Lesson 3

by Lorraine Gaudio

Lesson generated on August 16, 2025

BOISE STATE UNIVERSITY

Contents

1. Welcome back to R!

In lesson one and two you learned about objects, vectors, and created a simple dataframe. Now you'll learn how to *do things* with them using **functions**.

To begin Lesson 3, follow these steps:

1. Open your course project for RStudio
2. Create a new R script file (File > New File > R Script).
3. Type in the code provide in this document as you follow along with the video. Pause the video at anytime to answer assignment questions, dig deeper or add memo notes.

Lesson Overview

In the next few minutes you will learn how to:

- Distinguish a function *object* from a *function call.*
- Read help files to discover arguments and defaults.
- Use required vs. default, positional vs. named arguments.
- Handle missing values with na.rm.
- Generate random samples and set a seed for reproducibility.
- Nest functions to build compact pipelines.

2. Functions-Call Basics

 Concept A *function object* is stored code; a *function call* runs it. The parentheses trigger execution. Objects or strings without () just print. Type in the following into your R script and run each line. Comment out lines that produce an error.

```
thing_1            # object? (expect an error message: not found)

"thing_1"          # character string — prints literally

3                  # numeric literal

mean               # function *object* → prints the function body

mean()             # function *call*   → error: argument is missing
```

 Check-in: What happens if you try sum vs. sum(1:5) ?

3. Date and Time

☐ R has built-in date and time functions. Type the following into your R script and run each line. **Utility** Quickly time-stamp results for reports or file names.

```
Sys.Date()      # current date (yyyy-mm-dd)

Sys.time()      # current date + time

Sys.timezone()  # your local timezone string
```

☐ Dig deeper: I use format(Sys.Date(), '%B %d, %Y') to format dates in Lesson reports. What is the function of % in '%Y-%m-%d'?

```
# Create a learning skills memo note in your R script if you did
# the ☐ Check☐ in and☐ Dig deeper.

## 1. Explain what you did to discover more about Functions-Call Basics and
# Date and Time settings in reports, what your observations are, and what you
# expect to happen.

## 2. Demonstrate learning skill(s) in your memo
```

4. Using Help

 Why use Help? Nobody memorizes every argument. ?function_name opens the help file.

Type the following R code into your script. Read the help tab for more about the *Usage* section for defaults, and the *Arguments* list for details.

```
?mean              # opens in Help pane (bottom-right)

help(sample)        # alternate syntax works the same
```

 Pro-tip: args(mean) prints the argument names inline.

5. Required vs. Default

□ **Key Idea** If R supplies a default you can omit that argument; if not, you *must* provide it. Use named arguments for clarity. Type the following into your R script and run each line.

```
values <- c(1, 2, 3, 4, 5, NA)

mean(values)                        # default na.rm = FALSE → returns NA

mean(values, na.rm = TRUE)          # ignore missing values

mean(values, trim = 0.2, na.rm = TRUE)  # drop 20% from each end first
```

□ Identify: Try median(values, na.rm = TRUE). Why doesn't median()need trim?

□ Look deeper: What is bootstrapping in statistics?

Type the following code in your R script and run each line.

```
set.seed(123)          # reproducible randomness

numbers <- 1:10

sample(numbers, 3)                  # three unique picks

sample(numbers, 10, replace = TRUE)  # allow repeats
```

□ Challenge: Draw 5 numbers without replacement and sort them ascending.

```
# Create a memo note in your R script if you did □ Identify, □ Look deeper, and □ Challenge.

## 1. Explain what you did in Required vs. Default, what your observations are, and what you expect
# to happen.

## 2. Demonstrate learning skill(s) in your memo
```

6. Nesting Functions

 Idea Feed the output of one function straight into another. Type the following R script in your document script and run.

```r
mean(sample(numbers, size = 5, replace = TRUE))  # sample then average
```

 Explore and Play: Run the line above twice. Why does the result change each time even though the code is identical?

 Look deeper: From the functions we have learned about, which can be nested and how?

```r
# Create a memo note in your R script if you did  Explore and Play and  Look deeper.

## 1. Explain what you did in Nesting Functions, what your observations are, and what you expect
# to happen.

## 2. Demonstrate learning skill(s) in your memo
```

7. ▯ Assignment

Now it's your turn to practice creating and using vector objects. Follow the tasks below to complete part of the **technical skill practice assignment**.

1. Work through each task in order. Replace the TODO with your code.
2. Run each completed line to be sure no errors appear and objects show in the Environment.
3. When finished, save your workspace and submit this script plus the .RData file.

7.1 Task 1

▯ Build a random data set

Set a reproducible seed (pick any whole number you like). Use sample() and seq() together to create a vector called scores that contains 25 random numbers between 60 and 100, inclusive. Include at least 3 NA values in scores (hint: combine vectors with c()).

TODO

7.2 Task 2

▯ Explore function arguments

Open the help file for sample() **in the console**. In the comment block below, list TWO required arguments and TWO default arguments for sample() exactly as they appear in the Usage section.

1. Required arguments: TODO
2. Default arguments: TODO

7.3 Task 3

▯ Handle missing data

Calculate the mean of scores and store it in average_raw. Calculate the mean of scores again, this time **excluding** NA values, and store it in average_clean.

TODO

7.4 Task 4

☐ Practice named vs. positional arguments

Use sample() to draw 10 values **with replacement** from scores and store them in sample_scores_positional **WITHOUT naming any arguments**. Repeat the same call, but this time **name every argument**; store as sample_scores_named.

TODO

7.5 Task 5

☐ Nesting functions

Combine mean() and sample() to calculate the mean of 5 random draws from scores (allow replacement); store the result in mean_of_five. Do this in **one** line of code.

TODO

7.6 Task 6

☐ Quick time-stamp

Use Sys.Date() and Sys.time() to create two separate objects, today_date and now_time.

TODO

7.7 Task 7

☐ Reflection

In one sentence, explain the difference between a *function object* (e.g., mean) and a *function call* (e.g., mean()).

Answer: TODO

Reminder: Now would be a great time to click save (or press Ctrl + S) to save your script.

8. Practice Space

The following provides you more opportunities to earn points demonstrating **learning skills**

☐ Link today's skills with vectors from Lesson 2. Review Lesson 2, create vectors and then apply functions to them.

☐ Build ONE new object of your choice that demonstrates any two skills from Lesson 3 (e.g., help file reading, required/default arguments, nesting). Name it my_object and leave a brief comment describing what it does.

☐ Explore and Play IDEAS:

1. Use runif() to create 10 random decimals, then round() them.
2. Nest sample() inside paste() to build a random silly password.
3. Open ?sd and compute the standard deviation of values after removing NA.

```
# Create a memo note in your R script if you did the Practice Space.

## 1. Explain what you did, what your observations are, and what you expect
# to happen.

## 2. Demonstrate learning skill(s) in your memo
```

9. Save and Upload

1. You will be submitting **both** the R script and the workspace file. The workspace file saves all the objects in your environment that you created in this lesson. You can save the workspace by running the following command in your R script:

save.image("Assignment3_Workspace.RData")

Or you can click the “Save Workspace” button in the Environment pane.

Always save the R script before closing. If you don’t, you will lose your work.

2. Find the assignment in this week’s module in Canvas and upload **both** the R script and the workspace file.

10. Today you practiced:

- Recognizing function objects vs. calls.
- Reading help files and spotting default arguments.
- Controlling na.rm and other named parameters.
- Creating reproducible random samples.
- Nesting functions for concise workflows.

□ Great progress!