R 2019

20/03/2019

Repaso de la clase pasada

Conceptos principales

- Abrimos una sesión de R en una consola
- R usa comandos para ejecutar acciones
- Los comandos se distinguen por sus () finales
- Se escriben en la línea de comandos (que empieza con el *prompt*) y se ejecutan con *Enter*

Conceptos principales

Paquetes contienen **funciones** (aka, comandos)

```
install.packages("ggplot2") # instalo el paquete de Internet, si
library(ggplot2) # cargo el paquete ggplot2 en mi sesión
```

Funciones actuan sobre **datos y variables**

```
str(iris) # str() me describe la estructura de mis_datos
summary(iris) # summary() los resume estadísticamente
head(iris) # head() me muestra algunas primeras líneas
plot(iris$Sepal.Length, iris$Sepal.Width) # plot() los grafica
```

Visualización I

`base::plot()`

```r
# Defino un vector de 5 valores
cars <- c(1, 3, 6, 4, 9)

# Grafico los 5 valores en función de su posición en el vector
plot(cars)

# Idem, pero agrego una línea y lo coloreo de azul
plot(cars, type="o", col="blue")

# Agrego un título en rojo con cierto tamaño de letra
title(main="Autos", col.main="red", font.main=4)
```

Visualización II

`base::plot()`

```r
# Genero 1000 valores aleatorios de una distribución lognormal (?
r <- rlnorm(1000)

# Guardo en h un histograma sin dibujarlo, y con bins menores (?s
h <- hist(r, plot=F, breaks=c(seq(0,max(r)+1, .1)))

# Dibujo el histograma con ejes logarítmicos y con puntos azules
plot(h$counts, log="xy", pch=20, col="blue",
        main="Log-normal distribution",
        xlab="Value", ylab="Frequency")
```

Sobre las prácticas

ggplot2

qplot

ggplot2::qplot()

```
## "scatterplot" de displ vs. hwy del data.frame mpg
install.packages("ggplot2")
library(ggplot2)
qplot(displ, hwy, data = mpg)

## idem pero usando una escala de colores para "class"
qplot(displ, hwy, colour = class, data = mpg)
```

1. similar a plot()
2. para hacer figuras básicas está ok
3. para gráficas más elaboradas vamos a usar ggplot2()

ggplot2

```
p <- ggplot(mtcars)   # creo un objeto ggplot con los datos mtcar
p <- ggplot(mtcars) + aes(mpg, wt) # le agrego el mapeo de variab
p <- ggplot(mtcars) + aes(mpg, wt) + geom_point() # le agrego com
p                     # imprimo la figura
```

- *grammar of graphics* (Wilkinson, 2005)
- se trata de construir capas de código, cada una agregando un concepto a la figura final
- cada capa controla un aspecto independiente de la figura
- la figura *es* el código (se puede guardar, extender, reproducir, etc.)

En general:

```
ggplot(data = {DATA}) +
      {GEOM_FUNCTION}(mapping = aes( {MAPPINGS} ))
```

¡Cuidado! `ggplot2` trabaja solo con `data.frames`, al igual que el resto del tidyverse (dplyr, tidyr, etc.).

Más en ggplot2-cheatsheet-2.1.pdf (ver Google Classroom/Classwork/Class Drive Folder/R cheatseets).

¿Y qué vendría siendo un data.frame?

- un data.frame es una de las estructuras más comunes para manejar datos en R
- es una lista de vectores de igual dimensión

```
df <- data.frame() # creo un data.frame llamado df
is.data.frame(df)  # ¿es un data.frame?
[1] TRUE
is.list(df)        # ¿es una lista?
[1] TRUE
```

- tiene propiedades de data.frames y de listas
- ¡A practicar! Busquen un data.frame con data() y prueben las funciones `length()` y `nrow()`

Aesthetics y Geoms

```
data(economics)
?economics
str(economics)
ggplot(economics, aes(x = pce)) + geom_histogram()
ggplot(economics, aes(x = date, y = unemploy)) + geom_line()
ggplot(economics, aes(x = date, y = unemploy, colour = pce)) + ge
```

- aesthetics, `aes()`, controla mapeos entre variables y elementos visuales
- ejemplo: variable A <-> coordenada x, o variable C <-> forma del punto
- existen muchos tipos de `geoms`

Facets y *scales*

```
library(ggplot2)
ggplot(midwest, aes(x = popwhite, y = percadultpoverty, colour =
        geom_line() +
        facet_wrap(~ state) +
        scale_x_log10()
```

- *Facets* se usan para dividir la figura en varias, filtrando mediante alguna variable categórica
- *scales* permiten modificar el tipo de ejes de la figura

Glosario `ggplot2`

- **`data`**: el *dataframe* que contiene los datos a graficar
- **`geoms`**: el tipo de objeto geométrico que representa los datos: puntos, líneas, polígonos, etc.
- **`aesthetics`**: describe las características visuales que representan los datos, por ejemplo, posición, tamaño, color, forma, etc
- **`scale`**: para cada *aesthetic*, describe como se mapea la característica visual a valores por ejemplo, escala logarítmica, escala de color, de tamaño, de forma, etc.
- **`stats`**: describe transformaciones estadísticas que resumen los datos, e.g. una regresión

Práctica

Descargar práctica 2.