Advanced R by Hadley Wickham

Chapter 9: Functionals

Jake Riley

Oct. 1 2020

Contents

- 9.2 `map()`
- 9.3 `purrr` style
- 9.4 `map_*` variants
- 9.5 `reduce()` family
- Some functions that weren't covered
- Example application
- Well be using our old friend:

```
library(tidyverse) # focus on purrr
```

What are functionals

- alternatives to loops
- a functional is better than a `for` loop is better than `while` is better than `repeat`
- don't shoehorn it in if a loop is the easiest answer
    - ex: iterating over `eval()`

Benefits

- encourages function logic to be separated from iteration logic
- can collapse into vectors/data frames easily

9.2 `map()`

- `purrr::map()` is equivalent to `lapply()`
- returns a list and is the most general
- the length of the input == the length of the output

```
map(
  .x, # list/vector to iterate over
  .f, # function to use
  ... # arguments passed to .f
)
```

```
triple <- function(x) x * 3
map(1:3, triple)
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 6
##
## [[3]]
## [1] 9
```

9.2.1 Atomic vectors

- has 4 variants to return atomic vectors
  - `map_chr()`
  - `map_dbl()`
  - `map_int()`
  - `map_lgl()`

```
triple <- function(x) x * 3
map(1:3, triple)
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 6
##
## [[3]]
## [1] 9
```

```
map_dbl(1:3, triple)
```

```
## [1] 3 6 9
```

```
map_lgl(c(1, NA, 3), is.na)
```

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

9.2.2 Anonymous functions and shortcuts

- anonymous functions

```
map_dbl(mtcars, function(x) mean(x, na.rm = TRUE)) %>%
  head()
```

```
##        mpg        cyl       disp         hp       drat         wt
##  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250
```

- the "twiddle" uses a tilda `~` to set a formula
  - can use `.x` to reference the input `map(.x = ..., .f = )`

```
map_dbl(mtcars, ~ mean(.x, na.rm = TRUE))
```

- can be simplified further as

```
map_dbl(mtcars, mean, na.rm = TRUE)
```

9.3 `purrr` style

```
mtcars %>%
  map(head, 20) %>% # pull first 20 of each column
  map_dbl(mean) %>% # mean of each vector
  head()
```

```
##       mpg       cyl      disp        hp      drat        wt
##  20.13000   6.20000 233.93000 136.20000   3.54500   3.39845
```

An example from `tidytuesday`

```
tt <- tidytuesdayR::tt_load("2020-06-30")

# filter data & exclude columns with lost of nulls
list_df <-
  map(
    .x = tt[1:3],
    .f =
      ~ .x %>%
      filter(issue <= 152 | issue > 200) %>%
      mutate(timeframe = ifelse(issue <= 152, "first 5 years", "last 5 years")) %>%
      select_if(~mean(is.na(.)) < 0.2)
  )

# write to global environment
iwalk(
  .x = list_df,
  .f = ~ assign(x = .y, value = .x, envir = globalenv())
)
```

9.4 `map_*()` variants

There are many variants

	List	Atomic	Same type	Nothing
One argument	`map()`	`map_lgl()`, ...	`modify()`	`walk()`
Two arguments	`map2()`	`map2_lgl()`, ...	`modify2()`	`walk2()`
One argument + index	`imap()`	`imap_lgl()`, ...	`imodify()`	`iwalk()`
N arguments	`pmap()`	`pmap_lgl()`, ...	—	`pwalk()`

9.4.2 `map2_*()`

- raise each value `.x` by 2

```
map_dbl(
  .x = 1:5,
  .f = function(x) x ^ 2
)
```

```
## [1]  1  4  9 16 25
```

- raise each value `.x` by another value `.y`

```
map2_dbl(
  .x = 1:5,
  .y = 2:6,
  .f = ~ (.x ^ .y)
)
```

```
## [1]     1     8    81  1024 15625
```

9.4.3 `walk()`

- for steps that don't require "captured results" like generating plots, `write.csv()` or `ggsave()`, `map()` will print more info than you may want

```
map(1:3, ~cat(.x, "\n"))
```

```
## 1
## 2
## 3
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
```

- for these cases, use `walk()` instead

```
walk(1:3, ~cat(.x, "\n"))
```

```
## 1
## 2
## 3
```

9.4.4 `imap()`

- `imap()` is like `map2()` except that `.y` is derived from `names(.x)` if named or `seq_along(.x)` if not
- These two produce the same result

```
map2_chr(.x = mtcars, .y = names(mtcars), .f = ~ paste(.y, "has a mean of", round(mean(.x), 1)))
```

```
imap_chr(.x = mtcars, .f = ~ paste(.y, "has a mean of", round(mean(.x), 1))) %>%
  head()
```

```
##                         mpg                         cyl
##   "mpg has a mean of 20.1"    "cyl has a mean of 6.2"
##                        disp                          hp
## "disp has a mean of 230.7"   "hp has a mean of 146.7"
##                        drat                          wt
##   "drat has a mean of 3.6"     "wt has a mean of 3.2"
```

9.4.4 `imap()`

- When .x isn't named, the index will be used

```
set.seed(1234)

x <- map(1:3, ~ sample(100, 3))

imap_chr(
  .x = x,
  .f = ~ paste("The highest value in set", .y, "is", max(.x))
)
```

```
## [1] "The highest value in set 1 is 80" "The highest value in set 2 is 38"
## [3] "The highest value in set 3 is 98"
```

9.4.5 `pmap()`

- you can pass a named list or dataframe as arguments to a function
- for example `runif()` has the parameters `n`, `min` and `max`

```
params <- tibble::tribble(
  ~ n, ~ min, ~ max,
   1L,     1,    10,
   2L,    10,   100,
   3L,   100,  1000
)

pmap(params, runif)
```

- could also be

```
list(
  n = 1:3,
  min = 10 ^ (0:2),
  max = 10 ^ (1:3)
) %>%
  pmap(runif)
```

```
## [[1]]
## [1] 9.310901
##
## [[2]]
## [1] 36.30843 85.35661
##
## [[3]]
## [1] 357.6010 340.1387 268.0505
```

9.5 `reduce()` family

- you can use `reduce()` when the result should keep updating

```
set.seed(1234)

map(1:4, ~ sample(1:10, 15, replace = TRUE)) %>%
  reduce(intersect)
```

```
## [1] 10  8
```

- to see all intermediate steps, use `accumulate()`

```
set.seed(1234)

map(1:4, ~ sample(1:10, 15, replace = TRUE)) %>%
  accumulate(intersect)
```

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

Not covered: `map_df*()` variants

- `map_dfr()` = row bind the results
- `map_dfc()` = column bind the results

```
col_stats <- function(n) {
  head(mtcars, n) %>%
    summarise_all(mean) %>%
    mutate_all(floor) %>%
    mutate(n = paste("N =", n))
}

map((1:2) * 10, col_stats)
```

```
## [[1]]
##   mpg cyl disp  hp drat wt qsec vs am gear carb      n
## 1  20   5  208 122    3  3   18  0  0    3    2 N = 10
##
## [[2]]
##   mpg cyl disp  hp drat wt qsec vs am gear carb      n
## 1  20   6  233 136    3  3   18  0  0    3    2 N = 20
```

```
map_dfr((1:2) * 10, col_stats)
```

```
##   mpg cyl disp  hp drat wt qsec vs am gear carb      n
## 1  20   5  208 122    3  3   18  0  0    3    2 N = 10
## 2  20   6  233 136    3  3   18  0  0    3    2 N = 20
```

Not covered: `pluck()`

- `pluck()` will pull a single element from a list

```
my_list <- list(
  1:3,
  10 + (1:5),
  20 + (1:10)
)

pluck(my_list, 1)
```

```
## [1] 1 2 3
```

```
map(my_list, pluck, 1)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 11
##
## [[3]]
## [1] 21
```

```
map_dbl(my_list, pluck, 1)
```

```
## [1]  1 11 21
```

Not covered: `flatten()`

- `flatten()` will turn a list of lists into a simpler vector

```
my_list <-
  list(
    a = 1:3,
    b = list(1:3)
  )

map_if(my_list, is.list, pluck)
```

```
## $a
## [1] 1 2 3
##
## $b
## $b[[1]]
## [1] 1 2 3
```

```
map_if(my_list, is.list, flatten_int)
```

```
## $a
## [1] 1 2 3
##
## $b
## [1] 1 2 3
```

```
map_if(my_list, is.list, flatten_int) %>%
  flatten_int()
```

```
## [1] 1 2 3 1 2 3
```

Application

- parsing JSON or XML

```
library(repurrrsive)

repos <-
  repurrrsive::gh_repos %>%
  purrr::flatten()


str(repos[[1]], list.len = 10)
```

```
## List of 68
##  $ id                : int 61160198
##  $ name              : chr "after"
##  $ full_name         : chr "gaborcsardi/after"
##  $ owner             :List of 17
##   ..$ login               : chr "gaborcsardi"
##   ..$ id                  : int 660288
##   ..$ avatar_url          : chr "https://avatars.githubusercontent.com/u/660288?v=3"
##   ..$ gravatar_id         : chr ""
##   ..$ url                 : chr "https://api.github.com/users/gaborcsardi"
##   ..$ html_url            : chr "https://github.com/gaborcsardi"
##   ..$ followers_url       : chr "https://api.github.com/users/gaborcsardi/followers"
##   ..$ following_url       : chr "https://api.github.com/users/gaborcsardi/following{/other_user}"
##   ..$ gists_url           : chr "https://api.github.com/users/gaborcsardi/gists{/gist_id}"
##   ..$ starred_url         : chr "https://api.github.com/users/gaborcsardi/starred{/owner}{/repo}"
##   .. [list output truncated]
##  $ private           : logi FALSE
##  $ html_url          : chr "https://github.com/gaborcsardi/after"
##  $ description       : chr "Run Code in the Background"
##  $ fork              : logi FALSE
##  $ url               : chr "https://api.github.com/repos/gaborcsardi/after"
##  $ forks_url         : chr "https://api.github.com/repos/gaborcsardi/after/forks"
##   [list output truncated]
```

Application

```
tibble(
  repo_id = map_int(repos, pluck, "id"),
  repo_name = map_chr(repos, pluck, "name"),
  owner_name =
    map(repos, pluck, "owner") %>%
    map_chr(pluck, "login"),
  size = map_int(repos, pluck, "size"),
  issues = map_int(repos, pluck, "open_issues_count"),
  watchers = map_int(repos, pluck, "watchers_count"),
  forks = map_int(repos, pluck, "forks_count"),
  created_at = map_chr(repos, pluck, "created_at")
) %>%
  arrange(desc(watchers))
```

```
## # A tibble: 176 x 8
##     repo_id repo_name     owner_name   size issues watchers  forks created_at
##       <int> <chr>         <chr>       <int>  <int>    <int>  <int> <chr>
##  1 14204342 datasharing   jtleek        547    399     3558 161881 2013-11-07T1~
##  2  7751816 dataanalysis  jtleek     156389      5      616    605 2013-01-22T1~
##  3 55175084 tidytext      juliasilge  12355      5      265     33 2016-03-31T1~
##  4 24905212 genomicspape~ jtleek        197      1      211     93 2014-10-07T1~
##  5 20234724 capitalIn21s~ jtleek     374812      0      186    118 2014-05-27T2~
##  6 42663787 firstpaper    jtleek         45      0      133     37 2015-09-17T1~
##  7 40200563 maxygen       gaborcsar~    140      2       59      2 2015-08-04T1~
##  8 36437287 careerplanni~ jtleek        128      0       59     18 2015-05-28T1~
##  9 24343686 crayon        gaborcsar~    752      7       52      9 2014-09-22T2~
## 10 39025267 genstats      jtleek      15870      3       42    105 2015-07-13T1~
## # ... with 166 more rows
```