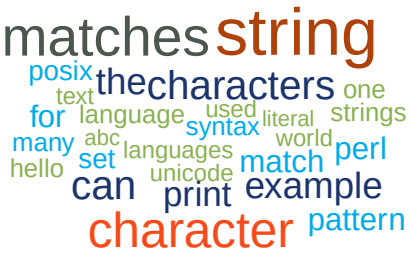
Introduction to Web Scraping with R

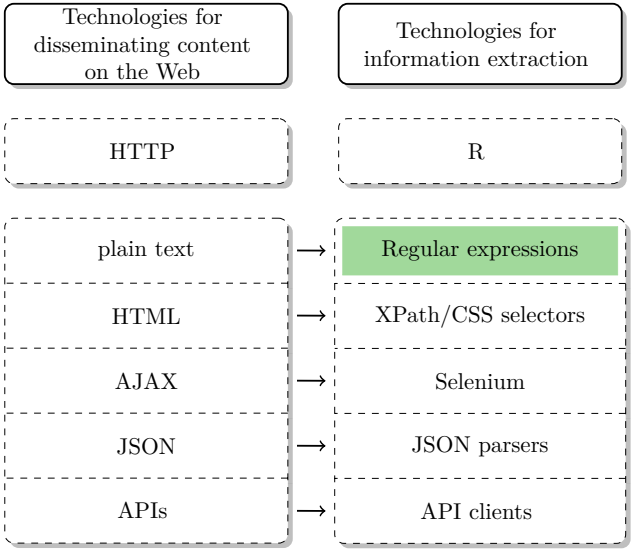
String Manipulation



Simon Munzert | IPSDS

String manipulation in R

Technologies of the World Wide Web



String manipulation in R

What is string manipulation?

- processing of string (character, text) data
- important operations: extraction of text patterns, data tidying, preparation of text corpora for statistical text processing
- web data often is text data!

String manipulation in R

What is string manipulation?

- processing of string (character, text) data
- important operations: extraction of text patterns, data tidying, preparation of text corpora for statistical text processing
- web data often is text data!

String manipulation with R

- base R provides basic string manipulation functionality but not a very consistent syntax
- comfortable string processing with Hadley Wickham's `stringr` package

String manipulation in R with the `stringr` package

Function	Description	Output
Functions using regular expressions		
`str_extract()`	Extracts first string that matches pattern	Character vector
`str_extract_all()`	Extracts all strings that match pattern	List of character vectors
`str_locate()`	Return position of first pattern match	Matrix of start/end positions
`str_locate_all()`	Return positions of all pattern matches	List of matrices
`str_replace()`	Replaces first pattern match	Character vector
`str_replace_all()`	Replaces all pattern matches	Character vector
`str_split()`	Split string at pattern	List of character vectors
`str_split_fixed()`	Split string at pattern into fixed number of pieces	Matrix of character vectors
`str_detect()`	Detect pattern in string	Boolean vector
`str_count()`	Count number of pattern occurrences in string	Numeric vector
Further useful functions		
`str_sub()`	Extract strings by position	Character vector
`str_subset()`	Extract strings for which condition applies	Character vector
`str_length()`	Length of string	Numeric vector
`str_trim()`	Discard string padding	Character vector

Useful functions

Regular expressions in R

Again, our example string:

R code

```
1  example.obj <- "1. A small sentence. - 2. Another tiny sentence."
```

end

String manipulation in R

```
example.obj <- "1.  A small sentence.  - 2.  Another tiny sentence."
```

String localization

R code

```
2  str_locate(example.obj, "small")
```

```
     start end
[1,]     6  10
```

end

String manipulation in R

```
example.obj <- "1.  A small sentence.  - 2.  Another tiny sentence."
```

String localization

R code

```
4  str_locate(example.obj, "small")
     start end
[1,]     6  10
```

end

Substring extraction

R code

```
5  str_sub(example.obj, start = 6, end = 10)
[1] "small"
```

end

String manipulation in R

`example.obj <- "1. A small sentence. - 2. Another tiny sentence."`

String replacement

R code

```
6 str_replace(example.obj, pattern = "tiny", replacement = "huge")
[1] "1. A small sentence. - 2. Another huge sentence."
```

end

String splitting

R code

```
7 unlist(str_split(example.obj, "-"))
[1] "1. A small sentence. "      " 2. Another tiny sentence."
```

end

String manipulation in R

Manipulation of several elements

- until this point we applied functions to vectors of length one
- however, it is common to apply functions to multi-element vectors

Example object with several elements:

R code

```
(char.vec <- c("this", "and this", "and that"))
```

```
[1] "this"     "and this" "and that"
```

end

String detection

R code

```
str_detect(char.vec, "this")
```

```
[1]  TRUE  TRUE FALSE
```

end

String manipulation in R

```
char.vec <- c("this", "and this", "and that")
```

String counting

R code

```
10 str_count(char.vec, "this")
   [1] 1 1 0
11 str_count(char.vec, "\\w+")
   [1] 1 2 2
12 str_length(char.vec)
   [1] 4 8 8
```

end

String subsetting

R code

```
13 str_subset(char.vec, "this")
   [1] "this"     "and this"
```

end

```
char.vec <- c("this", "and this", "and that")
```

String joining

R code

```
14 str_c("text", "manipulation", sep = " ")
   [1] "text manipulation"
15 cat(str_c(char.vec, collapse = "\n"))
   this
   and this
   and that
16 str_c("text", c("manipulation", "basics"), sep = " ")
   [1] "text manipulation" "text basics"
```

end

... but you might want prefer to stick to good old `paste()` and `paste0()`.

String manipulation in R

Approximate matching

- Matching of approximately equal strings, (e.g., “Beyoncé” and “Beyonce”)
- in principle, we could program naïve matching algorithms using regex
- better: use of more powerful algorithms, e.g. Levenshtein distance
- `agrep()` function in base R
- more extended functionality provided by the `stringdist` package

R code

```
17  agrep("Barack Obama", "Barack H. Obama", max.distance = list(all = 3))
    [1] 1
18  agrep("Barack Obama", "Michelle Obama", max.distance = list(all = 3))
    integer(0)
```

end

Summary

Summary

- being able to manipulate string data in R is very useful when you want to automate processing web data
- there are base R commands for string manipulation, but the `stringr` package provides many useful functions with a consistent syntax
- if you need more, check out the even more powerful `stringi` package

Source: http://kevingleong.blogspot.de/2011/01/string-manipulation-exercises.html