

# Visualización de datos con ggplot2

## 3º Encuentro R en Rosario

María Belén Allasia

Rosario | Marzo 2019

# 11 de Febrero DÍA DE LA MUJER Y LA NIÑA EN LA CIENCIA

Fue proclamado en 2015 por la Asamblea General de las Naciones Unidas con el fin de lograr el acceso y la participación plena y equitativa en la ciencia para las mujeres y las niñas, además para lograr la igualdad de género y el empoderamiento de las mujeres y las niñas.



**Ada King, Condesa de Lovelace**  
Primer Programadora  
Generó los primeros algoritmos



**Almirante Grace Hopper**  
Inventó el Primer Compilador  
Lenguajes independientes de las computadoras



**Dorothy Vaughan, Leslie Hunter & Vivian Adair**  
Primeras "operarias" de computadoras de la NASA

# 8 de Marzo DÍA INTERNACIONAL DE LA MUJER

El **Día Internacional de la Mujer Trabajadora**, también llamado **Día Internacional de la Mujer** o solamente **Día de la mujer**, tras ser institucionalizado por decisión de las Naciones Unidas en 1975, conmemora la lucha de la mujer por su participación, en pie de equidad de oportunidades muchas veces menores ante las del varón, en la sociedad y en su desarrollo íntegro como persona.



## ¿Cómo se manifiesta esta *rebelión de las mujeres en el mundo R?*

R-Ladies es una organización internacional cuya misión es promover la diversidad de género en la comunidad R, a través de reuniones presenciales, virtuales y mentorías en un espacio amigable.



Curiosidades a nivel internacional...

<https://gqueiroz.shinyapps.io/rshinylady/>



**145**  
CIUDADES

**45**  
PAÍSES

**42000+**  
PARTICIPANTES

# SeR MujeR

¿Y a nivel local?



**R-Ladies Buenos Aires**  
@RLadies\_BuenosAires  
R-Ladies Buenos Aires es parte de una organización mundial para promover la diversidad de géneros en la comunidad R. #RLadies #Rstats  
0 Buenos Aires, Argentina  
0 Meetups con 155 miembros...  
0 Se unió en enero de 2017

**R-Ladies Resistencia Corrientes**  
@RLadies\_resiste  
R-Ladies Resistencia Corrientes  
0 Buenos Aires, Argentina  
0 Meetups con 155 miembros...  
0 Se unió en enero de 2017

**R-Ladies Santa Fe**  
@RLadiesSantaFe  
R-Ladies Santa Fe es parte de una organización mundial para promover la diversidad de géneros en la comunidad R. #RLadies #Rstats  
0 Santa Fe, Argentina  
0 Meetups con 155 miembros...  
0 Se unió en mayo de 2018  
0 Fecha de nacimiento 01 de agosto

¿Y a nivel local?



# A graficaR!!

**Ahora sí, manos a la obra con el objetivo de hoy:  
Vamos a representar un conjunto de datos con ggplot2!**

El material de este taller está basado en la clase de gráficos del curso “Introducción a R” dictado en la Facultad de Ciencias Económicas y Estadística (UNR) por los docentes:

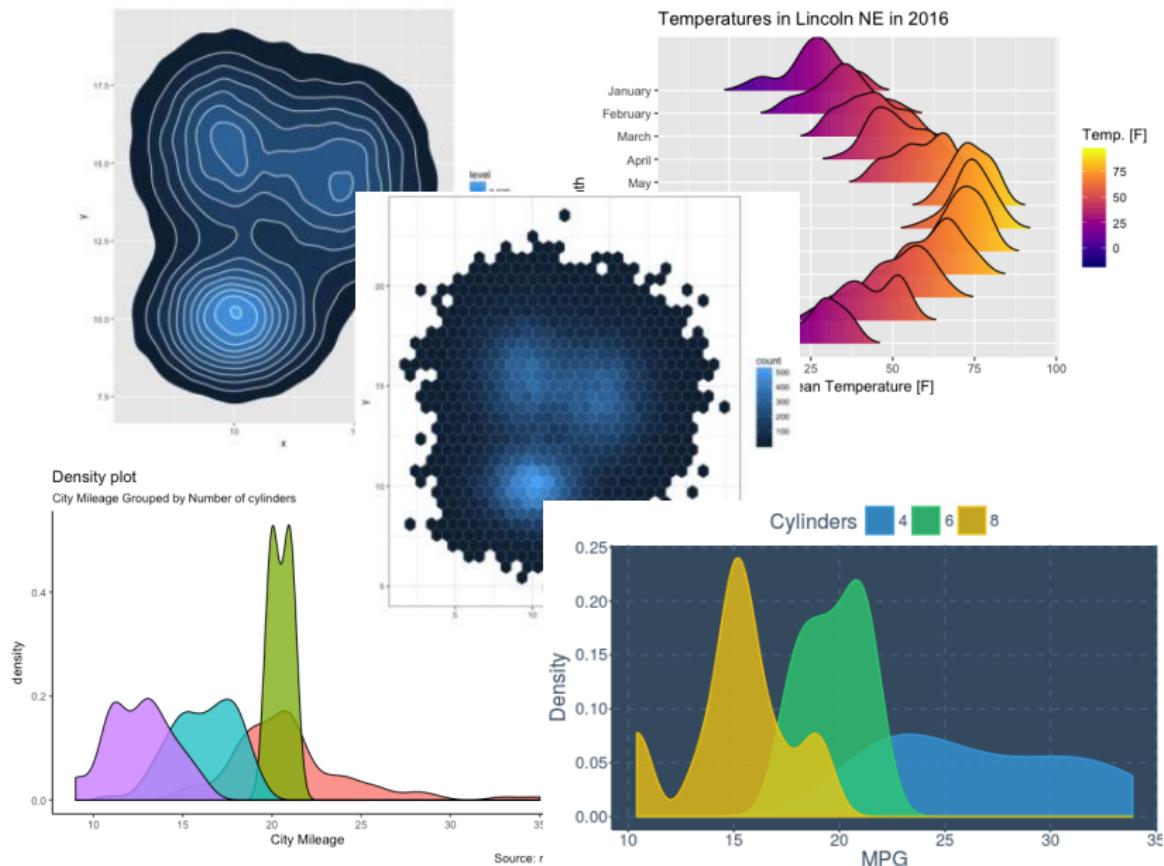
- Mg. Diego Marfetán
- Mg. Marcos Prunello
- Lic. María Belén Allasia

PREMISA:

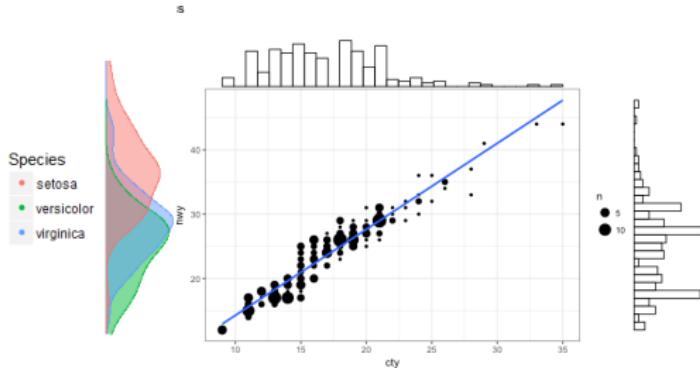
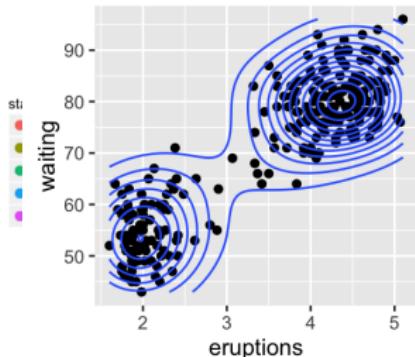
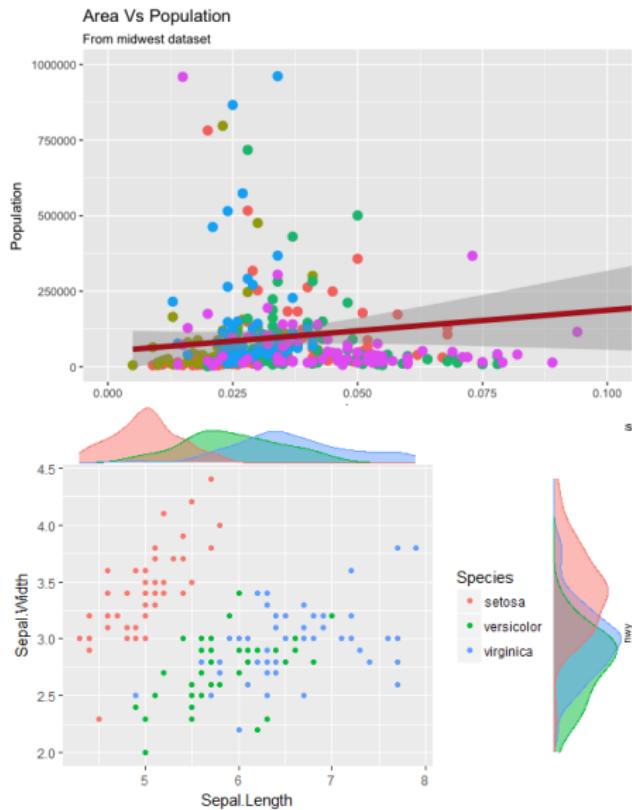
**Nunca comenzar un análisis aplicando un método estadístico.  
Siempre comenzar con un gráfico.**

- Esa es, para muchos, la regla fundamental del análisis de datos.
- Antes que nada, hay que intentar que los datos respondan nuestra pregunta de investigación por sí mismos.
- Y la mejor manera para dejar que se expresen, es graficarlos y que nos muestren lo que nos quieren decir!
- Por suerte, R es una gran herramienta para hacer gráficos!!

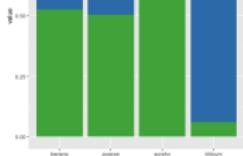
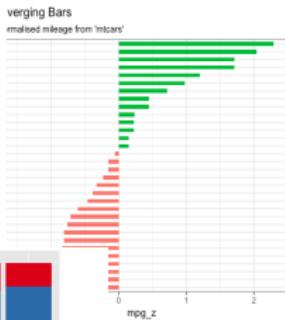
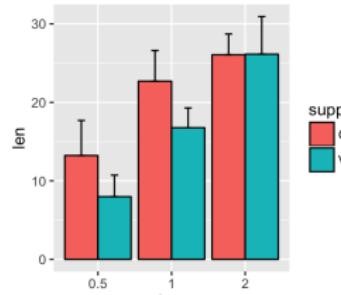
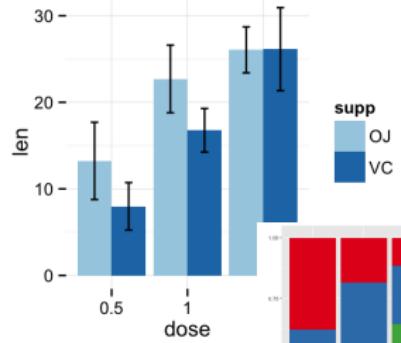
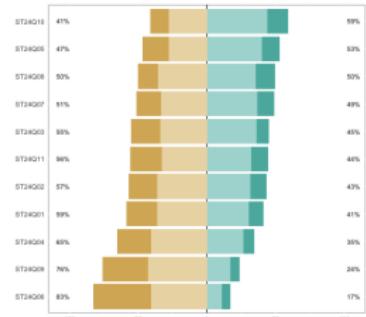
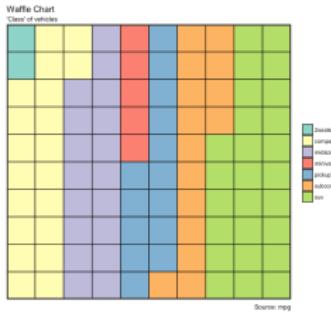
# Ejemplos: Densidades



# Ejemplos: Dispersión



## Ejemplos: Barras

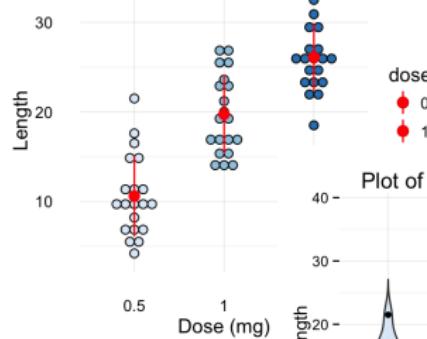
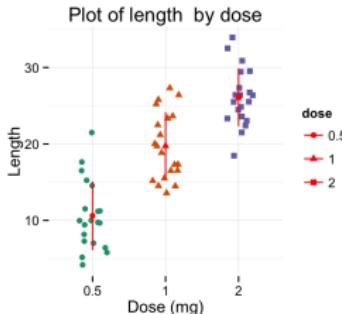
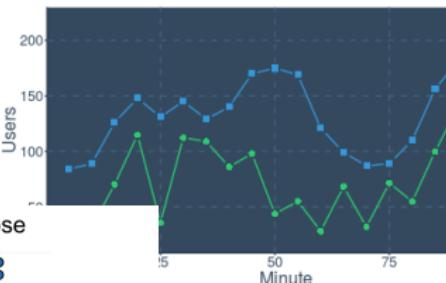


# Ejemplos: Puntos, boxplots, líneas

Time Series of Returns Percentage  
Drawn From Wide Data format

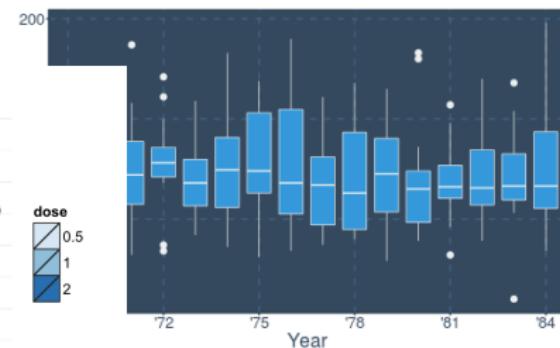
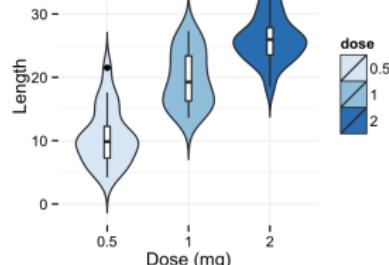


Measure Actual Jitter

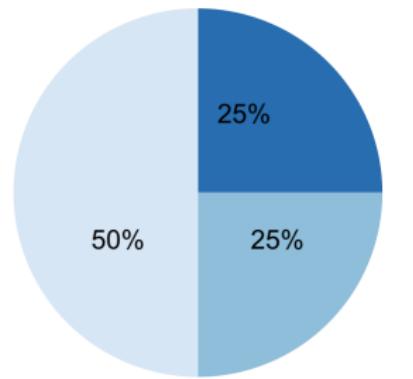
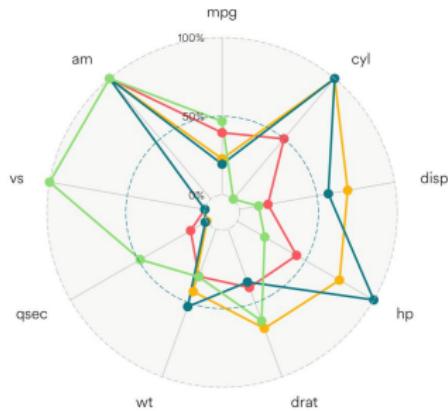
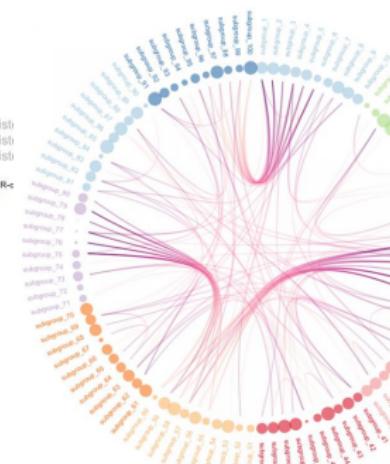
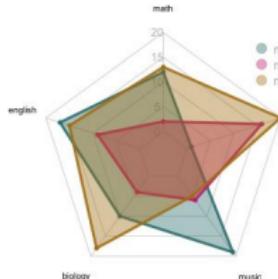
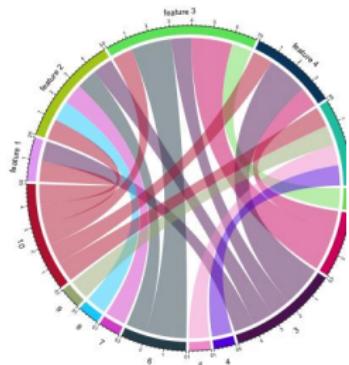


dose  
0.5  
1

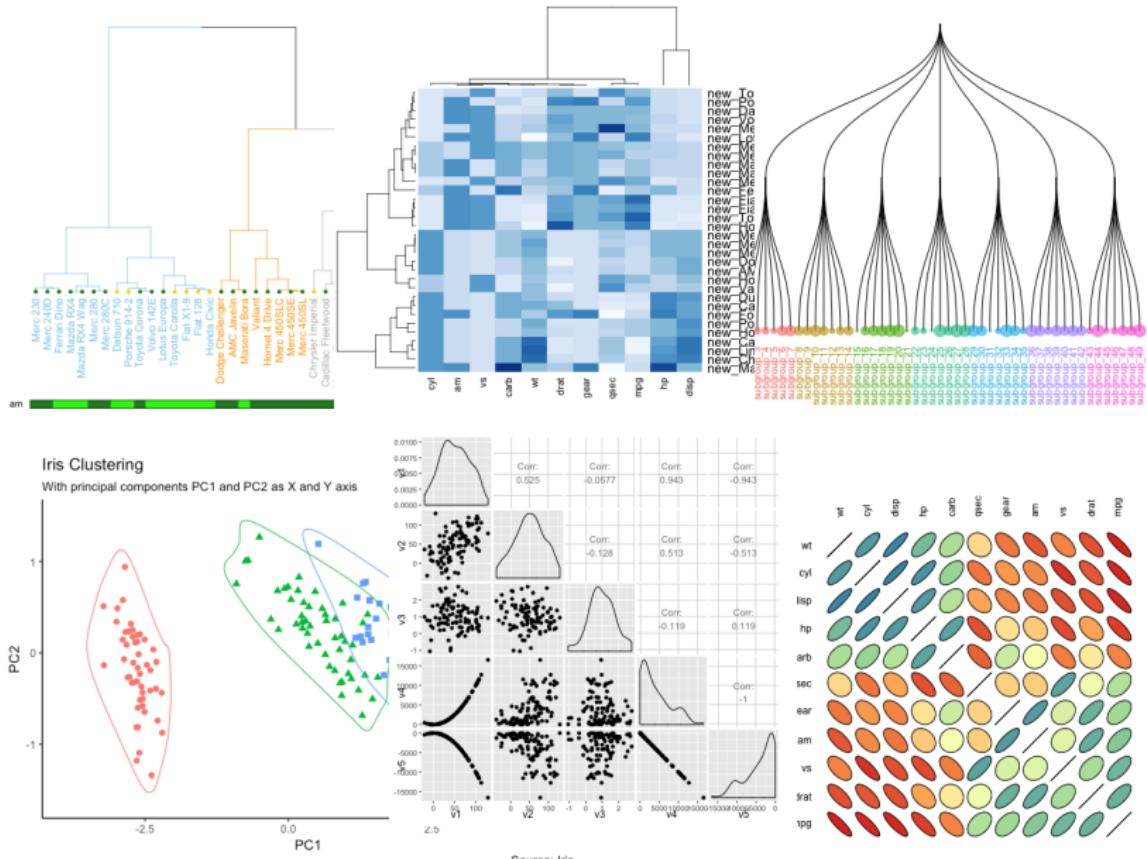
Plot of length by dose



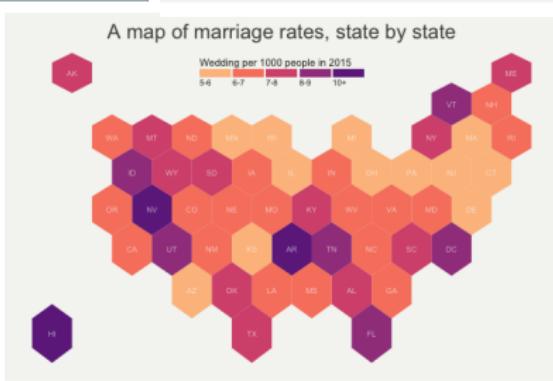
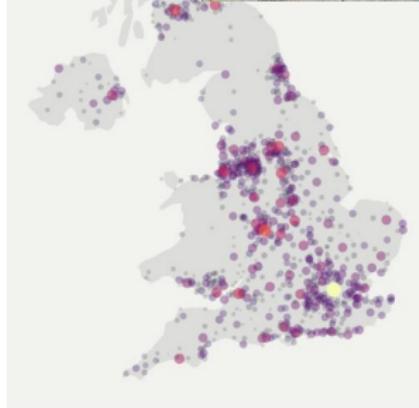
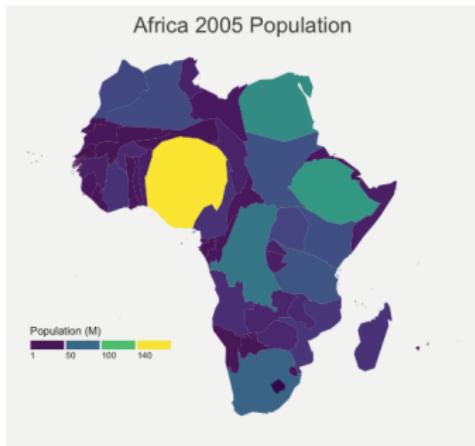
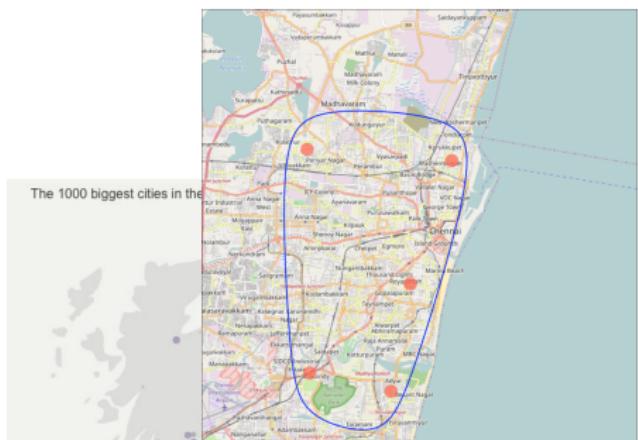
# Ejemplos: Circulares, radiales



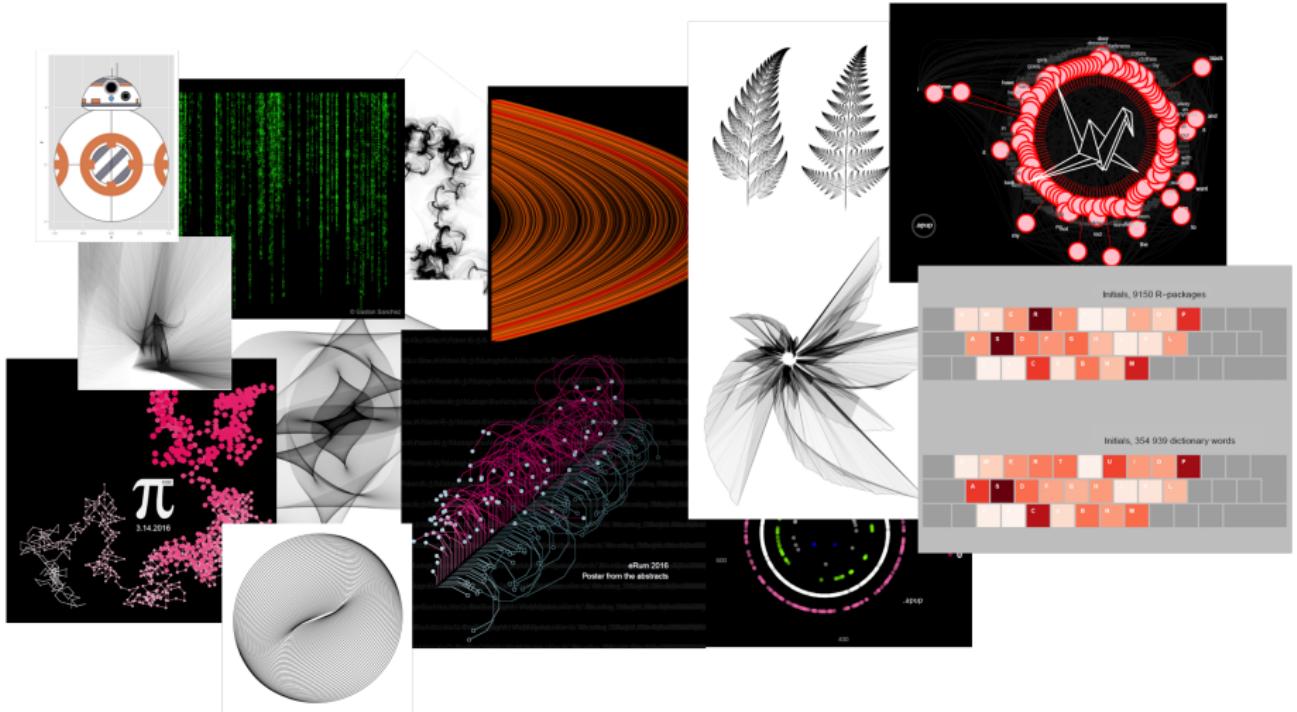
# Ejemplos: Asociación, conglomerados, PCA



# Ejemplos: Georreferenciamiento



# Ejemplos: artísticos. . .



# Los datos

- Vamos a hacer algunos gráficos con el dataset `compensacion.csv`, con datos sobre producción de manzanas (variable `fruta`, en kg).
- En su desarrollo los árboles habían sido injertados en rizomas de diferentes anchos (variable `raiz`, en mm), técnica que se usa para controlar el tamaño de los árboles sin modificar los atributos de los frutos (*rootstocks*).
- Además, algunos árboles están en huertos con pastoreo de ganado y otros no. El pastoreo puede reducir la cantidad de pasto, que puede competir con los árboles por los recursos del suelo.

## Los datos

```
library(readr)
library(ggplot2)
datos <- read_csv("compensacion.csv")
datos
```

```
# A tibble: 40 x 4
  raiz fruta pastoreo altura
  <dbl> <dbl>    <int>   <dbl>
1  6.22  59.8      1     2.1
2  6.49  61.0      1     2.6
3  4.92  14.7      1     2.4
4  5.13  19.3      1     2.7
5  5.42  34.2      1     2.3
6  5.36  35.5      1     2
7  7.61  87.7      1     2.1
8  6.35  63.2      1     2.8
9  4.97  24.2      1     2.1
10 6.02  64.2      1     2.6
```

# GraficaR

- Antes de comenzar, vamos a convertir a factor la variable `pastoreo`, teniendo en cuenta que el 1 es “sin pastoreo” y el 2 es “con pastoreo”.

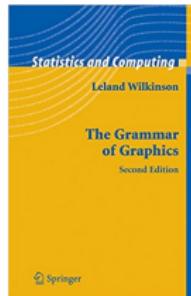
```
library(dplyr)
datos <- mutate(datos, pastoreo = factor(pastoreo, levels = 1:2, labels = c("Sin pastoreo",
  "Con pastoreo")))
datos
```

```
# A tibble: 40 x 4
  raiz fruta pastoreo     altura
  <dbl> <dbl> <fct>      <dbl>
1  6.22  59.8 Sin pastoreo  2.1 
2  6.49  61.0 Sin pastoreo  2.6 
3  4.92  14.7 Sin pastoreo  2.4 
4  5.13  19.3 Sin pastoreo  2.7 
5  5.42  34.2 Sin pastoreo  2.3 
6  5.36  35.5 Sin pastoreo  2    
7  7.61  87.7 Sin pastoreo  2.1 
8  6.35  63.2 Sin pastoreo  2.8 
9  4.97  24.2 Sin pastoreo  2.1 
10 6.93  64.3 Sin pastoreo  2.6 
# ... with 30 more rows
```

## Opciones de paquetes y sistemas de gráficos

- En R existen tres sistemas para realizar gráficos: **base**, **lattice** y **ggplot2**.

# ggplot2



- **gg** viene de *Grammar of Graphics*, una estructura ideada por Leland Wilkinson (*Grammar of Graphics*, 1999).
- Así como la gramática de un lenguaje estudia la estructura de las palabras, la manera en que se combinan y las reglas que hay que seguir para formar oraciones, la gramática de gráficos provee un sistema para combinar elementos gráficos que den como resultado figuras para mostrar datos de manera visualmente significativa.
- Este sistema fue popularizado por Hadley Wickham (CEO de RStudio) cuando creó el paquete *ggplot* basándose en estas ideas.

# ggplot2

- Los gráficos de ggplot2 se construyen tomando como base esta gramática, que consta de dos principios fundamentales:
  - Los gráficos están formados por distintas **capas** (*layers*) de elementos gramaticales.
  - Las capas se vinculan entre sí a través de un **mapeo estético** (*aesthetic mapping*)
- Las **capas** son como los adjetivos o sustantivos y el **mapeo estético** es como el conjunto de reglas gramaticales que ayudan a ensamblar ese vocabulario.

# ggplot2

- Hay 7 tipos elementos gráficos, pero los 3 esenciales, de presencia obligatoria, son:
  - **Datos** (*data*): los datos que queremos representar
  - **Aesthetics** (*aes*): describe cómo las variables en el data set se mapean (vinculan) a las propiedades visuales (*aesthetics*). Por ejemplo, cuál es la variable que van en el eje horizontal, cuál en el vertical, cuál la que indica de qué color va cada punto, etc.
  - **Geometrías** (*geometries, geom*): el aspecto o forma que los datos van a tomar en el gráfico (puntos, líneas, boxplots, etc.).

# ggplot2

- Los restantes tipos de elementos gráficos son opcionales:
  - **Facetas (facet)**: para graficar con paneles (por grupos).
  - **Estadísticas (stats)**: para agregar estadísticas o resultados de métodos estadísticos (como una curva ajustada).
  - **Coordenadas (coordinates)**: para transformar el espacio sobre el cual se grafican los datos (ej: escala logarítmica, coordenadas polares, etc.).
  - **Temas (themes)**: “tinta libre de datos” (*non-data ink*), controla el aspecto general (es lo que uno llamaría intuitivamente la parte estética).

# Graficando con ggplot2

- La primera capa contiene al 1º elemento que son los **datos** (obviamente!)

Data

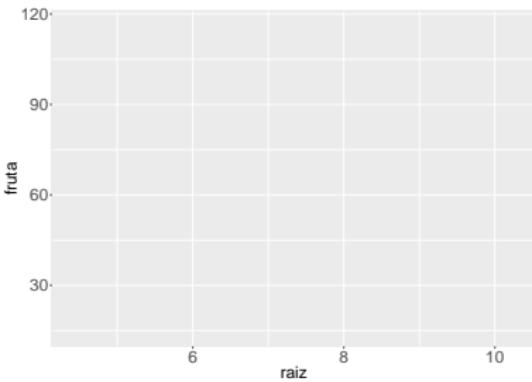
```
datos
# A tibble: 40 x 4
  raiz fruta pastoreo     altura
  <dbl> <dbl> <fct>      <dbl>
1  6.22  59.8 Sin pastoreo  2.1
2  6.49  61.0 Sin pastoreo  2.6
3  4.92  14.7 Sin pastoreo  2.4
4  5.13  19.3 Sin pastoreo  2.7
5  5.42  34.2 Sin pastoreo  2.3
6  5.36  35.5 Sin pastoreo  2
7  7.61  87.7 Sin pastoreo  2.1
8  6.35  63.2 Sin pastoreo  2.8
9  4.97  24.2 Sin pastoreo  2.1
10 6.93  64.3 Sin pastoreo  2.6
# ... with 30 more rows
```

# Graficando con ggplot2

- En la primera capa también incluimos al 2º elemento son las **aesthetics**, donde indicaremos los mapeos que utilizaremos para vincular las distintas capas.
- Vamos a hacer un diagrama de dispersión de **fruta** vs **raiz**, por lo que vamos a mapear la variable **fruta** al elemento estético **x** y la variable **raiz** al elemento estético **y**.

```
library(ggplot2)
ggplot(datos, aes(x = raiz, y = fruta))
```

Aesthetics  
Data

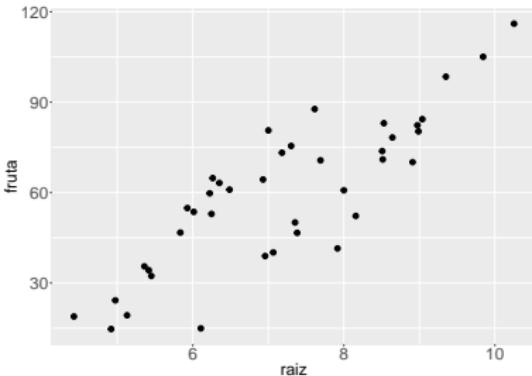


# Graficando con ggplot2

- El 3º elemento obligatorio es una **geometry**, sin al menos una de ellas no tenemos ningún gráfico!
- Por ejemplo, para un gráfico de dispersión, lo que necesitamos son puntos:

```
ggplot(datos, aes(x = raiz, y = fruta)) +  
  geom_point()
```

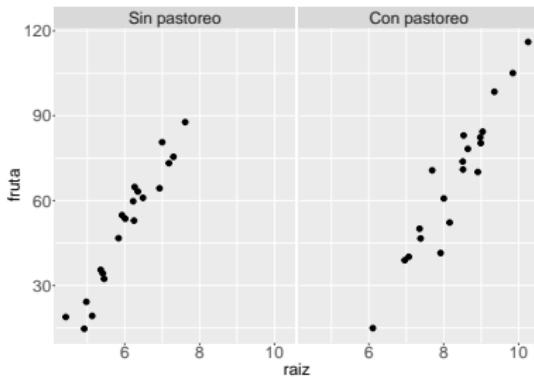
Geometries  
Aesthetics  
Data



# Graficando con ggplot2

- A partir de acá podemos considerar incluir o no los restantes elementos gramaticales optionales.
- Por ejemplo, podemos agregar un elemento tipo **facet** para hacer un panel para cada tipo de pastoreo:

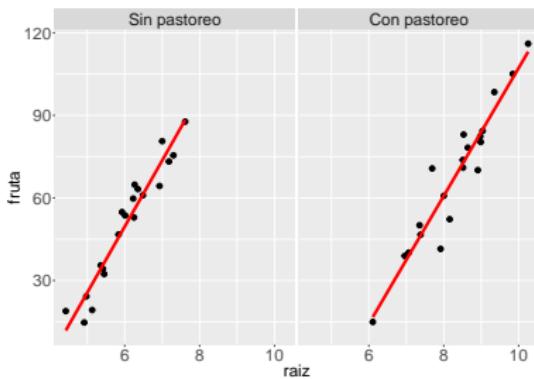
```
ggplot(datos, aes(x = raiz, y = fruta)) +  
  geom_point() +  
  facet_wrap(~ pastoreo)
```



# Graficando con ggplot2

- Podemos agregar una nueva capa con un elemento de tipo *stats*, por ejemplo para agregar la recta de regresión lineal simple:

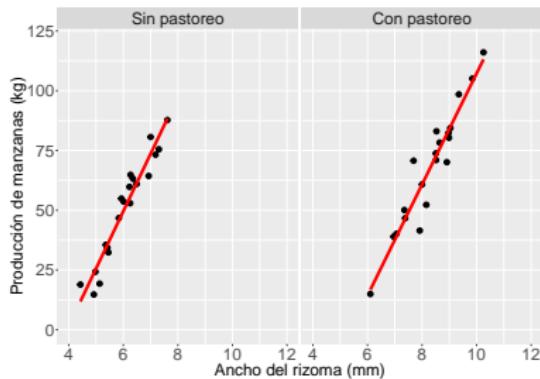
```
ggplot(datos, aes(x = raiz, y = fruta)) +  
  geom_point() +  
  facet_wrap(~ pastoreo) +  
  stat_smooth(method = "lm", se = F, col = "red")
```



# Graficando con ggplot2

- Ahora podemos hacer algo con las **coordinates**, agregando una capa para el eje horizontal y otra para el eje vertical:

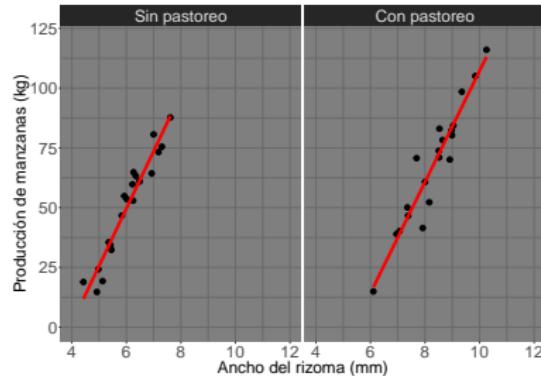
Coordinates  
Statistics  
Facets  
Geometries  
Aesthetics  
Data



```
ggplot(datos, aes(x = raiz, y = fruta)) +  
  geom_point() +  
  facet_wrap(~ pastoreo) +  
  stat_smooth(method = "lm", se = F, col = "red") +  
  scale_x_continuous("Ancho del rizoma (mm)", limits = c(4, 12)) +  
  scale_y_continuous("Producción de manzanas (kg)", limits = c(0, 120))
```

# Graficando con ggplot2

- Finalmente podemos modificar el aspecto general con una o varias capas **temáticas** (con estilos predefinidos o personalizados):



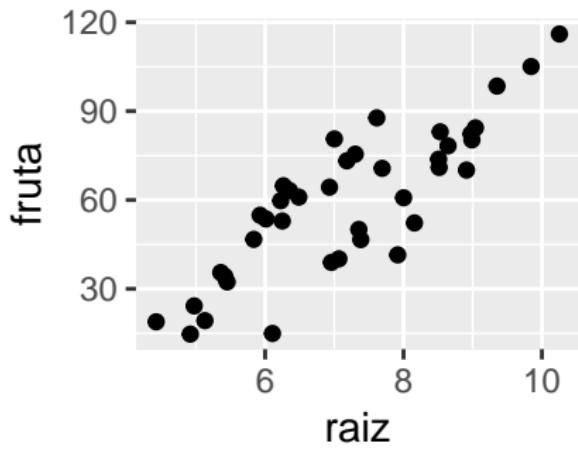
```
ggplot(datos, aes(x = raiz, y = fruta)) +  
  geom_point() +  
  facet_wrap(~ pastoreo) +  
  stat_smooth(method = "lm", se = F, col = "red") +  
  scale_x_continuous("Ancho del rizoma (mm)", limits = c(4, 12)) +  
  scale_y_continuous("Producción de manzanas (kg)", limits = c(0, 120)) +  
  theme_dark()
```

# Extra! Personalización de los puntos

- Vamos a retomar el diagrama de dispersión inicial para ver otras opciones.

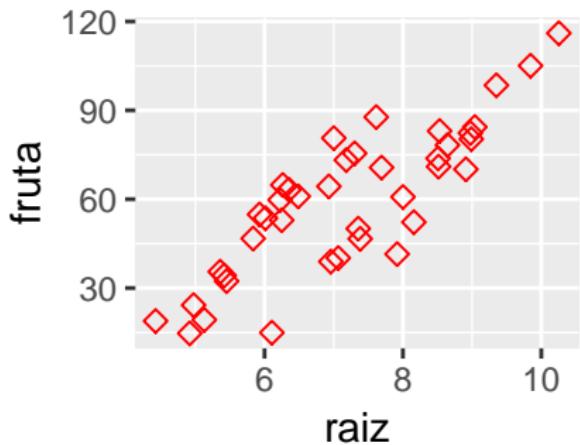
Gráfico original:

```
ggplot(datos, aes(x = raiz, y = fruta)) +  
  geom_point()
```



Cambiar el tamaño, color y forma de todos los puntos:

```
ggplot(datos, aes(x = raiz, y = fruta)) +  
  geom_point(color = "red", size = 2, shape = 23)
```



## Extra! Personalización de los puntos

0 □ 1 ○ 2 ▲ 3 + 4 ×

5 ◇ 6 ▽ 7 ☒ 8 \* 9 ☢

10 ⊕ 11 ☀ 12田 13⊗ 14□

15 ■ 16 ● 17▲ 18◆ 19●

20 ● 21 ● 22 ■ 23 ◇ 24 ▲ 25 ▽

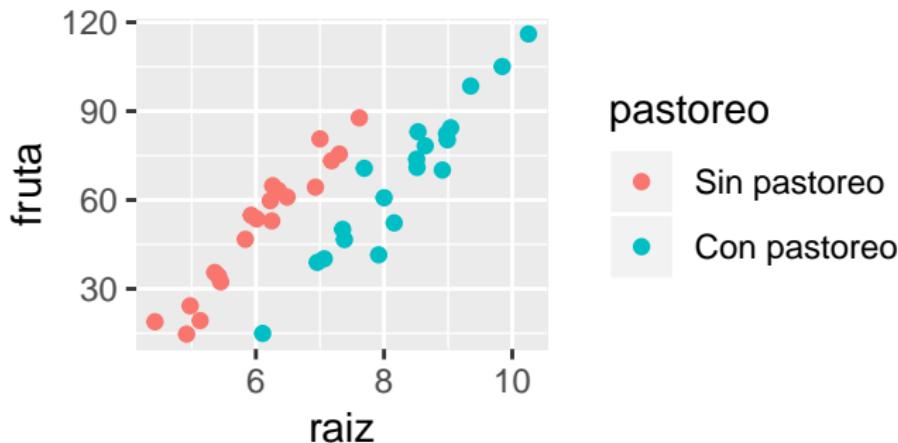
## Más opciones para *aesthetics*

- En el ejemplo anterior, hemos modificado la forma, el color y el tamaño para todos los puntos (*aesthetics setting*).
- Esas son **especificaciones estéticas fijas** que se definen dentro del *geom* correspondiente (*geom\_point*) y no forman parte de la función `aes()`.
- Por ejemplo, le asignamos un valor, el color `red` al argumento `color` de la función `geom_point`.
- Otra cosa distinta es emplear **especificaciones de mapeo estético** (*aesthetics mapping*). Estas se definen dentro de la función `aes()`.
- Por ejemplo, si quiero que a través de todas las capas el color esté “mapeado” a la variable `pastoreo` para que cada uno de los dos grupos tenga su propio color, deberé agregar `aes(color = pastoreo)`.

## Más opciones para *aesthetics*

- Esto inmediatamente genera el mapeo que asocia al color de todos nuestros *geoms* con la variable *pastoreo* y crea automáticamente la correspondiente leyenda.

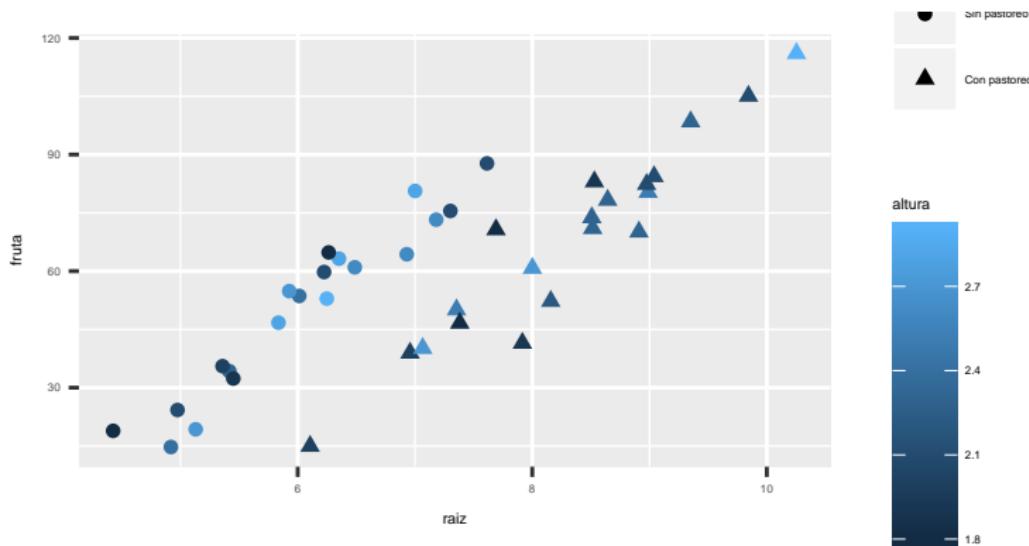
```
ggplot(datos, aes(x = raiz, y = fruta, color = pastoreo)) +  
  geom_point()
```



# Más opciones para *aesthetics*

- El mapeo estético del color puede hacerse en función de una variable continua:

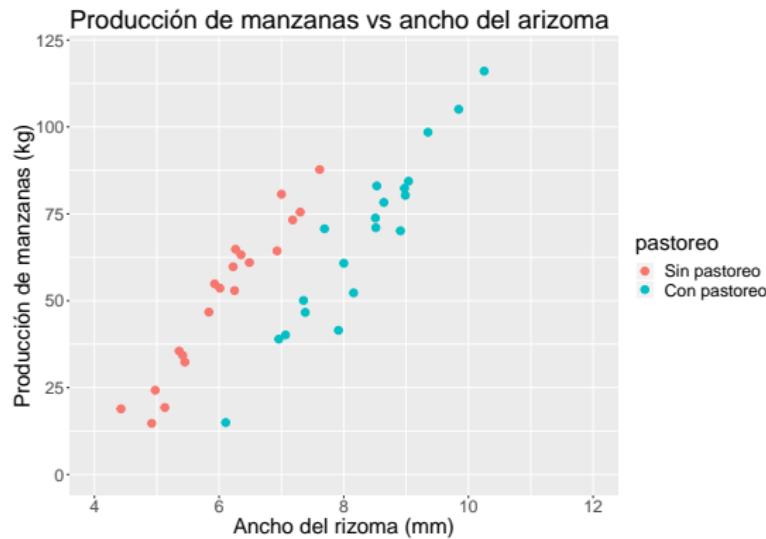
```
ggplot(datos, aes(x = raiz, y = fruta, color = altura, shape = pastoreo)) +  
  geom_point()
```



# Algunas opciones para editar la leyenda

Gráfico original:

```
g <- ggplot(datos, aes(x = raiz, y = fruta, color = pastoreo)) +  
  geom_point(size = 3) +  
  scale_x_continuous("Ancho del rizoma (mm)", limits = c(4, 12)) +  
  scale_y_continuous("Producción de manzanas (kg)", limits = c(0, 120)) +  
  gtitle("Producción de manzanas vs ancho del arizoma")  
g
```



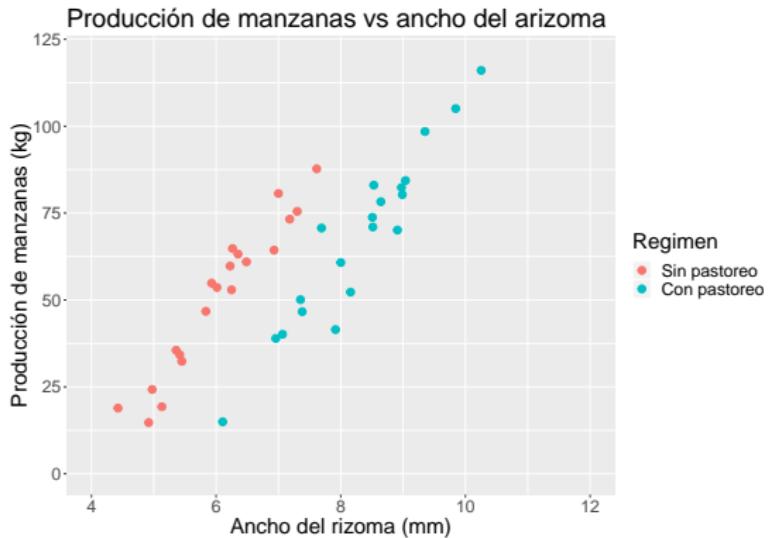
# Observación importante

- En el gráfico anterior le cambiamos el nombre a los ejes, usando las capas `scale_x_continuous` y `scale_y_continuous`.
- Las capas que permiten hacer este tipo de cambios comienzan con `scale_`, seguido por el mapeo estético que queremos editar (`x_`, `y_`, `color_`, `shape_`, etc.), y terminando con una palabra que describe cómo es ese mapeo (`continuous`, `discrete`, `manual`, etc.)

# Algunas opciones para editar la leyenda

Por ejemplo, si queremos cambiar el nombre de la leyenda usaremos:

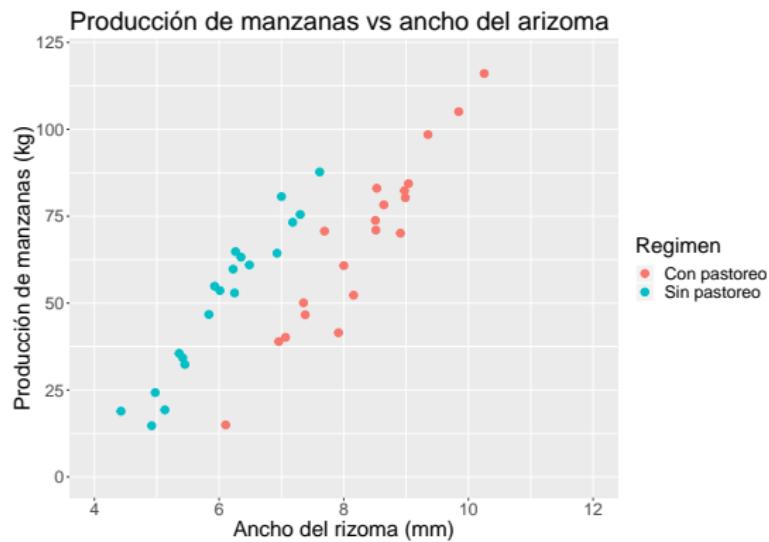
```
g + scale_color_discrete("Regimen")
```



# Algunas opciones para editar la leyenda

Cambiar el orden de las categorías en la leyenda:

```
g1 <- g + scale_color_discrete("Regimen",
                                limits = c("Con pastoreo", "Sin pastoreo"))
g1
```



# Observación importante

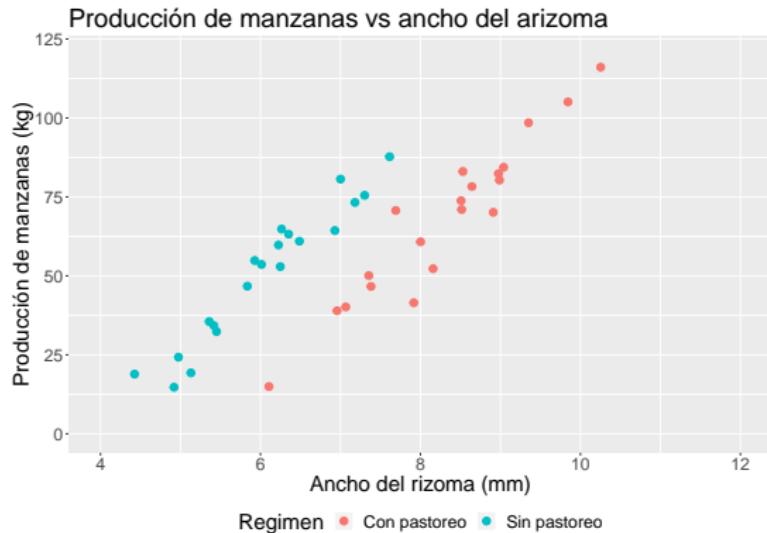
¿Podemos cambiar la posición de la leyenda?

- El lugar donde se ubica la leyenda no agrega ni quita información de los datos en sí, es parte de lo que se dice *non-data ink* y se lo configura como parte del estilo del gráfico, en `theme()`.
- Los argumentos de la capa `theme()` tienen nombres compuestos por palabras claves separadas por puntos.
- La primera palabra clave indica el elemento a editar (por ejemplo, `legend`).
- La segunda palabra clave indica qué aspecto le cambiaremos (por ejemplo, `legend.position`).

# Algunas opciones para editar la leyenda

Cambiar la posición de la leyenda:

```
g1 + theme(legend.position = "bottom")
```



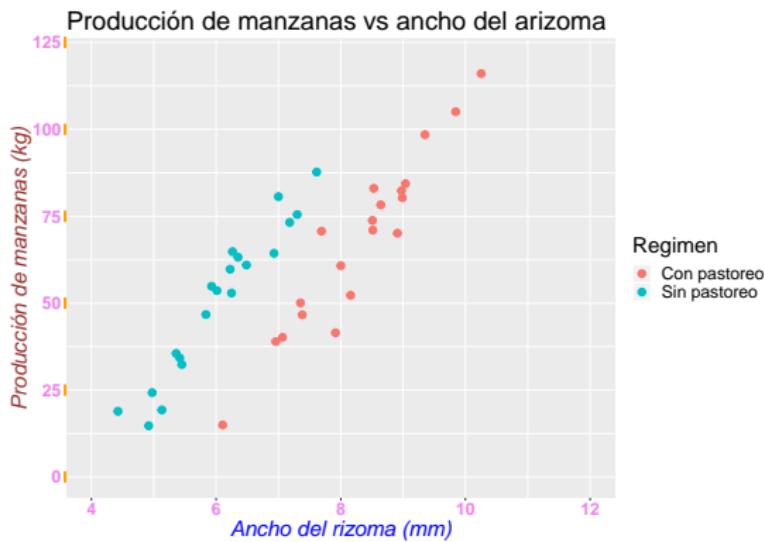
# Opciones para editar los ejes

- Ya vimos cómo cambiar el nombre de los ejes. Otras opciones de estilo para los ejes van dentro de `theme()`.
- Los argumentos comienzan con la palabra clave `axis`.
- Los atributos pueden ser `title`, `text`, `ticks`, etc.
- Y si queremos afectar sólo a uno de los ejes, terminamos con `x` o `y`.

# Opciones para editar los ejes

Ejemplo:

```
g1 + theme(  
  axis.title.x = element_text(color = "blue", size = 20, face = "italic"),  
  axis.title.y = element_text(color = "#993333", size = 20, face = "italic"),  
  axis.text = element_text(color = "violet", face = "bold"),  
  axis.ticks.y = element_line(color = "orange", size = 5)  
)
```



# Opciones para editar los ejes

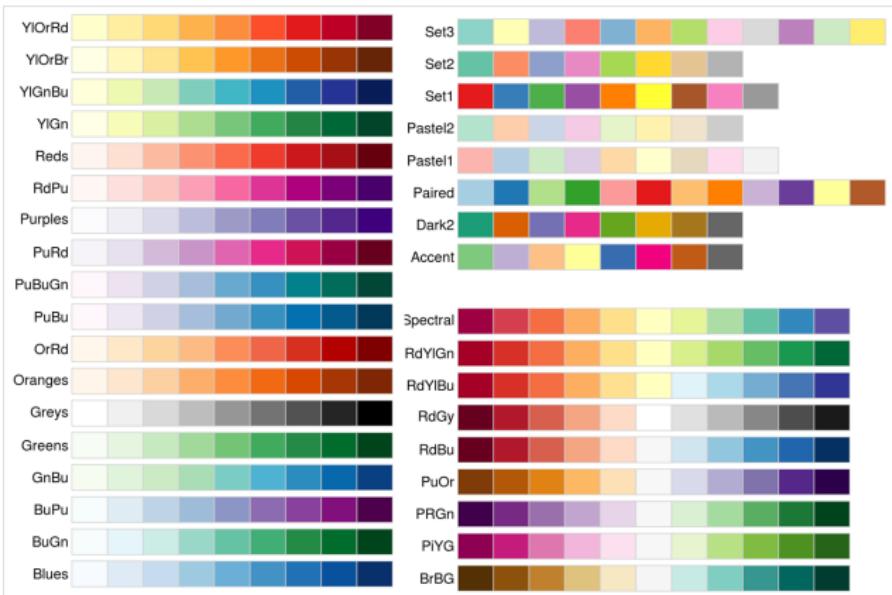
Ejemplo de cómo eliminar un componente con `element_blank()`:

```
g1 + theme(  
  axis.title = element_blank(),  
  axis.text = element_blank(),  
  axis.ticks = element_blank(),  
  panel.grid = element_blank(),  
  plot.title = element_blank(),  
  legend.position = "none"  
)
```



# Otras cosas útiles: Colores

- En algún ejemplo anterior usamos la capa `scale_color_manual()` para elegir los colores a utilizar.
- Otra opción es utilizar `scale_color_brewer()` para elegir una paleta de colores predefinidas, como las que se ven acá:



# Otras cosas útiles: Exportar los gráficos

Dos alternativas (entre otras):

- ① En la pestaña Plots en RStudio, con el botón Export.
  - Proporciona opciones para copiar el gráfico en el portapapeles o guardarlos como archivos de distintos formatos (.png, .jpg, .tiff, .pdf, etc.)
  - El cuadro de diálogo es muy útil para especificar el tamaño de la figura, la resolución y la ubicación.

## Otras cosas útiles: Exportar los gráficos

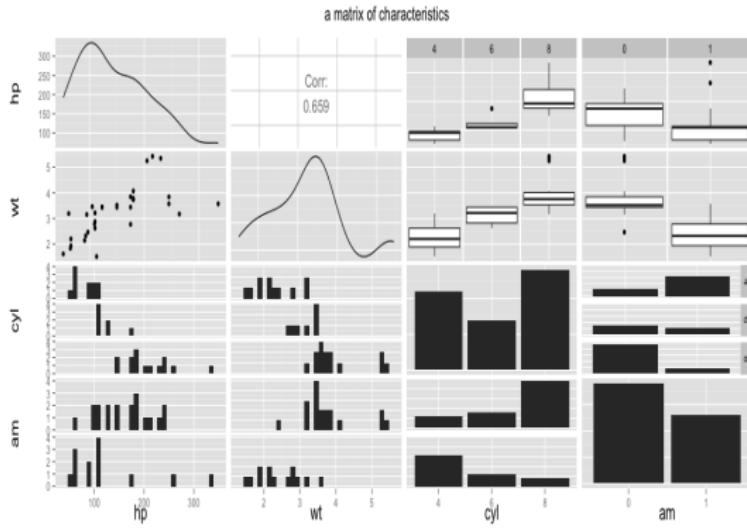
### ② Con la función `ggsave()` del paquete `ggplot2`.

- Es muy fácil de usar, muy útil para agregar en nuestro script si tenemos que hacer muchos gráficos!
- Por default guarda el último gráfico creado, o podemos indicarle el que nosotros queramos haciendo referencia a un objeto creado
- Identifica el formato automáticamente de la extensión que le pongamos en el nombre del archivo.

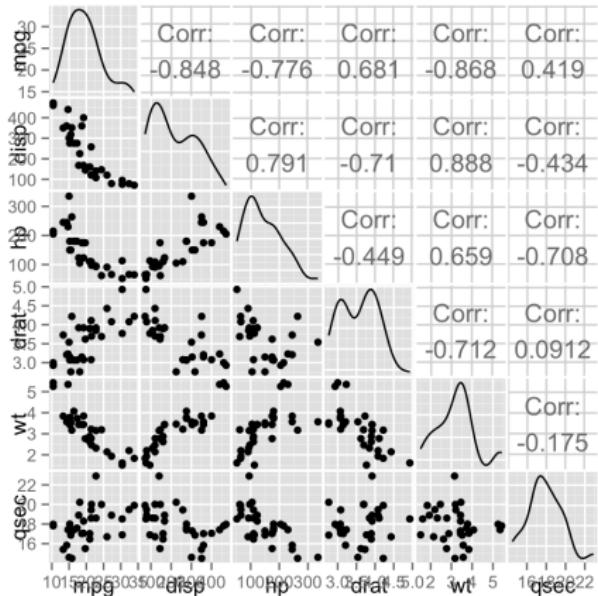
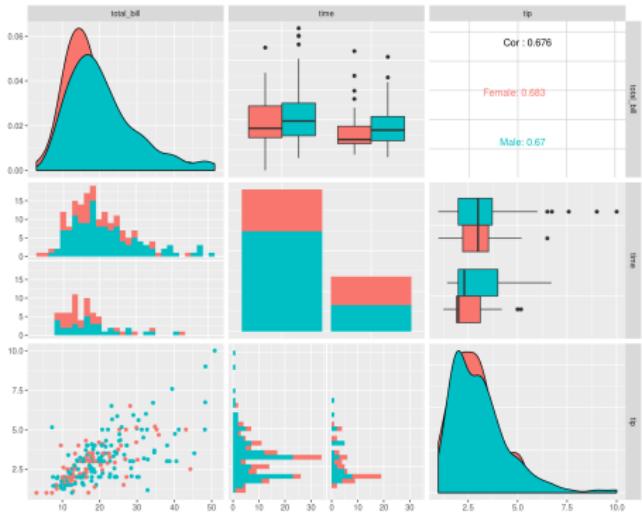
```
g <- ggplot(datos, aes(x = raiz, y = fruta)) + geom_point()
ggplot(datos, aes(x = raiz, y = fruta)) + geom_histogram()
# Guarda el último, es decir el histograma, en png:
ggsave("miDibujito.png")
# Guarda el diagrama de dispersión en pdf:
ggsave("miDibujito.pdf", g)
ggsave("miDibujito.jpg", g,
       width = 8, height = 5, units = "cm")
```

# Otras cosas útiles: paquetes que extienden ggplot2

- Hay paquetes que traen funciones para realizar gráficos ya predefinidos.
- Por ejemplo, el paquete [GGally](#) tiene muchas cosas lindas, entre ellas, la función `ggpairs()` que permite hacer **matrices de gráficos** para ver la relación de a pares entre conjuntos de variables:



# Otras cosas útiles: paquetes que extienden ggplot2



## Para conocer más.

- <http://ggplot2.tidyverse.org/>
- Libro para salir del paso: "[R Graphics Cookbook](#)"
- Libro para conocer más sobre la gramática de gráficos: "[ggplot2: Elegant Graphics for Data Analysis](#)"
- <http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html>