Lesson 8

by Lorraine Gaudio

Lesson generated on August 21, 2025

Contents

1. 􀀀 Welcome back to R!

In lesson 6 and 7, you learned about 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 data tasks.

To begin Lesson 8, follow these steps:

1. Open your course project for RStudio
2. Create a new file. From the file types we have used so far, pick which file type you want to use. (File > New File > ???).
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.

```
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.

```
library(dslabs)  # Data science labs package
library(dplyr)   # Data manipulation package
```

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.

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

□ Compare the above to head(mtcars). Create a memo note, demonstrate learning skill(s) used.

4. Selecting Columns

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

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

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

☐ Use –col_name to exclude to remove column(s).

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

☐ Link Concepts: Let's add a pipe!

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

☐ Link Concepts: Let's rename the data frame!

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

```
?gapminder
```

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

5. Filtering Rows

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

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

□ filter() uses logical operators!

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

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

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

□ Link Concepts: Let's add a pipe!

□ Commas act like &.

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

□ Explore and Play: How many cars have *both* mpg ≥ 25 *and* 4 cylinders? Use skills you developed in previous lessons to explore this data set. Create a memo note, demonstrate learning skill(s) used.

□ Link Concepts: Let's rename the data frame!

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
```

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

□ Identify: 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.

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
```

```
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. Create a memo note, demonstrate learning skill(s) used.

7. Application

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

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

Explanation

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; sample_n() picks random rows.

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

```
?movielens
```

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

sample_n(Movie_Night, 10)              # random sample of 10 rows
```

☐ Comment: What would happen if we didn't use unique()? Create a memo note, demonstrate learning skill(s) used.

☐ Challenge: 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)
```

☐ Check☐ in: What is the highest life expectancy in this subset?

☐ Check☐ in: How many countries are in the Top_LifeExp data frame?

☐ Check☐ in: What is the average life expectancy in this subset?

9. ▯ 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** this script and your .RData workspace and upload.

9.1 Task 1

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

9.2 Task 2

Column Selection

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

```
____ <- _____
```

9.3 Task 3

Single-Column Extraction

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

```
____ <- _____
```

9.4 Task 4

Row Filtering

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

```
____ <- _____
```

9.5 Task 5

Multiple Conditions

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

```
?mtcars
```

```
____ <- _____
```

9.6 Task 6

OR Logic + Column Slice

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

9.7 Task 7

Reflection

□ Write a short paragraph reflecting at least advantage of using dplyr verbs over base-R brackets for subsetting.

10. Save and Upload

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:

```r
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 RMD and the workspace file.

11. 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!