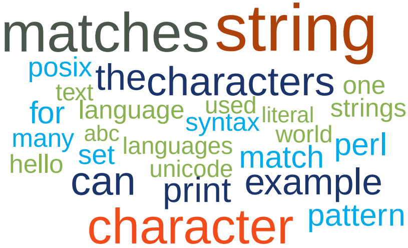
Introduction to Web Scraping with R

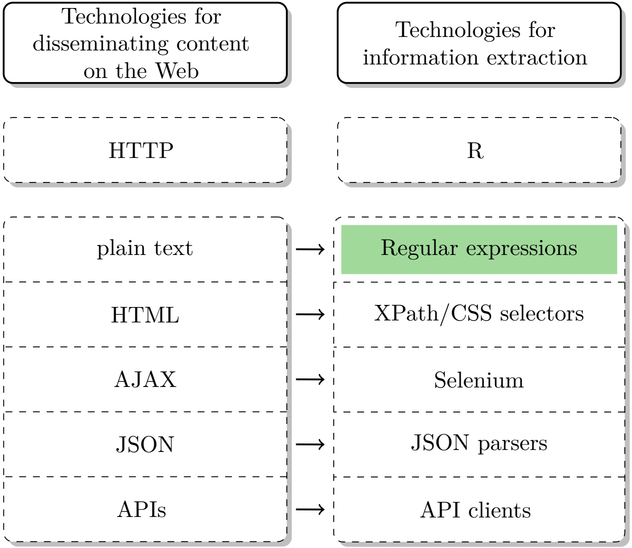
Regular Expressions Basics



Simon Munzert | IPSDS

Regular expressions

Technologies of the World Wide Web



Technologies for disseminating content on the Web		Technologies for information extraction
HTTP		R
plain text	→	Regular expressions
HTML	→	XPath/CSS selectors
AJAX	→	Selenium
JSON	→	JSON parsers
APIs	→	API clients

What are regular expressions?

Definition

- a.k.a. *regex* or *RegExp*
- origins in formal language theory
- sequences of characters that describe patterns in text
- implemented in many programming languages, including R

What are regular expressions?

Definition

- a.k.a. *regex* or *RegExp*
- origins in formal language theory
- sequences of characters that describe patterns in text
- implemented in many programming languages, including R

Why are regular expressions useful for web scraping?

- information on the web can often be described by patterns (think email addresses, numbers, cells in HTML tables, ...)
- if the data of interest follow specific patterns, we can match and extract them—regardless of page layout and HTML overhead
- whenever the information of interest is (stored in) text, regular expressions are useful for extraction and tidying purposes

Introductory example

Introductory example

R code

```r
raw.data <- "555-1239Moe Szyslak(636) 555-0113Burns, C. Montgomery
555-6542Rev. Timothy Lovejoy555 8904Ned Flanders636-555-3226
Simpson,Homer5553642Dr. Julius Hibbert"
```

end

- vector `raw.data` contains unstructured phonebook entries
- goal: extraction of entries
- problem: find a pattern that matches names and numbers
- solution: regex!

Introductory example

R code

```r
raw.data <- "555-1239Moe Szyslak(636) 555-0113Burns, C. Montgomery
555-6542Rev. Timothy Lovejoy555 8904Ned Flanders636-555-3226
Simpson,Homer5553642Dr. Julius Hibbert"
```

end

Solution:

- load package stringr (more on that later)
- a detective's work: construct regex for names
- apply regex on raw vector

R code

```r
library(stringr)
name <- unlist(str_extract_all(raw.data, "[[:alpha:]., ]{2,}"))
name
[1] "Moe Szyslak"          "Burns, C. Montgomery" "Rev. Timothy Lovejoy"
[4] "Ned Flanders"         "Simpson,Homer"        "Dr. Julius Hibbert"
```

end

Introductory example

Solution, *continued*:

- construct regex for phone numbers
- apply regex on raw vector
- combine both vectors

R code

```
phone <- unlist(str_extract_all(raw.data, "\\(?(\\d{3})?\\)?(-| )?\\d{3}(-| )?\\d{4}"))
phone
[1] "555-1239"       "(636) 555-0113" "555-6542"       "555 8904"
[5] "636-555-3226"   "5553642"
data.frame(name = name, phone = phone)
                  name          phone
1          Moe Szyslak       555-1239
2 Burns, C. Montgomery (636) 555-0113
3 Rev. Timothy Lovejoy       555-6542
4         Ned Flanders       555 8904
5        Simpson,Homer   636-555-3226
6   Dr. Julius Hibbert        5553642
```

end

Summary

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

Source: https://xkcd.com/208/ (Randall Munroe)