Presentacion ...

Manuel Hernández

19 de mayo de 2021

Contenido de la sección 1

```r
library(tidyverse)
library(datos)
```

Los 'objetos' (funciones, tablas de datos y operadores) que contiene el paquete `stringr` las podemos consultar mediante:

```
ls("package:stringr")
```

```
 [1] "%>%"             "boundary"        "coll"            "fixed"
 [5] "fruit"           "invert_match"    "regex"           "sentences"
 [9] "str_c"           "str_conv"        "str_count"       "str_detect"
[13] "str_dup"         "str_ends"        "str_extract"     "str_extract_all"
[17] "str_flatten"     "str_glue"        "str_glue_data"   "str_interp"
[21] "str_length"      "str_locate"      "str_locate_all"  "str_match"
[25] "str_match_all"   "str_order"       "str_pad"         "str_remove"
[29] "str_remove_all"  "str_replace"     "str_replace_all" "str_replace_na"
[33] "str_sort"        "str_split"       "str_split_fixed" "str_squish"
[37] "str_starts"      "str_sub"         "str_sub<-"       "str_subset"
[41] "str_to_lower"    "str_to_sentence" "str_to_title"    "str_to_upper"
[45] "str_trim"        "str_trunc"       "str_view"        "str_view_all"
[49] "str_which"       "str_wrap"        "word"            "words"
```

Los 'objetos' (funciones, tablas de datos y operadores) que contiene el paquete `stringr` las podemos consultar mediante:

```
ls("package:stringr")
```

```
 [1] "%>%"             "boundary"        "coll"            "fixed"
 [5] "fruit"           "invert_match"    "regex"           "sentences"
 [9] "str_c"           "str_conv"        "str_count"       "str_detect"
[13] "str_dup"         "str_ends"        "str_extract"     "str_extract_all"
[17] "str_flatten"     "str_glue"        "str_glue_data"   "str_interp"
[21] "str_length"      "str_locate"      "str_locate_all"  "str_match"
[25] "str_match_all"   "str_order"       "str_pad"         "str_remove"
[29] "str_remove_all"  "str_replace"     "str_replace_all" "str_replace_na"
[33] "str_sort"        "str_split"       "str_split_fixed" "str_squish"
[37] "str_starts"      "str_sub"         "str_sub<-"       "str_subset"
[41] "str_to_lower"    "str_to_sentence" "str_to_title"    "str_to_upper"
[45] "str_trim"        "str_trunc"       "str_view"        "str_view_all"
[49] "str_which"       "str_wrap"        "word"            "words"
```

Opa, ¿y esto?

```
ls("package:stringr")[38:39]
```

```
[1] "str_sub"   "str_sub<-"
```

```
string1 <- "Esta es una cadena de caracteres"
string2 <- 'Si quiero incluir "comillas" dentro de la cadena, uso comillas simples'
```

```
"que pasa si "no" hacemos eso"

Error: <text>:1:15: unexpected symbol
1: "que pasa si "no
                  ^
```

A ver esto:

```
string3 <- "Si quiero incluir 'comillas' dentro de la cadena, uso comillas simples"
```

```r
string1 <- "Esta es una cadena de caracteres"
string2 <- 'Si quiero incluir "comillas" dentro de la cadena, uso comillas simples'
```

```r
"que pasa si "no" hacemos eso"
```

```
Error: <text>:1:15: unexpected symbol
1: "que pasa si "no
                  ^
```

A ver esto:

```r
string3 <- "Si quiero incluir 'comillas' dentro de la cadena, uso comillas simples"
```

```r
sapply(ls(), get)
```

```
                                                                 string1
                                        "Esta es una cadena de caracteres"
                                                                 string2
"Si quiero incluir \"comillas\" dentro de la cadena, uso comillas simples"
                                                                 string3
  "Si quiero incluir 'comillas' dentro de la cadena, uso comillas simples"
```

```r
sapply(ls(), get) %>% lapply(writeLines) %>% invisible
```

```
Esta es una cadena de caracteres
Si quiero incluir "comillas" dentro de la cadena, uso comillas simples
Si quiero incluir 'comillas' dentro de la cadena, uso comillas simples
```

4 elementos básicas con cadenas:

- Averiguar el largo: `str_length()`
- Combinar: : `str_c()`
- Dividir (¿extraer?, *subsetting*) : `str_sub()`
- Locales (¿?): `str_XXX(..., locale = )`

```
str_length(c('Cuantos caracteres hay aca', 'Cuantoscaractereshayaca'))
```

```
[1] 26 23
```

```
str_c('Unime', 'esto', 'por', 'favor')
```

```
[1] "Unimeestoporfavor"
```

```
str_c('Unime', 'esto', 'por', 'favor', sep = ' ')
```

```
[1] "Unime esto por favor"
```

`str_c()` recicla:

```
nombres <- c("Alejandro", "Alejandro", "Gabriel", "Facundo", "Luciana", "Sebast
str_c("Hola",nombres, sep = ', ')

[1] "Hola, Alejandro" "Hola, Alejandro" "Hola, Gabriel"   "Hola, Facundo"
[5] "Hola, Luciana"   "Hola, Sebastian"
```

```
str_c(nombres, collapse = ', ')

[1] "Alejandro, Alejandro, Gabriel, Facundo, Luciana, Sebastian"
```

Si combinamos las dos cosas anteriores, y primero recicla y luego colapsa.

```
str_c('Hola, ', nombres, collapse = '; ')

[1] "Hola, Alejandro; Hola, Alejandro; Hola, Gabriel; Hola, Facundo; Hola, Luci
```

Si hay una cadena de largo 0, la omite. En este ejemplo, el quinto argumento es `" and HAPPY BIRTHDAY"` si `birthday` es `TRUE`, pero si no es un argumento que queda como vacío.

```
name <- "Hadley"
time_of_day <- "morning"
birthday <- FALSE

str_c(
  "Good ", time_of_day, " ", name,
  if (birthday) " and HAPPY BIRTHDAY",
  "."
)
```

```
[1] "Good morning Hadley."
```

También se pueden reemplazar `NA`.

```
str_replace_na(NA, 'algo')
```

```
[1] "algo"
```

Dividir, extraer, *subsetting*

str_sub

```
x <- c("Apple", "Banana", "Pear")
str_sub(x, 1, 3)

[1] "App" "Ban" "Pea"

str_sub(x, c(1, 2, 3), c(6, 6, 6)) # los argumentos estan "vectorizados

[1] "Apple" "anana" "ar"

str_sub(x, -3, -1) # a lo python, ¿no?

[1] "ple" "ana" "ear"
```

¿Qué hace str_sub()<- ?

```
x <- letters[1:4] %>%
  str_c(collapse = '')

str_sub(x,2,3) <- 'z'
x

[1] "azd"
```

Ejercicios

1. En ejemplos de código en los que no se utiliza stringr, verás usualmente `paste()` y `paste0()`. ¿Cuál es la diferencia entre estas dos funciones? ¿A qué función de stringr son equivalentes? ¿Cómo difieren estas dos funciones respecto de su manejo de los NA?

```
paste('Hola', "como", "estas")x <- letters[1:4] %>%
  str_c(collapse = '')

str_sub(x,2,3) <- 'z'
x

paste('Hola', "como", NA)
paste0('Hola', "como", "estas")
paste0('Hola', "como", NA)
str_c("Hola","como","andas", sep = ' ')

Error: <text>:1:31: unexpected symbol
1: paste('Hola', "como", "estas")x
                                 ^
```

2. Describe con tus propias palabras la diferencia entre los argumentos sep y collapse de la función. `str_c()` □
3. Utiliza `str_length()` y `str_sub()` para extraer el caracter del medio de una cadena. ¿Qué harías si el número de caracteres es par?

```
x1 <- 'abcdefg'
x2 <- 'abcdefgh'

pos <- ceiling(str_length(x1)/2)
str_sub(x1, pos, pos)
```

```
[1] "d"
```

```
pos2 <- ceiling(str_length(x1)/2)
str_sub(x2,pos2, -pos2)
```

```
[1] "de"
```

4. ¿Qué hace `str_wrap()`? (wrap = envolver) ¿Cuándo podrías querer utilizarla?

```
texto <- stringi::stri_rand_lipsum(4)

str_c(texto, collapse = '\n') %>% cat

Lorem ipsum dolor sit amet, quam nisl fermentum arcu a, tempus purus in
Sed ad in id ut tempus, non consequat tellus ut proin non pretium. Dui
Non ad proin nec sit magna. Sed sociis egestas curae metus justo lacus
Sociosqu, iaculis, sed nec posuere purus sed libero condimentum magnis

texto <- str_c(texto, collapse = '\n\n')

str_wrap(texto, width = .9) %>% cat()

Lorem
ipsum
dolor
sit
amet,
quam
nisl
fermentum
arcu
```

5. ¿Qué hace `str_trim()`? (trim = recortar) ¿Cuál es el opuesto de `str_trim()`?

```
str_trim('   Hola    ', side = 'left')
[1] "Hola    "
str_trim('   Hola    ', side = 'right')
[1] "   Hola"
str_trim('   Hola    ', side = 'both')
[1] "Hola"
str_pad("Hola", width = 20, side = 'both')
[1] "        Hola        "
```

6. Escribe una función que convierta, por ejemplo, el vector `c("a", "b", "c")` en la cadena `a, b y c`. Piensa con detención qué debería hacer dado un vector de largo 0, 1 o 2.

```
convertir <- function(x){
  if (length(x)>2) {
    str_c(x[-length(x)], collapse = ', ') %>%
  str_c(x[length(x)], sep = ' y ')
  }

  else
    if (length(x==2)) str_c(x, collapse = ' y ')
    else
      if (length(x <= 1)) return(x)
}

lapply(4:1, function(n) {
  convertir(letters[1:n])
  })
```

```
[[1]]
[1] "a, b, c y d"

[[2]]
[1] "a, b y c"

[[3]]
[1] "a y b"

[[4]]
[1] "a"
```

"Las expresiones regulares son un lenguaje conciso que te permite describir patrones en cadenas de caracteres". En muchos editores de texto, se pueden usar para el famoso "buscar y reemplazar". También se pueden incluir en un código.

```
x <- c("apple", "banana", "pear")
str_detect(x, '.a.')
```

```
[1] FALSE  TRUE  TRUE
```

```
str_extract(x, 'an')
```

```
[1] NA   "an" NA
```

```
str_extract_all(x, '.a')
```

```
[[1]]
character(0)

[[2]]
[1] "ba" "na" "na"

[[3]]
[1] "ea"
```

```
str_locate_all(x,'.a')

[[1]]
     start end

[[2]]
     start end
[1,]     1   2
[2,]     3   4
[3,]     5   6

[[3]]
     start end
[1,]     2   3

str_match_all(x,'.a')

[[1]]
     [,1]

[[2]]
     [,1]
[1,] "ba"
[2,] "na"
[3,] "na"
```

Escapar caracteres

Hay caracteres especiales que tienen como un "doble significado", se representan a sí mismos, pero también a algo más. Esto no pasa con la letra a, que solo representa a la letra.
Por ejemplo: `_ { | . \[`
Para referirnos a ellos mismos se dice que "los escapamos" y por lo general se usa \antes del caracter. El caracter \ se escapa dependiendo del contexto, por ejemplo si queremos referirnos en latex a \, usamos `\textbackslash`. Pueden llegar a ser necesarias varias \para referirse a \. En una expresión regular de **R** hay que usar \\\\.

Anclas

```
x <- c("pie de manzana", "manzana", "queque de manzana", "manzana con c
str_detect(x, "manzana")

[1] TRUE TRUE TRUE TRUE

str_detect(x, "^manzana")

[1] FALSE  TRUE FALSE  TRUE

str_detect(x, "manzana$")

[1]  TRUE  TRUE  TRUE FALSE

str_detect(x, "^manzana$")

[1] FALSE  TRUE FALSE FALSE

str_extract('$^$', '\\$\\^\\$')

[1] "$^$"
```

Categorías

- \d: coincide con cualquier dígito.
- \s: coincide con cualquier espacio en blanco (por ejemplo, espacio simple, tabulador, salto de línea).
- [abc]: coincide con a, b o c.
- [^abc]: coincide con todo menos con a, b o c.

```
datos::palabras %>%
  stringi::stri_trans_general(id = "Latin-ASCII") %>% # le saco los til
  str_subset('^[aeiou]') # o str_which nos devuelve los indices
```

```
  [1] "a"             "abril"          "accion"         "acciones"
  [5] "acerca"        "actitud"        "actividad"      "actividades"
  [9] "acto"          "actual"         "acuerdo"        "adelante"
 [13] "ademas"        "administracion" "afirmo"         "agua"
 [17] "ahi"           "ahora"          "aire"           "al"
 [21] "algo"          "alguien"        "algun"          "alguna"
 [25] "algunas"       "algunos"        "alla"           "alli"
 [29] "alrededor"     "alta"           "alto"           "ambiente"
 [33] "ambos"         "america"        "amigo"          "amigos"
 [37] "amor"          "analisis"       "animales"       "ante"
 [41] "anterior"      "antes"          "antonio"        "ano"
 [45] "anos"          "aparece"        "apenas"         "apoyo"
```

Escribir la fórmula para un modelo

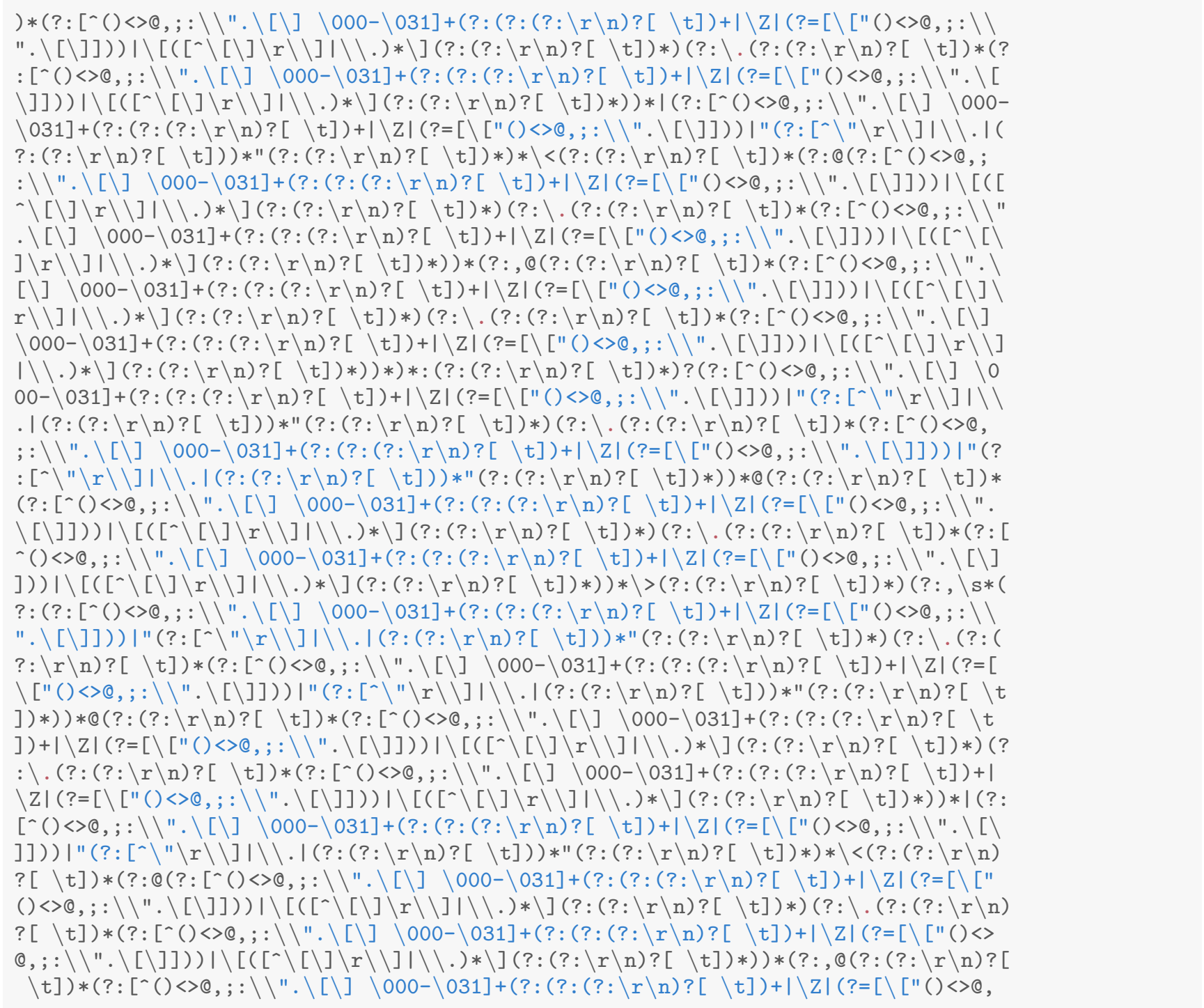
```
head(datos::diamantes, 3)

# A tibble: 3 x 10
  precio quilate corte   color claridad profundidad tabla     x     y     z
   <int>   <dbl> <ord>   <ord> <ord>          <dbl> <dbl> <dbl> <dbl> <dbl>
1    326    0.23 Ideal   E     SI2             61.5    55  3.95  3.98  2.43
2    326    0.21 Premium E     SI1             59.8    61  3.89  3.84  2.31
3    327    0.23 Bueno   E     VS1             56.9    65  4.05  4.07  2.31

names(datos::diamantes)

 [1] "precio"      "quilate"     "corte"       "color"       "claridad"
 [6] "profundidad" "tabla"       "x"           "y"           "z"

variables <- names(datos::diamantes)[-1]
numericas <- unlist(lapply(datos::diamantes, is.numeric))[-1]

rhs <- paste0(sprintf("log(%s) + %s", variables[numericas], variables[!numerica
sprintf("log(%s) ~ %s", names(datos::diamantes)[1], rhs) %>%
  as.formula(env = .GlobalEnv)

log(precio) ~ log(quilate) + corte + log(profundidad) + color +
    log(tabla) + claridad + log(x) + corte + log(y) + color +
    log(z) + claridad
```

Escribir la fórmula para un modelo

```
var.num <- variables[numericas]
var.nonum <- variables[!numericas]

str_glue('log({var.num})')

log(quilate)
log(profundidad)
log(tabla)
log(x)
log(y)
log(z)

str_glue('{var.nonum}')

corte
color
claridad

c(str_glue('log({var.num})'), str_glue('{var.nonum}')) %>%
  str_c(collapse = ' + ' )

[1] "log(quilate) + log(profundidad) + log(tabla) + log(x) + log(y) + log(z) + corte + color + claridad"

c(str_glue('log({var.num})'), str_glue('{var.nonum}')) %>%
  str_c(collapse = ' + ' ) %>%
  str_c('log(precio) ~ ', .) %>%
  as.formula(env = .GlobalEnv)

log(precio) ~ log(quilate) + log(profundidad) + log(tabla) +
    log(x) + log(y) + log(z) + corte + color + claridad
```

La expresión regular que chequea la validez de una dirección de correo electrónico:

```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:
\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(
?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[
\t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\0
31]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\
](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:
(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z
|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)
?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\
r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[
 \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)
?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t]
)*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[
 \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*
)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)
*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+
|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r
\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:
\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t
]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031
]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](
?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?
:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?
:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?
:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?
[ \t]))*"(?:(?:\r\n)?[ \t])*)*:(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\]
\000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|
\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>
@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"
(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t]
```

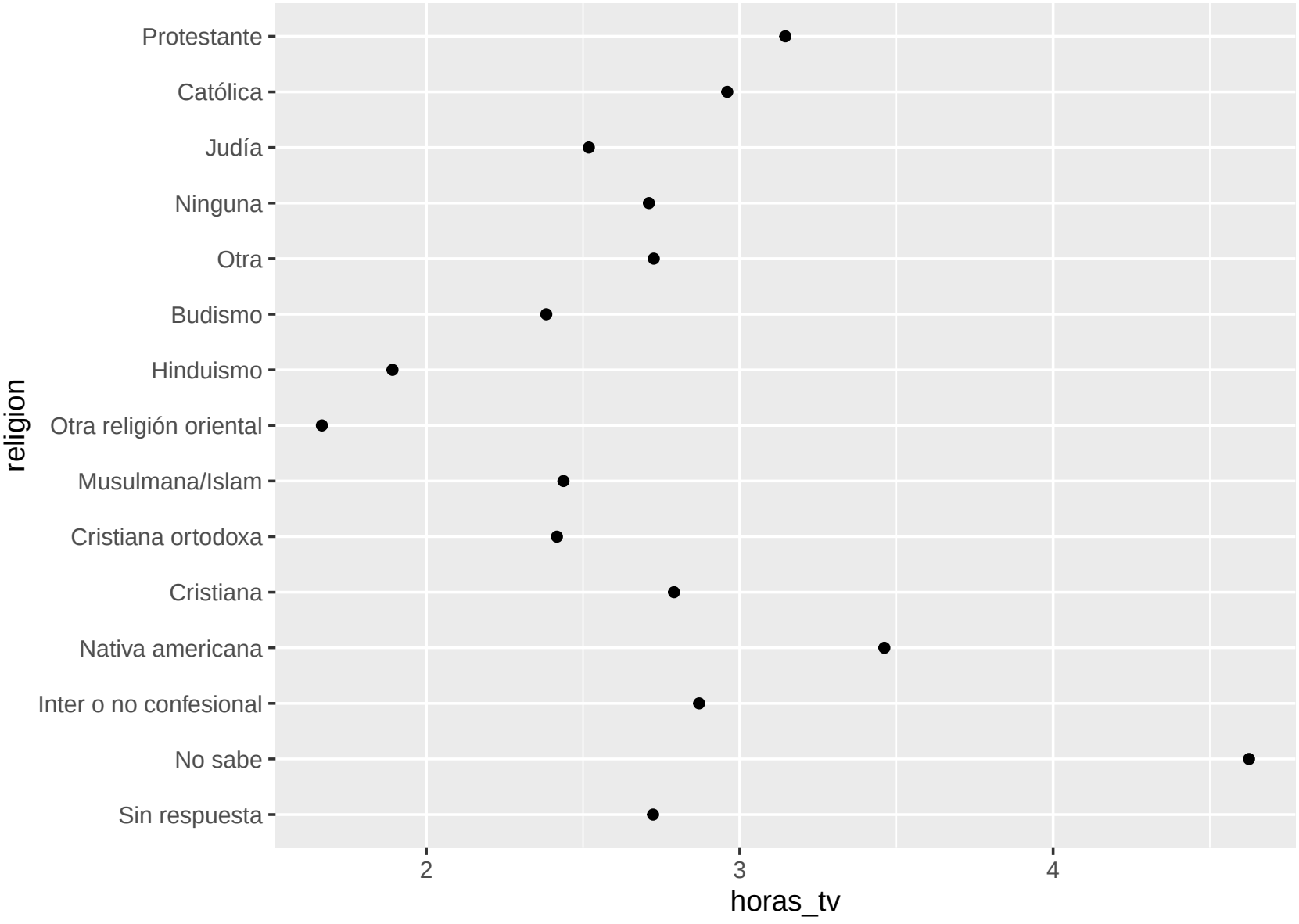

)*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*)(?:\.(?:(?:\r\n)?[\t])*(?
:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[
\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*))*|(?:[^()<>@,;:\\".\[\] \000-
\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(
?:(?:\r\n)?[\t]))*"(?:(?:\r\n)?[\t])*)*\<(?:(?:\r\n)?[\t])*(?:@(?:[^()<>@,;
:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([
^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*)(?:\.(?:(?:\r\n)?[\t])*(?:[^()<>@,;:\\"
.\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\
]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*))*(?:,@(?:(?:\r\n)?[\t])*(?:[^()<>@,;:\\".\
[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\
r\\]|\\.)*\](?:(?:\r\n)?[\t])*)(?:\.(?:(?:\r\n)?[\t])*(?:[^()<>@,;:\\".\[\]
\000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]
|\\.)*\](?:(?:\r\n)?[\t])*))*)*:(?:(?:\r\n)?[\t])*)?(?:[^()<>@,;:\\".\[\] \0
00-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\
.|(?:(?:\r\n)?[\t]))*"(?:(?:\r\n)?[\t])*)(?:\.(?:(?:\r\n)?[\t])*(?:[^()<>@,
;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?
:[^\"\r\\]|\\.|(?:(?:\r\n)?[\t]))*"(?:(?:\r\n)?[\t])*))*@(?:(?:\r\n)?[\t])*
(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*)(?:\.(?:(?:\r\n)?[\t])*(?:[
^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\]
]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*))*\>(?:(?:\r\n)?[\t])*)(?:,\s*(
?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[\t]))*"(?:(?:\r\n)?[\t])*)(?:\.(?:(
?:\r\n)?[\t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[
\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[\t]))*"(?:(?:\r\n)?[\t
])*))*@(?:(?:\r\n)?[\t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t
])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*)(?
:\.(?:(?:\r\n)?[\t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|
\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*))*|(?:
[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,;:\\".\[\
]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[\t]))*"(?:(?:\r\n)?[\t])*)*\<(?:(?:\r\n)
?[\t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["
()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*)(?:\.(?:(?:\r\n)
?[\t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>
@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[\t])*))*(?:,@(?:(?:\r\n)?[
 \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[\t])+|\Z|(?=[\["()<>@,

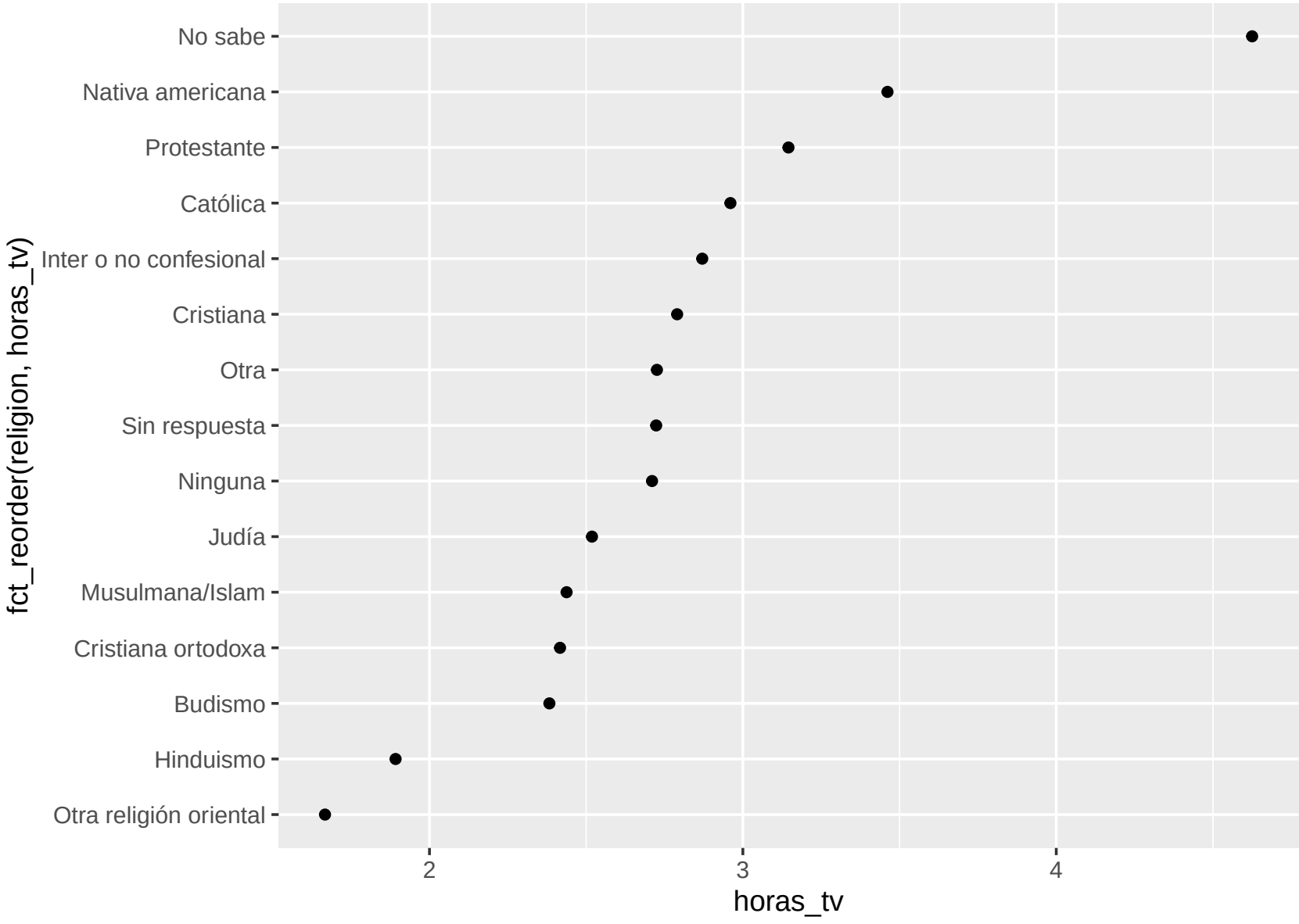
```
;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t]
)*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?
(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:
\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\[
"()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])
*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])
+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\
.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z
|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(
?:\r\n)?[ \t])*))*)?;\s*)
```

Contenido de la sección 2

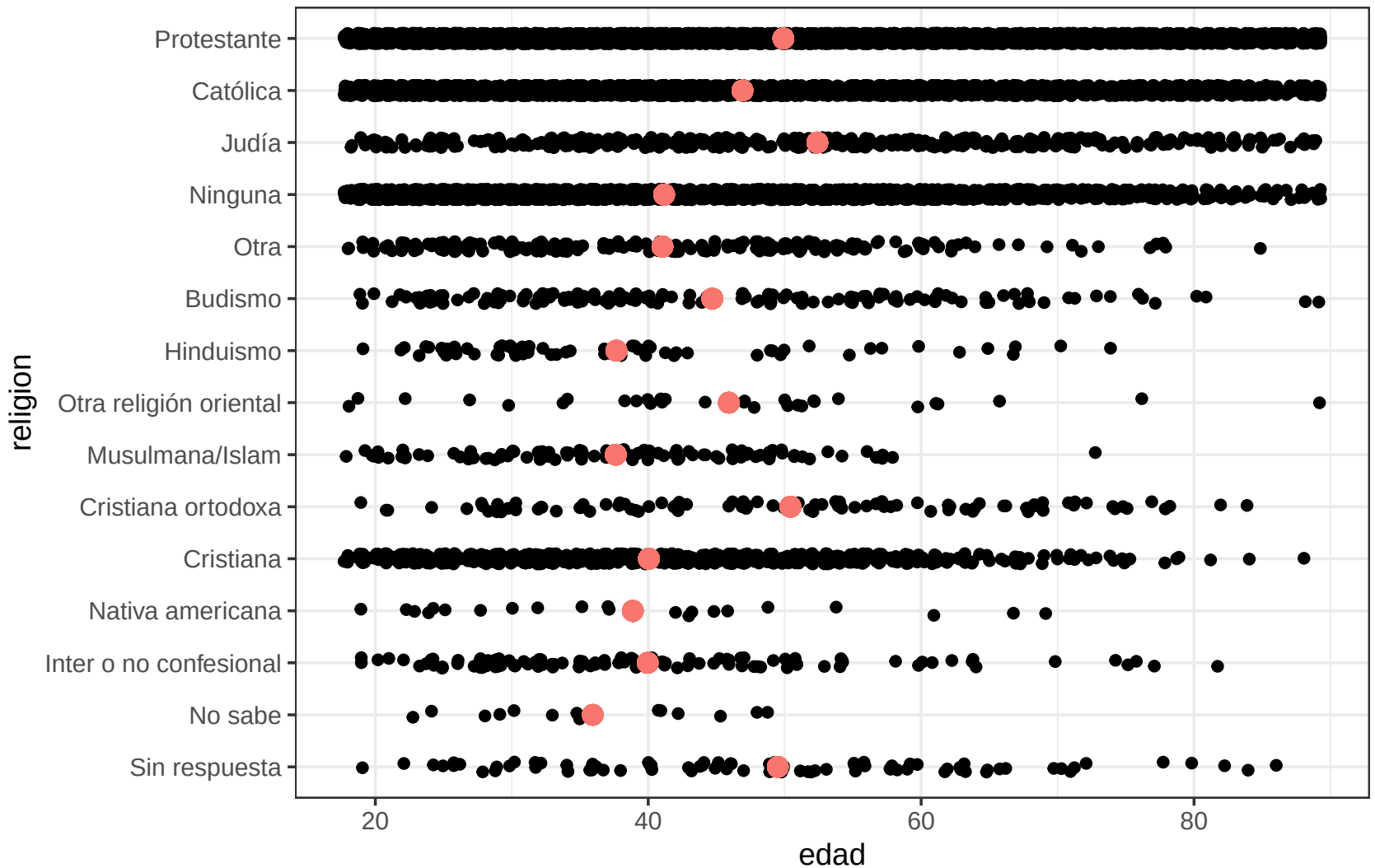
```r
resumen_religion <- encuesta %>%
  group_by(religion) %>%
  summarise(
    edad = mean(edad, na.rm = TRUE),
    horas_tv = mean(horas_tv, na.rm = TRUE),
    n = n()
  )
```

```r
ggplot(resumen_religion, aes(horas_tv, religion)) +
  geom_point()

ggplot(resumen_religion, aes(horas_tv, fct_reorder(religion, horas_tv))
  geom_point()
)
```

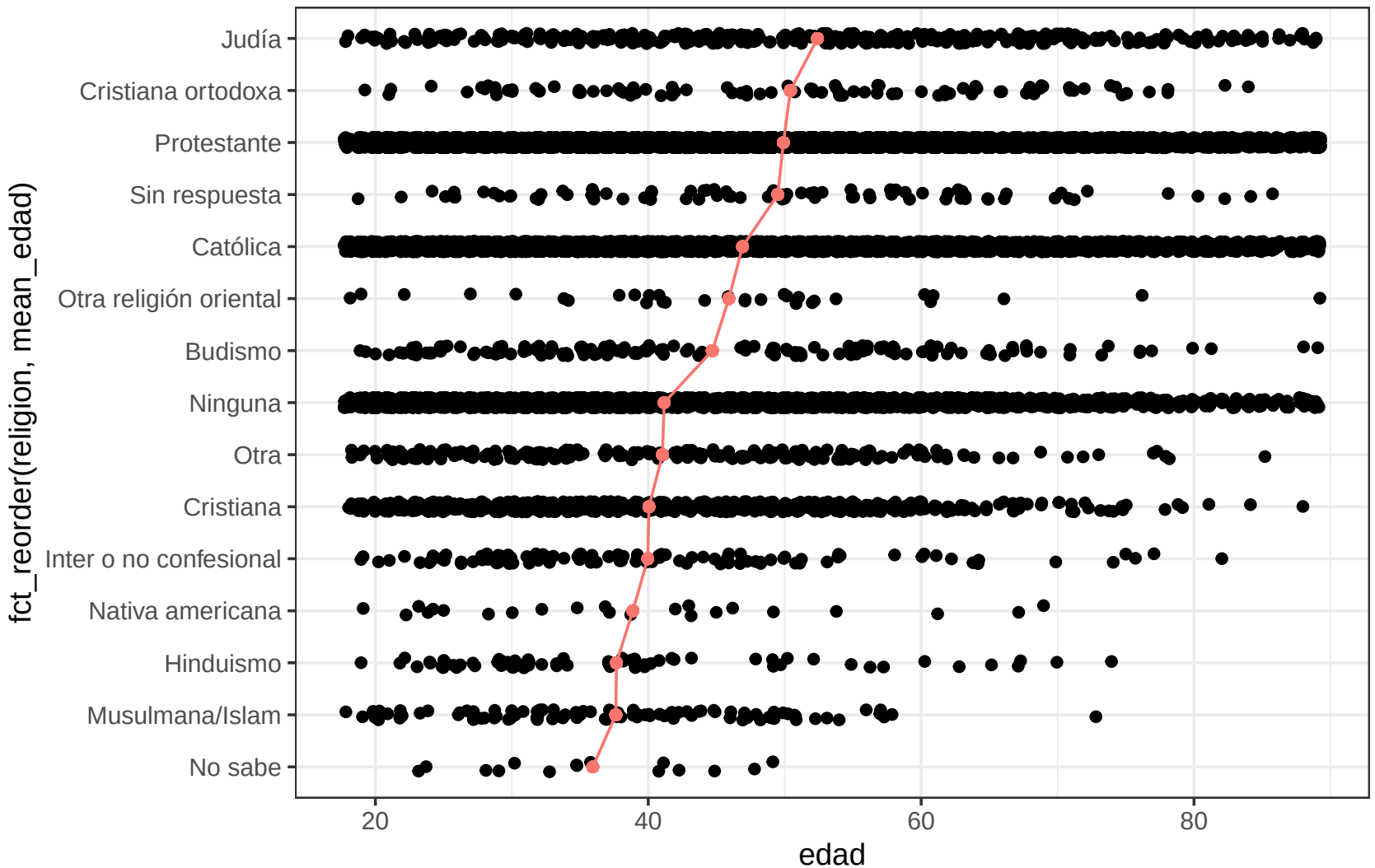

Protestante
Católica
Judía
Ninguna
Otra
Budismo
Hinduismo
Otra religión oriental
Musulmana/Islam
Cristiana ortodoxa
Cristiana
Nativa americana
Inter o no confesional
No sabe
Sin respuesta
religion
2
3
4
horas_tv


No sabe
Nativa americana
Protestante
Católica
Inter o no confesional
Cristiana
Otra
Sin respuesta
Ninguna
Judía
Musulmana/Islam
Cristiana ortodoxa
Budismo
Hinduismo
Otra religión oriental
fct_reorder(religion, horas_tv)
2
3
4
horas_tv

```r
encuesta %>%
  group_by(religion) %>%
  mutate(mean_edad = mean(edad, na.rm = T)) %>%
  ggplot() +
  geom_jitter(aes(edad, religion), height = .1, width = 0.3)+
  geom_point(aes(x = mean_edad, y = religion, col = 'media'), show.legend = F, size = 3) +
  theme_bw()
```



```r
encuesta %>%
  group_by(religion) %>%
  mutate(mean_edad = mean(edad, na.rm = T)) %>%
  ggplot() +
  geom_jitter(aes(edad, fct_reorder(religion, mean_edad)), height = .1, width = .3)+
  geom_point(aes(x = mean_edad, y = religion, col = 'media'), show.legend = F) +
  geom_line(aes(x = mean_edad, y = religion, group = 'media', col = 'media'), show.legend = F) +
  theme_bw()
```



Otros re-ordenamientos

Funciones para revisar que amplían posibilidades:
`fct_reorder2()`, `fct_infreq()`, `fct_rev()`.

Recodificar factores

`fct_recode()`, `fct_collapse()`, `fct_lump()`.

Contenido de la sección 3

- Datos solo de fechas. Para **R** es un `date`, en un tibble aparece como `<date>`
- Datos solo de hora. En un tibble aparece como `<tune>`
- Datos de fecha y hora. Esto para **R** es un POSIXct, pero en un tibble aparece como `<dttm>`

```
library(tidyverse)
library(lubridate)
library(datos)

today()
[1] "2021-05-19"
now() - 100000
[1] "2021-05-18 07:21:55 -03"
today() - 1 # este esta barbaro para fingir que estuviste holgado
[1] "2021-05-18"
today() %>% class
[1] "Date"
now() %>% class
[1] "POSIXct" "POSIXt"
```

```
now() %>% year
```

```
[1] 2021
```

```
now() %>% month
```

```
[1] 5
```

```
now() %>% day
```

```
[1] 19
```

```
(today() + 0:31) %>% mday
```

```
 [1] 19 20 21 22 23 24 25 26 27 28 29 30 31  1  2  3  4  5  6  7  8  9
[26] 13 14 15 16 17 18 19
```

```
(today() + 0:31) %>% yday
```

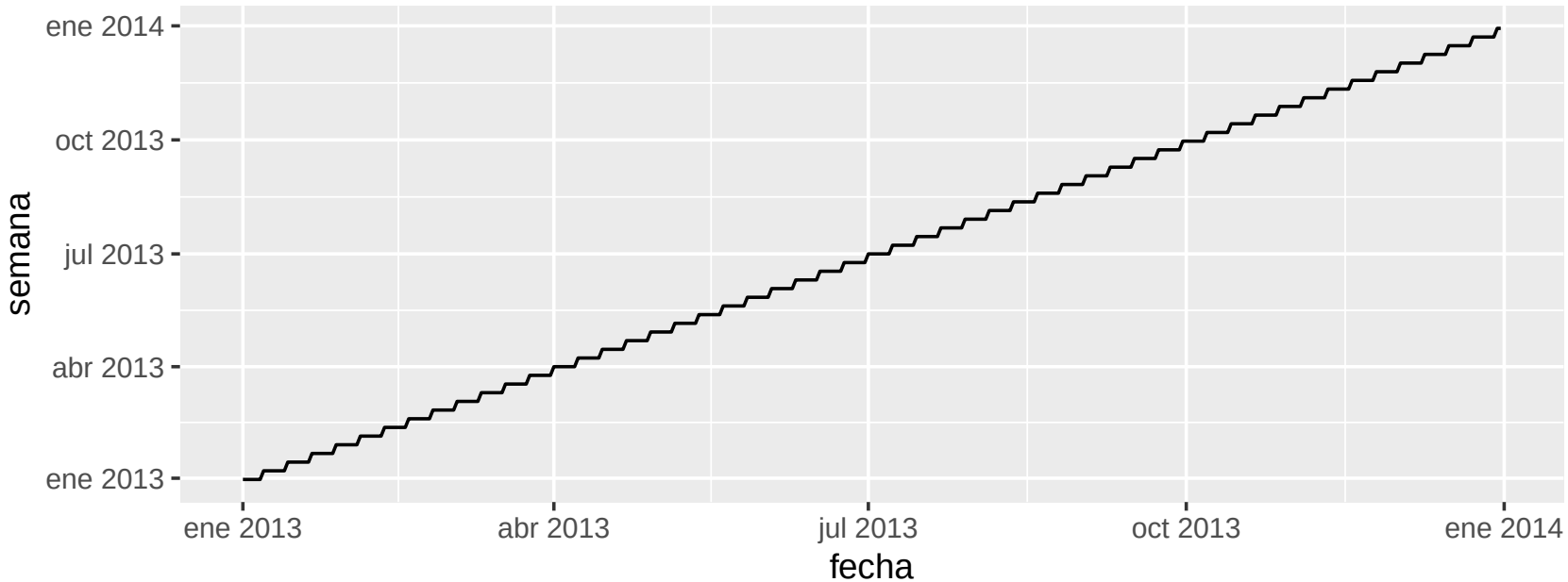
```
 [1] 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 15
[20] 158 159 160 161 162 163 164 165 166 167 168 169 170
```

```
(today() + 0:31) %>% wday
```

```
 [1] 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7
```

Redondeo de fechas

```
fechas <- vuelos %>%
  mutate(fecha = make_datetime(anio,mes,dia)) %>%
  select(fecha) %>%
  unique %>%
  mutate(semana = floor_date(fecha, unit = 'week', week_start = 1))

ggplot(fechas, aes(fecha, semana)) + geom_line()
```




```
floor_date(today(), 'week') # el 16 cayo domingo, quiero la semana del lunes
```

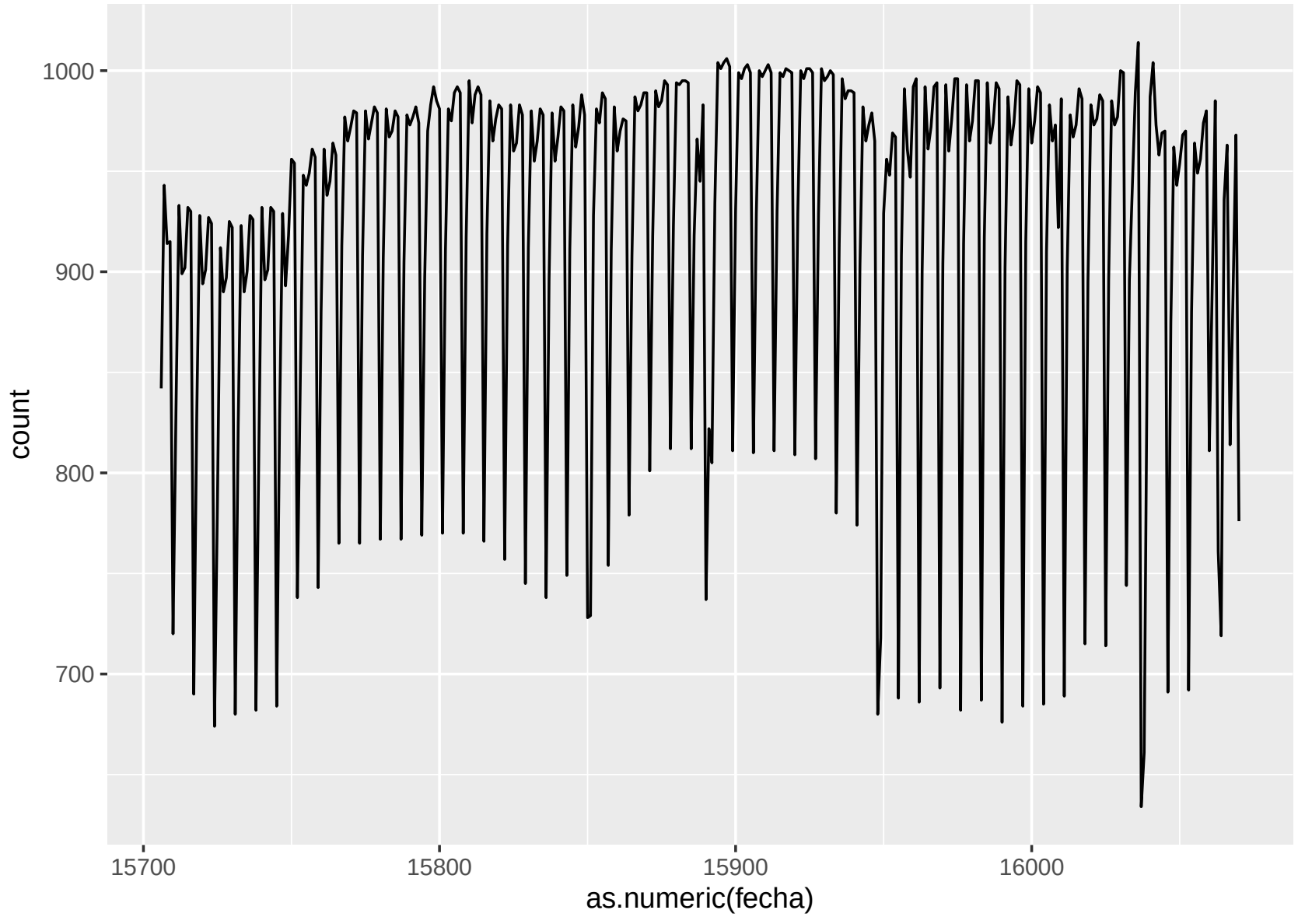
```
[1] "2021-05-16"
```

```
floor_date(today(), 'week', week_start = 1) # ahora si
```

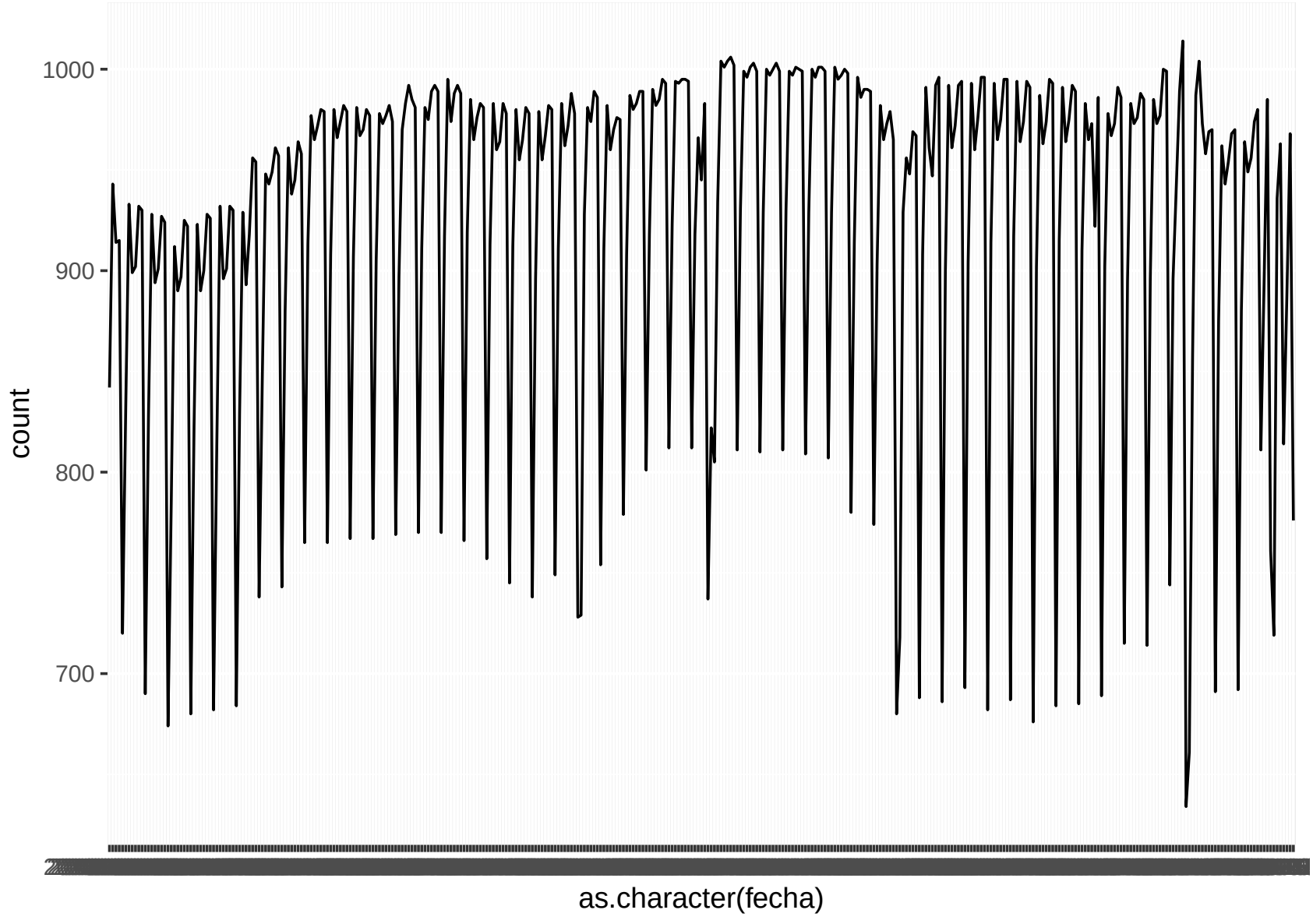
```
[1] "2021-05-17"
```

```r
vuelos_dt <- vuelos %>%
  mutate(fecha = make_date(anio, mes, dia)) %>%
  select(fecha, horario_salida) %>%
  group_by(fecha) %>%
  summarise(count = n())
```

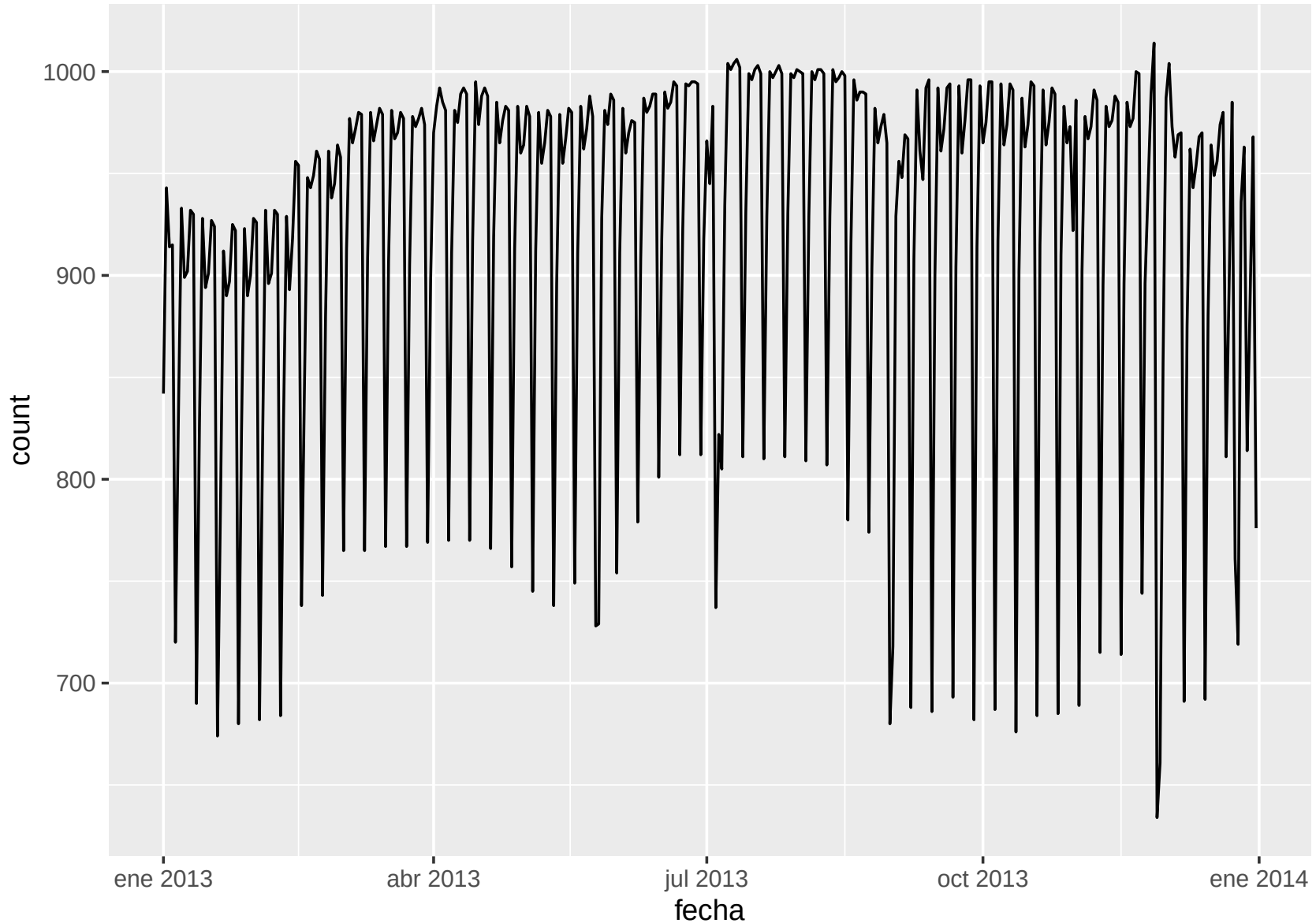
```
vuelos_dt %>%
  ggplot(aes(as.numeric(fecha), count)) +
  geom_line()
```



```
vuelos_dt %>%
  ggplot(aes(as.character(fecha), count, group = 1)) +
  geom_line()
```

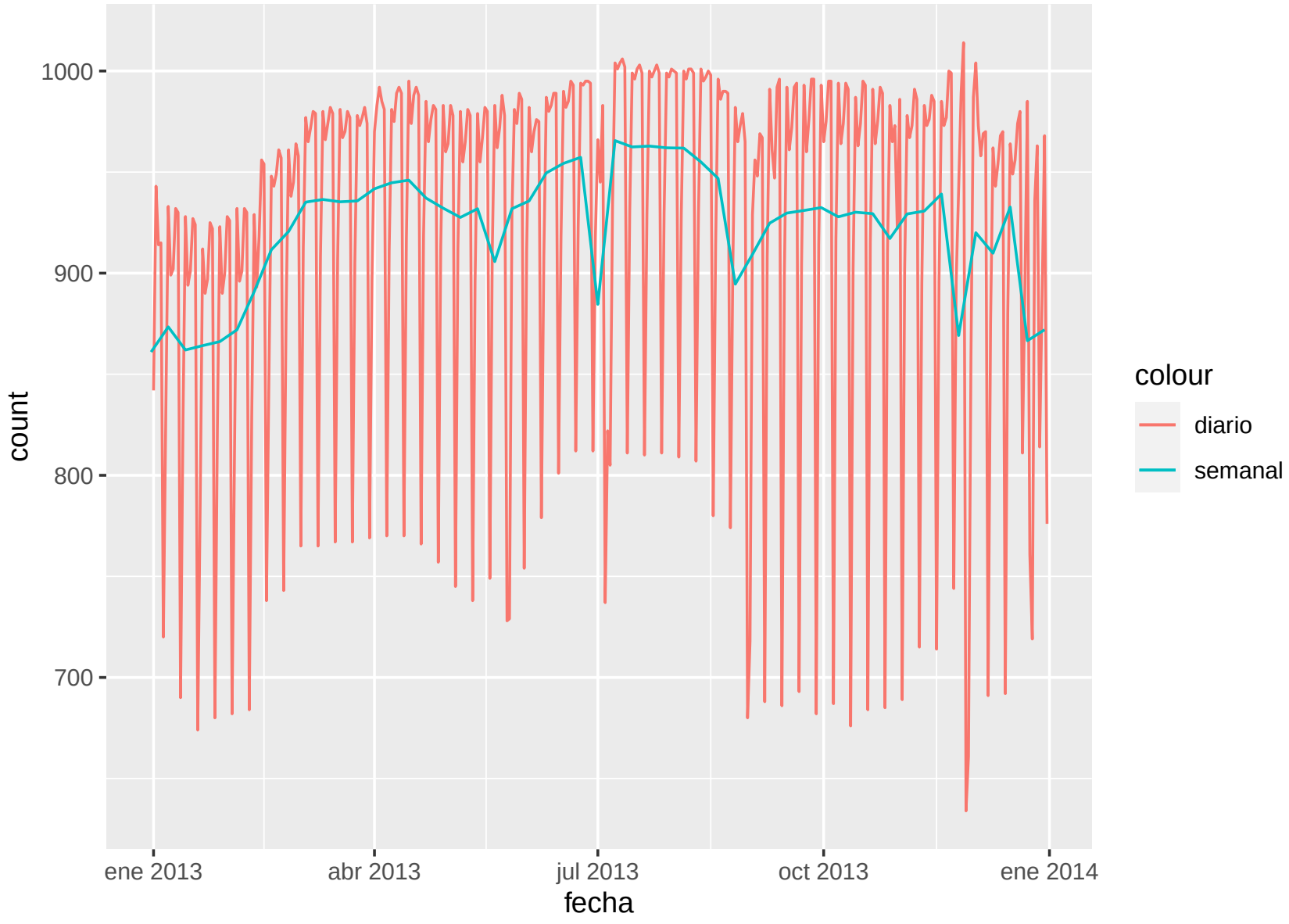


```
vuelos_dt %>%
  ggplot(aes(fecha, count)) +
  geom_line()
```



```r
vuelos_semana <- vuelos_dt %>%
  mutate(semana = floor_date(fecha, unit = 'week', week_start = 1)) %>%
  group_by(semana) %>%
  summarise(count = mean(count))
```

```
ggplot() +
geom_line(data = vuelos_dt, aes(fecha, count, col = 'diario')) +
geom_line(data = vuelos_semana, aes(semana, count, col = 'semanal'))
```



Nada que vero pero rimó

```
fecha_character <- today() %>% as.character()
fecha_character %>% fasttime::fastPOSIXct() %>% as.Date()
```

```
[1] "2021-05-19"
```