Lesson 12

by Lorraine Gaudio

Lesson generated on August 21, 2025

BOISE STATE UNIVERSITY

Contents

1. (maybe a dice icon?) Welcome Back to R!

We shift (gears) from the dyplr functions to using R's built in random number tools to make simuations. Simulation lets you create synthetic data, test ideas quickly, and understand probability without a math degree.

 No extra packages are needed today. We'll be using built-in stats functions.

To begin Lesson 12, 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 12 you will be able to:

1. Remember – Name sample(), rnorm(), and replicate().
2. Understand – Explain when to use replacement and probability weights in sample().
3. Apply – Simulate coins, dice, and normal data with code
4. Analyze – Visualize simulated results with hist() and interpret shapes.
5. Evaluate –Judge whether simulated outcomes match an expected distribution (fair vs. biased).

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

2. Warm-Up

- Peek at the help file

```
?sample      # skim Usage and Arguments
```

- NOTICE: Which argument controls replacement (TRUE/FALSE)?

3. Simulating Coin Flips

□ The GOAL: Flip 1,000 fair coins (0 = heads, 1 = tails) and count tails.

```
set.seed(123)  # for reproducibility
```

□ Look deeper: How does set.seed() work? Do you need to set seed every time before running a simulation? What happens when you don't set seed? What does it mean to "re-run without set.seed()"?

1. Start with the data

```
# sample(x, size, replace = FALSE, prob = NULL)
coins <- sample(x = 0:1, size = 1000, replace = TRUE)
```

□ Explanation: replace = TRUE because we "put the coin back" each flip.

```
head(coins)  # first few flips
```

□ Comment: What type of object is coins?

How many tails are in our data set? We use sum to count the number of tails because 1 = tails

```
sum(coins)
```

□ Reflect: How might we calculate the number of heads using coins?

□ Explore and Play: Can you make a new data set where heads = 1?

□ Break Things! What happens if replace = FALSE and the size = 1000?

4. Dice Rolls and Histograms

 The GOAL: Simulate 10,000 rolls of a fair 6 sided die and plot the results.

```
dice <- sample(1:6, 10000, replace = TRUE)
```

hist() creates a histogram of the dice rolls, showing the frequency of each face value.

```
?hist
```

```
hist(dice,                # The data to plot
    breaks = 6,
    col    = "lightblue",  # Color of the bars
    main   = "10,000 Fair Die Rolls", # Title of the plot
    xlab   = "Face Value") # Label for the x-axis
```

 Explanation:

- y-axis label is automatically set to "Frequency" by default.
- Bars should be roughly equal height — that's a uniform distribution.
- When you specify breaks = 6, R doesn't necessarily create bins that align perfectly with our integer values 1-6.

```
hist(dice,
    breaks = seq(0.5, 6.5, by = 1),  # Explicit breaks between integers
    col    = "lightblue",
    main   = "10,000 Fair Die Rolls",
    xlab   = "Face Value")
```

5. Adding Two Dice

- Look deeper: What is the *Central Limit Theorem*
- The GOAL: Roll two dice 10,000 times, add the results, plot.

```
sum_two <- sample(1:6, 10000, TRUE) + sample(1:6, 10000, TRUE)

hist(sum_two,
     breaks = 11,
     col    = "salmon",
     main   = "Sum of Two Dice (10,000 trials)",
     xlab   = "Total")
```

- How to Check for Issues
- Reflect: What shape do you expect: triangular, flat, or skewed?

```
h <- hist(sum_two, breaks = 11, plot = FALSE)
h$breaks  # Shows the actual break points used
```

- NOTICE: We stored the histogram in h without plotting it.

Use explicit breaks that align with the possible values

```
hist(sum_two,
     breaks = seq(1.5, 12.5, by = 1),  # Creates bins centered on integers 2-12
     col    = "salmon",
     main   = "Sum of Two Dice (10,000 trials)",
     xlab   = "Total")
```

6. Normal Data

 The GOAL: Simulate 1,000 adult female heights (mean 64 in, SD 2.5 in).

rnorm() generates a vector of random numbers that form a normal distribution.

```
?rnorm  # Check the help file for rnorm()
```

 The SYNTAX: rnorm(n, mean = 0, sd = 1)

```
heights <- rnorm(n = 1000, mean = 64, sd = 2.5)
```

```
hist(heights,
     breaks = "Scott",
     col    = "lavender",
     main   = "Simulated Female Heights",
     xlab   = "Height (inches)")
```

 Look deeper: What are “Scott” breaks in Histograms?

7. Repeating a Simulation

 The GOAL: How variable is the number of tails in 5,000 flips? Repeat 100 times.

```
?replicate  # Check the help file for replicate()
```

 The SYNTAX: replicate(n, expr, simplify = "array")

expr is the expression (a language object, usually a call) to evaluate repeatedly.

```
tails_100 <- replicate(n = 100, expr = sum(sample(0:1, 5000, TRUE)))
summary(tails_100)
```

 Check-in: What is expr in this case? Why is simplify = "array" not included?

 Explanation:

The replicate() function will:

1. Evaluate this entire expression 100 times (because n = 100)
2. Each evaluation is independent (gets its own random sampling)
3. Store the 100 results in a vector called tails_100

```
hist(tails_100,
     breaks = "Scott",
     col    = "gold",
     main   = "Distribution of Tails in 5,000 Coin Flips (100 reps)",
     xlab   = "Count of Tails")
```

 NOTICE: The spread shows sampling variability around the theoretical 2,500.

 The GOAL: Sum rnorm() for expr in replicate(). Count how many values in each sample are below a threshold.

```
values_below_zero <- replicate(n = 200, expr = sum(rnorm(50, mean = 2, sd = 3) < 0))
```

Order of Operations:

1. rnorm(n = 50, mean = 2, sd = 3) generates a sample of 50 random numbers.
2. sum(... < 0) counts how many of those numbers are below zero.
3. replicate(n = 200, expr = ...) repeats this process 200 times.

4. The result is a vector of counts, one for each of the 200 samples.

Simplified example

 The GOAL: Generate 10 random numbers, Compare each to zero, Count how many are below zero

```r
# Generate 10 random numbers
x <- rnorm(n = 10, mean = 2, sd = 3)
print(x)

# Compare each to zero
below_zero <- x < 0
print(below_zero)

# Count how many are below zero
count <- sum(below_zero)
print(count)
```

8. Fair Dice Check

Probability weights in sample() let you simulate biased outcomes.

 The SYNTAX: sample(x, size, replace = FALSE, prob = NULL)

The prob parameter specifies the probability weights for each element in the sampling pool.

```
first_time_prob <- sample(1:6, 1000, prob = c(.12,.13,.07,.23,.22,.23))
```

This simulates rolling a biased or "loaded" die 1000 times. The length of the prob vector must match the number of sampled values (x= 1:6). The probabilities should sum to 1'

 Check in: What does TRUE mean in the sample() calls?

 Your "friend's" dice look suspiciously unfair. Compare it to our fair dice.

```
set.seed(8000)
friend_sum <- sample(1:6, 1000, prob = c(.12,.13,.07,.23,.22,.23), TRUE) +
  sample(1:6, 1000, prob = c(.12,.13,.07,.23,.22,.23), TRUE)
```

```
fair_sum <- sample(1:6, 1000, TRUE) + sample(1:6, 1000, TRUE)
```

 Explanation: Understanding the Sum of Two Samples

The simulates rolling two biased dice 1000 times. This is like you and a friend each rolling a biased die 1000 times simultaneously and recording the sum for each pair of rolls:

```
hist(friend_sum,
     breaks = "Scott",
     col    = rgb(1,0,0,.4),
     freq   = FALSE,
     main   = "Friend vs. Fair Dice (1000 rolls)",
     xlab   = "Sum of Two Dice")
```

```
hist(fair_sum,
     breaks = "Scott",
     col    = rgb(0,0,1,.4),
     freq   = FALSE,
     add    = TRUE)
```

```
legend("topright", legend = c("Friend","Fair"),
       fill = c(rgb(1,0,0,.4), rgb(0,0,1,.4)))
```

□ Check□ in: Do the red bars lean high or low compared with blue?

9. ⛶ Practice Space

□ Practice: Replace the blanks (___) and run.

□ The GOAL: Simulate 300 rolls of a 20□ sided die (values 1:20) and store in **big_die**

```
big_die <- sample( ___ , size = ___ , replace = ___ )

# Using `any()`, did you roll a 20?
any(big_die == ___ )
```

□ Practice:

1. Simulate weekly demand for a product using rpois() (hint: ?rpois).
2. Try different probability weights in sample() to make a loaded coin.
3. Write a pipeline that creates 1,000 random heights, bins them with cut(), and counts the number in each bin.

Add your experiments below and include at least one □ check□ in of your own.

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

When you're done, your workspace should contain FOUR new objects: boring_weekend, d20_counts, blanketed_babies, weeks_too_long

10.1 Task 1

D20 marathon

Simulate 5221 rolls of a fair 20-sided die. Store as **boring_weekend** using sample().

```
____ <- _____
```

10.2 Task 2

☐ Quick frequency check

Create **d20_counts**: a table of how many times each face appeared.

```
____ <- _____
```

```
# Quick check
print(d20_counts)
```

10.3 Task 3

Hit rate

What proportion of rolls were natural 20s? (one line of code, no object)

```
____
```

10.4 Task 4

Baby-blanket test

Simulate 100 newborn lengths (inches) with mean 20.0, SD 0.9. Store as **blanketed_babies**. Use rnorm().

```
____ <- _____
```

10.5 Task 5

Too long?

Using any(), were ANY babies > 23 in? Also count how many. No object.

```
____(____)

____(____)
```

☐ Comment TODO: Write Yes / No – does the hospital need longer blankets?

☐ Comment: "___"

10.6 Task 6

☐ Week-to-week risk

Repeat Task 4 & 5 for 1000 simulated weeks. Store a vector **weeks_too_long** that records, for each week, the number of babies over 23 inches. Hint: replicate().

```
____ <- _____
```

10.7 Task 7

☐ Reflect

☐ Write a short paragraph reflecting on what the utility of replicate() was in task 6?

☐ 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("Assignment12_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:

- Used sample() for discrete simulations with or without replacement.
- Visualized uniform and aggregated dice results with histograms.
- Generated normal random numbers with rnorm() and plotted the shape.
- Repeated an experiment easily with replicate().
- Compared two simulated distributions to spot bias.