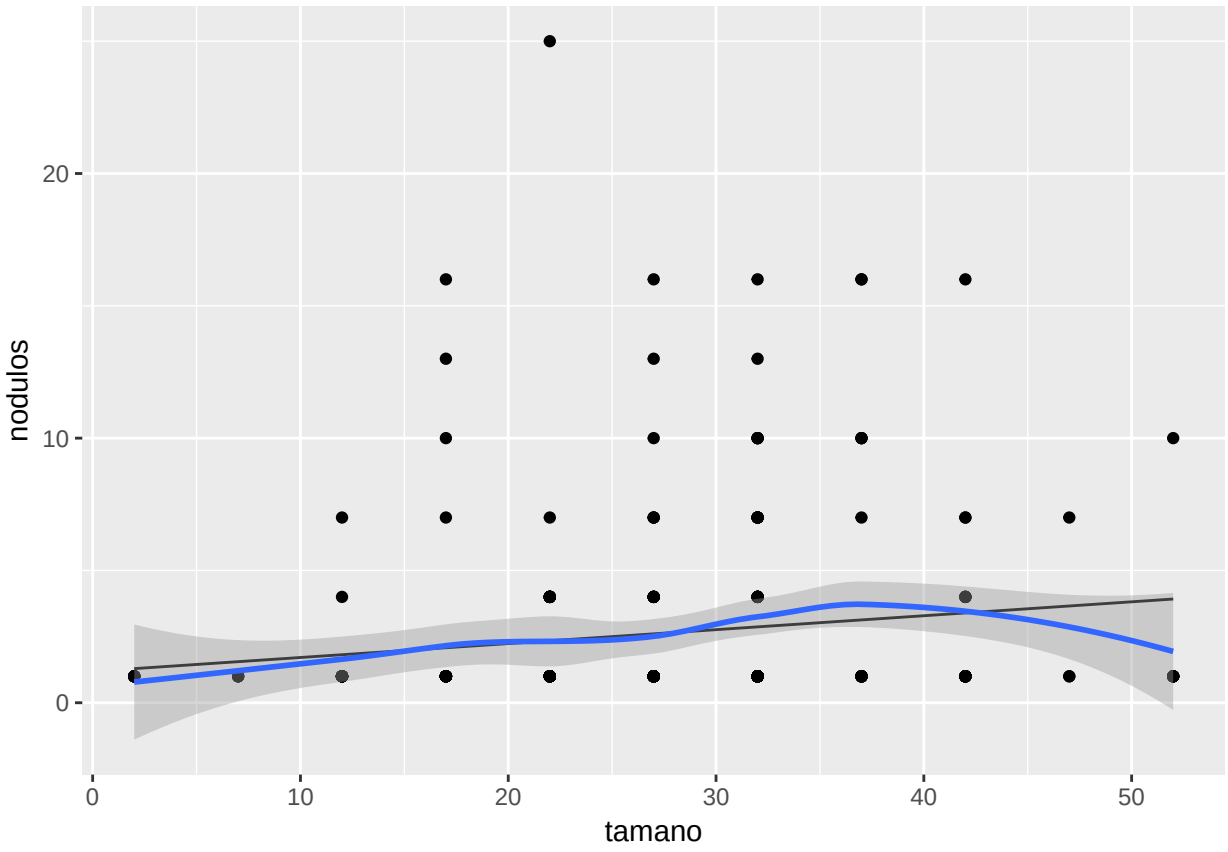


Smoothers

Algunos de los *geom* como los scatterplots tienen el problema del overplotting o solapamiento y no podemos conocer realmente la dispersión de los puntos. El método `geom_smooth` nos ayuda en este sentido mostrando una cinta sobre los puntos.

```
  p1 + geom_smooth()
```

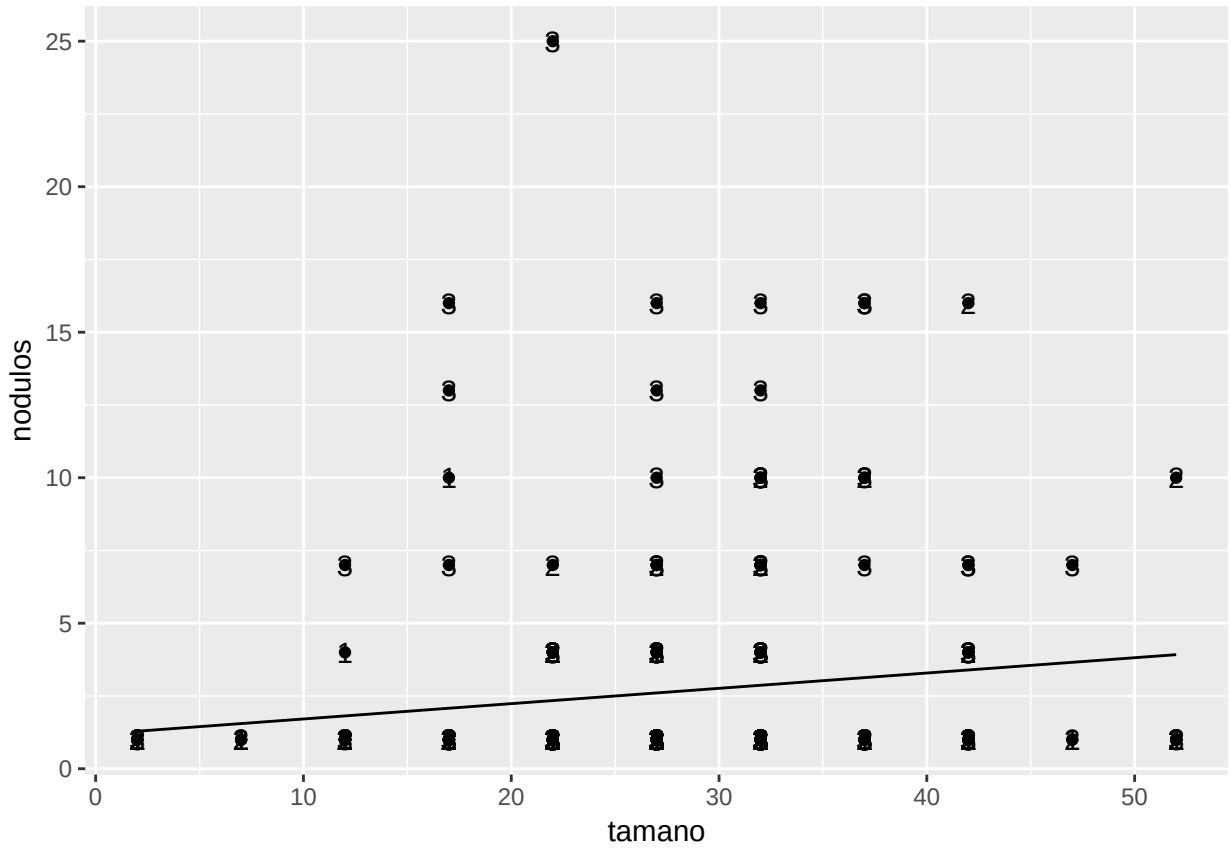
```
## `geom_smooth()` using method = 'loess'
```



Etiquetas sobre los puntos

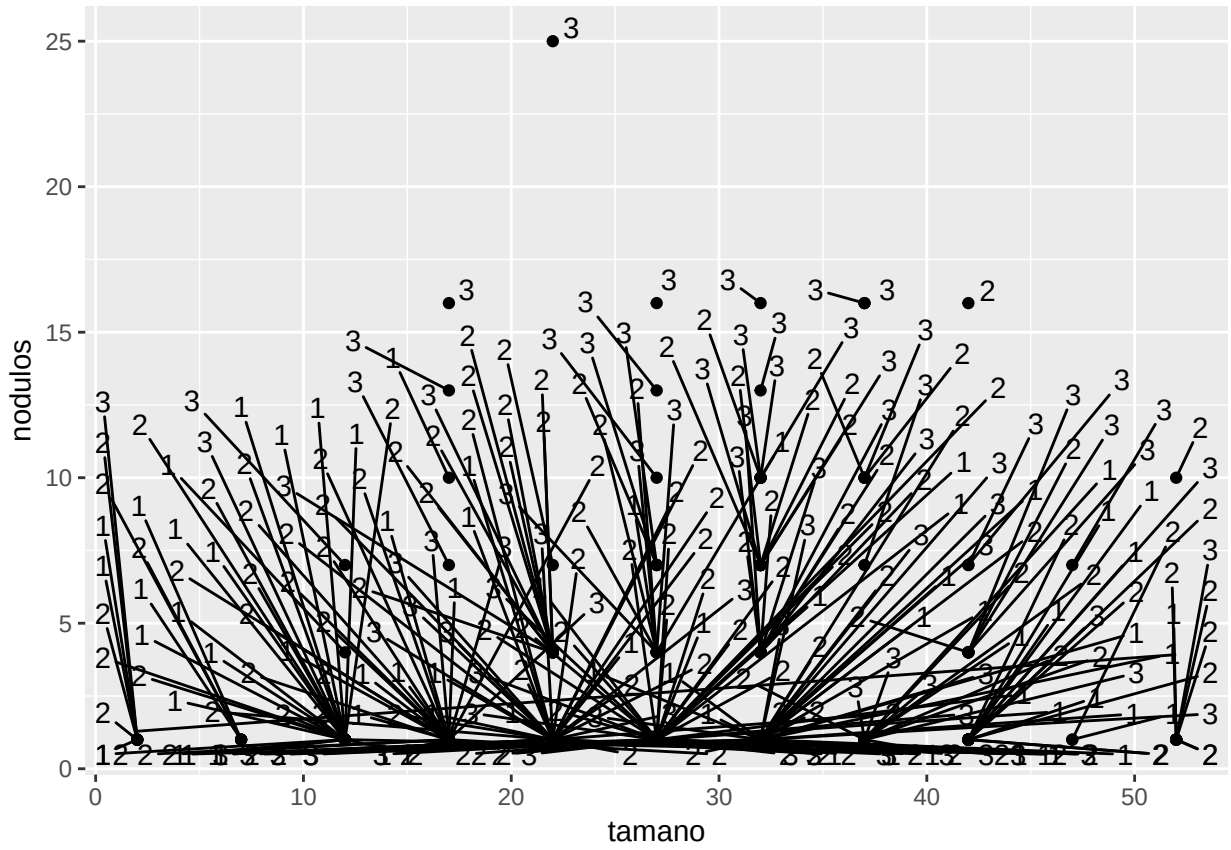
De esta manera se incluyen etiquetas sobre los puntos. No parece ser una buena idea

```
p1 + geom_text(aes(label=grado))
```



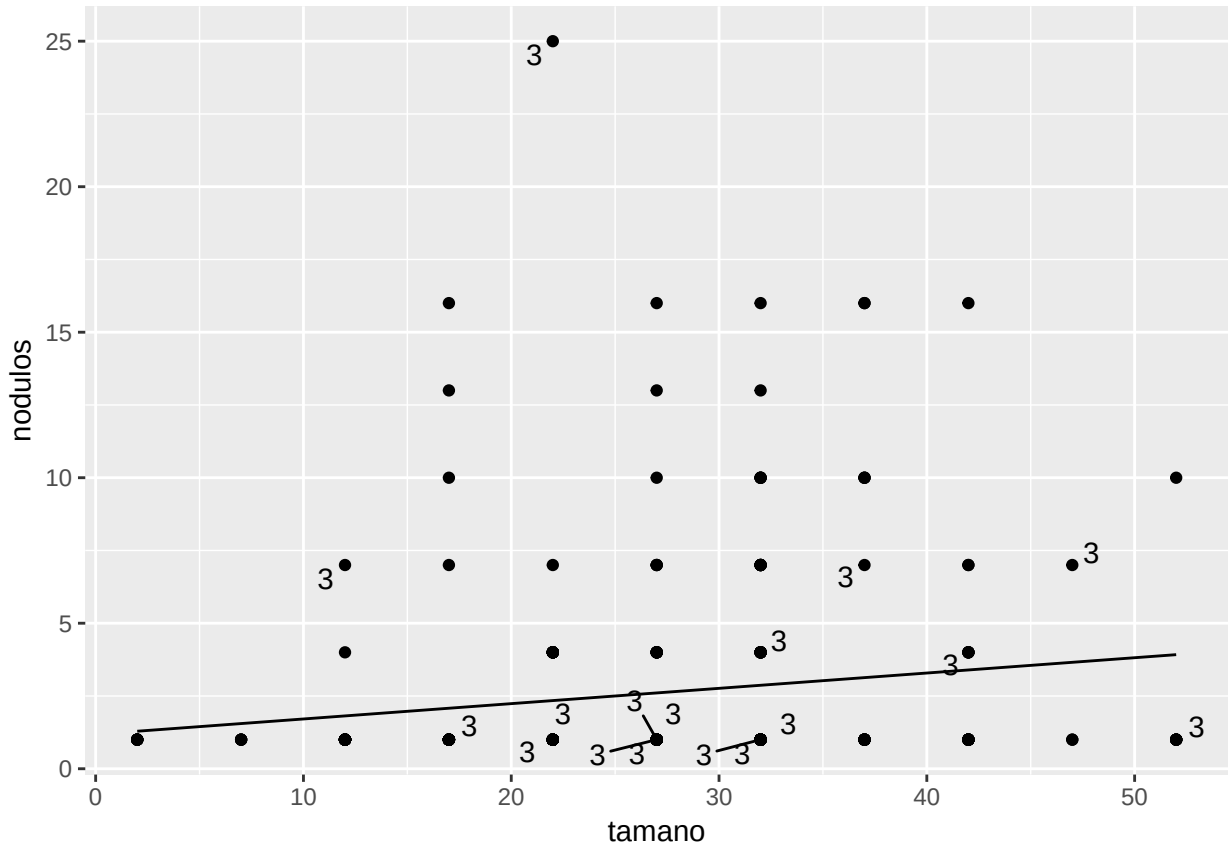
Existe un paquete que (intenta) ayudarnos con este problema

```
#install.packages("ggrepel")
library("ggrepel")

p1 + geom_text_repel(aes(label=grado))
```



No es un buen ejemplo... Pero si solo queremos marcar algún subconjunto...

```
p1 +
  geom_text_repel(
    aes(label=grado),
    data = subset(df1, grado==3 & edad>=5))
```



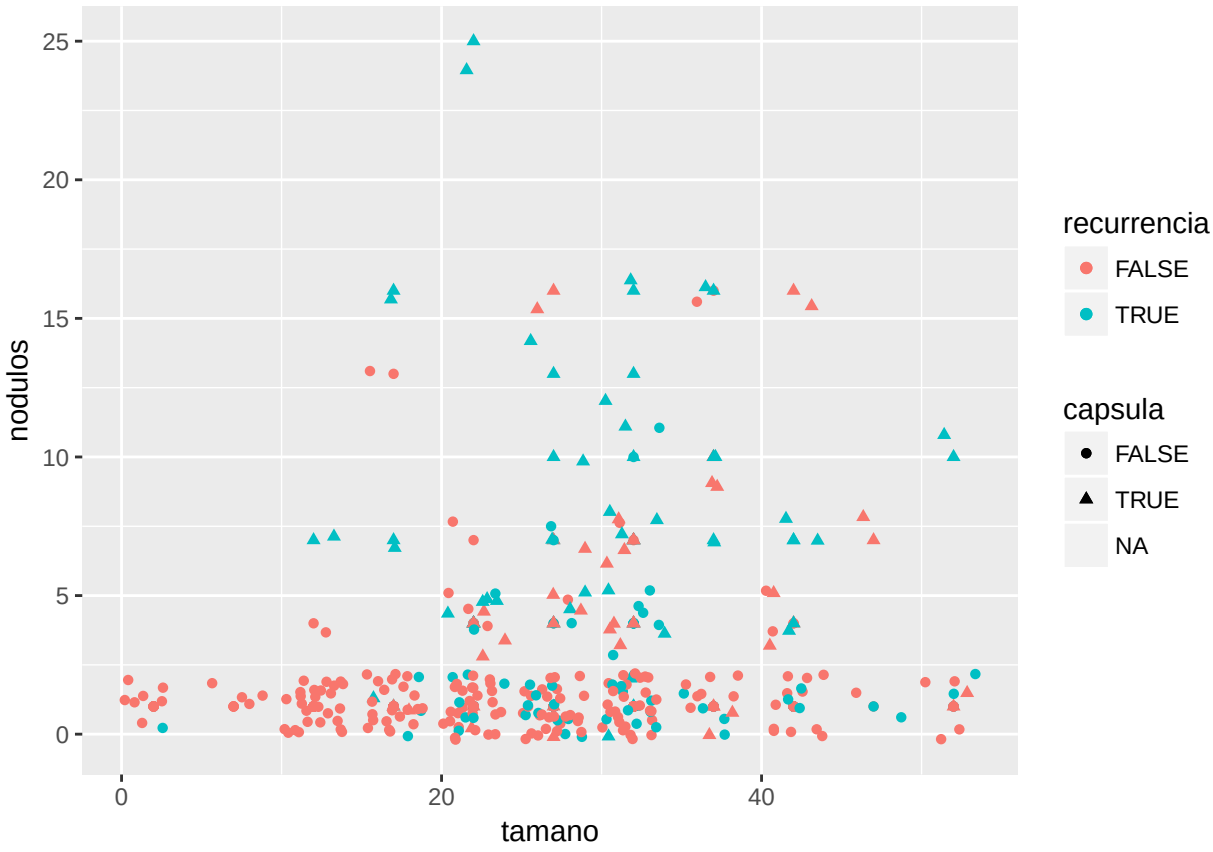
Otros mapeos visuales

Sumemos color y forma a este lío!

```
ggplot(df1, aes(x=tamano, y=nodulos, shape=capsula, color=recurrencia)) +
  geom_point() +
  geom_jitter()
```

```
## Warning: Removed 8 rows containing missing values (geom_point).
```

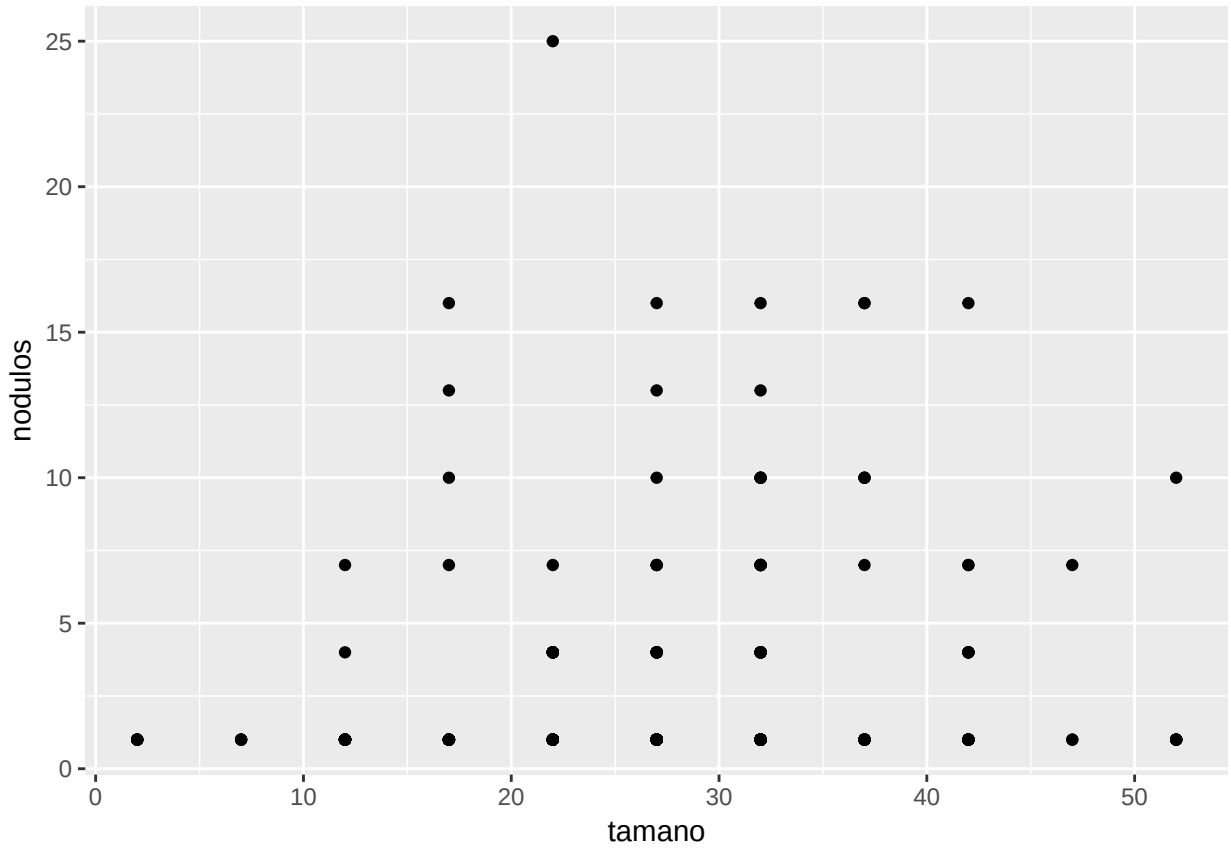
```
## Warning: Removed 8 rows containing missing values (geom_point).
```



Ejercitación

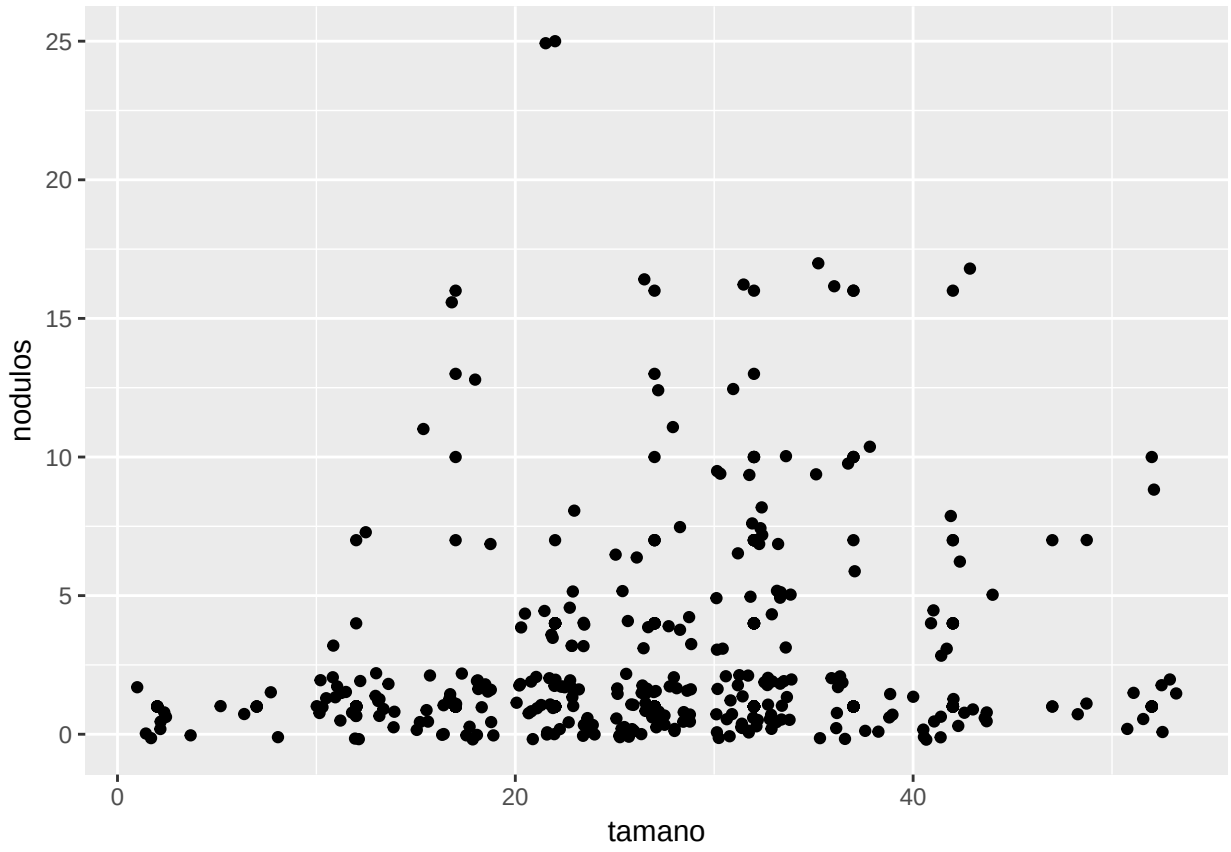
1. Crear un scatterplot con 2 variables numéricas.
2. Incluir jitter.
3. Utilizar el color según la *recurrencia*.
4. Crear boxplots del tamaño según la recurrencia.
5. Sumar una nube de puntos sobre los boxplots.
6. Crear un scatterplot con 2 variables numéricas.

```
ggplot(df1, aes(x=tamano, y=nodulos)) +
  geom_point()
```



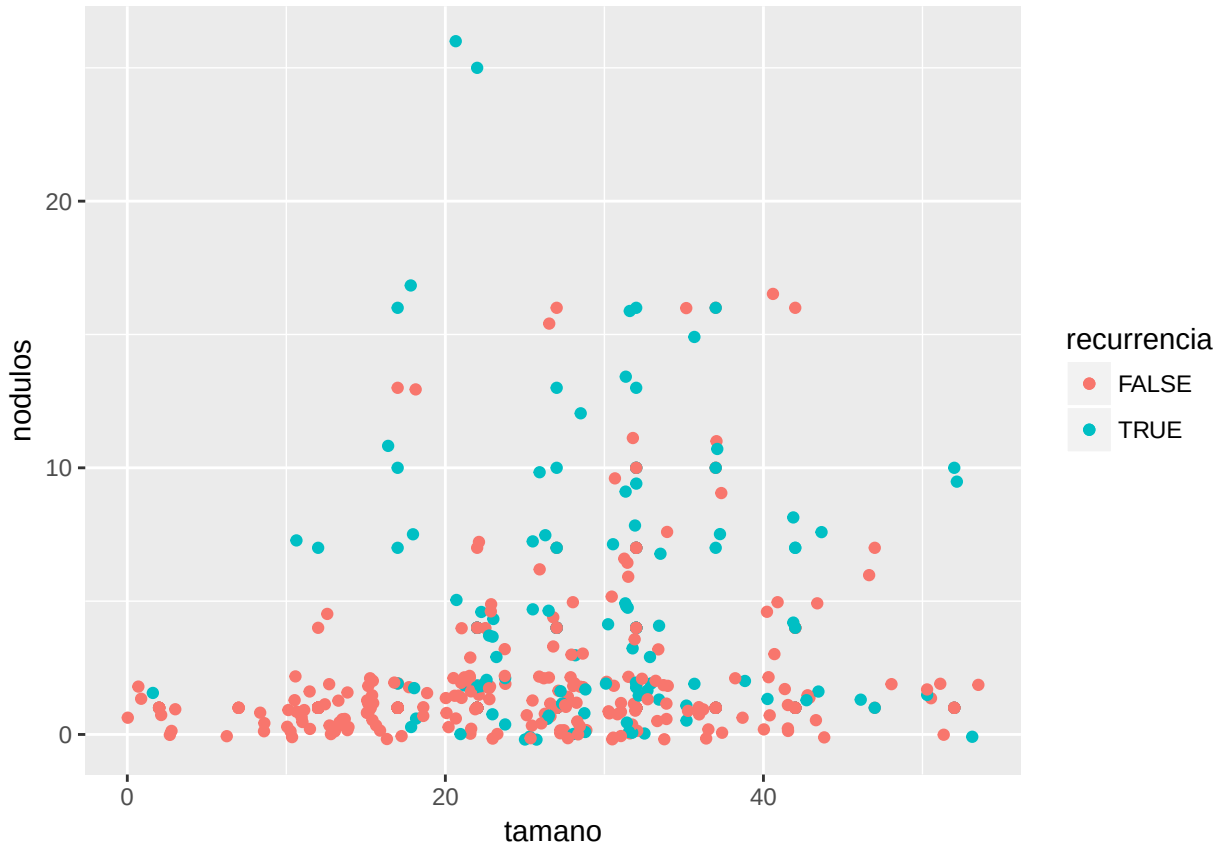
2. Incluir jitter.

```
ggplot(df1, aes(x=tamano, y=nodulos)) +
  geom_point() +
  geom_jitter()
```



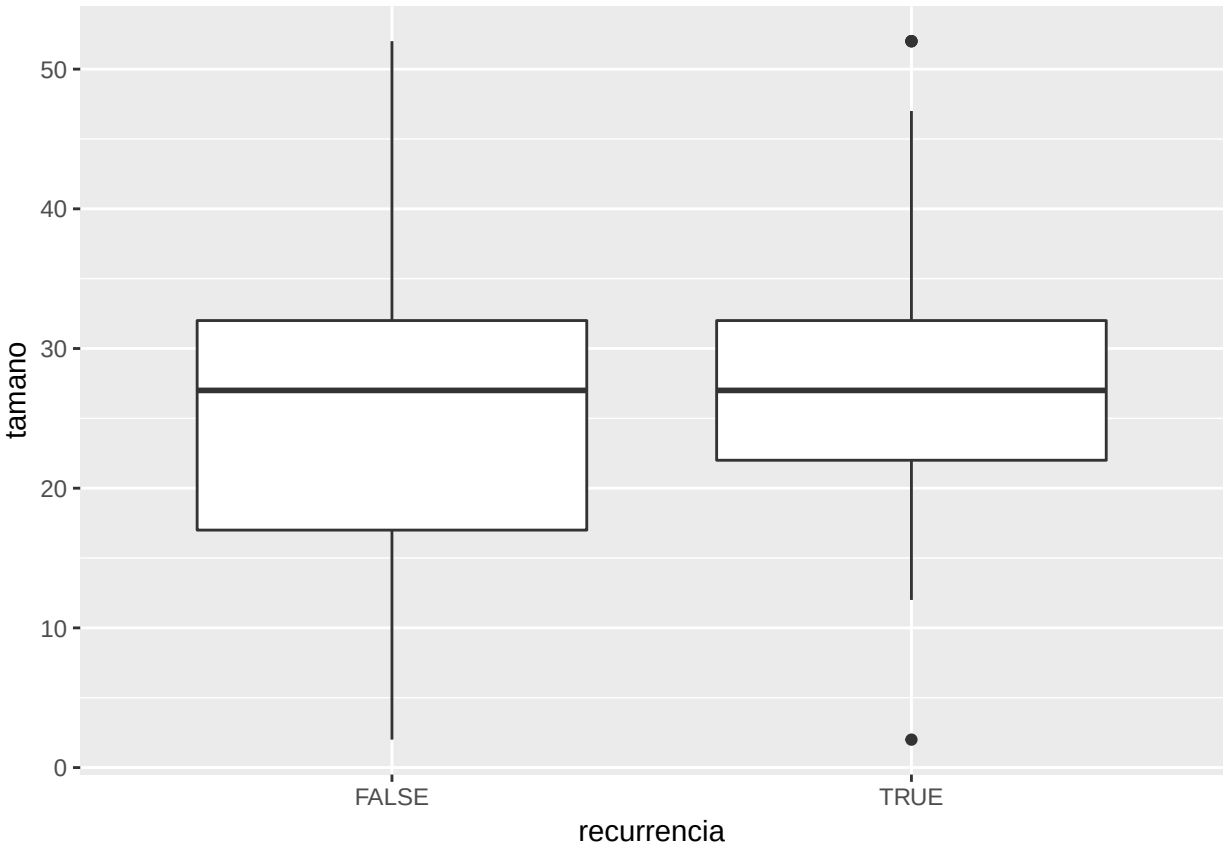
3. Utilizar el color según la *recurrencia*

```
ggplot(df1, aes(x=tamano, y=nodulos,color=recurrencia)) +
  geom_point() +
  geom_jitter()
```



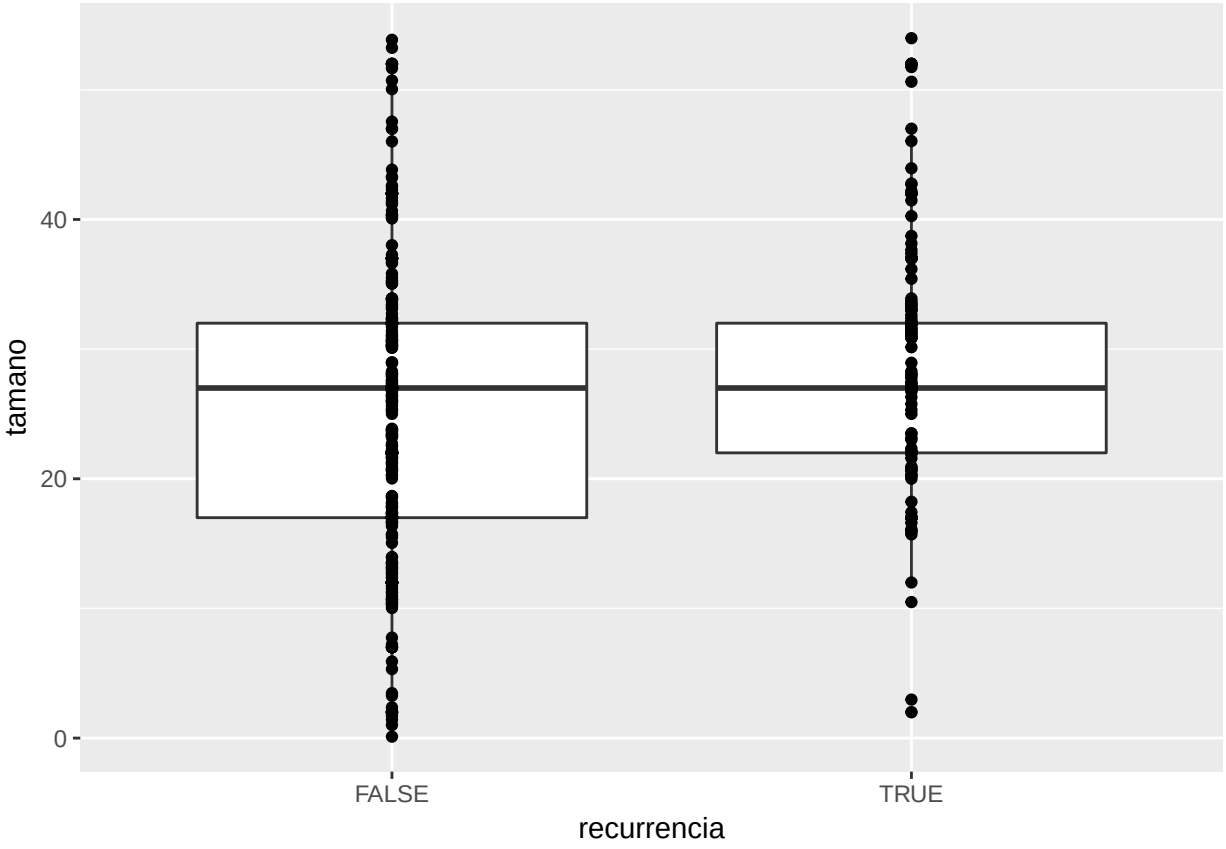
4. Crear boxplots del tamaño según la recurrencia.

```
ggplot(df1, aes(x=recurrencia, y=tamano)) +
  geom_boxplot()
```



5. Sumar una nube de puntos sobre los boxplots.

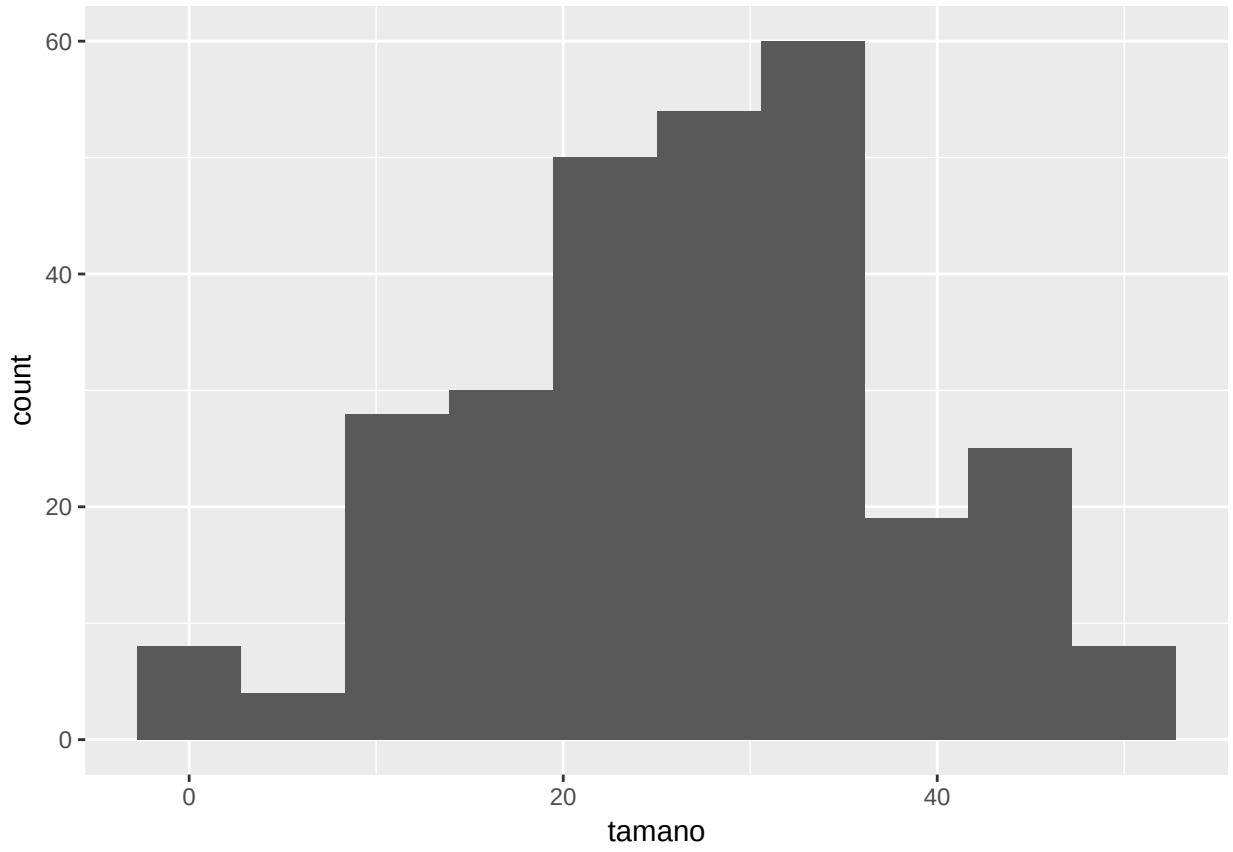
```
ggplot(df1, aes(x=recurrencia, y=tamano)) +
  geom_boxplot() +
  geom_point() +
  geom_jitter(width = 0)
```



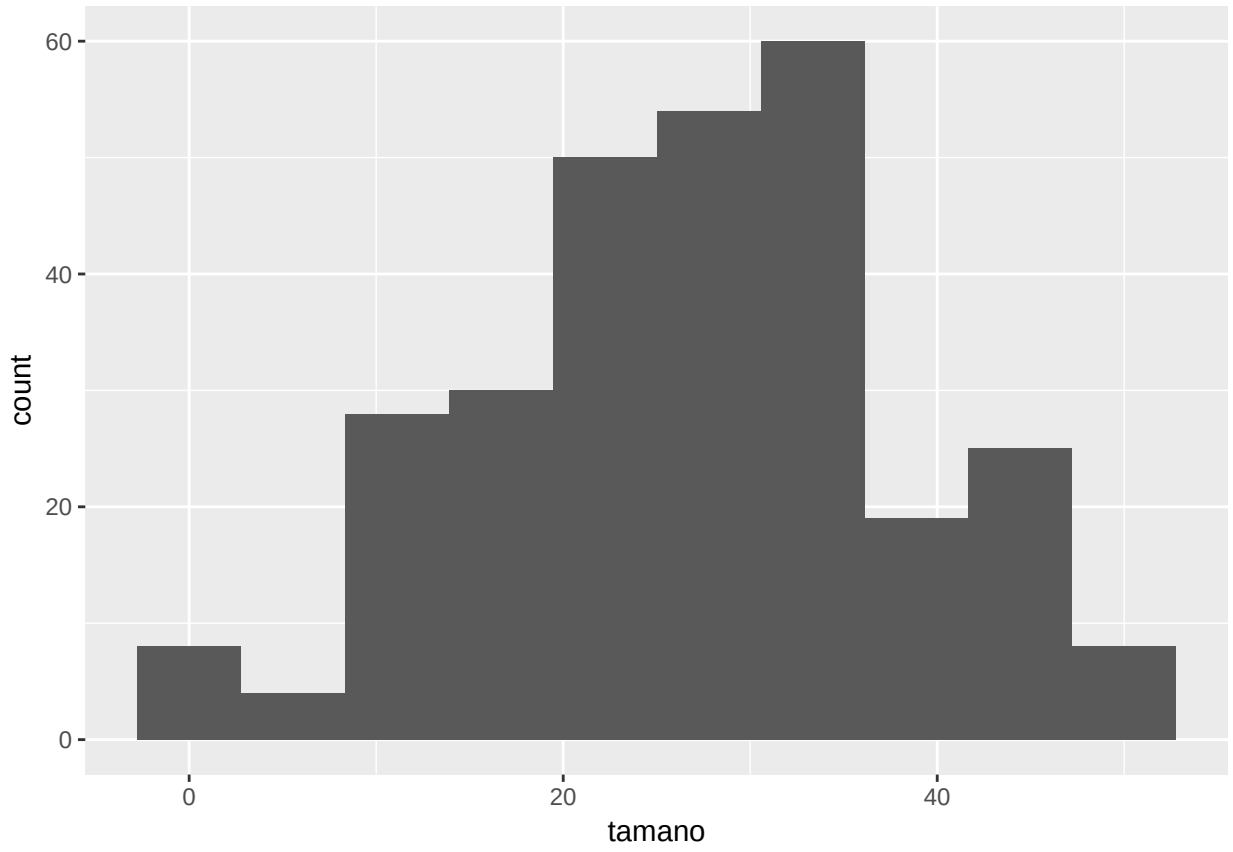
Histogramas y diagramas de densidad

Los histogramas y diagramas de densidad muestran la distribución de una sóla variable. Veamos cómo se resuelve.

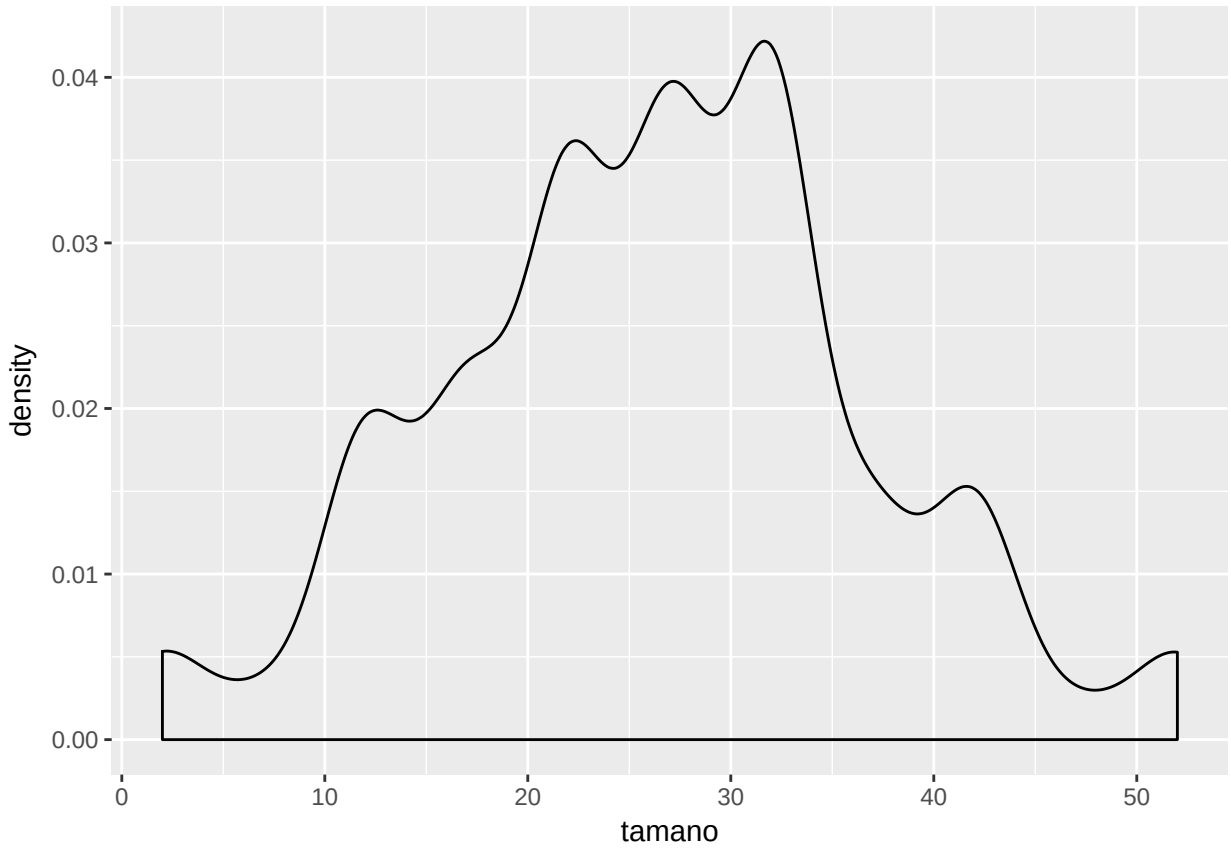
```
ggplot(df1, aes(x=tamano)) +
  geom_histogram(bins=10)
```




```
ggplot(df1, aes(x=tamano)) +
  geom_histogram(bins=10)
```



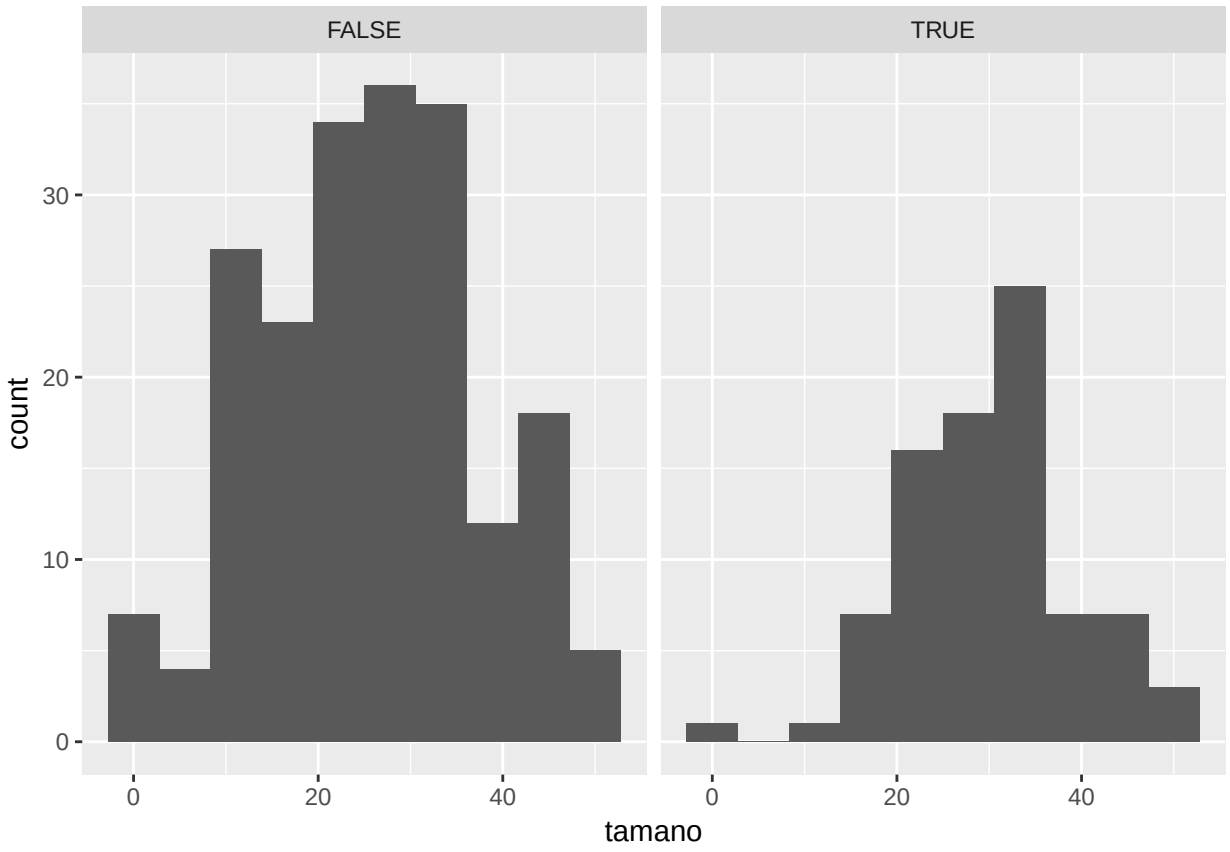

```
ggplot(df1, aes(x=tamano)) +
  geom_density()
```



Facetado

Una de las transformaciones más interesantes para facilitar la comparación es el facetado. En la literatura también se lo conoce como mínimos múltiplos. La función a usar es `facet_grid(VARIABLE ~ .)`

```
ggplot(df1, aes(x=tamano)) +
  geom_histogram(bins=10) +
  facet_grid(. ~ recurrencia)
```

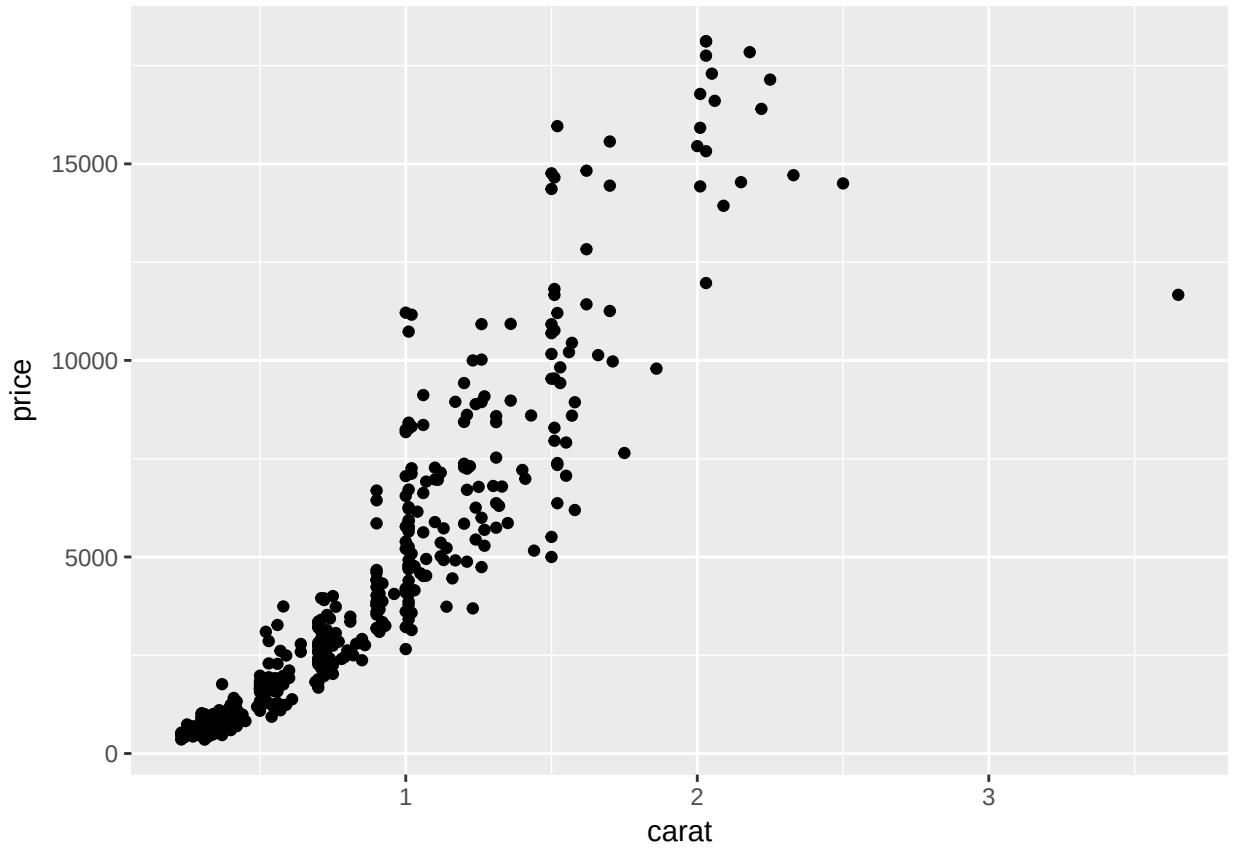


Transformaciones

Algunos datasets requieren que realicemos algunas transformaciones sobre las variables. Veamos cómo realizarlo, para esto usemos un dataset con estas características. El `ggplot2` viene con el dataset diamonds que consiste en un :

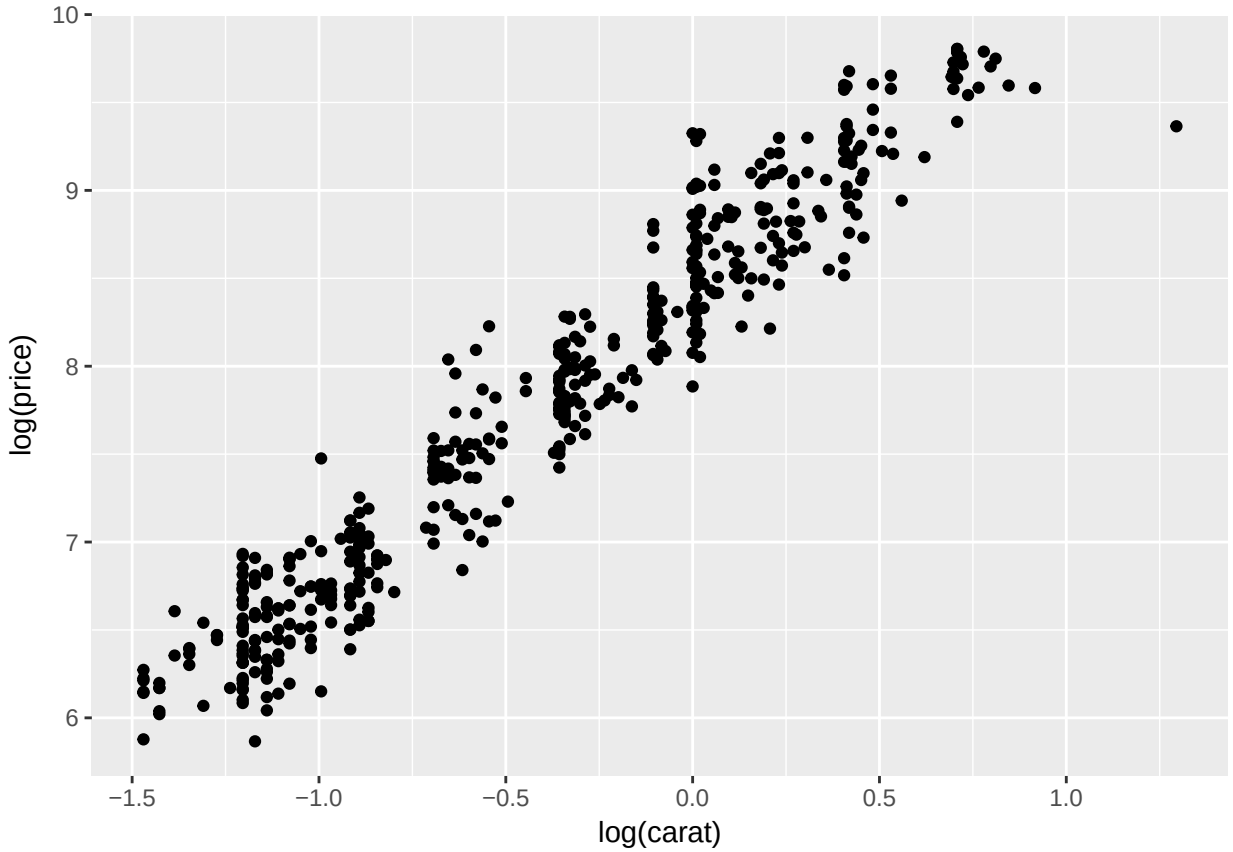
```
set.seed(1410)
dsmall <- diamonds[sample(nrow(diamonds), 500), ]
```

```
ggplot(dsmall, aes(x=carat, y=price)) +
  geom_point()
```



La relación tiene pinta de ser exponencial, veamos cómo se ve transformando las variables.

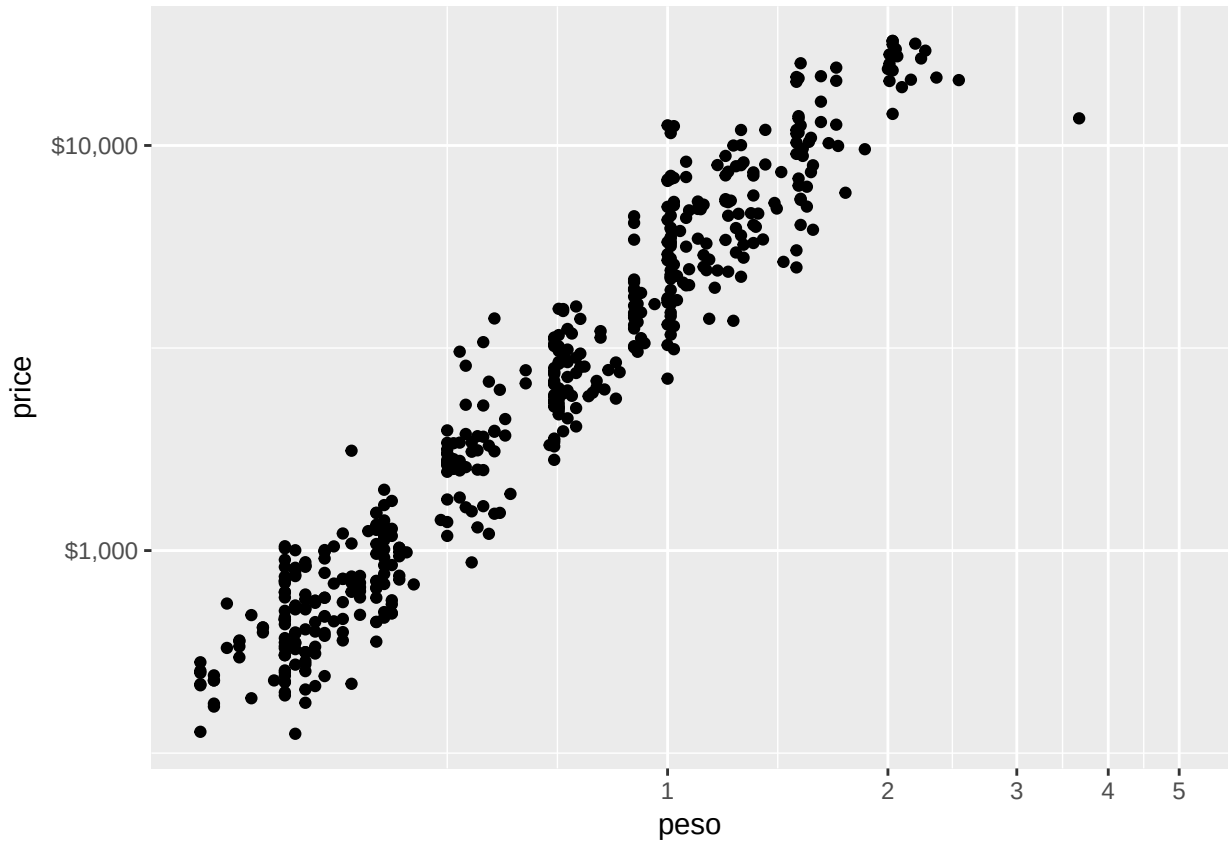
```
ggplot(dsmall, aes(x=log(carat), y=log(price))) +
  geom_point()
```



Pero nosotros no queremos saber el log del precio, sino queremos hacer que la escala del gráfico sea logarítmica. Para eso sumamos la siguiente función. Noten el argumento labels para indicar que se trata de un precio.

```
  ggplot(dsmall, aes(x=carat, y=price)) +
    geom_point() +
    scale_y_log10(labels = scales::dollar) +
    scale_x_log10("peso", breaks = 0:5, limits=c(-1,5))
```

```
## Warning in trans$transform(limits): NaNs produced
```



Y si queremos ponerle una línea de regresión? Se los dejo como ejercicio!

Referencias

- http://www.cookbook-r.com/Graphs/
- http://ggplot2.org/book/qplot.pdf
- http://docs.ggplot2.org/current/
- http://www.ats.ucla.edu/stat/r/seminars/ggplot2_intro/ggplot2_intro.htm
- https://github.com/izahn/workshops/tree/master/R/Rgraphics