Silesian University of Technology

RESEARCH UNIVERSITY
EXCELLENCE INITIATIVE
Ministry of Science and Higher Education

Programming for Cognitive Science

Lecture 4 – R for text-mining

Joanna Tobiasz, PhD
Anna Papiez, PhD

Department of Data Science and Engineering

Text-mining

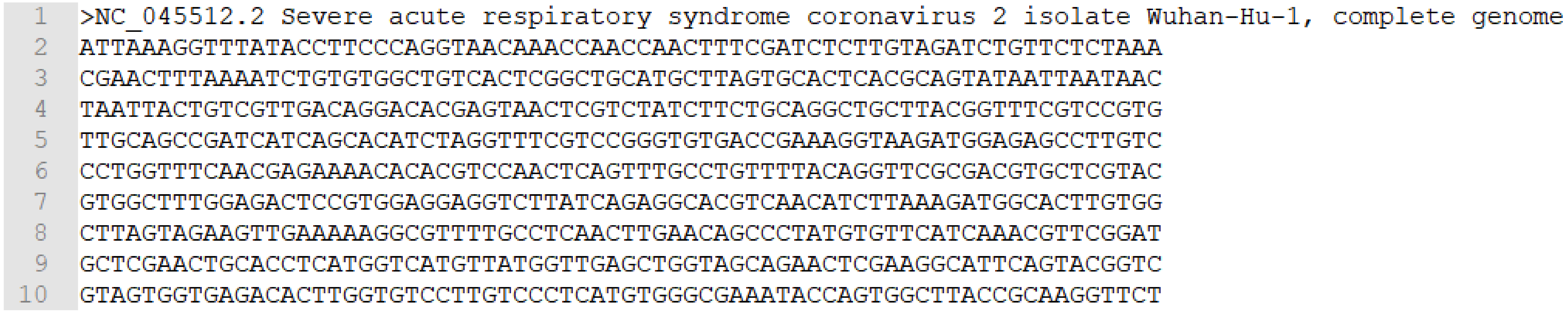
Kind of data mining methods dedicated to extracting data from the text and further analysis of those data.

Used especially for the analysis of:

- Social media data
- Articles, including scientific manuscripts
- E-mails
- Survey responses
- CV and reference letters in HR
- Chatbots
- Clinical information
- DNA, RNA, and protein sequences

Simple string analysis

```
library("seqinr")
sars <- read.fasta(file = “sarscov2.fasta")
```



```
>NC_045512.2 Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome
ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCTGTTCTCTAAA
CGAACTTTAAAATCTGTGTGGCTGTCACTCGGCTGCATGCTTAGTGCACTCACGCAGTATAATTAATAAC
TAATTACTGTCGTTGACAGGACACGAGTAACTCGTCTATCTTCTGCAGGCTGCTTACGGTTTCGTCCGTG
TTGCAGCCGATCATCAGCACATCTAGGTTTCGTCCGGGTGTGACCGAAAGGTAAGATGGAGAGCCTTGTC
CCTGGTTTCAACGAGAAAACACACGTCCAACTCAGTTTGCCTGTTTTACAGGTTCGCGACGTGCTCGTAC
GTGGCTTTGGAGACTCCGTGGAGGAGGTCTTATCAGAGGCACGTCAACATCTTAAAGATGGCACTTGTGG
CTTAGTAGAAGTTGAAAAAGGCGTTTTGCCTCAACTTGAACAGCCCTATGTGTTCATCAAACGTTCGGAT
GCTCGAACTGCACCTCATGGTCATGTTATGGTTGAGCTGGTAGCAGAACTCGAAGGCATTCAGTACGGTC
GTAGTGGTGAGACACTTGGTGTCCTTGTCCCTCATGTGGGCGAAATACCAGTGGCTTACCGCAAGGTTCT
```

Simple string analysis

```
length(sars[[1]])
[1] 29903
table(sars[[1]])
 a    c    g    t
8954 5492 5863 9594

head(sars[[1]])
[1] "a" "t" "t" "a" "a" "a„
tail(sars[[1]])
[1] "a" "a" "a" "a" "a" "a"
```

Simple string analysis

Count DNA words:

```
count(sars[[1]], 2)
  aa   ac   ag   at   ca   cc   cg   ct   ga   gc   gg   gt   ta   tc   tg   tt
2880 2023 1742 2308 2084  888  439 2081 1612 1168 1093 1990 2377 1413 2589 3215

count(sars[[1]], 3)
aaa aac aag aat aca acc acg act aga agc agg agt ata atc atg att caa cac cag cat cca ccc ccg
923 615 580 761 809 376 164 674 605 301 329 507 471 339 725 773 703 459 438 484 354 116  74
cct cga cgc cgg cgt cta ctc ctg ctt gaa gac gag gat gca gcc gcg gct gga ggc ggg ggt gta gtc
344  95  97  76 171 561 287 495 738 535 340 297 440 372 187  88 521 282 223 134 454 469 269
gtg gtt taa tac tag tat tca tcc tcg tct tga tgc tgg tgt tta ttc ttg ttt
552 700 719 609 427 622 549 209 113 542 630 547 554 858 876 518 817 100
```

Let’s move on to coding...

RStudio®

Regular expressions

Sequences of characters that define a search pattern:

```
^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$
```

Regular expressions

Sign		Meaning	Sign		Meaning
\	back slash	escape character	|	pipe	alternation [OR]
[ ]	square brackets	single character match	*	asterisk	zero or more times of repeat
{ }	curly braces	repeats	+	plus sign	one or more times of repeat
( )	parenthesis	reference or subexpression	?	question mark	occur 0 times or once
^	hat	beginning of a line	.	dot	any single character
$	dollar	end of a line	!	exclamation mark	negation [NOT]

Regular expressions

Sign	Meaning	Sign	Meaning
\d	digit [0-9]	\t	tab
\D	non-digit	\n	new line
\w	word character [a-zA-Z0-9]	\r	return
\W	non-word character	\f	end of page
\A	beginning of string	\s	white space
\z	end of string	\S	non white space

Text mining functions

- grep
- sub/gsub
- chartr
- regexpr
- regexec
- gregexpr
- regmatches
- substr
- strsplit
- paste/paste0
- tolower/toupper
- nchar

grep

Globally search a regular expression and print

```
grep(pattern,vector)

x <- c("abc", "bcd", "cde", "def")

grep("bc", x)
  [1] 1 2
```

grep

```
x <- c("abc", "bcd", "cde", "def")

grep("bc", x , value=TRUE)
[1] "abc" "bcd"

grep("bc", x, invert=TRUE)
[1] 3 4
```

No match

grep

```
x <- c("abc", "bcd", "cde", "def")

grep("BC", x, ignore.case=TRUE)
[1] 1 2

grepl("bc",x)
[1]  TRUE  TRUE FALSE FALSE

grep("[a-c]", x)
[1] 1 2 3

grep("[a-c]", x, fixed = T)
integer(0)
```

sub/gsub

```
sub("match","replace", input_vector)

x <- c("abc", "bcd", "cde", "def")

sub(".*(bc).*", "gh", x)
[1] "gh"  "gh"  "cde" "def"
```

chartr

```
chartr("string","replacement", input_vector)

x <- "This lecture is poor"

chartr("pr", "gd", x)
[1] "This lecture is good"
```

regexpr

First position of matched regular expression:

```
a <- "Mississippi contains a palindrome ississi."
regexpr("iss", a)
  [1] 2
  attr(,"match.length")
  [1] 3
  attr(,"useBytes")
  [1] TRUE
```

regexec

If there are parenthesized matching conditions, it will show both matched string position and the position of parenthesized matched string.

```
a <- "Mississippi contains a palindrome ississi."
regexec(".(ss)",a)
  [[1]]
  [1] 2 3
  attr(,"match.length")
  [1] 3 2
```

gregexpr

All positions of matched regular expression:

```
a <- "Mississippi contains a palindrome ississi."
gregexpr("iss", a)
  [[1]]
  [1]  2  5 35 38
  attr(,"match.length")
  [1] 3 3 3 3
  attr(,"useBytes")
  [1] TRUE
```

regmatches

Showing matched strings:

```
a <- "Mississippi contains a palindrome ississi."
b <- gregexpr("iss", a)
regmatches(a,b)
  [[1]]
  [1] "iss" "iss" "iss" "iss"
```

substr

Extract substring from input string:

```
substr(x, start, end)

x <- "abcdef"
substr(x, 3, 5)
[1] "cde"
```

substr

Replacing a substring:

```
x <- "abcdef"
substr(x,3,4) <- "CD"
[1] "abCDef"
```

strsplit

Split string on common separator:

```
strsplit("6/11/2015","/")
  [[1]]
  [1] "6"    "11"    "2015"
```

paste

Concatenate vectors after converting to character:

```
a <- unlist(strsplit("6/11/2015","/"))

paste(a, "/")
    [1] "6/11/2015"
```

paste and paste0

```
a <- 'My’
b <- 'string’
paste(a,b)
[1] "My string“
paste0(a,b)
[1] "Mystring“
paste(a,b, sep = '.')
[1] "My.string“
paste(a,b, sep = '')
[1] "Mystring"
```

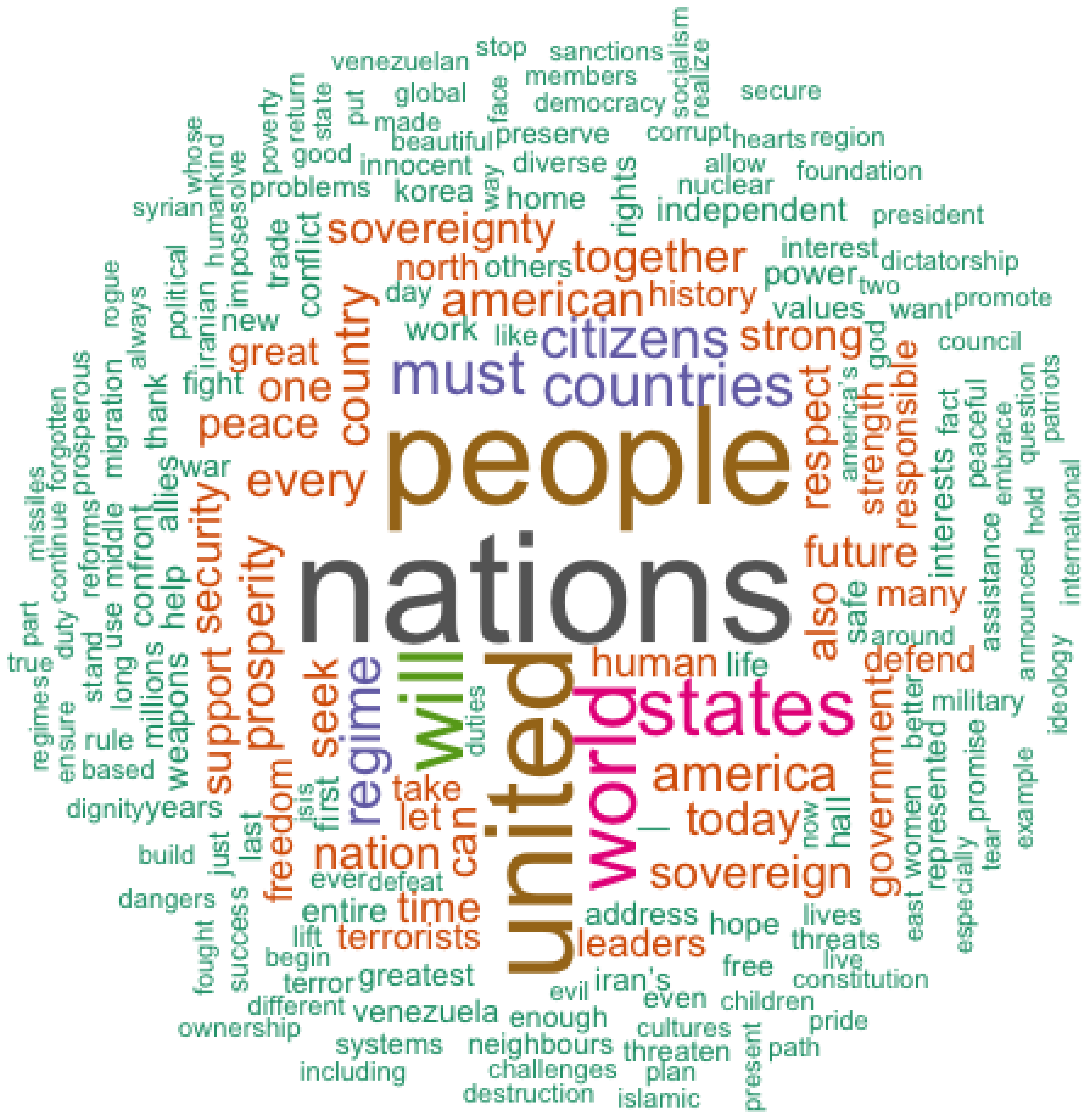
tolower and toupper

```
x <- "MiXeD cAsE 123"
tolower(x)
[1] "mixed case 123"
toupper(x)
[1] "MIXED CASE 123"
```

nchar

```
x <- “abcdf"
nchar(x)
[1] 5
```

Wordcloud plot



```
library(wordcloud)
wordcloud(words, frequencies,
          min.freq = 3, colors = brewer.pal(5, "Dark2"))
```

https://towardsdatascience.com/create-a-word-cloud-with-r-bde3e7422e8a

Practice

Download the genbank.RData dataset

1) Extract the accession number, the definition, and the organism.
2) Extract all MEDLINE article numbers which are mentioned in the entries.
3) Extract the DNA, merge the entire sequence and complement it.
   Hint: In the DNA guanine (g) is complemented by cytosine (c), adenine (a) by thymine (t).

Let’s move on to coding...

R Studio®

Silesian
University
of Technology

RESEARCH
UNIVERSITY
EXCELLENCE INITIATIVE
Ministry of Science
and Higher Education

I APPRECIATE YOUR ATTENTION