

# R Stats Bootcamp

1.4 - Data objects

Megan Lewis

2024-11-28

# R stats bootcamp - Module 1

## Schedule:

- Session 1: An introduction and script workflow
- Session 2: R language
- Session 3: R functions
- Session 4: Data objects
- Session 5: Data frames
- Session 6: Data subsetting



# Session 4 objectives:

- Basic data types in R
- Data with factors
- `class()` and converting variables
- Vectors and Matrices
- Practice exercises



# Data Objects in R

## R space

Imagine you are floating in space in the R Global Environment and any data object you can see, you can call on by name to manipulate with functions. Let's call it R Space.

# Data Objects in R

## R space

Imagine you are floating in space in the R Global Environment and any data object you can see, you can call on by name to manipulate with functions. Let's call it R Space.

- Fundamental way to analyse data in R is to be able to manipulate it with code
- Need to be able to store data somewhere



# The Global Environment

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Contains the R script "script-1.R". The code includes comments like "## What: I Bootcamp Setup", "## Last edited: 2022-10-09", and "## 00 Look at the "iris" data". It also contains a call to `head(iris)` and a boxplot command.
- Global Environment:** Shows the "iris" dataset loaded, with 150 observations and 5 variables.
- Plots:** A boxplot titled "SepalLength" is displayed, comparing Sepal.Length across three iris species: setosa, versicolor, and virginica. The y-axis ranges from 4.5 to 8.0. The boxplot shows that virginica has the highest median Sepal.Length, followed by versicolor, and then setosa.
- Console:** Displays the R session history, including the execution of the script and the resulting output.

# The Global Environment

The screenshot shows the RStudio interface with the "Global Environment" tab selected in the top navigation bar. The main pane displays the "iris" dataset, which contains 150 observations and 5 variables: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. The "Sepal.Length" variable is of type "num" and contains values ranging from 4.3 to 7.9. The "Sepal.Width" variable is of type "num" and contains values ranging from 2.0 to 4.4. The "Petal.Length" variable is of type "num" and contains values ranging from 1.0 to 2.5. The "Petal.Width" variable is of type "num" and contains values ranging from 0.1 to 1.9. The "Species" variable is of type "Factor" with levels "setosa", "versicolor", and "virginica". The RStudio interface also includes tabs for Environment, History, Connections, Git, and Tutorial, as well as various toolbars and a search bar.

# Basic data types

- Concept of “data objects”
- Data you can use in the global environment
- Data Type
  - refers to the kind of data (*Important for computers*)
  - determined automatically by R (*Passive aggressive butler*)
- Can tweak data types
  - May need or want to

# Basic data types

```
1 # numeric vector
2 variable_1 <-
3   c(4, 5, 7, 6, 5, 4, 5, 6, 7, 10, 3, 4, 5, 6)
4
5 # logical vector
6 variable_2 <-
7   c(TRUE, TRUE, TRUE, FALSE)
8
9 # character vector
10 variable_3 <-
11   c("Peter Parker", "Bruce Wayne", "Groo the Wanderer")
```

# Data Types Demo



HARUG R Stats Bootcamp by Ed Harris

# Naming conventions for variables (1)

- Can contain letters, numbers and symbolic characters
- **MUST** begin with a letter
- **MUST** not contain spaces
  - Technically can, but difficult
- Forbidden characters: @, %, #, math operators etc.
- Case sensitive

# Naming conventions for variables (2)

- Should be human readable
- Should be consistent
- Should be short and concise
  - Can use a “Data Dictionary” to help remember variables if needed

# Naming Conventions in R Demo



HARUG R Stats Bootcamp by Ed Harris

# Data with `factor()`

- What if you want to analyse categorical data?

# Data with `factor()`

- What if you want to analyse categorical data?
- Sometimes referred to as a factor

# Data with `factor()`

Two types of factors:

- Non-ordered factors
  - i.e., names of cows, variety of apples etc.
- Ordered factors
  - i.e., dose levels (Control, Half, Full)

# class() and converting variables

```
1 # non-ordered factor
2 variety <- c("short", "short", "short",
3             "early", "early", "early",
4             "hybrid", "hybrid", "hybrid")
```

# `class()` and converting variables

```
1 # non-ordered factor
2 variety <- c("short", "short", "short",
3             "early", "early", "early",
4             "hybrid", "hybrid", "hybrid")
```

```
1 # Ordered factor
2 day <- c("Monday", "Monday",
3          "Tuesday", "Tuesday",
4          "Wednesday", "Wednesday",
5          "Thursday")
```

# Factor demo



HARUG R Stats Bootcamp by Ed Harris

# Vectors and Matrices

Ways of organizing data

- **Vector:** One dimension from 1 to i, `my_vec[i]`
- **Matrix:** Two dimensions, rows and columns, `my_mat[i, j]`
- **Array:** Three (or more) dimensions, `my_array[i, j, k]`

## 💡 Bracket notation

Each value in a vector has an address (commonly referred to as an indices)

`my_vec[i]` - accessing values from the “address” of where value lives

# Vectors have an address?!

# Vectors have an address?!

```
my_vector <- c( 2, 5, 3, 9, 4, 6 )
```

# Vectors have an address?!

```
my_vector <- c( 2, 5, 3, 9, 4, 6 )
```



# Vectors have an address?!

```
my_vector <- c( 2, 5, 3, 9, 4, 6 )
```



# Vectors have an address?!



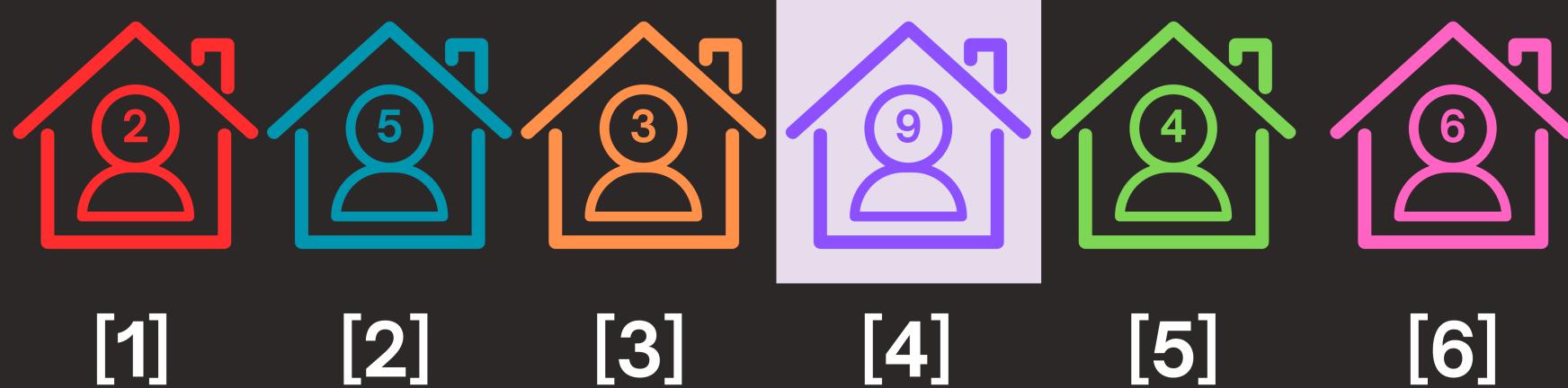
# Vectors have an address?!



# Vectors have an address?!



# Vectors have an address?!



# Vectors have an address?!



my\_vector[4] → 9

# Vectors have an address?!

```
my_vector_2 <- c( 4, 1, 5, 6, 3, 8 )
```

# Vectors have an address?!

```
my_vector_2 <- c( 4, 1, 5, 6, 3, 8 )
```



# Vectors have an address?!

```
my_vector <- c( 2, 5, 3, 9, 4, 6 )
```

```
my_vector_2 <- c( 4, 1, 5, 6, 3, 8 )
```



```
my_matrix <- matrix( c (vector1 , vector2),  
                      nrow = 2,  
                      byrow = TRUE)
```

# Vectors have an address?!

## Matrix

Street [1,]



Street [2,]



# Vectors have an address?!



# Vectors

- Vectors can contain any data type
- Only one data type at a time

```
1 # Yes
2 vec1 <- c(1,2,3,4,5,6)
3 vec2 <- c("Hello", " ", " ", "World", "!")
4
5 # No
6 vec3 <- c(1,2,3, "Hello", "World")
```

# Matrices

- May look similar to data set
- Only one type of data (i.e., numeric or categorical)
- Columns and rows assigned numbers by default

```
1 mat1 <- matrix(data = vec1,  
2                   ncol = 4,  
3                   byrow = FALSE)  
4  
5 mat1
```

# Vector and matrix demo



HARUG R Stats Bootcamp by Ed Harris

# Matrix challenge

- Challenge 1: Set the row names for Mat1 using the rownames() function

# Matrix challenge

- **Challenge 1:** Set the row names for Mat1 using the `rownames()` function
- **Challenge 2:** Make a matrix with 3 rows with the following vector,

```
vec2 <- c(2, 3, 5, 4, 5, 6, 7, 8, 9, 5, 3, 1),
```

- so that the first COLUMN contains numbers 2, 5, and 9, in the order for rows 1, 2, and 3 respectively ## Practice exercises

# Practice exercises



# Practice exercises 01

- Create a vector named `my_var1` that contains the following 6 integers: `3, 6, 12, 7, 5, 1`
- Create a second vector called `my_var2` that contains the following 2 integers: `2, 3`
- Evaluate the expression `my_var1 + my_var2`
- Explain the output in terms of R mechanics in your own words.

# Practice exercises 02

- Create a character vector with the names of the 12 months of the year.
- Convert the vector to a factor, with the month names in chronological order.
- Show your code.

# Practice exercises 03

- What is wrong with the following code? Describe, show the code, and justify a fix for the problem in your own words.

```
mymat <- matrix(data = c( 12,    23,    45,  
"34", "22", "31"))
```

# Practice exercises 04

- Use the `array()` function to make a  $2 \times 2 \times 3$  array to produce the following output:

```
, , 1          , , 3  
              [, 1]  [, 2]          [, 1]  [, 2]  
[1, ]  1      3          [1, ]  9      11  
[2, ]  2      4          [2, ]  10     12  
, , 2  
              [, 1]  [, 2]  
[1, ]  5      7  
[2, ]  6      8
```

# Practice exercises 05

- Show the code to make the following matrix:

	cat	dog
male	22	88
female	71	29

# Practice exercises 06

- Write a plausible practice question involving the use of the `matrix()` and `vector()` functions.