Hoja de Estilo del lenguaje R

Convenciones de estilo R

R es bastante indulgente con la forma en que escribimos código (a diferencia de otros lenguajes como Python, donde contar mal los espacios puede arruinar el trabajo).

Todas estas cosas harán exactamente lo mismo:

```
mpg |>
  filter(cty > 10, class == "compact")

mpg |> filter(cty > 10, class == "compact")

mpg |>
  filter(cty > 10,
         class == "compact")

mpg |> filter(cty>10, class=="compact")

filter(mpg,cty>10,class=="compact")

mpg |>
filter(cty > 10,
                         class == "compact")

filter ( mpg,cty>10,     class=="compact" )
```

Pero es evidente que sólo algunas de esas porciones de código (las primeras tres) son más fáciles de leer.

Para ayudar a mejorar la legibilidad y facilitar el compartir código con otros, existe una guía de estilo, publicada por el equipo de Tidyverse, para escribir código R. Es bastante breve y tiene muchos ejemplos de buenas y malas formas de escribir código (nombrar variables, manejar líneas largas, usar niveles de sangría adecuados, etc.).

Su enlace es https://style.tidyverse.org/

RStudio tiene una forma integrada de reidentar el código. Si seleccionamos la porción que deseamos y presionamos el atajo de teclado **Ctrl + i** (en Windows) R reformateará el código por nosotros. No siempre es perfecto, pero es realmente útil para lograr la sangría correcta sin tener que presionar manualmente espacio muchas veces.

Elementos más sobresalientes

Espaciado

Coloque espacios después de las comas

```
# Bien
filter(mpg, cty > 10)

# Mal
filter(mpg , cty > 10)
filter(mpg ,cty > 10)
filter(mpg,cty > 10)
```

Coloque espacios alrededor de operadores como +, -, >, =, etc.:

```
# Bien
filter(mpg, cty > 10)

# Mal
filter(mpg, cty>10)
filter(mpg, cty> 10)
filter(mpg, cty >10)
```

No coloque espacios alrededor de paréntesis que sean parte de funciones:

```
# Bien
filter(mpg, cty > 10)

# Mal
filter (mpg, cty > 10)
filter ( mpg, cty > 10)
filter( mpg, cty > 10 )
```

Líneas largas

En general, es una buena práctica no tener líneas de código muy largas. Una buena sugerencia es mantener las líneas con un máximo de 80 caracteres. En lugar de contar caracteres a mano, en RStudio vaya a **Tools** > **Global Options** > **Code** > **Display** y seleccione la casilla **Show margin**. Ahora deberías ver una línea muy delgada que indica 80 caracteres.

Conviene agregar saltos de línea dentro de líneas de código más largas. Los saltos de línea deben ir después de las comas y cosas como los argumentos de la función deben alinearse dentro de la función:

```
# Bien
filter(mpg, cty > 10, class == "compact")
```

```r
# Bien
filter(mpg, cty > 10,
       class == "compact")

# Bien
filter(mpg,
       cty > 10,
       class == "compact")

# Mal
filter(mpg, cty > 10, class %in% c("compact", "pickup", "midsize", "subcompact",
"suv", "2seater", "minivan"))

# Bien
filter(mpg,
       cty > 10,
       class %in% c("compact", "pickup", "midsize", "subcompact",
                    "suv", "2seater", "minivan"))
```

Tuberías (%>% o |>) y capas en ggplot2 (+)

Coloque cada capa de un gráfico ggplot en líneas separadas, con el + al final de la línea y debajo sangría de dos espacios:

```r
# Bien
ggplot(mpg, aes(x = cty, y = hwy, color = class)) +
  geom_point() +
  geom_smooth() +
  theme_bw()

# Mal
ggplot(mpg, aes(x = cty, y = hwy, color = class)) +
  geom_point() + geom_smooth() +
  theme_bw()

# Muy mal
ggplot(mpg, aes(x = cty, y = hwy, color = class)) + geom_point() + geom_smooth()
+ theme_bw()

# Tan mal que no funciona
ggplot(mpg, aes(x = cty, y = hwy, color = class))
  + geom_point()
  + geom_smooth()
  + theme_bw()
```

Coloque cada paso de una tubería en líneas separadas, con el %>% o |> al final de la línea y debajo una sangría de dos espacios:

```r
# Bien
mpg |>
  filter(cty > 10) |>
  group_by(class) |>
  summarize(avg_hwy = mean(hwy))

# Mal
mpg |> filter(cty > 10) |> group_by(class) |>
  summarize(avg_hwy = mean(hwy))

# Muy mal
mpg |> filter(cty > 10) |> group_by(class) |> summarize(avg_hwy = mean(hwy))

# Tan mal que no funciona
mpg |>
  filter(cty > 10)
  |> group_by(class)
  |> summarize(avg_hwy = mean(hwy))
```

Comentarios

Los comentarios deben comenzar con un símbolo de comentario y un solo espacio: #

```r
# Bien

#Mal

    #Mal
```

Si el comentario es muy corto (y no provoca que se supere los 80 caracteres en la línea), se puede incluir en la misma línea del código, separado por al menos dos espacios (funciona con un espacio, pero usando un par puede mejorar la legibilidad):

```r
mpg %>%
  filter(cty > 10) %>%  # filtro filas donde cty es 10 o más
  group_by(class) %>%  # estratifica por class
  summarize(avg_hwy = mean(hwy))  # resume la media de hwy por cada grupo
```

Se puede agregar espacios adicionales para alinear los comentarios en línea, si lo deseamos:

```r
mpg %>%
  filter(cty > 10) %>%           # filtro filas donde cty es 10 o más
  group_by(class) %>%            # estratifica por class
  summarize(avg_hwy = mean(hwy))  # resume la media de hwy por cada grupo
```

Si el comentario es muy largo, podemos dividirlo en varias líneas. RStudio tiene una herramienta en **Code** > **Reflow Comment** que lo automatiza.