Accelerator Crash Course - R

Center of Excellence for Women & Technology, Indiana University Bloomington

Jeffery (Shih-Chieh) Wang

2026-04-11

Course Overview

- **Learning materials:** jeffery-wang.com/r-basics/
- **Duration:** 4 hours (10:30–12:00; 13:00-15:30)
- **Goal**: By the end of this course, you will know the basics of R, able to import data into R, maniupate and tidy data sets using tidyverse tool, create visualizations, and get a better idea how to use generative AI to accelerate the R learning.
- **Prerequisites:** None! This course is designed for true beginners.
- **What you need:**
  - A laptop with R and RStudio installed
  - An internet connection (for installing packages and AI tools)

Schedule

10:30–11:30
Module 1: Getting Started with R & RStudio

11:30–12:00
Module 2: Importing Data (readr & readxl)

12:00–1:00
Lunch Break

13:00–14:00
Module 3: Data Manipulation with dplyr

14:00-15:30
Module 4: Data Tidying with tidyr

Module 1: Getting Started

Learning Objectives

- Navigate the RStudio interface (console, script editor, environment, files)
- Understand basic R concepts: objects, assignment, data types
- Install and load packages
- Understand what the tidyverse is

The RStudio Interface

RStudio is the most popular IDE for R language. It has four main panes:

- `IDE: Integrated Development Environment (code editing, building, testing, debugging, and many others)`

1. **Source Editor** (top-left) — where you write and save scripts
2. **Console** (bottom-left) — where R runs commands
3. **Environment** (top-right) — shows your data and objects
4. **Files/Plots/Help** (bottom-right) — file browser, plot viewer, documentation

- Note: You may customize the layout by going to `Tools` → `Global Options`→ `Pane Layout`

R Basics

Basic Calculations in R

- One of the easiest ways to start using R is to treat it like a calculator.
- R can perform basic arithmetic operations such as addition, subtraction, multiplication, division, and exponents.

```
# Some examples:

2 + 3        # addition
```

```
[1] 5
```

```
10 - 4       # subtraction
```

```
[1] 6
```

```
6 * 5        # multiplication
```

```
[1] 30
```

```
12 / 3        # division
```

```
[1] 4
```

```
2^3           # exponent
```

```
[1] 8
```

```
(2 + 3) * 4   # parentheses change the order of operations
```

```
[1] 20
```

Common R Objects

Objects are the basic building blocks for storing data in R. Five common types of R objects are vectors, factors, matrices, lists, and data frames.

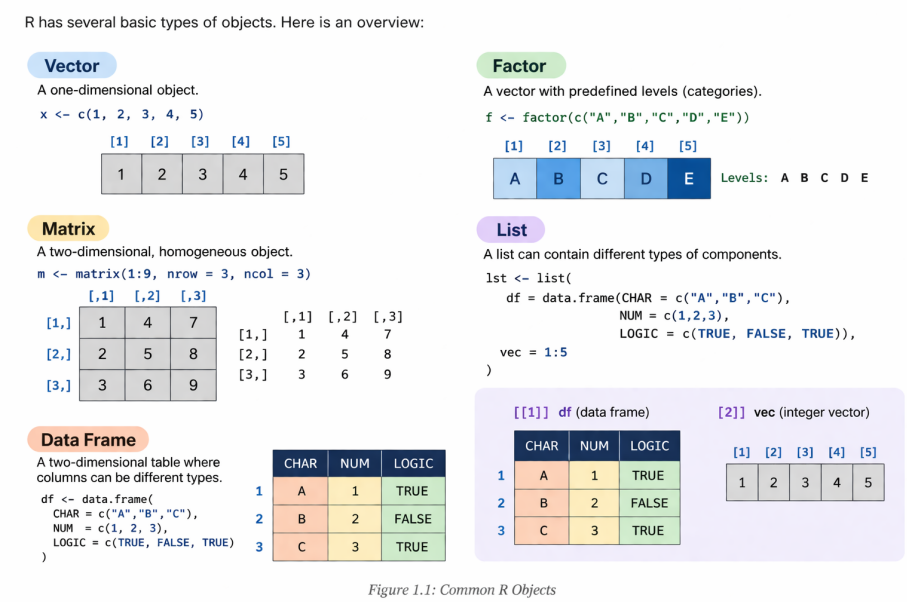


Figure 1.1: Common R Objects

```
# Assignment: use <- to store values in objects
my_name <- "World"
my_number <- 42

# Print values
my_name
```

```
[1] "World"
```

```
my_number
```

```
[1] 42
```

```r
# Vectors: a collection of values of the same type
ages <- c(25, 30, 35, 40, 45)
names <- c("Alice", "Bob", "Carol", "Dave", "Eve")

# Basic operations on vectors
mean(ages)
```

```
[1] 35
```

```r
length(ages)
```

```
[1] 5
```

```r
# Basic data types
class(my_name)    # character
```

```
[1] "character"
```

```r
class(my_number)  # numeric
```

```
[1] "numeric"
```

```r
class(1)          #
```

```
[1] "numeric"
```

```r
class(TRUE)       # logical (can be )
```

```
[1] "logical"
```

```r
# Example about checking logical values
is_adult <- 20 >= 18
class(is_adult)   # logical
```

```
[1] "logical"
```

```r
is_adult          # TRUE, because it stores the result of "20 >= 18"
```

```
[1] TRUE
```

```r
is_tall <- 150 > 180
is_tall
```

```
[1] FALSE
```

- Note: These create logical results
- `== # equal to`
- `!= # not equal to > # greater than`
- `< # less than`
- `>= # greater than or equal to`
- `<= # less than or equal to`

Installing and Loading the Tidyverse

```
# Install the tidyverse (only need to do this once)
install.packages("tidyverse")
```

```
# Load the tidyverse (do this every session)
library(tidyverse)
```

The tidyverse is a collection of packages that share a common design philosophy:

- **readr** — reading data files (CSV, TSV, etc.)
- **dplyr** — data manipulation (filter, select, mutate, summarize)
- **tidyr** — reshaping and tidying data
- **ggplot2** — data visualization
- **stringr** — string manipulation
- **forcats** — working with categorical data (factors)
- **purrr** — functional programming
- **tibble** — modern data frames

Data Frame?

```
# A data frame is a table - the fundamental data structure for analysis
# Let's look at a built-in dataset in ggplot2 (part of the idyverse)
mpg
```

```
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
 4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
# i 224 more rows
```

```
# Useful functions for exploring data frames
dim(mpg)       # rows x columns
```

```
[1] 234  11
```

```
names(mpg)     # column names
```

```
 [1] "manufacturer" "model"        "displ"        "year"         "cyl"
```

```
 [6] "trans"        "drv"          "cty"          "hwy"          "fl"
[11] "class"
```

```
glimpse(mpg)   # compact overview
```

```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
$ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
$ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
$ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

```
head(mpg)      # first 6 rows
```

```
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

- Another approach more common because you most like import your own data

```
# use vector
data_mpg <- mpg
glimpse(data_mpg )   # compact overview
```

```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
$ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
```

```
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
$ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
$ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

```
summary(data_mpg)   # summarize your data variables (basic stats, types, NAs)
```

```
 manufacturer          model               displ           year
 Length:234         Length:234         Min.   :1.600   Min.   :1999
 Class :character   Class :character   1st Qu.:2.400   1st Qu.:1999
 Mode  :character   Mode  :character   Median :3.300   Median :2004
                                       Mean   :3.472   Mean   :2004
                                       3rd Qu.:4.600   3rd Qu.:2008
                                       Max.   :7.000   Max.   :2008
      cyl           trans               drv                cty
 Min.   :4.000   Length:234         Length:234         Min.   : 9.00
 1st Qu.:4.000   Class :character   Class :character   1st Qu.:14.00
 Median :6.000   Mode  :character   Mode  :character   Median :17.00
 Mean   :5.889                                         Mean   :16.86
 3rd Qu.:8.000                                         3rd Qu.:19.00
 Max.   :8.000                                         Max.   :35.00
      hwy             fl                class
 Min.   :12.00   Length:234         Length:234
 1st Qu.:18.00   Class :character   Class :character
 Median :24.00   Mode  :character   Mode  :character
 Mean   :23.44
 3rd Qu.:27.00
 Max.   :44.00
```

```
dim(data_mpg )       # rows x columns
```

```
[1] 234  11
```

```
names(data_mpg )     # column names
```

```
 [1] "manufacturer" "model"        "displ"        "year"         "cyl"
 [6] "trans"        "drv"          "cty"          "hwy"          "fl"
[11] "class"
```

```
head(data_mpg)      # first 6 rows
```

```
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
```

```
6 audi          a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

```
# Let's see the car company list
length(data_mpg$manufacturer)
```

```
[1] 234
```

```
unique(data_mpg$manufacturer)
```

```
 [1] "audi"       "chevrolet"  "dodge"      "ford"       "honda"
 [6] "hyundai"    "jeep"       "land rover" "lincoln"    "mercury"
[11] "nissan"     "pontiac"    "subaru"     "toyota"     "volkswagen"
```

```
length(unique((data_mpg$manufacturer)))
```

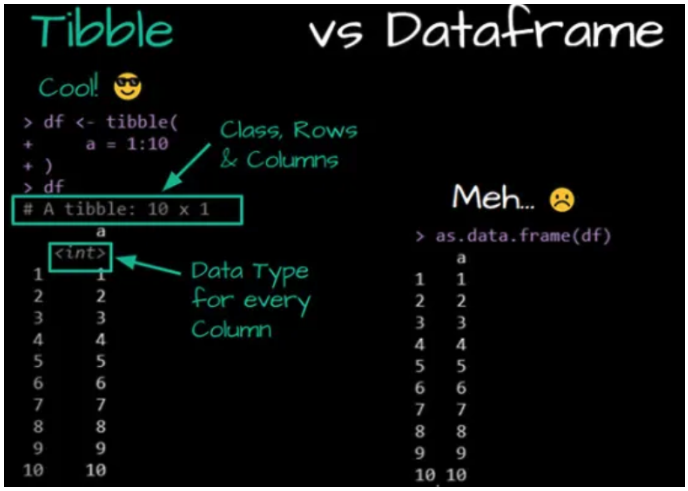
```
[1] 15
```

```
# And all the car models
unique(data_mpg$model)
```

```
 [1] "a4"                    "a4 quattro"            "a6 quattro"
 [4] "c1500 suburban 2wd"    "corvette"              "k1500 tahoe 4wd"
 [7] "malibu"                "caravan 2wd"           "dakota pickup 4wd"
[10] "durango 4wd"           "ram 1500 pickup 4wd"   "expedition 2wd"
[13] "explorer 4wd"          "f150 pickup 4wd"       "mustang"
[16] "civic"                 "sonata"                "tiburon"
[19] "grand cherokee 4wd"    "range rover"           "navigator 2wd"
[22] "mountaineer 4wd"       "altima"                "maxima"
[25] "pathfinder 4wd"        "grand prix"            "forester awd"
[28] "impreza awd"           "4runner 4wd"           "camry"
[31] "camry solara"          "corolla"               "land cruiser wagon 4wd"
[34] "toyota tacoma 4wd"     "gti"                   "jetta"
[37] "new beetle"            "passat"
```

```
sort(unique(data_mpg$model))
```

```
 [1] "4runner 4wd"           "a4"                    "a4 quattro"
 [4] "a6 quattro"            "altima"                "c1500 suburban 2wd"
 [7] "camry"                 "camry solara"          "caravan 2wd"
[10] "civic"                 "corolla"               "corvette"
[13] "dakota pickup 4wd"     "durango 4wd"           "expedition 2wd"
[16] "explorer 4wd"          "f150 pickup 4wd"       "forester awd"
[19] "grand cherokee 4wd"    "grand prix"            "gti"
[22] "impreza awd"           "jetta"                 "k1500 tahoe 4wd"
[25] "land cruiser wagon 4wd" "malibu"               "maxima"
[28] "mountaineer 4wd"       "mustang"               "navigator 2wd"
[31] "new beetle"            "passat"                "pathfinder 4wd"
[34] "ram 1500 pickup 4wd"   "range rover"           "sonata"
[37] "tiburon"               "toyota tacoma 4wd"
```

- `dbl: double-precision floating-point numbers, storing real numbers for those with decimal points`
- Tibble (modern) vs Dataframe
- 

  Source: www.r-bloggers.com

Exercise 1.1

```
# 1. Create a vector called 'fruits' containing: "apple", "banana", "cherry"
# 2. Create a vector called 'prices' with values: 1.20, 0.50, 2.75
# 3. Find the mean price
```

Exercise 1.2

```
# 1. Load the tidyverse and explore the 'diamonds' dataset using glimpse()

# 2. How many rows and columns?

# 3. What are the variable names?

# 4  What does 'cut' contain?
```

Module 2: Importing Data

Learning Objectives

- Read CSV files into R with `read_csv()`

- Understand file paths and working directories
- Inspect imported data for issues
- Know about other import functions (Excel, TSV, etc.)

Key Concepts

Your Working Directory

```
# Where is R looking for files?
getwd()

# You can set it in RStudio: Session > Set Working Directory > Choose Directory

# Or use code:
setwd("/path/to/your/folder") # Replace the path with the actual location of your file

list.files()
```

Reading CSV Files

```
# Reading a CSV file
# Replace the path with the actual location of your file
survey_data <- read_csv("sample_survey_data.csv")

# Look at the result
survey_data
```

```
# A tibble: 40 x 10
   respondent_id   age gender department  years_experience satisfaction_score
           <dbl> <dbl> <chr>  <chr>                  <dbl>              <dbl>
 1             1    28 Female Marketing                  3                4.2
 2             2    35 Male   Engineering                8                3.8
 3             3    42 Female Engineering               15                4.5
 4             4    25 Male   Sales                      1                3.2
 5             5    31 Female Marketing                  5                4
 6             6    38 Male   HR                        10                3.5
 7             7    29 Female Engineering                4                4.3
 8             8    45 Male   Sales                     18                3.9
 9             9    33 Female HR                        7                4.1
10            10    27 Male   Marketing                  2                3.6
# i 30 more rows
# i 4 more variables: monthly_hours <dbl>, remote_days_per_week <dbl>,
#   training_completed <lgl>, annual_salary <dbl>
```

```
glimpse(survey_data)
```

```
Rows: 40
Columns: 10
$ respondent_id        <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15~
$ age                  <dbl> 28, 35, 42, 25, 31, 38, 29, 45, 33, 27, 40, 36, 2~
$ gender               <chr> "Female", "Male", "Female", "Male", "Female", "Ma~
$ department           <chr> "Marketing", "Engineering", "Engineering", "Sales~
$ years_experience     <dbl> 3, 8, 15, 1, 5, 10, 4, 18, 7, 2, 12, 9, 1, 22, 4,~
$ satisfaction_score   <dbl> 4.2, 3.8, 4.5, 3.2, 4.0, 3.5, 4.3, 3.9, 4.1, 3.6,~
$ monthly_hours        <dbl> 165, 180, 170, 175, 160, 155, 185, 170, 150, 168,~
$ remote_days_per_week <dbl> 2, 3, 4, 1, 2, 3, 4, 1, 2, 2, 5, 1, 2, 3, 3, 0, 4~
$ training_completed   <lgl> TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE,~
$ annual_salary        <dbl> 52000, 85000, 105000, 45000, 58000, 72000, 78000,~
```

```
summary(survey_data)
```

```
 respondent_id        age           gender           department
 Min.   : 1.00   Min.   :23.00   Length:40          Length:40
 1st Qu.:10.75   1st Qu.:29.00   Class :character   Class :character
 Median :20.50   Median :33.50   Mode  :character   Mode  :character
 Mean   :20.50   Mean   :34.35
 3rd Qu.:30.25   3rd Qu.:39.25
 Max.   :40.00   Max.   :50.00

 years_experience satisfaction_score monthly_hours remote_days_per_week
 Min.   : 1.000   Min.   :3.000      Min.   :144   Min.   :0.00
 1st Qu.: 3.000   1st Qu.:3.575      1st Qu.:160   1st Qu.:1.00
 Median : 6.000   Median :4.000      Median :170   Median :2.00
 Mean   : 7.541   Mean   :3.930      Mean   :169   Mean   :2.35
 3rd Qu.:10.000   3rd Qu.:4.300      3rd Qu.:179   3rd Qu.:3.00
 Max.   :22.000   Max.   :4.800      Max.   :192   Max.   :5.00
 NA's   :3                           NA's   :1
 training_completed annual_salary
 Mode :logical      Min.   : 44000
 FALSE:10           1st Qu.: 56000
 TRUE :30           Median : 66000
                    Mean   : 69189
                    3rd Qu.: 75000
                    Max.   :115000
                    NA's   :3
```

Understanding the Output

When `read_csv()` imports data, it tells you:

- How many rows and columns it found
- What type it guessed for each column (character, double, integer, etc.)

Common Import Issues

```
# If your CSV uses semicolons instead of commas (common in Europe)
data <- read_csv2("european_file.csv")

# If your file uses tabs (Tab-Separated Values)
data <- read_delim("file.tsv", delim = "\t")

# Reading Excel files (need the readxl package)
library(readxl)
data <- read_excel("file.xlsx")
data <- read_excel("file.xlsx", sheet = "Sheet2")  # specific sheet
```

- Example of European Semicolon CSV (`read_csv2`)

  This format is the standard in many European countries where the **comma** is already used for decimals (e.g.,1,50 instead of 1.50). To separate the columns, they use a **semicolon**.

  ```
  Country;GDP_Growth;Inflation_Rate
  Germany;1,5;2,1
  France;1,2;1,8
  Italy;0,9;2,3
  ```

- Example of Tab-Separated Values

  In a "raw" view, tabs often look like big chunks of whitespace. In reality, they are a single invisible character (`\t`). TSVs are great because you almost never have a "tab" inside your actual data, so it's less likely to break your code.

  ```
  ID  Observation Status
  101 High Pressure   Stable
  102 Low Temperature Critical
  103 Ambient Stable
  ```

Working with Built-in Datasets

```
# R comes with many practice datasets
# The tidyverse adds even more
data(mpg)        # fuel economy data
data(diamonds)   # diamond prices
data(starwars)   # Star Wars characters

# View the starwars dataset
starwars
```

```
# A tibble: 87 x 14
```

```
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
 2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
 3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
 4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
 5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
 6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
 7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
 8 R5-D4        97    32 <NA>       white, red red             NA   none  mascu~
 9 Biggs D~    183    84 black      light      brown           24   male  mascu~
10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male  mascu~
# i 77 more rows
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

Exercise 2.1

```
# 1. Download the sample_survey_data.csv file to your computer
```

- Down here: Link

```
# 2. Read it into R using read_csv() and save it as 'survey'
# 3. Use summary() to examine the structure
# 4. How many rows and columns does it have?
# 5. What are the column names?
# 6. What variables have missing values?
```

Module 3: Data Manipulation with dplyr

Learning Objectives

- Chain operations with the pipe `|>` (or `%>%`)
- Sort data with `arrange()`; Note: `sort()` is a base R function
- Select columns with `select()`
- Filter rows with `filter()`
- Create new columns with `mutate()`
- Summarize data with `summarize()` and `group_by()`

Key Concepts

The Pipe: `|>` (or `%>%`)

The pipe takes the output of one function and feeds it as the first argument to the next function. Think of it as "and then."

```r
# Without the pipe (nested, hard to read)
head(arrange(filter(mpg, manufacturer == "toyota"), desc(hwy)), 5)
```

```
# A tibble: 5 x 11
  manufacturer model   displ  year   cyl trans     drv     cty   hwy fl    class
  <chr>        <chr>   <dbl> <int> <int> <chr>     <chr> <int> <int> <chr> <chr>
1 toyota       corolla   1.8  2008     4 manual(m~ f        28    37 r     comp~
2 toyota       corolla   1.8  1999     4 manual(m~ f        26    35 r     comp~
3 toyota       corolla   1.8  2008     4 auto(l4)  f        26    35 r     comp~
4 toyota       corolla   1.8  1999     4 auto(l4)  f        24    33 r     comp~
5 toyota       camry     2.4  2008     4 manual(m~ f        21    31 r     mids~
```

```r
# With the pipe (read left to right, top to bottom)
mpg |>
  filter(manufacturer == "toyota") |>
  arrange(desc(hwy)) |>
  head(5)
```

```
# A tibble: 5 x 11
  manufacturer model   displ  year   cyl trans     drv     cty   hwy fl    class
  <chr>        <chr>   <dbl> <int> <int> <chr>     <chr> <int> <int> <chr> <chr>
1 toyota       corolla   1.8  2008     4 manual(m~ f        28    37 r     comp~
2 toyota       corolla   1.8  1999     4 manual(m~ f        26    35 r     comp~
3 toyota       corolla   1.8  2008     4 auto(l4)  f        26    35 r     comp~
4 toyota       corolla   1.8  1999     4 auto(l4)  f        24    33 r     comp~
5 toyota       camry     2.4  2008     4 manual(m~ f        21    31 r     mids~
```

```r
# arrange() for sorting
arrange(mpg, hwy)        # hwy (highway) ascending, small to large
```

```
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 dodge        dakota pi~   4.7  2008     8 auto~ 4         9    12 e     pick~
 2 dodge        durango 4~   4.7  2008     8 auto~ 4         9    12 e     suv
 3 dodge        ram 1500 ~   4.7  2008     8 auto~ 4         9    12 e     pick~
 4 dodge        ram 1500 ~   4.7  2008     8 manu~ 4         9    12 e     pick~
 5 jeep         grand che~   4.7  2008     8 auto~ 4         9    12 e     suv
 6 chevrolet    k1500 tah~   5.3  2008     8 auto~ 4        11    14 e     suv
 7 jeep         grand che~   6.1  2008     8 auto~ 4        11    14 p     suv
 8 chevrolet    c1500 sub~   5.3  2008     8 auto~ r        11    15 e     suv
 9 chevrolet    k1500 tah~   5.7  1999     8 auto~ 4        11    15 r     suv
10 dodge        dakota pi~   5.2  1999     8 auto~ 4        11    15 r     pick~
# i 224 more rows
```

```r
arrange(mpg, desc(hwy))  # descending, large to small
```

```
# A tibble: 234 x 11
```

```
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 volkswagen   jetta        1.9  1999     4 manu~ f        33    44 d     comp~
 2 volkswagen   new beetle   1.9  1999     4 manu~ f        35    44 d     subc~
 3 volkswagen   new beetle   1.9  1999     4 auto~ f        29    41 d     subc~
 4 toyota       corolla      1.8  2008     4 manu~ f        28    37 r     comp~
 5 honda        civic        1.8  2008     4 auto~ f        25    36 r     subc~
 6 honda        civic        1.8  2008     4 auto~ f        24    36 c     subc~
 7 toyota       corolla      1.8  1999     4 manu~ f        26    35 r     comp~
 8 toyota       corolla      1.8  2008     4 auto~ f        26    35 r     comp~
 9 honda        civic        1.8  2008     4 manu~ f        26    34 r     subc~
10 honda        civic        1.6  1999     4 manu~ f        28    33 r     subc~
# i 224 more rows
```

select() — Choose Columns

```
names(mpg)
```

```
 [1] "manufacturer" "model"        "displ"        "year"         "cyl"
 [6] "trans"        "drv"          "cty"          "hwy"          "fl"
[11] "class"
```

```
glimpse(mpg)
```

```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
$ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
$ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
$ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

```
mpg
```

```
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
 4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
```

```
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
# i 224 more rows
```

```
# Select specific columns by name
mpg |>
  select(manufacturer, model, year, hwy) # Note: A `dplyr` pipeline only prints the result u
```

```
# A tibble: 234 x 4
   manufacturer model       year   hwy
   <chr>        <chr>      <int> <int>
 1 audi         a4          1999    29
 2 audi         a4          1999    29
 3 audi         a4          2008    31
 4 audi         a4          2008    30
 5 audi         a4          1999    26
 6 audi         a4          1999    26
 7 audi         a4          2008    27
 8 audi         a4 quattro  1999    26
 9 audi         a4 quattro  1999    25
10 audi         a4 quattro  2008    28
# i 224 more rows
```

```
  # For example:
  mpg_small <- mpg %>% select(manufacturer, model, year, hwy) # Don't forget that you can al

  # or
  mpg.s <- select (mpg, manufacturer, model, year, hwy )

# Remove columns with -
mpg |>
  select(-fl, -class)
```

```
# A tibble: 234 x 9
   manufacturer model      displ  year   cyl trans      drv     cty   hwy
   <chr>        <chr>      <dbl> <int> <int> <chr>      <chr> <int> <int>
 1 audi         a4           1.8  1999     4 auto(l5)   f        18    29
 2 audi         a4           1.8  1999     4 manual(m5) f        21    29
 3 audi         a4           2    2008     4 manual(m6) f        20    31
 4 audi         a4           2    2008     4 auto(av)   f        21    30
 5 audi         a4           2.8  1999     6 auto(l5)   f        16    26
 6 audi         a4           2.8  1999     6 manual(m5) f        18    26
 7 audi         a4           3.1  2008     6 auto(av)   f        18    27
```

```
 8 audi         a4 quattro   1.8  1999     4 manual(m5) 4        18    26
 9 audi         a4 quattro   1.8  1999     4 auto(l5)   4        16    25
10 audi         a4 quattro   2    2008     4 manual(m6) 4        20    28
# i 224 more rows
```

```
# Select a range of columns
mpg |>
  select(manufacturer:year)
```

```
# A tibble: 234 x 4
   manufacturer model      displ  year
   <chr>        <chr>      <dbl> <int>
 1 audi         a4           1.8  1999
 2 audi         a4           1.8  1999
 3 audi         a4           2    2008
 4 audi         a4           2    2008
 5 audi         a4           2.8  1999
 6 audi         a4           2.8  1999
 7 audi         a4           3.1  2008
 8 audi         a4 quattro   1.8  1999
 9 audi         a4 quattro   1.8  1999
10 audi         a4 quattro   2    2008
# i 224 more rows
```

```
# Helper functions
mpg |>
  select(starts_with("c"))   # columns starting with "c"
```

```
# A tibble: 234 x 3
     cyl   cty class
   <int> <int> <chr>
 1     4    18 compact
 2     4    21 compact
 3     4    20 compact
 4     4    21 compact
 5     6    16 compact
 6     6    18 compact
 7     6    18 compact
 8     4    18 compact
 9     4    16 compact
10     4    20 compact
# i 224 more rows
```

```
mpg |>
  select(where(is.numeric))  # only numeric columns
```

```
# A tibble: 234 x 5
   displ  year   cyl   cty   hwy
```

```
   <dbl> <int> <int> <int> <int>
 1   1.8  1999     4    18    29
 2   1.8  1999     4    21    29
 3   2    2008     4    20    31
 4   2    2008     4    21    30
 5   2.8  1999     6    16    26
 6   2.8  1999     6    18    26
 7   3.1  2008     6    18    27
 8   1.8  1999     4    18    26
 9   1.8  1999     4    16    25
10   2    2008     4    20    28
# i 224 more rows
```

filter() — Choose Rows

```
# Filter rows that meet a condition
mpg |>
  filter(manufacturer == "audi")
```

```
# A tibble: 18 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
 4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
11 audi         a4 quattro   2    2008     4 auto~ 4        19    27 p     comp~
12 audi         a4 quattro   2.8  1999     6 auto~ 4        15    25 p     comp~
13 audi         a4 quattro   2.8  1999     6 manu~ 4        17    25 p     comp~
14 audi         a4 quattro   3.1  2008     6 auto~ 4        17    25 p     comp~
15 audi         a4 quattro   3.1  2008     6 manu~ 4        15    25 p     comp~
16 audi         a6 quattro   2.8  1999     6 auto~ 4        15    24 p     mids~
17 audi         a6 quattro   3.1  2008     6 auto~ 4        17    25 p     mids~
18 audi         a6 quattro   4.2  2008     8 auto~ 4        16    23 p     mids~
```

```
# Multiple conditions (AND)
mpg |>
  filter(manufacturer == "audi", year == 2008)
```

```
# A tibble: 9 x 11
  manufacturer model      displ  year   cyl trans  drv     cty   hwy fl    class
```

```
  <chr>        <chr>      <dbl> <int> <int> <chr>  <chr> <int> <int> <chr> <chr>
1 audi         a4           2    2008     4 manua~ f        20    31 p     comp~
2 audi         a4           2    2008     4 auto(~ f        21    30 p     comp~
3 audi         a4           3.1  2008     6 auto(~ f        18    27 p     comp~
4 audi         a4 quattro   2    2008     4 manua~ 4        20    28 p     comp~
5 audi         a4 quattro   2    2008     4 auto(~ 4        19    27 p     comp~
6 audi         a4 quattro   3.1  2008     6 auto(~ 4        17    25 p     comp~
7 audi         a4 quattro   3.1  2008     6 manua~ 4        15    25 p     comp~
8 audi         a6 quattro   3.1  2008     6 auto(~ 4        17    25 p     mids~
9 audi         a6 quattro   4.2  2008     8 auto(~ 4        16    23 p     mids~
```

```
# OR conditions
mpg |>
  filter(manufacturer == "audi" | manufacturer == "bmw")
```

```
# A tibble: 18 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
 4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
11 audi         a4 quattro   2    2008     4 auto~ 4        19    27 p     comp~
12 audi         a4 quattro   2.8  1999     6 auto~ 4        15    25 p     comp~
13 audi         a4 quattro   2.8  1999     6 manu~ 4        17    25 p     comp~
14 audi         a4 quattro   3.1  2008     6 auto~ 4        17    25 p     comp~
15 audi         a4 quattro   3.1  2008     6 manu~ 4        15    25 p     comp~
16 audi         a6 quattro   2.8  1999     6 auto~ 4        15    24 p     mids~
17 audi         a6 quattro   3.1  2008     6 auto~ 4        17    25 p     mids~
18 audi         a6 quattro   4.2  2008     8 auto~ 4        16    23 p     mids~
```

```
# A shortcut for multiple OR on the same column
mpg |>
  filter(manufacturer %in% c("audi", "bmw", "toyota")) # This requires you know well your da
```

```
# A tibble: 52 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
```

```
 4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
# i 42 more rows
```

```
# Numeric comparisons
mpg |>
  filter(hwy > 30)
```

```
# A tibble: 22 x 11
   manufacturer model  displ  year   cyl trans     drv     cty   hwy fl    class
   <chr>        <chr>  <dbl> <int> <int> <chr>     <chr> <int> <int> <chr> <chr>
 1 audi         a4       2    2008     4 manual(m~ f        20    31 p     comp~
 2 honda        civic    1.6  1999     4 manual(m~ f        28    33 r     subc~
 3 honda        civic    1.6  1999     4 auto(l4)  f        24    32 r     subc~
 4 honda        civic    1.6  1999     4 manual(m~ f        25    32 r     subc~
 5 honda        civic    1.6  1999     4 auto(l4)  f        24    32 r     subc~
 6 honda        civic    1.8  2008     4 manual(m~ f        26    34 r     subc~
 7 honda        civic    1.8  2008     4 auto(l5)  f        25    36 r     subc~
 8 honda        civic    1.8  2008     4 auto(l5)  f        24    36 c     subc~
 9 hyundai      sonata   2.4  2008     4 manual(m~ f        21    31 r     mids~
10 nissan       altima   2.5  2008     4 auto(av)  f        23    31 r     mids~
# i 12 more rows
```

```
mpg |>
  filter(hwy >= 25, cty >= 20)
```

```
# A tibble: 56 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 2 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
 3 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
 4 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
 5 chevrolet    malibu       2.4  2008     4 auto~ f        22    30 r     mids~
 6 honda        civic        1.6  1999     4 manu~ f        28    33 r     subc~
 7 honda        civic        1.6  1999     4 auto~ f        24    32 r     subc~
 8 honda        civic        1.6  1999     4 manu~ f        25    32 r     subc~
 9 honda        civic        1.6  1999     4 manu~ f        23    29 p     subc~
10 honda        civic        1.6  1999     4 auto~ f        24    32 r     subc~
# i 46 more rows
```

mutate() — Create or Modify Columns

```
# Create a new column
mpg |>
  mutate(avg_mpg = (cty + hwy) / 2) # create avg_mpg at the last column
```

```
# A tibble: 234 x 12
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
 4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
# i 224 more rows
# i 1 more variable: avg_mpg <dbl>
```

```
mpg |>
  mutate(avg_mpg = (cty + hwy) / 2) |>
  select(manufacturer, model, cty, hwy, avg_mpg) # more control of the columns selected and
```

```
# A tibble: 234 x 5
   manufacturer model        cty   hwy avg_mpg
   <chr>        <chr>      <int> <int>   <dbl>
 1 audi         a4            18    29    23.5
 2 audi         a4            21    29    25
 3 audi         a4            20    31    25.5
 4 audi         a4            21    30    25.5
 5 audi         a4            16    26    21
 6 audi         a4            18    26    22
 7 audi         a4            18    27    22.5
 8 audi         a4 quattro    18    26    22
 9 audi         a4 quattro    16    25    20.5
10 audi         a4 quattro    20    28    24
# i 224 more rows
```

```
# Modify an existing column
mpg |>
  mutate(manufacturer = str_to_title(manufacturer)) |> # str_to_title() means convert text t
  select(manufacturer, model)
```

```
# A tibble: 234 x 2
```

```
   manufacturer model
   <chr>        <chr>
 1 Audi         a4
 2 Audi         a4
 3 Audi         a4
 4 Audi         a4
 5 Audi         a4
 6 Audi         a4
 7 Audi         a4
 8 Audi         a4 quattro
 9 Audi         a4 quattro
10 Audi         a4 quattro
# i 224 more rows
```

```
# Create multiple columns at once
mpg |>
  mutate(
    avg_mpg = (cty + hwy) / 2,
    fuel_efficiency = if_else(avg_mpg > 25, "Good", "Average") # if_else: if avg_mpg > 25, t
  ) |>
  select(manufacturer, model, avg_mpg, fuel_efficiency)
```

```
# A tibble: 234 x 4
   manufacturer model      avg_mpg fuel_efficiency
   <chr>        <chr>        <dbl> <chr>
 1 audi         a4            23.5 Average
 2 audi         a4            25   Average
 3 audi         a4            25.5 Good
 4 audi         a4            25.5 Good
 5 audi         a4            21   Average
 6 audi         a4            22   Average
 7 audi         a4            22.5 Average
 8 audi         a4 quattro    22   Average
 9 audi         a4 quattro    20.5 Average
10 audi         a4 quattro    24   Average
# i 224 more rows
```

```
mpg |>
  mutate(
    avg_mpg = (cty + hwy) / 2,
    fuel_efficiency = if_else(
      avg_mpg > 30, "Excellent",
      if_else(avg_mpg > 25, "Good", "Average")
    ) # If avg_mpg > 30 then "Excellent", otherwise if_else avg_mpg >25, then "Good", othwer
  ) |>
  select(manufacturer, model, avg_mpg, fuel_efficiency) |>
  arrange(desc(fuel_efficiency))
```

```
# A tibble: 234 x 4
   manufacturer model  avg_mpg fuel_efficiency
   <chr>        <chr>    <dbl> <chr>
 1 audi         a4        25.5 Good
 2 audi         a4        25.5 Good
 3 chevrolet    malibu    26   Good
 4 honda        civic     28   Good
 5 honda        civic     28.5 Good
 6 honda        civic     26   Good
 7 honda        civic     28   Good
 8 honda        civic     30   Good
 9 honda        civic     30   Good
10 hyundai      sonata    25.5 Good
# i 224 more rows
```

summarize() and group_by() — Aggregate Data

```
# Summarize the entire dataset
mpg |>
  summarize(
    avg_hwy = mean(hwy), # avg_hwy, the column name created; mean() is the function
    max_hwy = max(hwy), # maximum highway MPG
    count = n() # n() in dplyr means the number of rows in the current data.
  )
```

```
# A tibble: 1 x 3
  avg_hwy max_hwy count
    <dbl>   <int> <int>
1    23.4      44   234
```

```
# Group by a variable, then summarize
mpg |>
  group_by(manufacturer) |>
  summarize(
    avg_hwy = mean(hwy),
    count = n()
  ) |>
  arrange(desc(avg_hwy))
```

```
# A tibble: 15 x 3
   manufacturer avg_hwy count
   <chr>          <dbl> <int>
 1 honda           32.6     9
 2 volkswagen      29.2    27
 3 hyundai         26.9    14
 4 audi            26.4    18
 5 pontiac         26.4     5
```

```
 6 subaru          25.6    14
 7 toyota          24.9    34
 8 nissan          24.6    13
 9 chevrolet       21.9    19
10 ford            19.4    25
11 mercury         18       4
12 dodge           17.9    37
13 jeep            17.6     8
14 lincoln         17       3
15 land rover      16.5     4
```

```
# This code groups the data by manufacturer, so all cars made by the same manufacturer are p

# Group by multiple variables
mpg |>
  group_by(manufacturer, year) |> # the condition grouping by manufacturer, "and" not "or",
  summarize(
    avg_hwy = mean(hwy),
    count = n()
  )
```

```
# A tibble: 30 x 4
# Groups:   manufacturer [15]
   manufacturer  year avg_hwy count
   <chr>        <int>   <dbl> <int>
 1 audi          1999    26.1     9
 2 audi          2008    26.8     9
 3 chevrolet     1999    21.6     7
 4 chevrolet     2008    22.1    12
 5 dodge         1999    18.4    16
 6 dodge         2008    17.6    21
 7 ford          1999    18.6    15
 8 ford          2008    20.5    10
 9 honda         1999    31.6     5
10 honda         2008    33.8     4
# i 20 more rows
```

count() — A Handy Shortcut

```
# Count occurrences of each value
mpg |>
  count(manufacturer, sort = TRUE)
```

```
# A tibble: 15 x 2
   manufacturer     n
   <chr>        <int>
 1 dodge           37
```

```
 2 toyota          34
 3 volkswagen      27
 4 ford            25
 5 chevrolet       19
 6 audi            18
 7 hyundai         14
 8 subaru          14
 9 nissan          13
10 honda            9
11 jeep             8
12 pontiac          5
13 land rover       4
14 mercury          4
15 lincoln          3
```

```
  # or
  mpg |>
   count(manufacturer) |>
   arrange(desc(n)) # If for ascending: arrange(n)
```

```
# A tibble: 15 x 2
   manufacturer     n
   <chr>        <int>
 1 dodge           37
 2 toyota          34
 3 volkswagen      27
 4 ford            25
 5 chevrolet       19
 6 audi            18
 7 hyundai         14
 8 subaru          14
 9 nissan          13
10 honda            9
11 jeep             8
12 pontiac          5
13 land rover       4
14 mercury          4
15 lincoln          3
```

```
# Count combinations
mpg |>
  count(manufacturer, class, sort = TRUE)
```

```
# A tibble: 32 x 3
   manufacturer class          n
   <chr>        <chr>      <int>
 1 dodge        pickup        19
```

```
 2 audi         compact       15
 3 volkswagen   compact       14
 4 toyota       compact       12
 5 dodge        minivan       11
 6 chevrolet    suv            9
 7 ford         subcompact     9
 8 ford         suv            9
 9 honda        subcompact     9
10 jeep         suv            8
# i 22 more rows
```

Exercise 3.1: dplyr Practice

```
# Using the starwars dataset, answer these questions:

# 1. How many characters are from Tatooine? (filter + count or nrow)

# 2. What are the names and heights of the 5 tallest characters?
#    (select, arrange, head)

# 3. Create a new column called bmi = mass / (height/100)^2
#    Who has the highest BMI? (mutate, arrange)

# 4. What is the average height of characters grouped by species?
#    Only show species with more than 1 character. (group_by, summarize, filter)

# 5. How many characters have blue eyes? (filter, count)
```

Module 4: Data Tidying with tidyr

Learning Objectives

- Understand the concept of "tidy data"
- Reshape data from wide to long with `pivot_longer()`
- Reshape data from long to wide with `pivot_wider()`
- Handle missing values with `drop_na()` and `replace_na()`
- Separate and unite columns

Key Concepts

What is Tidy Data?

Tidy data has three rules:

1. Each **variable** has its own **column**
2. Each **observation** has its own **row**
3. Each **value** has its own **cell**

pivot_longer() — Wide to Long

```
# Create a wide dataset (common in spreadsheets)
grades_wide <- tibble(
  student = c("Alice", "Bob", "Carol"),
  math    = c(85, 92, 78),
  science = c(90, 88, 95),
  english = c(88, 76, 82)
)
grades_wide
```

```
# A tibble: 3 x 4
  student  math science english
  <chr>   <dbl>   <dbl>   <dbl>
1 Alice      85      90      88
2 Bob        92      88      76
3 Carol      78      95      82
```

```
  # or traditional dataframe
  grades_wide.d <- data.frame(
    student = c("Alice", "Bob", "Carol"),
    math    = c(85, 92, 78),
    science = c(90, 88, 95),
    english = c(88, 76, 82)
  )
  grades_wide.d
```

```
  student math science english
1   Alice   85      90      88
2     Bob   92      88      76
3   Carol   78      95      82
```

```
# Pivot to long format
grades_long <- grades_wide |>
  pivot_longer(
    cols = science:english, # columns to pivot; if only choose columns selectively, e.g., c
    #col = c(math,science,english),
    names_to = "subject",      # new column for the old column names
    values_to = "score"        # new column for the values
  )
grades_long # This is more preferable (or "Tidy") structure
```

```
# A tibble: 6 x 4
```

```
  student  math subject score
  <chr>   <dbl> <chr>   <dbl>
1 Alice      85 science    90
2 Alice      85 english    88
3 Bob        92 science    88
4 Bob        92 english    76
5 Carol      78 science    95
6 Carol      78 english    82
```

pivot_wider() — Long to Wide

```
# Convert back to wide format
grades_long |>
  pivot_wider(
    names_from = subject,
    values_from = score
  )
```

```
# A tibble: 3 x 4
  student  math science english
  <chr>   <dbl>   <dbl>   <dbl>
1 Alice      85      90      88
2 Bob        92      88      76
3 Carol      78      95      82
```

Handling Missing Values - drop_na() and replace_na()

```
names(starwars)
```

```
 [1] "name"       "height"     "mass"       "hair_color" "skin_color"
 [6] "eye_color"  "birth_year" "sex"        "gender"     "homeworld"
[11] "species"    "films"      "vehicles"   "starships"
```

```
# The starwars dataset has many missing values
starwars |>
  select(name, height, mass, hair_color) |>
  head(10)
```

```
# A tibble: 10 x 4
   name               height  mass hair_color
   <chr>               <int> <dbl> <chr>
 1 Luke Skywalker        172    77 blond
 2 C-3PO                 167    75 <NA>
 3 R2-D2                  96    32 <NA>
 4 Darth Vader           202   136 none
 5 Leia Organa           150    49 brown
 6 Owen Lars             178   120 brown, grey
```

```
 7 Beru Whitesun Lars    165    75 brown
 8 R5-D4                  97    32 <NA>
 9 Biggs Darklighter     183    84 black
10 Obi-Wan Kenobi        182    77 auburn, white
```

```
# Drop rows with ANY missing value
starwars |>
  select(name, height, mass) |>
  drop_na()
```

```
# A tibble: 59 x 3
   name               height  mass
   <chr>               <int> <dbl>
 1 Luke Skywalker        172    77
 2 C-3PO                 167    75
 3 R2-D2                  96    32
 4 Darth Vader           202   136
 5 Leia Organa           150    49
 6 Owen Lars             178   120
 7 Beru Whitesun Lars    165    75
 8 R5-D4                  97    32
 9 Biggs Darklighter     183    84
10 Obi-Wan Kenobi        182    77
# i 49 more rows
```

```
# Drop rows with missing values in specific columns only
starwars |>
  select(name, height, mass) |>
  drop_na(mass)
```

```
# A tibble: 59 x 3
   name               height  mass
   <chr>               <int> <dbl>
 1 Luke Skywalker        172    77
 2 C-3PO                 167    75
 3 R2-D2                  96    32
 4 Darth Vader           202   136
 5 Leia Organa           150    49
 6 Owen Lars             178   120
 7 Beru Whitesun Lars    165    75
 8 R5-D4                  97    32
 9 Biggs Darklighter     183    84
10 Obi-Wan Kenobi        182    77
# i 49 more rows
```

```
# Replace missing values
starwars |>
  select(name, hair_color) |>
```

```
  mutate(hair_color = replace_na(hair_color, "unknown")) |> # If `mutate()` uses the name of
  head(10)
```

```
# A tibble: 10 x 2
   name               hair_color
   <chr>              <chr>
 1 Luke Skywalker     blond
 2 C-3PO              unknown
 3 R2-D2              unknown
 4 Darth Vader        none
 5 Leia Organa        brown
 6 Owen Lars          brown, grey
 7 Beru Whitesun Lars brown
 8 R5-D4              unknown
 9 Biggs Darklighter  black
10 Obi-Wan Kenobi     auburn, white
```

separate() and unite()

```
# Create example data with combined columns
contacts <- tibble(
  name = c("Alice Smith", "Bob Jones", "Carol Lee"),
  phone = c("555-1234", "555-5678", "555-9012")
)

# Separate name into first and last
contacts |>
  separate(name, into = c("first_name", "last_name"), sep = " ") # into = ... is part of the
```

```
# A tibble: 3 x 3
  first_name last_name phone
  <chr>      <chr>     <chr>
1 Alice      Smith     555-1234
2 Bob        Jones     555-5678
3 Carol      Lee       555-9012
```

```
# Unite columns
contacts |>
  separate(name, into = c("first_name", "last_name"), sep = " ") |>
  unite("full_name",last_name, first_name, sep = ", ")
```

```
# A tibble: 3 x 2
  full_name    phone
  <chr>        <chr>
1 Smith, Alice 555-1234
2 Jones, Bob   555-5678
```

```
3 Lee, Carol   555-9012
```

Exercise 4.1: Tidying Data

```r
# 1. Create this messy dataset:
sales <- tibble(
  product = c("Widget", "Gadget", "Doohickey"),
  Q1_2024 = c(150, 200, 80),
  Q2_2024 = c(175, 180, 95),
  Q3_2024 = c(200, 210, 110),
  Q4_2024 = c(225, 190, 130)
)

# 2. Pivot it to long format with columns: product, quarter, revenue

# 3. Separate the 'quarter' column into 'quarter' and 'year'

# 4. Which product had the highest total revenue? (group_by + summarize)

# 5. Which quarter had the highest average revenue across all products?
```

Module 5: Visualization with Base R

Base R plotting comes built into every R installation — no packages needed. The functions are simple and direct: `plot()`, `hist()`, `boxplot()`, `barplot()`. You call a function, you get a picture.

```r
# Read and quick-clean the survey data
survey <- read.csv("sample_survey_data.csv")

summary(survey)
```

```
 respondent_id        age           gender           department
 Min.   : 1.00   Min.   :23.00   Length:40          Length:40
 1st Qu.:10.75   1st Qu.:29.00   Class :character   Class :character
 Median :20.50   Median :33.50   Mode  :character   Mode  :character
 Mean   :20.50   Mean   :34.35
 3rd Qu.:30.25   3rd Qu.:39.25
 Max.   :40.00   Max.   :50.00

 years_experience satisfaction_score monthly_hours remote_days_per_week
 Min.   : 1.000   Min.   :3.000      Min.   :144   Min.   :0.00
 1st Qu.: 3.000   1st Qu.:3.575      1st Qu.:160   1st Qu.:1.00
 Median : 6.000   Median :4.000      Median :170   Median :2.00
```

```
 Mean   : 7.541   Mean   :3.930      Mean   :169   Mean   :2.35
 3rd Qu.:10.000   3rd Qu.:4.300      3rd Qu.:179   3rd Qu.:3.00
 Max.   :22.000   Max.   :4.800      Max.   :192   Max.   :5.00
 NA's   :3                           NA's   :1
 training_completed annual_salary
 Mode :logical      Min.   : 44000
 FALSE:10           1st Qu.: 56000
 TRUE :30           Median : 66000
                    Mean   : 69189
                    3rd Qu.: 75000
                    Max.   :115000
                    NA's   :3
```

```
# Replace blank department with "Unknown"
survey$department[survey$department == ""] <- "Unknown" #  [ ] is used to access or replace

# Remove rows where years_experience is missing
survey <- survey[!is.na(survey$years_experience), ]

str(survey)
```

```
'data.frame':   37 obs. of  10 variables:
 $ respondent_id       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age                 : int  28 35 42 25 31 38 29 45 33 27 ...
 $ gender              : chr  "Female" "Male" "Female" "Male" ...
 $ department          : chr  "Marketing" "Engineering" "Engineering" "Sales" ...
 $ years_experience    : int  3 8 15 1 5 10 4 18 7 2 ...
 $ satisfaction_score  : num  4.2 3.8 4.5 3.2 4 3.5 4.3 3.9 4.1 3.6 ...
 $ monthly_hours       : int  165 180 170 175 160 155 185 170 150 168 ...
 $ remote_days_per_week: int  2 3 4 1 2 3 4 1 2 2 ...
 $ training_completed  : logi  TRUE TRUE TRUE FALSE TRUE TRUE ...
 $ annual_salary       : int  52000 85000 105000 45000 58000 72000 78000 68000 NA 48000 ...
```

```
nrow(survey)
```

```
[1] 37
```

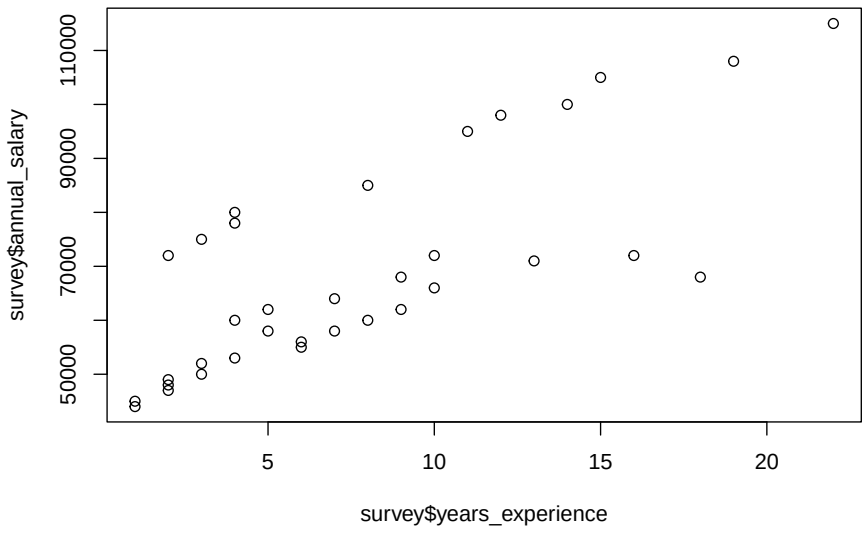
We have 37 rows with clean departments and no missing salaries. Let's plot.

Four Plots You'll Use All the Time

Scatter Plot — `plot()`

The workhorse of base R graphics. Give it an x and a y and it draws points.

```
# Basic scatter plot (two quantitative varaibales)
plot(survey$years_experience, survey$annual_salary)
```



That works, but it's ugly. Let's add labels, color, and a trend line.

```
# Color points by department

unique(survey$department)
```

```
[1] "Marketing"   "Engineering" "Sales"       "HR"          "Unknown"
```

```
dept_colors <- c(
  Engineering = "steelblue",
  HR = "tomato",
  Marketing = "forestgreen",
  Sales = "darkorange",
  Unknown = "gray50"
)

colors() # all named colors
```

```
  [1] "white"                "aliceblue"            "antiquewhite"
  [4] "antiquewhite1"        "antiquewhite2"        "antiquewhite3"
  [7] "antiquewhite4"        "aquamarine"           "aquamarine1"
 [10] "aquamarine2"          "aquamarine3"          "aquamarine4"
 [13] "azure"                "azure1"               "azure2"
 [16] "azure3"               "azure4"               "beige"
 [19] "bisque"               "bisque1"              "bisque2"
 [22] "bisque3"              "bisque4"              "black"
```

```
 [25] "blanchedalmond"      "blue"               "blue1"
 [28] "blue2"               "blue3"              "blue4"
 [31] "blueviolet"          "brown"              "brown1"
 [34] "brown2"              "brown3"             "brown4"
 [37] "burlywood"           "burlywood1"         "burlywood2"
 [40] "burlywood3"          "burlywood4"         "cadetblue"
 [43] "cadetblue1"          "cadetblue2"         "cadetblue3"
 [46] "cadetblue4"          "chartreuse"         "chartreuse1"
 [49] "chartreuse2"         "chartreuse3"        "chartreuse4"
 [52] "chocolate"           "chocolate1"         "chocolate2"
 [55] "chocolate3"          "chocolate4"         "coral"
 [58] "coral1"              "coral2"             "coral3"
 [61] "coral4"              "cornflowerblue"     "cornsilk"
 [64] "cornsilk1"           "cornsilk2"          "cornsilk3"
 [67] "cornsilk4"           "cyan"               "cyan1"
 [70] "cyan2"               "cyan3"              "cyan4"
 [73] "darkblue"            "darkcyan"           "darkgoldenrod"
 [76] "darkgoldenrod1"      "darkgoldenrod2"     "darkgoldenrod3"
 [79] "darkgoldenrod4"      "darkgray"           "darkgreen"
 [82] "darkgrey"            "darkkhaki"          "darkmagenta"
 [85] "darkolivegreen"      "darkolivegreen1"    "darkolivegreen2"
 [88] "darkolivegreen3"     "darkolivegreen4"    "darkorange"
 [91] "darkorange1"         "darkorange2"        "darkorange3"
 [94] "darkorange4"         "darkorchid"         "darkorchid1"
 [97] "darkorchid2"         "darkorchid3"        "darkorchid4"
[100] "darkred"             "darksalmon"         "darkseagreen"
[103] "darkseagreen1"       "darkseagreen2"      "darkseagreen3"
[106] "darkseagreen4"       "darkslateblue"      "darkslategray"
[109] "darkslategray1"      "darkslategray2"     "darkslategray3"
[112] "darkslategray4"      "darkslategrey"      "darkturquoise"
[115] "darkviolet"          "deeppink"           "deeppink1"
[118] "deeppink2"           "deeppink3"          "deeppink4"
[121] "deepskyblue"         "deepskyblue1"       "deepskyblue2"
[124] "deepskyblue3"        "deepskyblue4"       "dimgray"
[127] "dimgrey"             "dodgerblue"         "dodgerblue1"
[130] "dodgerblue2"         "dodgerblue3"        "dodgerblue4"
[133] "firebrick"           "firebrick1"         "firebrick2"
[136] "firebrick3"          "firebrick4"         "floralwhite"
[139] "forestgreen"         "gainsboro"          "ghostwhite"
[142] "gold"                "gold1"              "gold2"
[145] "gold3"               "gold4"              "goldenrod"
[148] "goldenrod1"          "goldenrod2"         "goldenrod3"
[151] "goldenrod4"          "gray"               "gray0"
[154] "gray1"               "gray2"              "gray3"
[157] "gray4"               "gray5"              "gray6"
[160] "gray7"               "gray8"              "gray9"
```

```
[163] "gray10"          "gray11"          "gray12"
[166] "gray13"          "gray14"          "gray15"
[169] "gray16"          "gray17"          "gray18"
[172] "gray19"          "gray20"          "gray21"
[175] "gray22"          "gray23"          "gray24"
[178] "gray25"          "gray26"          "gray27"
[181] "gray28"          "gray29"          "gray30"
[184] "gray31"          "gray32"          "gray33"
[187] "gray34"          "gray35"          "gray36"
[190] "gray37"          "gray38"          "gray39"
[193] "gray40"          "gray41"          "gray42"
[196] "gray43"          "gray44"          "gray45"
[199] "gray46"          "gray47"          "gray48"
[202] "gray49"          "gray50"          "gray51"
[205] "gray52"          "gray53"          "gray54"
[208] "gray55"          "gray56"          "gray57"
[211] "gray58"          "gray59"          "gray60"
[214] "gray61"          "gray62"          "gray63"
[217] "gray64"          "gray65"          "gray66"
[220] "gray67"          "gray68"          "gray69"
[223] "gray70"          "gray71"          "gray72"
[226] "gray73"          "gray74"          "gray75"
[229] "gray76"          "gray77"          "gray78"
[232] "gray79"          "gray80"          "gray81"
[235] "gray82"          "gray83"          "gray84"
[238] "gray85"          "gray86"          "gray87"
[241] "gray88"          "gray89"          "gray90"
[244] "gray91"          "gray92"          "gray93"
[247] "gray94"          "gray95"          "gray96"
[250] "gray97"          "gray98"          "gray99"
[253] "gray100"         "green"           "green1"
[256] "green2"          "green3"          "green4"
[259] "greenyellow"     "grey"            "grey0"
[262] "grey1"           "grey2"           "grey3"
[265] "grey4"           "grey5"           "grey6"
[268] "grey7"           "grey8"           "grey9"
[271] "grey10"          "grey11"          "grey12"
[274] "grey13"          "grey14"          "grey15"
[277] "grey16"          "grey17"          "grey18"
[280] "grey19"          "grey20"          "grey21"
[283] "grey22"          "grey23"          "grey24"
[286] "grey25"          "grey26"          "grey27"
[289] "grey28"          "grey29"          "grey30"
[292] "grey31"          "grey32"          "grey33"
[295] "grey34"          "grey35"          "grey36"
[298] "grey37"          "grey38"          "grey39"
```

```
[301] "grey40"               "grey41"               "grey42"
[304] "grey43"               "grey44"               "grey45"
[307] "grey46"               "grey47"               "grey48"
[310] "grey49"               "grey50"               "grey51"
[313] "grey52"               "grey53"               "grey54"
[316] "grey55"               "grey56"               "grey57"
[319] "grey58"               "grey59"               "grey60"
[322] "grey61"               "grey62"               "grey63"
[325] "grey64"               "grey65"               "grey66"
[328] "grey67"               "grey68"               "grey69"
[331] "grey70"               "grey71"               "grey72"
[334] "grey73"               "grey74"               "grey75"
[337] "grey76"               "grey77"               "grey78"
[340] "grey79"               "grey80"               "grey81"
[343] "grey82"               "grey83"               "grey84"
[346] "grey85"               "grey86"               "grey87"
[349] "grey88"               "grey89"               "grey90"
[352] "grey91"               "grey92"               "grey93"
[355] "grey94"               "grey95"               "grey96"
[358] "grey97"               "grey98"               "grey99"
[361] "grey100"              "honeydew"             "honeydew1"
[364] "honeydew2"            "honeydew3"            "honeydew4"
[367] "hotpink"              "hotpink1"             "hotpink2"
[370] "hotpink3"             "hotpink4"             "indianred"
[373] "indianred1"           "indianred2"           "indianred3"
[376] "indianred4"           "ivory"                "ivory1"
[379] "ivory2"               "ivory3"               "ivory4"
[382] "khaki"                "khaki1"               "khaki2"
[385] "khaki3"               "khaki4"               "lavender"
[388] "lavenderblush"        "lavenderblush1"       "lavenderblush2"
[391] "lavenderblush3"       "lavenderblush4"       "lawngreen"
[394] "lemonchiffon"         "lemonchiffon1"        "lemonchiffon2"
[397] "lemonchiffon3"        "lemonchiffon4"        "lightblue"
[400] "lightblue1"           "lightblue2"           "lightblue3"
[403] "lightblue4"           "lightcoral"           "lightcyan"
[406] "lightcyan1"           "lightcyan2"           "lightcyan3"
[409] "lightcyan4"           "lightgoldenrod"       "lightgoldenrod1"
[412] "lightgoldenrod2"      "lightgoldenrod3"      "lightgoldenrod4"
[415] "lightgoldenrodyellow" "lightgray"            "lightgreen"
[418] "lightgrey"            "lightpink"            "lightpink1"
[421] "lightpink2"           "lightpink3"           "lightpink4"
[424] "lightsalmon"          "lightsalmon1"         "lightsalmon2"
[427] "lightsalmon3"         "lightsalmon4"         "lightseagreen"
[430] "lightskyblue"         "lightskyblue1"        "lightskyblue2"
[433] "lightskyblue3"        "lightskyblue4"        "lightslateblue"
[436] "lightslategray"       "lightslategrey"       "lightsteelblue"
```

```
[439] "lightsteelblue1"      "lightsteelblue2"      "lightsteelblue3"
[442] "lightsteelblue4"      "lightyellow"          "lightyellow1"
[445] "lightyellow2"         "lightyellow3"         "lightyellow4"
[448] "limegreen"            "linen"                "magenta"
[451] "magenta1"             "magenta2"             "magenta3"
[454] "magenta4"             "maroon"               "maroon1"
[457] "maroon2"              "maroon3"              "maroon4"
[460] "mediumaquamarine"     "mediumblue"           "mediumorchid"
[463] "mediumorchid1"        "mediumorchid2"        "mediumorchid3"
[466] "mediumorchid4"        "mediumpurple"         "mediumpurple1"
[469] "mediumpurple2"        "mediumpurple3"        "mediumpurple4"
[472] "mediumseagreen"       "mediumslateblue"      "mediumspringgreen"
[475] "mediumturquoise"      "mediumvioletred"      "midnightblue"
[478] "mintcream"            "mistyrose"            "mistyrose1"
[481] "mistyrose2"           "mistyrose3"           "mistyrose4"
[484] "moccasin"             "navajowhite"          "navajowhite1"
[487] "navajowhite2"         "navajowhite3"         "navajowhite4"
[490] "navy"                 "navyblue"             "oldlace"
[493] "olivedrab"            "olivedrab1"           "olivedrab2"
[496] "olivedrab3"           "olivedrab4"           "orange"
[499] "orange1"              "orange2"              "orange3"
[502] "orange4"              "orangered"            "orangered1"
[505] "orangered2"           "orangered3"           "orangered4"
[508] "orchid"               "orchid1"              "orchid2"
[511] "orchid3"              "orchid4"              "palegoldenrod"
[514] "palegreen"            "palegreen1"           "palegreen2"
[517] "palegreen3"           "palegreen4"           "paleturquoise"
[520] "paleturquoise1"       "paleturquoise2"       "paleturquoise3"
[523] "paleturquoise4"       "palevioletred"        "palevioletred1"
[526] "palevioletred2"       "palevioletred3"       "palevioletred4"
[529] "papayawhip"           "peachpuff"            "peachpuff1"
[532] "peachpuff2"           "peachpuff3"           "peachpuff4"
[535] "peru"                 "pink"                 "pink1"
[538] "pink2"                "pink3"                "pink4"
[541] "plum"                 "plum1"                "plum2"
[544] "plum3"                "plum4"                "powderblue"
[547] "purple"               "purple1"              "purple2"
[550] "purple3"              "purple4"              "red"
[553] "red1"                 "red2"                 "red3"
[556] "red4"                 "rosybrown"            "rosybrown1"
[559] "rosybrown2"           "rosybrown3"           "rosybrown4"
[562] "royalblue"            "royalblue1"           "royalblue2"
[565] "royalblue3"           "royalblue4"           "saddlebrown"
[568] "salmon"               "salmon1"              "salmon2"
[571] "salmon3"              "salmon4"              "sandybrown"
[574] "seagreen"             "seagreen1"            "seagreen2"
```

```
[577] "seagreen3"           "seagreen4"           "seashell"
[580] "seashell1"           "seashell2"           "seashell3"
[583] "seashell4"           "sienna"              "sienna1"
[586] "sienna2"             "sienna3"             "sienna4"
[589] "skyblue"             "skyblue1"            "skyblue2"
[592] "skyblue3"            "skyblue4"            "slateblue"
[595] "slateblue1"          "slateblue2"          "slateblue3"
[598] "slateblue4"          "slategray"           "slategray1"
[601] "slategray2"          "slategray3"          "slategray4"
[604] "slategrey"           "snow"                "snow1"
[607] "snow2"               "snow3"               "snow4"
[610] "springgreen"         "springgreen1"        "springgreen2"
[613] "springgreen3"        "springgreen4"        "steelblue"
[616] "steelblue1"          "steelblue2"          "steelblue3"
[619] "steelblue4"          "tan"                 "tan1"
[622] "tan2"                "tan3"                "tan4"
[625] "thistle"             "thistle1"            "thistle2"
[628] "thistle3"            "thistle4"            "tomato"
[631] "tomato1"             "tomato2"             "tomato3"
[634] "tomato4"             "turquoise"           "turquoise1"
[637] "turquoise2"          "turquoise3"          "turquoise4"
[640] "violet"              "violetred"           "violetred1"
[643] "violetred2"          "violetred3"          "violetred4"
[646] "wheat"               "wheat1"              "wheat2"
[649] "wheat3"              "wheat4"              "whitesmoke"
[652] "yellow"              "yellow1"             "yellow2"
[655] "yellow3"             "yellow4"             "yellowgreen"
```

```r
point_cols <- dept_colors[survey$department]
# It looks at each department in survey$department, matches it to the corresponding named co

# Create a scatter plot of annual salary against years of experience.
# Each point is colored according to department using point_cols.
# pch = 19 uses solid circles, and cex = 1.3 makes the points slightly larger.
# xlab and ylab set the axis labels, and main adds the plot title.

plot(
  survey$years_experience, survey$annual_salary,
  col = point_cols,
  pch = 19, # solid circles (pch = 19)
  cex = 1.3, # point size
  xlab = "Years of Experience",
  ylab = "Annual Salary ($)",
  main = "Experience vs. Salary"
)
```

```r
# Common ones:
# pch = 1: open circle
# pch = 15: filled square
# pch = 16: filled circle
# pch = 17: filled triangle
# pch = 19: solid circle

# Add a trend line
abline(
  lm(annual_salary ~ years_experience, data = survey),
  lty = 2, col = "gray30", lwd = 2
)

# lty = 1: solid line
# lty = 2: dashed line
# lty = 3: dotted line
# lty = 4: dot-dash line
# lty = 5: long-dash line
# lty = 6: two-dash line

# Add a legend
legend("topleft",
       legend = names(dept_colors),
       col    = dept_colors,
       pch    = 19,
       cex    = 0.8,
       bty    = "n")          # no legend box
```


```
# Common bty values:
# bty = "o": full box around the plot
# bty = "l": only left and bottom lines
# bty = "7": top and right removed
# bty = "c": left removed
# bty = "u": top removed
# bty = "]": left and top removed
# bty = "n": no box
```

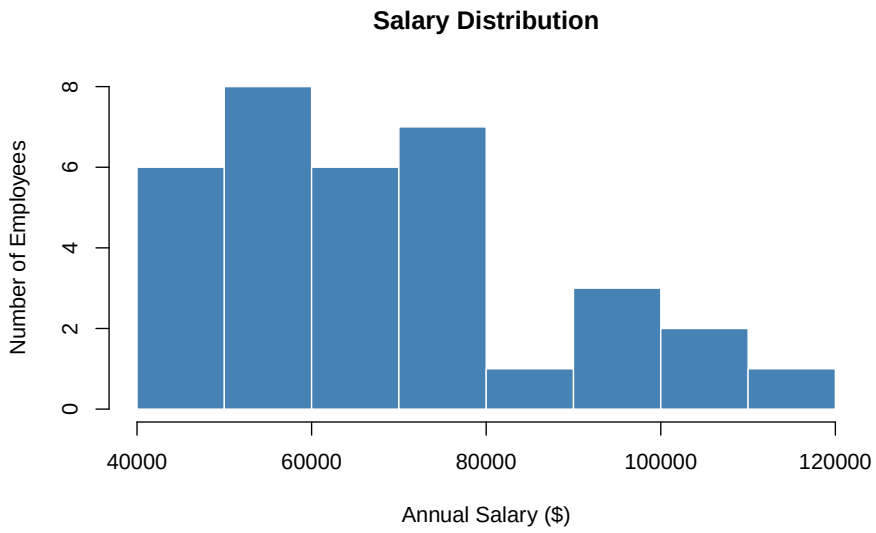
> **Tip**
>
> Key arguments for `plot()` `pch` = point shape (19 = solid circle), `cex` = size multiplier, `col` = color, `lwd` = line width, `lty` = line type (1 = solid, 2 = dashed).

Histogram — `hist()`

Shows how a single numeric variable is distributed.

```
hist(survey$annual_salary,
     breaks = 8,                          # number of bins
     col    = "steelblue",
     border = "white",
     main   = "Salary Distribution",
```

```
     xlab   = "Annual Salary ($)",
     ylab   = "Number of Employees")
```

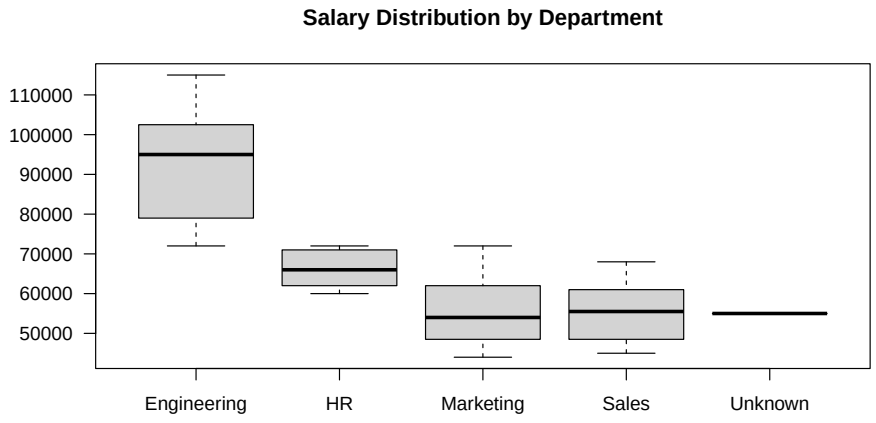


> **Tip**
>
> Choosing `breaks` `breaks` controls how many bins. More bins = more detail but noisier. Try `breaks = 5`, then `breaks = 15`, and see what tells the clearest story.

Box Plot — `boxplot()`

Compares the distribution of a number across groups (Quantitative vs Categorical).

```
boxplot(annual_salary ~ department,
        data = survey,
        # col  = c("steelblue", "tomato", "forestgreen",
        #          "darkorange", "gray80"),
        main = "Salary Distribution by Department",
        xlab = "",
        ylab = "",
        las  = 1)                          # horizontal y-axis labels
```

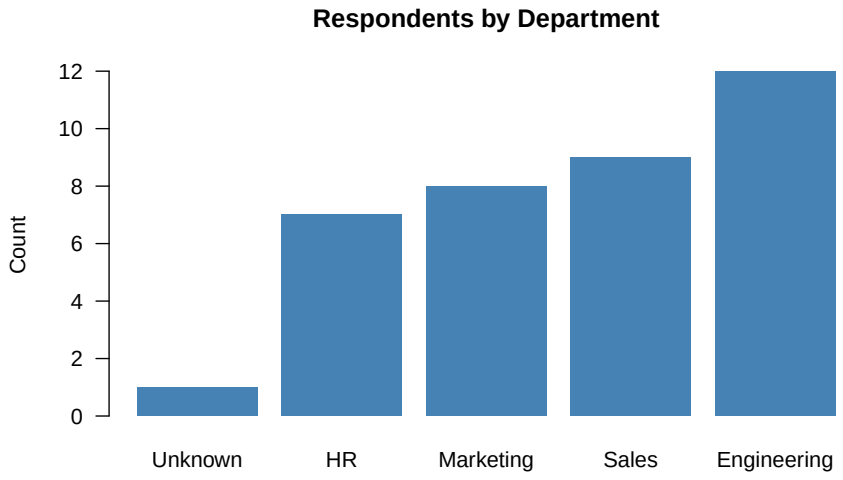


The box is the middle 50% (IQR). The thick line is the median. Dots beyond the whiskers are potential outliers.

Bar Plot — `barplot()`

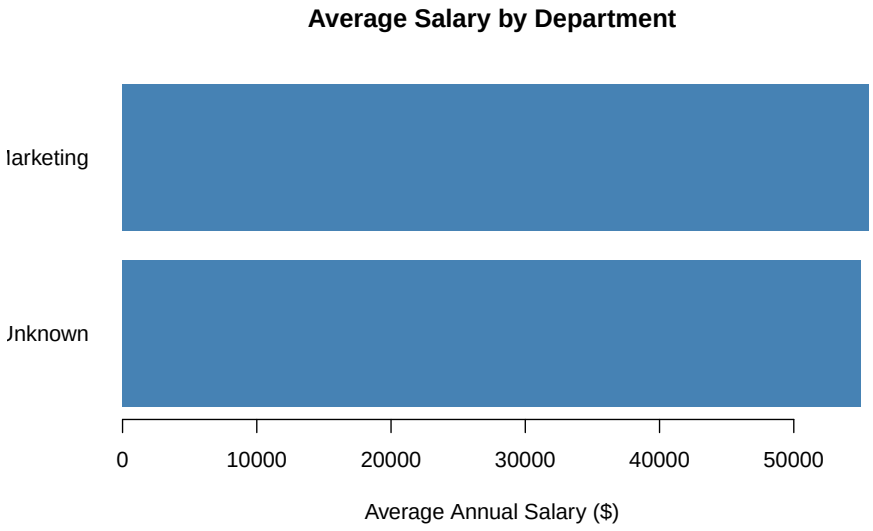
Shows counts or pre-calculated values as bars.

```
# Count respondents per department
dept_counts <- table(survey$department)

barplot(sort(dept_counts),
        col      = "steelblue",
        main     = "Respondents by Department",
        ylab     = "Count",
        las      = 1,
        border   = NA)                  # cleaner look without borders
```



For **pre-calculated values** (like averages), compute first then plot:

```
# Average salary by department
avg_sal <- tapply(survey$annual_salary, survey$department, mean)
avg_sal <- sort(avg_sal)

# Horizontal bar plot
barplot(avg_sal,
        horiz  = TRUE,
        col    = "steelblue",
        main   = "Average Salary by Department",
        xlab   = "Average Annual Salary ($)",
        las    = 1,
        border = NA)
```



> **Tip**
>
> `table()` and `tapply()` — your bar plot friends `table(x)` counts how many of each value. `tapply(value, group, function)` applies a function (like `mean`) to each group — perfect for feeding into `barplot()`.

Polish Your Plots

Titles and Labels

Every base R plot function accepts these arguments:

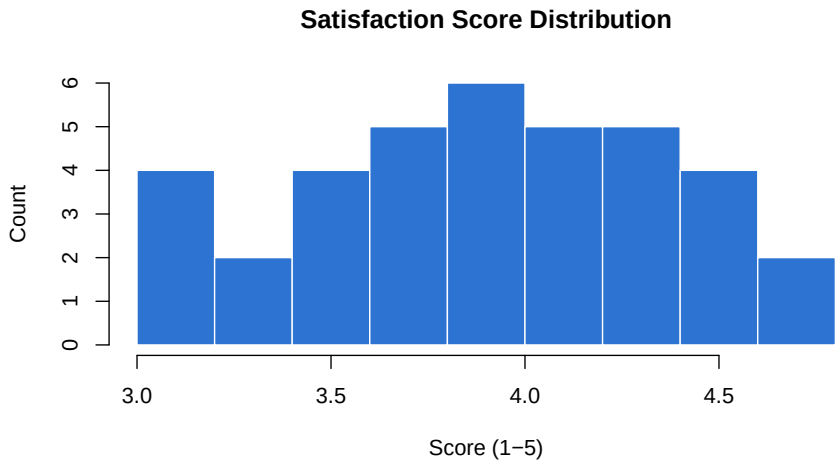
Argument	What it does	Example
`main`	Title above the plot	`main = "My Title"`
`sub`	Subtitle below the plot	`sub = "Source: survey"`
`xlab`	X-axis label	`xlab = "Age"`
`ylab`	Y-axis label	`ylab = "Salary ($)"`

Colors

You can use color names (`"steelblue"`, `"tomato"`) or hex codes (`"#2C73D2"`). Run `colors()` in R to see all 657 built-in color names.

```
hist(survey$satisfaction_score,
     breaks = 10,
```

```
     col    = "#2C73D2",
     border = "white",
     main   = "Satisfaction Score Distribution",
     xlab   = "Score (1-5)",
     ylab   = "Count")
```



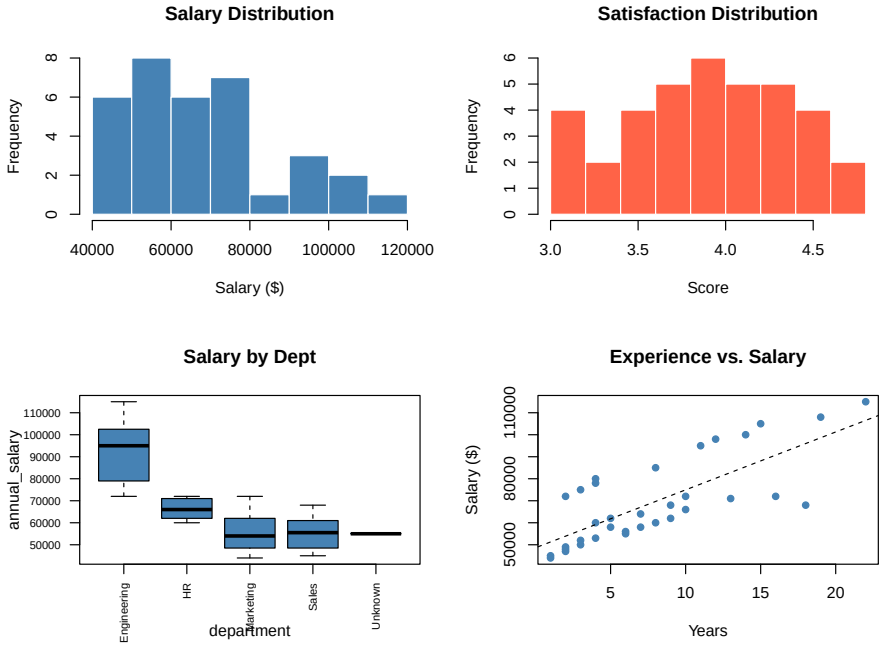
Adding Extra Elements

```
plot(survey$age, survey$annual_salary,
     pch  = 19,
     col  = "steelblue",
     cex  = 1.2,
     xlab = "Age",
     ylab = "Annual Salary ($)",
     main = "Age vs. Salary")

# Trend line
abline(lm(annual_salary ~ age, data = survey),
       col = "tomato", lwd = 2, lty = 2)

# Horizontal reference line at the mean salary
abline(h = mean(survey$annual_salary),
       col = "gray50", lty = 3)

# Label the reference line
text(x = 26, y = mean(survey$annual_salary) + 2500,
     labels = paste("Mean:", round(mean(survey$annual_salary))),
     cex = 0.8, col = "gray40")
```

```r
# Add a grid for readability
grid(col = "gray90")
```

Multi-Panel Layouts

```r
par(mfrow = c(2, 2))   # 2 rows, 2 columns

hist(survey$annual_salary, col = "steelblue", border = "white",
     main = "Salary Distribution", xlab = "Salary ($)")

hist(survey$satisfaction_score, col = "tomato", border = "white",
     main = "Satisfaction Distribution", xlab = "Score")

boxplot(annual_salary ~ department, data = survey,
        col = "steelblue", main = "Salary by Dept", las = 2, cex.axis = 0.7)

plot(survey$years_experience, survey$annual_salary,
     pch = 19, col = "steelblue", cex = 0.9,
     main = "Experience vs. Salary", xlab = "Years", ylab = "Salary ($)")
abline(lm(annual_salary ~ years_experience, data = survey), lty = 2)
```




```
par(mfrow = c(1, 1))   # reset to single panel
```

Saving Plots

```
getwd()

# Simplest method: Using the "Plots" tab to export plots

# Save to PNG
png("salary_scatter.png", width = 700, height = 450, res = 150)
plot(survey$years_experience, survey$annual_salary,
     pch = 19, col = "steelblue",
     xlab = "Experience", ylab = "Salary",
     main = "Experience vs. Salary")
abline(lm(annual_salary ~ years_experience, data = survey), lty = 2)
dev.off()   # IMPORTANT: closes the file

# You do not need an existing PNG file first. `png()` creates a new image file, and the plot

# Save to PDF (great for publications)
pdf("salary_scatter.pdf", width = 7, height = 4.5)
```

```
# ... same plot code ...
dev.off()
```

> **Warning**
>
> Don't forget `dev.off()`! When saving to a file, you must call `dev.off()` after your plot code. Otherwise the file won't be written and your plots will stop showing up in RStudio.

Cheat Sheet

Which Function Do I Need?

Question	Function	Example
How do two numbers relate?	`plot(x, y)`	`plot(age, salary)`
How is a number distributed?	`hist(x)`	`hist(salary)`
How do groups compare?	`boxplot(y ~ group)`	`boxplot(salary ~ dept)`
How many per group?	`barplot(table(x))`	`barplot(table(dept))`
What's the average per group?	`barplot(tapply(...))`	`barplot(tapply(sal, dept, mean))`

Common Appearance Arguments

Argument	Controls	Values
`col`	Fill color	`"steelblue"`, `"#FF6347"`, `c("red","blue")`
`border`	Border color	`"white"`, `NA` (no border)
`pch`	Point shape	19 = solid circle, 1 = open circle, 17 = triangle
`cex`	Size multiplier	1 = default, 1.5 = 50% bigger
`lwd`	Line width	1 = default, 2 = double
`lty`	Line type	1 = solid, 2 = dashed, 3 = dotted

Argument	Controls	Values
`las`	Axis label direction	1 = horizontal, 2 = perpendicular

Adding Extras to a Plot

Function	What it adds
`abline(lm(...))`	Trend line
`abline(h = value)`	Horizontal line
`abline(v = value)`	Vertical line
`legend(...)`	Legend
`text(x, y, label)`	Text label at a point
`grid()`	Grid lines
`points(x, y)`	More points on top
`lines(x, y)`	Lines on top

Module 6: Leveraging AI Tools

Learning Objectives

- Understand how generative AI can assist your R workflow
- Learn effective prompting strategies for R code generation
- Use AI to debug errors and understand code
- Know the limitations and best practices

How AI Can Help You Learn R

1. Writing Code from Natural Language

You can describe what you want to accomplish in plain English, and AI tools can generate the R code for you.

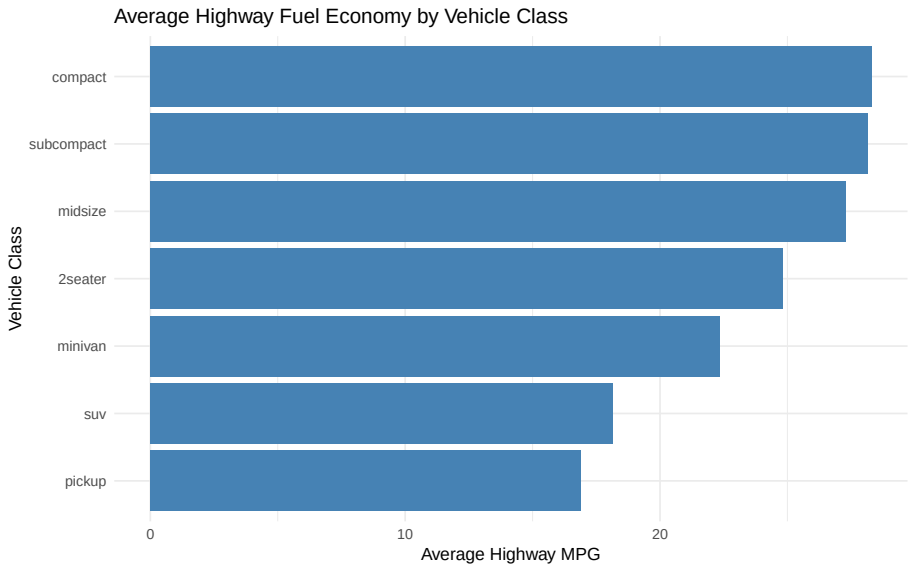
Example prompt:

> "Using the mpg dataset in R with the tidyverse, create a bar chart showing the average highway mpg for each car class, sorted from highest to lowest, with a clean minimal theme."

What you might get back:

```
mpg |>
  group_by(class) |>
  summarize(avg_hwy = mean(hwy)) |>
```

```r
ggplot(aes(x = reorder(class, avg_hwy), y = avg_hwy)) +
geom_col(fill = "steelblue") +
coord_flip() +
labs(x = "Vehicle Class", y = "Average Highway MPG",
     title = "Average Highway Fuel Economy by Vehicle Class") +
theme_minimal()
```



2. Debugging Error Messages

When you get an error, you can paste it into an AI tool and ask for help.

Example: "I'm getting this error in R. What does it mean and how do I fix it?"

```
Error in select(., name, height) :
  unused arguments (name, height)
```

AI would explain: This typically happens when another package (like MASS) has overwritten dplyr's `select()`. Fix it by being explicit: `dplyr::select()`.

3. Explaining Code

Paste code you don't understand and ask AI to explain it line by line.

4. Discovering New Methods

Ask AI things like:

- "What's the best way to remove duplicate rows in R?"
- "How do I join two datasets in the tidyverse?"

- "What alternatives exist for creating interactive plots in R?"

Tips for Effective AI Prompting for R

Be Specific About Your Data

```
# BAD prompt:  "Make a plot of my data"
# GOOD prompt: "Using ggplot2, create a scatter plot of the mpg dataset
#               with displ on the x-axis and hwy on the y-axis,
#               colored by the drv variable, with a smooth trend line."
```

Include Context

Tell the AI:

- What packages you're using (tidyverse, specific packages)
- What your data looks like (column names, types, sample rows)
- What you've already tried
- What output format you want

Ask for Explanations

Add "and explain each step" to your prompts to learn while coding.

Iterate

Start with a basic request, then refine: "Now change the colors to a blue-to-red gradient" or "Add error bars."

Practical AI Exercise

```
# Try these prompts with your AI tool of choice (ChatGPT, Claude, etc.):

# 1. "Write tidyverse code to find the top 5 most expensive diamond cuts
#     in the diamonds dataset, showing average price and count."

# 2. Paste this buggy code and ask AI to fix it:
#    mpg %>%
#      filter(class = "suv") %>%
#      sumamrize(mean_hwy == mean(hwy))

# 3. Ask: "Explain what the following code does step by step:"
#    diamonds |>
#      filter(carat > 1) |>
#      group_by(cut, color) |>
#      summarize(avg_price = mean(price), .groups = "drop") |>
#      pivot_wider(names_from = color, values_from = avg_price)
```

Best Practices and Caveats

1. **Always test AI-generated code** — Run it yourself and verify it works
2. **Understand what the code does** — Don't just copy-paste blindly
3. **AI can make mistakes** — Especially with newer packages or niche functions
4. **Use AI as a learning accelerator** — Not a replacement for understanding
5. **Start simple, build up** — Ask for basic code first, then add complexity
6. **Cross-reference with documentation** — Use `?function_name` in R for official docs

---

Additional: Data Visualization with ggplot2

Learning Objectives

- Understand the grammar of graphics (data + aesthetics + geometries)
- Create common plot types: scatter, bar, line, histogram, box plot
- Customize colors, labels, and themes
- Save plots to files

Key Concepts

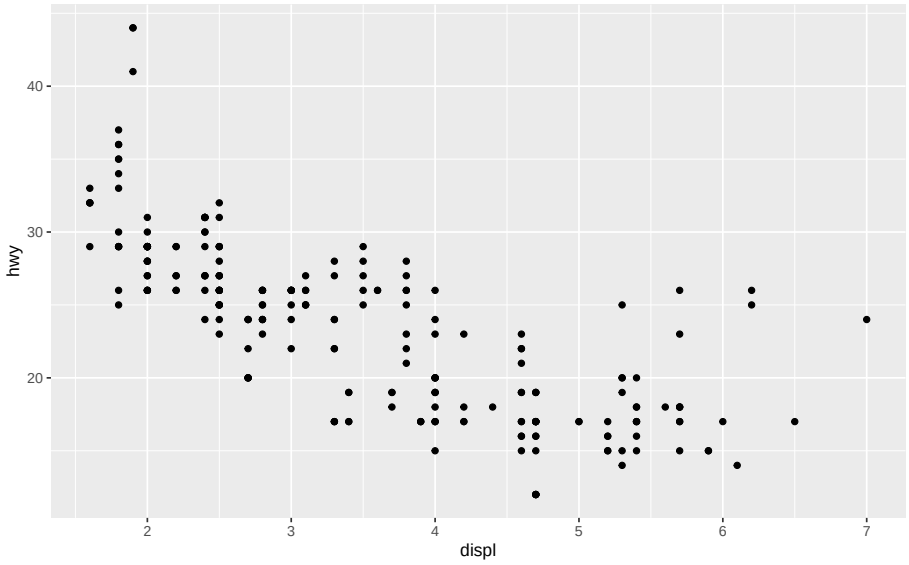
The Grammar of Graphics

Every ggplot has three essential components:

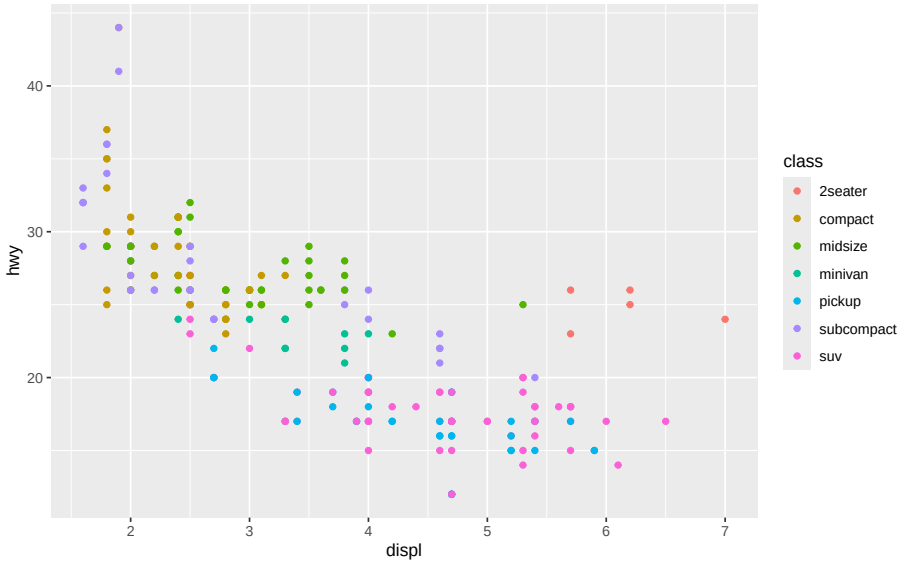
1. **Data** — the dataset
2. **Aesthetics** (`aes()`) — what variables map to x, y, color, size, etc.
3. **Geometries** (`geom_*()`) — how to draw the data (points, bars, lines, etc.)

Scatter Plots

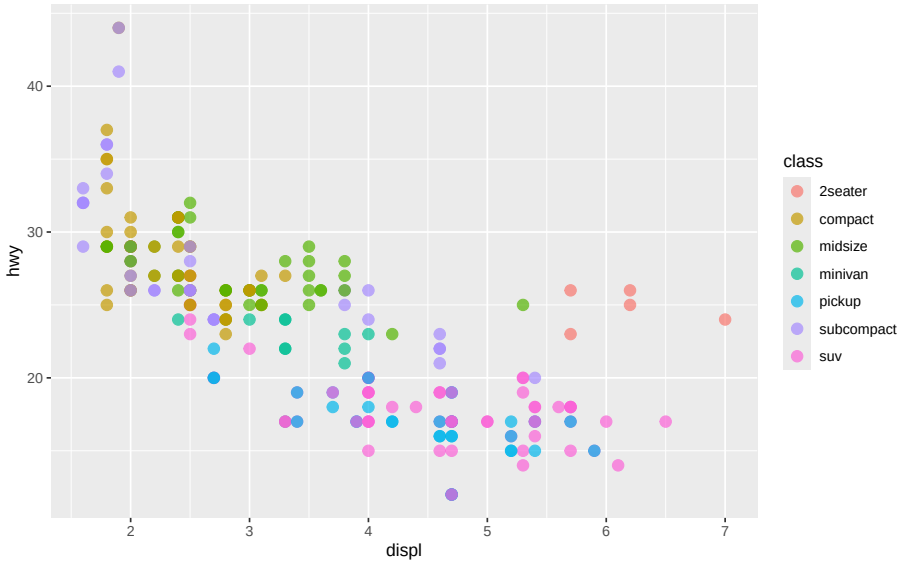
```
# Basic scatter plot
ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point()
```




```
# Add color by a variable
ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point()
```

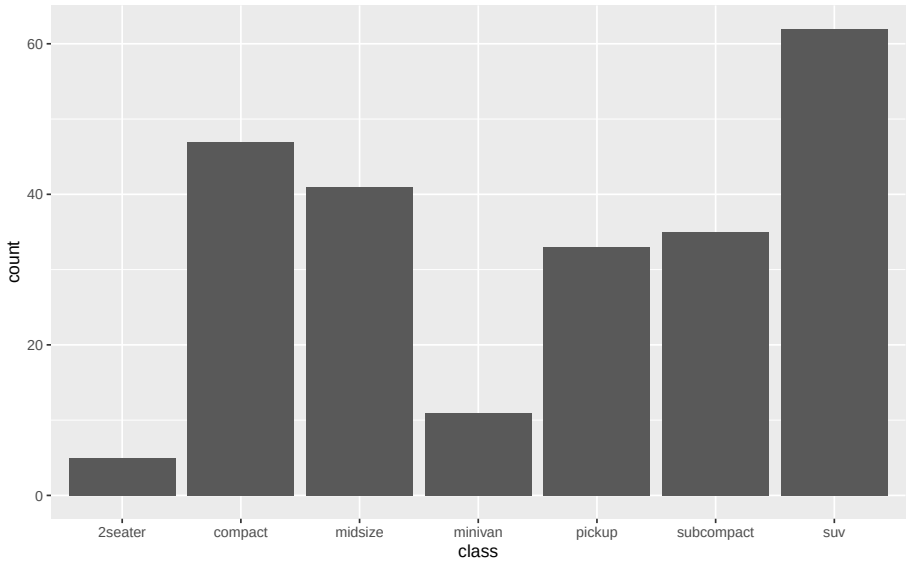



```
# Add size and transparency
ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point(size = 3, alpha = 0.7)
```



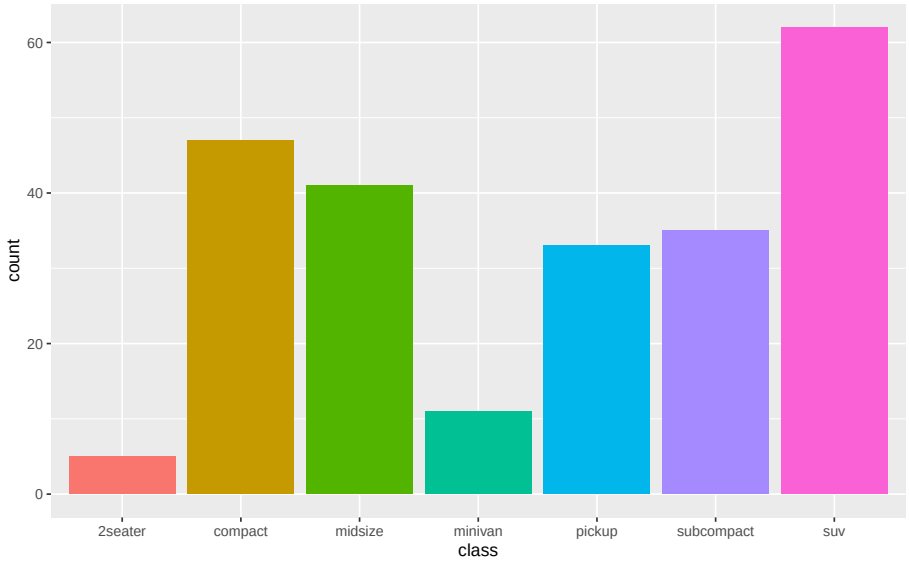
Bar Charts

```
# Count-based bar chart (the default)
ggplot(mpg, aes(x = class)) +
  geom_bar()
```

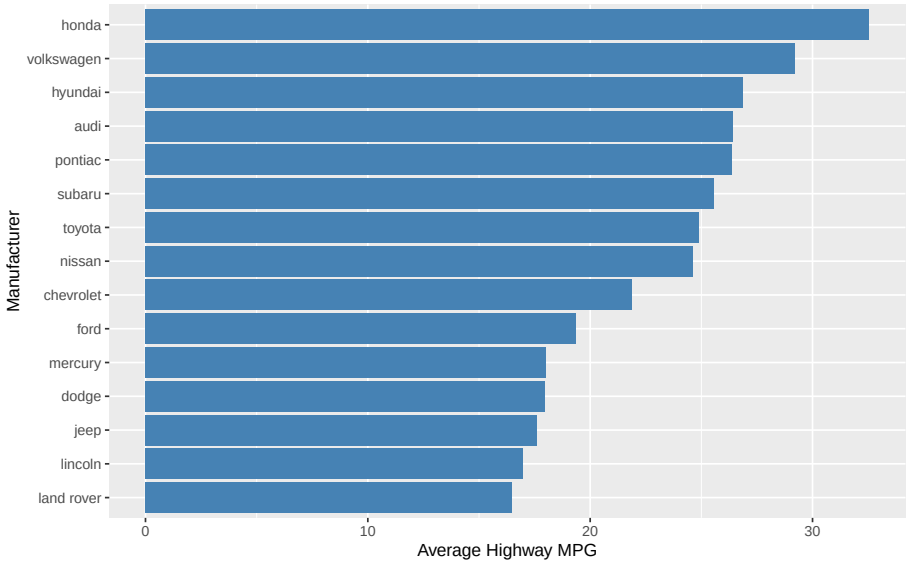



```
# Colored bars
ggplot(mpg, aes(x = class, fill = class)) +
```

```
  geom_bar() +
  theme(legend.position = "none")
```

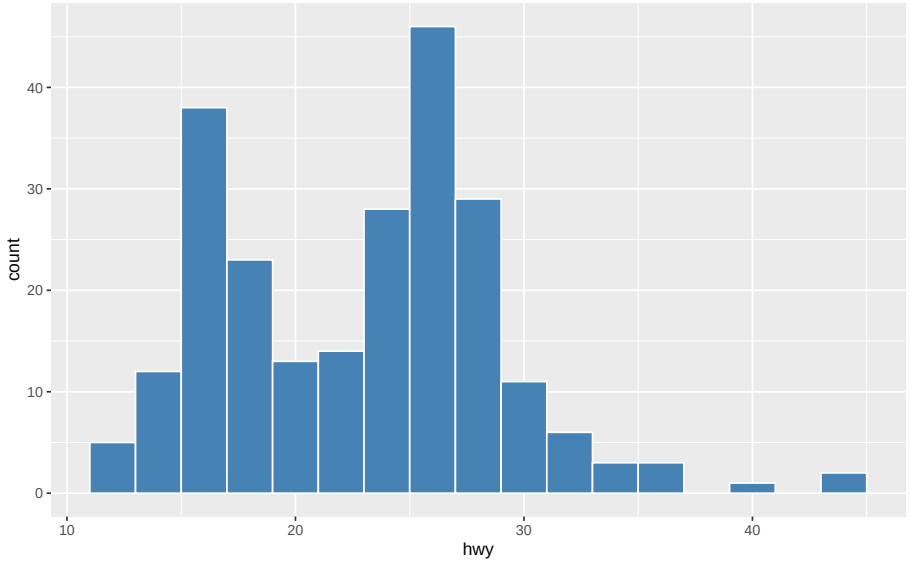



```
# Bar chart with a pre-calculated value
mpg |>
  group_by(manufacturer) |>
  summarize(avg_hwy = mean(hwy)) |>
  ggplot(aes(x = reorder(manufacturer, avg_hwy), y = avg_hwy)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(x = "Manufacturer", y = "Average Highway MPG")
```



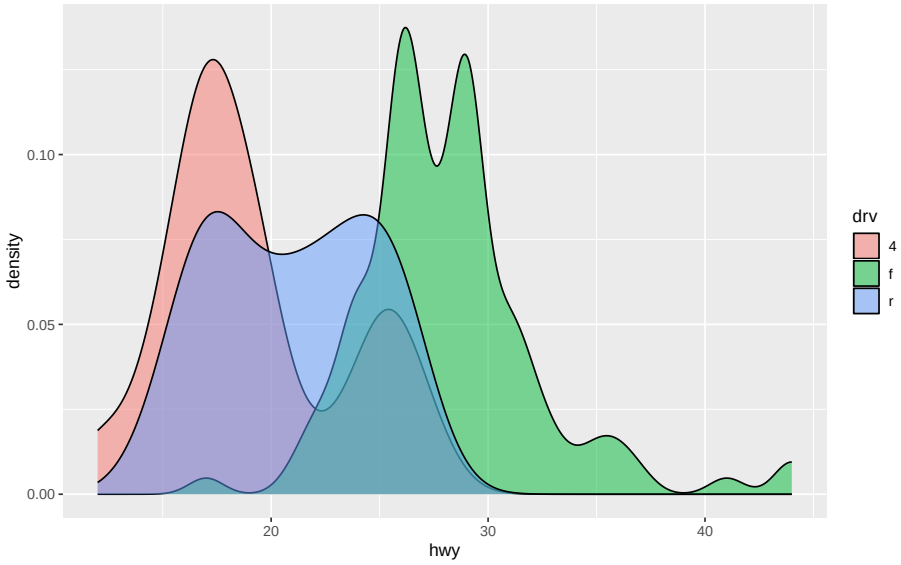
Histograms and Density Plots

```
# Histogram
ggplot(mpg, aes(x = hwy)) +
  geom_histogram(binwidth = 2, fill = "steelblue", color = "white")
```



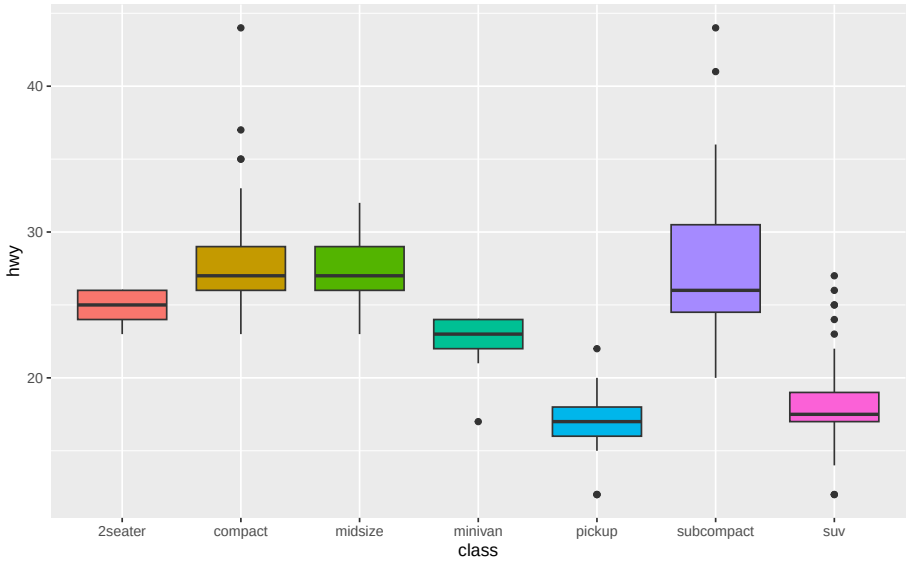

```
# Density plot
ggplot(mpg, aes(x = hwy, fill = drv)) +
```

```
  geom_density(alpha = 0.5)
```



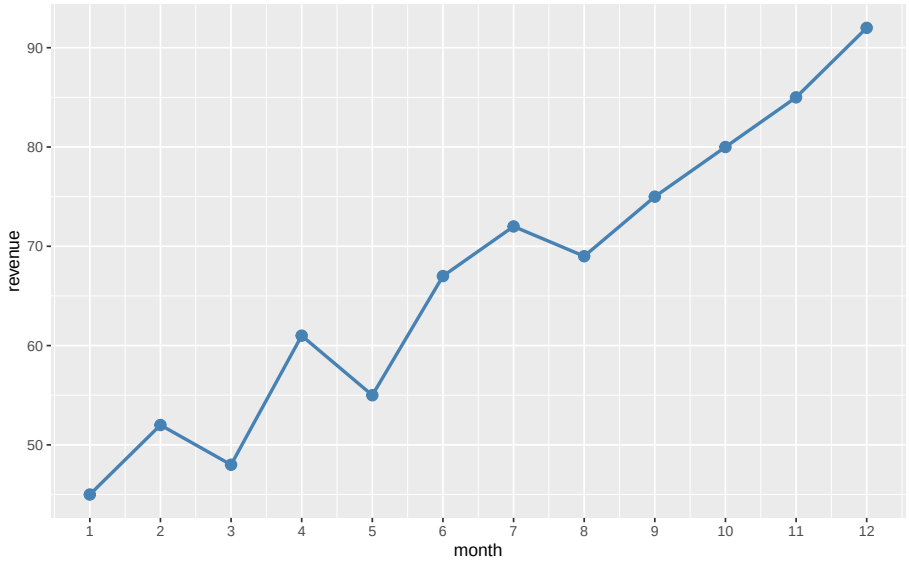
Box Plots

```
# Box plot comparing groups
ggplot(mpg, aes(x = class, y = hwy, fill = class)) +
  geom_boxplot() +
  theme(legend.position = "none")
```



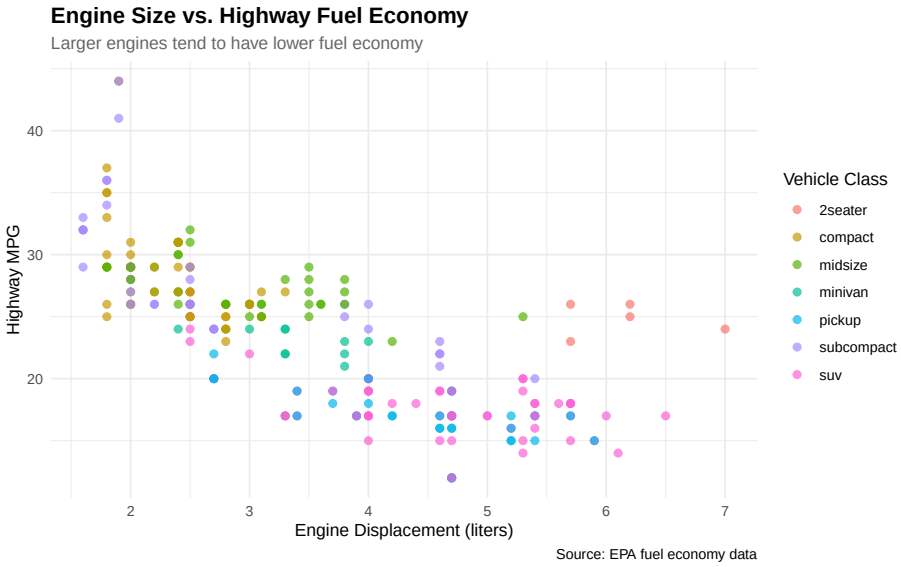
Line Charts

```
# Create some time series data
monthly_sales <- tibble(
  month = 1:12,
  revenue = c(45, 52, 48, 61, 55, 67, 72, 69, 75, 80, 85, 92)
)

ggplot(monthly_sales, aes(x = month, y = revenue)) +
  geom_line(color = "steelblue", linewidth = 1) +
  geom_point(color = "steelblue", size = 3) +
  scale_x_continuous(breaks = 1:12)
```



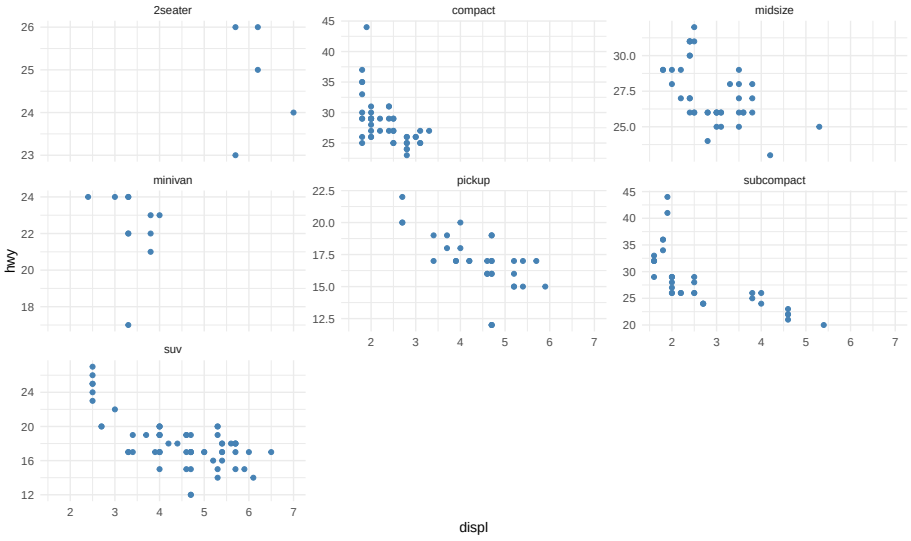
Adding Labels and Themes

```
ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point(size = 2, alpha = 0.7) +
  labs(
    title = "Engine Size vs. Highway Fuel Economy",
    subtitle = "Larger engines tend to have lower fuel economy",
    x = "Engine Displacement (liters)",
    y = "Highway MPG",
    color = "Vehicle Class",
    caption = "Source: EPA fuel economy data"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(color = "gray40")
  )
```



Facets — Small Multiples

```
# Split into panels by a variable
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(color = "steelblue") +
  facet_wrap(~ class, scales = "free_y") +
  theme_minimal()
```



Saving Plots

```
# Save the last plot you created
ggsave("my_plot.png", width = 8, height = 5, dpi = 300)

# Save a specific plot
my_plot <- ggplot(mpg, aes(x = displ, y = hwy)) + geom_point()
ggsave("scatter.pdf", plot = my_plot, width = 8, height = 5)
```

Addtional: R Packages (Three Common Live Places)

Not everything is on CRAN. Here's where to look:

Source	What's there	How to install
CRAN	~20,000 general-purpose packages	`install.packages("name")`
Bioconductor	~2,200 packages for genomics, genetics, bioinformatics	`BiocManager::install("name")`
GitHub	Development versions, niche tools, unreleased packages	`devtools::install_github("user/repo")`

CRAN — The Default

```
# Most packages you'll encounter
install.packages("tidyverse")
install.packages("ape")          # phylogenetics & population genetics

library(ape)
```

If `install.packages()` works, you're on CRAN. Nothing else needed.

Bioconductor — Genetics & Genomics

Many genetics and bioinformatics packages live on Bioconductor, not CRAN. You need `BiocManager` to install them.

```r
# Step 1: Install BiocManager (one time, from CRAN)
install.packages("BiocManager")

# Step 2: Use it to install Bioconductor packages
BiocManager::install("Biostrings")        # DNA/RNA sequence handling
BiocManager::install("VariantAnnotation") # VCF files & variant data
BiocManager::install("GenomicRanges")     # genomic intervals
BiocManager::install("DESeq2")            # differential gene expression

# Then load normally
library(Biostrings)
```

> **Tip**
>
> Finding Bioconductor packages Browse bioconductor.org/packages or search by topic (e.g., "Genetics", "Sequencing", "Variant"). Each package page has install instructions.

GitHub — devtools / remotes

Some packages aren't on CRAN or Bioconductor at all — they only exist on GitHub. Others have a development version on GitHub that's newer than the CRAN release.

```r
# Step 1: Install devtools or remotes (one time)
install.packages("devtools")
# or the lighter alternative:
install.packages("remotes")

# Step 2: Install from GitHub using "username/repository"
devtools::install_github("tidyverse/dplyr")          # dev version of dplyr
remotes::install_github("YuLab-SMU/ggtree")          # phylogenetic tree plots


# Then load normally
library(ggtree)
```

> **Note**
>
> `devtools` vs `remotes` `remotes` does just the installation part and is faster to install. `devtools` includes `remotes` plus tools for *building your own* packages. For just installing other people's packages, either works —

`remotes` is lighter.

Quick Decision Guide

```
Need a package?

  Try install.packages("name") first
     Works? → Done!

  "Package not available" error?
     Is it a bio/genetics package? → BiocManager::install("name")
     Is it on GitHub? → devtools::install_github("user/repo")

  Still can't find it?
      Search: google "R package [what you need]"
       or ask AI: "What R package does [task]?"
```

Common Genetics / Bioinformatics Packages

Package	Source	What it does
`ape`	CRAN	Phylogenetics, population genetics
`pegas`	CRAN	Population & evolutionary genetics
`adegenet`	CRAN	Multivariate genetics (PCA, DAPC)
`Biostrings`	Bioconductor	DNA/RNA/protein sequence manipulation
`GenomicRanges`	Bioconductor	Genomic intervals and annotations
`DESeq2`	Bioconductor	Differential gene expression (RNA-seq)
`VariantAnnotation`	Bioconductor	Read and annotate VCF files
`SNPRelate`	Bioconductor	SNP data analysis (PCA, relatedness)
`ggtree`	GitHub / Bioconductor	Phylogenetic tree visualization