Getting Started with R and RStudio

Cheng Peng

Contents

1 Introduction

This module outlines the open-source computer software program R (https://www.r-project.org/) and the platform RStudio (https://posit.co/downloads/) to be used in this and subsequent statistics courses.

2 Download and Installation

Both **R** and **RStudio** are free and open-source. **R** is a programming language widely used in statistics and data science, including machine learning, while **RStudio** is a data science platform that simplifies working with **R**. In other words, we need to install both **R** and **RStudio** and then use **R** through **RStudio**.

The following YouTube video by Tony Carlsen demonstrates the steps for downloading and installing both programs.

Please follow the steps to install these two programs on your machine. You can also use R Studio on WCU's Ramcloud.

3 Getting Started with R and RStudio

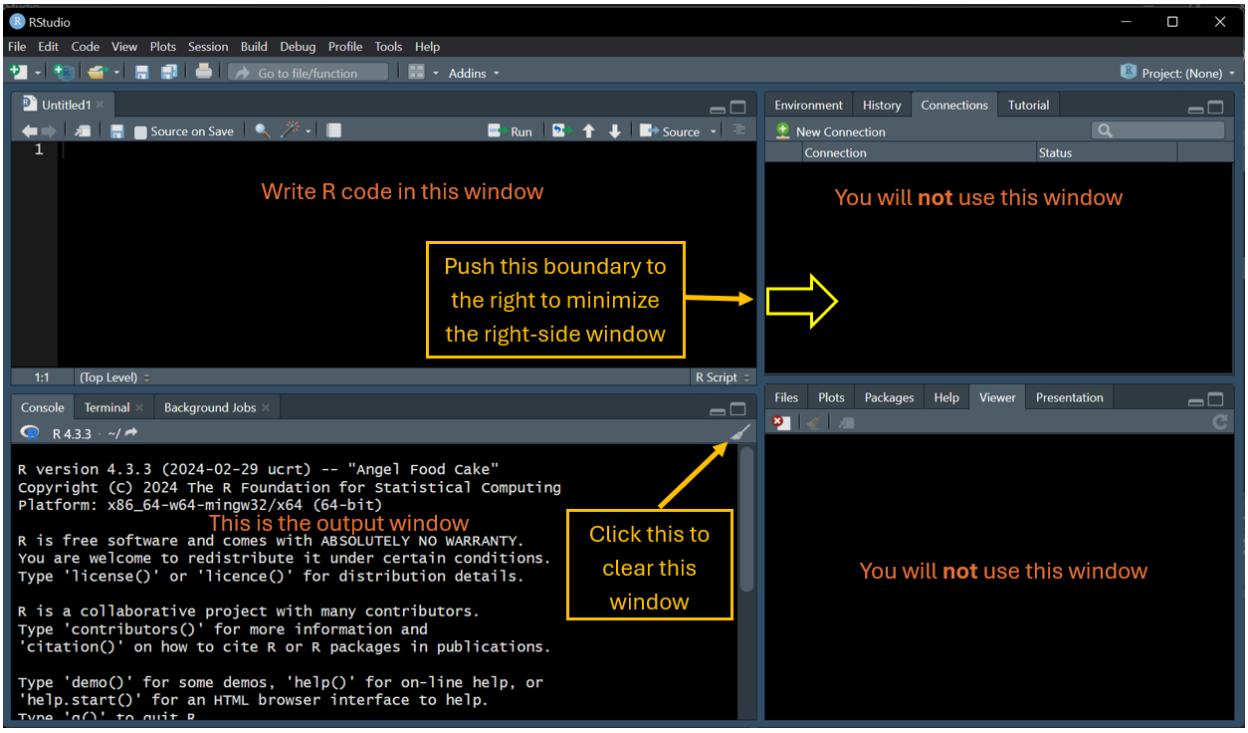
The next video shows how to use R through RStudio with some basic arithmetic operations and basic commands that will be used to compose some formulas in this course.

You can also change the appearance of the RStudio user interface (UI) to get a more comfortable and better UI by following the next few steps:

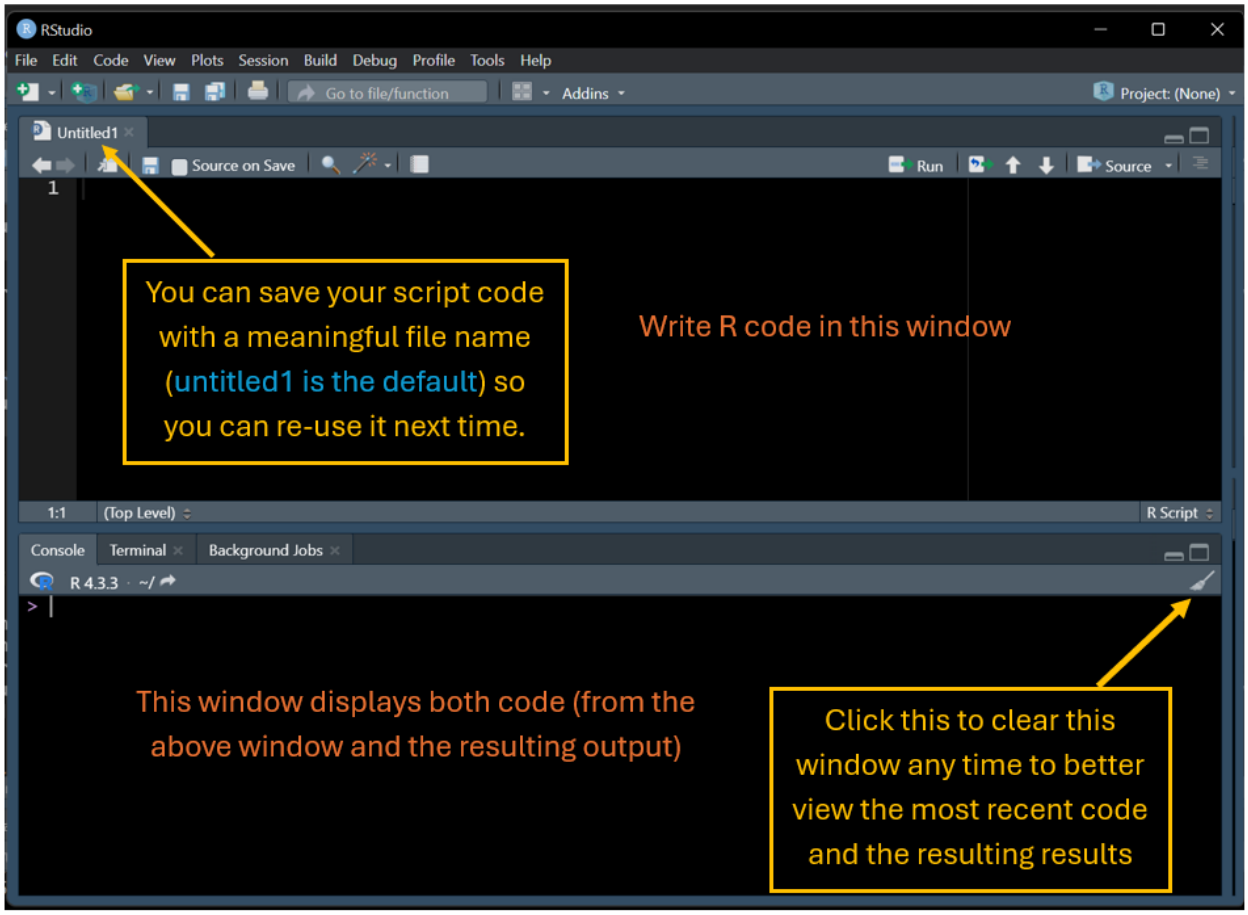
- From the menu bar, go to **Tools** > **Global Options**
- Click on **Appearance**
- Change the **Editor font size** if you want to
- Try out a few themes in the **Editor theme box**. (The default is **Textmate**. I prefer **Pastel on Dark**).
- Once you find something you like (or just stick with **Textmate** if you are happy with the default appearance), click on **OK**, and continue with this tutorial.

https://pengdsci.github.io/STA200/gif/Rsetup.gif

My own RStudio UI (user interface) is shown below (**File > New File > R Script**)



After clearing the **Console** (bottom-left window) and minimizing the right side windows (top-right and bottom-right windows), we have the following UI with **Script** window and **Console** window.



It is convenient for you to save a single file that includes all of your code to be drafted during the semester. We will discuss how to effectively organize your code for different modules later.

4 Using R As A Calculator

R can be used as a powerful calculator by entering equations directly at the prompt in the command console. Simply type your arithmetic expression and press ENTER. R will evaluate the expressions and respond with the result. While this is a simple interaction interface, there could be problems if you are not careful. R will normally execute your arithmetic expression by evaluating each item from left to right, but some operators have precedence in the order of evaluation. Let's start with some simple expressions as examples.

4.1 Simple Arithmetic Expressions

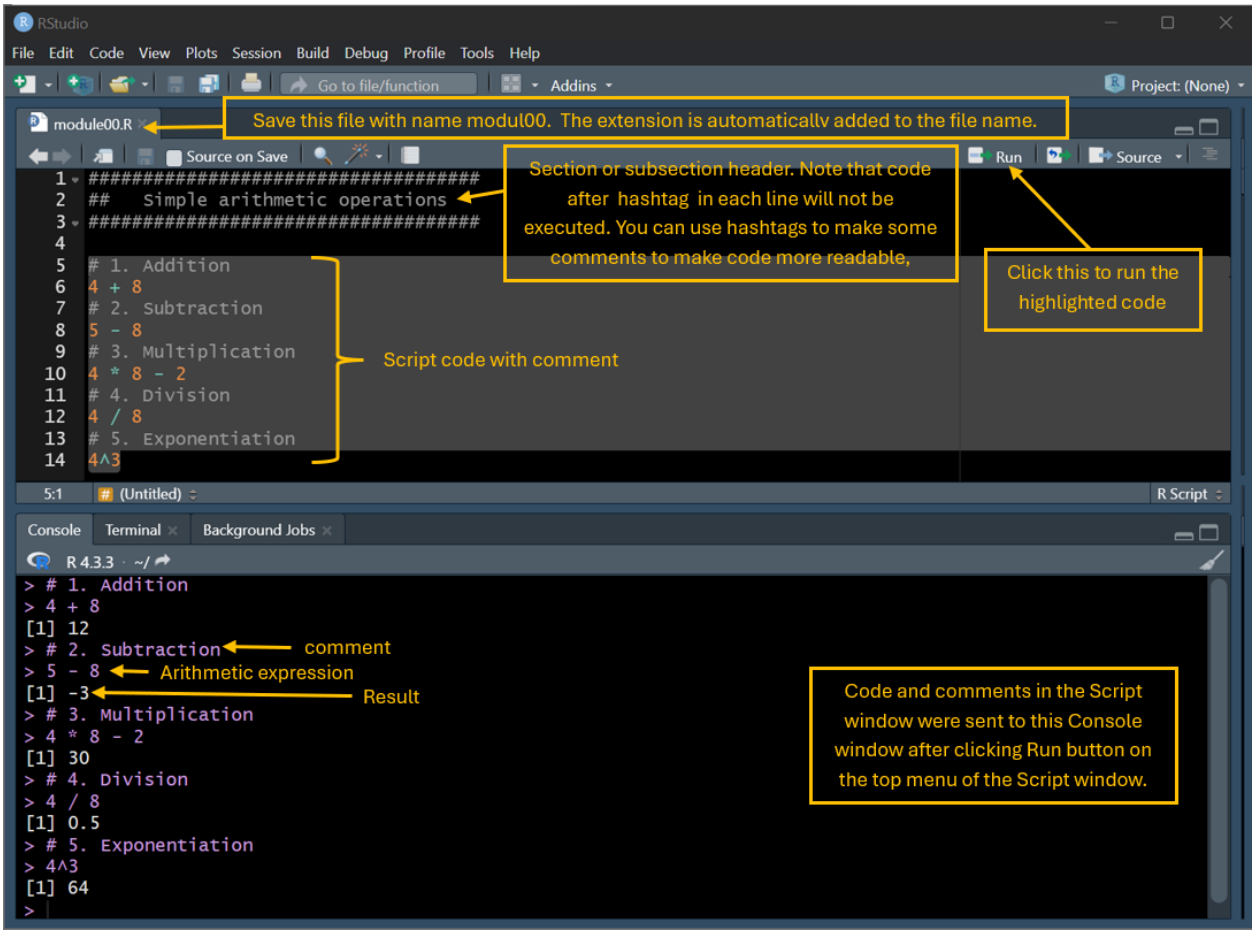
The operators R uses for basic arithmetic are: `+, -, *, /, ^`. The following table presents some examples.

Operator	Meaning	Example Expression	Result
`+`	Addition	`4 + 8`	12
`-`	Subtraction	`5 - 8`	-3
`*`	Multiplication	`4 * 8-2`	30
`/`	Division	`4 / 8`	0.5
`^`	Exponentiation	`4^3`	64

Here is how I performed the above operations in RStudio:

1. **Open RStudio** (click the RStudio icon, it will automatically open the script window, Console, and other windows on the right-hand side). Minimize the windows on the right-hand side to keep only Script and Console windows.
2. **Type the expressions** in the **Script window**.
3. **Highlight the expression** you want to run,
4. You will view **both code and results** in the **R Console**

The following is the screenshot of my RStudio UI (with some annotations)



From the above screenshot, you see that using hashtags can make your code more organized.

4.2 Input Data in R

In statistics, a data set consists of values of multiple measurements from multiple characteristics. For example, a data set contains **height**, **weight**, and **gender** taken from a group of n students.

observation ID	height (X)	weight (Y)	Gender (Z)
1	x_1	y_1	F
2	x_2	y_2	M
$\vdots$	$\vdots$	$\vdots$	$\vdots$

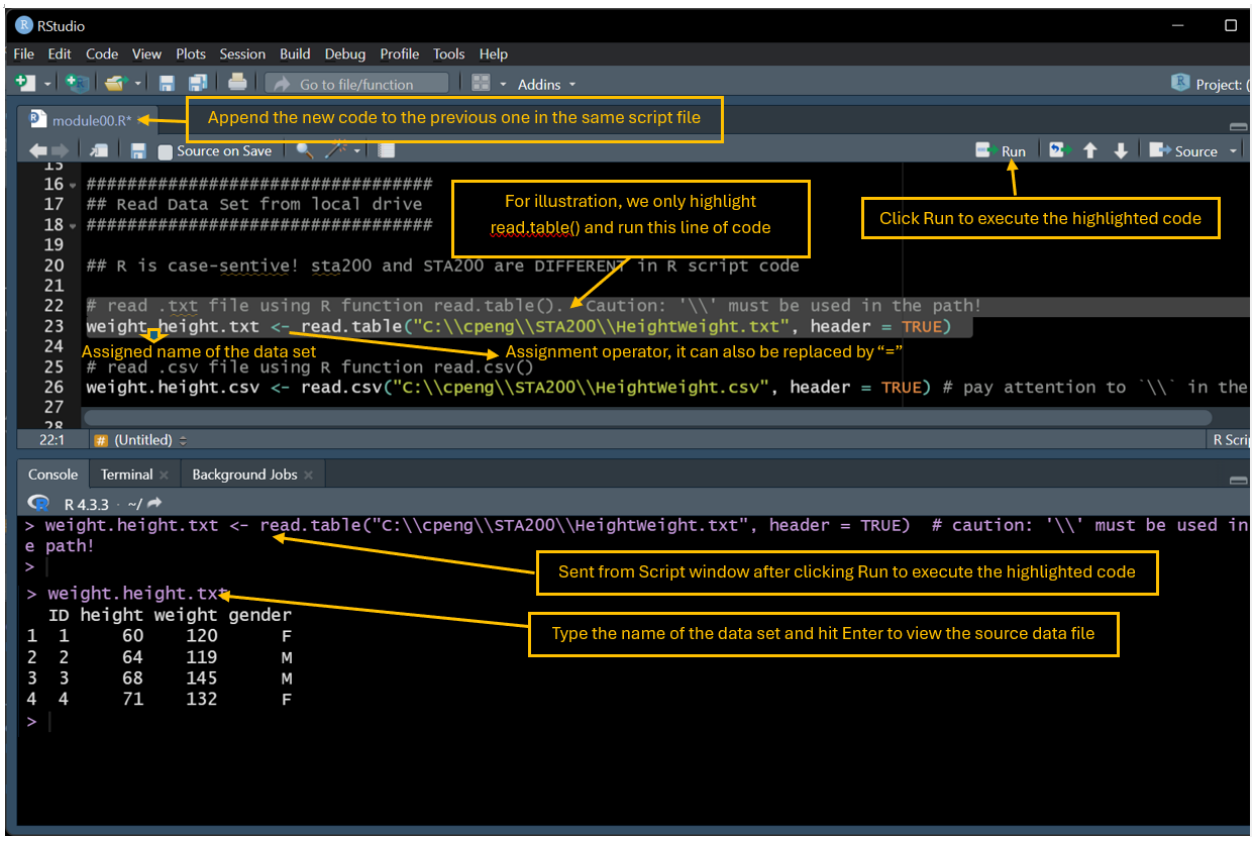
observation ID	height (X)	weight (Y)	Gender (Z)
$n-1$	x_{n-1}	y_{n-1}	M
n	x_n	y_n	F

The above data set has n rows, each row records a student's **height**, **weight**, and **gender**. Different columns represent different characteristics, which are commonly called variables. A dataset is usually saved in a different format. The most common formats of a flat data file are a text file `.txt` (plain text file). If Excel is used to store data, **comma-separated values** `.csv`, and **Microsoft Excel spreadsheets** (`.xls`) or **Excel Open XML Spreadsheet** (`.xlsx`). A data set with a different format required a different **R function** to read data into R.

As an example, I save the following data set in `C:\cpeng\STA200` in plain text format with extension `.txt` and comma-separated values with extension `.csv`.

```
"'{} ID   height    weight   gender 1     60         120      F 2     64         119
M 3     68         145      M 4     71         132      F
```

When reading the data set into R, you need to provide the path to the data file. The following screenshot shows how to use appropriate **R functions** to read the dataset.



We can also define individual variables and then make a data frame using the **R function** `data.frame()` as shown in the following **code chunk**.

```
# define individual variables first
ID <- c(1,2,3,4)     # ID = observation id, lower case c() is an R function used to define a vector.
height <- c(60, 64, 68, 71)
weight <- c(120, 119, 145, 132)
gender <- c("F", "M", "M", "F")    # Categorical values must be enclosed in double quotes and separated
```

```r
# put the above variables in a dataframe
height.weight.data <- data.frame(ID = ID, height = height, weight = weight, gender = gender)  # data.fr
```

You can also define the data frame directly using the following code.

```r
height.weight.data.02 <- data.frame(
                            ID = c(1,2,3,4),       # CAUTION: "=" CANNOT be replaced by "<-"!!!!
                            height = c(60, 64, 68, 71),
                            weight = c(120, 119, 145, 132),
                            gender = c("F", "M", "M", "F")
)
height.weight.data.02
```

```
##   ID height weight gender
## 1  1     60    120      F
## 2  2     64    119      M
## 3  3     68    145      M
## 4  4     71    132      F
```

4.3 Working With Data Frame

Quite often, we only work with one or two variables in a data frame instead of the entire data set. For example, we want to calculate the mean and standard deviation of the variable `height` in the above data set. We can extract `height` from the data frame we defined using the following code.

```r
height <- height.weight.data.02$height   # datasetname + $ + variablename
# Calculate mean and variance
xbar <- mean(height)  # compute the mean and store it in a variable under the name of xbar
xbar                  # print out the result
```

```
## [1] 65.75
```

```r
var.height <- var(height)
var.height
```

```
## [1] 22.91667
```

4.4 Some Basic Statistics and Mathematics Functions

Most of you have experience using graphing calculators and relevant functions. R has similar built-in functions for basic mathematical and statistical calculations. We use `height` and `weight` in examples in the following table.

Math & Stats function	R function	Example Code	Result
mean	`mean()`	`mean(height)`	65.75
variance	`var()`	`var(height)`	22.92
standard deviation	`sd()`	`sd(height)`	4.79
correlation coefficient	`cor()`	`cor(height, weight)`	0.691
summation of data values	`sum()`	`sum(height)`	263

4.5 Critical Values and Left-tail Probabilities

In testing hypotheses, we can use either the critical value or p-value methods to make a statistical decision. The next table lists the R functions for critical and p-values from normal and t tables.

Critical Value	degrees of freedom	Example Code	Result
95% normal critical value	NA	`qnorm(0.975)`	1.96
95% normal critical value	25	`qt(0.975, 25)`	2.059539

left-tail Probability	degrees of freedom	Example Code	Result
$P(TS < 1.45)$ normal table	NA	`pnorm(1.45)`	0.9264707
$P(TS < 1.45)$ t table	15	`pt(1.45, 15)`	0.9161772

4.6 Evaluate Formulas

Just like using a graphing calculator, we sometimes need to evaluate a formula. For example, when constructing a 95% normal confidence interval based on given descriptive statistics (rather than raw data), we use the following formula.

$$\bar{X} \pm Z_{\alpha/2}\frac{s}{\sqrt{n}}.$$

We use an example to illustrate how to write the above formula in R to calculate the confidence interval.

Example: The Dean wants to estimate the mean number of hours that students worked per week. A sample of 49 students showed a mean of 24 hours with a standard deviation of 4 hours. The point estimate is 24 hours (sample mean). What is the 95% confidence interval for the average number of hours worked per week by the students?

reasoning process: since $n = 39 > 30$, we can use the central limit theorem (CLT) to claim that $\bar{X}$ is approximated by a normal distribution. Therefore, $Z_{\alpha/2}$ can be found using `qnorm(1-alpha/2)`.

```
## Assign values to variables in the formula
alpha = 1-0.95   # alpha
xbar = 24   # sample mean
stdev = 4   # standard deviation
n = 49
## critical value
Z.0.975 = qnorm(0.975)    # alpha = 1 - 95% - 1 - 0.95 = 0.05, 1 - alpha/2 = 1 -0.025 = 0.975
## lower and upper confidence limits
LCL = xbar - Z.0.975*(stdev/sqrt(n))   # no square and curly bracket should be used in R
UCL = xbar + Z.0.975*(stdev/sqrt(n))
## Write the confidence interval
cbind(LCL = LCL, UCL = UCL)    # combined the two limits in a two-column table with one row
```

```
##           LCL      UCL
## [1,] 22.88002 25.11998
```

5 R Built-in Statistics Function

R has a rich built-in functions for various statistical analyses. Next, we list some of the functions that can perform all the analyses in introductory statistics like MAT121 at WCU. These functions are called when you have raw data stored in variables. **Remember, each column in a data frame is a variable.**

For convenience, we use the following raw data set collected from a diabetes study, which can be found at https://pengdsci.github.io/STA200/dataset/diabetes-dataset.csv

We first read the above data using the command given previously and extract variables to perform one-sample, two-sample tests, correlation coefficient, and least squares regression.

Data loading and variable extraction

Statistical Task	Built-in R Function	Example with Data
correlation coefficient	`cor()`	`cor(BMI, SkinThickness)`
five-number-summary	`summary()`	`summary(BMI)`
histogram	`hist()`	`hist(SkinThickness)`
scatter plot	`plot()`	`plot(BMI, SkinThickness)`
frequency table (categorical data)	`table()`	`table(Outcome)`
linear regression	`lm()`	`lm(BMI ~ diabets.status)`

6 R Packages

An R package is a collection of functions, data, and documentation that extends the capabilities of base R. Different R functions in different packages allow users to perform different statistical tasks. In this course, we will use a few functions and some packages. To use an R function in a specific package, you need to load the package using the following command.

```
if (!require("packageName")) {
   install.packages("packageName")
   library(packageName)
}
```

For example, if you want to perform a z-test (i.e., normal test), we can use the R function `z.test()` in the package. The following is the code for testing BMI Ho: mu <= 30 vs Ha: mu > 30.

```
## install and load package
if (!require("BSDA")) {
   install.packages("BSDA")
   library(BSDA)
}
## Call the function to perform a normal test
# Ho: mu = 30 vs Ha: mu != 30, the alternative is !=, this is a two-sided test
# IF the test is right-tailed, the alternative MUST be specified as "greater",
# Similarly, if the test is left-tailed, the alternative MUST be specified as "less".
z.test(x = BMI, sigma.x = sd(BMI), mu = 30, alternative = "two.sided")
```

```
##
##  One-sample z-Test
##
## data:  BMI
## z = 7.0039, p-value = 2.489e-12
## alternative hypothesis: true mean is not equal to 30
## 95 percent confidence interval:
##  31.43498 32.55018
## sample estimates:
## mean of x
##  31.99258
```

You can see that the output also provides a 95% confidence interval of the mean BMI.

Some commonly used packages come with the R base package - this means that you don't need to install and load these packages when you use any R functions. These packages will be automatically loaded

when you start an R session. For example, the following R function `prop.test()` for testing population proportion is in package {stats}:

```
prop.test(75, 137, p =0.57, alternative = "greater")
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  75 out of 137, null probability 0.57
## X-squared = 0.19977, df = 1, p-value = 0.6725
## alternative hypothesis: true p is greater than 0.57
## 95 percent confidence interval:
##  0.4736288 1.0000000
## sample estimates:
##         p
## 0.5474453
```

There are more than 20,000 (twenty thousand!) R packages are available for various applications. We will use about five packages that require installation and explicit loading to access specific R functions for analysis. You don't need to memorize the names of these packages—I encourage you to use AI tools like ChatGPT or related Copilot assistants to find the R functions you need for your analysis. I will also provide this information in my example code within the lecture notes.