Preliminaries in R

Object classes and vectors

What are object classes?

Objects can be structured in different ways, in terms of how they "hold" data.

These difference structures are called **object classes**.

One class of objects can be a subtype of a more general object class.

What are object classes?

Today, we'll look at two key object classes for working with data in R:

1. Vectors
2. Dataframes (tibbles)

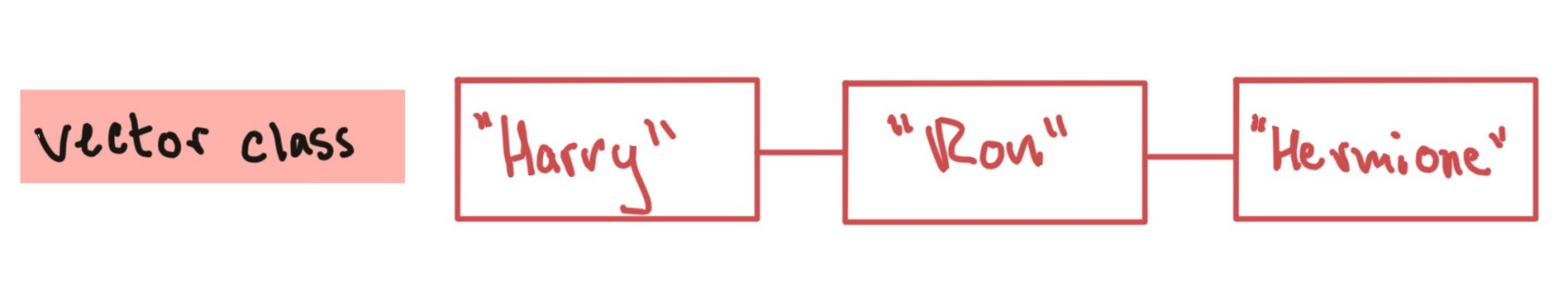
What are object classes?

For these two object classes (vectors and dataframes), we'll look at:

1. How that class is structured
2. How to make a new object with that class
3. How to extract values from objects with that class

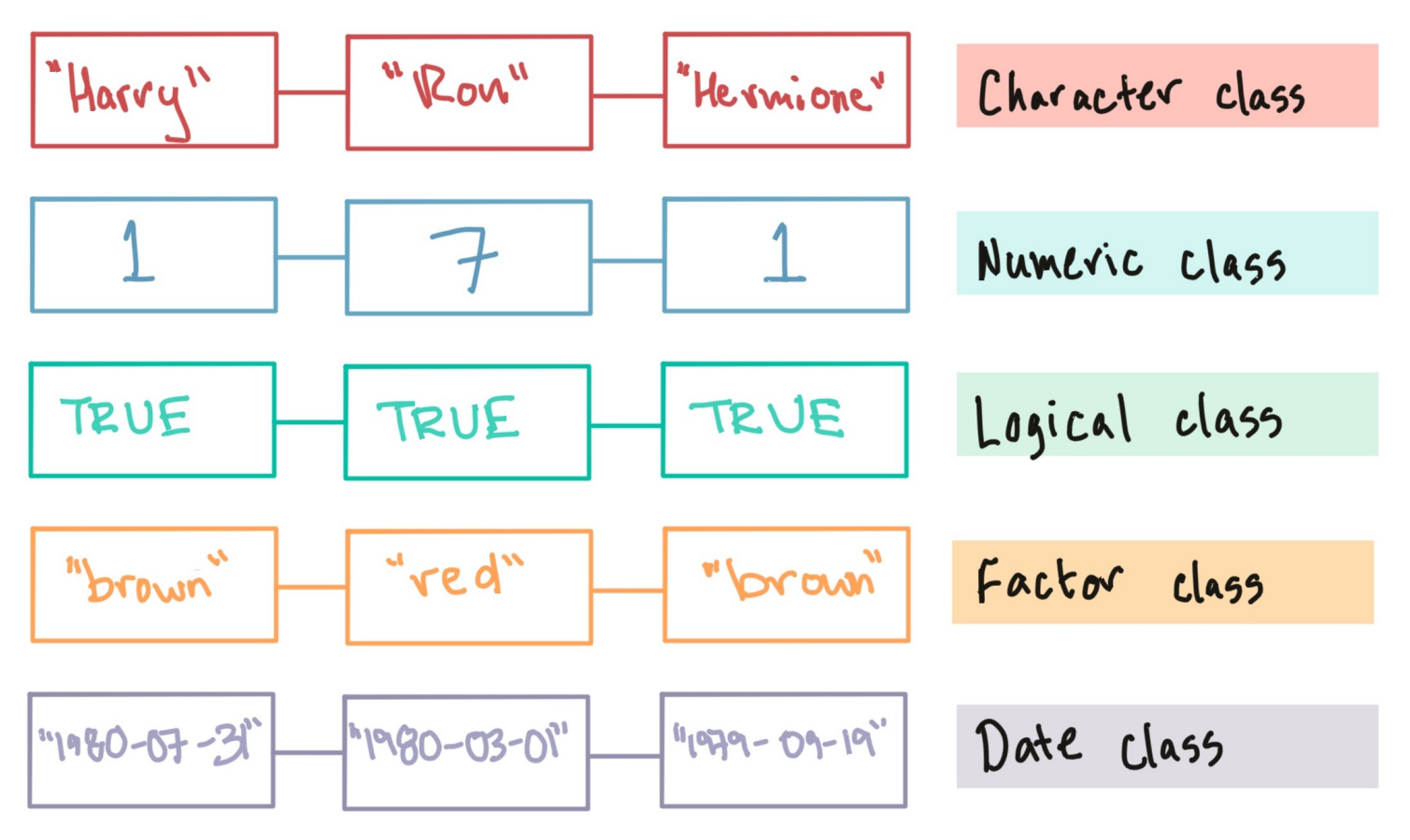
In later classes, we'll spend a lot of time learning how to do other things with objects from these two classes, plus learn some other classes.

Structure of vector class



A **vector** is an object class where the object is made of a string of values.

Structure of vector class



All the values in a vector must be of the same **type** (e.g., all numbers, all characters). There are different **classes** of vectors depending on the type of data they store.

Creating vectors

To create a vector object, you can use the **concatenate** function, `c`.

For example, to create a vector with the names of the three main characters in *Harry Potter*, use the R expression:

```
c("Harry", "Ron", "Hermione")
```

```
## [1] "Harry"    "Ron"      "Hermione"
```

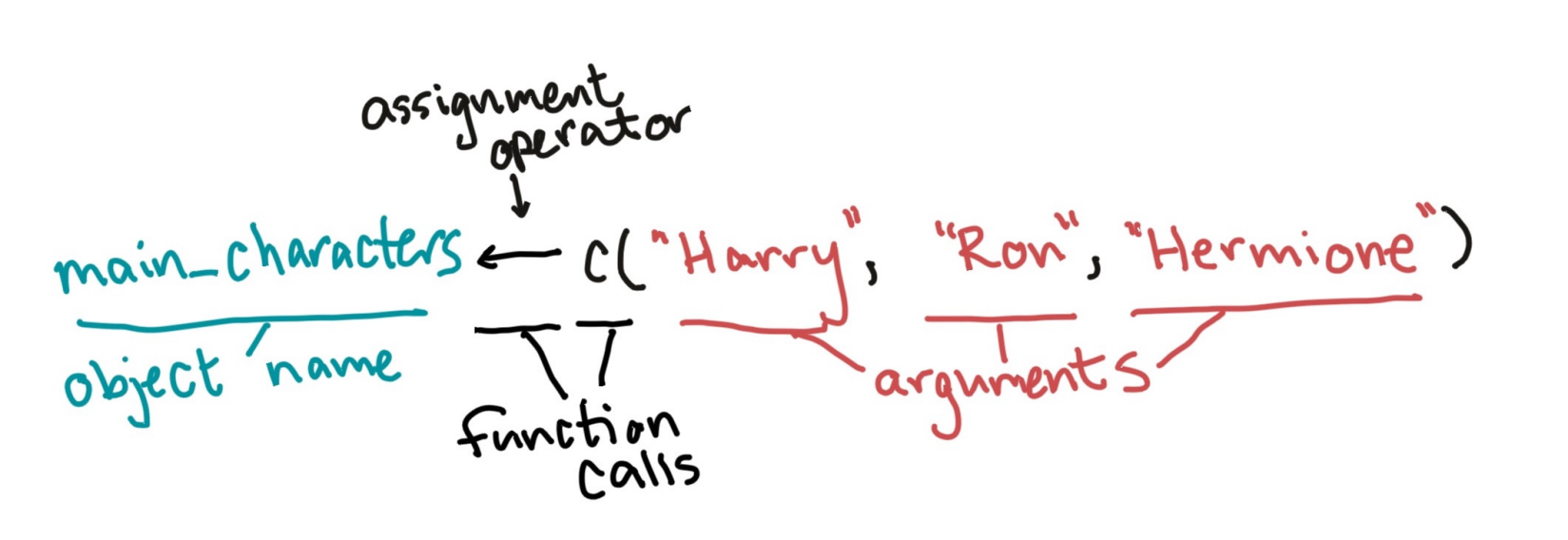
Creating vectors

If you want to use that object later, you can assign it an object name in the expression:

```
main_characters <- c("Harry", "Ron", "Hermione")
print(x = main_characters)
```

```
## [1] "Harry"    "Ron"      "Hermione"
```

Creating vectors



This **assignment expression** follows the structure we covered earlier for function calls and assignment expressions.

Creating vectors

Typically, when you use the c function, you'll be creating a numeric, character, or logical vector. More complex classes (like factors and dates) require a bit more work to create "from scratch".

- For character vectors, use quotation marks around each element.

```
main_characters <- c("Harry", "Ron", "Hermione")
```

- For numeric, don't use quotation marks.

```
n_kids <- c(1, 7, 1)
```

Vectors

You can use the `class` function to figure out the class of a vector.

```
class(x = main_characters)
```

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

```
class(x = n_kids)
```

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

Vectors

If you create a vector with a mix of classes, R will default the whole vector to the most generic class:

```
mixed_classes <- c(1, 3, "five")
mixed_classes
```

```
## [1] "1"    "3"    "five"
```

Vectors

The *length* of the vector is how many values it has. The `main_characters` vector includes three values ("Harry", "Ron", and "Hermione"), so it has a length of 3.

You can use the `length` function to figure out how long a vector is:

```
length(x = main_characters)
```

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

Extracting values from vectors

You can pull out certain values by using indexing (`[...]`) to identify the locations you want to get. For example, to get the second value from the `main_characters` vector, you can call:

```
main_characters[2] # Get the second value
```

```
## [1] "Ron"
```

Extracting values from vectors

You can also extract more than one value from a vector, by giving a vector of positions as the input in the square brackets.

For example, to get the first and third values from the `main_characters` vector, you can call:

```
main_characters[c(1, 3)] # Get first and third values
```

```
## [1] "Harry"    "Hermione"
```

Extracting values from vectors

There is an R operator that's very helpful with this. The `:` operator will create a sequence of values:

```
1:3
```

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

Therefore, to get the first three values from the `main_characters` vector, you can call:

```
main_characters[1:3]
```

```
## [1] "Harry"    "Ron"      "Hermione"
```