Data Science for High School

Summer 2021

Haim Bar, HaiYing Wang, and Jun Yan

Department of Statistics, University of Connecticut

1 Before We Begin...

In this class, we will use the R language, and the RStudio integrated development environment (IDE). Before the first meeting, install the necessary software on your personal computer by executing the following instructions. Go to the website of the Comprehensive R Archive Network `https://cran.r-project.org/` and download the latest version of R for your operating system (Windows, MacOS, or Linux). As of March 14, 2021, the latest version is 4.0.4. Follow the installation instructions.

Next, go the Rstudio download website `https://rstudio.com/products/rstudio/download/` and get the Desktop version (open source license). As of March 14, 2021, the version is 1.4.1106. Follow the installation instruction. An icon that looks like this: ⓡ will be on your computer's desktop. Double-click on this icon to start the R session. The Rstudio IDE will open, and look like this:
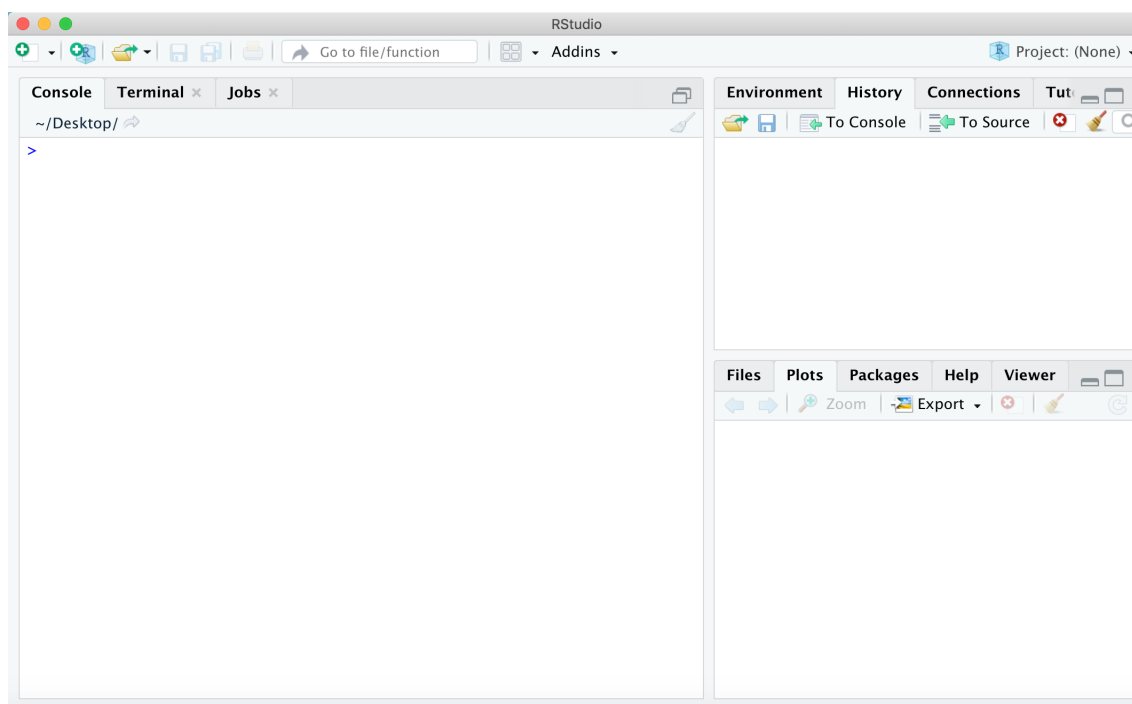
The left-hand side of the screen contains the Console tab. Notice the > sign (called the 'prompt'). When you see this character, it means the R is ready for the next command. Put the cursor there, and then type

```
2+2
```

and hit Enter on the keyboard. You should get the following on the Console:

```
[1] 4
```

Notice the top-right part of the Rstudio window. You should see an 'Environment' and a 'History' tab. Click on History. Notice that your previous input appears there. Try entering another calculation or command and see that they appear in the session's history. For example, try entering the following (without the prompt sign)

```
> 3^5
> date()
```

Click on the Environment tab. It should be empty when you start R for the first time. In the Console, type

```
> myFirstVariable <- factorial(5)
```

Notice that nothing was printed in the Console, but the Environment tab now contains a table with one row, with 'myFirstVariable' appearing in the cell on the left, and its value (120) on the right. Any object appearing in the Environment tab is available to you throughout your R session, and you don't have to recalculate it. For example, you can try the following:

```
> myFirstVariable/6
```

The Console should now display 20.

The lower-right side of the IDE contains a file browser (the Files tab), information about installed packages (more about it later), and any plot generated during the R session. It also contains a Help tab, to obtain information about built-in functions.

Finally, before we move on to the next section, in the Rstudio top menu, click on File, then on New File, and then on R Script. Alternatively, you can click on little green '+' icon in the top-left part of the IDE. This will split the left side of the Rstudio IDE into two parts – the lower part will contain the Console, and the top part will contain a tab labeled 'Untitled1'. This is were you can enter R code which you will save to a permanent file, and re-use later. For example, enter the following in the blank space in the Untitled1 tab:

```
# This is my first R program
cat('Hello, World!\n')
```

Then, click on File > Save, and in the 'Save As' box enter FirstProgram.R and click the Save button. Notice that the tab name is now FirstProgram.R.

In that part of the window, there should now be a small button called Source. Click on it. The program will be executed and the output will be shown in the Console. You can also execute individual lines in the source code. Just put the cursor anywhere in that line, and click on the Run button (which is near the Source button.)

That's it. In the rest of these notes we will see more features of Rstudio, but you are now ready to start learning programming in R.

2 Basic Operations in R

2.1 Generating random numbers

In this course, much of the learning will be done by using simulated data, so let's start by learning how to generate data. The basic function to generate random data is `runif` which is used to draw random numbers, uniformly between 0 and 1. In the following example we create 10,000 random draws from a uniform distribution, keep these numbers in a variable called simData, and plot a histogram to show the distribution of the data we have generated. The `set.seed` function is used to ensure that every time we run this code, we will get the same set of random numbers. This is called *reproducible code*.

```
1  # Generate 10,000 points from a uniform distribution
2  set.seed(210313)
3  n <- 10000
4  simData <- runif(n)
5  hist(simData)
```

This will produce the following:

Histogram of simData
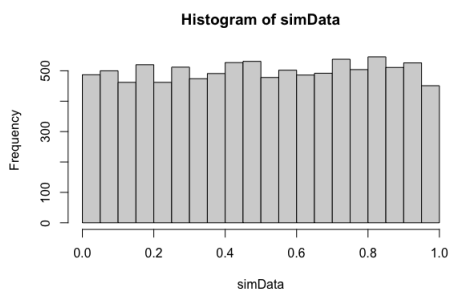
From the range and the flatness of the histogram we can see that the generated data is indeed uniform on [0,1]. The `runif` function can be used to draw random numbers uniformly on any finite

3

interval. For example, if we want our random numbers to be in the interval [1,5] we will run the following code:

```
simData <- runif(n, min=1, max=5)
```

Try it, and draw the histogram as in the previous example.

From a random draw of a uniform distribution, we can generate random numbers from other distributions. For example, we want to simulate coin flips, and count the number of Heads that we get. Let's say we want to simulate 200 coin tosses. We will draw 200 random numbers from a uniform distribution, and decide that we got Heads in the i-th toss if the i-the random number is less than 0.5, and Tails otherwise. Try the following code multiple times. What do you observe? We will discuss it further in a different lecture.

```
n.trials <- 200
cat("Number of Heads is: ", sum((runif(n.trials) < 0.5)), "\n")
```

Statistical inference is based on a mental exercise in which we ask, if we could repeat the same experiment infinitely many times, what would we see? With simulations, we can get a good approximation. For example, the 200 coin-tosses experiment can be repeated, say, 100 times. One way is to use loops, like in the following example:

```
n.trials <- 200  # the number of coin-tosses in each experiment
reps <- 100  # the number of experiments
Heads <- rep(0, reps)  # A vector to store the results (initialize with 0s)
for (i in 1:reps) {
  Heads[i] <- sum((runif(n.trials) < 0.5))
}
hist(Heads, breaks=30)
```

We've used a `for` loop, with an index variable called `i` which runs from 1 to 100. In each iteration we draw $n.trials = 200$ random numbers from a uniform distribution. For each value which turned out to be less than 0.5, `runif(n.trials) < 0.5` is set to TRUE. Otherwise, `runif(n.trials) < 0.5` is set to FALSE. When we use the `sum` function, the TRUE values count as 1 each, while FALSE ones counts as 0, so the result is the number of Heads. The results is stored in the i-th position in the vector we called `Heads`.

Using the uniform distribution, we have created a different distribution, called the 'binomial'. The mathematical notation is $Bin(N, p)$ where N is the number of trials where in each trial there can be exactly two outcomes (e.g., coin tosses), and p is the probability of the first possible outcome (e.g., Head), and $1 - p$ is the probability of the second possible outcome.

R has a built-in function to generate random numbers from many different distributions, so the code above can be replaced by the following, which uses the `rbinom` function:

```
1   Heads <- rbinom(reps, n.trials, 0.5)
2   Hist(Heads)
```

Exercises:

1. Change the number of simulated coin-toss datasets (`reps`) to 1,000 and rerun the code. Then change it to 10,000 and run it again. What do you notice?

2. Change the probability of Heads to 0.2 and run the code again. Then, change it to 0.8. What do you observe?

3. Try other values of the number of trials and the probability of Heads.

2.2   Summary statistics