STT 301: Practice with functions, loops, and conditionals

Shawn Santo

The exercises below will give you some additional practice at writing functions, loops, and conditional statements. Some of the exercises ask you to write a function or loop despite the fact that there is already a built-in R function or a way to use vectorization. The purpose is to get a deeper understanding of R programming basics and syntax.

Exercise 1

Write a function called `add.one` that takes any numeric value as an input and will add one to that value. The function should return the new value.

Exercise 2

Write a function called `operation.two` that has two arguments. One argument should be a numeric value and the other argument should be one of the four arithmetic operators in the form of a character ("+", "-", "*","/"). The function should add, subtract, multiply, or divide the numeric argument by 2 (depending on what operation the user enters) and return the result.

Exercise 3

Write a function that calculates the mean of a numeric vector `x`, ignoring the `s` smallest and `l` largest values (this is a trimmed mean). Thus, the function should have three arguments, with `s` and `l` being positive integers. Name the function `trimmed.mean`.

Exercise 4

Write a function called `my.grade1`. The function should have four arguments: quiz score, homework score, exam score, and project score. Each of these should be in terms of a percentage (but no % sign should be entered). The function should output the student's grade (as a percentage - again no % sign needs to show) according to the grading structure in the STT 301 syllabus.

Exercise 5

Write a function called `my.grade2`. The function should have one argument - student's grade as a percentage, but no % sign. Thus, the user should enter 95.8 or 74.0 for 95.8% and 74.0%, respectively. The function should output the student's grade on the 4.0 scale. Use the grading scale in the STT 301 syllabus.

Exercise 6

Write a for loop that prints the first 10 uppercase letters of the alphabet. Make use of the `print()` function. `LETTERS` in R will give the vector of uppercase letters.

Exercise 7

Write a while loop that prints the last 10 uppercase letters of the alphabet. Make use of the `print()` function. `LETTERS` in R will give the vector of uppercase letters.

Exercise 8

Create a loop that will store all uppercase and lowercase letters of the alphabet in a vector. The output should be as follows:

"A" "a" "B" "b" "C" "c"…and so on until "Z" "z"

The `LETTERS` and `letters` objects will give the vector of uppercase and lowercase letters of the alphabet, respectively.

Exercise 9

Create a function called `even.count`. The function should take an integer vector as its argument and should output the number of even numbers from the vector.

Exercise 10

Create a function called `even.odd.count`. The function should take an integer vector as its argument and should output the number of even and odd numbers from the vector. The output of the function should be in the form of a list.

Exercise 11

Create loop to compute the mean of each column of the `mtcars` data frame. The data set `mtcars` is built into R. Just type `mtcars` and you will have access. You should store your results in a vector and display the results.

Exercise 12

Create a loop to compute the number of unique values in each column of the `iris` data frame. The data set `iris` is built into R. Just type `iris` and you will have access. You should store your results in a vector and display the results.

Exercise 13

Write a function that displays the mean of each numeric column in a data frame, along with its name. Use the `iris` data set that is mentioned in exercise 11 to test your function. Name your function `show.mean`. For example, `show.mean(iris)` would print:

Sepal.Length Sepal.Width Petal.Length Petal.Width

```
    5.84         3.06         3.76         1.20
```

Exercise 14

Write a while loop to determine how many tries it takes to get three 1's in a row if you randomly sample from the Bernoulli distribution with probability 0.50. To generate a random number from a Bernoulli random variable use `rbinom(1,1,0.5)`.

Exercise 15

Create a function named `trials.count`. Use what you did in exercise 14 to now create a function to determine how many tries it takes to get to a user-specified number of 1's with a user specified probability of success on each trial. To modify the probability of success, change `rbinom(1, 1, 0.5)` to `rbinom(1, 1, p)` where p is the probability of success you want to use.