L09 Strings

Data Science I (STAT 301-1)

YOUR NAME

Contents

Overview

The goal of this lab is to learn and understand string manipulation in R. You'll learn the basics of how strings work and how to create them by hand, but the focus of this chapter will be on regular expressions, or regexps for short (regex for even shorter). Regular expressions are useful because strings usually contain unstructured or semi-structured data, and regexps are a concise language for describing patterns in strings. You'd be surprised how often you'll need to use string manipulation in R, whether for text analysis, efficiently loading data files/objects, or extracting information from the internet or standardized forms.

These resources may help:

- `stringr` tidyverse homepage
- regex tutorial
- regex practice/game

Datasets

All datasets are found within downloaded R packages. New packages include `stringr`,`htmlwidgets`, and `stringi`.

Exercises

Please complete the following exercises. Be sure that your solutions are clearly indicated and that the document is neatly formatted.

Load Packages You should always begin by loading all necessary packages towards the beginning of your documents. Assume that that all necessary packages have been installed. User should be able to determine if a package needs to be installed either through knowing their R repository or an error message. **Your code should never have install commands.**

```
library(stringr)
library(tidyverse)
```

Special Note The output for this assignment becomes extremely long and hinders presentation. Please set `results = "hide"` for your R code chunks where appropriate. Use your best judgement in deciding when to set `results = "hide"`. Remember that you can break the R code for exercises into many separate code chunks, so you can hide the long parts of solution output without hiding the shorter parts.

Exercise 1 (Website: 14.2.5 Ex. 1)

In code that doesn't use `stringr`, you'll often see `paste()` and `paste0()`. What's the difference between the two functions? What `stringr` function are they equivalent to? How do `paste()` and `paste0()` differ in their handling of `NA`? **`paste()` concatenates vectors after converting to character, and will include spaces in between strings. `paste0()` will concatenate several vectors in a vectorized way, and it doesn't include spaces between strings. They both are equivalent to `strinr::str_c()`. Whereas `paste()` prints the NA character to "NA", `str_c()` does not unless you use `str_replace_na()`.**

Exercise 2 (Website: 14.2.5 Ex. 2)

In your own words, describe the difference between the `sep` and `collapse` arguments to `str_c()`. Then demonstrate this difference by using the provided strings, `x` and `y`, to produce the single string `"I heart data science!"` or `"I tolerate data science!"` (your choice) twice. Produce the string once using `sep` and once using `collapse`. **`sep` will insert the separator character just between the arguments of the function, while `collapse` will insert the character between any elements of the character vector to make a string of length one.**

```
# Provided strings
x <- "I"
y <- "data science!"

str_c(x, 'heart', y, sep = ' ')
```

```
## [1] "I heart data science!"
```

```
str_c(c(x, ' heart ', y), collapse = '')
```

```
## [1] "I heart data science!"
```

Exercise 3 (Website: 14.2.5 Ex. 3)

Use `str_length()` and `str_sub()` to extract the middle character from the provided strings. Make a decision (and explain your decision) regarding how to handle a string that has an even number of characters.

```
# Provided strings
st1 <- "data"
st2 <- "science"
st1_pos <- str_length(st1)/2
st2_pos <- str_length(st2)%/%2 + 1
str_sub(st1, st1_pos, st1_pos)
```

```
## [1] "a"
```

```
str_sub(st2, st2_pos, st2_pos)
```

```
## [1] "e"
```

For even lengthed strings, I chose the lower middle one for simplicity.

Exercise 4 (Website: 14.2.5 Ex. 4)

What does `str_wrap()` do? When might you want to use it? **`str_wrap()` formats strings into paragraphs, where you can specifiy width, indent, and exdent. This would be useful when reading files and displaying their contents nicely.**

Exercise 5 (Website: 14.2.5 Ex. 5)

What does `str_trim()` do? What's the opposite of `str_trim()`? **`str_trim()` removes whitespace from the start and end of a string. The opposite is `str_pad()`.**

Exercise 6 (Website: 14.3.1.1 Ex. 2)

How would you match the sequence `"'\`?

```
str_view("\"'\\", "\"'\\\\")
```

```
"'\
```

Exercise 7 (Website: 14.3.1.1 Ex. 3)

What patterns will the regular expression `\..\..\..` match? Provide a basic example of a string that it would match (and demonstrate). **It will match any pattern that is .x.x.x where x is any character.**

```
str_view(c('.x.y.z', '.a.b.'), "\\..\\..\\..")
```

```
.x.y.z
.a.b.
```

Exercise 8 (Website: 14.3.2.1 Ex. 1)

How would you match the literal string `"$^$"`?

```
str_view("$^$", "^\\$\\^\\$$")
```

```
$^$
```

Exercise 9 (Website: 14.3.2.1 Ex. 2)

Given the corpus of common words in `stringr::words`, create regular expressions to match all words that:

- start with "y".

```
str_view(stringr::words, "^y", match = TRUE)
```

year
yes
yesterday
yet
you
young

* end with “x”.

```
str_view(stringr::words, "x$", match = TRUE)
```

box
sex
six
tax

* are exactly three letters long (don’t cheat by using `str_length()`!).

```
str_view(stringr::words, "^...$", match = TRUE)
```

act
add
age
ago
air
all
and
any
arm
art
ask
bad
bag
bar
bed
bet
big
bit
box
boy
bus
but
buy
can
car
cat
cup
cut
dad
day
die
dog
dry
due
eat
egg
end
eye
far
few
fit
fly
for
fun
gas
get
god
guy
hit
hot
how
job
key
kid
lad
law
lay
leg
let
lie
lot
low
man
may
mrs
new
non
not
now
odd
off
old
one
out
own
pay
per
put
red
rid
run
say
see
set
sex
she
sir
sit
six
son
sun
tax
tea
ten
the
tie
too
top
try
two
use
war
way
wee
who
why
win
yes
yet
you

Since these lists are long, you might want to use the `match` argument of `str_view()` to display only the matching or non-matching words.

Exercise 10 (Website: 14.3.3.1 Ex. 1)

Given the corpus of common words in `stringr::words`, create regular expressions to match all words that:

- start with a vowel.
- contain only consonants (hint: think about matching "not"-vowels).
- end with `ed` but NOT with `eed`.
- end with `ing` OR `ise`.

```
#start with a vowel
str_subset(words, "^[aeiou]")
```

```
##   [1] "a"           "able"        "about"       "absolute"    "accept"
##   [6] "account"     "achieve"     "across"      "act"         "active"
##  [11] "actual"      "add"         "address"     "admit"       "advertise"
##  [16] "affect"      "afford"      "after"       "afternoon"   "again"
##  [21] "against"     "age"         "agent"       "ago"         "agree"
##  [26] "air"         "all"         "allow"       "almost"      "along"
##  [31] "already"     "alright"     "also"        "although"    "always"
##  [36] "america"     "amount"      "and"         "another"     "answer"
##  [41] "any"         "apart"       "apparent"    "appear"      "apply"
##  [46] "appoint"     "approach"    "appropriate" "area"        "argue"
##  [51] "arm"         "around"      "arrange"     "art"         "as"
##  [56] "ask"         "associate"   "assume"      "at"          "attend"
##  [61] "authority"   "available"   "aware"       "away"        "awful"
##  [66] "each"        "early"       "east"        "easy"        "eat"
##  [71] "economy"     "educate"     "effect"      "egg"         "eight"
##  [76] "either"      "elect"       "electric"    "eleven"      "else"
##  [81] "employ"      "encourage"   "end"         "engine"      "english"
##  [86] "enjoy"       "enough"      "enter"       "environment" "equal"
##  [91] "especial"    "europe"      "even"        "evening"     "ever"
##  [96] "every"       "evidence"    "exact"       "example"     "except"
## [101] "excuse"      "exercise"    "exist"       "expect"      "expense"
## [106] "experience"  "explain"     "express"     "extra"       "eye"
## [111] "idea"        "identify"    "if"          "imagine"     "important"
## [116] "improve"     "in"          "include"     "income"      "increase"
## [121] "indeed"      "individual"  "industry"    "inform"      "inside"
## [126] "instead"     "insure"      "interest"    "into"        "introduce"
## [131] "invest"      "involve"     "issue"       "it"          "item"
## [136] "obvious"     "occasion"    "odd"         "of"          "off"
## [141] "offer"       "office"      "often"       "okay"        "old"
## [146] "on"          "once"        "one"         "only"        "open"
## [151] "operate"     "opportunity" "oppose"      "or"          "order"
## [156] "organize"    "original"    "other"       "otherwise"   "ought"
## [161] "out"         "over"        "own"         "under"       "understand"
## [166] "union"       "unit"        "unite"       "university"  "unless"
## [171] "until"       "up"          "upon"        "use"         "usual"
```

```
#contain only consonants
str_view(words, "[aeiou]", match = FALSE)
```

by
dry
fly
mrs
try
why

```
#end with `ed` but NOT with `eed`
str_view(words, "[^e]ed$", match = TRUE)
```

bed
hundred
red

```
#end with `ing` or `ise`
str_view(words, "ing|ise$", match = TRUE)
```

advertise
bring
during
evening
exercise
king
meaning
morning
otherwise
practise
raise
realise
ring
rise
sing
single
surprise
thing

Exercise 11 (Website: 14.3.3.1 Ex. 2 & 3)

Use the corpus of common words in `stringr::words` to empirically verify:

- the rule "`i` before `e` except after `c`".
- whether `q` is always followed by `u`.

```
(cei <- str_subset(words, "cei"))
```

```
## [1] "receive"
```

```
(cie <- str_subset(words, "cie"))
```

```
## [1] "science" "society"
```

```
(ie <- str_subset(words, "[^c]ie"))
```

```
##  [1] "achieve"    "believe"    "brief"      "client"     "die"
##  [6] "experience" "field"      "friend"     "lie"        "piece"
## [11] "quiet"      "tie"        "view"
```

```
(ei <- str_subset(words, "[^c]ei"))
```

```
## [1] "weigh"
```

Based on the words in the `words` dataset, It does seem like "i before e except after c" holds, but there are many execptions. The word 'weigh' is the best example on an exception, because the words 'society' and 'science' have separate syllables for the 'i' and 'e'.

```
(qu <- str_subset(words, "qu"))
```

```
##  [1] "equal"    "quality"  "quarter"  "question" "quick"    "quid"
##  [7] "quiet"    "quite"    "require"  "square"
```

```
(q. <- str_subset(words, "q[^u]"))
```

```
## character(0)
```

There aren't any letters with a q that arent followed by a u, thus the rule holds pretty well.

Exercise 12 (Website: 14.3.3.1 Ex. 5)

Create a regular expression that will match telephone numbers as commonly written in the US. (Note that there are several valid solutions for this, depending on your definition of "as commonly written in the US.")
To match the phone number expression, I will use the (999)999-9999 format.

```
str_view('(999)999-9999', '\\(\\d\\d\\d\\)\\d\\d\\d-\\d\\d\\d\\d')
```

(999)999-9999

Exercise 13 (Website: 14.3.4.1 Ex. 2)

Describe in your own words what these regular expressions match. (Hint: Read carefully to determine whether these are a regular expression or a string that defines a regular expression.)

- `^.*$`
- `"\\{.+\\}"`
- `\d{4}-\d{2}-\d{2}`
- `"\\\\{4}"` **`^.*$` matches any string. This is because the `.*` means 0 or more of any character, so this regex essentially means any string that starts and ends with any character.**

`"\\{.+\\}"` matches any string with curly brackets around one or more character.

`\d{4}-\d{2}-\d{2}` matches a string of the form XXXX-XX-XX, where X is any digit.

"\\\\{4}" matches four forward slashes in a row

Exercise 14 (Website: 14.3.4.1 Ex. 3)

Create regular expressions to match all words that:

- start with three consonants.
- have three or more vowels in a row.
- have two or more vowel-consonant pairs in a row.

```
#start with three consonants: "^[^aieou]{3}"
str_view(words, "^[^aeiou]{3}", match = TRUE)
```

Christ
Christmas
dry
fly
mrs
scheme
school
straight
strategy
street
strike
strong
structure
system
three
through
throw
try
type
why

```
#have three or more vowels in a row: "[aeiou]{3}
str_view(words, "[aeiou]{3}", match = TRUE)
```

beauty
obvious
previous
quiet
serious
various

```
#have two or more consonant vowel pairs in a row: "([aeiou][^aeiou]){2,}"
str_view(words, "([aeiou][^aeiou]){2,}", match = TRUE)
```

absolute
agent
along
america
another
apart
apparent
authority
available
aware
away
balance
basis
become
before
begin
behind
benefit
business
character
closes
community
consider
cover
debate
decide
decision
definite
department
depend
design
develop
difference
difficult
direct
divide
document
during
economy
educate
elect
electric
eleven
encourage
environment
europe
even
evening
ever
every
evidence
exact
example
exercise
exist
family
figure
final
finance
finish
friday
future
general
govern
holiday
honest
hospital
however
identify
imagine
individual
interest
introduce
item
jesus
level
likely
limit
local
major
manage
meaning
measure
minister
minus
minute
moment
money
music
nature
necessary
never
notice
okay
open
operate
opportunity
organize
original
over
paper
paragraph
parent
particular
photograph
police
policy
politic
position
positive
power
prepare
present
presume
private
probable
process
produce
product
project
proper
propose
protect
provide
quality
realise
reason
recent
recognize
recommend
record
reduce
refer
regard
relation
remember
report
represent
result
return
saturday
second
secretary
secure
separate
seven
similar
specific
strategy
student
stupid
telephone
television
therefore
thousand
today
together
tomorrow
tonight
total
toward
travel
unit
unite
university
upon
visit
water
woman

Exercise 15 (Website: 14.3.5.1 Ex. 1)

Describe in words what these expressions will match:

- `(.)\1\1`
- `"(.)(.)\\2\\1"`
- `(..)\1`
- `"(.).\\1.\\1"` **`(.)\1\1` matches any string with a character repeated three times in a row.** `"(.)(.)\\2\\1"` **matches any string with two characters that appear in forward and reverse order.** `(..)\1` **matches two characters that are repeated, ex: 'abab'.** `"(.).\\1.\\1"` **matches strings of this form: "axaya", where x may or may not equal y.**

Exercise 16 (Website: 14.3.5.1 Ex. 2)

Construct regular expressions to match all words that:

- start and end with the same character.
- contain a repeated pair of letters (e.g., `church` contains the pair `ch` repeated twice).
- contain one letter repeated in at least three places (e.g., the word `eleven` contains `e` repeated in three places).

```
#start and end with same character
str_view(words, "^(.)((.*\\1$)|\\1?$)", match = TRUE)
```

a
america
area
dad
dead
depend
educate
else
encourage
engine
europe
evidence
example
excuse
exercise
expense
experience
eye
health
high
knock
level
local
nation
non
rather
refer
remember
serious
stairs
test
tonight
transport
treat
trust
window
yesterday

```
#contain a repeated pair of letters
str_view(words, "([A-Za-z][A-Za-z]).*\\1", match = TRUE)
```

appropriate
church
condition
decide
environment
london
paragraph
particular
photograph
prepare
pressure
remember
represent
require
sense
therefore
understand
whether

```
#contain one letter repeated in three places
str_view(words, "([a-z]).*\\1.*\\1", match = TRUE)
```

appropriate
available
believe
between
business
degree
difference
discuss
eleven
environment
evidence
exercise
expense
experience
individual
paragraph
receive
remember
represent
telephone
therefore
tomorrow

Exercise 17 (Website: 14.4.2 Ex. 1)

Use `stringr::words` to accomplish the following tasks. (Hint: You might want to change `stringr::words` to a tibble.)

- Identify which word(s) have the highest number of vowels. (As an example, the word `achieve` has four vowels.)
- Calculate the proportion of vowels for each word and display the dataset to determine which words have the highest proportion of vowels.
- Calculate the ratio of consonants to vowels for each word. What do you observe about the words that have the maximum possible ratio?

```
myWords <- tibble(words)

#highest number of vowels
myWords <- myWords %>%
  mutate(num_vowels = str_count(words, '[aeiou]')) %>%
```

```
  arrange(desc(num_vowels))
myWords
```

```
## # A tibble: 980 x 2
##    words       num_vowels
##    <chr>            <int>
##  1 appropriate          5
##  2 associate            5
##  3 available            5
##  4 colleague            5
##  5 encourage            5
##  6 experience           5
##  7 individual           5
##  8 television           5
##  9 absolute             4
## 10 achieve              4
## # ... with 970 more rows
```

```
#highest proportion of vowels
myWords <- myWords %>%
  mutate(prop_vowels = num_vowels/str_length(words)) %>%
  arrange(desc(prop_vowels))
myWords
```

```
## # A tibble: 980 x 3
##    words  num_vowels prop_vowels
##    <chr>       <int>       <dbl>
##  1 a               1       1
##  2 area            3       0.75
##  3 idea            3       0.75
##  4 europe          4       0.667
##  5 age             2       0.667
##  6 ago             2       0.667
##  7 air             2       0.667
##  8 die             2       0.667
##  9 due             2       0.667
## 10 eat             2       0.667
## # ... with 970 more rows
```

```
#highest proportion of consonants to vowels
myWords %>%
  mutate(num_cons = str_count(words, '[^aeiou]')) %>%
  mutate(prop_cons_to_vowels = num_cons/num_vowels) %>%
  arrange(desc(prop_cons_to_vowels))
```

```
## # A tibble: 980 x 5
##    words  num_vowels prop_vowels num_cons prop_cons_to_vowels
##    <chr>       <int>       <dbl>    <int>               <dbl>
##  1 by              0       0            2                 Inf
##  2 dry             0       0            3                 Inf
##  3 fly             0       0            3                 Inf
##  4 mrs             0       0            3                 Inf
##  5 try             0       0            3                 Inf
##  6 why             0       0            3                 Inf
##  7 Christ          1       0.167        5                   5
```

```
##  8 church          1       0.167        5                   5
##  9 pretty          1       0.167        5                   5
## 10 slight          1       0.167        5                   5
## # ... with 970 more rows
```

The words with the highest proportion of consonants to vowels are the words with no vowels.

Exercise 18 (Website: 14.4.3.1 Ex. 2)

From the Harvard `sentences` data, extract:

- the first word from each sentence.
- all words ending in `ing`.

```
#first word from each sentence
head(str_extract(sentences, "[A-Za-z][A-Za-z']*"))
```

```
## [1] "The"   "Glue"  "It's"  "These" "Rice"  "The"
```

```
#all words ending in 'ing'
end_with_ing <- str_extract(sentences, "\\b[A-Za-z]+ing\\b")
(end_with_ing <- unique(end_with_ing[!is.na(end_with_ing)]))
```

```
##  [1] "spring"    "evening"   "morning"   "winding"   "living"    "king"
##  [7] "Adding"    "making"    "raging"    "playing"   "sleeping"  "ring"
## [13] "glaring"   "sinking"   "dying"     "Bring"     "lodging"   "filing"
## [19] "wearing"   "wading"    "swing"     "nothing"   "sing"      "painting"
## [25] "walking"   "bring"     "shipping"  "puzzling"  "landing"   "thing"
## [31] "waiting"   "whistling" "timing"    "changing"  "drenching" "moving"
## [37] "working"
```

Exercise 19 (Website: 14.4.4.1 Ex. 1)

From the `sentences` data, find all words that come after a "number" like `one`, `two`, ..., `ten`. Pull out both the numbers and the words.

```
numword <- "\\b(one|two|three|four|five|six|seven|eight|nine|ten) +(\\w+)"
sentences[str_detect(sentences, numword)] %>%
  str_extract(numword)
```

```
##  [1] "seven books"   "two met"       "two factors"   "three lists"
##  [5] "seven is"      "two when"      "ten inches"    "one war"
##  [9] "one button"    "six minutes"   "ten years"     "two shares"
## [13] "two distinct"  "five cents"    "two pins"      "five robins"
## [17] "four kinds"    "three story"   "three inches"  "six comes"
## [21] "three batches" "two leaves"
```

Exercise 20 (Website: 14.5.1 Ex. 2)

What are the five most common words in `sentences`?

```
sentence_words <- tibble(word = unlist(str_extract_all(sentences, boundary("word"))))
sentence_words %>%
  mutate(word = str_to_lower(word)) %>%
  count(word) %>%
  arrange(desc(n))
```

```
## # A tibble: 1,904 x 2
##    word      n
##    <chr> <int>
##  1 the     751
##  2 a       202
##  3 of      132
##  4 to      123
##  5 and     118
##  6 in       87
##  7 is       81
##  8 was      66
##  9 on       60
## 10 with     51
## # ... with 1,894 more rows
```

'the', 'a', 'of', 'to', and 'and' are the five most common words.

Exercise 21 (Website: 14.5.1 Ex. 2)

Find the `stringi` functions that:

- count the number of words.
- find duplicated strings.
- generate random text. **Count the number of words: `stri_count_words()`. Find duplicate strings: `stri_duplicated()`. Generate random text: `stri_stri_rand_shuffle()`.**

Exercise 22

Describe what this code does in your own words.

```
dir(path = "../", pattern = "\\.Rmd$", recursive = TRUE)
```

This finds all R-markdown files in your current folder.

Challenge 1 (Website: 14.2.5 Ex. 6) – NOT REQUIRED.

Write a function that turns a character vector of the format `c("a", "b", "c")` into a string of the format `a, b, and c`. Think carefully about what your function should do if it is given a vector of length 0, 1, or 2.

Challenge 2 (Website: 14.4.5.1 Ex. 3) – NOT REQUIRED.

Switch the first and last letters of all the words in the corpus `stringr::words`. Which of the resulting strings are still words?