Data\_types

Martin Andrzejak

November 2, 2021

In order to understand R in general, it is useful to understand that data comes in many different types. The easiest types in R are probably

1. Numeric.

Every number is a numeric value, but not every number is the same type of number. You're maybe surprised on how many types of numbers there are.

1.1. Integer:the easiest case, an integer for example is 1 or 2 or 123. They are numbers without decimal numbers. This type is often used for computer and calculations. A lot of glms (Poisson distribution) can only consist of integers. Everything that you can count will be an integer. Computers in general like integers, as they have the exact number. It is possible to store those as integer, instead of numeri.

1.2. floats: Every number with decimal numbers. 1.2, 1.4, 1234.5. Floats can be as long as you want, just note that the computer will always assign an end to a float. For example, 1.2343564 could be going on eternal but the memory you have is limited. A lot of data sets (for example temperature in CHELSA) multiply their values by 10 (or anything that makes a integer) to save memory. R will store those simply as numeric. Note that if you try to store something like 2.4 as an integer it will always be stored as (in this example 2) as a decimal number can not be a integer.

1.3 complex numbers: You will encounter those when dealing with demography data. They are not really one number but a continuous number. They are usually written like a+b\*i. Very simplified one could say every complex number consists of real numbers.

## numeric  
class(2)

## [1] "numeric"

class(2.4)

## [1] "numeric"

## create integer, also from a float  
class(as.integer(2))

## [1] "integer"

class(as.integer(2.4))

## [1] "integer"

## a complex number  
as.complex(2)

## [1] 2+0i

class(as.complex(2))

## [1] "complex"

2. Character.

Besides numeric we also can have characters in R. Characters are all not numeric values meaning: letters, words, sentences. To be correct: A character is usually just one letter and everything consisting of more than one letter would be a string. However, R does not differ between strings and characters. We can also make numbers into characters by putting it in quotes: "8"

class("L")

## [1] "character"

class("I am a string, but in R I am classified as character")

## [1] "character"

class("8")

## [1] "character"

1. Factor.

Now another data class can be a factor, sometimes it happens that you read data and a column in a data frame is classified as factor. That sometimes happens when you have numeric and characters both in one column, and you didn't tell R that it should treat strings always as characters and never as factors (which you can do in global options or in the read function). Factors themselves can be really useful but can also harm your statistics quite badly. Factors always have a numeric value behind the actual value of your data. For example, let’s say you have two words: dead and alive. Now if you translate those in factors each word gets assigned a level coded by a number. So, if you call dead for example R will print dead but actually calls a 1 and for alive it would call a 2 but print alive. Both words are stored as a numeric value. That can also be true for numbers which are stored as factor. So if a 10 is stored as factor it doesn't mean the computer thinks it is a number 10, but rather as a “category” 10.

## lets create a factor  
example <- factor(c("dead", "alive"), levels = c("dead", "alive"))  
example

## [1] dead alive  
## Levels: dead alive

## if we now translate it into numeric we see how R stored those  
as.numeric(example)

## [1] 1 2

data.frame(factor = example, how\_stored = as.numeric(example))

## factor how\_stored  
## 1 dead 1  
## 2 alive 2

## now an example with numeric values  
example2 <- factor(c(10, 888), levels = c(888, 10))  
  
as.numeric(example2)

## [1] 2 1

data.frame(factor = example2, how\_stored = as.numeric(example2))

## factor how\_stored  
## 1 10 2  
## 2 888 1

Now factors can be really useful to order plots or your data in a specific way you must be careful when dealing with them as for models you run. R will always use the internal value assigned to the factor.

It is always important to know which class your data is as, some classes are required for certain statistical methods. A lm with a normal distribution just requires that your values are continuous and numeric, while Poisson distribution models require integer values. Always check your data before you do something with it.