Lesson 7

by Lorraine Gaudio

Lesson generated on August 21, 2025

BOISE STATE UNIVERSITY

Contents

1. More Subsetting in R!

In lesson five, you learned about subsetting, extraction, and insertion in data frames. This week, we cover advanced subsetting in base R. Clean analysis starts with pulling out the exact rows and columns you need. R's logical operators (>, <, ==, !=, & , |, %in%) and how they power precise subsetting with brackets [rows , cols].

To begin Lesson 7, follow these steps:

1. Open your course project for RStudio
2. Create a new file. Today, let's try "Quarto Document" (File > New File > Quarto Document).
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.

Lesson Overview

By the end of Lesson 7 you will be able to:

1. Remember – List R's primary logical operators and their purpose.
2. Understand – Describe how logical tests return TRUE/FALSE/NA.
3. Apply – Use logical conditions inside [ ] and which() to subset
4. Analyze – Compare results from different operators or thresholds.
5. Evaluate –Choose operators and subsetting strategy for a goal.

Keep these goals in mind as you move through each section.

2. Quick Warm□up

We will use the built-in mtcars dataset. View it for context.

```
data("mtcars")
?mtcars      # help file (opens in Help tab)
View(mtcars) # spreadsheet view (top□ left pane)
```

□ Hypotheses then Test: What will mtcars[, -c(8,9)] return? □ Give your best guess and explain why that is your guess. □ Test and explain the results. Create a memo note, demonstrate learning skill(s) used.

3. Excluding Columns

 Review: Subsetting template: object[ROWS , COLS]

Type the following code in a new code chunk and run.

```
no_cat <- mtcars[ , -c(8, 9)]
head(no_cat)
```

 NOTICE that the row names (car models) and remaining columns are untouched.

 Up Next! Logical operators: the key to subsetting data frames.

Operator	Meaning
&	and
|	or
%in%	in set
!	not
==	equal to
!=	not equal to
>	greater than
<	less than
>=	greater *or* equal
<=	less *or* equal

4. Equality and Inequality

== asks "is exactly equal to?"

!= asks "is NOT equal to?".

Both return a logical vector.

```
mtcars$cyl == 6      # TRUE where cylinders equal 6
mtcars$gear != 4     # TRUE where gears are NOT 4
```

Check in: How many cars in the dataset have mpg ≤ 15?

```
sum(mtcars$mpg <= 15)
```

Explore and Play: Create your own questions based on the available data in mtcars *then* Write the script to answer those questions. Create a memo note, demonstrate learning skill(s) used.

Other applications: Using these TRUE/FALSE results inside [ ] keeps only rows that satisfy the test.

```
# Example 1: cars with exactly 6 cylinders
six_cyl <- mtcars[mtcars$cyl == 6, ]
head(six_cyl)
```

```
# Example 2: cars with average mile per gallon over 22.
high_mpg <- mtcars[mtcars$mpg > 22, ]
# Verify
unique(high_mpg$mpg)
```

Remember how negative column indices work?

```
selection <- mtcars[mtcars$hp >= 100, -c(3, 5, 8:9)]
head(selection)
```

"-c(3,5,8:9)" drops columns 3, 5, 8, 9. This keeps other variables.

5. **which()**: From Logical to Position

which() converts TRUE positions to row indices (numbers). This avoids errors when NA is present and is handy for re□ use.

```
rows_fast <- which(mtcars$qsec < 16)  # quarter□ mile < 16 sec
mtcars[rows_fast, c("mpg", "qsec")]  # inspect subset
```

6. Combining Conditions

6.1 And &

□ Goal: automatic (am == 0) *and* average miles per gallon is less than or equal to 22.

Type the following code in a new code chunk and run.

```
auto_efficient <- mtcars[mtcars$am == 0 & mtcars$mpg >= 22, ]
auto_efficient
```

□ Explanation: both tests must be TRUE for a row to survive.

6.2 Or |

□ Goal: cars with 4 *or* 8 cylinders.

Type the following code in a new code chunk and run.

```
four_or_eight <- mtcars[mtcars$cyl == 4 | mtcars$cyl == 8, ]
```

□ Common Pain Point: using | where you meant & (or vice□ versa). Think in words first: "I want rows that are both A **and** B" □ use &. "I want rows that are either A **or** B" □ use |.

6.3 Membership %in%

Same task as above but scalable.

```
four_or_eight_2 <- mtcars[mtcars$cyl %in% c(4, 8), ]
```

□ Compare nrow(four_or_eight) to nrow(four_or_eight_2).

```
nrow(four_or_eight) == nrow(four_or_eight_2)
```

□ Look deeper: How are | and %n% similar and different from each other? □ Create a memo note, demonstrate learning skill(s) used.

6.4 Putting It All Together

 Scenario: Pick cars that weigh < 3,000 lb, have hourse power between 110 and 180, and cylinders {4,6}.

```
light_midpower <- mtcars[mtcars$wt < 3 &
                         mtcars$hp >= 110 & mtcars$hp <= 180 &
                         mtcars$cyl %in% c(4, 6), ]
light_midpower
```

 Common Pain Point: Forgetting weight is in 1000s of lbs.

```
snack_box[1:10, "Shape"]
```

 Practice: fill in the blank prompt

```
# Replace the blanks and run:
sporty <- mtcars[mtcars$am == __ & mtcars$qsec < __ , ]

nrow(sporty)
```

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 ___ placeholder with your code or short written answer.
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 R Markdown file (RMD) plus the .RData file.

7.1 Task 0

Setup: Load the built-in data set mtcars and take a quick look. Identify what the columns contain and meaning of that content.

```
data("mtcars")   # already in memory but this keeps the workflow explicit
?mtcars        # help file for variable descriptions
View(mtcars)   # spreadsheet view (optional)
```

7.2 Task 1

Time Traveler

I want to travel back to the '70s and buy a *very* thirsty muscle car. Create a data frame named **Time_Traveler** that keeps cars that less than 11 mile per gallon **AND** have 8 cylinders, **AND** weigh more than 4000 lb. After creating the object, print rownames to see which cars are thirsty.

```
# TODO 1 A: Write the subsetting code
Time_Traveler <- ___
```

```
# TODO 1 B: Display the matching models.
                  # this should list 2–3 cars when correct
```

7.3 Task 2

☐ Negative Indexing

Run the chunk below, then describe what the *selection* holds and how negative column indices work.

```
mtcars              # for comparison
selection <- mtcars[ mtcars$hp >= 100 , -c(3, 5, 8:9) ]
selection
```

Type your brief explanation describing the selection and how negative column indices work.

☐ EXPLANATION: "___"

7.4 Task 3

☐ Say Hello to iris

We will switch datasets for a moment. Explore *iris* so you know its layout.

```
data("iris")
?iris
View(iris)
```

```
# Explore iris
```

7.5 Task 4

Long-and-Short Virginica

Create **Long_and_Short_Virginica** that keeps Species equals "virginica", **AND** Petal.Length greater than 6.0 **OR** Petal.Length less then 5.0. Keep *all* columns.

```
# TODO 4: Write the extraction line.
Long_and_Short_Virginica <- ___
```

```
# Check your work
# option:
nrow(Long_and_Short_Virginica)
```

7.6 Task 5

 stringr Detective

Install/load *stringr* and re run the supplied code. Explain what str_detect() accomplished.

```
# install.packages("stringr")  # uncomment if first time
library(stringr)
?str_detect

mtcars                  # for comparison
mtcars[str_detect(row.names(mtcars), "Merc"), ]
```

Type your brief explanation explaining str_detect and how it worked in this script

 EXPLANATION: "___"

8. Save and Upload

1. You will be submitting **both** the R Markdown 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 a code chunk of the R Markdown document:

```
save.image("Assignment7_Workspace.RData")
```

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

 Always save the R documents before closing.

2. Find the assignment in this week's module in Canvas and upload **both** the RMD and the workspace file.

9. Today you practiced:

- Learned R's logical operators and how they create TRUE/FALSE vectors.
- Practiced negative indices to drop columns.
- Used which() to convert logical results into row positions.
- Combined conditions with &, |, and %in% for flexible subsetting.
- Built step□ by□ step filters to answer targeted questions about mtcars and iris.

□ Keep experimenting! Logic is the engine behind every data filter you will ever write in R. In our next lesson, we explore subsetting with the with dplyr package.