Lab 4: Text Pre-Processing

Iulia Cioroianu

Basic String Manipulation using Regular Expressions

We are going to be using the stringr package for this section:

```
#install.packages("stringr")
library(stringr)
```

Basic Commands for Cleaning Text ###

Let's start with an example string:

```
my_string <- "I want 2 learn text analysis 4 my research!
  Where is the best place to do that?"
# Print it out:
my_string
```

```
## [1] "I want 2 learn text analysis 4 my research! \n  Where is the best place to do that?"
```

How many characters does it have?

```
length(my_string)
```

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

```
str_count(my_string)
```

```
## [1] 82
```

Lowercase:

```
lower_string <- tolower(my_string)
lower_string
```

```
## [1] "i want 2 learn text analysis 4 my research! \n  where is the best place to do that?"
```

We can also combine strings using the paste() command:

```
second_string <- "
ESS 2020."
```

```
my_string <- paste(my_string,second_string,sep = " ")
my_string
```

```
## [1] "I want 2 learn text analysis 4 my research! \n  Where is the best place to do that? \nESS 2020.
```

It is also possible to pull out all substrings matching a given string argument.

```
str_extract_all(my_string, "is")
```

```
## [[1]]
## [1] "is" "is"
```

Use | to for "or" Put individual elements in square brackets [] to look for any of them.

```
str_extract_all(my_string, "2|the|[!?]")
```

```
## [[1]]
## [1] "2"   "!"   "the" "?"   "2"   "2"
```

"[0-9]+" is a regular expression. The + is for one of more of what't in the brackets. In the brackets we have digits 0-9. So this matches all numbers.

```
str_extract_all(my_string,"[0-9]+")
```

```
## [[1]]
## [1] "2"    "4"    "2020"
```

0-9 isn't the only range you can look for:

```
str_extract_all(my_string,"[a-zA-Z]+") #  sequences of letters small OR big
```

```
## [[1]]
##  [1] "I"        "want"     "learn"    "text"     "analysis" "my"
##  [7] "research" "Where"    "is"       "the"      "best"     "place"
## [13] "to"       "do"       "that"     "ESS"
```

One thing we have to note is that there are a number of special characters in regular expressions that need to be escaped: These include [ ] { } ( ) " ' $ ^ & * ? . - + These should all be escaped with a \ as in \$ if you want to literally match them.

Instead of 0-9, we can just say "\d" for digits

```
str_extract_all(my_string,"\\d+")
```

```
## [[1]]
## [1] "2"    "4"    "2020"
```

Or try:

```
str_extract_all(my_string,"\\w+") # strings of alphanumeric characters
```

```
## [[1]]
##  [1] "I"        "want"     "2"        "learn"    "text"     "analysis"
##  [7] "4"        "my"       "research" "Where"    "is"       "the"
## [13] "best"     "place"    "to"       "do"       "that"     "ESS"
## [19] "2020"
```

Extract string between characters:

```
matched_string <- str_match(my_string, "2 \\s*(.*?)\\s* 4")
matched_string
```

```
##      [,1]                     [,2]
## [1,] "2 learn text analysis 4" "learn text analysis"
```

```
matched_string[1]
```

```
## [1] "2 learn text analysis 4"
```

Exercise:

Find all sequences of small letters, all sequences of big letter and all strings of digits .

```
str_extract_all(my_string,"[a-z]+|[A-Z]+|\\d+") #  sequences of letters small OR big
```

```
## [[1]]
##  [1] "I"        "want"     "2"        "learn"    "text"     "analysis"
##  [7] "4"        "my"       "research" "W"        "here"     "is"
## [13] "the"      "best"     "place"    "to"       "do"       "that"
## [19] "ESS"      "2020"
```

We use "|" to mean "or" in regular expressions:

```
str_extract_all(my_string,"th(e|at)")
```

```
## [[1]]
## [1] "the"  "that"
```

Fing location of certain character or set of characters within a string:

```
str_locate_all(my_string, "[?!]|the")
```

```
## [[1]]
##      start end
## [1,]    43  43
## [2,]    57  59
## [3,]    82  82
```

Split up strings on a particular character sequence or sets of characters:

```
str_split(my_string, "[!?]")
```

```
## [[1]]
## [1] "I want 2 learn text analysis 4 my research"
## [2] " \n  Where is the best place to do that"
## [3] " \nESS 2020."
```

```
str_split(my_string, "\n")
```

```
## [[1]]
## [1] "I want 2 learn text analysis 4 my research! "
## [2] "  Where is the best place to do that? "
## [3] "ESS 2020."
```

The str_replace_all function can be used to replace all instances of a given string, with an alternative string:

```
str_replace_all(my_string, "4", "for")
```

```
## [1] "I want 2 learn text analysis for my research! \n  Where is the best place to do that? \nESS 202
```

Exercise:

It's not very nice to write 4 instead of "for" and 2 instead of "to". Create a new string called correct_string and fix these errors. Make sure that it looks good in the end!

```
str_replace_all(my_string, "2", "to")
```

```
## [1] "I want to learn text analysis 4 my research! \n  Where is the best place to do that? \nESS to0t
```

```
# Not exactly what we wanted, though, is it?
# How about:
str_replace_all(my_string, " 2 ", " to ")
```

```
## [1] "I want to learn text analysis 4 my research! \n  Where is the best place to do that? \nESS 2020
```

```
#Much better. Now let's change it:
my_string <- str_replace_all(my_string, " 4 ", " for ")
my_string <- str_replace_all(my_string, " 2 ", " to ")
```

You can also split it into lines based on end of line characters:

```
str_split(my_string,"\\n")
```

```
## [[1]]
## [1] "I want to learn text analysis for my research! "
## [2] "  Where is the best place to do that? "
## [3] "ESS 2020."
```

Turn character number into string:

```
number_two <- as.numeric("2")
number_two
```

```
## [1] 2
```

```
typeof(number_two)
```

```
## [1] "double"
```

Exercise: Extracting elements from text and put them into a dataset.

The text below is from a BBC News collection on official statistics reporting:

```
text <- "In December 2019 - the most recent borrowing forecasts from the OBR - the UK's
budget deficit (the shortfall between government spending and tax income) was forecast
to rise from £41.0bn in 2018-19 to as much as £47.6bn in 2019-20, before falling slightly
to £40.2bn in 2020-21, then £37.6bn in 2021-22, £35.4bn in 2022-23 and £33.3bn in 2023-24."
```

- We need to extract and put into a data frame called Deficit_data:
- the budget year (such as 2022-2023) as a variable called "Year";
- the budget as a numeric variable called "Deficit".

```
# 1. Start by printing to console all contiguous sequences of numbers of length one or greater.
str_extract_all(text,"[0-9]+")
```

```
## [[1]]
##  [1] "2019" "41"   "0"    "2018" "19"   "47"   "6"    "2019" "20"   "40"
## [11] "2"    "2020" "21"   "37"   "6"    "2021" "22"   "35"   "4"    "2022"
## [21] "23"   "33"   "3"    "2023" "24"
```

```
# Extract all pound amounts of the form "£10.4bn" and save them in a new variable called "Spending"
Deficit <- str_extract_all(text,"£[,.0-9]+bn")[[1]]
Deficit
```

```
## [1] "£41.0bn" "£47.6bn" "£40.2bn" "£37.6bn" "£35.4bn" "£33.3bn"
```

```
# That says "give me everything that
# starts with a "£"
# followed by one or more commas or dots, or digits 0 to 9.

# Let's try to get year ranges:
Year <- str_extract_all(text,"[0-9]+\\-[0-9]+")[[1]]
Year
```

```
## [1] "2018-19" "2019-20" "2020-21" "2021-22" "2022-23" "2023-24"
```

```
# Remove £, "," and bn, add zeros and turn to numeric.
# Put them all in a data frame.
Deficit <- str_replace_all(Deficit, "£","")
Deficit <- str_replace_all(Deficit, "\\.","")
Deficit <- str_replace_all(Deficit, "bn","00000000")
Deficit <- as.numeric(Deficit)
Deficit
```

```
## [1] 4.10e+10 4.76e+10 4.02e+10 3.76e+10 3.54e+10 3.33e+10
```

```
options(scipen=999)

# Let's put it into a data frame:
Deficit_data <- data.frame(Year, Deficit)
```

Read more about regular expressions at: https://cran.r-project.org/web/packages/stringr/vignettes/stringr.html

Text pre-processing with Quanteda

```
library(quanteda)
```

```
## Warning in stringi::stri_info(): Your current locale is not in the list
## of available locales. Some functions may not work properly. Refer to
## stri_locale_list() for more details on known locale specifiers.

## Warning in stringi::stri_info(): Your current locale is not in the list
## of available locales. Some functions may not work properly. Refer to
## stri_locale_list() for more details on known locale specifiers.
```

```
## Package version: 3.0.0
## Unicode version: 13.0
## ICU version: 66.1
```

```
## Parallel computing: 16 of 16 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

Tokenize text

```
raw_text="Hi! Welcome to Intro to Text Analysis at ESS 2020. I am @iuliacioroianu - http://www.iuliacio
tks <- tokens(raw_text)
tks
```

```
## Tokens consisting of 1 document.
## text1 :
##  [1] "Hi"       "!"        "Welcome"  "to"       "Intro"    "to"
##  [7] "Text"     "Analysis" "at"       "ESS"      "2020"     "."
## [ ... and 5 more ]
```

```
paste(tks, collapse=" ")
```

```
## [1] "Hi ! Welcome to Intro to Text Analysis at ESS 2020 . I am @iuliacioroianu - http://www.iuliacio
```

More options:

```
?tokens
tks <- tokens(raw_text,
              remove_punct=TRUE,
```

```
              remove_symbols=TRUE,
              remove_numbers=TRUE,
              remove_url=TRUE)
tks
```

```
## Tokens consisting of 1 document.
## text1 :
##  [1] "Hi"              "Welcome"         "to"              "Intro"
##  [5] "to"              "Text"            "Analysis"        "at"
##  [9] "ESS"             "I"               "am"              "@iuliacioroianu"
```

```
paste(tks, collapse=" ")
```

```
## [1] "Hi Welcome to Intro to Text Analysis at ESS I am @iuliacioroianu"
```

Turn tokens to lower case

```
tokens_tolower(tks)
```

```
## Tokens consisting of 1 document.
## text1 :
##  [1] "hi"              "welcome"         "to"              "intro"
##  [5] "to"              "text"            "analysis"        "at"
##  [9] "ess"             "i"               "am"              "@iuliacioroianu"
```

Remove stopwords

```
tks <- tokens_remove(tks, pattern = stopwords('en'))
```

Stem

```
tokens_wordstem(tks, language = quanteda_options("language_stemmer"))
```

```
## Tokens consisting of 1 document.
## text1 :
## [1] "Hi"              "Welcom"          "Intro"           "Text"
## [5] "Analysi"         "ESS"             "@iuliacioroianu"
```

Get ngrams

```
tokens_ngrams(tks, n=2:3)
```

```
## Tokens consisting of 1 document.
## text1 :
##  [1] "Hi_Welcome"                  "Welcome_Intro"
##  [3] "Intro_Text"                  "Text_Analysis"
##  [5] "Analysis_ESS"                "ESS_@iuliacioroianu"
##  [7] "Hi_Welcome_Intro"            "Welcome_Intro_Text"
##  [9] "Intro_Text_Analysis"         "Text_Analysis_ESS"
## [11] "Analysis_ESS_@iuliacioroianu"
```

NLP with spacyR

```
library(spacyr)
#spacy_install()
#spacy_download_langmodel()
spacy_initialize(model = "en_core_web_sm")
```

```
## Found 'spacy_condaenv'. spacyr will use this environment
## successfully initialized (spaCy Version: 3.1.0, language model: en_core_web_sm)
## (python options: type = "condaenv", value = "spacy_condaenv")
txtparsed <- spacy_parse(raw_text, tag = TRUE, pos = TRUE)

txtparsed
```

```
##    doc_id sentence_id token_id                          token
## 1   text1           1        1                             Hi
## 2   text1           1        2                              !
## 3   text1           2        1                        Welcome
## 4   text1           2        2                             to
## 5   text1           2        3                          Intro
## 6   text1           2        4                             to
## 7   text1           2        5                           Text
## 8   text1           2        6                       Analysis
## 9   text1           2        7                             at
## 10  text1           2        8                            ESS
## 11  text1           2        9                           2020
## 12  text1           2       10                              .
## 13  text1           3        1                              I
## 14  text1           3        2                             am
## 15  text1           3        3                 @iuliacioroianu
## 16  text1           3        4                              -
## 17  text1           3        5 http://www.iuliacioroianu.info
##                             lemma   pos  tag   entity
## 1                              hi  INTJ   UH
## 2                               ! PUNCT    .
## 3                         welcome  INTJ   UH
## 4                              to   ADP   IN
## 5                           Intro PROPN  NNP PERSON_B
## 6                              to   ADP   IN
## 7                           Text PROPN  NNP
## 8                         Analysis PROPN  NNP
## 9                              at   ADP   IN
## 10                            ESS PROPN  NNP
## 11                           2020   NUM   CD   DATE_B
## 12                              . PUNCT    .
## 13                              I  PRON  PRP
## 14                             be   AUX  VBP
## 15                 @iuliacioroianu  NOUN   NN
## 16                              - PUNCT HYPH
## 17 http://www.iuliacioroianu.info  NOUN   NN
```

What we do with this? We include the POS tags in our future models as features.