

Reap what you saw: web surfing to analysis

Kristina Martin

February 18, 2021

Situation: run across data embedded in a webpage

- Tables in text (Wikipedia, news article)
- Daily reports (weather, stocks)
- Scores (sports, students)

Example - CO2 Emissions Rankings

**The 20 countries that emitted
the most carbon dioxide in 2018**



Rank	Country	CO ₂ emissions (total)
1	China	10.06GT
2	United States	5.41GT
3	India	2.65GT
4	Russian Federation	1.71GT
5	Japan	1.16GT

Goal: import data for analysis



quickmeme.com

Options

- Hand copy to spreadsheet
- Import the flat HTML

```
url_emissions <- "https://www.ucsusa.org/resources/each-country-s-carbon-emissions"
flat_emissions <- readLines(url_emissions)
head(flat_emissions)

## [1] "<!DOCTYPE html>"
## [2] "<html lang=\"en\" dir=\"ltr\" prefix=\"og: https://ogp.me/ns#\">"
## [3] "  <head>"
## [4] "    <title>Each Country's Share of CO2 Emissions</title>"
## [5] "    <meta charset=\"utf-8\" /><script type=\"text/javascript\" src=\"https://www.ucsusa.org/resources/each-country-s-carbon-emissions.js\"></script>"
## [6] "<meta name=\"twitter:card\" content=\"summary_large_card\" />"
```

- A better way????

rvest helps you scrape (or harvest) data from web pages





Hadley Wickham



@hadleywickham



I'm starting to prepare for the release of rvest 1.0.0. It includes big improvements to text and table extraction, and a bunch of minor API improvements. If you're an rvest user, please try out the dev version and let me know how it goes: github.com/tidyverse/rvest... #rstats



tidyverse/rvest

Simple web scraping for R. Contribute to tidyverse/rvest development by creating an account on GitHub.

[github.com](https://github.com/tidyverse/rvest)

528 12:13 PM - Jan 7, 2021



81 people are talking about this



Scraping emissions rankings with rvest

```
# devtools::install_github("tidyverse/rvest")
library(rvest)

##
## Attaching package: 'rvest'
## The following object is masked from 'package:readr':
##       guess_encoding

scrape_emissions <- rvest::read_html(url_emissions)
tables <- scrape_emissions %>%
  html_elements("table") %>%
  html_table()
```

Scraping emissions with rvest

```
tables[1]
```

```
## [[1]]  
## # A tibble: 21 x 3  
##   Rank Country `CO2 emissions (total)`  
##   <int> <chr> <chr>  
## 1     1 China   10.06GT  
## 2     2 United States 5.41GT  
## 3     3 India   2.65GT  
## 4     4 Russian Federation 1.71GT  
## 5     5 Japan   1.16GT  
## 6     6 Germany 0.75GT  
## 7     7 Islamic Republic of Iran 0.72GT  
## 8     8 South Korea 0.65GT  
## 9     9 Saudi Arabia 0.62GT
```

Key functions

- `read_html()`
 - Returns an `xml_document` object.
- `html_elements()`
 - Extract specific components of the HTML.
 - Examples:
 - `html_elements("table")`
 - `html_elements("title")`
 - `html_elements("li")`
- `html_table()`
 - Converts tabular HTML data to `data.frame` (`rvest_0.99.0.9000`) or `tibble` (`rvest_1.0.0`)

Finding the right CSS



2020 IN REVIEW

THE BEST COOKBOOKS OF 2020

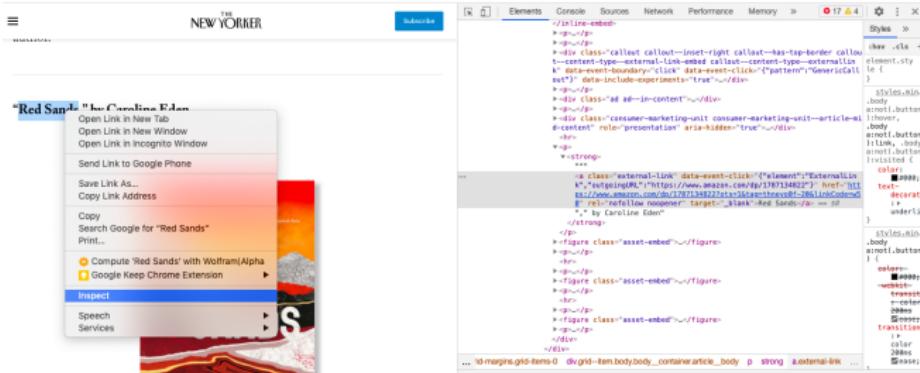
Despite everything, it's been a hell of a year for cookbooks. Here are ten of the best.

By Helen Rosner

December 15, 2020

Finding the right CSS

- In the browser, “Developer tools” or right-click “Inspect”



- Selector Gadget
 - “A JavaScript bookmarklet that allows you to interactively figure out what css selector you need to extract desired components from a page.”
 - YMMV

Finding the right CSS

```
url_cookbooks <- "https://www.newyorker.com/culture/2020-in-  
scrape_cb <- read_html(url_cookbooks)  
  
titles <- scrape_cb %>%  
  html_elements("strong") %>%  
  html_text2() %>% .[-1]  
  
titles  
  
## [1] ""Red Sands," by Caroline Eden"  
## [2] ""New World Sourdough," by Bryan Ford"  
## [3] ""In Bibi's Kitchen," by Hawa Hassan"  
## [4] ""50 Ways to Cook a Carrot," by Peter Hertzmann"  
## [5] ""Blood," by Jennifer McLagan"  
## [6] ""The Flavor Equation," by Nik Sharma"
```

Finding the right CSS

The screenshot shows a browser's developer tools element inspector. A specific element, a `` tag, is selected. The status bar at the bottom of the inspector window displays the CSS selector: `strong == $0`. The element itself contains the following HTML code:

```
<a class="external-link" data-event-click="{"element":"ExternalLink", "outgoingURL":"https://www.amazon.com/dp/1787134822"}> https://www.amazon.com/dp/1787134822?ots=1&tag=thneyo0f-20&linkCode=w50</a>  
" , " by Caroline Eden"  
</strong>
```

Finding the right CSS

```
df.cookbooks <- data.frame(  
  titles = scrape_cb %>%  
    html_elements("strong") %>%  
    html_text2() %>% .[ -1 ],  
  links = scrape_cb %>%  
    html_elements("strong") %>%  
    html_elements(".external-link") %>%  
    html_attr("href")) %>%  
  separate(titles,  
    into = c("title", "author"),  
    sep = " by ") %>%  
  mutate(title = str_replace(title, ',', '' , ' '))
```

```
head(df.cookbooks, 4)
```

```
##                                     title          author
## 1           "Red Sands"      Caroline Eden
## 2 "New World Sourdough"      Bryan Ford
## 3       "In Bibi's Kitchen"      Hawa Hassan
## 4 "50 Ways to Cook a Carrot" Peter Hertzmann
##                                     links
## 1 https://www.amazon.com/dp/1787134822
## 2 https://www.amazon.com/dp/B088Q27PP8
## 3 https://www.amazon.com/dp/1984856731
## 4 https://www.amazon.com/dp/1909248630
```



Kristina Martin

Reap what you saw: web surfing to analysis

Science and Art

ESPN NFL NBA Soccer NCAAM MMA Tennis ...

Soccer Home Scores Schedule Transfers Teams More

		Home	Scores	Schedule	Transfers	Teams	More				
		GP	W	D	L	E	A	GD	P	PPG	
11	 Chicago Fire FC	23	5	8	10	33	39	-6	23	1	
12	 Atlanta United FC	23	6	4	13	23	30	-7	22	0.96	
13	 DC United	23	5	6	12	25	41	-16	21	0.91	
14	 FC Cincinnati	23	4	4	15	12	36	-24	16	0.7	
WESTERN CONFERENCE		GP	W	D	L	E	A	GD	P	PPG	
1	 Sporting Kansas City	21	12	3	6	38	25	+13	39	1.86	
2	 Seattle Sounders FC	22	11	6	5	44	23	+21	39	1.77	
3	 Portland Timbers	23	11	6	6	46	35	+11	39	1.7	
4	 Minnesota United FC	21	9	7	5	36	26	+10	34	1.62	
5	 Colorado Rapids	18	8	4	6	32	28	+4	28	1.56	
6	 FC Dallas	22	9	7	6	28	24	+4	34	1.55	
7	 LAFC	22	9	5	8	47	39	+8	32	1.45	
8	 San Jose Earthquakes	23	8	6	9	35	51	-16	30	1.3	
9	 Vancouver Whitecaps	23	9	0	14	27	44	-17	27	1.17	
10	 LA Galaxy	22	6	4	12	27	46	-19	22	1	
11	 Real Salt Lake	22	5	7	10	25	35	-10	22	1	
12	 Houston Dynamo FC	23	4	9	10	30	40	-10	21	0.91	

Positions 1, 2, 3, 4, 5, 6: Qualifies for MLS Cup playoffs in both Conferences
Positions 7, 8: Qualifies for MLS Cup playoffs (Western); qualifies for play-in round (Eastern)
Positions 9, 10: Qualifies for play-in round (Eastern only)

GLOSSARY

GP: Games Played L: Losses GD: Goal Difference
W: Wins F: Goals For P: Points

Kristina Martin

Reap what you saw: web surfing to analysis

Science and Art

The screenshot shows a web browser displaying the ESPN Major League Soccer Table 2020 standings. The page title is "Major League Soccer Table 2020". The table lists 12 Eastern Conference teams with their records, goals scored, and points per game. The browser's developer tools are open, specifically the Elements tab, which highlights the HTML structure of the table. The table has a class of "standings_table" and is a "ResponsiveTable". The table header includes columns for Rank, Team, W, L, D, E, A, GS, P, and PPG. The table body contains 12 rows, each representing a team with its corresponding statistics.

	Team	W	L	D	E	A	GS	P	PPG
1	PHI	23	14	5	4	44	20	+24	2.04
2	TOR	23	13	5	5	33	26	+7	2.44
3	CLB	23	12	5	6	36	21	+15	2.11
4	ORL	23	11	8	4	40	25	+15	1.78
5	NYC	23	12	3	8	37	25	+12	3.9
6	NY	23	9	5	9	29	31	-2	3.39
7	NSH	23	8	8	7	24	22	+2	1.39
8	NE	23	8	8	7	26	25	+1	3.39
9	MTL	23	8	2	13	33	43	-10	2.6
10	MIA	23	7	3	13	25	35	-10	2.4
11	CHI	23	6	8	10	33	39	-6	1.04
12	ATL	23	6	4	13	29	30	-7	2.2

Kristina Martin

Reap what you saw: web surfing to analysis

Science and Art

```
url_soccer <- "https://www.espn.com/soccer/standings/_/league_id/1000"
scrape_soccer <- read_html(url_soccer)

scrape_soccer %>%
  html_element(".standings__table") %>%
  html_table()

## # A tibble: 56 x 9
##   X1                  X2      X3      X4      X5      X6
##   <chr>                <chr>  <chr>  <chr>  <chr>  <chr>
## 1 Eastern Conference    <NA>   <NA>   <NA>   <NA>   <NA>
## 2 Philadelphia Union    <NA>   <NA>   <NA>   <NA>   <NA>
## 3 Toronto FC            <NA>   <NA>   <NA>   <NA>   <NA>
## 4 Columbus Crew SC      <NA>   <NA>   <NA>   <NA>   <NA>
## 5 Orlando City SC       <NA>   <NA>   <NA>   <NA>   <NA>
```

Science and Art

```
scrape_soccer %>%
  html_element(".standings__table") %>%
  html_table() %>%
  .[29:56, ]
```

```
## # A tibble: 28 x 9
##   X1     X2     X3     X4     X5     X6     X7     X8     X9
##   <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 GP      W      D      L      F      A      GD     P      PPG
## 2 23     14     5      4     44     20    +24    47    2.04
## 3 23     13     5      5     33     26     +7    44    1.91
## 4 23     12     5      6     36     21    +15    41    1.78
## 5 23     11     8      4     40     25    +15    41    1.78
## 6 23     12     3      8     37     25    +12    39    1.7
## 7 23      9     5      9     29     31     -2    32    1.39
```

Science and Art

```
df.soccer <- bind_cols(teams, stats) %>%
  janitor::row_to_names(row_number = 1) %>%
  rename(team = 'Eastern Conference') %>%
  filter(!grepl('Conference', team)) %>%
  mutate(team = str_remove(team, "[2-9]|1[0-4]?")) %>%
  mutate(abbreviation = str_sub(team, 1, 3),
        team = str_remove(team, ".{0,3}"))
```

Science and Art

```
df.soccer
```

```
## # A tibble: 26 x 10
##   team          W   D   L   F   A   GD
##   <chr>     <chr> <chr> <chr> <chr> <chr> <chr>
## 1 Philadelphia Un~ 14     5     4    44   20   +24
## 2 Toronto FC       13     5     5    33   26    +7
## 3 Columbus Crew SC 12     5     6    36   21   +15
## 4 Orlando City SC 11     8     4    40   25   +15
## 5 New York City FC 12     3     8    37   25   +12
## 6 New York Red Bul~  9     5     9    29   31   -2
## 7 Nashville SC      8     8     7    24   22   +2
## 8 New England Revo~  8     8     7    26   25   +1
## 9 CF Montréal       8     2    13   33   43  -10
## 10 Inter Miami CF   7     3    13   25   35  -10
```

Session functions

- <https://rvest.tidyverse.org/reference/index.html#section-session>

Session

```
session()  is.session()  
session_jump_to()  
session_follow_link()  session_back()  
session_forward()  session_history()  
session_submit()
```

Simulate a session in web browser

Other ideas

- Recipe app
- NLP with articles
- Social media
- DREAM AND EXPLORE!

References and Resources

- rvest vignette: Web Scraping 101
- Amelia McNamara's talk at noRth 2020
- Kasia Kulma: Webscraping with R - from messy & unstructured to blissfully tidy