

Package: dataverifyr (via r-universe)

March 5, 2025

Type Package

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

Version 0.1.8

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

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

RemoteUrl <https://github.com/davzim/dataverifyr>

RemoteRef HEAD

RemoteSha 8b6a645329b74044bd188884b7582a8578ac2c7b

Contents

bind_rules	2
check_data	2

dataverifyr_plus	3
detect_backend	4
filter_fails	5
plot_res	6
rule	7
ruleset	8
write_rules	9

Index	10
--------------	-----------

bind_rules	<i>Programatically Combine a List of Rules and Rulesets into a Single Ruleset</i>
------------	---

Description

Programatically Combine a List of Rules and Rulesets into a Single Ruleset

Usage

bind_rules(rule_ruleset_list)

Arguments

rule_ruleset_list
a list of rules and rulesets you wish to combine into a single list

Value

a ruleset which consolidates all the inputs

check_data	<i>Checks if a dataset confirms to a given set of rules</i>
------------	---

Description

Checks if a dataset confirms to a given set of rules

Usage

```
check_data(  
  x,  
  rules,  
  xname = deparse(substitute(x)),  
  stop_on_fail = FALSE,  
  stop_on_warn = FALSE,  
  stop_on_error = FALSE  
)
```

Arguments

x	a dataset, either a <code>data.frame</code> , <code>dplyr::tibble</code> , <code>data.table::data.table</code> , <code>arrow::arrow_table</code> , <code>arrow::open_dataset</code> , or <code>dplyr::tbl</code> (SQL connection)
rules	a list of <code>rules</code>
xname	optional, a name for the x variable (only used for errors)
stop_on_fail	when any of the rules fail, throw an error with stop
stop_on_warn	when a warning is found in the code execution, throw an error with stop
stop_on_error	when an error is found in the code execution, throw an error with stop

Value

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

See Also

`detect_backend()`

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)
```

dataverifyr_plus	<i>Add Rules and Rulesets Together</i>
------------------	--

Description

- allows you to add rules and rulesets into larger rulesets. This can be useful if you want to create a ruleset for a dataset out of checks for other datasets.

Usage

```
dataverifyr_plus(a, b)

## S3 method for class 'ruleset'
a + b

## S3 method for class 'rule'
a + b
```

Arguments

- a the first ruleset you wish to add
- b the second ruleset you wish to add

detect_backend	<i>Detects the backend which will be used for checking the rules</i>
----------------	--

Description

The detection will be made based on the class of the object as well as the packages installed. For example, if a `data.frame` is used, it will look if `data.table` or `dplyr` are installed on the system, as they provide more speed. Note the main functions will revert the

Usage

```
detect_backend(x)
```

Arguments

- x The data object, ie a `data.frame`, `tibble`, `data.table`, `arrow`, or `DBI` object

Value

a single character element with the name of the backend to use. One of `base-r`, `data.table`, `dplyr`, `collectibles` (for `arrow` or `DBI` objects)

See Also

[check_data\(\)](#)

Examples

```
data <- mtcars
detect_backend(data)
```

filter_fails	<i>Filters a result dataset for the values that failed the verification</i>
--------------	---

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 <code>check_data()</code> or a ruleset
x	a dataset that was used in <code>check_data()</code>
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)  
  
filter_fails(res, mtcars)  
filter_fails(res, mtcars, per_rule = TRUE)  
  
# alternatively, the first argument can also be a ruleset  
filter_fails(rules, mtcars)  
filter_fails(rules, mtcars, per_rule = TRUE)
```

`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

<code>res</code>	a data.frame as returned by <code>check_data()</code>
<code>main</code>	the title of the plot
<code>colors</code>	a named list of colors, with the names pass and fail
<code>labels</code>	whether the values should be displayed on the barplot
<code>table</code>	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 missing 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	<i>Creates a single data rule</i>
------	-----------------------------------

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 <code>check_data()</code> , within the given framework. That means, for example if a the data given to <code>check_data()</code> 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 in the data? default value is FALSE. Note that when NAs are introduced in the expression, <code>allow_na</code> has no effect. Eg when the rule <code>as.numeric(vs) %in% c(0, 1)</code> finds the values of <code>vs</code> as <code>c("1", "A")</code> , the rule will throw a fail regardless of the value of <code>allow_na</code> as the NA is introduced in the expression and is not found in the original data. However, when the values of <code>vs</code> are <code>c("1", NA)</code> , <code>allow_na</code> will have an effect.
negate	is the rule negated, only applies to the expression not <code>allow_na</code> , that is, if <code>expr = mpg > 10</code> , <code>allow_na = TRUE</code> , and <code>negate = TRUE</code> , it would match all <code>mpg <= 10</code> 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 <- 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	<i>Creates a set of rules</i>
---------	-------------------------------

Description

Creates a set of rules

Usage

```

ruleset(...)

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

```

Arguments

...	a list of rules
x	a ruleset to print
n	a maximum number of rules to print

Value

the list of rules as a ruleset

Methods (by generic)

- `print(ruleset)`: Prints a ruleset

Examples

```
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(dis))
)
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 = ".yaml")
write_rules(rr, file)
```

Index

`+.rule(dataverifyr_plus)`, 3
`+.ruleset(dataverifyr_plus)`, 3

`arrow::arrow_table`, 3
`arrow::open_dataset`, 3

`bind_rules`, 2

`check_data`, 2
`check_data()`, 4–6

`data.frame`, 3
`data.table::data.table`, 3
`dataverifyr_plus(dataverifyr_plus)`, 3
`dataverifyr_plus`, 3
`detect_backend`, 4
`detect_backend()`, 3
`dplyr::tbl`, 3
`dplyr::tibble`, 3

`filter_fails`, 5

`plot_res`, 6
`print.rule(rule)`, 7
`print.ruleset(ruleset)`, 8

`read_rules(write_rules)`, 9
`rule`, 3, 7
`ruleset`, 8

`write_rules`, 9