regex_hw2

Emre Batarlar

Regex Homework 2

Hi everyone! I am Emre. I will talk about key functions that can detect the presence or absence of a match and count the number of matches.

as always I am using tidyverse package for my functions and babynames package for some examples

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.3     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(babynames)
library(ggplot2)
```

Meet **str_detect()**, a function spotting patterns in data.

```r
str_detect(c("a", "b", "c"), "[aeiou]")
```

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

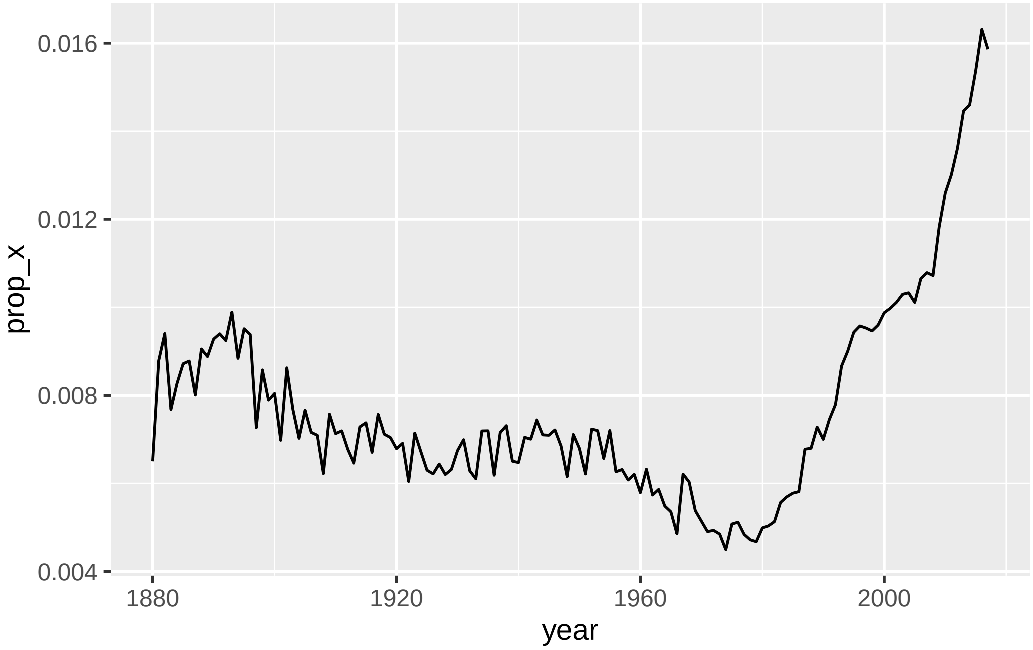
Since `str_detect()` returns a logical vector of the same length as the initial vector, it pairs well with `filter()`. For example, this code finds all the most popular names containing a lower-case "x":

```
babynames |>
  filter(str_detect(name, "x")) |>
  count(name,wt=n, sort = TRUE)
```

```
# A tibble: 974 x 2
   name            n
   <chr>       <int>
 1 Alexander  665492
 2 Alexis     399551
 3 Alex       278705
 4 Alexandra  232223
 5 Max        148787
 6 Alexa      123032
 7 Maxine     112261
 8 Alexandria  97679
 9 Maxwell     90486
10 Jaxon       71234
# i 964 more rows
```

We can also use `str_detect()` with `summarize()` by pairing it with `sum()` or `mean()`: `sum(str_detect(x, pattern))` tells you the number of observations that match and `mean(str_detect(x, pattern))` tells you the proportion that match. For example, the following snippet computes and visualizes the proportion of baby names[4] that contain "x", broken down by year. It looks like they've radically increased in popularity lately!

```
babynames |>
  group_by(year) |>
  summarize(prop_x = mean(str_detect(name, "x"))) |>
  ggplot(aes(x = year, y = prop_x)) +
  geom_line()
```



Now, **str_count()** steps in, counting occurrences

```
x <- c("apple", "banana", "pear")
str_count(x, "p")
```

```
[1] 2 0 1
```

Note that each match starts at the end of the previous match, i.e. regex matches never overlap. For example, in "abababa", how many times will the pattern "aba" match? Regular expressions say two, not three:

```
str_count("abababa", "aba")
```

```
[1] 2
```

```
str_view("abababa", "aba")
```

```
[1] | <aba>b<aba>
```

It's natural to use str_count() with mutate(). The following example uses str_count() with character classes to count the number of vowels and consonants in each name.

```
babynames |>
  count(name) |>
  mutate(
    vowels = str_count(name, "[aeiou]"),
    consonants = str_count(name, "[^aeiou]")
  )
```

```
# A tibble: 97,310 x 4
   name          n vowels consonants
   <chr>     <int>  <int>      <int>
 1 Aaban        10      2          3
 2 Aabha         5      2          3
 3 Aabid         2      2          3
 4 Aabir         1      2          3
 5 Aabriella     5      4          5
 6 Aada          1      2          2
 7 Aadam        26      2          3
 8 Aadan        11      2          3
 9 Aadarsh      17      2          5
10 Aaden        18      2          3
# i 97,300 more rows
```

If you look closely, you'll notice that there's something off with our calculations: "Aaban" contains three "a"s, but our summary reports only two vowels. That's because regular expressions are case sensitive. There are three ways we could fix this:

- Add the upper case vowels to the character class: `str_count(name, "[aeiouAEIOU]")`.
- Tell the regular expression to ignore case: `str_count(name, regex("[aeiou]", ignore_case = TRUE))`. We'll talk about more in Section 15.5.1.
- Use str_to_lower() to convert the names to lower case: `str_count(str_to_lower(name), "[aeiou]")`.

```
babynames |>
  count(name) |>
  mutate(
    name = str_to_lower(name),
    vowels = str_count(name, "[aeiou]"),
    consonants = str_count(name, "[^aeiou]")
  )
```

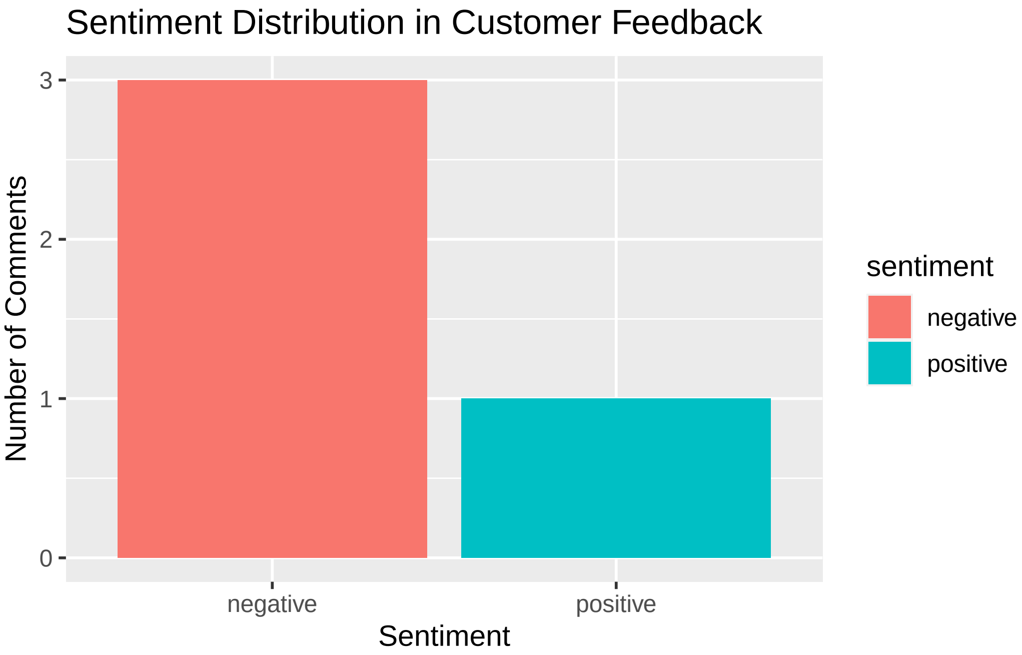
```
# A tibble: 97,310 x 4
   name           n vowels consonants
   <chr>      <int>  <int>      <int>
 1 aaban         10      3          2
 2 aabha          5      3          2
 3 aabid          2      3          2
 4 aabir          1      3          2
 5 aabriella      5      5          4
 6 aada           1      3          1
 7 aadam         26      3          2
 8 aadan         11      3          2
 9 aadarsh       17      3          4
10 aaden         18      3          2
# i 97,300 more rows
```

BONUS Example

```
# Sample dataset
customer_feedback <- tibble(
  comment = c("Great service! Very satisfied.",
              "Not happy with the product.",
              "Amazing experience. Will buy again.",
              "Disappointed with the delivery time.")
)

# Categorize comments as positive or negative using str_detect()
customer_feedback <- customer_feedback |>
  mutate(sentiment = ifelse(str_detect(comment, "great|amazing|satisfied"), "positive", "n

# Count the occurrences of positive and negative sentiments using str_count()
sentiment_counts <- customer_feedback |>
  group_by(sentiment) |>
  summarise(comment_count = n())

# Visualize the sentiment distribution using ggplot2
sentiment_distribution <- ggplot(sentiment_counts, aes(x = sentiment, y = comment_count, f
  geom_bar(stat = "identity") +
  labs(title = "Sentiment Distribution in Customer Feedback",
       x = "Sentiment",
       y = "Number of Comments")

# Display the results and visualization
sentiment_counts
```

```
# A tibble: 2 x 2
  sentiment comment_count
  <chr>             <int>
1 negative              3
2 positive              1
```

```
sentiment_distribution
```


Sentiment Distribution in Customer Feedback
Number of Comments
negative
positive
Sentiment
sentiment
negative
positive