Graphs with ggplot

September 25, 2025

1 Introducción a la Programación para Ciencia de Datos

1.1 Lenguaje de programación R

Rocío Romero Zaliz - rocio@decsai.ugr.es

Gráficos en R * Base R * Paquete `ggplot2`

[ ]:
```
plot(mtcars[,c("mpg","wt","hp")])
```

[ ]:
```
hist(mtcars$mpg)
```

1.2 GGPLOT ("Grammar of Graphics")

- Plots:
  - datos y mapeados estéticos (aesthetic),
  - objetos geométricos,
  - escalas,
  - transformaciones estadísticas, y
  - un sistema de coordenadas
- Los plots pueden tener varias capas
- Los plots pueden ser asignados a variables

[ ]:
```
# Load library
library(ggplot2)
library(tidyverse)

# Example dataset
diamonds
```

[ ]:
```
help(diamonds)
```

[ ]:
```
diamonds %>% str
```

[ ]:
```
summary(diamonds)
```

[ ]:
```
# Start ggplot
# The aes() function is used to specify the X and Y axes
ggplot(data=diamonds, mapping=aes(x=carat, y=price))
```

[ ]:
```
?aes
```

- Ejes (x, y)
- Color (color, fill)
- Tamaño (size)
- Forma (shape)
- Transparencia (alpha)
- Grupo (group)
- …

1.3 Scatterplots

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price)) + geom_point()
```

[ ]:
```
plot(diamonds$carat, diamonds$price)
```

1.3.1 Export plots

- PDF
- PNG
- SVG
- JPEG
- BMP
- TIFF

[ ]:
```
# R base
pdf("grafico.pdf")
# R sentences
ggplot(diamonds, aes(x=carat, y=price)) + geom_point()
dev.off()
```

[ ]:
```
# GGplot
ggsave("otro_grafico.pdf")
```

[ ]:
```
?ggsave
```

1.3.2 Tuning plots

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price)) +
    geom_point() +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1")
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price)) +
    geom_point(color="red", size=1) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1")
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
    geom_point(size=2) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut")
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, color=cut, size=clarity)) +
    geom_point() +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is")
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, color=cut, size=clarity)) +
    geom_point() +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is") +
    scale_color_brewer(palette = "Blues")
```

[ ]:
```
library(RColorBrewer)
brewer.pal.info
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, color=cut, size=clarity)) +
    geom_point() +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5)) +
    scale_y_continuous(breaks=seq(0, 20000, 1000))
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, color=cut, size=clarity)) +
    geom_point() +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is") +
```

```
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000))
```

[ ]:
```
print(names(diamonds))
ggplot(diamonds, aes(x=carat, y=price, color=cut, size=clarity)) +
    geom_point() +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000)) +
    geom_text(aes(label=x))
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, size=clarity)) +
    geom_point(aes(color=cut)) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000)) +
    geom_text(aes(label=x))
```

[ ]:
```
ggplot(data=diamonds, aes(x=carat, y=price, size=clarity)) +
    geom_point(aes(color=cut)) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000)) +
    geom_text(data=subset(diamonds, price > 10000),
            aes(x=carat, y=price, label=price))
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, size=clarity)) +
    geom_point(aes(color=cut)) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000)) +
    geom_text(data=subset(diamonds, price > 10000),
            aes(x=carat, y=price, label=price)) +
    geom_hline(yintercept = 10000, colour = "blue", linetype="dotted")
```

[ ]:
```
print(summary(diamonds))
print(names(diamonds))
ggplot(diamonds, aes(x=carat, y=price, size=clarity, shape=color)) +
    geom_point(aes(color=cut)) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", color="Quality of the cut",␣
 ↪size="How clear is", shape="Color") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000))
```

[ ]:
```
sum(diamonds$color == 'J')
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, size=clarity, shape=cut)) +
    geom_point(aes(color=color)) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", shape="Quality of the cut",␣
 ↪size="How clear is", color="Color") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000))
```

[ ]:
```
ggplot(diamonds, aes(x=carat, y=price, size=clarity, shape=as.character(cut))) +
    geom_point(aes(color=color)) +
    labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the diamond",␣
 ↪y="Price in US dollars", caption="Fig. 1", shape="Quality of the cut",␣
 ↪size="How clear is", color="Color") +
    scale_colour_brewer(palette = "Oranges") +
    scale_x_continuous(breaks=seq(0, 5, 0.5), labels = letters[1:11]) +
    scale_y_continuous(breaks=seq(0, 20000, 1000))
```

1.4 Facetting

[ ]:
```
p <- ggplot(diamonds, aes(x=carat, y=price, size=clarity)) +
        geom_point(aes(color=cut)) +
        labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the␣
 ↪diamond", y="Price in US dollars", caption="Fig. 1", color="Quality of the␣
 ↪cut", size="How clear is", shape="Color") +
        scale_colour_brewer(palette = "Oranges")
p
```

[ ]:
```
?formula
```

[ ]:
```
p + facet_wrap(~ cut)
```

[ ]:
```
p + facet_wrap(~ color, nrow=3)
```

[ ]:
```
# Dangerous!!!! Avoid!!!!
p + facet_wrap(~ cut, nrow=3, scales = "free")
```

[ ]:
```
p + facet_wrap(~ cut + clarity)
```

También existe el `facet_grid`

1.5 Histograms

[ ]:
```
q <- ggplot(diamonds, aes(x=carat)) +
        labs(title="Diamonds", subtitle="Pay the price!", x="Weight of the␣
 ↪diamond ", caption="Fig. 1") +
        scale_colour_brewer(palette = "Oranges")
q
```

[ ]:
```
q + geom_histogram()
```

[ ]:
```
q + geom_histogram(bins = 10)
```

[ ]:
```
q + geom_histogram(binwidth = 2)
```

[ ]:
```
q + geom_histogram(color="red", fill="blue", bins = 15)
```

1.6 Boxplots

[ ]:
```
summary(diamonds)
```

[ ]:
```
ggplot(diamonds, aes(x=carat)) +
    geom_boxplot()
```

[ ]:
```
ggplot(diamonds, aes(x=carat)) +
    geom_boxplot() + coord_flip()
```

[ ]:
```
diamonds %>% head(5)
```

[ ]:
```
ndiamonds <- diamonds %>% pivot_longer(cols=c(x, y, z))
ndiamonds %>% head(6)
```

El contrario a `pivot_longer` es `pivot_wider`

[ ]:
```
ggplot(ndiamonds, aes(x=name, y=value)) +
    geom_boxplot()
```

[ ]:
```
ggplot(ndiamonds, aes(x=name, y=value)) +
    geom_boxplot(aes(fill=name))
```

[ ]:
```
ggplot(ndiamonds, aes(x=name, y=value)) +
    geom_boxplot(aes(fill=name)) +
    facet_wrap(~ cut)
```

[ ]:
```
ggplot(ndiamonds, aes(x=name, y=value)) +
    geom_boxplot(aes(fill=name)) +
    facet_wrap(~ cut + color)
```

[ ]:
```
ggplot(ndiamonds, aes(x=name, y=value)) +
    geom_boxplot(aes(fill=name), outlier.shape = NA) +
    facet_wrap(~ cut)
```

[ ]:
```
ggplot(ndiamonds, aes(x=name, y=value)) +
    geom_boxplot(aes(fill=name), outlier.shape = NA) +
    facet_wrap( ~ cut) +
    scale_y_continuous(limits = c(0, 10))
```

[ ]:
```
# Alternativa...
ggplot(ndiamonds, aes(x=name, y=value)) +
    geom_boxplot(aes(fill=name), outlier.shape = NA) +
    facet_wrap( ~ cut) +
    coord_cartesian(ylim=c(0,10))
```

1.7 Line plots

[ ]:
```
# It does not make any sense...
ggplot(diamonds, aes(x=carat, y=price)) + geom_line()
```

[ ]:
```
# Quiero visualizar el precio maximo y minimo por cada carat
ndiamonds <- diamonds %>% group_by(carat) %>%
    summarize(max_price_carat=max(price), min_price_carat=min(price))
ndiamonds %>% head(5)
```

[ ]:
```
ggplot(ndiamonds, aes(x=carat, y=max_price_carat)) + geom_line()
```

[ ]:
```
ggplot(ndiamonds, aes(x=carat, y=min_price_carat)) + geom_line()
```

[ ]:
```
ggplot(ndiamonds, aes(x=carat)) +
    geom_line(aes(y=max_price_carat)) +
    geom_line(aes(y=min_price_carat))
```

[ ]:
```
ggplot(ndiamonds, aes(x=carat, color="red")) +
    geom_line(aes(y=max_price_carat)) +
    geom_line(aes(y=min_price_carat))
```

[ ]:
```
ggplot(ndiamonds, aes(x=carat)) +
    geom_line(aes(y=max_price_carat), color="red") +
    geom_line(aes(y=min_price_carat), color="blue")
```

[ ]:
```
ggplot(ndiamonds, aes(x=carat)) +
    geom_line(aes(y=max_price_carat), color="red", linetype="dotted") +
    geom_line(aes(y=min_price_carat), color="blue", linetype="dashed")
```

[ ]:
```
ndiamonds <- ndiamonds %>% pivot_longer(cols=c(max_price_carat,␣
 ↪min_price_carat))
ndiamonds %>% head(5)
```

[ ]:
```
ggplot(ndiamonds, aes(x=carat, y=value, linetype=name, color=name)) +
    geom_line()
```

[ ]:
```
# Datos de edad y altura sobre personas...
nlme::Oxboys %>% head(15)
```

[ ]:
```
ggplot(nlme::Oxboys, aes(age, height)) + geom_line()
```

[ ]:
```
ggplot(nlme::Oxboys, aes(age, height, group=Subject)) + geom_line()
```

[ ]:
```
ggplot(nlme::Oxboys, aes(age, height, group=Subject, linetype=Subject)) +␣
 ↪geom_line()
```

[ ]:
```
ggplot(nlme::Oxboys, aes(age, height, group=Subject, color=Subject)) +␣
 ↪geom_line() + geom_point()
```

1.8 Bar plots

[ ]:
```
ggplot(diamonds, aes(x=cut)) + geom_bar() + coord_flip()
```

[ ]:
```
diamonds %>% count(cut)
```

[ ]:
```
diamonds %>% count(cut, color)
```

[ ]:
```
ggplot(diamonds, aes(x=cut, fill=color)) + geom_bar()
```

[ ]:
```
ggplot(diamonds, aes(x=cut, fill=color)) + geom_bar(position = "dodge")
```

```
[ ]: ggplot(diamonds, aes(x=cut, y=price)) + geom_bar(stat="identity") # Warning!␣
      ↪Summing up prices!
     ggplot(diamonds, aes(x=cut, y=price)) + geom_col() # Same!
```

```
[ ]: diamonds %>% select(cut, price) %>% add_count(cut) %>% group_by(cut, n) %>%
         summarise(sum_price = sum(price))
```

```
[ ]: ndiamonds <- diamonds %>% group_by(cut) %>% mutate(max_price=max(price))
     print(ndiamonds)

     ggplot(ndiamonds, aes(x=cut, y=max_price)) + geom_col()
```

```
[ ]: ggplot(ndiamonds, aes(x=cut, y=max_price)) + geom_point()
```

```
[ ]: ndiamonds %>% select(cut, max_price) %>% unique
```

```
[ ]: ggplot(ndiamonds %>% select(cut, max_price) %>% unique,
            aes(x=cut, y=max_price)) + geom_col()
```

```
[ ]: ggplot(ndiamonds %>% select(cut, color, max_price) %>% unique(),
            aes(x=cut, y=max_price, fill=color)) +
            geom_bar(stat="identity", color="black")
```

2 Customizing plots

- Use of "themes"

theme(line, rect, text, title, aspect.ratio, axis.title, axis.title.x, axis.title.x.top, axis.title.x.bottom, axis.title.y, axis.title.y.left, axis.title.y.right, axis.text, axis.text.x, axis.text.x.top, axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right, axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.length, axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y, axis.line.y.left, axis.line.y.right, legend.background, legend.margin, legend.spacing, legend.spacing.x, legend.spacing.y, legend.key, legend.key.size, legend.key.height, legend.key.width, legend.text, legend.text.align, legend.title, legend.title.align, legend.position, legend.direction, legend.justification, legend.box, legend.box.just, legend.box.margin, legend.box.background, legend.box.spacing, panel.background, panel.border, panel.spacing, panel.spacing.x, panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor, panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x, panel.grid.minor.y, panel.ontop, plot.background, plot.title, plot.subtitle, plot.caption, plot.tag, plot.tag.position, plot.margin, strip.background, strip.background.x, strip.background.y, strip.placement, strip.text, strip.text.x, strip.text.y,

strip.switch.pad.grid, strip.switch.pad.wrap, ..., complete = FALSE, validate = TRUE)

[ ]:
```
w <- ggplot(diamonds, aes(x=carat, y=color, fill=cut)) +
       coord_flip() +
       geom_bar(stat="identity")
w
```

[ ]:
```
w + theme(legend.position="top") # "none", "left", "top", "bottom", "right" or␣
↪specific location using c()
```

More on legends: http://www.sthda.com/english/wiki/ggplot2-legend-easy-steps-to-change-the-position-and-the-appearance-of-a-graph-legend-in-r-software

[ ]:
```
w + theme_dark()
```

[ ]:
```
w + theme_light()
```

[ ]:
```
w + theme(
      panel.background = element_rect(fill = "pink",
                                      colour = "lightblue",
                                      linewidth = 0.5, linetype = "solid"),
      panel.grid.major = element_line(linewidth = 0.1, linetype = 'dashed',
                                      colour = "blue"),
      panel.grid.minor = element_blank(),
      plot.background = element_rect(fill = "green"),

      text = element_text(size=10, family="Courier"),
      axis.text.x = element_text(angle=30)
  )
```

More on themes: http://www.sthda.com/english/wiki/ggplot2-themes-and-background-colors-the-3-elements

2.1 Pie charts

[ ]:
```
ggplot(diamonds, aes(x=1, fill=cut)) + geom_bar(width = 1)
```

[ ]:
```
ggplot(diamonds, aes(x=1, fill=cut)) + geom_bar(width = 1) +
    coord_polar(theta="y", start=0)
```

[ ]:
```
ggplot(diamonds, aes(x=1, fill=cut)) + geom_bar(width = 1) +
    scale_fill_manual(values=c("#FFFFFF", "#AAAAAA", "#BBBBBB", "#CCCCCC",␣
↪"#DDDDDD"))
```

[ ]:
```
ggplot(diamonds, aes(x=1, fill=cut)) + geom_bar(width = 1) +
```

```
    scale_fill_manual(values=c("#FFFFFF", "#AAAAAA", "#BBBBBB", "#CCCCCC",␣
 ↪"#DDDDDD")) +
    coord_polar(theta="y", start=0)
```

[ ]:
```
ggplot(diamonds, aes(x=1, fill=cut)) + geom_bar(width = 1) +
    coord_polar(theta="x", start=0)
```

[ ]:
```
# Coxcomb plot
ggplot(diamonds, aes(x=1, y=clarity, fill=clarity)) + geom_bar(stat =␣
 ↪"identity") +
    coord_polar(theta="y")
```

2.2 Heatmap

[ ]:
```
ggplot(diamonds, aes(x = carat, y = color, fill = cut)) + geom_tile()
```

[ ]:
```
ggplot(diamonds, aes(x = carat, y = color, fill = price)) + geom_tile() +
    scale_fill_gradient(low = "white", high = "darkred")
```

[ ]:
```
diamonds %>% ggplot(aes(x = carat, y = color, fill = price)) + geom_tile() +
    scale_fill_gradient(low = "white", high = "darkred")
```

2.3 Ejercicios

1) Realiza un scatterplot que muestre la variable distance en función del stretch en el siguiente conjunto de datos:
2) Los siguientes datos tienen diez observaciones tomadas durante los años 1970-79, sobre la cubierta de nieve de octubre para Eurasia (la cubierta de nieve está en millones de kilómetros cuadrados).
   - Grafica snow.cover versus year (para series de tiempo se recomienda usar lineas y no solo puntos)
   - Grafica un histograma de los valores de snow.cover
3) Dados los siguientes datos:

- Transforma las columnas de temperatura de ºF a ºC
- Transforma las columnas de pulgadas a mm
- Grafica el año frente a la temperatura mínima más cálida.
- Grafica el año frente a la temperatura mínima más cálida y la temperatura mínima más fría. No olvides poner una leyenda.

4) Utilizando el conjunto de datos `starwars` muestra:
   - la diversidad de especies en los diferentes mundos
   - la altura y masa de cada personaje, ¿es la misma en ambos sexos?
5) Utilizando el conjunto de datos `mtcars` muestra la distribución de millas por galón, caballos brutos y peso para cada número de cilindros.

2.3.1 Ejercicio colectivo

Dado el dataset de `starwars` mostrar (a definir en clase)

2.3.2 Bibliografía & Referencias

- The Layered Grammar of Graphics. Hadley Wickham. Journal of Computational and Graphical Statistics, Volume 19, Number 1, Pages 3–28 DOI: 10.1198/jcgs.2009.07098.
- R graphics. Paul Murrell. Computer Science and Data Analysis Series. Chapman & Hall/CRC. 2006.
- http://www.r-bloggers.com
- http://www.statmethods.net
- http://www.cookbook-r.com/