Introduction to Base R

Reminder to Start Zoom Recording

Announcements & Reminders

- Attendance important & cameras on please
- Six Ways to Get Help & Course Collaboration Leaders
- Moodle & assignments debrief
- Joing the class late? Please reach out to me.

Base R: What Beginners Need to Know

What is Base R?

- **Base R** refers to the core set of functions, data types, and utilities included with the default R installation.
- It provides essential tools for working with vectors, data frames, lists, and matrices, as well as basic **data manipulation**, **statistics**, and **plotting** functions.

Alternatives to Base R

- **Tidyverse**: A collection of packages designed for more intuitive data manipulation and visualization. Syntax is often cleaner for larger data workflows. (**We will learn this next.**)
- **Data.table**: A faster alternative to Base R data frames, optimized for large datasets. (**Beyond scope of class.**)
- **Matrix-specific Packages**: For large numerical datasets, for efficient matrix operations. (**Beyond scope of class.**)

Why Learn Base R?

- Provides a **strong foundation**: Most R packages build upon the principles of Base R, so understanding it helps you with advanced techniques.
- **Lightweight**: Base R functions are fast and simple for smaller or less complex tasks.
- **Always available**: Base R is built-in, meaning you don't need to install any additional packages.

Some syntax and terminology (mostly a review)

- An R **object** can store different types of data, such as:
  - **Numbers** (e.g., `10`, `3.14`)
  - **Text** (e.g., `"Hello"`, `"R is fun!"`)
  - **Vectors** (e.g., a list of numbers or text)
  - **Data Frames** (e.g., tables of data)
- We assign values to objects using the <- operator or =.
  - Tip: <- is very much preferred, but = also works.

- R **functions** are commands that perform specific tasks in R.
  - A function takes *input* (called arguments) and returns *output*.
  - You can pass objects as inputs into functions to operate on them.
  - Some functions inspect objects by telling us info about it.

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

Overview of Data Frames in Base R

A data frame is a two-dimensional table-like structure that can hold **columns of different types** (e.g., numeric, character, and logical).

Each column in a data frame is a vector, and the length of each vector (number of rows) must be the same across all columns.

Built-in Data Frames in R

We use a R package in base R called `datasets` to access built-in toy data frames.

Viewing the Data Frame

Let's inspect the data frame `mtcars` by looking at the first 6 rows using the function `head()`:

```
                  mpg cyl disp  hp drat    wt  qsec vs am
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0
```

Let's look at the structure of the data frame using `str()`:

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
```

Column and Row Names

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"
[11] "carb"

 [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 71
 [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
 [7] "Duster 360"          "Merc 240D"           "Merc 230
[10] "Merc 280"            "Merc 280C"           "Merc 450
[13] "Merc 450SL"          "Merc 450SLC"         "Cadillac
[16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128
[19] "Honda Civic"         "Toyota Corolla"      "Toyota Co
[22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z
[25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche
[28] "Lotus Europa"        "Ford Pantera L"      "Ferrari
[31] "Maserati Bora"       "Volvo 142E"
```

Accessing Columns in Data Frames

Like with matrices, we can use `[]` to designate a column. But it is not a good coding practice to use numbers within the brackets. If our data frame gets rearranged, our code would be wrong. Instead, we use column names as follows:

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

```
[1] "data.frame"
```

Accessing Rows in Data Frames

We can also use brackets to get particular rows. Note here, we need to put the row name before a comma. And it returns a data.frame.

```
[1] "data.frame"
```

Subsetting Rows

We can also get multiple rows:

```
              mpg cyl disp  hp drat    wt  qsec vs am gear
Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4
```

More Advanced Row Subsetting

Let's get all the Mercedes without having to type them out.

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
[13]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
             mpg cyl  disp  hp drat   wt qsec vs am gear ca
Merc 240D   24.4   4 146.7  62 3.69 3.19 20.0  1  0    4
Merc 230    22.8   4 140.8  95 3.92 3.15 22.9  1  0    4
Merc 280    19.2   6 167.6 123 3.92 3.44 18.3  1  0    4
Merc 280C   17.8   6 167.6 123 3.92 3.44 18.9  1  0    4
Merc 450SE  16.4   8 275.8 180 3.07 4.07 17.4  0  0    3
Merc 450SL  17.3   8 275.8 180 3.07 3.73 17.6  0  0    3
Merc 450SLC 15.2   8 275.8 180 3.07 3.78 18.0  0  0    3
```

Using $ to Access Columns

We can also use “data.frame.name$” to refer to columns/variables.

```
[1] 20.09062
```

Creating a New Column

We can also use this approach to add columns to a data frame. In the example below, we create a new column kpg that converts miles per gallon (mpg) to kilometers per gallon.

```
                   mpg cyl disp  hp drat    wt  qsec vs am
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0
```

Reading and Writing Data Files

.RData (or .Rda) files

- .RData files can store multiple R objects. Sometimes data files are stored as .rds files, which only store a single object.
- After loading, the objects contained in the .RData file will be available in your R environment with their original names.

.csv files or .txt files

A common way to import external data into R is from a CSV file.
Use the read.csv() function.
For plain text files, use read.table() or readLines():

Tips

- File Paths: File paths can be absolute or relative.
  - Absolute paths specify the exact location on your computer.
  - Relative paths are relative to your current working directory in R.
  - **We set up R Studio so that it should be easy to use relative paths!**
- Working Directory:
  - Use `getwd()` to find your current working directory
  - Use `setwd("path/to/directory")` to change it.
  - **But we set up R Studio so that if you `.Rproj` is loaded, your current working directory is set.**
- Viewing Data: After reading a file, use head(data) to view the first few lines of your data frame.
- Handling Errors: If R can't read your file, check for typos in the file path and ensure the file format is correct.