



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

Working with unnamed lists

Auriel Fournier
Instructor



But first, pipes

```
output <- function_one() %>%  
  function_two()
```

Instead of needing

```
output1 <- function_one()  
  
output <- function_two(output1)
```



Does my list have names?

Without Pipes

```
names(survey_data)
```

```
[1] "LakeErieS" "LakeErieN" "LakeErieW" "LakeErieE"
```

With Pipes

```
survey_data %>%  
  names()
```

```
[1] "LakeErieS" "LakeErieN" "LakeErieW" "LakeErieE"
```



No names? Set some!

```
library(repurrrsive)
data(sw_films)
str(sw_films)
```

```
List of 14
 $ title      : chr "A New Hope"
 $ episode_id : int 4
 $ opening_crawl: chr "It is a period of __truncated__"
 $ director   : chr "George Lucas"
 ...
```

```
sw_films <- sw_films %>%
  set_names(map_chr(sw_films, "title"))

names(sw_films)
```

```
[1] "A New Hope"           "Attack of the Clones"
[3] "The Phantom Menace"   "Revenge of the Sith"
[5] "Return of the Jedi"   "The Empire Strikes Back"
[7] "The Force Awakens"
```



Pipes within map()

```
waterfowl_data  
  
$LakeErieS  
[1] 0 0 10 5  
  
$LakeErieN  
[1] 0 0 1000 5  
  
$LakeErieW  
[1] 10000 0 0 1  
  
$LakeErieE  
[1] 10 10 5 0
```

```
map(waterfowl_data, ~.x %>%  
    sum() %>%  
    log())  
  
$LakeErieS  
[1] 2.70805  
  
$LakeErieN  
[1] 6.912743  
  
$LakeErieW  
[1] 9.21044  
  
$LakeErieE  
[1] 3.218876
```



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

Let's purrr-actice!



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

More map()

Auriel Fournier
Instructor



Simulate data

```
list_of_means
```

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

```
[[2]]  
[1] 2
```

```
[[3]]  
[1] 300
```

```
[[4]]  
[1] 15
```

```
list_of_df <- map(list_of_means,  
  ~data.frame(a=rnorm(mean = .x,  
                      n = 200,  
                      sd = (5/2))))
```

```
head(list_of_df[[1]])
```

```
      a  
1 4.518015  
2 3.915059  
3 5.306956  
4 7.039757  
5 8.609741  
6 1.478696
```


Run linear models

```
str(education_data[[1]])
```

```
List of 2
 $ district_a:List of 2
  ..$ education_level: chr [1:200] "B.S." "K-12" ...
  ..$ income          : num [1:200] 487256 493378 ...
```

```
models <- education_data %>%
  map(~ lm(income ~ education_level, data=.x)) %>%
  map(summary)
```

map_*() flavors

```
map(livingthings,  
    ~.x[["species"]])
```

```
[[1]]  
[1] "Purple Flowers"
```

```
[[2]]  
[1] "Green Grass"
```

```
[[3]]  
[1] "Brown Dog"
```

```
map_chr(livingthings,  
        ~.x[["species"]])
```

```
[1] "Purple Flowers"  
"Green Grass"  
"Brown Dog"
```

```
map_lgl(livingthings,  
        ~.x[["species"]]=="Purple Flowers")
```

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

map_*() flavors with numbers

```
map(bird_measurements,  
    ~.x[["wing length"]])
```

```
$sora  
[1] 75
```

```
$robin  
[1] 12
```

```
map_dbl(bird_measurements,  
         ~.x[["wing length"]])
```

```
sora robin  
75    12
```

```
map_int(bird_measurements,  
         ~.x[["wing length"]])
```

```
sora robin  
75    12
```

```
map(bird_measurements,  
    ~.x[["weight"]])
```

```
$sora  
[1] 96.4
```

```
$robin  
[1] 76.5
```

```
map_dbl(bird_measurements,  
         ~.x[["weight"]])
```

```
sora robin  
96.4 76.5
```

```
map_int(bird_measurements,  
         ~.x[["weight"]])
```

```
Error: Can't coerce element 1  
from a double to a integer
```



map_df()

```
bird_measurements %>%  
  map_df(~ data_frame(weight=.x[["weight"]],  
                      wing_length = .x[["wing length"]]))
```

```
# A tibble: 2 x 2  
  weight wing_length  
  <dbl>      <dbl>  
1   96.4         75  
2   76.5         12
```



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

Let's purrr-actice!



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

map2() and pmap()

Auriel Fournier
Instructor

More complex iterations

```
list_of_means
```

```
[[1]]  
[1] 5  
[[2]]  
[1] 2  
[[3]]  
[1] 300  
[[4]]  
[1] 15
```

```
list_of_sd
```

```
[[1]]  
[1] 0.5  
[[2]]  
[1] 0.01  
[[3]]  
[1] 20  
[[4]]  
[1] 1
```

```
simdata <- map2(list_of_means,  
               list_of_sd,  
               ~data.frame(a = rnorm(mean=.x,  
                                     n=200, sd=.y),  
                           b = rnorm(mean=200,  
                                     n=200, sd=15)))
```

```
head(simdata[[1]])
```

	a	b
1	4.986100	195.1436
2	5.216531	222.7807
3	4.249028	201.0155
4	5.125663	189.3022
5	4.430192	231.3301
6	5.557537	185.3563



What if we didn't use purrr?

```
for(i in list_of_means){  
  for(j in list_of_sd){  
    for(k in list_of_sample_size){  
      num <- 1  
      simdata[[1]] <- rnorm(mean=i, sd=j, n = k)  
      num <- num + 1  
    }  
  }  
}
```




pmap() inputs

```
list_of_means
```

```
[[1]]  
[1] 5  
[[2]]  
[1] 2  
...
```

```
list_of_sd
```

```
[[1]]  
[1] 0.5  
[[2]]  
[1] 0.01  
...
```

```
list_of_samplesize
```

```
[[1]]  
[1] 200  
[[2]]  
[1] 50  
...
```

```
input_list <- list(  
  means = list_of_means,  
  sd = list_of_sd,  
  samplesize = list_of_samplesize)
```

```
str(input_list)
```

```
List of 3
```

```
$ means      :List of 4
```

```
..$ : num 5  
..$ : num 2  
..$ : num 300  
..$ : num 15
```

```
$ sd         :List of 4
```

```
..$ : num 0.5  
..$ : num 0.01  
..$ : num 20  
..$ : num 1
```

```
$ samplesize:List of 4
```

```
..$ : num 200  
..$ : num 50  
..$ : num 500  
..$ : num 100
```



pmap()

```
simdata <- pmap(inputs_list,  
  function(means, sd, samplesize)  
    data.frame(a = rnorm(mean=means,  
                        n=samplesize,  
                        sd=sd) ))  
  
head(simdata[[1]])
```

```
      a  
1 5.862376  
2 5.308204  
3 4.771946  
4 5.173814  
5 4.674113  
6 4.681016
```



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

Let's purrr-actice!