TEXT MINING

Lecture 04

TEXT EXPLORATION

KEUNGOUI KIM

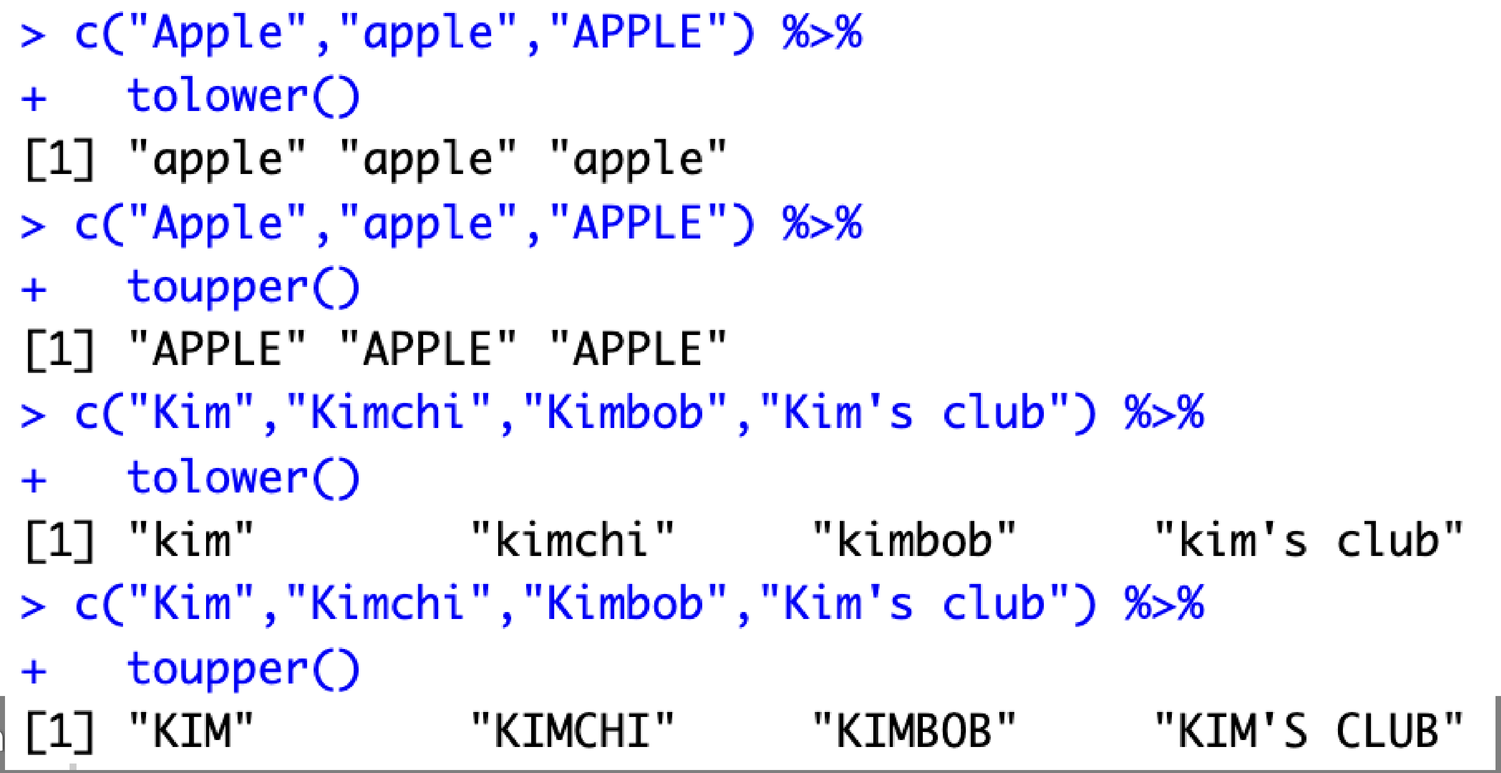
awekim@handong.edu

Handong Global University, 2024-2.

Key Functions for Text Mining

Text Conversion

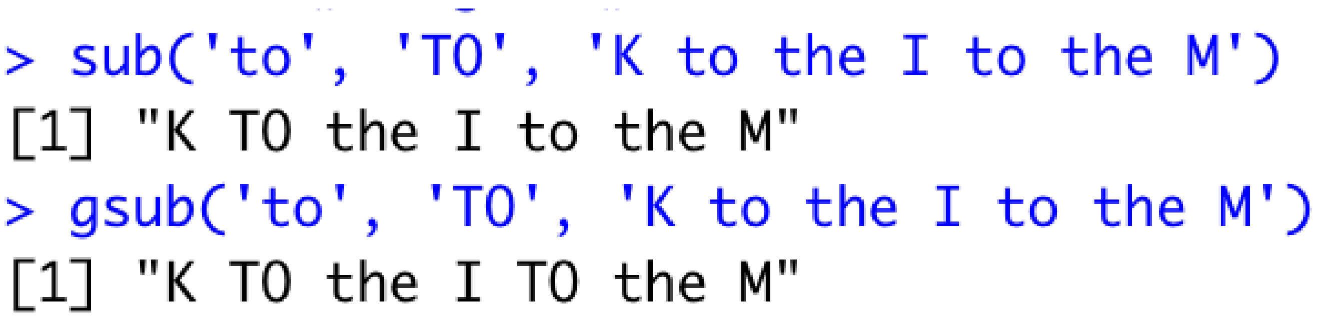
- Very basic programming knowledge
  - Apple vs. apple vs. APPLE ➔ Same meaning, but treated differently in programming
  - As a simple task, tolower() or toupper() is used to create a unified set of texts.
  - However, be aware that there are some cases where texts have to be written differently, such as someone’s name, a brand name, a country name, etc.



```
> c("Apple","apple","APPLE") %>%
+   tolower()
[1] "apple" "apple" "apple"
> c("Apple","apple","APPLE") %>%
+   toupper()
[1] "APPLE" "APPLE" "APPLE"
> c("Kim","Kimchi","Kimbob","Kim's club") %>%
+   tolower()
[1] "kim"          "kimchi"      "kimbob"      "kim's club"
> c("Kim","Kimchi","Kimbob","Kim's club") %>%
+   toupper()
[1] "KIM"          "KIMCHI"      "KIMBOB"      "KIM'S CLUB"
```

Text Conversion

- sub()
  - Substitutes the first searching text
- gsub()
  - Substitutes all searching text

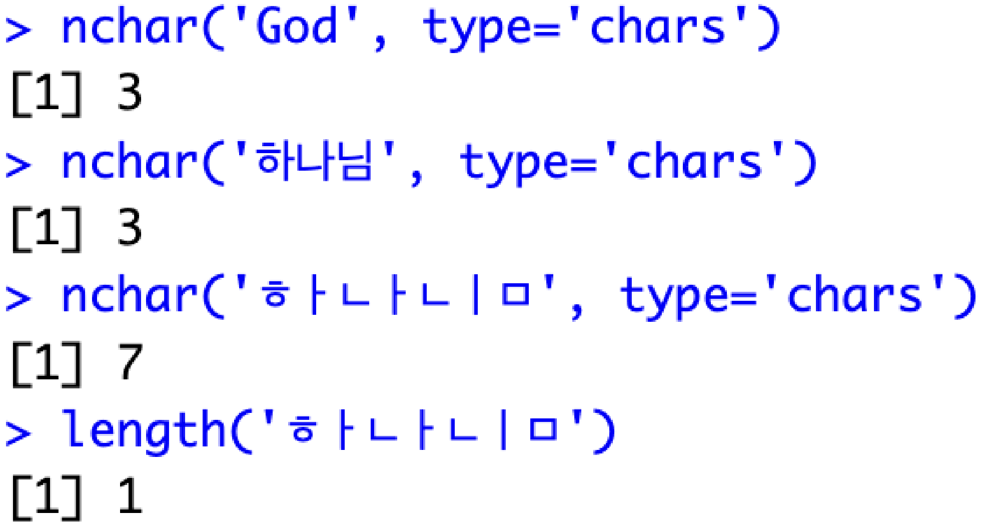


```
> sub('to', 'TO', 'K to the I to the M')
[1] "K TO the I to the M"
> gsub('to', 'TO', 'K to the I to the M')
[1] "K TO the I TO the M"
```

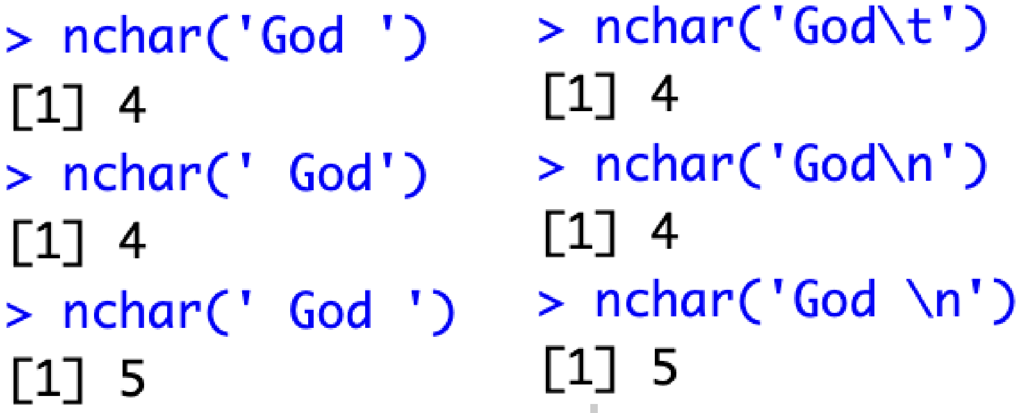
- Text substitution is a technique that is not only used for “correcting” the wrong text, but also for “easing” the text analysis process
  - Proper noun (ex. Son Heung Min, Samsung Electronics, etc.)
  - Stopwords

Text Conversion

- nchar()
  - Counts number of characters
  - While length() is used to count the number of elements, nchar() allows us to measure *the length of words*
  - **Counts “ ” as a text →** can be used for counting the number of spaces
  - \t: tab
  - \n: line breaking



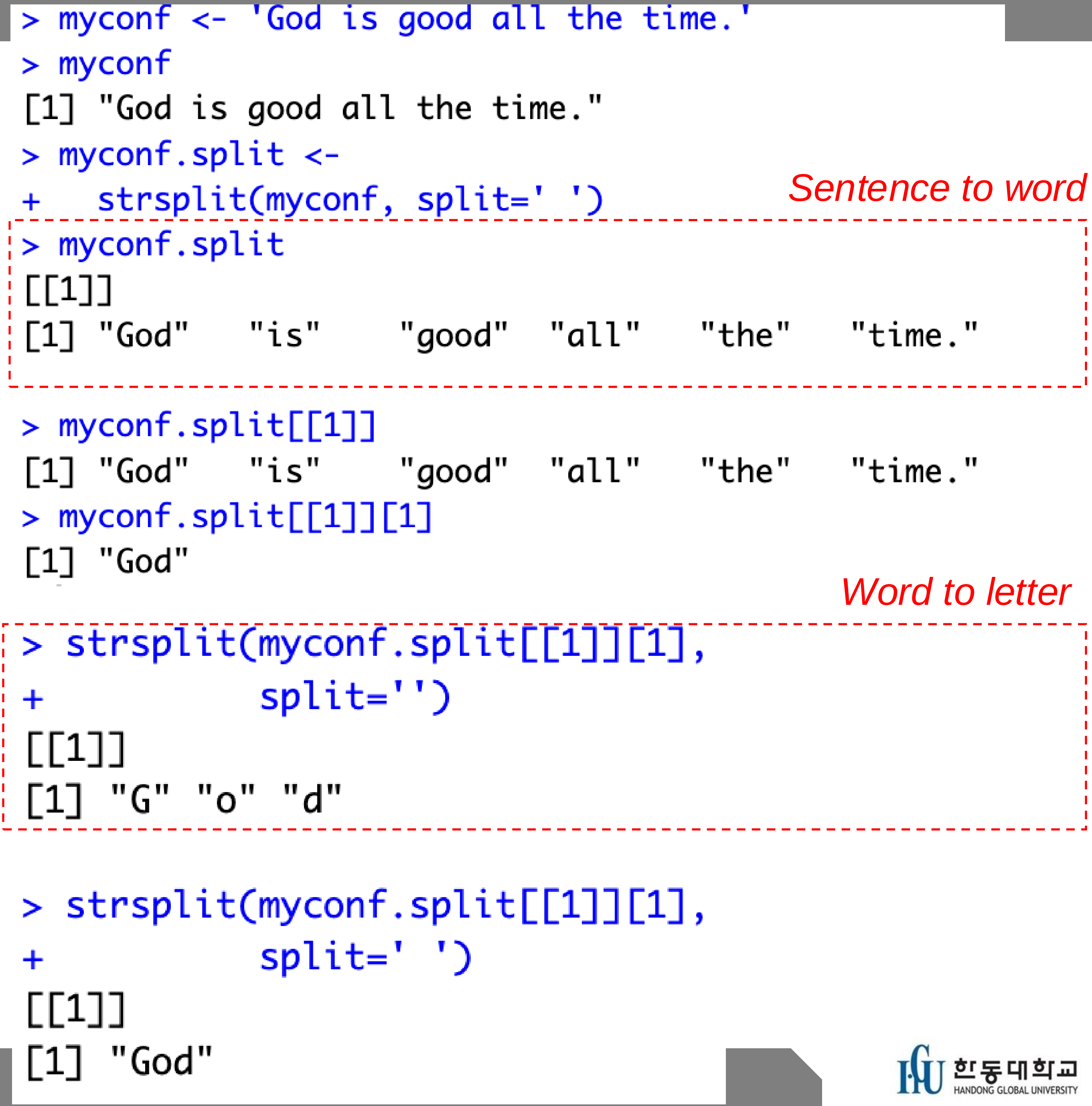
```
> nchar('God', type='chars')
[1] 3
> nchar('하나님', type='chars')
[1] 3
> nchar('ㅎㅏㄴㅏㄴㅣㅁ', type='chars')
[1] 7
> length('ㅎㅏㄴㅏㄴㅣㅁ')
[1] 1
```



```
> nchar('God ')         > nchar('God\t')
[1] 4                   [1] 4
> nchar(' God')         > nchar('God\n')
[1] 4                   [1] 4
> nchar(' God ')        > nchar('God \n')
[1] 5                   [1] 5
```

Text Separation

- strsplit()
  - String split
  - Split the elements of a character vector x into substrings according to the matches to substrings split within them.
  - Returns a list
  - split: regular expression for splitting



```
> myconf <- 'God is good all the time.'
> myconf
[1] "God is good all the time."
> myconf.split <-
+   strsplit(myconf, split=' ')
```

Sentence to word

```
> myconf.split
[[1]]
[1] "God"   "is"    "good"  "all"   "the"   "time."
```

```
> myconf.split[[1]]
[1] "God"   "is"    "good"  "all"   "the"   "time."
> myconf.split[[1]][1]
[1] "God"
```

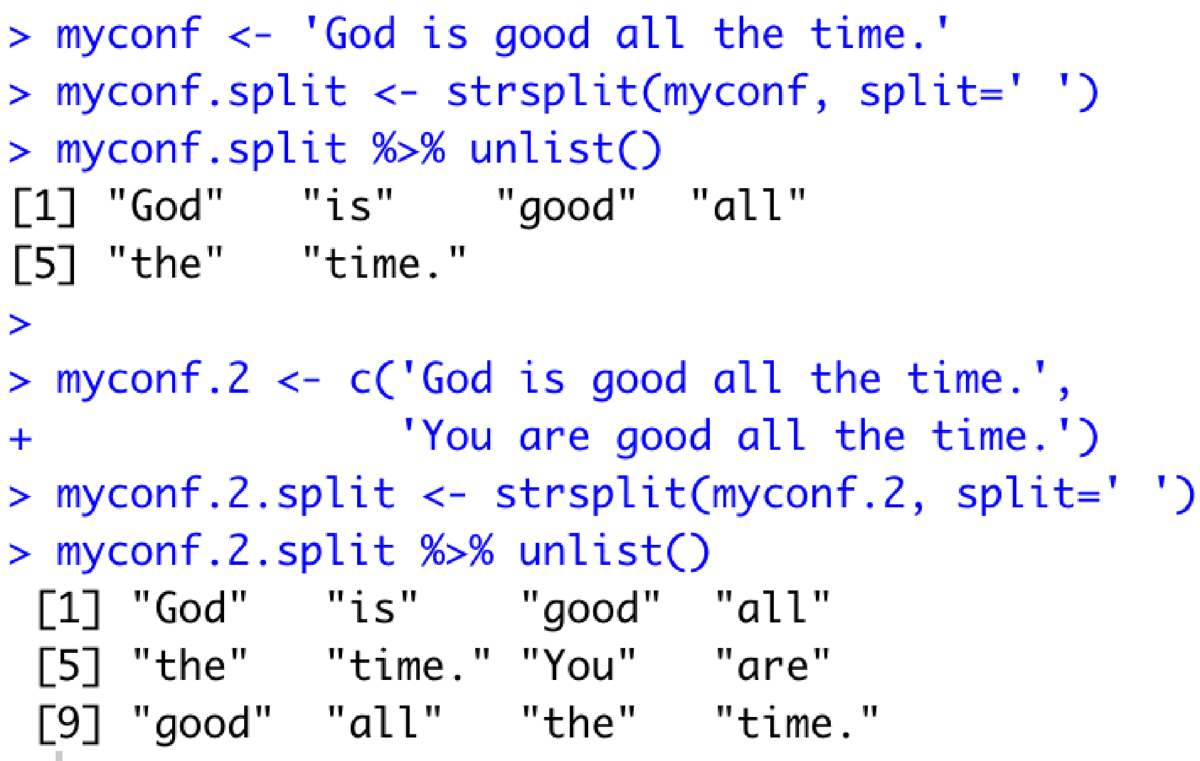
Word to letter

```
> strsplit(myconf.split[[1]][1],
+          split='')
[[1]]
[1] "G" "o" "d"
```

```
> strsplit(myconf.split[[1]][1],
+          split=' ')
[[1]]
[1] "God"
```

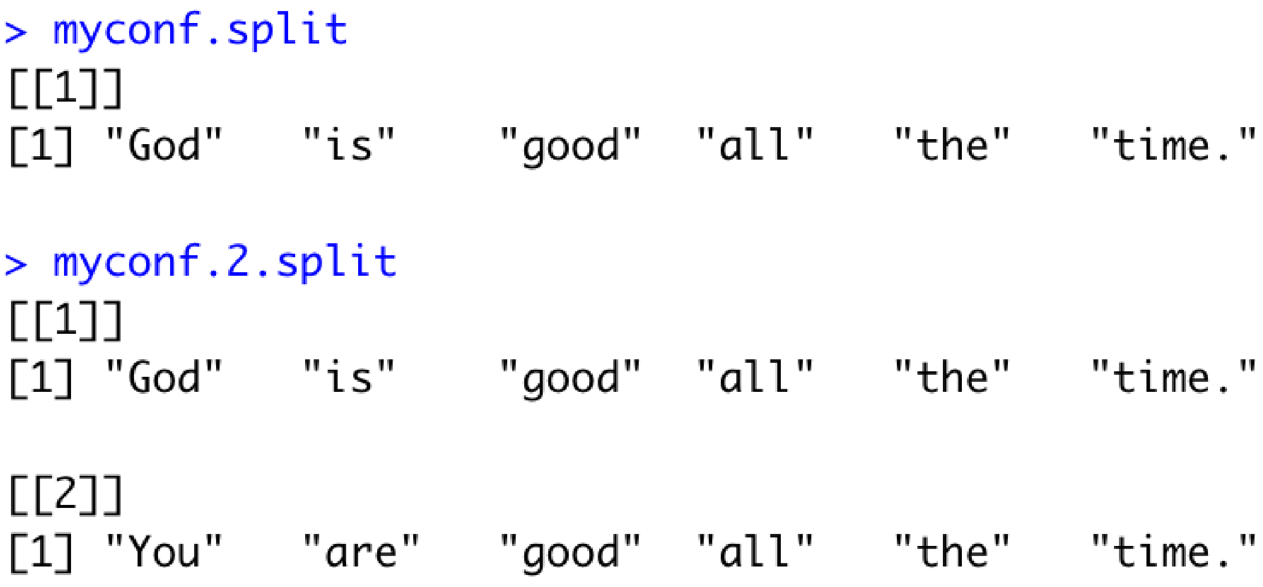
Text Separation

- strsplit() + unlist
  - By combining strsplit and unlist, we can create a vector of words



```
> myconf <- 'God is good all the time.'
> myconf.split <- strsplit(myconf, split=' ')
> myconf.split %>% unlist()
[1] "God"   "is"    "good"  "all"
[5] "the"   "time."
>
> myconf.2 <- c('God is good all the time.',
+               'You are good all the time.')
> myconf.2.split <- strsplit(myconf.2, split=' ')
> myconf.2.split %>% unlist()
 [1] "God"   "is"    "good"  "all"
 [5] "the"   "time." "You"   "are"
 [9] "good"  "all"   "the"   "time."
```

Useful if we want to manage all words without the consideration of the "origin"



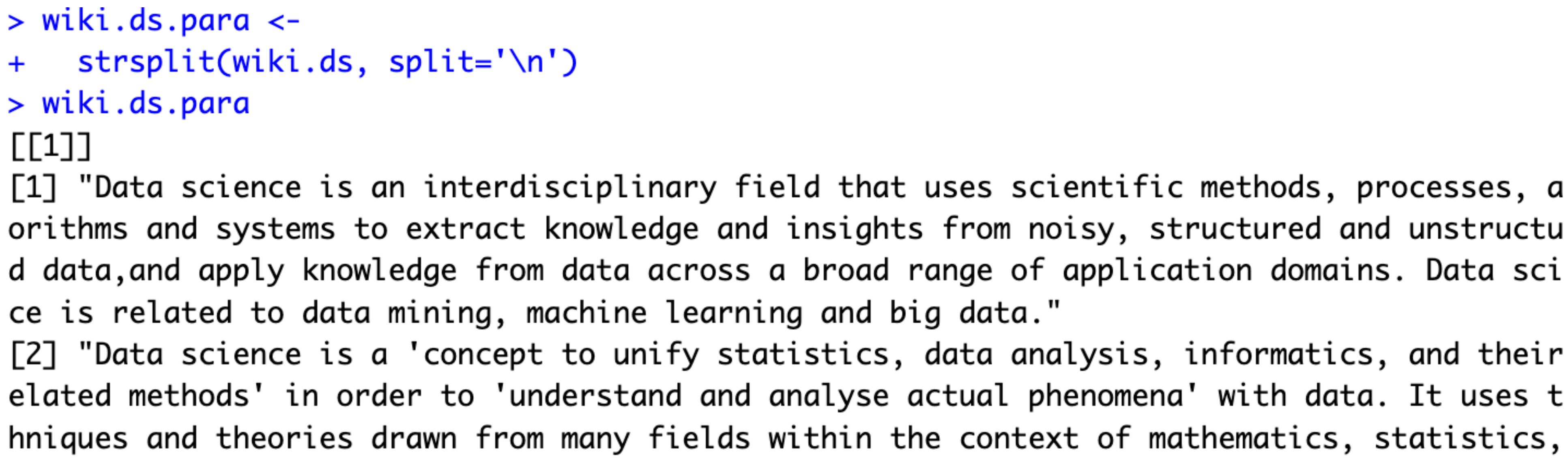
```
> myconf.split
[[1]]
[1] "God"   "is"    "good"  "all"   "the"   "time."

> myconf.2.split
[[1]]
[1] "God"   "is"    "good"  "all"   "the"   "time."

[[2]]
[1] "You"   "are"   "good"  "all"   "the"   "time."
```

Useful if we want to manage words by its "origin"

strsplit() with Paragraphs

- Divide text by paragraph
  - How do we distinguish a paragraph?
  - strsplit(X, split = ‘\n’)

```
wiki.ds <-
"Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to
extract knowledge and insights from noisy, structured and unstructured data,and apply knowledge from data
across a broad range of application domains. Data science is related to data mining, machine learning and big
data.
Data science is a 'concept to unify statistics, data analysis, informatics, and their related methods' in order to
'understand and analyse actual phenomena' with data. It uses techniques and theories drawn from many
fields within the context of mathematics, statistics, computer science, information science, and domain
knowledge. However, data science is different from computer science and information science. Turing Award
winner Jim Gray imagined data science as a 'fourth paradigm' of science (empirical, theoretical, computational,
and now data-driven) and asserted that 'everything about science is changing because of the impact of
information technology' and the data deluge.
A data scientist is someone who creates programming code and combines it with statistical knowledge to
create insights from data."
```



```
> wiki.ds.para <-
+   strsplit(wiki.ds, split='\n')
> wiki.ds.para
[[1]]
[1] "Data science is an interdisciplinary field that uses scientific methods, processes, a
orithms and systems to extract knowledge and insights from noisy, structured and unstructu
d data,and apply knowledge from data across a broad range of application domains. Data sci
ce is related to data mining, machine learning and big data."
[2] "Data science is a 'concept to unify statistics, data analysis, informatics, and their
elated methods' in order to 'understand and analyse actual phenomena' with data. It uses t
hniques and theories drawn from many fields within the context of mathematics, statistics,
```

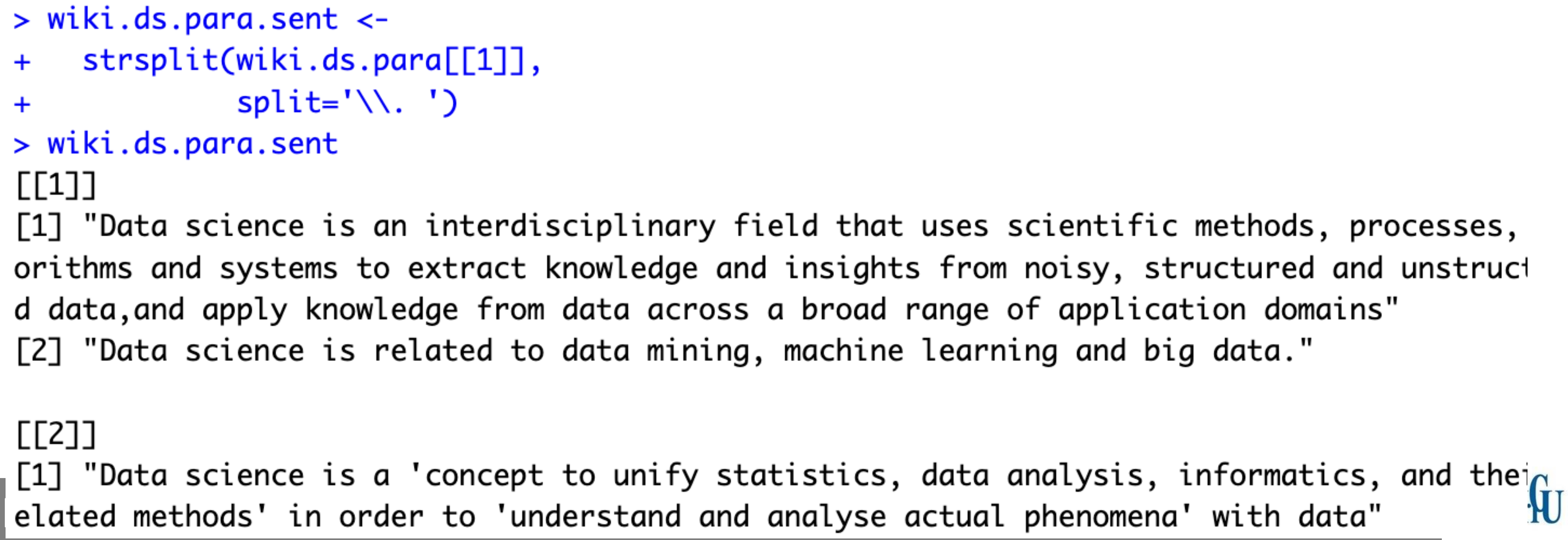
strsplit() with Paragraphs

- Divide text by sentence
  - How do we distinguish a sentence?
  - strsplit(X, split= ‘\\. ’)

Why do we need space?

wiki.ds <-
"Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from noisy, structured and unstructured data,and apply knowledge from data across a broad range of application domains. Data science is related to data mining, machine learning and big data.
Data science is a 'concept to unify statistics, data analysis, informatics, and their related methods' in order to 'understand and analyse actual phenomena' with data. It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, information science, and domain knowledge. However, data science is different from computer science and information science. Turing Award winner Jim Gray imagined data science as a 'fourth paradigm' of science (empirical, theoretical, computational, and now data-driven) and asserted that 'everything about science is changing because of the impact of information technology' and the data deluge.
A data scientist is someone who creates programming code and combines it with statistical knowledge to create insights from data."

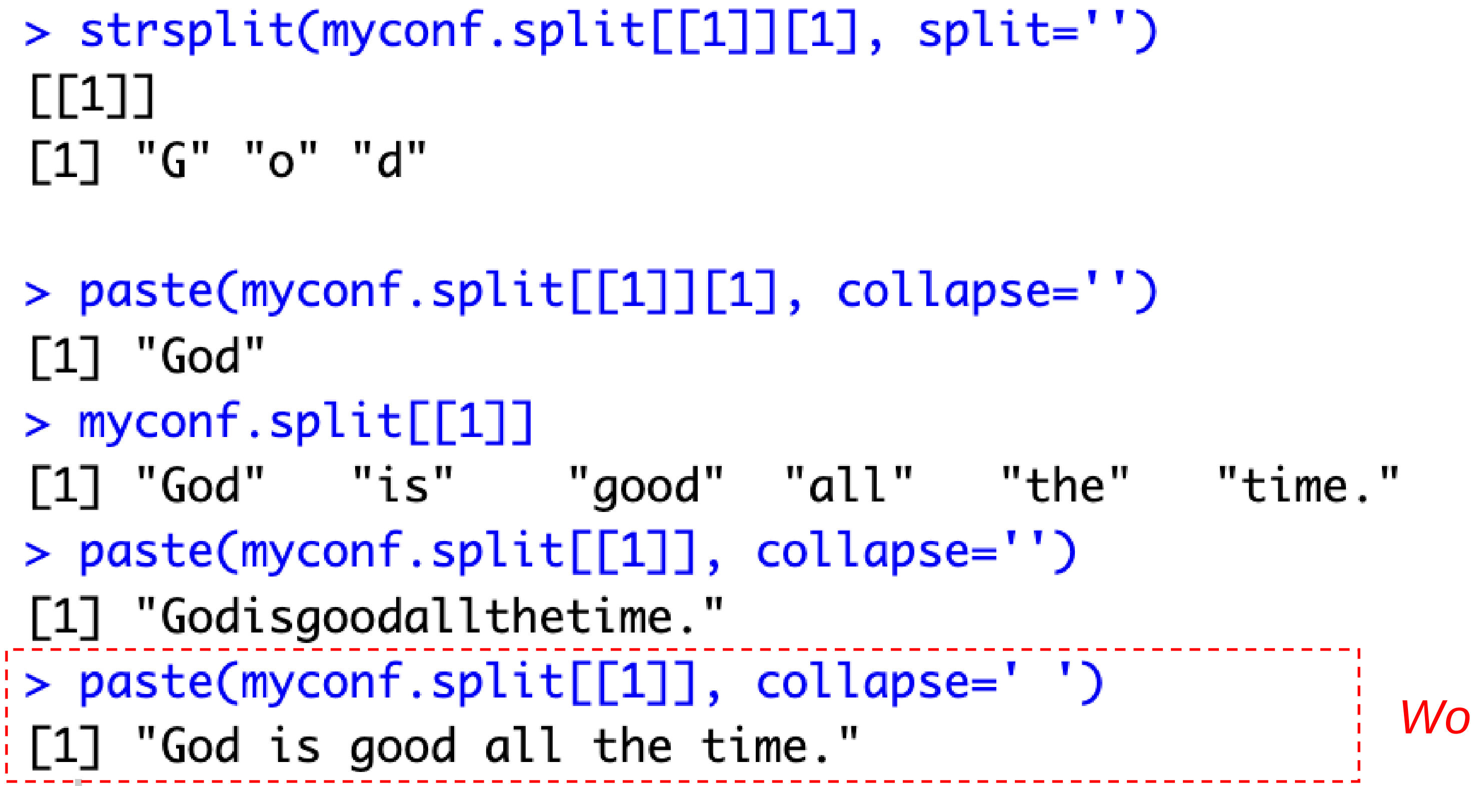
- Whenever using a comma(,) or dot(.), two backslashes(\\) should be used.



```
> wiki.ds.para.sent <-
+   strsplit(wiki.ds.para[[1]],
+            split='\\. ')
> wiki.ds.para.sent
[[1]]
[1] "Data science is an interdisciplinary field that uses scientific methods, processes,
orithms and systems to extract knowledge and insights from noisy, structured and unstruct
d data,and apply knowledge from data across a broad range of application domains"
[2] "Data science is related to data mining, machine learning and big data."

[[2]]
[1] "Data science is a 'concept to unify statistics, data analysis, informatics, and thei
elated methods' in order to 'understand and analyse actual phenomena' with data"
```

Text Integration

- paste()
  - Concatenate vectors after converting to character
  - collapse: optional character string to separate the result
  - The collapse option allows us to merge strings as a *single string*.



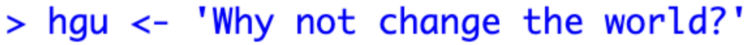
```
> strsplit(myconf.split[[1]][1], split='')
[[1]]
[1] "G" "o" "d"

> paste(myconf.split[[1]][1], collapse='')
[1] "God"
> myconf.split[[1]]
[1] "God"   "is"    "good"  "all"   "the"   "time."
> paste(myconf.split[[1]], collapse='')
[1] "Godisgoodallthetime."
> paste(myconf.split[[1]], collapse=' ')
[1] "God is good all the time."
```

Words to sentence

Key Functions Exercise

Exercise

- (1-1) Count the number of spaces in a variable called ‘hgu’ and save it to a variable called ‘nspace’
- (1-2) Using the for statement, create a list called ‘hgu.str’ which contains each word for each list element
- (1-3) Using ‘hgu.str’, create a string vector, “Why not change the world?”



```
> hgu <- 'Why not change the world?'
```

Exercise

- Create a word vector called “love.para.sent.word”
- What is the fourth paragraph’s last sentence?
- What is the second paragraph’s second sentence’s first word?

```
love <-
"Love encompasses a range of strong and positive emotional and mental states, from the most sublime virtue or good habit, the deepest interpersonal affection, to the simplest pleasure. An example of this range of meanings is that the love of a mother differs from the love of a spouse, which differs from the love for food. Most commonly, love refers to a feeling of a strong attraction and emotional attachment.
Love is considered to be both positive and negative, with its virtue representing human kindness, compassion, and affection, as 'the unselfish loyal and benevolent concern for the good of another' and its vice representing human moral flaw, akin to vanity, selfishness, amour-propre, and egotism, as potentially leading people into a type of mania, obsessiveness or codependency. It may also describe compassionate and affectionate actions towards other humans, one's self, or animals. In its various forms, love acts as a major facilitator of interpersonal relationships and, owing to its central psychological importance, is one of the most common themes in the creative arts. Love has been postulated to be a function that keeps human beings together against menaces and to facilitate the continuation of the species.
Ancient Greek philosophers identified six forms of love: essentially, familial love (in Greek, Storge), friendly love or platonic love (Philia), romantic love (Eros), self-love (Philautia), guest love (Xenia), and divine love (Agape). Modern authors have distinguished further varieties of love: unrequited love, empty love, companionate love, consummate love, infatuated love, self-love, and courtly love. Numerous cultures have also distinguished Ren, Yuanfen, Mamihlapinatapai, Cafuné, Kama, Bhakti, Mettā, Ishq, Chesed, Amore, Charity, Saudade (and other variants or symbioses of these states), as culturally unique words, definitions, or expressions of love in regards to a specified 'moments' currently lacking in the English language.
Scientific research on emotion has increased significantly over the past two decades. The color wheel theory of love defines three primary, three secondary and nine tertiary love styles, describing them in terms of the traditional color wheel. The triangular theory of love suggests 'intimacy, passion and commitment' are core components of love. Love has additional religious or spiritual meaning. This diversity of uses and meanings combined with the complexity of the feelings involved makes love unusually difficult to consistently define, compared to other emotional states."
```

Exercise

- Create a word vector called “love.para.sent.word”
- What is the fourth paragraph’s last sentence?
- What is the second paragraph’s second sentence’s first word?

Exercise

- Create a word vector called “love.para.sent.word”
  - Document → paragraph

Exercise

- Create a word vector called “love.para.sent.word”
  - Paragraph → sentence

Exercise

- Create a word vector called “love.para.sent.word”
  - Sentence → word

Exercise

- What is the fourth paragraph’s last sentence?

- What is the second paragraph’s second sentence’s first word?

Text Exploration

Text Indexing

- In typical data analysis, “indexing” is one of the fundamental tasks
  - With indexing, we can extract and edit the text
  - Text indexing with text vectors
  - Text indexing with text documents

“Text Mining is really fun to learn.”

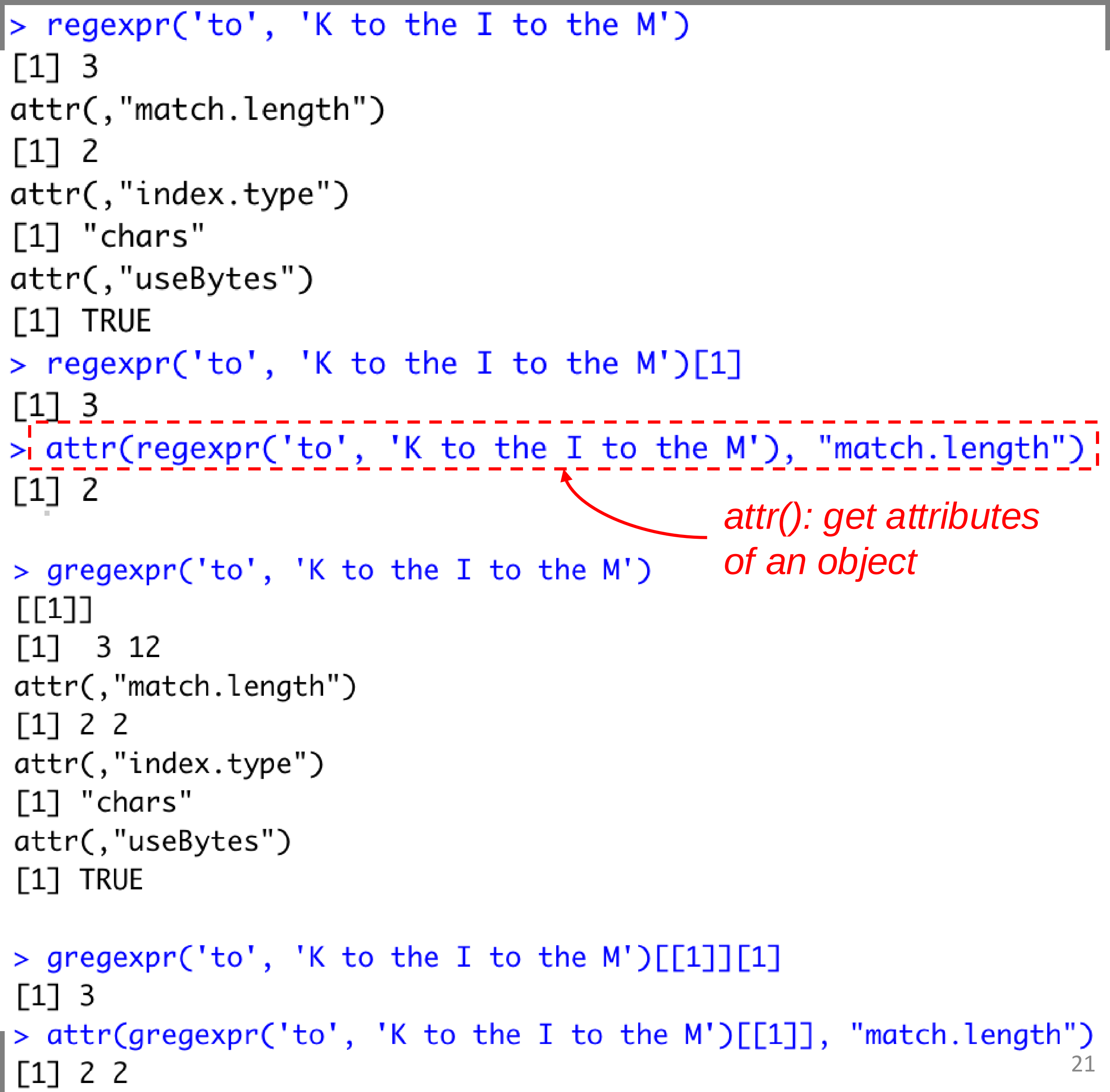
“Text Mining is really fun to learn.”
said prof. Kim.”
“But I knew that he was lying, and
he will give us a full stress.”
“…”

regexpr() & gregexpr()

- regexpr()
  - Returns the *index of the "first" searching text*, attributes for the length of the searching text, the data type, and bytes used

- gregexpr()
  - Global + regexpr()
  - Returns the *index of "all" the searching text*, attributes for the length of searching text, data type, bytes used

- -1 is returned if not found

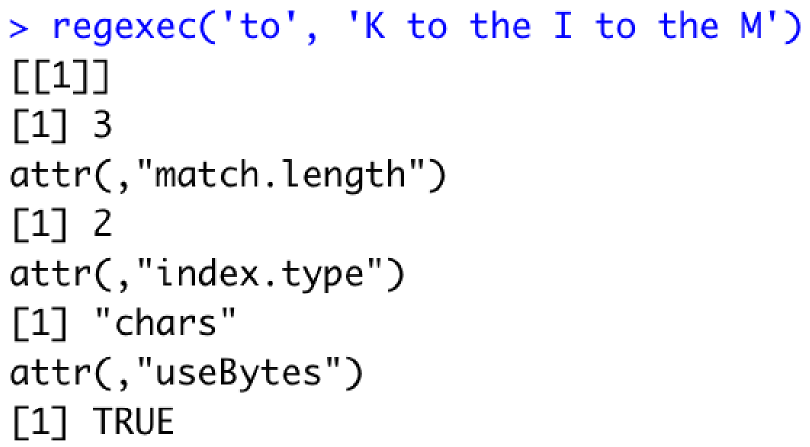


```
> regexpr('to', 'K to the I to the M')
[1] 3
attr(,"match.length")
[1] 2
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
> regexpr('to', 'K to the I to the M')[1]
[1] 3
> attr(regexpr('to', 'K to the I to the M'), "match.length")
[1] 2

> gregexpr('to', 'K to the I to the M')
[[1]]
[1]  3 12
attr(,"match.length")
[1] 2 2
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE

> gregexpr('to', 'K to the I to the M')[[1]][1]
[1] 3
> attr(gregexpr('to', 'K to the I to the M')[[1]], "match.length")
[1] 2 2
```

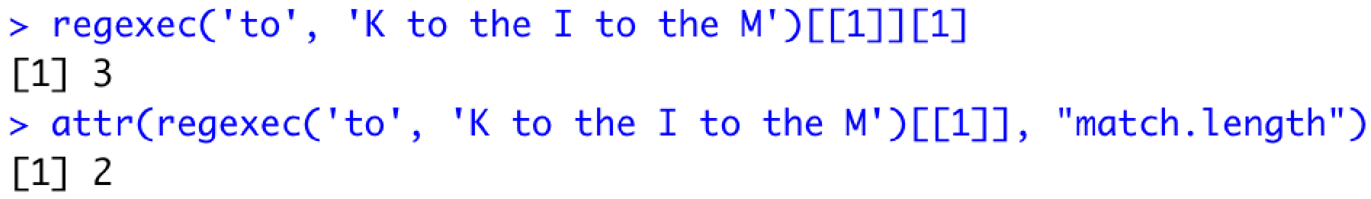
attr(): get attributes of an object

regexec()

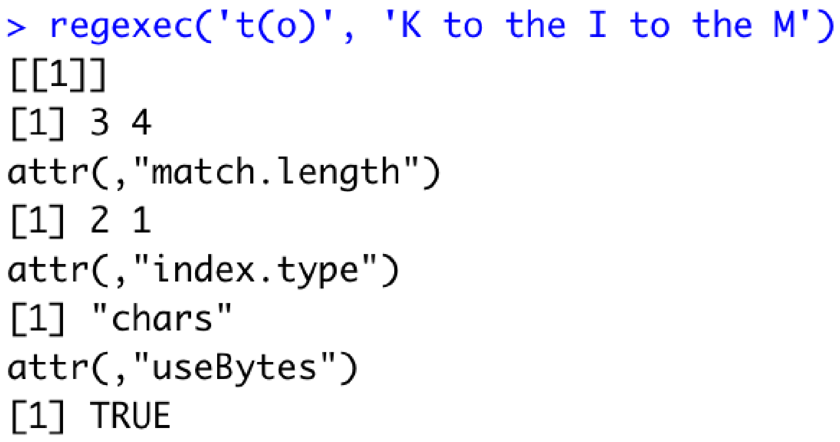
- regexpr() & regexec() returns the similar results
- When using regexec(), ‘()’ can be used to detect additional expressions
  - Especially useful when finding the index of the first and the last letter



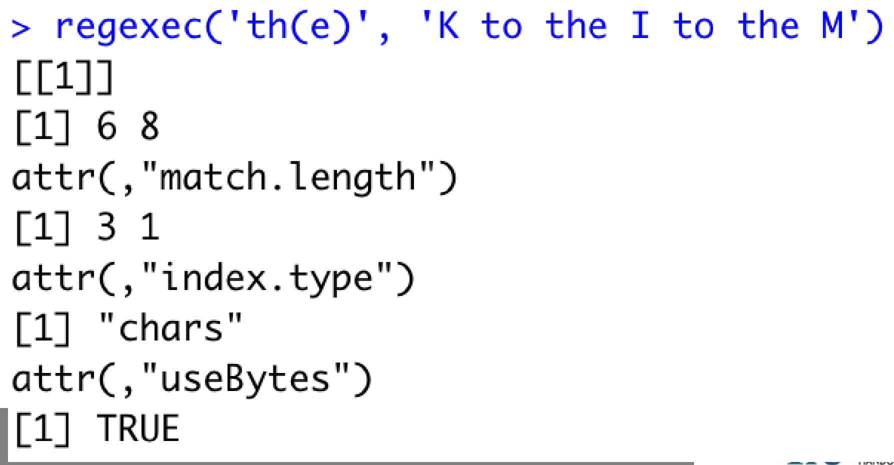
```
> regexec('to', 'K to the I to the M')
[[1]]
[1] 3
attr(,"match.length")
[1] 2
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```



```
> regexec('to', 'K to the I to the M')[[1]][1]
[1] 3
> attr(regexec('to', 'K to the I to the M')[[1]], "match.length")
[1] 2
```



```
> regexec('t(o)', 'K to the I to the M')
[[1]]
[1] 3 4
attr(,"match.length")
[1] 2 1
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

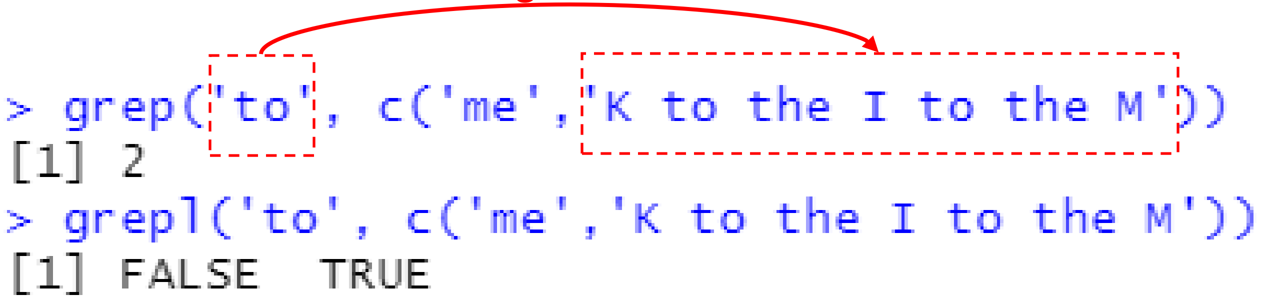


```
> regexec('th(e)', 'K to the I to the M')
[[1]]
[1] 6 8
attr(,"match.length")
[1] 3 1
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

grep() & grepl()

- grep()
  - Returns the index of object containing searched text
- grepl()
  - grep() + logical values
  - Returns TRUE if the object contains the searched text and FALSE otherwise

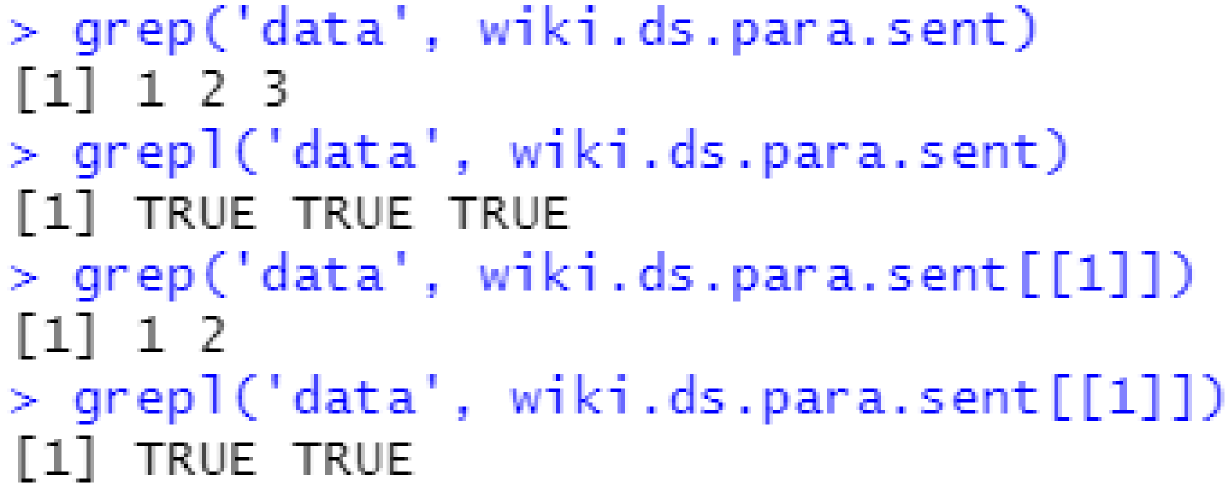
Finding the text



```
> grep('to', c('me','K to the I to the M'))
[1] 2
> grepl('to', c('me','K to the I to the M'))
[1] FALSE  TRUE
```

Find in which paragraph the text can be found → 1st, 2nd, 3rd paragraph

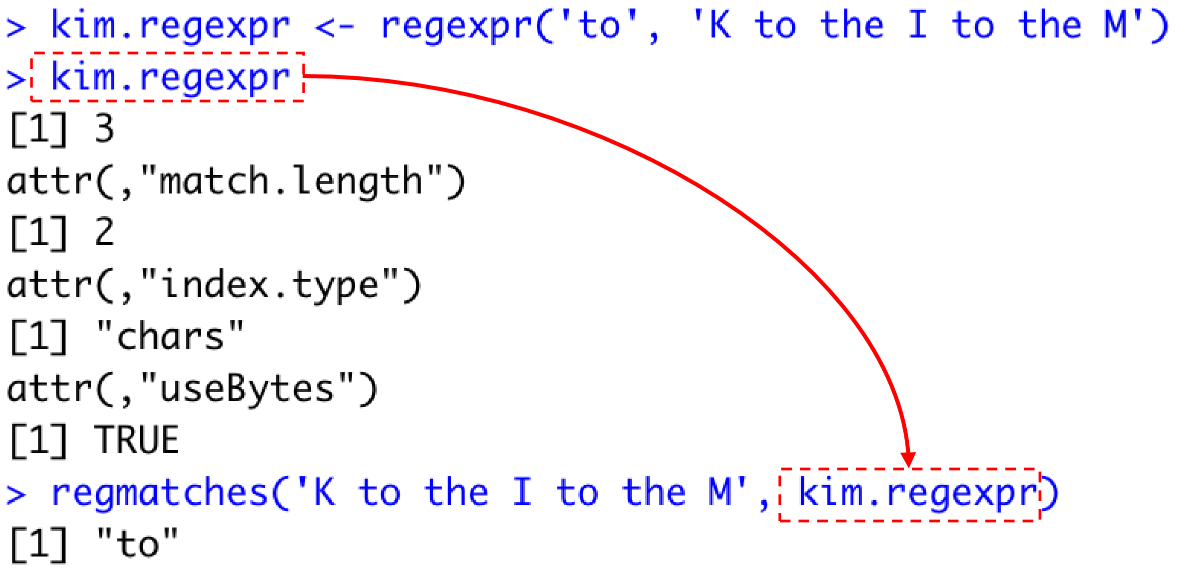
Find in which sentence the text can be found → 1st & 2nd sentence



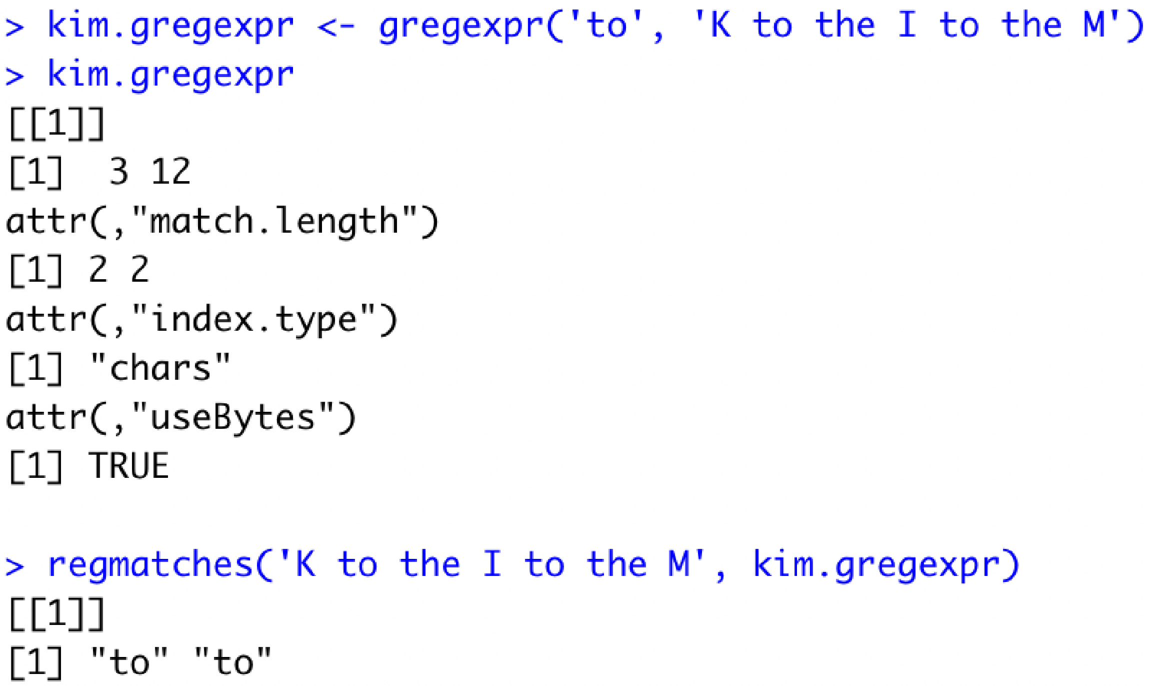
```
> grep('data', wiki.ds.para.sent)
[1] 1 2 3
> grepl('data', wiki.ds.para.sent)
[1] TRUE TRUE TRUE
> grep('data', wiki.ds.para.sent[[1]])
[1] 1 2
> grepl('data', wiki.ds.para.sent[[1]])
[1] TRUE TRUE
```

regmatches()

- regmatches()
  - Extract or replace matched substrings from match data obtained by regexpr, gregexpr, regexec or gregexec
  - Uses regexpr() or gregxpr() objects as an input
  - For us to check whether we found the correct text (ex. “to” → “together”, “today”, etc.)



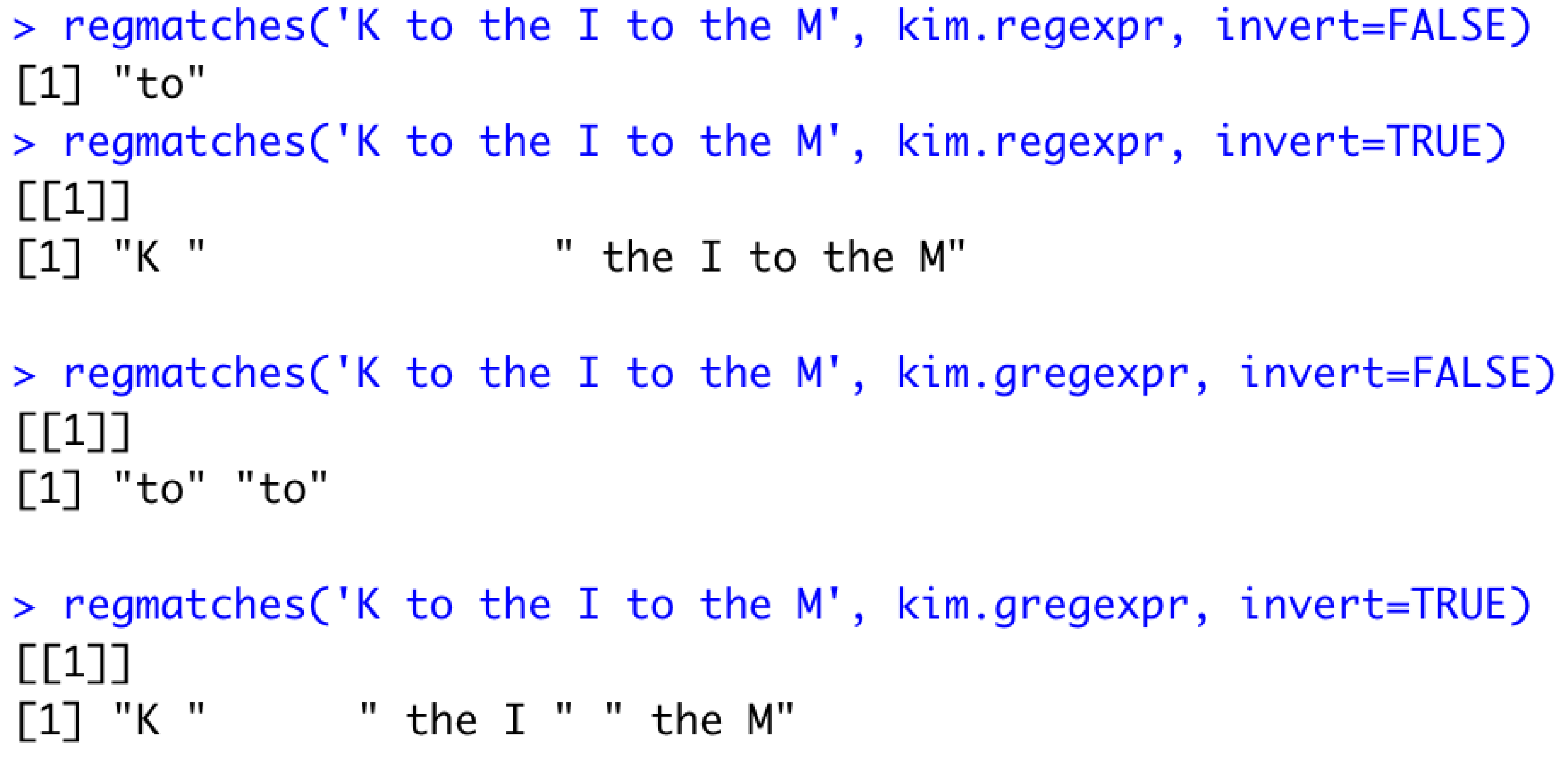
```
> kim.regexpr <- regexpr('to', 'K to the I to the M')
> kim.regexpr
[1] 3
attr(,"match.length")
[1] 2
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
> regmatches('K to the I to the M', kim.regexpr)
[1] "to"
```



```
> kim.gregexpr <- gregexpr('to', 'K to the I to the M')
> kim.gregexpr
[[1]]
[1]  3 12
attr(,"match.length")
[1] 2 2
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE

> regmatches('K to the I to the M', kim.gregexpr)
[[1]]
[1] "to" "to"
```

regmatches()

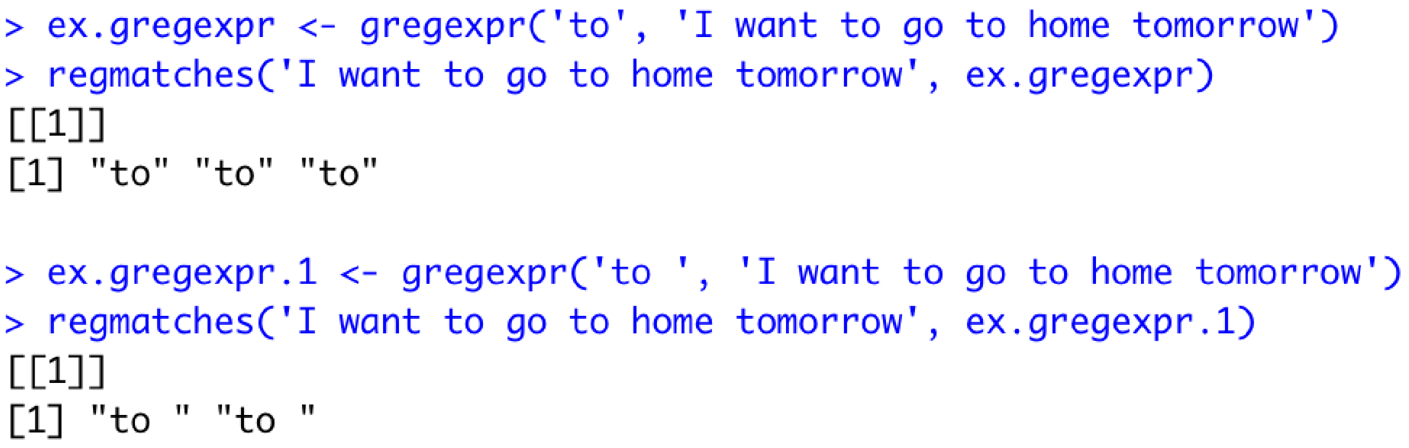
- regmatches()
  - invert option: TRUE or FALSE
  - To find the text ➔ invert = FALSE
  - Find everything except the text ➔ invert = TRUE
  - We can use the invert option to remove meaningless texts



```
> regmatches('K to the I to the M', kim.regexpr, invert=FALSE)
[1] "to"
> regmatches('K to the I to the M', kim.regexpr, invert=TRUE)
[[1]]
[1] "K "                  " the I to the M"

> regmatches('K to the I to the M', kim.gregexpr, invert=FALSE)
[[1]]
[1] "to" "to"

> regmatches('K to the I to the M', kim.gregexpr, invert=TRUE)
[[1]]
[1] "K "       " the I " " the M"
```

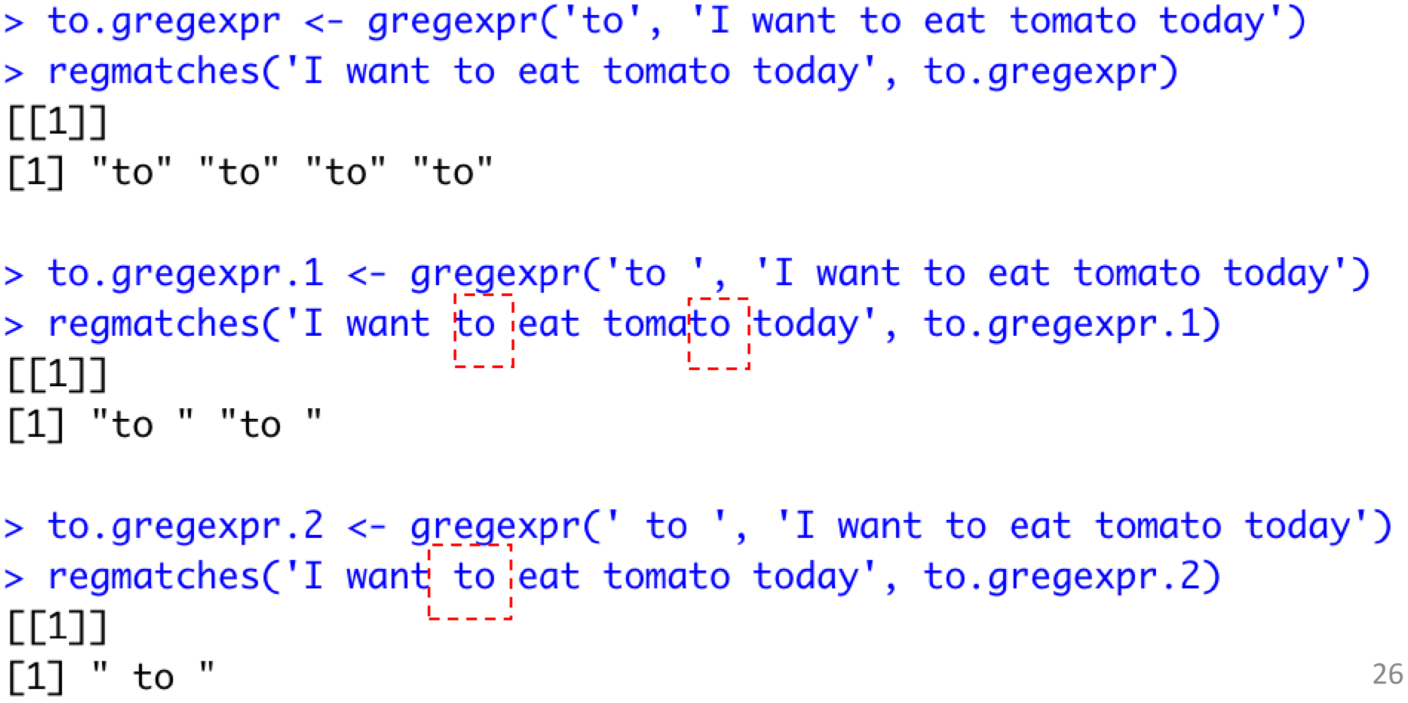
Text Indexing with Patterns

- Find ‘to’(to 부정사) in text, ‘I want to go to home tomorrow’
  - To avoid searching every text containing “to”, we can use the space.



```
> ex.gregexpr <- gregexpr('to', 'I want to go to home tomorrow')
> regmatches('I want to go to home tomorrow', ex.gregexpr)
[[1]]
[1] "to" "to" "to"

> ex.gregexpr.1 <- gregexpr('to ', 'I want to go to home tomorrow')
> regmatches('I want to go to home tomorrow', ex.gregexpr.1)
[[1]]
[1] "to " "to "
```

- Find ‘to’ in ‘I want to eat tomato today’

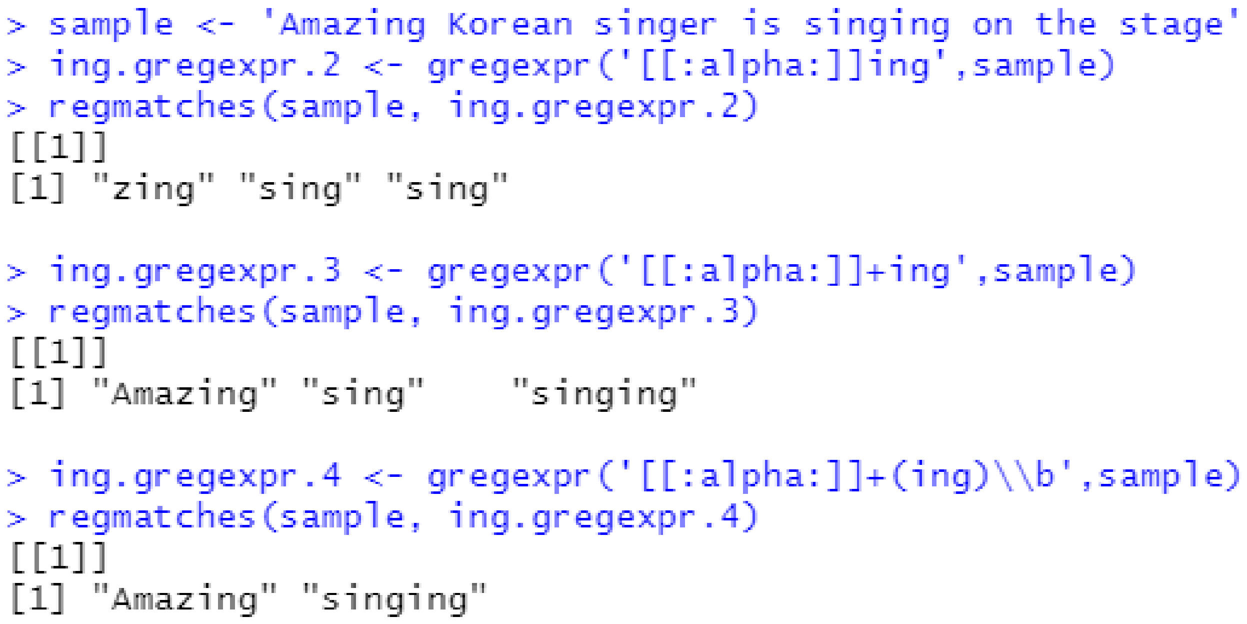


```
> to.gregexpr <- gregexpr('to', 'I want to eat tomato today')
> regmatches('I want to eat tomato today', to.gregexpr)
[[1]]
[1] "to" "to" "to" "to"

> to.gregexpr.1 <- gregexpr('to ', 'I want to eat tomato today')
> regmatches('I want to eat tomato today', to.gregexpr.1)
[[1]]
[1] "to " "to "

> to.gregexpr.2 <- gregexpr(' to ', 'I want to eat tomato today')
> regmatches('I want to eat tomato today', to.gregexpr.2)
[[1]]
[1] " to "
```

Text Indexing with Patterns

- When indexing (or finding) text, recognizing the key pattern is important
  - Unfortunately, patterns are not as simple as we hope…
  - Find ‘ing’ in the text, ‘Amazing Korean singer is singing on the stage’

- Regular expressions for finding patterns
  - [:alpha:] → lower & upper alphabets
  - \b → any expression finishes or ends
  - More will be covered in the next lecture



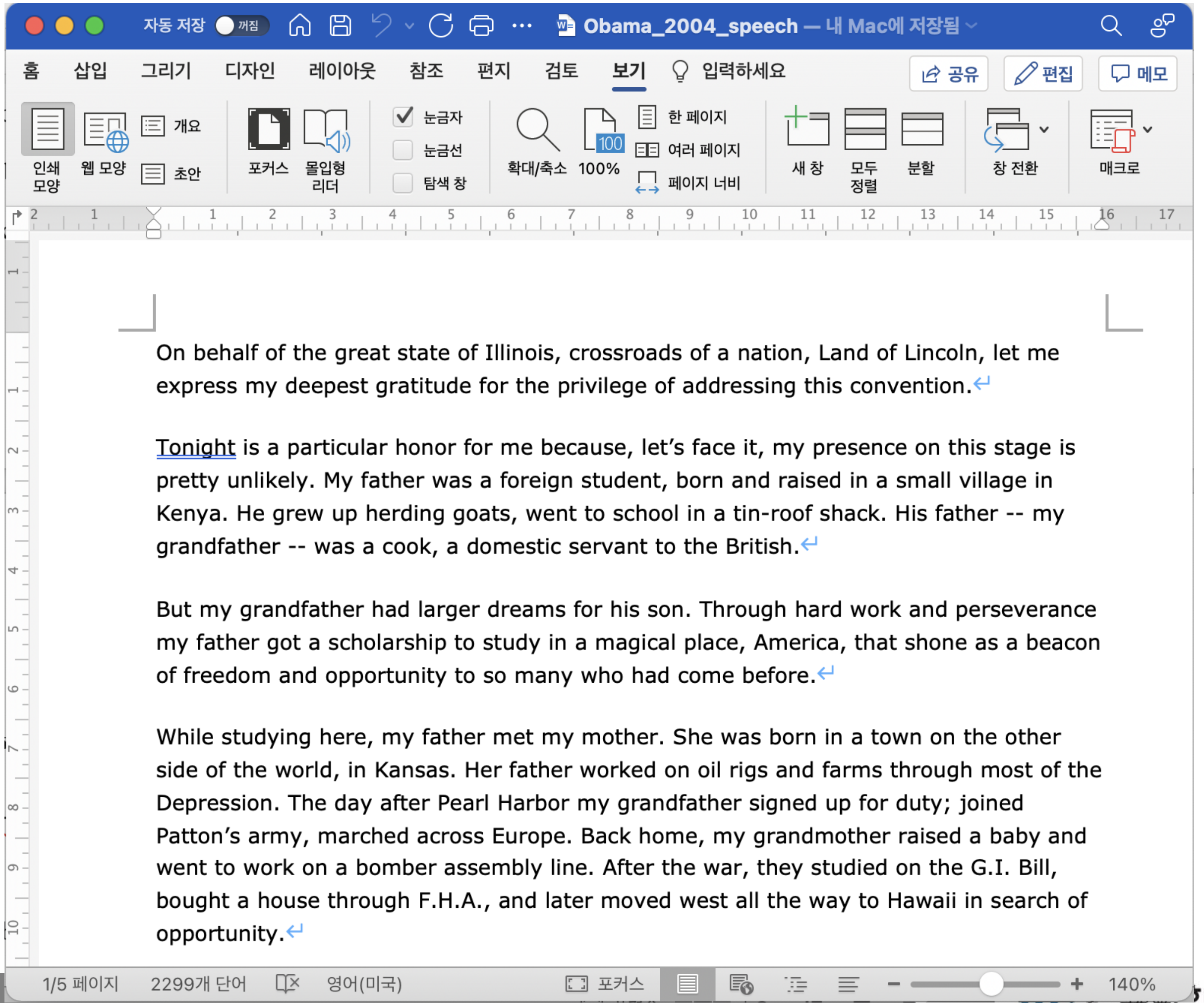
```
> sample <- 'Amazing Korean singer is singing on the stage'
> ing.gregexpr.2 <- gregexpr('[[:alpha:]]ing',sample)
> regmatches(sample, ing.gregexpr.2)
[[1]]
[1] "zing" "sing" "sing"

> ing.gregexpr.3 <- gregexpr('[[:alpha:]]+ing',sample)
> regmatches(sample, ing.gregexpr.3)
[[1]]
[1] "Amazing" "sing"    "singing"

> ing.gregexpr.4 <- gregexpr('[[:alpha:]]+(ing)\\b',sample)
> regmatches(sample, ing.gregexpr.4)
[[1]]
[1] "Amazing" "singing"
```

Text Mining Word Documents

Text Mining Word Documents – Obama Speech

- Load “Obama_2004_speech.docx” to R
  - Barack Obama’s speech at the 2004 Democratic National Convention.

https://www.americanprogress.org/article/obama-at-the-dnc-how-different-is-america-from-what-he-hoped-for-in-2004/



자동 저장 꺼짐
Obama_2004_speech — 내 Mac에 저장됨
홈 삽입 그리기 디자인 레이아웃 참조 편지 검토 보기 입력하세요
공유 편집 메모
인쇄 모양 웹 모양 개요 초안 포커스 몰입형 리더 눈금자 눈금선 탐색 창 확대/축소 100% 한 페이지 여러 페이지 페이지 너비 새 창 모두 정렬 분할 창 전환 매크로
1/5 페이지 2299개 단어 영어(미국) 포커스 140%

On behalf of the great state of Illinois, crossroads of a nation, Land of Lincoln, let me express my deepest gratitude for the privilege of addressing this convention.

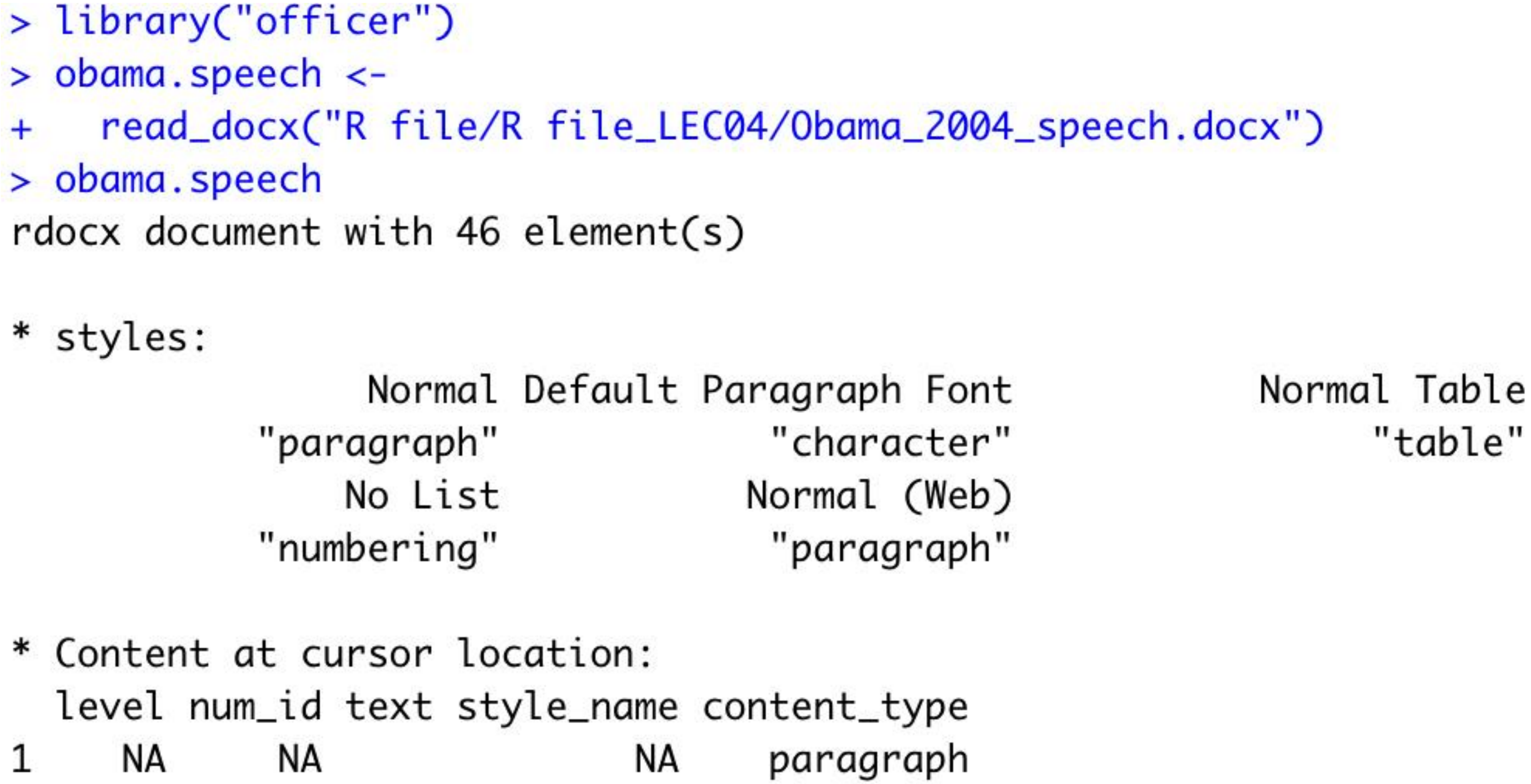
Tonight is a particular honor for me because, let’s face it, my presence on this stage is pretty unlikely. My father was a foreign student, born and raised in a small village in Kenya. He grew up herding goats, went to school in a tin-roof shack. His father -- my grandfather -- was a cook, a domestic servant to the British.

But my grandfather had larger dreams for his son. Through hard work and perseverance my father got a scholarship to study in a magical place, America, that shone as a beacon of freedom and opportunity to so many who had come before.

While studying here, my father met my mother. She was born in a town on the other side of the world, in Kansas. Her father worked on oil rigs and farms through most of the Depression. The day after Pearl Harbor my grandfather signed up for duty; joined Patton’s army, marched across Europe. Back home, my grandmother raised a baby and went to work on a bomber assembly line. After the war, they studied on the G.I. Bill, bought a house through F.H.A., and later moved west all the way to Hawaii in search of opportunity.

Text Mining Word Documents – Obama Speech

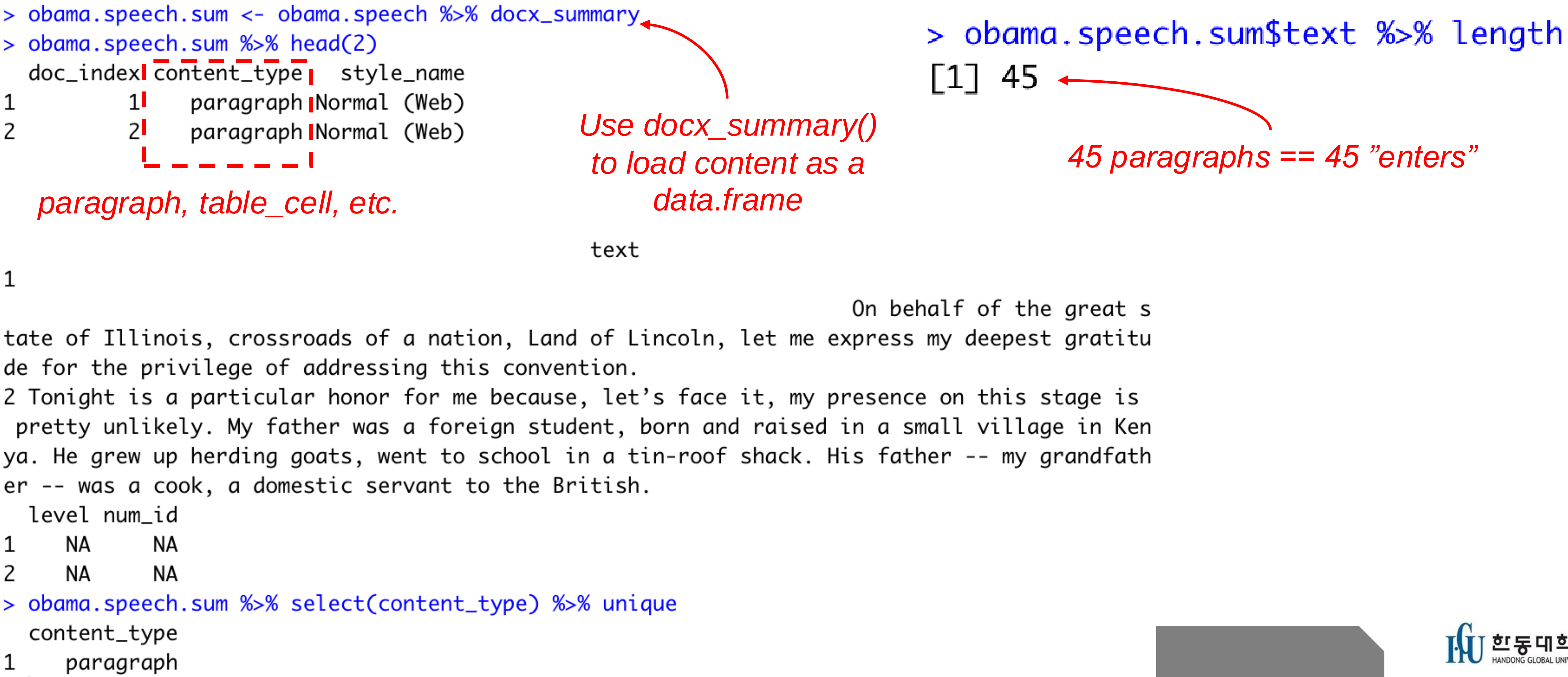
- Load “Obama_2004_speech.docx” to R
  - Use officer::read_docx()
  - officer::read_docx(): loads word document and creates a word document object



```
> library("officer")
> obama.speech <-
+   read_docx("R file/R file_LEC04/Obama_2004_speech.docx")
> obama.speech
rdocx document with 46 element(s)

* styles:
                Normal Default Paragraph Font           Normal Table
           "paragraph"            "character"                "table"
               No List           Normal (Web)
           "numbering"            "paragraph"

* Content at cursor location:
  level num_id text style_name content_type
1    NA     NA              NA    paragraph
```

Text Mining Word Documents – Obama Speech

- Load “Obama_2004_speech.docx” to R
  - officer::docx_summary() : reads the content of a Word document and returns a data.frame representing the document



```
> obama.speech.sum <- obama.speech %>% docx_summary
> obama.speech.sum %>% head(2)
  doc_index content_type   style_name
1         1    paragraph Normal (Web)
2         2    paragraph Normal (Web)





                                            text
1
                                                                    On behalf of the great s
tate of Illinois, crossroads of a nation, Land of Lincoln, let me express my deepest gratitu
de for the privilege of addressing this convention.
2 Tonight is a particular honor for me because, let’s face it, my presence on this stage is
 pretty unlikely. My father was a foreign student, born and raised in a small village in Ken
ya. He grew up herding goats, went to school in a tin-roof shack. His father -- my grandfath
er -- was a cook, a domestic servant to the British.
  level num_id
1    NA     NA
2    NA     NA
> obama.speech.sum %>% select(content_type) %>% unique
  content_type
1    paragraph
```

```
> obama.speech.sum$text %>% length
[1] 45
```

Use docx_summary() to load content as a data.frame

paragraph, table_cell, etc.

45 paragraphs == 45 ”enters”

Exercise

- Find out in which paragraph the word ‘free’ is used.
- Check how the word ‘free’ is used in each case.

Exercise

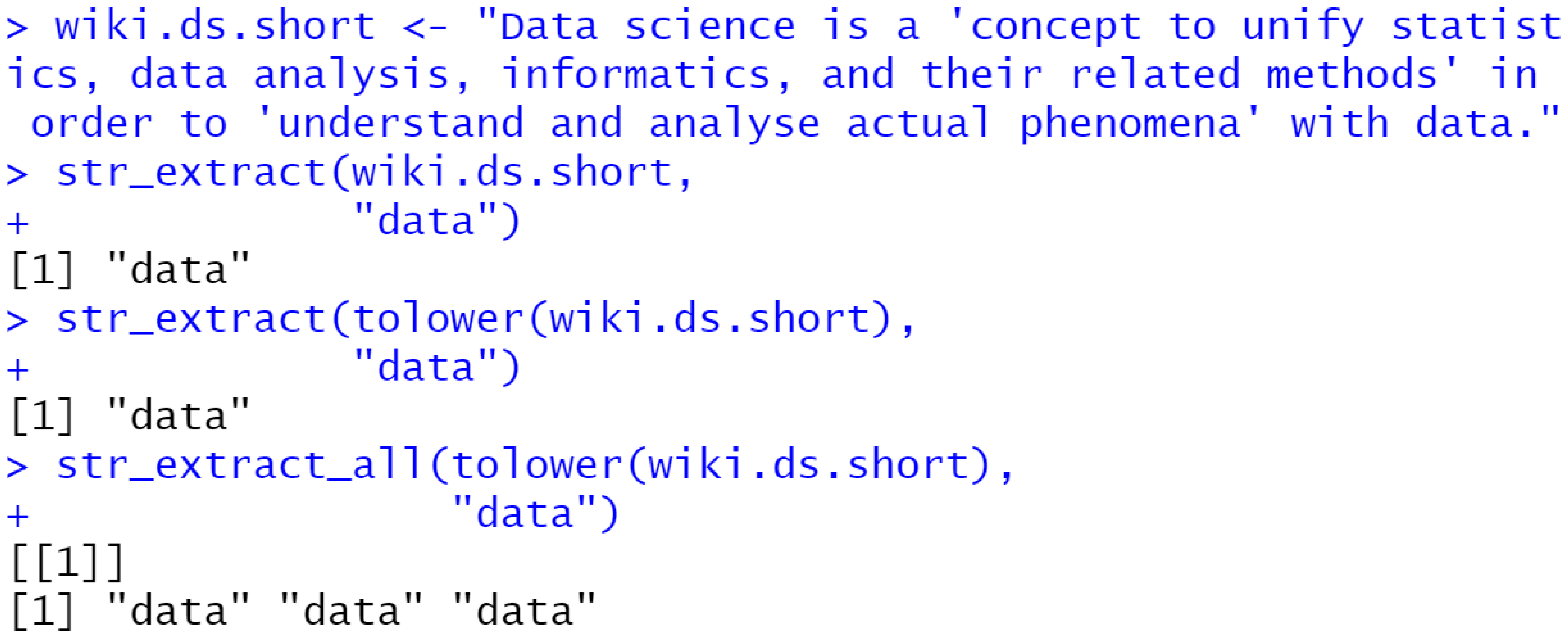
- How many times does the word “America” appear in the speech?

stringr() package

stringr::str_extract() & stringr::str_extract_all()

- str_extract() & str_extract_all()
  - Similar to a combination of regexpr and regmatches
  - str_extract(): extracts first instance
  - str_extract_all(): extracts all instances

wiki.ds.short <- "Data science is a 'concept to unify statistics, data analysis, informatics, and their related methods' in order to 'understand and analyse actual phenomena' with data."

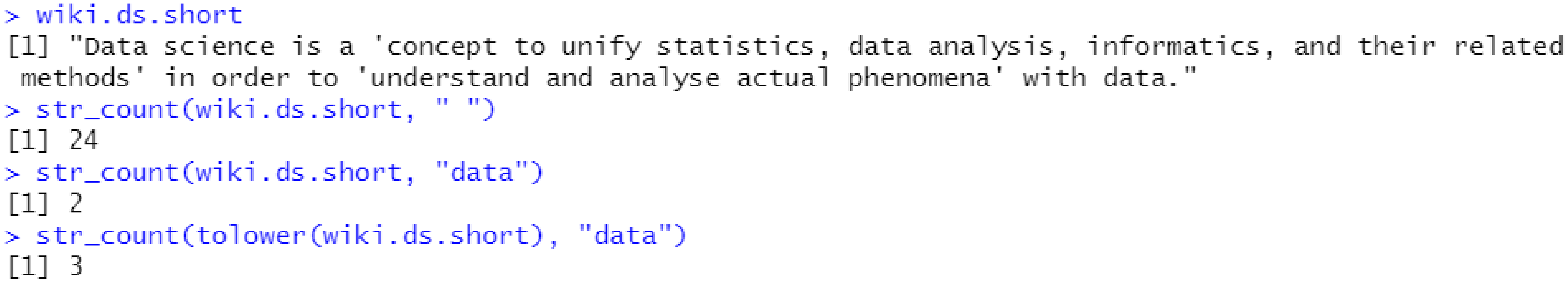


```
> wiki.ds.short <- "Data science is a 'concept to unify statist
ics, data analysis, informatics, and their related methods' in
 order to 'understand and analyse actual phenomena' with data."
> str_extract(wiki.ds.short,
+             "data")
[1] "data"
> str_extract(tolower(wiki.ds.short),
+             "data")
[1] "data"
> str_extract_all(tolower(wiki.ds.short),
+                 "data")
[[1]]
[1] "data" "data" "data"
```

stringr::str_count()

```
wiki.ds.short <- "Data science is a 'concept to unify statistics, data analysis, informatics, and their related methods' in order to 'understand and analyse actual phenomena' with data."
```

- str_count()
  - Counts the number of matches
  - Allows us to find how many words are included in the text

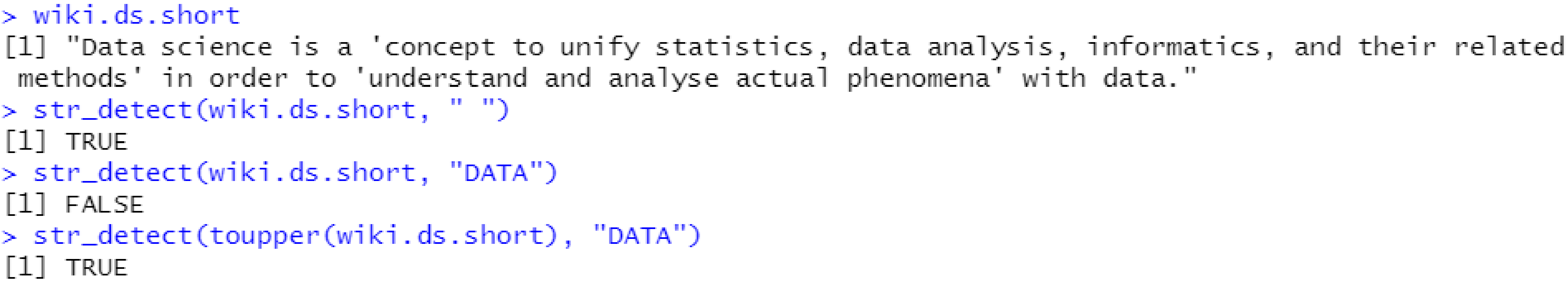


```
> wiki.ds.short
[1] "Data science is a 'concept to unify statistics, data analysis, informatics, and their related
 methods' in order to 'understand and analyse actual phenomena' with data."
> str_count(wiki.ds.short, " ")
[1] 24
> str_count(wiki.ds.short, "data")
[1] 2
> str_count(tolower(wiki.ds.short), "data")
[1] 3
```

stringr::str_detect()

wiki.ds.short <- "Data science is a 'concept to unify statistics, data analysis, informatics, and their related methods' in order to 'understand and analyse actual phenomena' with data."

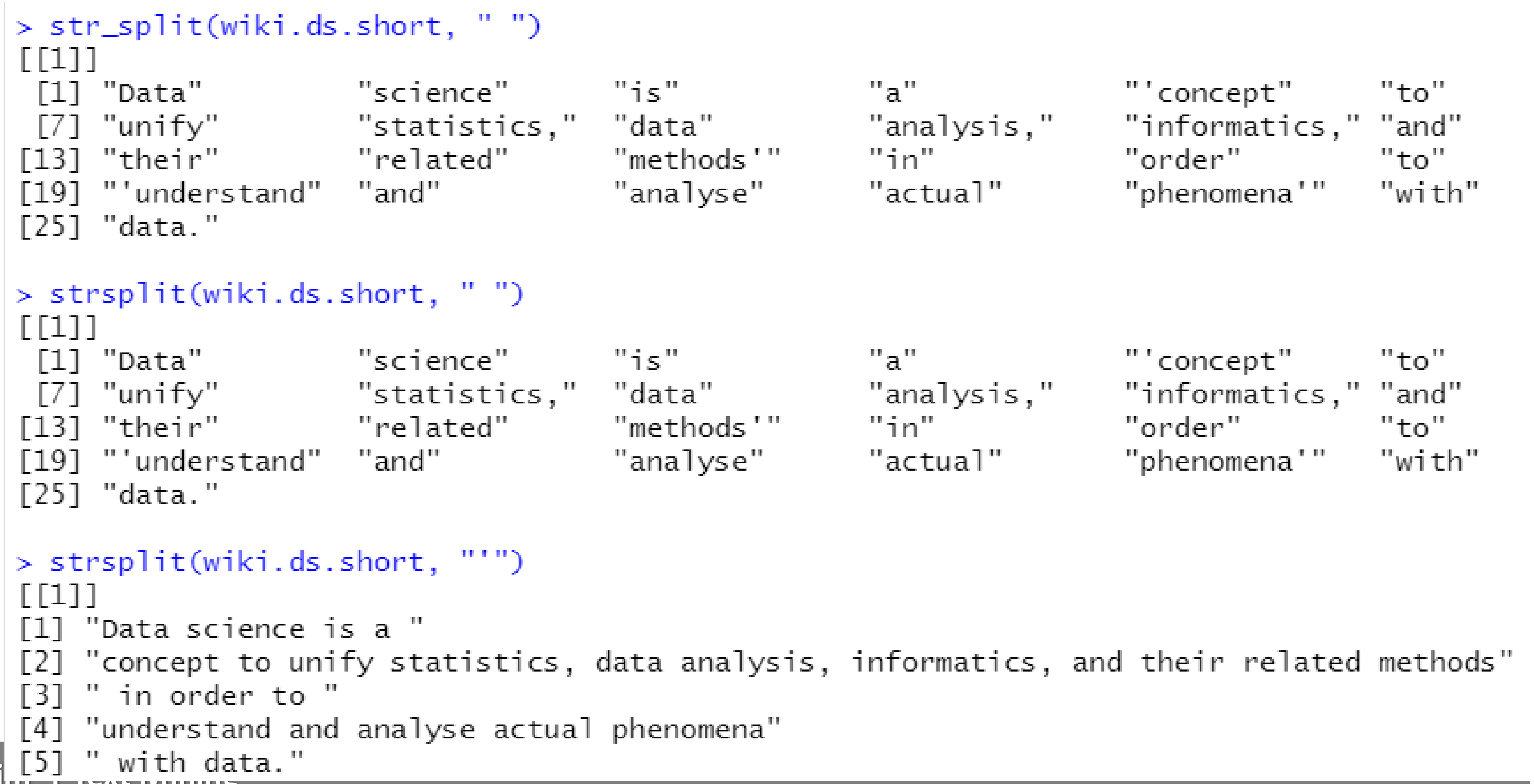
- str_detect()
  - Detect the presence or absence of a pattern in a string
  - Returns TRUE if detected, FALSE otherwise



```
> wiki.ds.short
[1] "Data science is a 'concept to unify statistics, data analysis, informatics, and their related
 methods' in order to 'understand and analyse actual phenomena' with data."
> str_detect(wiki.ds.short, " ")
[1] TRUE
> str_detect(wiki.ds.short, "DATA")
[1] FALSE
> str_detect(toupper(wiki.ds.short), "DATA")
[1] TRUE
```

stringr::str_split()

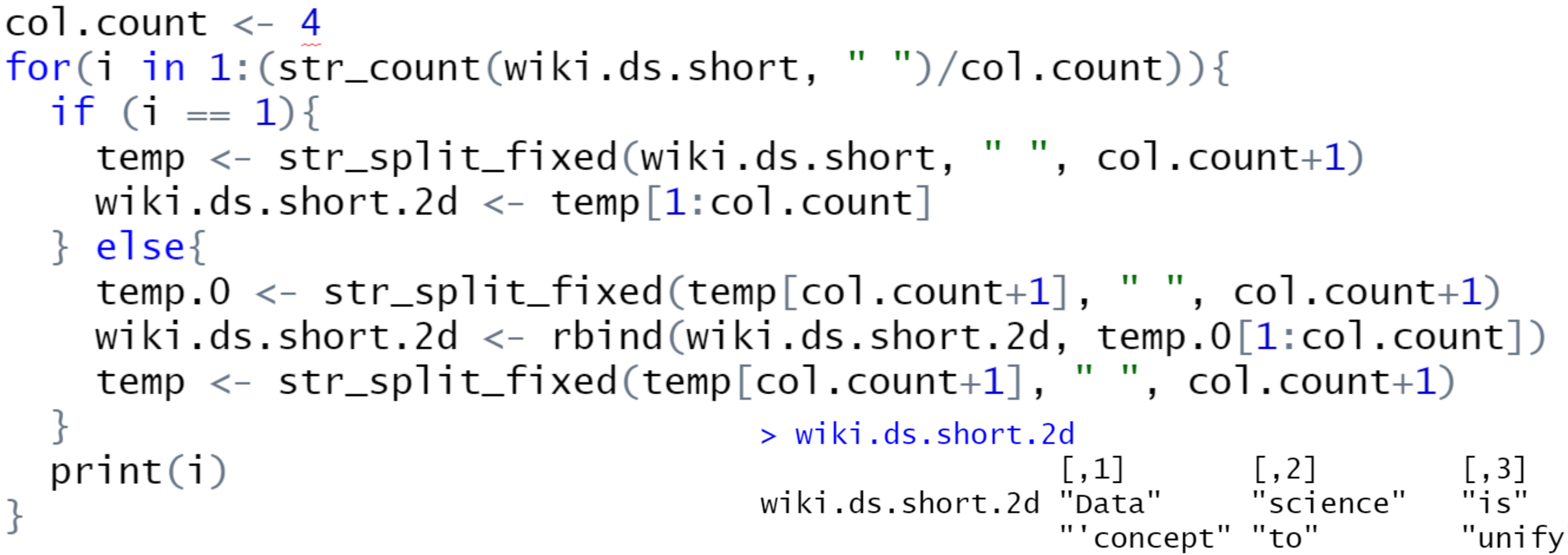
- str_split() *
  - Split up a string into pieces
  - Similar to strsplit() → same



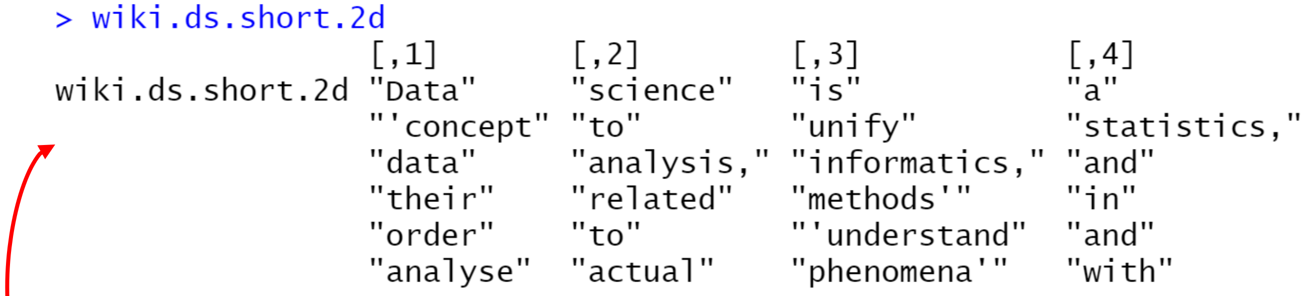
```
> str_split(wiki.ds.short, " ")
[[1]]
 [1] "Data"         "science"      "is"           "a"            "'concept"     "to"
 [7] "unify"        "statistics,"  "data"         "analysis,"    "informatics," "and"
[13] "their"        "related"      "methods'"     "in"           "order"        "to"
[19] "'understand"  "and"          "analyse"      "actual"       "phenomena'"   "with"
[25] "data."

> strsplit(wiki.ds.short, " ")
[[1]]
 [1] "Data"         "science"      "is"           "a"            "'concept"     "to"
 [7] "unify"        "statistics,"  "data"         "analysis,"    "informatics," "and"
[13] "their"        "related"      "methods'"     "in"           "order"        "to"
[19] "'understand"  "and"          "analyse"      "actual"       "phenomena'"   "with"
[25] "data."

> strsplit(wiki.ds.short, "'")
[[1]]
[1] "Data science is a "
[2] "concept to unify statistics, data analysis, informatics, and their related methods"
[3] " in order to "
[4] "understand and analyse actual phenomena"
[5] " with data."
```

stringr::str_split_fixed()

- str_split_fixed()
  - Split up a string into a fixed number of pieces → Similar to strsplit()
  - Using str_split_fixed(), we can create a fixed data format of text

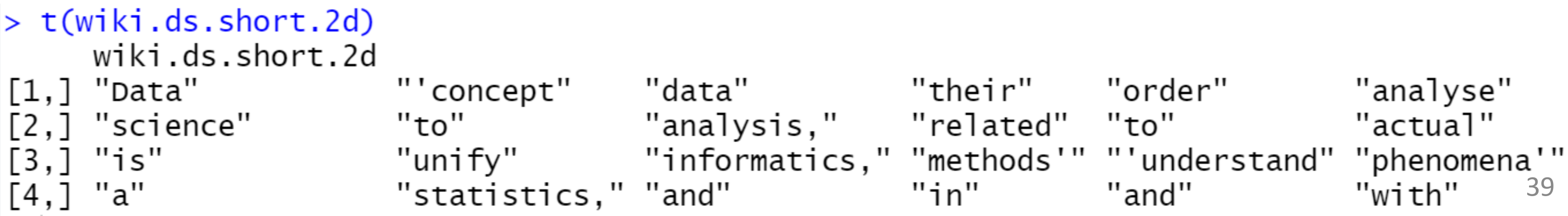


```
col.count <- 4
for(i in 1:(str_count(wiki.ds.short, " ")/col.count)){
  if (i == 1){
    temp <- str_split_fixed(wiki.ds.short, " ", col.count+1)
    wiki.ds.short.2d <- temp[1:col.count]
  } else{
    temp.0 <- str_split_fixed(temp[col.count+1], " ", col.count+1)
    wiki.ds.short.2d <- rbind(wiki.ds.short.2d, temp.0[1:col.count])
    temp <- str_split_fixed(temp[col.count+1], " ", col.count+1)
  }
  print(i)
}
```



```
> wiki.ds.short.2d
                 [,1]       [,2]        [,3]            [,4]
wiki.ds.short.2d "Data"     "science"   "is"            "a"
                 "'concept" "to"        "unify"         "statistics,"
                 "data"     "analysis," "informatics,"  "and"
                 "their"    "related"   "methods'"      "in"
                 "order"    "to"        "'understand"   "and"
                 "analyse"  "actual"    "phenomena'"    "with"
```

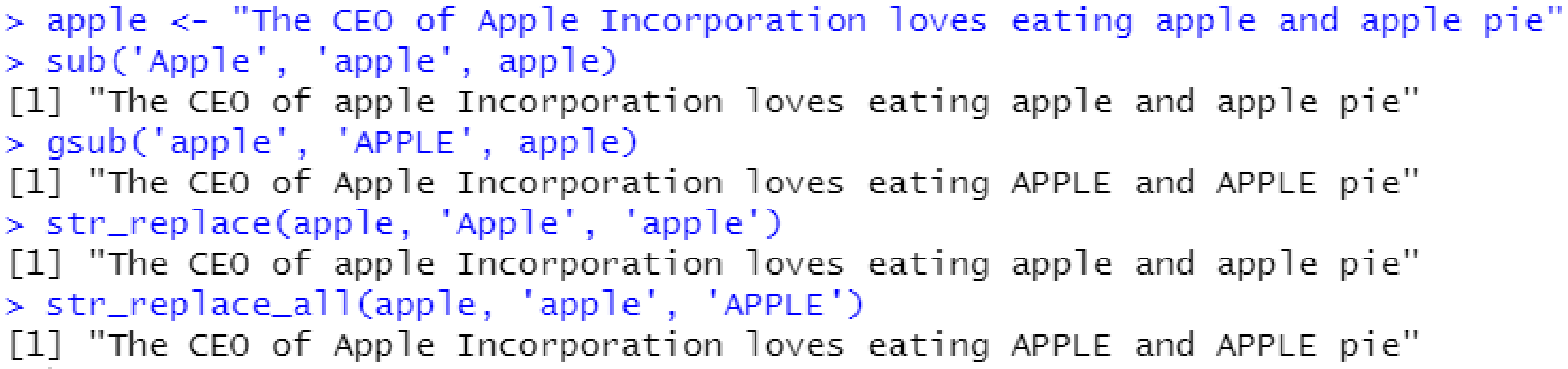
Useful when converting matrix transformation



```
> t(wiki.ds.short.2d)
     wiki.ds.short.2d
[1,] "Data"          "'concept"    "data"         "their"    "order"        "analyse"
[2,] "science"       "to"          "analysis,"    "related"  "to"           "actual"
[3,] "is"            "unify"       "informatics," "methods'" "'understand"  "phenomena'"
[4,] "a"             "statistics," "and"          "in"       "and"          "with"
```

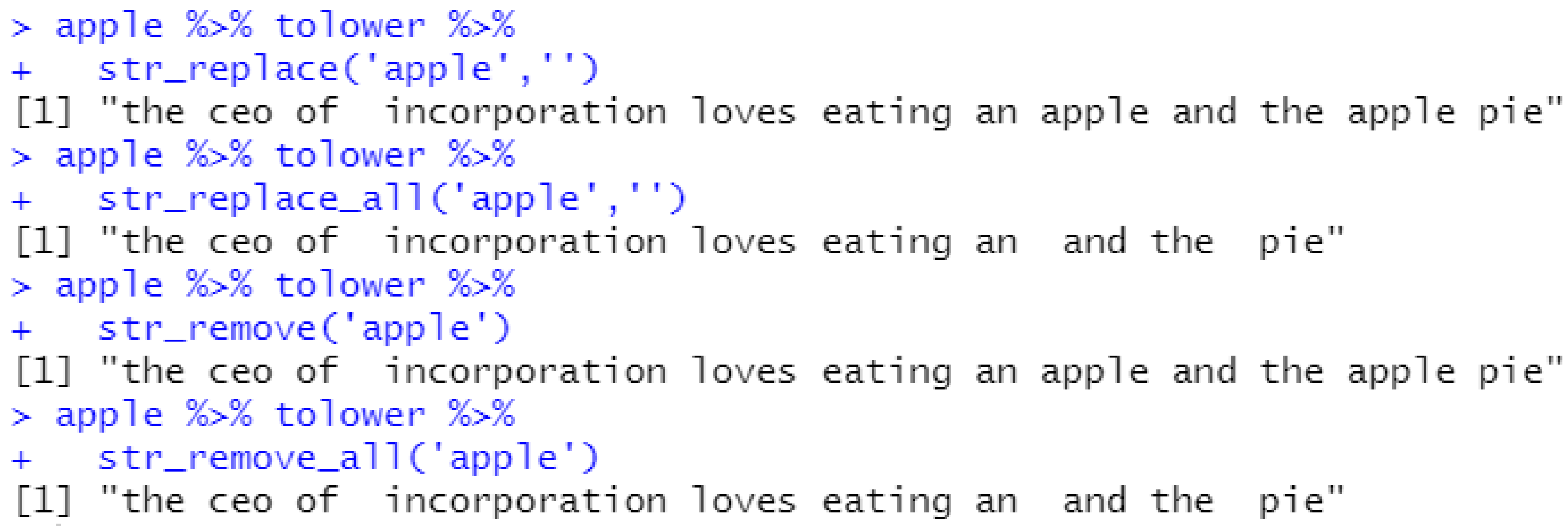
stringr::str_replace() & stringr::str_replace_all()

- str_replace() & str_replace_all()
  - similar to sub() and gsub()
  - str_replace(string, pattern, “”) == str_remove(string, pattern)



```
> apple <- "The CEO of Apple Incorporation loves eating apple and apple pie"
> sub('Apple', 'apple', apple)
[1] "The CEO of apple Incorporation loves eating apple and apple pie"
> gsub('apple', 'APPLE', apple)
[1] "The CEO of Apple Incorporation loves eating APPLE and APPLE pie"
> str_replace(apple, 'Apple', 'apple')
[1] "The CEO of apple Incorporation loves eating apple and apple pie"
> str_replace_all(apple, 'apple', 'APPLE')
[1] "The CEO of Apple Incorporation loves eating APPLE and APPLE pie"
```

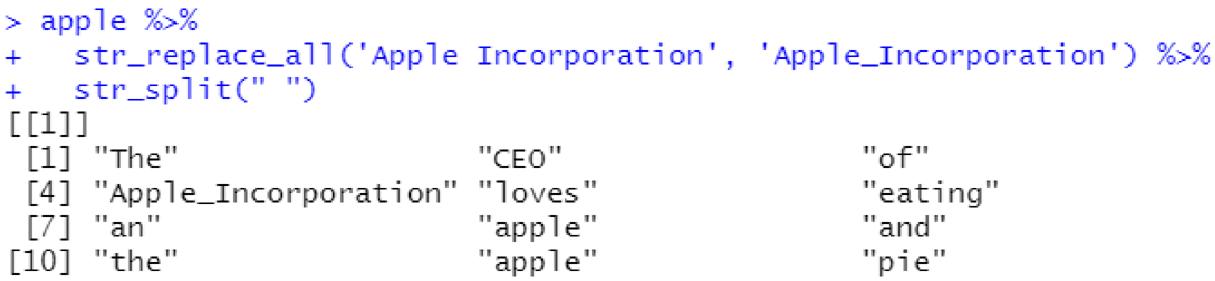
Useful when using pipe operator



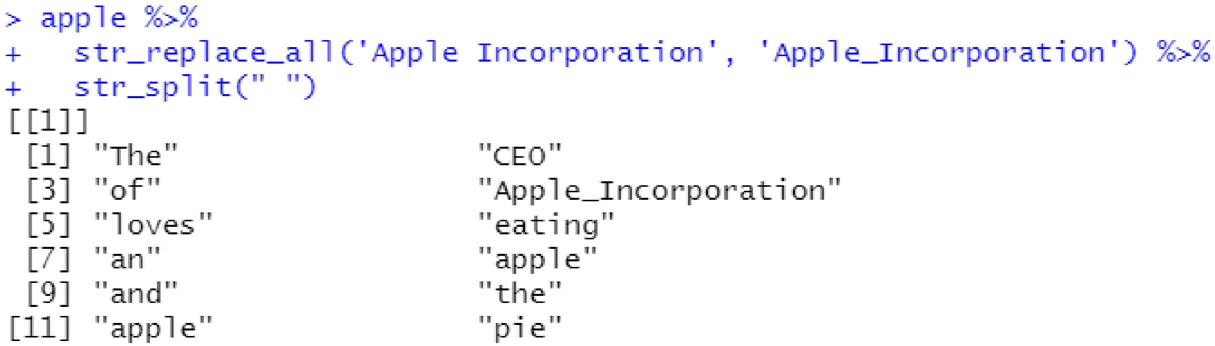
```
> apple %>% tolower %>%
+   str_replace('apple','')
[1] "the ceo of  incorporation loves eating an apple and the apple pie"
> apple %>% tolower %>%
+   str_replace_all('apple','')
[1] "the ceo of  incorporation loves eating an  and the  pie"
> apple %>% tolower %>%
+   str_remove('apple')
[1] "the ceo of  incorporation loves eating an apple and the apple pie"
> apple %>% tolower %>%
+   str_remove_all('apple')
[1] "the ceo of  incorporation loves eating an  and the  pie"
```

stringr::str_replace() & stringr::str_replace_all()

- str_replace() & str_replace_all()
  - Useful when handling “unique text” and removing meaningless text

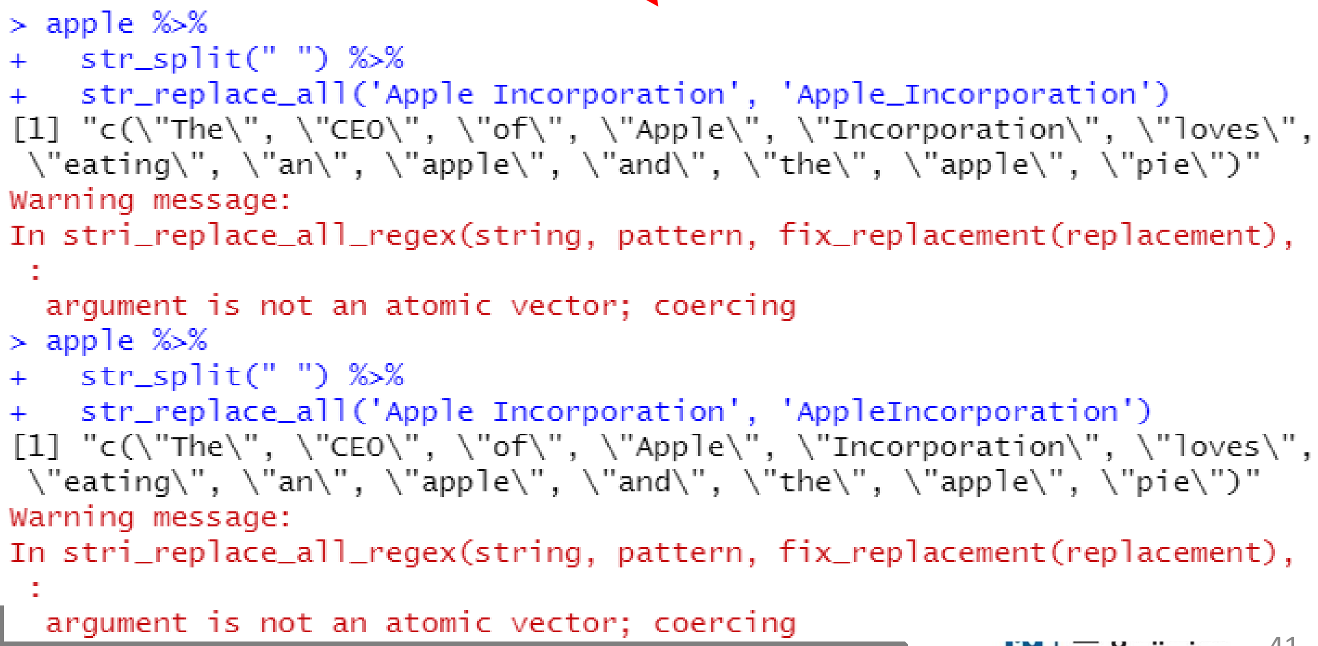


```
> apple %>%
+   str_replace_all('Apple Incorporation', 'Apple_Incorporation') %>%
+   str_split(" ")
[[1]]
 [1] "The"                 "CEO"                 "of"
 [4] "Apple_Incorporation" "loves"               "eating"
 [7] "an"                  "apple"               "and"
[10] "the"                 "apple"               "pie"
```



```
> apple %>%
+   str_replace_all('Apple Incorporation', 'Apple_Incorporation') %>%
+   str_split(" ")
[[1]]
 [1] "The"                 "CEO"
 [3] "of"                  "Apple_Incorporation"
 [5] "loves"               "eating"
 [7] "an"                  "apple"
 [9] "and"                 "the"
[11] "apple"               "pie"
```

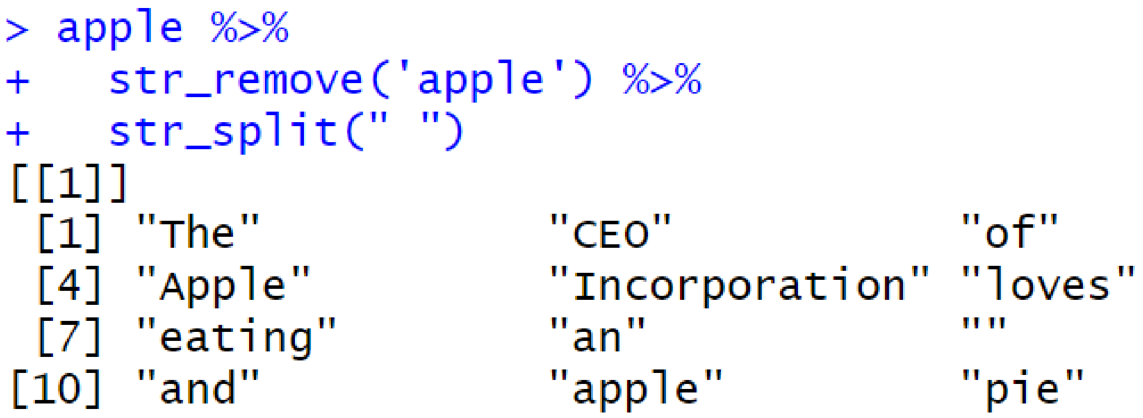
The order of
operation matters



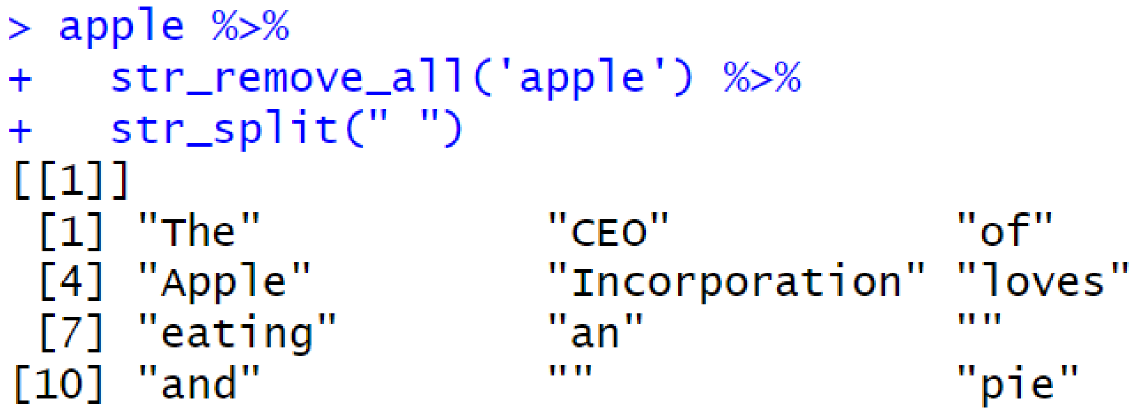
```
> apple %>%
+   str_split(" ") %>%
+   str_replace_all('Apple Incorporation', 'Apple_Incorporation')
[1] "c(\"The\", \"CEO\", \"of\", \"Apple\", \"Incorporation\", \"loves\",
 \"eating\", \"an\", \"apple\", \"and\", \"the\", \"apple\", \"pie\")"
Warning message:
In stri_replace_all_regex(string, pattern, fix_replacement(replacement),
  :
  argument is not an atomic vector; coercing
> apple %>%
+   str_split(" ") %>%
+   str_replace_all('Apple Incorporation', 'AppleIncorporation')
[1] "c(\"The\", \"CEO\", \"of\", \"Apple\", \"Incorporation\", \"loves\",
 \"eating\", \"an\", \"apple\", \"and\", \"the\", \"apple\", \"pie\")"
Warning message:
In stri_replace_all_regex(string, pattern, fix_replacement(replacement),
  :
  argument is not an atomic vector; coercing
```

stringr::str_remove() & stringr::str_remove_all()

- str_remove() & str_remove_all()
  - Useful when handling “unique text” and removing meaningless text



```
> apple %>%
+   str_remove('apple') %>%
+   str_split(" ")
[[1]]
 [1] "The"           "CEO"           "of"           
 [4] "Apple"         "Incorporation" "loves"        
 [7] "eating"        "an"            ""             
[10] "and"           "apple"         "pie"          
```



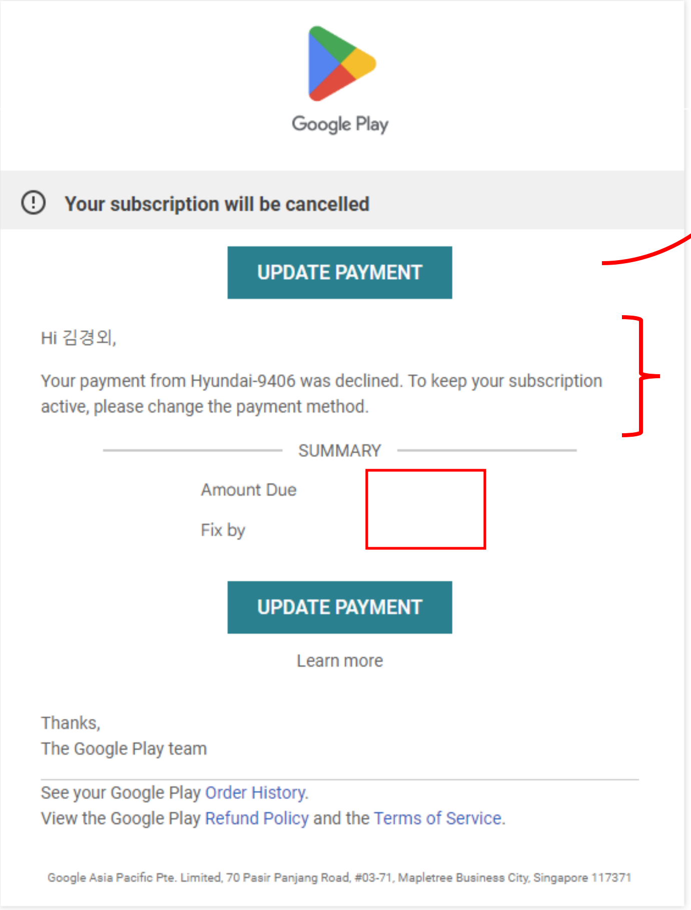
```
> apple %>%
+   str_remove_all('apple') %>%
+   str_split(" ")
[[1]]
 [1] "The"           "CEO"           "of"           
 [4] "Apple"         "Incorporation" "loves"        
 [7] "eating"        "an"            ""             
[10] "and"           ""              "pie"          
```

stringr::str_glue()

- str_glue()
  - Allows us to create a text consisting "fixed text" and "changeable text"

user.name *card.name*

Hi 김경외,Your payment from Hyundai-9406 was declined. To keep your subscription active, please change the payment method.



Google Play

Your subscription will be cancelled

UPDATE PAYMENT

Hi 김경외,

Your payment from Hyundai-9406 was declined. To keep your subscription active, please change the payment method.

SUMMARY

Amount Due

Fix by

UPDATE PAYMENT

Learn more

Thanks,
The Google Play team

See your Google Play Order History.
View the Google Play Refund Policy and the Terms of Service.

Google Asia Pacific Pte. Limited, 70 Pasir Panjang Road, #03-71, Mapletree Business City, Singapore 117371

```
> user.name <- "김경외"
> card.name <- "Hyundai-9406"
> str_glue("Hi {user.name},Your payment from {card.name} was declined. To
 keep your subscription active, please change the payment method.")
Hi 김경외,Your payment from Hyundai-9406 was declined. To keep your subscr
iption active, please change the payment method.
> df.set <- data.frame(user.name,card.name)
> df.set
  user.name    card.name
1    김경외 Hyundai-9406
> str_glue("Hi {df.set$user.name},Your payment from {df.set$card.name} wa
s declined. To keep your subscription active, please change the payment m
ethod.")
Hi 김경외,Your payment from Hyundai-9406 was declined. To keep your subscr
iption active, please change the payment method.
```