Apply Family of Functions

Justin Post

Efficient Code

For loops vs Vectorized Functions

`apply()` Family

- `apply()` family of functions *pretty* fast
- Check `help(apply)`!
  - We'll look at `apply()`, `sapply()`, `lapply()`

`apply()` Family

- `apply()` family of functions *pretty* fast
- Check `help(apply)`!
  - We'll look at `apply()`, `sapply()`, `lapply()`
- Consider our Batting data set

```
library(Lahman)
my_batting <- Batting[, c("playerID", "teamID", "G", "AB", "R", "H", "X2B", "X3B", "HR")] |>
  as_tibble()
my_batting
```

```
## # A tibble: 108,789 x 9
##   playerID  teamID     G    AB     R     H   X2B   X3B    HR
##   <chr>     <fct>  <int> <int> <int> <int> <int> <int> <int>
## 1 abercda01 TRO        1     4     0     0     0     0     0
## 2 addybo01  RC1       25   118    30    32     6     0     0
## 3 allisar01 CL1       29   137    28    40     4     5     0
## 4 allisdo01 WS3       27   133    28    44    10     2     2
## 5 ansonca01 RC1       25   120    29    39    11     3     0
## # i 108,784 more rows
```

`apply()` Family

- Use `apply()` to find summary for the batting data

```
apply(X = my_batting,
      MARGIN = 2,
      FUN = summary,
      na.rm = TRUE)
```

```
##        playerID    teamID      G           AB          R           H
## Length "108789"    "108789"    "108789"    "108789"    "108789"    "108789"
## Class  "character" "character" "character" "character" "character" "character"
## Mode   "character" "character" "character" "character" "character" "character"
##        X2B         X3B         HR
## Length "108789"    "108789"    "108789"
## Class  "character" "character" "character"
## Mode   "character" "character" "character"
```

`apply()` Family

- Let's try it with just numeric data!

```
batting_summary <- apply(X = my_batting |>
                           select(where(is.numeric)),
                         MARGIN = 2,
                         FUN = summary,
                         na.rm = TRUE)
batting_summary
```

```
##                 G       AB        R         H       X2B       X3B       HR
## Min.      1.00000   0.0000   0.0000   0.00000  0.000000  0.000000  0.00000
## 1st Qu.  12.00000   4.0000   0.0000   0.00000  0.000000  0.000000  0.00000
## Median   34.00000  46.0000   4.0000   8.00000  1.000000  0.000000  0.00000
## Mean     50.74049 139.2413  18.4835  36.38861  6.202024  1.247075  2.85015
## 3rd Qu.  79.00000 224.0000  27.0000  56.00000  9.000000  1.000000  2.00000
## Max.    165.00000 716.0000 198.0000 262.00000 67.000000 36.000000 73.00000
```

Anonymous Functions

- We often use our own custom functions with the `apply()` family
  - Called anonymous functions or lambda functions

Anonymous Functions

- We often use our own custom functions with the `apply()` family
  - Called anonymous functions or lambda functions

```
custom_batting_summary <- apply(X = my_batting |>
                                  select(where(is.numeric)),
                                MARGIN = 2,
                                FUN = function(x){
                                  temp <- c(mean(x), sd(x))
                                  names(temp) <- c("mean", "sd")
                                  temp
                                }
)
custom_batting_summary
```

```
##              G       AB        R        H      X2B      X3B       HR
## mean 50.74049 139.2413 18.48350 36.38861 6.202024 1.247075 2.850150
## sd   46.88959 183.6021 28.04323 52.18888 9.627314 2.595462 6.368678
```

Anonymous Functions

- Anonymous functions can take other arguments

```
custom_batting_summary <- apply(X = my_batting |>
                                  select(where(is.numeric)),
                                MARGIN = 2,
                                FUN = function(x, trim){
                                  temp <- c(mean(x, trim), sd(x))
                                  names(temp) <- c("mean", "sd")
                                  temp
                                },
                                trim = 0.1
)
custom_batting_summary
```

```
##              G       AB        R        H      X2B       X3B       HR
## mean 44.74459 105.1568 12.30237 25.69437 4.058495 0.5909368 1.168660
## sd   46.88959 183.6021 28.04323 52.18888 9.627314 2.5954618 6.368678
```

`lapply()`

- Use `lapply()` to apply function to lists
- Obtain a list object

```
set.seed(10)
my_list <- list(rnorm(100), runif(10), rgamma(40, shape = 1, rate = 1))
```

`lapply()`

- Apply `mean()` function to each list element

```
lapply(X = my_list, FUN = mean)
```

```
## [[1]]
## [1] -0.1365489
##
## [[2]]
## [1] 0.5997619
##
## [[3]]
## [1] 1.108209
```

`lapply()`

- To give additional arguments to `FUN` we add them on afterward

```
lapply(X = my_list, FUN = mean, trim = 0.1, na.rm = TRUE)
```

```
## [[1]]
## [1] -0.1359629
##
## [[2]]
## [1] 0.6062252
##
## [[3]]
## [1] 0.9563087
```

sapply()

- Similar function but it attempts to simplify when possible

```
sapply(X = my_list, FUN = mean, trim = 0.1, na.rm = TRUE)
```

```
## [1] -0.1359629  0.6062252  0.9563087
```

Recap!

- Vectorized functions fast!
- `apply()` family is sort of vectorized
- `lapply()` and `sapply()` to apply a function to a list
- `aggregate()`, `replicate()`, `tapply()` `vapply()`, and `mapply()` also exist!