Regular Expressions

Julian Gerez

March 1, 2019

Introduction to regular expressions

Have you ever wanted to merge two datasets only to hopelessly learn that the `id` variable in each of the datasets is mismatched (e.g. the `id`s in the first dataset are looking something like "Atlántico", "Bolívar", "Boyacá", while the `id` in the second datasets are "Atlantico", "Bolivar", "Boyaca")? Have you ever used webscraping only to find that a lot of the text you just extracted is just full of (*regularly* repeating) nonsense? Have you ever wanted to read in all the files of a certain extension into a folder using R? All of these desires and more can be fulfilled with regular expressions!

So what are regular expressions, even?

If you've ever worked with a big dataset in Stata you might have had several variables each starting with the same name, say `oil_exp1994`, `oil_exp1995`, `oil_exp1996` and so on. You might have learned (like me, after having tediously typed out many variable names in the past) that you can use `oil_exp*` to refer to *any* variable that starts with `oil_exp` and then has as many characters afterward as needed. Though this is a very basic example, this is *kind of* like a regular expression.

A regular expression, or regex for short, is a special text string for describing a search pattern in text.

Here is a more complex regular expression than that single asterisk:

```
[A-z]*\\.*? +[0-9]{1,2}-[0-9]{1,2}
```

No, I did not just randomly mash on my keyboard: I can assure you that bit of code has meaning, and in this workshop, you're going to learn how to interpret it!

This workshop will walk you toward understanding of *regex*, first going through how they work and function, and then using some applied examples. The tutorial is structured in `R`, and does pressupose some basic `R` knowledge, but the good news is that most of what you'll learn today is applicable for any software that has compatibility with regex (the syntax slightly varies from one platform to another), like Java, Python, Perl, Ruby, whatever you like. The biggest differences are going to be that we'll be learning functions in `R` that perform certain tasks, and the names for those will almost certainly be different in other softwares. *What's nice is that the underlying logic of regex will be the same.*

This workshop is heavily based off of Chapter 14 in **R for Data Science** by Hadley Wickham, but deviates at many different points. An answer key, as well as the data specified in the "Real World Examples" section, can be found on my GitHub at https://github.com/JulianEGerez/RegEx_Workshop.

A simple example where regex might be useful

You may also have heard of string manipulation functions and are wondering if they are the same as regex. Not quite, though the two are complementary. String manipulation are also used to extract information from text variables, but are often much more simple than regular expressions. For example, suppose you have a list of email addresses like so:

```
emails <- c("aaa@gmail.com", "bbb@gmail.com", "ccc@gmail.com", "abc@gmail.com")
```

Say you want to get (the technical term here is *extract*) the usernames from the emails. Pretend like this is a much longer list and it doesn't make sense to write it out by hand. We can use the `substr` command in `R` to do so very easily:

```
substr(emails, 1, 3)
```

```
## [1] "aaa" "bbb" "ccc" "abc"
```

This commands works as follows: `substr(x, start, stop)`, where `x` is a vector of strings, `start` is an integer and represents the starting value of your "request", and `stop` likewise represents the last value. If we switch the `stop` value in the code above to 2, what will happen? The `start` value?

So it turns out we actually want the domain names of the emails, not the usernames. Write some code that will extract the domain name (everything after the @ sign):

```
# insert your code here
```

Well done! But now I'm going to throw a wrench into your plan. What if our list of emails looks like this?

```
emails <- c("lionel@gmail.com", "sergio@wikipedia.org", "gonzalo@columbia.edu", "angel@gmail.com")
```

You can quickly see how simple string manipulation commands won't get us very far. All hope is not lost though! This is exactly what regular expressions are for. Come to the workshop to learn more!