# Package 'rjsoncons'

October 14, 2022

**Title** 'C++' Header-Only 'jsoncons' Library for 'JSON' Queries

**Version** 1.0.0

**Description** The 'jsoncons' <https://danielaparker.github.io/jsoncons/> 'C++' header-only library constructs representations from a 'JSON' character vector, and provides extensions for flexible queries and other operations on 'JSON' objects. This package has simple 'R' wrappers to support 'JSONpath' and 'JMESpath' queries into 'JSON' strings or 'R' objects. The 'jsoncons' library is also be easily linked to other packages for direct access to 'C++' functionality.

**Imports** jsonlite

**Suggests** tinytest, BiocStyle, knitr, rmarkdown

**License** BSL-1.0

**LinkingTo** cpp11

**NeedsCompilation** yes

**SystemRequirements** C++11

**Encoding** UTF-8

**BugReports** https://github.com/mtmorgan/rjsoncons/issues

**RoxygenNote** 7.2.1

**VignetteBuilder** knitr

**Author** Martin Morgan [aut, cre] (<https://orcid.org/0000-0002-5874-8148>), Marcel Ramos [aut] (<https://orcid.org/0000-0002-3242-0582>), Daniel Parker [aut, cph] (jsoncons C++ library maintainer)

**Maintainer** Martin Morgan <mtmorgan.bioc@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-09-29 08:20:05 UTC

## R topics documented:

version *Query the jsoncons C++ library*

## Description

`version()` reports the version of the C++ jsoncons library in use.

`jsonpath()` executes a query against a json string using the 'jsonpath' specification

`jmespath()` executes a query against a json string using the 'jmespath' specification.

## Usage

```
version()

jsonpath(data, path, object_names = "asis", ...)

jmespath(data, path, object_names = "asis", ...)
```

## Arguments

| | |
|---|---|
| `data` | an *R* object. If `data` is a scalar (length 1) character vector, it is treated as a single JSON string. Otherwise, it is parsed to a JSON string using `jsonlite::toJSON()`. Use `I()` to treat a scalar character vector as an *R* object rather than JSON string, e.g., `I("A")` will be parsed to `["A"]` before processing. |
| `path` | character(1) jsonpath or jmespath query string. |
| `object_names` | character(1) order `data` object elements `"asis"` (default) or `"sort"` before filtering on `path`. |
| `...` | arguments passed to `jsonlite::toJSON` when `data` is not a scalar character vector. For example, use `auto_unbox = TRUE` to automatically 'unbox' vectors of length 1 to JSON scalar values. |

## Value

`version()` returns a character(1) major.minor.patch version string .

`jsonpath()` returns a character(1) json string representing the result of the query.

`jmespath()` return a character(1) json string representing the result of the query.

## Examples

```
version()

json <- '{
  "locations": [
    {"name": "Seattle", "state": "WA"},
    {"name": "New York", "state": "NY"},
    {"name": "Bellevue", "state": "WA"},
    {"name": "Olympia", "state": "WA"}
```

```
  ]
 }'

jsonpath(json, "$..name") |>
    cat("\n")

## create a list with state and name as scalar vectors
lst <- jsonlite::fromJSON(json, simplifyVector = FALSE)

## objects other than scalar character vectors are automatically
## coerced to JSON; use `auto_unbox = TRUE` to represent R scalar
## vectors in the object as JSON scalar vectors
jsonpath(lst, "$..name", auto_unbox = TRUE) |>
    cat("\n")

## a scalar character vector like "Seattle" is not valid JSON...
try(jsonpath("Seattle", "$[0]"))

## use I("Seattle") to coerce to a JSON object ["Seattle"]
jsonpath(I("Seattle"), "$[0]")      |> cat("\n")

## different ordering of object names -- 'asis' (default) or 'sort'
json_obj <- '{"b": "1", "a": "2"}'
jsonpath(json_obj, "$")             |> cat("\n")
jsonpath(json_obj, "$.*")           |> cat("\n")
jsonpath(json_obj, "$", "sort")   |> cat("\n")
jsonpath(json_obj, "$.*", "sort") |> cat("\n")

path <- "locations[?state == 'WA'].name | sort(@)"
jmespath(json, path) |>
    cat("\n")

## original filter always fails, e.g., '["WA"] != 'WA'
jmespath(lst, path)  # empty result set, '[]'

## filter with unboxed state, and return unboxed name
jmespath(lst, "locations[?state[0] == 'WA'].name[0] | sort(@)") |>
    cat("\n")

## automatically unbox scalar values when creating the JSON string
jmespath(lst, path, auto_unbox = TRUE) |>
    cat("\n")
```

# Index