Package ‘fdq’

November 19, 2018

Type Package

Title Forest Data Quality

Date 2018-11-19

Version 0.11

Maintainer Caíque de Oliveira de Souza <forestgrowthsoftware@gmail.com>

Description Forest data quality is a package that contains methods of
analysis of forest databases, the purpose of the analyzes is to evaluate
the quality of the data present in the databases focusing on the dimensions
of consistency, pountuality and completeness. Databases can range from forest
inventory data to growth model data. The package has methods to work with large
volumes of data quickly, in addition in certain analyzes it is possible to generate
the graphs for a better understanding of the analysis and reporting of the analyzed analysis.

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat

Depends R(>= 3.0), Fgmutils

Imports data.table, sqldf, randomcoloR, ggplot2, plyr, utils, stats

RoxygenNote 6.1.1

NeedsCompilation no

Author Caíque de Oliveira de Souza [aut, cre],
Clayton Vieira Fraga Filho [ctb, dtc],
Miquéias Fernandes [ctb]

Repository CRAN

Date/Publication 2018-11-19 17:10:03 UTC

R topics documented:

`check.integer` *Ckeck Integer*

Description

checks if a variable is integer

Usage

```
check.integer(x)
```

Arguments

x	any variable

Value

TRUE if "x" is integer, FALSE if "x" not is interger

Examples

```
x = 5
check.integer(x)
```

`check_ages` *check_ages*

Description

This analysis verifies age differences on a paired basis, if the rounded ages are in months the check is if the difference is 12 months, if it is in year the consecutive ages should only present difference of 1 year, doubts about how to pair your base consult The Fgmutils package

Usage

```
check_ages(data_base, rounded_age1, rounded_age2, months = FALSE)
```

Arguments

`data_base`	data.frame data.table
`rounded_age1`	string name of column rounde age one
`rounded_age2`	string name of column rounde age two
`months`	TRUE for age in months or FALSE for age in years

`check_clones_different_parcel` *check_clones_different_pacel*

Description

This function checks if the clones of a tree have different plots

Usage

```
check_clones_different_parcel(database, parcel_name, clone_name,
  variables_to_group)
```

Arguments

`database`	data.frame, data.table or any database
`parcel_name`	string name of the field containing the parcels
`clone_name`	string name of the field containing the clones
`variables_to_group`	string(s) variable (s) that you want to group the result of the analysis

`check_dead_state` *check_dead_state*

Description

This function checks if the base state field is equal to dead (M) and there is some kind of measurement

Usage

```
check_dead_state(data_base, state, measurement_variables)
```

Arguments

`data_base`	data.frame data.table or any database
`state`	string field name representing state column in database
`measurement_variables`	string vector that contains a set of measurement variables to be analyzed, this variables are names of columns in database

`check_existing_ages` *check_existing_ages*

Description

This function checks if a given set of ages exists in a database column

Usage

```
check_existing_ages(database, ages_name, ages_to_check)
```

Arguments

`database`	data.frame data.table or any database
`ages_name`	string name of the column representing ages
`ages_to_check`	string name/vector of the column (s) representing ages to be checked

`check_existing_place` *check_existing_place*

Description

This function checks whether a particular set of sites or locations exists in a database column

Usage

```
check_existing_place(database, place_name, places_to_check)
```

Arguments

`database`	data.frame, data.table or any database
`place_name`	string name of the column representing site or place
`places_to_check`	value(s) to be checked, example: c(12,21,33)

`check_existing_plots` *check_existing_plots*

Description

This function checks if a particular set of parcels exists in a database column

Usage

```
check_existing_plots(database, plots_name, plots_to_check)
```

Arguments

`database`	data.frame, data.table or any database
`plots_name`	string column name representing parcels in the base
`plots_to_check`	value(s) to be checked, example: c(356,122)

`check_measurements_state` *check_measurements_state*

Description

This function checks if there is a measurement variable with value equal to 0 and if the respective states are different from M, F, A

Usage

```
check_measurements_state(data_base, measurement_variables, state)
```

Arguments

`data_base`	data.frame, data.table or any database
`measurement_variables`	set of variables to be analyzed, this set can be a vector of string with names of colunms
`state`	string name of the field that represents the state in database

`check_measurement_ages` *check_measurement_ages*

Description

This function verifies if measurement variables have records of type DAP2 <DAP1, HT2 <HT1 in consecutive ages i + 1 and i tt is necessary that the base is already paired to perform such analysis, to know more about pairing consult the Fgmutils package

Usage

```
check_measurement_ages(data_base, measurement_variable1,
  measurement_variable2)
```

Arguments

`data_base`	data.frame, data.table or any database
`measurement_variable1`	string field containing the measurement variables at age 1
`measurement_variable2`	string field containing the measurement variables at age 2

`check_parcel_different_spacing` *check_parcel_different_spacing*

Description

This function checks for partitions with different spacing at i and i + 1 ages, it is necessary that the base be paired including the field representing the spacing, doubts about how to pair its base see the Fgmutils package

Usage

```
check_parcel_different_spacing(database, parcel_name, spacing_age1,
  spacing_age2, variables_to_group)
```

Arguments

`database`	data.frame, data.table or any database
`parcel_name`	string containing the field name parcels in database
`spacing_age1`	string containing the name of the field spacing in the first age
`spacing_age2`	string containing the name of the field spacing in the second age
`variables_to_group`	variable (s) that you want to group the result of the analysis, this can be a vector os strings or strign name to group

`check_size_age_parcel` *check_size_age_parcel*

Description

This function checks if the age field is more than one age, returning TRUE to for yes and FALSE for no

Usage

```
check_size_age_parcel(database, age_name)
```

Arguments

`database`	data.frame, data.table or any database
`age_name`	string containing the name of the column that represents age

`check_undefined_spacing` *check_undefined_spacing*

Description

This function checks if there is any record with undefined spacing (0 or NA)

Usage

```
check_undefined_spacing(data_base, spacings)
```

Arguments

`data_base`	data.frame, data.table or any database
`spacings`	string vector containing the name of the variable (s) than represent spacings in database

`check_variables` *check_variables*

Description

This function checks if the entered column exists within the base

Usage

```
check_variables(database, variables)
```

Arguments

`database`	data.frame, data.table or any database
`variables`	vector of strings with names of columns

Value

TRUE for all variables in database, or FALSE for variables not present in columns

Examples

```
test <- data.frame("tree","diametrer","N")
check_variables(test,c("tree","diameter"))
```

`check_zero_measurement` *check_zero_measurement*

Description

This analysis verifies which measurement variables have values equal to 0 and then checks if there are variables in the states that the user reported

Usage

```
check_zero_measurement(data_base, measurement_variables, state_name,
  states_to_check)
```

Arguments

`data_base`	data.frame, data.table or any database
`measurement_variables`	string vector containing name of the field(s) it represents measurement variable(s) to be analyzed
`state_name`	string vector containing the name of the variable than represents state in database
`states_to_check`	string vector containing the name of the the states to be checked, the user can inform this names in a string vector like ("F","N")

`find_missing_age` *find_missing_age*

Description

This function identifies the missing age values in the database and notifies them to the user.

Usage

```
find_missing_age(database, age_name, ages_to_check)
```

Arguments

`database`	data.frame, data.table or any database
`age_name`	string that contains the field name that represents age in database
`ages_to_check`	vector containing the values of ages to be checked like c(12,23,48)

`find_missing_place` *find_missing_place*

Description

This function identifies values of sites or locations in the database and notifies them to the user

Usage

```
find_missing_place(database, place_name, places_to_check)
```

Arguments

`database`	data.frame, data.table or any database
`place_name`	string that contains the field name representing site or place in database
`places_to_check`	vector containing the values of places/sites to be checked like c(21,33,48)

`find_missing_variable` *find_missing_variable*

Description

This function identifies non-existent column names in the database and informs the user

Usage

```
find_missing_variable(data_base, variables)
```

Arguments

`data_base`	data.frame, data.table or any database
`variables`	vector string that contains the name(s) of columns to be checked in database

generate_diameter_classes

generate_diameter_classes

Description

This function identifies non-existent column names in the database and informs the user

Usage

```
generate_diameter_classes(database, diameter_names, amplitude,
  name_of_diameter_class)
```

Arguments

`database`	data.frame, data.table or any database
`diameter_names`	string with name of the field that contains the diameters of database
`amplitude`	desired amplitude for class creation, example: 1,2,4,6,7
`name_of_diameter_class`	string with name you want for the field class of diameter

generate_initial_diameter_class

generate_initial_diameter_class

Description

This function generates the initial class field to aid in the process of diametric increasing

Usage

```
generate_initial_diameter_class(database, plot_name, age_name)
```

Arguments

`database`	data.frame, data.table or any database
`plot_name`	string with the name of field representing plots in database
`age_name`	string with the name of field representing rounded age

generate_new_color *generate_new_color*

Description

This function generates a new random color without repeating the ones that were entered in the last field as parameter

Usage

```
generate_new_color(colors)
```

Arguments

`colors`	vector of strings containing existing colors, exemple: c("#6140bc" "#e75bf7" "#d15102" "#6a0b9e" "#e8ad4e")

generate_number_hectare *generate_number_hectare*

Description

This function generates the NHa, field that represents the number of surviving trees per hectare

Usage

```
generate_number_hectare(database, area_name, n_name, nha_name = "NHa")
```

Arguments

`database`	data.frame, data.table or any database
`area_name`	string with the name of field containing area in database
`n_name`	string with the name of field containing numbers of trees in database
`nha_name`	string with name you want for the field number of trees per hectare

getColors *getColors*

Description

This function generates a new random color for each diameter class in the base

Usage

```
getColors(database, diameter_classe_name)
```

Arguments

`database`	data.frame, data.table or any database
`diameter_classe_name`	string with the name of field (column) containing the diameter classes

get_ages *get_ages*

Description

This function concatenates age values in a string for a query and returns the same

Usage

```
get_ages(database, age_name, age_values)
```

Arguments

`database`	data.frame, data.table or any database
`age_name`	string with the name of field (column) containing the ages
`age_values`	vector with the age values you want to assemble string to made query, example: c(12,24,36)

get_max *get_max*

Description

This function returns the maximum value of one or more fields of measurement variables

Usage

```
get_max(database, variables)
```

Arguments

`database`	data.frame, data.table or any database
`variables`	string vector with name(s) of the column (s) you want to know the maximum value

get_min *get_min*

Description

This function returns the minimum value of one or more fields of measurement variables

Usage

```
get_min(database, variables)
```

Arguments

`database`	data.frame, data.table or any database
`variables`	string vector with name(s) of the column (s) you want to know the minimum value

get_place *get_place*

Description

This function returns a database from a particular site or location present in the original database

Usage

```
get_place(database, place_name, place_value)
```

Arguments

`database`	data.frame, data.table or any database
`place_name`	string with the name of the column that represents the place
`place_value`	vector with values of that you want to filter the sites/places of the database

mount_query *mount_query*

Description

This auxiliary function checks that need to group fields of certain measurements

Usage

```
mount_query(database, select_names, group_names, option)
```

Arguments

`database`	data.frame, data.table or any database
`select_names`	string vector with the name(s) of the column(s) you want to include in the selection
`group_names`	string vector with the name(s) of the column(s) you want to group the results
`option`	options to make the query, can be 1,2,3 each one for one use in the analysis functions

`sort_columns_crescent` *sort_columns_crescent*

Description

Sorts the database incrementally based on the selected column

Usage

```
sort_columns_crescent(database, column)
```

Arguments

`database`	data.frame, data.table or any database
`column`	string with the name of the column you want sort the database

Index