purrr for (R)odeo

Ethan Addicott

11/5/2019

Slides and RMarkdown File

http://github.com/eaddicott/library

Motivating purrr

Iteration is one of the most useful tasks a computer can perform for a researcher.

- Repetition
  - Static
  - Dynamic
- The programming *Rule of 3*
  - Never copy and paste more than twice

Enter purrR

We loop over vectors and dataframes, do some manipulation of the data, and save the results somewhere so often that there is a whole package designed to help us with this common task:

purrR

Thankfully this package is a part of the tidyverse, so we don't need to invoke it separately if we're already in the tidyverse library.

Exercise: Summary Statistics

Your Task: Using your favorite R data (I'm using mtcars), generate column-by-column summary statistics

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1
```

An Aside

Let's take the mean of a vector of numbers from 1 to 5

```
# The mean of a list of numbers
```

Attempt 2

```
output <- summary(mtcars)
output
```

```
##       mpg             cyl             disp               h
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.
##  Median :19.20   Median :6.000   Median :196.3   Median
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.
##       drat             wt             qsec              v
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.
##  Median :3.695   Median :3.325   Median :17.71   Median
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean
```

Attempt 3: Super Common Issue

```
#Preallocate the vector for means
means <- vector("double", ncol(mtcars))
# for loop to take the column by column means
for (i in length(mtcars)){
  means[[i]] <- mean(mtcars[[i]])
}
# see the mean vector
means
```

```
##  [1] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0
## [11] 2.8125
```

```
colnames(mtcars)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "v
## [11] "carb"
```

purrr Solution

```
#library(tidyverse)
means <- vector("double", ncol(mtcars))
means <- map(mtcars,mean)
means
```

```
## $mpg
## [1] 20.09062
##
## $cyl
## [1] 6.1875
##
## $disp
## [1] 230.7219
##
## $hp
```

Survey says...

```
#library(microbenchmark)
mbm <- microbenchmark(
  "loop" = {means <- vector("double", ncol(mtcars))
            for (i in seq_along(mtcars)){
              means[[i]] <- mean(mtcars[[i]])
            }
  },
  "purrR" = {means <- vector("double", ncol(mtcars))
             means <- map(mtcars,mean)
             means <- as.numeric(means)
})
mbm
```

```
## Unit: microseconds
##   expr     min      lq     mean  median      uq     max ne
```

The Details

- purrr functions run in C (read: fast!)
- map() writes to a vector
- There is a family of map functions that can write to a vector rather than a list (faster!)
  - map_dbl()
  - map_chr()
  - map_dfc(), map_dfr()
  - map_lgl()
  - map_int()

Example: map_dbl()

```
microbenchmark("map" = {means <- vector("double", ncol(mtca
                        means <- map(mtcars,mean)
                        means <- as.numeric(means)},
       "map_dbl()" = {means <- vector("double", ncol(mtcar
                        means <- map_dbl(mtcars,mean)})
```

```
## Unit: microseconds
##        expr   min     lq    mean median     uq   max neva
##         map 109.1 111.25 117.240 112.70 119.70 159.1   10
##  map_dbl() 108.9 110.75 120.893 111.65 119.35 278.1   10
```

Only slightly faster, but those microseconds can add up!

Nested Dataframes

A short dive into the tidyverse here to explore some additional benefits in the purrr package.

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1
```

Let's group the data by the number of cylinders and create a nested dataframe.

Nest by Cylinders

Models with Nests

Now we'll run a model (or set of models) over our nested dataframe.

```
#Regressing weight on fuel economy
model_fn <- function(df){
  lm(mpg ~ wt,data = df)
}
#apply model over nested data and save to new model column
 m_df <- nested %>%
  mutate(model = map(data,model_fn))
#take a look
 m_df
```

```
## # A tibble: 3 x 3
##     cyl data                 model
##   <dbl> <list>               <list>
## 1     6 <tibble [7 x 10]>    <lm>
```

map2: Another feature in purrr

- Sick of figuring out how to use an apply function over two objects?
- mapply got you down?
- FEAR NOT!

Using map2

```
mtcars %>% mutate(hp_wt_ratio = map2_dbl(hp,wt, ~ .x / .y))
```

```
##    hp_wt_ratio
## 1     41.98473
## 2     38.26087
## 3     40.08621
## 4     34.21462
## 5     50.87209
## 6     30.34682
```

Resources

- https://r4ds.had.co.nz/iteration.html
- purrR Cheatsheet: https://github.com/rstudio/cheatsheets/blob/master/purrr.pdf
- https://github.com/cwickham/purrr-tutorial
- https://emoriebeck.github.io/R-tutorials/purrr/