Package ‘data.validator’

Type Package
Title Automatic Data Validation and Reporting
Version 0.1.5
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.

BugReports https://github.com/Appsilon/data.validator/issues
License MIT + file LICENSE
Encoding UTF-8
LazyData TRUE
RoxygenNote 7.1.1
Imports shiny, assertr (>= 2.8), shiny.semantic (>= 0.3.3), knitr, purrr, dplyr, tidyr, utils, R6, rlang, rmarkdown, htmltools, htmlwidgets, tibble
Suggests testthat, covr
Collate 'results_parsers.R' 'semantic_report_constructors.R' 'utils.R'
'report.R' 'assertions.R'
NeedsCompilation no
Author Krystian Igras [aut], Marcin Dubel [aut], Paweł Przytuła [aut], Dominik Krzeminski [ctb, cre], Appsilon Sp. z o.o. [cph]
Maintainer Dominik Krzeminski <dominik@appsilon.com>
Repository CRAN
Date/Publication 2021-03-02 14:40:02 UTC
R topics documented:

add_results ................................................. 2
convert_error_df ........................................... 3
create_summary_row ......................................... 3
data_validation_report ..................................... 4
display_results ............................................. 4
error_class .................................................. 5
find_chain_parts ............................................ 5
generate_id ................................................... 5
get_assertion_type ........................................... 6
get_assert_method ........................................... 6
get_first_name .............................................. 7
get_results .................................................. 7
get_results_number ......................................... 8
get_semantic_report_ui ..................................... 8
makeAccordion_container .................................. 9
makeAccordion_element ..................................... 9
make_summary_table ......................................... 10
make_table_row ............................................. 10
parse_errors_to_df ......................................... 11
parse_results_to_df ....................................... 11
parse_successes_to_df ..................................... 12
prepare_modal_content ...................................... 12
render_raw_report_ui ....................................... 13
render_semantic_report_ui .................................. 13
result_table .................................................. 14
save_report ................................................... 14
save_results ................................................. 15
save_summary .................................................. 15
segment ....................................................... 16
validate ....................................................... 16
validate_cols .................................................. 17
validate_if .................................................... 18
validate_rows .................................................. 19

Index 21

---

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.
**convert_error_df**

**Usage**

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

**Arguments**

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

---

**Description**

Convert error table column types

**Usage**

```python
convert_error_df(error_df)
```

**Arguments**

- `error_df`: Table consisting assert error details

---

**create_summary_row**

**Create summary table row.**

**Description**

Create summary table row.

**Usage**

```python
create_summary_row(id, number, color, label)
```

**Arguments**

- `id`: ID.
- `number`: Number to display.
- `color`: Color of the label.
- `label`: Label to display.

**Value**

Summary table row.
**data_validation_report**

*Create new validator object*

**Description**

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

**Usage**

```r
data_validation_report()
```

**display_results**

*Displays results of validations.*

**Description**

Displays results of validations.

**Usage**

```r
display_results(data, n_passes, n_fails, n_warns)
```

**Arguments**

- `data` Report data.
- `n_passes` Number of successful assertions.
- `n_fails` Number of warning assertions.
- `n_warns` Number of violation assertions.

**Value**

Validation report.
**error_class Constants**

**Description**

Constants

**Usage**

```r
error_class
```

**Format**

An object of class `character` of length 1.

---

**find_chain_parts**

*Find all chain parts in parent frame*

**Description**

Find all chain parts in parent frame

**Usage**

```r
find_chain_parts()
```

---

**generate_id**

*Generate a random ID.*

**Description**

Generate a random ID.

**Usage**

```r
generate_id()
```

**Value**

A characters corresponding to random ID.
**get_assertion_type**  
*get assertion type*

**Description**

get assertion type

**Usage**

```r
get_assertion_type(assertion)
```

**Arguments**

- `assertion`  
  assertion object (check `assertr` package for details)

**Value**

character with id of assertion: "error", "success", "warning"

---

**get_assert_method**  
*Match proper method depending on predicate type*

**Description**

Match proper method depending on predicate type

**Usage**

```r
get_assert_method(  
  predicate,  
  method = list(direct = assertr::assert, generator = assertr::insist)  
)
```

**Arguments**

- `predicate`  
  Predicate or predicate generator function.

- `method`  
  optional list with fields direct and generator of assertions
**get_first_name**

*Get first name of the data frame*

**Description**

Get first name of the data frame

**Usage**

```r
get_first_name(df)
```

**Arguments**

- `df` : data.frame

**Value**

deparsed chain part

---

**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**

```r
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.
**get_results_number**  
*Get results number*

**Description**
Get results number.

**Usage**
```
get_results_number(results)
```

**Arguments**
- `results`: assertion results

**Value**
- table with results number

---

**get_semantic_report_ui**  
*Generate HTML report.*

**Description**
Generate HTML validation report.

**Usage**
```
get_semantic_report_ui(n_passes, n_fails, n_warns, validation_results)
```

**Arguments**
- `n_passes`: Number of passed validations
- `n_fails`: Number of failed validations.
- `n_warns`: Number of warnings.
- `validation_results`: Data frame with validation results.

**Value**
- HTML validation report.
make_accordion_container

Create a UI accordion container.

Description
Create a UI accordion container.

Usage
make_accordion_container(...)

Arguments
... Additional arguments inside accordion container.

Value
Accordion container.

make_accordion_element

Create a UI accordion element.

Description
Create a UI accordion element.

Usage
make_accordion_element(
    results, 
    color = "green", 
    label, 
    active = FALSE, 
    type, 
    mark 
)

Arguments
results Results to display.
color Color of the label icon.
label Label.
active Is active?
type Result type.
mark Icon to display.
**Value**

Accordion.

---

**make_summary_table**

*Create summary table.*

---

**Description**

Create summary table.

**Usage**

```make_summary_table(n_passes, n_fails, n_warns)```

**Arguments**

- `n_passes` Number of passed validations.
- `n_fails` Number of failed validations.
- `n_warns` Number of warnings.

**Value**

Summary table.

---

**make_table_row**

*Create table row.*

---

**Description**

Create table row.

**Usage**

```make_table_row(results, type, mark)```

**Arguments**

- `results` Results to display in a row.
- `type` Result type.
- `mark` Icon to display.

**Value**

Table row.
**Description**

Parse errors to data.frame

**Usage**

```
parse_errors_to_df(data)
```

**Arguments**

- `data` object of assertr error class (check assertr package for details)

**Value**

data.frame with errors

---

**Description**

Parse results to data.frame

**Usage**

```
parse_results_to_df(data)
```

**Arguments**

- `data` assertr object (check assertr package for details)

**Value**

data.frame with successes and errors
parse_successes_to_df Parse successes to data.frame

Description
Parse successes to data.frame

Usage
parse_successes_to_df(data)

Arguments
data object of assertr success class (check assertr package for details)

Value
data.frame with successes

prepare_modal_content Prepare modal content.

Description
Prepare modal content.

Usage
prepare_modal_content(error)

Arguments
error Assertr error.

Value
Modal content.
**render_raw_report_ui**  
*Render simple version of report*

**Description**

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

**Usage**

```r
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**

```r
render_semantic_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?

**result_table**  
*Create table with results.*

---

**Description**

Create table with results.

**Usage**

```r
result_table(results, type, mark)
```

**Arguments**

- `results`  
  Result to display in table.
- `type`  
  Result type.
- `mark`  
  Icon to display.

**Value**

Table row.

---

**save_report**  
*Saving results as a HTML report*

---

**Description**

Saving results as a HTML report.

**Usage**

```r
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"),
  ...
)
```
### save_results

**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.

**Description**

Saving results table to external file

**Usage**

```r
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).
- **...**
  - Remaining parameters passed to method.

---

### save_summary

**Description**

Save simple validation summary in text file

**Usage**

```r
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?

---

**segment**

Create a UI segment element.

Description

Create a UI segment element.

Usage

```r
segment(title, ...)
```

Arguments

- `title`: Title of the segment.
- `...`: Additional arguments inside segment.

Value

Segment.

---

**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

```r
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

```r
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
- **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

```r
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

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

* datasets
  error_class, 5
  add_results, 2, 7
  convert_error_df, 3
  create_summary_row, 3
  data_validation_report, 4
  display_results, 4
  error_class, 5
  find_chain_parts, 5
  generate_id, 5
  get_assert_method, 6
  get_assertion_type, 6
  get_first_name, 7
  get_results, 7, 13–15
  get_results_number, 8
  get_semantic_report_ui, 8
  in_set, 17, 19
  language, 17, 19

makeAccordionContainer, 9
makeAccordionElement, 9
make_summary_table, 10
make_table_row, 10
num_row_NAs, 19

parse_errors_to_df, 11
parse_results_to_df, 11
parse_successes_to_df, 12
prepare_modal_content, 12

render_raw_report_ui, 13
render_semantic_report_ui, 13
result_table, 14

save_report, 14
save_results, 15
save_summary, 15
segment, 16
validate, 16, 17–19
validate_cols, 17, 18
validate_if, 18
validate_rows, 18, 19
within_n_sds, 17, 19