Objects & Indexing

Part 2

Plan for today

- Recap of **objects** & **classes**
- More sophisticated objects
    - **data.frames**
    - briefly mention others (lists, matrices, tibble)

Objects

- Objects are the nouns of programming languages
- They have names and they store something
    - single value or text string (character)
    - vector of objects
    - **data.frames**
    - **models**
    - *and more*

Classes

Objects can be of a different **classes**. *What type of information is stored in the object?* Some of the options are:

- **Numeric:** Decimals (3.141593)
- **Integer:** Natural numbers (0,1,2, etc.)
- **Character:** Text or string characters:
  - Always inside quotation marks
  - **Factors** (or categories)
- **Logical:** True or False:
  - No quotations
  - 2 possible values: `TRUE` or `FALSE`
  - or `T`/`F`...
  - but NOT `True`/`False`/`t`/`f`
- **Missing Value:** `NA`

Data.frames

Another object class is a **data.frame**. You can think of this as an Excel sheet. `empire` is an example of a data.frame. When you view it in `R`, it looks like this:

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Data.frames

Typically, this is what we want our data to look like. In `empire`, we have 6 column vectors. But they are *NOT* stored as 6 separate objects -- they are combined because they are all related to one another.

Data.frames are 2-dimensional

- Rows & columns
- Prettier spreadsheet

Data.frames

Every **row** has **6** pieces of data that are associated with one another...

Data.frames

Every **column** has **10** observations...

Indexing Data.frames

Data.frames can be indexed just like vectors.

Except: Data.frames have 2 dimensions!

Indexing Data.frames

`data.frame[rows, columns]`

What *should* we get if we typed `empire[1:6,5]`?

Indexing Data.frames

`data.frame[rows, columns]`

What *do* we get if we type `empire[1:6,5]`?

```
empire[1:6,5]
```

```
## [1] "Tatooine" "Tatooine" "Naboo"    "Tatooine" "Alderaan" "Stewjon"
```

Indexing Data.frames

If you want **all** of something, leave it blank.

All the rows of column 2

```
empire[,2]
```

```
##  [1] 172 167  96 202 150 182 228 180  66 183
```

All the columns of row 5

```
empire[5,]
```

```
##          name height mass    sex homeworld species
## 5 Leia Organa    150   49 female  Alderaan   Human
```

Finding Your Data

Sometimes it's easy enough to remember the row index or column index that you want. But often, we forget!

One of the benefits of a data.frame is that you can access a column by using the column name.

```
data.frame$column.name
```

Finding Your Data

```
empire$height
```

```
##  [1] 172 167  96 202 150 182 228 180  66 183
```

- *Note the tab-complete!*

Other object types

- Matrix
- Tibble
- List

Matrix

- Very similar to data.frame
- No column names
- No real reason to use matrices
- Can convert to data.frame easily

```
# Make a matrix

testMatrix <- matrix(data =  1:12, nrow = 4, ncol = 3)
testMatrix
```

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

```
# Try to access "V1"
testMatrix$V1
```

```
## Error in testMatrix$V1: $ operator is invalid for atomic vectors
```

Matrix

```
# Convert to data.frame

testDataFrame <- as.data.frame(testMatrix)
testDataFrame
```

```
##   V1 V2 V3
## 1  1  5  9
## 2  2  6 10
## 3  3  7 11
## 4  4  8 12
```

```
# Now try to access column 2 using the "V2" heading
testDataFrame$V2
```

```
## [1] 5 6 7 8
```

Tibble

- Even more similar to a data.frame than matrices are!
- It works particularly well with a suite of packages called the `tidyverse`
- If you use `class()` on a tibble, it might show up as `tbl_df`
- At this point, for our purposes, there is not notable difference between a tibble and data.frame.

List

- Contain elements of different types (e.g., numbers, strings, vectors, data.frames, matrices, and more)
- If you store a statistical model as an object, it will likely be in a list format
  - Besides dealing with models, we will (for the most part) not be dealing with lists
  - But they can be SUPER useful
  - Ex: You have 2 large data.frames that have the same variables, but data were collected on different groups (e.g., patients vs. controls). You want to perform the same actions on both datasets. You can store these as a list, and run the same analysis on each, rather than copying/pasting code.

Indexing Lists

- For the most part, you can index lists the same way you would a vector
- *(For these examples, let's only look at the first 3 rows of `empire`)*

```
exampleList <- list("hello", empire[1:3,], c(2:12))

# To get the first element (the word "hello")
exampleList[1]
```

```
## [[1]]
## [1] "hello"
```

Indexing Lists

- For the most part, you can index lists the same way you would a vector
- *(For these examples, let's only look at the first 3 rows of `empire`)*

```
exampleList <- list("hello", empire[1:3,], c(2:12))

# To get the second element (the `empire` data.frame)
exampleList[2]
```

```
## [[1]]
##              name height mass  sex homeworld species
## 1 Luke Skywalker    172   77 male  Tatooine   Human
## 2           C-3PO    167   75 none  Tatooine   Droid
## 3           R2-D2     96   32 none     Naboo   Droid
```

Indexing Lists

- For the most part, you can index lists the same way you would a vector
- *(For these examples, let's only look at the first 3 rows of `empire`)*

```
exampleList <- list("hello", empire[1:3,], c(2:12))

# To get the third element (the vector of numbers 2 through 12)
exampleList[3]
```

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

Indexing Lists

- **BUT!** Lists have the equivalent of an empty book page that says "Chapter 2" before getting to the actual chapter.
- In order to open the chapter, we use double brackets `[[ ]]`

```
# Single brackets
exampleList[2]
```

```
## [[1]]
##              name height mass  sex homeworld species
## 1 Luke Skywalker    172   77 male  Tatooine   Human
## 2          C-3PO    167   75 none  Tatooine   Droid
## 3          R2-D2     96   32 none     Naboo   Droid
```

```
# Single brackets doesn't get you "in"
exampleList[2]$mass
```

```
## NULL
```

Indexing Lists

- **BUT!** Lists have the equivalent of an empty book page that says "Chapter 2" before getting to the actual chapter.
- In order to open the chapter, we use double brackets `[[ ]]`

```
# Single brackets
exampleList[2]
```

```
## [[1]]
##              name height mass  sex homeworld species
## 1 Luke Skywalker    172   77 male  Tatooine   Human
## 2          C-3PO    167   75 none  Tatooine   Droid
## 3          R2-D2     96   32 none     Naboo   Droid
```

```
# Double brackets
exampleList[[2]]
```

```
##              name height mass  sex homeworld species
## 1 Luke Skywalker    172   77 male  Tatooine   Human
## 2          C-3PO    167   75 none  Tatooine   Droid
## 3          R2-D2     96   32 none     Naboo   Droid
```

Indexing Lists

- **BUT!** Lists have the equivalent of an empty book page that says "Chapter 2" before getting to the actual chapter.
- In order to open the chapter, we use double brackets `[[ ]]`

```
# Double brackets
exampleList[[2]]
```

```
##             name height mass  sex homeworld species
## 1 Luke Skywalker    172   77 male  Tatooine   Human
## 2          C-3PO    167   75 none  Tatooine   Droid
## 3          R2-D2     96   32 none     Naboo   Droid
```

```
# Double brackets gets you "in"
exampleList[[2]]$mass
```

```
## [1] 77 75 32
```