# Hassle-free HTML tables with htmltab

Fonte: <https://cran.r-project.org/web/packages/htmltab/vignettes/htmltab.html>

Christian Rubba

2020-03-22

HTML tables are a standard way to display tabular information online. Getting HTML table data into R is fairly straightforward with the readHTMLTable() function of the XML package. But tables on the web are primarily designed for displaying and consuming data, not for analytical purposes. Peculiar design choices for HTML tables are therefore frequently made which tend to produce useless outputs when run through readHTMLTable(). I found that sometimes these outputs could be saved with a little bit of (tedious) post-processing, but just as often they could not. To make working with HTML tables easier and less time-consuming, I developed htmltab, a package for the R system that tries to alleviate these problems directly in the parsing stage when the structural information is still available. Its main advantages over readHTMLTable() are twofold:

* Consideration of row and column spans in the HTML table body and header cells
* More control over the process that translates HTML cells into R table cells

This vignette discusses the application of htmltab for two use cases where the package provides a significant improvement over readHTMLTable().

(I make use of the R packages tidyr and stringr to process table outputs. Neither of the three is required for running htmltab.)

## How to read HTML tables with htmltab()

The principal function of htmltab is htmltab(). The behaviour of htmltab() is modeled closely after readHTMLTable(), and many argument names are identical. Any function call requires passing a value to its docargument. This value may be of three kinds:

1. a URL or file path for the HTML document where the table lives
2. a parsed HTML object of the entire page of class HTMLInternalDocument
3. a table nodeset of class XMLNodeSet

The last of these methods returns a single R table object. For the first two, htmltab() requires users to be specific about the table they would like to have returned. This is done via the which argument. This may be either a numeric value for the table’s position in the page, or a character value that describes an XPath statement.

## Corrections for rowspans and colspans by default

In many HTML tables, spans are used to allow cell values to extend across multiple cells. htmltab() recognizes spans and expands tables automatically. To illustrate this feature, take a look at the HTML table in the Language section of this [Wikipedia page about Demography in the UK](http://en.wikipedia.org/wiki/Demography_of_the_United_Kingdom#Languages). The header information spans across three consecutive rows. To get the table into R, we have to pass an identifiying information to the which argument. I use an XPath statement that I wrote while exploring the HTML page with Web Developer Tools. One that works is “//th[text() = ‘Ability’]/ancestor::table”:

**library**(htmltab)

url <- "http://christianrubba.com/cran/htmltab/vignette/Demography%20of%20the%20United%20Kingdom%20-%20Wikipedia.html"

ukLang <- **htmltab**(doc = url, which = "//th[text() = 'Ability']/ancestor::table")

**head**(ukLang)

## Ability Wales >> Welsh >> Number

## 4 Understands but does not speak, read or write 157,792

## 5 Speaks, reads and writes 430,717

## 6 Speaks but does not read or write 80,429

## 7 Speaks and reads but does not write 45,524

## 8 Reads but does not speak or write 44,327

## 9 Other combination of skills 40,692

## Wales >> Welsh >> % Scotland >> Scottish Gaelic >> Number

## 4 5.15% 23,357

## 5 14.06% 32,191

## 6 2.63% 18,966

## 7 1.49% 6,218

## 8 1.45% 4,646

## 9 1.33% 1,678

## Scotland >> Scottish Gaelic >> % Scotland >> Scots >> Number

## 4 0.46% 267,412

## 5 0.63% 1,225,622

## 6 0.37% 179,295

## 7 0.12% 132,709

## 8 0.09% 107,025

## 9 0.03% 17,381

## Scotland >> Scots >> % Northern Ireland >> Irish >> Number

## 4 5.22% 70,501

## 5 23.95% 71,996

## 6 3.50% 24,677

## 7 2.59% 7,414

## 8 2.09% 5,659

## 9 0.34% 4,651

## Northern Ireland >> Irish >> % Northern Ireland >> Ulster-Scots >> Number

## 4 4.06% 92,040

## 5 4.15% 17,228

## 6 1.42% 10,265

## 7 0.43% 7,801

## 8 0.33% 11,911

## 9 0.27% 959

## Northern Ireland >> Ulster-Scots >> %

## 4 5.30%

## 5 0.99%

## 6 0.59%

## 7 0.45%

## 8 0.69%

## 9 0.06%

The header information has been recast into a format that respects the hierarchical order of the variables and yet only spans a single line in the R table. If you prefer a different seperator between variables, pass it to the headerSep argument. This format was chosen to make further processing of the table easy. For example, using functionality from the tidyr package, the next couple of data cleaning steps may be the following:

**library**(tidyr)

ukLang <- **gather**(ukLang, key, value, -Ability)

This statement restructures the variables in a more useful long format. From this we can separate the variables using an appropriate regular expression such as " >> ".

ukLang <- **separate**(ukLang, key, into = **c**("region", "language", "statistic"), sep = " >> ")

**head**(ukLang)

## Ability region language statistic

## 1 Understands but does not speak, read or write Wales Welsh Number

## 2 Speaks, reads and writes Wales Welsh Number

## 3 Speaks but does not read or write Wales Welsh Number

## 4 Speaks and reads but does not write Wales Welsh Number

## 5 Reads but does not speak or write Wales Welsh Number

## 6 Other combination of skills Wales Welsh Number

## value

## 1 157,792

## 2 430,717

## 3 80,429

## 4 45,524

## 5 44,327

## 6 40,692

htmltab() also automatically expands row and column spans when they appear in the table’s body.

## More control over cell value conversion

htmltab() offers you more control over what part of the HTML table is used in the R table. You can exert this control via htmltab()’s body, header, bodyFun, headerFun, rm\_escape, rm\_footnote, rm\_superscript, rm\_nodata\_cols, rm\_invisible and rm\_whitespace arguments.

### body and header arguments

It is not possible for htmltab() to correctly identify header and body elements in all the tables. Although there is a semantically correct way to organize header and body elements in HTML tables, web designers do not necessarily need to adhere to them to produce visually appealing tables. htmltab employs heuristics for identification but they are no guarantee. If you find that the table is not correctly assembled, you can try to give the function more information through its header and body arguments. These arguments are used to pass information about which rows should be used for the contruction of the header and the body. Both accept numeric values for the rows, but a more robust way is to use an XPath that identifies the respective rows. To illustrate, take a look at this [Wikipedia page about the New Zealand General Election in 2002](http://en.wikipedia.org/wiki/New_Zealand_general_election,_2002#Electorate_results). The table uses cells that span the entire column range to classify General and Maori electorates (yellow background). We need to control for this problem explicitly in the assembling stage. I pass the XPath “//tr[./td[not(@colspan = '10')]]” to the body argument to explicitly discard all rows from the body that have a <td> cell with a colspan attribute of 10:

url <- "http://christianrubba.com/cran/htmltab/vignette/New%20Zealand%20general%20election,%202002%20-%20Wikipedia.html"

xp <- "//caption[starts-with(text(), 'Electorate results')]/ancestor::table"

body\_xp <- "//tr[./td[not(@colspan = '8')]]"

nz1 <- **htmltab**(doc = url, which = xp, body = body\_xp, encoding = "UTF-8")

## No encoding supplied: defaulting to UTF-8.

## Warning: Columns [Incumbent >> Incumbent,Majority >> Majority,Runner up >>

## Runner-up] seem to have no data and are removed. Use rm\_nodata\_cols = F to

## suppress this behavior

**head**(nz1)

## Electorate >> Electorate Incumbent >> Incumbent Winner >> Winner

## 2 Aoraki Jim Sutton Jim Sutton

## 3 Auckland Central Judith Tizard Judith Tizard

## 4 Banks Peninsula Ruth Dyson Ruth Dyson

## 5 Bay of Plenty Tony Ryall Tony Ryall

## 6 Christchurch Central Tim Barnett Tim Barnett

## 7 Christchurch East Lianne Dalziel Lianne Dalziel

## Winner >> Winner.1 Runner up >> Runner-up

## 2 Jim Sutton Wayne Marriott

## 3 Judith Tizard Pansy Wong

## 4 Ruth Dyson David Carter

## 5 Tony Ryall Peter Brown

## 6 Tim Barnett Nicky Wagner

## 7 Lianne Dalziel Stephen Johnston

### Using table information that intercept body rows

In the previous example, we discarded the two intercepting rows in the body which signified the region of the electorate. You might object that ideally these rows should not be discarded, but used for what they are – variable/header information! As of version 0.6.0, htmltab can process these sort of table designs more efficiently and prepend the information accurately in a new column variable. Information to the header argument can now be passed in the form of X1 + X2 + X3 + …, where X1 codifies the main header (i.e. the one that spans the body grid), and X2, X3, … signify groups of header information that appear in the body. Please note that the in-body information (X2, X3, …) must not identify row elements (tr) but individual cells (td or th) from which the value of the new variable can be generated (usually from the node value). To illustrate, consider the following snippet:

nz2 <- **htmltab**(doc = url, which = xp, header = 1 + "//tr/td[@colspan = '8']",

body = "//tr[./td[not(@colspan = '10')]]", encoding = "UTF-8")

## No encoding supplied: defaulting to UTF-8.

## Warning: Columns [Header\_1,Incumbent,Majority,Runner up] seem to have no data

## and are removed. Use rm\_nodata\_cols = F to suppress this behavior

Here, we pass ‘1’ to signify that the main header information appear in the first row. We add to this the XPath “//td[@colspan = '10']” that refer to the two rows. Generally, you are free to use numeric information or XPath to refer to the values that are takes as header variable. htmltab extracts these information and prepends them to the main table.

**tail**(nz2, 9)

## Electorate Incumbent Winner Winner.1

## 63 Wigram Jim Anderton <NA> Jim Anderton

## 65 Electorate Incumbent Winner Winner

## 66 Ikaroa-Rāwhiti Parekura Horomia Parekura Horomia Parekura Horomia

## 67 Tainui New electorate <NA> Nanaia Mahuta

## 68 Tāmaki Makaurau New electorate <NA> John Tamihere

## 69 Te Tai Hauāuru Tariana Turia Tariana Turia Tariana Turia

## 70 Te Tai Tokerau Dover Samuels Dover Samuels Dover Samuels

## 71 Te Tai Tonga Mahara Okeroa Mahara Okeroa Mahara Okeroa

## 72 Waiariki Mita Ririnui Mita Ririnui Mita Ririnui

## Runner up

## 63 Mike Mora

## 65 Runner-up

## 66 Glenis Philip-Barbara

## 67 Willie Jackson

## 68 Metiria Turei

## 69 Ken Mair

## 70 Mere Mangu

## 71 Bill Karaitiana

## 72 Rihi Vercoe

For more information on this feature take a look at this [blog post](http://r-datacollection.com/blog/htmltab-v0.6.0/) and the Details section of the htmltab function in the package documentation.

### Removal of unneeded information

Many HTML tables include additional information which are of little interest to data analysts such as information encoded in superscript and footnote tags, as well as escape sequences. By default, htmltab() removes information from the first two and replaces all escape sequences by a whitespace. You can change this behaviour through the rm\_superscript, rm\_footnotes, rm\_escape, rm\_nodata\_cols, rm\_invisible and rm\_whitespace arguments.

## Conclusion

HTML tables are a valuable data source but they frequently violate basic principles of data well-formedness. This is usually for good reason since their primary purpose is to increase readability of tabular information. htmltab’s goal is to reduce the need for users to interfere when working with HTML tables by relying on available structural information as well as making some assumptions about the table’s design. However, you are free to exert more control over the transformation by specifying various function arguments.