Package ‘dataverifyr’

May 2, 2023

Type Package

Title A Lightweight, Flexible, and Fast Data Validation Package that Can Handle All Sizes of Data

Version 0.1.5

Description Allows you to define rules which can be used to verify a given dataset.
The package acts as a thin wrapper around more powerful data packages such as ‘dplyr’, ‘data.table’, ‘arrow’, and ‘DBI’ (‘SQL’), which do the heavy lifting.

License MIT + file LICENSE

URL https://github.com/DavZim/dataverifyr,
https://davzim.github.io/dataverifyr/

BugReports https://github.com/DavZim/dataverifyr/issues

Imports yaml

Suggests arrow, data.table, DBI, dplyr, dbplyr, duckdb, RSQLite, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author David Zimmermann-Kollenda [aut, cre]

Maintainer David Zimmermann-Kollenda <david_j_zimmermann@hotmail.com>

Repository CRAN

Date/Publication 2023-05-02 18:30:06 UTC

R topics documented:

  check_data ................................................................. 2
  filter_fails .................................................................. 3
  plot_res ................................................................. 3
check_data

Checks if a dataset confirms to a given set of rules

Description

Checks if a dataset confirms to a given set of rules

Usage

check_data(x, rules, fail_on_warn = FALSE, fail_on_error = FALSE)

Arguments

x          a dataset, either a data.frame, dplyr::tibble, data.table::data.table, arrow::arrow_table, arrow::open_dataset, or dplyr::tbl (SQL connection)
rules      a list of rules
fail_on_warn if the function should throw an error on a warning
fail_on_error if the function should throw an error on a failed rule

Value

a data.frame-like object with one row for each rule and its results

Examples

rs <- ruleset(
  rule(mpg > 10),
  rule(cyl %in% c(4, 6)), # missing 8
  rule(qsec >= 14.5 & qsec <= 22.9)
)
rs

check_data(mtcars, rs)
filter_fails

Filters a result dataset for the values that failed the verification

Description

Filters a result dataset for the values that failed the verification

Usage

filter_fails(res, x, per_rule = FALSE)

Arguments

res a result data.frame as outputted from check_data()

x a dataset that was used in check_data()

per_rule if set to TRUE, a list of filtered data is returned, one for each failed verification rule. If set to FALSE, a data.frame is returned of the values that fail any rule.

Value

the dataset with the entries that did not match the given rules

Examples

rules <- ruleset(
  rule(mpg > 10 & mpg < 30), # mpg goes up to 34
  rule(cyl %in% c(4, 8)), # missing 6 cyl
  rule(vs %in% c(0, 1), allow_na = TRUE)
)

res <- check_data(mtcars, rules)

fails <- filter_fails(res, mtcars)

fails

plot_res

Visualize the results of a data validation

Description

Visualize the results of a data validation
Usage

```
plot_res(
    res,
    main = "Verification Results per Rule",
    colors = c(pass = "#308344", fail = "#E66820"),
    labels = TRUE,
    table = TRUE
)
```

Arguments

- `res`: a data.frame as returned by `check_data()`
- `main`: the title of the plot
- `colors`: a named list of colors, with the names pass and fail
- `labels`: whether the values should be displayed on the barplot
- `table`: show a table in the legend with the values

Value

a base r plot

Examples

```
rs <- ruleset(
    rule(Ozone > 0 & Ozone < 120, allow_na = TRUE), # some mising values and > 120
    rule(Solar.R > 0, allow_na = TRUE),
    rule(Solar.R < 200, allow_na = TRUE),
    rule(Wind > 10),
    rule(Temp < 100)
)

res <- check_data(airquality, rs)
plot_res(res)
```

---

**rule**

*Creates a single data rule*

Description

Creates a single data rule

Usage

```
rule(expr, name = NA, allow_na = FALSE, negate = FALSE, ...)
```

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

**expr**  
an expression which dictates which determines when a rule is good. Note that the expression is evaluated in `check_data()`, within the given framework. That means, for example if a the data given to `check_data()` is an arrow dataset, the expression must be mappable from `arrow` (see also arrow documentation). The expression can be given as a string as well.

**name**  
an optional name for the rule for reference

**allow_na**  
does the rule allow for NA values? default value is FALSE

**negate**  
is the rule negated, only applies to the expression not allow_na, that is, if `expr = mpg > 10, allow_na = TRUE, and negate = TRUE, it would match all mpg <= 10 as well as NAs.

...  
additional arguments that are carried along for your documentation, but are not used. Could be for example date, person, contact, comment, etc

**x**  
a rule to print

Value

The rule values as a list

Methods (by generic)

- `print(rule)`: Prints a rule

Examples

```r
r <- rule(mpg > 10)
r

r2 <- rule(mpg > 10, name = "check that mpg is reasonable", allow_na = TRUE,
          negate = FALSE, author = "me", date = Sys.Date())
r2

check_data(mtcars, r)

rs <- ruleset(
  rule(mpg > 10),
  rule(cyl %in% c(4, 6)), # missing 8
  rule(qsec >= 14.5 & qsec <= 22.9)
)
rs
check_data(mtcars, rs)
```
**ruleset**  
*Creates a set of rules*

**Description**

Creates a set of rules

**Usage**

```r
ruleset(...)  
```

## S3 method for class 'ruleset'
print(x, n = 3, ...)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>a list of rules</td>
</tr>
<tr>
<td>x</td>
<td>a ruleset to print</td>
</tr>
<tr>
<td>n</td>
<td>a maximum number of rules to print</td>
</tr>
</tbody>
</table>

**Value**

the list of rules as a ruleset

**Methods (by generic)**

- `print(ruleset)`: Prints a ruleset

**Examples**

```r
r1 <- rule(mpg > 10)  
r2 <- rule(mpg < 20)  
rs <- ruleset(r1, r2)  
rs

rs <- ruleset(  
    rule(cyl %in% c(4, 6, 8)),  
    rule(is.numeric(disp))  
)  
rs
```
write_rules

Read and write rules to a yaml file

Description
Read and write rules to a yaml file

Usage
write_rules(x, file)
read_rules(file)

Arguments
x a list of rules
file a filename

Value
the filename invisibly

Functions
• read_rules(): reads a ruleset back in

Examples
rr <- ruleset(
  rule(mpg > 10),
  rule(cyl %in% c(4, 6, 8))
)
file <- tempfile(fileext = "yml")
write_rules(rr, file)
Index

arrow::arrow_table, 2
arrow::open_dataset, 2
check_data, 2
check_data(), 3, 4
data.frame, 2
data.table::data.table, 2
dplyr::tbl, 2
dplyr::tibble, 2
filter_fails, 3
plot_res, 3
print.rule(rule), 4
print.ruleset(ruleset), 6
read_rules(write_rules), 7
rule, 2, 4
ruleset, 6
write_rules, 7