Homework Week 3

CUNY MSDA DATA 607

Duubar Villalobos Jimenez mydvtech@gmail.com

February 19, 2017

Please deliver links to an R Markdown file (in GitHub and rpubs.com) with solutions to problems 3 and 4 from chapter 8 of Automated Data Collection in R. Problem 9 is extra credit.

Library definition

```
# Need to employ stringr for Regular Expressions
library(stringr)
```

Problems

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

Original Example

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

Extracting the vector name

```
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"
```

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

```
# Original Name vector obtained by using Regular Expressions
name
```

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

```
# Split name for those names that have the last name first separated by a comma
split_name <- str_split(name, ",")
split_name
```

```
## [[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"
```

```
# Create a data frame to work from there
split_name <- data.frame(split_name)
# Display the new data frame, in order to vizualice the two rows, from here we can observe the differen
split_name
```

```
##   X.Moe.Szyslak. c..Burns.....C..Montgomery.. X.Rev..Timothy.Lovejoy.
## 1    Moe Szyslak                       Burns    Rev. Timothy Lovejoy
## 2    Moe Szyslak               C. Montgomery    Rev. Timothy Lovejoy
##   X.Ned.Flanders. c..Simpson.....Homer.. X.Dr..Julius.Hibbert.
## 1    Ned Flanders               Simpson    Dr. Julius Hibbert
## 2    Ned Flanders                 Homer    Dr. Julius Hibbert
```

```
# Assing the "Last Names" Row
ln <- data.frame(split_name[1,])
# Assing the "First Names" Row
fn <- data.frame(split_name[2,])

# Compare rows and proceed to create the desired "Firt Name then Last name" output by employing rbind
split_name <- ifelse(fn == ln, ln , rbind(fn, ln))
split_name
```

```
## [[1]]
## [1] Moe Szyslak
## Levels: Moe Szyslak
##
## [[2]]
## [1]  C. Montgomery Burns
## Levels:  C. Montgomery Burns
##
## [[3]]
## [1] Rev. Timothy Lovejoy
## Levels: Rev. Timothy Lovejoy
##
## [[4]]
## [1] Ned Flanders
## Levels: Ned Flanders
##
## [[5]]
## [1]  Homer  Simpson
## Levels:  Homer Simpson
##
## [[6]]
```

```
## [1] Dr. Julius Hibbert
## Levels: Dr. Julius Hibbert
```

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

```
# Defining Tiles Vector
title <- c("Rev.","Dr.")

# Find out if the title is part of the name
names_wtitle <- ifelse(str_detect(name,title) == TRUE, "YES", "NO")

# Creating a data frame to represent if title is part of the name
names_wtitle <- data.frame (name=name, title=names_wtitle)
names_wtitle
```

```
##                   name title
## 1          Moe Szyslak    NO
## 2 Burns, C. Montgomery    NO
## 3 Rev. Timothy Lovejoy   YES
## 4         Ned Flanders    NO
## 5       Simpson, Homer    NO
## 6   Dr. Julius Hibbert   YES
```

(c) Construct a logical vector indicating whether a character has a second name.

```
# Identify if the name has a Middle name. The Regular expressions are generally and initial followed by
middle_name <- ifelse(str_detect(name,"[A-Z]\\.") == TRUE, "YES", "NO")
middle_name
```

```
## [1] "NO"  "YES" "NO"  "NO"  "NO"  "NO"
```

```
# Report next to a name in a data frame.
names_wmname <- data.frame (name, middle=middle_name)
names_wmname
```

```
##                   name middle
## 1          Moe Szyslak     NO
## 2 Burns, C. Montgomery    YES
## 3 Rev. Timothy Lovejoy     NO
## 4         Ned Flanders     NO
## 5       Simpson, Homer     NO
## 6   Dr. Julius Hibbert     NO
```

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

(a) [0-9]+\$

This one represent a one digit from [0-9] repeating multiple times to the right at the end of a expression with a dollar ($) sign at the end of the number.

```
raw.vector <- c("This is my example where 1234567890$ is represented at the end of the vector; it will
unlist(str_extract_all(raw.vector, "[0-9]+\\$"))
```

```
## [1] "1234567890$" "1234567890$" "123$"
```

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

In this example it will display all the words that are surrounded by edges \b on both sides and composed of four letters or less but with LOWER CASE only.

```
raw.vector <- c("In this example it will display all the words that are surrounded by edges \\b on both
unlist(str_extract_all(raw.vector, "\\b[a-z]{1,4}\\b"))
```

```
##  [1] "this" "it"   "will" "all"  "the"  "that" "are"  "by"   "b"    "on"
## [11] "both" "and"  "of"   "four" "or"   "less" "but"  "with" "only"
```

(c) .*?\.txt$

In this example it will display all the sentences that are composed of a period . followed by a word that has the asterisk or star that could be optional and then followed by the extension.txt, for example:

```
raw.vector <- c("In this example it will display all the sentences that are composed of a period . follo
unlist(str_extract_all(raw.vector, ".*?\\.txt$"))
```

```
## [1] "this is good my.homework.txt"
## [2] "but this one is great my.homew*rk.txt"
```

(d) \d{2}/\d{2}/\d{4}

This is for dates composed of up to two digit month, two digit day and four digit year separated with the slash symbol.

```
raw.vector <- c("in this example we will extract 02/04/2016 and 08/09/1977","Also we will not be able t
unlist(str_extract_all(raw.vector, "\\d{2}/\\d{2}/\\d{4}"))
```

```
## [1] "02/04/2016" "08/09/1977"
```

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

After doing some research, this represents the Vector containing strings with any type of HTML tag. The back reference removes the outer HTML tags.

The \1 is doing a recall of <(.+?)> at the end of the regular expression.

```
raw.vector <- c("<!DOCTYPE html><html><body>Hello World</body></html></html>")
unlist(str_extract_all(raw.vector, "<(.+?)>.+?</\\1>"))
```

```
## [1] "<html><body>Hello World</body></html>"
```

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.

clcopCow1zmstc0d87wnkig7OvdicpNuggvhryn92Gjuwczi8hqrfpRxs5Aj5dwpn0Tanwo Uwisdij7Lj8kpf03AT5Idr3coc0bt7yczjatO
d6vrfUrbz2.2bkAnbhzgv4R9i05zEcrop.wAgnb.SqoU65fPa1otfb7wEm24k6t3sR9zqe5 fy89n6Nd5t9kc4fE905gmc4Rgxo5nhDk!gr

```
# Raw vector
raw.vector <- "clcopCow1zmstc0d87wnkig7OvdicpNuggvhryn92Gjuwczi8hqrfpRxs5Aj5dwpn0TanwoUwisdij7Lj8kpf03A
# I noticed there are some upper case letters and some periods in between, so I run the code for Alphan
hidden_message <- unlist(str_extract_all(raw.vector, "[[:upper:].?]{1,}"))
hidden_message
```

```
##  [1] "C"  "O"  "N"  "G"  "R"  "A"  "T"  "U"  "L"  "AT" "I"  "O"  "N"  "S"
## [15] "."  "Y"  "O"  "U"  "."  "A"  "R"  "E"  "."  "A"  ".S" "U"  "P"  "E"
## [29] "R"  "N"  "E"  "R"  "D"
```

```
# Since the periods work as a separator, we can replace them for blank spaces and also we can put all t
hidden_message <- str_replace_all(paste(hidden_message, collapse = ''), "[.]", " ")
# Final message
hidden_message
```

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