Package ‘validata’

October 12, 2022

Title  Validate Data Frames
Version  0.1.0
Maintainer  Harrison Tietze <Harrison4192@gmail.com>
Description  Functions for validating the structure and properties of data frames. Answers essential questions about a data set after initial import or modification. What are the unique or missing values? What columns form a primary key? What are the properties of the numeric or categorical columns? What kind of overlap or mapping exists between 2 columns?
License  MIT + file LICENSE
URL  https://harrison4192.github.io/validata/,
     https://github.com/Harrison4192/validata
BugReports  https://github.com/Harrison4192/validata/issues
Encoding  UTF-8
LazyData  true
RoxygenNote  7.1.1
Imports  dplyr, stringr, janitor, rlang, tidyselect, purrr, magrittr,
         tidyr, tibble, gtools, listviewer, data.table, scales, utils,
         BBmisc, framecleaner, badger, rlist
Suggests  knitr, rmarkdown, testit, webshot
VignetteBuilder  knitr
Depends  R (>= 2.10)
NeedsCompilation  no
Author  Harrison Tietze [aut, cre]
Repository  CRAN
Date/Publication  2021-10-05 08:20:02 UTC

R topics documented:

  confirm_distinct ................................................. 2
  confirm_mapping .................................................. 3
confirm_distinct

Description

Confirm whether the rows of a data frame can be uniquely identified by the keys in the selected columns. Also reports whether the dataframe has duplicates. If so, it is best to remove duplicates and re-run the function.

Usage

confirm_distinct(.data, ...)

Arguments

.data A dataframe

... (ID) columns

Value

da Logical value invisibly with description printed to console

Examples

iris %>% confirm_distinct(Species, Sepal.Width)
confirm_mapping

Confirm structural mapping between 2 columns

Description
The mapping between elements of 2 columns can have 4 different relationships: one - one, one -
many, many - one, many - many. This function returns a view of the mappings by row, and prints a
summary to the console.

Usage

confirm_mapping(.data, col1, col2, view = T)

Arguments

.data          a data frame
col1           column 1
col2           column 2
view           View results?

Value
A view of mappings. Also returns the view as a data frame invisibly.

Examples

iris %>% confirm_mapping(Species, Sepal.Width, view = FALSE)

close

confirm_overlap

Confirm Overlap

Description
Prints a venn-diagram style summary of the unique value overlap between two columns and also
invisibly returns a dataframe that can be assigned to a variable and queried with the overlap helpers.
The helpers can return values that appeared only the first col, second col, or both cols.

Usage

confirm_overlap(vec1, vec2, return_tibble = F)

co_find_only_in_1(co_output)

co_find_only_in_2(co_output)

co_find_in_both(co_output)
confirm_strlen

**Argument**

- **vec1**: vector 1
- **vec2**: vector 2
- **return_tibble**: logical. If TRUE, returns a tibble. otherwise by default returns the database invisibly to be queried by helper functions.
- **co_output**: dataframe output from confirm_overlap

**Value**

tibble. overlap summary or overlap table

**Examples**

```r
confirm_overlap(iris$Sepal.Width, iris$Sepal.Length) -> iris_overlap
iris_overlap
iris_overlap %>%
  co_find_only_in_1()
iris_overlap %>%
  co_find_only_in_2()
iris_overlap %>%
  co_find_in_both()
```

---

**confirm_strlen**

**confirm string length**

**Description**

returns a count table of string lengths for a character column. The helper function choose_strlen filters dataframe for rows containing specific string length for the specified column.

**Usage**

```r
confirm_strlen(mdb, col)
choose_strlen(cs_output, len)
```

**Arguments**

- **mdb**: dataframe
- **col**: unquoted column
- **cs_output**: dataframe. output from confirm_strlen
- **len**: integer vector.
**Determine Distinct**

**Value**

prints a summary and returns a dataframe invisibly
dataframe with original columns, filtered to the specific string length

**Examples**

```r
data(iris)
iris %>%
tibble::as_tibble() %>%
confirm_strlen(Species) -> iris_cs_output

iris_cs_output

iris_cs_output %>%
choose_strlen(6)
```

**Description**

Uses confirm_distinct in an iterative fashion to determine the primary keys.

**Usage**

```r
determine_distinct(df, ..., listviewer = TRUE)
```

**Arguments**

- `df` a data frame
- `...` columns or a tidyselect specification. defaults to everything
- `listviewer` logical. defaults to TRUE to view output using the listviewer package

**Details**

The goal of this function is to automatically determine which columns uniquely identify the rows of a dataframe. The output is a printed description of the combination of columns that form unique identifiers at each level. At level 1, the function tests if individual columns are primary keys. At level 2, the function tests \( \binom{n}{2} \) combinations of columns to see if they form primary keys. The final level is testing all columns at once.

- For completely unique columns, they are recorded in level 1, but then dropped from the dataframe to facilitate the determination of multi-column primary keys.
- If the dataset contains duplicated rows, they are eliminated before proceeding.

**Value**

list
determine_mapping

**Examples**

```r
sample_data1 %>%
  head

## on level 1, each column is tested as a unique identifier. the VAL columns have no
## duplicates and hence qualify, even though they normally would be considered as IDs
## on level 3, combinations of 3 columns are tested. implying that ID_COL 1,2,3 form a unique key
## level 2 does not appear, implying that combinations of any 2 ID_COLs do not form a unique key

sample_data1 %>%
determine_distinct(listviewer = FALSE)
```

determine_mapping  
*Determine pairwise structural mappings*

**Description**

Determine pairwise structural mappings

**Usage**

```r
determine_mapping(df, ..., listviewer = TRUE)
```

**Arguments**

- `df`  
  a data frame
- `...`  
  columns or a tidyselect specification
- `listviewer`  
  logical. defaults to TRUE to view output using the listviewer package

**Value**

description of mappings

**Examples**

```r
iris %>%
determine_mapping(listviewer = FALSE)
```
**determine_overlap**  
*Determine Overlap*

**Description**

Uses `confirm_overlap` in a pairwise fashion to see venn style comparison of unique values between the columns chosen by a tidyselect specification.

**Usage**

```r
determine_overlap(db, ...)
```

**Arguments**

- `db`: a data frame
- `...`: tidyselect specification. Default being everything.

**Value**

tibble

**Examples**

```r
iris %>%
determine_overlap()
```

---

**diagnose**  
*diagnose*

**Description**

This function is inspired by the excellent `dlookr` package. It takes a dataframe and returns a summary of unique and missing values of the columns.

**Usage**

```r
diagnose(df, ...)
```

**Arguments**

- `df`: dataframe
- `...`: tidyselect
Value
dataframe summary

Description
counts the distinct entries of categorical variables. The \texttt{max\_distinct} argument limits the scope to categorical variables with a maximum number of unique entries, to prevent overflow.

Usage
diagnose\_category(.\texttt{data}, ..., \texttt{max\_distinct} = 5)

Arguments
\begin{itemize}
  \item \texttt{.data} \hspace{1cm} \texttt{dataframe}
  \item \texttt{...} \hspace{1cm} \texttt{tidyselect}
  \item \texttt{max\_distinct} \hspace{1cm} \texttt{integer}
\end{itemize}

Examples
diagnose\_category()
**diagnose_missing**

Description

faster than diagnose if emphasis is on diagnosing missing values. Also, only shows the columns with any missing values.

Usage

```r
diagnose_missing(df, ...)
```

Arguments

- `df` dataframe
- `...` optional tidyselect

Value
tibble summary

Examples

```r
iris %>%
  framecleaner::make_na(Species, vec = "setosa") %>%
  diagnose_missing()
```

**diagnose_numeric**

Description

Inputs a dataframe and returns various summary statistics of the numeric columns. For example zeros returns the number of 0 values in that column. minus counts negative values and infs counts Inf values. Other rarer metrics are also returned that may be helpful for quick diagnosis or understanding of numeric data. mode returns the most common value in the column (chooses at random in case of tie), and mode_ratio returns its frequency as a ratio of the total rows

Usage

```r
diagnose_numeric(.data, ...)
```

Arguments

- `.data` dataframe
- `...` tidyselect
Value
dataframe

Examples

library(framecleaner)

iris %>%
diagnose_numeric

view_missing

Description

View rows of the dataframe where columns in the tidyselect specification contain missings by default, detects missings in any column. The result is by default displayed in the viewer pane. Can be returned as a tibble optionally.

Usage

view_missing(df, ..., view = TRUE)

Arguments

<table>
<thead>
<tr>
<th>df</th>
<th>dataframe</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>tidyselect</td>
</tr>
<tr>
<td>view</td>
<td>logical. if false, returns tibble</td>
</tr>
</tbody>
</table>

Value
tibble

Examples

iris %>%
framecleaner::make_na(Species, vec = "setosa") %>%
view_missing(view = FALSE)
Index

choose_strlen(confirm_strlen), 4
co_find_in_both(confirm_overlap), 3
c_0nd_only_in_1(confirm_overlap), 3
c_0nd_only_in_2(confirm_overlap), 3
c_0nfirm_distinct, 2
c_0nfirm_mapping, 3
c_0nfirm_overlap, 3
c_0nfirm_strlen, 4
determine_distinct, 5
determine_mapping, 6
determine_overlap, 7
diagnose, 7
diagnose_category, 8
diagnose_missing, 9
diagnose_numeric, 9
view_missing, 10