Package ‘errorlocate’

February 6, 2020

Type Package
Title Locate Errors with Validation Rules
Version 0.3.0
Description Errors in data can be located and removed using validation rules from package ‘validate’.
License GPL-3
LazyData TRUE

URL https://github.com/data-cleaning/errorlocate

BugReports https://github.com/data-cleaning/errorlocate/issues
Depends validate
Imports lpSolveAPI, methods
Suggests testthat (>= 2.1.0), magrittr, covr
RoxygenNote 6.1.1
Encoding UTF-8

Collate 'MipRules.R' 'addnoise.R' 'categorical.R' 'conditional.R'
'dnf.R' 'errorlocalizer.R' 'errorlocate-package.R'
'errorlocation.R' 'expr_manip.R' 'expr_simplify.R' 'linear.R'
'local_variable.R' 'locate-errors.R' 'mip_lpsolve.R'
'mip_rule.R' 'replace-errors.R' 'soft-rule.R' 'utils.R'
'values.R'

NeedsCompilation no
Author Edwin de Jonge [aut, cre] (<https://orcid.org/0000-0002-6580-4718>),
Mark van der Loo [aut]
Maintainer Edwin de Jonge <edwindjonge@gmail.com>
Repository CRAN
Date/Publication 2020-02-06 10:00:03 UTC
**R topics documented:**

- `errorlocate-package` ........................................ 2
- `add_noise` .................................................. 3
- `ErrorLocalizer-class` ..................................... 4
- `errorlocation-class` ....................................... 4
- `errors_removed` ........................................... 5
- `expect_values` .............................................. 5
- `FHLocalizer-class` ......................................... 6
- `is_categorical` ............................................ 6
- `is_conditional` ............................................. 7
- `is_linear` .................................................. 7
- `locate_errors` ............................................. 8
- `MipRules-class` ............................................ 9
- `replace_errors` ............................................. 10
- `substitute_` ................................................ 11
- `translate_mip_lp` ......................................... 12

**Index**

13

---

**errorlocate-package**  
Find errors in data given a set of validation rules.

**Description**

Find errors in data given a set of validation rules. The `errorlocate` helps to identify obvious errors in raw datasets.

**Details**

It works in tandem with the package `validate`. With `validate` you formulate data validation rules to which the data must comply. For example:

"age cannot be negative": age >= 0

While `validate` can identify if a record is valid or not, it does not identify which of the variables are responsible for the invalidation. This may seem a simple task, but is actually quite tricky: a set of validation rules form a web of dependent variables: changing the value of an invalid record to repair for rule 1, may invalidate the record for rule 2.

Errorlocate provides a small framework for record based error detection and implements the Felligi Holt algorithm. This algorithm assumes there is no other information available then the values of a record and a set of validation rules. The algorithm minimizes the (weighted) number of values that need to be adjusted to remove the invalidation.

The `errorlocate` package translates the validation and error localization problem into a mixed integer problem and uses a mip solver to find a solution.
**add_noise**

**Author(s)**

Maintainer: Edwin de Jonge <edwindjonge@gmail.com> (0000-0002-6580-4718)
Authors:

- Mark van der Loo <mark.vanderloo@gmail.com>

**References**

Van der Loo, M., de Jonge, E, Data Cleaning With Applications in R

**See Also**

Useful links:

- [https://github.com/data-cleaning/errorlocate](https://github.com/data-cleaning/errorlocate)

---

### add_noise

**Add (a small amount of) noise**

**Description**

Utility function to add some small positive noise to weights. This is mainly done to randomly choose between solutions of equal weight. Without adding noise to weights lp solvers may return an identical solution over and over while there are multiple solutions of equal weight. The generated noise is positive to prevent that weights will be zero or negative.

**Usage**

```r
add_noise(x, max_delta = NULL, ...)
```

**Arguments**

- `x`: numeric vector or matrix. When `x` is a matrix, the function will be applied to each row of the matrix.
- `max_delta`: when supplied noise will be drawn from $[0, max_delta]$ otherwise see details
- `...`: currently not used

**Details**

When no `max_delta` is supplied, `add_noise` will use the minimum difference larger than zero divided by the `length(x)`. 
Value

numeric vector/matrix with noise applied.

ErrorLocalizer-class  Base class for class locate errors based on rules and data

Description

ErrorLocalizer can be used as a base class to implement a new error localization algorithm. The derived class must implement two methods: initialize, which is called before any error localization is done and locate which operates upon data. The extra parameter ... can used to supply algorithmic specific parameters.

errorlocation-class  Error location object

Description

Errorlocation contains the result of a error detection. Errors can record based or variable based.

• A record based error is restricted within one observation. errorlocate using the Felligi Holt algorithm assumes errors are record based.

• A variable based error is a flaw in uni- or multivariate distribution. To correct this error multiple observations or the aggregated number should be adjusted.

Details

Current implementation assumes that errors are record based. The error locations can be retrieved using the method values and are a matrix of rows and columns, with the same dimensions are the data.frame that was checked. For errors that are purely column based, or dataset based, error-locations will return a matrix with all rows or cells set to TRUE. The values return NA for missing values.

Fields

• $errors: matrix indicating which values are erroneous (TRUE), missing (NA) or valid (FALSE)

• $weight: The total weight per record. A weight of 0 means no errors were detected.
**errors_removed**

Get location of removed errors from a ‘cleaned’ data set

**Description**

errors_removed retrieves the errors detected by replace_errors

**Usage**

errors_removed(x, ...)

**Arguments**

- **x**
  - data.frame that was checked for errors
- **...**
  - not used

**Value**

errorlocation-class object

---

**expect_values**

**Description**

expect values

**Usage**

expect_values(values, weights, ...)

**Arguments**

- **values**
  - named list of values.
- **weights**
  - named numeric of equal length as values.
- **...**
  - not used
FHLocalizer-class  Feligi-Holt ErrorLocalizer

Description

Implementation of the Feligi-Holt algorithm using the ErrorLocalizer base class. Given a set of validation rules and a dataset the Feligi-Holt algorithm finds for each record the smallest (weighted) combination of variables that are erroneous (if any).

Note

Most users do not need this class and can use locate_errors.

ErrorLocalizer implements feligi holt using a MIP-solver. For problems in which coefficients of the validation rules or the data are too different, you should consider scaling the data.

is_categorical  Check if rules are categorical

Description

Check if rules are categorical

Usage

is_categorical(x, ...)

Arguments

x  
validator or expression object

...  
not used

Value

logical indicating which rules are purely categorical/logical

Examples

v <- validator( A %in% c("a1", "a2")
    , B %in% c("b1", "b2")
    , if (A == "a1") B == "b1"
    , y > x
    )

is_categorical(v)
is_conditional

is_conditional

Check if rules are conditional rules

Description
Check if rules are conditional rules

Usage
is_conditional(rules, ...)

Arguments
rules
validator object containing validation rules

Value
logical indicating which rules are conditional

Examples
v <- validator( A %in% c("a1", "a2")
, B %in% c("b1", "b2")
, if (A == "a1") x > 1 # conditional
, if (y > 0) x >= 0 # conditional
, if (A == "a1") B == "b1" # categorical
)

is_conditional(v)

is_linear

is_linear

Check which rules are linear rules.

Description
Check which rules are linear rules.

Usage
is_linear(x, ...)

Arguments
x
validator object containing data validation rules

Value
not used

Examples


locate_errors

Locate errors in data

Description
Locate erroneous fields in rows of data using validation rules or a specific errorlocalizer object. This method returns found errors, according to the specified method x. If these errors are to be removed automatically use method replace_errors.

Usage
locate_errors(data, x, ..., timeout = 60)

## S4 method for signature 'data.frame,validator'
locate_errors(data, x, weight = NULL,
              ref = NULL, ..., timeout = 60)

## S4 method for signature 'data.frame,ErrorLocalizer'
locate_errors(data, x,
              weight = NULL, ref = NULL, ..., timeout = 60)

Arguments
data data to be checked
x validation rules or errorlocalizer object to be used for finding possible errors.
... optional parameter to be used by a specific method
timeout maximum number of seconds that the localizer should use per record.
weight numeric optional weight vector to be used in the error localization.
ref data.frame optional reference data to be used in the rules checking

Value
errorlocation-class object describing the errors found.

Examples
rules <- validator ( profit + cost == turnover
, cost - 0.6*turnover >= 0
, cost>= 0
, turnover >= 0
)
data <- data.frame(profit=755, cost=125, turnover=200)
le <- locate_errors(data, rules)
MipRules-class

Create a mip object from a validator object

Description

Create a mip object from validator object. This is a utility class that translates a validator object into a mixed integer problem that can be solved. Most users should use locate_errors which will handle all translation and execution automatically. This class is provided so users can implement or derive an alternative solution.

Methods

The MipRules class contains the following methods:

- $execute calls the mip solver to execute the rules.
- $to_lp: transforms the object into a lp_solve object
- $is_infeasible Checks if the current system of mixed integer rules is feasible.
- $set_values: set values and weights for variables (determines the objective function).
**Examples**

```r
rules <- validator(x > 1)
mr <- miprules(rules)
mr$to_lp()
mr$set_values(list(x=0, weight=list(x=1)))
mr$execute()
```

---

**replace_errors**

*Replace erroneous fields with NA or a suggested value*

**Description**

Find erroneous fields using `locate_errors` and replace these fields automatically with NA or a suggestion that is provided by the error detection algorithm.

**Usage**

```r
replace_errors(data, x, ref = NULL, ..., value = c("NA", "suggestion"))
```

```r
## S4 method for signature 'data.frame,validator'
replace_errors(data, x, ref = NULL, ..., value = c("NA", "suggestion"))
```

```r
## S4 method for signature 'data.frame,ErrorLocalizer'
replace_errors(data, x, ref = NULL, ..., value = c("NA", "suggestion"))
```

```r
## S4 method for signature 'data.frame,errorlocation'
replace_errors(data, x, ref = NULL, ..., value = c("NA", "suggestion"))
```

**Arguments**

- `data`  
  data to be checked
- `x`  
  `validator` object
- `ref`  
  optional reference data set
- `...`  
  these parameters are handed over to `locate_errors`
- `value`  
  NA

**Value**

data with erroneous values removed.
Note

In general it is better to replace the erroneous fields with NA and apply a proper imputation methods. Suggested values from the error localization method may introduce an unwanted bias.

The errors that were removed from the data.frame can be retrieved with the function `errors_removed`. For more control over error localization see `locate_errors`.

See Also

`errorlocation-class`

Examples

```r
library(magrittr)

rules <- validator( profit + cost == turnover
                     , cost - 0.6*turnover >= 0
                     , cost>= 0
                     , turnover >= 0
)
data <- data.frame(profit=755, cost=125, turnover=200)
data_no_error <-
data  %>%
  replace_errors(rules)
# faulty data was replaced with NA
data_no_error

errors_removed(data_no_error)
# a bit more control
error_locations <- locate_errors(data, rules)
data  %>%
  replace_errors(error_locations)
```

substitute_  substitute an existing language object

Description

substitute an existing language object

Usage

`substitute_(x, values = list())`
translate_mip_lp

Arguments

x expression or language object!
values list of values

translate_mip_lp translate linear rules into an lp problem

Description

translate linear rules into an lp problem

Usage

translate_mip_lp(rules, objective = NULL, eps = 0.001, ...)

Arguments

rules mip rules
objective function
eps accuracy for equality/inequality
... additional lp.control parameters that are set for the mip problem
Index

add_noise, 3
create_errorlocation (errorlocation-class), 4
ErrorLocalizer-class, 4
errorlocate, 4
errorlocate (errorlocate-package), 2
errorlocate-package, 2
errorlocation-class, 4
errors_removed, 5, 11
expect_values, 5

fh_localizer (FHLocalizer-class), 6
FHLocalizer-class, 6

is_categorical, 6
is_conditional, 7
is_linear, 7

locate_errors, 6, 8, 9–11
locate_errors, data.frame, ErrorLocalizer-method (locate_errors), 8
locate_errors, data.frame, validator-method (locate_errors), 8
lp.control, 12

miprules (MipRules-class), 9
MipRules-class, 9

replace_errors, 5, 8, 10
replace_errors, data.frame, ErrorLocalizer-method (replace_errors), 10
replace_errors, data.frame, errorlocation-method (replace_errors), 10
replace_errors, data.frame, validator-method (replace_errors), 10

substitute_, 11

translate_mip_lp, 12