DATA SCIENCE INDONESIA

Sentiment Analysis with **R**

DSI Jatim Camp #3, 29 September 2018

Outline

- Overview Sentiment Analysis
- Intro to R Programming Language
- **Intro to Text Analysis**
- Preprocessing
- Visualize
- Sentiment

Introduction to Text Analysis

String Manipulation

```
> string1 <- "Setiap hari ibu pergi ke Pasar untuk membeli Pisang Ambon"
> string2 <- ‘To include a "quote" inside a string, I use single quotes'
> string1
[1] "Setiap hari ibu pergi ke Pasar untuk membeli Pisang Ambon"
> string2
[1] "If I want to include a "quote" inside a string, I use single quotes"

> nchar(string1)
[1] 64
> nchar(string2)
[1] 67

> strsplit(string1, split = ' ')
[[1]]
 [1] "Setiap"  "hari"    "ibu"     "pergi"   "ke"      "Pasar"   "untuk"
 [8] "membeli" "Pisang"  "Ambon"
> strsplit(string2, split = ' ')
[[1]]
 [1] "To"         "include"   "a"         "\"quote\"" "inside"    "a"
 [7] "string,"   "I"         "use"       "single"    "quotes"
```

String Manipulation

```
> strsplit(string2, split = ' ‘)
> Multiple_sentences = c("Monday: The doctor's appointment is at 2:45pm.",
                         "Tuesday: The dentist's appointment is at 11:30 am.",
                         "Wednesday: At 7:00pm, there is a basketball game!",
                         "Thursday: Be back home by 11:15 pm at the latest.",
                         "Friday: Take the train at 08:10 am, arrive at 09:00am.")
> Multiple_sentences
[1] "Monday: The doctor's appointment is at 2:45pm."
[2] "Tuesday: The dentist's appointment is at 11:30 am."
[3] "Wednesday: At 7:00pm, there is a basketball game!"
[4] "Thursday: Be back home by 11:15 pm at the latest."
[5] "Friday: Take the train at 08:10 am, arrive at 09:00am."
```

String Manipulation

```
> as.data.frame(Multiple_sentences)
                                      Multiple_sentences
1         Monday: The doctor's appointment is at 2:45pm.
2     Tuesday: The dentist's appointment is at 11:30 am.
3      Wednesday: At 7:00pm, there is a basketball game!
4      Thursday: Be back home by 11:15 pm at the latest.
5 Friday: Take the train at 08:10 am, arrive at 09:00am.

> paste(string1, string2, sep = ". ")
[1] "Setiap hari ibu pergi ke Pasar untuk membeli Pisang Ambon. To include a "quote"
inside a string, I use single quotes"

> toupper(string1)
[1] "SETIAP HARI IBU PERGI KE PASAR UNTUK MEMBELI PISANG AMBON"

> tolower(string2)
[1] "to include a \"quote\" inside a string, i use single quotes"
```

String Manipulation

```
> a = "Hari Sabtu"
> b = 1 + 6
> print(a)
[1] "Hari Sabtu"

> print(b)
[1] 7

> is.character(a)
[1] TRUE

> is.character(b)
[1] FALSE
> as.character(b)
[1] "7"
```

```
> data.frame(numbers = 1:5, letters = letters[1:5])
  numbers letters
1       1       a
2       2       b
3       3       c
4       4       d
5       5       e

> paste("The life of", pi)
[1] "The life of 3.14159265358979"

> paste("I", "love", "R", sep = "--")
[1] "I--love--R"
```

String Manipulation

`print()`	generic printing
`noquote()`	print with no quotes
`cat()`	concatenation
`format()`	special formats
`toString()`	convert to string
`sprintf()`	printing
`nchar()`	number of characters
`tolower()`	convert to lower case
`toupper()`	convert to upper case
`casefold()`	case folding
`chartr()`	character translation
`abbreviate()`	abbreviation
`substring()`	substrings of a character vector
`substr()`	substrings of a character vector

Regular Expression

A sequence of characters that define a search pattern, mainly for use in pattern matching with text strings.

Typically, regex patterns consist of a combination of alphanumeric characters as well as special characters. The pattern can also be as simple as a single character or it can be more complex and include several characters.

Regular Expression

Main Regex Function

Function	Purpose	Characteristic
`grep()`	finding regex matches	which elements are matched (index or value)
`grepl()`	finding regex matches	which elements are matched (TRUE & FALSE)
`regexpr()`	finding regex matches	positions of the first match
`gregexpr()`	finding regex matches	positions of all matches
`regexec()`	finding regex matches	hybrid of regexpr() and gregexpr()
`sub()`	replacing regex matches	only first match is replaced
`gsub()`	replacing regex matches	all matches are replaced
`strsplit()`	splitting regex matches	split vector according to matches

Regular Expression

Metacharacters

Metacharacters consist of non-alphanumeric symbols

```
> money = "$money"
> sub(pattern = "$", replacement = "", x
  = money)
[1] "$money"
> sub(pattern = "\\$", replacement = "",
  x = money)
[1] "money"
```

Metacharacter	Literal meaning	Escape in R
.	the period or dot	\\.
$	the dollar sign	\\$
*	the asterisk or star	*
+	the plus sign	\\+
?	the question mark	\\?
|	the vertical bar or pipe symbol	\\\|
n	the backslash	\\\\
^	the caret	\\^
[	the opening square bracket	\\[
]	the closing square bracket	\\]
f	the opening curly bracket	\\f
g	the closing curly bracket	\\g
(	the opening round bracket	\\(
)	the closing round bracket	\\)

Regular Expression

Quanitifiers

When we want to match a **certain number** of characters that meet a certain criteria we can apply quantifiers to our pattern searches.

Quantifier	**Description**
`?`	The preceding item is optional and will be matched at most once
`*`	The preceding item will be matched zero or more times
`+`	The preceding item will be matched one or more times
`{n}`	The preceding item is matched exactly n times
`{n,}`	The preceding item is matched n or more times
{n,m}	The preceding item is matched at least n times, but not more than m times

Regular Expression

Quanitifiers

```
> people = c("rori", "emilia", "matteo", "mehmet", "filipe", "anna", "tyler",
        "rasmus", "jacob", "youna", "flora", "adi")


> grep(pattern = "m?", people, value = TRUE)
 [1] "rori"   "emilia" "matteo" "mehmet" "filipe" "anna"   "tyler"  "rasmus"
 [9] "jacob"  "youna"  "flora"  "adi"
> grep(pattern = "m{1}", people, value = TRUE, perl = FALSE)
[1] "emilia" "matteo" "mehmet" "rasmus"
> grep(pattern = "m*t", people, value = TRUE)
[1] "matteo" "mehmet" "tyler"
```

Regular Expression

Sequences

To match a sequence of characters we can apply short-hand notation which captures the fundamental types of sequences.

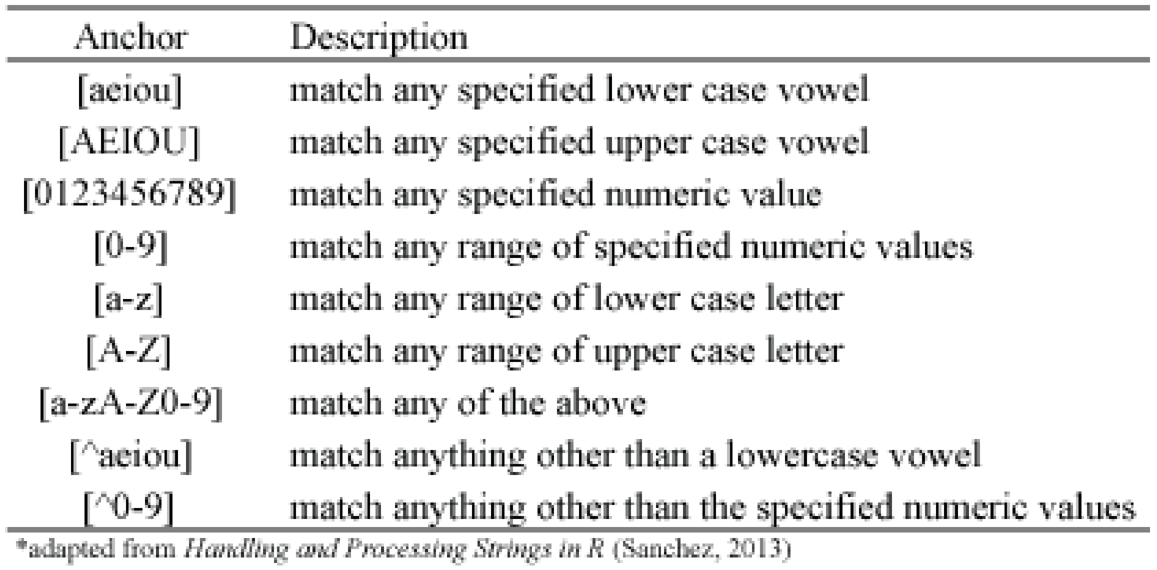
```
> sub("\\d", "_", "the dandelion war 2010")
[1] "the dandelion war _010"
> gsub("\\d", "_", "the dandelion war 2010")
[1] "the dandelion war ____"
> sub("\\D", "_", "the dandelion war 2010")
[1] "_he dandelion war 2010"
> gsub("\\D", "_", "the dandelion war 2010")
[1] "__________________2010"
```

Anchor	Description
\\d	match a digit character
\\D	match a non-digit character
\\s	match a space character
\\S	match a non-space character
\\w	match a word character
\\W	match a non-word character
\\b	match a word boundary
\\B	match a non-(word boundary)
\\h	match a horizontal space
\\H	match a non-horizontal space
\\v	match a vertical space
\\V	match a non-vertical space

Regular Expression

Character classes

To match one of several characters in a specified set we can enclose the characters of concern with square brackets [ ]. In addition, to match any characters not in a specified character set we can include the caret ^ at the beginning of the set within the brackets..



Anchor	Description
[aeiou]	match any specified lower case vowel
[AEIOU]	match any specified upper case vowel
[0123456789]	match any specified numeric value
[0-9]	match any range of specified numeric values
[a-z]	match any range of lower case letter
[A-Z]	match any range of upper case letter
[a-zA-Z0-9]	match any of the above
[^aeiou]	match anything other than a lowercase vowel
[^0-9]	match anything other than the specified numeric values

*adapted from *Handling and Processing Strings in R* (Sanchez, 2013)

Regular Expression

Character classes

```
> x <- c("RStudio", "v.0.99.484", "2015", "09-22-2015", "grep vs. grepl")

# find any strings with numeric values between 0-9
> grep(pattern = "[0-9]", x, value = TRUE)
## [1] "v.0.99.484" "2015"       "09-22-2015"

# find any strings with numeric values between 6-9
> grep(pattern = "[6-9]", x, value = TRUE)
## [1] "v.0.99.484" "09-22-2015"

# find any strings with the character R or r
> grep(pattern = "[Rr]", x, value = TRUE)
## [1] "RStudio"        "grep vs. grepl"

# find any strings that have non-alphanumeric characters
> grep(pattern = "[^0-9a-zA-Z]", x, value = TRUE)
## [1] "v.0.99.484"     "09-22-2015"     "grep vs. grepl"
```

DATA SCIENCE INDONESIA

Thank You