Lesson 2

by Lorraine Gaudio

Lesson generated on September 05, 2025

BOISE STATE UNIVERSITY

Contents

1. ☒ Welcome back!

In lesson 1 you learned how to run code and create simple objects. Lesson 2 moves one step further: you'll meet *vectors* which are ordered collections of values that form the backbone of nearly everything in R.

To begin Lesson 2, 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 provided 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.

1.1 Lesson Overview

In the next few minutes you will learn how to:

- □ Build numeric, character, and logical vectors.
- □ Check the structure, length, and data type of any object.
- □ Explain how R coerces mixed□ type vectors.
- □ Subset vectors by position or logical tests.
- □ Summarize vectors with built□ in, *vectorised* functions.
- □ Remove objects you no longer need.

2. Functions

Many mathematical operations in R are performed using **functions**. A function is a reusable block of code that takes inputs, processes them, and returns an output. Functions can be built□ in or user-defined. Arguments are the inputs to a function, and they can be required or optional. Functions can return a single value or multiple values, and they can be nested within other functions. We'll be using functions throughout this course. We'll start with the very basics in this lesson to help up create vectors and a data frame.

3. Numeric Vectors

A **vector** is like a row in a spreadsheet: a single dimension of related values. Use c() to *combine*, : to create a simple sequence, or seq() for custom steps. Please type the following into your R script and run each line.

3.1 c()

```
numbers   <- c(4, 8, 15, 16, 23, 42)          # numeric via c()

numbers3  <- c(1.5, 2.5, -3.5)               # numeric with decimals and negative
```

3.2 seq()

Generating a sequence of numbers is useful for creating evenly spaced values. Use seq() to specify the start, end, and step size. Type the following into your R script and run each line.

```
numbers5  <- seq(from = 1, to = 10, by = 2)

steps     <- seq(from = 0, to = 1, by = 0.2)
```

☐ Check-in: What did seq do? Run "numbers3" in the Console to see the result.

3.3 :

Please type the following into your R script and run.

```
series    <- 1:10                    # shortcut 1,2,…,10
```

☐ Explore and Play: What are other ways you can create numeric vectors? For example, try rep() to repeat values.

```
# Create a memo note in your R script  Explore and Play.
# This is where you could earn points for demonstrating learning skills.
## 1. Explain what you did in creating vectors, what your observations are, and
# what you expect to happen.

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

4. Vector arithmetic

You can perform arithmetic operations on vectors. R will apply the operation element wise, meaning it will add, subtract, multiply, or divide corresponding elements in the vectors.

Recall that we created the objects:

```
numbers  <- c(4, 8, 15, 16, 23, 42)
steps    <- seq(from = 0, to = 1, by = 0.2)
```

What is the length of numbers and steps?

Let's do some vector arithmetic with these two vectors. Type the following into your R script and run each line.

```
numbers - steps
```

That's numbers minus steps.

5. ☒ Part I 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 ___ with your code.
2. Run each completed line to be sure no errors appear and objects show in the Environment.

5.1 Task 1

Build two numeric vectors where both vectors are the same length.

1. Create a vector called sales_q1 with FOUR numbers of your choice.
2. Create a vector called sales_q2 with FOUR numbers of your choice.

5.2 Task 2

Using vector arithmetic, create an object named growth that stores the difference between sales_q2 and sales_q1.

□ Explore and Play: What happens if you try to subtract a vector of different lengths?

```
# Create a memo note in your R script if you did the □ Explore and Play.
# This is where you could earn points for demonstrating learning skills.

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

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

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

□ Nice! Now, we'll dig a little deeper into other types of vectors.

6. Character Vectors

Frequently used as plot labels, text data, or categories. Use quotes to create character strings. Type the following into your R script and run each line.

```
chars  <- c("apple", "banana", "cherry")   # character vector

chars2  <- c("red", "yellow", "green")      # another character vector

chars3  <- c("A", "B", "C")                 # single letters as characters
```

□ Check-in: What happens if you don't put quotations around character strings? Try it with c(apple, banana, cherry) and see what R does. What happens if you put the entire vector in quotes like this c("apple, banana, cherry")? What happens if you quote numbers like c("1", "2", "3")?

```
# Create a memo note in your R script if you completed this □ Check-in.
# This is where you could earn points for demonstrating learning skills.

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

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

7. Missing Values

Sometimes data is missing or unknown. R uses NA to represent these missing values. You can create a vector with NA like this:

```
z <- c(1, 2, NA, 4, 5)  # vector with a missing value
```

□ Check-in: What happens if you run z + 2? What happens if you type 0/0 or Inf - Inf in the console?

□ Look deeper: What is Inf? Type ?Inf in the console. What happens?

Note: NaN stands for "Not a Number" and is used when a calculation does not produce a valid number.

```
# Create a memo note in your R script if you did the □ Check-in and □ Look deeper.
# This is where you could earn points for demonstrating learning skills.

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

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

8. Logical Vectors

Logical vectors are used for conditions and comparisons. They contain TRUE or FALSE values.

```
logicals  <- c(TRUE, FALSE, TRUE, NA)          # logical (TRUE/FALSE)

logicals2 <- c(1 > 2, 3 < 5, 4 == 4)           # logical comparisons
```

The logical operators are:

- > greater than
- < less than
- == equal to
- >= greater than or equal to
- <= less than or equal to
- != not equal to

Additionally, if logical and logical2 are logical expressions, then logical & logical2 is their intersection (*and* operator), and logical | logical2 is their union (*or* operator). We'll cover logical vectors in more detail in a later lesson.

 Check-in: Run each name (numbers, chars, …) in your environment by typing the object name. Note how R prints them. After running, peek at the Environment (Grid view) to see each vector shows its type & size.

```
# Create a memo note in your R script if you did the  Check-in.
# This is where you could earn points for demonstrating learning skills.

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

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

9. Indexing

Indexing is how you access specific elements in a vector. R uses 1-based indexing, meaning the first element is at position 1. You can use square brackets [ ] to index vectors.

Recall that we created:

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

numbers3  <- c(1.5, 2.5, -3.5)
```

9.1 Sub-setting Vectors

□ **Goal** Pull out the pieces you need. Use [ ] with positions or a logical test.

□ Remember: positions start at 1 in R (not 0).

Type the following into your R script and run each line.

```
z[1]      # first element by position

z[3]      # third element by position

z[2:4]    # last two elements by position

z[-c(1, 5)] # all but first and last elements
```

□ Break Things! What happens if you try to index with a number larger than the vector length, like z[6]? □ Explain any error message you see.

□ Challenge: Extract the last two fruits from *chars*.

9.2 Logical Indexing

Type the following into your R script and run.

```
numbers3[numbers3 > 0] # keep only positives
```

 Look deeper: What happens if you runy <- z[(!is.na(z))]? Hint: The function is.na() checks for missing values (NA) in the vector. The ! operator negates the logical vector. Print y. What happened? Explain

```
# Create a memo note in your R script if you did  Break Things and  Challenge
# Or if you did the  Look deeper, and  What happened?.
# This is where you could earn points for demonstrating learning skills.

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

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

10. Inspecting Vectors

Before using data, we know confirm *what* it is. str() gives a compact summary, length() counts elements, and class() reveals type. Please type the following into your R script and run each line.

10.1 Length

Type the following into your R script and run.

```
length(chars)
```

☐ Explore the length of objects in the Environment pane.

10.2 Class

class(logicals)

☐ Explore the class of objects in the Environment pane.

Some types of classes in R include:

- **numeric**: numbers (e.g., 1, 3.14)
- **character**: text strings (e.g., "Hello World", "R is fun")
- **logical**: TRUE or FALSE values
- **integer**: whole numbers (e.g., 1L, 2L)
- **factor**: categorical data (e.g., "red", "blue") is a special type of character vector used for categorical data.
- **NULL**: represents an empty object (e.g., NULL)
- **complex**: complex numbers (e.g., 1 + 2i)
- **Index**: a special type of vector used for indexing (e.g., 1:10)
- **list**: a collection of objects (e.g., list(a = 1, b = "text"))
- **matrix**: a two-dimensional array (e.g., matrix(1:6, nrow = 2))
- **array**: a multi-dimensional array (e.g., array(1:12, dim = c(2, 3, 2)))
- **data.frame**: a table-like structure (e.g., data.frame(name = c("Alice", "Bob"), age = c(25, 30)))

10.3 Structure

Type the following into your R script and run.

```
str(numbers)    # shows type = num, length = 6
```

□ Look deeper: What other functions can you use to inspect objects?

```
# Create a memo note in your R script if you did □ Look deeper.
# This is where you could earn points for demonstrating learning skills.

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

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

11. Part II Assignment

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

1. Work through each task in order.
2. Run each completed line to be sure no errors appear and objects show in the Environment.

11.1 Task 3: Inspect

1. Use class() and str() to inspect the object growth.
2. Write a one-line comment describing what str() reveals.

11.2 Task 4: Subset

1. Create a new vector positive_growth that contains only values in growth that are greater than zero. (hint: see logical vectors)
2. Create a new vector last_two that contains the last TWO elements of sales_q2 (use numeric positions, not names).

 Nice! Next, lets try to break some rules and learn more functions.

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

12. Data Coercion

 Key Idea A vector can only hold *one* type. Mixing types triggers *coercion* to the most flexible type (logical → numeric → character).

What happens if you mix numbers and characters?

```
mixed <- c("candy", 4, "book", 92)

class(mixed)          # character: all elements coerced to character
```

The as.numeric() function tries to convert characters to numbers. If it can't, it returns NA (not available).

```
as.numeric(mixed)          # character → numeric: text converts to NA

as.numeric(c("1", "2", "3")) # character → numeric: text converts to numbers
```

 Look deeper: What are other ways to convert class types?

```
# Create a memo note in your R script if you did  Look deeper.
# This is where you could earn points for demonstrating learning skills.

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

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

13. Vectorized Functions

 Why it's cool Functions like sum(), mean(), and sd() automatically loop over every element—no for-loop needed. This is called *vectorization*.

 Look deeper: What is a for-loop?

Recall we created series <- 1:10

Type the following R code into your script and run. Comment about the output.

```
sum(series)

mean(series)

sd(series)
```

14. Cleaning Up

 When to sweep Large projects can crowd the Environment. Use rm() to remove objects you no longer need and free memory. Type the following into your R script and run.

```
rm(mixed)      # mixed disappears from the Environment pane
```

15. Dataframe function

Vectors form the foundation for data frames.

 What & Why A data frame is like a spreadsheet: a table of rows and columns. Use data.frame() to create one from vectors of the same length. Each vector becomes a column.

```
column_1 <- c(1, 2, 3)

column_2 <- c("A", "B", "C")

column_3 <- c(TRUE, FALSE, TRUE)

data_frame <- data.frame(column_1, column_2, column_3)
```

16. Part III Assignment

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

1. Work through each task in order.
2. Run each completed line to be sure no errors appear and objects show in the Environment.

16.1 Task 5: Vectorised summaries

1. Calculate the average of sales_q2; save as avg_q2.
2. Calculate the total of sales_q1; save as total_q1.

16.2 Task 6: Mini data frame

1. Create a character vector months that names four months (e.g., "Jan").
2. Combine months, sales_q1, sales_q2 and growth into a data frame called sales.
3. Check the structure of sales with str(). Explain what it shows.

16.3 Task 7 Reflection

 In one sentence, explain why numeric values may be coerced to character if you mix them inside a single vector.

17. 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("Assignment2_Workspace.RData")
```

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

Always save the R documents 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.

18. Today you practiced

- Building numeric, character, and logical vectors.
- Inspecting structure, length, and data type.
- Understanding and witnessing type coercion.
- Sub-setting vectors with positions and logical tests.
- Applying vectorised summary functions.
- Tidying your workspace with `rm()`.
- Creating a data frame from vectors.

 Great progress! In the next lesson, we'll learn more about functions.