Package 'validR6'

January 12, 2026

Type Package

Title Validation framework for members in 'R6::R6Class'

Version 1.0

Date 2025-09-19

Author Simon Lenau

Maintainer Simon Lenau <lenau@cispa.de>

Description Validation framework to ensure that members in 'R6::R6Class' objects match formalized validation criteria, e.g. in term of type, structure or content.

License GPL (>= 3)

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE,r6=TRUE)

Encoding UTF-8

Depends R (>= 4.3.0)

Imports R6,
knitr

Suggests rmarkdown,
testthat (>= 3.0.0)

 # VignetteBuilder knitr

Config/testthat/edition 3

Collate 'internals-.capture.R'
'internals-doc_topic*.R'
'internals-documentation.R'
'class-R6Member.R'
'class-R6MemberList.R'
'internals-DOC_DECLARE_REF.R'
'internals-.doc_ref*.R'
'internals-documentation-types.R'
'class-R6MemberRestriction.R'
'internals-.check_if.R'
'internals-.cut_string.R'
'internals-.doc_arg*.R'
'internals-.doc_data*.R'
'internals-.doc_in_current_package.R'
'internals-.doc_make_link.R'
'internals-.doc_pad.R'

'internals-.doc_tbl*.R'
'internals-documentation-default_from_class.R'
'internals-documentation-formatting.R'
'internals-documentation-phrases.R'
'internals-documentation-tagging.R'
'internals-documentation-values.R'
'internals-indent.R'
'methods-S3-R6MemberList.R'
'validR6_package.R'

VignetteBuilder knitr

Contents

`validR6-package` *The* `validR6` *package*

Description

The validR6-package provides a validation framework for members (fields, methods and active bindings) in `R6Classes`. This ensures that members meet validation criteria, e.g. in terms of their type, structure or content. Validation happens *before* assignment and is reusable across `R6Classes`.

Details

To declare validation rules for `R6Classes` members', this package revolves around the three cooperating `R6Classes`: `R6MemberRestriction`, `R6Member` and `R6MemberList`. These are outlined in the following.

R6MemberRestriction:

`R6MemberRestriction` objects are wrappers for user-defined validation rules, standardizing how rules are executed and errors are captured.

R6Member:

`R6Member` objects encapsulate exactly one member (field, method or active binding) of `R6Classes` and store the member's current value. They may contain an optional set of `R6MemberRestrictions`, which are used for validating new values before assignment via the `R6MemberRestriction$check()` method. The use of a `R6Member` object as field, method, or active binding inside `R6Classes` is simplified through its helper methods.

R6MemberList:

`R6MemberList` objects are lightweight wrappers around any number of `R6Member` instances. They provide helper methods to build fully validated `R6Classes` by generating the corresponding `public`, `private` and `active` `lists` used as arguments in `R6::R6Class()`.

Typical workflow

1. Define one or more `R6MemberRestriction` objects with validation rules.
2. Define one `R6Member` object for each member in a `R6Class` to be defined and specify the `R6MemberRestrictions` this member must satisfy.
3. Combine the `R6Members` in a `R6MemberList` and call `R6MemberList$R6Class()` to build a fully validated `R6Classes`.

This keeps validation logic decoupled, reusable, and testable, while letting `R6` remain concise.

Key benefits

- **Consistency**: uniform validation and error reporting across members and classes.
- **Reusability**: compose simple rules into complex constraints.
- **Safety**: validation occurs before assignment, preventing invalid state.
- **Convenience**: helpers generate `R6Classes`, minimizing boilerplate.

`R6Member` *Container for members in* `R6Class` *definitions.*

Description

A container that stores name, value and (optional) validity restrictions of type `R6MemberRestriction` for members to be used in `R6Class` declarations.

A `R6Member` object provides

- a setter that checks restrictions
- a getter returning the value
- helper methods for using it as member (attribute or method) in `R6Classes`

See the sections on details, examples and methods.

Details

`R6Member` objects provide a formalized interface for setting up members of `R6Classes` and provide methods for automatically generating the corresponding `R6Class()` arguments `public`, `private` and `active`.

Validation rules for `$value` :

Validation rules for the `$value` field are specified in the `$restrictions` field.

`$restrictions` contains a list of zero or more `R6MemberRestrictions` describing the validation rules for the `$value` field. Any new `value` is passed to the `R6MemberRestriction$check()` method of all elements in `$restrictions` before `R6Member$value` is changed. This ensures all validation criteria are met.

Combining multiple `R6Members` and declaring `R6Classes` :

For declaring `R6Classes`, `R6Members` are most conveniently combined in a `R6MemberList`.

Active bindings

`is_method`

`is_private`

`locked`

`name`

`private_name`

`restrictions`

`value`

`value_locked`

Methods

Public methods:

- `R6Member$accessor()`
- `R6Member$new()`
- `R6Member$print()`
- `R6Member$declare_active_binding()`
- `R6Member$declare_method()`
- `R6Member$declare_initialization()`
- `R6Member$clone()`

Method `accessor()`:

Usage:

```
R6Member$accessor(field_value, field_name)
```

Arguments:

`field_value`

`field_name`

Details: Access or modify one of the internal fields of `R6Members`

Returns:

If `field_value` is **not** provided (getter case):	the field's current value – see active bindings for the fields' types.
If `field_value` is provided (setter case):	`NULL`

Method `new()`:

Usage:

```
R6Member$new(
  name,
  value = NULL,
  restrictions = NULL,
  locked = TRUE,
  value_locked = FALSE,
  is_private = FALSE,
  is_method = NULL
)
```

Arguments:

name

value

restrictions

locked

value_locked

is_private

is_method

Details: Construct a new R6Member object

Returns: R6Member object

Method print()**:**

Usage:

```
R6Member$print(value_level = TRUE, restriction_level = 0, print_class = TRUE)
```

Arguments:

value_level

restriction_level

print_class

Details: Print an R6Member object.

Returns: NULL

Method declare_active_binding()**:**

Usage:

```
R6Member$declare_active_binding()
```

Details: Create active binding function for R6Member object. *To be used when declaring* R6Classes, see also the R6MemberList$R6Class() method and ?R6Class.

Returns: function to be used in the active argument of R6Class().

Method declare_method()**:**

Usage:

```
R6Member$declare_method()
```

Details: Create method function for R6Member object. *To be used when declaring* R6Classes, see also the R6MemberList$R6Class() method and ?R6Class.

Returns: function to be used in the method argument of R6Class().

Method declare_initialization()**:**

Usage:

```
R6Member$declare_initialization()
```

Details: Create (initial) assignment expression for R6Member object. *To be used when declaring* R6Classes, see also the R6MemberList$R6Class() method and ?R6Class.

Returns: expression containing the assignment statement to be used in the public[["initialize"]] (constructor) method of R6Class().

Method clone()**:** The objects of this class are cloneable with this method.

Usage:

```
R6Member$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

R6MemberList *R6MemberList: Wrapping toolbox for using* R6Members *in* R6Class *definitions.*

Description

A R6MemberList provides utility methods for declaring R6Classes with members wrapped in R6Member objects. As such, the R6MemberList encloses zero or more R6Member objects that define the members in a R6Class.

The core method for this purpose is $R6Class().

Details

Once a R6MemberList is instanciated, the $R6Class() method can be used to define a R6Class and delegate the enclosed R6Member objects as members to this new R6Class.

The core method for this purpose is $R6Class(), which calls the worker methods $public_list(), $private_list() and $active_bindings(). These set up the lists defining the arguments public, private and active for the call to R6::R6Class().

Retrieving private and public names:

Use $names() to retrieve public or private member names of the R6Member objects in the R6Class created by $R6Class().

Subsetting:

The $subset() method allows access to the enclosed R6Member objects by name. Alternatively, the S3-operators [and [[allow subsetting for R6MemberList as well.

See the methods and examples section for more details and examples.

Methods

Public methods:

- R6MemberList$new()
- R6MemberList$active_bindings()
- R6MemberList$R6Class()
- R6MemberList$public_list()
- R6MemberList$private_list()
- R6MemberList$print()
- R6MemberList$names()
- R6MemberList$subset()
- R6MemberList$clone()

Method new(): Construct a new R6MemberList object from the supplied R6Member objects.

Usage:

```
R6MemberList$new(...)
```

Arguments:

...

Returns: R6MemberList object containing the provided R6Member objects.

Method `active_bindings()`**:** Create named `list` that represent the active bindings provided by the stored `R6Members`. To be used as argument '`active`' in `R6Class()` definition. *See also the documentation of argument* '`active`' *in* `?R6Class`.

Usage:

```
R6MemberList$active_bindings()
```

Returns: A named `list` that represent the active bindings provided by the stored `R6Members`. To be used as argument '`active`' in `R6Class()` definition.

Method `R6Class()`**:** Declare `R6Class` from `R6MemberList`

Usage:

```
R6MemberList$R6Class(classname, print_method = "default")
```

Arguments:

`classname`

`print_method`

Returns: A `R6Class` with members defined by the `R6MemberList`

Method `public_list()`**:** Create named `list` that represent the public fields provided by the stored `R6Members`. To be used as argument '`public`' in `R6Class()` definition. *See also the documentation of argument* '`public`' *in* `?R6Class`.

Usage:

```
R6MemberList$public_list(print_method = "default")
```

Arguments:

`print_method`

Returns: A named `list` that represent the public fields provided by the stored `R6Members`. To be used as argument '`public`' in `R6Class()` definition.

Method `private_list()`**:** Create named `list` that represent the private fields provided by the stored `R6Members`. To be used as argument '`private`' in `R6Class()` definition. *See also the documentation of argument* '`private`' *in* `?R6Class`.

Usage:

```
R6MemberList$private_list()
```

Returns: A named `list` that represent the private fields provided by the stored `R6Members`. To be used as argument '`private`' in `R6Class()` definition.

Method `print()`**:** Print method for `R6MemberList`

Usage:

```
R6MemberList$print()
```

Method `names()`**:** `names` method for `R6MemberList`

Usage:

```
R6MemberList$names(type = "public", subset = "all")
```

Arguments:

`type`

`subset`

Returns: Character vector of names

Method `subset()`**:** subset method for `R6MemberList`

Usage:

```
R6MemberList$subset(member, new_value, inner = TRUE)
```

Arguments:

`member`

`new_value`

`inner`

Returns: subset

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
R6MemberList$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`R6MemberRestriction` *R6MemberRestriction: Validity constraints for* `R6Member` *objects*

Description

A `R6MemberRestriction` encapsulates and manages a single validation rule – a `restriction` – as a reusable object. This validation rule is as a user-supplied `function` to check whether `values` meets a constraint.

A `R6MemberRestriction` handles the execution of this rule, manages `error` reporting, and provides a consistent interface. `R6MemberRestriction` objects are primarily used to validate the `value` attribute of `R6Member` objects, but can be applied to any `value` for which a validation rule needs to be enforced.

Details

A `R6MemberRestriction` is characterized by its `restriction`, a `function` specifying a validation rule for an object. While the validation logic is entirely defined by the `restriction`, the `R6MemberRestriction` class provides the infrastructure for how the rule is invoked and how results are reported. This separation promotes modularity and consistency across validation workflows.

Typical Examples for `Restrictions`:

- `value` must be of a certain type (such as `numeric`, `list` or `function`)
- `value`'s elements *together* must meet a *common* restriction, e.g. in terms of dimensionality (`length(value) == k`) or totals (`sum(value) > t`)
- `value`'s elements *each* must meet an *individual* restriction, e.g. be non-negative (`all(value >= 0)`)

Structure of `Restrictions`:

The `restriction` must be a `function` taking a single argument and returning a single `logical` value to indicate if a restriction is met (`TRUE`) or not (`FALSE`). Alternatively, if validation fails, the `function` may throw an `error` providing detailed information on how or why the `restriction` is violated. The is `restriction` must be able to take arguments of all kinds and shapes. It is tested using a range of sample inputs as its argument; any violation raises a descriptive `error` so users know how to fix their rule.

Validation Workflow:

The R6MemberRestriction is executed via the R6MemberRestriction$check() method, which applies the rule to a provided value and returns a list containing both the validation decision and any error message encountered. This encapsulation allows validation results to be easily captured, logged, or aggregated.

Composition of Rules:

Multiple R6MemberRestriction objects are commonly combined to define complex validation logic, typically for specifying R6Member objects. This compositional design keeps individual rules small, testable, and focused on a single concern, promoting code clarity and maintainability.

See the R6Member examples for practical examples.

Active bindings

id

restriction

Methods

Public methods:

- R6MemberRestriction$check()
- R6MemberRestriction$new()
- R6MemberRestriction$print()
- R6MemberRestriction$clone()

Method check():

Usage:

```
R6MemberRestriction$check(value, include_id = TRUE, include_value = TRUE)
```

Arguments:

value

include_id

include_value

Details: Check whether a given value satisfies the restriction. *This is the core method of* R6MemberRestriction *objects.*

Returns: list with entries

decision	(logical)	indicator whether the value meets (TRUE) or violates (FALSE) the restriction.
message	(character)	error message resulting from restriction (if any), or empty string otherwise

Method new():

Usage:

```
R6MemberRestriction$new(
  id = format(Sys.time(), "%Y-%M-%d - %H:%M:%S"),
  restriction = function(x) TRUE
)
```

Arguments:

id r .doc_tag(tag = "sec-params")

`restriction`

Details: Construct a new `R6MemberRestriction` object with identifier `R6MemberRestriction` and restriction `R6MemberRestriction`

Returns: `R6MemberRestriction` object.

Method `print()`**:**

Usage:

```
R6MemberRestriction$print(restriction_level = 2)
```

Arguments:

`restriction_level`

Details: Print method for `R6MemberRestriction` objects that displays information (`id` and `restriction`).

Returns: `NULL`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
R6MemberRestriction$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index