Base R, Part II

Andie Creel

1 Set up: Picking Up from Base R, Part I

In this lecture, we will talk about functions, loops and if statements. We will start basic and work up to using them with a data frame. Let's make the data frame we had constructed during the last lecture (Base R, Part I).

```r
# Create the data frame
myPpl <-    data.frame(
   gender = c("Male", "non-binary","Female"),
   male = c(T, F, F),
   height = c(152, 171.5, 165),
   weight = c(81, 93, 78),
   age = c(42,38,26)
)

# Try referencing one column
myPpl$male # version 1
```

```
## [1]  TRUE FALSE FALSE
```

```r
myPpl[,2] #version 2
```

```
## [1]  TRUE FALSE FALSE
```

```r
# Try referencing one row
myPpl[1]
```

```
##       gender
## 1       Male
## 2 non-binary
## 3     Female
```

```r
# Try referencing one cell
myPpl$height[1] # version 1
```

```
## [1] 152
```

```r
myPpl[1,3] # version 2
```

```
## [1] 152
```

2 Functions

Functions: once you have initialized them, they take in an input, perform a set of operations on them, and then give you some return value.

Example on board

- consider the function: myF(x) { y <- x + 3; return(y)}
- what does myF(3) return? 6

Points:

- These are helpful when you have something that you do often
- Rule of thumb: if you're copying and pasting code 3 times or more, make function
- (I say if you are going to copy past ever, because even if you think it'll only be twice it'll probably be more)
- Recent example for me:
  - wrote a function to take a date and return the season
  - Wrote a function to get kelvin and return Fahrenheit

```
myF <- function(x){
  y <- x - x^2
  return(y)
}

myF(.5)
```

```
## [1] 0.25
```

```
myF(.25)
```

```
## [1] 0.1875
```

```
myF(.7)
```

```
## [1] 0.21
```

3 Loops

- for loops: iterates through a task for a set number of times
- Consider these loops (psuedo code):
  - For (i in 1 through 4) { print i }
  - For (i in 1 through 4) { print i / 4}
- Can be helpful when
  - Iterating through a column of data and do something to each row
  - Construct a new column and want to construct each row by scratch

```r
# Complicated code that is simplified by the loop
print(1)
```

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

```r
print(2)
```

```
## [1] 2
```

```r
print(3)
```

```
## [1] 3
```

```r
print(4)
```

```
## [1] 4
```

```r
# The following loop does the exact same thing
for (i in 1:4){
  print(i)
}
```

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

```r
# more involved
for (i in 1:4){
  print(i/4)
}
```

```
## [1] 0.25
## [1] 0.5
## [1] 0.75
## [1] 1
```

```r
# combining loop and function
for (i in 1:4){

  y = myF(i/4)

  print(y)
}
```

```
## [1] 0.1875
## [1] 0.25
## [1] 0.1875
## [1] 0
```

```r
# Example

# Making a new column
# Version one: Line by line
myPpl$age_new_a[1] <- myPpl$age[1] + 1
myPpl$age_new_a[2] <- myPpl$age[2] + 1
myPpl$age_new_a[3] <- myPpl$age[3] + 1

# Version two: loop
for (i in 1:length(myPpl$age)) {
  myPpl$age_new_b[i] <- myPpl$age[i] + 1 # everyone aged one year
}
```

4 If statements

- sometimes you want to execute a task ONLY if a certain condition is met.
- Open the myPpl df:
  - Our RA did not record men's ages right
  - All men are actually 3 years older than what's recorded
  - What would the correct DF look like?
- If statements let you fix a mistake like this
- Also demonstrates why the Boolean (true/false or indicator) variable is so powerful

```r
# goes through each row and changes age if someone is male
for (i in 1:length(myPpl$male)) {

   if (myPpl$male[i] == TRUE) {
    myPpl$age_new_m[i] <- myPpl$age[i] - 3
  }else{
    myPpl$age_new_m[i] <- myPpl$age[i]
  }
}
```

5 Other R Tutorials

UCLA Getting Started with R

git lab intro

6 Some specific packages

We haven't covered packages yet, but a few good resources for the future

ggplot

dplyr and tidyr