Stringr

J. Wall

2020-04-07

String Basics

Read sections 14.1 and 14.2 in (https://r4ds.had.co.nz/strings.html#string-basics (https://r4ds.had.co.nz/strings.html#string-basics)) Another resource is (https://stringr.tidyverse.org/articles/regular-expressions.html (https://stringr.tidyverse.org/articles/regular-expressions.html)).

Use of the escape character and string length

The "" says the thing inside is a string. is an escape character. If you want to include a “\” in a string as a character, you must use “\\”. Other common commands are “\n” (new line) and “\t” (tab).

```
"\"" # printed representation shows the escapes
```

```
## [1] "\""
```

```
# writeLines shows the raw contents of the string
writeLines("\"") # escape the quotes
```

```
## "
```

```
writeLines("'")
```

```
## '
```

```
writeLines('"')
```

```
## "
```

```
x <- c("\"", "\\")
x # printed representation shows the escapes
```

```
## [1] "\"" "\\"
```

```
writeLines(x)
```

```
## "
## \
```

Combining strings

There are several ways to find the lengths of strings and concatenate strings.

```
x <- c("Here is a string", "another string", "strong string")

length(x)
```

```
## [1] 3
```

```
nchar(x)
```

```
## [1] 16 14 13
```

```
str_length(x)
```

```
## [1] 16 14 13
```

```
# combine strings with str_c and paste
str_c("Here is a string", "another string", "strong string")
```

```
## [1] "Here is a stringanother stringstrong string"
```

```
# but
str_c(x)
```

```
## [1] "Here is a string" "another string"   "strong string"
```

```
str_c("Here is a string", "another string", "strong string", sep="; ")
```

```
## [1] "Here is a string; another string; strong string"
```

```
paste("Here is a string", "another string", "strong string")
```

```
## [1] "Here is a string another string strong string"
```

```
paste0("Here is a string", "another string", "strong string")
```

```
## [1] "Here is a stringanother stringstrong string"
```

```
str_c("O no!", x, " again")
```

```
## [1] "O no!Here is a string again" "O no!another string again"
## [3] "O no!strong string again"
```

```
paste("O no!", x, " again")
```

```
## [1] "O no! Here is a string  again" "O no! another string  again"
## [3] "O no! strong string  again"
```

```
# collapse into a single string
str_c("O no!", x, " again", collapse = ";")
```

```
## [1] "O no!Here is a string again;O no!another string again;O no!strong string again"
```

```
# use \ to write non-English character
mu <- "\u00b5"
mu
```

```
## [1] "µ"
```

-**Exercise 1:** How would you write the following expression? Use `writeLines()` to see the raw result.

```
## She said: "Let's all go!" \never mind! I said "I don't want to."
```

-**Exercise 2:** Use str_c to put (before the area codes followed by) and a space followed by the phone number where area_codes <- c(703, 863, 404, 202) and phone_nums <- c(5551212, 1234567, 7891234, 4799747)

```
## [1] "(703) 5551212" "(863) 1234567" "(404) 7891234" "(202) 4799747"
```

Subsetting strings

Use str_sub for string subset

```
x <- c("Apple", "Banana", "Pear")
str_sub(x, 1, 3) # get positions 1 to 3
```

```
## [1] "App" "Ban" "Pea"
```

```
# negative numbers count backwards from end
str_sub(x, -3, -1) # get positions 3 from end to last position
```

```
## [1] "ple" "ana" "ear"
```

```
# if the string is too short, will return all
str_sub(x,3,5)
```

```
## [1] "ple" "nan" "ar"
```

```
str_to_lower(str_sub(x,1,1))
```

```
## [1] "a" "b" "p"
```

```
# use this to replace items if desired
str_sub(x,1,1) <- str_to_lower(str_sub(x,1,1))
x
```

```
## [1] "apple"  "banana" "pear"
```

```
# can also use str_to_upper and str_to_title
x <- str_to_title(x)
x
```

```
## [1] "Apple"  "Banana" "Pear"
```

```
y <- c("apple", "eggplant", "banana")
#  uses locale argument for sorting, title
#  default is your current locale as provided by the operating system
str_sort(y, locale = "en") # English locale
```

```
## [1] "apple"    "banana"   "eggplant"
```

```
str_sort(y, locale = "haw") # Hawaiian locale
```

```
## [1] "apple"    "eggplant" "banana"
```

-Exercise 3: Look up `str_trim()` and `str_squish()`. Use them to replace the string

```
## [1] "   Help me to      get rid    of extra spaces?      "
```

with the following:

```
## [1] "Help me to      get rid    of extra spaces?      "
```

```
## [1] "   Help me to      get rid    of extra spaces?"
```

```
## [1] "Help me to      get rid    of extra spaces?"
```

```
## [1] "Help me to get rid of extra spaces?"
```

-Exercise 4: Use str_length() and str_sub() to extract the middle character from a string. What will you do if the string has an even number of characters? Test your function on the strings “hamburger” and “hotdog”.

```
## [1] "u"
```

```
## [1] "u"
```

```
## [1] "td"
```

Matching Patterns with Reg Expressions

Basic matches

Read 14.3 in online book, (https://r4ds.had.co.nz/strings.html#matching-patterns-with-regular-expressions (https://r4ds.had.co.nz/strings.html#matching-patterns-with-regular-expressions))

Special functions that help us when working with regular expressions are str_view() and str_view_all(). These commands will not knit to .pdf or .doc but only to .html. Most useful when trying out code in the console.

```
x <- c("Apple", "Banana", "Pear")
x
```

```
## [1] "Apple"  "Banana" "Pear"
```

```
str_view(x,"an") # matches exact string and returns first occurrence
```

Apple

Banana

Pear

```
str_view_all(x,"an")
```

Apple

Banana

Pear

```
str_view(x,".a.") # . matches any character
```

Apple

Banana

Pear

```
# how to find a . in the text then?
y <- c("tomdoc", "tom.doc", "tom.doc.com")
str_view(y,"\\.")
```

tomdoc

tom.doc

tom.doc.com

```
str_view_all(y,"\\.")
```

tomdoc

tom.doc

tom.doc.com

```
# how to find a \ in your text
#  needs \\\\
z <- c("jane\\doe", "jane.doe")
z
```

```
## [1] "jane\\doe" "jane.doe"
```

```
writeLines(z)
```

```
## jane\doe
## jane.doe
```

```
str_view(z,"\\\\")
```

jane\doe

jane.doe

-Exercise 5: How would you match the sequence "'\? Note this is a double quote, single quote, backslash and question mark. Build it up one piece at a time. Use it to identify that sequence contained in s2 below.

```
s <- "\"'\\?"
s
```

```
## [1] "\"'\\?"
```

```
writeLines(s)
```

```
## "'\?
```

```
s2 <- str_c("some stuff",s,"more!")
s2
```

```
## [1] "some stuff\"'\\?more!"
```

```
writeLines(s2)
```

```
## some stuff"'\?more!
```

some stuff"'\?more!

some stuff"'\?more!

some stuff"'\?more!

some stuff"'\?more!

Anchors

- ^ matches the start of the string

- $ matches the end of the string
- can use both to match exact entire string

mnemonic - starts with power, ends with money

```
x <- c("apple", "banana", "pear")
str_view(x,"a")
```

apple

banana

pear

```
str_view(x,"^a")
```

apple

banana

pear

```
str_view(x,"a$")
```

apple

banana

pear

```
y <- c("hair", "hair cut", "long hair")
y
```

```
## [1] "hair"      "hair cut"  "long hair"
```

```
str_view(y,"hair")
```

hair

hair cut

long hair

```
str_view(y,"^hair")
```

hair

hair cut

long hair

```
str_view(y,"hair$")
```

hair

hair cut

long hair

```
str_view(y,"^hair$")
```

hair

hair cut

long hair

How might we find all the words that are exactly eleven letters long (without using str_length())?

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

appropriate

environment

opportunity

responsible

-**Exercise 6:** Given the corpus of common words in stringr::words, create regular expressions that find all words that:

- Start with “y”
- End with “x”

year

yes

yesterday

yet

you

young

box

sex

six

tax

Character classes and alternatives

- . matches any character
- \d matches any digit
- \D matches any non-digit character
- \s matches any whitespace (space, tab, newline)
- \S matches any non-whitespace character
- [abc] matches a, b or c
- [^abc] matches anything except a,b, or c
- more at (https://stringr.tidyverse.org/articles/regular-expressions.html (https://stringr.tidyverse.org/articles/regular-expressions.html))

Alternatives - use | to pick between one or more alternative patterns.

```
str_view(c("gray","grey","whey"),"gr(e|a)y")
```

gray

grey

whey

```
str_view(c("gray","grey","whey"),"gr[ea]y")
```

gray

grey

whey

```
str_view(words, ".*(ei|ie).*", match = TRUE)
```

achieve

believe

brief

client

die

eight

either

experience

field

friend

lie

piece

quiet

receive

science

society

tie

view

weigh

We can use the [ ] to choose special characters as well in a regex.

```
str_view(c("abc", "a.c", "a*c", "a c"), "a[ ]c")
```

abc

a.c

a*c

a c

```
str_view(c("abc", "a.c", "a*c", "a c"), "a[*]c")
```

abc

a.c

a*c

a c

```
str_view(c("abc", "a.c", "a*c", "a c"), "a[.]c")
```

abc

a.c

a*c

a c

There are a few exceptions to this that must be handled with backslash escapes (for instance], -, ^ and \).

How might you find all words that start with either j, q, x or z?

```
str_view(words, "^[jqxz]", match = TRUE)
```

jesus

job

join

judge

jump

just

quality

quarter

question

quick

quid

quiet

quite

Is the letter q always followed by u?

```
str_view(words, "q[^u]", match = TRUE)
```

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

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

-**Exercise 7:** (2 points) Create regular expressions to find all words that:

- (1 pt) End with ed, but not with eed.
- (1/2 pt)End with ing or ise.
- (1/2 pt) Words that do not follow the rule “i before e except after c”.

```
## [1] "bed"     "hundred" "red"
```

```
##  [1] "advertise" "bring"     "during"    "evening"   "exercise"  "king"
##  [7] "meaning"   "morning"   "otherwise" "practise"  "raise"     "realise"
## [13] "ring"      "rise"      "sing"      "surprise"  "thing"
```

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

Repetition - control how many times a pattern matches

- ? 0 or 1
- + 1 or more
- * 0 or more
- {n} exactly n
- {n,} n or more
- {,m} at most m
- {n,m} between n and m

```
x <- "1888 is the longest year in Roman numerals: MDCCCLXXXVIII"
# view all the digits
str_view_all(x, "\\d")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x, "CC?")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view_all(x, "CC?")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x, "CC+")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x,"C[LX]+")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x,"I{2}")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x,"I{2,}")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x,"C{1,3}") #  greedy - matches longest string possible
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x,"C{1,3}?") # lazy - matches shortest string possible
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x,"C[LX]+?")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

```
str_view(x,"L[XV]+?")
```

1888 is the longest year in Roman numerals: MDCCCLXXXVIII

-**Exercise 8:**(2 pts) Create regular expressions to find all words that:

- (1/2 point)Find all words that start or end with the letter x.
- (1 point) Find all words that start with at least two vowels and end with at least two consonants.
- (1/2 point) Contains three or more vowels in a row.

```
## [1] "box" "sex" "six" "tax"
```

```
## [1] "authority" "each"      "early"     "east"      "easy"      "eight"
## [7] "ought"
```

```
## [1] "beauty"   "obvious"  "previous" "quiet"    "serious"  "various"
```

Grouping and Back-references

The parentheses create a numbered group. You can then use the group number ed to refer to a previous group using \1, \2 etc.

```
str_view(fruit,"(..)\\1", match = TRUE)
```

banana

coconut

cucumber

jujube

papaya

salal berry

For each of the following examples, see if you can figure out what it will match before we try it out. Were you correct?

- “(.)\1\1”
- “(.)\\1\\1”
- “(.)(.)\\2\\1”
- “(..)\1”
- “(.).\\1.\\1”
- "(.)(.)(.).*\\3\\2\\1"

```
# "(.)\1\1"
str_view_all(x, "(.)\1\1")
```

```
1888 is the longest year in Roman numerals: MDCCCLXXXVIII
```

```
# "(.)\\1\\1"
str_view_all(x, "(.)\\1\\1")
```

```
1888 is the longest year in Roman numerals: MDCCCLXXXVIII
```

```
# "(.)(.)\\2\\1"
str_view_all(fruit, "(.)(.)\\2\\1", match = TRUE)
```

```
bell pepper
chili pepper
```

```
# "(..)\1"
str_view_all(fruit, "(..).\1", match = TRUE)
```

```
# "(.).\\1.\\1"
str_view_all(fruit, "(.).\\1.\\1", match = TRUE)
```

```
banana
papaya
```

```
# "(.)(.)(.).*\\3\\2\\1"
str_view_all(fruit, "(.)(.)(.).*\\3\\2\\1", match = TRUE)
```

-**Exercise 9:** (2 points) Construct regular expressions to find words that:

- (1/2 point) Start and end with the same character
- (1/2 point) Contain a repeated pair of letters (e.g. “church” contains “ch” repeated twice.)
- (1 point) Contain one letter other than e that is repeated in at least three places (e.g. “appropriate” contains three “p”s.)

```
##  [1] "america"    "area"       "dad"        "dead"       "depend"
##  [6] "educate"    "else"       "encourage"  "engine"     "europe"
## [11] "evidence"   "example"    "excuse"     "exercise"   "expense"
## [16] "experience" "eye"        "health"     "high"       "knock"
## [21] "level"      "local"      "nation"     "non"        "rather"
## [26] "refer"      "remember"   "serious"    "stairs"     "test"
## [31] "tonight"    "transport"  "treat"      "trust"      "window"
## [36] "yesterday"
```

```
##  [1] "appropriate" "church"      "condition"   "decide"      "environment"
##  [6] "london"      "paragraph"   "particular"  "photograph"  "prepare"
## [11] "pressure"    "remember"    "represent"   "require"     "sense"
## [16] "therefore"   "understand"  "whether"
```

```
## [1] "appropriate" "available"   "business"    "discuss"     "environment"
## [6] "individual"  "paragraph"   "tomorrow"
```

#Tools

Read 14.4 - 14.7 in book (https://r4ds.had.co.nz/strings.html#tools (https://r4ds.had.co.nz/strings.html#tools))

Main functions of stringr:

- str_detect(x, pattern) tells you if there is any match to the pattern.
- str_count(x, pattern) counts the number of patterns.
- str_subset(x, pattern) extracts the matching components.
- str_locate(x, pattern) gives the position of the match.
- str_extract(x, pattern) extracts the text of the match.
- str_match(x, pattern) extracts parts of the match defined by parentheses.
- str_replace(x, pattern, replacement) replaces the matches with new text.
- str_split(x, pattern) splits up a string into multiple pieces.

Detect matches

str_detect() returns a logical vector the same length as the input

```
x <- c("calculus","statistics","data science")
str_detect(x,"t")
```

```
## [1] FALSE  TRUE  TRUE
```

```
sum(str_detect(x,"t"))
```

```
## [1] 2
```

```
mean(str_detect(x,"t"))
```

```
## [1] 0.6666667
```

When doing something complex, combine str_detects using logical arguments rather than making a complicated reg expression.

Find all words that don't contain a vowel.

```
words[!str_detect(words,"[aieou]")]
```

```
## [1] "by"  "dry" "fly" "mrs" "try" "why"
```

To pick out the elements that match, use str_subset(). Can use this in combination with filter to get the rows we want.

```
str_subset(x,"s$")
```

```
## [1] "calculus"   "statistics"
```

```
str_subset(words, "^s.*x$")
```

```
## [1] "sex" "six"
```

```
# or
as_tibble(words) %>%
  filter(str_detect(words,"^s.*x$"))
```

```
## # A tibble: 2 x 1
##   value
##   <chr>
## 1 sex
## 2 six
```

str_count gives number of matches in a string

```
x
```

```
## [1] "calculus"     "statistics"   "data science"
```

```
str_count(x,"s")
```

```
## [1] 1 3 1
```

```
mean(str_count(x,"s"))
```

```
## [1] 1.666667
```

```
str_count("bananana","ana") # matches dont overlap
```

```
## [1] 2
```

```
# use with mutate
as_tibble(words) %>%
  rename(word = value) %>%
  mutate(
    vowels = str_count(word,"[aeiou]"),
    consonants = str_count(word,"[^aeiou]")
  )
```

```
## # A tibble: 980 x 3
##    word     vowels consonants
##    <chr>     <int>      <int>
##  1 a             1          0
##  2 able          2          2
##  3 about         3          2
##  4 absolute      4          4
##  5 accept        2          4
##  6 account       3          4
##  7 achieve       4          3
##  8 across        2          4
##  9 act           1          2
## 10 active        3          3
## # ... with 970 more rows
```

-**Exercise 10:** (1 1/2 points) Use `str_detect()` to find any words that contain at least one of each different vowel? How about at least 4 of the 5 vowels?

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

```
##  [1] "absolute"    "appropriate" "associate"   "authority"   "colleague"
##  [6] "continue"    "encourage"   "introduce"   "organize"    "previous"
## [11] "question"    "relation"    "serious"     "situate"     "various"
```

-**Exercise 11:** (1 1/2 points) Switch the first and last letters in each word of words. Which of those strings are still words?

```
##  [1] "a"          "america"    "area"       "dad"        "dead"
##  [6] "deal"       "dear"       "depend"     "dog"        "educate"
## [11] "else"       "encourage"  "engine"     "europe"     "evidence"
## [16] "example"    "excuse"     "exercise"   "expense"    "experience"
## [21] "eye"        "god"        "health"     "high"       "knock"
## [26] "lead"       "level"      "local"      "nation"     "no"
## [31] "non"        "on"         "rather"     "read"       "refer"
## [36] "remember"   "serious"    "stairs"     "test"       "tonight"
## [41] "transport"  "treat"      "trust"      "window"     "yesterday"
```

##Extract matches

str_extract extracts the first match from each element

```
head(sentences)
```

```
## [1] "The birch canoe slid on the smooth planks."
## [2] "Glue the sheet to the dark blue background."
## [3] "It's easy to tell the depth of a well."
## [4] "These days a chicken leg is a rare dish."
## [5] "Rice is often served in round bowls."
## [6] "The juice of lemons makes fine punch."
```

```
colors <- c("red", "Red","orange", "yellow", "green", "blue", "purple")
color_match <- str_c(colors, collapse = "| ")
color_match
```

```
## [1] "red| Red| orange| yellow| green| blue| purple"
```

```
# select the sentences that contain a color, and then extract the color to figure out which one it is
has_color <- str_subset(sentences, color_match)
matches <- str_extract(has_color, color_match)
# select all the sentences that have more than 1 match
more <- sentences[str_count(sentences, color_match) > 1]
str_view_all(more, color_match)
```

It is hard to erase `blue` or `red` ink.

The `green` light in the brown box flicke`red`.

The sky in the west is tinged with `orange` `red`.

```
str_extract(more,color_match) # gets first color mentioned in each sentence
```

```
## [1] " blue"   " green"  " orange"
```

```
str_extract_all(more, color_match) # returns them all in a list
```

```
## [[1]]
## [1] " blue" "red"
##
## [[2]]
## [1] " green" "red"
##
## [[3]]
## [1] " orange" "red"
```

```
str_extract_all(more, color_match, simplify = TRUE)
```

```
##      [,1]     [,2]
## [1,] " blue"   "red"
## [2,] " green"  "red"
## [3,] " orange" "red"
```

- **Exercise 12: * Modify the regex so it no longer picks up the word flickered as a color. First, show all the sentences that contain at least one color with the word highlighted using `str_view_all()`. Then use `str_extract_all()` to produce an array with the colors that appear multiple times in a sentence.

Glue the sheet to the dark `blue` background.

Two `blue` fish swam in the tank.

A wisp of cloud hung in the `blue` air.

Leaves turn brown and `yellow` in the fall.

The spot on the blotter was made by `green` ink.

The sofa cushion is `red` and of light weight.

The sky that morning was clear and bright `blue`.

A `blue` crane is a tall wading bird.

It is hard to erase `blue` or `red` ink.

The lamp shone with a steady `green` flame.

The box is held by a bright `red` snapper.

The houses are built of `red` clay bricks.

The `red` tape bound the smuggled food.

Hedge apples may stain your hands `green`.

The plant grew large and `green` in the window.

The `purple` tie was ten years old.

Bathe and relax in the cool `green` grass.

The lake sparkled in the `red` hot sun.

Mark the spot with a sign painted `red`.

The couch cover and hall drapes were `blue`.

A man in a `blue` sweater sat at the desk.

The small `red` neon lamp went out.

Paint the sockets in the wall dull `green`.

Wake and rise, and step into the `green` outdoors.

The `green` light in the brown box flickered.

Tear a thin sheet from the `yellow` pad.

The sky in the west is tinged with `orange` `red`.

The `red` paper brightened the dim stage.

The big `red` apple fell to the ground.

```
##      [,1]     [,2]
## [1,] " blue"   " red"
## [2,] " orange" " red"
```

It is hard to erase `blue` or `red` ink.

The sky in the west is tinged with `orange` `red`.

##Grouped matches

Use parentheses to extract parts of a complex match

```
# use a space followed by one or more non-spaces to mean a word
noun <- "(a|the|A|The) ([^ ]+)"
has_noun <- sentences %>%
  str_subset(noun) %>%
  head(7)
has_noun
```

```
## [1] "The birch canoe slid on the smooth planks."
## [2] "Glue the sheet to the dark blue background."
## [3] "It's easy to tell the depth of a well."
## [4] "These days a chicken leg is a rare dish."
## [5] "The juice of lemons makes fine punch."
## [6] "The box was thrown beside the parked truck."
## [7] "The hogs were fed chopped corn and garbage."
```

```
has_noun %>%
  str_extract(noun) # gives the complete match
```

```
## [1] "The birch" "the sheet" "the depth" "a chicken" "The juice" "The box"
## [7] "The hogs"
```

```
has_noun %>%
  str_match(noun) # returns an array - first column the complete match
```

```
##      [,1]        [,2]  [,3]
## [1,] "The birch" "The" "birch"
## [2,] "the sheet" "the" "sheet"
## [3,] "the depth" "the" "depth"
## [4,] "a chicken" "a"   "chicken"
## [5,] "The juice" "The" "juice"
## [6,] "The box"   "The" "box"
## [7,] "The hogs"  "The" "hogs"
```

```
                # other column for each group of the match
```

- **Exercise 13:** Find all words that come after a “number” like “one”, “two”, … “twelve”. Pull out both the number and the word.

```
##       [,1]             [,2]     [,3]
##  [1,] " seven books"   " seven" "books"
##  [2,] " two met"       " two"   "met"
##  [3,] " two factors"   " two"   "factors"
##  [4,] " three lists"   " three" "lists"
##  [5,] " seven is"      " seven" "is"
##  [6,] " two when"      " two"   "when"
##  [7,] " ten inches."   " ten"   "inches."
##  [8,] " one war"       " one"   "war"
##  [9,] " one button"    " one"   "button"
## [10,] " six minutes."  " six"   "minutes."
## [11,] " ten years"     " ten"   "years"
## [12,] " two shares"    " two"   "shares"
## [13,] " two distinct"  " two"   "distinct"
## [14,] " five cents"    " five"  "cents"
## [15,] " two pins"      " two"   "pins"
## [16,] " five robins."  " five"  "robins."
## [17,] " four kinds"    " four"  "kinds"
## [18,] " three story"   " three" "story"
## [19,] " three inches"  " three" "inches"
## [20,] " six comes"     " six"   "comes"
## [21,] " three batches" " three" "batches"
## [22,] " two leaves."   " two"   "leaves."
```

##Replacing Matches

str_replace replaces the first occurrence while str_replace_all replaces them all

```
x <- c("apple", "pear", "banana")
str_replace(x, "[aeiou]", "-")
```

```
## [1] "-pple"  "p-ar"   "b-nana"
```

```
str_replace_all(x, "[aeiou]", "-")
```

```
## [1] "-ppl-"  "p--r"   "b-n-n-"
```

```
x <- c("1 house", "2 cars", "3 people")
# do multiple replacements with one vector
str_replace_all(x, c("1" = "one", "2" = "two", "3" = "three"))
```

```
## [1] "one house"    "two cars"     "three people"
```

- **Exercise 14:** View all sentences that contain the word “good” or the word “bad”. Get the sentence numbers where those words occur. Use `str_replace_all()` to replace the word “bad” with the word “horrible” and the word “good” with the word “great”. Look at the sentence

numbers you found before to verify the words were replaced correctly.

The wrist was badly strained and hung limp.

We frown when events take a bad turn.

We admire and love a good cook.

Sell your gift to a buyer at a good gain.

These pills do less good than others.

It takes a good trap to capture a bear.

Much of the story makes good sense.

The price is fair for a good antique clock.

The water in this well is a source of good health.

The team with the best timing looks good.

To send it. now in large amounts is bad.

The good book informs of what we ought to know.

It was a bad error on the part of the new judge.

```
##  [1] "The wrist was badly strained and hung limp."
##  [2] "We frown when events take a bad turn."
##  [3] "We admire and love a good cook."
##  [4] "Sell your gift to a buyer at a good gain."
##  [5] "These pills do less good than others."
##  [6] "It takes a good trap to capture a bear."
##  [7] "Much of the story makes good sense."
##  [8] "The price is fair for a good antique clock."
##  [9] "The water in this well is a source of good health."
## [10] "The team with the best timing looks good."
## [11] "To send it. now in large amounts is bad."
## [12] "The good book informs of what we ought to know."
## [13] "It was a bad error on the part of the new judge."
```

```
##  [1]  36 139 187 195 259 360 389 460 508 612 656 668 692
```

```
##  [1] "The wrist was horriblely strained and hung limp."
##  [2] "We frown when events take a horrible turn."
##  [3] "We admire and love a great cook."
##  [4] "Sell your gift to a buyer at a great gain."
##  [5] "These pills do less great than others."
##  [6] "It takes a great trap to capture a bear."
##  [7] "Much of the story makes great sense."
##  [8] "The price is fair for a great antique clock."
##  [9] "The water in this well is a source of great health."
## [10] "The team with the best timing looks great."
## [11] "To send it. now in large amounts is horrible."
## [12] "The great book informs of what we ought to know."
## [13] "It was a horrible error on the part of the new judge."
```

##String Splitting

```
sentences %>%
  head(5) %>%
  str_split(" ")
```

```
## [[1]]
## [1] "The"     "birch"   "canoe"   "slid"    "on"      "the"     "smooth"
## [8] "planks."
##
## [[2]]
## [1] "Glue"        "the"         "sheet"       "to"          "the"
## [6] "dark"        "blue"        "background."
##
## [[3]]
## [1] "It's"  "easy"  "to"    "tell"  "the"   "depth" "of"    "a"     "well."
##
## [[4]]
## [1] "These"   "days"    "a"       "chicken" "leg"     "is"      "a"
## [8] "rare"    "dish."
##
## [[5]]
## [1] "Rice"   "is"     "often"  "served" "in"     "round"  "bowls."
```

```
sentences %>%
  head(5) %>%
  str_split(" ") %>%
  .[[1]]
```

```
## [1] "The"     "birch"   "canoe"   "slid"    "on"      "the"     "smooth"
## [8] "planks."
```

```
# Use simplify = TRUE to make into array rather than a list.
sentences %>%
  head(5) %>%
  str_split(" ", simplify = TRUE)
```

```
##      [,1]    [,2]    [,3]    [,4]      [,5]  [,6]    [,7]     [,8]
## [1,] "The"   "birch" "canoe" "slid"    "on"  "the"   "smooth" "planks."
## [2,] "Glue"  "the"   "sheet" "to"      "the" "dark"  "blue"   "background."
## [3,] "It's"  "easy"  "to"    "tell"    "the" "depth" "of"     "a"
## [4,] "These" "days"  "a"     "chicken" "leg" "is"    "a"      "rare"
## [5,] "Rice"  "is"    "often" "served"  "in"  "round" "bowls." ""
##      [,9]
## [1,] ""
## [2,] ""
## [3,] "well."
## [4,] "dish."
## [5,] ""
```

```
# Specify the number of pieces to split into
sentences %>%
  head(5) %>%
  str_split(" ", n = 3, simplify = TRUE)
```

```
##      [,1]    [,2]    [,3]
## [1,] "The"   "birch" "canoe slid on the smooth planks."
## [2,] "Glue"  "the"   "sheet to the dark blue background."
## [3,] "It's"  "easy"  "to tell the depth of a well."
## [4,] "These" "days"  "a chicken leg is a rare dish."
## [5,] "Rice"  "is"    "often served in round bowls."
```

```
# Instead of splitting up strings by patterns,
# you can also split up by character, line,
# sentence and word boundarys:
sentences %>%
  head(5) %>%
  str_split(boundary("word"), simplify = TRUE)
```

```
##      [,1]    [,2]    [,3]    [,4]      [,5]  [,6]    [,7]     [,8]
## [1,] "The"   "birch" "canoe" "slid"    "on"  "the"   "smooth" "planks"
## [2,] "Glue"  "the"   "sheet" "to"      "the" "dark"  "blue"   "background"
## [3,] "It's"  "easy"  "to"    "tell"    "the" "depth" "of"     "a"
## [4,] "These" "days"  "a"     "chicken" "leg" "is"    "a"      "rare"
## [5,] "Rice"  "is"    "often" "served"  "in"  "round" "bowls"  ""
##      [,9]
## [1,] ""
## [2,] ""
## [3,] "well"
## [4,] "dish"
## [5,] ""
```

Since the regular expression is passed to the regex() functions, there are options that can be included by calling regex() explicitly

```
x <- c("Please do NOT yell", "DO NOT YELL", "Yellow submarine")
str_view(x,"yell")
```

Please do NOT `yell`

DO NOT YELL

Yellow submarine

```
str_view(x, regex("yell"))
```

Please do NOT `yell`

DO NOT YELL

Yellow submarine

```
str_view(x, regex("yell", ignore_case = TRUE))
```

Please do NOT `yell`

DO NOT `YELL`

`Yell`ow submarine

Other options:

- multiline = TRUE allows ^ and $ to match the start and end of each line instead of each string.
- comments = TRUE lets you use comments and white space in a regex
- dotall = TRUE lets . match everything including
- fixed() matches exactly what is in the "" and is very fast, but may not work how you want with non-English characters.

Many more functions available in stringi.

- **Exercise 15:** Install and library the ISLR package. Then take Names <- row.names(College) . Show the first twenty rows that result if you split Names at each space into 3 parts. Do the same by splitting the first 20 rows on every word boundary.

```
##       [,1]                [,2]           [,3]
##  [1,] "Abilene"           "Christian"    "University"
##  [2,] "Adelphi"           "University"   ""
##  [3,] "Adrian"            "College"      ""
##  [4,] "Agnes"             "Scott"        "College"
##  [5,] "Alaska"            "Pacific"      "University"
##  [6,] "Albertson"         "College"      ""
##  [7,] "Albertus"          "Magnus"       "College"
##  [8,] "Albion"            "College"      ""
##  [9,] "Albright"          "College"      ""
## [10,] "Alderson-Broaddus" "College"      ""
## [11,] "Alfred"            "University"   ""
## [12,] "Allegheny"         "College"      ""
## [13,] "Allentown"         "Coll."        "of St. Francis de Sales"
## [14,] "Alma"              "College"      ""
## [15,] "Alverno"           "College"      ""
## [16,] "American"          "International" "College"
## [17,] "Amherst"           "College"      ""
## [18,] "Anderson"          "University"   ""
## [19,] "Andrews"           "University"   ""
## [20,] "Angelo"            "State"        "University"
```

```
##       [,1]        [,2]            [,3]         [,4] [,5]      [,6] [,7]
##  [1,] "Abilene"   "Christian"     "University" ""   ""        ""   ""
##  [2,] "Adelphi"   "University"    ""           ""   ""        ""   ""
##  [3,] "Adrian"    "College"       ""           ""   ""        ""   ""
##  [4,] "Agnes"     "Scott"         "College"    ""   ""        ""   ""
##  [5,] "Alaska"    "Pacific"       "University" ""   ""        ""   ""
##  [6,] "Albertson" "College"       ""           ""   ""        ""   ""
##  [7,] "Albertus"  "Magnus"        "College"    ""   ""        ""   ""
##  [8,] "Albion"    "College"       ""           ""   ""        ""   ""
##  [9,] "Albright"  "College"       ""           ""   ""        ""   ""
## [10,] "Alderson"  "Broaddus"      "College"    ""   ""        ""   ""
## [11,] "Alfred"    "University"    ""           ""   ""        ""   ""
## [12,] "Allegheny" "College"       ""           ""   ""        ""   ""
## [13,] "Allentown" "Coll"          "of"         "St" "Francis" "de" "Sales"
## [14,] "Alma"      "College"       ""           ""   ""        ""   ""
## [15,] "Alverno"   "College"       ""           ""   ""        ""   ""
## [16,] "American"  "International" "College"    ""   ""        ""   ""
## [17,] "Amherst"   "College"       ""           ""   ""        ""   ""
## [18,] "Anderson"  "University"    ""           ""   ""        ""   ""
## [19,] "Andrews"   "University"    ""           ""   ""        ""   ""
## [20,] "Angelo"    "State"         "University" ""   ""        ""   ""
```