Foundations of Functional Programming with purrr_Simplifying Iteration and Lists With purrr

dizhen

5/12/2020

The power of iteration

```r
# setwd("D:/git/Datacamp--r/Datacamp/_Foundations of Functional Programming with purrr/")
dir <- "data/"

library(readr)
files <- list("data_from_2005.csv","data_from_2004.csv",
              "data_from_2003.csv","data_from_2002.csv",
              "data_from_2001.csv","data_from_2000.csv",
              "data_from_1999.csv","data_from_1998.csv",
              "data_from_1997.csv","data_from_1996.csv",
              "data_from_1995.csv","data_from_1994.csv",
              "data_from_1993.csv","data_from_1992.csv",
              "data_from_1991.csv","data_from_1990.csv")
# Initialize list
all_files <- list()

# For loop to read files into a list
for(i in seq_along(files)){
  all_files[[i]] <- read_csv(file = paste(dir, files[[i]], sep = ""))
}
```

```
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
```

```
# Output size of list object
length(all_files)
```

```
## [1] 16
```

```
# Load purrr library
library(purrr)
files1 <- paste(dir, files,sep = "")
# Use map to iterate
all_files_purrr <- map(files1, read_csv)
```

```
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
## Parsed with column specification:
## cols(
##   years = col_double(),
##   a = col_double(),
##   b = col_double()
## )
```

```r
# Output size of list object
length(all_files_purrr)
```

```
## [1] 16
```

```r
list1 <- c("1","2","3","4")
list_of_df <- list(list1, list1, list1, list1,list1,
                   list1, list1, list1, list1,list1)

# Check the class type of the first element
class(list_of_df[[1]])
```

```
## [1] "character"
```

```r
# Change each element from a character to a number
for(i in seq_along(list_of_df)){
    list_of_df[[i]] <- as.numeric(list_of_df[[i]])
}

# Check the class type of the first element
class(list_of_df[[1]])
```

```
## [1] "numeric"
```

```
# Print out the list
list_of_df
```

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

```
# Check the class type of the first element
class(list_of_df[[1]])
```

```
## [1] "numeric"
```

```
# Change each character element to a number
list_of_df <- map(list_of_df, as.numeric)

# Check the class type of the first element again
class(list_of_df[[1]])
```

```
## [1] "numeric"
```

```
# Print out the list
list_of_df
```

```
## [[1]]
## [1] 1 2 3 4
```

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

Subsetting lists; it's not that hard!

```r
# Load repurrrsive package, to get access to the wesanderson dataset
library(repurrrsive)

# Load wesanderson dataset
data(wesanderson)

# Get structure of first element in wesanderson
str(wesanderson)
```

```
## List of 15
##  $ GrandBudapest : chr [1:4] "#F1BB7B" "#FD6467" "#5B1A18" "#D67236"
##  $ Moonrise1     : chr [1:4] "#F3DF6C" "#CEAB07" "#D5D5D3" "#24281A"
##  $ Royal1        : chr [1:4] "#899DA4" "#C93312" "#FAEFD1" "#DC863B"
##  $ Moonrise2     : chr [1:4] "#798E87" "#C27D38" "#CCC591" "#29211F"
##  $ Cavalcanti    : chr [1:5] "#D8B70A" "#02401B" "#A2A475" "#81A88D" ...
##  $ Royal2        : chr [1:5] "#9A8822" "#F5CDB4" "#F8AFA8" "#FDDDA0" ...
##  $ GrandBudapest2: chr [1:4] "#E6A0C4" "#C6CDF7" "#D8A499" "#7294D4"
##  $ Moonrise3     : chr [1:5] "#85D4E3" "#F4B5BD" "#9C964A" "#CDC08C" ...
##  $ Chevalier     : chr [1:4] "#446455" "#FDD262" "#D3DDDC" "#C7B19C"
##  $ Zissou        : chr [1:5] "#3B9AB2" "#78B7C5" "#EBCC2A" "#E1AF00" ...
##  $ FantasticFox  : chr [1:5] "#DD8D29" "#E2D200" "#46ACC8" "#E58601" ...
##  $ Darjeeling    : chr [1:5] "#FF0000" "#00A08A" "#F2AD00" "#F98400" ...
##  $ Rushmore      : chr [1:5] "#E1BD6D" "#EABE94" "#0B775E" "#35274A" ...
##  $ BottleRocket  : chr [1:7] "#A42820" "#5F5647" "#9B110E" "#3F5151" ...
##  $ Darjeeling2   : chr [1:5] "#ECCBAE" "#046C9A" "#D69C4E" "#ABDDDE" ...
```

```
# Get structure of GrandBudapest element in wesanderson
str(wesanderson$GrandBudapest)
```

```
##  chr [1:4] "#F1BB7B" "#FD6467" "#5B1A18" "#D67236"
```

```
# Third element of the first wesanderson vector
wesanderson[["GrandBudapest"]][3]
```

```
## [1] "#5B1A18"
```

```
# Fourth element of the GrandBudapest wesanderson vector
wesanderson$GrandBudapest[4]
```

```
## [1] "#D67236"
```

```
# Subset the first element of the sw_films data
sw_films[[1]]
```

```
## $title
## [1] "A New Hope"
##
## $episode_id
## [1] 4
##
## $opening_crawl
## [1] "It is a period of civil war.\r\nRebel spaceships, striking\r\nfrom a hidden base, have won\r\ntl
##
## $director
## [1] "George Lucas"
##
## $producer
## [1] "Gary Kurtz, Rick McCallum"
##
## $release_date
## [1] "1977-05-25"
##
## $characters
##  [1] "http://swapi.co/api/people/1/"  "http://swapi.co/api/people/2/"
##  [3] "http://swapi.co/api/people/3/"  "http://swapi.co/api/people/4/"
##  [5] "http://swapi.co/api/people/5/"  "http://swapi.co/api/people/6/"
##  [7] "http://swapi.co/api/people/7/"  "http://swapi.co/api/people/8/"
##  [9] "http://swapi.co/api/people/9/"  "http://swapi.co/api/people/10/"
## [11] "http://swapi.co/api/people/12/" "http://swapi.co/api/people/13/"
## [13] "http://swapi.co/api/people/14/" "http://swapi.co/api/people/15/"
## [15] "http://swapi.co/api/people/16/" "http://swapi.co/api/people/18/"
## [17] "http://swapi.co/api/people/19/" "http://swapi.co/api/people/81/"
##
```

```
## $planets
## [1] "http://swapi.co/api/planets/2/" "http://swapi.co/api/planets/3/"
## [3] "http://swapi.co/api/planets/1/"
##
## $starships
## [1] "http://swapi.co/api/starships/2/"  "http://swapi.co/api/starships/3/"
## [3] "http://swapi.co/api/starships/5/"  "http://swapi.co/api/starships/9/"
## [5] "http://swapi.co/api/starships/10/" "http://swapi.co/api/starships/11/"
## [7] "http://swapi.co/api/starships/12/" "http://swapi.co/api/starships/13/"
##
## $vehicles
## [1] "http://swapi.co/api/vehicles/4/" "http://swapi.co/api/vehicles/6/"
## [3] "http://swapi.co/api/vehicles/7/" "http://swapi.co/api/vehicles/8/"
##
## $species
## [1] "http://swapi.co/api/species/5/" "http://swapi.co/api/species/3/"
## [3] "http://swapi.co/api/species/2/" "http://swapi.co/api/species/1/"
## [5] "http://swapi.co/api/species/4/"
##
## $created
## [1] "2014-12-10T14:23:31.880000Z"
##
## $edited
## [1] "2015-04-11T09:46:52.774897Z"
##
## $url
## [1] "http://swapi.co/api/films/1/"
```

```r
# Subset the first element of the sw_films data, title column
sw_films[[1]]$title
```

```
## [1] "A New Hope"
```

The many flavors of map()

1. map(list, function)
2. map(list, ~function(.x))

```r
# Map over wesanderson, and determine the length of each element
map(wesanderson, ~length(.x))
```

```
## $GrandBudapest
## [1] 4
##
## $Moonrise1
## [1] 4
##
## $Royal1
## [1] 4
##
## $Moonrise2
## [1] 4
##
## $Cavalcanti
## [1] 5
##
## $Royal2
## [1] 5
##
## $GrandBudapest2
## [1] 4
##
## $Moonrise3
## [1] 5
##
## $Chevalier
## [1] 4
##
## $Zissou
## [1] 5
##
## $FantasticFox
## [1] 5
##
## $Darjeeling
## [1] 5
##
## $Rushmore
## [1] 5
##
## $BottleRocket
## [1] 7
##
## $Darjeeling2
## [1] 5
```

```r
# Map over wesanderson and determine the length of each element
map(wesanderson, length)
```

```
## $GrandBudapest
## [1] 4
##
## $Moonrise1
## [1] 4
##
## $Royal1
## [1] 4
##
## $Moonrise2
## [1] 4
##
## $Cavalcanti
## [1] 5
##
## $Royal2
## [1] 5
##
## $GrandBudapest2
## [1] 4
##
## $Moonrise3
## [1] 5
##
## $Chevalier
## [1] 4
##
## $Zissou
## [1] 5
##
## $FantasticFox
## [1] 5
##
## $Darjeeling
## [1] 5
##
## $Rushmore
## [1] 5
##
## $BottleRocket
## [1] 7
##
## $Darjeeling2
## [1] 5
```

```r
# Create a numcolors column and fill with length of each wesanderson element
data.frame(numcolors = map_dbl(wesanderson, ~length(.x)))
```

```
##                numcolors
## GrandBudapest          4
## Moonrise1              4
## Royal1                 4
## Moonrise2              4
## Cavalcanti             5
## Royal2                 5
## GrandBudapest2         4
## Moonrise3              5
## Chevalier              4
## Zissou                 5
## FantasticFox           5
## Darjeeling             5
## Rushmore               5
## BottleRocket           7
## Darjeeling2            5
```