Package 'cleaninginspectoR'

December 10, 2019

Title Basic checks that data cleaning ocurred

Version 0.0.0.9000

Description This package checks that data cleaning ocurred according to IMPACT standards. It requires the user to upload the data, cleaning log, and fill out several parameters

License What license it uses

Encoding UTF-8

LazyData true

Suggests testthat,
rmarkdown

VignetteBuilder knitr

RoxygenNote 6.1.1

Roxygen list(markdown = TRUE)

Depends reshape,
knitr,
dplyr

Imports purrr,
questionr,
data.table,
assertthat,
tidyr,
magrittr

R topics documented:

check_time *Check kobo interview time*

Description

Check kobo interview time

Usage

```
check_time(data, duration_threshold_lower = 15,
  duration_threshold_upper = 100)
```

Arguments

data	Dataframe with "start" and "end" columns.
duration_threshold_lower	Minimum number of minutes to complete the form
duration_threshold_upper	Maximum number of minutes to complete the form

Value

A dataframe with one row per potential issue. It has columns for the corresponding row index in the original data; the suspicious value (survey that is longer or shorter than it should be); the variable name in the original dataset in which the suspicious value occured; A description of the issue type.

cleaninginspectoR *Data Cleaning Checks rudimentary checks to flag potentially problematic values in a dataset*

Description

Data Cleaning Checks rudimentary checks to flag potentially problematic values in a dataset

Details

Functionality

`inspect_all()`: applies all checks listed below `find_duplicates()` looks for duplicates in columns that should be unique `find_duplicates_uuid()` looks for a function containing "uuid" in the name and looks for duplicates `find_outliers()` looks for outliers in numerical columns `find_other_responses()` looks for values in potential "if other, specify.." type of columns

Output The output has the same structure for all functions of this package: A data frame with the following columns: value variable has_issue issue_type

`index`: the index of the row of the original data in which the issue occured. (NA if applies to multiple rows) `value` the suspicious value `variable` The column containing the suspicious value `has_issue` logical TRUE/FALSE: currently always true, can be ignored. `issue_type` a description/name of the potential issue

Limitations

1. These checks are under no circumstances sufficient or complete in any way, and more context and data specific checks are always necessary.
2. Any data protection related checks are *rudimentary* at best and are under no circumstances sufficient to ensure data protection in any way
3. Although tempting, this package should *never* be used to automatically remove any flagged values without double checking them manually. This would seriously skew your dataset, and make uncertainty estimates invalid. Potential issues must be investigated on a case by case basis.

`find_duplicates` *Find duplicates / non-unique values in a variable*

Description

Find duplicates / non-unique values in a variable

Usage

```
find_duplicates(data, duplicate.column.name)
```

Arguments

`data`	a dataframe
`duplicate.column.name`	the name of the column the dataframe to be checked for duplicates as a string (in quotes)

Value

A dataframe with one row per potential issue. It has columns for the corresponding row index in the original data; the suspicious value; the variable name in the original dataset in which the suspicious value occured; A description of the issue type.

Examples

```
# a test dataset with 1000 rows; one numeric variable and one id variable
testdf <- data.frame(numeric_var = runif(10), unique_ids = c(1, 2, 3, 4, 5, 6, 7, 8, 1, 3))
# find duplicates in the unique_ids column:
find_duplicates(data, "unique_ids")
```

find_duplicates_uuid *Search UUID column, then find duplicates / non-unique values in it*

Description

Search UUID column, then find duplicates / non-unique values in it

Usage

```
find_duplicates_uuid(data)
```

Arguments

data	a dataframe

Details

searches for "uuid" (not case sensitive) in the variable names. Identifies duplicate values in the first variable that matches the search. This function uses the more generic `find_duplicates()` function, which you should use if your id column doesn't contain "uuid"

Value

A dataframe with one row per potential issue. It has columns for the corresponding row index in the original data; the suspicious value; the variable name in the original dataset in which the suspicious value occured; A description of the issue type.

Examples

```
# a test dataset with 1000 rows; one numeric variable and one id variable
testdf <- data.frame(numeric_var = runif(10), unique_ids = c(1, 2, 3, 4, 5, 6, 7, 8, 1, 3))
# find duplicates in the unique_ids column:
find_duplicates_uuid(data)
```

find_other_responses *Find all responses in all columns that might be "specify other" responses to a multiple choice question*

Description

Find all responses in all columns that might be "specify other" responses to a multiple choice question

Usage

```
find_other_responses(data)
```

Arguments

data	a dataframe

Details

Performs a non-case sensitive search for "other" in english and french along the column names of the dataframe and returns all unique values and their frequency.

Value

A dataframe with one row per potential issue. It has columns for the corresponding row index in the original data; the suspicious value; the variable name in the original dataset in which the suspicious value occured; A description of the issue type.

---

find_outliers *Find outliers in all numerical columns of a dataset*

Description

Find outliers in all numerical columns of a dataset

Usage

```
find_outliers(data)
```

Arguments

data	a dataframe

Details

Searches for values that are outside more than three standard deviations from the mean. If fewer outliers are found when the data is log-transformed before the check, only outliers in the log-transformed data are returned.

Value

A dataframe with one row per potential issue. It has columns for the corresponding row index in the original data; the suspicious value; the variable name in the original dataset in which the suspicious value occured; A description of the issue type.

---

hasdata *has data removes NA, empty strings and non-finite values from a vector*

Description

has data removes NA, empty strings and non-finite values from a vector

Usage

```
hasdata(x, return.index = F)
```

Arguments

x	vector
return.index	if true, returns indices of the vector that have valid data. Defaults to FALSE.

Value

returns the values of the input vector that contain valid data

---

inspect_all *Apply general data cleaning checks*

Description

Uses all other cleaning checks available in this package at once

Usage

```
inspect_all(df, uuid.column.name = NULL)
```

Arguments

uuid.column.name	optional: The name of the column containing the uuids. If none is provided, will seach variable names for "uuid".
data	a dataframe

Details

for details see ?cleaninginspectoR

Value

A dataframe with one row per potential issue, containing the row index, value and variable name in the original dataset, as well as a description of the issue type

Examples

```
inspect_all(my_df)
```

inspect_all_csv_in_dir

run inspect_all() on all csv files in a folder and subfolders

Description

run inspect_all() on all csv files in a folder and subfolders

Usage

```
inspect_all_csv_in_dir(source_dir = "./", pattern = "csv$",
  target_dir = "./", recursive = TRUE)
```

Arguments

source_dir	folder to search for csv files that inspect_all() should be run on
pattern	a regex pattern on which files in the folders to select. The default is "csv$"; this should be kept at the end of the pattern
recursive	logical: if TRUE (default), also searches all subfolders of the current working directory
target_folder	path to the folder where the issue tables should be saved

Value

a list of data frames with the outputs from each csv file in the current working directory (and subdirectories)#

inspect_all_in_folder *run inspect_all() on all csv files in a folder and subfolders*

Description

run inspect_all() on all csv files in a folder and subfolders

Usage

```
inspect_all_in_folder(source_dir = "./", pattern = "csv$",
  recursive = TRUE, write.to.csv = FALSE, target_dir = "./")
```

Arguments

source_dir	folder to search for csv files that inspect_all() should be run on
pattern	a regex pattern on which files in the folders to select. The default is "csv$"; this should be kept at the end of the pattern
recursive	logical: if TRUE (default), also searches all subfolders of the current working directory
write.to.csv	logical: whether or not to write csv files with the issue tables to files (folder can be specified with target_dir)
target_folder	path to the folder where the issue tables should be saved

Value

a list of data frames with the outputs from each csv file in the current working directory (and subdirectories)#

`sensitive_columns` *Search column names for words often used in senstive variables*

Description

Search column names for words often used in senstive variables

Usage

```
sensitive_columns(data, i.know.this.check.is.insufficient = F)
```

Arguments

`data`	a dataframe
`i.know.this.check.is.insufficient`	optional: if not set to TRUE, this function throws a warning.

Details

Searches column headers for keywords "gps", "phone","latitude", "longitude" and "phone" (not case sensitive) WARNING: this check is rudimentary and does not suffice in any way to insure protection of sensitive information.

Value

A dataframe with one row per potential issue. It has columns for the corresponding row index in the original data; the suspicious value; the variable name in the original dataset in which the suspicious value occured; A description of the issue type.

Index