

Package ‘yaml’

November 12, 2016

Type Package

Title Methods to Convert R Data to YAML and Back

Version 2.1.14

Suggests testthat

Date 2016-11-12

Author Jeremy Stephens <jeremy.stephens@vanderbilt.edu>

Maintainer Jeremy Stephens <jeremy.stephens@vanderbilt.edu>

License BSD_3_clause + file LICENSE

Description Implements the 'libyaml' 'YAML' 1.1 parser and emitter (<<http://pyyaml.org/wiki/LibYAML>>) for R.

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-11-12 19:55:17

R topics documented:

as.yaml	1
yaml.load	3

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

as.yaml	<i>Convert an R object into a YAML string</i>
---------	---

Description

Convert an R object into a YAML string

Usage

```
as.yaml(x, line.sep = c("\n", "\r\n", "\r"), indent = 2, omap = FALSE,  
        column.major = TRUE, unicode = TRUE, precision = getOption('digits'),  
        indent.mapping.sequence = FALSE)
```

Arguments

x	the object to be converted
line.sep	the line separator character(s) to use
indent	the number of spaces to use for indenting
omap	determines whether or not to convert a list to a YAML omap; see Details
column.major	determines how to convert a data.frame; see Details
unicode	determines whether or not to allow unescaped unicode characters in output
precision	number of significant digits to use when formatting numeric values
indent.mapping.sequence	determines whether or not to indent sequences in mapping context

Details

If you set the `omap` option to `TRUE`, `as.yaml` will create ordered maps (or omaps) instead of normal maps.

The `column.major` option determines how a data frame is converted. If `TRUE`, the data frame is converted into a map of sequences where the name of each column is a key. If `FALSE`, the data frame is converted into a sequence of maps, where each element in the sequence is a row. You'll probably almost always want to leave this as `TRUE` (which is the default), because using `yaml.load` on the resulting string returns an object which is much more easily converted into a data frame via `as.data.frame`.

Value

Returns a YAML string which can be loaded using `yaml.load` or copied into a file for external use.

Author(s)

Jeremy Stephens <jeremy.stephens@vanderbilt.edu>

References

YAML: <http://yaml.org>

YAML omap type: <http://yaml.org/type/omap.html>

See Also

[yaml.load](#)

Examples

```
as.yaml(1:10)
as.yaml(list(foo=1:10, bar=c("test1", "test2")))
as.yaml(list(foo=1:10, bar=c("test1", "test2")), indent=3)
as.yaml(list(foo=1:10, bar=c("test1", "test2")), indent.mapping.sequence=TRUE)
as.yaml(data.frame(a=1:10, b=letters[1:10], c=11:20))
as.yaml(list(a=1:2, b=3:4), omap=TRUE)
```

```
as.yaml("multi\nline\nstring")
as.yaml(function(x) x + 1)
as.yaml(list(foo=list(list(x = 1, y = 2), list(x = 3, y = 4))))
```

yml.load

Convert a YAML string into R objects

Description

Parse a YAML string and return R objects.

Usage

```
yml.load(string, as.named.list = TRUE, handlers = NULL)
yml.load_file(input, ...)
```

Arguments

string	the YAML string to be parsed
as.named.list	whether or not to return a named list for maps (TRUE by default)
handlers	named list of custom handler functions for YAML types (see Details).
input	a filename or connection
...	arguments to pass to yml.load

Details

Use `yml.load` to load a YAML string. For files and connections, use `yml.load_file`, which calls `yml.load` with the contents of the specified file or connection.

Sequences of uniform data (i.e. a sequence of integers) are converted into vectors. If the sequence is not uniform, it's returned as a list. Maps are converted into named lists by default, and all the keys in the map are converted to strings. If you don't want the keys to be coerced into strings, set `as.named.list` to `FALSE`. When it's `FALSE`, a list will be returned with an additional attribute named 'keys', which is a list of the un-coerced keys in the map (in the same order as the main list).

You can specify custom handler functions via the `handlers` argument. This argument must be a named list of functions, where the names are the YAML types (i.e., 'int', 'float', 'seq', etc). The functions you provide will be passed one argument. Custom handler functions for string types (all types except sequence and map) will receive a character vector of length 1. Custom sequence functions will be passed a list of objects. Custom map functions will be passed the object that the internal map handler creates, which is either a named list or a list with a 'keys' attribute (depending on `as.named.list`). ALL functions you provide must return an object. See the examples for custom handler use.

This function uses the YAML parser provided by `libyaml`, which conforms to the YAML 1.1 specification.

Value

If the root YAML object is a map, a named list or list with an attribute of 'keys' is returned. If the root object is a sequence, a list or vector is returned, depending on the contents of the sequence. A vector of length 1 is returned for single objects.

Author(s)

Jeremy Stephens <jeremy.stephens@vanderbilt.edu>

References

YAML: <http://yaml.org>

libyaml: <http://pyyaml.org/wiki/LibYAML>

See Also

[as.yaml](#)

Examples

```
yaml.load("- hey\n- hi\n- hello")
yaml.load("foo: 123\nbar: 456")
yaml.load("- foo\n- bar\n- 3.14")
yaml.load("foo: bar\n123: 456", as.named.list = FALSE)

## Not run:
# reading from a file (uses readLines internally)
cat("foo: 123", file="foo.yml", sep="\n")
yaml.load_file('foo.yml')
unlink("foo.yml") # tidy up

## End(Not run)

# custom scalar handler
my.float.handler <- function(x) { as.numeric(x) + 123 }
yaml.load("123.456", handlers=list("float#fix"=my.float.handler))

# custom sequence handler
yaml.load("- 1\n- 2\n- 3", handlers=list(seq=function(x) { as.integer(x) + 3 })))

# custom map handler
yaml.load("foo: 123", handlers=list(map=function(x) { x$foo <- x$foo + 123; x })))

# handling custom types
yaml.load("!sqrt 555", handlers=list(sqrt=function(x) { sqrt(as.integer(x)) })))
yaml.load("!foo\n- 1\n- 2", handlers=list(foo=function(x) { as.integer(x) + 1 })))
yaml.load("!bar\nnone: 1\ntwo: 2", handlers=list(foo=function(x) { x$one <- "one"; x })))

# loading R expressions
doc <- yaml.load("inc: !expr function(x) x + 1")
doc$inc(1)
```

Index

*Topic **data**

as.yaml, 1

yaml.load, 3

*Topic **manip**

as.yaml, 1

yaml.load, 3

*Topic **programming**

yaml.load, 3

as.data.frame, 2

as.yaml, 1, 4

yaml.load, 2, 3

yaml.load_file(yaml.load), 3