# Package ‘errorlocate’

March 30, 2018

**Type** Package

**Title** Locate Errors with Validation Rules

**Version** 0.1.3

**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, magrittr, covr

**Collate** 'addnoise.R' 'categorical.R' 'conditional.R' 'mip.R' 'errorlocalizer.R' 'errorlocate-package.r' 'errorlocation.R' 'expr_manip.R' 'expr_simplify.R' 'linear.r' 'locate-errors.R' 'mip_lpsolve.R' 'mip_rule.R' 'replace-errors.R' 'soft-rule.R' 'values.R'

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Author** Edwin de Jonge [aut, cre],
Mark van der Loo [aut]

**Maintainer** Edwin de Jonge <edwindjonge@gmail.com>

**Repository** CRAN

**Date/Publication** 2018-03-30 12:25:05 UTC

## R topics documented:

---

`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.

## Author(s)

**Maintainer**: Edwin de Jonge `<edwindjonge@gmail.com>`

Authors:

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

### References

T. De Waal (2003) Processing of Erroneous and Unsafe Data. PhD thesis, University of Rotterdam.

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

E. De Jonge and Van der Loo, M. (2012) Error localization as a mixed-integer program in editrules.

lp_solve and Kjell Konis. (2011). lpSolveAPI: R Interface for lp_solve version 5.5.2.0. R package version 5.5.2.0-5. http://CRAN.R-project.org/package=lpSolveAPI

### See Also

Useful links:

- https://github.com/data-cleaning/errorlocate
- Report bugs at https://github.com/data-cleaning/errorlocate/issues

---

## `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

```
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 algoritmic 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, errorlocations will return a matrix with all rows or cells set to `TRUE`. The `values` return `NA` for missing values.

### Fields

- `$errors`: `matrix` indicating which values are erronuous (`TRUE`), missing (`NA`) or valid (`FALSE`)
- `$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` *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 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 *Check if rules are conditional rules*

### Description

Check if rules are conditional rules

### Usage

```
is_conditional(rules, ...)
```

### Arguments

| | |
|---|---|
| rules | validator object containing validation rules |
| ... | not used |

### 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 *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 |
| ... | not used |

## Value

`logical` indicating which rules are (purely) linear.

---

locate_errors *Locate errors in data*

---

## Description

Locate erronuous 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, ...)

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

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

## 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 |
| `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)

print(le)
```

```
summary(le)

v_categorical <- validator( A %in% c("a1", "a2")
                          , B %in% c("b1", "b2")
                          , if (A == "a1") B == "b1"
)

data <- data.frame(A = c("a1", "a2"), B = c("b2", "b2"))
locate_errors(data, v_categorical)

v_logical <- validator( A %in% c(TRUE, FALSE)
                      , B %in% c(TRUE, FALSE)
                      ,  if (A == TRUE) B == TRUE
                      )

data <- data.frame(A = TRUE, B = FALSE)
locate_errors(data, v_logical, weight=c(2,1))

# try a condinational rule
v <- validator( married %in% c(TRUE, FALSE), if (married==TRUE) age >= 17 )
data <- data.frame( married = TRUE, age = 16)
locate_errors(data, v, weight=c(married=1, age=2))
```

---

`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 validor 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 a 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

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

### Usage

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

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

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

## 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 erronous values removed.

### Note

In general it is better to replace the erronuous 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

```
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())
```

## 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 |

# Index