VISUALIZACIÓN DE DATOS Y COMUNICACIÓN DE RESULTADOS. INFORMES DINÁMICOS

Instituto Tecnológico

ÍNDICE

TU RETO EN ESTA UNIDAD

Seguro que a lo largo de tu vida has tenido que repetir algún trabajo que hiciste hace mucho tiempo. Y cuando has ido a recuperar aquello que hiciste años atrás, has olvidado casi completamente como lo hiciste y tienes que empezar prácticamente de cero.

En esta unidad vas a aprender una herramienta para crear documentación que refleja no solo los resultados que obtuviste, sino que también guarda como llegaste a ellos y además te permite compartir el trabajo con otras personas. Este paradigma de llama "investigación reproducible".

¿Quieres que tu trabajo sea reproducible con el paso de tiempo, por ti u otras personas?

1. INTRODUCCIÓN

Hasta ahora hemos aprendido las herramientas para cargar los datos en R, transformarlos de manera adecuada, calcular métricas estadísticas y a representarlos gráficamente para entender las relaciones que existen entre ellos.

Sin embargo, no importa lo buenos que sean tus análisis si no eres capaz de comunicar tus resultados a los demás de manera eficaz.

La comunicación es el tema que vamos a desarrollar en esta unidad, que vamos a dividir en dos secciones principales:

- **Gráficos para comunicación:** donde aprenderemos como transformar los gráficos que hemos realizado en nuestros análisis exploratorios en gráficos preparados para exponer o publicar. El objetivo de estos gráficos es que nuestra audiencia los comprenda con facilidad.
- **Documentos dinámicos con R markdown:** donde aprenderemos R markdown, que es una herramienta muy potente que permite integrar texto con código y resultados. Estos documentos pueden generarse en diferentes formatos en función de las necesidades y del público objetivo: podemos generar notebooks para comunicación con nuestros analistas colaboradores, o informes para un público más general en html, pdf, o documentos de Word.

2. GRÁFICOS PARA COMUNICACIÓN

Hasta ahora hemos aprendido a hacer gráficos orientados al análisis exploratorio de datos. Cuando se hacen gráficos exploratorios, conocemos perfectamente que variables estamos utilizando y realizamos a veces cientos de gráficos, la mayoría de los cuales los desechamos.

Pero cuando hemos llegado a entender los datos, hemos extraído conclusiones de ellos y queremos comunicar los resultados al resto del mundo, nuestro público objetivo no tendrá el mismo conocimiento de los datos que tenemos nosotros. Deberemos esforzarnos en crear unos gráficos que sean fáciles de comprender por nuestra audiencia.

En este capítulo, aprenderás algunas de las herramientas que ggplot2 proporciona para crear gráficos eficaces, bien diseñados para la difusión y la publicación.

Vamos a trabajar con los datos del paquete gapminder, que nos proporcionan información de población, riqueza y esperanza de vida de un buen número de países del mundo desde los años 50 del siglo XX. Si no lo tienes instalado.

```
install.packages("gapminder")
```

2.1. ETIQUETAS

El sitio más fácil para comenzar a convertir un gráfico exploratorio en un gráfico publicable es mediante un buen etiquetado.

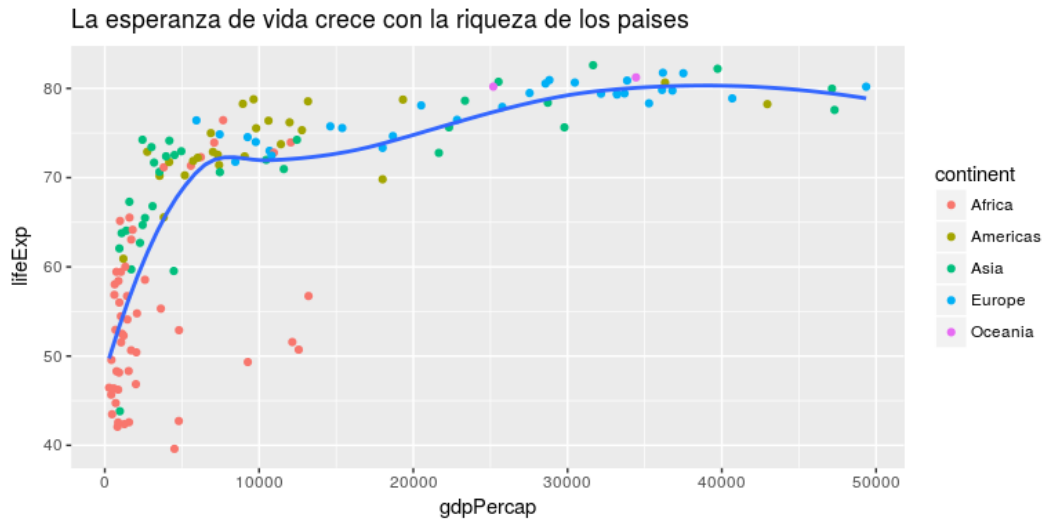
Añade etiquetas con la función labs(). Podemos añadir:

- Títulos.
- Subtítulos.
- Captions (explicaciones a pie de imagen).

El siguiente ejemplo muestra para el año 2007 la relación entre la esperanza de vida y el producto interior bruto per cápita. Le añadimos un título.

```
library(gapminder)
library(ggplot2)
library(dplyr)

gapminder %>% filter(year==2007) %>%
ggplot(aes(gdpPercap,lifeExp)) + geom_point(aes(color=continent))
+
  geom_smooth(se=FALSE) +
  labs(title="La esperanza de vida crece con la riqueza de los
paises")
```

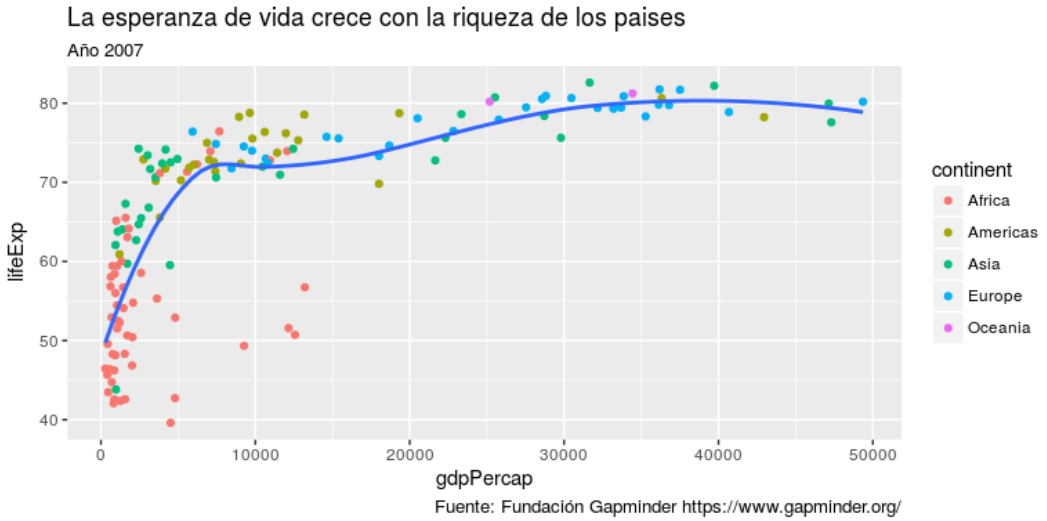


El propósito del título de un gráfico es resumir el hallazgo principal. Debemos evitar, en la medida de lo posible, títulos que simplemente describan lo que representa el gráfico, como por ejemplo: "Gráfico de dispersión de la esperanza de vida frente a la renta per cápita".

Si necesitas agregar más texto, hay dos etiquetas útiles que puede utilizar en ggplot2 (a partir de la versión 2.2.0):

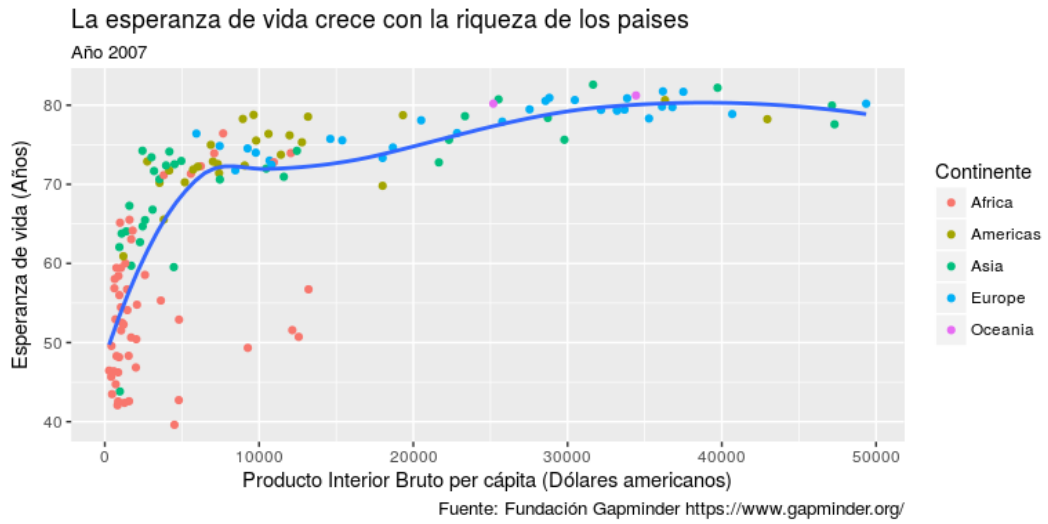
- `subtitle`: el subtítulo agrega detalles adicionales usando una fuente más pequeña debajo del título.
- `caption`: añade texto en la parte inferior de la gráfica. Se usa a menudo para especificar la fuente de los datos.

```
gapminder %>% filter(year==2007) %>%
ggplot(aes(gdpPercap,lifeExp)) + geom_point(aes(color=continent))
+
  geom_smooth(se=FALSE) +
  labs(title="La esperanza de vida crece con la riqueza de los
paises",
       subtitle="Año 2007",
       caption= "Fuente: Fundación Gapminder
https://www.gapminder.org/")
```



También podemos utilizar labs() para cambiar las etiquetas de los ejes y los títulos de las leyendas. Es una buena costumbre reemplazar los nombres de variables cortas con descripciones más detalladas, e incluir las unidades.

```
gapminder %>% filter(year==2007) %>%
  ggplot(aes(gdpPercap,lifeExp)) +
geom_point(aes(color=continent)) +
    geom_smooth(se=FALSE) +
    labs(title="La esperanza de vida crece con la riqueza de los
paises",
       subtitle="Año 2007",
       caption= "Fuente: Fundación Gapminder
https://www.gapminder.org/",
       x = "Producto Interior Bruto per cápita (Dólares america-
nos)",
       y = "Esperanza de vida (Años)",
       color = "Continente")
```

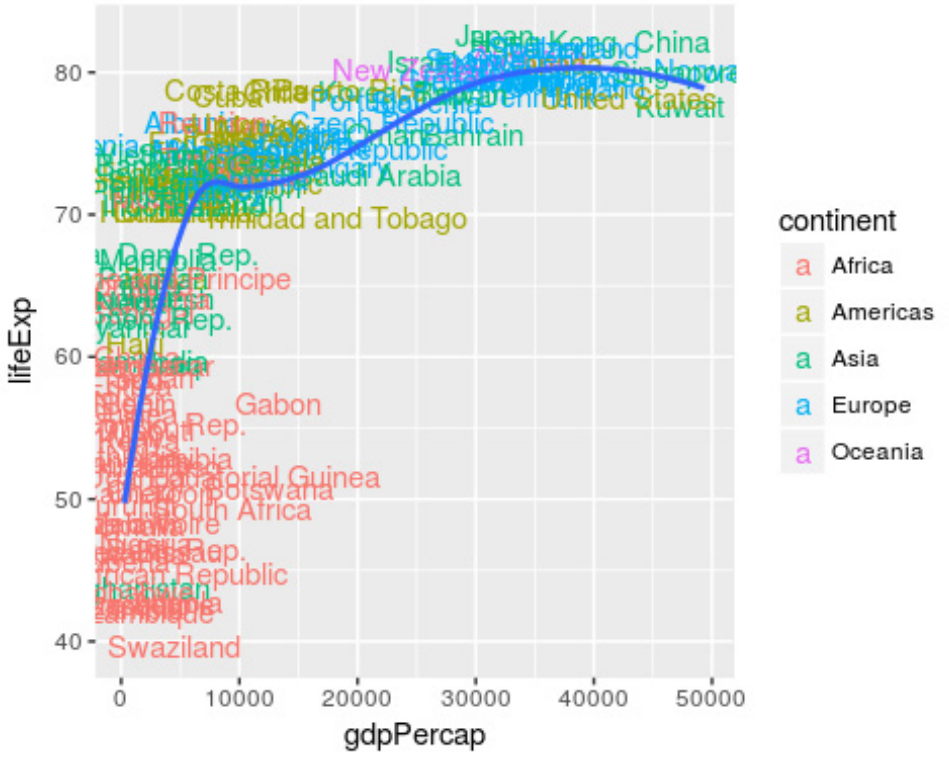


2.2. ANOTACIONES

Además de etiquetar los componentes principales del gráfico, a menudo es útil etiquetar observaciones individuales o grupos de observaciones. La primera herramienta que vamos a ver es `geom_text()`. Es similar a `geom_point()`, pero tiene una estética adicional: `label`. Esto hace posible agregar etiquetas textuales a los gráficos.

Como ejemplo podríamos representar cada país por su nombre en lugar de por un punto.

```
gapminder %>% filter(year==2007) %>%
ggplot(aes(gdpPercap,lifeExp)) + ge-
om_text(aes(label=country,color=continent)) +
  geom_smooth(se=FALSE)
```



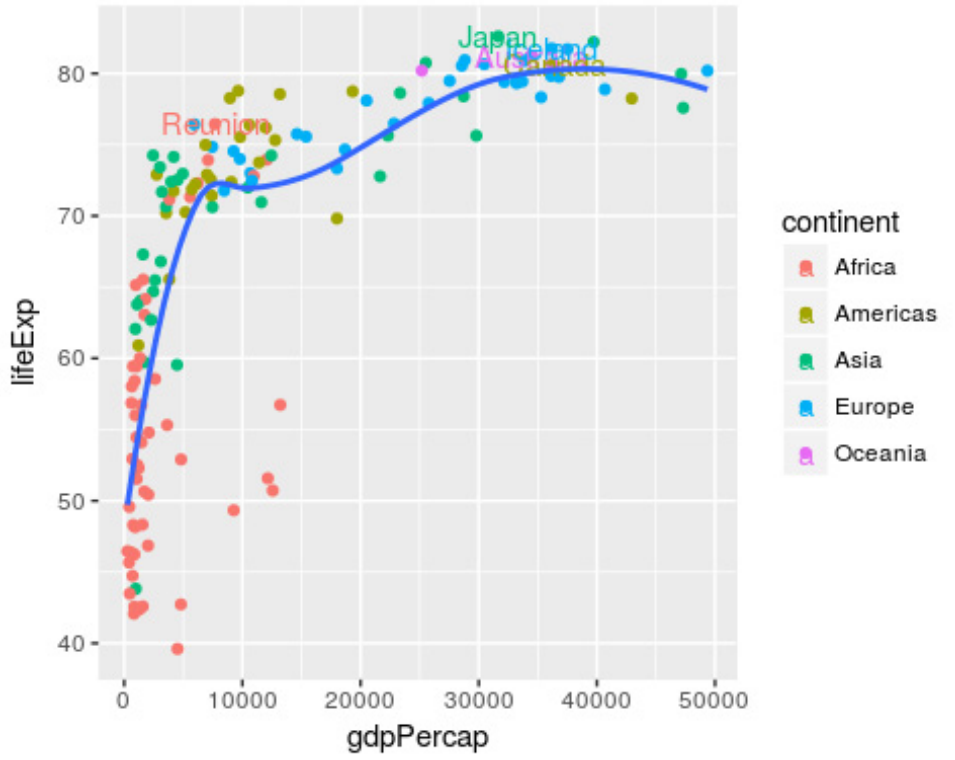
Aunque resulta un poco difícil de leer debido a la gran cantidad de países representados. Podríamos limitar esta cantidad, por ejemplo escribiendo solo el nombre del país con mayor esperanza de vida en cada continente. Para ello necesitamos un poco de dplyr.

```
gap2007 <- gapminder %>% filter(year==2007)

paises_longevos <- gap2007 %>%  group_by(continent) %>% fil-
ter(row_number(desc(lifeExp)) == 1)

ggplot(gap2007, aes(gdpPercap,lifeExp)) + ge-
```

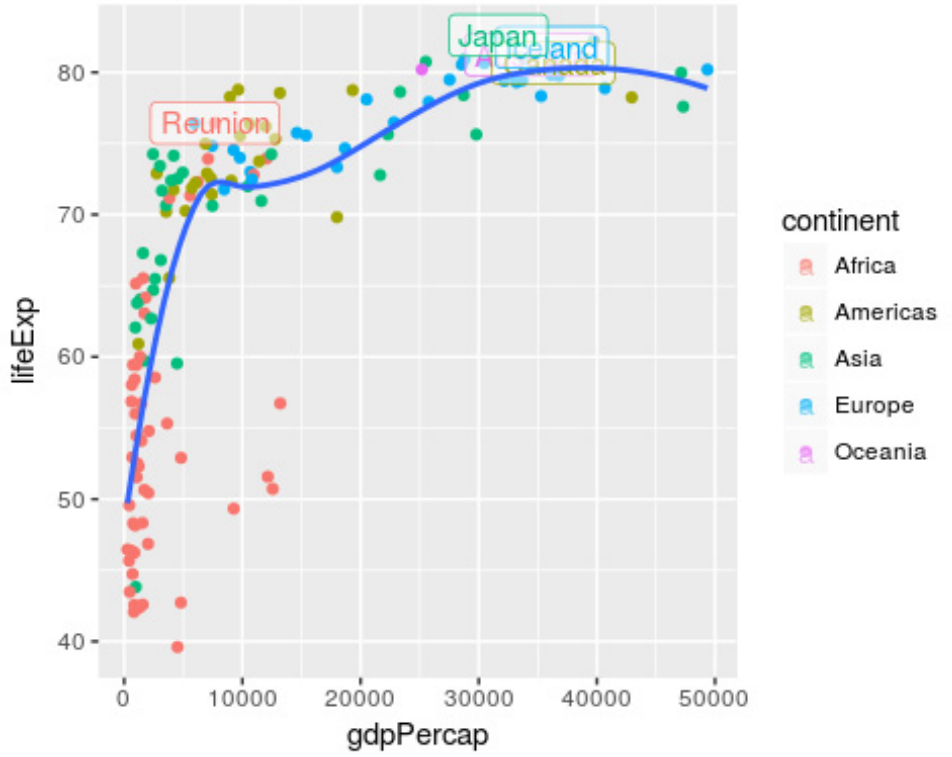
```
om_point(aes(color=continent)) +
  ge-
om_text(aes(label=country,color=continent),data=paises_longevos)
+
  geom_smooth(se=FALSE)
```



Aun escribiendo solo 5 etiquetas, no acaban de leerse bien los nombres de los países ya que se solapan unos con otros. Mediante `geom_label()` podemos añadir texto rodeado por un recuadro. Esto mejora algo la legibilidad.

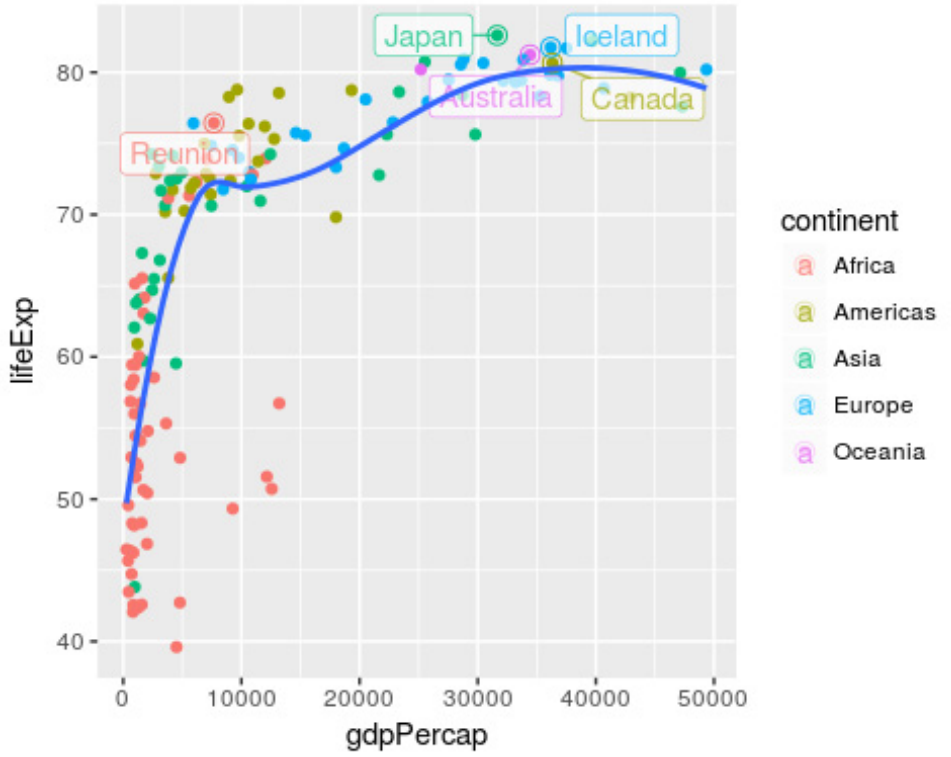
```
# Ponemos la transparencia de las etiquetas a 0.5 para ver debajo
de lo solapado
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
geom_point(aes(color=continent)) +

geom_label(aes(label=country,color=continent),data=paises_longevo
s, alpha=0.5) +
  geom_smooth(se=FALSE)
```



Pero seguimos sin evitar el problema de las etiquetas solapadas. Para mejorar este aspecto podemos usar el paquete ggrepel (si no lo tienes instalado ya sabes como hacerlo) que tiene una función que ajusta la posición de las etiquetas para que no se solapen.

```
ggplot(gap2007, aes(gdpPercap,lifeExp)) + ge-
om_point(aes(color=continent)) +
  geom_point(aes(color=continent),size=3, shape = 1, da-
ta=paises_longevos) +
  ggre-
pel::geom_label_repel(aes(label=country,color=continent),data=pai
ses_longevos,alpha=0.7) +
  geom_smooth(se=FALSE)
```



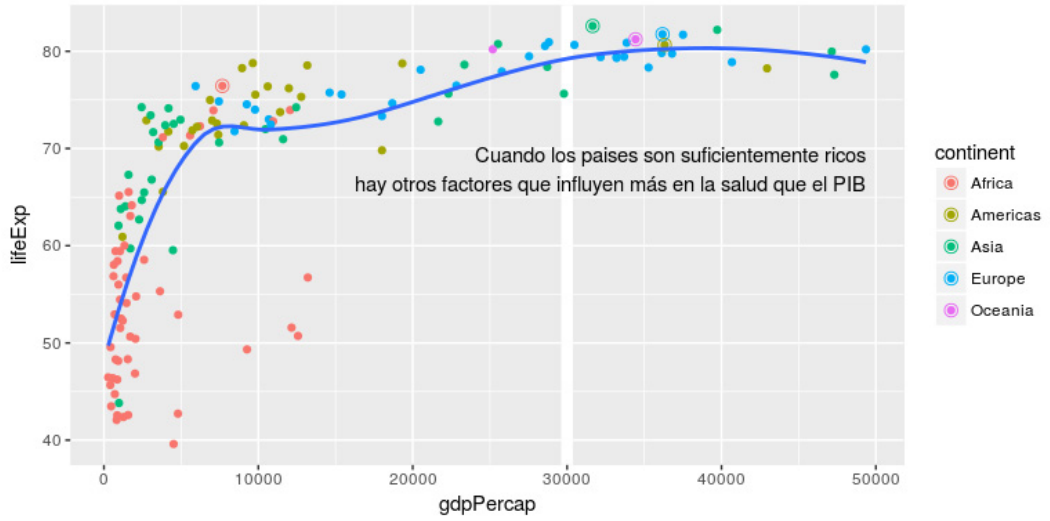
Date cuanta que además hemos añadido una capa nueva al gráfico, que muestra los puntos de los países seleccionados de manera resaltada.

A veces simplemente queremos añadir un texto explicativo en algún lugar del interior del gráfico, pero cuyo texto no está contenido en los datos. Para ello tenemos que crear una estructura de datos para esta anotación. Veamos un ejemplo:

```
label <- gap2007 %>%
  summarise(
    lifeExp = 0.85*max(lifeExp),
    gdpPercap = max(gdpPercap),
    label = "Cuando los paises son suficientemente ricos\n hay
otros factores que influyen más en la salud que el PIB"
  )

ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_vline(aes(xintercept=30000),size=3,color="white") +
  geom_point(aes(color=continent)) +
  geom_point(aes(color=continent),size=3, shape = 1, da-
```

```
ta=paises_longevos) +
  geom_smooth(se=FALSE) +
#  geom_vline(aes(xintercept=30000),linetype=2,size=0.5) +
  geom_text(aes(label = label), data = label, vjust = "top",
hjust = "right")
```



Hagamos algunas observaciones al último gráfico:

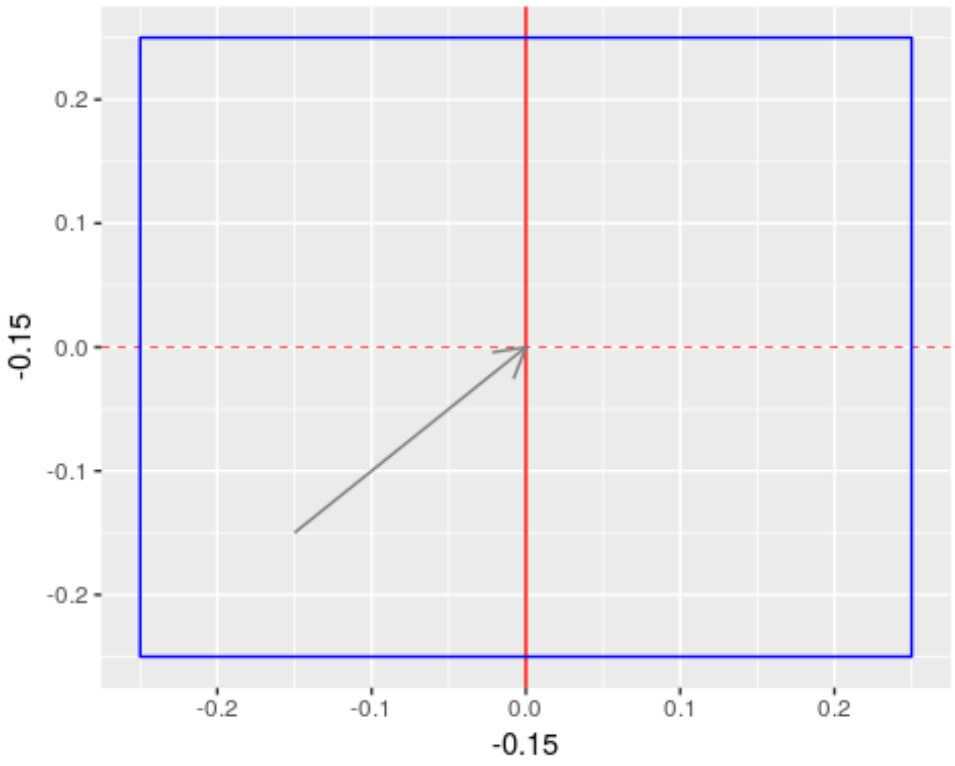
- Hemos creado el data.frame label para posicionar la anotación.
- Para justificar, vertical y horizontalmente, el texto correctamente hemos usado los argumentos `vjust = "top"`, `hjust = "right"` en `geom_text()`
- Hemos añadido un salto manual de linea a la etiqueta mediante ""
- Hemos añadido otro elemento de anotación en el gráfico: una linea vertical mediante `geom_vline()`

Además de `geom_text()` y `geom_label()` tenemos a disposición otras "geoms" en ggplot2 útiles para anotar nuestros gráficos. Algunas ideas:

- Utiliza `geom_hline()` y `geom_vline()` para agregar líneas de referencia. Deben ser discretas para interferir lo menos posible con los elementos principales del gráfico. En el ejemplo he usado una linea blanca gruesa en la primera capa del gráfico. Pueden usarse tambien lineas finas y punteadas.

- Utiliza `geom_rect()` para dibujar un rectángulo alrededor de los puntos de interés. Los límites del rectángulo están definidos por la `aes(xmin, xmax, ymin, ymax)`
- Utiliza `geom_segment()` con el argumento arrow para llamar la atención de un punto con una flecha. Utilice la estética x e y para definir la posición inicial y `xend` e `yend` para definir la ubicación de la punta.

```
ggplot(NULL) +
  geom_vline(aes(xintercept=0),color="red") +
  geom_hline(aes(yintercept=0),color="red",linetype=2,size=0.25)
+
  geom_rect(aes(xmin=-0.25,xmax=0.25,ymin=-
0.25,ymax=0.25),fill=NA,color="blue") +
  geom_segment(aes(x=-0.15,y=-0.15,xend=0,yend=0), arrow
=arrow(length = unit(0.05,"npc")) , color="grey50")
```



2.3. ESCALAS

Otra manera de mejorar los gráficos de cara a la comunicación es ajustar las escalas. Las escalas controlan el mapeo de valores de datos a elementos que podemos percibir. Normalmente, ggplot2 agrega automáticamente las escalas. Por ejemplo, cuando escribimos:

```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent))
```

ggplot2 agrega automáticamente las escalas predeterminadas silenciosamente. Añade una escala para cada estética que añadimos al gráfico, en este caso `x`,`y` y `color`. El gráfico anterior es equivalente a:

```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent)) +
  scale_x_continuous() +
  scale_y_continuous() +
  scale_colour_discrete()
```

Observa el esquema de nomenclatura para las funciones que controlan las escalas: `scale_` seguido del nombre de la estética, seguido de `_`, y del nombre de la escala.

Las escalas predeterminadas se nombran según el tipo de variable con el que se asocian: continuo, discreto, fecha o fecha-hora. Hay un montón de escalas no predeterminadas que veremos a continuación.

Las escalas predeterminadas han sido cuidadosamente seleccionadas para funcionar bien con un rango amplio de datos de entrada. Sin embargo, es posible que, en ocasiones, queramos reemplazar los valores predeterminados por dos razones:

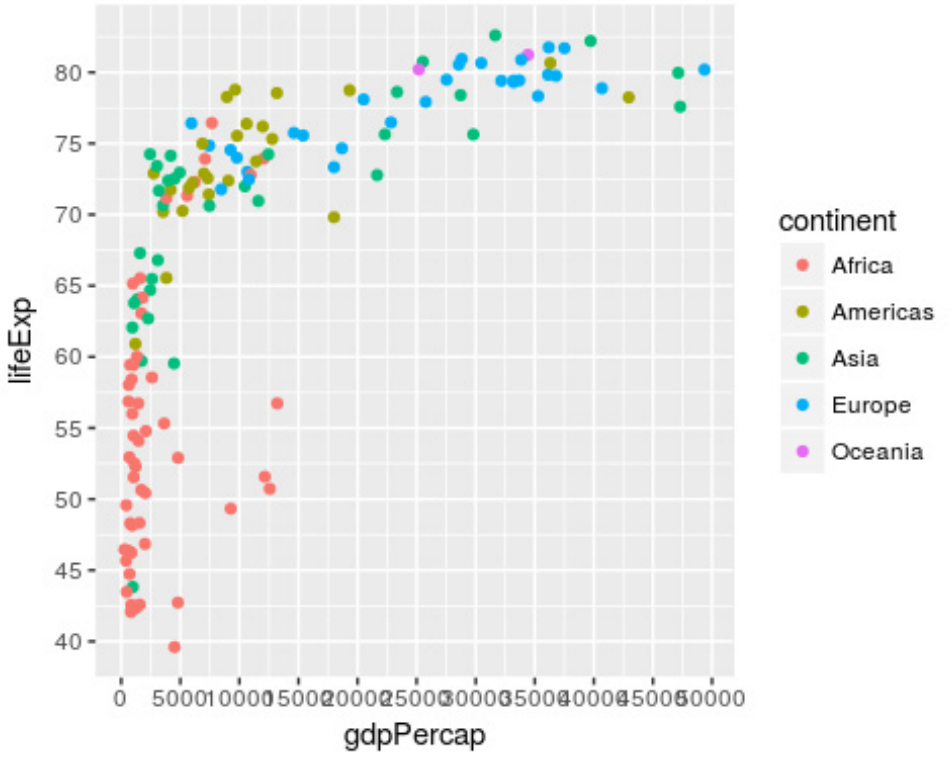
- Es posible que queramos modificar algunos de los parámetros de la escala predeterminada. Esto nos permite hacer cosas como cambiar las marcas en los ejes, o las etiquetas de la leyenda.
- Es posible que queramos sustituir la escala por completo, ya que en algunas ocasiones nos puede interesar una escala diferente a la por defecto.

2.3.1. EJES

Podemos controlar las marcas y etiquetas de los ejes mediante los argumentos `breaks` y `labels` de la función `scale_` correspondiente. Por ejemplo:

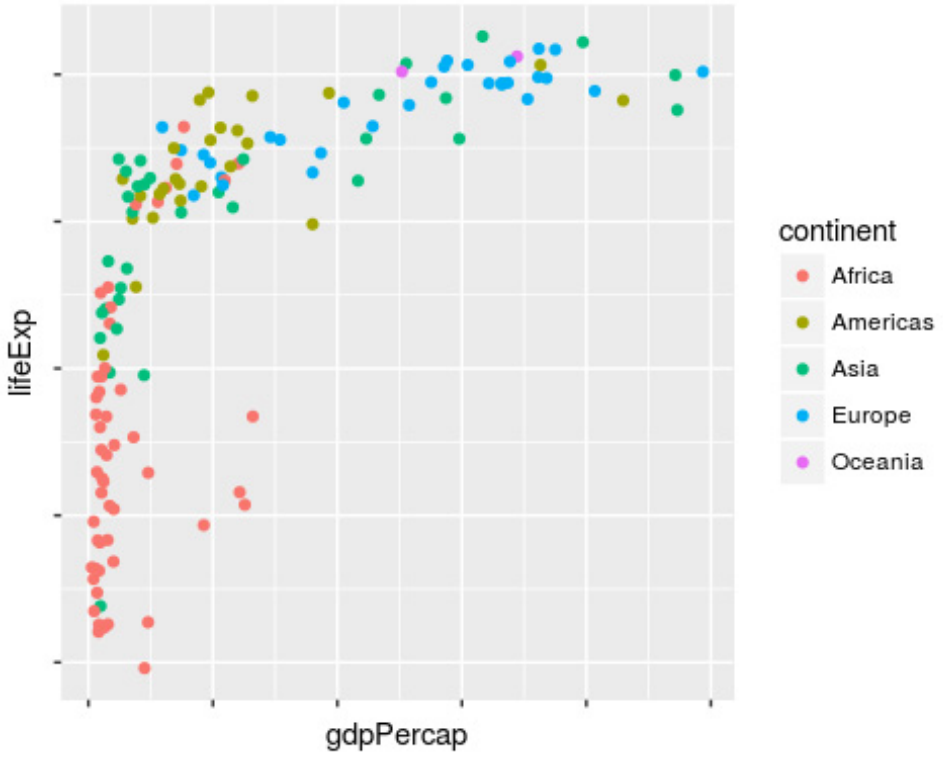
- Marcas en eje y cada 5 años y en el eje x cada 5000 $.

```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent)) +
  scale_x_continuous(breaks = seq(0, 50000, by = 5000)) +
  scale_y_continuous(breaks = seq(35, 85, by = 5))
```



- Si quisiéramos ocultar completamente las marcas de los ejes, por ejemplo por motivos de confidencialidad.

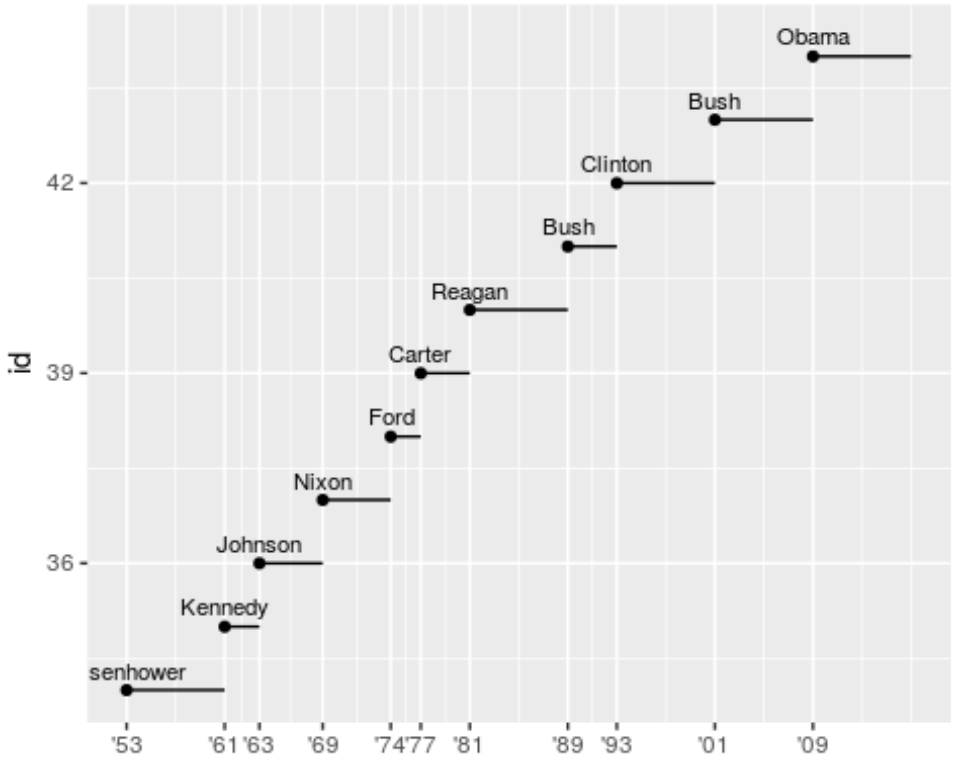
```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent)) +
  scale_x_continuous(labels = NULL) +
  scale_y_continuous(labels = NULL)
```



Otro uso de breaks es cuando tenemos pocos puntos de datos y queremos resaltar exactamente donde se producen las observaciones. Por ejemplo, hagamos un gráfico que muestra cuando cada presidente de EE.UU. comenzó y terminó su mandato. Modificamos las marcas del eje x para que coincidan con el inicio del mandato de cada presidente. El conjunto de datos `presidential` viene incluido en el paquete ggplot2.

```
presidential %>%
  mutate(id = 33 + row_number()) %>% # Creo una variable nueva
para posicionar los presidentes en el eje y
  ggplot(aes(start, id)) +
    geom_point() +
```

```
    geom_segment(aes(xend = end, yend = id)) +
    geom_text(aes(y=id+0.3,label=name),size=3) +
    scale_x_date(NULL, breaks = presidential$start, date_labels =
"'%y")
```



En este caso, es interesante observar como la especificación de los breaks y labels para variables de fecha o fecha-hora es algo diferente a las variables numéricas:

- `date_labels`: toma una especificación de formato del estilo de `format.Date` o `format.POSIXct` (teclea en la consola `?strptime` para más información).
- `date_breaks`: (no la hemos usado) admite una cadena de texto como "2 days" o "1 month".

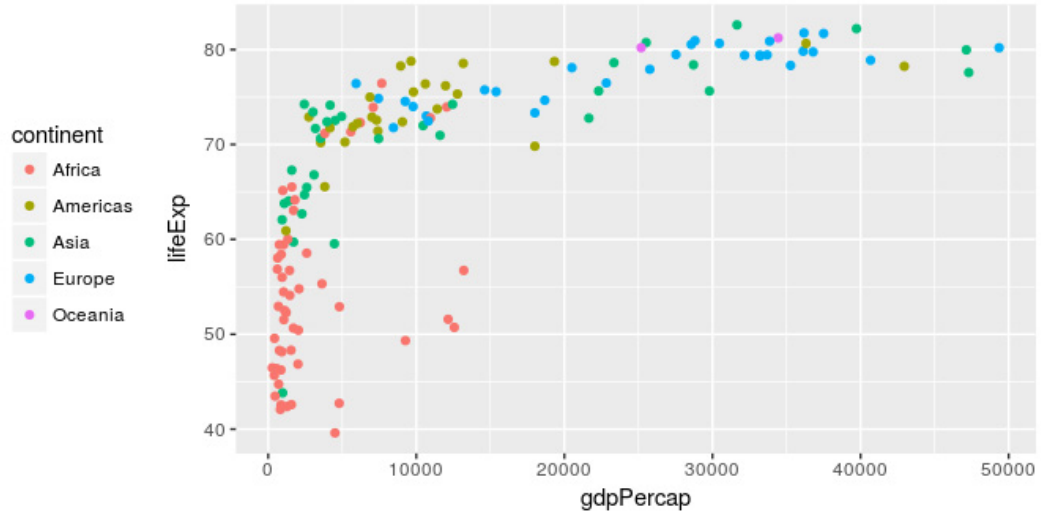
2.3.2. LEYENDAS

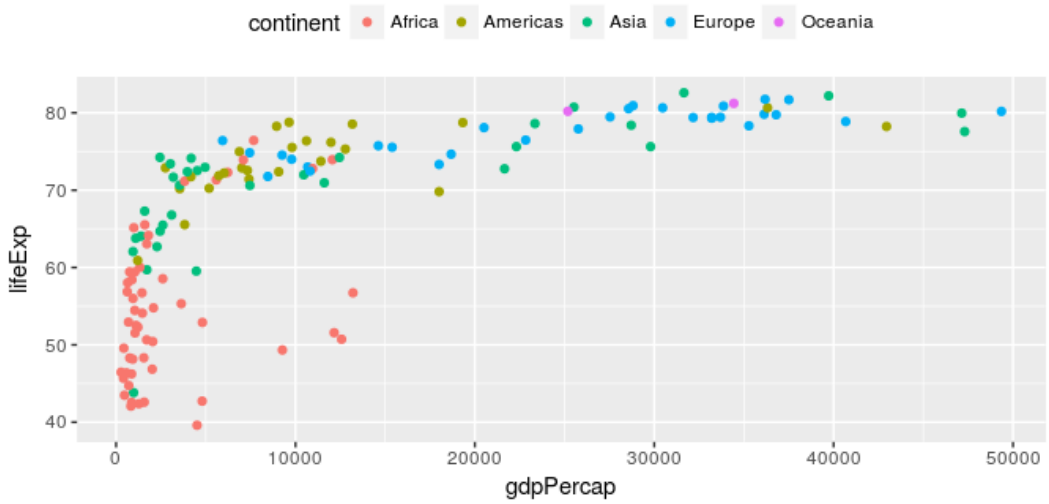
Los argumentos `breaks` y `labels` que hemos visto para ajustar los ejes también funcionan con las leyendas, aunque con estas últimas hay otras modificaciones que son más habituales de utilizar.

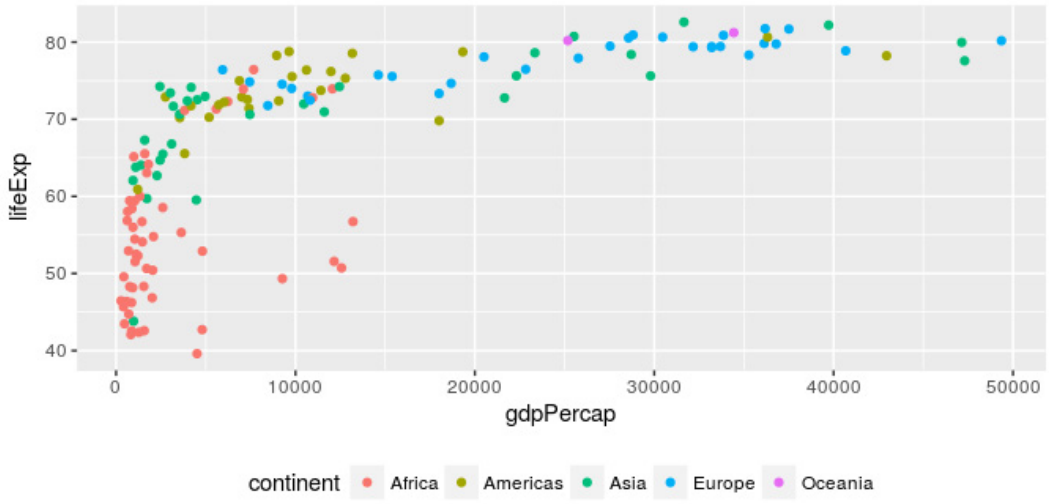
Para controlar la posición de la leyenda, debemos usar una configuración de `theme()`. Veremos con más detalle los temas al final del capítulo, pero en resumen, controlan las partes del gráfico que no tienen que ver con los datos. El argumento `legend.position` controla donde se dibuja la leyenda:

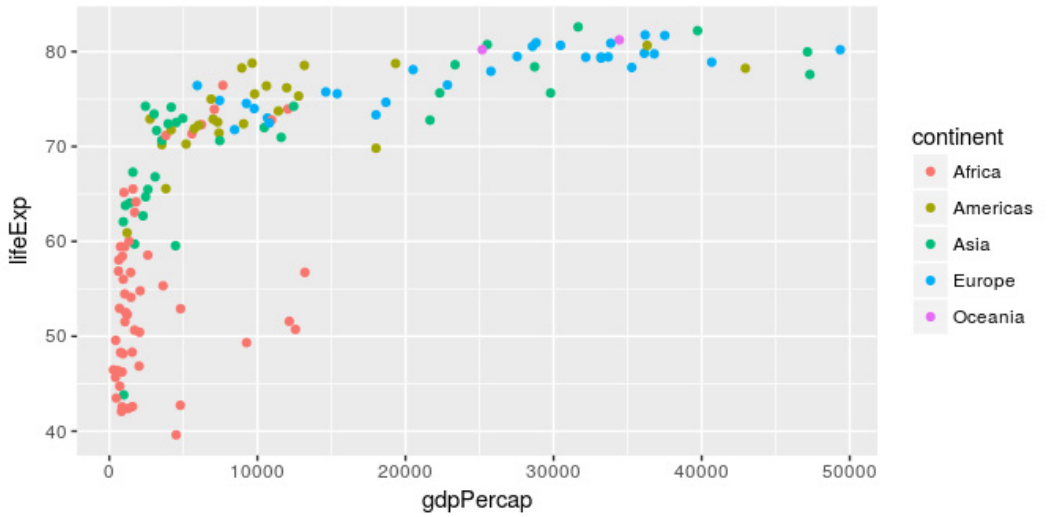
```
base <- ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent))

base + theme(legend.position = "left")
base + theme(legend.position = "top")
base + theme(legend.position = "bottom")
base + theme(legend.position = "right")
```




continent
Africa
Americas
Asia
Europe
Oceania
lifeExp
80
70
60
50
40
0
10000
20000
30000
40000
50000
gdpPercap


lifeExp
80
70
60
50
40
0
10000
20000
30000
40000
50000
gdpPercap
continent
Africa
Americas
Asia
Europe
Oceania

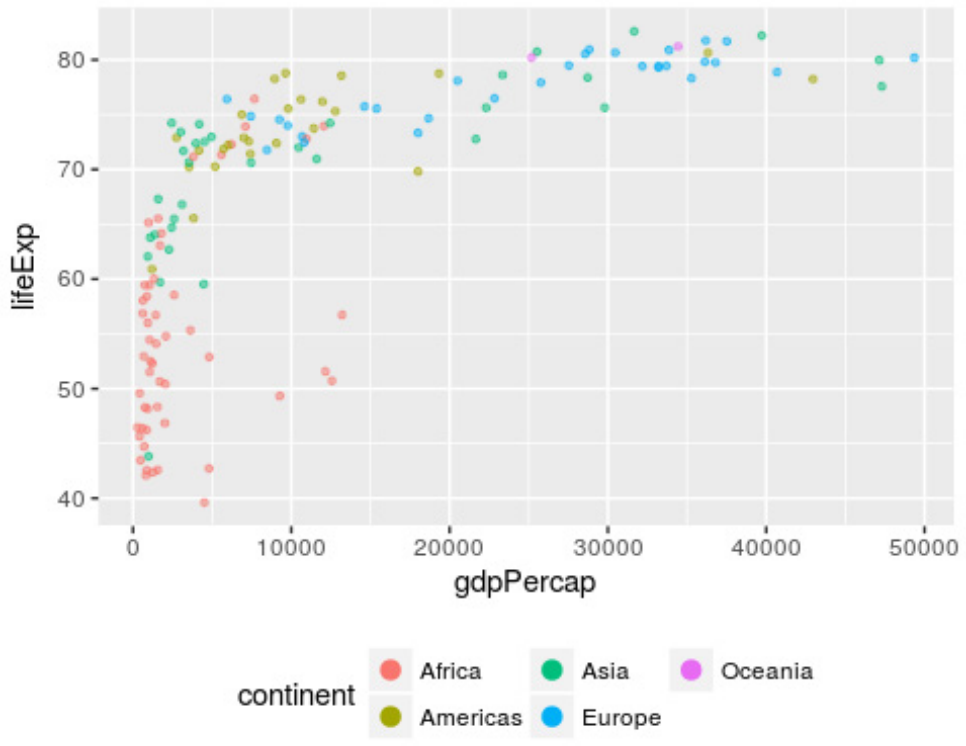

lifeExp
80
70
60
50
40
0
10000
20000
30000
40000
50000
gdpPercap
continent
Africa
Americas
Asia
Europe
Oceania

Puedes emplear `legend.position = "none"` para suprimir la leyenda del gráfico.

Para controlar los detalles visuales de cada leyenda, usaremos `guides()` junto con `guide_legend()` o `guide_colourbar()`.

El siguiente ejemplo controla dos aspectos importantes en la leyenda: primero el número de filas en la leyenda mediante nrow y por último el tamaño de los puntos y su transparencia mediante override.aes. Esto es útil cuando hagamos gráficos con un tamaño de punto muy pequeño o con transparencia alta.

```
# Prueba a hacer el gráfico comentando la última linea
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent),size=1,alpha=0.5) +
  theme(legend.position = "bottom") +
  guides(color = guide_legend(nrow = 2, override.aes = list(size
= 3,alpha=1)))
```

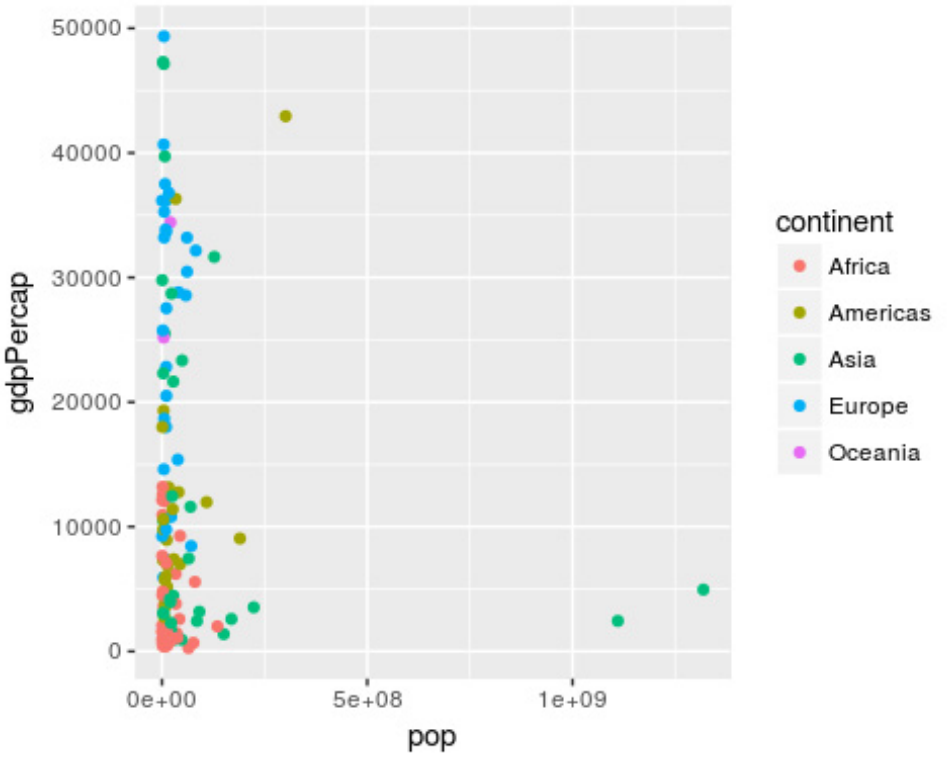


2.3.3. REEMPLAZAR LA ESCALA

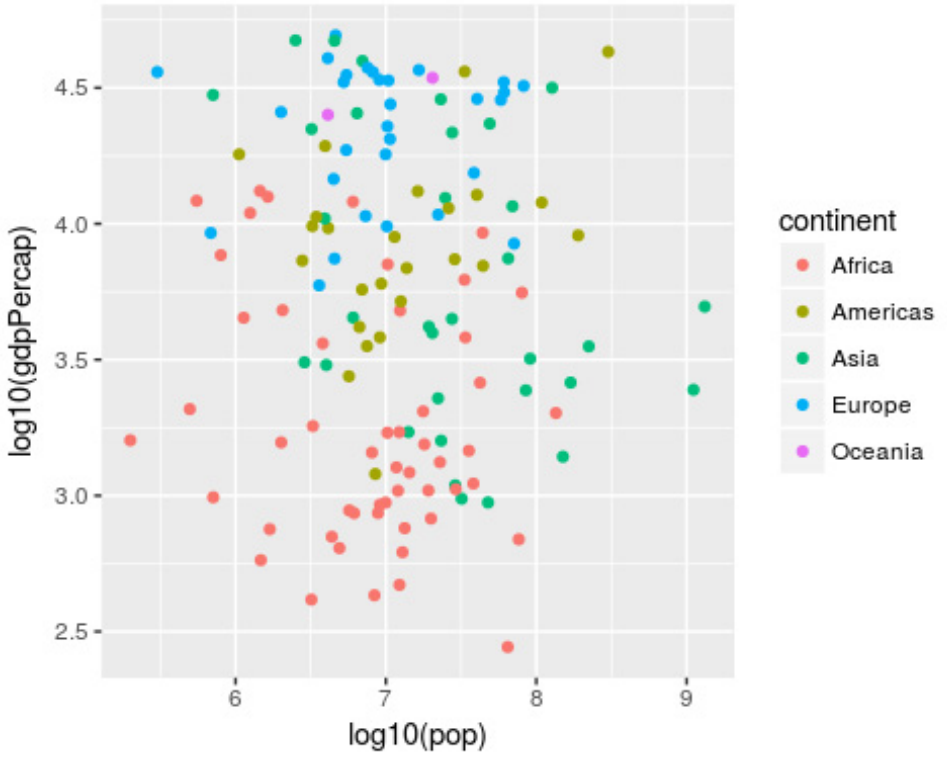
En lugar de ajustar los detalles de una escala, es posible cambiarla por completo.

Cuando alguna variable toma valores en ordenes de magnitud muy diferentes es útil hacer una transformación logarítmica de ellos. Esto lo podemos hacer transformando directamente los datos.

```
ggplot(gap2007, aes(pop,gdpPercap)) +
  geom_point(aes(color=continent))
```

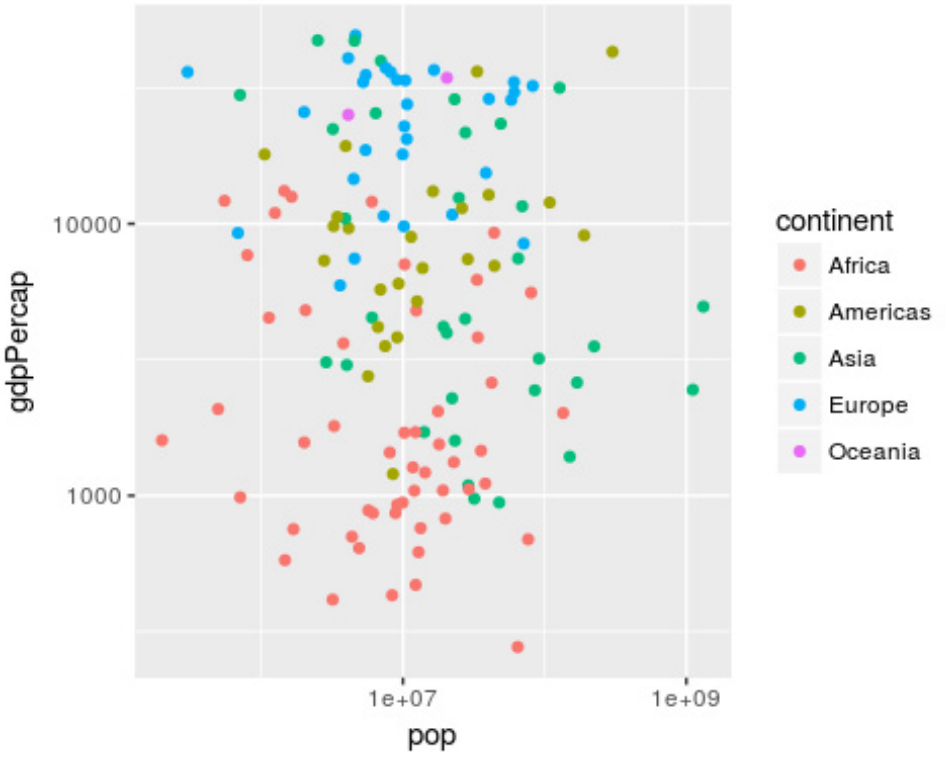



```
ggplot(gap2007, aes(log10(pop),log10(gdpPercap))) +
  geom_point(aes(color=continent))
```



Sin embargo, la desventaja de esta transformación es que los ejes están ahora marcados con los valores transformados, lo que hace difícil interpretar el gráfico. En lugar de realizar la transformación en los datos, podemos hacerlo con la escala. Esto es visualmente idéntico, excepto que los ejes están etiquetados en la escala de datos original.

```
ggplot(gap2007, aes(pop,gdpPercap)) +
  geom_point(aes(color=continent)) +
    scale_x_log10() +
  scale_y_log10()
```



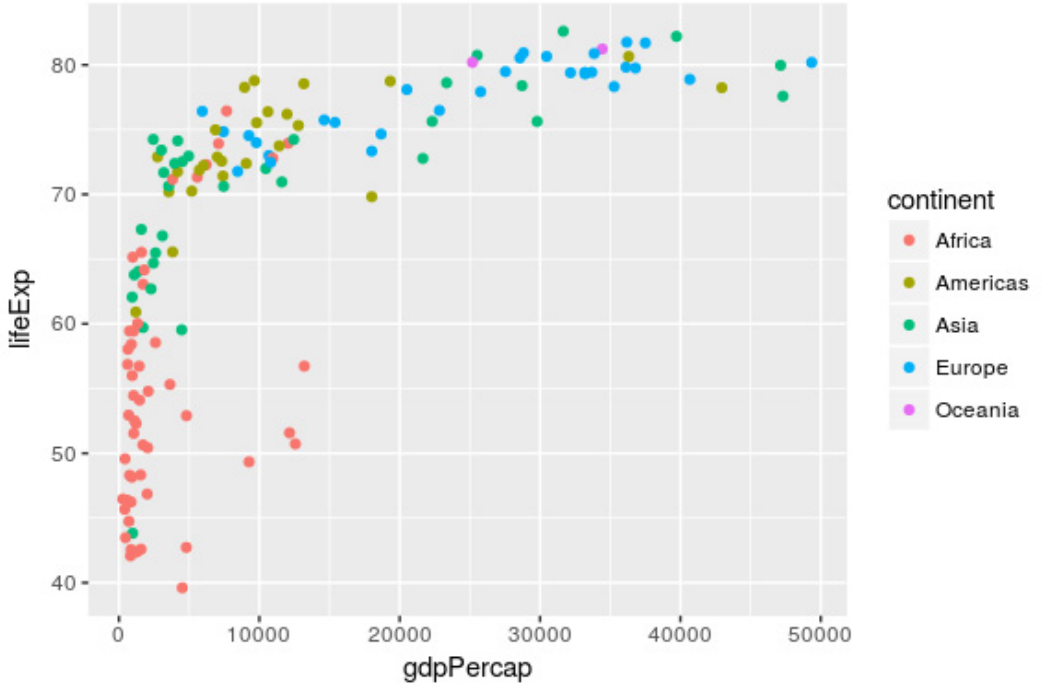
2.3.4. ESCALAS DE COLOR

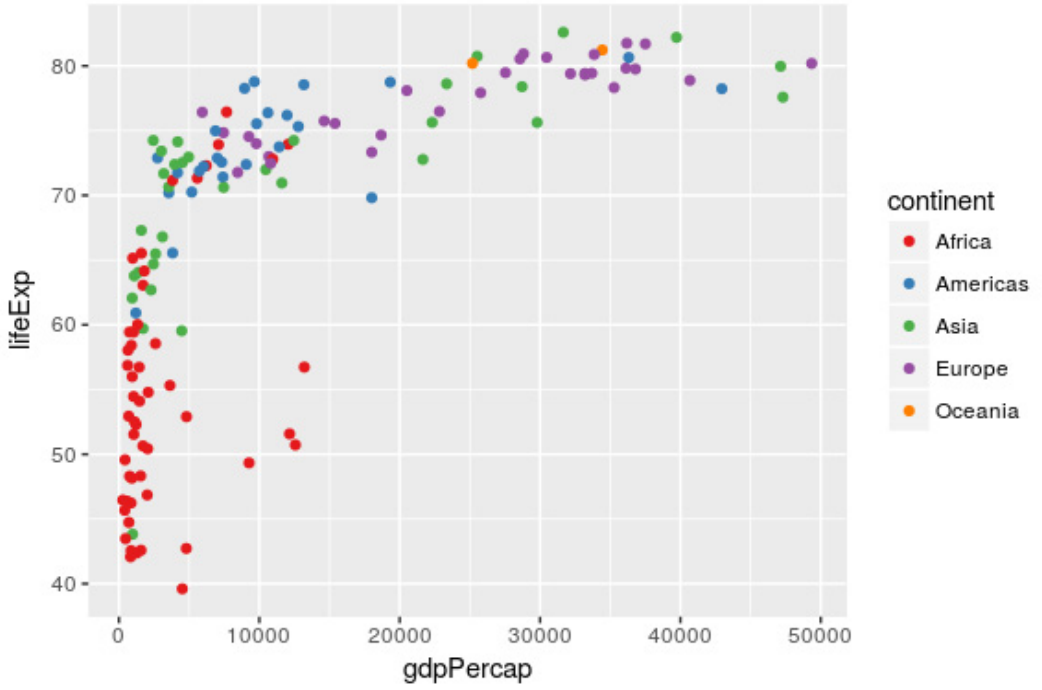
Otra escala que habitualmente se personaliza es el color.

Cuando se colorea según los valores de una variable categórica, la escala de color discreta predeterminada selecciona los colores uniformemente espaciados alrededor de la rueda de color. A veces esta escala es difícil de distinguir por personas con problemas de daltonismo. Una alternativa útil son las escalas que proporciona el paquete RColorBrewer que han sido afinadas manualmente de cara a mejorar la distinción de los colores. Veamos un ejemplo:

```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
    geom_point(aes(color=continent))

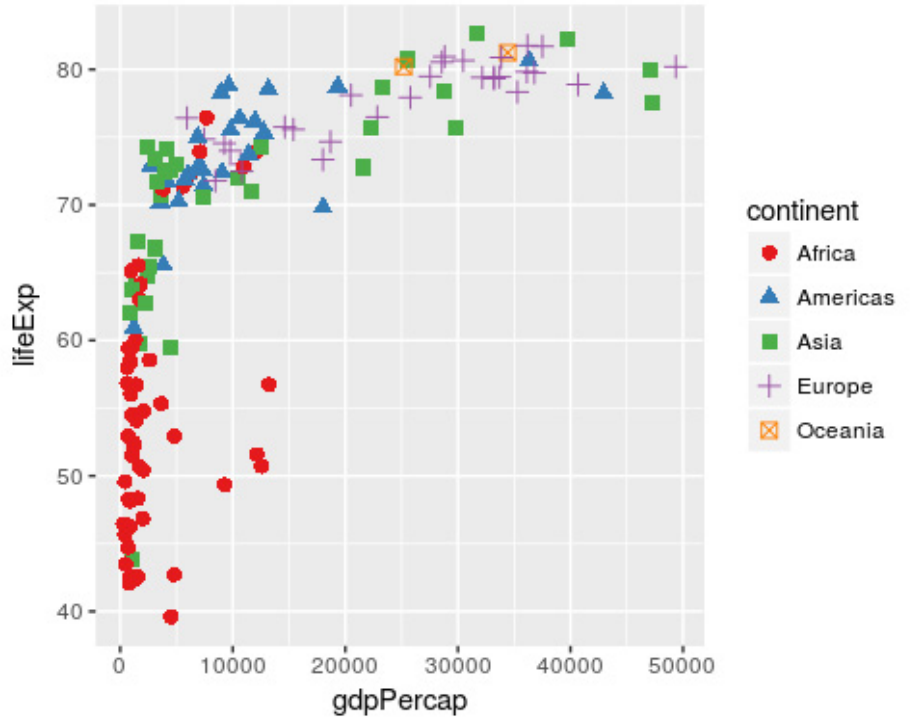
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent)) +
  scale_colour_brewer(palette = "Set1")
```





Y no olvidemos las técnicas más sencillas. Si solo hay unas pocas categorías podemos usar como estética de forma redundante la forma de los puntos. Esto asegura que el gráfico se puede interpretar en una impresión en blanco y negro.

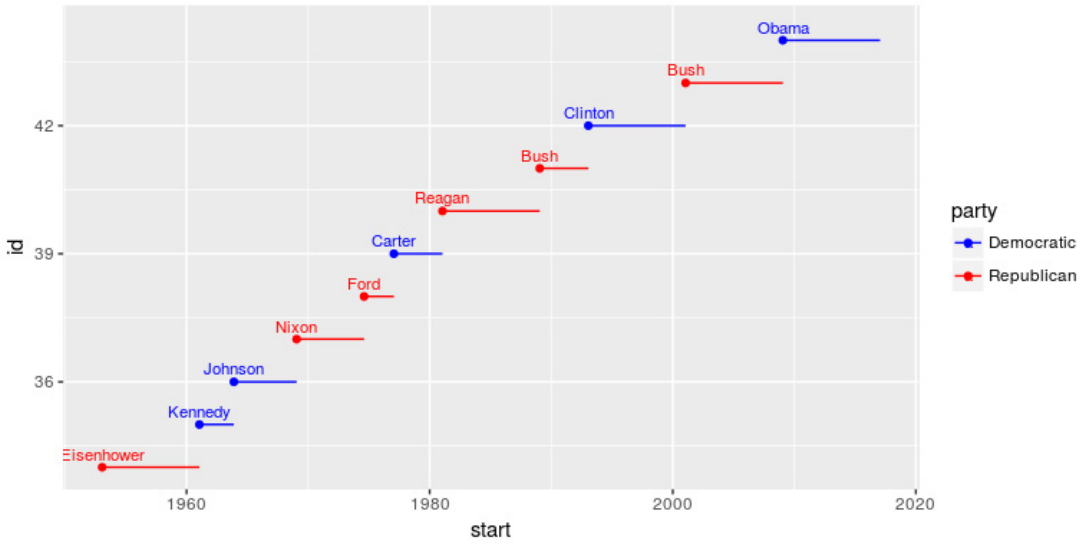
```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent,shape=continent),size=2.5) +
  scale_colour_brewer(palette = "Set1")
```



Las escalas ColorBrewer están documentadas en **http://colorbrewer2.org/** y puedes consultar las paletas disponibles mediante el comando.

Cuando tengamos una relación predefinida entre valores y colores, utilizaremos `scale_colour_manual()`. Por ejemplo, si asignamos al partido presidencial un color, queremos usar el mapeo estándar de rojo para republicanos y azul para demócratas:

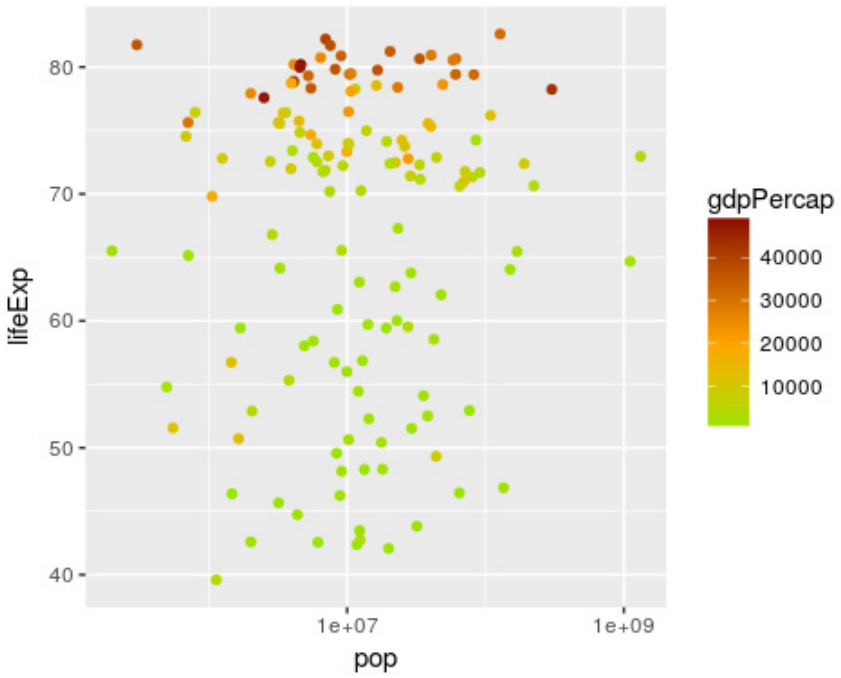
```
presidential %>%
  mutate(id = 33 + row_number()) %>%
  ggplot(aes(start, id, colour = party)) +
    geom_point() +
    geom_segment(aes(xend = end, yend = id)) +
    geom_text(aes(y=id+0.3,label=name),size=3) +
    scale_colour_manual(values = c(Republican = "red", Democratic
= "blue"))
```



Cuando el coloreado se realiza usando una variable continua, es buena idea usar `scale_colour_gradient2()`. Eso te permite, por ejemplo, asignar diferentes colores para valores positivos y negativos de la variable, o si deseas distinguir puntos por encima o por debajo de la media.

```
ggplot(gap2007, aes(pop,lifeExp)) +
  geom_point(aes(color=gdpPercap)) +
  scale_x_log10() +

scale_colour_gradient2(low="green",mid="orange",high="red4",midpo
int =20000)
```



2.4. ZOOM

Hay 3 maneras de controlar los límites de un gráfico:

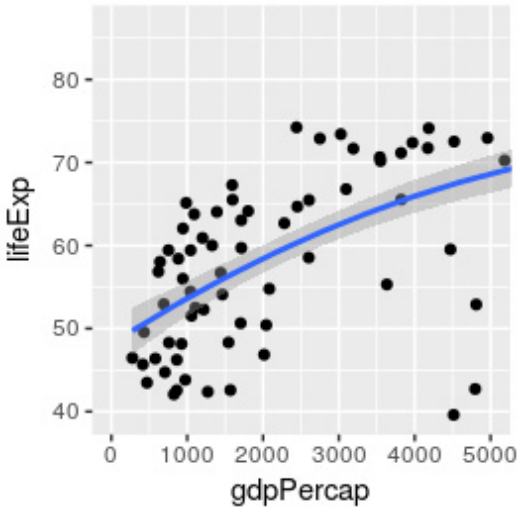
- Ajuste manual de los datos que se representan.
- Establecer los límites en cada escala.
- Configuración de xlim y ylim en `coord_cartesian()`

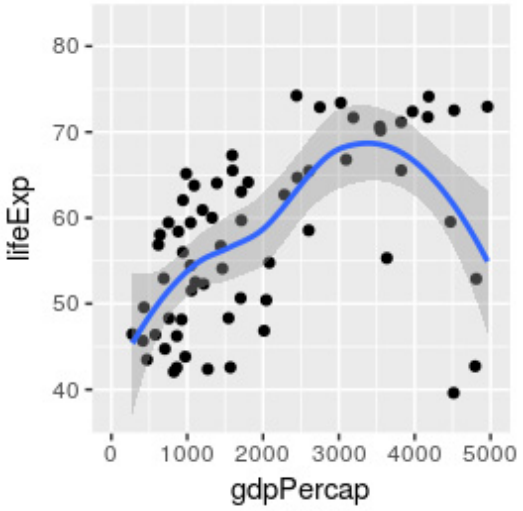
Para ampliar una región del gráfico, generalmente es bastante mejor la opción de utilizar `coord_cartesian()`. Compara los siguientes gráficos:

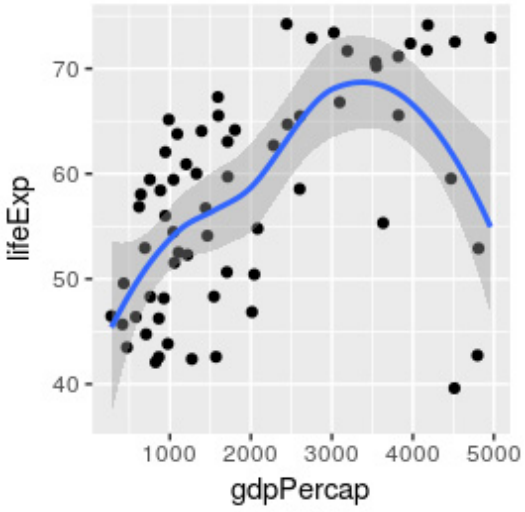
```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point() + geom_smooth() +
  coord_cartesian(xlim = c(0, 5000))

ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point() + geom_smooth() +
  scale_x_continuous(limits = c(0, 5000))

gap2007 %>% filter(gdpPercap<5000) %>%
ggplot(aes(gdpPercap,lifeExp)) +
  geom_point() + geom_smooth()
```


lifeExp
80
70
60
50
40
0
1000
2000
3000
4000
5000
gdpPercap


lifeExp
80
70
60
50
40
0
1000
2000
3000
4000
5000
gdpPercap


lifeExp
70
60
50
40
1000
2000
3000
4000
5000
gdpPercap

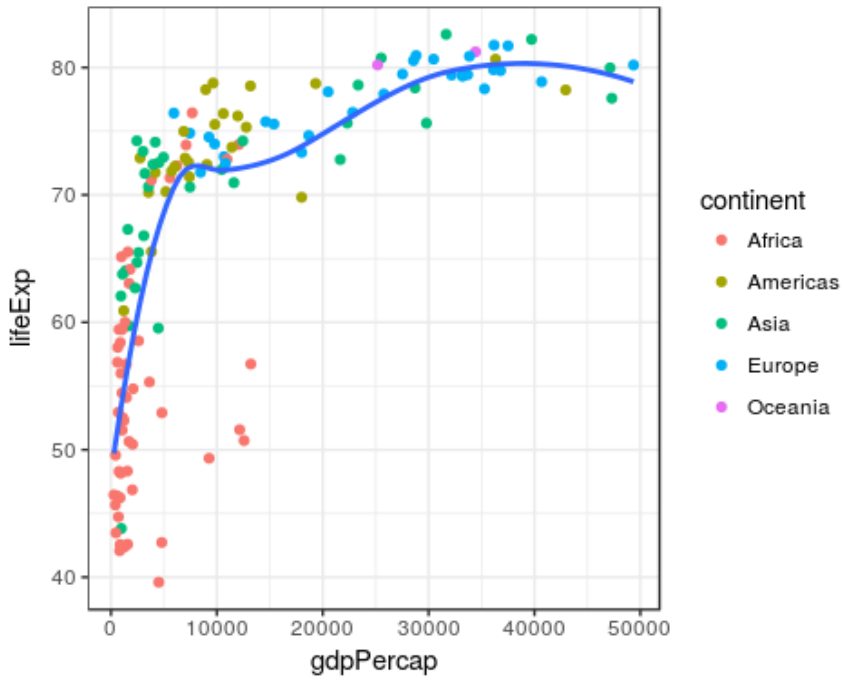
La diferencia entre ellos está en:

- El primero, que usa `coord_catesian()`, hace el ajuste sobre la nube completa de puntos y posteriormente hace el zoom en la región especificada.
- Los dos últimos seleccionan los puntos primero y hacen el ajuste después, difieren ligeramente en los límites precisos de los valores máximos y mínimos que muestran los ejes.

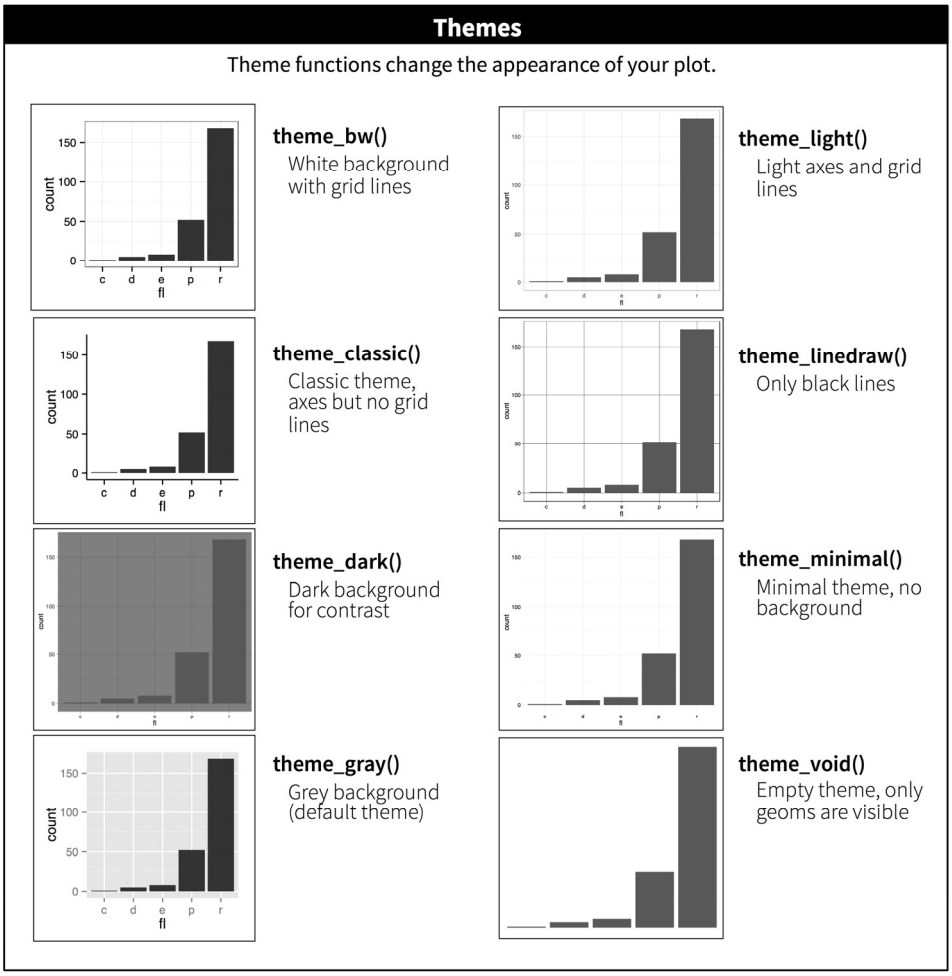
2.5. TEMAS

Finalmente, podemos personalizar los elementos del gráfico no relacionados con los datos mediante un tema. Por ejemplo el siguiente gráfico usa el tema bw, que sustituye el habitual fondo gris por uno blanco.

```
ggplot(gap2007, aes(gdpPercap,lifeExp)) +
  geom_point(aes(color=continent)) + geom_smooth(se=FALSE) +
  theme_bw()
```



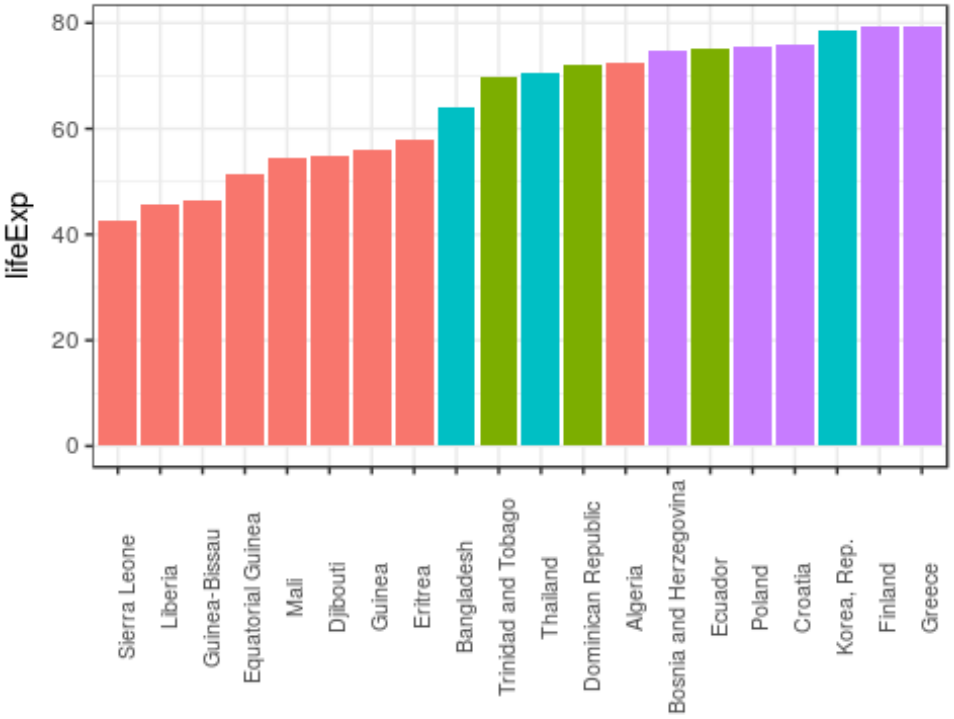
ggplot2 incluye ocho temas de forma predeterminada. Muchos más se incluyen en paquetes adicionales como ggthemes.



Temas incluidos en el paquete ggplot2

Hay otras características de apariencia del gráfico que pueden cambiarse mediante argumentos de la función `theme()` Una modificación muy habitual es cambiar la orientación de las etiquetas del eje x cuando corresponden a una variable categórica:

```
gap2007 %>% sample_n(20) %>%
ggplot() + geom_bar(aes(reorder(country,lifeExp),lifeExp,fill=continent),stat="identity") +
  theme_bw() + theme(legend.position = 'none', axis.text.x = element_text(angle=90,size=8))+
  labs(x="")
```


lifeExp
80
60
40
20
0
Sierra Leone
Liberia
Guinea-Bissau
Equatorial Guinea
Mali
Djibouti
Guinea
Eritrea
Bangladesh
Trinidad and Tobago
Thailand
Dominican Republic
Algeria
Bosnia and Herzegovina
Ecuador
Poland
Croatia
Korea, Rep.
Finland
Greece

3. R MARKDOWN

R Markdown es un entorno de edición de texto que permite combinar código, resultados y texto con formato.

Este tipo de documentos se emplean habitualmente con tres propósitos diferentes:

1. Para comunicar resultados de manera formal, ya sea para un informe/presentación puntual como para generar documentación automatizada de forma periódica. En este caso, se prioriza presentar los resultados y las conclusiones y no el código que hay detrás de los análisis.
2. Para colaborar con otros analistas o científicos de datos, que están interesados en las conclusiones y como se llego a ellas, es decir el código.
3. Como cuaderno de laboratorio del científico de datos, en que capturemos no solo lo que hicimos sino también lo que pensábamos, los análisis exitosos o los menos satisfactorios.

Ojo al dato

Para ti R markdown no es algo nuevo, la documentación de este curso está desarrollada en R markdown y a estas alturas de curso ya habrás abierto varios documentos .Rmd para ejecutar el código que acompaña a nuestras explicaciones. Ahora vas a aprender a crear este tipo de documentos desde cero.

3.1. FUNDAMENTOS DE R MARKDOWN

Para empezar a generar documentos con R markdown solo necesitas el paquete rmarkdown. Aunque desde Rstudio, no es necesario que lo instales, ni lo cargues explícitamente, él se encarga de hacerlo cuando lo necesites.

Un fichero de R Markdown es un fichero de texto plano con la extensión .Rmd. Veamos un ejemplo:

````
---
title: "Gapminder"
output:
  pdf_document: default
  html_document: default
date: '2017-03-22'
---

```{r setup, include = FALSE}
library(ggplot2)
library(dplyr)
library(gapminder)
library(knitr)

data2007 <- gapminder %>% filter(year==2007)
```

#  Paises más poblados

El __pais más poblado__ en el año 2007 es `r da-
ta2007$country[which.max(data2007$pop)]` con una población de `r
max(data2007$pop)` habitantes

Veamos los paises más poblados de cada continente

```{r, echo = FALSE}
maxpop<- gapminder %>% group_by(continent) %>%
  filter(min_rank(-pop)==1)
kable(maxpop)
```

# Evoloución temporal
````

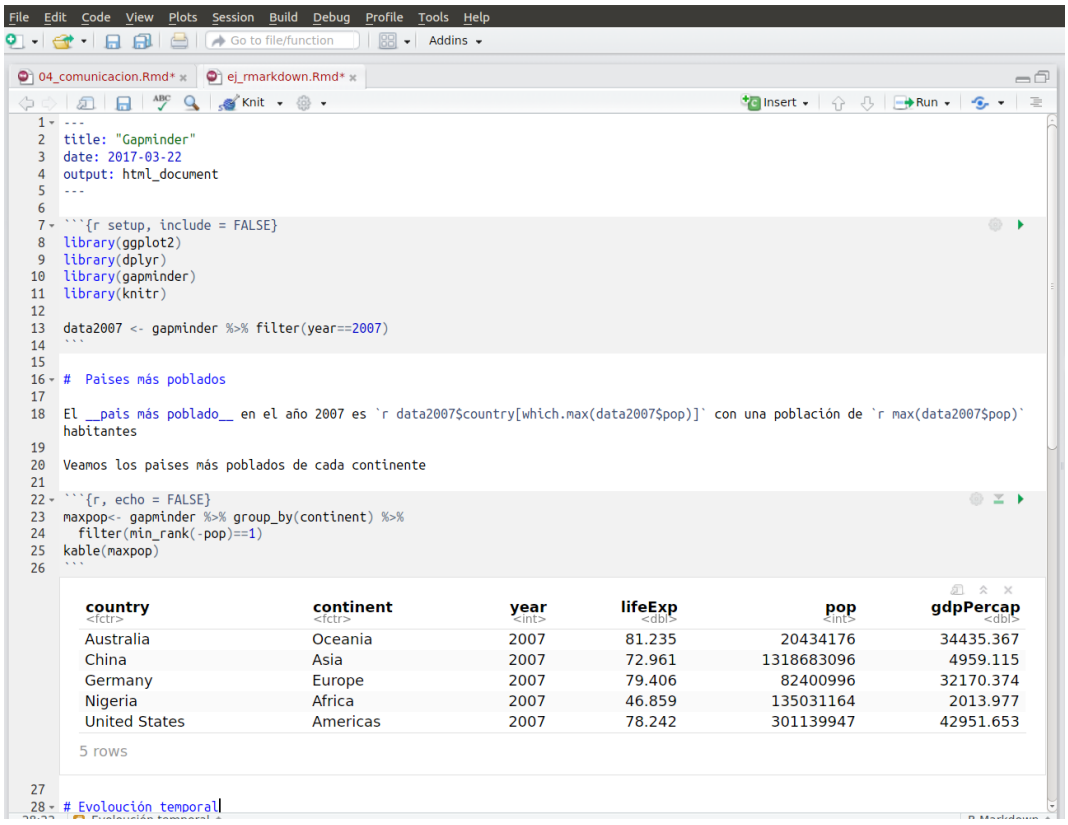
````
La  evolución de la población en estos 5 paises ha sido

```{r,echo=FALSE}
library(gapminder)
gapminder %>% filter(country %in% maxpop$country) %>%
  ggplot() + geom_line(aes(year,pop,color=country,group=country))
```
````

Contiene tres tipos distintos de contenido:

- Una cabecera YAML (opcional) delimitada por `---` .
- Bloques (chunks) de código delimitados por ```` ``` ````.
- Texto que mezcla texto simple con texto formateado como `# Título` y `__negrita__`.

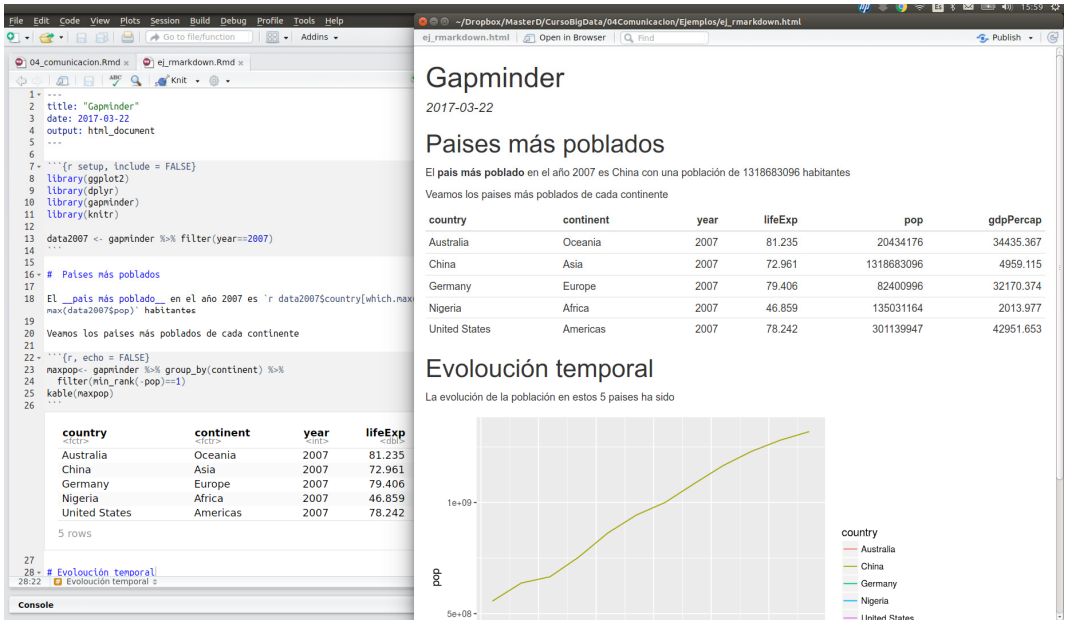
Cuando abres un fichero .Rmd en Rstudio lo que ves es una interfaz tipo notebook en donde el código y el texto están entrelazados, tal y como muestra la siguiente figura:



Puedes ejecutar los chunks de código pulsando el icono de Run que tienes encima de cada chunk (el que parece un botón de play) o tecleando `Ctrl + Shift + Enter`. Rstudio ejecuta el código y mostrará los resultados justo debajo.

Para producir un informe completo que contenga todo el texto, código y resultados, haz clic en "Knit" o presiona `Ctrl + Mayús + K`. También puede ejecutarse en la consola escribiendo `rmarkdown::render("tufichero.Rmd")`.

Esto mostrará el informe en un visor y creará fichero HTML (o el formato especificado) que puedes abrir en tu navegador y compartir con otras personas.



Cuando se procesa el documento R Markdown envía el archivo .Rmd a knitr (**http://yihui.name/knitr/**), que ejecuta todos los fragmentos de código y crea un nuevo documento de markdown (.md) que incluye el código y su salida. El archivo de markdown generado por knitr es procesado por pandoc (**http://pandoc.org/**), que es el responsable de crear el archivo finalizado. La ventaja de este flujo de trabajo en dos pasos es que se puede crear una amplia gama de formatos de salida para un mismo documento, como veremos más adelante.

Puedes comenzar por tu cuenta a crear un fichero .Rmd. Selecciona `File -> New File -> R Markdown` en la barra de menús. RStudio creara un documento plantilla con contenido útil que le recuerda cómo funcionan las características principales de R Markdown.

```
---
title: "Untitled"
author: "MasterD"
date: "27 de marzo de 2017"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

3.2. TEXTO EN MARKDOWN

El texto en archivos .Rmd está escrito en Markdown, es un lenguaje de marcado ligero que proporciona un conjunto pequeño de instrucciones para dar formato al texto desde un fichero de texto plano. Markdown está pensado para que se pueda leer y escribir con facilidad.

A continuación te mostramos una pequeña guía de cómo usar Markdown:

```
Formato de texto
-----------------------------------------------------------------
*cursiva*  or _cursiva_
**negrita**   __así también en negrita__
`código`
superíndice^2^ and subscript~2~


Títulos
-----------------------------------------------------------------

# Título Nivel 1

## Título Nivel 2

### Título Nivel 3


Listas
-----------------------------------------------------------------

*   Lista con viñetas -  Elemento  1

*   Elemento 2

    * Elemento 2a

    * Elemento 2b


1.  Lista numerada -  Elemento  1

1.  Elemento 2. Los números se incrementan automaticamente en la
salida.


Links and images
-----------------------------------------------------------------

<http://ejemplo.es>
```

```
[Texto del hipervínculo](http://ejemplo.es)

![Texto al pie de imagen, opcional](path/to/img.png)


Tablas
---------------------------------------------------------

Cabecera 1    | Cabecera 2
------------- | -------------
aaaa          | 1.23
bbb           | 5.67
```

Puedes obtener ayuda sobre los comando básicos de Markdow en Rstudio mediante el menú Help -> Markdown Quick Reference.

Reto

Crea un nuevo fichero de Markdown e introduce todos los elementos de Markdown mostrados arriba. Compílalo pulsando el botón de Knit y observa el documento resultante. Ten cuidado si insertas imágenes que se encuentren en la ruta especificada.

3.3. FRAGMENTOS DE CÓDIGO (CODE CHUNKS)

Para ejecutar código dentro de un documento de Markdown de R, debes insertar un fragmento de código(chunk). Hay tres formas de hacerlo:

- El atajo de teclado `Ctrl + Alt + I`
- El botón "Insert" en la barra de herramientas.
- Escribiendo manualmente ```` ```{r} ```` y ```` ``` ````

Puedes ejecutar el código del chunk, comando a comando, mediante el atajo de teclado que ya conoces `Ctrl + Enter`. Para ejecutar el bloque completo debes usar `Ctrl + Shift + Enter` o el icono parecido al "Play"" situado en la parte superior derecha del chunk.

El contenido de cada fragmento debe estar centrado en una única tarea. Si quieres hacer más de una cosa inserta diferentes chunks, así estará mejor organizado tu documento.

3.3.1. NOMBRE DEL CHUNK

Es posible dar un nombre (opcional) a los chunks mediante: ```` ```{r nombre-chunk}``` ````. Esto tiene dos ventajas aparentes:

- Es más fácil navegar sobre los chunks mediante el desplegable de la parte inferior izquierda del editor.
- Los gráficos que producen los chunks tienen nombres de fichero fáciles de identificar para poder utilizarlos en otra parte.

3.3.2. OPCIONES DEL CHUNK

La salida de los chunks se puede personalizar mediante opciones, es decir mediante argumentos suministrados en la cabecera de chunk.

Knitr proporciona casi 60 opciones que pueden utilizar para personalizar los chunks de código.

Aquí vamos a ver las opciones más importantes, las que controlan si se ejecuta el bloque de código y qué resultados se insertan en el documento final:

- `eval = FALSE` evita que se evalúe el código. (Y, obviamente, si el código no se ejecuta, no se generarán resultados). Esto es útil para mostrar código de ejemplo.
- `include = FALSE` ejecuta el código, pero no muestra el código o los resultados en el documento final. Utilízalo para el código de configuración o pre-cálculos necesarios para el informe.
- `echo = FALSE` no muestra el código, pero si los resultados. Utilízalo cuando redactes informes dirigidos a personas que no deseen ver el código R subyacente.

- `message = FALSE` o `warning = FALSE` impide que los mensajes o warnings provocados en la ejecución del código R aparezcan en el documento final.
- `results = 'hide'` oculta la salida impresa y `fig.show = 'hide'` no muestra los gráficos.
- `error = TRUE` hace que el procesado continúe, aunque el código ejecutado devuelva un error. Esto no es lo deseable en la versión final de tu informe, pero puede ser muy útil si necesitas depurar exactamente lo que está pasando dentro de tu .Rmd.

También se controlan mediante argumentos de los chunks las características de los gráficos que genera el código. A esto dedicamos una sección del texto más adelante.

Puedes ver la lista completa de opciones en **http://yihui.name/knitr/options/**

3.3.3. TABLAS

De forma predeterminada, R Markdown imprime los data frames y las matrices como los verías en la consola:

```
mtcars[1:5, ]
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am
gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
3    2
```

Si prefieres que los datos se muestren con mejor formato, utilizaremos la función `knitr :: kable`. El código siguiente genera la siguiente tabla:

```
knitr::kable(mtcars [1:5,1:9], caption = "tabla formateada por
kable")
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0

Tabla formateada por *kable*

Si lees la ayuda `?knitr::kable` podrás ver muchas opciones en las que puedes personalizar la tabla.

3.3.4. GRÁFICOS

Cuando un chunk de código devuelve una figura se mostrará en el documento final. Sin embargo controlar el tamaño y la alineación final para que se muestre de la manera deseada puede ser a veces difícil.

Si no nos gustan los resultados por defecto, podemos controlar el tamaño de la figura mediante 5 parámetros del chunk que podemos utilizar: `fig.width, fig.height, fig.asp, out.width` y `out.height`. Pero no es necesario modificar todos a la vez:

- Hay que distinguir entre el tamaño de la figura que crea R (`fig.width` y `fig.height`) y el tamaño en que la inserta en el documento final (`out.width` y `out.height`). `fig.width` y `fig.height` toman un valor numérico y representa el tamaño en pulgadas. Por defecto vale 7. Los parámetros `out.width` y `out.height` son más flexibles, pueden ser un porcentaje del ancho del documento, `out.width="100%"` o un valor numérico con unidades `out.height="300 px"`

- `fig.asp` controla el ratio entre la altura y la anchura de la gráfica. Si esta fijado la `fig.heigth` de calcula como `fig.width*fig.asp`.
- `out.width` y `out.height` funcionan para documentos html o pdf, pero no para documentos word o .odt
- Si fijamos `out.width ="90%"`, entonces cambiar los valores de fig.width cambia el tamaño del texto y los elementos del gráfico respecto al marco global.

Otras opciones importantes a tener en cuenta en los chunks que devuelven imágenes son:

- `fig.align`: controla la alineación de la figura en el documento. `fig.align="center"` mostrará la figura centrada.
- `fig.show='hold'` permite en un chunk que muestra varios gráficos que estos se muestren después del código y no intercalados.
- `fig.cap = "texto"` : Permite insertar un caption a la figura, es decir texto explicativo en la parte inferior.
- `dev` : especificamos el formato de la imagen del gráfico generada. Por ejemplo si la salida es html, `dev="png"` por defecto. Sin embargo si la salida es un documento PDF `dev="pdf"` por defecto. Si un gráfico contiene muchos puntos, en formato pdf ocupará mucho y tardará mucho tiempo en cargarse, en ese caso es recomendable cambiar en el chunk el formato de salida haciendo dev="png".
- Es una buena idea nombrar chunks de código que producen gráficos, incluso si no etiquetamos rutinariamente otros chunks. La etiqueta del chunk se utiliza para generar el nombre de archivo del gráfico en el disco, así que de esta manera es más fácil identificar los ficheros del gráfico y reutilizarlos en otro contexto, como un email o un tweet.

3.3.5. OPCIONES GLOBALES

A medida que trabajes más con knitr, descubrirás que algunas de las opciones predeterminadas de los chunks no se adaptan a tus necesidades y querrás cambiarlas. Puedes hacerlo mediante `knitr::opts_chunk$set()` en un fragmento de código.

Por ejemplo, si estás preparando un informe y quieres ocultar todo el código, puedes establecer:

```
knitr::opts_chunk$set(echo = FALSE)
```

Esto ocultará el código de forma predeterminada, por lo que solo mostrará los bloques de código que deliberadamente elijas mostrar (con echo = TRUE).

3.3.6. CÓDIGO EN LINEA

Hay otra forma de insertar código R en un documento de R Markdown: directamente en el texto, con: `` `r codigo` ``. En el ejemplo que pusimos al principio del capítulo, decía:

El **pais más poblado** en el año 2007 es `` `r data2007$country[which.max(data2007$pop)]` `` con una población de `` `r max(data2007$pop)` `` habitantes.

que después de procesarse queda en:

El pais más poblado en el año 2007 es China con una población de 1318683096 habitantes

R realiza los cálculos e incrusta los resultados en linea con el texto.

Cuando introduzcamos cálculos en linea que puedan dar como resultado números con decimales debemos usar format o round para controlar el número de decimales que se imprimen en pantalla. Veamos la diferencia.

El número pi es `` `r pi` ``, se redondea con tres cifras decimales a `` `r round(pi,3)` `` y si queremos limitar las cifras significativas a dos `` `r format(pi,digits=2` ``

Al procesarlo resulta:

El número pi es 3.1415927, se redondea con tres cifras decimales a 3.142 y si queremos limitar las cifras significativas a dos 3.1

3.3.7. ECUACIONES MATEMÁTICAS

Podemos escribir ecuaciones matemáticas usando la sintaxis del sistema de edición de textos científicos Latex.

https://en.wikibooks.org/wiki/LaTeX/Mathematics te puede servir como referencia rápida para escribir ecuaciones en este sistema.

Podemos escribir ecuaciones en linea `$\frac{1}{n}\sum_{i}x^{2}_{i}$` o como objeto propio.

```
$$ f(x) = \frac{A_0}{\sqrt{1+x^3}} $$
```

Que una vez procesado el documento se visualizaría como:

La ecuación en linea se ve así $\frac{1}{n}\sum_{i}x_{i}^{2}$ y la independiente:

$$f(x) = \frac{A_0}{\sqrt{1+x^3}}$$

3.4. FORMATOS DE SALIDA

Para controlar algunos detalles del documento de salida usaremos la cabecera YAML. YAML significa "yet another markup language"" (otro lenguaje de marcado más), que está diseñado para representar datos jerárquicos de una manera fácil de leer y escribir.

Hasta ahora los ejemplos de documentos de R Markdown que hemos dado creaban un documento de salida en html. Hay dos maneras de cambiar el formato de salida:

1. Permanentemente, modificando la cabecera YAML:

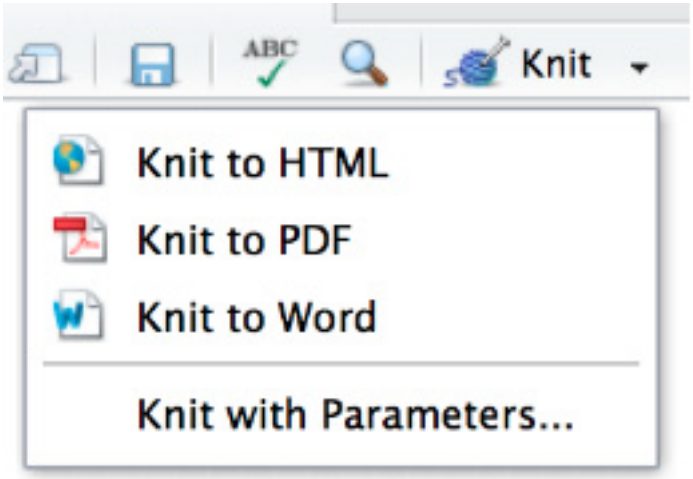
```
title: "Demo R markdown"
output: pdf_document
```

2. De forma transitoria ejecutando `rmarkdown::render()` a mano en la consola:

   ```
   rmarkdown::render("ej_rmarkdown.Rmd", output_format =
   "word_document")
   ```

   Esta forma es útil cuando queremos, mediante un programa, generar documentos en múltiples formatos.

El botón Knit de Rstudio procesa el fichero al primer formato especificado en el campo 'output' de la cabecera. Pero es posible procesar a otros formatos mediante el menú desplegable al lado del botón Knit.



Es posible cambiar la configuración por defecto de los documentos de salida mediante un campo `output` expandido. Por ejemplo, si quieres crear un `html_document` con una tabla de índice flotante, utiliza:

```
output:
  html_document:
    toc: true
    toc_float: true
```

También puedes especificar opciones para formatos múltiples:

```
output:
  html_document:
    toc: true
    toc_float: true
  pdf_document: default
```

Fíjate en la sintaxis especial si no quieres modificar los parámetros por defecto en tipo de documento, una vez que hemos expandido el campo output.

3.4.1. DOCUMENTOS

Los principales formatos de salida son:

- `html_document` crea un documento html. Es la opción por defecto.
- `pdf_document` crea un documento PDF mediante LaTeX (es un sistema para creación de documentación científica), que necesitarás instalar si no lo has hecho anteriormente. RStudio te avisará en caso de necesitarlo.
- `word_document` para documentos de Microsoft Word (`.docx`).
- `odt_document` para documentos en formato OpenDocument Text (`.odt`).
- `rtf_document` para documentos de texto enriquecido (`.rtf`).

Recuerda que si quieres esconder todo el código para el documento final lo puedes hacer cambiando una opción global de knitr mediante un chunk de código:

```
knitr::opts_chunk$set(echo = FALSE)
```

Si la salida es en html, hay otra opción que es hacer que los chunks de código estén ocultos inicialmente pero puedan hacerse visibles con un click.

```
output:
  html_document:
    code_folding: hide
```

3.4.2. NOTEBOOKS

Un notebook, `html_notebook`, es una variante de `html_document`. La salida procesada es muy parecida pero el propósito es diferente. Un `html_document` se centra en la comunicación con el público general, mientras que un notebook se centra en la colaboración con otros científicos de datos. El documento de salida del notebook es un documento `.nb.html` que aparentemente es igual al del documento html pero que se puede utilizar de dos formas:

1. Lo puedes abrir en un explorador web y ver la salida procesada. Pero, al contrario que un `html_document`, este procesado siempre incluye una copia del código que ha generado el documento.
2. Puedes abrirlo y editarlo desde RStudio. Cuando abres un fichero `.nb.html`, RStudio crea el fichero .Rmd que lo generó.

3.4.3. PRESENTACIONES

También se puede usar R Markdown para crear presentaciones. Se obtiene menos control visual que con una herramienta como PowerPoint, pero insertar automáticamente los resultados de tu código R en una presentación te puede ahorrar una gran cantidad de tiempo.

Las presentaciones funcionan dividiendo su contenido en diapositivas, con una nueva diapositiva comenzando en cada primer (`#`) o segundo (`##`) encabezado de nivel.

R Markdown viene con tres formatos de presentaciones incorporados:

1. `ioslides_presentation` - Presentación HTML con ioslides.

2. `slidy_presentation` - Presentación HTML con W3C Slidy.
3. `beamer_presentation` - Presentación en PDF con LaTeX Beamer.

Otros formatos populares se proporcionan a través de paquetes, como por ejemplo:

- `revealjs :: revealjs_presentation` - Presentación en HTML con reveal.js. Requiere el paquete **revealjs**. Es el usamos para las presentaciones en este curso.

Puedes crear un fichero de presentación mediante File -> New File -> R Markdown, eligiendo como tipo de formato "Presentation" y el motor preferido. Para una presentación usando reveal.js debes elegir como formato "From Template" y elegir la plantilla de reveal.js, siempre tengamos el paquete instalado.

3.4.4. OTROS FORMATOS

Es posible generar otros formatos de salida como:

- Cuadros de mando (Dashboards).
- Libros.
- Artículos de revistas científicas.
- Sitios web.

Y puedes añadir elementos interactivos mediante shiny:

- **http://shiny.rstudio.com/**

Esto no está contenido en este curso por razones de tiempo pero te animamos a que los explores. Puedes empezar leyendo el capítulo dedicado en el libro R for Data Science:

- **http://r4ds.had.co.nz/r-markdown-formats.html**

¿QUÉ HAS APRENDIDO?

En esta unidad has aprendido a elaborar documentos y graficos para comunicar de forma efectiva y vistosa tus resultados.

La forma más fácil de transmitir resultados cuantitativos es mediante buenos gráficos, y a ello hemos dedicado gran parte de esta unidad.

Aquí te hemos proporcionado una introducción a algunas herramientas y te hemos dado buenos consejos que te van a ayudar a comunicar mejor los resultados que obtengas en proyectos de ciencia de datos, pero la maestría la alcanzarás experimentando por tu cuenta y mediante la experiencia.

Además recuerda que has aprendido el manejo de herramientas que permiten:

Que tu trabajo sea reproducible, por ti mismos pasado el tiempo, o por otras personas.

Automatizar tareas: podrás programar informes, dashboards o sitios web que se actualicen automáticamente a partir de una única plantilla.

AUTOCOMPROBACIÓN

1. **El propósito principal del título de un gráfico es:**

   a) Describir el contenido del gráfico.

   b) Resumir el hallazgo principal.

   c) Proporcionar la fuente de los datos.

   d) Ninguna de las anteriores.

2. **¿Cuál de las siguientes geometrías de ggplot no te sirve para anotar un gráfico?**

   a) geom_smooth.

   b) geom_segment.

   c) geom_text.

   d) geom_vline.

3. **¿Qué paquete de R nos puede ayudar a evitar que etiquetas de datos se solapen en un gráfico?**

   a) GGally.

   b) ggplot.

   c) ggrepel.

   d) gglabels.

4. ¿Con qué función de ggplot2 cambiarias la escala de opacidad de los puntos de un gráfico? (Usa la ayuda de R si tienes dudas).

a) scale_size().

b) scale_alpha().

c) scale_shape().

d) scale_radius().

5. Para escribir el título de una sección Markdown escribimos:

a) `# Título`

b) `**Título**`

c) `__Título__`

d) `*Título*`

6. Un chunk de código en Rmarkdown devuelve un data.frame. ¿Que comando usarías para formatear la salida en forma de tabla?

a) render.

b) mable.

c) ggtable.

d) kable.

7. ¿Qué paquete es necesario instalar para realizar una presentación como las que usamos en este curso?

a) rpoint.

b) beamer.

c) revealjs.

d) knotes.

8. ¿Si quieres mostrar el código de un chunk pero no quieres que se ejecute, ¿qúe argumento usarías para el chunk?

a) eval=TRUE.

b) eval=FALSE.

c) include=TRUE.

d) include=FALSE.

9. En R markdown, para especificar el formato de salida de un documento, ¿cómo puedes hacelo?

a) Mediante el campo output de la cabecera YAML del documento.

b) Procesándolo desde la consola usando:

```
rmarkdown::render("midocumento.Rmd",output_format = "pdf_document")
```

c) Mediante el menú desplegable que hay junto al botón Knit.

d) Todas las opciones anteriores son válidas.

10. Si en un documento R markdown escribo: El área de una circunferencia de radio `` `r 1+2` `` es `` `r pi*3^2` ``. En el documento final veremos:

a) El área de una circunferencia de radio 1+2 es 28.27433.

b) El área de una circunferencia de radio 3 es 28.2.

c) El área de una circunferencia de radio 3 es 28.27433.

d) El área de una circunferencia de radio 3 es 28.

SOLUCIONARIO

1.	b	**2.**	a	**3.**	c	**4.**	b	**5.**	a
6.	d	**7.**	c	**8.**	b	**9.**	d	**10.**	c

BIBLIOGRAFÍA

- R for Data Science, Garrett Grolemund, Hadley Wickham. O'Reilly (2016) Disponible online en http://r4ds.had.co.nz
- Authoring Books and Technical Documents with R Markdown, Yihui Xie: https://bookdown.org/yihui/bookdown/
- Documentación oficial de R markdown: http://rmarkdown.rstudio.com/
- Documentación oficial Knitr: https://yihui.name/knitr/
- R Markdown Cheat Sheet:

  https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf