Representaciones gráficas en R

En este documento veremos cómo podemos obtener representaciones gráficas de los datos usando la librería ***ggplot2***, disponible dentro del ecosistema tidyverse. Sería el equivalente a librerías como Matplotlib para Python.

1. Instalación y primeros pasos

Instalaremos el paquete o bien incorporando el conjunto *tidyverse*, o bien especificando la librería en sí. En ambos casos tendremos que usar la instrucción `install.packages`:

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

Una vez instalada, la incorporamos a los programas que la vayan a usar:

```
library(ggplot2)
```

1.1. Las capas de *ggplot*

Para construir un gráfico *ggplot2* organiza la información en capas:

- En una primera capa estarán los datos con los que hay que trabajar (típicamente un *data frame* o similar).
- En una segunda capa están los *aesthetics*, donde indicaremos qué parte de los datos queremos incluir en el gráfico, y asociaremos colores y otros elementos de estilo.
- La tercera capa hace referencia a la *geometría*, donde indicamos el tipo de gráfico que queremos construir (de barras, de puntos...).
- Por encima de estas tres capas básicas existen otras más complejas, aunque menos relevantes, como la posibilidad de hacer *facets* (varios gráficos dentro de un mismo gráfico), elegir temas de diseño, etc.

Ilustraremos los siguientes ejemplos con los datos de este *data frame*:

	Nombre	Edad	Email	Telefono	EstadoCivil	Pais
1	Juan Pérez	70	jperez@gmail.com	611223344	Casado	España
2	Ana Sánchez	41	asg@hotmail.com	655443322	Soltero	Argentina
3	Pedro Carrillo	56	p.carrillo@gmail.com	677889900	Casado	España
...	...	...	...	...	...	...

2. Construcción básica de un gráfico. Ejemplos sencillos.

La función `ggplot` se encarga de iniciar el proceso. Recibe normalmente como parámetros:

- El conjunto de datos (*data frame*) con que trabajar, en un parámetro `data`.
- Las estéticas (*aesthetics*) que se quieren utilizar, en un parámetro `mapping`.

Por ejemplo, si queremos mostrar en el eje X los nombres y en el eje Y las edades de las personas del *data frame* anterior, haríamos algo así:

```
ggplot(data=datos, mapping=aes(x=Nombre, y=Edad))
```

2.1. Tipos de gráficos habituales

La instrucción anterior no va a construir ningún gráfico. De hecho, si la ejecutamos veremos el área preparada, con los ejes definidos, pero nada más. Nos falta escoger la tercera capa, la geometría, a través de las funciones *geom* que incorpora *ggplot*, y que enlazamos con lo anterior mediante el operador `+`. Existen diferentes tipos de geometría. Algunas de las más habituales son:

- `geom_histogram`: representan histogramas de frecuencias para los valores de una determinada serie (o columna de datos en una tabla)
- `geom_point`: gráficos de puntos o de dispersión. Se puede combinar con `geom_smooth` para dibujar una línea de ajuste de regresión.
- `geom-smooth`: ajusta una línea (curva) mediante regresión lineal. Es habitual utilizarla junto a la geometría anterior
- `geom_col`: para gráficos de barras verticales (columnas)
- `geom_boxplot`: para gráficos de cajas que muestren la distribución de valores para una o varias variables categóricas

Por ejemplo, de esta forma construimos un gráfico de columnas que muestre cada nombre con su edad:

```
ggplot(data=datos, mapping=aes(x=Nombre, y=Edad)) + geom_col()
```

Y así construiríamos un histograma de edades:

```
ggplot(data=datos, mapping=aes(x=Edad)) + geom_histogram()
```

El parámetro *mapping* se puede pasar alternativamente a la función *geom*, en lugar de a *ggplot*:

```
ggplot(data=datos) + geom_histogram(mapping=aes(x=Edad))
```

De este otro modo dibujaríamos un gráfico de dispersión junto con su recta de regresión:

```
ggplot(data=datos, mapping=aes(x=..., y=...)) + geom_point() + geom_smooth(method=lm)
```

2.2. El caso de los gráficos circulares

Como particularidad, no existe una función específica para crear gráficos circulares (*pie charts*). Lo que se hace es crear un gráfico de barras con una sola barra, donde las Y sean los valores de cada categoría. Después se utiliza la función `coord_polar` para convertir el gráfico en circular. Aquí vemos un ejemplo que construye un gráfico circular con las proporciones de cada tipo de nota de un conjunto:

```
datos = data.frame(calificaciones = c('Suspensos', 'Aprobados', 'Notables',
  'Sobresalientes'), valores = c(5, 3, 8, 4))

ggplot(datos, aes(x='', y=valores, fill=calificaciones)) +
  geom_bar(stat='identity') + coord_polar("y", start=0)
```

3. Opciones adicionales

Existen algunas opciones adicionales que podemos añadir a los gráficos, como por ejemplo:

- Colores a través del parámetro `color` (línea o punto) o `fill` (relleno) en la función `aes`. Podemos especificar un color distinto por un campo (por ejemplo `color=Pais` para que pinte de colores diferentes los datos según la columna *Pais*), o bien especificar un color determinado (`color="blue"`).
- Formas a través del parámetro `shape`, también en la función `aes`
- Etiquetas en los ejes, con las funciones `xlab` e `ylab`
- Título general del gráfico con `ggtitle`
- Alternativamente podemos usar la función `labs` y pasarle como parámetro las etiquetas de todo (X, Y, general, etc)
- Diferentes subgráficos con `facet_grid`
- Diferentes temas con las funciones `theme_*`
- Algunas opciones adicionales dependientes de cada gráfico. Por ejemplo, en los histogramas tenemos un parámetro `bins` que indica cuántas divisiones queremos que tenga el histograma: `geom_histogram(bins=10)`.
- Si queremos reordenar los valores de un eje de categorías podemos emplear la función `reorder` en los *aesthetics*

Todo esto se puede ir enlazando y acumulando en una variable. Por ejemplo:

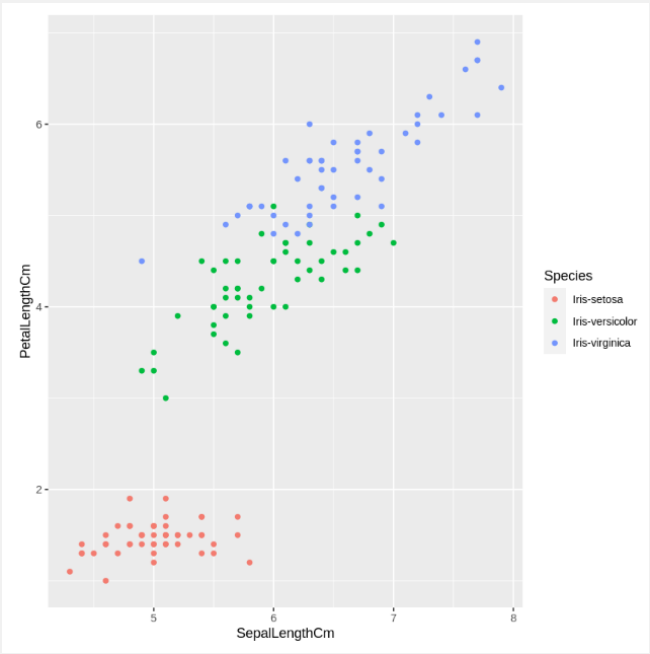
```
# Definimos datos y estética básicas y color basado en país
g <- ggplot(data=datos, mapping=aes(x=Nombre, y=Edad, fill=Pais))
# Enlazamos geometría (gráfico de columnas)
g <- g + geom_col()
# Rótulo de los ejes y título
g <- g + xlab("Nombres de personas")
g <- g + ylab("Edades de personas")
g <- g + ggtitle("Relación de nombres y edades")
# Alternativa para hacer lo mismo con los títulos y etiquetas
g <- g + labs(title = "Relación de nombres y edades",
    x = "Nombres de personas", y = "Edades de personas")
# Tema en blanco y negro (sin fondo gris)
g <- g + theme_bw()
# Reordenar datos en X (nombres) por la edad descendente
g <- ggplot(data=datos, mapping=aes(x=reorder(Nombre, -Edad), y=Edad, fill=Pais))
```

Adicionalmente podemos guardar el/los gráfico(s) que generemos con la función `ggsave`, indicando el nombre del archivo y el gráfico a guardar. Se admiten distintos formatos, como por ejemplo PDF, JPG o PNG.

```
ggsave("grafico.pdf", g)
```

Ejercicio 1:

Utiliza este CSV de clasificación de las diferentes especies de la flor *Iris* y muestra un gráfico de dispersión o puntos que muestre la relación entre la longitud del sépalo y la del pétalo, pintando con diferentes colores los puntos de las diferentes especies (columna *Species*).



AYUDA: vídeo con solución del ejercicio

Ejercicio 2:

Utiliza este CSV sobre cotizaciones de acciones y muestra un gráfico de barras con los valores máximos de las 5 acciones de mayor valor, ordenados de mayor a menor, y pintando las columnas de azul.