Lesson 8

by Lorraine Gaudio

Lesson generated on October 10, 2025

BOISE STATE UNIVERSITY

Contents

1. Welcome back to R!

In lesson 6 and 7, you learned about subsetting, extraction, and insertion in data frames using base R functions. Today, you will learn easier subsetting with dplyr (select() & filter()). The dplyr package simplifies column and row extraction, making code shorter and easier to read than base-R brackets. Mastering select(), filter(), and the pipe %>% will speed up every day tasks.

To begin Lesson 8, 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 8 you will be able to:

1. Remember – List what select(), filter(), and %>% do.
2. Understand – Explain how select() (columns) differs from filter() (rows).
3. Apply – Use select() and filter() to subset a data frame.
4. Analyze – Chain multiple verbs with %>% to answer a real question.
5. Evaluate – Decide when dplyr is preferable to base-R brackets.

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

2. 📦 Packages

📌 Recall that we need to download packages **once**. If needed, install the required packages we will use in this lesson. After installation, comment # out the install.packages code.

```
#install.packages("dslabs")
install.packages("dplyr")
```

After installing the package, you need to load it into your R session using the library() function. Each time you open R, you will need to "library" the packages. This makes the functions in the package available for use. Type the following code in a new code chunk and run.

📌 Pro tip: You can load multiple packages in one library() call by separating package names with commas.

```
library(dslabs, dplyr)
```

3. The Pipe (%>%)

 The pipe operator passes the result of the left side into the first argument of the right side, reading like “and then”. It cuts repetition and keeps code in logical order.

```
# Test
mtcars %>%        # start with data
  head()          # shows first six rows
```

 Link Concepts: What other ways have you learned to write code that will view the first six rows of a data frame? Write this code and a memo to demonstrate learning skills.

4. Selecting Columns

□ The GOAL of select() is to keep or drop specific **columns**.

□ The SYNTAX is select(data, col1, col2, …)

4.1 Positive Selection

□ Goal: keep specific columns by name.

```
#□ Test
select(mtcars, mpg, cyl, hp)    # keep three columns
```

4.2 Negative Selection

□ Goal: remove columns with a leading -.

```
select(mtcars, -gear)   # drop one column
```

4.3 Piping with Select

□ Goal: Let's add a pipe!

```
mtcars %>%
  select(mpg, cyl, hp)              # easier to read
```

4.4 Subset with Select

We'll mix it up by using a different data set, gapminder from dslabs. This data set contains information about countries over time.

```
?gapminder
```

□ Goal: Create a copy of the gapminder dataset that contains only specific columns!

```
# Test:
I_Demo <- gapminder %>%
  select(country, year, infant_mortality, continent)
```

```
# Verify
head(I_Demo)
```

Explain: A new dataset called I_Demo was created from the gapminder dataset that only contains the columns country, year, infant_mortality, and continent.

Practice: fill in the blank prompt

You try to design your own subsetting operation.

Goal: Write an example that selects only the columns you choose.

```
# Test:


# Verify
```

Explain: What columns did you choose and why? Create a memo note, demonstrate learning skill(s) used.

5. Filtering Rows

 The GOAL of filter() is to keep or drop specific **rows** based on conditions.

 The SYNTAX is filter(data, condition1, condition2, …)

5.1 Single Condition

 filter() uses logical operators!

```
# Test:
filter(mtcars, mpg >= 25)          # high mileage cars
```

5.2 Piping with Filter

We'll mix it up by using a different data set, murders from dslabs. This data set contains information about gun murders in the US.

```
?murders
```

 Goal: Create a copy of the murders dataset that excludes Washington state.

 Prediction: 50 states

```
# Test:
nrow(murders)
```

 Explain: … Wait, what? … Question: What are these 51 states?

```
# Test:
unique(murders$state)  # 51 states
```

 Explain: Ohhhhh! "District of Columbia" is included in the dataset.

Back to the Goal: Create a copy of the murders dataset that excludes Washington state.

```
# Test:
No_Washington <- murders %>%
  filter(state != "Washington")
```

```
# □ Verify
nrow(No_Washington)  # 49 states
```

□ Explain: A new dataset called No_Washington was created from the murders dataset that excludes Washington state.

□ Review: What is the difference between == and !=?

□ Break Things! What happens when you use = instead of ==? or =! instead of !=? Create a memo note, demonstrate learning skill(s) used.

5.3 Multiple Conditions with AND

□ In filter(), commas = AND (&).

□ Goal: Let's filter with two conditions and while we are at it, let's add a pipe!

□ Question: How many cars have *both* mpg ≥ 25 *and* 4 cylinders?

```
# □ Test:
mtcars %>%
  filter(mpg >= 25, cyl == 4)          # two conditions
```

□ Verify: How many cars have *both* mpg ≥ 25 *and* 4 cylinders? What code would you use to verify the success of this filter? Create a memo note, demonstrate learning skill(s) used by writing and running the code.

5.4 Multiple Conditions with OR

□ In dplyr::filter(), the OR operator is the same as base R | or %in%.

5.4.1 |

□ Goal: Let's filter with two conditions using OR (|).

□ Question: How many cars have *either* mpg ≥ 25 *or* 4 cylinders?

```
# □ Test:
mtcars %>%
  filter(mpg >= 25 | cyl == 4)          # two conditions
```

□ Verify: How many cars have *either* mpg ≥ 25 *or* 4 cylinders? What code would you use to verify the success of this filter? Create a memo note, demonstrate learning skill(s) used by writing and running the code.

5.4.2 %in%

 Goal: The %in% operator is used to check if a value is in a set. It's like "is this value one of these?"

```
#  Test
filter(mtcars, cyl %in% c(4, 6))        # 4 or 6 cylinders
```

6. filter() AND select()

 The GOAL of combining filter() and select() is to **subset** a data frame by both rows and columns.

 Best practice: filter rows first, then select columns.

 The SYNTAX is data %>% filter(condition) %>% select(col1, col2, …)

We'll mix it up by using a different data set, us_contagious_diseases from dslabs. This data set contains information about contagious diseases in the US.

```
?us_contagious_diseases
```

```
# Test
us_contagious_diseases %>%
  filter(state == "Alaska", year >= 1980) %>%   # rows
  select(year, disease, count)                  # columns
```

 Explore and Play: What are the unique diseases in this Alaska subset? Use skills you developed in previous lessons to explore this data set. Compare filter and select with subsetting with brackets [ ]. Create a memo note, demonstrate learning skill(s) used.

7. One Application

 Let's say you have the research question, "Which automatic cars (am == 0) in mtcars have horsepower > 150? Goal: Your report should show only hp and mpg.

```
# Test
mtcars %>%
  filter(am == 0, hp > 150) %>%
  select(hp, mpg) %>%
  head()
```

 Explain:

1. filter() keeps matching rows.
2. select() trims to needed columns.
3. head() previews results. Bonus: Add row count for exploration.

8. Duplicates & Sampling

Sometimes, you may need to remove duplicate rows or sample a subset of rows. unique() removes duplicate rows; slice_sample() picks random rows. Type ?slice_sample() to learn more about this dplyr function.

 Goal: We'll mix it up by using a different data set, movielens from dslabs. This data set contains information about movies and ratings.

```
?movielens
```

```
nrow(movielens)   # how many rows at start?
```

```
#  Test
Movie_Night <- movielens %>%
  filter(rating >= 4) %>%        # keep high ratings
  select(title, rating) %>%      # keep title and rating
  unique()                       # remove duplicates
```

```
#  Verify
nrow(Movie_Night)  # how many rows after filter, select, and unique?
```

Filter kept only the rows with ratings 4 or higher. Select kept only the title and rating columns. Unique removed duplicate rows, so each movie title appears only once.

 Question: What movie should I watch on movie night?

 Goal: Let's randomly sample 10 movies with ratings four or higher.

```
set.seed(420)
#  Test
Movie_Night %>%
  slice_sample(n=10)  # random sample of 10 rows
```

 Comment: What movie would you choose?

 Explore and Play: What would happen if we didn't use unique()? Why might unique be useful for our question? Create a memo note, demonstrate learning skill(s) used.

9. Extra Practice

 Practice: Demonstrate learning skill(s) by completing this bonus practice. Fill the blanks. Use the gapminder dataset.

```
___ %>%
  filter(life_expectancy >= quantile(life_expectancy, ___)) %>%
  select(country, ___, life_expectancy) -> Top_LifeExp
View(Top_LifeExp)
```

 Explain: What is the -> Top_LifeExp do in the code above?

 Question: What is the highest life expectancy in this subset?

 Question: How many countries are in the Top_LifeExp data frame?

 Question: What is the average life expectancy in this subset?

10. ☒ Assignment

Replace each ____ placeholder (and any TODO comments) with working code or a short written answer. Run each section; be sure the requested objects appear in the Environment. When finished, save **BOTH** your script (which should include your lesson notes and completed assignment) and your .RData workspace and upload to Canvas Assignment 8 in Module 8.

10.1 Task 0

☐ Make sure dplyr and dslabs packages are attached so their functions / datasets load.

10.2 Task 1

Column Selection

☐ Create **C_Overload** that keeps only year and carbon_emissions from the temp_carbon data frame. Use select() with or without %>%.

```
____ <- _____
```

10.3 Task 2

Single-Column Extraction

☐ Pipe the stars dataset in dslabs through select() to keep only the star column. Name the result **star_names**.

```
____ <- _____
```

10.4 Task 3

Row Filtering

☐ From murders, remove every row where state equals "Florida". Name the object **No_Florida** (any method is fine).

```
____ <- _____
```

10.5 Task 4

Multiple Conditions

 From mtcars, use filter() + %>%, pull cars in mtcars that are manual, have 8 cylinders, and have qsec < 15. Store as **Fast_n_Furious**.

```
summary(mtcars$qsec)  # check qsec range
```

Is qsec < 15 within range?

```
____ <- _____
```

10.6 Task 6

OR Logic + Column Slice

In mtcars, show only rows where disp > 250 OR disp < 160 and display just the first three columns.

```
____ <- _____
```

10.7 Task 7

Reflection

 Write a few sentences comparing dplyr verbs over base-R brackets (from lesson 6 & 7) for subsetting.

11. Save and Upload

0. Self-check checklist:

For the assignment protion, your environment should contain the following items:

- C_Overload exists and has only year and carbon_emissions.
- star_names exists and has one column named star.
- No_Florida has 0 rows with state == 'Florida'.
- Fast_n_Furious with rows that only contain manual, 8 cylinders, and where qsec < 15.
- **All code in your Quarto document runs top-to-bottom without errors.**

1. You will be submitting **both** the Quarto document 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 Quarto Document document:

```
save.image("Assignment8_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 .qmd and the workspace file (RData).

12. Today you practiced:

- Installed / loaded packages for extra tools.
- Used the pipe %>% to chain steps left-to-right.
- Selected columns with select() (including negative selection).
- Filtered rows with filter() using logical tests and %in%.
- Combined verbs to answer data questions concisely.

 Great job! Keep practicing piping and you'll soon write clear, readable R code!