

# Package ‘yyjsonr’

July 21, 2025

**Type** Package

**Title** Fast 'JSON', 'NDJSON' and 'GeoJSON' Parser and Generator

**Version** 0.1.21

**Maintainer** Mike Cheng <mikefc@coolbutuseless.com>

**Description** A fast 'JSON' parser, generator and validator which converts 'JSON', 'NDJSON' (Newline Delimited 'JSON') and 'GeoJSON' (Geographic 'JSON') data to/from R objects. The standard R data types are supported (e.g. logical, numeric, integer) with configurable handling of NULL and NA values. Data frames, atomic vectors and lists are all supported as data containers translated to/from 'JSON'. 'GeoJSON' data is read in as 'simple features' objects. This implementation wraps the 'yyjson' 'C' library which is available from <<https://github.com/ibireme/yyjson>>.

**License** MIT + file LICENSE

**URL** <https://github.com/coolbutuseless/yyjsonr>,  
<https://coolbutuseless.github.io/package/yyjsonr/>

**BugReports** <https://github.com/coolbutuseless/yyjsonr/issues>

**Encoding** UTF-8

**Language** en-AU

**RoxygenNote** 7.3.2

**Suggests** bit64, knitr, rmarkdown, jsonlite, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**Copyright** The included 'yyjson' code is Copyright (c) 2020 YaoYuan.  
See 'COPYRIGHTS' for LICENSE for included code.

**Depends** R (>= 4.1.0)

**NeedsCompilation** yes

**Author** Mike Cheng [aut, cre, cph],  
Yao Yuan [aut, cph] (Author of bundled yyjson),  
Murat Tasan [ctb] ('json\_verbatim' handling)

**Repository** CRAN  
**Date/Publication** 2025-06-02 22:20:01 UTC

**Contents**

|                              |           |
|------------------------------|-----------|
| as_scalar . . . . .          | 2         |
| opts_read_geojson . . . . .  | 3         |
| opts_read_json . . . . .     | 4         |
| opts_write_geojson . . . . . | 5         |
| opts_write_json . . . . .    | 6         |
| read_geojson_str . . . . .   | 8         |
| read_json_connn . . . . .    | 8         |
| read_json_file . . . . .     | 9         |
| read_json_raw . . . . .      | 10        |
| read_json_str . . . . .      | 11        |
| read_ndjson_file . . . . .   | 11        |
| read_ndjson_raw . . . . .    | 13        |
| read_ndjson_str . . . . .    | 14        |
| validate_json_file . . . . . | 15        |
| write_geojson_str . . . . .  | 16        |
| write_json_file . . . . .    | 17        |
| write_json_raw . . . . .     | 17        |
| write_json_str . . . . .     | 18        |
| write_ndjson_file . . . . .  | 19        |
| write_ndjson_raw . . . . .   | 20        |
| write_ndjson_str . . . . .   | 20        |
| yyjson_read_flag . . . . .   | 21        |
| yyjson_version . . . . .     | 22        |
| yyjson_write_flag . . . . .  | 23        |
| <b>Index</b>                 | <b>25</b> |

---

|           |                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------|
| as_scalar | <i>Tag an atomic vector with a single element as class 'scalar'. When output to JSON it will be output as a scalar not a vector</i> |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------|

---

**Description**

Tag an atomic vector with a single element as class 'scalar'. When output to JSON it will be output as a scalar not a vector

**Usage**

as\_scalar(x)

**Arguments**

x                      atomic vector with length = 1

**Value**

If `x` is an atomic vector with length = 1, then object is returned with class 'scalar', otherwise object is returned unmodified

---

|                   |                                       |
|-------------------|---------------------------------------|
| opts_read_geojson | <i>Options for reading in GeoJSON</i> |
|-------------------|---------------------------------------|

---

**Description**

Options for reading in GeoJSON

**Usage**

```
opts_read_geojson(
  type = c("sf", "sfc"),
  property_promotion = c("string", "list"),
  property_promotion_lgl = c("integer", "string")
)
```

**Arguments**

`type` 'sf' or 'sfc'

`property_promotion`

What is the most general container type to use when properties differ across a FEATURECOLLECTION? E.g. if the property exists both as a numeric and a string, should all values be promoted to a 'string', or contained as different types in a 'list'. Default: 'string' will behave like geojsonsf package.

`property_promotion_lgl`

when `property_promotion = "string"` should logical values become words (i.e. "TRUE"/"FALSE") or integers (i.e. "1"/"0"). Default: "integer" in order to match geojsonsf package

**Value**

Named list of options specific to reading GeoJSON

**Examples**

```
# Create a set of options to use when reading geojson
opts_read_geojson()
```

---

|                |                                                      |
|----------------|------------------------------------------------------|
| opts_read_json | Create named list of options for parsing R from JSON |
|----------------|------------------------------------------------------|

---

## Description

Create named list of options for parsing R from JSON

## Usage

```
opts_read_json(
  promote_num_to_string = FALSE,
  digits_promote = 6,
  df_missing_list_elem = NULL,
  obj_of_arrs_to_df = TRUE,
  arr_of_objs_to_df = TRUE,
  str_specials = c("string", "special"),
  num_specials = c("special", "string"),
  int64 = c("string", "double", "bit64"),
  length1_array_asis = FALSE,
  single_null = NULL,
  yyjson_read_flag = 0L
)
```

## Arguments

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| promote_num_to_string | Should numeric values be promoted to strings when they occur within an array with other string values? Default: FALSE means to keep numerics as numeric value and promote the <i>container</i> to be a list rather than an atomic vector when types are mixed. If TRUE then array of mixed string/numeric types will be promoted to all string values and returned as an atomic character vector. Set this to TRUE if you want to emulate the behaviour of <code>jsonlite::fromJSON()</code> |
| digits_promote        | When promoting numbers to be strings, how many decimal places should be used in the representation. Default: 6. This option is only valid if <code>promote_num_to_string = TRUE</code> . Valid range 0 to 30                                                                                                                                                                                                                                                                                 |
| df_missing_list_elem  | R value to use when elements are missing in list columns in data.frames. Default: NULL                                                                                                                                                                                                                                                                                                                                                                                                       |
| obj_of_arrs_to_df     | logical. Should a named list of equal-length vectors be promoted to a data.frame? Default: TRUE. If FALSE, then result will be left as a list.                                                                                                                                                                                                                                                                                                                                               |
| arr_of_objs_to_df     | logical. Should an array or objects be promoted to a data.frame? Default: TRUE. If FALSE, then results will be read as a list-of-lists.                                                                                                                                                                                                                                                                                                                                                      |
| str_specials          | Should 'NA' in a JSON string be converted to the 'special' NA value in R, or left as a 'string'. Default: 'string'                                                                                                                                                                                                                                                                                                                                                                           |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| num_specials       | Should JSON strings 'NA'/'Inf'/'NaN' in a numeric context be converted to the 'special' R numeric values NA, Inf, NaN, or left as a 'string'. Default: 'special'                                                                                                                                                                                                                                                                       |
| int64              | how to encode large integers which do not fit into R's integer type. 'string' imports them as a character vector. 'double' will convert the integer to a double precision numeric value. 'bit64' will use the 'integer64' type from the 'bit64' package. Note that the 'integer64' type is a <i>signed</i> integer type, and a warning will be issued if JSON contains an <i>unsigned</i> integer which cannot be stored in this type. |
| length1_array_asis | logical. Should JSON arrays with length = 1 be marked with class AsIs. Default: FALSE                                                                                                                                                                                                                                                                                                                                                  |
| single_null        | R object to return for isolated JSON null values. Default: NULL. Note: JSON null values in arrays may still be promoted to NAs of the appropriate type if possible.                                                                                                                                                                                                                                                                    |
| yyjson_read_flag   | integer vector of internal yyjson options. See yyjson_read_flag in this package, and read the yyjson API documentation for more information. This is considered an advanced option.                                                                                                                                                                                                                                                    |

**Value**

Named list of options for reading JSON

**See Also**

[yyjson\\_read\\_flag\(\)](#)

**Examples**

```
opts_read_json()
```

---

|                    |                                                      |
|--------------------|------------------------------------------------------|
| opts_write_geojson | <i>Options for writing from sf object to GeoJSON</i> |
|--------------------|------------------------------------------------------|

---

**Description**

Currently no options available.

**Usage**

```
opts_write_geojson()
```

**Value**

Named list of options specific to writing GeoJSON

## Examples

```
# Create a set of options to use when writing geojson
opts_write_geojson()
```

---

|                 |                                                        |
|-----------------|--------------------------------------------------------|
| opts_write_json | Create named list of options for serializing R to JSON |
|-----------------|--------------------------------------------------------|

---

## Description

Create named list of options for serializing R to JSON

## Usage

```
opts_write_json(
  digits = -1L,
  digits_secs = 0L,
  digits_signif = -1L,
  pretty = FALSE,
  auto_unbox = FALSE,
  dataframe = c("rows", "columns"),
  factor = c("string", "integer"),
  name_repair = c("none", "minimal"),
  num_specials = c("null", "string"),
  str_specials = c("null", "string"),
  fast_numerics = FALSE,
  json_verbatim = FALSE,
  yyjson_write_flag = 0L
)
```

## Arguments

|               |                                                                                                                                                                                                                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| digits        | decimal places to keep for floating point numbers. Default: -1. Positive values specify number of decimal places. Using zero will write the numeric value as an integer. Values less than zero mean that the floating point value should be written as-is (the default). This argument is ignored if digits_signif is greater than zero. |
| digits_secs   | decimal places for fractional seconds when converting times to a string representation. Default: 0. Valid range: 0 to 6                                                                                                                                                                                                                  |
| digits_signif | significant decimal places to store in floating point numbers. Default: -1 means to output the number as-is (while respecting the digits) argument. Values above 0 will produce rounding to the given number of places and the digits argument will be ignored.                                                                          |
| pretty        | Logical value indicating if the created JSON string should have whitespace for indentation and linebreaks. Default: FALSE. Note: this option is equivalent to yyjson_write_flag = write_flag\$YYJSON_WRITE_PRETTY                                                                                                                        |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| auto_unbox        | automatically unbox all atomic vectors of length 1 such that they appear as atomic elements in JSON rather than arrays of length 1.                                                                                                                                                                                                                                                                                                                                                      |
| dataframe         | how to encode data.frame objects. Options 'rows' or 'columns'. Default: 'rows'                                                                                                                                                                                                                                                                                                                                                                                                           |
| factor            | how to encode factor objects: must be one of 'string' or 'integer' Default: 'string'                                                                                                                                                                                                                                                                                                                                                                                                     |
| name_repair       | How should unnamed items in a partially named list be handled? 'none' means to leave their names blank in JSON (which may not be valid JSON). 'minimal' means to use the integer position index of the item as its name if it is missing. Default: 'none'                                                                                                                                                                                                                                |
| num_specials      | Should special numeric values (i.e. NA, NaN, Inf) be converted to a JSON null value or converted to a string representation e.g. "NA"/"NaN" etc. Default: 'null'                                                                                                                                                                                                                                                                                                                         |
| str_specials      | Should a special value of NA in a character vector be converted to a JSON null value, or converted to a string "NA"? Default: 'null'                                                                                                                                                                                                                                                                                                                                                     |
| fast_numerics     | Does the user guarantee that there are no NA, NaN or Inf values in the numeric vectors? Default: FALSE. If TRUE then numeric and integer vectors will be written to JSON using a faster method. Note: if there are NA, NaN or Inf values, an error will be thrown. Expert users are invited to also consider the YYJSON_WRITE_ALLOW_INF_AND_NAN and YYJSON_WRITE_INF_AND_NAN_AS_NULL options for yyjson_write_flags and should consult the yyjson API documentation for further details. |
| json_verbatim     | Write strings with class 'json' directly into the result? Default: FALSE. If json_verbatim = TRUE and a string has the class "json", then it will be written verbatim into the output.                                                                                                                                                                                                                                                                                                   |
| yyjson_write_flag | integer vector corresponding to internal yyjson options. See yyjson_write_flag in this package, and read the yyjson API documentation for more information. This is considered an advanced option.                                                                                                                                                                                                                                                                                       |

**Value**

Named list of options for writing JSON

**See Also**

[yyjson\\_write\\_flag\(\)](#)

**Examples**

```
write_json_str(head(iris, 3), opts = opts_write_json(factor = 'integer'))
```

---

|                  |                                  |
|------------------|----------------------------------|
| read_geojson_str | <i>Load GeoJSON as sf object</i> |
|------------------|----------------------------------|

---

### Description

Coordinate reference system is always WGS84 in accordance with GeoJSON RFC.

### Usage

```
read_geojson_str(str, opts = list(), ..., json_opts = list())
read_geojson_file(filename, opts = list(), ..., json_opts = list())
```

### Arguments

|           |                                                                                                                                                                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| str       | Single string containing GeoJSON.                                                                                                                                                                                                                                                                              |
| opts      | Named list of GeoJSON-specific options. Usually created with <code>opts_read_geojson()</code> . Default: empty <code>list()</code> to use the default options.                                                                                                                                                 |
| ...       | Any extra named options override those in GeoJSON-specific options - <code>opts</code>                                                                                                                                                                                                                         |
| json_opts | Named list of vanilla JSON options as used by <code>read_json_str()</code> . This is usually created with <code>opts_read_json()</code> . Default value is an empty <code>list()</code> which means to use all the default JSON parsing options which is usually the correct thing to do when reading GeoJSON. |
| filename  | Filename                                                                                                                                                                                                                                                                                                       |

### Value

sf object

### Examples

```
geojson_file <- system.file("geojson-example.json", package = 'yyjsonr')
read_geojson_file(geojson_file)
```

---

|                |                                                |
|----------------|------------------------------------------------|
| read_json_conn | <i>Parse JSON from an R connection object.</i> |
|----------------|------------------------------------------------|

---

### Description

Currently, this is not very efficient as the entire contents of the connection are read into R as a string and then the JSON parsed from there.

### Usage

```
read_json_conn(conn, opts = list(), ...)
```

**Arguments**

|      |                                                                                                                                                                       |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| conn | connection object. e.g. <code>url('https://jsonplaceholder.typicode.com/todos/1')</code>                                                                              |
| opts | Named list of options for parsing. Usually created by <code>opts_read_json()</code>                                                                                   |
| ...  | Other named options can be used to override any options in <code>opts</code> . The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |

**Details**

For plain text files it is faster to use `read_json_file()`.

**Value**

R object

**See Also**

Other JSON Parsers: [read\\_json\\_file\(\)](#), [read\\_json\\_raw\(\)](#), [read\\_json\\_str\(\)](#), [read\\_ndjson\\_file\(\)](#), [read\\_ndjson\\_raw\(\)](#), [read\\_ndjson\\_str\(\)](#)

**Examples**

```
if (interactive()) {
  read_json_conn(url("https://api.github.com/users/hadley/repos"))
}
```

---

|                |                          |
|----------------|--------------------------|
| read_json_file | <i>Convert JSON to R</i> |
|----------------|--------------------------|

---

**Description**

Convert JSON to R

**Usage**

```
read_json_file(filename, opts = list(), ...)
```

**Arguments**

|          |                                                                                                                                                                       |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename | full path to text file containing JSON.                                                                                                                               |
| opts     | Named list of options for parsing. Usually created by <code>opts_read_json()</code>                                                                                   |
| ...      | Other named options can be used to override any options in <code>opts</code> . The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |

**Value**

R object

**See Also**

Other JSON Parsers: [read\\_json\\_conn\(\)](#), [read\\_json\\_raw\(\)](#), [read\\_json\\_str\(\)](#), [read\\_ndjson\\_file\(\)](#), [read\\_ndjson\\_raw\(\)](#), [read\\_ndjson\\_str\(\)](#)

**Examples**

```
tmp <- tempfile()
write_json_file(head(iris, 3), tmp)
read_json_file(tmp)
```

---

read\_json\_raw

---

*Convert JSON in a raw vector to R*


---

**Description**

Convert JSON in a raw vector to R

**Usage**

```
read_json_raw(raw_vec, opts = list(), ...)
```

**Arguments**

|         |                                                                                                                                                         |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| raw_vec | raw vector                                                                                                                                              |
| opts    | Named list of options for parsing. Usually created by <a href="#">opts_read_json()</a>                                                                  |
| ...     | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |

**Value**

R object

**See Also**

Other JSON Parsers: [read\\_json\\_conn\(\)](#), [read\\_json\\_file\(\)](#), [read\\_json\\_str\(\)](#), [read\\_ndjson\\_file\(\)](#), [read\\_ndjson\\_raw\(\)](#), [read\\_ndjson\\_str\(\)](#)

**Examples**

```
raw_str <- as.raw(utf8ToInt('[1, 2, 3, "four"]'))
read_json_raw(raw_str)
```

---

|               |                                                |
|---------------|------------------------------------------------|
| read_json_str | <i>Convert JSON in a character string to R</i> |
|---------------|------------------------------------------------|

---

**Description**

Convert JSON in a character string to R

**Usage**

```
read_json_str(str, opts = list(), ...)
```

**Arguments**

|      |                                                                                                                                                                       |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| str  | a single character string                                                                                                                                             |
| opts | Named list of options for parsing. Usually created by <code>opts_read_json()</code>                                                                                   |
| ...  | Other named options can be used to override any options in <code>opts</code> . The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |

**Value**

R object

**See Also**

Other JSON Parsers: [read\\_json\\_conn\(\)](#), [read\\_json\\_file\(\)](#), [read\\_json\\_raw\(\)](#), [read\\_ndjson\\_file\(\)](#), [read\\_ndjson\\_raw\(\)](#), [read\\_ndjson\\_str\(\)](#)

**Examples**

```
read_json_str("4294967297", opts = opts_read_json(int64 = 'string'))
```

---

|                  |                                                     |
|------------------|-----------------------------------------------------|
| read_ndjson_file | <i>Parse an NDJSON file to a data.frame or list</i> |
|------------------|-----------------------------------------------------|

---

**Description**

If reading as `data.frame`, each row of NDJSON becomes a row in the `data.frame`. If reading as a list, then each row becomes an element in the list.

**Usage**

```
read_ndjson_file(
  filename,
  type = c("df", "list"),
  nread = -1,
  nskip = 0,
  nprobe = 100,
  opts = list(),
  ...
)
```

**Arguments**

|          |                                                                                                                                                         |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename | Path to file containing NDJSON data. May be a vanilla text file or a gzipped file                                                                       |
| type     | The type of R object the JSON should be parsed into. Valid values are 'df' or 'list'. Default: 'df' (data.frame)                                        |
| nread    | Number of records to read. Default: -1 (reads all JSON strings)                                                                                         |
| nskip    | Number of records to skip before starting to read. Default: 0 (skip no data)                                                                            |
| nprobe   | Number of lines to read to determine types for data.frame columns. Default: 100. Use -1 to probe entire file.                                           |
| opts     | Named list of options for parsing. Usually created by <code>opts_read_json()</code>                                                                     |
| ...      | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |

**Details**

If parsing NDJSON to a data.frame it is usually better if the json objects are consistent from line-to-line. Type inference for the data.frame is done during initialisation by reading through nprobe lines. Warning: if there is a type-mismatch further into the file than it is probed, then you will get missing values in the data.frame, or JSON values not captured in the R data.

No flattening of the namespace is done i.e. nested object remain nested.

**Value**

NDJSON data read into R as list or data.frame depending on 'type' argument

**See Also**

Other JSON Parsers: [read\\_json\\_conn\(\)](#), [read\\_json\\_file\(\)](#), [read\\_json\\_raw\(\)](#), [read\\_json\\_str\(\)](#), [read\\_ndjson\\_raw\(\)](#), [read\\_ndjson\\_str\(\)](#)

**Examples**

```
tmp <- tempfile()
write_ndjson_file(head(mtcars), tmp)
read_ndjson_file(tmp)
```

---

|                 |                                                                    |
|-----------------|--------------------------------------------------------------------|
| read_ndjson_raw | <i>Parse an NDJSON within a raw vector to a data.frame or list</i> |
|-----------------|--------------------------------------------------------------------|

---

## Description

If reading as data.frame, each row of NDJSON becomes a row in the data.frame. If reading as a list, then each row becomes an element in the list.

## Usage

```
read_ndjson_raw(
  x,
  type = c("df", "list"),
  nread = -1,
  nskip = 0,
  nprobe = 100,
  opts = list(),
  ...
)
```

## Arguments

|        |                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| x      | string containing NDJSON                                                                                                                                |
| type   | The type of R object the JSON should be parsed into. Valid values are 'df' or 'list'. Default: 'df' (data.frame)                                        |
| nread  | Number of records to read. Default: -1 (reads all JSON strings)                                                                                         |
| nskip  | Number of records to skip before starting to read. Default: 0 (skip no data)                                                                            |
| nprobe | Number of lines to read to determine types for data.frame columns. Default: 100. Use -1 to probe entire file.                                           |
| opts   | Named list of options for parsing. Usually created by <code>opts_read_json()</code>                                                                     |
| ...    | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |

## Details

If parsing NDJSON to a data.frame it is usually better if the json objects are consistent from line-to-line. Type inference for the data.frame is done during initialisation by reading through nprobe lines. Warning: if there is a type-mismatch further into the file than it is probed, then you will get missing values in the data.frame, or JSON values not captured in the R data.

No flattening of the namespace is done i.e. nested object remain nested.

## Value

NDJSON data read into R as list or data.frame depending on 'type' argument

**See Also**

Other JSON Parsers: [read\\_json\\_conn\(\)](#), [read\\_json\\_file\(\)](#), [read\\_json\\_raw\(\)](#), [read\\_json\\_str\(\)](#), [read\\_ndjson\\_file\(\)](#), [read\\_ndjson\\_str\(\)](#)

**Examples**

```
js <- write_ndjson_raw(head(mtcars))
js
read_ndjson_raw(js, 'list')
read_ndjson_raw(js, 'df')
```

---

read\_ndjson\_str

---

*Parse an NDJSON string to a data.frame or list*


---

**Description**

If reading as data.frame, each row of NDJSON becomes a row in the data.frame. If reading as a list, then each row becomes an element in the list.

**Usage**

```
read_ndjson_str(
  x,
  type = c("df", "list"),
  nread = -1,
  nskip = 0,
  nprobe = 100,
  opts = list(),
  ...
)
```

**Arguments**

|        |                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| x      | string containing NDJSON                                                                                                                                |
| type   | The type of R object the JSON should be parsed into. Valid values are 'df' or 'list'. Default: 'df' (data.frame)                                        |
| nread  | Number of records to read. Default: -1 (reads all JSON strings)                                                                                         |
| nskip  | Number of records to skip before starting to read. Default: 0 (skip no data)                                                                            |
| nprobe | Number of lines to read to determine types for data.frame columns. Default: 100. Use -1 to probe entire file.                                           |
| opts   | Named list of options for parsing. Usually created by <a href="#">opts_read_json()</a>                                                                  |
| ...    | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |

**Details**

If parsing NDJSON to a data.frame it is usually better if the json objects are consistent from line-to-line. Type inference for the data.frame is done during initialisation by reading through nprobe lines. Warning: if there is a type-mismatch further into the file than it is probed, then you will get missing values in the data.frame, or JSON values not captured in the R data.

No flattening of the namespace is done i.e. nested object remain nested.

**Value**

NDJSON data read into R as list or data.frame depending on 'type' argument

**See Also**

Other JSON Parsers: [read\\_json\\_conn\(\)](#), [read\\_json\\_file\(\)](#), [read\\_json\\_raw\(\)](#), [read\\_json\\_str\(\)](#), [read\\_ndjson\\_file\(\)](#), [read\\_ndjson\\_raw\(\)](#)

**Examples**

```
tmp <- tempfile()
json <- write_ndjson_str(head(mtcars))
read_ndjson_str(json, type = 'list')
```

---

|                    |                                        |
|--------------------|----------------------------------------|
| validate_json_file | <i>Validate JSON in file or string</i> |
|--------------------|----------------------------------------|

---

**Description**

Validate JSON in file or string

**Usage**

```
validate_json_file(filename, verbose = FALSE, opts = list(), ...)
```

```
validate_json_str(str, verbose = FALSE, opts = list(), ...)
```

**Arguments**

|          |                                                                                                                                                         |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| filename | path to file containing JSON                                                                                                                            |
| verbose  | logical. If the JSON is not valid, should a warning be shown giving details?                                                                            |
| opts     | Named list of options for parsing. Usually created by <a href="#">opts_read_json()</a>                                                                  |
| ...      | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_read_json()</a> |
| str      | character string containing JSON                                                                                                                        |

**Value**

Logical value. TRUE if JSON validates as OK, otherwise FALSE

**Examples**

```
tmp <- tempfile()
write_json_file(head(iris, 3), tmp)
validate_json_file(tmp)
str <- write_json_str(iris)
validate_json_str(str)
```

---

|                   |                                   |
|-------------------|-----------------------------------|
| write_geojson_str | <i>Write SF to GeoJSON string</i> |
|-------------------|-----------------------------------|

---

**Description**

Coordinate reference system is always WGS84 in accordance with GeoJSON RFC.

**Usage**

```
write_geojson_str(x, opts = list(), ..., json_opts = list())

write_geojson_file(x, filename, opts = list(), ..., json_opts = list())
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x         | sf object. Supports sf or sfc                                                                                                                                                                                                                                                                                    |
| opts      | named list of options. Usually created with <code>opts_write_geojson()</code> . Default: empty <code>list()</code> to use the default options.                                                                                                                                                                   |
| ...       | any extra named options override those in <code>opts</code>                                                                                                                                                                                                                                                      |
| json_opts | Named list of vanilla JSON options as used by <code>write_json_str()</code> . This is usually created with <code>opts_write_json()</code> . Default value is an empty <code>list()</code> which means to use all the default JSON writing options which is usually the correct thing to do when writing GeoJSON. |
| filename  | filename                                                                                                                                                                                                                                                                                                         |

**Value**

Character string containing GeoJSON, or NULL if GeoJSON written to file.

**Examples**

```
geojson_file <- system.file("geojson-example.json", package = 'yyjsonr')
sf <- read_geojson_file(geojson_file)
cat(write_geojson_str(sf, json_opts = opts_write_json(pretty = TRUE)))
```

---

|                 |                                      |
|-----------------|--------------------------------------|
| write_json_file | <i>Convert R object to JSON file</i> |
|-----------------|--------------------------------------|

---

**Description**

Convert R object to JSON file

**Usage**

```
write_json_file(x, filename, opts = list(), ...)
```

**Arguments**

|          |                                                                                                                                                          |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| x        | the object to be encoded                                                                                                                                 |
| filename | filename                                                                                                                                                 |
| opts     | Named list of serialization options. Usually created by <a href="#">opts_write_json()</a>                                                                |
| ...      | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_write_json()</a> |

**Value**

None

**See Also**

Other JSON Serializer: [write\\_json\\_raw\(\)](#), [write\\_json\\_str\(\)](#), [write\\_ndjson\\_file\(\)](#), [write\\_ndjson\\_raw\(\)](#), [write\\_ndjson\\_str\(\)](#)

**Examples**

```
tmp <- tempfile()
write_json_file(head(iris, 3), tmp)
read_json_file(tmp)
```

---

|                |                                                      |
|----------------|------------------------------------------------------|
| write_json_raw | <i>Convert R object to a raw vector of JSON data</i> |
|----------------|------------------------------------------------------|

---

**Description**

Convert R object to a raw vector of JSON data

**Usage**

```
write_json_raw(x, opts = list(), ...)
```

**Arguments**

|      |                                                                                                                                                          |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | the object to be encoded                                                                                                                                 |
| opts | Named list of serialization options. Usually created by <a href="#">opts_write_json()</a>                                                                |
| ...  | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_write_json()</a> |

**Value**

Raw vector containing the JSON

**See Also**

Other JSON Serializer: [write\\_json\\_file\(\)](#), [write\\_json\\_str\(\)](#), [write\\_ndjson\\_file\(\)](#), [write\\_ndjson\\_raw\(\)](#), [write\\_ndjson\\_str\(\)](#)

**Examples**

```
js <- write_json_raw(head(iris, 3))
js
read_json_raw(js)
```

---

|                |                                        |
|----------------|----------------------------------------|
| write_json_str | <i>Convert R object to JSON string</i> |
|----------------|----------------------------------------|

---

**Description**

Convert R object to JSON string

**Usage**

```
write_json_str(x, opts = list(), ...)
```

**Arguments**

|      |                                                                                                                                                          |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | the object to be encoded                                                                                                                                 |
| opts | Named list of serialization options. Usually created by <a href="#">opts_write_json()</a>                                                                |
| ...  | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_write_json()</a> |

**Value**

Single string containing JSON

**See Also**

Other JSON Serializer: [write\\_json\\_file\(\)](#), [write\\_json\\_raw\(\)](#), [write\\_ndjson\\_file\(\)](#), [write\\_ndjson\\_raw\(\)](#), [write\\_ndjson\\_str\(\)](#)

## Examples

```
write_json_str(head(iris, 3), pretty = TRUE)
```

---

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| write_ndjson_file | <i>Write list or data.frame object to NDJSON in a file</i> |
|-------------------|------------------------------------------------------------|

---

## Description

For list input, each element of the list is written as a single JSON string. For data.frame input, each row of the data.frame is written as aJSON string.

## Usage

```
write_ndjson_file(x, filename, opts = list(), ...)
```

## Arguments

|          |                                                                                                                                                          |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| x        | data.frame or list to be written as multiple JSON strings                                                                                                |
| filename | JSON strings will be written to this file one-line-per-JSON string.                                                                                      |
| opts     | Named list of serialization options. Usually created by <a href="#">opts_write_json()</a>                                                                |
| ...      | Other named options can be used to override any options in opts. The valid named options are identical to arguments to <a href="#">opts_write_json()</a> |

## Value

None

## See Also

Other JSON Serializer: [write\\_json\\_file\(\)](#), [write\\_json\\_raw\(\)](#), [write\\_json\\_str\(\)](#), [write\\_ndjson\\_raw\(\)](#), [write\\_ndjson\\_str\(\)](#)

## Examples

```
tmp <- tempfile()
write_ndjson_file(head(mtcars), tmp)
read_ndjson_file(tmp)
```

---

|                  |                                                                  |
|------------------|------------------------------------------------------------------|
| write_ndjson_raw | <i>Write list or data.frame object to NDJSON in a raw vector</i> |
|------------------|------------------------------------------------------------------|

---

### Description

For `list` input, each element of the list is written as a single JSON string. For `data.frame` input, each row of the `data.frame` is written as a JSON string.

### Usage

```
write_ndjson_raw(x, opts = list(), ...)
```

### Arguments

|                   |                                                                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>    | <code>data.frame</code> or <code>list</code> to be written as multiple JSON strings                                                                                    |
| <code>opts</code> | Named list of serialization options. Usually created by <a href="#">opts_write_json()</a>                                                                              |
| <code>...</code>  | Other named options can be used to override any options in <code>opts</code> . The valid named options are identical to arguments to <a href="#">opts_write_json()</a> |

### Value

Raw vector containing NDJSON character data

### See Also

Other JSON Serializer: [write\\_json\\_file\(\)](#), [write\\_json\\_raw\(\)](#), [write\\_json\\_str\(\)](#), [write\\_ndjson\\_file\(\)](#), [write\\_ndjson\\_str\(\)](#)

### Examples

```
js <- write_ndjson_raw(head(mtcars))
js
read_ndjson_raw(js, 'list')
```

---

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| write_ndjson_str | <i>Write list or data.frame object to NDJSON in a string</i> |
|------------------|--------------------------------------------------------------|

---

### Description

For `list` input, each element of the list is written as a single JSON string. For `data.frame` input, each row of the `data.frame` is written as a JSON string.

### Usage

```
write_ndjson_str(x, opts = list(), ...)
```

**Arguments**

|                   |                                                                                                                                                                     |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>    | data.frame or list to be written as multiple JSON strings                                                                                                           |
| <code>opts</code> | Named list of serialization options. Usually created by <code>opts_write_json()</code>                                                                              |
| <code>...</code>  | Other named options can be used to override any options in <code>opts</code> . The valid named options are identical to arguments to <code>opts_write_json()</code> |

**Value**

String containing multiple JSON strings separated by newlines.

**See Also**

Other JSON Serializer: `write_json_file()`, `write_json_raw()`, `write_json_str()`, `write_ndjson_file()`, `write_ndjson_raw()`

**Examples**

```
write_ndjson_str(head(mtcars))
```

---

|                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| <code>yyjson_read_flag</code> | <i>Advanced: Values for setting internal options directly on YYJSON library</i> |
|-------------------------------|---------------------------------------------------------------------------------|

---

**Description**

This is a list of integer values used for setting flags on the yyjson code directly. This is an ADVANCED option and should be used with caution.

**Usage**

```
yyjson_read_flag
```

**Format**

An object of class list of length 9.

**Details**

Some of these settings overlap and conflict with code needed to handle the translation of JSON values to R.

```
opts_read_json(yyjson_read_flag = c(yyjson_read_flag$x, yyjson_read_flag$y, ...))
```

**YYJSON\_READ\_NOFLAG** Default option (RFC 8259 compliant):

- Read positive integer as `uint64_t`.
- Read negative integer as `int64_t`.
- Read floating-point number as double with round-to-nearest mode.

- Read integer which cannot fit in `uint64_t` or `int64_t` as double.
- Report error if double number is infinity.
- Report error if string contains invalid UTF-8 character or BOM.
- Report error on trailing commas, comments, `inf` and `nan` literals.

**YYJSON\_READ\_INSITU** Read the input data in-situ. This option allows the reader to modify and use input data to store string values, which can increase reading speed slightly. The caller should hold the input data before free the document. The input data must be padded by at least `YYJSON_PADDING_SIZE` bytes. For example: `"[1,2]"` should be `"[1,2]\0\0\0\0"`, input length should be 5.

**YYJSON\_READ\_STOP\_WHEN\_DONE** Stop when done instead of issuing an error if there's additional content after a JSON document. This option may be used to parse small pieces of JSON in larger data, such as `"NDJSON"`

**YYJSON\_READ\_ALLOW\_TRAILING\_COMMAS** Allow single trailing comma at the end of an object or array, such as `"[1, 2, 3, ]"`

**YYJSON\_READ\_ALLOW\_COMMENTS** Allow C-style single line and multiple line comments (non-standard).

**YYJSON\_READ\_ALLOW\_INF\_AND\_NAN** Allow `inf/nan` number and literal, case-insensitive, such as `1e999`, `NaN`, `inf`, `-Infinity` (non-standard).

**YYJSON\_READ\_NUMBER\_AS\_RAW** Read all numbers as raw strings (value with `"YYJSON_TYPE_RAW"` type), `inf/nan` literal is also read as raw with `"ALLOW_INF_AND_NAN"` flag.

**YYJSON\_READ\_ALLOW\_INVALID\_UNICODE** Allow reading invalid unicode when parsing string values (non-standard). Invalid characters will be allowed to appear in the string values, but invalid escape sequences will still be reported as errors. This flag does not affect the performance of correctly encoded strings. **WARNING:** Strings in JSON values may contain incorrect encoding when this option is used, you need to handle these strings carefully to avoid security risks.

**YYJSON\_READ\_BIGNUM\_AS\_RAW** Read big numbers as raw strings. These big numbers include integers that cannot be represented by `"int64_t"` and `"uint64_t"`, and floating-point numbers that cannot be represented by finite `"double"`. The flag will be overridden by `"YYJSON_READ_NUMBER_AS_RAW"` flag.

## Examples

```
read_json_str(
    '[12.3]',
    opts = opts_read_json(yyjson_read_flag = yyjson_read_flag$YYJSON_READ_ALLOW_TRAILING_COMMAS)
)
```

---

|                             |                                             |
|-----------------------------|---------------------------------------------|
| <code>yyjson_version</code> | <i>Version number of 'yyjson' C library</i> |
|-----------------------------|---------------------------------------------|

---

## Description

Version number of 'yyjson' C library

**Usage**

```
yyjson_version()
```

**Value**

Version of included yyjson C library as a string

**Examples**

```
yyjson_version()
```

---

|                   |                                                                                 |
|-------------------|---------------------------------------------------------------------------------|
| yyjson_write_flag | <i>Advanced: Values for setting internal options directly on YYJSON library</i> |
|-------------------|---------------------------------------------------------------------------------|

---

**Description**

This is a list of integer values used for setting flags on the yyjson code directly. This is an ADVANCED option and should be used with caution.

**Usage**

```
yyjson_write_flag
```

**Format**

An object of class list of length 9.

**Details**

Some of these settings overlap and conflict with code needed to handle the translation of JSON values to R.

```
opts_write_json(yyjson_write_flag = c(write_flag$x, write_flag$y, ...))
```

**YYJSON\_WRITE\_NOFLAG** Default value.

- Write JSON minify.
- Report error on inf or nan number.
- Report error on invalid UTF-8 string.
- Do not escape unicode or slash.

**YYJSON\_WRITE\_PRETTY** Write JSON pretty with 4 space indent.

**YYJSON\_WRITE\_ESCAPE\_UNICODE** Escape unicode as uXXXX, make the output ASCII only.

**YYJSON\_WRITE\_ESCAPE\_SLASHES** Escape '/' as '\V'.

**YYJSON\_WRITE\_ALLOW\_INF\_AND\_NAN** Write inf and nan number as 'Infinity' and 'NaN' literal (non-standard).

**YYJSON\_WRITE\_INF\_AND\_NAN\_AS\_NULL** Write inf and nan number as null literal. This flag will override YYJSON\_WRITE\_ALLOW\_INF\_AND\_NAN flag.

**YYJSON\_WRITE\_ALLOW\_INVALID\_UNICODE** Allow invalid unicode when encoding string values (non-standard). Invalid characters in string value will be copied byte by byte. If YYJSON\_WRITE\_ESCAPE\_UNICODE flag is also set, invalid character will be escaped as U+FFFD (replacement character). This flag does not affect the performance of correctly encoded strings.

**YYJSON\_WRITE\_PRETTY\_TWO\_SPACES** Write JSON pretty with 2 space indent. This flag will override YYJSON\_WRITE\_PRETTY flag.

**YYJSON\_WRITE\_NEWLINE\_AT\_END** Adds a newline character at the end of the JSON. This can be helpful for text editors or NDJSON

### Examples

```
write_json_str("hello/there", opts = opts_write_json(  
  yyjson_write_flag = yyjson_write_flag$YYJSON_WRITE_ESCAPE_SLASHES  
))
```

# Index

## \* JSON Parsers

- [read\\_json\\_conn](#), 8
- [read\\_json\\_file](#), 9
- [read\\_json\\_raw](#), 10
- [read\\_json\\_str](#), 11
- [read\\_ndjson\\_file](#), 11
- [read\\_ndjson\\_raw](#), 13
- [read\\_ndjson\\_str](#), 14

## \* JSON Serializer

- [write\\_json\\_file](#), 17
- [write\\_json\\_raw](#), 17
- [write\\_json\\_str](#), 18
- [write\\_ndjson\\_file](#), 19
- [write\\_ndjson\\_raw](#), 20
- [write\\_ndjson\\_str](#), 20

## \* datasets

- [yyjson\\_read\\_flag](#), 21
- [yyjson\\_write\\_flag](#), 23

[as\\_scalar](#), 2

- [opts\\_read\\_geojson](#), 3
- [opts\\_read\\_json](#), 4
- [opts\\_read\\_json\(\)](#), 9–15
- [opts\\_write\\_geojson](#), 5
- [opts\\_write\\_json](#), 6
- [opts\\_write\\_json\(\)](#), 17–21

- [read\\_geojson\\_file\(read\\_geojson\\_str\)](#), 8
- [read\\_geojson\\_str](#), 8
- [read\\_json\\_conn](#), 8, 10–12, 14, 15
- [read\\_json\\_file](#), 9, 9, 10–12, 14, 15
- [read\\_json\\_raw](#), 9, 10, 10, 11, 12, 14, 15
- [read\\_json\\_str](#), 9, 10, 11, 12, 14, 15
- [read\\_ndjson\\_file](#), 9–11, 11, 14, 15
- [read\\_ndjson\\_raw](#), 9–12, 13, 15
- [read\\_ndjson\\_str](#), 9–12, 14, 14

- [validate\\_json\\_file](#), 15
- [validate\\_json\\_str\(validate\\_json\\_file\)](#), 15

- [write\\_geojson\\_file\(write\\_geojson\\_str\)](#), 16
- [write\\_geojson\\_str](#), 16
- [write\\_json\\_file](#), 17, 18–21
- [write\\_json\\_raw](#), 17, 17, 18–21
- [write\\_json\\_str](#), 17, 18, 18, 19–21
- [write\\_ndjson\\_file](#), 17, 18, 19, 20, 21
- [write\\_ndjson\\_raw](#), 17–19, 20, 21
- [write\\_ndjson\\_str](#), 17–20, 20
- [yyjson\\_read\\_flag](#), 21
- [yyjson\\_read\\_flag\(\)](#), 5
- [yyjson\\_version](#), 22
- [yyjson\\_write\\_flag](#), 23
- [yyjson\\_write\\_flag\(\)](#), 7