R- String and Regular Expressions

What are Regular Expressions ?

Regular Expressions (regex) are a set of pattern matching commands used to detect string sequences in a large text data. These commands are designed to match a family (alphanumeric, digits, words) of text which makes then versatile enough to handle any text / string class.

What is String Manipulation ?

String manipulation comprises a series of functions used to extract information from text variables.

In machine learning, these functions are being widely used for doing feature engineering, i.e., to create new features out of existing string features.

In R, we have **packages such as stringr and stringi** which are loaded with all string manipulation functions.

In addition, R also comprises several base functions for string manipulations. These functions are designed to complement regular expressions.

The practical differences between string manipulation functions and regular expressions are

- We use string manipulation functions to do simple tasks such as splitting a string, extracting the first three letters, etc.. We use regular expressions to do more complicated tasks such as extract email IDs or date from a set of text.

- String manipulation functions are designed to respond in a certain way. They don't deviate from their natural behavior. Whereas, we can customize regular expressions in any way we want.

List of String Manipulation Functions

```
text <- "san francisco“          #a string is any value enclosed in quotes (" ")
typeof(text)
[1] "character"

num <- c("24","34","36")
typeof(num)
[1] "character"
```

List of String Manipulation Functions CONT.

R's base paste function is used to combine (or paste) set of strings.

```
var3 <- paste("Var1","Var2",sep = "-")
var3
[1] "Var1-Var2"
```

```
paste(1:5,c("?","!"),sep = "-")
[1] "1-?" "2-!" "3-?" "4-!" "5-?"
```

R also allows you to print and concatenate strings without quotes. It is done using the cat function.

```
cat(text,"USA",sep = "-")
san francisco-USA
```

```
cat(month.name[1:5],sep = " ")
January February March April May
```

In stringr package, its substitute function is str_c() or str_join().

The toString function allows you to convert any non-character value to a string.

```
toString (1:10)
[1] "1,2,3,4,5,6,7,8,9,10“
```

Functions	**Description**
nchar()	It counts the number of characters in a string or vector. In the stringr package, it's substitute function is str_length()
tolower()	It converts a string to the lower case. Alternatively, you can also use the str_to_lower() function
toupper()	It converts a string to the upper case. Alternatively, you can also use the str_to_upper() function
chartr()	It is used to replace each character in a string. Alternatively, you can use str_replace() function to replace a complete string
substr()	It is used to extract parts of a string. **Start and end positions need to be specified**. Alternatively, you can use the str_sub() function

setdiff()	**It is used to determine the difference between two vectors**
setequal()	It is used to check if the two vectors have the same string values
abbreviate()	It is used to abbreviate strings. The length of abbreviated string needs to be specified
strsplit()	It is used to split a string based on a criterion. It returns a list. Alternatively, you can use the str_split() function. This function lets you convert your list output to a character matrix
sub()	It is used to find and replace the first match in a string
gsub()	It is used to find and replace all the matches in a string / vector. Alternatively, you can use the str_replace() function

```
> library(stringr)
> string <- "Los Angeles, officially the City of Los Angeles and often known by its initials L.A.";
> strwrap(string)
[1] "Los Angeles, officially the City of Los Angeles and often known by its"
[2] "initials L.A."

> #count number of characters
> nchar(string)
[1] 84
> str_length(string)
[1] 84

#replace strings
> chartr("and","for",x = string)                #letters a,n,d get replaced by f,o,r
[1] "Los Aogeles, officiflly the City of Los Aogeles for ofteo koowo by its ioitifls L.A."
> str_replace_all(string = string, pattern = c("City"),replacement = "state")        #this is case sentitive
[1] "Los Angeles, officially the state of Los Angeles and often known by its initials L.A."
```

```
>#extract parts of string
 > substr(x = string,start = 5,stop = 11)
[1] "Angeles“

#extract angeles
>str_sub(string = string, start = 5, end = 11)
[1] "Angeles“

> #get difference between two vectors
> setdiff(c("monday","tuesday","wednesday"),c("monday","thursday","friday"))
[1] "tuesday"   "wednesday“

> #check if strings are equal
> setequal(c("monday","tuesday","wednesday"),c("monday","tuesday","wednesday"))
[1] TRUE

> setequal(c("monday","tuesday","thursday"),c("monday","tuesday","wednesday"))
[1] FALSE

> #abbreviate strings
> abbreviate(c("monday","tuesday","wednesday"),minlength = 3)
   monday   tuesday wednesday
    "mnd"     "tsd"     "wdn“
```

```
> #split strings
> strsplit(x = c("ID-101","ID-102","ID-103","ID-104"), split = "-")
[[1]]
[1] "ID"  "101"

[[2]]
[1] "ID"  "102"

[[3]]
[1] "ID"  "103"

[[4]]
[1] "ID"  "104"

> str_split(string = c("ID-101","ID-102","ID-103","ID-104"), pattern = "-",simplify = T)
     [,1] [,2]
[1,] "ID" "101"
[2,] "ID" "102"
[3,] "ID" "103"
[4,] "ID" "104"
```

```
> #find and replace first match
> sub(pattern = "L", replacement = "B", x = string, ignore.case = T)
[1] "Bos Angeles, officially the City of Los Angeles and often known by its initials L.A.“
```

```
> #find and replace all matches
> gsub(pattern = "Los",replacement = "Bos",x = string,ignore.case = T)
[1] "Bos Angeles, officially the City of Bos Angeles and often known by its initials L.A.“
```

List of Regular Expression Commands

Along with above functions, there are several other functions specially designed to deal with regular expressions (a.k.a regex).

R is equally powerful when it comes to parsing text data. In regex, there are multiple ways of doing a certain task. Therefore, it's essential for you to stick to a particular method to avoid confusion.

For using regular expressions, the available base regex functions are grep(), grepl(), regexpr(), gregexpr(), regexec(), and regmatches(). Here's a quick preview of these functions:

Function	**Description**
grep	returns the **index or value of the matched string**
grepl	returns the **Boolean value (True or False) of the matched string**
regexpr	return the **index of the first match**
gregexpr	returns the **index of all matches**
regexec	**is a hybrid of regexpr and gregexpr**
regmatches	returns **the matched string at a specified index**. It is used in conjunction with regexpr and gregexpr.

Regular expressions in R can be divided into 5 categories:

- Metacharacters
- Sequences
- Quantifiers
- Character Classes
- POSIX character classes