Module 2_4 - Simulation and Iteration

Keaton Wilson

1/7/2020

Review

Let's take a minute to review as a class what we've talking about, what the problem is and where we're at with it.
1. What is the main problem we're trying to address?
2. What did we cover last time in class?
3. Where did we leave off?

Ultimately, we have a contaminated food supply that we think is making people sick. We've noticed a correlation between the amount of fish in someone's diet and now we're using simulation to compare our expectations for disease in the tanks to what we measured in our small (50 tank sample).

Last time, we worked through building a function to test this, but we realized it has limited utility because it only generates one mean… and this mean changes every time, because of the nature of random draws from a distribution.

Iteration

Today we're going to dive into one of the most important topics in programming, iteration. Something we touched on before, but haven't really explored. The ultimate goal here is to build a way to run through our function multiple times and calculate a 'mean of means', which should give us a better overall picture of what our expected value is.

loops

The most basic way to do iteration is with loops. It's essentially telling R to keep doing things a certain number of times, and then doing something with the outputs for each iteration.

```r
# Let's use a basic example, but one which we would never actually use, as R
# does basic vectorized stuff like this by default.
# Let's generate a logarithm of each value in a vector and output this to a new
# vector.

vec_in = c(2, 6, 8, 9, 5)

for(i in 1:length(vec_in)){
  log(vec_in[i])
}

# Ok... but nothing happened....
# We have to set up some other stuff first
# output vector to feed into
vec_out = c()
for(i in 1:length(vec_in)){
  x = log(vec_in[i])
  vec_out[i] = x
}
vec_out
```

```
## [1] 0.6931472 1.7917595 2.0794415 2.1972246 1.6094379
```

Things to remember about loops:
1. Remember to initialize an object before the loop to feed into
2. Proper syntax - curly braces!
3. Iterating through pieces with the variable you're using (i, in the example above).

Applying loops to our simulation function

We're now going to run our simulation function 100 times. Quick check in - why are we doing this?!

```r
# loading in data
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ---------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
fish_tank_data = read_csv("https://tinyurl.com/tbyskxl")
```

```
## Parsed with column specification:
## cols(
##   tank_id = col_double(),
##   species = col_character(),
##   avg_daily_temp = col_double(),
##   num_fish = col_double(),
##   day_length = col_double(),
##   tank_volume = col_double(),
##   size_day_30 = col_double()
## )
```

```r
# function
fish_sick_simulator = function(mean = NULL, species = "trout", sd = NULL){
  if(species == 'trout'){
    sick_fish_vec = round(rnorm(n=17, mean = mean, sd = sd))
    return(mean(sick_fish_vec)/74)
  } else {
    sick_fish_vec = round(rnorm(n=33, mean = mean, sd = sd))
    return(mean(sick_fish_vec)/100)
  }
}

# Double check our function to make sure it's still working
# We're working on Tilapia first!

fish_sick_simulator(mean = 4, species = "tilapia", sd = 2)
```

```
## [1] 0.04151515
```

```r
# Let's loop it!
sim_vec = c()
for(i in 1:100){
  sim_vec[i] = fish_sick_simulator(mean = 4,
                                   species = "tilapia",
                                   sd = 2)
}

sim_vec
```

```
##   [1] 0.04030303 0.04272727 0.03848485 0.04363636 0.03424242 0.04424242
##   [7] 0.04393939 0.03696970 0.03939394 0.03575758 0.04393939 0.04393939
##  [13] 0.04151515 0.04000000 0.04696970 0.04212121 0.04363636 0.03757576
##  [19] 0.04363636 0.04545455 0.03939394 0.03757576 0.04272727 0.04333333
##  [25] 0.03696970 0.03666667 0.04030303 0.04363636 0.03939394 0.03969697
##  [31] 0.03757576 0.03272727 0.04151515 0.04272727 0.04212121 0.04424242
##  [37] 0.03969697 0.03848485 0.04242424 0.04333333 0.03969697 0.04454545
##  [43] 0.04454545 0.04757576 0.03545455 0.04030303 0.04000000 0.04030303
##  [49] 0.04121212 0.04000000 0.03848485 0.04272727 0.04727273 0.04030303
##  [55] 0.04575758 0.03757576 0.03454545 0.04212121 0.04060606 0.04212121
##  [61] 0.03636364 0.03151515 0.03696970 0.04242424 0.04393939 0.04242424
##  [67] 0.04181818 0.04575758 0.03575758 0.03787879 0.04242424 0.04757576
##  [73] 0.04242424 0.03666667 0.04515152 0.03757576 0.03454545 0.04151515
##  [79] 0.03818182 0.04333333 0.02969697 0.03848485 0.04030303 0.03757576
##  [85] 0.03818182 0.03848485 0.03727273 0.04090909 0.03606061 0.03696970
##  [91] 0.03848485 0.03818182 0.04212121 0.04303030 0.03848485 0.04181818
##  [97] 0.03515152 0.04090909 0.03515152 0.03696970
```

```r
mean(sim_vec)
```

```
## [1] 0.04026667
```

Group Challenge

Construct a similar simulation for trout. Be prepared to answer the following questions:
1. Do the sickness percentages measured by the aquaculture team line up with our expecations? 2. Are there differences between the two species? What does this point to?