Clase 3.0
Scripts, funciones y control de flujo

Marcos Rosetti y Luis Pacheco-Cobos

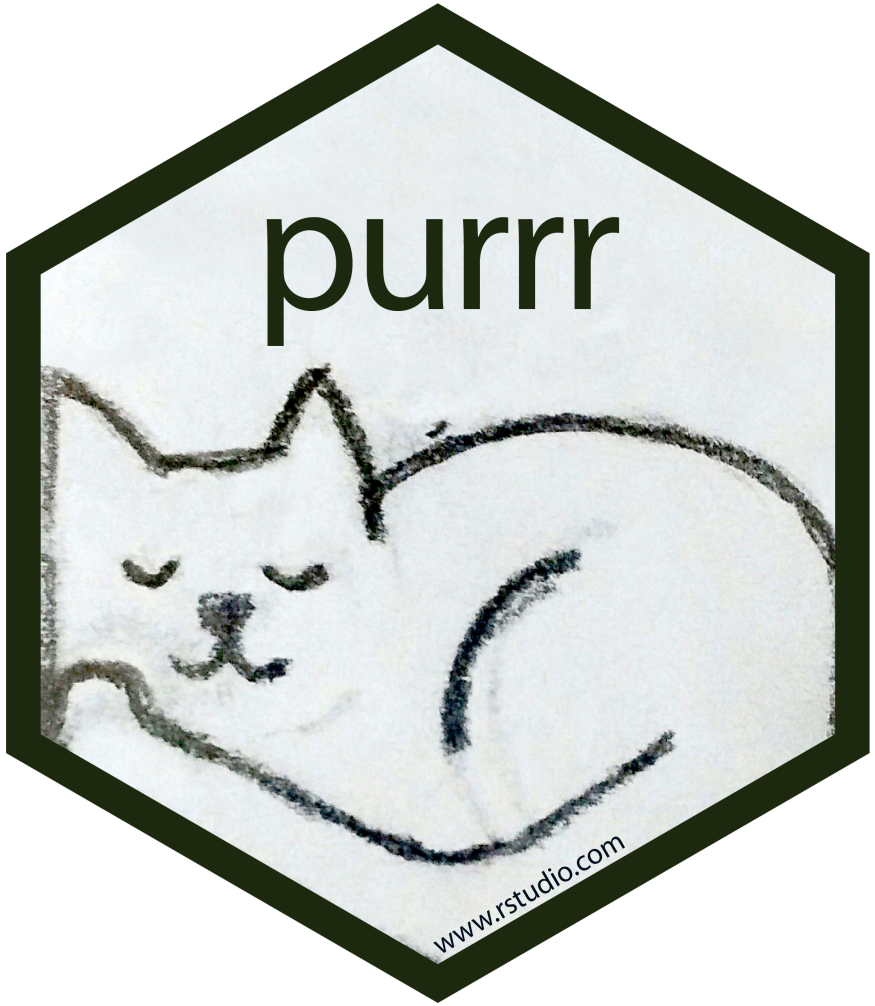
Estadística y Manejo de Datos con R (EMDR) — Virtual

Programación funcional

Programación funcional

- Es un paradigma de programación declarativa basado en el uso de funciones matemáticas.

Programación funcional



Programación funcional

- `purrr`
- Instalamos las bibliotecas.

```
install.packages("purrr")
install.packages("tidyverse")
install.packages("broom")
```

- Cargamos las bilbiotecas.

```
library(purrr)
library(tidyverse)
library(broom)
```

Programación funcional

- `nest()` y `unnest()`

```
head(mtcars)
?mtcars
```

```
n_mtcars <- mtcars %>%
  nest(-cyl) # produce un df de listas
```

```
## Warning: All elements of `...` must be named.
## Did you want `data = c(mpg, disp, hp, drat, wt, qsec, vs, am, gear, carb)`?
```

```
n_mtcars
```

```
## # A tibble: 3 × 2
##     cyl data
##   <dbl> <list>
## 1     6 <tibble [7 × 10]>
## 2     4 <tibble [11 × 10]>
## 3     8 <tibble [14 × 10]>
```

Programación funcional

- `unnest()`

```
n_mtcars %>%
  unnest()
```

```
## Warning: `cols` is now required when using unnest().
## Please use `cols = c(data)`
```

```
## # A tibble: 32 × 11
##      cyl   mpg  disp    hp  drat    wt  qsec    vs    am  gear  carb
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     6  21    160    110  3.9   2.62  16.5     0     1     4     4
##  2     6  21    160    110  3.9   2.88  17.0     0     1     4     4
##  3     6  21.4  258    110  3.08  3.22  19.4     1     0     3     1
##  4     6  18.1  225    105  2.76  3.46  20.2     1     0     3     1
##  5     6  19.2  168.   123  3.92  3.44  18.3     1     0     4     4
##  6     6  17.8  168.   123  3.92  3.44  18.9     1     0     4     4
##  7     6  19.7  145    175  3.62  2.77  15.5     0     1     5     6
##  8     4  22.8  108     93  3.85  2.32  18.6     1     1     4     1
##  9     4  24.4  147.    62  3.69  3.19  20       1     0     4     2
## 10     4  22.8  141.    95  3.92  3.15  22.9     1     0     4     2
## # … with 22 more rows
```

Programación funcional

- `map()`

```
my_test <- function(x) {
  lm(mpg ~ wt, data=x)
  }

mtcars %>%
  nest(-cyl) %>%
  mutate(res = map(data, my_test))
```

```
## Warning: All elements of `...` must be named.
## Did you want `data = c(mpg, disp, hp, drat, wt, qsec, vs, am, gear, carb)`?
```

```
## # A tibble: 3 × 3
##     cyl data               res
##   <dbl> <list>             <list>
## 1     6 <tibble [7 × 10]>  <lm>
## 2     4 <tibble [11 × 10]> <lm>
## 3     8 <tibble [14 × 10]> <lm>
```

Programación funcional

- `map()`

```
my_test <- function(x) {
  lm(mpg ~ wt, data=x)
  }

mtcars %>%
  nest(-cyl) %>%
  mutate(res = map(data, my_test)) %>%
  mutate(glance_lm = res %>% map(glance))
```

```
## Warning: All elements of `...` must be named.
## Did you want `data = c(mpg, disp, hp, drat, wt, qsec, vs, am, gear, carb)`?
```

```
## # A tibble: 3 × 4
##     cyl data               res    glance_lm
##   <dbl> <list>             <list> <list>
## 1     6 <tibble [7 × 10]>  <lm>   <tibble [1 × 12]>
## 2     4 <tibble [11 × 10]> <lm>   <tibble [1 × 12]>
## 3     8 <tibble [14 × 10]> <lm>   <tibble [1 × 12]>
```

Programación funcional

- `map()`

```
my_test <- function(x) {
  lm(mpg ~ wt, data=x)
  }

mtcars %>%
  nest(-cyl) %>%
  mutate(res = map(data, my_test)) %>%
  mutate(glance_lm = res %>% map(glance)) %>%
  unnest(glance_lm)
```

```
## Warning: All elements of `...` must be named.
## Did you want `data = c(mpg, disp, hp, drat, wt, qsec, vs, am, gear, carb)`?
```

```
## # A tibble: 3 × 15
##     cyl data  res   r.squared adj.r.squared sigma statistic p.value    df logLik
##   <dbl> <lis> <lis>     <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl>
## 1     6 <tib… <lm>      0.465         0.357  1.17      4.34  0.0918     1  -9.83
## 2     4 <tib… <lm>      0.509         0.454  3.33      9.32  0.0137     1 -27.7
## 3     8 <tib… <lm>      0.423         0.375  2.02      8.80  0.0118     1 -28.7
## # … with 5 more variables: AIC <dbl>, BIC <dbl>, deviance <dbl>,
## #   df.residual <int>, nobs <int>
```

Programación funcional

- `map()`, otra versión.

```
mtcars %>%
  split(.$cyl) # de R base
```

```
## $`4`
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
##
## $`6`
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Valiant        18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
##
## $`8`
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
```

Programación funcional

- `map()`, otra versión.

```
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .))
```

```
## $`4`
##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Coefficients:
## (Intercept)           wt
##      39.571       -5.647
##
##
## $`6`
##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Coefficients:
## (Intercept)           wt
##       28.41        -2.78
##
##
## $`8`
##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Coefficients:
```

Programación funcional

- `map()`, otra versión.

```
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .)) %>%
  map(summary)
```

```
## $`4`
##
## Call:
## lm(formula = mpg ~ wt, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1513 -1.9795 -0.6272  1.9299  5.2523
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   39.571      4.347   9.104 7.77e-06 ***
## wt            -5.647      1.850  -3.052   0.0137 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.332 on 9 degrees of freedom
## Multiple R-squared:  0.5086, Adjusted R-squared:  0.454
## F-statistic: 9.316 on 1 and 9 DF,  p-value: 0.01374
##
##
## $`6`
##
## Call:
## lm(formula = mpg ~ wt, data = .)
```

Programación funcional

- `map()`, otra versión.

```
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .)) %>%
  map(summary) %>%
  map("r.squared")
```

```
## $`4`
## [1] 0.5086326
##
## $`6`
## [1] 0.4645102
##
## $`8`
## [1] 0.4229655
```

Programación funcional

- `map()`, otra versión.

```
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .)) %>%
  map(summary) %>%
  map_dbl("r.squared")
```

```
##         4         6         8
## 0.5086326 0.4645102 0.4229655
```

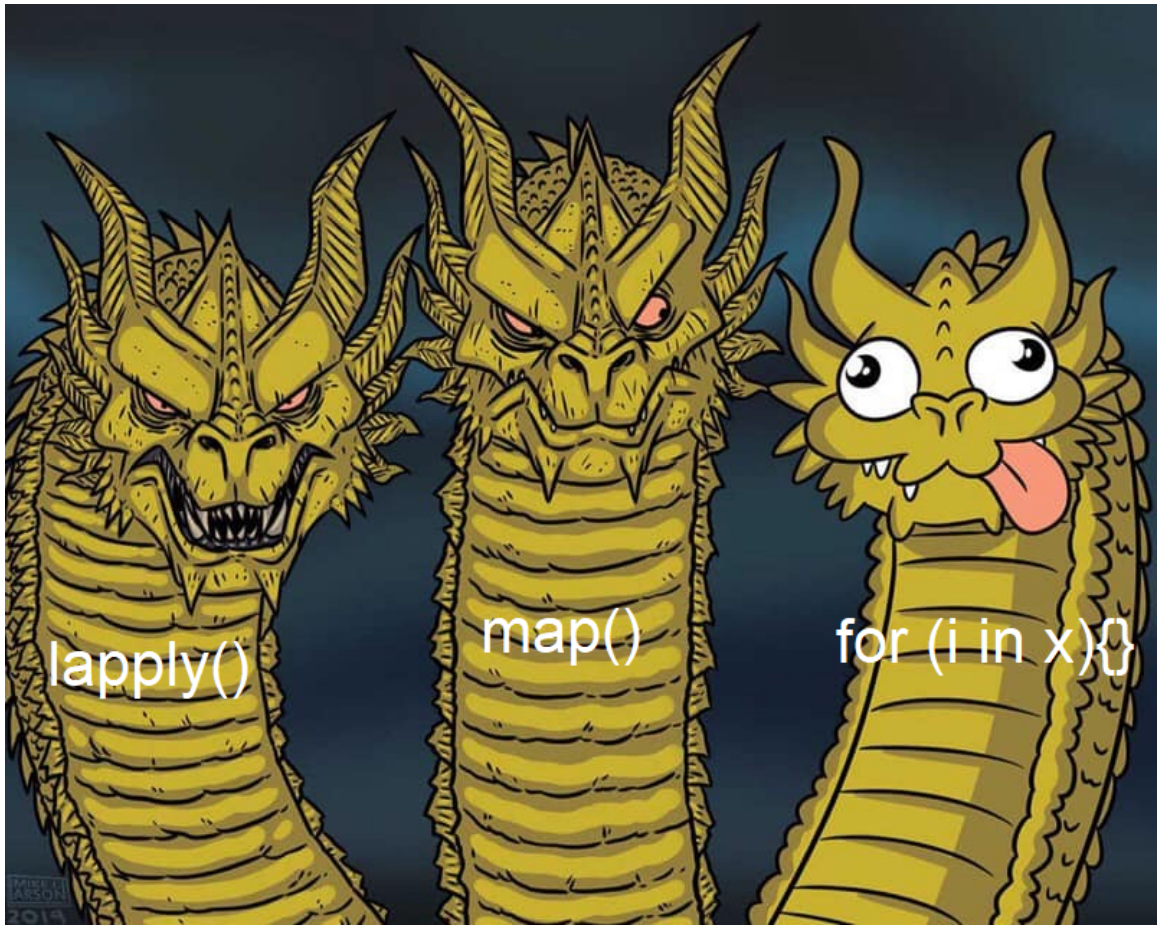
Programación funcional

- `map()`, otra versión.

```
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .)) %>%
  map(summary) %>%
  map_df("r.squared")
```

```
## # A tibble: 1 × 3
##     `4`   `6`   `8`
##   <dbl> <dbl> <dbl>
## 1 0.509 0.465 0.423
```

Programación funcional



Licencia CC BY