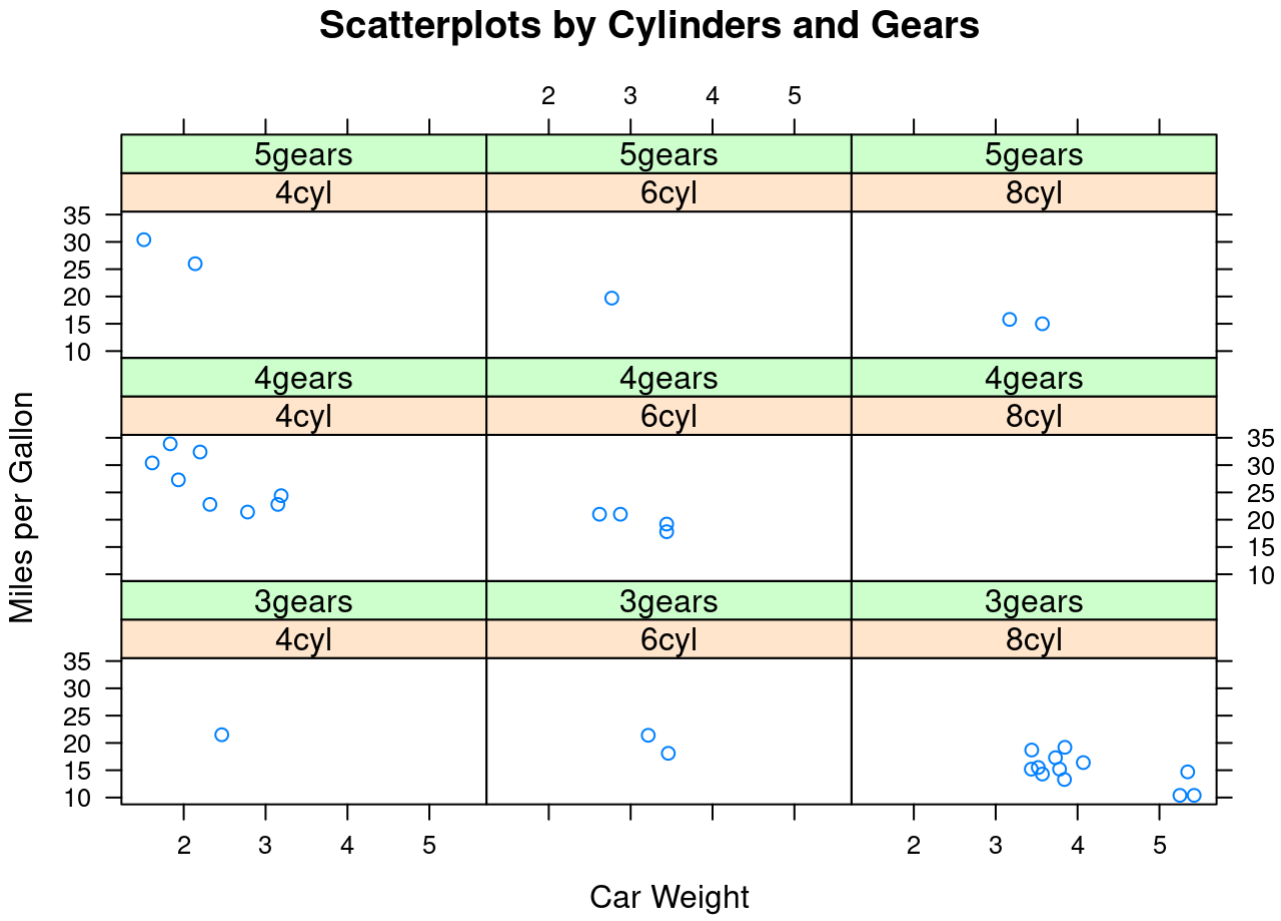
Tomado de: https://www.datanalytics.com/libro_r/0800_ggplot.html

Gráficos con ggplot2

R es un lenguaje para el análisis estadístico de datos. Como tal, destaca por sus gráficos. En la mayor parte de los lenguajes de programación, la capacidad de crear gráficos la proporcionan liberías adicionales ajenas a su núcleo. Sin embargo, en R, los gráficos son nativos.

Existen dos motores gráficos en R. Un motor gráfico es un conjunto de funciones que permiten realizar manipulaciones gráficas básicas: generar lienzos (o canvas), trazar líneas, dibujar puntos, etc. Un usuario de R no manipula generalmente esas funciones directamente: utiliza funciones de alto nivel, como plot. La función plot es la encargada de invocar esas funciones de bajo nivel que pintan los segmentos, círculos, etc. que conforman un gráfico estadístico, con sus ejes, sus etiquetas, etc.

Funciones de R tales como plot, hist, barplot, boxplot y otras se apoyan en el motor tradicional de R, que es suficiente para esos fines. Sin embargo, se queda corto para construir otro tipo de gráficos más avanzados. Por eso, en 2001, Paul Murrell desarrolló un motor gráfico alternativo, grid. Uno de sus objetivos era facilitar la generación en R de un tipo de gráficos conocidos como de Trellis, de celosía o de pequeños múltiplos.



Los gráficos de Trellis permiten seguir el comportamiento de unas variables de interés a través de los distintos niveles de otras, disponiendo la información en una retícula que facilita el descubrimiento de patrones por inspección visual. El gráfico anterior muestra la relación entre el peso y el consumo de gasolina de una serie de vehículos en función de su número de cilindros y el número de marchas. La relación es más evidente usando un

gráfico de Trellis que, por ejemplo, usando colores (o formas) para representar el número de cilindros o marchas en un único gráfico de dispersión.

Hay muchas funciones y paquetes que crean gráficos apoyándose en el motor gráfico tradicional. Otros, utilizan grid. Dos de los más conocidos son lattice (con el que está generado el gráfico anterior) y ggplot2. De hecho, lattice y ggplot2 se solapan funcionalmente y la mayoría de los usuarios de R se decantan por uno u otro y lo usan predominantemente.

ggplot2 es unos años posterior a lattice pero, a pesar de ello, más popular. ggplot2 es una implementación de las ideas recogidas en el artículo de The Language of Graphics escrito por Leland Wilkinson y sus coautores en 2000. Ese artículo recogía una serie de ideas novedosas sobre qué es la representación gráfica de información estadística y cómo debería hacerse. Los gráficos tradicionales de R tienen un importante elemento de adhoquismo: las funciones para representar diagramas de dispersión, de cajas, de barras, etc. utilizan datos con distintos formatos, tienen parámetros no siempre coincidentes en nombre, etc. Lo revolucionario de planteamiento del artículo es la de poner de manifiesto que todos esos tipos de gráficos (y otros) pueden generarse mediante un lenguaje más o menos regular, con su sintaxis, su semiótica, etc. De la misma manera que el lenguaje natural organiza sonidos mediante ciertas reglas comunes, conocidas y regulares para generar mensajes con significado, es posible construir una serie de reglas comunes, conocidas y regulares para crear representaciones visuales de datos de interés estadístico.

De ese lenguaje, implementado en el paquete ggplot2, se ocupan las siguientes secciones.

8.1 Una primera toma de contacto

En primer lugar vamos a cargar los paquetes ggplot2 y reshape2:

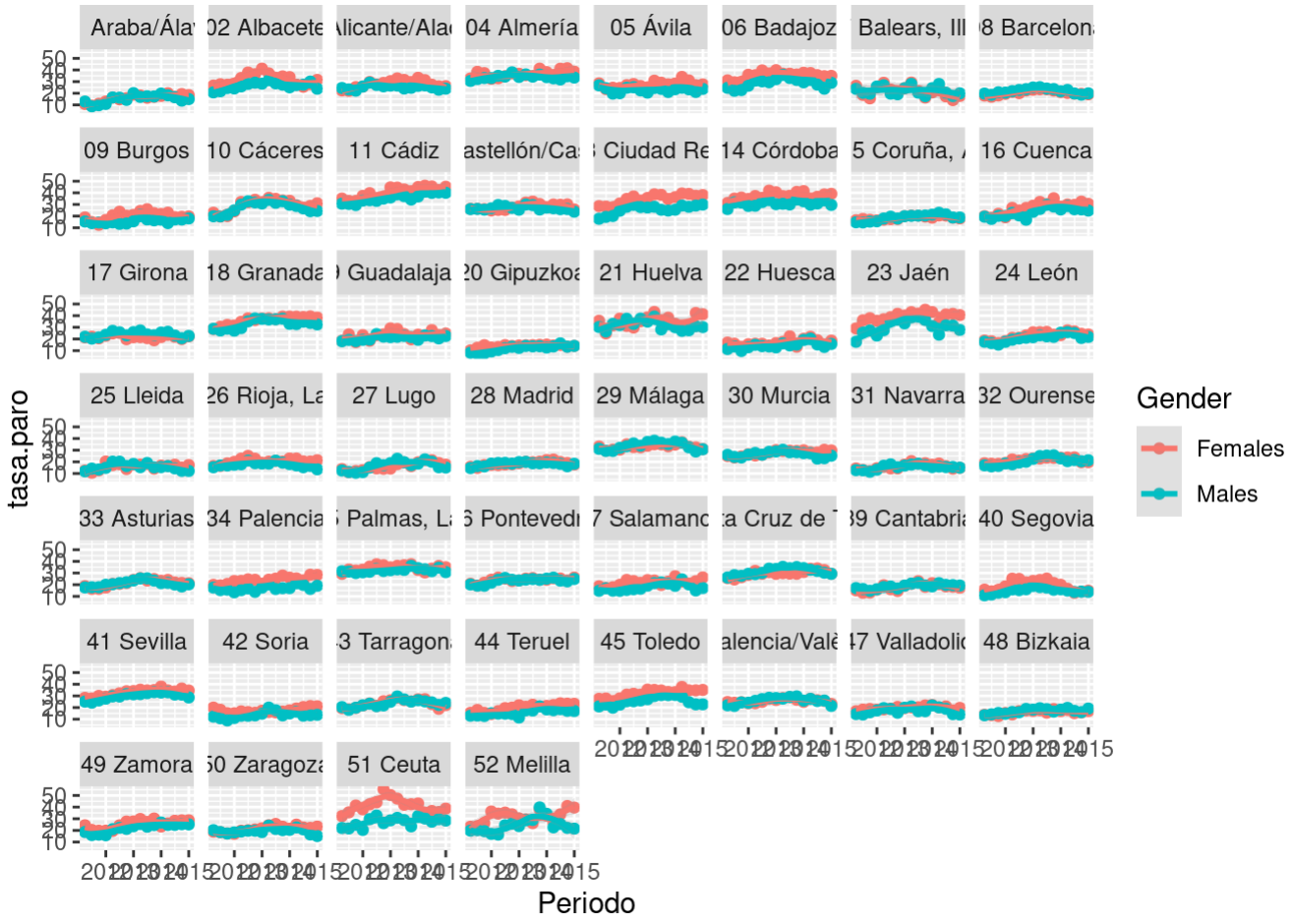
```
library(ggplot2)
library(reshape2)
```

Para el primer gráfico con ggplot2 vamos a leer y preprocesar de nuevo los datos del paro del INE:

```
paro <- read.table("data/paro.csv", header = T, sep = "\t")
paro <- dcast(paro, Gender + Provinces + Periodo ~ Situation)
paro$tasa.paro <- 100 * paro$unemployed / paro$active

paro$Periodo <- as.character(paro$Periodo)
paro$Periodo <- gsub("Q1", "-03-31", paro$Periodo)
paro$Periodo <- gsub("Q2", "-06-30", paro$Periodo)
paro$Periodo <- gsub("Q3", "-09-30", paro$Periodo)
paro$Periodo <- gsub("Q4", "-12-31", paro$Periodo)
paro$Periodo <- as.Date(paro$Periodo)
```

La expresión fundamental para crear un gráfico con ggplot2 es:

```
ggplot(paro, aes(x = Periodo, y = tasa.paro, col = Gender)) +
  geom_point() + geom_smooth(alpha = 0.2) +
  facet_wrap(~ Provinces)
```



La expresión anterior combina varios elementos que discutiremos con detalle más adelante:

- Datos: siempre una tabla.
- Estéticas, que asocian a elementos representables gráficamente (la posición `x` e `y`, el color, etc.) columnas de la tabla de datos.
- Geometrías (o capas): puntos, rectas, histogramas, densidades, etc. También se llaman capas porque pueden superponerse.
- Facetas: parten un gráfico en sublienzos preservando las escalas (pequeños múltiplos)

8.2 Elementos de un gráfico en `ggplot2`

Un gráfico en `ggplot2` se construye combinando una serie de elementos básicos y comunes a muchos tipos de gráficos distintos mediante una sintaxis sencilla. Esta sección describe esa sintaxis y los elementos que articula.

8.2.1 Datos

Uno de los elementos más importantes de un gráfico son los datos que se quieren representar. Una particularidad de `ggplot2` es que solo acepta un tipo de datos: `data.frames`. Otras funciones gráficas (p.e., `hist`) admiten vectores, listas u otro tipo de estructuras. `ggplot2` no.

```
p <- ggplot(iris)
```

El código anterior crea un objeto, `p` que viene a ser un protográfico: contiene los datos que vamos a utilizar, los del conjunto de datos `iris`. Obviamente, el código anterior es insuficiente para crear un gráfico: aún no hemos indicado qué queremos hacer con `iris`.

8.2.2 Estéticas

En un conjunto de datos hay columnas: edad, altura, ingresos, temperatura, etc. En un gráfico hay, en la terminología de ggplot2, *estéticas*. Estéticas son, por ejemplo, la distancia horizontal o vertical, el color, la forma (de un punto), el tamaño (de un punto o el grosor de una línea), etc. Igual que al hablar asociamos a un conjunto de sonidos (p.e., m-e-s-a) un significado (el objeto que conocemos como mesa), al realizar un gráfico asociamos a elementos sin significado propio (p.e., los colores) uno: el que corresponde a una columna determinada de los datos.

En ggplot2, dentro del lenguaje de los gráficos que implementa, es muy importante esa asociación explícita de significados a significantes, es decir, de columnas de datos a *estéticas*.

En el código

```
p <- p + aes(x = Petal.Length, y = Petal.Width, colour = Species)
```

se están añadiendo a p información sobre las estéticas que tiene que utilizar y qué variables de iris tiene que utilizar:

- La distancia horizontal, x, vendrá dada por la longitud del pétalo.
- La distancia vertical, y, por su anchura.
- El color, por la especie.

Hay que hacer notar la sintaxis del código anterior, bastante particular y propia del paquete ggplot2.
Al *protográfico* se le han sumado las estéticas. En las secciones siguientes se le *sumarán* otros elementos adicionales. Lo importante es recordar cómo la suma es el signo que combina los elementos que componen el lenguaje de los gráficos.

De todos modos, es habitual combinar ambos pasos en una única expresión

```
p <- ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species))
```

El objeto p resultante aún no es un gráfico ni se puede representar. Le faltan capas, que es el objeto de la siguiente sección. No obstante, se puede inspeccionar así:

```
summary(p)
```

```
data: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, Species
 [150x5]
mapping:  x = ~Petal.Length, y = ~Petal.Width, colour = ~Species
faceting: <ggproto object: Class FacetNull, Facet, gg>
    compute_layout: function
    draw_back: function
    draw_front: function
    draw_labels: function
    draw_panels: function
    finish_data: function
    init_scales: function
    map_data: function
    params: list
    setup_data: function
    setup_params: function
```

shrink: TRUE

train_scales: function

vars: function

super: <ggproto object: Class FacetNull, Facet, gg>

Ahí están indicados los datos que va a utilizar y la relación (o *mapeo*) entre estéticas y columnas de los datos.

¿Cuántas estéticas existen? Alrededor de una docena, aunque se utilizan, generalmente, menos:

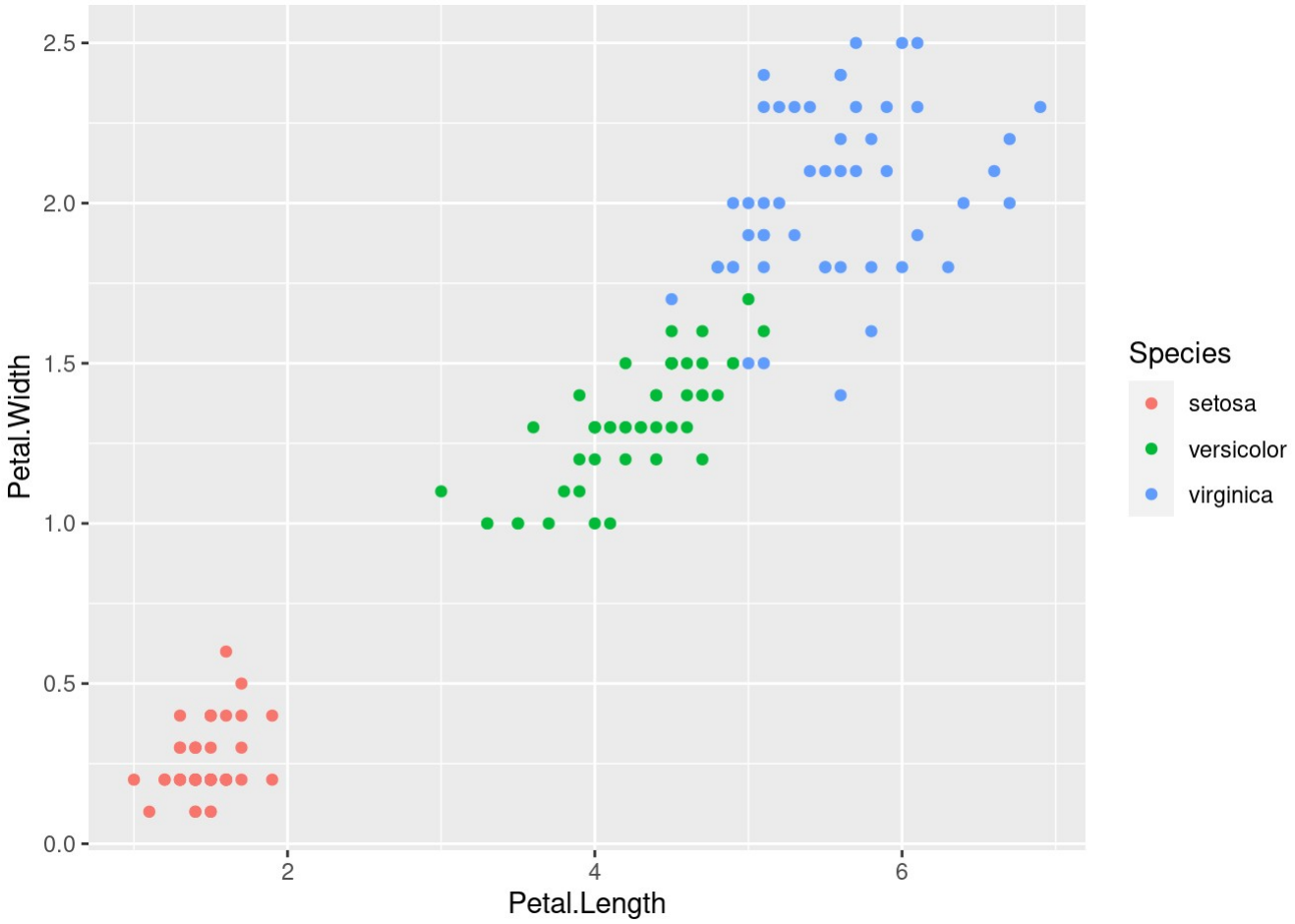
- x e y, distancias horizontal y vertical.
- colour, para el color.
- size, para el tamaño.
- shape, que indica la forma de los puntos (cuadrados, triángulos, etc.) de los puntos o del trazo (continuo, punteado) de las líneas.
- alpha para la transparencia: los valores más altos tendrían formas opacas y los más bajos, casi transparentes. De ahí la utilizad del canal alfa: da peso e importancia a las observaciones que la merecen.
- fill, para el color de relleno de las formas sólidas (barras, etc.).

Hay que advertir que no todas las *estéticas* tienen la misma potencia en un gráfico. El ojo humano percibe fácilmente longitudes distintas. Pero tiene problemas para comparar áreas (que es lo que regula la estética size) o intensidades de color. Se recomienda usar las estéticas más potentes para representar las variables más importantes.

8.2.3 Capas

Las capas (o geoms para ggplot2) son los verbos del lenguaje de los gráficos. Indican qué hacer con los datos y las estéticas elegidas, cómo representarlos en un lienzo. Y, en efecto, el siguiente código crea el correspondiente gráfico.

```
p <- p + geom_point()
p
```

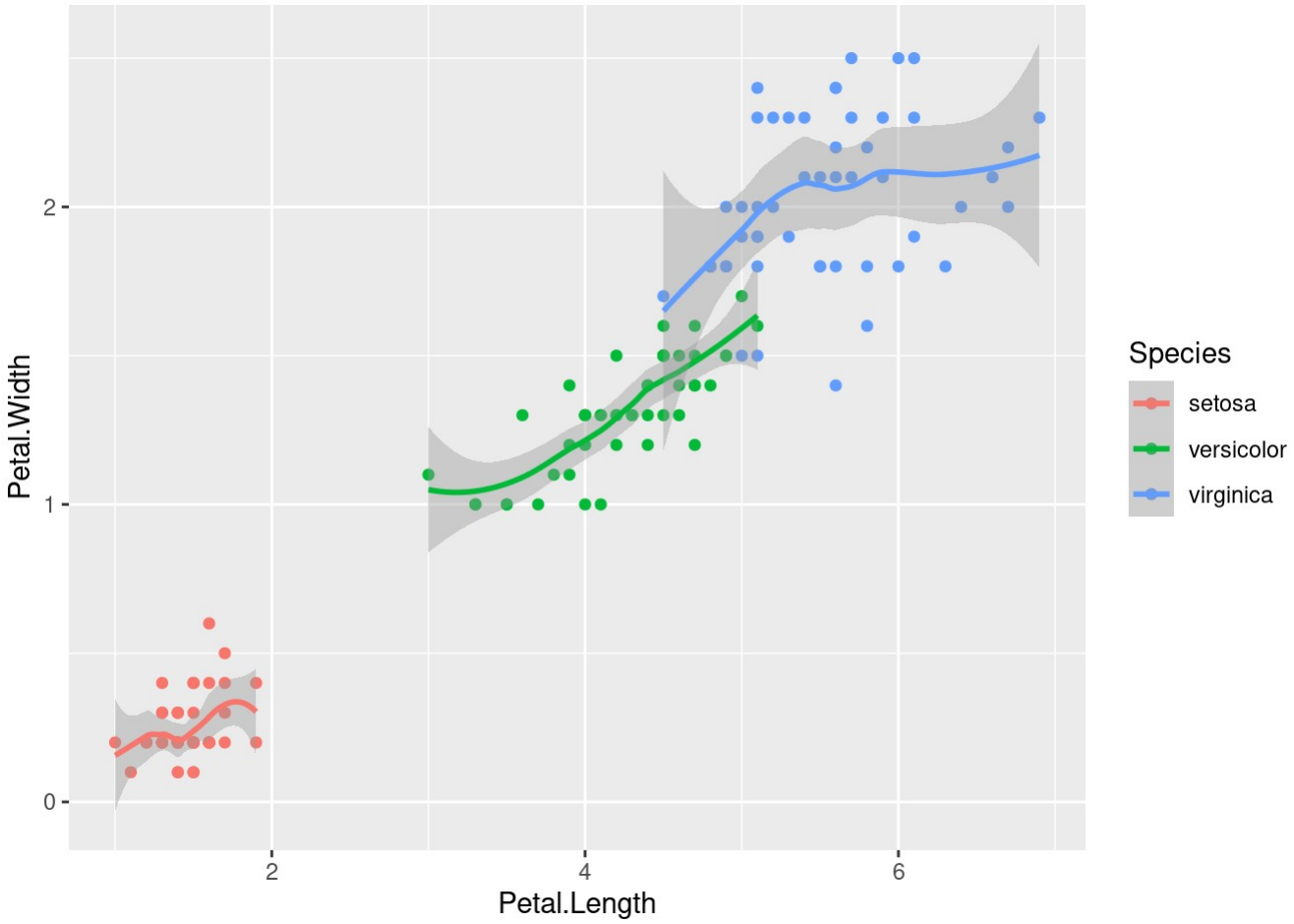


Una vez añadida una capa al gráfico, este puede pintarse (que es lo que ocurre al llamar a `p`). Se obtiene el mismo resultado haciendo, en una única línea,

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) + geom_point()
```

Una característica de las capas, y de ahí su nombre, es que pueden superponerse. Por ejemplo,

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point() + geom_smooth()
```



añade al gráfico una curva suavizada (con sus intervalos de confianza en gris).

Existen muchos tipos de capas. Los más usuales son geom_point, geom_line, geom_histogram, geom_bar y geom_boxplot. Pero hay más. En la página http://docs.ggplot2.org/current/ se muestra una lista de los disponibles (en la versión más actualizada de ggplot2). En esa página se indica qué *geom* hay que utilizar en función de una representación esquemática del tipo de gráfico que se quiere construir. Además, hay capas específicas que exigen estéticas especiales. Para algunas tiene sentido, p.e., shape. Para otras no. Esas especificidades están indicadas en dicha página, que es más útil que la ayuda general de R.

Una vez creado un gráfico, es posible exportarlo a png, jpg, etc. La función ggsave guarda en un fichero el último gráfico generado con ggplot2. Lo hace, además, en el formato indicado en el nombre del fichero que se quiere generar. Así,

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) + geom_point()
ggsave("mi_grafico.png")
```

guarda la figura creada en la primera línea en formato png en el fichero mi_grafico.png del directorio de trabajo.

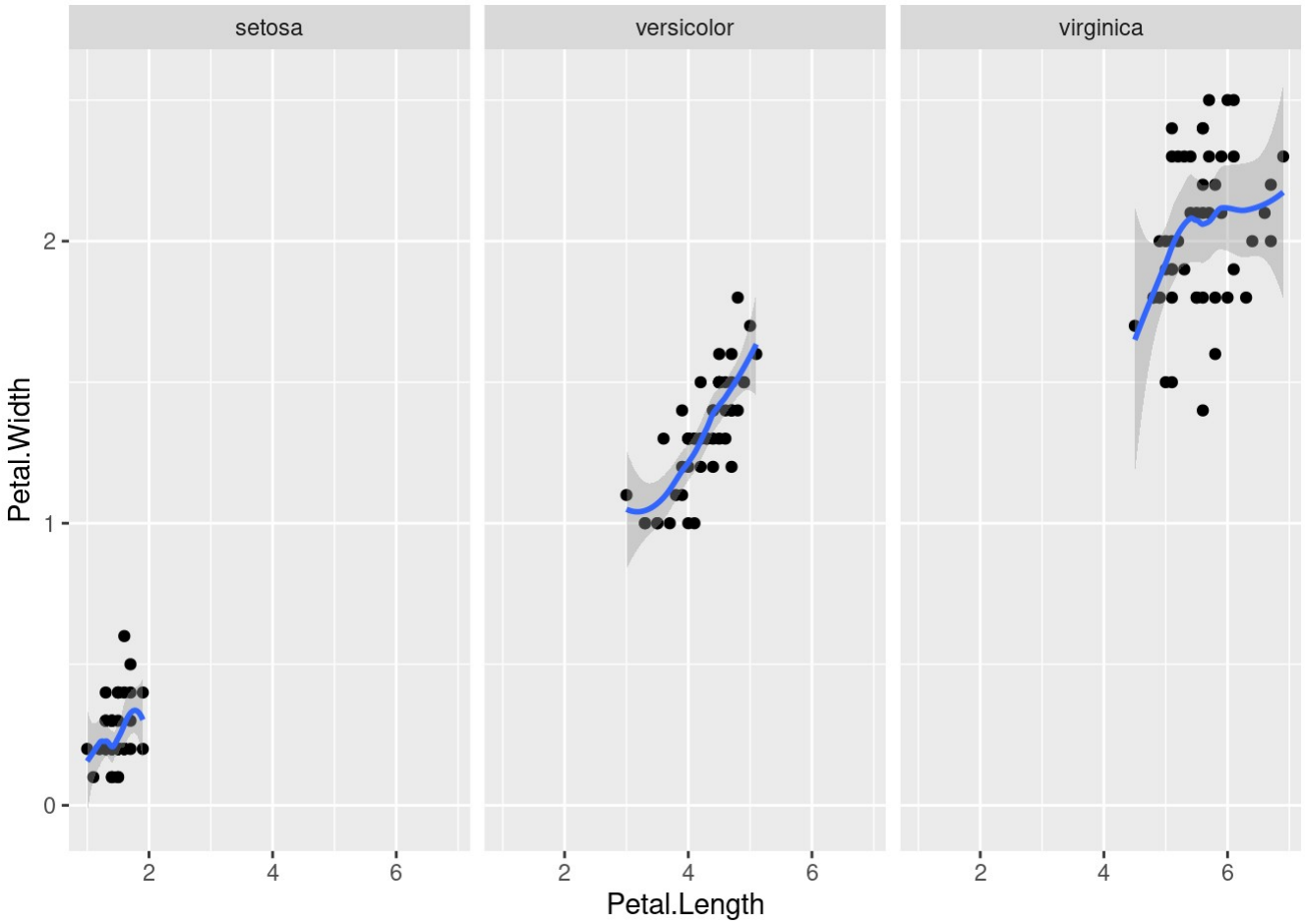
8.2.4 Facetas

Muchos de los gráficos que pueden generarse con los elementos anteriores pueden reproducirse sin mucho esfuerzo (exceptuando, tal vez, cuestiones de aspecto) usando los gráficos tradicionales de R, pero no los que usan facetas.

Las facetas implementan los gráficos de Trellis mencionados antes. Por ejemplo,

```
ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) +
```

```
  geom_point() + geom_smooth() +
  facet_grid(~ Species)
```

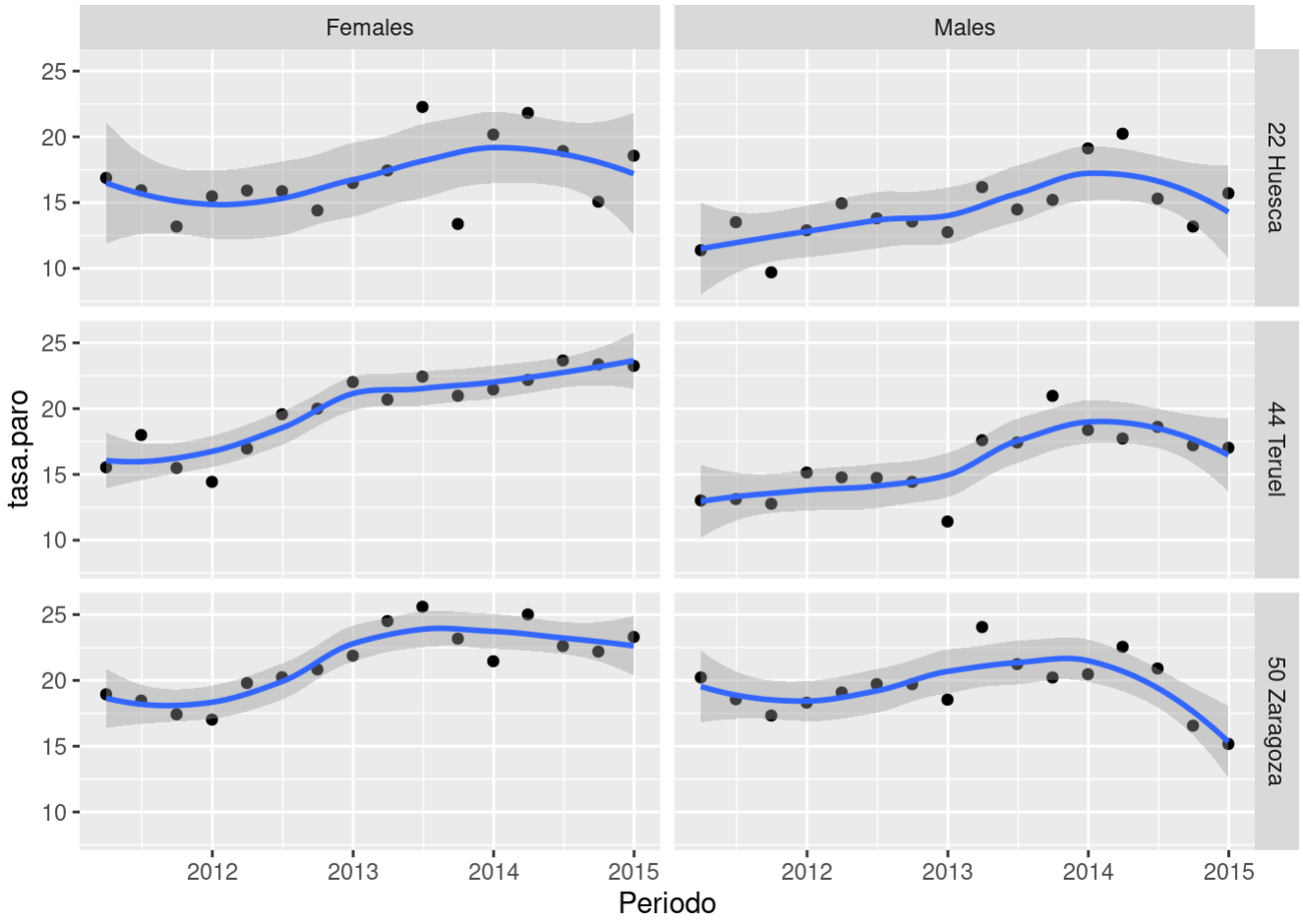


crea tres gráficos dispuestos horizontalmente que comparan la relación entre la anchura y la longitud del pétalo de las tres especies de iris. Una característica de estos gráficos, que es crítica para poder hacer comparaciones adecuadas, es que comparten ejes.

Los gráficos podrían disponerse verticalmente reemplazando `facet_grid(~ Species)` por `facet_grid(Species ~ .)` en el código anterior. Además, se puede subdividir el lienzo por dos (¡o más!) variables así:

```
tmp <- paro[paro$Provinces %in% c("50 Zaragoza", "22 Huesca", "44 Teruel"),]
ggplot(tmp, aes(x = Periodo, y = tasa.paro)) +
  geom_point() + geom_smooth() +
  facet_grid(Provinces~Gender)
```

`geom_smooth()` using method = 'loess' and formula = 'y ~ x'



En caso de haber muchas categorías (p.e., provincia), puede usarse la función facet_wrap para distribuir las subgráficas en una cuadrícula.

8.2.5 Más sobre estéticas

Las estéticas se pueden etiquetar con la función labs. Además, se le puede añadir un título al gráfico usando la función ggtitle. Por ejemplo, en el gráfico anterior se pueden reetiquetar los ejes y la leyenda haciendo

```
p <- p + ggtitle("Petal length and width") +
    labs(x = "Petal length",
         y = "Petal width",
         colour = "Species")
```

8.2.6 Temas

Los *temas* de ggplot2 permiten modificar aspectos estéticos del gráfico que no tienen que ver con los datos en sí. Eso incluye los ejes, etiquetas, colores de fondo, el tamaño de los márgenes, etc. No es habitual (y se desaconseja a los usuarios menos expertos) tener que alterar los temas que ggplot2 usa por defecto. Solo se vuelve necesario cuando los gráficos tienen que adecuarse a una imagen corporativa o atenerse a algún criterio de publicación exigente.

Un tema es una colección de elementos (p.e., panel.background, que indica el color, transparencia, etc. del lienzo sobre el que se representa el gráfico) modificables. El tema que usa ggplot2 por defecto es theme_grey. Al escribir theme_grey() en la consola de R, se muestran alrededor de cuarenta elementos modificables y sus atributos tal y como los define dicho tema.

¿Qué se puede hacer con los temas? Una primera opción es elegir otro. Por ejemplo, se puede reemplazar el habitual por otros disponibles en el paquete como `theme_bw` (o `theme_classic`) haciendo

```
p <- p + theme_bw()
```

Es posible usar tanto los temas que incluye `ggplot2` por defecto como otros creados por la comunidad. Algunos, por ejemplo, tratan de imitar el estilo de publicaciones reconocidas como The Economist o similares. Algunos están recogidos en paquetes como, por ejemplo, `ggthemes`.

Alternativamente (o adicionalmente), es posible modificar un tema dado en un gráfico. Por ejemplo, haciendo

```
p <- p +
  theme_bw() +
  theme(
    panel.background = element_rect(fill = "lightblue"),
    panel.grid.minor = element_line(linetype = "dotted")
  )
```

se está modificando el atributo de color del lienzo de un gráfico y el tipo de la línea con que se dibuja la malla.

Finalmente, es posible construir temas propios y personalizados. Aunque no es un proceso complicado, los detalles quedan fuera del alcance de esta introducción.

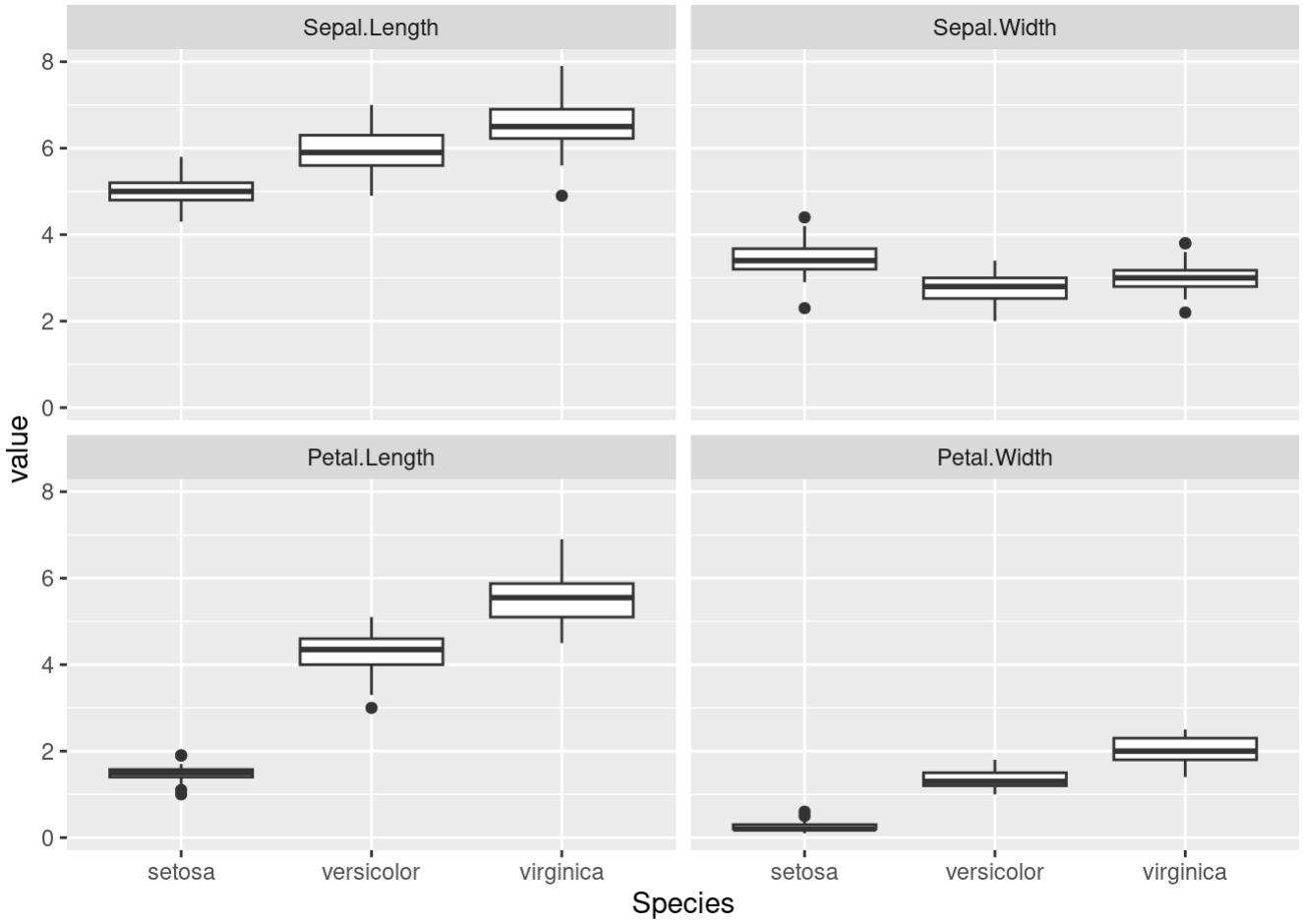
8.3 Ejemplos

En esta sección se van a explorar algunos de los gráficos estadísticos más básicos.

8.3.1 Diagramas de cajas (y de violín)

Los diagramas de caja (*boxplots*) describen de manera cruda la distribución de una variable continua en función de una discreta. En el ejemplo que aparece a continuación, se explora el conjunto de datos `iris` que contiene 50 observaciones de características métricas de cada una de las tres subespecies de iris, una flor. Es un conjunto de datos de larga tradición en estadística y se recopiló para ilustrar algoritmos de clasificación, es decir, cómo crear criterios para distinguir los tres tipos de iris.

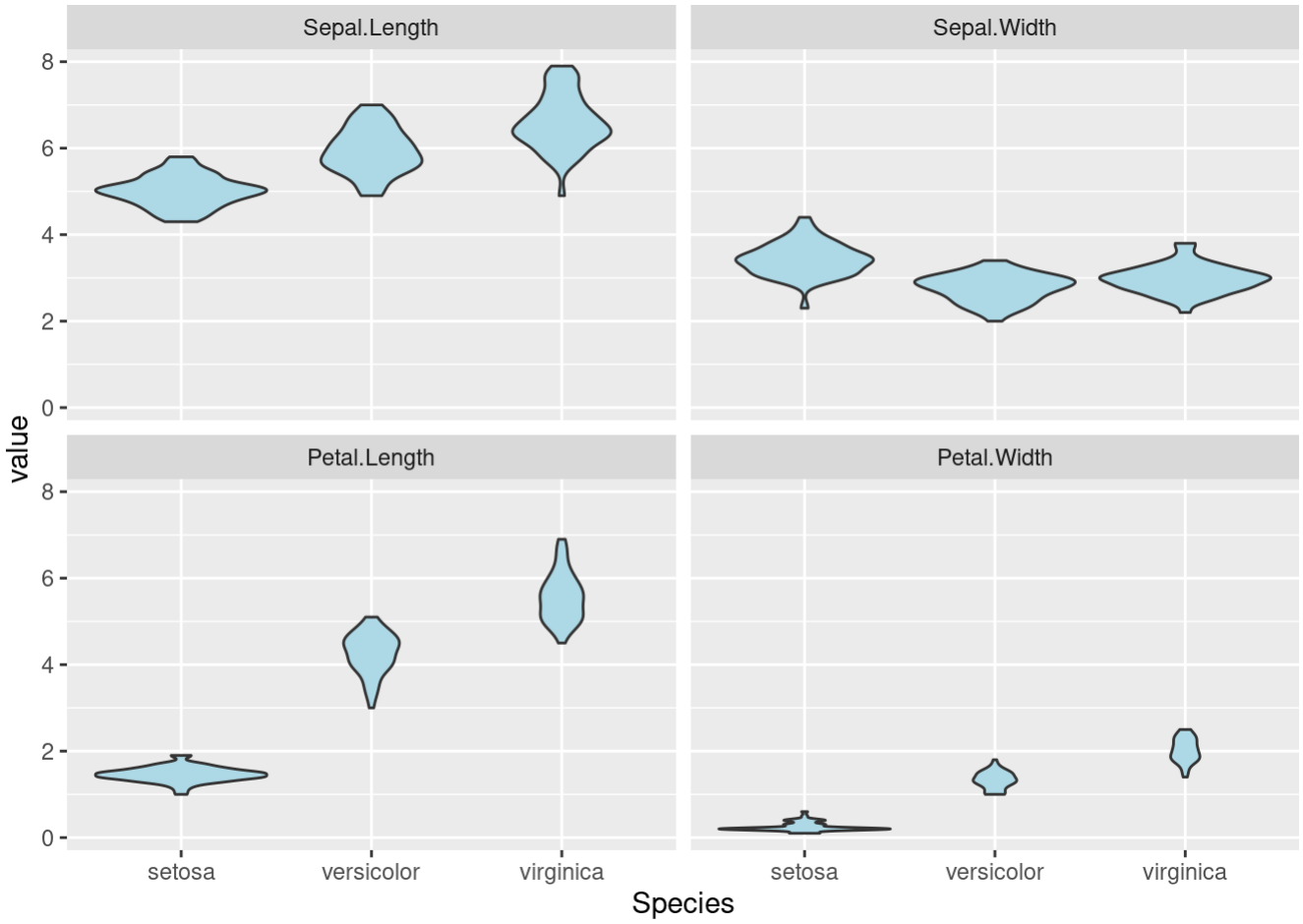
```
library(reshape2)
tmp <- melt(iris)
ggplot(tmp, aes(x = Species, y = value)) + geom_boxplot() + facet_wrap(~ variable)
```



El gráfico utiliza las facetas para crear cuatro paneles, uno por variable. Así resume rápidamente la información contenida en el conjunto de datos y revela eficazmente los patrones que encierra: por ejemplo, cómo la especie setosa tiene el pétalo sensiblemente más estrecho y corto que las otras dos. Este tipo de gráficos son fundamentales como herramienta exploratoria previa a la aplicación de técnicas de análisis estadístico (discriminante, en este caso).

Los gráficos de cajas son muy básicos: apenas muestran cinco puntos característicos de una distribución: la mediana, los extremos y los cuartiles. Son herencia de una época en que apenas había recursos, principalmente informáticos, para realizar representaciones más sofisticadas. Una versión moderna de los gráficos de cajas es la de gráficos de violín. Como los de cajas, resumen la distribución de las variables. Pero en lugar de una representación sucinta, tratan de dibujar la distribución real de los datos: son verdaderos gráficos de densidad, solo que dispuestos de otra manera para facilitar la comparación.

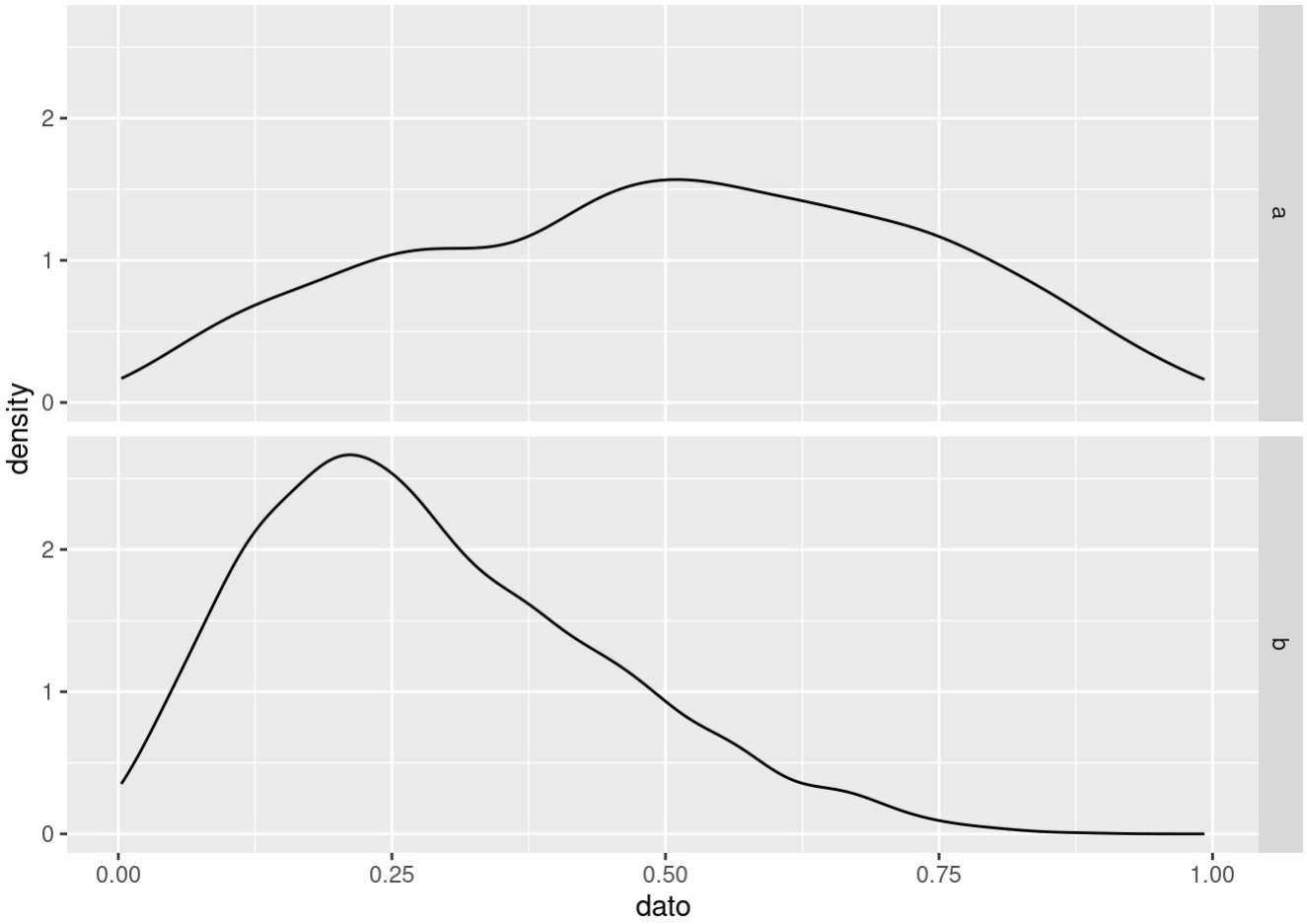
```
ggplot(tmp, aes(x = Species, y = value)) +
  geom_violin(fill = "lightblue") +
  facet_wrap(~ variable)
```



8.3.2 Comparación de dos densidades

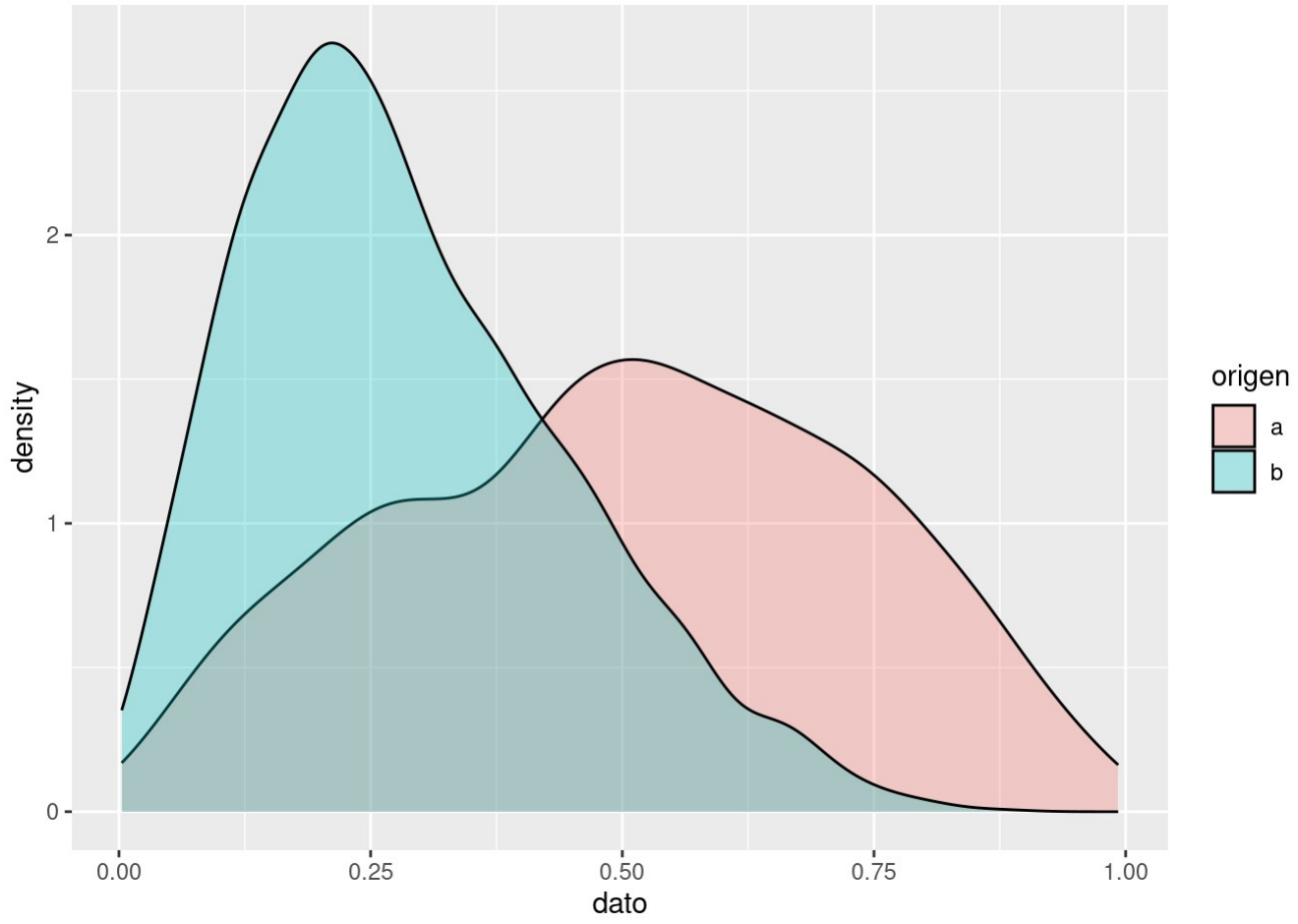
El análisis de datos exige en ocasiones comparar dos distribuciones continuas. Se pueden usar gráficos de cajas o de violín, como arriba, pero también se puede dibujar la distribuición completa como en el siguiente gráfico:

```
# datos (simulados)
a <- rbeta(1000, 2, 2)
b <- rbeta(2000, 2, 5)

# construcción de un dataframe a partir de ellos
tmp <- rbind(data.frame(origen = "a", dato = a),
      data.frame(origen = "b", dato = b))

ggplot(tmp, aes(x = dato)) + geom_density() + facet_grid(origen ~ .)
```



Alternativamente, se pueden solapar ambas distribuciones. El uso del parámetro `alpha`, que controla la transparencia, es fundamental en este caso:

```
ggplot(tmp, aes(x = dato, fill = origen)) + geom_density(alpha = 0.3)
```



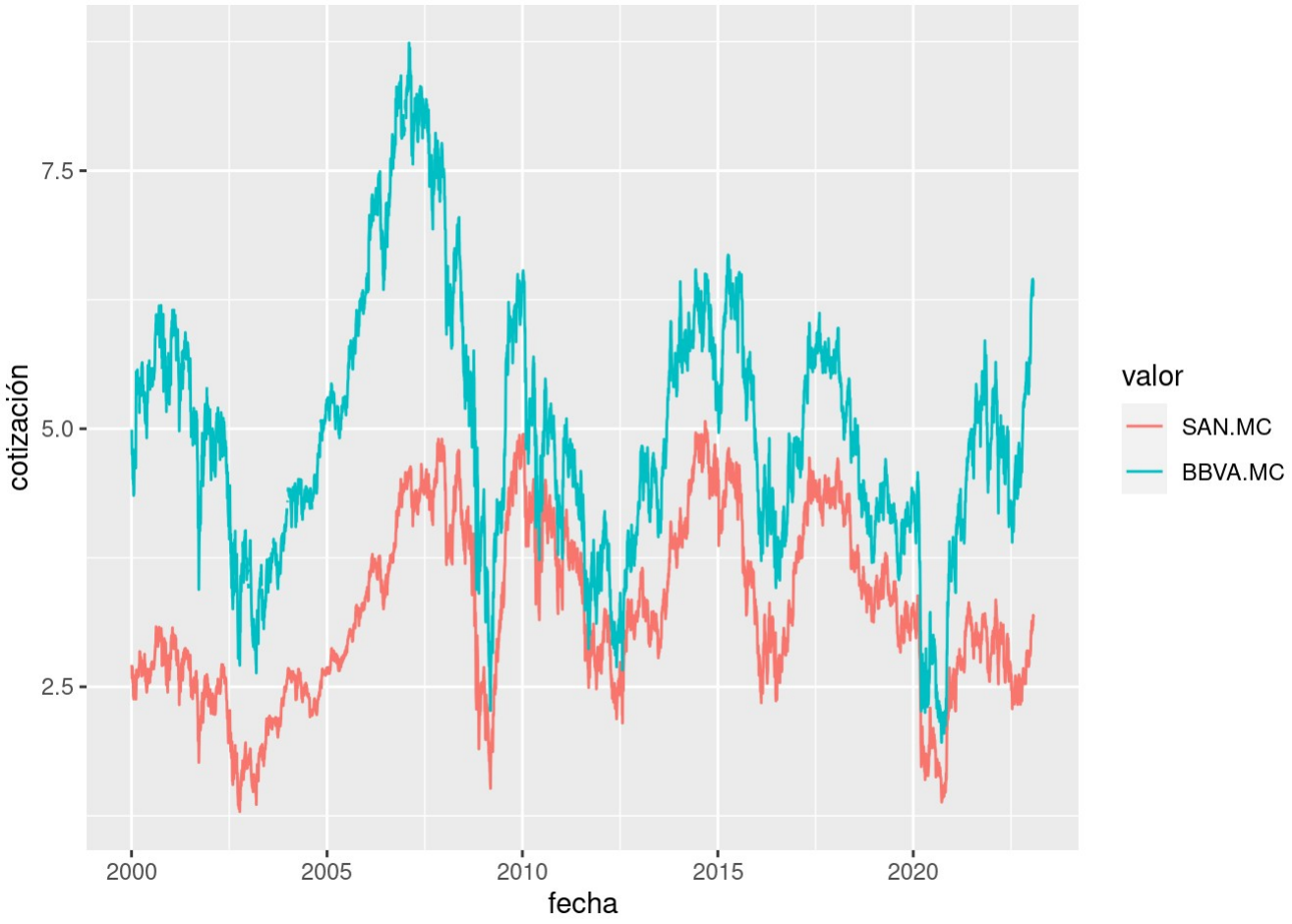
8.3.3 Series temporales

ggplot2 entiende ciertos tipos de datos particulares, como por ejemplo, series temporales. En el siguiente ejemplo se descargan las cotizaciones bursátiles de dos banco españoles directamente de Yahoo! Finance y se representan gráficamente:

```
library(tseries)
library(zoo)
library(reshape2)
# función para descargar las cotizaciones
cotizaciones <- function(valor){
 res <- get.hist.quote(valor, provider = "yahoo",
            quote = "AdjClose", quiet = TRUE)
 colnames(res) <- valor
 res
}
# combinación de ambas series temporales
res <- merge(cotizaciones("SAN.MC"),
      cotizaciones("BBVA.MC"))
# construcción de un dataframe con un índice de tipo fecha
res <- data.frame(fecha = index(res), as.data.frame(res))
res <- melt(res, id.var = "fecha")
ggplot(res, aes(x = fecha, y = value)) + geom_line() + facet_grid(variable ~ .)
```

Una versión alternativa del gráfico es la que superpone las series usando colores para distinguirlas:

```
ggplot(res, aes(x = fecha, y = value, colour = variable)) + geom_line() +
  labs(colour = "valor", y = "cotización")
```



8.4 Resumen y referencias

8.5 Ejercicios adicionales

Ejercicio 8.1 Construye un diagrama de cajas de las temperaturas en NY por mes (sin facetas). Nota: ten en cuenta que `ggplot` se confunde cuando la estética `x` del histograma no es categórica; por lo tanto, tendrás que convertir la variable mes en categórica (usando `factor`).

Ejercicio 8.2 Construye un histograma de las temperaturas en NY por mes (con facetas).

Ejercicio 8.3 Prueba con los gráficos de violín (que son una mezcla de los dos anteriores).

Ejercicio 8.4 Superpón las distribuciones de las temperaturas de NY por mes como en el ejemplo anterior de la superposición de dos densidades (o como aquí).

Ejercicio 8.5 Haz gráficos con tus propios datos.