Text processing with R: `exact.matches` and other functions

Stefan Th. Gries[1,*]

1. University of California, Santa Barbara
*Contact author: stgries@linguistics.ucsb.edu

Keywords: text processing, pattern matching, regular expression, corpus linguistics

While *R* has been extremely widely and successfully adopted as a programming language for statistical data processing and analysis, its use for text processing and analysis is much less widespread. For instance, in many digital humanities or linguistics contexts, *Perl* or *Python* are still more common even though *R* offers very much the same functionality for text processing plus the advanced statistical and graphical tools that usually follow textual analyses in these fields. Over the last few years, however, R has become more popular in these fields, too, in part because of (i) the availability of a first textbook on text processing with *R* (Gries, 2009) as well as workshops and bootcamps and (ii) because of a variety of functions that provide convenient text processing abilities that are more challenging to implement with the regular base *R* functions. In this talk, I will showcase several functions that are now enjoying wider use in linguistics; specifically, I will discuss

– `exact.matches`: a function that allows to retrieve the exact matches of a search expression in a character vector with many output options: just the matches, matches with the tab-delimited rest of the elements of the character vector with matches, as shown below, …

```
> cat(exact.matches("qwe", "1 1 1 qwe 2 2 2 qwe 3 3 3")[[4]], sep="\n")
[1] 1 1 1               qwe         2 2 2 qwe 3 3 3
[2] 1 1 1 qwe 2 2 2     qwe         3 3 3
```

… matches with user-defined numbers of characters or vector elements as contexts; in addition, contrary to competing functions, the function allows to find multiple overlapping matches:

```
> exact.matches.new(c("s", "s"), "this is a second sentence")[[1]]
[1] "s is"           "s is a s"       "s is a second s" "s a s"
[5] "s a second s"   "second s"
```

– `word.grammy`: a function that generates *n*-grams of text vectors:

```
> word.grammy(c("this","is","a","brand","news","example"), 3, " ")[[1]]
[1] "this is a"          "is a brand"         "a brand news"
[4] "brand news example"
```

– `char.grammy`: a function that generates *n*-grams of characters of text vectors:

```
> char.grammy(c("expressions"), 2)[[1]]
[[1]]
[1] "ex" "xp" "pr" "re" "es" "ss" "si" "io" "on" "ns"

[[2]]
[1] "te" "ex" "xt" "ts"
```

References

Gries, Stefan Th. (2009). *Quantitative Corpus Linguistics with R*, London & New York: Taylor & Francis. URL <http://tinyurl.com/QuantCorpLingWithR>.