

Package: yaml12 (via r-universe)

September 25, 2026

Title Fast 'YAML' 1.2 Parser and Formatter

Version 0.2.0.9000

Description A fast, correct, safe, and ergonomic 'YAML' 1.2 parser and generator written in 'Rust'. Convert between 'YAML' and simple 'R' objects with full support for multi-document streams, tags, anchors, and aliases. Offers opt-in handlers for custom tag behavior and round-trips common 'R' data structures. Implements the 'YAML' 1.2.2 specification from the 'YAML' Language Development Team (2021) <<https://yaml.org/spec/1.2.2/>>. Proudly supported by Posit.

License MIT + file LICENSE

URL <https://posit-dev.github.io/r-yaml12/>,
<https://github.com/posit-dev/r-yaml12>

BugReports <https://github.com/posit-dev/r-yaml12/issues>

Depends R (>= 4.2)

Suggests jsonlite, knitr, rmarkdown, testthat (>= 3.0.0), waldo, withr

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate, rmarkdown, bench,
ggplot2, ggbeeswarm, dplyr, tidyr, yaml

Config/testthat/edition 3

Config/testthat/parallel false

Encoding UTF-8

Roxygen list(markdown = TRUE)

SystemRequirements Cargo (Rust's package manager), rustc >= 1.71.0,
xz. On Windows ARM64, source installs also require Microsoft
C++ Build Tools with ARM64 components.

Config/roxygen2/version 8.1.0

Config/pak/sysreqs xz-utils libclang-dev

Repository <https://posit-dev.r-universe.dev>

Date/Publication 2026-08-26 15:55:44 UTC

RemoteUrl <https://github.com/posit-dev/r-yaml12>

RemoteRef HEAD

RemoteSha 4c37a8b6cab48cb6f8d85ef51dd79316db36eda2

Contents

format_yaml	2
parse_yaml	3
Index	5

format_yaml	<i>Format or write R objects as YAML 1.2.</i>
-------------	---

Description

format_yaml() returns YAML as a character string. write_yaml() writes a YAML stream to a file or stdout and starts every document with a document start (---) marker. Both functions honor a yaml_tag attribute on values (see examples).

Long single-line strings and multiline strings containing unindented, single-line paragraphs separated by exactly one blank line are automatically wrapped when a line would be wider than width columns and has a safe word boundary. Wrapped strings use a YAML folded block scalar: >- preserves no final newline, while > preserves exactly one. Paragraph breaks and all other string content round-trip through parse_yaml() unchanged. Strings that cannot be folded losslessly use a literal or quoted representation, and mapping keys are never wrapped.

Literal blocks use explicit indentation indicators when needed to preserve leading spaces or tabs. They keep physical blank lines empty and preserve later-indented lines.

Usage

```
format_yaml(value, multi = FALSE, width = 80L)
```

```
write_yaml(value, path = NULL, multi = FALSE, append = FALSE, width = 80L)
```

Arguments

value	Any R object composed of lists, atomic vectors, and scalars.
multi	When TRUE, treat value as a list of YAML documents and encode a stream.
width	Target maximum line width in columns. Long single-line strings and unindented paragraphs with safe word boundaries are wrapped. Individual lines may still exceed width when there is no safe break point (e.g. a single long word) or under deep indentation. Use NULL or Inf to disable wrapping.
path	Scalar string file path to write YAML to when using write_yaml(). Tilde prefixes (~) are expanded as by <code>base::path.expand()</code> . When NULL (the default), write to R's standard output connection.
append	When TRUE, append to path instead of replacing it. Defaults to FALSE.

Value

`format_yaml()` returns a scalar character string containing YAML. `write_yaml()` invisibly returns value.

Examples

```
cat(format_yaml(list(foo = 1, bar = list(TRUE, NA))))

docs <- list("first", "second")
cat(format_yaml(docs, multi = TRUE))

tagged <- structure("1 + 1", yaml_tag = "!expr")
cat(tagged_yaml <- format_yaml(tagged), "\n")

dput(parse_yaml(tagged_yaml))

write_yaml(list(foo = 1, bar = list(2, "baz")))

write_yaml(list("foo", "bar"), multi = TRUE)

tagged <- structure("1 + 1", yaml_tag = "!expr")
write_yaml(tagged)
```

 parse_yaml

Parse YAML 1.2 document(s) into base R structures.

Description

`parse_yaml()` takes strings of YAML; `read_yaml()` reads from a file path.

Usage

```
parse_yaml(text, multi = FALSE, simplify = TRUE, handlers = NULL)

read_yaml(path, multi = FALSE, simplify = TRUE, handlers = NULL)
```

Arguments

<code>text</code>	Character vector; elements are concatenated with <code>"\n"</code> .
<code>multi</code>	When <code>TRUE</code> , return a list containing all documents in the stream.
<code>simplify</code>	When <code>FALSE</code> , keep YAML sequences as R lists instead of simplifying to atomic vectors.
<code>handlers</code>	Named list of R functions with names corresponding to YAML tags; matching handlers transform tagged values.
<code>path</code>	Scalar string path to a YAML file. Tilde prefixes (<code>~</code>) are expanded as by <code>base::path.expand()</code> .

Details

YAML tags without a corresponding handler are preserved in a `yaml_tag` attribute. Mappings with keys that are not all simple scalar strings are returned as a named list with a `yaml_keys` attribute.

Value

When `multi = FALSE`, returns a parsed R object for the first document. When `multi = TRUE`, returns a list of parsed documents.

Duplicate mapping keys

YAML 1.2 requires mapping keys to be unique, but `parse_yaml()` and `read_yaml()` do not currently reject duplicates. When a key is repeated with the same spelling and quoting style, only its last value is retained. Keys written differently, such as `key` and `"key"`, may remain as separate entries even when they produce the same R name. A future release may warn or error for these cases, so this behavior should not be relied upon.

Examples

```
dput(parse_yaml("foo: [1, 2, 3]"))

# homogeneous sequences simplify by default.
# YAML null maps to NA in otherwise homogeneous sequences.
dput(parse_yaml("foo: [1, 2, 3, null]"))

# mixed type sequence never simplify
dput(parse_yaml("[1, true, cat]"))

# use `simplify=FALSE` to always return sequences as lists.
str(parse_yaml("foo: [1, 2, 3, null]", simplify = FALSE))

# Parse multiple documents when requested.
stream <- "
---
first: 1
---
second: 2
"
str(parse_yaml(stream, multi = TRUE))

# Read from a file; keep sequences as lists.
path <- tempfile(fileext = ".yaml")
writeLines("alpha: [true, null]\nbeta: 3.5", path)
str(read_yaml(path, simplify = FALSE))
```

Index

`base::path.expand()`, [2](#), [3](#)

`format_yaml`, [2](#)

`parse_yaml`, [3](#)

`read_yaml (parse_yaml)`, [3](#)

`write_yaml (format_yaml)`, [2](#)