

Package ‘ndjson’

September 18, 2018

Type Package

Title Wicked-Fast Streaming 'JSON' ('ndjson') Reader

Version 0.7.0

Date 2018-08-16

Maintainer Bob Rudis <bob@rud.is>

Description Streaming 'JSON' ('ndjson') has one 'JSON' record per-line and many modern 'ndjson' files contain large numbers of records. These constructs may not be columnar in nature, but it is often useful to read in these files and ``flatten'' the structure out to enable working with the data in an R 'data.frame'-like context. Functions are provided that make it possible to read in plain 'ndjson' files or compressed ('gz') 'ndjson' files and either validate the format of the records or create ``flat'' 'data.table' structures from them.

URL <http://gitlab.com/hrbrmstr/ndjson>

BugReports <https://gitlab.com/hrbrmstr/ndjson/issues>

SystemRequirements zlib, C++14

NeedsCompilation yes

License MIT + file LICENSE

Encoding UTF-8

Suggests testthat

Depends R (>= 3.2.0)

Imports Rcpp, data.table, dplyr, dtplyr

LinkingTo Rcpp

RoxygenNote 6.0.1.9000

Author Bob Rudis [aut, cre] (<<https://orcid.org/0000-0001-5670-2640>>),
Niels Lohmann [aut] (C++ json parser),
Deepak Bandyopadhyay [aut] (C++ gzstream),
Lutz Kettner [aut] (C++ gzstream)

Repository CRAN

Date/Publication 2018-09-18 09:00:03 UTC

R topics documented:

flatten	2
ndjson	2
stream_in	3
validate	3

Index	5
--------------	----------

flatten	<i>Flatten a character vector of individual JSON lines into a tbl_dt</i>
---------	--

Description

Flatten a character vector of individual JSON lines into a `tbl_dt`

Usage

```
flatten(x, cls = c("dt", "tbl"))
```

Arguments

<code>x</code>	character vector of individual JSON lines to flatten
<code>cls</code>	the package uses <code>data.table::rbindlist</code> for speed but that's not always the best return type for everyone, so you have option of keeping it a <code>tbl_dt</code> via "dt" or converting it to a <code>tbl</code>

Value

`tbl_dt` or `tbl` or `data.frame`

Examples

```
flatten('{"top":{"next":{"final":1,"end":true},"another":"yes"},"more":"no"}')
```

ndjson	<i>Wicked-fast Streaming JSON ('ndjson) Reader</i>
--------	--

Description

Wicked-fast Streaming JSON ('ndjson) Reader

Author(s)

Bob Rudis (bob@rud.is)

stream_in	<i>Stream in & flatten an ndjson file into a tbl_dt</i>
-----------	---

Description

Given a file of streaming JSON (ndjson) this function reads in the records and creates a flat `data.table` / `tbl_dt` from it.

Usage

```
stream_in(path, cls = c("dt", "tbl"))
```

Arguments

path	path to file (supports "gz" files)
cls	the package uses <code>data.table::rbindlist</code> for speed but that's not always the best return type for everyone, so you have option of keeping it a <code>tbl_dt</code> via "dt" or converting it to a <code>tbl</code>

Value

`tbl_dt` or `tbl` or `data.frame`

References

<http://ndjson.org/>

Examples

```
f <- system.file("extdata", "test.json", package="ndjson")
nrow(stream_in(f))

gzf <- system.file("extdata", "testgz.json.gz", package="ndjson")
nrow(stream_in(gzf))
```

validate	<i>Validate ndjson file</i>
----------	-----------------------------

Description

Given a file of streaming JSON (ndjson) this function reads in the records and validates that they are all legal JSON records. If the `verbose` parameter is `TRUE` and errors are found, the line numbers of the errant records will be displayed.

Usage

```
validate(path, verbose = FALSE)
```

Arguments

path path to file (supports "gz" files)
verbose display verbose information (filename and line numbers with bad records)

Value

logical

References

<http://ndjson.org/>

Examples

```
f <- system.file("extdata", "test.json", package="ndjson")
validate(f)

gzf <- system.file("extdata", "testgz.json.gz", package="ndjson")
validate(gzf)
```

Index

`flatten`, [2](#)

`ndjson`, [2](#)

`ndjson-package (ndjson)`, [2](#)

`stream_in`, [3](#)

`validate`, [3](#)