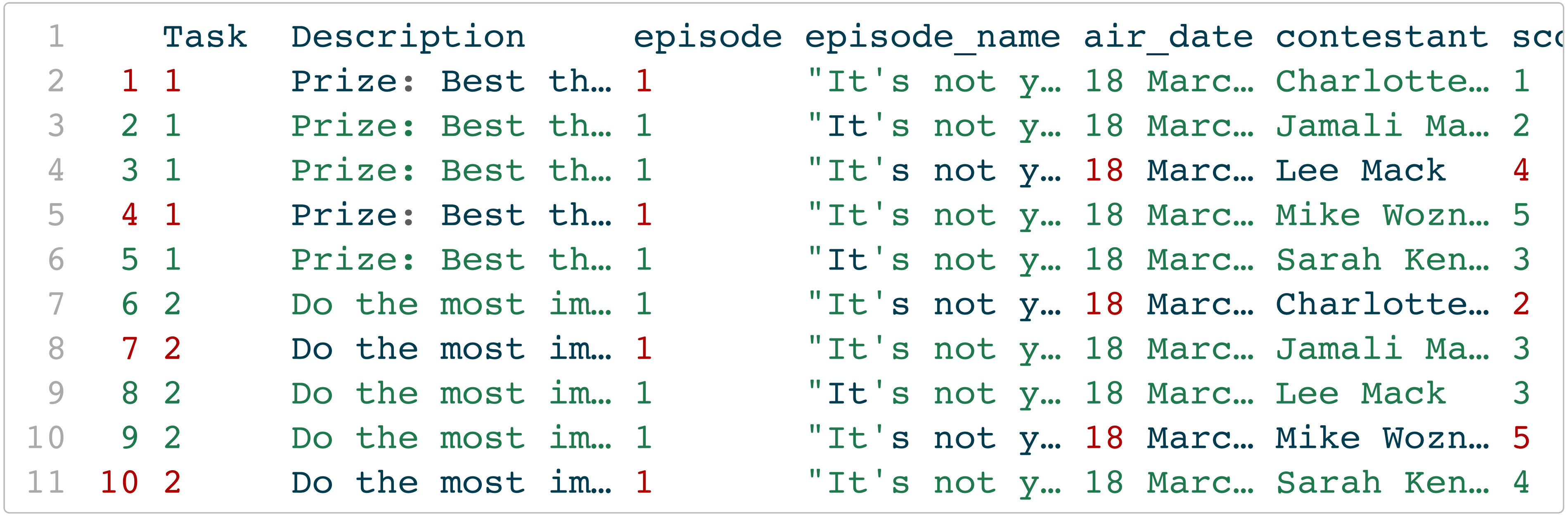
# Lecture 21: Intro to regular expressions

# Last time: scraping and wrangling Taskmaster data

What we ultimately want:



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

```r
results <- read_html("https://taskmaster.fandom.com/wiki/Series_11")
  html_element(".tmtable") |>
  html_table() |>
  mutate(episode = ifelse(startsWith(Task, "Episode"), Task, NA)) |>
  fill(episode, .direction = "down") |>
  filter(!startsWith(Task, "Episode"),
         !(Task %in% c("Total", "Grand Total"))) |>
  pivot_longer(cols = -c(Task, Description, episode),
               names_to = "contestant",
               values_to = "score") |>
  mutate(series = 11)
```

# What we have so far

```
   Task  Description       episode    contestant score series
  1      Prize: Best thing… Episode 1… Charlotte… 1         11
  1      Prize: Best thing… Episode 1… Jamali Ma… 2         11
  1      Prize: Best thing… Episode 1… Lee Mack   4         11
  1      Prize: Best thing… Episode 1… Mike Wozn… 5         11
  1      Prize: Best thing… Episode 1… Sarah Ken… 3         11
  2      Do the most…       Episode 1… Charlotte… 2         11
  2      Do the most…       Episode 1… Jamali Ma… 3[1]      11
  2      Do the most…       Episode 1… Lee Mack   3         11
  2      Do the most…       Episode 1… Mike Wozn… 5         11
  2      Do the most…       Episode 1… Sarah Ken… 4         11
```

Currently, the episode column contains entries like

```
"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

```r
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?

handle special symbols

- remove footnotes ([1], e.g.)
- DQ → 0 (separate column for DQ?)
- separately keep track of tie breakers?

# Extracting numeric information

Suppose we have the following string:

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

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

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

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

pattern to match

extracts part of a String which matches a pattern

String to check (look for the pattern)

# Extracting numeric information

Suppose we have the following string:

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

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

```r
str_extract("3[1]", "\\d")
```

regular expression

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

```r
str_extract("4[1]", "\\d")
```

```
[1] "4"
```

```r
str_extract("DQ", "\\d")
```

```
[1] NA
```

Regular expression: \d means "any digit" (0-9)

In R: \\d in a string means \d

Multiple digits: \\d+ "one or more digits in a row"

# 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:

```r
str_extract("My cat is 3 years old", "cat")
```

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

```r
str_extract("My cat is 3 years old", "3")
```

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

# Matching multiple options

We can also provide multiple options for the match

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

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

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

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

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

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

first match

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

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

all the matches

# Matching groups of characters

What if I want to extract a *number*?

```r
str_extract("My cat is 3 years old", "\\d")
```

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

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

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

# Matching groups of characters

What if I want to extract a *number*?

```r
str_extract("My cat is 3 years old", "\\d")
```

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

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

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

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

# Matching groups of characters

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

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

```
[1] "10"
```

one or more

# Extracting from multiple strings

```r
strings <- c("My cat is 3 years old", "My dog is 10 years old")
str_extract(strings, "\\d+")
```

```
[1] "3"  "10"
```

# Extracting episode information

Currently, the `episode` column contains entries like:

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

How would I extract just the episode number?

nd t

# Extracting episode information

Currently, the `episode` column contains entries like:

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

How would I extract just the episode number?

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

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

first match

sequence of digits

# Extracting episode information

Currently, the `episode` column contains entries like:

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

How would I extract the episode name?

ends before .

Starts after the :

# Extracting episode information

```
"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”

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

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

```
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”

```r
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:

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

```
[1] "."
```

# Extracting episode information

Getting everything between the `:` and the `.`

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

```
[1] ": The pie whisperer."
```

ends with a period

contains any character at least once

pattern starts with a :

# Extracting episode information

Getting everything between the `:` and the `.`

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

```
[1] "The pie whisperer"
```

came after :

something

came before .

# Lookbehinds

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

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

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

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

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

fill in

regular expression

# Lookaheads

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

fill in regular expression

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

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

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

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

. any character

+ one or more time

=> .+ a sequence of at least one character

# Extracting air date

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

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

between parentheses

after (

before )

(?<=\\() .+ (?=\\))

after ( something before )

# Extracting air date

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

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
```

# Wrangling the episode info

One option:

```r
results |>
  mutate(episode_name = str_extract(episode,
                                    "(?<=: ).+(?=\\.)"),
         air_date = str_extract(episode, "(?<=\\().+(?=\\))"),
         episode = str_extract(episode, "\\d+"))
```

last b/c overwriting episode

```
# 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
```

# Wrangling the episode info

## Another option:

separate episode column

```r
results |>
  separate_wider_regex(episode,
                       patterns = c(".+ ",
                                    episode = "\\d+",
                                    ": ",
                                    episode_name = ".+",
                                    "\\. \\(",
                                    air_date = ".+",
                                    "\\)"))
```

something

episode #

into episode, episode-name, air-date

:

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 18 March 2021