Lesson 9

by Lorraine Gaudio

Lesson generated on August 21, 2025

Contents

1. ⌧ Clean up on aisle R!

In lesson 8, we learned about about subsetting with dplyr (select() & filter()). This week, we will look at other functions in the dplyr package (mutate()and ifelse()) . Real-world data often needs *new* or *cleaner* variables before analysis. The **dplyr** verb mutate() lets you create or modify columns, while base□ R ifelse() supplies the logic for quick recoding. Together they form a duo you will use in nearly every data project.

To begin Lesson 9, 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 9 you will be able to:

1. □ Remember – State the purposes of mutate() and ifelse().
2. □ Understand – Explain how ifelse() replaces values based on a test.
3. □ Apply – Overwrite or add columns using mutate() + ifelse()
4. □ Analyze – Build multi□ level recodes with nested ifelse().
5. □ Evaluate – Choose between overwriting vs. keeping the original data.

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

2. 󠀠 Packages

Install once (if needed): install.packages("dplyr"); install.packages("dslabs")

Load the packages at the start of every session:

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

3. Warm￼Up

1. Logical Test Review

```r
# Run and read: TRUE appears where the test is met.
c(1, 2, 3, 4, 5) > 3
```

☐ Reflect: How many TRUE values do you expect? Run and read: TRUE appears where the test is met.

2. sample() Review

Recall sample() is a base-R function that randomly selects elements from a vector.

☐ Practice using sample()

4. sample_n()

 The SYNTAX:sample_n(data, size, replace = FALSE, weight = NULL)

```
sample_n(mtcars, 5)  # Sample 5 rows from mtcars data frame
mtcars %>% sample_n(10)  # Using with pipe
```

We'll come back to sample_n() latter in this lesson.

5. **ifelse()**

ifelse() tests the data.

☐ The SYNTAX: ifelse(test, yes, no)

- *test* – logical vector (TRUE/FALSE/NA)
- *yes* – value to insert where test is TRUE
- *no* – value to insert where test is FALSE or NA

```r
ifelse(test = c(1, 2, 3, 4, 5) > 3, yes = "Big", no = "Small")
```

☐ Check-in: Which numbers were labeled “Big”?

6. Simple Recoding

```r
object <- c("Yellow", "Jello", "Gefilte Fish")
ifelse(object == "Yellow", yes = "color", no = "not a color")
```

☐ Explore and Play: Try swapping the test to identify food words.

7. Multi‐Level Recoding

Nested ifelse()

```
object2 <- c(1, 1, 2, 2, 3, 3, 4, 4, 5, 5)
ifelse(object2 == 1, "One",
       ifelse(object2 == 2, "Two",
              ifelse(object2 == 3, "Three",
                     ifelse(object2 == 4, "Four", "Five"))))
```

☐ Reflect: Why didn't we need an explicit test for 5?

8. mutate()

8.1 Overwrite Columns

mutate() is a function that adds new columns or modifies existing ones

 The SYNTAX: mutate(data, new_col = value)

1. Create a copy of a column

```r
names(mtcars) # Check the column names
```

```r
mtcars2 <- mtcars %>%
  mutate(new_column_copy = mpg)
```

```r
names(mtcars2) # Check the column names again
```

2. Overwriting approach - replaces original column

```r
cars_overwrite <- mtcars %>%
  mutate(am = ifelse(am == 0, "Automatic", "Manual"))
```

3. Keeping original approach - creates new column

```r
cars_keep <- mtcars %>%
  mutate(transmission = ifelse(am == 0, "Automatic", "Manual"))
```

 Comment: What is the difference between the code that creates cars_overwrite and cars_keep?

```r
cars_overwrite$am
```

```r
cars_overwrite %>% select(am)
```

```r
cars_keep$transmission
```

8.2 Overwrite with Factors

Let's create a sample of the movielens dataset using sample_n() in to practice overwriting a column variables.

```
data("movielens")
?movielens
```

```
set.seed(99)

# Sample 100 movies from the movielens dataset
smaller_movies <- movielens %>%
  sample_n(100) %>%
  select(title, rating)

head(smaller_movies)
```

We'll apply mutate() to smaller_movies to recode the rating column to "Good" or "Bad".

 The SYNTAX: mutate(data, new_col = ifelse(test, yes, no))

```
names(smaller_movies) # Check the column names)
movies_overwrite <- smaller_movies %>%
  mutate(rating = ifelse(rating >= 4, "Good", "Bad"))
head(movies_overwrite)
```

 Overwriting keeps the column name but changes its contents.

8.3 Bonus Example

```
smaller_movies %>%
  mutate(rating_label = ifelse(rating >= 4, "Good", "Bad"))
head(movies_newcol) -> movies_newcol
```

 NOTICE: What is happening here -> movies_newcol?

 Break Things! How can you reorganize this script?

8.4 Calculations

 Using mtcars, let's use create a column l_per_100km that converts from US units (miles per gallon) to metric units (liters per 100km). 235.215 is the conversion factor between these units

- round(..., 1) rounds to 1 decimal place

```
mtcars_units <- mtcars %>%
  mutate(l_per_100km = round(235.215 / mpg, 1))
head(mtcars_units)
```

 Check in: Which car is most fuel efficient in l/100 km terms?

9. □ Practice Space

□ Practice: Recode the mpg column. Use mutate() and ifelse()

Fill in the Blanks

```r
mtcars %>%
  ______(mpg_class = ______(mpg > 25, "High MPG",
                          ______(mpg < 15, "Low MPG", "Medium MPG"))) -> mpg_recode
View(mpg_recode)
```

10. 󰞗 Assignment

Replace each ____ placeholder 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.

10.1 Task 1

☐ Library it up!

Make sure there is script in your document to load dplyr and dslabs packages so their functions / datasets load.

10.2 Task 2

☐ Recode ranges (overwrite)

Using quakes, recode mag into THREE ordered levels of *your choice* (e.g., "Low", "Mid", "High") **overwriting** the original column. Use mutate() and ifelse() chain and pipe (%>%). Store as **quake_recode**.

```
____ <- _____
```

```
# Quick check
table(quake_recode$mag)   # should show exactly 3 levels
```

10.3 Task 3

Recode categories (new col)

☐ With stars, turn the one-letter *type* into a descriptive phrase of your own in a **new** column called full_type. Keep the original type. Store as **stars_recode**.

```
____ <- _____
```

```
# Quick check
head(select(stars_recode, type, full_type))
```

10.4 Task 4

Unit conversion

Create international_friendly_heights by adding a height_cm column to heights (inches × 2.54). Leave original height.

```
____ <- _____
```

- [ ] Reflect: Why add—not overwrite—the height column?
- [ ] EXPLANATION: "___"

11. 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:

```
save.image("Assignment9_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.

12. Today you practiced:

- Practiced logical tests and the structure of ifelse().
- Created binary and multi□ level recodes.
- Used mutate() to overwrite or create columns.
- Performed inline calculations for unit conversion.

□ Continue experimenting. Recoding and mutation are the building blocks of clean, analysis□ ready data!