Objects & Classes

Part 1

Plan for today

- What is an **object**?
- **Assignment**
- More than 1 thing with **vectors**
- Object **classes**

What do we *want*?

We *want* our data to look something like this...

name	height	mass	sex	homeworld	species
Luke Skywalker	172	77.0	male	Tatooine	Human
C-3PO	167	75.0	none	Tatooine	Droid
R2-D2	96	32.0	none	Naboo	Droid
Darth Vader	202	136.0	male	Tatooine	Human
Leia Organa	150	49.0	female	Alderaan	Human
Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
Chewbacca	228	112.0	male	Kashyyyk	Wookiee
Han Solo	180	80.0	male	Corellia	Human
Yoda	66	17.0	male	NA	Yoda's species
Boba Fett	183	78.2	male	Kamino	Human

What do we *want*?

What R sees...

```
## # A tibble: 10 x 6
##    name           height  mass sex    homeworld species
##    <chr>           <int> <dbl> <chr>  <chr>     <fct>
##  1 Luke Skywalker    172  77   male   Tatooine  Human
##  2 C-3PO             167  75   none   Tatooine  Droid
##  3 R2-D2              96  32   none   Naboo     Droid
##  4 Darth Vader       202 136   male   Tatooine  Human
##  5 Leia Organa       150  49   female Alderaan  Human
##  6 Obi-Wan Kenobi    182  77   male   Stewjon   Human
##  7 Chewbacca         228 112   male   Kashyyyk  Wookiee
##  8 Han Solo          180  80   male   Corellia  Human
##  9 Yoda               66  17   male   <NA>      Yoda's species
## 10 Boba Fett         183  78.2 male   Kamino    Human
```

How do we get there?

Objects

- A basic concept in (statistical) programming is called an **object**
- An **object** allows you to store a value or a thing:

An object can be...

An object can be...

name	height	mass	sex	homeworld	species
Luke Skywalker	172	77	male	Tatooine	Human
C-3PO	167	75	none	Tatooine	Droid
R2-D2	96	32	none	Naboo	Droid
Darth Vader	202	136	male	Tatooine	Human
Leia Organa	150	49	female	Alderaan	Human
Obi-Wan Kenobi	182	77	male	Stewjon	Human
Chewbacca	228	112	male	Kashyyyk	Wookiee
Han Solo	180	80	male	Corellia	Human
Yoda	66	17	male	NA	Yoda's species
Boba Fett	183	78.2	male	Kamino	Human

An object can be...

An object can be...

Important:

- Objects have *names*
- We are going to refer to objects by their *names*
- Since they have names, we can **store** objects and use them later

Storing data in objects

If you want to use an object later on (you do!), you have to name it.

This is called **assignment** or **assigning** a name to an object

In takes the form of:

```
nameOfMyObject <- objectToStore
```

What's the point of storing objects?

Remembering things sucks! Let R hold on to all the stuff you don't want to remember or write down right away.

Let's do an example with a series of math equations:

$$y = 17 * 8$$

$$z = \frac{y}{3}$$

How do we solve this?:

1. Solve for y, which is 136. Either remember 136 or write the number down.
2. Plug it in in the second equation, so that you have 136 divided by 3.

Let's do this with R code!

Let's do an example with a series of math equations:

$$y = 17 * 8$$

$$z = \frac{y}{3}$$

With code:

```
y <- 17*8 # first, solve for y
z <- y/3 # now, solve for z
```

We didn't even need to know that `17*8` is `136`. We stored the value of `136` as an object with the name `y`.

Then, we could tell R to simply use the name `y` anytime we wanted to refer to the number `136`

Who cares?

Remembering a single number seems a little ridiculous. But remember, an object in R can really be anything. Some objects you definitely might want to store for later:

- A data set like `empire`
- A correlation coefficient
- The output of a linear regression model
- p-values and other statistics
- The mean of a variable, so you can subtract the mean from every individual's score
- and lots, lots more!

If you do not assign a name to an object, R will not remember it!

Example:

```
17*8
```

```
## [1] 136
```

```
y/3
```

```
## Error in eval(expr, envir, enclos): object 'y' not found
```

The error message `object 'y' not found` is very common!

R cannot perform the operation because you never told it to remember `17*8`

One type of object: Vectors

A group of objects is called a **vector**

Vectors are *ONE-DIMENSIONAL*. You can think of this as either a row...

One type of object: Vectors

A group of objects is called a **vector**

Vectors are *ONE-DIMENSIONAL*. You can think of this as either a row... ... or a column

Making Vectors

In your R code, you will type `c()` in order to create a vector

The `c` stands for *"combine"* or *"concatenate"*

Some examples:

```
subjectID <- c("Subject 1", "Subject 2", "Subject 3", "Subject 4", "Subj
passedStats <- c(TRUE, FALSE, FALSE, TRUE, TRUE)
favoriteNumbers <- c(7, 3, 6, 10, 100)
countries <- c(0, 3, 10, 1, 8)
```

```
## [1] "Subject 1" "Subject 2" "Subject 3" "Subject 4" "Subject 5"
```

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

```
## [1]   7   3   6  10 100
```

```
## [1]  0  3 10  1  8
```

Vectors

Because these items are grouped together, you can do something to them all at once!

Let's say these 5 people all went on a trip together, and they visited 2 countries. We can add `2` to the entire vector, rather than each individual number:

```
countries + 2
```

```
## [1]  2  5 12  3 10
```

Basic data classes

Objects can be of a different `class`. You can think of it more as *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`
- **Missing Value:** `NA`

Basic data classes

To check what data class your object is, you can type `class()`.

```
class(subjectID)
```

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

```
class(passedStats)
```

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

```
class(countries)
```

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

```
class(empire$species)
```

```
## [1] "factor"
```

Pro Tip #1: Special Values

RStudio will change the color of the words you type so that as you code, you can quickly see what you're dealing with.

Pro Tip #1: Special Values

Character objects are red (or green), and use quotation marks:

```
subjectID <- c("Subject 1", "Subject 2", "Subject 3", "Subject 4", "Subj
```

Numeric objects are green (or blue) and do *NOT* use quotation marks:

```
favoriteNumbers <- c(7, 3, 6, 10, 100)
```

Exception #1: `NA`, without quotation marks, is recognized as "missing" by R.

```
animals <- c("cow", "dog", NA, "chicken")
```

Exception #2: `TRUE` and `FALSE` does *not* require quotation marks. Must be either ALL CAPS or just the first letter capitalized (`T` or `F`).

```
variable <- c(T, FALSE, F, TRUE) # this line is correct
variable <- c(t, false, f, true) # this line is incorrect
```

Back to data classes

When you combine objects, the new object will have the class of the **least specific** object. For example:

```
numbers <- c(5, 6, 7, "eight", 9)
class(numbers)
```

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

All numbers could theoretially be wrapped in quotes and considered text. But there is no way for the computer to understand that the character string "eight" actually refers to 8. So the `character` class is less specific than the `numeric` class.

Pro Tip #2: Naming objects

- An object name can *never **start*** with a number, like `3myObject <- 7`
- You can include underscores `_` and periods `.` in object names, like `my_object <- "hi"` or `correlation.2 <- .57`
- RStudio allows for tab-complete. Start typing in an object name, and it should appear! Once you see it, either hit tab or enter on your keyboard, and it will fill in the object name for you
    - This means you should name your objects with clear, meaningful names!
    - It does not matter how long the name is
- Use capitalization to your benefit, like camel case 🐫 `myObject` or `patientsVsControls`
- Names should be human & computer readable

Recap

1. **Objects** are things, with names -- use the names!
2. Single values, or a group of values like **vectors**
3. The stuff within objects can belong to different data types or **data classes**
4. A lot of the error messages you'll get will relate to these!
5. Next up will be accessing our objects!