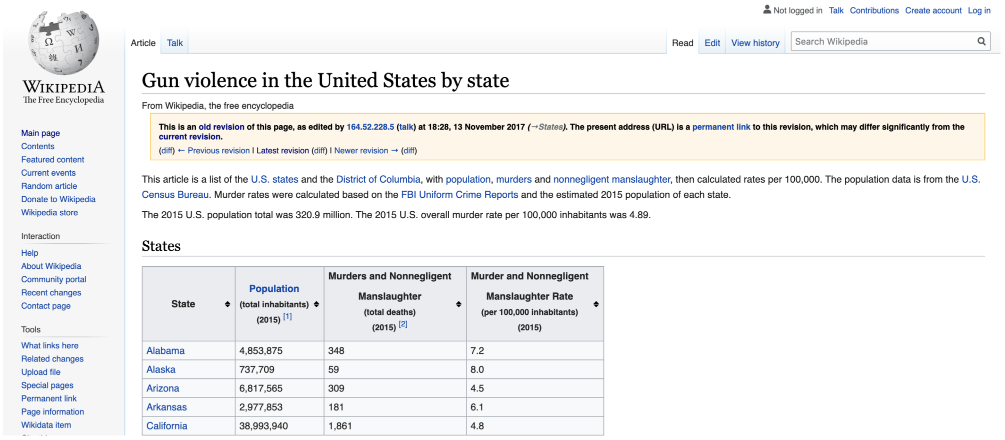
Chapter 23 Web scraping

The data we need to answer a question is not always in a spreadsheet ready for us to read. For example, the US murders dataset we used in the R Basics chapter originally comes from this Wikipedia page:

```
url <- paste0("https://en.wikipedia.org/w/index.php?title=",
              "Gun_violence_in_the_United_States_by_state",
              "&direction=prev&oldid=810166167")
```

for later

You can see the data table when you visit the webpage:



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

Gun violence in the United States by state

From Wikipedia, the free encyclopedia

This is an old revision of this page, as edited by 164.52.228.5 (talk) at 18:28, 13 November 2017 (→States). The present address (URL) is a permanent link to this revision, which may differ significantly from the current revision.
(diff) ← Previous revision | Latest revision (diff) | Newer revision → (diff)

This article is a list of the U.S. states and the District of Columbia, with population, murders and nonnegligent manslaughter, then calculated rates per 100,000. The population data is from the U.S. Census Bureau. Murder rates were calculated based on the FBI Uniform Crime Reports and the estimated 2015 population of each state.

The 2015 U.S. population total was 320.9 million. The 2015 U.S. overall murder rate per 100,000 inhabitants was 4.89.

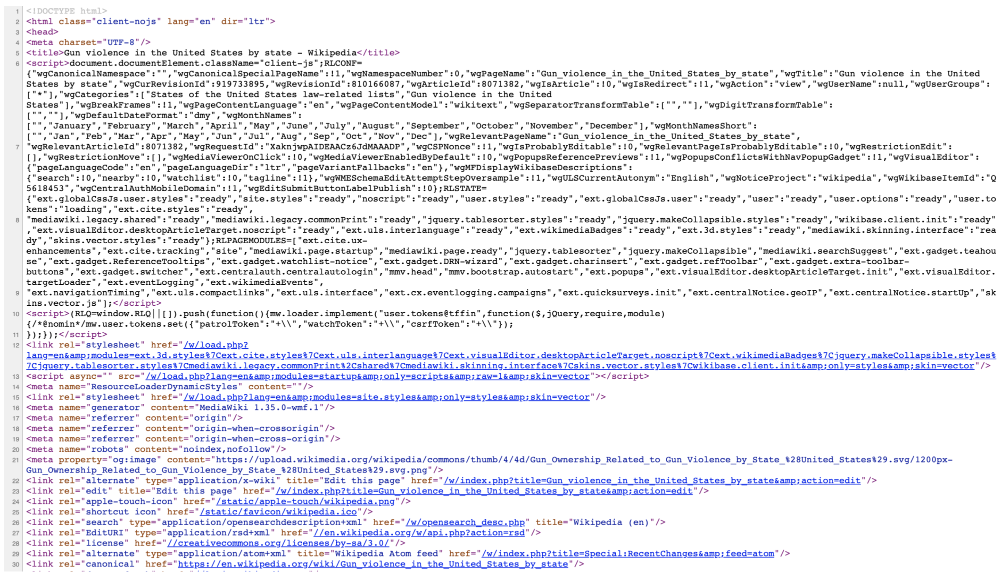
States

State	Population (total inhabitants) (2015) [1]	Murders and Nonnegligent Manslaughter (total deaths) (2015) [2]	Murder and Nonnegligent Manslaughter Rate (per 100,000 inhabitants) (2015)
Alabama	4,853,875	348	7.2
Alaska	737,709	59	8.0
Arizona	6,817,565	309	4.5
Arkansas	2,977,853	181	6.1
California	38,993,940	1,861	4.8

(Web page courtesy of Wikipedia[81]. CC-BY-SA-3.0 license[82]. Screenshot of part of the page.)

Unfortunately, there is no link to a data file. To make the data frame that is loaded when we type `data(murders)` , we had to do some *web scraping*.

Web scraping, or *web harvesting*, is the term we use to describe the process of extracting data from a website. The reason we can do this is because the information used by a browser to render webpages is received as a text file from a server. The text is code written in hyper text markup language (HTML). Every browser has a way to show the html source code for a page, each one different. On Chrome, you can use Control-U on a PC and command+alt+U on a Mac. You will see something like this:



```
<!DOCTYPE html>
<html class="client-nojs" lang="en" dir="ltr">
<head>
<meta charset="UTF-8"/>
<title>Gun violence in the United States by state - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF=
{"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":!1,"wgNamespaceNumber":0,"wgPageName":"Gun_violence_in_the_United_States_by_state","wgTitle":"Gun violence in the United
States by state","wgCurRevisionId":919733895,"wgRevisionId":810166087,"wgArticleId":8071382,"wgIsArticle":!0,"wgIsRedirect":!1,"wgAction":"view","wgUserName":null,"wgUserGroups":
["*"],"wgCategories":["States of the United States law-related lists","Gun violence in the United
States"],"wgBreakFrames":!1,"wgPageContentLanguage":"en","wgPageContentModel":"wikitext","wgSeparatorTransformTable":["",""],"wgDigitTransformTable":
["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":
["","January","February","March","April","May","June","July","August","September","October","November","December"],"wgMonthNamesShort":
["","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"],"wgRelevantPageName":"Gun_violence_in_the_United_States_by_state",
"wgRelevantArticleId":8071382,"wgRequestId":"XaknjwpAIDEAACz6JdMAAADP","wgCSPNonce":!1,"wgIsProbablyEditable":!0,"wgRelevantPageIsProbablyEditable":!0,"wgRestrictionEdit":
[],"wgRestrictionMove":[],"wgMediaViewerOnClick":!0,"wgMediaViewerEnabledByDefault":!0,"wgPopupsReferencePreviews":!1,"wgPopupsConflictsWithNavPopupGadget":!1,"wgVisualEditor":
{"pageLanguageCode":"en","pageLanguageDir":"ltr","pageVariantFallbacks":"en"},"wgMFDisplayWikibaseDescriptions":
{"search":!0,"nearby":!0,"watchlist":!0,"tagline":!1},"wgWMESchemaEditAttemptStepOversample":!1,"wgULSCurrentAutonym":"English","wgNoticeProject":"wikipedia","wgWikibaseItemId":"Q
5618453","wgCentralAuthMobileDomain":!1,"wgEditSubmitButtonLabelPublish":!0};RLSTATE=
{"ext.globalCssJs.user.styles":"ready","site.styles":"ready","noscript":"ready","user.styles":"ready","ext.globalCssJs.user":"ready","user":"ready","user.options":"ready","user.to
kens":"loading","ext.cite.styles":"ready",
"mediawiki.legacy.shared":"ready","mediawiki.legacy.commonPrint":"ready","jquery.tablesorter.styles":"ready","jquery.makeCollapsible.styles":"ready","wikibase.client.init":"ready"
,"ext.visualEditor.desktopArticleTarget.noscript":"ready","ext.uls.interlanguage":"ready","ext.wikimediaBadges":"ready","ext.3d.styles":"ready","mediawiki.skinning.interface":"rea
dy","skins.vector.styles":"ready"};RLPAGEMODULES=["ext.cite.ux-
enhancements","ext.cite.tracking","site","mediawiki.page.startup","mediawiki.page.ready","jquery.tablesorter","jquery.makeCollapsible","mediawiki.searchSuggest","ext.gadget.teahou
se","ext.gadget.ReferenceTooltips","ext.gadget.watchlist-notice","ext.gadget.DRN-wizard","ext.gadget.charinsert","ext.gadget.refToolbar","ext.gadget.extra-toolbar-
buttons","ext.gadget.switcher","ext.centralauth.centralautologin","mmv.head","mmv.bootstrap.autostart","ext.popups","ext.visualEditor.desktopArticleTarget.init","ext.visualEditor.
targetLoader","ext.eventLogging","ext.wikimediaEvents",
"ext.navigationTiming","ext.uls.compactlinks","ext.uls.interface","ext.cx.eventlogging.campaigns","ext.quicksurveys.init","ext.centralNotice.geoIP","ext.centralNotice.startUp","sk
ins.vector.js"];</script>
<script>(RLQ=window.RLQ||[]).push(function(){mw.loader.implement("user.tokens@tffin",function($,jQuery,require,module)
{/*@nomin*/mw.user.tokens.set({"patrolToken":"+\\","watchToken":"+\\","csrfToken":"+\\"});
});});</script>
<link rel="stylesheet" href="/w/load.php?
lang=en&amp;modules=ext.3d.styles%7Cext.cite.styles%7Cext.uls.interlanguage%7Cext.visualEditor.desktopArticleTarget.noscript%7Cext.wikimediaBadges%7Cjquery.makeCollapsible.styles%
7Cjquery.tablesorter.styles%7Cmediawiki.legacy.commonPrint%2Cshared%7Cmediawiki.skinning.interface%7Cskins.vector.styles%7Cwikibase.client.init&amp;only=styles&amp;skin=vector"/>
<script async="" src="/w/load.php?lang=en&amp;modules=startup&amp;only=scripts&amp;raw=1&amp;skin=vector"></script>
<meta name="ResourceLoaderDynamicStyles" content=""/>
<link rel="stylesheet" href="/w/load.php?lang=en&amp;modules=site.styles&amp;only=styles&amp;skin=vector"/>
<meta name="generator" content="MediaWiki 1.35.0-wmf.1"/>
<meta name="referrer" content="origin"/>
<meta name="referrer" content="origin-when-crossorigin"/>
<meta name="referrer" content="origin-when-cross-origin"/>
<meta name="robots" content="noindex,nofollow"/>
<meta property="og:image" content="https://upload.wikimedia.org/wikipedia/commons/thumb/4/4d/Gun_Ownership_Related_to_Gun_Violence_by_State_%28United_States%29.svg/1200px-
Gun_Ownership_Related_to_Gun_Violence_by_State_%28United_States%29.svg.png"/>
<link rel="alternate" type="application/x-wiki" title="Edit this page" href="/w/index.php?title=Gun_violence_in_the_United_States_by_state&amp;action=edit"/>
<link rel="edit" title="Edit this page" href="/w/index.php?title=Gun_violence_in_the_United_States_by_state&amp;action=edit"/>
<link rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png"/>
<link rel="shortcut icon" href="/static/favicon/wikipedia.ico"/>
<link rel="search" type="application/opensearchdescription+xml" href="/w/opensearch_desc.php" title="Wikipedia (en)"/>
<link rel="EditURI" type="application/rsd+xml" href="//en.wikipedia.org/w/api.php?action=rsd"/>
<link rel="license" href="//creativecommons.org/licenses/by-sa/3.0/"/>
<link rel="alternate" type="application/atom+xml" title="Wikipedia Atom feed" href="/w/index.php?title=Special:RecentChanges&amp;feed=atom"/>
<link rel="canonical" href="https://en.wikipedia.org/wiki/Gun_violence_in_the_United_States_by_state"/>
```

23.1 HTML

Because this code is accessible, we can download the HTML file, import it into R, and then write programs to extract the information we need from the page. However, once we look at HTML code, this might seem like a daunting task. But we will show you some convenient tools to facilitate the process. To get an idea of how it works, here are a few lines of code from the Wikipedia page that provides the US murders data:

```
<table class="wikitable sortable">
<tr>
<th>State</th>
<th><a href="/wiki/List_of_U.S._states_and_territories_by_population"
title="List of U.S. states and territories by population">Population</a><br />
<small>(total inhabitants)</small><br />
<small>(2015)</small> <sup id="cite_ref-1" class="reference">
<a href="#cite_note-1">[1]</a></sup></th>
<th>Murders and Nonnegligent
<p>Manslaughter<br />
<small>(total deaths)</small><br />
<small>(2015)</small> <sup id="cite_ref-2" class="reference">
<a href="#cite_note-2">[2]</a></sup></p>
</th>
<th>Murder and Nonnegligent
<p>Manslaughter Rate<br />
<small>(per 100,000 inhabitants)</small><br />
<small>(2015)</small></p>
</th>
</tr>
<tr>
<td><a href="/wiki/Alabama" title="Alabama">Alabama</a></td>
<td>4,853,875</td>
<td>348</td>
<td>7.2</td>
</tr>
<tr>
<td><a href="/wiki/Alaska" title="Alaska">Alaska</a></td>
<td>737,709</td>
<td>59</td>
<td>8.0</td>
</tr>
<tr>
```

You can actually see the data, except data values are surrounded by html code such as `<td>`. We can also see a pattern of how it is stored. If you know HTML, you can write programs that leverage knowledge of these patterns to extract what we want. We also take advantage of a language widely used to make webpages look "pretty" called Cascading Style Sheets (CSS). We say more about this in Section 23.3.

Although we provide tools that make it possible to scrape data without knowing HTML, as a data scientist it is quite useful to learn some HTML and CSS. Not only does this improve your scraping skills, but it might come in handy if you are creating a webpage to showcase your work. There are plenty of online courses and tutorials for learning these. Two examples are Codeacademy[83] and W3schools[84].

23.2 The rvest package

The **tidyverse** provides a web harvesting package called **rvest**. The first step using this package is to import the webpage into R. The package makes this quite simple:

```
library(tidyverse)
library(rvest)
h <- read_html(url)
```

Note that the entire Murders in the US Wikipedia webpage is now contained in `h`. The class of this object is:

```
class(h)
#> [1] "xml_document" "xml_node"
```

The **rvest** package is actually more general; it handles XML documents. XML is a general markup language (that's what the ML stands for) that can be used to represent any kind of data. HTML is a specific type of XML specifically developed for representing webpages. Here we focus on HTML documents.

Now, how do we extract the table from the object `h`? If we print `h`, we don't really see much:

```
h
#> {html_document}
#> <html class="client-nojs" lang="en" dir="ltr">
#> [1] <head>\n<meta http-equiv="Content-Type" content="text/html; chars ...
#> [2] <body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns- ...
```

We can see all the code that defines the downloaded webpage using the `html_text` function like this:

```
html_text(h)
```

We don't show the output here because it includes thousands of characters, but if we look at it, we can see the data we are after are stored in an HTML table: you can see this in this line of the HTML code above `<table class="wikitable sortable">` . The different parts of an HTML document, often defined with a message in between `<` and `>` are referred to as *nodes*. The **rvest** package includes functions to extract nodes of an HTML document: `html_nodes` extracts all nodes of different types and `html_node` extracts the first one. To extract the tables from the html code we use:

```
tab <- h %>% html_nodes("table")
```

Now, instead of the entire webpage, we just have the html code for the tables in the page:

```
tab
#> {xml_nodeset (2)}
#> [1] <table class="wikitable sortable"><tbody>\n<tr>\n<th>State\n</th> ...
#> [2] <table class="nowraplinks hlist mw-collapsible mw-collapsed navbo ...
```

The table we are interested is the first one:

```
tab[[1]]
#> {html_node}
#> <table class="wikitable sortable">
#> [1] <tbody>\n<tr>\n<th>State\n</th>\n<th>\n<a href="/wiki/List_of_U.S ...
```

This is clearly not a tidy dataset, not even a data frame. In the code above, you can definitely see a pattern and writing code to extract just the data is very doable. In fact, **rvest** includes a function just for converting HTML tables into data frames:

```
tab <- tab[[1]] %>% html_table
class(tab)
#> [1] "data.frame"
```

We are now much closer to having a usable data table:

```
tab <- tab %>% setNames(c("state", "population", "total", "murder_rate"))
head(tab)
#>        state population total murder_rate
#> 1    Alabama  4,853,875   348         7.2
#> 2     Alaska    737,709    59         8.0
#> 3    Arizona  6,817,565   309         4.5
#> 4   Arkansas  2,977,853   181         6.1
#> 5 California 38,993,940 1,861         4.8
#> 6   Colorado  5,448,819   176         3.2
```

We still have some wrangling to do. For example, we need to remove the commas and turn characters into numbers. Before continuing with this, we will learn a more general approach to extracting information from web sites.

23.3 CSS selectors

The default look of a webpage made with the most basic HTML is quite unattractive. The aesthetically pleasing pages we see today are made using CSS to define the look and style of webpages. The fact that all pages for a company have the same style usually results from their use of the same CSS file to define the style. The general way these CSS files work is by defining how each of the elements of a webpage will look. The title, headings, itemized lists, tables, and links, for example, each receive their own style including font, color, size, and distance from the margin. CSS does this by leveraging patterns used to define these elements, referred to as *selectors*. An example of such a pattern, which we used above, is `table`, but there are many, many more.

If we want to grab data from a webpage and we happen to know a selector that is unique to the part of the page containing this data, we can use the `html_nodes` function. However, knowing which selector can be quite complicated. In fact, the complexity of webpages has been increasing as they become more sophisticated. For some of the more advanced ones, it seems almost impossible to find the nodes that define a particular piece of data. However, selector gadgets actually make this possible.

SelectorGadget[85] is piece of software that allows you to interactively determine what CSS selector you need to extract specific components from the webpage. If you plan on scraping data other than tables from html pages, we highly recommend you install it. A Chrome extension is available which permits you to turn on the gadget and then, as you click through the page, it highlights parts and shows you the selector you need to extract these parts. There are various demos of how to do this including **rvest** author Hadley Wickham's vignette[86] and other tutorials based on the vignette[87].[88]

23.4 JSON

Sharing data on the internet has become more and more common. Unfortunately, providers use different formats, which makes it harder for data scientists to wrangle data into R. Yet there are some standards that are also becoming more common. Currently, a format that is widely being adopted is the JavaScript Object Notation or JSON. Because this format is very general, it is nothing like a spreadsheet. This JSON file looks more like the code you use to define a list. Here is an example of information stored in a JSON format:

```
#> [
#>   {
#>     "name": "Miguel",
#>     "student_id": 1,
#>     "exam_1": 85,
#>     "exam_2": 86
#>   },
#>   {
#>     "name": "Sofia",
#>     "student_id": 2,
#>     "exam_1": 94,
#>     "exam_2": 93
#>   },
#>   {
#>     "name": "Aya",
#>     "student_id": 3,
#>     "exam_1": 87,
#>     "exam_2": 88
#>   },
#>   {
#>     "name": "Cheng",
#>     "student_id": 4,
#>     "exam_1": 90,
#>     "exam_2": 91
#>   }
#> ]
```

The file above actually represents a data frame. To read it, we can use the function `fromJSON` from the **jsonlite** package. Note that JSON files are often made available via the internet. Several organizations provide a JSON API or a web service that you can connect directly to and obtain data. Here is an example:

```
library(jsonlite)
res <- fromJSON('http://ergast.com/api/f1/2004/1/results.json')
                                                  This file is still there.
citi_bike <- fromJSON("http://citibikenyc.com/stations/json")
                                      This one no longer seems to be there.
```

This downloads a list. The first argument tells you when you downloaded it:

```
citi_bike$executionTime
```

and the second is a data table:

```
citi_bike$stationBeanList %>% as_tibble()
```

You can learn much more by examining tutorials and help files from the **jsonlite** package. This package is intended for relatively simple tasks such as converting data into tables. For more flexibility, we recommend `rjson` .

23.5 Exercises

1. Visit the following web page:
https://web.archive.org/web/20181024132313/http://www.stevetheump.com/Payrolls.htm

Notice there are several tables. Say we are interested in comparing the payrolls of teams across the years. The next few exercises take us through the steps needed to do this.

Start by applying what you learned to read in the website into an object called `h` .

2. Note that, although not very useful, we can actually see the content of the page by typing:

```
html_text(h)
```

The next step is to extract the tables. For this, we can use the `html_nodes` function. We learned that tables in html are associated with the `table` node. Use the `html_nodes` function and the `table` node to extract the first table. Store it in an object `nodes` .

3. The `html_nodes` function returns a list of objects of class `xml_node`. We can see the content of each one using, for example, the `html_text` function. You can see the content for an arbitrarily picked component like this:

```
html_text(nodes[[8]])
```

If the content of this object is an html table, we can use the `html_table` function to convert it to a data frame. Use the `html_table` function to convert the 8th entry of `nodes` into a table.

4. Repeat the above for the first 4 components of `nodes`. Which of the following are payroll tables:

   a. All of them.
   b. 1
   c. 2
   d. 2-4

5. Repeat the above for the first **last** 3 components of `nodes`. Which of the following is true:

   a. The last entry in `nodes` shows the average across all teams through time, not payroll per team.
   b. All three are payroll per team tables.
   c. All three are like the first entry, not a payroll table.
   d. All of the above.

6. We have learned that the first and last entries of `nodes` are not payroll tables. Redefine `nodes` so that these two are removed.

7. We saw in the previous analysis that the first table node is not actually a table. This happens sometimes in html because tables are used to make text look a certain way, as opposed to storing numeric values. Remove the first component and then use `sapply` and `html_table` to convert each node in `nodes` into a table. Note that in this case, `sapply` will return a list of tables. You can also use `lapply` to assure that a list is applied.

8. Look through the resulting tables. Are they all the same? Could we just join them with `bind_rows`?

9. Create two tables, call them `tab_1` and `tab_2` using entries 10 and 19.

10. Use a `full_join` function to combine these two tables. Before you do this you will have to fix the missing header problem. You will also need to make the names match.

11. After joining the tables, you see several NAs. This is because some teams are in one table and not the other. Use the `anti_join` function to get a better idea of why this is happening.

12. We see see that one of the problems is that Yankees are listed as both *N.Y. Yankees* and *NY Yankees*. In the next section, we will learn efficient approaches to fixing problems like this. Here we can do it "by hand" as follows:

```
tab_1 <- tab_1 %>%
  mutate(Team = ifelse(Team == "N.Y. Yankees", "NY Yankees", Team))
```

Now join the tables and show only Oakland and the Yankees and the payroll columns.

13. Advanced: extract the titles of the movies that won Best Picture from this website: https://m.imdb.com/chart/bestpicture/

81. https://en.wikipedia.org/w/index.php?title=Gun_violence_in_the_United_States_by_state&direction=prev&oldid=810166167↩
82. https://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License↩
83. https://www.codecademy.com/learn/learn-html↩
84. https://www.w3schools.com/↩
85. http://selectorgadget.com/↩
86. https://cran.r-project.org/web/packages/rvest/vignettes/selectorgadget.html↩
87. https://stat4701.github.io/edav/2015/04/02/rvest_tutorial/↩
88. https://www.analyticsvidhya.com/blog/2017/03/beginners-guide-on-web-scraping-in-r-using-rvest-with-hands-on-knowledge/↩