Package ‘IPDFileCheck’

June 12, 2020

Type Package

Title Basic Functions to Check Readability, Consistency, and Content of an Individual Participant Data File

Version 0.6.3

Author Sheeja Manchira Krishnan

Maintainer Sheeja Manchira Krishnan <sheejamk@gmail.com>

Description Basic checks needed with an individual level participant data from randomised controlled trial. This
checks files for existence, read access and individual columns for formats. The checks on format is currently implemented for gender and age formats.

Imports dplyr, testthat (>= 1.0.2), GlobalOptions (>= 0.1.0), lubridate, methods, eeptools

License CC0

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-12 16:00:15 UTC

R topics documented:

calculate_age_from_dob

Function to calculate age from date of birth

Description

Function to calculate age from date of birth

Usage

```
calculate_age_from_dob(data, columnname, dateformat = "dmy",
  nrcode = NA)
```

Arguments

data	a data frame
columnname	name of column corresponding to date of birth
dateformat	format of date e.g. dmy default is dmy
nrcode	non response code corresponding to date of birth

Value

data if success error if failure

Examples

```
library(IPDFileCheck)
this.df <- data.frame(c("1987-05-28", "1987-06-18"), c(1, 2), stringsAsFactors = FALSE)
colnames(this.df) <- c("dob", "num")
calculate_age_from_dob(this.df, "dob", "ymd")
```

`calculate_age_from_year` *Function to calculate age from year of birth*

Description

Function to calculate age from year of birth

Usage

```
calculate_age_from_year(data, columnname, nrcode = NA)
```

Arguments

`data`	a data frame
`columnname`	name of column corresponding to year of birth
`nrcode`	non response code corresponding to date of birth

Value

data, if success error if failure

Examples

```
this.data.frame <- data.frame(c(1951, 1980), c("John", "Dora"))
colnames(this.data.frame) <- c("yob", "name")
calculate_age_from_year(this.data.frame, "yob", NA)
```

`check_colno_pattern_colname` *## Function to return the column number if a given pattern is contained in the column names of a data*

Description

Function to return the column number if a given pattern is contained in the column names of a data

Usage

```
check_colno_pattern_colname(pattern, column_names)
```

Arguments

`pattern`	a string that needs to be checked
`column_names`	column names actually have

Value

TRUE , if success FALSE, if failure

Examples

```
check_colno_pattern_colname("age", "female_age")
```

`check_column_exists` *Function to check the given column exists*

Description

Function to check the given column exists

Usage

```
check_column_exists(column_name, data)
```

Arguments

`column_name`	a column name
`data`	data frame

Value

0 if success error if failure

Examples

```
check_column_exists("age", data.frame("Age" = c(21, 15), "Name" = c("John", "Dora")))
```

`check_load_packages` *Function to check the package is installed, if not install*

Description

Function to check the package is installed, if not install

Usage

```
check_load_packages(pkg)
```

Arguments

`pkg`	name of package(s)

Value

0, if packages cant be installed and loaded, else error

Examples

```
check_load_packages("dplyr")
```

`cohensd` *Function to find the effect size Cohen's d*

Description

Function to find the effect size Cohen's d

Usage

```
cohensd(x, y)
```

Arguments

`x,`	a vector
`y,`	another vector

Value

cohens d estimated with 95

Examples

```
cohensd(c(1, 2, 3, 4), c(3, 4, 5, 6))
```

convert_date_numeric_stdform

Helper function to keep date formats in year-month-date

Description

Helper function to keep date formats in year-month-date

Usage

```
convert_date_numeric_stdform(column, index, orderby = "dmy")
```

Arguments

`column` a data frame or a vector

`index` those correspond to valid date in numeic form (omitting non response code or no entry)

`orderby` give the order such as mdy, dmy etc where d refers to day, m to month and y to year

Value

entry corrected entries as in standard date format

Examples

```
convert_date_numeric_stdform(c("01/01/2000", "02/02/2002"), c(1, 2), "dmy")
```

convert_date_numeric_stdform_old

Helper function to keep date formats in year-month-date

Description

Helper function to keep date formats in year-month-date

Usage

```
convert_date_numeric_stdform_old(entry, index, orderby = "dmy")
```

Arguments

`entry` a data frame or a vector

`index` those correspond to valid date in numeic form (omitting non response code or no entry)

`orderby` give the order such as mdy, dmy etc where d refers to day, m to month and y to year

Value

entry corrected entries as in standard date format

Examples

```
convert_date_numeric_stdform(c("01/01/2000", "02/02/2002"), c(1, 2), "dmy")
```

`convert_date_string_stdform` *Helper function to keep date formats in year-month-date*

Description

Helper function to keep date formats in year-month-date

Usage

```
convert_date_string_stdform(entry, orderby)
```

Arguments

`entry`	a date e.g 1 Jan 2020 with no commas
`orderby`	give the order such as mdy, dmy etc where d refers to day, m to month and y to year

Value

entry corrected entries as in standard date format

Examples

```
convert_date_string_stdform("Jan-1-2020", "mdy")
```

`convert_to_number` *Function that convert a number represented as character array*

Description

Function that convert a number represented as character array

Usage

```
convert_to_number(character_array)
```

Arguments

`character_array`
a character array of numbers

Value

converted_number in numeric form

Examples

```
convert_to_number(c("1", "9", "8"))
```

---

descriptive_stats_col *Function to return descriptive statistics, sum, no of observations, mean, mode. median, range, standard deviation and standard error*

Description

Function to return descriptive statistics, sum, no of observations, mean, mode. median, range, standard deviation and standard error

Usage

```
descriptive_stats_col(data, column_name, nrcode = NA)
```

Arguments

`data`	data frame
`column_name`	the column name
`nrcode`	non response code corresponding to the column

Value

the descriptive statistics for success , error for failure

Examples

```
descriptive_stats_col(data.frame("age" = c(21, 15), "Name" = c("John", "Dora")), "age", NA)
```

get_colno_pattern_colname *Function to return the column number if a given pattern is contained in the column names of a data*

Description

Function to return the column number if a given pattern is contained in the column names of a data

Usage

```
get_colno_pattern_colname(pattern, column_names)
```

Arguments

pattern	a string that needs to be checked
column_names	column names actually have

Value

column number, if success error, if failure

Examples

```
get_colno_pattern_colname("age", "female_age")
```

get_columnno_fornames *Function to return the column number for column name*

Description

Function to return the column number for column name

Usage

```
get_columnno_fornames(data, column_name)
```

Arguments

data	a data frame
column_name	column names of the data frame

Value

column number, if success error, if failure

Examples

```
get_columnno_fornames(data.frame("Age" = c(21, 15), "Name" = c("John", "Dora")), "Name")
```

`get_contents_cols` *Function to return the unique contents of the column given the column name*

Description

Function to return the unique contents of the column given the column name

Usage

```
get_contents_cols(data, colname)
```

Arguments

`data`	a data frame
`colname`	name of column corresponding to year of birth

Value

the contents of the column, if success error if failure

Examples

```
get_contents_cols(data.frame(
  "yob" =  c(1951, 1980),
  "Name" =  c("John", "Dora")
), "yob")
```

`get_mode_from_vector` *## Function to return mode*

Description

Function to return mode

Usage

```
get_mode_from_vector(v)
```

Arguments

`v`	a vector

Value

mode

Examples

```
get_mode_from_vector(c(1, 1, 2, 3))
```

`get_sem` *##*
## *Function to estimate standard error of the mean*

Description

Function to estimate standard error of the mean

Usage

```
get_sem(x)
```

Arguments

`x,`	a vector

Value

SE the standard error of the mean

Examples

```
get_sem(c(1, 2, 3, 4))
```

`present_mean_sd_rmna_text` *##*
## *Function to present the mean and sd of a data set in the form Mean (SD)*

Description

Function to present the mean and sd of a data set in the form Mean (SD)

Usage

```
present_mean_sd_rmna_text(data, column_name, nrcode = NA)
```

Arguments

`data`	data frame
`column_name`	the column name
`nrcode`	non response code corresponding to the column

Value

the mean(sd), error for failure

Examples

```
present_mean_sd_rmna_text(data.frame(
  "age" = c(21, 15),
  "Name" = c("John", "Dora")
), "age", NA)
```

represent_categorical_data

Function to find the number and percentages of categories

Description

Function to find the number and percentages of categories

Usage

```
represent_categorical_data(data, variable, nrcode = NA)
```

Arguments

`data,`	a data frame
`variable`	the column name
`nrcode`	non response code

Value

number and percentages or error if failure

Examples

```
this.df <- data.frame(c(11, 78), c("m", "f"), stringsAsFactors = FALSE)
colnames(this.df) <- c("mark", "gender")
represent_categorical_data(this.df, "gender", NA)
```

represent_categorical_textdata

Function to represent categorical data in the form - numbers (percentage)

Description

Function to represent categorical data in the form - numbers (percentage)

Usage

```
represent_categorical_textdata(data, variable, nrcode)
```

Arguments

`data`	data frame
`variable`	column name
`nrcode`	non response code

Value

the numbers (percentage) , error for failure

Examples

```
df <- data.frame(c(11, 78), c("m", "f"), stringsAsFactors = FALSE)
colnames(df) <- c("mark", "gender")
represent_categorical_textdata(df, "gender", NA)
```

return_subgroup_omitna

Function to return a subgroup when certain variable equals the given value while omitting those with NA

Description

Function to return a subgroup when certain variable equals the given value while omitting those with NA

Usage

```
return_subgroup_omitna(data, variable, value)
```

Arguments

`data`	data frame
`variable`	that corresponds to a column
`value`	a value that can be taken by the variable

Value

subgroup a data frame if success error if failure

Examples

```
return_subgroup_omitna(data.frame(
  "age" = c(21, 15),
  "Name" = c("John", "Dora")
), "age", 10)
```

`test_age` *Function to check the format of ’age’ in data*

Description

Function to check the format of ’age’ in data

Usage

```
test_age(data, agecolumn = "age", nrcode = NA)
```

Arguments

`data`	a data frame
`agecolumn`	column name that corresponds to age or date pf birth
`nrcode`	non response code corresponding to age column

Value

0, if success error if failure

Examples

```
df <- data.frame("Age" = c(21, 15), "Name" = c("John", "Dora"))
test_age(df, "age", 999)
```

`test_columnnames` *Function to test column names of a data being different from what specified*

Description

Function to test column names of a data being different from what specified

Usage

```
test_columnnames(column_names, data)
```

Arguments

`column_names`	column names of the data frame
`data`	a data frame

Value

0, if success error, if failure

Examples

```
test_columnnames(c("name", "age"), data.frame(
  "Age" =  c(21, 15),
  "Name" =  c("John", "Dora")
))
```

`test_column_contents` *Function to check the format of column contents*

Description

Function to check the format of column contents

Usage

```
test_column_contents(data, column, code, nrcode = NA)
```

Arguments

`data`	a data frame
`column`	column name for gender
`code`	how column values are coded
`nrcode`	non response code corresponding to gender column

Value

0, if success error if failure

Examples

```
test_column_contents(data.frame(
  "sex" = c("m", "f"),
  "Name" = c("John", "Dora")
), "sex", c("m", "f"), 999)
```

`test_data_numeric` *Function to check the format of a numeric column*

Description

Function to check the format of a numeric column

Usage

```
test_data_numeric(column_name, data, nrcode = NA, minval, maxval)
```

Arguments

`column_name`	the column name
`data`	data frame
`nrcode`	non response code corresponding to the column
`minval`	minimum value allowed
`maxval`	maximum value allowed

Value

0, if success error, if failure

Examples

```
test_data_numeric("age", data.frame(
  "Age" =  c(21, 15),
  "Name"  = c("John", "Dora")
), -99, 0, 100)
```

`test_data_numeric_norange` *Function to check the format of a numeric column when the values are not bounded*

Description

Function to check the format of a numeric column when the values are not bounded

Usage

```
test_data_numeric_norange(column_name, data, nrcode = NA)
```

Arguments

`column_name`	the column name
`data`	data frame
`nrcode`	non response code corresponding to the column

Value

0, if success error, if failure

Examples

```
test_data_numeric_norange("marks", data.frame(
  "marks"  =  c(210, 99),
  "Name" =  c("John", "Dora")
), -99)
```

`test_data_string` *Function to check the format of a string column*

Description

Function to check the format of a string column

Usage

```
test_data_string(data, column_name, nrcode = NA)
```

Arguments

`data`	data frame
`column_name`	the column name
`nrcode`	non response code corresponding to the column

Value

0, if success error, if failure

Examples

```
test_data_string(data.frame("Age" = c(21, 15), "Name" = c("John", "Dora")), "name", -999)
```

---

test_data_string_restriction

Function to check the format of a string column when the string values are given

---

Description

Function to check the format of a string column when the string values are given

Usage

```
test_data_string_restriction(data, column_name, nrcode = NA,
  allowed_strings)
```

Arguments

`data`	data frame
`column_name`	the column name
`nrcode`	non response code corresponding to the column
`allowed_strings`	allowed strings or characters to represent meaningful entry

Value

0, if success error, if failure

Examples

```
test_data_string_restriction(
  data.frame("Age" = c(21, 15), "sex" = c("m", "f")),
  "sex", -999, c("f", "m")
)
```

test_file_exist_read *Function to throw error on invalid directory or file and if not readable*

Description

Function to throw error on invalid directory or file and if not readable

Usage

```
test_file_exist_read(filename)
```

Arguments

filename	name of a file or dir

Value

0, if success error, if failure

Examples

```
test_file_exist_read(system.file("extdata", "blank.txt",
  package = "IPDFileCheck"
))
```

test_gender *Function to check the format of 'gender' column in data*

Description

Function to check the format of 'gender' column in data

Usage

```
test_gender(data, gendercode, gendercolumn = "gender", nrcode = NA)
```

Arguments

data	a data frame
gendercode	how gender is coded
gendercolumn	column name for gender
nrcode	non response code corresponding to gender column

Value

0, if success error if failure

Examples

```
test_gender(data.frame("sex" = c("m", "f"), "Name" = c("John", "Dora")), c("f", "m"), "sex", 999)
```

Index