

Homework 4

Rosemary Kinuthia

2/28/2018

```
#packages required
library(stringr)
library(stringi)
```

Section 1.

Suppose we have a vector vText as follows: `vText <- c('nurse', 'nut', 'ninja', 'nutrient', 'under', 'unusual')` We want to write a regular expression that matches n, nu, un, or unu in vText and replaces the matching patterns with .. To do this, the R code is as follows:

```
vText <- c('nurse', 'nut', 'ninja', 'nutrient', 'under', 'unusual')
pattern <- 'u?nu?'
gsub(pattern, '.', vText)
```

```
## [1] ".rse" ".t" ".i.ja" ".trie.t" ".der" ".sual"
```

Problem 1.

Suppose you have another vector vText as follows: `vText <- c("google", "logo", "dig", "blog", "boogie")` You want to match g, og, go, or ogo and replace with .. Write the R code that will make that happen.

Answer to Problem 1.

```
vText <- c("google", "logo", "dig", "blog", "boogie")
pattern <- 'g?og?'
gsub(pattern, '.', vText)
```

```
## [1] "..le" "l.." "dig" "bl." "b..ie"
```

Section 2.

You have 3 strings of text that you wish to merge. One way to do this is to use the paste function.

```
x <- "I AM SAM. I AM SAM. SAM I AM."  
y <- "THAT SAM-I-AM! THAT SAM-I-AM! I DO NOT LIKE THAT SAM-I-AM!"  
z <- "DO YOU LIKE GREEN EGGS AND HAM?"  
paste(x, y, z, collapse = NULL)
```

```
## [1] "I AM SAM. I AM SAM. SAM I AM. THAT SAM-I-AM! THAT SAM-I-AM! I DO NOT LIKE THA  
T SAM-I-AM! DO YOU LIKE GREEN EGGS AND HAM?"
```

Extra credit: What is the difference if you use the paste0 function instead of the paste function above?

Paste links together a series of strings (in this case x, y, z) while paste0 does not link a series of strings.

Problem 2.

Suppose that you now have 4 lines of text as follows: W <- "Hey Diddle Diddle, the cat and the fiddle," X <- "The cow jumped over the moon." Y <- "The little boy laughed to see such a sport," Z <- "And the dish ran away with the spoon." Write the R code below to merge these 3 strings.

Answer to Problem 2.

```
W <- "Hey Diddle Diddle, the cat and the fiddle,"  
X <- "The cow jumped over the moon."  
Y <- "The little boy laughed to see such a sport,"  
Z <- "And the dish ran away with the spoon."  
paste(W, X, Y, Z, collapse = NULL)
```

```
## [1] "Hey Diddle Diddle, the cat and the fiddle, The cow jumped over the moon. The  
little boy laughed to see such a sport, And the dish ran away with the spoon."
```

Section 3.

An alternative way to merge these text strings is to concatenate them with the str_c function from the stringr package. Suppose we want to concatenate the 3 strings we did above, but also NA. We can do this in these two ways. What is the difference? (Answer to yourself)

```
paste(X, Y, Z, NA, collapse = NULL)
```

```
## [1] "The cow jumped over the moon. The little boy laughed to see such a sport, And  
the dish ran away with the spoon. NA"
```

```
str_c(X, Y, Z, NA, collapse = NULL)
```

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

What is the difference between the two results? (Answer to yourself.)

Problem 3.

We now want to concatenate our 4 vectors and NA. Do this using both methods. W <- "Hey Diddle Diddle, the cat and the fiddle," X <- "The cow jumped over the moon." Y <- "The little boy laughed to see such a sport," Z <- "And the dish ran away with the spoon."

Answer to Problem 3.

```
#Using paste function  
W <- "Hey Diddle Diddle, the cat and the fiddle,"  
X <- "The cow jumped over the moon."  
Y <- "The little boy laughed to see such a sport,"  
Z <- "And the dish ran away with the spoon."  
paste(W, X, Y, Z, NA, collapse = NULL)
```

```
## [1] "Hey Diddle Diddle, the cat and the fiddle, The cow jumped over the moon. The  
little boy laughed to see such a sport, And the dish ran away with the spoon. NA"
```

```
#Using str_c function  
str_c(W, X, Y, Z, NA, collapse=NULL)
```

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

Section 4.

We can use the str_sub function to extract parts of strings. Suppose I wanted to extract the last 5 letter of my name.

```
myName <- "Vicki Hertzberg"
Length <- str_length(myName)
last5letters <- str_sub(myName, Length-4, Length)
last5letters
```

```
## [1] "zberg"
```

Problem 4.

Suppose Melinda Higgins wants to extract the last 6 letters of her name. herName <- "Melinda Higgins" Write the code below to extract the last 6 letters of her name.

Answer to Problem 4.

```
herName <- "Melinda Higgins"
Length <- str_length(herName)
last6letters <- str_sub(herName, Length-5, Length)
last6letters
```

```
## [1] "iggins"
```

Section 5

Suppose I have a string and I want to split it into unique words based on the occurrence of a separator, as follows:

```
myString <- "The_quick_brown_fox_jumped_over_the_lazy_dog"

#the separator is the character "_"

mySeparatedString <- str_split(myString, "_")
mySeparatedString
```

```
## [[1]]
## [1] "The"      "quick"    "brown"    "fox"      "jumped"  "over"     "the"      "lazy"
## [9] "dog"
```

```
##If you look in your environment you will see that mySeparatedString is a List of 1.
```

Problem 5.

I want to separate the following string into separate words: myNewString <-

"Now_is_the_time_for_all_good_men_to_come_to_the_aid_of_their_country" Split this new string into separate words:

Answer to Problem 5.

```
myNewString <- "Now_is_the_time_for_all_good_men_to_come_to_the_aid_of_their_country"
myNewSeparatedString <- str_split(myNewString, "_")
myNewSeparatedString
```

```
## [[1]]
## [1] "Now"      "is"       "the"      "time"     "for"      "all"      "good"
## [8] "men"      "to"       "come"     "to"       "the"      "aid"      "of"
## [15] "their"    "country"
```

Section 6.

On another occasion, I need the same string split so that the last word comes off, and the rest remains intact. I can achieve that in the following way:

```
myString <- "The_quick_brown_fox_jumped_over_the_lazy_dog"
myNewSplitSpring <- str_split(myString, "_", n=2)
myNewSplitSpring
```

```
## [[1]]
## [1] "The"
## [2] "quick_brown_fox_jumped_over_the_lazy_dog"
```

Problem 6.

Suppose we wanted to split off the first “word” from myNewString. Again, we have

myNewString <- “Now_is_the_time_for_all_good_men_to_come_to_the_aid_of_their_country” Split off the first word but leave the rest intact.

Answer 6.

```
myNewString <- "Now_is_the_time_for_all_good_men_to_come_to_the_aid_of_their_country"  
myNewSplitString <- str_split(myNewString, "_", n=2)  
myNewSplitString
```

```
## [[1]]  
## [1] "Now"  
## [2] "is_the_time_for_all_good_men_to_come_to_the_aid_of_their_country"
```

Section 7.

With the stringi package there is functionality to count the number of words in a string.

```
newString <- "The quick brown fox jumps over the lazy dog."  
stri_count_words(newString)
```

```
## [1] 9
```

Problem 7.

yourNewString <- "Now is the time for all good men to come to the aid of their country" Use the stri_count_words function as above to count the number of distinct words in yourNewString.

Answer 7.

```
yourNewString <- "Now is the time for all good men to come to the aid of their countr  
y"  
stri_count_words(yourNewString)
```

```
## [1] 16
```

Section 8.

Let's say you have a string listing famous mathematicians and you want to know if there are any duplicates in the list. You would do this as follows:

```
mathematicians <- c("Goedel", "Euler", "Gauss", "Hilbert", "Goedel", "Fermat", "LaGra  
nge", "Gauss")  
mathematicians[stri_duplicated(mathematicians)]
```

```
## [1] "Goedel" "Gauss"
```

Problem 8.

Suppose you have string listing famous nurses and you want to find the duplicates. Here is the list:

```
nurses <-c("Nightingale", "Barton", "Dix", "Sanger", "Barton", "Woodruff", "Lincoln", "Dix", "Peplau")
```

Answer 8.

```
nurses <-c("Nightingale", "Barton", "Dix", "Sanger", "Barton", "Woodruff", "Lincoln",
"Dix", "Peplau")
nurses[stri_duplicated(nurses)]
```

```
## [1] "Barton" "Dix"
```

Section 9.

The LETTERS object is a vector of length 26, consisting of all of the capital letters. Suppose we wanted to use this object to create the string A-B_C-D_E-F_G-H_I-J_K-L_M-N_O-P_Q-R_S-T_U-V_W-X_Y-Z_. To achieve this, we use the following commands:

```
stri_join(LETTERS, separators = c("-", "_"), collapse = "")
```

```
## [1] "A-B_C-D_E-F_G-H_I-J_K-L_M-N_O-P_Q-R_S-T_U-V_W-X_Y-Z_"
```

Problem 9.

Suppose we create the object DIGITS as follows:

```
DIGITS <- c("0", "1", "2", "3", "4", "5", "6", "7", "8", "9") How can we form the string `0_1-2_3-4_5-6_7-8_9-`?
```

Answer 9.

```
DIGITS <- c("0", "1", "2", "3", "4", "5", "6", "7", "8", "9")
stri_join(DIGITS, separators = c("-", "_"), collapse = "")
```

```
## [1] "0-1_2-3_4-5_6-7_8-9_"
```

Section 10

Suppose we want to replace statistician with mathematician and average with median in the following pun:

```
pun <- "A statistician can have his head in an oven and his feet in ice, and he will  
say that on the average he feels fine"  
punModified <- stri_replace_all_fixed(pun, c("statistician", "average"), c("mathemati  
cian", "median"), vectorize_all = FALSE)  
punModified
```

```
## [1] "A mathematician can have his head in an oven and his feet in ice, and he will  
say that on the median he feels fine"
```

Problem 10

Using the original pun, replace his with her and he with she:

Answer 10

Note that the word head also starts with he but we don't want to substitute she into there.

```
pun <- "A statistician can have his head in an oven and his feet in ice, and he will  
say that on the average he feels fine"  
punModified <- stri_replace_all_fixed(pun, c("his", " he "), c("her", " she "), vecto  
rize_all = FALSE)  
punModified
```

```
## [1] "A statistician can have her head in an oven and her feet in ice, and she will  
say that on the average she feels fine"
```

... The Github repository for this assignment can be accessed via this link:

(<https://github.com/RosemaryKinuthia/N741-Homework-4.git> (<https://github.com/RosemaryKinuthia/N741-Homework-4.git>) [<https://github.com/RosemaryKinuthia/N741-Homework-4.git> (<https://github.com/RosemaryKinuthia/N741-Homework-4.git>)] ...