

Lecture 20: Intro to regular expressions

Last time: scraping and wrangling Taskmaster data

What we ultimately want:

1	Task	Description	episode	episode_name	air_date	contestant	score
2	1 1	Prize: Best th...	1	"It's not y...	18 Marc...	Charlotte...	1
3	2 1	Prize: Best th...	1	"It's not y...	18 Marc...	Jamali Ma...	2
4	3 1	Prize: Best th...	1	"It's not y...	18 Marc...	Lee Mack	4
5	4 1	Prize: Best th...	1	"It's not y...	18 Marc...	Mike Wozn...	5
6	5 1	Prize: Best th...	1	"It's not y...	18 Marc...	Sarah Ken...	3
7	6 2	Do the most im...	1	"It's not y...	18 Marc...	Charlotte...	2
8	7 2	Do the most im...	1	"It's not y...	18 Marc...	Jamali Ma...	3
9	8 2	Do the most im...	1	"It's not y...	18 Marc...	Lee Mack	3
10	9 2	Do the most im...	1	"It's not y...	18 Marc...	Mike Wozn...	5
11	10 2	Do the most im...	1	"It's not y...	18 Marc...	Sarah Ken...	4

colnames: Task, Description, episode, episode_name,
air_date, contestant, score, series

Last time: scraping and wrangling Taskmaster data

```
1 results <- read_html("https://taskmaster.fandom.com/wiki/Series_11")
2   html_element(".tmttable") |>
3     html_table() |>
4       mutate(episode = ifelse(startsWith(Task, "Episode"), Task, NA)) |>
5         fill(episode, .direction = "down") |>
6         filter(!startsWith(Task, "Episode"),
7             !(Task %in% c("Total", "Grand Total")))) |>
8           pivot_longer(cols = -c(Task, Description, episode),
9                           names_to = "contestant",
10                          values_to = "score") |>
11    mutate(series = 11)
```

What we have so far

1	Task	Description	episode	contestant	score	series
2	1	Prize: Best thing...	Episode 1...	Charlotte...	1	11
3	1	Prize: Best thing...	Episode 1...	Jamali Ma...	2	11
4	1	Prize: Best thing...	Episode 1...	Lee Mack	4	11
5	1	Prize: Best thing...	Episode 1...	Mike Wozn...	5	11
6	1	Prize: Best thing...	Episode 1...	Sarah Ken...	3	11
7	2	Do the most...	Episode 1...	Charlotte...	2	11
8	2	Do the most...	Episode 1...	Jamali Ma...	3 [1]	11
9	2	Do the most...	Episode 1...	Lee Mack	3	11
10	2	Do the most...	Episode 1...	Mike Wozn...	5	11
11	2	Do the most...	Episode 1...	Sarah Ken...	4	11

Currently, the episode column contains entries like

```
1 "Episode 1: It's not your fault. (18 March 2021)"
```

Next steps

1. Separate episode info into episode number, episode name, and air date columns
2. Clean up the score column
3. Combine data from multiple series

Goal for today: start learning some tools for 1. and 2.

Cleaning the score column

```
1 table(results$score)
```

-	✓	✗	0	1	2	3	3[1]	3[2]	4	4[2]	5	DQ
7	1	1	11	37	42	48	1	3	50	1	55	13

How do we want to clean these scores? How should the scores be stored?

special
symbols

- remove footnotes
- DQ → 0? (possible confusion w/ 0 pts?)
(separate column for DQ?)
- separately keep track of tie breakers?

Extracting numeric information

Suppose we have the following string:

```
1 "3[1]"
```

And we want to extract just the number “3”:

```
1 str_extract("3[1]", "3")
```

[1] "3"

↑

extracts
part of a
string
which
matches
a
pattern

~~~~~  
A ↗

pattern to match

String to check

# Extracting numeric information

Suppose we have the following string:

```
1 "3[1]"
```

What if we don't know which number to extract?

```
1 str_extract("3[1]", "\d")  
[1] "3"  
  
1 str_extract("4[1]", "\d")  
[1] "4"  
  
1 str_extract("DQ", "\d")  
[1] NA
```

returns  
the  
first  
match

\d means "any digit" (0-9)

In R: pass regular expressions in a string  
so add extra \ to the beginning  
\d means "any digit"

# Regular expressions

A *regular expression* is a pattern used to find matches in text.

The simplest regular expressions match a specific character or sequence of characters:

```
1 str_extract("My cat is 3 years old", "cat")  
[1] "cat"  
  
1 str_extract("My cat is 3 years old", "3")  
[1] "3"
```

# Matching multiple options

We can also provide multiple options for the match

```
1 str_extract("My cat is 3 years old", "cat|dog")
```

```
[1] "cat"
```

*match "cat" or "dog"*

```
1 str_extract("My dog is 10 years old", "cat|dog")
```

*"dog"*

# Matching multiple options

We can also provide multiple options for the match

```
1 str_extract("My cat is 3 years old", "cat|dog")
```

```
[1] "cat"
```

```
1 str_extract("My dog is 10 years old", "cat|dog")
```

```
[1] "dog"
```

```
1 str_extract("My dog is 10 years old, my cat is 3 years old",
2                 "cat|dog")
```

"dog"

# Matching multiple options

We can also provide multiple options for the match

```
1 str_extract("My cat is 3 years old", "cat|dog")
```

```
[1] "cat"
```

```
1 str_extract("My dog is 10 years old", "cat|dog")
```

```
[1] "dog"
```

```
1 str_extract("My dog is 10 years old, my cat is 3 years old",
2                      "cat|dog")
```

```
[1] "dog"
```

```
1 str_extract_all("My dog is 10 years old, my cat is 3 years old",
2                      "cat|dog")
```

```
[[1]]
```

```
[1] "dog" "cat"
```

first  
match

return all matches to either "cat" or "dog"

# Matching groups of characters

What if I want to extract a *number*?

```
1 str_extract("My cat is 3 years old", "\\d")  
[1] "3"
```

What do you think will happen when I run the following code?

```
1 str_extract("My dog is 10 years old", "\\d")
```

# Matching groups of characters

What if I want to extract a *number*?

```
1 str_extract("My cat is 3 years old", "\\d")
[1] "3"
```

What do you think will happen when I run the following code?

```
1 str_extract("My dog is 10 years old", "\\d")
[1] "1"
```

↑  
only returns the first match

# Matching groups of characters

The + symbol in a regular expression means “repeated one or more times”

```
1 str_extract("My dog is 10 years old", "\\d+")
[1] "10"
```

↑  
one or more instances

$\backslash d^+$  means a sequence  
of one or more  
digits

# Extracting from multiple strings

```
1 strings <- c("My cat is 3 years old", "My dog is 10 years old")
2 str_extract(strings, "\\d+")
[1] "3"   "10"   ↑
```

if we input a vector, we apply the pattern to  
each entry in the vector

# Extracting episode information

Currently, the episode column contains entries like:

```
1 "Episode 2: The pie whisperer. (4 August 2015)"
```

How would I extract just the episode number?

ll d +

# Extracting episode information

Currently, the episode column contains entries like:

```
1 "Episode 2: The pie whisperer. (4 August 2015)"
```

How would I extract just the episode number?

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)", "\\d+")
[1] "2"
```

# Extracting episode information

Currently, the episode column contains entries like:

```
1 "Episode 2: The pie whisperer. (4 August 2015)"
```

How would I } extract the episode name?  
                    ↑  
                    end before .

Starts after the colon :

# Extracting episode information

```
1 "Episode 2: The pie whisperer. (4 August 2015)"
```

Pattern to match: *anything* that starts with a :, ends with a

■

**Note:** The . character in a regex means “any character”

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)", ".")
```

```
[1] "E"
```

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)", ".+")
```

```
[1] "Episode 2: The pie whisperer. (4 August 2015)"
```

# Extracting episode information

**Note:** The `.` character in a regex means “any character”

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)", ".")  
[1] "E"
```

We use an *escape character* when we actually want to choose a period:

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)", "\\.")  
[1] "."
```

\.                    "I actually went the period"  
\\.                    add additional backslash in R

# Extracting episode information

Getting everything between the : and the .

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)",  
2      ":.+\\.")
```

[1] ": The pie whisperer."

Start with a colon:  
match anything else  
end with the period

probably want to remove : and . from name

# Extracting episode information

# Getting everything between the : and the .

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)",  
2             "(?<=: ).+(?=\\.)")
```

[1] "The pie whisperer") ↑      ↗      ↑  
                                something      comes before  
                                came s  
                                after

# Lookbehinds

( ?<=) is a *positive lookbehind*. It is used to identify expressions which are *preceded* by a particular expression.

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)",  
2           "(?<=: ).+")
```

```
[1] "The pie whisperer. (4 August 2015)"
```

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)",  
2           "(?<=\\" ).+")
```

```
[1] "(4 August 2015)"
```

(? . L =   )  
      ↑  
fill in regular expression

# Lookaheads

(?=) is a *positive lookahead*. It is used to identify expressions which are *followed* by a particular expression.

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)",  
2           ".+(?=\\.)")
```

```
[1] "Episode 2: The pie whisperer"
```

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)",  
2           ".+(?=:)")
```

```
[1] "Episode 2"
```

(?<sub>1</sub>=  )  
      ↑  
      fill in  
      regular  
      expression

# Extracting air date

I want to extract just the air date. What pattern do I want to match?

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)", )
```

4 August 2015

after (   
 before )

(?.L=\\"( ) .+ (?.=\\") )  
after (      Something      before )

# Extracting air date

```
1 str_extract("Episode 2: The pie whisperer. (4 August 2015)",  
2           "(?=<\\().+(?=\\))")
```

4 August 2015

# Wrangling the episode info

Currently:

```
# A tibble: 270 × 1  
  episode  
  <chr>  
1 Episode 1: It's not your fault. (18 March 2021)  
2 Episode 1: It's not your fault. (18 March 2021)  
3 Episode 1: It's not your fault. (18 March 2021)  
4 Episode 1: It's not your fault. (18 March 2021)  
5 Episode 1: It's not your fault. (18 March 2021)  
6 Episode 1: It's not your fault. (18 March 2021)  
7 Episode 1: It's not your fault. (18 March 2021)  
8 Episode 1: It's not your fault. (18 March 2021)  
9 Episode 1: It's not your fault. (18 March 2021)  
10 Episode 1: It's not your fault. (18 March 2021)  
# i 260 more rows
```

Episode 1: It's not your fault. (18 March 2021)  
↓                  ↓                  ↓  
episode\_num      episode-name    episode\_date

option \:

```
mutate(  
  episode_num = str_extract(...),  
  episode_name = str_extract(...),  
  episode_date = str_extract(...))
```

# Wrangling the episode info

One option:

```
1 results |>
2   mutate(episode_name = str_extract(episode,
3                                     "( ?<=: ).+(?=\\.)"),
4         air_date = str_extract(episode, "( ?<=\\().+(?=\\))"),
5         episode = str_extract(episode, "\\d+"))
```

# A tibble: 270 × 3

|    | episode | episode_name        | air_date      |
|----|---------|---------------------|---------------|
|    | <chr>   | <chr>               | <chr>         |
| 1  | 1       | It's not your fault | 18 March 2021 |
| 2  | 1       | It's not your fault | 18 March 2021 |
| 3  | 1       | It's not your fault | 18 March 2021 |
| 4  | 1       | It's not your fault | 18 March 2021 |
| 5  | 1       | It's not your fault | 18 March 2021 |
| 6  | 1       | It's not your fault | 18 March 2021 |
| 7  | 1       | It's not your fault | 18 March 2021 |
| 8  | 1       | It's not your fault | 18 March 2021 |
| 9  | 1       | It's not your fault | 18 March 2021 |
| 10 | 1       | It's not your fault | 18 March 2021 |

# i 260 more rows

option 2 : separate wider regex

(describing all pieces of the string, and specifying (i.e., naming)  
the pieces we want to keep)

Episode 1: It's not your fault. (18 March 2021)  
↓                          ↓                          ↓  
episode-num      episode-name      episode-date

Episode 1: It's not your fault. (18 March 2021)  
↓                  ↓                  ↓                  ↓  
".+"      ":"      ".+"      "11. ("      ".+"      "11)"  
  " "      " "      " "      " "      " "  
episode      episode-name      episode-date

# Wrangling the episode info

Another option:

```
1 results |>
2   separate_wider_regex(episode,
3     patterns = c(".+", "episode",
4                   ":", "episode_name",
5                   "\\\\.\\(\"",
6                   "air_date = ".+",
7                   "\\\")"))
```

Episode 1 : It's not your fault. (18 March 2021)  
+ not in episode name ( air-date )

```
# A tibble: 270 × 3
  episode episode_name      air_date
  <chr>   <chr>          <chr>
1 1       It's not your fault 18 March 2021
2 1       It's not your fault 18 March 2021
3 1       It's not your fault 18 March 2021
4 1       It's not your fault 18 March 2021
5 1       It's not your fault 18 March 2021
6 1       It's not your fault 18 March 2021
7 1       It's not your fault 18 March 2021
8 1       It's not your fault 18 March 2021
9 1       It's not your fault 18 March 2021
```

10 1

It's not your fault 10 March 2021

