DATA 607 Assignment 3

Alejandro Osborne

February 18, 2018

```
library(stringi)
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.4.3
```

```
library(knitr)
```

Question 3

```
#Copy the introductory example. The vector name Stores the extracted names

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

#Extract step
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"
```

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

```
#Split step
names<-str_split(name,"(\\,)")
names
```

```
## [[1]]
## [1] "Moe Szyslak"
##
## [[2]]
## [1] "Burns"          " C. Montgomery"
##
## [[3]]
## [1] "Rev. Timothy Lovejoy"
##
## [[4]]
## [1] "Ned Flanders"
##
## [[5]]
## [1] "Simpson" " Homer"
##
## [[6]]
## [1] "Dr. Julius Hibbert"
```

```
#reverse order step
names <- str_replace_all(name, "(.+)(, .+)$", "\\2 \\1")
names
```

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

```
#remove punctuation
newname <- str_replace_all(names, ", ", "")
newname
```

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

```
#no salutations

finalnames <- str_replace_all(newname, "[A-Z][a-z]([a-z]?)\\.", "")
finalnames
```

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

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

```
NamesTable <- data.frame(newname)
NamesTable$Title <- str_detect(newname, "[:alpha:]{2,}\\.")
kable(NamesTable)
```

newname	Title
Moe Szyslak	FALSE
C. Montgomery Burns	FALSE
Rev. Timothy Lovejoy	TRUE
Ned Flanders	FALSE
Homer Simpson	FALSE
Dr. Julius Hibbert	TRUE
##3.3 Construct a logic	al vector indicating whether a character has a second name.

```
NamesTable$SecondName <- str_detect(string = newname, pattern = "[A-Z]{1}\\.")
kable(NamesTable)
```

newname	Title	SecondName
Moe Szyslak	FALSE	FALSE
C. Montgomery Burns	FALSE	TRUE
Rev. Timothy Lovejoy	TRUE	FALSE
Ned Flanders	FALSE	FALSE
Homer Simpson	FALSE	FALSE
Dr. Julius Hibbert	TRUE	FALSE

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

4.1 [0-9]+\$

```
tester <- c("gjkaeilksx8582347$gdsfwsdlkj", "aerfvdlk$avvvvxcdln", "753$129", "4567", "$93sb")
newtester <- unlist(str_extract_all(tester, pattern = "[0-9]+\\$" ))
newtester
```

```
## [1] "8582347$" "753$"
```

This expression will display up to nine digits and then the dollar sign, so long as they are in the respective order

4.2 [a-z]1,4

```
testee <- c("bdb, sfhrr, wrwwrwf", "simpson homer jay", "bring all then 345", "year", "2589", "man")
lasttestee <- str_extract_all(testee, pattern = "\\b[a-z]{1,4}\\b" )
lasttestee
```

```
## [[1]]
## [1] "bdb"
##
## [[2]]
## [1] "jay"
##
## [[3]]
## [1] "all"  "then"
##
## [[4]]
## [1] "year"
##
## [[5]]
## character(0)
##
## [[6]]
## [1] "man"
```

This will find/produce all the words that are 4 characters or less so long as they are all lowercase.

4.3 .*?\.txt$

```
testing <- c("amazon.txt", "allindata.txt  fasttimes.txt", "abdefkltxt", "teachme.txt", "txt", ".txt")
firsttesting <- str_extract_all(testing, pattern = ".*?\\.txt$" )
firsttesting
```

```
## [[1]]
## [1] "amazon.txt"
##
## [[2]]
```

```
## [1] "allindata.txt  fasttimes.txt"
##
## [[3]]
## character(0)
##
## [[4]]
## [1] "teachme.txt"
##
## [[5]]
## character(0)
##
## [[6]]
## [1] ".txt"
```

This will select anything that ends in ".txt"

4.4 \d{2}/\d{2}/\d{4}

```
testerson <- c("59/72/9101", "are rents high", "12/48/2554", "1/1/00", "WTF", "lol", "we/ar/star")
testesterson <- str_extract_all(testerson, pattern = "\\d{2}/\\d{2}/\\d{4}")
testesterson
```

```
## [[1]]
## [1] "59/72/9101"
##
## [[2]]
## character(0)
##
## [[3]]
## [1] "12/48/2554"
##
## [[4]]
## character(0)
##
## [[5]]
## character(0)
##
## [[6]]
## character(0)
##
## [[7]]
## character(0)
```

Here we are looking for 2 digits followed by a forward slash and then 2 digits followed by another forward slash then 4 digits (i.e. MM/DD/YYYY format).

4.5 <(.+?)>.+?</\1>

This appears to be looking for corresponding comparison operators (HTML/XML tags). This one is harder to figure out than the others so we have to test for clarity:

```
test <- c("<body> to </body>", "<head> is on fire not your </head>", "852/743 <div>", "<test/> captiali
finaltest <- str_extract_all(test, pattern = "<(.+?)>.+?</\\1>")
finaltest
```

```
## [[1]]
## [1] "<body> to </body>"
##
## [[2]]
## [1] "<head> is on fire not your </head>"
##
## [[3]]
## character(0)
##
## [[4]]
## character(0)
##
## [[5]]
## character(0)
##
## [[6]]
## character(0)
```

So yes thise expression is definitely looking for the markup language style tags. As long as the beginning and ending tags are matched properly it will produce whatever is written in between.

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

```
secret = "clcopCow1zmstc0d87wnkig7OvdicpNuggvhryn92Gjuwczi8hqrfpRxs5Aj5dwpn0Tanwo Uwisdij7Lj8kpf03AT5Idi
decoder = "[[:lower:].]+"
str_replace_all(paste(unlist(str_extract_all(secret, decoder)),collapse=""),pattern="[\\.]+",replacement
```

```
## [1] "clcopowzmstcdwnkigvdicpuggvhrynjuwczihqrfpxsjdwpnanwowisdijjkpfdrcocbtyczjataootjtjnecfek rwwwoj
```

This doesn't make sense, so lets try numbers:

```
secret1 = "clcopCow1zmstc0d87wnkig7OvdicpNuggvhryn92Gjuwczi8hqrfpRxs5Aj5dwpn0Tanwo Uwisdij7Lj8kpf03AT5Ic
newdecoder = "[[:digit:].]+"
str_replace_all(paste(unlist(str_extract_all(secret1, newdecoder)),collapse=""),pattern="[\\.]+",replace
```

```
## [1] "10877928550780353075533641162 24905 651724639589659490545"
```

This is once again, jibberish. We now go to Capital letters:

```
lastsecret = "clcopCow1zmstc0d87wnkig7OvdicpNuggvhryn92Gjuwczi8hqrfpRxs5Aj5dwpn0Tanwo Uwisdij7Lj8kpf03AT
finaldecoder = "[[:upper:].]+"
str_replace_all(paste(unlist(str_extract_all(lastsecret, finaldecoder)),collapse=""),pattern="[\\.]+",re
```

```
## [1] "CONGRATULATIONS YOU ARE A SUPERNERD"
```

Yay!