Lecture 14 Programming III

Today, I’m going to go over the rest part of lecture 13 first. Then discuss loop functions in R. Finally, we’re going to work on analysis 3 together.

**Lecture 13**

**Practicing Functions: T-Test**

In the next example, we are going to build function for t-test. Let’s write a t-test function to test the following hypotheses.

* The null hypothesis is: the average number of hours spent watching TV per day is (a number specified) hours in the USA.
* The alternative hypothesis is the opposite.
* We are interested in whether the alternative hypothesis is true. And, we will use the TV time column in general social survey data to answer the question.
* This is an example of using t-test to test population mean.
* (next slide)
* Let’s first recall how to do t-test for population mean.
* To do t-test, we first need to specify alpha, the type I error and make a guess on the population mean.
* Then the most important part, use the sample mean, our guess of population mean, sample standard deviation to calculate the t-test stats.
* Then, use the t-test stats to find corresponding p-value. If the p-value is smaller than alpha, we reject the null hypothesis, otherwise we cannot reject it.

**Building Functions (2 slides)**

Before writing code for the function, we should first figure out what the input and expected output are for this function.

* First of all, we need sample data to calculate the sample mean and standard deviation.
* We also need a guess of population mean to complete the hypothesis.
* Finally, we need alpha to reach a conclusion.

And what should the function produce?

* It depends on what information you’re interested in. Test-statistic, p-value are important information of the test. With the two, you can tell whether we should reject the null hypothesis under different alpha threshold.
* The decision can directly be provided.
* We can also visualize the test with ggplot.
* (Next slide)
* After figuring out the input and expected output, we can write the code of the function.
* The function name is ttest. It takes the sample data, null hypothesis and alpha as input.
* First, the t-test is calculated with the input.
* Since it’s two-tailed t-test, p-value is found by the pt() function with specified t-test stats and degree of freedom.
* With the p-value and specified alpha, we can use if else statement to get the conclusion. If p-value is smaller than alpha, we reject the null hypothesis, otherwise we cannot reject the null hypothesis.
* Finally, we visualize the test with ggplot. A histogram of the data is first generated, then a vertical line of the null hypothesis is put on the plot.

**Results (2 slides)**

The results of the test is provided.

* In this slide, the test result of null hypothesis: average number of hours spent watching TV per day is 4 with alpha value 0.05, is provided.
* You can see the null hypothesis as a vertical line on the histogram of the sample data.
* The line is on the left side of the center pike area of the distribution. Without seeing the test result, you can guess that the null hypothesis should be rejected.
* The t-test statistic, p-value and conclusion provided by the function proves my guess.
* (Next slide)
* In this slide, the test result of null hypothesis: average number of hours spent watching TV per day is 3 with alpha value 0.05, is provided.
* Based on the plot, we can see that the null hypothesis may not be rejected. The line is around the center pike arear of the distribution.
* With alpha equals to 0.5, we cannot reject the null hypothesis.

**Practicing Functions: CLT**

Central limit theorem is one of the most important theorems in statistics. Let’s review the theorem and try to visualize it with R function.

* Let X be a random variable with any distribution.
* We get a random sample of n observations from this distribution with mean mu and standard deviation sigma.
* The sample mean X \_bar will approximately follow a normal distribution with mean mu, and standard error sigma divided by square-root of n.
* The larger the sample size, the better will be the normal approximation.
* (Next slide)
* Let’s try to visualize the theorem with R function.
* First of all, we need a distribution to sample the data. Here we use two distributions, one is standard normal, another one is gamma distribution.
* To derive a simulated distribution of sample mean, we need to sample data of size n from the distribution and calculate the sample mean many times. Here S is the number of simulation times, and n is the size of random sample we got each time.
* To show the central limit theorem, we need to compare the simulated mean and standard error with the theoretical mean and standard error.
* The histogram of sample mean can also provide some information.

**Writing Functions**

Here’s the code of the function.

* We specify sample size, simulation times, and Use D to specify which distribution the sample data are from. D equals to 1 means standard normal, and D equals to 2 means gamma distribution with rate equals to 1.
* In the first part, we generate 1 million data point from the corresponding distribution. Sample data with such large size can be used to present the population.
* Then we calculate the theoretical mean and standard error based on the populatin data.
* In the second part, we perform simulation. Since we are going to sample the data from the initial distribution S times, we create an empty vector with length S to store the mean of each sample.
* A for loop is used here to do the simulation. In each iteration, n random sample is sampled from the initial distribution. The mean of the random sample is calculated and saved to the sample mean vector.
* We take the average of all the means. This is the simulated mean of sample mean distribution. The standard deviation of all the means is the simulated standard error.
* (The standard deviation a measure of the variability of a random variable. The standard error is the standard deviation of the estimator in repeated samples from the population.)
* A histogram of the simulated sample means are generated to better visualize the process.
* And all the results are saved to a list as output of the function.

**Results: Gamma**

In this slide, I showed you the results of n equals to 1, S equals to 1000, and initial distribution is gamma distribution.

* This means we generate random sample from gamma distribution 1000 times, but every time we only sample 1 data point from the distribution.
* Therefore, visualize the sample mean distribution of the 1000 samples is the same as visualize 1000 data points from the gamma distribution.
* The plot here is roughly of the shape of gamma distribution with rate 1, which is our second initial distribution.

**Results: n=10, 100, 1000**

In this slide, I increased the sample size n to 10, and repeated the simulation 1000 times. The initial distribution is still gamma distribution.

* But you can see that, this time, as n gets larger, the sample mean distribution is closer to normal distribution.
* And the center of the distribution is close to the theoretical mean.
* (Next slide)
* If I increase the sample size to 100, the first thing to notice is that the spread of the simulated sample mean distribution gets narrower. This is expected as the theoretically, the standard error of sample mean decreases as n increases. And the simulated mean gets closer to the theoretical mean as n increases.
* (Next slide)
* The last slide shows sample size equals to 1000. Now the spread of the distribution is even more narrower and the simulated statistics get very close to the theoretical ones. This shows `The larger the sample size, the better will be the normal approximation.`

Lecture 14

Loop Functions

In this part, I’m going to introduce several loop functions in R. These functions are a family of functions in R which allow you to repetitively perform an action on multiple chunks of data. An apply function is essentially a loop, but run faster than loops and often requires less code.

* Three functions I’m going to discuss are lapply, sapply and apply.
* lapply and sapply are functions that will loop a function through data in a list or vector.
* apply() function operations on arrays.
* Loop functions are not covered in the textbook, but you can find more information of loop functions in Chapter 18 of another reference book, R Programming for Data Science.
* There more than 3 loop functions. I’m not going to discuss all of them. Once you know how to use the three apply functions introduced in today’s lecture, you’ll be able to transfer the knowledge to other apply functions easily.

lapply() (2 slides)

Let’s start with the lapply function. Lapply function can apply a given function to every element of a list and obtain a list as result. When you type ?lapply in the console, you’ll see the execution details of the function:

* The first argument would be the input.
* The second argument is the function to be applied on each element of X.
* In the rest part, you can specify other optimal arguments.
* (Next slide)
* Let’s see some examples to help you understand it.
* You can apply minimum or mean functions on each column of the dataset CARS using lapply function.
* The loop function will return the corresponding minimum and mean of the columns.

sapply() (2slides)

Sapply() function is similar to lapply() function, but the output is simplified to the most elementary data structure that is possible. This means that instead of returning a list like lapply, it will return a vector instead if the output is simplifiable.

* The first argument is the input.
* Second argument is the function to be applied on each element of X.
* You can see in the simple example. If we use sapply to get minimum of each column in CARS, the output is a vector not a list.
* (Next slide)
* There is another example of using sapply(). In previous examples, the inputs are data frames. Here, in this example, the input is a vector.
* We have done this exercise in Tutorial 08: Generating 100 random samples from normal distributions with means range from 0 to 99.
* In tutorial 8, we use for-loop to iterate from 0 to 99 and sample 1 random point each time.
* Now you have learned loop functions. Loop function can achieve the same goal without writing a loop.
* In the code chunk, the first part is solution using for-loop. The results are saved to vector a.
* In the second part, I use sapply function to generate the results. It’s just one line of code. The first argument of the sapply function is a sequence from 0 to 99. This means the mean of normal distribution changes from 0 to 99.
* Since we would like to apply random normal function 100 times to the sequence, we specify the function name as random normal. The number of data point sampled is 1 each time and the standard deviation is always 1.
* The two methods get exactly the same results.

Apply() (2 slides)

The last loop function I’d like to discuss is apply() function.

* This function takes data frame or matrix as an input and operates.
* In the second argument, you specify the margin to operate on. Margin = 1, means the function will be applied to each row of the input.
* Margin=2, means applying the function to each columns of the input.
* Margin=c(1,2), means applying the function to all the elements.
* The last argument specifies which function to be applied.
* (Next slide)
* Here are some examples for apply() function.
* I created a 100 by 10 matrix. To calculate the sum of each row, you can use apply() function on the matrix, specify margin equals to 1, and specify the function to be applied is sum.
* Change the margin to 2, you’ll get column sums.
* In the second example, I use apply function on a data frame to calculate five summary statistics.
* First, specify the input data frame, then specify margin equals to 2. This means applying all the functions on the columns.
* Then, for different summary statistic, different functions are specified.

Analysis 3