

Package: data.checker (via r-universe)

June 8, 2026

Title Data Checker for Validating Data Frames Against Defined Schema

Version 2.0.0

Description Validates data frames against a defined schema. Produces a report of the checks performed and any issues found, with index and entry value where appropriate. Backend checks are performed using pointblank Richard Iannone et al (2025)
<[doi:10.32614/CRAN.package.pointblank](https://doi.org/10.32614/CRAN.package.pointblank)>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr, glue, jsonlite, knitr, lubridate, magrittr, tools, yaml, tomledit, utils, cli, rlang, pointblank, tidyselect, tidyr, stringr, hms

Suggests rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://onsdigital.github.io/data.checker/>

NeedsCompilation no

Author Crown Copyright [cph], Analysis Standards and Pipelines Team (ONS) [cre, aut]

Maintainer Analysis Standards and Pipelines Team (ONS)
<ASAP@ons.gov.uk>

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

Date/Publication 2026-06-08 19:40:07 UTC

RemoteUrl <https://github.com/cran/data.checker>

RemoteRef HEAD

RemoteSha c8fa73eece97f86768ef5f49afe9ab2bc936f089

Contents

add_check	2
add_check_custom	3
add_qa_entry	4
check	4
check_and_export	5
check_backseries	7
check_colnames	7
check_column_contents	8
check_completeness	8
check_duplicates	9
check_schema_contents_against_df	9
check_types	10
export	10
export.Validator	11
hard_checks_status	11
iqr_bounds	12
is_column_contents_valid	12
is_type_valid	13
is_valid_column_values	13
is_valid_schema	14
log_html	14
log_pointblank_outcomes	15
log_to_table	15
new_validator	16
print.Validator	17
run_checks	17
types_to_classes	18
validate_and_convert_date_formats	18
z_score	19
Index	20

add_check	<i>Add a custom check to the validator</i>
-----------	--------------------------------------------

Description

This function allows you to add a custom check to the Validator object.

Usage

```
add_check(validator, description, condition)
```

Arguments

validator	A Validator object to which the custom check will be added.
description	A description of the custom check.
condition	Expression to be evaluated or logical conditions to define the custom check. Optional if outcome is set

Value

The updated Validator object with the custom check added.

add_check_custom	<i>Add a custom check to the validator</i>
------------------	--------------------------------------------

Description

This function allows you to add a custom check outcomes to the validator log. The outcomes must be a logical vector.

Usage

```
add_check_custom(
  validator,
  description,
  outcome,
  type = c("error", "warning", "note")
)
```

Arguments

validator	A Validator object to which the custom check will be added.
description	A description of the custom check.
outcome	Logical vector indicating the result of the check (TRUE/FALSE). Outcome must be logical.
type	The type of the check, can be "error", "warning", or "note".

Value

The updated Validator object with the custom check added.

add_qa_entry	<i>Add a QA Entry to the validator's QA Log</i>
--------------	-------------------------------------------------

Description

This function adds a new entry to the validator's QA log with details such as a description, type of entry, timestamp, pass status, and failing IDs.

Usage

```
add_qa_entry(
  validator,
  description,
  failing_ids,
  outcome = NA,
  entry_type = c("info", "warning", "error")
)
```

Arguments

validator	a Validator object.
description	A character string describing the QA entry.
failing_ids	Optional: A vector of IDs that failed the QA check. If more than 10 IDs are provided, only the first 10 are stored, with a note indicating the additional count.
outcome	Optional: A logical value indicating whether the QA check passed. If not provided or invalid, defaults to NA.
entry_type	Optional: A character string specifying the type of entry. Must be one of "info", "warning", or "error". Defaults to "info".

Value

The updated validator object with the new entry appended to its QA log.

check	<i>Validate a Validator Object</i>
-------	------------------------------------

Description

This function runs the full suite of validation checks on a Validator object.

Usage

```
check(validator, ...)
```

Arguments

validator An object of class Validator to be validated.
... Additional arguments (currently unused).

Value

The validated Validator object if all checks pass. If any check fails, an error is thrown.

Examples

```
# create schema
schema <- list(
  check_duplicates = FALSE,
  check_completeness = FALSE,
  columns = list(
    age = list(type = "double", optional = FALSE),
    sex = list(type = "character", optional = FALSE)
  )
)

# create dataframe
df <- data.frame(
  age = c(10, 11, 13, 15, 22, 34, 80),
  sex = c("M", "F", "M", "F", "M", "F", "M")
)

# create validator object
validator <- new_validator(
  data = df,
  schema = schema
)

# validate the data
validator <- check(validator)
```

check_and_export

Validate data against a schema and output results

Description

This function validates data against a given schema, performs checks, and exports the validation results to a specified file in a given format.

Usage

```
check_and_export(
  data,
  schema,
```

```

    file,
    format,
    hard_check = FALSE,
    backseries = NULL,
    name = deparse(substitute(data))
  )

```

Arguments

data	The data to be validated.
schema	The schema to validate against.
file	The file path where the validation results will be exported.
format	The format in which the validation results will be exported.
hard_check	logical. Optional - FALSE by default. If TRUE, raises an error if there are any failed checks. Otherwise, raises a warning.
backseries	A previous version of the data to check against (optional).
name	validator name - defaults to the name of the dataframe object supplied to "data" (Optional). Must be a single character string.

Value

The exported validation results.

Examples

```

# create schema
schema <- list(
  check_duplicates = FALSE,
  check_completeness = FALSE,
  columns = list(
    age = list(type = "double", optional = FALSE),
    sex = list(type = "character", optional = FALSE)
  )
)

# create dataframe
df <- data.frame(
  age = c(10, 11, 13, 15, 22, 34, 80),
  sex = c("M", "F", "M", "F", "M", "F", "M")
)

# validate and export log
check_and_export(
  data = df,
  schema = schema,
  file = paste0(tempdir(), "\\validation_results_example.html"),
  format = "html",
  hard_check = TRUE
)

```

check_backseries	<i>Check backseries consistency</i>
------------------	-------------------------------------

Description

Checks if the latest data is consistent with previous data.

Usage

```
check_backseries(validator)
```

Arguments

validator A Validator object containing the schema and agent.

Value

The updated Validator object with outcomes logged.

check_colnames	<i>Check Column Names against schema</i>
----------------	------------------------------------------

Description

This function performs checks on the column names of a Validator object to ensure they follow specific naming conventions and meet schema conditions.

Usage

```
check_colnames(validator)
```

Arguments

validator A Validator object containing the column names to be checked.

Details

The function performs the following checks on the column names:

- Ensures column names do not contain spaces.
- Ensures column names do not contain symbols other than underscores.
- Ensures column names do not contain uppercase letters.

For each check, a QA entry is added to the Validator object with details about the check, whether it passed, and the IDs of failing columns (if any). # nolint: line_length_linter.

Value

The updated Validator object with QA entries for each check.

check_column_contents *Check Column Contents against schema and checks*

Description

This function performs checks on the columns of `Validator$data` to ensure they meet the specified schema conditions and checks.

Usage

```
check_column_contents(validator)
```

Arguments

validator A `Validator` object containing the column names to be checked.

Value

The updated `Validator` object with QA entries for each check.

check_completeness *Check dataset for missing columns*

Description

Check dataset for missing columns

Usage

```
check_completeness(validator)
```

Arguments

validator data `Validator` object

Value

The updated `validator` object with new log entries appended.

check_duplicates	<i>Check for duplicate rows. Can use subset of columns to check for duplicates if duplicates_cols is specified in the schema. Otherwise, all columns are used for duplicate check.</i>
------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Description

Check for duplicate rows. Can use subset of columns to check for duplicates if duplicates_cols is specified in the schema. Otherwise, all columns are used for duplicate check.

Usage

```
check_duplicates(validator)
```

Arguments

validator Validator object

Value

The updated validator object with new log entries appended.

check_schema_contents_against_df	<i>Check schema contents against the data frame provided</i>
----------------------------------	--------------------------------------------------------------

Description

This function checks that the contents of the schema are consistent with the data frame provided. It checks for unused schema entries, incompatible schema entries, and that any columns specified in the schema are present in the data frame.

Usage

```
check_schema_contents_against_df(validator)
```

Arguments

validator A Validator object containing the data and schema to check against.

Value

The updated Validator object with QA entries added for any issues found in the schema.

check_types	<i>Check Column Types and Classes</i>
-------------	---------------------------------------

Description

This function checks the types and classes of the columns in the data against the schema defined in the Validator object.

Usage

```
check_types(validator)
```

Arguments

validator	A Validator object containing the data and schema to check against. The schema should define the expected type and optionally the class for each column.
-----------	----------------------------------------------------------------------------------------------------------------------------------------------------------

Value

The updated Validator object with quality assurance (QA) entries added for type and class checks. Each QA entry includes a description, pass/fail status, and any failing column IDs.

export	<i>Generic export function</i>
--------	--------------------------------

Description

This function exists For ease of use - see export.Validator() for details.

Usage

```
export(object, ...)
```

Arguments

object	The object to be checked.
...	Additional arguments passed to specific methods.

Value

The result of the export operation, specific to the object type.

export.Validator	<i>Export Validator Log</i>
------------------	-----------------------------

Description

This function exports the log of a Validator object to a file in the specified format.

Usage

```
## S3 method for class 'Validator'
export(object, file, format = c("yaml", "json", "html", "csv"), ...)
```

Arguments

object	A Validator object containing the log to be exported.
file	A string specifying the file path where the log will be exported. The file extension must match the specified format.
format	A string specifying the format of the output file. Supported formats are "yaml", "json", "html", and "csv".
...	Additional arguments passed to specific methods.

Value

Writes the log to the specified file. No value is returned.

hard_checks_status	<i>Check the status of errors and warnings in the validator log</i>
--------------------	---------------------------------------------------------------------

Description

This function raises errors or warnings if any checks flagged as error or warnings fail.

Usage

```
hard_checks_status(validator, hard_check)
```

Arguments

validator	A Validator object to check the log.
hard_check	A logical value indicating whether to perform hard checks (default is TRUE).

Value

Warning if there are any warnings or errors in the log when hard_check is FALSE. Error if there are any errors and hard_check is TRUE.

iqr_bounds	<i>Flag outliers based on Interquartile Range (IQR). Outliers are flagged if they are below $Q1 - (\text{multiplier} * \text{IQR})$ or above $Q3 + (\text{multiplier} * \text{IQR})$.</i>
------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Description

Flag outliers based on Interquartile Range (IQR). Outliers are flagged if they are below $Q1 - (\text{multiplier} * \text{IQR})$ or above $Q3 + (\text{multiplier} * \text{IQR})$.

Usage

```
iqr_bounds(x, multiplier = 1.5)
```

Arguments

x	A numeric vector.
multiplier	A numeric value to multiply the IQR by (default is 1.5).

Value

A vector the same size as x, with TRUE for values that are outliers and FALSE otherwise

is_column_contents_valid	<i>Check column contents valid</i>
--------------------------	------------------------------------

Description

This wrapper calls is_valid_column_values for each column in the schema

Usage

```
is_column_contents_valid(schema)
```

Arguments

schema	the validator schema
--------	----------------------

Value

TRUE if all column values are valid, otherwise an error is raised.

is_type_valid	<i>Check type of column in schema is valid</i>
---------------	------------------------------------------------

Description

Check type of column in schema is valid

Usage

```
is_type_valid(schema)
```

Arguments

schema	the validator schema
--------	----------------------

Value

TRUE if all column types are valid, otherwise an error is raised.

is_valid_column_values	<i>Check that max values are not less than min values in column schema</i>
------------------------	----------------------------------------------------------------------------

Description

This function checks that for any column schema, the max values (e.g., max_string_length, max_date) are not less than the corresponding min values (e.g., min_string_length, min_date). If any such inconsistency is found, an error is raised with a descriptive message.

Usage

```
is_valid_column_values(column_schema, col_name)
```

Arguments

column_schema	A list representing the schema for a specific column, which may contain max and min value specifications.
col_name	The name of the column being checked, used for error messages.

Value

TRUE if all max values are greater than or equal to their corresponding min values, otherwise an error is raised.

is_valid_schema	<i>Check if the schema is valid</i>
-----------------	-------------------------------------

Description

Check if the schema is valid

Usage

```
is_valid_schema(schema)
```

Arguments

schema	A list to validate.
--------	---------------------

Value

TRUE if the schema is a valid named list, otherwise FALSE.

log_html	<i>Generate HTML Representation of a Log</i>
----------	----------------------------------------------

Description

Generate HTML Representation of a Log

Usage

```
log_html(validator)
```

Arguments

validator	A Validator object containing a log to be converted into HTML. It is expected to be in a format compatible with log_to_table.
-----------	-------------------------------------------------------------------------------------------------------------------------------

Value

A string containing the HTML representation of the log.

`log_pointblank_outcomes`*Log pointblank validation outcomes to a validator log*

Description

This function extracts validation results from a pointblank agent and appends them to the validator's log.

Usage

```
log_pointblank_outcomes(validator)
```

Arguments

`validator` A list containing a pointblank agent and a log. The agent should have a `validation_set` from a pointblank interrogation.

Details

Each entry in the log will contain the timestamp, description, outcome, failing row indices, number of failures, and entry type for each validation step.

Value

The updated validator list with new log entries appended.

`log_to_table`*Convert Validator Log to Table*

Description

This function converts a validator log into a formatted data frame (table) for exports.

Usage

```
log_to_table(log)
```

Arguments

`log` A list representing the validator log, where each element is a log entry.

Value

A data frame containing the formatted log entries.

new_validator	<i>Validator Constructor</i>
---------------	------------------------------

Description

Creates a Validator object to validate data against a given schema.

Usage

```
new_validator(  
  data,  
  schema,  
  backseries = NULL,  
  name = deparse(substitute(data))  
)
```

Arguments

data	A data frame to validate against the schema.
schema	A schema object that defines the validation rules. See the vignette for more details on schema structure. This can also be a file path to a JSON, YAML, or TOML file containing the schema.
backseries	A previous version of the data to check against (optional).
name	validator name - defaults to the name of the dataframe object supplied to "data" (optional). Must be a single character string.

Value

An object of class Validator.

Examples

```
# create schema  
schema <- list(  
  check_duplicates = FALSE,  
  check_completeness = FALSE,  
  columns = list(  
    age = list(type = "double", optional = FALSE),  
    sex = list(type = "character", optional = FALSE)  
  )  
)  
  
# create dataframe  
df <- data.frame(  
  age = c(10, 11, 13, 15, 22, 34, 80),  
  sex = c("M", "F", "M", "F", "M", "F", "M")  
)
```

```
# create validator object
validator <- new_validator(
  data = df,
  schema = schema
)
```

print.Validator	<i>Print Validator Log</i>
-----------------	----------------------------

Description

This function prints the log of a Validator object in a markdown table format.

Usage

```
## S3 method for class 'Validator'
print(x, ...)
```

Arguments

x	A Validator object containing a log to be printed.
...	Additional arguments passed to specific methods.

Value

A markdown-formatted table of the validator log.

run_checks	<i>Run column checks</i>
------------	--------------------------

Description

To be used by check_column_contents - not intended to be run separately.

Usage

```
run_checks(validator, i_col)
```

Arguments

validator	Validator object passed from check_column_contents.
i_col	column index

Value

validator object

types_to_classes	<i>Convert complex types to the correct types and classes</i>
------------------	---------------------------------------------------------------

Description

This function modifies a schema by converting column types to their corresponding R classes.

Usage

```
types_to_classes(schema)
```

Arguments

schema	A list containing a columns element, where each column is a list with a type field.
--------	-------------------------------------------------------------------------------------

Value

The modified schema with updated type and class fields for each column.

validate_and_convert_date_formats	<i>Validate date formats in the schema This function checks that any date formats specified in the schema are valid and can be parsed correctly.</i>
-----------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------

Description

Validate date formats in the schema This function checks that any date formats specified in the schema are valid and can be parsed correctly.

Usage

```
validate_and_convert_date_formats(schema)
```

Arguments

schema	A list containing a columns element, where each column may have min_date and max_date fields.
--------	-----------------------------------------------------------------------------------------------

Value

The original schema if all date formats are valid. If any date format is invalid, an error is thrown with a message indicating the issue.

`z_score`*Check Z Score of Numeric Columns*

Description

This function calculates the maximum z-score for a numeric column.

Usage

```
z_score(x)
```

Arguments

`x` A numeric vector.

Value

A vector of the same length as `x`, indicating the z-score for each element.

Index

[add_check](#), 2
[add_check_custom](#), 3
[add_qa_entry](#), 4

[check](#), 4
[check_and_export](#), 5
[check_backseries](#), 7
[check_colnames](#), 7
[check_column_contents](#), 8
[check_completeness](#), 8
[check_duplicates](#), 9
[check_schema_contents_against_df](#), 9
[check_types](#), 10

[export](#), 10
[export.Validator](#), 11

[hard_checks_status](#), 11

[iqr_bounds](#), 12
[is_column_contents_valid](#), 12
[is_type_valid](#), 13
[is_valid_column_values](#), 13
[is_valid_schema](#), 14

[log_html](#), 14
[log_pointblank_outcomes](#), 15
[log_to_table](#), 15

[new_validator](#), 16

[print.Validator](#), 17

[run_checks](#), 17

[types_to_classes](#), 18

[validate_and_convert_date_formats](#), 18

[z_score](#), 19