Web Data Collection with R

Regular Expressions / RegEx

Peter Meißner / 2016-02-29 – 2016-03-04 / ECPR WSMT

How Regular Expressions work ...

Patterns

Functions

How Regular Expressions work . . .

What is it all about?

1. Regular Expressions refer to combination of two things
   - a **syntax** that allows to define string patterns
     - e.g.: "`[pP]eter`", "`\\d{4}-\\d{1,2}-\\d{1,2}`"
   - a set of **functions** doing string handling
     - base R has `grep()`, `grepl()`, `substring()`, ... nice because of options ignore.case and invert and bcause build in
     - more convenient stringr/stringi functions: `str_detect()`, `str_replace()`, `str_extract()`, ...

Patterns

Patterns

2. Regular Expressions providing string patterns

pattern	description
`"Hallo"`	1:1
`"."`	any character
`"[]"`	placeholder for one character
`"[abc]"`	set of characters (e.g. a,b, and c)
`"[a-z]"`	range of characters (e.g. a-z, not è, ä, . . .)
`"a*"` / `"a+"`	none or more / one or more
`"a{2,4}"`	two up to four
`"ac|b"`	ac or b
`"[^ab]"`	non of those
`"^a"`	starting with a
`"a$"`	ending with a

Special Characters

3. Expressing Patterns

pattern	description
`"\n"`	newline
`"\r"`	carriage return
`"\t"`	tab
`"\b"`	word boundary (between `"\\w"` and `"\\W"`)
`"\122"`	[matches ASCII character number 82 (octal)
`"\x52"`	matches ASCII character number 82 (hexadecimal)
`"\u0052"`	matches Unicode character number 82 (hexadecimal)

Character Classes

3. Expressing Character Classes

pattern	description
`"\\d" / "\\D"`	digit / no digit
`"\\w" / "\\W"`	word char. / no word char
`"\\s" / "\\S"`	white space char. / no ws char
`"\\p{Currency_Symbol}"`	unicode groups and blocks
`"[[:digit:]]"`	digit
`"[[:alpha:]]"`	characters (also è)
`"[[:alphanum:]]"`	word char.
`"[^[:alphanum:]]"`	white space char.

Syntax Characters

4. some characters have special meaning and cannot be used literally

- . $ ^ { [(|)] } * + ?

character	description	matching
"\"	escapes "\", extra chars	`grep("\\\\","\\")`
"{"	numeral classifier	`grep("\\{","{")`
...	...	...

Functions

functions for string detection

5. base functions for string detection / manipulation

name	description
`grep()`	searches for pat. and returns numeric index/content
`grepl()`	searches for pat. and returns logical index
`gregexpr()`	gives back each position of match
`nchar()`	length of string
`substr()`	extracts sequence of characters
`sub()`	replace one pat. match in string with other string
`gsub()`	replace all pat. matches in string with other string
`paste()`	concatonates vector elements into one string
`paste0()`	concatonates vector elements into one string
-	duplicates string
-	removes leading /trailing whitespace
-	adds whitespace to left, right, or both
-	returns matrix of strings x matches + 1
`cat()`	prints text to screen

functions for string detection

6. stringr/stringi functions for string detection / manipulation

name	description
-	searches for pat. and ret. numeric index/cont.
`str_detect()`	searches for pat. and returns logical index
`str_locate()`	gives back each position of match
`str_length()`	length of str.
`str_sub()`	extracts sequence of characters
`str_replace`	repl. one pat. match in str. with other str.
`str_replace_all()`	repl. all pat. matches in str. with other str.
-	concatonates vector elements into one str.
`str_c()`	concatonates vector elements into one str.
`str_dup`	duplicates string
`str_trim`	removes leading /trailing whitespace
`str_pad`	adds whitespace to left, right, or both
`str_match`	returns matrix of strings x matches + 1
`cat()`	prints text to screen

base / stringr / stringi

- currently there are 3 packages with regular expression engines and string manipulation functions
- base functions might be a little less coonvenient but they are availible out of the box and are solid (and most likely to not change in the near future)
- stringr used to be its own package but is nowadays based on stringi
- stringi is based on a very solid, fast, and powerful C-library
- note, that the RegexEngines of stringi (stringr) and base work differently (see following slide)

base / stringr / stringi

```
library(stringr)
grepl("^a.*d$", "abc\nefgd")
```

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

```
str_detect("abc\nefgd", "^a.*d$")
```

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

```
str_detect("abc\nefgd", regex("a.*d",  dotall=TRUE))
```

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