Package 'trustmebro'

May 9, 2025

Type Package

Title Inspect and Clean Subject-Generated ID Codes and Related Data

Version 1.0.0

Maintainer Annemarie Pläschke <anneplaeschke@gmail.com>

Description Makes data wrangling with ID-related aspects more comfortable.
Provides functions that make it easy to inspect various subject-generated ID codes (SGIC) for plausibility.
Also helps with inspecting other common identifiers, ensuring that your data stays clean and reliable.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

Imports dplyr, tibble, rlang

VignetteBuilder knitr

URL https://kuuuwe.github.io/trustmebro/,
https://github.com/kuuuwe/trustmebro

Language en-US

BugReports https://github.com/kuuuwe/trustmebro/issues

NeedsCompilation no

Author Annemarie Pläschke [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0005-7115-8790>),
Tobias Brändle [aut] (ORCID: <https://orcid.org/0000-0001-8872-9872>)

Repository CRAN

Date/Publication 2025-05-09 14:10:02 UTC

Contents

`find_dupes` *Identify duplicate cases*

Description

Identify duplicate cases in a data frame or tibble based on specific variables. A logical column 'has_dupes' is added, that indicates whether or not a row has duplicate values based on the provided variables.

Usage

```
find_dupes(data, ...)
```

Arguments

`data`	A data frame or tibble
`...`	Variable names to check for duplicates

Value

The original data frame or tibble with an additional logical column 'has_dupes' which is 'TRUE' for rows that have duplicates based on the specified variables and 'FALSE' otherwise.

Examples

```
# Example data
print(sailor_students)

# Find duplicate cases based on 'sgic', 'school' and 'class'
sailor_students_dupes <- find_dupes(sailor_students, sgic, school, class)

# Rows where 'has_dupes' is `TRUE` indicate duplicates based on the provided columns
print(sailor_students_dupes)
```

inspect_birthday *Inspect birthday-component of a string*

Description

Check whether a given string contains exactly one two-digit number that represents a valid day of the month (between 01 and 31). The string is assumed to be a code (e.g., a SGIC), which may include letters and digits.

Usage

```
inspect_birthday(code)
```

Arguments

code	A character string containing a SGIC or similar code that may include a numeric birthday-component.

Value

A logical value: ‘TRUE‘ if the string contains only one valid birthday-component (between 01 and 31), otherwise ‘FALSE‘.

Examples

```
inspect_birthday("DEF66") # FALSE - 66 is not a valid day
inspect_birthday("GHI02") # TRUE - 02 is a valid day
inspect_birthday("ABC12DEF34") # FALSE - Multiple numeric components
inspect_birthday("XYZ") # FALSE - No numeric component
inspect_birthday("JKL31") # TRUE - 31 is a valid day
```

inspect_birthdaymonth *Inspect birthday- and birthmonth-component of a string*

Description

Checks whether a given string contains exactly one four-digit number representing a valid combination of a day (birthday) and a month (birth month). Numeric components can be interpreted in either "DDMM" (day-month) or "MMDD" (month-day) format, depending on the specified format. The string is assumed to be a code (e.g., a SGIC), which may include letters and digits.

Usage

```
inspect_birthdaymonth(code, format = "DDMM")
```

Arguments

code	A character string containing a SGIC or similar code that may include a numeric component representing a birthday and birth month.
format	A string specifying the format of the date of birth components in code. Use "DDMM" for day-month format and "MMDD" for month-day format. Default is "DDMM".

Value

A logical value: 'TRUE' if the string contains exactly one valid numeric component that forms a valid birthday (day and month), otherwise 'FALSE'.

Examples

```
inspect_birthdaymonth("DEF2802") # TRUE - 28th of February is a valid date
inspect_birthdaymonth("GHI3002") # FALSE - 30th of February is invalid
inspect_birthdaymonth("XYZ3112") # TRUE - 31st of December is valid
inspect_birthdaymonth("18DEF02") # FALSE - Multiple numeric components
inspect_birthdaymonth("XYZ") # FALSE - No numeric components
inspect_birthdaymonth("ABC1231", format = "MMDD") # TRUE - December 31st is valid
```

---

inspect_birthmonth *Inspect birthmonth-component of a string*

Description

Check whether a given string contains exactly one two-digit number that represents a valid month of the year (between 01 and 12). The string is assumed to be a code (e.g., a SGIC), which may include letters and digits.

Usage

```
inspect_birthmonth(code)
```

Arguments

code	A character string containing a SGIC or similar code that may include a numeric birth month-component.

Value

A logical value: 'TRUE' if the string contains only one valid birth month-component (between 01 and 12), otherwise 'FALSE'.

Examples

```
inspect_birthday("DEF66") # FALSE - 66 is not a valid month
inspect_birthday("GHI02") # TRUE - 02 (February) is a valid month
inspect_birthday("ABC12DEF10") # FALSE - Multiple numeric components
inspect_birthday("XYZ") # FALSE - No numeric component
inspect_birthday("JKL11") # TRUE - 11 (November) is a valid day
```

---

inspect_characterid *Inspect if a string matches an expected pattern*

Description

Check whether a given string matches a specified pattern using regular expressions (regex). The string is assumed to be a code (e.g., a SGIC), which should follow a predefined format.

Usage

```
inspect_characterid(code, pattern)
```

Arguments

code	A character string containing a SGIC or similar code that should follow a predefined format.
pattern	A character string specifying the expected pattern using regular expressions (regex). The pattern defines the format 'code' should match.

Value

A logical value: 'TRUE' if the code matches the expected pattern, otherwise 'FALSE'

Examples

```
inspect_characterid("ABC1234", "^[A-Za-z]{3}[0-9]{4}$") #TRUE - Matches the pattern
inspect_characterid("12DBG45FG", "^[A-Za-z]{3}[0-9]{4}$") #FALSE - Does not match the pattern
```

---

inspect_numberid *Inspect if a number has the expected length*

Description

Check whether a given numeric value has the expected number of digits.

Usage

```
inspect_numberid(number, expected_length)
```

Arguments

number	A numeric value.
expected_length	An integer specifying the expected number of digits.

Value

A logical value: ‘TRUE‘ if ‘number‘ has the expected length and consists only of digits, otherwise ‘FALSE‘.

Examples

```
inspect_numberid(12345, 5)  # TRUE - 5 digits
inspect_numberid(1234, 5)    # FALSE - 4 digits
```

---

inspect_valinvec *Inspect if a value is in a recode map*

Description

Check whether a given value is present as a key in a specified recode map. Inputs can be validated against a set of predefined categories or labels.

Usage

```
inspect_valinvec(value, recode_map)
```

Arguments

value	A single value to inspect, which is checked against the keys of a recode map.
recode_map	A named vector where the names represent the keys to check against. The values of the vector are ignored.

Value

A logical value: ‘TRUE‘ if the ‘value‘ is a key in the ‘recode_map‘, otherwise ‘FALSE‘.

Examples

```
recode_map <- c(male = "M", female = "F")
inspect_valinvec("female", recode_map) # TRUE - "female" is a key in the recode map
inspect_valinvec("other", recode_map) # FALSE - "other" is not a key in the recode map
```

purge_string *Purge strings in a data frame*

Description

Clean specified character columns in a data frame or tibble by removing non-alphanumeric characters, replacing them with a specified character (default is "#"). Also replaces NA values and allows for additional characters to keep in the cleaned strings. The resulting strings are converted to uppercase.

Usage

```
purge_string(data, ..., replacement = "#", keep = "")
```

Arguments

data	A data frame or tibble containing columns to be cleaned.
...	Variables to clean. If none are provided, all character columns will be processed.
replacement	A character string used to replace unwanted characters and empty strings. Default is "#".
keep	A character string containing any additional characters that should be retained in the cleaned strings.

Value

A data frame or tibble with the specified character columns cleaned and modified as per the given parameters.

Examples

```
# Example data
print(sailor_students)

# Clean all character columns, replacing unwanted characters with "#", retaining "-"
sailor_students_cleaned <-
purge_string(sailor_students, sgic, school, class, gender, keep = "-")

# Tibble with cleaned 'sgic', 'school', 'class' and 'gender' columns
print(sailor_students_cleaned)
```

`recode_valinvec` *Recode a variable*

Description

Recode a specified variable in a data frame or tibble based on a provided recode map. If the recode map is empty, the original variable is retained under a new name.

Usage

```
recode_valinvec(data, var, recode_map, new_var)
```

Arguments

`data`	A data frame or tibble.
`var`	A variable to be recoded.
`recode_map`	A named vector specifying the recode map.
`new_var`	Name of the new variable holding the recoded values.

Value

A data frame or tibble with the new variable added.

Examples

```
# Example data
print(sailor_students)

# Define a recode map for gender
recode_map_gender <- c("Female" = "F", "Male" = "M", "Other" = "X")

# Recode gender
sailor_students_recoded <-
recode_valinvec(sailor_students, gender, recode_map_gender, recode_gender)

# A tibble with a recoded gender variable
print(sailor_students_recoded)
```

sailor_keys *key data on students from the sailor moon universe*

Description

A fictional key data set.

Usage

```
sailor_keys
```

Format

'sailor_keys' A tibble with 12 rows and 6 columns:

schoolyear schoolyear

guid hexadecimal ID number

name, birthday, sex student information

school, schoolnumber, class, grade_level school information

sgic1, sgic2, sgic3 subject generated ID

sailor_students *assessment data on students from the sailor moon universe*

Description

A fictional assessment data set.

Usage

```
sailor_students
```

Format

'sailor_students' A tibble with 12 rows and 6 columns:

sgic Subject generated ID

school schoolnumber

class class designation

gender gender

testscore_language, testscore_calculus testscores

Index