R Overview: Learning to Speak R

2023-08-24

Basic Terminology

Two programs we'll use:

- `R` is a statistical programming language
- `RStudio` is an IDE (integrated development environment)

`RStudio` is *how* we'll work with `R`.

Base `R` and Packages

`R` has lots of functions included by default

- e.g., `mean()`, `lm()`, etc.
- These functions are known as "Base `R`"

In this class, we'll use several *packages* like `tidyverse`

- Packages are collections of functions that aren't in base `R`

Using packages in `R`

- Install them using the `install.packages()` function
- *Load* them using the `library()` function.

This tells `R`, "I want to use the functions in this package."

Scripts and Running Code in `R`

In this class, we'll general work with `R` scripts

- These files will have a `.R` file extension.
- Scripts are collections of `R` code.

To run code in an `R` script:

- Click on the line of code (or select it with your mouse)
- `Ctrl + Enter` (or `Apple logo / Cmd key + Enter` on Mac)

If you want to run several lines of code at one time,

- Highlight all of the lines you want to run, then run
- You can also use the `Run` button

Scripts and the Command Line

The `Console` is one of the four default panels in `Rstudio`

- You can run code by clicking next to the > symbol
- Convenient way to test a line of code
- Use it to access documentation using "?" - e.g, `?mean`

AN IMPORTANT NOTE

- When something's important, make sure its saved in a script!
- *Don't* rely on the command line (you'll lose your work)

Comments in your Code

An important part of well-written code is *comments*.

- Start a line with a hash tag - `#`.
- This tells `R`, "ignore this next line."

Documenting your code means explaining what you're doing

- Helps other people (and you, later!) understand your code

```
# This is a comment - R will ignore this line


2 + 2
```

```
## [1] 4
```

Basic Data Structures in R and the Assignment Operator

<- is the *assignment operator* - use it to create objects

- Type it using the hot key combination `Alt + -`
- Example code below:

```
# Create object named a storing the value 5...

a <- 5

# Now we can use a in equations, functions, etc.

a + 2
```

```
## [1] 7
```

Object Types in `R`

In `R`, we can create different *types* of objects, including:

- **Numeric**: stores a single value (like `a` stores 5 above)
- **Logical**: stores binary values `TRUE` or `FALSE`
- **Character**: also known as string objects, these stores strings of text (like "hello world")

Given an object, we can use "`is`" functions to check types

- These functions will return either `TRUE` or `FALSE`
- Example on next slide

Object Types in R and the is Function

```r
# Check if a object from above is numeric

is.numeric(a)
```

```
## [1] TRUE
```

```r
# Now we can create a character object and check it

a.string <- "123 Main St"

is.character(a.string)
```

```
## [1] TRUE
```

Vectors in R

Vectors are the basic “building blocks” of data in R

- Vectors are collections of items stored together
- To create vectors, we’ll use the c() function

```r
# Create a vector with 3 elements - note the commas!

b.vector <- c(1, 2, 3)

b.vector
```

```
## [1] 1 2 3
```

Dataframes in Base R

In Base R, data sets are stored as *data frames*

- Data frames are *collections* of vectors

```
# Start with vectors storing various object types

person.ID <- c(12, 24, 54, 65)
address   <- c("123 Main St", "274 Long St",
               "789 Right St", "467 Left St")
employed  <- c(TRUE, TRUE, FALSE, TRUE)
wage.inc  <- c(12500, 15750, 0, 14100)
```

Dataframes in Base R

```r
# Combine individual vectors into a data frame

data <- data.frame(person.ID, address,
                   employed, wage.inc)

head(data)
```

```
##   person.ID      address employed wage.inc
## 1        12  123 Main St     TRUE    12500
## 2        24  274 Long St     TRUE    15750
## 3        54 789 Right St    FALSE        0
## 4        65  467 Left St     TRUE    14100
```