Functions in R

Modified from materials by Heike Hofmann, STAT 579

Functions in R

- Have been using functions a lot, now we want to write them ourselves!
- Idea: avoid repetitive coding (errors will creep in)
- Instead: extract common core, wrap it in a function, make it reusable

Basic Structure

- Name
- Input arguments
  - names,
  - default values
- Body
- Output values

A first function

```
mean <- function(x) {
  return(sum(x)/length(x))
}
!
mean(1:15)
mean(c(1:15, NA))
!
mean <- function(x, na.rm=F) {
# if na.rm is set to true, we would like to delete the missing values from x
  return(sum(x)/length(x))
}
!
```

Conditionals

- if (condition) {
  statement

  } else {
  statement

  }

- condition is a length one logical value, i.e. either TRUE or FALSE
- use & and | to combine several conditions
- ! negates condition

A first function

```
mean <- function(x, na.rm=FALSE) {
  if (na.rm==TRUE) x <- na.omit(x)
  return(sum(x)/length(x))
}
!
mean(1:15)
mean(c(1:15, NA), na.rm=T)
```

Function `mean`

- Name: `mean`
- Input arguments `x, na.rm=T`
  - names,
  - default values
- Body `if(na.rm) x <- na.omit(x)`
- Output values `return(sum(x)/length(x))`

Function writing

- Start simple, then extend
- Test out each step of the way
- Don’t try too much at once

Your Turn

Similar to what we did with mean, write a function sd that computes the standard deviation of variable x ‘from scratch’. Include a parameter na.rm in it.

Good Practice

- Use tabulators to structure blocks of statements
- Build complex blocks of codes step by step, i.e. try with single state first, try to generalize
- `# write comments!`

Testing

- Always test the functions you've written!
- Even better: let somebody else test them for you
- Switch seats with your neighbor, test their function!

Your Turn

Switch seats with your neighbor, test their function!

Your Turn

What do these functions do? Brainstorm some better names for them

```
f1 <- function(string, prefix) {
  substr(string, 1, nchar(prefix)) == prefix
}
f2 <- function(x) {
  if (length(x) <= 1) return(NULL)
  x[-length(x)]
}
f3 <- function(x, y) {
  rep(y, length.out = length(x))
}
```

What to do when things go wrong ...

- Debugging code is an art - it’s hard, much like finding your own errors in writing...
- What you can do yourself:

  - check your code step by step

  - include print statements to check intermediate results (and assumptions)

  - use browser()

  - investigate all warnings

- ask a friend to look over your code

Looking at functions

- For any function loaded into your environment, you can see its code by typing its name in the Console and hitting enter
- Try this with `count`
  - What do you see?
- Try this with your `sd` function
  - What do you see?
- Try this with `mutate`
  - What do you see?

We'll talk more about generic functions later!

Iterations

- Want to run the same block of code multiple times:

```
for (i in players) {
        Block of commands
        Output
}
```

  - Loop or iteration

Iterations

- Want to run the same block of code multiple times:

```
for (i in players) {

        Block of commands

        print(avg)


}
```

  - Loop or iteration

Iterations

- Want to run the same block of code multiple times:

```
for (i in players) {

      player <- subset(baseball, playerID == i)
      avg <- sum(H/AB, na.rm=T)

      print(avg)


}
```

  - Loop or iteration

Code style

- https://style.tidyverse.org/