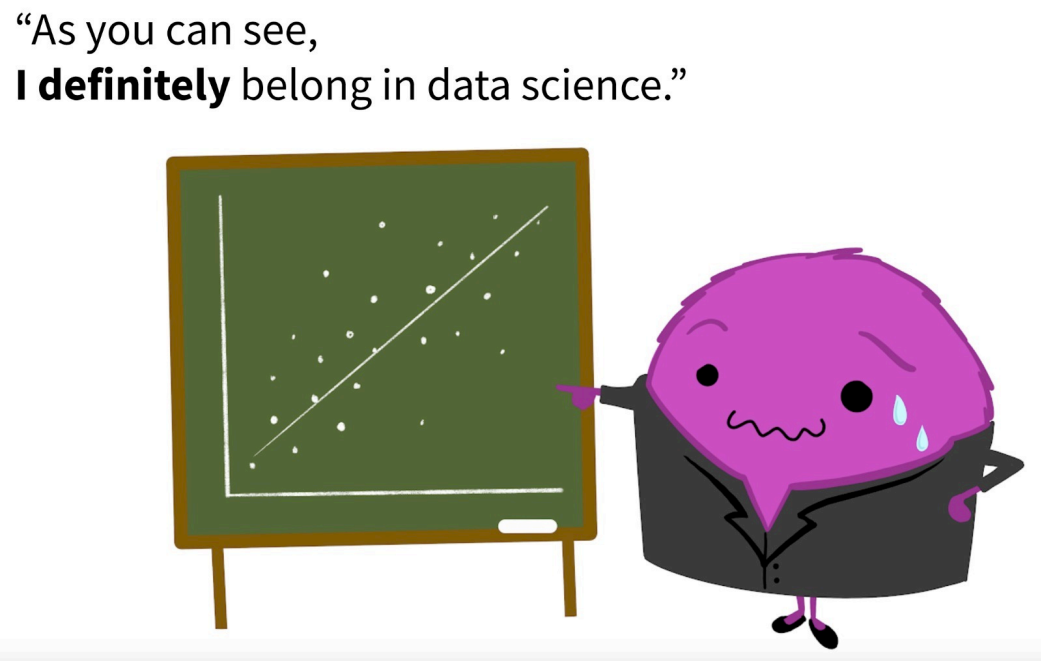
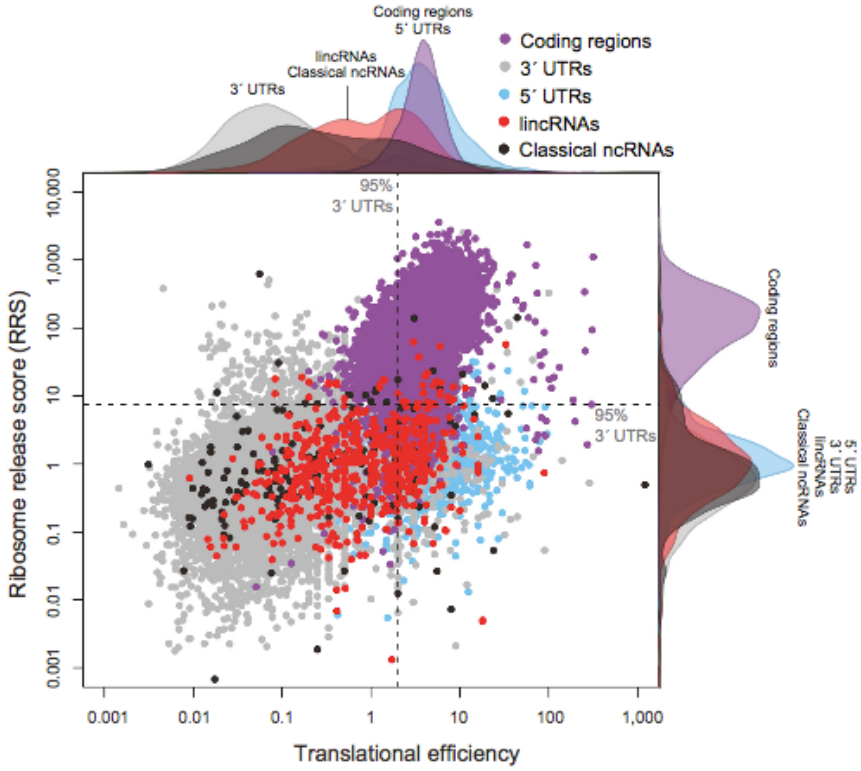
GRÁFICOS EN R – 2 PARTE!



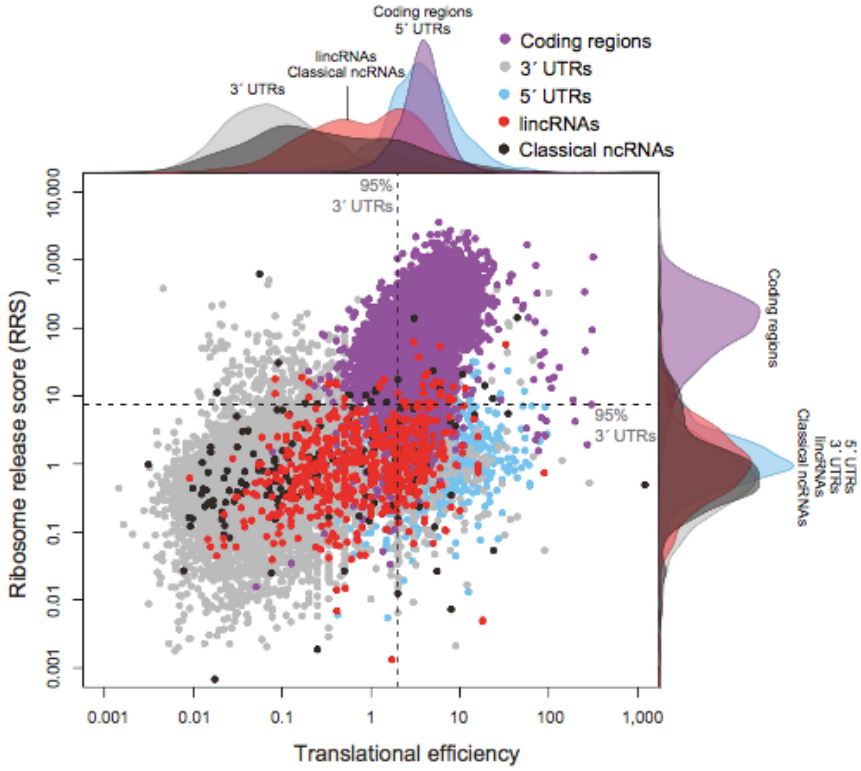
Gráficos en R, Parte II

- **Graficas Multi-panel**
- **Diseño**
- **ggplot**



Gráficos en R, Parte II

- **Graficas Multi-panel**
- **Diseño**
- ggplot



Múltiples paneles

- A menudo es muy útil hacer varios gráficos juntos
- Como? Con una matriz de graficas usando:

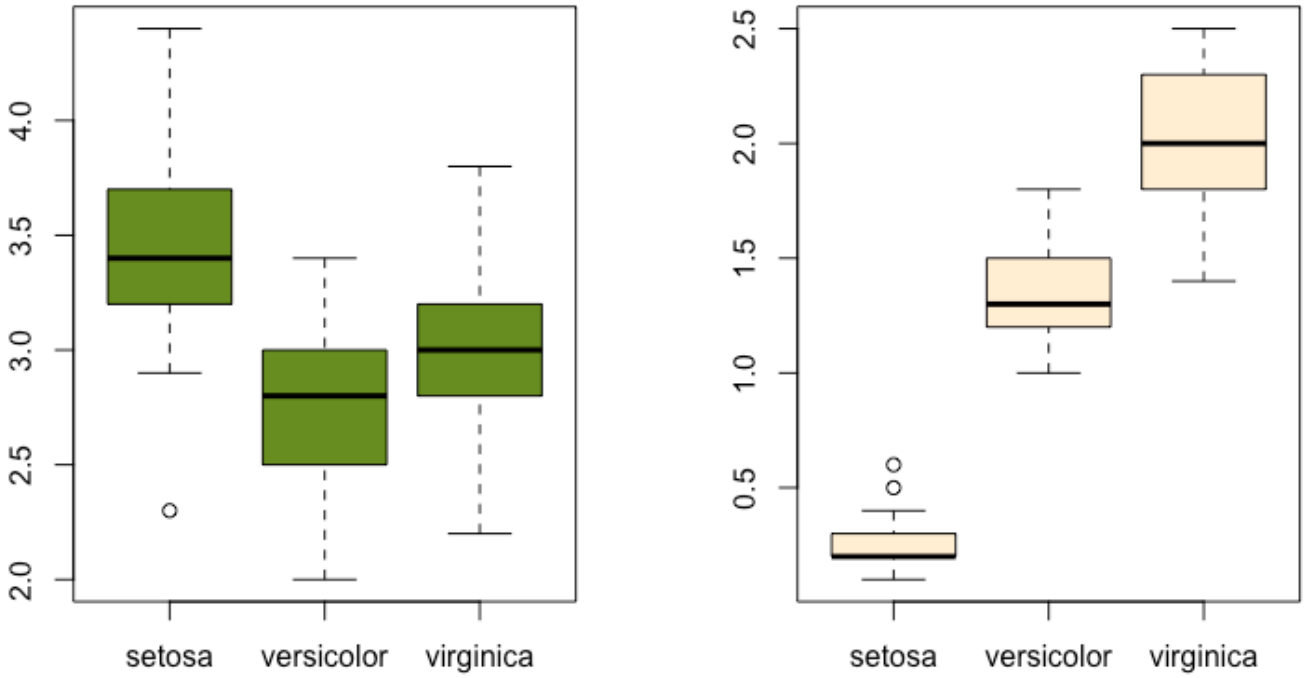
  par(mfrow=c(nrows, ncols))

  c(nrows, ncols) = número de filas y número de columnas
- Use par(mfrow=c(nr, nc)) antes de hacer los gráficos

Intenten esto:

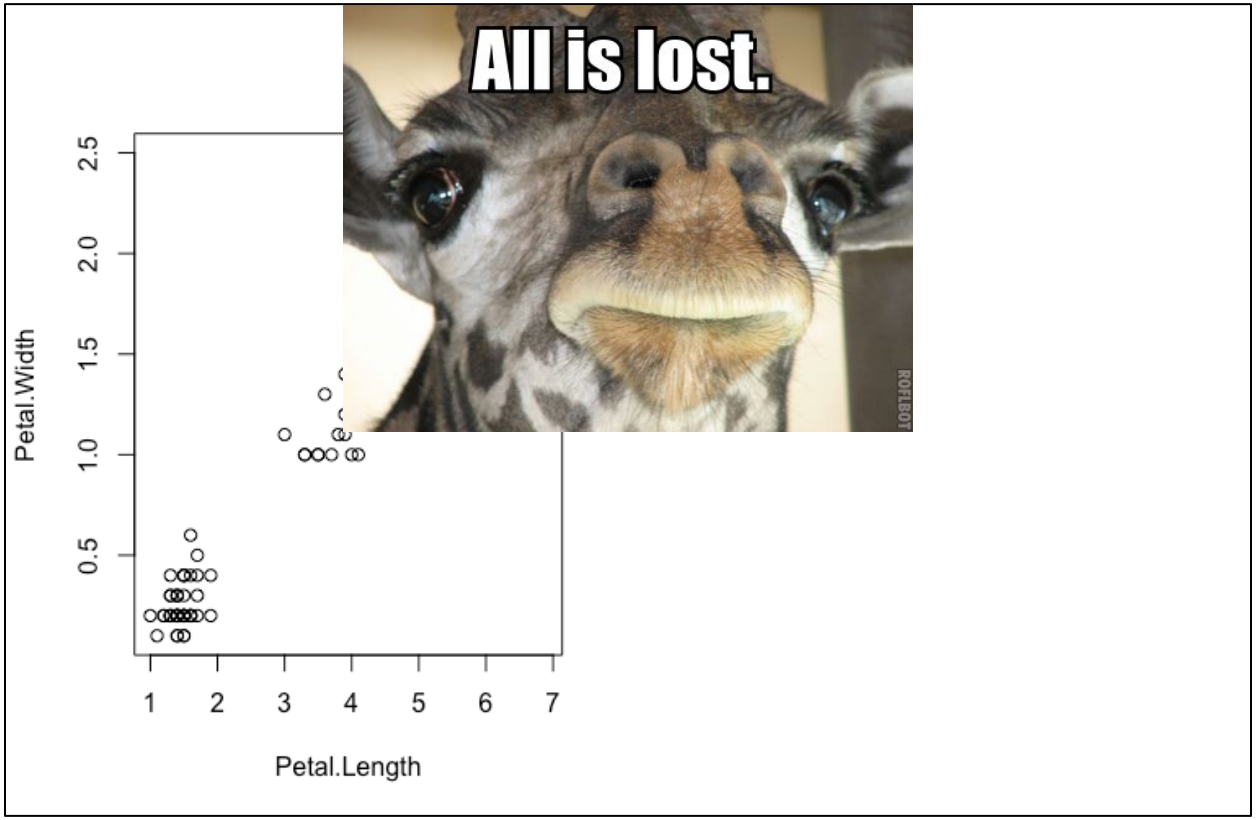
```
par(mfrow=c(1,2))

boxplot(Sepal.Width~Species,data=iris,
     col="olivedrab")

boxplot(Petal.Width~Species,data=iris,
     col="papayawhip")
```

Dos paneles!


2.0
2.5
3.0
3.5
4.0
setosa
versicolor
virginica
0.5
1.0
1.5
2.0
2.5
setosa
versicolor
virginica

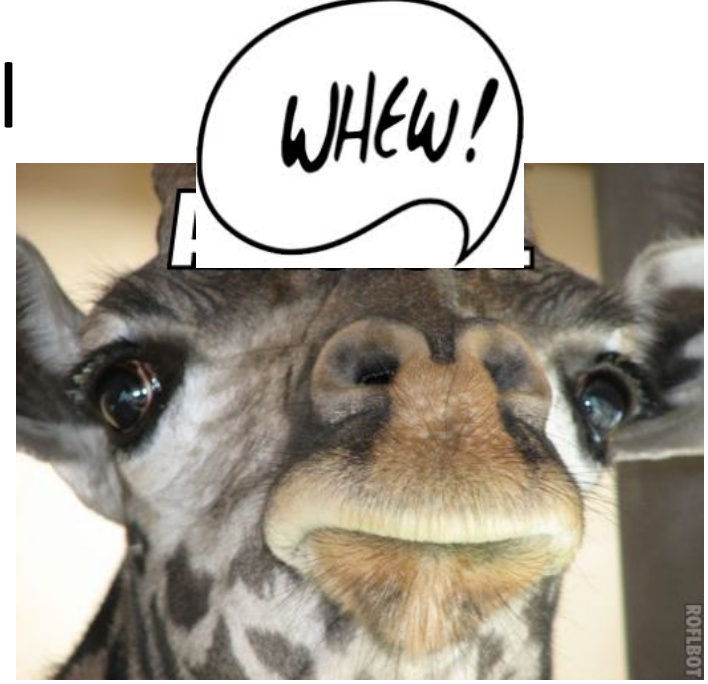
Ahora agregue otra gráfica

```
plot(Petal.Width~Petal.Length,data=iris)
```



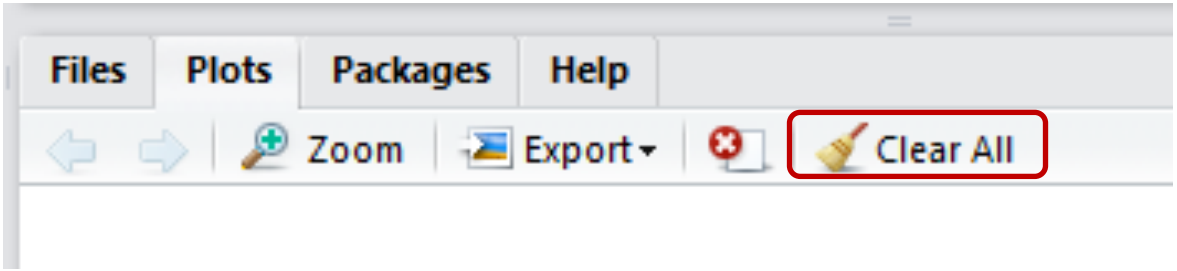
AHHHHHHH ESTAMOS ATRAPADOS EN UN MUNDO ETERNO DE DOS PANELES!!!!!!

Para volver a un solo panel




```
par(mfrow=c(1,1))
```

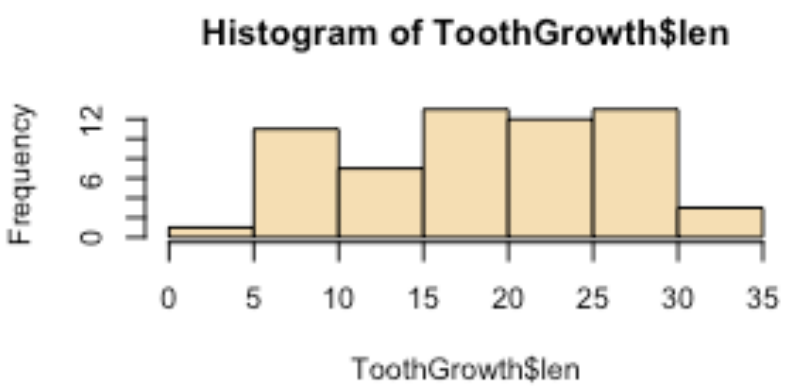
- O `graphics.off()`
- O, recuerda:

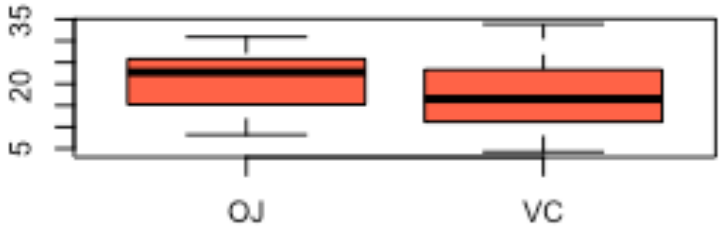


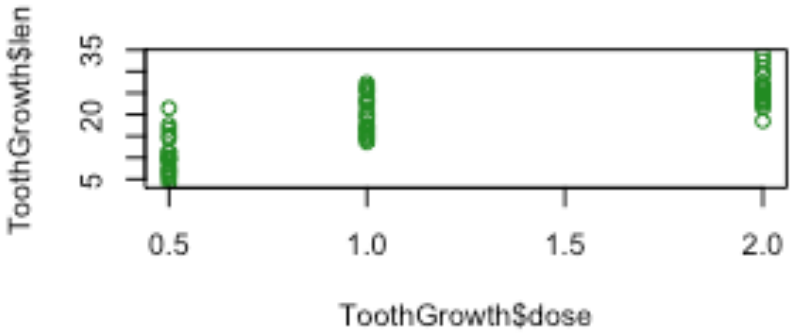
Seleccione **Clear All** en la ventana plots: restablece todas las graficas y par() a los valores predeterminados.

Optimización del espacio: Cambio de márgenes

¡Algunas graficas son en su mayoría espacios en blanco!







Optimizar espacio: Cambiar los márgenes

- Para modificar los márgenes alrededor de cada plot, utilice mar

- `par(mar = c(bottom, left, top, right))`
  - El valor predeterminado es:

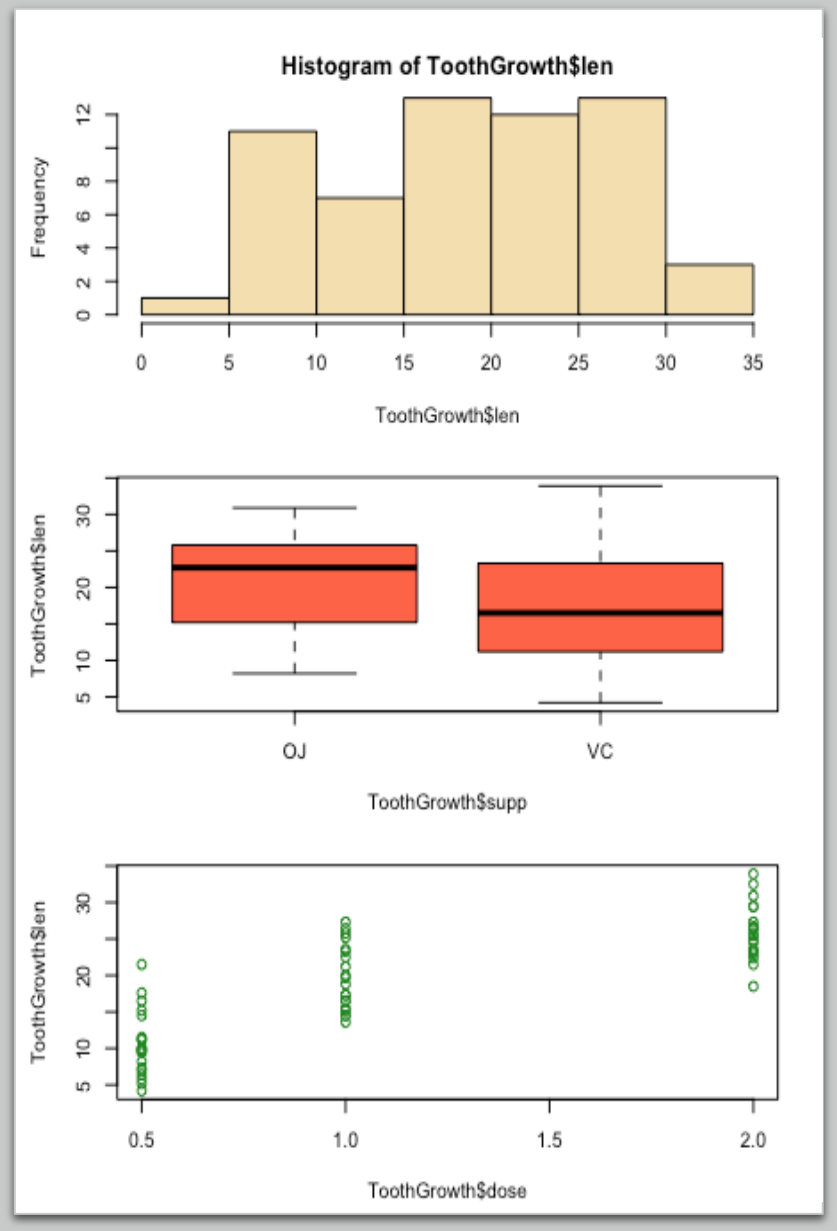
    `par(mar = c(5,4,4,2) + 0.1)`

- Para modificar los márgenes alrededor de toda la figura, utilice oma:

- `par(oma = c(bottom, left, top, right))`
  - El valor predeterminado es

    `par(oma = c(0,0,0,0))`

```
par(mfrow=c(3,1))
par(mar=c(4,4,2,2))

hist(ToothGrowth$len,
     col="wheat”)

boxplot(ToothGrowth$len~
     ToothGrowth$supp,
     col="tomato")

plot(ToothGrowth$len~
     ToothGrowth$dose,
     col="forestgreen")
```



Diseños personalizados

- `layout()` da alternativas más flexibles que `mfrow`
- `layout()` permite la creación de múltiples regiones de figuras de tamaños desiguales
- Toma una matriz con el número de filas y columnas en el diseño de la figura
- Valores de matriz = las filas y columnas que ocupará cada figura

`layout()`

```
layout(mat, widths, heights, ...)
```

- `mat` una matriz que da la ubicación de las siguientes N figuras.
- `widths` un vector de valores para los anchos de las columnas.
- `heights` un vector de valores para las alturas de las filas.

¿Quieres 15 plots de diferentes anchuras y alturas? ¡No hay problema!

```
mat <- matrix(1:15,nrow=3, ncol=5, byrow=T)

layout(mat=mat, widths = c(1,3,5,1,3),
       heights = c(2,2,1) )

layout.show(n=15)
```

15 plots

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Orden mas complejo

```
mat <- matrix(c( 1, 1, 3, 4, 5,
                 1, 1, 3, 6, 7,
                 2, 2, 2, 2, 7),
              nrow=3, ncol=5, byrow=T)
```

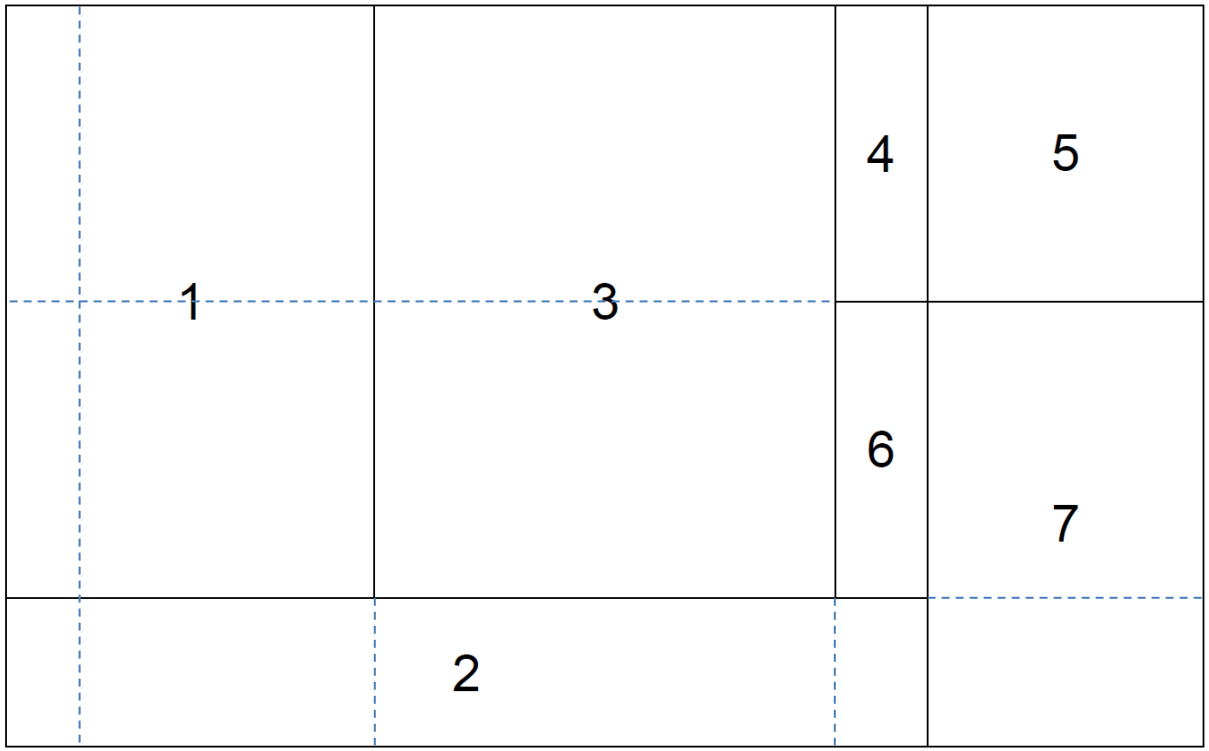
1	1	3	4	5
1	1	3	6	7
2	2	2	2	7

```
layout(mat=mat, widths = c(1,3,5,1,3),
       heights = c(2,2,1) )
```

```
layout.show(n=7)
```

Orden mas complejo

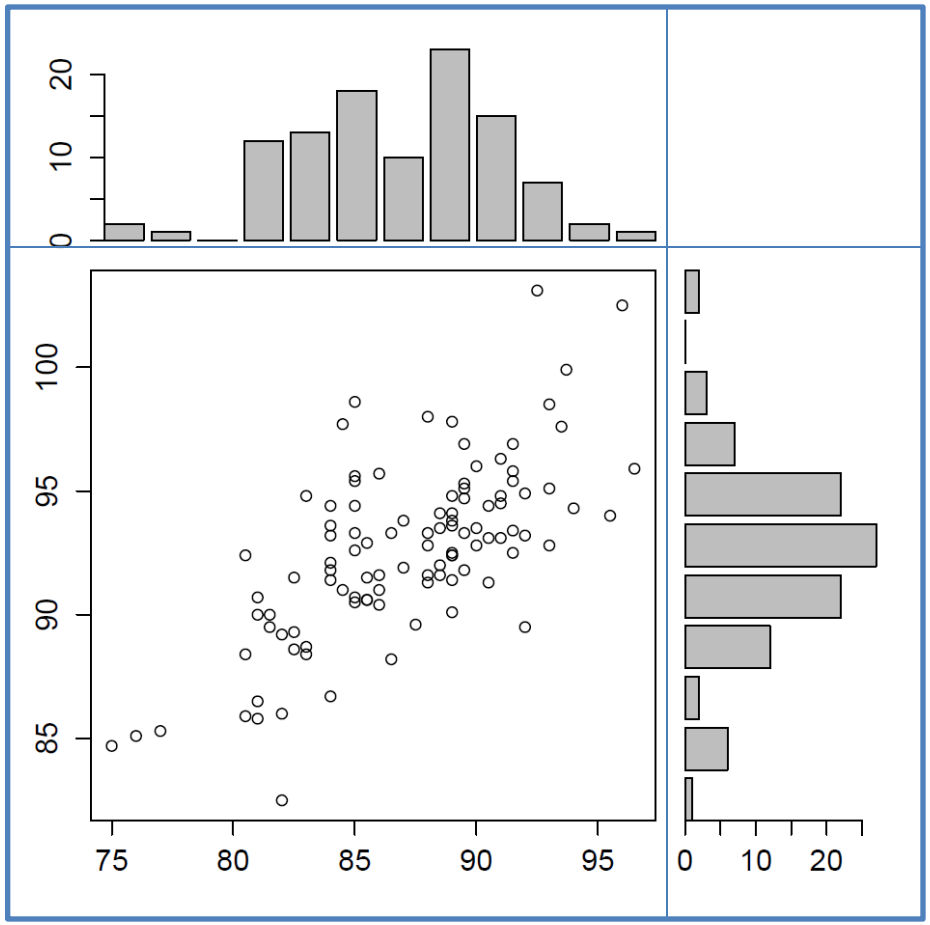
1	1	3	4	5
1	1	3	6	7
2	2	2	2	7




```
widths = c(1,3,5,1,3)
heights = c(2,2,1) )
```

Cero para espacios vacíos

2	0
1	3



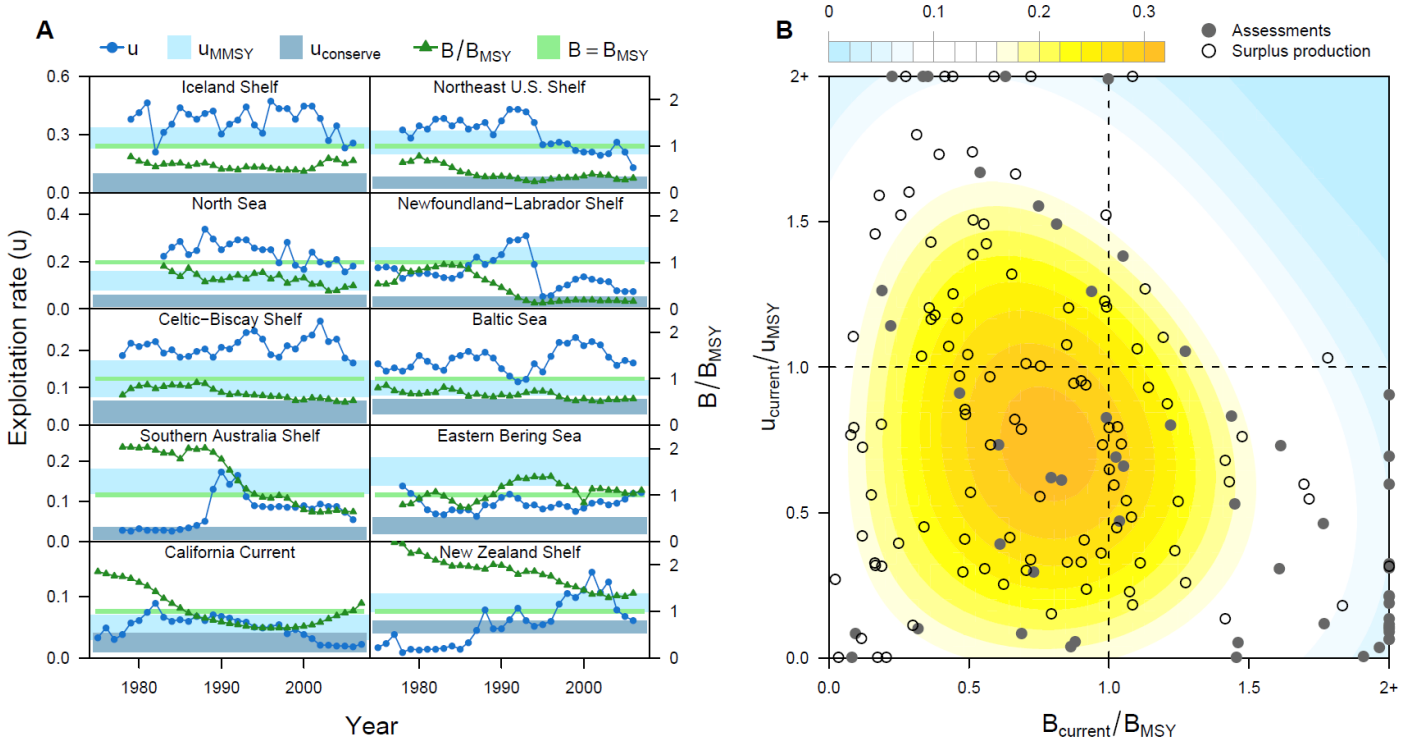
Ejemplo

11	11	0	13
1	6	0	12
2	7	0	12
3	8	0	12
4	9	0	12
5	10	0	12

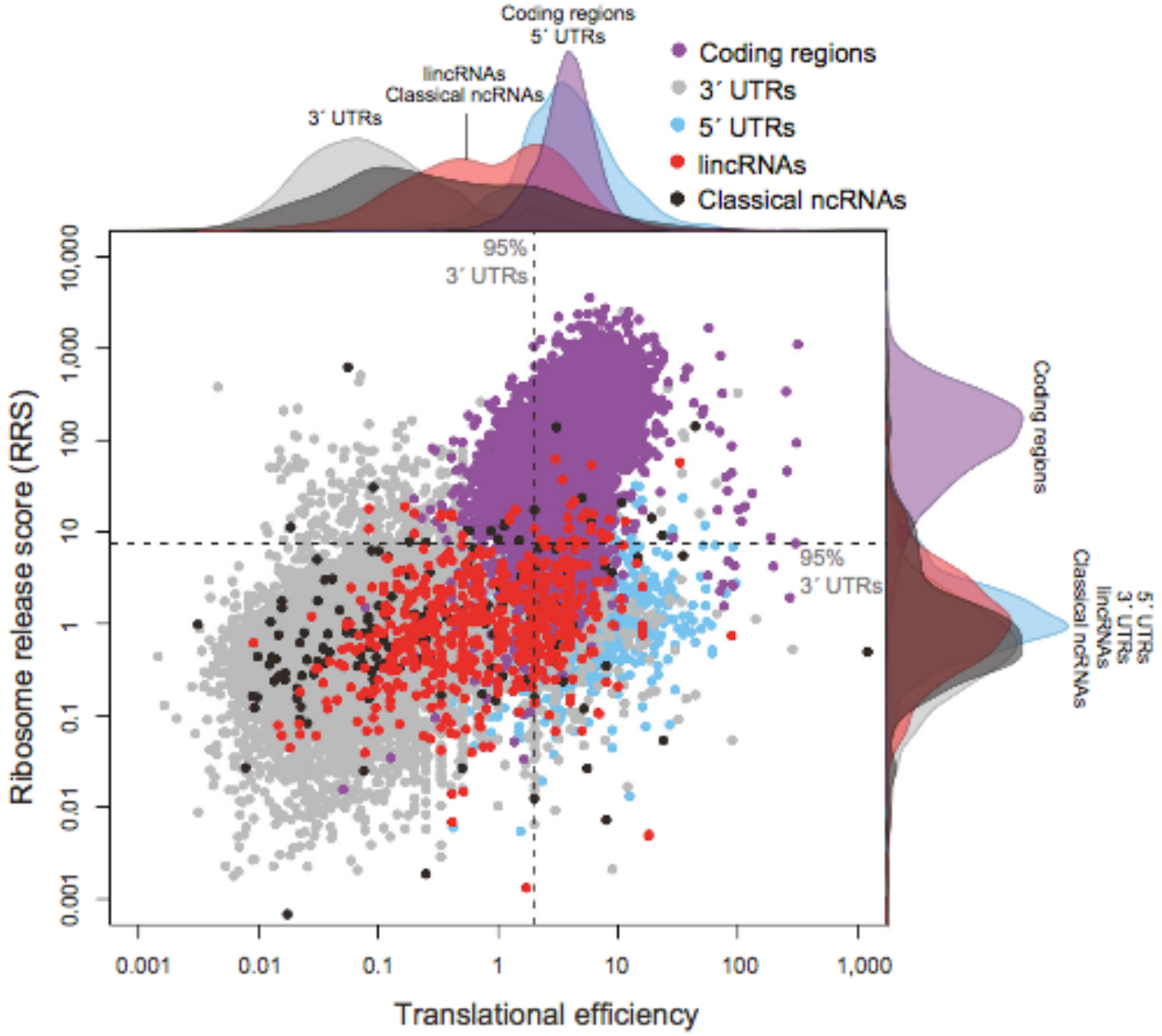
11			13
1	6		
2	7		
3	8		12
4	9		
5	10		

Despues hacen las graficas...

11	11	0	13
1	6	0	12
2	7	0	12
3	8	0	12
4	9	0	12
5	10	0	12

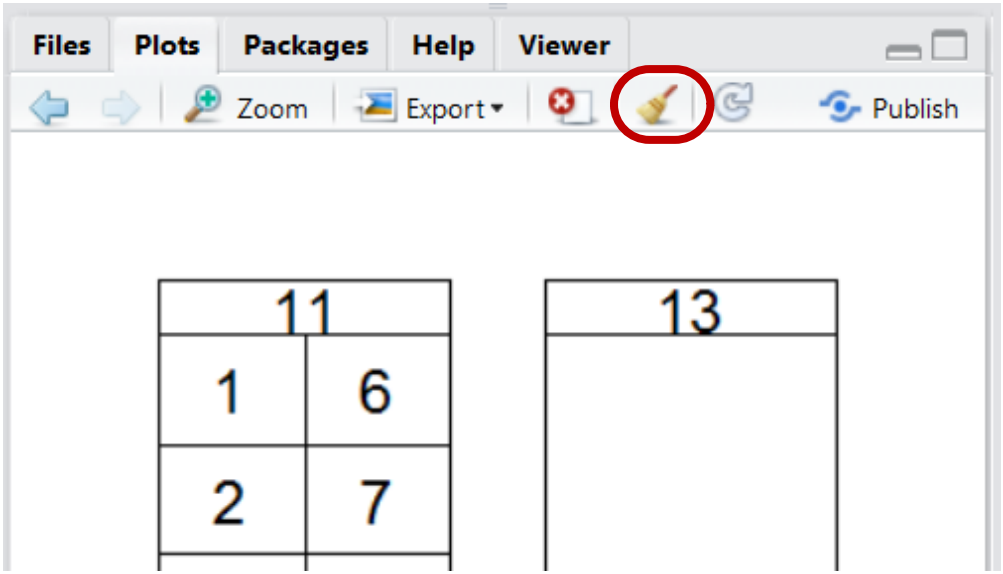


Worm et al. (2009) Rebuilding global fisheries. Science 325:578-585


Coding regions
5′ UTRs
lincRNAs
Classical ncRNAs
3′ UTRs
Coding regions
3′ UTRs
5′ UTRs
lincRNAs
Classical ncRNAs
95%
3′ UTRs
95%
3′ UTRs
Ribosome release score (RRS)
0.001
0.01
0.1
1
10
100
1,000
10,000
0.001
0.01
0.1
1
10
100
1,000
Translational efficiency
Coding regions
5′ UTRs
3′ UTRs
lincRNAs
Classical ncRNAs

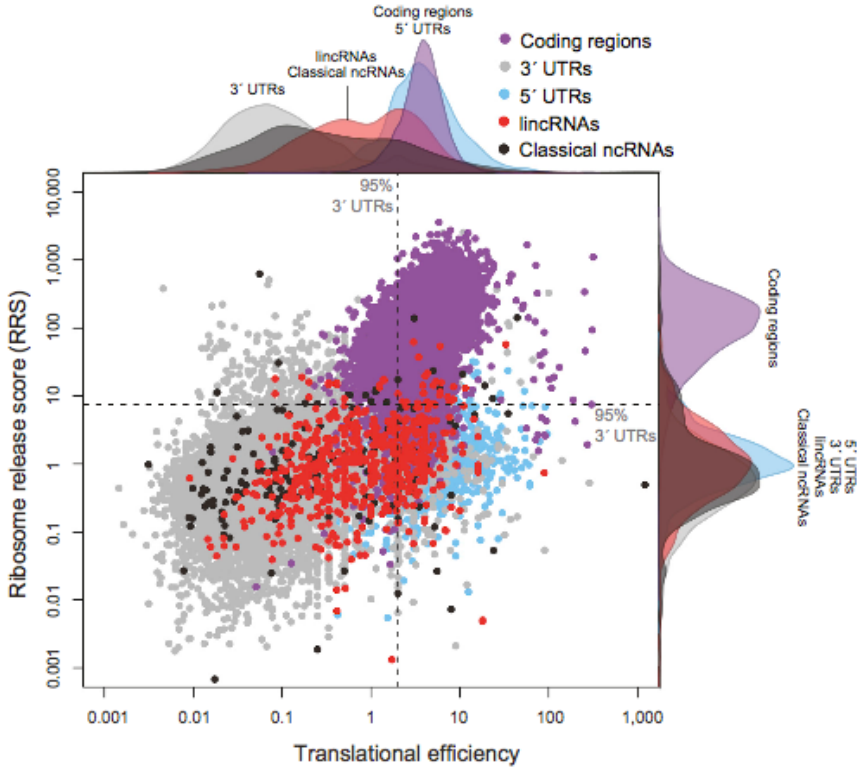
Después de usar `layout()` limpien la ventana de plots para volver a los parámetros predeterminados.

`graphics.off()`..



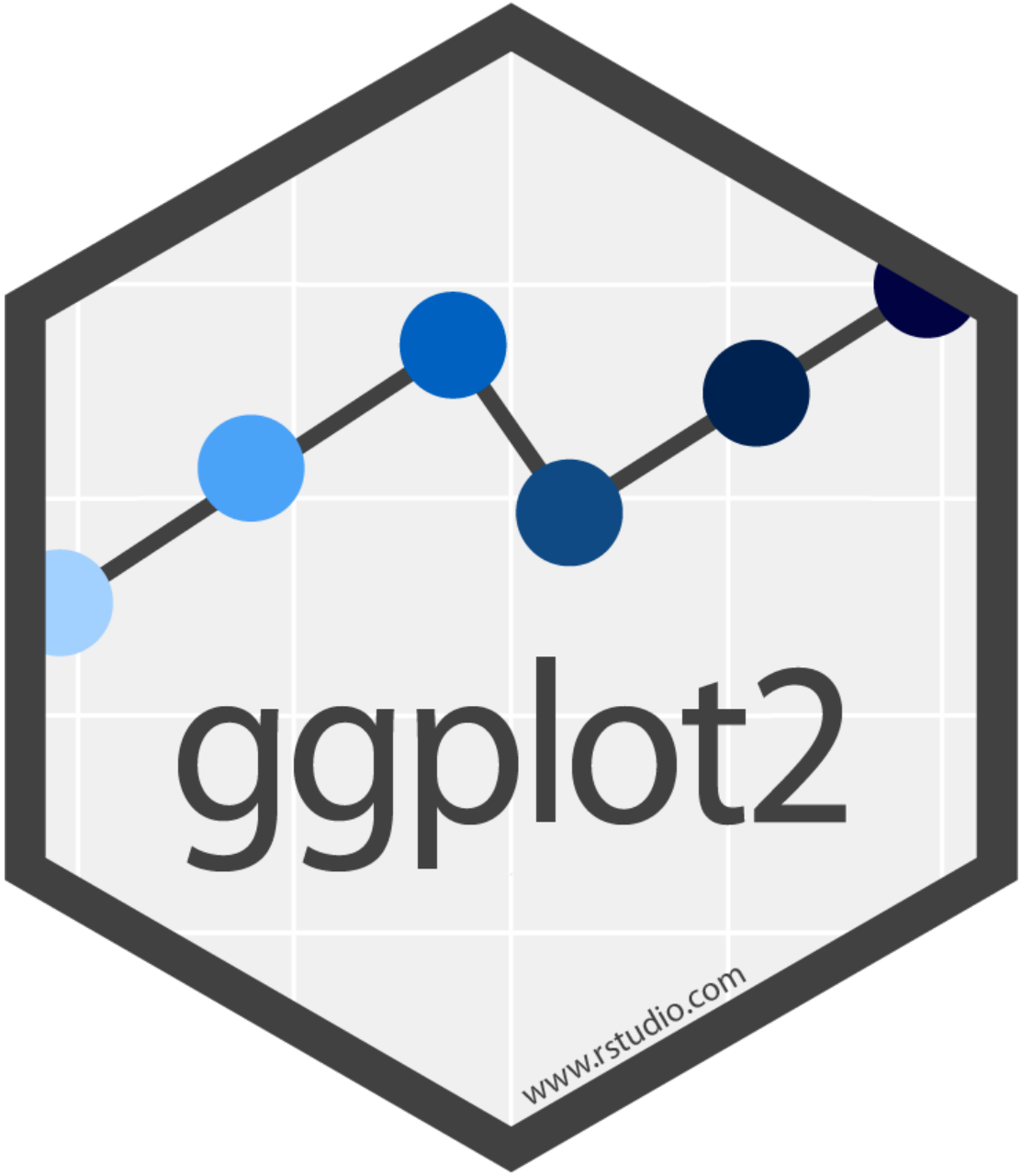
Gráficos en R, Parte II

- Graficas Multi-panel
- Diseño
- **ggplot**



Gráficos en R: resumen

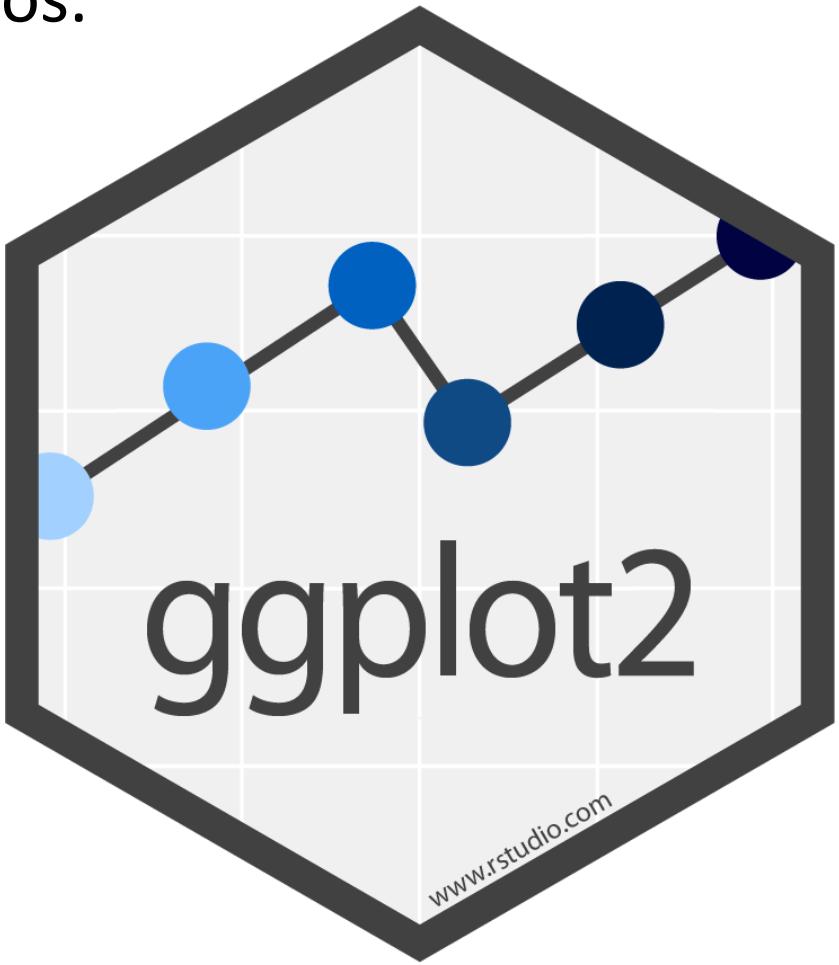
- Hay varias maneras de hacer gráficos en R
  - **Gráficos base (los cambios en el diseño son bastante fáciles, altamente modificables)**
  - **ggplot2** (bueno para graficas multipanel, vistas alternativas rápidas de datos, cambios en el diseño básico pueden ser difíciles)


ggplot2
www.rstudio.com

Qué es ggplot?

- ggplot es un paquete (ggplot2) para la "exploración rápida de datos” con graficos.

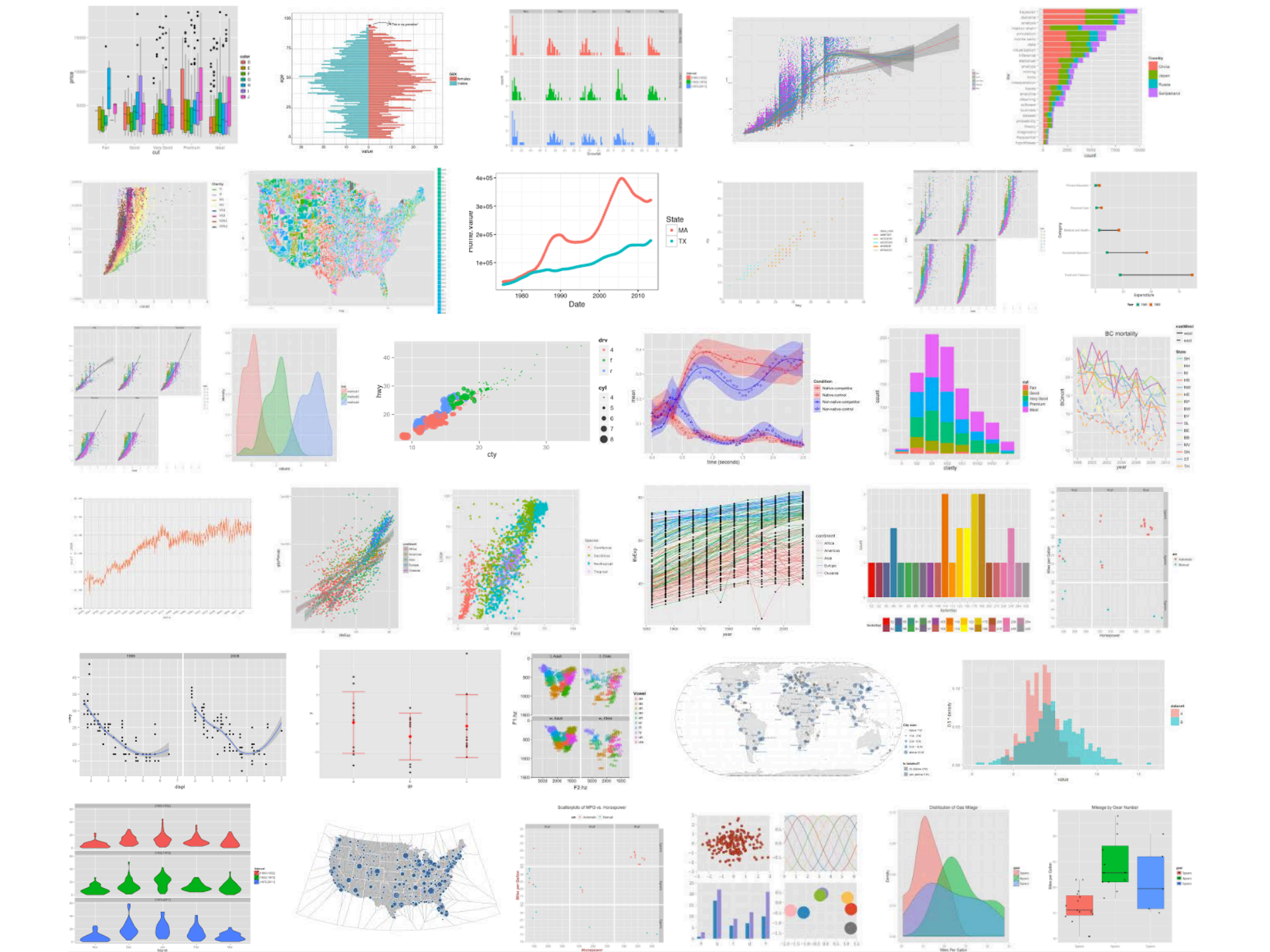
- Es parte de ’tidyverse’



ggplot vs. gráficos base

- **ggplot**: exploración rápida de datos

- **Base graphics**: personalización / complejidad

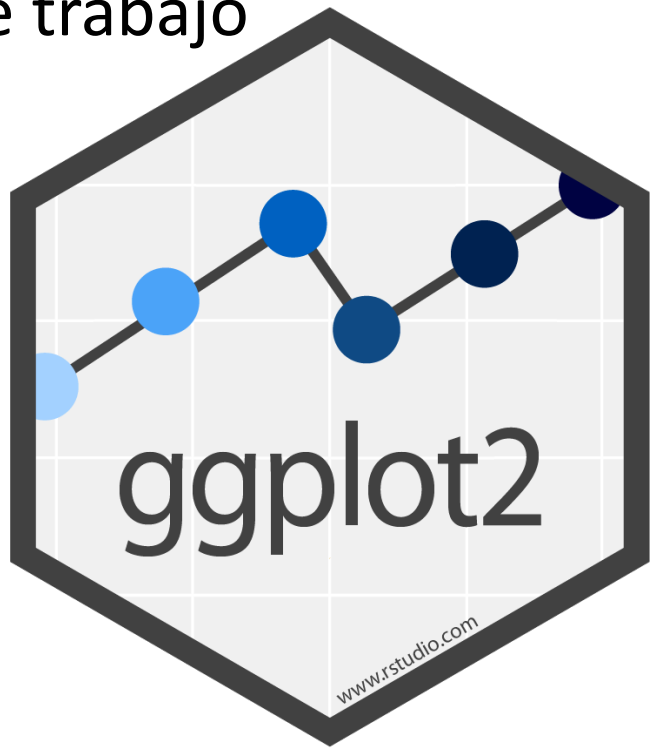


- Para instalar ggplot (una vez por ordenador):

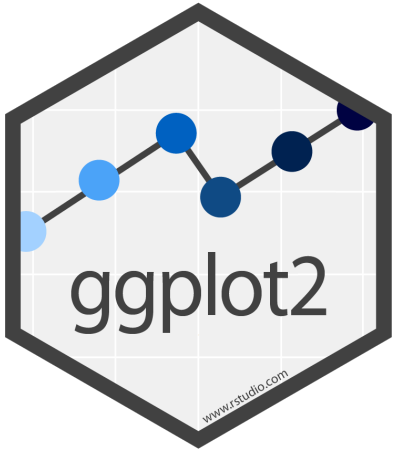
```
install.packages("ggplot2")
```

- Para cargar ggplot en el espacio de trabajo
- (una vez por sesión):

```
library("ggplot2")
```



por qué gg?



- **gg** es por “grammar of graphics”

- Utiliza un conjunto de términos que define los componentes básicos de un grafico

- Produzca figuras utilizando una sintaxis coherente (amigable con el usuario)

ggplot

ggplot(data = _____, aes(___)) +

geom (type of plotted data) +

other stuff.. (stats, labels, facets, themes..)

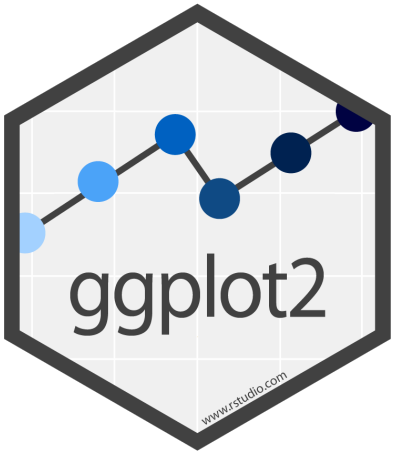
aesthetics: Cómo se representan los datos visualmente x, y, color, tamaño, forma, etc.

…usar '+' para añadir capas:

geometry: Geometrías de objetos: puntos, líneas, polígonos, etc.

Otras capas se pueden agregar con otra +

Qué es **G**rammer of **G**raphics?

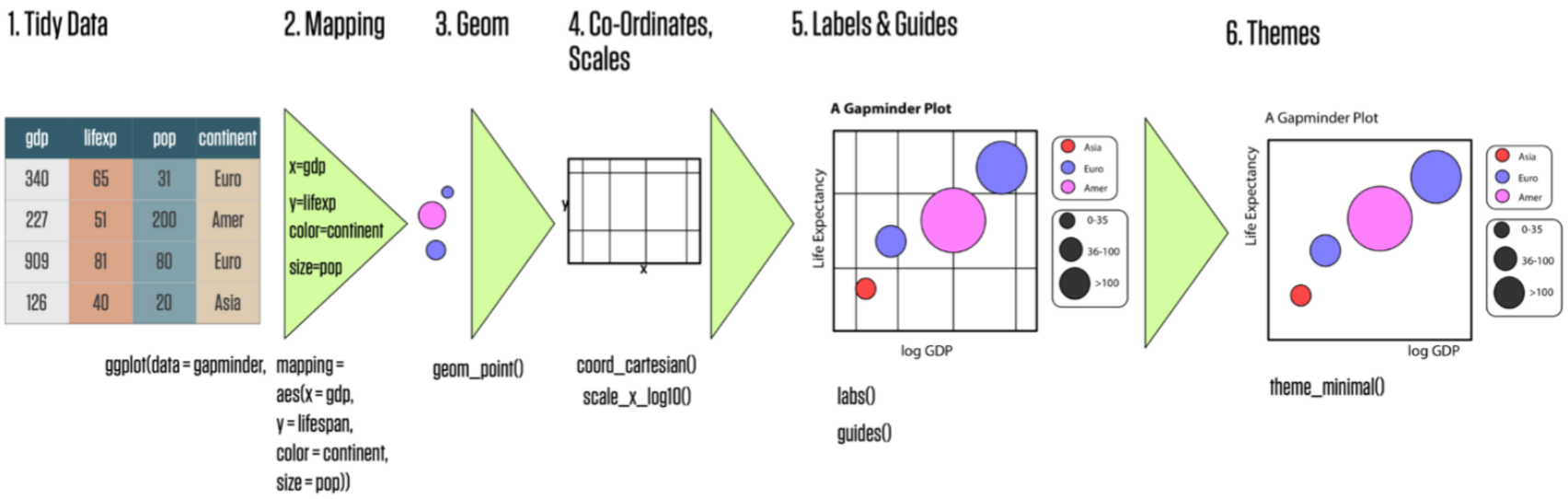


- Una gráfica básica de ggplot2 consiste en:
  - **data**: Debe ser un data.frame
  - **aes**thetics: Cómo se representan visualmente los datos x, y, color, tamaño, forma, etc.

    +
  - **geom**etry: Geometrías de objetos - puntos, líneas, polígonos, etc.
  - y más…

Qué es **G**rammer of **G**raphics?

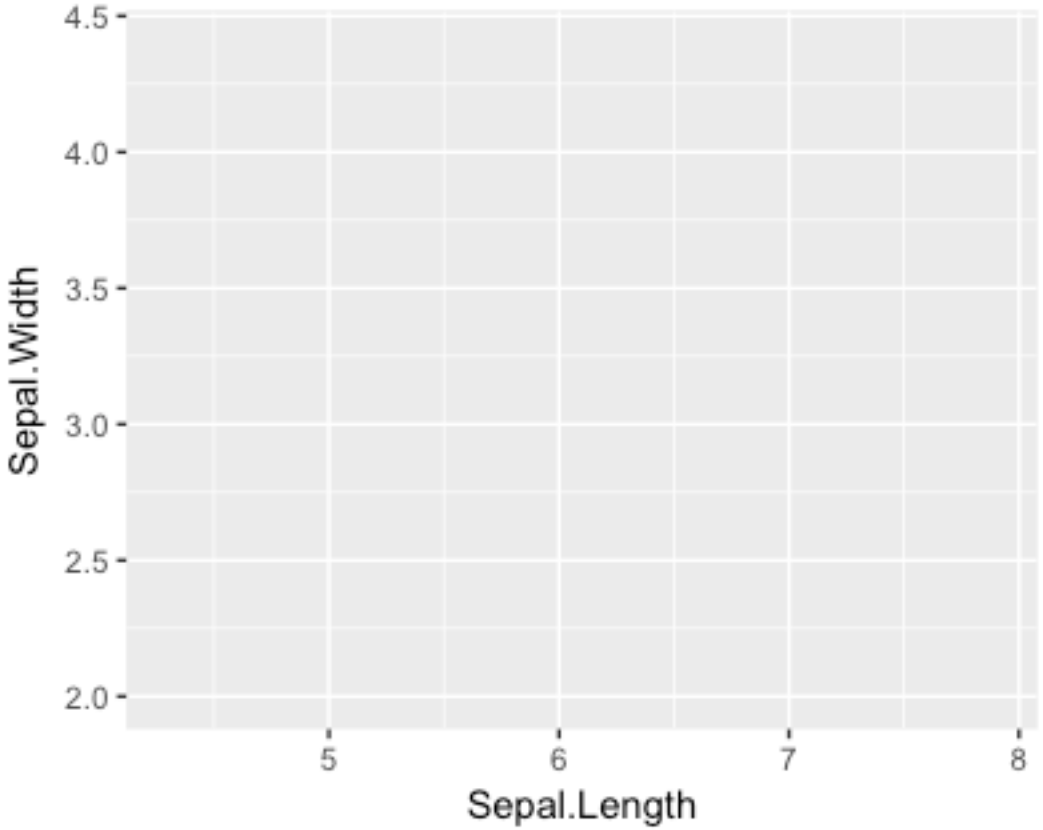
- Una gráfica básica de ggplot2 consiste en:



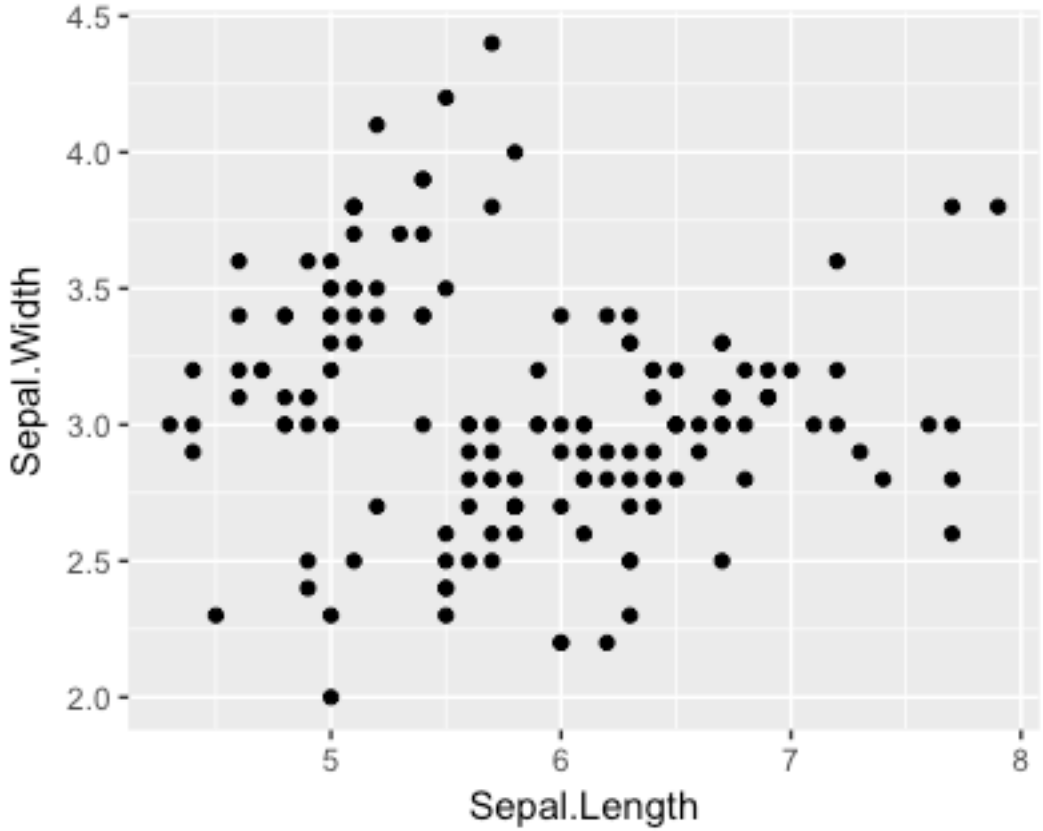
```
x <- ggplot(data=iris,
aes(x=Sepal.Length, y=Sepal.Width))
```

Luego, escriba:

```
x
```



```
x + geom_point()
```

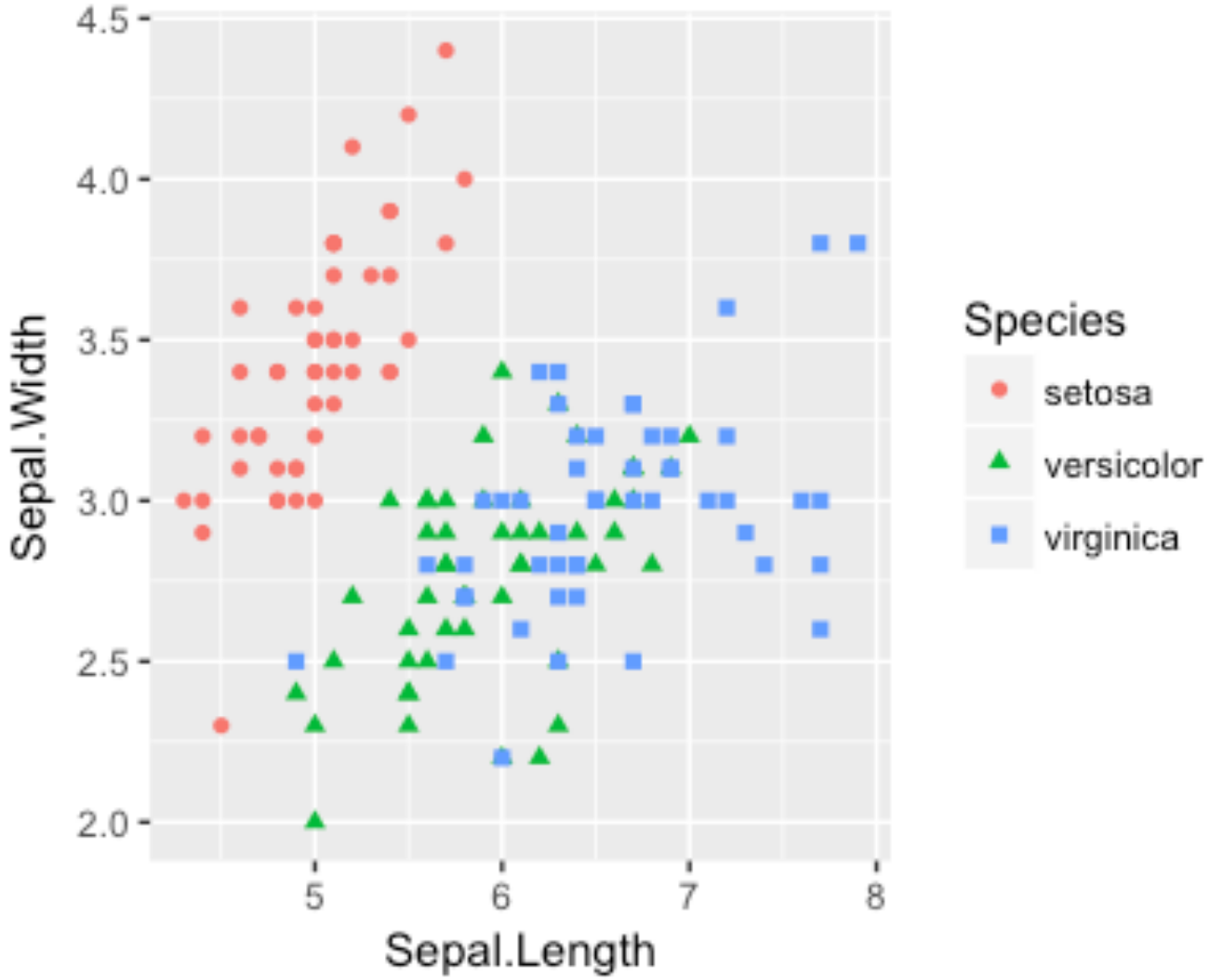


Recordatorio:

```
x<- ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width))
```

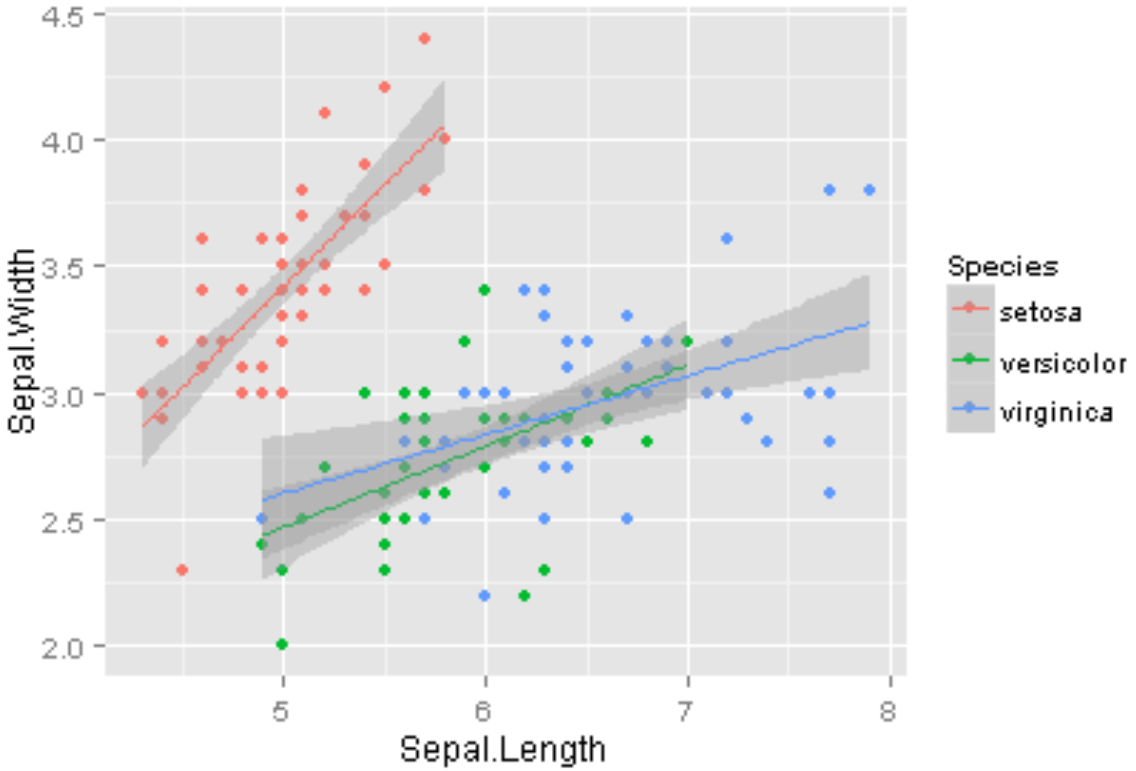
```
ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width,
      color=Species, shape=Species)) +
      geom_point()
```

Cambie la visualización con **aes()**



Podemos usar otras funciones : e.g., stats

```
x <- ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width,
                      color=Species))+
                      geom_point() +
                      stat_smooth(method="lm")
x
```

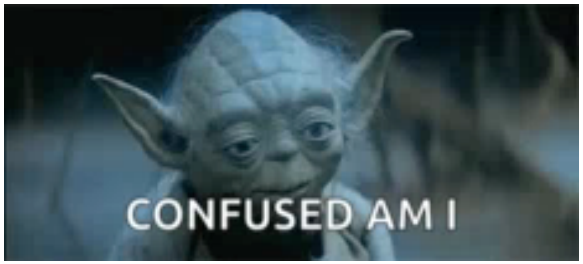


Agregar aesthetics a geoms

- Agregar aes() en el original ggplot es global (aes son heredados por cualquier otro geom que se construyen en la parte superior).

- Agregar aes() en un geom sólo se utilizará en ese geom.

- Si utiliza alguna aes() en geom, anulan la configuración en ggplot.

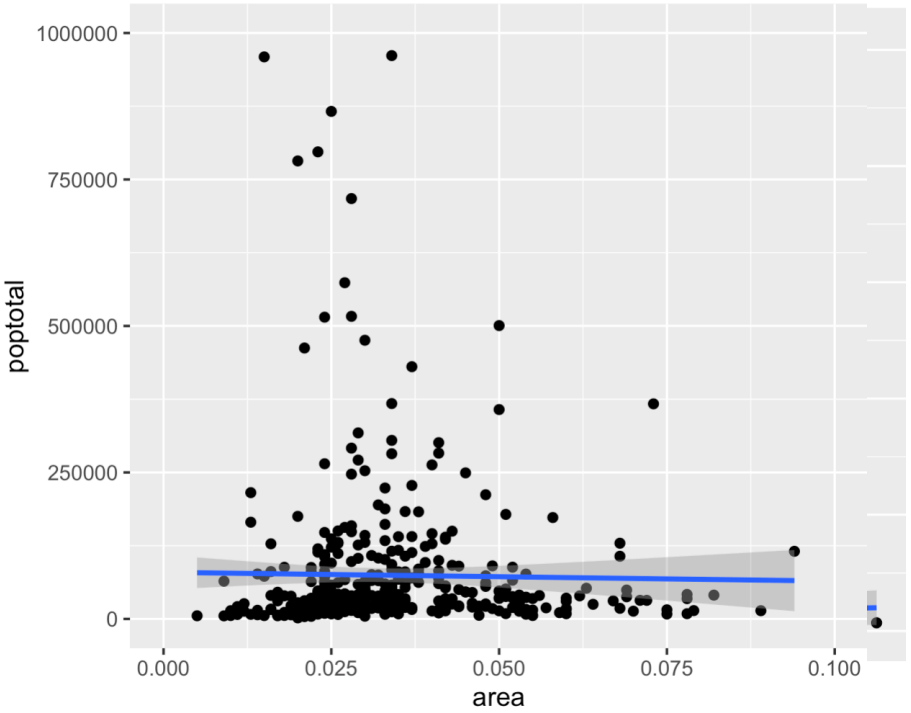



```
ggplot(midwest,aes(x=area, y=poptotal, size = 2)) +
       geom_point(aes(size = 4))
```

Ajuste de los límites de los ejes X e Y

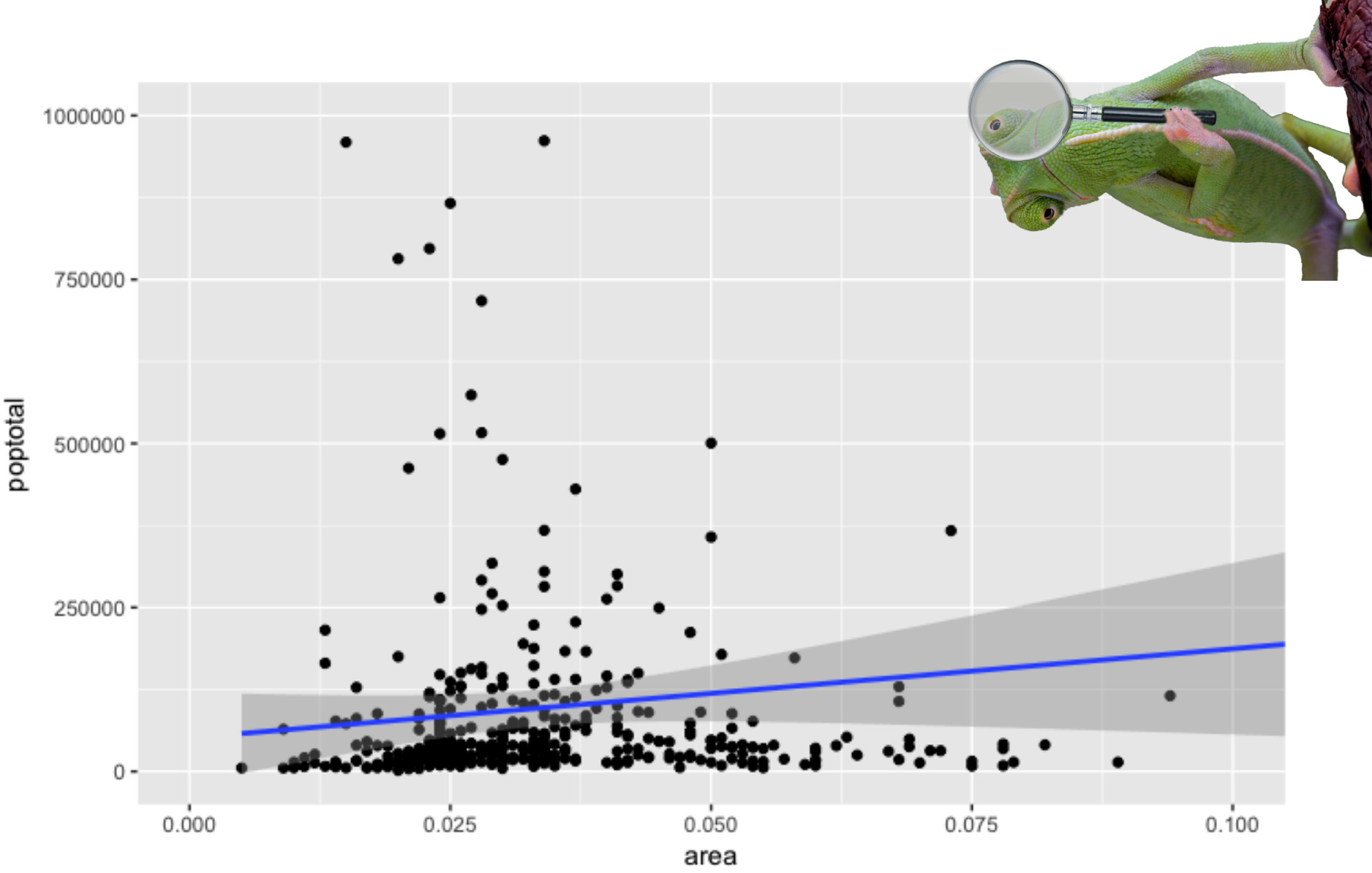
```
g <- ggplot(midwest, aes(x=area, y=poptotal))+
     geom_point() +
     geom_smooth(method="lm")
g
```

```
# Eliminar los puntos
fuera de los límites

g + xlim(c(0, 0.1)) +
ylim(c(0, 1000000))
```



Zoom

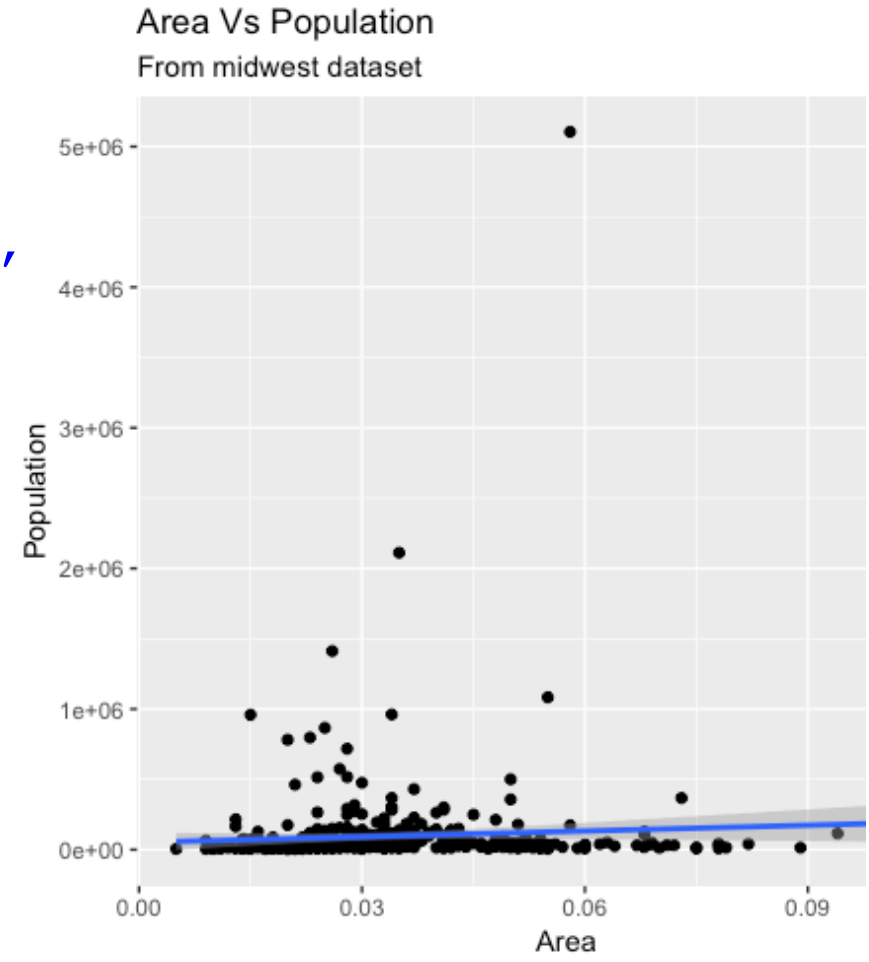
```
# Acércte sin eliminar los puntos fuera de los límites.

g +
coord_cartesian(xlim=c(0,0.1),
ylim=c(0, 1000000))

# Como resultado, la línea de mejor ajuste es la misma que la gráfica original.
```




```
g + coord_cartesian(xlim=c(0,0.1), ylim=c(0, 1000000))
```

Cambio de etiquetas y títulos de ejes

```
g + ggtitle("Area Vs Population",
subtitle="From midwest dataset")
+ xlab("Area") +
ylab("Population")
```



Cómo guardar un ggplot: ggsave

```
ggsave("plot.png", width = 5, height = 5)
```

Guarda la última gráfica como archivo de 5' x 5' denominado "plot.png" en el directorio de trabajo. Coincide con el tipo de archivo con la extensión de archivo.

“El valor predeterminado es guardar el último trazado que se muestra y, para un tamaño predeterminado.

…Esto significa que el único argumento que necesita proporcionar es el nombre de archivo.”

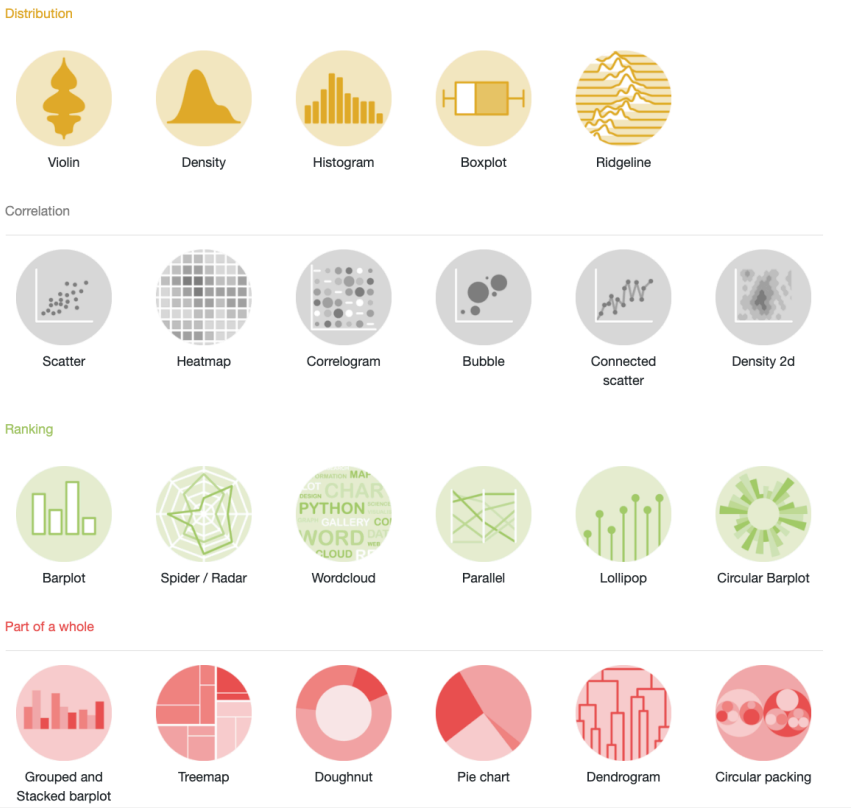
ggplot: más allá de las graficas de puntos

geom se refiere a un objeto geométrico. Determina la "forma" de los elementos de la grafico.

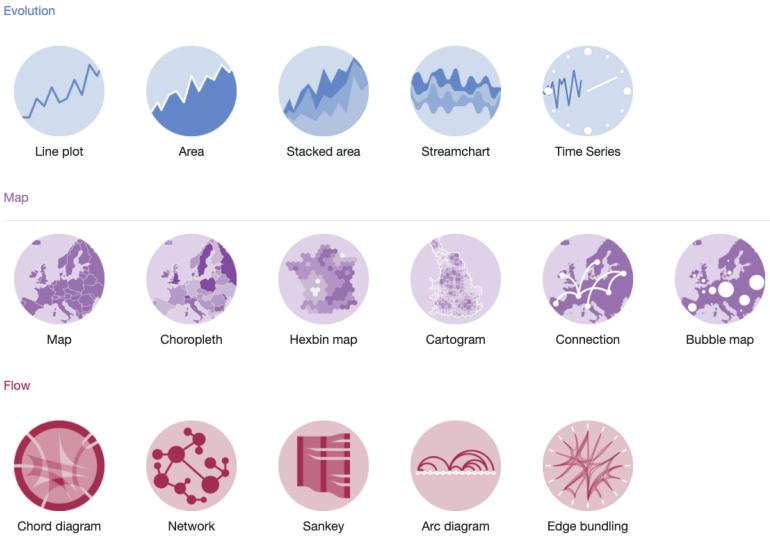
Some common geoms:

geom	Description
geom_point()	Points
geom_line()	Lines
geom_ribbon()	Ribbons, y range with continuous x values
geom_polygon()	Polygon, a filled path
geom_pointrange()	Vertical line with a point in the middle
geom_linerange()	An interval represented by a vertical line
geom_path()	Connect observations in original order
geom_histogram()	Histograms
geom_text()	Text annotations
geom_violin()	Violin plot (another name for a beanplot)
geom_map()	Polygons on a map

THE R GRAPH GALLERY



Different plots with example code!



https://www.r-graph-gallery.com/index.html

Conocimiento general

General knowledge

Ggplot2

Animation

Interactivity

3D

ggplot2

`ggplot2` is a `R` package dedicated to data visualization. It can greatly improve the quality and aesthetics of your graphics, and will make you much more efficient in creating them.

`ggplot2` allows to build almost any type of chart. The R graph gallery focuses on it so almost every section there starts with `ggplot2` examples.

This page is dedicated to general `ggplot2` tips that you can apply to any chart, like customizing a title, adding annotation, or using faceting.

If you're new to `ggplot2`, a good starting point is probably this online course.

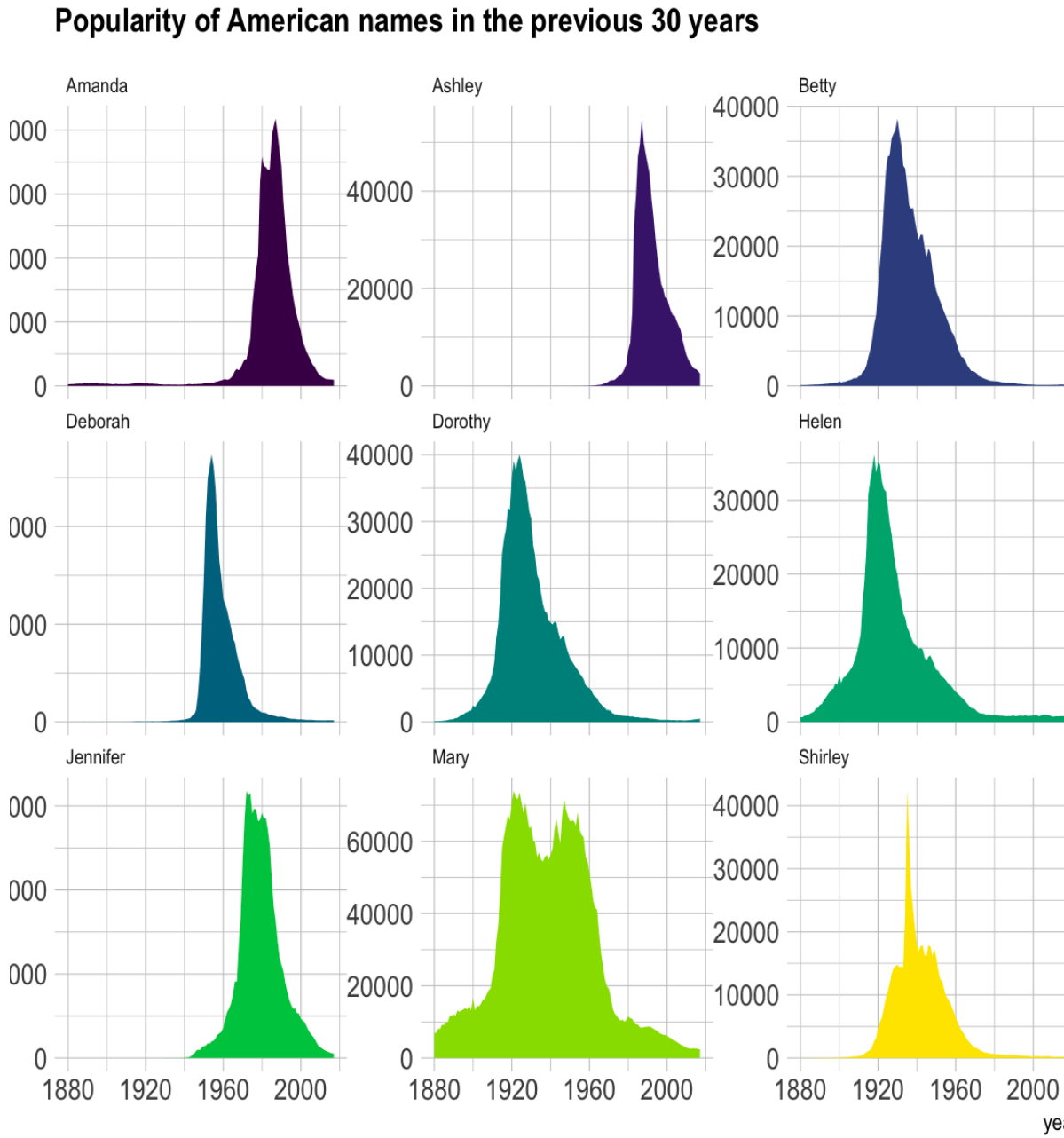
Multi-panel

facet_wrap()

y

facet_grid()

funciones comunes para graficos multipanel con ggplot2

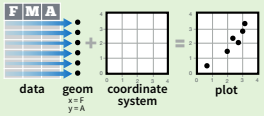


Data Visualization with ggplot2 Cheat Sheet

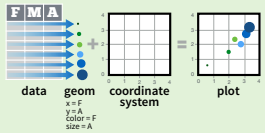
RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system.**



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **ggplot()** or **qplot()**

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

data

```
ggplot(mpg, aes(hwy, cty)) +
 geom_point(aes(color = cyl)) +
 geom_smooth(method ="lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

add layers, elements with +

layer = geom + default stat + layer specific mappings

additional elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

aesthetic mappings | data | geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point"**)**

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot()

Returns the last plot

ggsave("plot.png", width = 5, height = 5)

Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

Graphical Primitives

```
a <- ggplot(seals, aes(x = long, y = lat))
b <- ggplot(economics, aes(date, unemploy))
```

a + **geom_blank()**
(Useful for expanding limits)

a + **geom_curve(**aes(yend = lat + delta_lat, xend = long + delta_long, curvature = z)**)**
x, xend, y, yend, alpha, angle, color, curvature, linetype, size

b + **geom_path(**lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, group, linetype, size

b + **geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, group, linetype, size

a + **geom_rect(**aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

b + **geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)**
x, ymax, ymin, alpha, color, fill, group, linetype, size

a + **geom_segment(**aes(yend=lat + delta_lat, xend = long + delta_long)**)**
x, xend, y, yend, alpha, color, linetype, size

a + **geom_spoke(**aes(yend = lat + delta_lat, xend = long + delta_long)**)**
x, y, angle, radius, alpha, color, linetype, size

One Variable

Continuous

```
c <- ggplot(mpg, aes(hwy))
```

c + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size
a + geom_area(aes(y = ..density..), stat = "bin")

c + **geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

c + **geom_dotplot()**
x, y, alpha, color, fill

c + **geom_freqpoly()**
x, y, alpha, color, group, linetype, size
a + geom_freqpoly(aes(y = ..density..))

c + **geom_histogram(**binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
a + geom_histogram(aes(y = ..density..))

Discrete

```
d <- ggplot(mpg, aes(fl))
```

d + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Two Variables

Continuous X, Continuous Y

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + **geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + **geom_jitter(**height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

e + **geom_point()**
x, y, alpha, color, fill, shape, size, stroke

e + **geom_quantile()**
x, y, alpha, color, group, linetype, size, weight

e + **geom_rug(**sides = "bl"**)**
x, y, alpha, color, linetype, size

e + **geom_smooth(**method = lm**)**
x, y, alpha, color, fill, group, linetype, size, weight

e + **geom_text(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

```
f <- ggplot(mpg, aes(class, hwy))
```

f + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight

f + **geom_boxplot()**
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + **geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill, group

f + **geom_violin(**scale = "area"**)**
x, y, alpha, color, fill, group, linetype, size, weight

Discrete X, Discrete Y

```
g <- ggplot(diamonds, aes(cut, color))
```

g + **geom_count()**
x, y, alpha, color, fill, shape, size, stroke

Continuous Bivariate Distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + **geom_bin2d(**binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + **geom_density2d()**
x, y, alpha, colour, group, linetype, size

h + **geom_hex()**
x, y, alpha, colour, fill, size

Continuous Function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + **geom_area()**
x, y, alpha, color, fill, linetype, size

i + **geom_line()**
x, y, alpha, color, group, linetype, size

i + **geom_step(**direction = "hv"**)**
x, y, alpha, color, group, linetype, size

Visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

j + **geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + **geom_errorbar()**
x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + **geom_linerange()**
x, ymin, ymax, alpha, color, group, linetype, size

j + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

Maps

```
data <- data.frame(murder = USArrests$Murder,
  state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

k + **geom_map(**aes(map_id = state), map = map**)** + **expand_limits(**x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Three Variables

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
l <- ggplot(seals, aes(long, lat))
```

l + **geom_contour(**aes(z = z)**)**
x, y, z, alpha, colour, group, linetype, size, weight

l + **geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

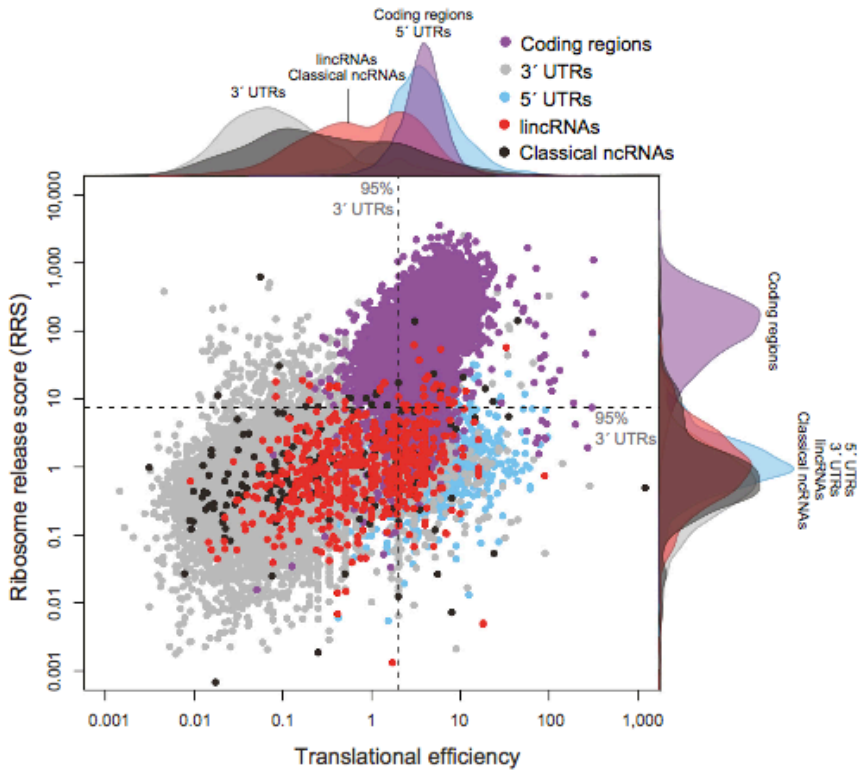
l + **geom_tile(**aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size, width

Lo que hemos aprendido

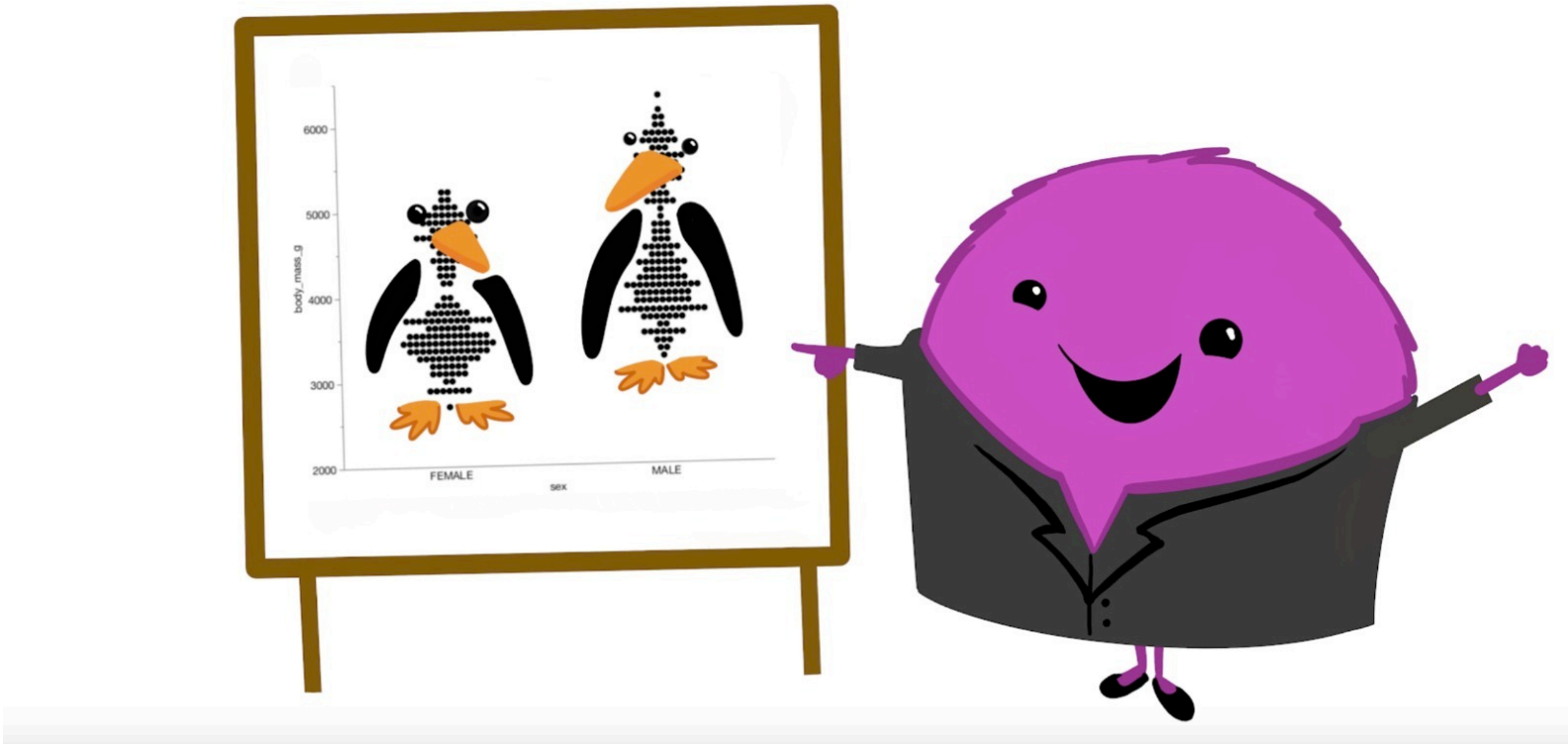
- Inspiración

- Gráficos base en R
- Gráfica de puntos
- Manipulación de graficos
  - Otros tipos básicos de graficos
  - Figuras multipanel y diseño

- Introducción a ggplot

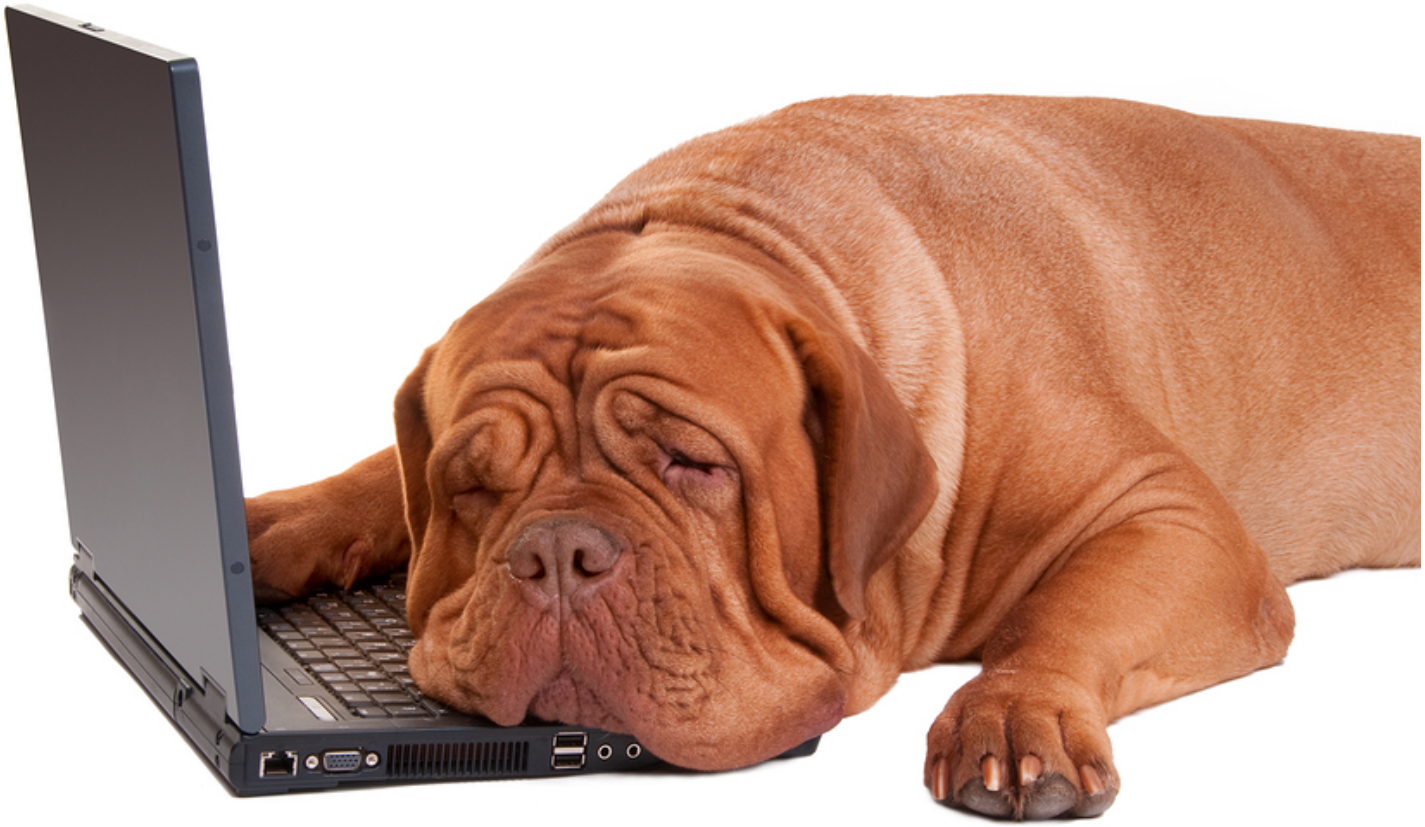


“As you can see,
you definitely belong in data science.”



Una nota sobre la graficación en R

A veces está bien arreglarlo en powerpoint



Decida si vale la pena el esfuerzo. A menudo, si. Pero no siempre!