BSDS 100: Intro to Data Science with R Assignment 8

Chase Darlington (University of San Francisco)

11/13/2018

Directions: For all questions in this assignment, write complete sentences and fully answer any question that is asked, and use R to answer each question. Provide all R code and solutions by knitting your final RStudio file into a single file named [your name] CA8.pdf. Late assignments will automatically have 10 points deducted, and will not be accepted after the answer key is posted (1 week following the due date).

1. From https://r4ds.had.co.nz/strings.html Create regular expressions to find all words from stringr::words that:

SETUP

```
words <- stringr::words
str(words)
```

```
##  chr [1:980] "a" "able" "about" "absolute" "accept" "account" ...
```

```
words <- data.frame(words)
nvowels <- str_count(tolower(words[,1]), "[aeoiu]")
total_letters <- str_count(tolower(words[,1]), "\\w")
starts_with_vowel <- grepl("^[aeiou]", tolower(words[,1]))
ends_with_vowel <- grepl("[aeiou]$", tolower(words[,1]))
ends_with_ed <- grepl("ed$", tolower(words[,1]))
ends_with_eed <- grepl("eed$", tolower(words[,1]))
ends_with_ed_not_eed <- ends_with_ed - ends_with_eed
ends_with_ing <- grepl("ing$", tolower(words[,1]))
ends_with_ise <- grepl("ise$", tolower(words[,1]))
ends_with_ing_or_ise <- ends_with_ing + ends_with_ise
words <- cbind(words,nvowels, total_letters, starts_with_vowel, ends_with_ed_not_eed, ends_with_ing_or_
```

(a) (2 points) Start with a vowel.

```
head(words[words$starts_with_vowel==TRUE,])
```

```
##      words nvowels total_letters starts_with_vowel ends_with_ed_not_eed
## 1        a       1             1              TRUE                     0
## 2     able       2             4              TRUE                     0
## 3    about       3             5              TRUE                     0
## 4 absolute       4             8              TRUE                     0
## 5   accept       2             6              TRUE                     0
## 6  account       3             7              TRUE                     0
##   ends_with_ing_or_ise
## 1                    0
## 2                    0
## 3                    0
## 4                    0
## 5                    0
## 6                    0
```

```
tail(words[words$starts_with_vowel==TRUE,])
```

```
##      words nvowels total_letters starts_with_vowel ends_with_ed_not_eed
## 908 unless       2             6              TRUE                     0
## 909  until       2             5              TRUE                     0
## 910     up       1             2              TRUE                     0
## 911   upon       2             4              TRUE                     0
## 912    use       2             3              TRUE                     0
## 913  usual       3             5              TRUE                     0
##     ends_with_ing_or_ise
## 908                    0
## 909                    0
## 910                    0
## 911                    0
## 912                    0
## 913                    0
```

(b) (2 points) That only contain consonants. (Hint: thinking about matching "not"-vowels.)

```
words[words$nvowels==0,]
```

```
##     words nvowels total_letters starts_with_vowel ends_with_ed_not_eed
## 123    by       0             2             FALSE                     0
## 249   dry       0             3             FALSE                     0
## 328   fly       0             3             FALSE                     0
## 538   mrs       0             3             FALSE                     0
## 895   try       0             3             FALSE                     0
## 952   why       0             3             FALSE                     0
##     ends_with_ing_or_ise
## 123                    0
## 249                    0
## 328                    0
## 538                    0
## 895                    0
## 952                    0
```

```
#technically y is a vowel in these words, but I believe this answers the question
```

(c) (2 points) End with ed, but not with eed.

```
words[words$ends_with_ed_not_eed==TRUE,]
```

```
##       words nvowels total_letters starts_with_vowel ends_with_ed_not_eed
## 82      bed       1             3             FALSE                     1
## 410 hundred       2             7             FALSE                     1
## 690     red       1             3             FALSE                     1
##     ends_with_ing_or_ise
## 82                     0
## 410                    0
## 690                    0
```

(d) (2 points) End with -ing or -ise.

```
words[words$ends_with_ing_or_ise==TRUE,]
```

```
##         words nvowels total_letters starts_with_vowel ends_with_ed_not_eed
## 15  advertise       4             9              TRUE                     0
## 113     bring       1             5             FALSE                     0
```

```
## 251    during       2             6             FALSE                    0
## 280   evening       3             7              TRUE                    0
## 288  exercise       4             8              TRUE                    0
## 448      king       1             4             FALSE                    0
## 512   meaning       3             7             FALSE                    0
## 533   morning       2             7             FALSE                    0
## 588 otherwise       4             9              TRUE                    0
## 637  practise       3             8             FALSE                    0
## 674     raise       3             5             FALSE                    0
## 681   realise       4             7             FALSE                    0
## 709      ring       1             4             FALSE                    0
## 710      rise       2             4             FALSE                    0
## 765      sing       1             4             FALSE                    0
## 834  surprise       3             8             FALSE                    0
## 860     thing       1             5             FALSE                    0
##     ends_with_ing_or_ise
## 15                     1
## 113                    1
## 251                    1
## 280                    1
## 288                    1
## 448                    1
## 512                    1
## 533                    1
## 588                    1
## 637                    1
## 674                    1
## 681                    1
## 709                    1
## 710                    1
## 765                    1
## 834                    1
## 860                    1
```

2. (4 points) Write and test a regular expression that will match a phone number in the format 555-555-5555

This variable complexity function is more efficient than a for loop, because it executes only as many commands as necessary to invalidate/validate a US number in the format 555-555-5555 (admittedly, I can think of a few quicker ways, especially if you strip out the "-")

```
US_phone_validation <- function(x){
  if(nchar(mynum)==12){
    hyphen1 <- substr(mynum, start = 4, stop = 4)
    hyphen2 <- substr(mynum, start = 8, stop = 8)
    areacode <- substr(mynum, start = 1, stop = 3)
    numpart1 <- substr(mynum, start = 1, stop = 3)
    numpart2 <-substr(mynum, start = 1, stop = 3)
    if(hyphen1=="-"){
      if(hyphen2=="-"){
        if(is.na(as.integer(areacode))==F){
          if(is.na(as.integer(numpart1))==F){
            if(is.na(as.integer(numpart2))==F){
              print("Thank you for your input. We will reach out to you soon.")
            }else{print("Please edit your entry to match the format: 555-555-5555")}
          }else{print("Please edit your entry to match the format: 555-555-5555")}
```

```
        }else{print("Please edit your entry to match the format: 555-555-5555")}
      }else{print("Please edit your entry to match the format: 555-555-5555")}
    }else{print("Please edit your entry to match the format: 555-555-5555")}
  }else{print("Please edit your entry to match the format: 555-555-5555")}
}

mynum <- ("253-948-2507")
US_phone_validation(mynum)
```

```
## [1] "Thank you for your input. We will reach out to you soon."
```

```
mynum <- ("2539482507")
US_phone_validation(mynum)
```

```
## [1] "Please edit your entry to match the format: 555-555-5555"
```

3. (6 points) Use for() and if() to loop through the numbers 1 to 30 and print "divisible by 3" for any that are divisible by 3 and print "not divisible by 3" for any that are not divisible by 3

```
#one way
numbers <- 1:30
is_divisible_by_3 <- NA
num <- data.frame(cbind(numbers, is_divisible_by_3))
for(i in 1:length(numbers)){
  j <- num$numbers[i]
  if(num$numbers[j]%%3==0){
    num$is_divisible_by_3[i] <- "divisible by 3"
    }
  else{
      num$is_divisible_by_3[i] <- "not divisible by 3"
    }
}
num
```

```
##    numbers  is_divisible_by_3
## 1        1 not divisible by 3
## 2        2 not divisible by 3
## 3        3     divisible by 3
## 4        4 not divisible by 3
## 5        5 not divisible by 3
## 6        6     divisible by 3
## 7        7 not divisible by 3
## 8        8 not divisible by 3
## 9        9     divisible by 3
## 10      10 not divisible by 3
## 11      11 not divisible by 3
## 12      12     divisible by 3
## 13      13 not divisible by 3
## 14      14 not divisible by 3
## 15      15     divisible by 3
## 16      16 not divisible by 3
## 17      17 not divisible by 3
## 18      18     divisible by 3
## 19      19 not divisible by 3
## 20      20 not divisible by 3
## 21      21     divisible by 3
```

```
## 22      22 not divisible by 3
## 23      23 not divisible by 3
## 24      24     divisible by 3
## 25      25 not divisible by 3
## 26      26 not divisible by 3
## 27      27     divisible by 3
## 28      28 not divisible by 3
## 29      29 not divisible by 3
## 30      30     divisible by 3
```

```
#another way
n <- 1:30
for(i in 1:length(n)){
  if(n[i]%%3==0){
    print(paste(n[i], "is divisible by 3", sep=" "))
    }
  else{
    print(paste(n[i], "is not divisible by 3", sep=" "))
    }
}
```

```
## [1] "1 is not divisible by 3"
## [1] "2 is not divisible by 3"
## [1] "3 is divisible by 3"
## [1] "4 is not divisible by 3"
## [1] "5 is not divisible by 3"
## [1] "6 is divisible by 3"
## [1] "7 is not divisible by 3"
## [1] "8 is not divisible by 3"
## [1] "9 is divisible by 3"
## [1] "10 is not divisible by 3"
## [1] "11 is not divisible by 3"
## [1] "12 is divisible by 3"
## [1] "13 is not divisible by 3"
## [1] "14 is not divisible by 3"
## [1] "15 is divisible by 3"
## [1] "16 is not divisible by 3"
## [1] "17 is not divisible by 3"
## [1] "18 is divisible by 3"
## [1] "19 is not divisible by 3"
## [1] "20 is not divisible by 3"
## [1] "21 is divisible by 3"
## [1] "22 is not divisible by 3"
## [1] "23 is not divisible by 3"
## [1] "24 is divisible by 3"
## [1] "25 is not divisible by 3"
## [1] "26 is not divisible by 3"
## [1] "27 is divisible by 3"
## [1] "28 is not divisible by 3"
## [1] "29 is not divisible by 3"
## [1] "30 is divisible by 3"
```

4. (6 points) Now make a vector of the numbers 1 to 30, and use an ifelse() statement to return a vector where 1 is returned if the original number was divisible by 3 and 0 if it was not

```r
n <- 1:30
vector <- ifelse(n%%3==0,
       1,
       0)
matrix <- as.matrix(ifelse(n%%3==0,
       1,
       0))

combined_vector <- rbind(n, ifelse(n%%3==0,
       1,
       0))

vector
```

```
##  [1] 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
```

```r
matrix
```

```
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    1
##  [4,]    0
##  [5,]    0
##  [6,]    1
##  [7,]    0
##  [8,]    0
##  [9,]    1
## [10,]    0
## [11,]    0
## [12,]    1
## [13,]    0
## [14,]    0
## [15,]    1
## [16,]    0
## [17,]    0
## [18,]    1
## [19,]    0
## [20,]    0
## [21,]    1
## [22,]    0
## [23,]    0
## [24,]    1
## [25,]    0
## [26,]    0
## [27,]    1
## [28,]    0
## [29,]    0
## [30,]    1
```

```r
combined_vector
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## n    1    2    3    4    5    6    7    8    9    10    11    12    13
##      0    0    1    0    0    1    0    0    1     0     0     1     0
##   [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
```

```
## n    14    15    16    17    18    19    20    21    22    23    24    25
##       0     1     0     0     1     0     0     1     0     0     1     0
##   [,26] [,27] [,28] [,29] [,30]
## n    26    27    28    29    30
##       0     1     0     0     1
```

5. (3 points) Consider the following loop. What is wrong with it? What could be done to fix it? n <- 0 while(n < 10){ print("hello world") n <- n - 1 }

When properly formatted, the function provided is an infinite loop. The while function prints so long as n is less than 10, and counts n down (n <- n-1) at the same time. Subsequently, n is continuously less than 10, and the function runs an infinite number of times.

```
n <- 0
while(n < 10){
  print("hello world")
  n = n+1
  }
```

```
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
## [1] "hello world"
```

To resolve the function, simply count n up (n <- n+1), and it will no longer run an infinite number of times.

6. (3 points) Write and test a switch function that returns "woof" when passed "dog", "meow" when passed "cat", and "???" when passed "fox"

```
n <- "dog"
switch(n, "dog"="woof", "cat"="meow", "fox"="???")
```

```
## [1] "woof"
```

```
n <- "cat"
switch(n, "dog"="woof", "cat"="meow", "fox"="???")
```

```
## [1] "meow"
```

```
n <- "fox"
switch(n, "dog"="woof", "cat"="meow", "fox"="???")
```

```
## [1] "???"
```

```
#What does the fox say?
```