Package ‘data.validator’

October 13, 2022

Type Package

Title Automatic Data Validation and Reporting

Version 0.1.6

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

RoxygenNote 7.1.2

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, cre], Paweł Przytulą [aut], Dominik Krzeminski [ctb], Servet Ahmet Çizmeli [ctb], Appsilon Sp. z o.o. [cph]

Maintainer Marcin Dubel <marcin@appsilon.com>

Repository CRAN

Date/Publication 2022-01-19 16:32:42 UTC
add_results

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**

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

**Arguments**

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

---

**convert_error_df**

**Convert error table column types**

**Description**

Convert error table column types

**Usage**

```python
convert_error_df(error_df)
```

**Arguments**

- `error_df` (Table consisting assertr 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**

data_validation_report()

---

display_results

*Displays results of validations.*

**Description**

Displays results of validations.

**Usage**

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.
<table>
<thead>
<tr>
<th>error_class</th>
<th>Constants</th>
</tr>
</thead>
</table>

**Description**

Constants

**Usage**

```r
error_class
```

**Format**

An object of class character of length 1.

<table>
<thead>
<tr>
<th>find_chain_parts</th>
<th>Find all chain parts in parent frame</th>
</tr>
</thead>
</table>

**Description**

Find all chain parts in parent frame

**Usage**

```r
find_chain_parts()
```

<table>
<thead>
<tr>
<th>generate_id</th>
<th>Generate a random ID.</th>
</tr>
</thead>
</table>

**Description**

Generate a random ID.

**Usage**

```r
generate_id()
```

**Value**

A characters corresponding to random ID.
### get_assertion_type

**Description**

get assertion type

**Usage**

`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

**Description**

Match proper method depending on predicate type

**Usage**

`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**

```python
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**

```python
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.
**makeAccordionContainer**

*Create a UI accordion container.*

**Description**

Create a UI accordion container.

**Usage**

```r
makeAccordionContainer(...)```

**Arguments**

- ... Additional arguments inside accordion container.

**Value**

Accordion container.

---

**makeAccordionElement**

*Create a UI accordion element.*

**Description**

Create a UI accordion element.

**Usage**

```r
makeAccordionElement(
    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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_passes</td>
<td>Number of passed validations.</td>
</tr>
<tr>
<td>n_fails</td>
<td>Number of failed validations.</td>
</tr>
<tr>
<td>n_warns</td>
<td>Number of warnings.</td>
</tr>
</tbody>
</table>

**Value**

Summary table.

---

**make_table_row**

Create table row.

---

**Description**

Create table row.

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>results</td>
<td>Results to display in a row.</td>
</tr>
<tr>
<td>type</td>
<td>Result type.</td>
</tr>
<tr>
<td>mark</td>
<td>Icon to display.</td>
</tr>
</tbody>
</table>

**Value**

Table row.
### parse_errors_to_df

**Parse errors to data.frame**

**Description**

Parse errors to data.frame

**Usage**

```r
parse_errors_to_df(data)
```

**Arguments**

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

**Value**

data.frame with errors

### parse_results_to_df

**Parse results to data.frame**

**Description**

Parse results to data.frame

**Usage**

```r
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
```r
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
```r
prepare_modal_content(error)
```

#### Arguments
- `error`  
  Assertr error.

#### Value
- Modal content.
**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 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

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>report</td>
<td>Report object that stores validation results. See get_results.</td>
</tr>
<tr>
<td>output_file</td>
<td>Html file name to write report to.</td>
</tr>
<tr>
<td>output_dir</td>
<td>Target report directory.</td>
</tr>
<tr>
<td>ui_constructor</td>
<td>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.</td>
</tr>
<tr>
<td>template</td>
<td>Path to Rmd template in which ui_constructor is rendered. See data.validatormarkdown template to see basic construction - the one is used as a default template.</td>
</tr>
</tbody>
</table>

... Additional parameters passed to ui_constructor.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>report</td>
<td>Report object that stores validation results. See get_results.</td>
</tr>
<tr>
<td>file_name</td>
<td>Name of the resulting file (including extension).</td>
</tr>
<tr>
<td>method</td>
<td>Function that should be used to save results table (write.csv default).</td>
</tr>
</tbody>
</table>

... 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
)
validate

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>report</td>
<td>Report object that stores validation results.</td>
</tr>
<tr>
<td>file_name</td>
<td>Name of the resulting file (including extension).</td>
</tr>
<tr>
<td>success</td>
<td>Should success results be presented?</td>
</tr>
<tr>
<td>warning</td>
<td>Should warning results be presented?</td>
</tr>
<tr>
<td>error</td>
<td>Should error results be presented?</td>
</tr>
</tbody>
</table>

segment

Create a UI segment element.

Description

Create a UI segment element.

Usage

segment(title, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Title of the segment.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments inside segment.</td>
</tr>
</tbody>
</table>

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

validate(data, name, description = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>data.frame or tibble to test</td>
</tr>
<tr>
<td>name</td>
<td>name of validation object (will be displayed in the report)</td>
</tr>
<tr>
<td>description</td>
<td>description of validation object (will be displayed in the report)</td>
</tr>
</tbody>
</table>
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` 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 = asserttr::success_append,
error_fun = asserttr::error_append,
defect_fun = asserttr::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
exterior_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
  - make_accordion_container, 9
  - make_accordion_element, 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

21