Loops and Functions

Sampling

Sampling one value from a set

The sample() function in R can be used to sample some numbers from a set of possible outcomes with given probabilities. The following code can be used to select one value (denoted by "size") from the set of possibilities (0,1), where the probability of drawing a 0 is 40% (or .4) and the probability of drawing a 1 is 60% (or .6). Note: in R, we use c() to combine values into a set (called a vector).

Run the following a couple of times to understand the output.

```
sample(c(0, 1), size = 1, prob = c(0.2, 0.8))
```

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

```
# What is the output?  One number that is either 0 or 1. 0 has a probability
# of 0.20 and 1 has a probability of 0.80.
```

Now let's change some of the some of the parameters so that it samples from the values (0,1,2) with probabilities (0.3, 0.5, 0.2)

```
# Insert code here
```

Sampling multiple values from a set

We can also change the size parameter to sample multiple values at once. However, we may need to use the replace parameter if we want to sample from the same set repeatedly.

Running the following code will produce an error because we only have 2 values in our sample space and want to draw 3 values without replacement. For example if I draw a 1, then 3 will be the only value for me to draw from for my 2nd draw. This leave me no values to draw from for my third draw.

```
# Sampling 3 values from a set containing only 2 values with replace = FALSE
# sample(c(1,3), size = 3, prob = c(.4,.6), replace = FALSE)
```

```
# Sampling 3 values from a set containing only 2 values with replace = TRUE
sample(c(1, 3), size = 3, prob = c(0.4, 0.6), replace = TRUE)
```

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

Sampling from a set of character strings

In the previous examples, we sampled from numeric values, but we do not necessarily sample from only numbers. We can use quotes to denote character strings. The code can we used to sample one color from the set of primary colors ("red", "yellow", "blue") with probabilities (0.3, 0.3, 0.4).

```
sample(c("red", "yellow", "blue"), size = 1, prob = c(0.3, 0.3, 0.4), replace = TRUE)
```

```
## [1] "red"
```

If/else conditional statements

We can use an if/else conditional statement: "if (condition) {Execute some code} else {execute some other code}". This will check if the condition is met. If so, it will execute the first chunk of code. Otherwise (else), it will execute the second chunk of code.

Please note that the following example is not representative of an actual credit card application process. There are many of ther factors that are taken into consideration that we are not accounting for.

The following code will go sample from two sets of credit scores ("300-599", "600-850") with probabilities (0.3, 0.7). If the outcome is "300-599" then our credit card application is not approved. If the outcome is "600-850", we will sample from the credit ranges "600-699", "700-850" with probabilities (0.4, 0.6). If your credit score is between "600-699" your credit card application is approved with a credit limit of 2,000, else (your credit score is between "700-850") your credit card application is approved with credit limit of 10,000.

```
# First we determine whether our credit card application is approved based
# off credit score
creditscore1 <- sample(c("300-599", "600-850"), size = 1, prob = c(0.3, 0.7),
    replace = FALSE)

# If our credit score is between 300-599, our credit card application is not
# approved
if (creditscore1 == "300-599") {
    creditcard = "not approved"
} else {
    # otherwise...

    # If our credit score is between 600-850 we can go the the next step:
    # creditscore2 creditscore2 tells us that our credit card application was
    # approved and our credit limit
    creditscore2 <- sample(c("600-699", "700-850"), size = 1, prob = c(0.4,
        0.6), replace = FALSE)

    # If our credit score is between 600-699, our credit card is approved with a
    # credit limit of $2000
    if (creditscore2 == "600-699") {
        creditcard = "approved - credit limit: $2,000"

    } else {
        # otherwise...

        # If our credit score is between 600-699, our credit card is approved with a
        # credit limit of $10000
        creditcard = "approved - credit limit: $10,000"
    }
}
paste("Your credit card application was", creditcard)  #This line prints out the outcome of our credit
```

```
## [1] "Your credit card application was not approved"
```

Functions

Functions are created using 'function()' and are stored as R obejcts in your global environemnt. They are R objects of class "function".

Functions follow the following structure:

'myFunction <- function(arguments){ execute some interesting code }'

Using the above example with credit card application, we can create a function that takes in a numerical value (x) and determine if the application was approved. We will use the same conditions as above and for the sake of simplicity, if the credit card is not approved the credit limit will equal 0.

We will first set the condition if x is less than 600, then the credit card application will be denied and receive a credit limit of 0. If the credit score is between 600 and 700, then the application will be approved with a credit limit of 2000. Lastly, if the credit score is above 700, then the credit card application is approved with a credit limit of 10000. Once it meets one of the conditions, the function will print out out creditscore.

```
cc_app_function <- function(x) {
    # name the function and set the arguments of the function

    if (x < 600) {
        # sets the 1st condition
        creditlimit = 0  # what happens when the 1st condition is met

    } else if (x >= 600 && x < 700) {
        # sets the 2nd condition
        creditlimit = 2000  # what happens when the 2nd condition is met

    } else {
        x > 700  # sets the 3rd condition
        creditlimit = 10000  # what happens when the 3rd condition is met
    }
    creditlimit  # calls the result of the credit card application
}
```

Now let's make sure the function works. What is the result of my credit card application if my credit score is 500? 650? 790?

```
# Insert code here to see what the result of your credit card application is
```

For loops

Let's say we have 500 samples with ~30% of our sample with a credit score less than 600, ~28% with a a credit score between 600 and 700, and ~42% with a credit score greater than 700 and we want to determine the credit card application process of each credit score. We could find the credit card of each application through the following code...

```
set.seed(333)
N <- 500  # Sets the sample size and the number of loops we will run
creditscores <- sample(c(rnorm(n = round(N * 0.3), mean = 500, sd = 25), rnorm(n = round(N *
    0.28), mean = 650, sd = 20), rnorm(n = round(N * 0.42), mean = 750, sd = 20)),
    500)  # Creates random sample of credit scores

cc_app_function(creditscores[1])  # Finds ths result of a credit card application with the creditscore
```

```
## [1] 2000
```

```
cc_app_function(creditscores[2])  # Finds ths result of a credit card application with the creditscore
```

```
## [1] 10000
```

```
cc_app_function(creditscores[3])  # Finds ths result of a credit card application with the creditscore
```

```
## [1] 2000
```

```
cc_app_function(creditscores[4])  # Finds ths result of a credit card application with the creditscore
```

```
## [1] 2000
```

```
cc_app_function(creditscores[5])  # Finds ths result of a credit card application with the creditscore
```

```
## [1] 2000
```

```
# We can continue this process to cc_app_function(creditscores[500]) but
# that is redundant and can be done much more efficiently with a for loop
```

We can better model this process with a for loop to repeat the process a bunch of times and save the results in a vector. The general format for loops is to create an empty vector, matrix, dataframe, etc... to store your results.

In the example below we will store our results in an empty vector. Then we initialize the loop to begin at the position we want to start from (ie. if we want to start from the very beginning we will start at 1) through where we want the loop to stop (if want to go through all the credit scores we can use 'length(creditscores)' but since we are creating the set of credit scores we will end at the Nth position).

In the loop, we will assign our results to the empty vector that we created (cc_app_results) and store the results for every iteration. In each iteration, the result of out credit card application, for each credit score, is stored. We can look at the probabilites of our credit card application by using the 'table' command and dividing our results by the total number of values/iterations (N).

```
cc_app_results <- rep(NA, N)  # Create an empty vector to store your results

for (i in 1:N) {
    # repeat the process for every credit score, the creditscore vector is of
    # size N
    cc_app_results[i] <- cc_app_function(creditscores[i])  # store our results from the cc_app_function
}

# We can call the results from the loop to see if they are the same as from
# above
cc_app_results[1]
```

```
## [1] 2000
```

```
cc_app_results[2]
```

```
## [1] 10000
```

```
table(cc_app_results)/N  # Probabilites of credit limits
```

```
## cc_app_results
##     0  2000 10000
## 0.300 0.278 0.422
```

Is this what you would expect? 1. What is the probability of obtaining a credit limit of 2000 dollars? 2. What is the probability of obtaining a credit limit of 10000 dollars?

While loops

The credit card company can only approve 50 credit card application with a credit limit of 10000. Let's walk through a while loop to see how many credit card applications will go through before approving 50 card applications.

Practice Problems