

# Package ‘htmldf’

August 17, 2021

**Title** Simple Scraping and Tidy Webpage Summaries

**Version** 0.4.0

**Author** Alastair Rushworth

**Maintainer** Alastair Rushworth <alastairrushworth@gmail.com>

**Description** Simple tools for scraping webpages, extracting common html tags and parsing contents to a tidy, tabular format. Tools help with extraction of page titles, links, images, rss feeds, social media handles and page metadata.

**License** GPL-2

**Imports** cld3, dplyr, httr, lubridate, magrittr, processx, progress, R.utils, ranger, rvest, stringr, tibble, tidyr, tools, urltools, xml2

**Depends** R (>= 3.5.0)

**Encoding** UTF-8

**Language** en\_GB

**URL** <https://github.com/alastairrushworth/htmldf/>

**BugReports** <https://github.com/alastairrushworth/htmldf/issues>

**RoxygenNote** 7.1.1

**Suggests** testthat

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-08-17 07:30:01 UTC

## R topics documented:

html_df . . . . .	2
<b>Index</b>	<b>4</b>

---

`html_df`*Get a tabular summary of webpage content from a vector of urls*

---

### Description

From a vector of urls, `html_df()` will attempt to fetch the html. From the html, `html_df()` will attempt to look for a page title, rss feeds, images, embedded social media profile handles and other page metadata. Page language is inferred using the package `clld3` which wraps Google's Compact Language Detector 3.

### Usage

```
html_df(  
  urlx,  
  max_size = 5e+06,  
  wait = 0,  
  time_out = 10,  
  show_progress = TRUE,  
  keep_source = TRUE,  
  chrome_bin = NULL  
)
```

### Arguments

<code>urlx</code>	A character vector containing urls. Local files must be prepended with <code>file://</code> .
<code>max_size</code>	Maximum size in bytes of pages to attempt to parse, defaults to 5000000. This is to avoid reading very large pages that may cause <code>read_html()</code> to hang.
<code>wait</code>	Time in seconds to wait between successive requests. Defaults to 0.
<code>time_out</code>	Time in seconds to wait for <code>httr::GET()</code> to complete before exiting. Defaults to 10.
<code>show_progress</code>	Logical, defaults to TRUE. Whether to show progress during download.
<code>keep_source</code>	Logical argument - whether or not to retain the contents of the page source column in the output tibble. Useful to reduce memory usage when scraping many pages. Defaults to TRUE.
<code>chrome_bin</code>	(Optional) Path to a Chromium install to use Chrome in headless mode for scraping

### Value

A tibble with columns

- `url` the original vector of urls provided
- `title` the page title, if found
- `lang` inferred page language
- `url2` the fetched url, this may be different to the original, for example if redirected

- links a list of tibbles of hyperlinks found in <a> tags
- rss a list of embedded RSS feeds found on the page
- tables a list of tables found on the page in descending order of size, coerced to tibble wherever possible.
- images list of tibbles containing image links found on the page
- social list of tibbles containing twitter, linkedin and github user info found on page
- code\_lang numeric indicating inferred code language. A negative values near -1 indicates high likelihood that the language is python, positive values near 1 indicate R. If not code tags are detected, or the language could not be inferred, value is NA.
- size the size of the downloaded page in bytes
- server the page server
- accessed datetime when the page was accessed
- published page publication or last updated date, if detected
- generator the page generator, if found
- status HTTP status code
- source character string of xml documents. These can each be coerced to xml\_document for further processing using rvest using xml2::read\_html().

### Author(s)

Alastair Rushworth

### Examples

```
# Examples require an internet connection...
urlx <- c("https://github.com/alastairrushworth/html_df",
         "https://alastairrushworth.github.io/")
dl <- html_df(urlx)
# preview the dataframe
head(dl)
# social tags
dl$social
# page titles
dl$title
# page language
dl$lang
# rss feeds
dl$rss
# inferred code language
dl$code_lang
# print the page source
dl$source
```

# Index

html\_df, 2