HW4

Nicole Carlson

3/3/2018

### 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:

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.

pattern <- 'o?go?'  
gsub(pattern, '.', vText)

## [1] "..le" "l." "di." "bl." "bo.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 THAT 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?

To answer that question, I tried using both:

paste0(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 THAT SAM-I-AM!DO YOU LIKE GREEN EGGS AND HAM?"

Looks like paste0 takes out the space between each input when it concatenates the 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.

I also added some code here to change to lowercase all letters following a comma. Don’t be too impressed–a computer programmer helped me with this. The exercise prompted a fascinating conversation about regular expressions.

cow <- paste(W, X, Y, Z, collapse = NULL)  
pattern <- ', (.)'  
gsub (pattern, ', \\L\\1', cow, perl=TRUE)

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

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"

str\_c(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"

### 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. Write the code below to extract the last 6 letters of her name.

#### Answer to Problem 4.

herName <- "Melinda Higgins"  
Length <- str\_length(myName)  
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.

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.

myNewSplitSpring2 <- str\_split(myNewString, "\_", n=2)  
myNewSplitSpring2

## [[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.

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", "LaGrange", "Gauss")  
mathematicians[stri\_duplicated(mathematicians)]

## [1] "Goedel" "Gauss"

#### Problem 8.

Suppose you have string listing famouse 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[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 stringA-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.

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("mathematician", "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:

punModified2 <- stri\_replace\_all\_fixed(pun, c("his", "he"), c("her", "she"), vectorize\_all = FALSE)  
punModified2

## [1] "A statistician can have sher shead in an oven and sher feet in ice, and she will say that on tshe average she feels fine"

#### Answer 10

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

This was a hard question. I found this [link](http://www.endmemo.com/program/R/gsub.php) about gsub package that had a table of Regular Expression Syntax to use when instructing R to operate among strings. I used the syntax for ‘space’, \s, in the operation below. It took care of the ‘shed’ output from before, but I cannot get R to recognize ‘he’ in the string for substitution to ‘she’.

punModified2 <- stri\_replace\_all\_fixed(pun, c("his", "\\s\\he"), c("her", "she"), vectorize\_all = FALSE)  
punModified2

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

The github repository for this homework can be found [here](https://github.com/nicolecarlson/2018Week7)