Lecture 11: Strings & Regex in R

Economía Laboral

Junghanss, Juan Cruz

Universidad del CEMA

2nd Semester 2022

Contents

Today's lecture content:

- Strings Basics
- Regex Basics
- Cómputo de informalidad para TP Nº3

Strings

A lot of data for (micro)economists is represented by strings (from surveys mostly) and it can be pretty messy.

For example: “Avenida Cabildo 1110” is the same as “Cabildo Av. 1110”

Sometimes, people responsible of data entry don’t care about this. So you need to have the neccesary skills to work with strings.

Strings with stringr

The most important package to work with strings in R is **stringr** (it's not part of the core tidyverse, you have to load it). It will provide you methods like the following ones:

- str_length(): to compute the length of a string.
- str_c(): to combine two or more strings.
- str_sub(): to subset, i.e. substract parts of a string.
- str_to_upper() / str_to_lower() are for changing to upper or lowercase the string.

All the stringr functions start with "str_"

Strings with stringr

Regarding the use of quotes:

- It’s the same whether to use ’ or ”
- However, if you want quotes inside a string, you need to use single quotes:

Example

```
string1 <- "This is a string"
string2 <- ’To put a "quote" inside a string, use single
quotes’
```

- You can also use the **escape character** ”\” to write special characters such as quotes or even backslashs

Regular Expressions

Regular expressions or just "**regex**" are a form to describe *patterns* in strings. They're pretty useful because strings usually contain unstructured or semi-structured data, and regexps are a concise language for describing patterns in strings. Common use cases are:

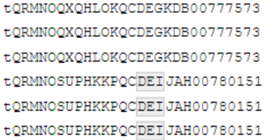
- Data validation
- Text search
- Search engine
- Data Collection: Web scraping

Regular Expressions

Consider the functions str_view() and str_view_all(). These functions take a character vector and a regular expression as parameters, and show how they match.

Example

```
str_view(codusu, "DEI")
```



```
tQRMNOQXQHLOKQCDEGKDB00777573
tQRMNOQXQHLOKQCDEGKDB00777573
tQRMNOQXQHLOKQCDEGKDB00777573
tQRMNOSUPHKKPQCDEIJAH00780151
tQRMNOSUPHKKPQCDEIJAH00780151
tQRMNOSUPHKKPQCDEIJAH00780151
```

Regular Expressions - The tricky part

Regex, that will match strings, are also built with strings. So we will encounter that special characters are tricky to put in regex.

Regex ".e." will match any character on the left and right of "e". So if you want to match a literal ".", you use the backslash \:

1. one backslash for the regex
2. another backslash for the string

As a result, to represent a simple "." you'll have an expression of "\\."

Regular Expressions - The tricky part

And what if you want to match a backslash “\”?

If it’s used as an escape character in regex, we have to escape it, creating the regular expression \\. To create that regular expression, you need to use a string, which also needs to escape \.

That means to match a literal \you need to write “\\\\”-you need four backslashes to match one.

Regular Expressions - Anchors

By default, the regex will just match ANY part of the string. But what if you want to match only at the beginning or at the end of the string?

- To match only the beginning you use the symbol ^
- To match only the end you use the symbol $

Regular Expressions - Character Classes and Alternatives

Patterns that match more than one character:

- "." will match any character
- "\d" matches any digit
- "\s" matches any whitespace (tab, newlines, etc.)
- "[abc]" matches a, b, OR c.
- "[^abc]" matches anything EXCEPT a, b or c.

See examples in our R code.

Regular Expressions - Repetition

How many times can a pattern match?

- ? for 0 or 1
- + for 1 or more
- * for 0 or more
- {n}: exactly n
- {n,}: n or more
- {,m}: at most m
- {n,m}: between n and m

See examples in our R code.

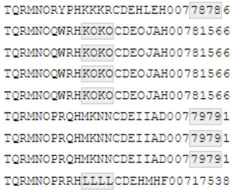
Regular Expressions - Grouping and Backreferences

We can match groups or even refer to groups with backreferences (for example \1, \2).

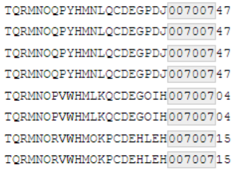
Example

```
str_view(codusu, "(..)\1", match = TRUE) # dos caracteres
que se repitan 1 vez

str_view(codusu, "(...)\1", match = TRUE) # tres caracteres
que se repitan 1 vez
```



```
TQRMNORYPHKKKRCDEHLEH00778786
TQRMNOQWRHKOKOCDEOJAH00781566
TQRMNOQWRHKOKOCDEOJAH00781566
TQRMNOQWRHKOKOCDEOJAH00781566
TQRMNOQWRHKOKOCDEOJAH00781566
TQRMNOPRQHMKNNCDEIIAD00779791
TQRMNOPRQHMKNNCDEIIAD00779791
TQRMNOPRQHMKNNCDEIIAD00779791
TQRMNOPRRHLLLLCDEHMHF00717538
```



```
TQRMNOQPYHMNLQCDEGPDJ00700747
TQRMNOQPYHMNLQCDEGPDJ00700747
TQRMNOQPYHMNLQCDEGPDJ00700747
TQRMNOQPYHMNLQCDEGPDJ00700747
TQRMNOPVWHMLKQCDEGOIH00700704
TQRMNOPVWHMLKQCDEGOIH00700704
TQRMNORVWHMOKPCDEHLEH00700715
TQRMNORVWHMOKPCDEHLEH00700715
```

Regular Expressions - Other functions

1. Detect Matches

To determine if a character vector matches a pattern, use str_detect():

Example

```
x <- c("apple", "banana", "pear")
str_detect(x, "e")
#> [1] TRUE FALSE TRUE
```

This is useful to count questions like “how many common words start with t”, for example.

Example

```
sum(str_detect(words, "^t"))
#> [1] 65
```

Regular Expressions - Other functions

2. Extract Matches

To extract the actual text of a match, use str_extract():

Example

```
str_extract(fecha_nacimiento_eph, "1998")

#> [1] NA "1998" NA NA NA "1998" ...
```

Regular Expressions - Other functions

3. Replace Matches

To replace the actual text of a match with new strings, use str_replace() or str_replace_all():

Example

```
x <- c("apple", "banana", "pear")
str_replace(x, "[aeiou]", "-")
#> [1] "-pple" "p-ar" "b-nana"

str_replace_all(x, "[aeiou]", "-")
#> [1] "-ppl-" "p--r" "b-n-n-"
```

Regular Expressions - Other functions

3. Replace Matches

Going back to our initial example of addresses, we can solve the data entry problem as following:

Example

```
input <- vector de direcciones de una ciudad
str_replace_all(input, "Av.", "Avenida")
```

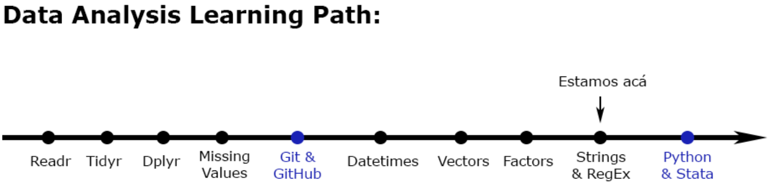
El output será el mismo vector de direcciones, pero estandarizando las palabras claves.

Regular Expressions - Applications

General Rule: Instead of creating one complex regular expression, it's often easier to create a series of simpler regexps. If you get stuck trying to create a single regexp that solves your problem, take a step back and think if you could break the problem down into smaller pieces, solving each challenge before moving on to the next one.

Where are we standing?

¿Dónde estamos parados? ¿Qué vimos y qué faltaría?



Data Analysis Learning Path:

Estamos acá

Readr

Tidyr

Dplyr

Missing Values

Git & GitHub

Datetimes

Vectors

Factors

Strings & RegEx

Python & Stata

Informalidad en Argentina - Datos EPH

La economía argentina se caracteriza por elevados niveles de informalidad (i.e. mercados negros). Sin embargo, ¿cómo podemos aproximar estimaciones de la magnitud de la informalidad laboral con datos de la EPH?

Al menos para datos del mercado de trabajo, tenemos la posibilidad de indagar más sobre este tema con diferentes variables. Justamente la encuesta busca, con ciertas preguntas, determinar si hay individuos en situación de trabajo "negro" o no.

Estas preguntas derivan principalmente de la categoría ocupacional (variable CAT_OCUP) que contempla patrones, cuentapropistas, empelados y trabajadores familiares.

Informalidad en Argentina - Datos EPH

Dado que la informalidad en la categoría de patrón o cuentapropista sería más dificil de determinar, podemos primero enfocarnos en la masa de asalariados (i.e. empleados en relación de dependencia).

En el diseño de registro, a partir de la página 27, aparecen las preguntas vinculadas a dicha actividad laboral. Nótese como, PP07H por ejemplo, hace referencia al descuento jubilatorio de dicho trabajo. Recordemos como vimos en la clase Nº7 acerca de impuestos y aportes relacionados al trabajo, la jubilación era uno de estos. En otras palabras, aquella actividad que no posea descuentos jubilatorios, no será una actividad formal.

Informalidad en Argentina - Datos EPH

Si bien nuestro análisis sobre la informalidad en asalariados resulta en números correctos, pues coinciden con los resultados del informe de INDEC, sería interesante pensar que esta tendencia es generalizable para otras ocupaciones (como cuentapropistas). ¿O acaso los cuentapropistas no evaden monotributos y/o la formalidad en su actividad comercial para poder subsistir?

Aunque a priori no tenemos una pregunta clara y concisa que determine si es informal la ocupación independiente, sería interesante aproximar tasas a partir de ciertas variables, pero quedará en vuestra creatividad y capacidad analítica.