Package 'data.validator'

December 11, 2023

Type Package

Title Automatic Data Validation and Reporting

Version 0.2.1

Description

Validate dataset by columns and rows using convenient predicates inspired by 'assertr' package. Generate good looking HTML report or print console output to display in logs of your data processing pipeline.

URL https://appsilon.github.io/data.validator/,

https://github.com/Appsilon/data.validator

BugReports https://github.com/Appsilon/data.validator/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Language en-US

Imports assertr (>= 2.8), shiny.semantic (>= 0.3.3), knitr, purrr, dplyr, tidyr, utils, R6, rlang, rmarkdown, htmltools, htmlwidgets, tibble

Suggests covr, fixtuRes, fs, lintr, magrittr, rcmdcheck, readr, shiny, spelling, targets, testthat, visNetwork, withr

Collate 'results_parsers.R' 'semantic_report_constructors.R' 'utils.R' 'report.R' 'assertions.R'

NeedsCompilation no

Author Marcin Dubel [aut, cre],
Paweł Przytuła [aut],
Jakub Nowicki [aut],
Krystian Igras [aut],
Dominik Krzeminski [ctb],
Servet Ahmet Çizmeli [ctb],
Appsilon Sp. z o.o. [cph]

Maintainer Marcin Dubel <opensource+marcin@appsilon.com>

Repository CRAN

Date/Publication 2023-12-11 10:20:06 UTC

R topics documented:

---

add_results   *Add validation results to the Report object*

Description

This function adds results to validator object with aggregating summary of success, error and warning checks. Moreover it parses assertr results attributes and stores them inside usable table.

Usage

```
add_results(data, report)
```

Arguments

data	Data that was validated.
report	Report object to store validation results.

---

data_validation_report
        Create new validator object

Description

The object returns R6 class environment responsible for storing validation results.

Usage

```
data_validation_report()
```

---

get_results *Get validation results*

Description

The response is a list containing information about successful, failed, warning assertions and the table stores important information about validation results. Those are:

- table_name - name of validated table
- assertion.id - id used for each assertion
- description - assertion description
- num.violations - number of violations (assertion and column specific)
- call - assertion call
- message - assertion result message for specific column
- type - error, warning or success
- error_df - nested table storing details about error or warning result (like violated indexes and values)

Usage

```
get_results(report, unnest = FALSE)
```

Arguments

report	Report object that stores validation results. See add_results.
unnest	If TRUE, error_df table is unnested. Results with remaining columns duplicated in table.

---

render_raw_report_ui *Render simple version of report*

Description

Renders content of simple report version that prints `validation_results` table.

Usage

```
render_raw_report_ui(
  validation_results,
  success = TRUE,
  warning = TRUE,
  error = TRUE
)
```

Arguments

validation_results

> Validation results table (see [get_results](#)).

success Should success results be presented?

warning Should warning results be presented?

error Should error results be presented?

---

render_semantic_report_ui

Render semantic version of report

Description

Renders content of semantic report version.

Usage

```
render_semantic_report_ui(
  validation_results,
  success = TRUE,
  warning = TRUE,
  error = TRUE,
  df_error_head_n = 6L
)
```

Arguments

validation_results

> Validation results table (see [get_results](#)).

success Should success results be presented?

warning Should warning results be presented?

error Should error results be presented?

df_error_head_n

> Number of rows to display in error table. Works in the same way as head function.

---

save_report   *Saving results as a HTML report*

Description

Saving results as a HTML report

Usage

```
save_report(
  report,
  output_file = "validation_report.html",
  output_dir = getwd(),
  ui_constructor = render_semantic_report_ui,
 template = system.file("rmarkdown/templates/standard/skeleton/skeleton.Rmd", package =
    "data.validator"),
  ...
)
```

Arguments

report	Report object that stores validation results.
output_file	Html file name to write report to.
output_dir	Target report directory.
ui_constructor	Function of `validation_results` and optional parameters that generates HTML code or HTML widget that should be used to generate report content. See `custom_report` example.
template	Path to Rmd template in which ui_constructor is rendered. See `data.validator` rmarkdown template to see basic construction - the one is used as a default template.
...	Additional parameters passed to ui_constructor. For example: `df_error_head_n`

---

save_results   *Saving results table to external file*

Description

Saving results table to external file

Usage

```
save_results(report, file_name = "results.csv", method = utils::write.csv, ...)
```

Arguments

report	Report object that stores validation results. See [get_results](#).
file_name	Name of the resulting file (including extension).
method	Function that should be used to save results table (write.csv default) The function passed to `method` should have 'x' and 'file' arguments. Functions with different arguments can be passed by creating a wrapper function for it. See example save_results_methods.
...	Remaining parameters passed to `method`.

---

save_summary   *Save simple validation summary in text file*

Description

Saves `print(validator)` output inside text file.

Usage

```
save_summary(
  report,
  file_name = "validation_log.txt",
  success = TRUE,
  warning = TRUE,
  error = TRUE
)
```

Arguments

report	Report object that stores validation results.
file_name	Name of the resulting file (including extension).
success	Should success results be presented?
warning	Should warning results be presented?
error	Should error results be presented?

---

validate	*Prepare data for validation chain*

Description

Prepare data for validation and generating report. The function prepares data for chain validation and ensures all the validation results are gathered correctly. The function also attaches additional information to the data (name and description) that is then displayed in validation report.

Usage

```
validate(data, name, description = NULL)
```

Arguments

data	data.frame or tibble to test
name	name of validation object (will be displayed in the report)
description	description of validation object (will be displayed in the report)

---

validate_cols	*Validation on columns*

Description

Validation on columns

Usage

```
validate_cols(
  data,
  predicate,
  ...,
  obligatory = FALSE,
  description = NA,
  skip_chain_opts = FALSE,
  success_fun = assertr::success_append,
  error_fun = assertr::error_append,
  defect_fun = assertr::defect_append
)
```

Arguments

`data`	A data.frame or tibble to test
`predicate`	Predicate function or predicate generator such as [`in_set`](#) or [`within_n_sds`](#)
`...`	Columns selection that `predicate` should be called on. All tidyselect [`language`](#) methods are supported. If not provided, all [`everything`](#) will be used.
`obligatory`	If TRUE and assertion failed the data is marked as defective. For defective data, all the following rules are handled by defect_fun function
`description`	A character string with description of assertion. The description is then displayed in the validation report
`skip_chain_opts`	
	While wrapping data with [validate](#) function, `success_fun` and `error_fun` parameters are rewritten with `success_append` and `error_append` respectively. In order to use parameters assigned to the function directly set skip_chain_opts to TRUE
`success_fun`	Function that is called when the validation pass
`error_fun`	Function that is called when the validation fails
`defect_fun`	Function that is called when the data is marked as defective

See Also

validate_if validate_rows

`validate_if`	*Verify if expression regarding data is TRUE*

Description

The function checks whether all the logical values returned by the expression are TRUE. The function is meant for handling all the cases that cannot be reached by using [validate_cols](#) and [validate_rows](#) functions.

Usage

```
validate_if(
  data,
  expr,
  description = NA,
  obligatory = FALSE,
  skip_chain_opts = FALSE,
  success_fun = assertr::success_append,
  error_fun = assertr::error_append,
  defect_fun = assertr::defect_append
)
```

Arguments

`data`	A data.frame or tibble to test
`expr`	A Logical expression to test for, e.g. `var_name > 0`
`description`	A character string with description of assertion. The description is then displayed in the validation report
`obligatory`	If TRUE and assertion failed the data is marked as defective. For defective data, all the following rules are handled by defect_fun function
`skip_chain_opts`	
	While wrapping data with [validate](#) function, `success_fun` and `error_fun` parameters are rewritten with `success_append` and `error_append` respectively. In order to use parameters assigned to the function directly set skip_chain_opts to TRUE
`success_fun`	Function that is called when the validation pass
`error_fun`	Function that is called when the validation fails
`defect_fun`	Function that is called when the data is marked as defective

See Also

validate_cols validate_rows

`validate_rows`	*Validation on rows*

Description

Validation on rows

Usage

```
validate_rows(
  data,
  row_reduction_fn,
  predicate,
  ...,
  obligatory = FALSE,
  description = NA,
  skip_chain_opts = FALSE,
  success_fun = assertr::success_append,
  error_fun = assertr::error_append,
  defect_fun = assertr::defect_append
)
```

Arguments

data	A data.frame or tibble to test
row_reduction_fn	
	Function that should reduce rows into a single column that is passed to validation e.g. num_row_NAs
predicate	Predicate function or predicate generator such as in_set or within_n_sds
...	Columns selection that row_reduction_fn should be called on. All tidyselect language methods are supported. If not provided, all everything will be used.
obligatory	If TRUE and assertion failed the data is marked as defective. For defective data, all the following rules are handled by defect_fun function
description	A character string with description of assertion. The description is then displayed in the validation report
skip_chain_opts	
	While wrapping data with validate function, success_fun and error_fun parameters are rewritten with success_append and error_append respectively. In order to use parameters assigned to the function directly set skip_chain_opts to TRUE.
success_fun	Function that is called when the validation pass
error_fun	Function that is called when the validation fails
defect_fun	Function that is called when the data is marked as defective

See Also

validate_cols validate_if

Index