Package 'schemate'

June 12, 2026

Title Schema Inference, Editing, and Validation with 'checkmate'

Version 0.1.0

Author Hongyuan Jia [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-0075-8183>)

Maintainer Hongyuan Jia <hongyuanjia@cqust.edu.cn>

Description Provides a compact schema domain-specific language for inferring, editing, and validating R data structures with 'checkmate' checks. Schemas can be serialized to and restored from JSON for storage and review. A generated standalone bundle supports vendoring the schema tools into other R packages.

License MIT + file LICENSE

Encoding UTF-8

URL https://hongyuanjia.github.io/schemate/

BugReports https://github.com/hongyuanjia/schemate/issues

RoxygenNote 7.3.3

Imports checkmate (>= 2.0.0), S7

Suggests jsonlite, testthat (>= 3.0.0)

Collate 'schema-utils.R' 'schema-spec.R' 'schema-doc.R' 'schema-compact.R' 'schema-edit.R' 'schema-flat.R' 'schema-infer.R' 'schema-json.R' 'schema-query.R' 'schema-validate.R' 'schemate-package.R' 'zzz.R'

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-06-12 10:00:02 UTC

Contents

`schema_add_def` *Add a schema definition*

Description

Add a schema definition

Usage

```
schema_add_def(x, name, value, overwrite = FALSE)
```

Arguments

`x`	A `SchemaDoc`.
`name`	Definition name to add.
`value`	Schema fragment using the same list syntax accepted by `schema_doc()`, or a fragment produced by helpers such as `schema_check()`, to store in `$defs`.
`overwrite`	Logical flag indicating whether an existing definition of the same name should be replaced.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(schema_check("string"))
schema <- schema_add_def(schema, "text", schema_check("string"))
schema

names(as.list(schema)$`$defs`)
```

---

schema_add_field *Add a field schema to a container node*

Description

Add a field schema to a container node

Usage

```
schema_add_field(x, name, field, path = "$", overwrite = FALSE)
```

Arguments

x	A SchemaDoc.
name	Field name to add.
field	Schema fragment using the same list syntax accepted by schema_doc(), or a fragment produced by helpers such as schema_check().
path	Path to the target container node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
overwrite	Logical flag indicating whether an existing field of the same name should be replaced.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(check = list(kind = "list")))
schema
schema <- schema_add_field(schema, "id", schema_check("int", lower = 1))
schema

schema_validate(schema, list(id = 1L), mode = "test")
```

schema_add_group *Add a schema group to a container node*

Description

Add a schema group to a container node

Usage

```
schema_add_group(x, group, path = "$")
```

Arguments

x	A SchemaDoc.
group	Schema group fragment using the same list syntax accepted by schema_doc(), or a fragment produced by schema_group().
path	Path to the target container node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(check = list(kind = "list")))
schema
schema <- schema_add_group(schema, schema_group(c("x", "y"), schema_check("number")))
schema

schema_validate(schema, list(x = 1, y = 2), mode = "test")
```

schema_add_position *Add a position schema to an unnamed container node*

Description

Add a position schema to an unnamed container node

Usage

```
schema_add_position(x, index, value, path = "$")
```

Arguments

x	A SchemaDoc.
index	1-based insertion index. 1 inserts at the front and length(positions) + 1 appends.
value	Schema fragment using the same list syntax accepted by schema_doc(), or a fragment produced by helpers such as schema_check().
path	Path to the target unnamed container node. Use $ for the root node.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(
    check = list(kind = "list"),
    keys = list(type = "unnamed")
))
schema <- schema_add_position(schema, 1, schema_check("string"))
schema <- schema_add_position(schema, 2, schema_check("int"))
schema

schema_validate(schema, list("a", 1L), mode = "test")
```

---

schema_all *Create an* all *schema combinator fragment*

Description

Create an all schema combinator fragment

Usage

```
schema_all(..., description = NULL)
```

Arguments

...	Branch schema fragments.
description	Optional node description.

Value

A raw schema fragment accepted by schema_doc() and schema edit verbs.

Examples

```
schema <- schema_doc(schema_all(
    schema_check("string"),
    schema_check("string", min.chars = 1)
))
schema

schema_validate(schema, "ok", mode = "test")
```

---

schema_any *Create an* any *schema combinator fragment*

Description

Create an any schema combinator fragment

Usage

```
schema_any(..., description = NULL)
```

Arguments

`...`	Branch schema fragments.
`description`	Optional node description.

Value

A raw schema fragment accepted by `schema_doc()` and schema edit verbs.

Examples

```
schema <- schema_doc(schema_any(schema_check("int"), schema_check("string")))
schema

schema_validate(schema, "ok", mode = "test")
```

schema_check *Create a schema check fragment*

Description

schema_check() creates a raw schema fragment with a check operator. The helper performs only lightweight structural validation; semantic validation of kind and check arguments is handled by schema_doc() and schema edit verbs.

Usage

```
schema_check(kind, ..., description = NULL)
```

Arguments

kind	Check kind string.
...	Additional named checkmate arguments stored inside check.
description	Optional node description.

Value

A raw schema fragment accepted by schema_doc() and schema edit verbs.

Examples

```
schema_check("string", min.chars = 1)
schema <- schema_doc(schema_check("string", min.chars = 1))
schema
```

schema_compact *Compact a schema document*

Description

schema_compact() simplifies schema documents produced by inference or hand-authoring. It can merge observed array element alternatives and group sibling fields that share identical schemas.

Usage

```
schema_compact(x, arrays = TRUE, groups = TRUE)
```

Arguments

x	A SchemaDoc or raw schema DSL list.
arrays	Whether to merge compatible any branches, especially the observed element alternatives produced by schema_infer(arrays = "rest").
groups	Whether to combine sibling fields with identical schemas into groups.

Value

A compacted SchemaDoc.

Examples

```
schema <- schema_infer(
    list(items = list(list(id = 1L, name = "a"), list(id = 2L, label = "b"))),
    keys = "named",
    arrays = "rest"
)
schema

schema_compact(schema)
```

---

schema_del_def *Delete a schema definition*

Description

Delete a schema definition

Usage

```
schema_del_def(x, name, missing = "error")
```

Arguments

x	A SchemaDoc.
name	Definition name to remove.
missing	Missing-target behavior. Use "error" to raise an error or "ignore" to leave the schema unchanged.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(schema_check("string"))
schema <- schema_add_def(schema, "text", schema_check("string"))
schema <- schema_del_def(schema, "text")
schema

as.list(schema)$`$defs`
```

---

schema_del_field *Delete a field schema from a container node*

Description

Delete a field schema from a container node

Usage

```
schema_del_field(x, name, path = "$", missing = "error")
```

Arguments

x	A SchemaDoc.
name	Field name to remove.
path	Path to the target container node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
missing	Missing-target behavior. Use "error" to raise an error or "ignore" to leave the schema unchanged.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(check = list(kind = "list")))
schema
schema <- schema_add_field(schema, "id", schema_check("int"))
schema
schema <- schema_del_field(schema, "id")
schema
```

schema_del_group *Delete a schema group from a container node*

Description

Delete a schema group from a container node

Usage

```
schema_del_group(x, index, path = "$", missing = "error")
```

Arguments

x	A SchemaDoc.
index	1-based group index to remove.
path	Path to the target container node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
missing	Missing-target behavior. Use "error" to raise an error or "ignore" to leave the schema unchanged.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(
    check = list(kind = "list"),
    groups = list(schema_group(c("x", "y"), schema_check("number")))
))
schema

schema_del_group(schema, 1)
```

schema_del_keys *Delete a schema node keys rule*

Description

Delete a schema node keys rule

Usage

```
schema_del_keys(x, path = "$", missing = "error")
```

Arguments

x	A SchemaDoc.
path	Path to the target schema node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
missing	Missing-target behavior. Use "error" to raise an error or "ignore" to leave the schema unchanged.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(check = list(kind = "list"), keys = list(type = "named")))
schema <- schema_del_keys(schema)
schema

as.list(schema)$keys
```

---

schema_del_position *Delete a position schema from an unnamed container node*

Description

Delete a position schema from an unnamed container node

Usage

```
schema_del_position(x, index, path = "$", missing = "error")
```

Arguments

x	A SchemaDoc.
index	1-based position index to remove.
path	Path to the target unnamed container node. Use $ for the root node.
missing	Missing-target behavior. Use "error" to raise an error or "ignore" to leave the schema unchanged.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(check = list(kind = "list"), keys = list(type = "unnamed")))
schema <- schema_add_position(schema, 1, schema_check("string"))
schema <- schema_del_position(schema, 1)
schema

as.list(schema)$positions
```

---

schema_del_rest *Delete a container rest schema*

Description

Delete a container rest schema

Usage

```
schema_del_rest(x, path = "$", missing = "error")
```

Arguments

x	A SchemaDoc.
path	Path to the target container node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
missing	Missing-target behavior. Use "error" to raise an error or "ignore" to leave the schema unchanged.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(check = list(kind = "list")))
schema <- schema_set_rest(schema, schema_check("string"))
schema <- schema_del_rest(schema)
schema

as.list(schema)$rest
```

schema_doc *Parse schema documents*

Description

schema_doc() parses a schema DSL list into a schemate schema document object.

Usage

```
schema_doc(x, path = NULL)
```

Arguments

x	A schema DSL list or an existing schemate schema document.
path	Optional source path stored as runtime metadata.

Details

Normal users usually create schema documents with schema_infer(), schema_read(), or the edit helpers. Use schema_doc() when you are hand-authoring a schema as an R list.

Value

A schemate schema document object.

Examples

```
doc <- schema_doc(list(check = list(kind = "string", min.chars = 1)))
doc

schema_validate(doc, "ok")
schema_validate(doc, 1L, mode = "check")
```

schema_group *Create a schema group fragment*

Description

Create a schema group fragment

Usage

```
schema_group(names, value, description = NULL)
```

Arguments

names	Field names covered by the group.
value	Schema node fragment containing exactly one primary operator.
description	Optional group description.

Value

A raw schema group fragment accepted in a schema document groups list or by schema_add_group().

Examples

```
schema <- schema_doc(list(
    check = list(kind = "list"),
    groups = list(schema_group(c("x", "y"), schema_check("number")))
))
schema

schema_validate(schema, list(x = 1, y = 2), mode = "test")
```

---

schema_infer *Infer a conservative schema from example data*

Description

schema_infer() builds a SchemaDoc from example data using conservative, structural inference only. It infers base/container check kinds and nested field structure, but does not guess higher-level authoring constructs such as $defs, $ref, keys, groups, or combinators.

Usage

```
schema_infer(
  x,
  version = NULL,
  keys = c("none", "named", "required", "exact"),
  arrays = c("none", "rest")
)
```

Arguments

x	Example data to infer from.
version	Optional schema document version string.
keys	Strategy for inferring optional keys rules from observed names. Use "none" to skip names-rule inference, "named" to require named inputs, "required" to require the observed names to be present, or "exact" to require the observed names in the observed order.

arrays	Strategy for inferring unnamed lists. Use "none" to keep unnamed lists generic, or "rest" to infer them as unnamed containers whose observed element schemas are stored in rest.

Details

To parse an existing schema DSL document, use schema_doc() or schema_read() instead.

Value

A SchemaDoc inferred from x.

Examples

```
payload <- list(items = list(list(id = 1L), list(id = 2L)))
schema_infer(payload, keys = "named", arrays = "rest")
```

---

schema_modify_where *Modify schema nodes selected by a predicate*

Description

schema_modify_where() modifies every schema node matched by where. schema_replace_where() is a convenience wrapper that replaces all matched nodes with the same schema fragment.

Usage

```
schema_modify_where(x, where, fn, defs = TRUE, missing = "ignore")

schema_replace_where(x, where, value, defs = TRUE, missing = "ignore")
```

Arguments

x	A schema document or raw schema DSL list.
where	Predicate function with signature function(path, node).
fn	Function with signature function(path, node) returning a schema fragment or SchemaNode.
defs	Whether to include root $defs entries.
missing	Missing-match behavior. Use "error" to raise an error when where matches no paths or "ignore" to leave the schema unchanged.
value	Replacement schema fragment or SchemaNode.

Details

Batch edits operate on logical paths. Editing paths inside a grouped schema field may split the group into per-field bindings. If where matches both a node and one of its descendants in the same call, the edit errors and asks you to narrow the selector.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_compact(schema_infer(list(
    issued = list(`date-parts` = list(list(2024L))),
    created = list(`date-parts` = list(list(2024L)))
), arrays = "rest"))
schema <- schema_add_def(schema, "year", schema_check("int", lower = 0))
schema

schema_find(schema, schema_where_path("(^|\\$)`date-parts`\\$rest$"))

schema <- schema_replace_where(
    schema,
    schema_where_path("(^|\\$)`date-parts`\\$rest$"),
    schema_ref("year")
)
schema
```

---

schema_not *Create a* not *schema combinator fragment*

Description

Create a not schema combinator fragment

Usage

```
schema_not(branch, description = NULL)
```

Arguments

branch	Branch schema fragment.
description	Optional node description.

Value

A raw schema fragment accepted by schema_doc() and schema edit verbs.

Examples

```
schema <- schema_doc(schema_not(schema_check("null")))
schema

schema_validate(schema, "ok", mode = "test")
```

schema_one *Create a* one *schema combinator fragment*

Description

Create a one schema combinator fragment

Usage

```
schema_one(..., description = NULL)
```

Arguments

`...`	Branch schema fragments.
`description`	Optional node description.

Value

A raw schema fragment accepted by `schema_doc()` and schema edit verbs.

Examples

```
schema <- schema_doc(schema_one(schema_check("int"), schema_check("string")))
schema

schema_validate(schema, "ok", mode = "test")
```

schema_paths *Query schema paths and matching nodes*

Description

`schema_paths()` lists editable logical schema paths. `schema_find()` returns paths whose schema node satisfies a predicate.

Usage

```
schema_paths(x, defs = TRUE)

schema_find(x, where, defs = TRUE)
```

Arguments

`x`	A schema document or raw schema DSL list.
`defs`	Whether to include root `$defs` entries.
`where`	Predicate function with signature `function(path, node)`.

Details

Logical paths describe fields as users see them in the validated data. Grouped fields are expanded to one path per field.

Value

A character vector of schema paths.

Examples

```
schema <- schema_compact(schema_infer(list(
    issued = list(`date-parts` = list(list(2024L))),
    created = list(`date-parts` = list(list(2024L)))
), arrays = "rest"))
schema

schema_find(schema, schema_where_path("(^|\\$)`date-parts`\\$rest$"))
schema_find(schema, schema_where_check("int"))
```

---

schema_ref *Create a schema reference fragment*

Description

`schema_ref()` creates a local `$defs` reference fragment. `name` may be either a bare definition name such as `"text"` or a local ref string of the form `"#/$defs/text"`.

Usage

```
schema_ref(name, description = NULL)
```

Arguments

`name`	Definition name or local `$defs` ref string.
`description`	Optional node description.

Value

A raw schema fragment accepted by `schema_doc()` and schema edit verbs.

Examples

```
schema <- schema_doc(list(
    `$defs` = list(text = schema_check("string")),
    `$ref` = "#/$defs/text"
))
schema

schema_validate(schema, "ok", mode = "test")
schema_ref("text")
```

---

schema_replace *Replace a schema node*

Description

Replace a schema node

Usage

```
schema_replace(x, path = "$", value)
```

Arguments

x	A SchemaDoc.
path	Path to the target schema node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
value	Replacement schema fragment using the same list syntax accepted by schema_doc(), or a fragment produced by helpers such as schema_check() or schema_ref().

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(
    check = list(kind = "list"),
    fields = list(id = schema_check("int"))
))
schema <- schema_replace(schema, "$id", schema_check("int", lower = 1))
schema

schema_validate(schema, list(id = 1L), mode = "test")
```

schema_set_desc *Set or remove a schema node description*

Description

Set or remove a schema node description

Usage

```
schema_set_desc(x, path = "$", description = NULL)
```

Arguments

x	A SchemaDoc.
path	Path to the target schema node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
description	Optional description string. Use NULL to remove the description.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(schema_check("string"))
schema_set_desc(schema, "$", "A non-empty label.")
```

schema_set_keys *Set a schema node keys rule*

Description

Set a schema node keys rule

Usage

```
schema_set_keys(x, path = "$", ...)
```

Arguments

x	A SchemaDoc.
path	Path to the target schema node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .
...	Named keys rule arguments passed through to the schema DSL.

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(check = list(kind = "list")))
schema
schema_validate(schema, list(id = 1L), mode = "test")

schema <- schema_set_keys(schema, type = "named", must.include = "id")
schema
schema_validate(schema, list(id = 1L), mode = "assert")
```

---

schema_set_rest *Set or replace a container rest schema*

Description

Set or replace a container rest schema

Usage

```
schema_set_rest(x, field, path = "$")
```

Arguments

x	A SchemaDoc.
field	Schema fragment using the same list syntax accepted by schema_doc(), or a fragment produced by helpers such as schema_check(), to store as the rest schema.
path	Path to the target container node. Use $ for the root node. Bare field segments such as $id implicitly traverse container fields. Use $fields$id to write the explicit field path. Backtick-quote field names that contain path operators, for example $`a$b` .

Value

A modified SchemaDoc.

Examples

```
schema <- schema_doc(list(
    check = list(kind = "list"),
    keys = list(type = "unnamed")
))
schema <- schema_set_rest(schema, schema_check("string"))
schema

schema_validate(schema, list("a", "b"), mode = "test")
```

---

schema_validate *Validate input against a schema*

Description

schema_validate() validates an R object against a SchemaDoc, SchemaFlat, or compiled flat schema node.

Usage

```
schema_validate(schema, x, mode = "assert", name = NULL, ...)
```

Arguments

schema	A SchemaDoc, SchemaFlat, or compiled flat schema node.
x	Input object to validate.
mode	One of "assert", "check", "test", or "expect".
name	Optional display name used in validation messages.
...	Reserved for future extension.

Value

In "assert" mode, invisibly returns x or throws an error. In "check" mode, returns TRUE or a diagnostic string. In "test" mode, returns TRUE or FALSE. In "expect" mode, returns a testthat-style expectation object.

Examples

```
schema <- schema_doc(list(
    check = list(kind = "list"),
    fields = list(id = list(check = list(kind = "int", lower = 1)))
))
schema

schema_validate(schema, list(id = 1L), mode = "test")
schema_validate(schema, list(id = 0L), mode = "check", name = "payload")
```

`schema_where_path` *Create schema query predicates*

Description

Create schema query predicates

Usage

```
schema_where_path(pattern, fixed = FALSE)

schema_where_check(kind = NULL)
```

Arguments

`pattern`	Pattern passed to `grepl()` for matching schema paths.
`fixed`	Whether `pattern` should be matched literally.
`kind`	Optional check kind to match, such as `"list"` or `"int"`.

Value

A predicate function with signature `function(path, node)`.

Examples

```
by_path <- schema_where_path("(^|\\$)`date-parts`\\$rest$")
by_int <- schema_where_check("int")

schema <- schema_infer(list(id = 1L))
schema

schema_find(schema, by_int)
```

schema_write *Read and write schema JSON*

Description

schema_read() reads schema JSON into a SchemaDoc. schema_write() serializes schema objects to schema JSON. Both functions require the suggested package jsonlite.

Usage

```
schema_write(x, path, overwrite = FALSE, pretty = TRUE, auto_unbox = TRUE)

schema_read(txt)
```

Arguments

x	A schema object accepted by as.list(), usually a SchemaDoc.
path	Output file path.
overwrite	Whether an existing output file may be overwritten.
pretty	Whether JSON output should be pretty-printed.
auto_unbox	Passed to jsonlite::write_json().
txt	JSON text, local file path, or URL accepted by jsonlite::fromJSON().

Value

schema_read() returns a SchemaDoc. schema_write() invisibly returns path.

Examples

```
schema <- schema_infer(list(id = 1L))
schema

path <- tempfile(fileext = ".json")
schema_write(schema, path)

schema_read(path)
```

Index