INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Discovering the dataset

Colin Fay
Data Scientist & R Hacker at ThinkR

The dataset

`rstudioconf`: a list of 5055 tweets

```
length(rstudioconf)
[1] 5055

length(rstudioconf[[1]])
[1] 31

library(purrr)
vec_depth(rstudioconf)
[1] 4
```

Source : ThinkR-open/datasets

JSON — A typical API output

JSON == nested lists

```
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                { "value": "New", "onclick": "CreateNewDoc" },
                { "value": "Open", "onclick": "OpenDoc" },
                { "value": "Close", "onclick": "CloseDoc" }
            ]
        }
    }
}
```

Predicate refresher

We've seen:

- `map_*()`
- `discard()`
- `keep()`

Predicate functionals:

- Take an element & a predicate
- Use the predicate on the element

keep() & discard()

`keep()` the elements that meet a condition:

```
keep(1:10, ~ .x < 5)
[1] 1 2 3 4
```

`discard()` the elements that meet a condition:

```
discard(1:10, ~ .x < 5)
[1]  5  6  7  8  9 10
```

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Extracting information from the dataset

Colin Fay
Data Scientist & R Hacker @ ThinkR

Function manipulation

We've seen:

- `partial()`
- `compose()`

partial()

`partial()` prefills a function:

```
sum_no_na <- partial(sum, na.rm = TRUE)

map_dbl(airquality, sum_no_na)
```

```
  Ozone  Solar.R     Wind     Temp    Month      Day
 4887.0  27146.0   1523.5  11916.0   1070.0   2418.0
```

compose()

`compose()` a function:

```
rounded_sum <- compose(round, sum_no_na)

map_dbl(airquality, rounded_sum)
```

```
  Ozone Solar.R    Wind    Temp   Month     Day
   4887   27146    1524   11916    1070    2418
```

Cleaner lists

Removing `NULL`:

- `compact()`

Removing one level:

- `flatten()`

compact()

```
l <- list(NULL, 1, 2, 3, NULL)
l
```

```
[[1]]
NULL

[[2]]
[1] 1

[[3]]
[1] 2

[[4]]
[1] 3

[[5]]
NULL
```

```
l <- list(NULL, 1, 2, 3, NULL)
compact(l)
```

```
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3
```

flatten()

```
my_list <- list(
  list(a = 1),
  list(b = 2)
)

my_list
```

```
[[1]]
[[1]]$a
[1] 1


[[2]]
[[2]]$b
[1] 2
```

```
my_list <- list(
  list(a = 1),
  list(b = 2)
)

flatten(my_list)
```

```
$a
[1] 1

$b
[1] 2
```

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Manipulating URLs

Colin Fay
Data Scientist & R Hacker at ThinkR

Creating mappers

`as_mapper()`

```
library(purrr)

mult <- as_mapper(~ .x * 2)

map(
  list(airquality, mtcars),
  mult
  )
[[1]]
    Ozone Solar.R Wind Temp Month Day
1      82     380 14.8  134    10   2
2      72     236 16.0  144    10   4
3      24     298 25.2  148    10   6
4      36     626 23.0  124    10   8
5      NA      NA 28.6  112    10  10
6      56      NA 29.8  132    10  12
...
```

stringr::str_detect()

Pattern detection:

```
library(stringr)

lyrics <- c("Is this the real life?",
            "Is this just fantasy?",
            "Caught in a landslide",
            "No escape from reality")

str_detect(a, "life")
```

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

Side note on logicals

Summing logicals:

```
sum(FALSE, TRUE, TRUE, FALSE, TRUE)
```

```
[1] 3
```

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Identifying influencers

Colin Fay
Data Scientist at ThinkR

map_at()

`map_at()` a specific place:

```
my_list <- list(
  a = 1:10,
  b = 1:100,
  c = 12
)

map_at(.x = my_list, .at = "b", .f = sum)
```

```
$a
 [1]  1  2  3  4  5  6  7  8  9 10

$b
[1] 5050

$c
[1] 12
```

negate()

Predicate inversion:

```
not_character <- negate(is.character)
my_list <- list(
  a = 1:10,
  b = "a",
  c = iris
)

map(my_list, not_character)
```

```
$a
[1] TRUE

$b
[1] FALSE

$c
[1] TRUE
```

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Congratulations!

Colin Fay
Data-Scientist & R-Hacker @ ThinkR

Function creation

Lambda functions:

```
map(1:5, ~ .x*10)
[[1]]
[1] 10

[[2]]
[1] 20

[[3]]
[1] 30

[[4]]
[1] 40

[[5]]
[1] 50
```

Reusable mappers:

```
ten_times <- as_mapper(~ .x * 10)
map(1:5, ten_times)
[[1]]
[1] 10

[[2]]
[1] 20

[[3]]
[1] 30

[[4]]
[1] 40

[[5]]
[1] 50
```

Function manipulation

Functionals:

- `map()` & friends
- `keep()` & `discard()`
- `some()` & `every()`

Function operators:

- `safely()` & `possibly()`
- `partial()`
- `compose()`
- `negate()`

Cleaner code

```
library(purrr)
rounded_mean <- compose(
  partial(round, digits = 1),
  partial(mean, trim = 2, na.rm = TRUE)
  )

map(
  list(airquality, mtcars),
  ~ map_dbl(.x, rounded_mean)
)

## [[1]]
##   Ozone Solar.R    Wind    Temp   Month     Day
##    31.5   205.0     9.7    79.0     7.0    16.0
##
## [[2]]
##   mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##  19.2   6.0 196.3 123.0   3.7   3.3  17.7   0.0   0.0   4.0   2.0
```

Where to go next?

- Go and try `purrr` in the wild ;)
- DataCamp tidyverse courses
- Advanced R
- Continue to explore `purrr`

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

See you soon!