Webscraping Text

Emily Malcolm-White

```r
#LOAD PACKAGES
library(tidyverse)
library(rvest)
```

Webscraping Text

Let's look at the Science Books on Good Reads

```r
URL <- read_html("https://www.goodreads.com/shelf/show/science")
```

Notice that the data for all these books isn't housed inside a `<table>` element!

Titles

For example, check out the lines of html code for the book "Sapians":

```
<div class="elementList" style="padding-top: 10px;">
<div class="left" style="width: 75%;">
<a title="Sapiens: A Brief History of Humankind" class="leftAlignedImage" href="/book/show/23
<a class="bookTitle" href="/book/show/23692271-sapiens">Sapiens: A Brief History of Humankin
<br />
<span class='by'>by</span>
<span itemprop='author' itemscope='' itemtype='http://schema.org/Person'>
<div class='authorName__container'>
<a class="authorName" itemprop="url" href="https://www.goodreads.com/author/show/395812.Yuval
</div>
</span>
(shelved 6592 times as <em>science</em>)
<br />
<span class="greyText smallText">
avg rating 4.35 —
1,168,542 ratings  —
published 2011
</span>
</div>
```

Titles

In this case, we want to look for the class `bookTitle` using `html_elements(".bookTitle")`

> Tip
>
> In this case, we want ALL titles so we used `html_elements()`. If we had only wanted the first title we would have used `html_element()`

Authors

In this case, we want to look for the class `authorName` using `html_elements(".authorName")`

> Tip
>
> In this case, we want ALL titles so we used `html_elements()`. If we had only wanted the first title we would have used `html_element()`

Other Info

```
<span class="greyText smallText">
avg rating 4.35 -
1,168,542 ratings  -
published 2011
</span>
```

In this case, we need to reference the class `greyText`

Putting it all together

Oops! They aren't all the same length, so we can't put them in the same dataset. There should be 50 books! We need to look at this by hand…

The 13th element of `authors` just says "(Contributers)". Let's remove that.

Our `info` vector has some excessive info (ie. `info[58] = (GoodReads Author)`)

Sometimes webscraping requires some manual massaging. We could try using the full `class = "greyText smallText` but using multiple classes `html_elements(".greyText.smallText")`

All Together Now

Now they are all the same length:

Seperating Data in Columns

We will need some help of `regex` (short for regular expression) – a powerful way to search, match, and extract patterns in text.

```
# books <- books %>%
#   mutate(
#     # Remove newlines and trim spaces
#     Info = str_squish(Info),
#
#     # Extract each piece using regex
#     avg_rating = str_extract(Info, "(?<=avg rating )\\d+\\.\\d+"),
#     num_ratings = str_extract(Info, "\\d{1,3}(,\\d{3})*(?= ratings)"),
#     yr_published = str_extract(Info, "(?<=published )\\d{4}")
#   ) %>%
#   select(-Info)
```

regex expressions can be very challenging depending on the complexity of the text you are trying to format. You are encouraged to use Google and/or generative AI to help you come up with the appropriate expressions. Below is a quick guide to some of the basics:

```
| Pattern | Meaning                                     | Example
|---------|---------------------------------------------|---------------------------------
| `.`     | Any character except newline                | `a.b` matches `acb`, `a2b`
| `\d`    | Any digit (0-9)                             | `\d+` matches `2023`, `55`
| `\w`    | Any word character (letter, number, _)      | `\w+` matches `hello`, `abc123`
| `\s`    | Whitespace (space, tab, newline)            | `\s+` matches spaces/tabs
| `+`     | One or more of the preceding                | `a+` matches `a`, `aa`, `aaa`
| `*`     | Zero or more of the preceding               | `a*` matches ``, `a`, `aa`
| `?`     | Optional (zero or one)                      | `colou?r` matches `color`, `colo
| `^`     | Start of line/string                        | `^The` matches lines starting wi
| `$`     | End of line/string                          | `end$` matches lines ending with
| `[]`    | Match one character inside brackets         | `[aeiou]` matches any vowel
| `[^]`   | Match anything **except** what's in brackets | `[^0-9]` matches non-digits
| `()`    | Grouping for extracting or repeating        | `(\d{4})` extracts 4-digit numbe
| `|`     | OR                                           | `cat|dog` matches `cat` or `dog
| `{n}`   | Exactly n repetitions                       | `\d{4}` matches exactly four dig
```

Formatting Columns

```
# books <- books %>%
#   mutate(
#     num_ratings = str_remove_all(num_ratings, ",") %>%  as.numeric(),
#     avg_rating = avg_rating %>%  as.numeric(),
#     num_ratings = num_ratings %>%  as.numeric(),
#     yr_published = yr_published %>%  as.numeric()
#   )
```