Package: webreadr (via r-universe)

September 7, 2024

Type Package

Title Tools for Reading Formatted Access Log Files

Version 0.4.999999999

Date 2018-01-26

Author Oliver Keyes [aut, cre], Andrew Martin [ctb]

Maintainer Oliver Keyes <ironholds@gmail.com>

Description Read and tidy various common forms of web request log,
including the Common and Combined Web Log formats and various
Amazon access log types.

License MIT + file LICENSE

BugReports https://github.com/Ironholds/webreadr/issues

URL https://github.com/Ironholds/webreadr

Suggests iptools, urltools, rgeolocate, knitr, testthat

LinkingTo Rcpp

Imports Rcpp, readr

VignetteBuilder knitr

RoxygenNote 7.1.0

Repository https://ironholds.r-universe.dev

RemoteUrl https://github.com/ironholds/webreadr

RemoteRef HEAD

RemoteSha 545932629e3e7082911c91ee41a9ac0474832e1d

Contents

`read_aws` *read Amazon CloudFront access logs*

Description

Amazon CloudFront uses access logs with a standard format described on their website. `read_aws` reads these files in; due to the Amazon treatment of header lines, it is capable of organically detecting whether files lack common fields, and compensating for that. See "Details"

Usage

```
read_aws(file)
```

Arguments

`file`	the full path to the AWS file you want to read.

Details

Amazon CloudFront uses tab-separated files with Amazon-specific fields. This can be changed by individual CloudFront users, however, to exclude particular fields, and historically has contained fewer fields than it now does. Luckily, Amazon's insistence on standardisation in field names means that we can organically detect if fields are missing, and compensate for that before reading in the file.

If no fields are missing, the fields returned will be:

- date: the date and time when the request was *completed*
- time_elapsed: the amount of time (in milliseconds) that the connection and fulfilment of the request lasted for.
- edge_location: the Amazon edge location that served the request, identified by a three-letter code. See the Amazon documentation for more details.
- bytes_sent: a count of the number of bytes sent by the server to the client, including headers, to fulfil the request.
- ip_address: the IP address of the client making the request.
- http_method: the HTTP method (POST, GET, etc) used.
- host: the CloudFront host name.
- path: the path to the requested asset.
- status_code: the HTTP status code associated with the request.

- referer: the referer associated with the request.
- user_agent: the user agent of the client that made the request.
- query: the query string associated with the request; if there is no query string, this will be a dash.
- cookie: the cookie header from the request, stored as name-value pairs. When no cookie header is provided, or it is empty, this will be a dash.
- result_type: the result of the request. This is similar to Squid response codes (see `read_squid`) but Amazon-specific; their documentation contains details on what each code means.
- request_id: A hashed unique identifier for each request.
- host_header: the host header of the requested asset. While `host` will always be the CloudFront host name, `host_header` contains alternate domain names (or 'CNAMES') when the CloudFront distribution is using them.
- protocol: the protocol used in the request (http/https).
- bytes_received: client-to-server bytes, including headers.
- time_elapsed: the time elapsed, in seconds, between the time the request was received and the time the server completed responding to it.
- forwarded_for: If the viewer used an HTTP proxy or a load balancer to send the request, the value of `ip_address` is the IP address of the proxy or load balancer. In that case, x-forwarded-for is the IP address of the viewer that originated the request. If the viewer did not use an HTTP proxy or a load balancer, the value of `forwarded_for` is a hyphen (-).
- ssl_protocol: When `cs_protocol` is https, the SSL protocol that the client and CloudFront negotiated for encrypting the request and response. When `cs_protocol` is http, the value for `ssl-protocol` is a hyphen (-).
- ssl_cipher: When `cs_protocol` is https, the SSL cipher that the client and CloudFront negotiated for encrypting the request and response. When `cs_protocol` is http, the value for `ssl_cipher` is a hyphen (-).
- response_result_type: How CloudFront classified the response just before returning the response to the viewer.
- protocol_version: The HTTP version that the viewer specified in the request.
- fle_status: When field-level encryption is configured for a distribution, this field contains a code that indicates whether the request body was successfully processed.
- fle_encrypted_fields: The number of fields that CloudFront encrypted and forwarded to the origin.
- port: The port number of the request from the viewer.
- time_to_first_byte: The number of seconds between receiving the request and writing the first byte of the response, as measured on the server.
- detailed_result_type: When `result_type` is not Error, this field contains the same value as `result_type`.
- content_type: The value of the HTTP Content-Type header of the response.
- content_length: The value of the HTTP Content-Length header of the response.
- content_range_start: When the response contains the HTTP Content-Range header, this field contains the range start value.
- content_range_end: When the response contains the HTTP Content-Range header, this field contains the range end value.

See Also

read_s3, for Amazon S3 files, read_clf for the Common Log Format, read_squid and read_combined.

Examples

```
#Read in an example CloudFront file provided with the webreadr package.
data <- read_aws(system.file("extdata/log.aws", package = "webreadr"))
```

read_bro *Read bro logfiles*

Description

This function reads logfiles exported from 'bro', automatically detecting and handling different types of log and returning them as data.frames. The full range of bro files are not yet supported, but more will be added over time.

Usage

```
read_bro(file)
```

Arguments

file	the path to the logfile you want to read in

See Also

read_clf, read_squid and other readers for different log formats.

Examples

```
# Read an FTP log
data <- read_bro(system.file("extdata/ftp.log", package = "webreadr"))
```

`read_clf` *read CLF-formatted logs*

Description

Read a file of request logs stored in the Common Log Format.

Usage

```
read_clf(file, has_header = FALSE)
```

Arguments

`file`	the full path to the CLF-formatted file you want to read.
`has_header`	whether or not the file has a header row. Set to FALSE by default.

Details

the CLF is a standardised format for web request logs. It consists of the fields:

- ip_address: the IP address of the remote host that made the request. The CLF does not (by default) include the de-facto standard X-Forwarded-For header
- remote_user_ident: the RFC 1413 remote user identifier.
- local_user_ident: the identifier the user has authenticated with locally.
- timestamp: the timestamp associated with the request, stored as "[08/Apr/2001:17:39:04 -0800]", where "-0800" represents the time offset (minus eight hours) of the timestamp from UTC.
- request: the actual user request, containing the HTTP method used, the asset requested, and the HTTP Protocol version used.
- status_code: the HTTP status code returned.
- bytes_sent: the number of bytes sent

While outdated as a standard, systems using the CLF are still around; the Squid caching system, for example, uses the CLF as one of its default log formats (the other, the squid "native" format, can be read with `read_squid`).

Value

a data.frame consisting of seven fields, as discussed above, with normalised timestamps.

See Also

`read_combined` for the /Combined/ Log Format, and `split_clf` for splitting out the "requests" field.

Examples

```
#Read in an example CLF-formatted file provided with the webreadr package.
data <- read_clf(system.file("extdata/log.clf", package = "webreadr"))
```

read_combined *read Combined Log Format files*

Description

read requests logs following the Combined Log Format.

Usage

```
read_combined(file, has_header = FALSE)
```

Arguments

file	the full path to the CLF-formatted file you want to read.
has_header	whether or not the file has a header row. Set to FALSE by default.

Details

the Combined Log Format (CLF) is the same as the Common Log Format (CLF, because software engineers and naming go together like chalk and cheese), which is documented at read_clf. In addition to the fields described there, the Combined Log Format also includes:

- referer: the referer associated with the request.
- user_agent: the user agent of the user that made the request.

read_combined handles these fields, as well as the CLF-standard ones. This is (amongst other things) the default logging format for nginx servers

See Also

read_clf for the /Common/ Log Format, and split_clf for splitting out the "requests" field.

Examples

```
#Read in an example Combined-formatted file provided with the webreadr package.
data <- read_combined(system.file("extdata/combined_log.clf", package = "webreadr"))
```

read_iis *Read Microsoft IIS Logs*

Description

read_iis provides a reader for Microsoft IIS access logs.

Usage

```
read_iis(file)
```

Arguments

file	the full path to the IIS log file you want to read.

Examples

```
# Using the inbuilt testing dataset
iis_data <- read_iis(system.file(file.path("extdata", "iis.log"), package = "webreadr"))
```

read_s3 *Read Amazon S3 Access Logs*

Description

read_s3 provides a reader for Amazon's S3 service's access logs, described here.

Usage

```
read_s3(file)
```

Arguments

file	the full path to the S3 file you want to read.

Details

S3 access logs contain information about requests to S3 buckets, and follow a standard format described here.

The fields for S3 files are:

- owner: the owner of the S3 bucket; a hashed user ID
- bucket: the bucket that processed the request.
- request_time: the time that a request was received. Formatted as POSIXct timestamps.

- remote_ip: the IP address that made the request.
- requester: the user ID of the person making the request; `Anonymous` if the request was not authenticated.
- operation: the actual operation performed with the request.
- key: the request's key, normally an encoded URL fragment or NA if the operation did not contain a key.
- uri: the full URI for the request, as well as the HTTP method and version. `split_clf` works to split this into a data.frame of 3 columns.
- status: the HTTP status code associated with the request.
- error: the error code, if an error occurred; NA otherwise. See here for more information about S3 error codes.
- sent: the number of bytes returned in response to the request.
- size: the total size of the returned object.
- time: the number of milliseconds between the request being sent and the response being sent, from the server's perspective.
- turn_around: the number of milliseconds the S3 bucket spent processing the request.
- referer: the referer associated with the request.
- user_agent: the user agent associated with the request.
- version_id: the version ID of the request; NA if the requested operation does not involve a version ID.

See Also

`read_aws` for reading Amazon Web Services (AWS) access log files, and `split_clf`, which works well on the `uri` field from S3 files.

Examples

```
# Using the inbuilt testing dataset
s3_data <- read_s3(system.file("extdata/s3.log", package = "webreadr"))
```

read_squid *read Squid files*

Description

the Squid default log formats are either the CLF - for which, use `read_clf` - or the "native" Squid format, which is described in more detail below. `read_squid` allows you to read the latter.

Usage

```
read_squid(file, has_header = FALSE)
```

Arguments

`file` the full path to the CLF-formatted file you want to read.

`has_header` whether or not the file has a header row. Set to FALSE by default.

Details

The log format for Squid servers can be custom-set, but by default follows one of two patterns; it's either the Common Log Format (CLF), which you can read in with `read_clf`, or the "native log format", a Squid-specific format handled by this function. It consists of the fields:

- timestamp: the timestamp identifying when the request was received. This is stored (from the file's point of view) as a count of seconds, in UNIX time: `read_squid` turns them into POSIXlt timestamps, assuming UTC as an origin timezone.
- time_elapsed: the amount of time (in milliseconds) that the connection and fulfilment of the request lasted for.
- ip_address: the IP address of the remote host making the request.
- status_code: the status code and Squid response code associated with that request, stored as a single field. This can be split into two distinct fields with `split_squid`
- bytes_sent: the number of bytes sent
- http_method: the HTTP method (POST, GET, etc) used.
- url: the URL of the requested asset.
- remote_user_ident: the RFC 1413 remote user identifier.
- peer_info: the status of how forwarding to a peer server was handled and, if the request was forwarded, the server it was sent to.

See Also

`read_clf` for the Common Log Format (also used by Squids), and `split_squid` for splitting the "status_code" field into its component parts.

Examples

```
#Read in an example Squid file provided with the webreadr package.
data <- read_squid(system.file("extdata/log.squid", package = "webreadr"))
```

`split_clf` *split requests from a CLF-formatted file*

Description

CLF (Combined/Common Log Format) files store the HTTP method, protocol and asset requested in the same field. `split_clf` takes this field as a vector and returns a data.frame containing these elements in distinct columns. The function also works nicely with the `uri` field from Amazon S3 files (see `read_s3`).

Usage

```
split_clf(requests)
```

Arguments

`requests`	the "request" field from a CLF-formatted file, read in with `read_clf` or `read_combined`.

Value

a data.frame of three columns - "method", "asset" and "protocol" - representing, respectively, the HTTP method used ("GET"), the asset requested ("/favicon.ico") and the protocol used ("HTTP/1.0"). In cases where the request is not intact (containing, for example, just the protocol or just the asset), NAs will be returned.

See Also

`read_clf` and `read_combined` for reading in these files.

Examples

```
# Grab CLF data and split out the request.
data <- read_combined(system.file("extdata/combined_log.clf", package = "webreadr"))
requests <- split_clf(data$request)

# An example using S3 files
s3_data <- read_s3(system.file("extdata/s3.log", package = "webreadr"))
s3_requests <- split_clf(s3_data$uri)
```

split_squid *split the "status_code" field in a Squid-formatted dataset.*

Description

the Squid data format (which can be read in with `read_squid`) stores the squid response and the HTTP status code as a single field. `split_squid` allows you to split these into a data.frame of two distinct columns.

Usage

```
split_squid(status_codes)
```

Arguments

`status_codes`	a `status_code` column from a Squid file read in with `read_squid`

Value

a data.frame of two columns - "squid_code" and "http_status" - representing, respectively, the Squid response to the request and the HTTP status of it. In cases where the status code is not intact (containing, for example, just the squid_code), NAs will be returned.

See Also

`read_squid` for reading these files in, and `split_clf` for similar parsing of multi-field columns in Common/Combined Log Format (CLF) data.

Examples

```
#Read in an example Squid file provided with the webtools package, then split out the codes
data <- read_squid(system.file("extdata/log.squid", package = "webreadr"))
statuses <- split_squid(data$status_code)
```

`webreadr` *A package for reading various common forms of request log*

Description

see the introductory vignette for more details!

Index