DATA607 Assignment 3

Albert Gilharry

15 February 2018

Contents

```r
library("stringr")
```

3. Copy the introductory example. The vector *name* stores the extracted names.

```r
raw.data <- "555-1239Moe Szyslak(636) 555-0113Burns, C. Montgomery555 -6542Rev. Timothy Lovejoy555 8904
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"
```

- Use the tools of this chapter to rearrange the vector so that all elements conform to the standard first_name last_name.

Assuming that this question refers to names that are in reversed order and separated by a comma, and

```r
names <- str_split(name,", ")

for( i in 1:length(name) ){
   if(length(names[[i]]) == 2){
     name[i]<-str_c(names[[i]][2], names[[i]][1], sep=" ")
   }
}
```

Result:

```r
print(name)
```

```
## [1] "Moe Szyslak"          "C. Montgomery Burns"  "Rev. Timothy Lovejoy"
## [4] "Ned Flanders"         "Homer Simpson"        "Dr. Julius Hibbert"
```

- Construct a logical vector indicating whether a character has a title (i.e., Rev. and Dr.).

My assumption here is that titles start with a single uppercase letter followed by 1 to 3 alphabetic cha

```r
str_detect(name,"[:upper:]{1}[:alpha:]{1,3}\\.")
```

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

- Construct a logical vector indicating whether a character has a second name.

My approach to this is to separate the expression into two categories, people with a title and second na

```r
  str_detect(name,"([:upper:]{1}\\.([:space:][:upper:][:alpha:]+){2})|([:upper:]{1}[:lower:]+\\.([:spac
```

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

4. Describe the types of strings that conform to the following regular expressions and construct an example that is matched by the regular expression.

- [0-9]+\$

  This expression represents numbers with a minimum length of 1 digit followed by a dollar sign *$*.

  Example: *0235$*

```r
str_extract("0235$","[0-9]+\\$")
```

```
## [1] "0235$"
```

- \b[a-z]{1,4}\b

  This expression represents words that are up to four letters long.

  Example: *a word*

```r
str_extract_all("a word","\\b[a-z]{1,4}\\b")
```

```
## [[1]]
## [1] "a"    "word"
```

- .*?\.txt$

  This expression represents strings starts with 0 or more characters and ends with *.txt*. This is repr

  Example: *.txt*

```r
str_detect(".txt",".*?\\.txt$")
```

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

- \d{2}/\d{2}/\d{4}

  This represents 8 digits separated by a forward slash after the second digit and one after the fouth

  Example: 24/65/9987

```r
str_extract("29/22/2333","\\d{2}/\\d{2}/\\d{4}")
```

```
## [1] "29/22/2333"
```

- <(.+?)>.+?</\1>

  The represents a text with at minimum 1 character enclosed in an XML type tag. The opening must be
  Example: <d>x</d>

```r
str_extract("<d>x</d>","<(.+?)>.+?</\\1>")
```

```
## [1] "<d>x</d>"
```

9. The following code hides a secret message. Crack it with R and regular expressions. Hint: Some of the characters are more revealing than others! The code snippet is also available in the materials at www.r-datacollection.com.

clcopCow1zmstc0d87wnkig7OvdicpNuggvhryn92Gjuwczi8hqrfpRxs5Aj5dwpn0TanwoUwisdij7Lj8kpf03AT5Idr3c
oc0bt7yczjatOaootj55t3Nj3ne6c4Sfek.r1w1YwwojigOd6vrfUrbz2.2bkAnbhzgv4R9i05zEcrop.wAgnb.
SqoU65fPa1otfb7wEm24k6t3sR9zqe5fy89n6Nd5t9kc4fE905gmc4Rgxo5nhDk!gr

Extract the uppercase letters and punctuations from the text.

```r
txt <- "clcopCow1zmstc0d87wnkig7OvdicpNuggvhryn92Gjuwczi8hqrfpRxs5Aj5dwpn0TanwoUwisdij7Lj8kpf03AT5Idr3c

str_extract_all(txt,"[:upper:]|[:punct:]")
```

```
## [[1]]
##  [1] "C" "O" "N" "G" "R" "A" "T" "U" "L" "A" "T" "I" "O" "N" "S" "." "Y"
## [18] "O" "U" "." "A" "R" "E" "." "A" "." "S" "U" "P" "E" "R" "N" "E" "R"
## [35] "D" "!"
```

This reveals "CONGRATULATIONS YOU ARE A SUPERNERD!".