purrr for (R)odeo

Ethan Addicott

11/5/2019

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: Column by Column Summary Statistics

```
#mean(mtcars)
#means(mtcars)
```

These won't even run. Not a thing.

Attempt 2

```
output <- summary(mtcars)
means <- output[4,]
means
```

```
##                    mpg                  cyl                  disp
##   "Mean   :20.09  "   "Mean   :6.188  "   "Mean   :230.7  "
##                     hp                 drat                    wt
##   "Mean   :146.7  "   "Mean   :3.597  "   "Mean   :3.217  "
##                   qsec                   vs                   am
##   "Mean   :17.85  "  "Mean   :0.4375  "  "Mean   :0.4062  "
##                   gear                 carb
##   "Mean   :3.688  "   "Mean   :2.812  "
```

Gross

Attempt 3: Super Common Issue

```
means <- vector("double", ncol(mtcars))
for (i in length(mtcars)){
  means[[i]] <- mean(mtcars[[i]])
}
means
```

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

Attempt 3: Redux

```
means <- vector("double", ncol(mtcars))
for (i in seq_along(mtcars)){
  means[[i]] <- mean(mtcars[[i]])
}
means
```

```
##   [1]   20.090625    6.187500 230.721875 146.687500    3.596
##   [7]   17.848750    0.437500    0.406250    3.687500    2.812
```

Better...

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
```

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
##         map 110.201 111.6000 119.214 113.551 117.6010 199
##  map_dbl() 108.700 111.0515 119.606 112.851 116.9505 260
```

Nested Dataframes

```
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
```

```
nested <- mtcars %>%
  group_by(cyl) %>%
  nest()
nested
```

```
## # A tibble: 3 x 2
```

Models with Nests

```
model_fn <- function(df){
  lm(mpg ~ wt,data = df)
}

 m_df <- nested %>%
  mutate(model = map(data,model_fn))

 m_df
```

```
## # A tibble: 3 x 3
##     cyl data                model
##   <dbl> <list>              <list>
## 1     6 <tibble [7 x 10]>   <lm>
## 2     4 <tibble [11 x 10]> <lm>
## 3     8 <tibble [14 x 10]> <lm>
```

View the results

```
results <- m_df$model %>% map(summary)
results
```

```
## [[1]]
##
## Call:
## lm(formula = mpg ~ wt, data = df)
##
## Residuals:
##       1       2       3       4       5       6       7
## -0.1250  0.5840  1.9292 -0.6897  0.3547 -1.0453 -1.0080
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   28.409      4.184   6.789  0.00105 **
```

Also 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
## 7     68.62745
## 8     19.43574
## 9     30.15873
## 10    35.75581
## 11    35.75581
## 12    44.22604
```

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/