ITERATIVE FUNCTIONS

Olivia Johnson

Adelaide Code Club

28 July 2022

OUTLINE FOR TODAY

- Recap of iterative functions
- For loops
- If statments
- Apply functions

QUICK RECAP

- **glue(**“text and **{**values to iterates over**}**”**)**
  - year <- 1998:2022
  - year_searches <- glue(“wombats AND {year}[PDAT]”)

QUICK RECAP

- **glue(**“text and **{**values to iterates over**}**”**)**
  - year <- 1998:2022
  - year_searches <- glue(“wombats AND {year}[PDAT]”)

- **map_dbl(.x =** things to iterate over, **.f=**function or formula (need ~ before function))
  - map_dbl(.x=year_searches, .f=~entrez_search(db=“pubmed”, term= .x))

FOR LOOPS

- Most basic iterative function
- Can be slow
- ```
  for (i in sequence){
      statement
  }
  ```
  ```

FOR LOOP EXAMPLES

```
for (i in 1:10){
        print(i)
}
```

FOR LOOP EXAMPLES

```
for (i in 1:10){
        print(i)
}


groups <- c(1:25)
all_groups=NULL
for (g in groups){
        parameters <- fread(file=paste0("~/data/parameters_”, g, “.txt"))
        all_groups = cbind(all_groups, parameters)
}
```

NESTED LOOPS

```
for (x in x_vals){
 for (y in y_vals){
  print(paste("x =", x, ", y =", y))
 }
}
```

IF ELSE STATEMENTS

- If statements have the same syntax
- Can leave out else if just want to use the if statement

```
if (condition){
    statement
} else {
    statement
}
```

APPLY FUNCTIONS

- Base iterative functions
- Often used as faster than a for loop
- **`apply(`**`x, MARGIN (1 for rows, 2 for columns), function`**`)`**
  - For dataframe or matrix

APPLY FUNCTIONS

- Base iterative functions
- Often used as faster than a for loop
- **`apply(x, MARGIN (1 for rows, 2 for columns), function)`**
  - For dataframe or matrix
- **`lapply(x, function)`**
  - Output is a list

APPLY FUNCTIONS

- Base iterative functions
- Often used as faster than a for loop
- **`apply(x, MARGIN (1 for rows, 2 for columns), function)`**
  - For dataframe or matrix
- **`lapply(x, function)`**
  - Output is a **l**ist
- **`sapply(x, function)`**
  - **S**implified, output is a vector.

EXAMPLE - APPLY

```
> df
  x y
1 1 5
2 2 6
3 3 7
4 4 8
> apply(df, MARGIN = 1, sum)  ## will apply sum across rows
[1]  6  8 10 12
>
> apply(df, MARGIN=2, sum)  ##will apply sum along columns
 x  y
10 26
```

EXAMPLE - LAPPLY

```
> animals = c("KANGAROO", "WOMBAT", "BILBY")
> animals
[1] "KANGAROO" "WOMBAT"   "BILBY"
> lapply(animals, tolower)
[[1]]
[1] "kangaroo"

[[2]]
[1] "wombat"

[[3]]
[1] "bilby"
```

EXAMPLE - SAPPLY

```
> sapply(animals, tolower)
  KANGAROO      WOMBAT       BILBY
"kangaroo"    "wombat"     "bilby"
> sapply(df, sum)
 x  y
10 26
```

EXERCISE TIME!

SUMMARY

For loops – the most basic iterative function

- `for ( x in sequence ) { statement }`

If else statements can be useful in iterative functions

- `if ( condition ) { statement } else { statement }`

Apply functions – often faster than for loops

- `apply(), lapply(), sapply()`