

# Week 1: Introduction to R and RStudio

Welcome to R Programming!

Dr. Basim Alsaedi

2025-08-31

# Welcome to R Programming!

## Your Journey into Data Analysis Begins

Welcome to R - the language that powers data analysis at Google, Facebook, and thousands of companies worldwide! Today, you'll write your first code and begin your transformation into a data analyst.

### **What you'll learn this week:**

- What R is and why it's powerful
- How to use R as a calculator
- Creating and using variables
- Your first real analysis

**Our Project:** Help "Bean There Café" analyze their daily operations

# What is R?

R is a programming language designed for data analysis and statistics.

## Think of R as:

- A super-powered calculator that remembers everything
- A tool that transforms numbers into insights
- Your assistant for making data-driven decisions

## Real-world fact: R is used by:

- Netflix to recommend shows
- Facebook to analyze user behavior
- Scientists to discover new medicines
- Banks to detect fraud

In this class, you join millions of R users worldwide!

# Your R Environment

**The Console:** Where you type commands and see results

```
# This is where you type  
> 2 + 2  
[1] 4
```

## Key points:

- > means R is ready for your command
- Type your command and press Enter
- [1] appears before results (ignore it for now)
- # starts a comment (notes for humans, R ignores these)

# R as Your Calculator

Let's start with something familiar - math!

*# Addition*

5 + 3

*# Subtraction*

10 - 4

*# Multiplication (use \*)*

6 \* 7

*# Division (use /)*

20 / 4

*# Exponents (use ^)*

2 ^ 3

**Try it!** Calculate your age in days: age \* 365

# Order of Operations

R follows standard math rules (PEMDAS):

```
# Without parentheses  
2 + 3 * 4  
# Result: 14 (multiplication first)  
  
# With parentheses  
(2 + 3) * 4  
# Result: 20 (parentheses first)  
  
# Complex calculation  
(100 - 20) / 4 + 3 * 2  
# Result: 26
```

**Rule:** When in doubt, use parentheses to be clear!

# Real Coffee Shop Calculations

Let's help Bean There Café:

```
# Morning sales calculation
# Sold 45 coffees at $4.50 each
45 * 4.50
# Result: 202.5

# Daily profit calculation
# Revenue: $2847, Costs: $1823
2847 - 1823
# Result: 1024

# Average per customer
# Total: $2847, Customers: 127
2847 / 127
# Result: 22.42
```

You're already doing business analysis!

# The Problem with Repeating Numbers

What if coffee price changes?

```
# Monday calculations  
45 * 4.50  # Morning  
67 * 4.50  # Afternoon  
38 * 4.50  # Evening  
  
# If price changes to $4.75, we must update everywhere!  
45 * 4.75  # Update here  
67 * 4.75  # And here  
38 * 4.75  # And here too!
```

**Problem:** Updating the same number multiple times is error-prone

**Solution:** Variables!

# Introduction to Variables

Variables are named storage boxes for values.

```
# Create a variable with <-  
coffee_price <- 4.50  
  
# The arrow points the value INTO the variable name  
# Read as: "coffee_price gets 4.50"
```

**The arrow <- is crucial:**

- Points from right to left
- Stores the value in the variable
- Like putting something in a labeled box

# Using Variables

Once created, use variables anywhere:

```
# Store the price once  
coffee_price <- 4.50  
  
# Use it many times  
morning_sales <- 45 * coffee_price  
afternoon_sales <- 67 * coffee_price  
evening_sales <- 38 * coffee_price  
  
# Check a value  
morning_sales  
# Result: 202.5
```

## Benefits:

- Change price in ONE place
- Code is clearer
- Fewer errors

# Variable Naming Rules

## Good variable names:

```
daily_sales <- 2847  
customer_count <- 127  
avg_purchase <- 22.42  
coffee_price <- 4.50
```

## Rules:

- Use descriptive names
- No spaces (use underscore \_ )
- Can't start with numbers
- Case sensitive (Sales  $\neq$  sales)

## Bad examples:

```
# Don't do these:  
x <- 2847           # Not descriptive  
coffee price <- 4.50 # No spaces allowed!  
2nd_place <- "silver" # Can't start with number
```

# Displaying Values with print()

The print() function shows values on screen:

```
# Calculate and store  
total_sales <- 2847  
  
# Display the value  
print(total_sales)  
# Output: [1] 2847  
  
# Can print calculations directly  
print(45 * 4.50)  
# Output: [1] 202.5  
  
# Print with variables  
coffee_price <- 4.50  
print(coffee_price * 100)  
# Output: [1] 450
```

**Note:** print() is our first function - it does something with what's in the parentheses

# Updating Variables

Variables can be changed:

```
# Initial price
coffee_price <- 4.50
print(coffee_price)
# Output: [1] 4.50

# Price increase
coffee_price <- 4.75
print(coffee_price)
# Output: [1] 4.75

# Using the new value
revenue <- 100 * coffee_price
print(revenue)
# Output: [1] 475
```

The variable remembers the most recent value!

# Building Your First Analysis

Let's analyze Bean There Café's daily performance:

```
# Store the data
morning_customers <- 45
afternoon_customers <- 67
evening_customers <- 38

# Calculate total
total_customers <- morning_customers + afternoon_customers + evening_customers
print(total_customers)
# Output: [1] 150

# Average transaction value
total_revenue <- 2847
avg_per_customer <- total_revenue / total_customers
print(avg_per_customer)
# Output: [1] 18.98
```

# Working with Decimals

R handles decimal numbers naturally:

```
# Precise calculations
price <- 4.99
tax_rate <- 0.08
quantity <- 3

# Calculate with decimals
subtotal <- price * quantity
tax <- subtotal * tax_rate
total <- subtotal + tax

print(total)
# Output: [1] 16.1676

# Round to 2 decimal places (we'll learn more later)
# For now, just know R is precise!
```

# Common Beginner Mistakes

## Mistake 1: Wrong arrow direction

```
# Wrong:  
4.50 -> coffee_price # Backwards!  
  
# Right:  
coffee_price <- 4.50 # Into the variable
```

## Mistake 2: Forgetting to run lines

```
coffee_price <- 4.50  
# Must press Enter to execute!
```

## Mistake 3: Typos in variable names

```
coffee_price <- 4.50  
print(coffe_price) # Missing 'e' - Error!
```

# Complete Daily Report

Let's create a full analysis:

```
# Daily sales data
coffees_sold <- 145
pastries_sold <- 89
sandwiches_sold <- 56

# Prices
coffee_price <- 4.50
pastry_price <- 3.75
sandwich_price <- 8.95

# Calculate revenues
coffee_revenue <- coffees_sold * coffee_price
pastry_revenue <- pastries_sold * pastry_price
sandwich_revenue <- sandwiches_sold * sandwich_price

# Total daily revenue
total_revenue <- coffee_revenue + pastry_revenue + sandwich_revenue

# Display results
print(total_revenue)
# Output: [1] 1488
```

# Understanding Output

When R shows results:

```
# Single value  
age <- 25  
print(age)  
# Output: [1] 25  
  
# The [1] means "starting at position 1"  
# For single values, always [1]  
# We'll see [1], [2], etc. with multiple values later
```

**Don't worry about [1] for now** - just focus on your actual result!

# Practice Calculations

Try these on your own:

## ① Calculate weekly revenue:

- Daily average: \$2847
- Days: 7

## ② Find profit margin:

- Revenue: \$2847
- Costs: \$1823
- $\text{Margin} = (\text{Revenue} - \text{Costs}) / \text{Revenue}$

## ③ Employee wage total:

- 3 employees
- 8 hours each
- \$15.50 per hour

Remember: Create variables, calculate, print results!

# What You've Accomplished Today!

- ✓ **Used R as a calculator** - All operations
- ✓ **Created variables** - Stored values with `<-`
- ✓ **Named variables properly** - Descriptive names
- ✓ **Used `print()`** - Your first function!
- ✓ **Built real analyses** - Coffee shop calculations
- ✓ **Updated variables** - Changed stored values

**You are officially an R programmer!**

Every data scientist started exactly where you are now!

