Package 'restrictR'

March 9, 2026

Title Composable Runtime Contracts for R

Version 0.1.0

Description Build reusable validators from small building blocks using the base pipe operator. Define runtime contracts once with 'restrict()' and enforce them anywhere in code. Validators compose naturally, support dependent rules via formulas, and produce clear, path-aware error messages. No DSL, no operator overloading, just idiomatic R.

License MIT + file LICENSE

Language en-US

Encoding UTF-8

RoxygenNote 7.3.3

URL https://gillescolling.com/restrictR/, https://github.com/gcol33/restrictR

BugReports https://github.com/gcol33/restrictR/issues

Depends R (>= 4.1.0)

Suggests knitr, rmarkdown, svglite, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Gilles Colling [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3070-6066>)

Maintainer Gilles Colling <gilles.colling051@gmail.com>

Repository CRAN

Date/Publication 2026-03-09 11:20:02 UTC

Contents

`as_contract_block` *Convert a Validator to a Multi-Line Block*

Description

Produces a multi-line text summary suitable for roxygen `@details` documentation. Each step appears on its own line as a bullet point.

Usage

```
as_contract_block(x)
```

Arguments

`x`	a `restriction` object.

Value

A character(1) string with one step per line.

See Also

Other core: `as_contract_text()`, `require_custom()`, `restrict()`

Examples

```
v <- restrict("x") |> require_numeric(no_na = TRUE) |> require_length(1L)
as_contract_block(v)
```

`as_contract_text` *Convert a Validator to Plain Text*

Description

Produces a single-line text summary suitable for roxygen `@param` documentation. Use with inline R code in roxygen: `` `r as_contract_text(validator)` ``.

Usage

```
as_contract_text(x)
```

Arguments

`x`	a `restriction` object.

Value

A character(1) string describing the validation contract.

See Also

Other core: `as_contract_block()`, `require_custom()`, `restrict()`

Examples

```
v <- restrict("x") |> require_numeric(no_na = TRUE) |> require_length(1L)
as_contract_text(v)
```

require_between *Require Value in Range*

Description

Validates that all elements of a numeric value fall within a specified range.

Usage

```
require_between(
  restriction,
  lower = -Inf,
  upper = Inf,
  exclusive_lower = FALSE,
  exclusive_upper = FALSE
)
```

Arguments

restriction	a restriction object.
lower	numeric(1) lower bound (default -Inf).
upper	numeric(1) upper bound (default Inf).
exclusive_lower	logical; if TRUE, lower bound is exclusive.
exclusive_upper	logical; if TRUE, upper bound is exclusive.

Value

The modified restriction object.

See Also

Other value checks: require_one_of()

require_character *Require Character Type*

Description

Validates that the value is character. Optionally checks for NA values.

Usage

```
require_character(restriction, no_na = FALSE)
```

Arguments

restriction	a restriction object.
no_na	logical; if TRUE, rejects NA values.

Value

The modified restriction object.

See Also

Other type checks: require_df(), require_integer(), require_logical(), require_numeric()

---

require_col_between *Require Column Values in Range*

Description

Validates that all values in a column fall within a specified range.

Usage

```
require_col_between(
  restriction,
  col,
  lower = -Inf,
  upper = Inf,
  exclusive_lower = FALSE,
  exclusive_upper = FALSE
)
```

Arguments

restriction	a restriction object.
col	character(1) column name.
lower	numeric(1) lower bound (default -Inf).
upper	numeric(1) upper bound (default Inf).
exclusive_lower	logical; if TRUE, lower bound is exclusive.
exclusive_upper	logical; if TRUE, upper bound is exclusive.

Value

The modified restriction object.

See Also

Other column checks: require_col_character(), require_col_numeric(), require_col_one_of()

require_col_character *Require Character Column*

Description

Validates that a specific column in a data.frame is character. Produces path-aware error messages.

Usage

```
require_col_character(restriction, col, no_na = FALSE)
```

Arguments

restriction	a restriction object.
col	character(1) column name.
no_na	logical; if TRUE, rejects NA values in the column.

Value

The modified restriction object.

See Also

Other column checks: require_col_between(), require_col_numeric(), require_col_one_of()

require_col_numeric *Require Numeric Column*

Description

Validates that a specific column in a data.frame is numeric. Produces path-aware error messages (e.g. newdata$x2: must be numeric).

Usage

```
require_col_numeric(restriction, col, no_na = FALSE, finite = FALSE)
```

Arguments

restriction	a restriction object.
col	character(1) column name.
no_na	logical; if TRUE, rejects NA values in the column.
finite	logical; if TRUE, rejects non-finite values in the column.

Value

The modified `restriction` object.

See Also

Other column checks: `require_col_between()`, `require_col_character()`, `require_col_one_of()`

`require_col_one_of` *Require Column Values from a Set*

Description

Validates that all values in a column are among the allowed values.

Usage

```
require_col_one_of(restriction, col, values)
```

Arguments

`restriction`	a `restriction` object.
`col`	character(1) column name.
`values`	vector of allowed values.

Value

The modified `restriction` object.

See Also

Other column checks: `require_col_between()`, `require_col_character()`, `require_col_numeric()`

`require_custom` *Create a Custom Validation Step*

Description

Allows advanced users to define their own validation step without growing the package's built-in API surface. The step function receives `(value, name, ctx)` and should call `testthat::fail()` on validation failure.

Usage

```
require_custom(restriction, label, fn, deps = character(0L))
```

Arguments

restriction	a restriction object.
label	character(1) human-readable description for printing.
fn	a function with signature function(value, name, ctx) that calls stop() or restrictR:::fail() on failure.
deps	character vector of context names this step requires (default: none).

Value

A new restriction object with the custom step appended.

See Also

Other core: as_contract_block(), as_contract_text(), restrict()

Examples

```
# Custom step: require all values to be unique
require_unique_id <- restrict("id") |>
  require_custom(
    label = "must contain unique values",
    fn = function(value, name, ctx) {
      dupes <- which(duplicated(value))
      if (length(dupes) > 0L) {
        stop(sprintf("%s: contains %d duplicate value(s)",
                     name, length(dupes)), call. = FALSE)
      }
    }
  )
```

---

require_df *Require a Data Frame*

Description

Validates that the value is a data.frame.

Usage

```
require_df(restriction)
```

Arguments

restriction	a restriction object.

Value

The modified `restriction` object.

See Also

Other type checks: `require_character()`, `require_integer()`, `require_logical()`, `require_numeric()`

`require_finite` *Require Finite Values*

Description

Validates that a numeric value contains no `Inf`, `-Inf`, or `NaN` values. Does not check for `NA` (use `require_no_na()` for that).

Usage

```
require_finite(restriction)
```

Arguments

`restriction`	a `restriction` object.

Value

The modified `restriction` object.

See Also

Other missingness checks: `require_no_na()`

`require_has_cols` *Require Specific Columns*

Description

Validates that a data.frame contains all specified columns.

Usage

```
require_has_cols(restriction, cols)
```

Arguments

`restriction`	a `restriction` object.
`cols`	character vector of required column names.

Value

The modified `restriction` object.

See Also

Other structure checks: `require_length()`, `require_length_matches()`, `require_length_max()`, `require_length_min()`, `require_nrow_matches()`, `require_nrow_min()`

`require_integer` *Require Integer Type*

Description

Validates that the value is integer (not just numeric with integer values). Optionally checks for NA values.

Usage

```
require_integer(restriction, no_na = FALSE)
```

Arguments

`restriction`	a `restriction` object.
`no_na`	logical; if `TRUE`, rejects NA values.

Value

The modified `restriction` object.

See Also

Other type checks: `require_character()`, `require_df()`, `require_logical()`, `require_numeric()`

`require_length` *Require Specific Length*

Description

Validates that the value has exact length `n`.

Usage

```
require_length(restriction, n)
```

Arguments

restriction	a restriction object.
n	integer(1) required length.

Value

The modified restriction object.

See Also

Other structure checks: require_has_cols(), require_length_matches(), require_length_max(), require_length_min(), require_nrow_matches(), require_nrow_min()

---

require_length_matches

Require Length Matching an Expression

Description

Validates that length(value) equals the result of evaluating a formula. The formula is evaluated using only explicitly passed context arguments, plus .value (the validated value) and .name (the restriction name).

Usage

```
require_length_matches(restriction, formula)
```

Arguments

restriction	a restriction object.
formula	a one-sided formula (e.g. ~ nrow(newdata)).

Value

The modified restriction object.

See Also

Other structure checks: require_has_cols(), require_length(), require_length_max(), require_length_min(), require_nrow_matches(), require_nrow_min()

require_length_max *Require Maximum Length*

Description

Validates that the value has at most length `n`.

Usage

```
require_length_max(restriction, n)
```

Arguments

`restriction`	a `restriction` object.
`n`	integer(1) maximum length.

Value

The modified `restriction` object.

See Also

Other structure checks: `require_has_cols()`, `require_length()`, `require_length_matches()`, `require_length_min()`, `require_nrow_matches()`, `require_nrow_min()`

require_length_min *Require Minimum Length*

Description

Validates that the value has at least length `n`.

Usage

```
require_length_min(restriction, n)
```

Arguments

`restriction`	a `restriction` object.
`n`	integer(1) minimum length.

Value

The modified `restriction` object.

See Also

Other structure checks: require_has_cols(), require_length(), require_length_matches(), require_length_max(), require_nrow_matches(), require_nrow_min()

---

require_logical *Require Logical Type*

Description

Validates that the value is logical. Optionally checks for NA values.

Usage

```
require_logical(restriction, no_na = FALSE)
```

Arguments

restriction	a restriction object.
no_na	logical; if TRUE, rejects NA values.

Value

The modified restriction object.

See Also

Other type checks: require_character(), require_df(), require_integer(), require_numeric()

---

require_no_na *Require No NA Values*

Description

Validates that the value contains no NA values. Works on any atomic type.

Usage

```
require_no_na(restriction)
```

Arguments

restriction	a restriction object.

Value

The modified restriction object.

See Also

Other missingness checks: `require_finite()`

`require_nrow_matches` *Require Row Count Matching an Expression*

Description

Validates that `nrow(value)` equals the result of evaluating a formula. The formula is evaluated using only explicitly passed context arguments, plus `.value` (the validated value) and `.name` (the restriction name).

Usage

```
require_nrow_matches(restriction, formula)
```

Arguments

`restriction`	a `restriction` object.
`formula`	a one-sided formula (e.g. `~ nrow(reference)`).

Value

The modified `restriction` object.

See Also

Other structure checks: `require_has_cols()`, `require_length()`, `require_length_matches()`, `require_length_max()`, `require_length_min()`, `require_nrow_min()`

`require_nrow_min` *Require Minimum Number of Rows*

Description

Validates that a data.frame has at least `n` rows.

Usage

```
require_nrow_min(restriction, n)
```

Arguments

`restriction`	a `restriction` object.
`n`	integer(1) minimum row count.

Value

The modified `restriction` object.

See Also

Other structure checks: `require_has_cols()`, `require_length()`, `require_length_matches()`, `require_length_max()`, `require_length_min()`, `require_nrow_matches()`

`require_numeric` *Require Numeric Type*

Description

Validates that the value is numeric. Optionally checks for NA and non-finite values.

Usage

```
require_numeric(restriction, no_na = FALSE, finite = FALSE)
```

Arguments

`restriction`	a `restriction` object.
`no_na`	logical; if `TRUE`, rejects NA values.
`finite`	logical; if `TRUE`, rejects `Inf`/`-Inf`/`NaN`.

Value

The modified `restriction` object.

See Also

Other type checks: `require_character()`, `require_df()`, `require_integer()`, `require_logical()`

`require_one_of` *Require Value from a Set*

Description

Validates that all elements of the value are among the allowed values.

Usage

```
require_one_of(restriction, values)
```

Arguments

`restriction`	a `restriction` object.
`values`	vector of allowed values.

Value

The modified `restriction` object.

See Also

Other value checks: `require_between()`

`restrict` *Create a Composable Validator*

Description

Creates a callable validation object that accumulates checks via the base pipe operator `|>`. The resulting object behaves like a function: call it with a value to validate.

Usage

```
restrict(name)
```

Arguments

`name`	character(1) name used in error messages (e.g. `"newdata"`).

Value

A `restriction` object (callable function) with no validation steps.

Calling convention

Validators accept `value` as the first argument, plus context via named arguments in `...` or as a named list in `.ctx`:

```
require_pred(out, newdata = df)
require_pred(out, .ctx = list(newdata = df))
```

Named arguments in `...` take precedence over `.ctx` entries with the same name. If a step declares dependencies (e.g. `require_length_matches(~ nrow(newdata))`), the validator checks that all required context is present before running any steps and errors early if not.

See Also

Other core: `as_contract_block()`, `as_contract_text()`, `require_custom()`

Examples

```
# Define a validator
require_positive <- restrict("x") |>
  require_numeric(no_na = TRUE) |>
  require_between(lower = 0, exclusive_lower = TRUE)

# Use it
require_positive(5)   # passes silently

# Compose with pipe
require_score <- restrict("score") |>
  require_numeric() |>
  require_length(1L) |>
  require_between(lower = 0, upper = 100)
```

Index