Menos for, más purrr: Programación funcional con R

Nacho Evangelista

18 de febrero de 2020

Contenido

```
Warning: package 'dplyr' was built under R version 3.6.3

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

A

- Nuestro objetivo: reemplazar los *for loops* o estructuras de repetición. Los lenguajes de programación puramente funcionales utilizan funciones para lograr los mismos resultados.
- En R base existen las funciones de la familia `apply`. En el paquete `purrr`, estas funciones se reemplazan por la familia `map`, más fáciles de usar. Se llaman funcionales: reciben una función como argumento.
- En una estructura funcional, no es necesario crear una lista vacía para ir guardando resultados, el código es más conciso.
- Premisa: arrancar con porciones de código pequeños y fáciles de entender (funciones). Combinar estos bloques en estructuras más complejas.
- Evitar duplicación de código. La duplicación hace que los errores y *bugs* sean más frecuentes. También se hace más difícil modificar el código.
- Principio: no repetirse a uno mismo (DRY: *don't repeat yourself*)
- R programmers prefer to solve this type of problem by applying an

Listas

- El bloque fundamental de `purrr` son las listas:
  - Un vector es un objeto que guarda elementos individuales del mismo tipo
  - Un dataframe es una estructura que guarda varios vectores de la misma longitud pero de distinto tipo.
  - Una lista es una estructura que permite guardar objetos de distinto tipo y longitud.
- https://cran.r-project.org/web/packages/listviewer/index.html

A

- Muchas operaciones de R simplemente funcionan en forma vectorizada; cuando proveemos vectores como entrada, la función se aplica elemento a elemento (una suerte de iteración)
- Muchas funciones no tienen esa capacidad
- `purrr` sirve para iterar
- La función (funcional) más básica de `purrr` es `map`: toma un vector y una función, y aplica la función a cada elemento del vector, devolviendo los resultados en una lista.
- `map(1:3,f)` es equivalente a `list(f(1), f(2), f(3))`

A

A

map2

Pero yo nunca usé listas...

Columnas lista y dataframes anidados

- Some crazy stuff starts happening when you learn that tibble columns can be lists (as opposed to vectors, which is what they usually are).
- For instance, a tibble can be “nested” where the tibble is essentially split into separate data frames based on a grouping variable, and these separate data frames are stored as entries of a list

¿Cómo se construyen?

- Definición:
- Usando `nest`
- Como resultado de una operación...

¿Por qué necesito purrr?

Secuencia de fechas

Contamos con los movimientos de dos empresas. Interesa tener la serie temporal de eventos para cada empresa y producto.

```
datos ← tibble::tribble(
  ~empresa, ~producto,       ~fecha, ~evento,
       "A",       "A1", "02/06/2018",     112,
       "A",       "A1", "06/06/2018",     141,
       "A",       "A1", "13/07/2018",     119,
       "A",       "A2", "01/05/2018",      53,
       "A",       "A2", "04/05/2018",      67,
       "B",       "B1", "01/07/2018",     127,
       "B",       "B1", "05/07/2018",     301,
       "B",       "B1", "10/07/2018",      98,
       "B",       "B1", "11/07/2018",     167)
datos$fecha ← as.Date(datos$fecha,format = "%d/%m/%Y")
```

Idea:

1. Determinar la primera y última fecha de cada grupo
2. Generar una secuencia de fechas (`seq.Date`) para cada grupo y construir una tabla con todas las fechas
3. Unir esta tabla con la original

```
fechas_todas ←
  datos %>%
  group_by(empresa, producto) %>%
  summarise(fecha_inicial = min(fecha),
            fecha_final = max(fecha)) %>%
  mutate(fechas = map2(fecha_inicial,
                       fecha_final,
                       ~seq.Date(.x,.y,by="1 day"))) %>%
  select(-fecha_inicial,-fecha_final)
```

```
fechas_todas %>%
  unnest(fechas) %>%
  left_join(datos, by = c("empresa","producto","fechas" = "fecha"))
```

```
# A tibble: 57 x 4
# Groups:   empresa [2]
   empresa producto fechas     evento
   <chr>   <chr>    <date>      <dbl>
 1 A       A1       2018-06-02    112
 2 A       A1       2018-06-03     NA
 3 A       A1       2018-06-04     NA
 4 A       A1       2018-06-05     NA
 5 A       A1       2018-06-06    141
 6 A       A1       2018-06-07     NA
 7 A       A1       2018-06-08     NA
 8 A       A1       2018-06-09     NA
 9 A       A1       2018-06-10     NA
10 A       A1       2018-06-11     NA
# ... with 47 more rows
```

Separar datos en filas

```
peleas ← tibble::tribble(
 ~pelea, ~horario,                          ~casting,
      1,  "20:30",          "thf028,fez195,yfm179",
      2,  "20:50",                "thf028,yfm179",
      3,  "19:40", "jfa348,fez195,gky651,wpx281",
      4,  "21:00",                "thf028,fez195",
      5,  "19.20",                "phb625,yfm179",
      6,  "20:10", "yfm179,gsf901,thf028,fez195")
```

```
luchadores ← tibble::tribble(
   ~codigo,            ~nombre,
  "gsf901",  "Vicente Viloni",
  "thf028",     "Hip Hop Man",
  "wpx281",         "La Masa",
  "fez195", "Fulgencio Mejía",
  "jfa348",         "Mc Floyd",
  "phb625",     "Mario Morán",
  "gky651",      "Rulo Verde",
  "yfm179",    "Steve Murphy")
```

Abrir varios archivos a la vez

```
# A tibble: 4 x 1
  .
  <chr>
1 archivo_1.csv
2 archivo_2.csv
3 archivo_3.csv
4 data.csv
```

Rotar puntos

Train/test

Múltiples salidas

```
# A tibble: 7 x 3
  banda          cancion              frase
  <chr>          <chr>                <chr>
1 Los Wachiturr~ Este es el pasito    El que no hace palmas es un ga
2 La Base        Sabor sabrosón       Según la moraleja, el que no h
3 Damas Gratis   Me va a extrañar     ATR perro cumbia cajeteala pic
4 Altos Cumbier~ No voy a llorar      Andy, fijate que volvieron, qu
5 Los Pibes Cho~ Llegamos los Pibes~ Llegamos los pibes chorros que
6 La Liga        Se re pudrió         El que no hace palmas tiene fa
7 Los Palmeras   La cola              A la una, a la dos, a la one t
```

Múltiples plots

```
# A tibble: 10 x 2
   `"x"` `"y"`
   <dbl> <dbl>
 1   211   184
 2   230   147
 3   587   413
 4   414   252
 5   419   252
 6   157   272
 7   327   158
 8   222   158
 9   451   249
10   296   127
```

possibly

Resumen