Comparación de sintaxis de R : : HOJA DE REFERENCIA

Sintaxis de signo peso

`objetivo(data$x, data$y)`

ESTADÍSTICOS DE RESUMEN:

una variable continua:
```
mean(mtcars$mpg)
```

una variable categórica:
```
table(mtcars$cyl)
```

dos variables categóricas:
```
table(mtcars$cyl, mtcars$am)
```

una continua, una categórica:
```
mean(mtcars$mpg[mtcars$cyl==4])
mean(mtcars$mpg[mtcars$cyl==6])
mean(mtcars$mpg[mtcars$cyl==8])
```

GRAFICAR:

una variable continua:
```
hist(mtcars$disp)
boxplot(mtcars$disp)
```

una variable categórica:
```
barplot(table(mtcars$cyl))
```

dos variables continuas:
```
plot(mtcars$disp, mtcars$mpg)
```

dos variables categóricas:
```
mosaicplot(table(mtcars$am, mtcars$cyl))
```

una continua, una categórica:
```
histogram(mtcars$disp[mtcars$cyl==4])
histogram(mtcars$disp[mtcars$cyl==6])
histogram(mtcars$disp[mtcars$cyl==8])

boxplot(mtcars$disp[mtcars$cyl==4])
boxplot(mtcars$disp[mtcars$cyl==6])
boxplot(mtcars$disp[mtcars$cyl==8])
```

MANEJO DE DATOS:

crear subconjuntos:
```
mtcars[mtcars$mpg>30, ]
```

crear una nueva variable:
```
mtcars$efficient[mtcars$mpg>30] <- TRUE
mtcars$efficient[mtcars$mpg<30] <- FALSE
```

Sintaxis de fórmula

`objetivo(y~x|z, data=data, group=w)`

ESTADÍSTICOS DE RESUMEN:

una variable continua:
```
mosaic::mean(~mpg, data=mtcars)
```

una variable categórica:
```
mosaic::tally(~cyl, data=mtcars)
```

dos variables categóricas:
```
mosaic::tally(cyl~am, data=mtcars)
```

una continua, una categórica:
```
mosaic::mean(mpg~cyl, data=mtcars)
```

virgulilla

GRAFICAR:

una variable continua:
```
lattice::histogram(~disp, data=mtcars)
lattice::bwplot(~disp, data=mtcars)
```

una variable categórica:
```
mosaic::bargraph(~cyl, data=mtcars)
```

dos variables continuas:
```
lattice::xyplot(mpg~disp, data=mtcars)
```

dos variables categóricas:
```
mosaic::bargraph(~am, data=mtcars, group=cyl)
```

una continua, una categórica:
```
lattice::histogram(~disp|cyl, data=mtcars)

lattice::bwplot(cyl~disp, data=mtcars)
```

Sintaxis del Tidyverse

`data %>% objetivo(x)`

ESTADÍSTICOS DE RESUMEN:

una variable continua:
```
mtcars %>% dplyr::summarize(mean(mpg))
```

una variable categórica:
```
mtcars %>% dplyr::group_by(cyl) %>%
dplyr::summarize(n())
```

el *pipe*

dos variables categóricas:
```
mtcars %>% dplyr::group_by(cyl, am) %>%
dplyr::summarize(n())
```

una continua, una categórica:
```
mtcars %>%  dplyr::group_by(cyl) %>%
  dplyr::summarize(mean(mpg))
```

GRAFICAR:

una variable continua:
```
ggplot2::qplot(x=mpg, data=mtcars, geom="histogram")
ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")
```

una variable categórica:
```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```

dos variables continuas:
```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```

dos variables categóricas:
```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +
       facet_grid(.~am)
```

una continua, una categórica:
```
ggplot2::qplot(x=disp, data=mtcars, geom="histogram") +
       facet_grid(.~cyl)

ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
               geom="boxplot")
```

MANEJO DE DATOS:

crear subconjuntos:
```
mtcars %>% dplyr::filter(mpg>30)
```

crear una nueva variable:
```
mtcars <- mtcars %>%
dplyr::mutate(efficient = if_else(mpg>30, TRUE, FALSE))
```

La variedad de sintaxis de R te permite "decir" lo mismo de diferentes modos

lee **a lo ancho** de esta hoja para ver cómo las diferentes sintaxis se aproximan al mismo problema

Comparación de sintaxis de R : : HOJA DE REFERENCIA

La **sintaxis** es el conjunto de reglas que definen qué código funciona y qué código no funciona en un lenguaje de programación. Si bien la mayoría de los lenguajes de programación ofrecen una sola sintaxis estandarizada, R permite a sus desarrolladores especificar la suya propia. Como resultado, existe una gran variedad de sintaxis en R (igualmente válidas).

Las tres sintaxis de R más predominantes son:

1. La **sintaxis de signo peso,** a veces llamada **sintaxis de R base**, es la esperada por la mayoría de las funciones de R base. Se caracteriza por el uso de `dataset$nombrevariable` y se asocia a la creación de subconjuntos con corchetes, como en `dataset[1,2]`. Casi todas las funciones de R aceptarán elementos que se entreguen con esta sintaxis.
2. La **sintaxis de fórmula**, es la usada para funciones de modelado como `lm()`, gráficos de `lattice` y estadísticos de resumen de `mosaic`. Usa una virgulilla (~) para conectar una variable de respuesta y uno (o más) predictores. Muchas funciones de R base aceptan este tipo de sintaxis.
3. La **sintaxis del Tidyverse** es la usada por `dplyr`, `tidyr` y muchos otros paquetes. Estas funciones esperan que los datos sean el primer argumento, lo que permite trabajar con el "pipe" (%>%) del paquete `magrittr`. Aunque `ggplot2` se considera parte del Tidyverse, tiene su propia variedad de sintaxis, que usa signos de suma (+) para encadenar las distintas partes del código. Hadley Wickham, autor de `ggplot2`, ha dicho que el paquete tendría una sintaxis distinta si lo hubiera desarrollado después de conocer el *pipe*.

Si bien usualmente se intenta enseñar R en el marco de un tipo de sintaxis, la mayoría de las personas programa usando una combinación de ellas.

Tips para buscar en la web:

Si al buscar en Google, StackOverflow u otro recurso en línea de tu preferencia te encuentras con código escrito en una sintaxis que no reconoces:

- chequea si el código utiliza alguna de las tres sintaxis comunes listadas en esta hoja de referencia.
- intenta buscar de nuevo agregando alguna palabra clave del nombre de la sintaxis ("tidyverse") o de un paquete relevante ("mosaic").

❗ Hay ocasiones en que una determinada sintaxis puede funcionar, pero se considera peligroso su uso porque es muy fácil producir errores. Por ejemplo, pasar nombres de variables sin asignarlas a un argumento con nombre.

Aun **más** formas de decir lo mismo

Incluso dentro de una misma sintaxis suelen haber variaciones que son igualmente válidas. Como caso de estudio, miremos la sintaxis de `ggplot2`. `ggplot2` es el paquete para graficar incluido en el Tidyverse. Si lees esta columna hacia abajo, todo el código produce el mismo gráfico.

lee hacia abajo esta columna para ver bloques de código en una sintaxis que parece diferente, pero produce el mismo gráfico

quickplot

`qplot()` hace referencia a *quickplot*, y te permite hacer gráficos rápidos. No tiene todo el poder de `ggplot2` y usa una sintaxis levemente diferente al resto del paquete.

```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```

```
ggplot2::qplot(x=disp, y=mpg, data=mtcars)
```
❗

```
ggplot2::qplot(disp, mpg, data=mtcars)
```
❗❗

ggplot

Para liberar todo el poder de `ggplot2`, necesitas usar la función `ggplot()` (que prepara una región para graficar) y agregar `geoms` (parámetros geométricos) al gráfico.

```
ggplot2::ggplot(mtcars) +
    geom_point(aes(x=disp, y=mpg))
```

```
ggplot2::ggplot(data=mtcars) +
    geom_point(mapping=aes(x=disp, y=mpg))
```

el '+' agrega capas

```
ggplot2::ggplot(mtcars, aes(x=disp, y=mpg)) +
    geom_point()
```

```
ggplot2::ggplot(mtcars, aes(x=disp)) +
    geom_point(aes(y=mpg))
```

ggformula

La "tercera forma y media" es usar la sintaxis de fórmula, pero para obtener gráficos tipo ggplot2

```
ggformula::gf_point(mpg~disp, data= mtcars)
```

fórmulas en gráficos de R base

Los gráficos de R base también aceptan la sintaxis de fórmula, aunque no es un uso habitual

```
plot(mpg~disp, data=mtcars)
```