purrr functions 2

DanielH

August 15, 2018

In this part we will consider a list of lists from the package `repurrrsive`.

The `gotchars_list` is a list of 30 lists composed of 18 elements(vectors) each

`map()`, `pluck()`

We want to extract an element from a list.

```
# extract by name
got_chars %>%
  map("id") %>%
  flatten_int()
```

```
##  [1] 1022 1052 1074 1109 1166 1267 1295  130 1303 1319  148  149  150  168
## [15] 2066  208  216  232  238  339  529  576  583   60  605  743  751  844
## [29]  954  957
```

```
# extract by position
got_chars %>%
  map(2) %>%
  flatten_int()
```

```
##  [1] 1022 1052 1074 1109 1166 1267 1295  130 1303 1319  148  149  150  168
## [15] 2066  208  216  232  238  339  529  576  583   60  605  743  751  844
## [29]  954  957
```

```
# using 'pluck' by name
got_chars %>%
  modify_depth(1, pluck, "id") %>%
  flatten_int()
```

```
##  [1] 1022 1052 1074 1109 1166 1267 1295  130 1303 1319  148  149  150  168
## [15] 2066  208  216  232  238  339  529  576  583   60  605  743  751  844
## [29]  954  957
```

```
# using 'pluck' by position
got_chars %>%
  modify_depth(1, pluck, 2) %>%
  flatten_int()
```

```
##  [1] 1022 1052 1074 1109 1166 1267 1295  130 1303 1319  148  149  150  168
## [15] 2066  208  216  232  238  339  529  576  583   60  605  743  751  844
## [29]  954  957
```

```
# using magrittr::extract()
got_chars %>%
  modify_depth(1, extract, "id") %>%
  flatten() %>%
  flatten_int()
```

```
##  [1] 1022 1052 1074 1109 1166 1267 1295  130 1303 1319  148  149  150  168
## [15] 2066  208  216  232  238  339  529  576  583   60  605  743  751  844
## [29]  954  957
```

explore the list

```
# check if elements of the 30 lists are all length 18
got_chars %>%
  every(~length(.x) == 18)
```

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

```
# name the list
list_names <-
  got_chars %>%
  modify_depth(1, pluck, "name") %>%
  flatten_chr()

got_chars <-
  got_chars %>%
  set_names(list_names)

# explore the list
got_chars %>%
  sample(2) %>%
  str(max.level = 2, list.len = 3)
```

```
## List of 2
##  $ Sansa Stark:List of 18
##   ..$ url        : chr "https://www.anapioficeandfire.com/api/characters/957"
##   ..$ id         : int 957
##   ..$ name       : chr "Sansa Stark"
##   .. [list output truncated]
##  $ Varamyr    :List of 18
##   ..$ url        : chr "https://www.anapioficeandfire.com/api/characters/2066"
##   ..$ id         : int 2066
##   ..$ name       : chr "Varamyr"
##   .. [list output truncated]
```

```
# extract a named element
got_chars %>%
  extract(c("Theon Greyjoy", "Will")) %>%
  str(list.len = 3)
```

```
## List of 2
##  $ Theon Greyjoy:List of 18
##   ..$ url        : chr "https://www.anapioficeandfire.com/api/characters/1022"
##   ..$ id         : int 1022
##   ..$ name       : chr "Theon Greyjoy"
##   .. [list output truncated]
##  $ Will         :List of 18
##   ..$ url        : chr "https://www.anapioficeandfire.com/api/characters/1109"
##   ..$ id         : int 1109
##   ..$ name       : chr "Will"
##   .. [list output truncated]
```

```r
# extract multiple elements
got_chars[2:4] %>%
  map(extract, c("id", "culture")) %>%  # return a list of 3 vector lists
  map(str, max.level = 1) %>%
  compact()  # remove elements that are NULL
```

```
## List of 2
##  $ id     : int 1052
##  $ culture: chr ""
## List of 2
##  $ id     : int 1074
##  $ culture: chr "Ironborn"
## List of 2
##  $ id     : int 1109
##  $ culture: chr ""
```

```
## named list()
```

has_element()

Here we want to see which(named) element of the list contains an element "Bronson Webb" and which one contains "Maester"

```r
got_chars %>%
  modify_depth(1, has_element, "Bronson Webb") %>%
  keep(~.x == TRUE)
```

```
## $Will
## [1] TRUE
```

```r
got_chars %>%
  modify_depth(1, has_element, "Maester") %>%
  keep(~.x == TRUE)
```

```
## $Cressen
## [1] TRUE
```

It's the 'Will' element.

head_while()

```r
got_chars %>%
  modify_depth(1, has_element, "Ironborn") %>%
  keep(~. == TRUE)
```

```
## $`Theon Greyjoy`
## [1] TRUE
##
## $`Victarion Greyjoy`
## [1] TRUE
##
## $`Asha Greyjoy`
## [1] TRUE
##
```

```
## $`Aeron Greyjoy`
## [1] TRUE
```

```
got_chars %>%
  modify_depth(1, has_element, "Ironborn") %>%
  head_while(~. == TRUE)
```

```
## $`Theon Greyjoy`
## [1] TRUE
```

```
got_chars %>%
  modify_depth(1, has_element, "Norvoshi") %>%
  keep(~. == TRUE)
```

```
## $`Areo Hotah`
## [1] TRUE
```

```
got_chars %>%
  modify_depth(1, has_element, "Norvoshi") %>%
  head_while(~. == FALSE)
```

```
## $`Theon Greyjoy`
## [1] FALSE
##
## $`Tyrion Lannister`
## [1] FALSE
##
## $`Victarion Greyjoy`
## [1] FALSE
##
## $Will
## [1] FALSE
```

```
got_chars %>%
  modify_depth(1, has_element, "Ironborn") %>%
  tail_while(~. == FALSE)
```

```
## $`Kevan Lannister`
## [1] FALSE
##
## $Melisandre
## [1] FALSE
##
## $`Merrett Frey`
## [1] FALSE
##
## $`Quentyn Martell`
## [1] FALSE
##
## $`Samwell Tarly`
## [1] FALSE
##
## $`Sansa Stark`
## [1] FALSE
```

map_if(), map_at()

```
# map_if
got_chars %>%
  modify_depth(1, map_if, is_integer, sqrt) %>%     #sqrt of integer elements
  modify_depth(1, extract, "id") %>%     # extract the square roots, 'id'
  modify_depth(1, flatten_dbl) %>%     # remove one level of hierarchy
  unname() %>%    # remove list names
  as_vector() %>%    # convert to vector
  round(2)
```

```
##  [1] 31.97 32.43 32.77 33.30 34.15 35.59 35.99 11.40 36.10 36.32 12.17
## [12] 12.21 12.25 12.96 45.45 14.42 14.70 15.23 15.43 18.41 23.00 24.00
## [23] 24.15  7.75 24.60 27.26 27.40 29.05 30.89 30.94
```

```
# map_to

# 1
got_chars %>%
  modify_depth(1, map_at, "gender", str_count) %>%    #count gender characters
  modify_depth(1, extract, "gender") %>%     # extract the gender elements
  modify_depth(1, flatten_int) %>%     # remove one level of hierarchy
  unname() %>%      # remove list names
  as_vector() %>%     # convert to vector
  map_chr(~if_else(. == 4, "female", "male"))    # back to character vec
```

```
##  [1] "female" "female" "female" "female" "female" "female" "female"
##  [8] "male"   "male"   "female" "male"   "female" "male"   "female"
## [15] "female" "female" "male"   "male"   "male"   "female" "female"
## [22] "female" "female" "female" "female" "male"   "female" "female"
## [29] "female" "male"
```

```
# 2
got_chars[1:5] %>%
  modify_depth(1, map_at, "id", ~.x + 100) %>%  # +100 where TRUE
  map("id") %>%
  flatten_dbl()
```

```
## [1] 1122 1152 1174 1209 1266
```

from list to dataframe

```
# create tibble
got_chars_tibble <-
  got_chars %>% {
  tibble(name = map_chr(., "name"),
         culture = map_chr(., "culture"),
         gender = map_chr(., "gender"),
         id = map_int(., "id"),
         born = map_chr(., "born"),
         alive = map_lgl(., "alive"))
  }
```

```
# set NAs in the culture column
got_chars_tibble <-
  got_chars_tibble %>%
  mutate(culture = parse_character(culture, na = c("", "NA" )))
```

map2()

If we need to map a function over two vectors or lists in parallel, we can use `map2()` for that.

Here is the usage:

```
# create list 1
nms <-
  got_chars %>%
  map("name")
# create list 2
brn <-
  got_chars %>%
  map("born")

# map paste(...) in parallel using map2
nms %>%
  map2(brn, ~paste(.x, "was born", .y)) %>%
  tail(3)
```

```
## $`Quentyn Martell`
## [1] "Quentyn Martell was born In 281 AC, at Sunspear, Dorne"
##
## $`Samwell Tarly`
## [1] "Samwell Tarly was born In 283 AC, at Horn Hill"
##
## $`Sansa Stark`
## [1] "Sansa Stark was born In 286 AC, at Winterfell"
```

pmap()

to map a function over two or more vectors or lists in parallel, we use `pmap()`

```
# create df/tibble
df <-
  got_chars %>%
  {tibble(name = map_chr(., "name"),
          aliases = map(., "aliases"),
          allegiances = map(., "allegiances"))}

# define function
my_fun <-
  function(name, aliases, allegiances) {
    paste(name, "has", length(aliases), "aliases and",
          length(allegiances), "allegiances")
}

# map the function with pmap
```

```r
df %>%
  pmap_chr(my_fun) %>%
  tail(4)
```

```
## [1] "Merrett Frey has 1 aliases and 1 allegiances"
## [2] "Quentyn Martell has 4 aliases and 1 allegiances"
## [3] "Samwell Tarly has 7 aliases and 1 allegiances"
## [4] "Sansa Stark has 3 aliases and 2 allegiances"
```