

# Package: polite (via r-universe)

October 29, 2024

**Version** 0.1.3

**Title** Be Nice on the Web

**Description** Be responsible when scraping data from websites by following polite principles: introduce yourself, ask for permission, take slowly and never ask twice.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**ByteCompile** true

**URL** <https://github.com/dmi3kno/polite>,  
<https://dmi3kno.github.io/polite/>

**BugReports** <https://github.com/dmi3kno/polite/issues>

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Imports** htr, magrittr, memoise, ratelimitr, robotstxt, rvest, stats, usethis

**Suggests** dplyr, testthat, covr, webmockr

**Repository** <https://dmi3kno.r-universe.dev>

**RemoteUrl** <https://github.com/dmi3kno/polite>

**RemoteRef** HEAD

**RemoteSha** 8cb4893e4d6029aa31be001d553469dec9aab1e4

## Contents

bow . . . . .	2
guess_basename . . . . .	3
html_attrs_dfr . . . . .	3
nod . . . . .	4
politely . . . . .	5

print.polite . . . . .	6
rip . . . . .	6
scrape . . . . .	7
set_scrape_delay . . . . .	8
use_manners . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

bow	<i>Introduce yourself to the host</i>
-----	---------------------------------------

---

## Description

Introduce yourself to the host

## Usage

```
bow(
  url,
  user_agent = "polite R package",
  delay = 5,
  times = 3,
  force = FALSE,
  verbose = FALSE,
  ...
)
```

```
is.polite(x)
```

## Arguments

url	URL
user_agent	character value passed to user agent string
delay	desired delay between scraping attempts. Final value will be the maximum of desired and mandated delay, as stipulated by robots.txt for relevant user agent
times	number of times to attempt scraping. Default is 3.
force	refresh all memoised functions. Clears up robotstxt and scrape caches. Default is FALSE
verbose	TRUE/FALSE
...	other curl parameters wrapped into <code>httr::config</code> function
x	object of class polite, session

## Value

object of class polite, session

**Examples**

```
library(polite)

host <- "https://www.cheese.com"
session <- bow(host)
session
```

---

guess_basename	<i>Guess download file name from the URL</i>
----------------	--

---

**Description**

Guess download file name from the URL

**Usage**

```
guess_basename(x)
```

**Arguments**

x                      url to guess basename from

**Value**

guessed file name

**Examples**

```
guess_basename("https://bit.ly/polite_sticker")
```

---

html_attrs_dfr	<i>Convert collection of html nodes into data frame</i>
----------------	---

---

**Description**

Convert collection of html nodes into data frame

**Usage**

```
html_attrs_dfr(
  x,
  attrs = NULL,
  trim = FALSE,
  defaults = NA_character_,
  add_text = TRUE
)
```

**Arguments**

<code>x</code>	xml_nodelist object, containing text and attributes of interest
<code>attrs</code>	character vector of attribute names. If missing, all attributes will be used
<code>trim</code>	if TRUE, will trim leading and trailing spaces
<code>defaults</code>	character vector of default values to be passed to <code>rvest::html_attr()</code> . Recycled to match length of <code>attrs</code>
<code>add_text</code>	if TRUE, node content will be added as <code>.text</code> column (using <code>rvest::html_text</code> )

**Value**

data frame with one row per xml node, consisting of an `html_text` column with text and additional columns with attributes

**Examples**

```
library(polite)
library(rvest)
bow("https://en.wikipedia.org/wiki/List_of_cognitive_biases") %>%
  scrape() %>%
  html_nodes("tr td:nth-child(1) a") %>%
  html_attrs_dfr()
```

---

 nod

---

*Agree modification of session path with the host*


---

**Description**

Agree modification of session path with the host

**Usage**

```
nod(bow, path, verbose = FALSE)
```

**Arguments**

<code>bow</code>	object of class <code>polite</code> , session created by <code>polite::bow()</code>
<code>path</code>	string value of path/URL to follow. The function accepts either a path (string part of URL following domain name) or a full URL
<code>verbose</code>	TRUE/FALSE

**Value**

object of class `polite`, session with modified URL

**Examples**

```
library(polite)

host <- "https://www.cheese.com"
session <- bow(host) %>%
  nod(path="by_type")
session
```

---

 politely

*Give your web-scraping function good manners polite*


---

**Description**

Give your web-scraping function good manners polite

**Usage**

```
politely(
  fun,
  user_agent = paste0("polite ", getOption("HTTPUserAgent"), " bot"),
  robots = TRUE,
  force = FALSE,
  delay = 5,
  verbose = FALSE,
  cache = memoise::cache_memory()
)
```

**Arguments**

fun	function to be turned "polite". Must contain an argument named url, which contains url to be queried.
user_agent	optional, user agent string to be used. Defaults to paste("polite", getOption("HTTPUserAgent"), "bot")
robots	optional, should robots.txt be consulted for permissions. Default is TRUE
force	whether or not to force fresh download of robots.txt
delay	minimum delay in seconds, not less than 1. Default is 5.
verbose	output more information about querying process
cache	memoise cache function for storing results. Default memoise::cache_memory()

**Value**

polite function

**Examples**

```
polite_GET <- politely(httr::GET)
```

---

<code>print.polite</code>	<i>Print host introduction object</i>
---------------------------	---------------------------------------

---

**Description**

Print host introduction object

**Usage**

```
## S3 method for class 'polite'
print(x, ...)
```

**Arguments**

<code>x</code>	object of class polite, session
<code>...</code>	other parameters passed to methods

---

<code>rip</code>	<i>Polite file download</i>
------------------	-----------------------------

---

**Description**

Polite file download

**Usage**

```
rip(
  bow,
  destfile = NULL,
  ...,
  mode = "wb",
  path = tempdir(),
  overwrite = FALSE
)
```

**Arguments**

<code>bow</code>	host introduction object of class polite, session created by <code>bow()</code> or <code>nod()</code>
<code>destfile</code>	optional new file name to use when saving the file. If missing, it will be guessed from <code>'basename(url)'</code>
<code>...</code>	other parameters passed to <code>download.file</code>
<code>mode</code>	character. The mode with which to write the file. Useful values are <code>w</code> , <code>wb</code> (binary), <code>a</code> (append) and <code>ab</code> . Not used for methods <code>wget</code> and <code>curl</code> .
<code>path</code>	character. Path where to save the <code>destfile</code> . By default is temporary directory created with <code>tempdir()</code> Ignored if <code>destfile</code> contains path along with filename.
<code>overwrite</code>	if <code>TRUE</code> will overwrite file on disk

**Value**

Full path to the locally saved file indicated by the user in destfile (and path)

**Examples**

```
bow("https://en.wikipedia.org/") %>%
  nod("wiki/Flag_of_the_United_States#/media/File:Flag_of_the_United_States.svg") %>%
  rip()
```

---

 scrape

*Scrape the content of authorized page/API*


---

**Description**

Scrape the content of authorized page/API

**Usage**

```
scrape(
  bow,
  query = NULL,
  params = NULL,
  accept = "html",
  content = NULL,
  verbose = FALSE
)
```

**Arguments**

bow	host introduction object of class <code>polite</code> , session created by <code>bow()</code> or <code>nod()</code>
query	named list of parameters to be appended to URL in the format <code>list(param1=valA, param2=valB)</code>
params	deprecated. Use query argument above.
accept	character value of expected data type to be returned by host (e.g. <code>html</code> , <code>json</code> , <code>xml</code> , <code>csv</code> , <code>txt</code> , etc.)
content	MIME type (aka internet media type) used to override the content type returned by the server. See <a href="http://en.wikipedia.org/wiki/Internet_media_type">http://en.wikipedia.org/wiki/Internet_media_type</a> for a list of common types. You can add the <code>charset</code> parameter to override the server's default encoding
verbose	extra feedback from the function. Defaults to <code>FALSE</code>

**Value**

Object of class `httr::response` which can be further processed by functions in `rvest` package

**Examples**

```
library(rvest)
bow("https://en.wikipedia.org/wiki/List_of_cognitive_biases") %>%
  scrape(content="text/html; charset=UTF-8") %>%
  html_nodes(".wikitable") %>%
  html_table()
```

---

set_scrape_delay	<i>Reset scraping/ripping rate limit</i>
------------------	--

---

**Description**

Reset scraping/ripping rate limit

**Usage**

```
set_scrape_delay(delay)
```

```
set_rip_delay(delay)
```

**Arguments**

delay                    Delay between subsequent requests. Default for package is 5 sec. It can be set lower only under the condition of specifying a custom user-agent string.

**Value**

Updates rate-limit property of scrape and rip functions, respectively.

**Examples**

```
library(polite)

host <- "https://www.cheese.com"
session <- bow(host)
session
```



---

`use_manners`*Use manners in your own package or script*

---

**Description**

Creates collection of polite functions for scraping and downloading

**Usage**

```
use_manners(save_as = "R/polite-scrape.R", open = TRUE)
```

**Arguments**

<code>save_as</code>	File where function should be created Defaults to "R/polite-scrape.R"
<code>open</code>	if TRUE, open the resultant files

# Index

`bow`, 2

`guess_basename`, 3

`html_attrs_dfr`, 3

`is.polite (bow)`, 2

`nod`, 4

`politely`, 5

`print.polite`, 6

`rip`, 6

`scrape`, 7

`set_rip_delay (set_scrape_delay)`, 8

`set_scrape_delay`, 8

`use_manners`, 9