STATS 220

Working with text

String manipulation

- create

Fixed pattern

Create strings with ' or "

```
library(tidyverse) # library(stringr)
string <- "lzDHk3orange2o5ghte"
string
```

```
#> [1] "lzDHk3orange2o5ghte"
```

```
fruit <- c("cherry", "banana")
fruit
```

```
#> [1] "cherry" "banana"
```


- create
- join

Fixed pattern

Join strings

```
c(string, fruit)
```

```
#> [1] "lzDHk3orange2o5ghte" "cherry"              "banana"
```

```
str_c(string, fruit, sep = ", ")
```

```
#> [1] "lzDHk3orange2o5ghte, cherry" "lzDHk3orange2o5ghte, banana"
```

```
str_c(string, fruit, collapse = ", ")
```

```
#> [1] "lzDHk3orange2o5ghtecherry, lzDHk3orange2o5ghtebanana"
```

- create
- join
- **detect**

Fixed pattern

Determine which strings match a pattern

```
str_detect(string, "orange")
```

```
#> [1] TRUE
```

```
str_detect(fruit, "orange")
```

```
#> [1] FALSE FALSE
```


- create
- join
- detect
- **locate**

Fixed pattern

Find the positions of matches

```
str_locate(string, "orange")
```

```
#>      start end
#> [1,]     7  12
```

```
str_locate(fruit, "orange")
```

```
#>      start end
#> [1,]    NA  NA
#> [2,]    NA  NA
```


- create
- join
- detect
- locate
- **extract**

Fixed pattern

Extract the content of matches

```
str_sub(string, 7, 12)
```

```
#> [1] "orange"
```

```
str_extract(string, "orange")
```

```
#> [1] "orange"
```

```
str_extract(fruit, "orange")
```

```
#> [1] NA NA
```


- create
- join
- detect
- locate
- extract
- **replace**

Fixed pattern

Replace matches with new values

```
str_replace(string, "orange", "apple")
```

```
#> [1] "lzDHk3apple2o5ghte"
```

```
str_replace(fruit, "orange", "apple")
```

```
#> [1] "cherry" "banana"
```

Regular expressions (aka regex/regexp)

an extremely concise language for describing patterns

- period

Regex

. matches any character (except a newline)

```
str_extract(string, "o....e")
```

```
#> [1] "orange"
```

```
str_extract_all(string, "o....e")
```

```
#> [[1]]
#> [1] "orange" "o5ghte"
```


- period
- qualifier

Regex

repetition

```
str_extract_all(string, "o.{4}e")
```

```
#> [[1]]
#> [1] "orange" "o5ghte"
```

```
str_extract_all(string, "o.*e")
```

```
#> [[1]]
#> [1] "orange2o5ghte"
```

```
str_extract_all(string, "o.*?e")
```

```
#> [[1]]
#> [1] "orange" "o5ghte"
```


- period
- qualifier

Regex

repetition

- `?`: 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

- period
- qualifier
- **escape**

Regex

if `.` matches any character, how to match a literal `"."`?

- use the backslash `\` to escape special behaviour `\.`
- `\` is also used as an escape symbol in strings
- end up using `"\\."` to create the regular expression `\.`

```
str_view_all(string, "o\\.{4}e")
```

> `lzDHk3orange2o5ghte`

```
str_view_all("a.b.c", "\\.")
```

> `a.b.c`

- period
- qualifier
- escape
- **meta**

Regex

- `\d`: matches any digit. *(metacharacter)*
- `\s`: matches any whitespace (e.g. space, tab `\t`, newline `\n`).
- `[abc]`: matches a, b, or c. *(make character classes by hand)*

```
str_view_all(string, "\\d")
```

> `lzDHk3orange2o5ghte`

```
str_view_all(string, "[0-9]")
```

> `lzDHk3orange2o5ghte`

- period
- qualifier
- escape
- **meta**

Regex

- `\D`: matches anything except digits.
- `\S`: matches anything except whitespaces.
- `[^abc]`: matches anything except a, b, or c.

```
str_view_all(string, "\\D")
```

> `lzDHk3orange2o5ghte`

```
str_view_all(string, "[^0-9]")
```

> `lzDHk3orange2o5ghte`

- period
- qualifier
- escape
- meta
- **POSIX**

Regex

- `[:digit:]`: matches any digit.
- `[:space:]`: matches any whitespace.
- `[:alpha:]`: matches any alphabetic character.
- more on ?base::regex

```
str_view_all(string, "[:digit:]")
```

> `lzDHk3orange2o5ghte`

```
str_view_all(string, "[:alpha:]")
```

> `lzDHk3orange2o5ghte`

- period
- qualifier
- escape
- meta
- POSIX
- **anchor**

Regex

- `^` matches the start of the string.
- `$` matches the end of the string.

```
str_view_all(fruit, "a")
```

- cherry
- banana

```
str_view_all(fruit, "a$")
```

- cherry
- banana

```
str_view_all(fruit, "^a")
```

- cherry
- banana

Working with strings in a tibble

Gapminder

```
gapminder <- read_rds("data/gapminder.rds") %>%
  group_by(country) %>%
  slice_tail() %>%
  ungroup()
gapminder
```

```
#> # A tibble: 142 x 6
#>   country     continent  year lifeExp      pop gdpPercap
#>   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
#> 1 Afghanistan Asia       2007    43.8 31889923      975.
#> 2 Albania     Europe     2007    76.4  3600523     5937.
#> 3 Algeria     Africa     2007    72.3 33333216     6223.
#> 4 Angola      Africa     2007    42.7 12420476     4797.
#> 5 Argentina   Americas   2007    75.3 40301927    12779.
#> 6 Australia   Oceania    2007    81.2 20434176    34435.
#> # … with 136 more rows
```


Gapminder

- `"i.a"` matches “ina”, “ica”, “ita”, and more.

```
gapminder %>%
  filter(str_detect(country, "i.a"))
```

```
#> # A tibble: 16 x 6
#>    country          continent  year lifeExp     pop gdpPercap
#>    <fct>            <fct>     <int>   <dbl>   <int>     <dbl>
#>  1 Argentina        Americas   2007    75.3  4.03e7    12779.
#>  2 Bosnia and Her… Europe     2007    74.9  4.55e6     7446.
#>  3 Burkina Faso     Africa     2007    52.3  1.43e7     1217.
#>  4 Central Africa… Africa     2007    44.7  4.37e6      706.
#>  5 China            Asia       2007    73.0  1.32e9     4959.
#>  6 Costa Rica       Americas   2007    78.8  4.13e6     9645.
#>  7 Dominican Repu… Americas   2007    72.2  9.32e6     6025.
#>  8 Hong Kong, Chi… Asia       2007    82.2  6.98e6    39725.
#>  9 Jamaica          Americas   2007    72.6  2.78e6     7321.
#> 10 Mauritania       Africa     2007    64.2  3.27e6     1803.
#> 11 Nicaragua        Americas   2007    72.9  5.68e6     2749.
#> 12 South Africa     Africa     2007    49.3  4.40e7     9270.
#> 13 Swaziland        Africa     2007    39.6  1.13e6     4513.
#> 14 Taiwan           Asia       2007    78.4  2.32e7    28718.
```

Gapminder

- `"i.a$"` matches **the end** of “ina”, “ica”, “ita”, and more.

```
gapminder %>%
  filter(str_detect(country, "i.a$"))
```

```
#> # A tibble: 7 x 6
#>   country         continent  year lifeExp      pop gdpPercap
#>   <fct>           <fct>     <int>   <dbl>    <int>     <dbl>
#> 1 Argentina       Americas   2007    75.3   4.03e7    12779.
#> 2 Bosnia and Her… Europe     2007    74.9   4.55e6     7446.
#> 3 China           Asia       2007    73.0   1.32e9     4959.
#> 4 Costa Rica      Americas   2007    78.8   4.13e6     9645.
#> 5 Hong Kong, Chi… Asia       2007    82.2   6.98e6    39725.
#> 6 Jamaica         Americas   2007    72.6   2.78e6     7321.
#> 7 South Africa    Africa     2007    49.3   4.40e7     9270.
```


Gapminder

- `"[nls]ia$"` matches `ia` at the end of the country name, preceded by one of the characters in the class given inside `[]`.

```
gapminder %>%
  filter(str_detect(country, "[nls]ia$"))
```

```
#> # A tibble: 11 x 6
#>    country    continent  year lifeExp       pop gdpPercap
#>    <fct>      <fct>     <int>   <dbl>     <int>     <dbl>
#>  1 Albania    Europe     2007    76.4   3600523     5937.
#>  2 Australia  Oceania    2007    81.2  20434176    34435.
#>  3 Indonesia  Asia       2007    70.6 223547000     3541.
#>  4 Malaysia   Asia       2007    74.2  24821286    12452.
#>  5 Mauritania Africa     2007    64.2   3270065     1803.
#>  6 Mongolia   Asia       2007    66.8   2874127     3096.
#>  7 Romania    Europe     2007    72.5  22276056    10808.
#>  8 Slovenia   Europe     2007    77.9   2009245    25768.
#>  9 Somalia    Africa     2007    48.2   9118773      926.
#> 10 Tanzania   Africa     2007    52.5  38139640     1107.
#> 11 Tunisia    Africa     2007    73.9  10276158     7093.
```


Gapminder

- `"[^nls]ia$"` matches `ia` at the end of the country name, preceded by anything but one of the characters in the class given inside `[]`.

```
gapminder %>%
  filter(str_detect(country, "[^nls]ia$"))
```

```
#> # A tibble: 17 x 6
#>    country      continent  year lifeExp        pop gdpPercap
#>    <fct>        <fct>     <int>   <dbl>      <int>     <dbl>
#>  1 Algeria      Africa     2007    72.3   33333216     6223.
#>  2 Austria      Europe     2007    79.8    8199783    36126.
#>  3 Bolivia      Americas   2007    65.6    9119152     3822.
#>  4 Bulgaria     Europe     2007    73.0    7322858    10681.
#>  5 Cambodia     Asia       2007    59.7   14131858     1714.
#>  6 Colombia     Americas   2007    72.9   44227550     7007.
#>  7 Croatia      Europe     2007    75.7    4493312    14619.
#>  8 Ethiopia     Africa     2007    52.9   76511887      691.
#>  9 Gambia       Africa     2007    59.4    1688359      753.
#> 10 India        Asia       2007    64.7 1110396331     2452.
#> 11 Liberia      Africa     2007    45.7    3193942      415.
```


Gapminder

- `"[:punct:]"` matches country names that contain punctuation.

```
gapminder %>%
  filter(str_detect(country, "[:punct:]"))
```

```
#> # A tibble: 8 x 6
#>   country          continent  year lifeExp     pop gdpPercap
#>   <fct>            <fct>     <int>   <dbl>   <int>     <dbl>
#> 1 Congo, Dem. Rep. Africa     2007    46.5 6.46e7      278.
#> 2 Congo, Rep.      Africa     2007    55.3 3.80e6     3633.
#> 3 Cote d'Ivoire    Africa     2007    48.3 1.80e7     1545.
#> 4 Guinea-Bissau    Africa     2007    46.4 1.47e6      579.
#> 5 Hong Kong, China Asia       2007    82.2 6.98e6    39725.
#> 6 Korea, Dem. Rep. Asia       2007    67.3 2.33e7     1593.
#> 7 Korea, Rep.      Asia       2007    78.6 4.90e7    23348.
#> 8 Yemen, Rep.      Asia       2007    62.7 2.22e7     2281.
```

Text mining

🎼 Waiting for the Sun ☀️

```
lyrics <- c("This will be an uncertain time for us my love",
            "I can hear the echo of your voice in my head",
            "Singing my love",
            "I can see your face there in my hands my love",
            "I have been blessed by your grace and care my love",
            "Singing my love")
text_tbl <- tibble(line = seq_along(lyrics), text = lyrics)
text_tbl
```

```
#> # A tibble: 6 x 2
#>    line text
#>   <int> <chr>
#> 1     1 This will be an uncertain time for us my love
#> 2     2 I can hear the echo of your voice in my head
#> 3     3 Singing my love
#> 4     4 I can see your face there in my hands my love
#> 5     5 I have been blessed by your grace and care my love
#> 6     6 Singing my love
```


- tokenise

unigram

```
library(tidytext)
text_tbl %>%
  unnest_tokens(output = word, input = text)
```

```
#> # A tibble: 49 x 2
#>    line word
#>   <int> <chr>
#> 1     1 this
#> 2     1 will
#> 3     1 be
#> 4     1 an
#> 5     1 uncertain
#> 6     1 time
#> # … with 43 more rows
```


- tokenise

unigram

```
text_tbl %>%
  unnest_tokens(output = word, input = text) %>%
  count(word, sort = TRUE)
```

```
#> # A tibble: 32 x 2
#>   word      n
#>   <chr> <int>
#> 1 my        7
#> 2 love      5
#> 3 i         3
#> 4 your      3
#> 5 can       2
#> 6 in        2
#> # … with 26 more rows
```

how often we see each word in this corpus

letters

- tokenise

```
text_tbl %>%
  unnest_characters(output = word, input = text)
```

```
#> # A tibble: 171 x 2
#>    line word
#>   <int> <chr>
#> 1     1 t
#> 2     1 h
#> 3     1 i
#> 4     1 s
#> 5     1 w
#> 6     1 i
#> # … with 165 more rows
```


n-gram

- tokenise

```
text_tbl %>%
  unnest_ngrams(output = word, input = text, n = 2)
```

```
#> # A tibble: 43 x 2
#>    line word
#>   <int> <chr>
#> 1     1 this will
#> 2     1 will be
#> 3     1 be an
#> 4     1 an uncertain
#> 5     1 uncertain time
#> 6     1 time for
#> # … with 37 more rows
```


sentiment analysis

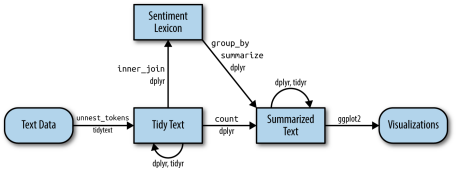


image credit: Text Mining with R

sentiment analysis

- import

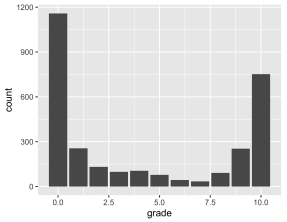
```
user_reviews <- read_tsv(
  "data/animal-crossing/user_reviews.tsv")
user_reviews
```

```
#> # A tibble: 2,999 x 4
#>   grade user_name   text                        date
#>   <dbl> <chr>       <chr>                       <date>
#> 1     4 mds27272    My gf started playing before… 2020-03-20
#> 2     5 lolo2178    While the game itself is gre… 2020-03-20
#> 3     0 Roachant    My wife and I were looking f… 2020-03-20
#> 4     0 Houndf      We need equal values and opp… 2020-03-20
#> 5     0 ProfessorF… BEWARE!  If you have multipl… 2020-03-20
#> 6     0 tb726       The limitation of one island… 2020-03-20
#> # … with 2,993 more rows
```


grade distribution

- import
- glimpse

```
user_reviews %>%
  ggplot(aes(grade)) +
  geom_bar()
```



positive vs negative reviews

- import
- glimpse

```
user_reviews %>%
  slice_max(grade,
    with_ties = FALSE) %>%
  pull(text)
```

```
#> [1] "Cant stop playing!"
```

```
user_reviews %>%
  slice_min(grade,
    with_ties = FALSE) %>%
  pull(text)
```

```
#> [1] "My wife and I were
looking forward to playing this
game when it released. I bought
it, I let her play first she
made an island and played for a
bit. Then I decided to play only
to discover that Nintendo only
allows one island per switch!
Not only that, the second player
cannot build anything on the
island and tool building is
considerably harder to do. So,
if you have more than one
personMy wife and I were looking
forward to playing this game
when it released. I bought it, I
```


clean a bit from web scraping ...

- import
- glimpse
- tokenise

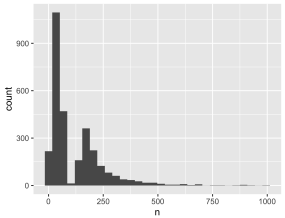
```
user_reviews_words <- user_reviews %>%
  mutate(text = str_remove(text, "Expand$")) %>%
  unnest_tokens(output = word, input = text)
user_reviews_words
```

```
#> # A tibble: 362,729 x 4
#>   grade user_name date       word
#>   <dbl> <chr>     <date>     <chr>
#> 1     4 mds27272  2020-03-20 my
#> 2     4 mds27272  2020-03-20 gf
#> 3     4 mds27272  2020-03-20 started
#> 4     4 mds27272  2020-03-20 playing
#> 5     4 mds27272  2020-03-20 before
#> 6     4 mds27272  2020-03-20 me
#> # … with 362,723 more rows
```

- import
- glimpse
- tokenise
- **vis**

distribution of words per review

```
user_reviews_words %>%
  count(user_name) %>%
  ggplot(aes(x = n)) +
  geom_histogram()
```



- import
- glimpse
- tokenise
- vis
- **stop words**

the most common words

```
user_reviews_words %>%
  count(word, sort = TRUE)
```

```
#> # A tibble: 13,454 x 2
#>   word      n
#>   <chr> <int>
#> 1 the   17739
#> 2 to    11857
#> 3 game   8769
#> 4 and    8740
#> 5 a      8330
#> 6 i      7211
#> # … with 13,448 more rows
```


- import
- glimpse
- tokenise
- vis
- **stop words**

lexicon

```
get_stopwords()
```

```
#> # A tibble: 175 x 2
#>   word   lexicon
#>   <chr>  <chr>
#> 1 i      snowball
#> 2 me     snowball
#> 3 my     snowball
#> 4 myself snowball
#> 5 we     snowball
#> 6 our    snowball
#> # … with 169 more rows
```

1. In computing, stop words are words which are filtered out before or after processing of natural language data (text).
2. They usually refer to the most common words in a language, but there is not a single list of stop words used by all natural language processing tools.

- import
- glimpse
- tokenise
- vis
- **stop words**

remove stop words

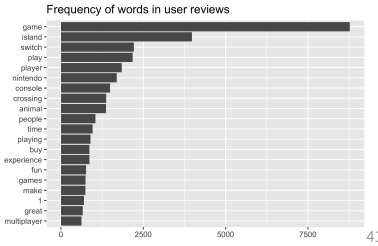
```
stopwords_smart <- get_stopwords(source = "smart")
user_reviews_smart <- user_reviews_words %>%
  anti_join(stopwords_smart)
user_reviews_smart
```

```
#> # A tibble: 145,444 x 4
#>   grade user_name date       word
#>   <dbl> <chr>     <date>     <chr>
#> 1     4 mds27272  2020-03-20 gf
#> 2     4 mds27272  2020-03-20 started
#> 3     4 mds27272  2020-03-20 playing
#> 4     4 mds27272  2020-03-20 option
#> 5     4 mds27272  2020-03-20 create
#> 6     4 mds27272  2020-03-20 island
#> # … with 145,438 more rows
```

- import
- glimpse
- tokenise
- vis
- stop words
- **count**

the most common words

Plot | Code



- import
- glimpse
- tokenise
- vis
- stop words
- count
- **sentiments**

sentiment lexicons

- AFINN lexicon measures sentiment with a numeric score b/t -5 & 5.
- Other lexicons categorise words in a binary fashion, either positive or negative.

```
get_sentiments("afinn")
```

```
#> # A tibble: 2,477 x 2
#>   word       value
#>   <chr>      <dbl>
#> 1 abandon       -2
#> 2 abandoned     -2
#> 3 abandons      -2
#> 4 abducted      -2
#> 5 abduction     -2
#> 6 abductions    -2
#> # … with 2,471 more rows
```

```
get_sentiments("loughran")
```

```
#> # A tibble: 4,150 x 2
#>   word         sentiment
#>   <chr>        <chr>
#> 1 abandon      negative
#> 2 abandoned    negative
#> 3 abandoning   negative
#> 4 abandonment  negative
#> 5 abandonments negative
#> 6 abandons     negative
#> # … with 4,144 more rows
```


- import
- glimpse
- tokenise
- vis
- stop words
- count
- **sentiments**

sentiment lexicons

```
sentiments_bing <- get_sentiments("bing")
sentiments_bing
```

```
#> # A tibble: 6,786 x 2
#>   word       sentiment
#>   <chr>      <chr>
#> 1 2-faces    negative
#> 2 abnormal   negative
#> 3 abolish    negative
#> 4 abominable negative
#> 5 abominably negative
#> 6 abominate  negative
#> # … with 6,780 more rows
```


- import
- glimpse
- tokenise
- vis
- stop words
- count
- **sentiments**

join sentiments

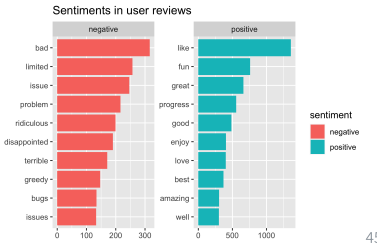
```
user_reviews_sentiments <- user_reviews_words %>%
  inner_join(sentiments_bing) %>%
  count(sentiment, word, sort = TRUE)
user_reviews_sentiments
```

```
#> # A tibble: 1,622 x 3
#>   sentiment word        n
#>   <chr>     <chr>   <int>
#> 1 positive  like     1357
#> 2 positive  fun       760
#> 3 positive  great     661
#> 4 positive  progress  556
#> 5 positive  good      486
#> 6 positive  enjoy     405
#> # … with 1,616 more rows
```

visualise sentiments

- import
- glimpse
- tokenise
- vis
- stop words
- count
- sentiments
- **vis**

› Plot › Code



Would Animal Crossing be considered to be a delightful game?

```
user_reviews_sentiments %>%
  group_by(sentiment) %>%
  summarise(n = sum(n)) %>%
  mutate(p = n / sum(n))
```

```
#> # A tibble: 2 x 3
#>   sentiment     n     p
#>   <chr>     <int> <dbl>
#> 1 negative  11097 0.464
#> 2 positive  12825 0.536
```

- import
- glimpse
- tokenise
- vis
- stop words
- count
- sentiments
- **vis**

Reading

› Strings
› {stringr} cheatsheet

› Sentiment analysis with tidy data