Package ‘basecamb’

February 6, 2022

Type Package

Title Utilities for Streamlined Data Import, Imputation and Modelling

Version 1.1.0

Description Provides functions streamlining the data analysis workflow:
  Outsourcing data import, renaming and type casting to a *.csv.
  Manipulating imputed datasets and fitting models on them. Summarizing models.

Depends R (>= 4.0.0)

Imports assertive.types, assertthat, dplyr, mice, Hmisc, survival,
  stats, purrr

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.2

Suggests testthat (>= 3.0.0), rms

Config/testthat/edition 3

URL https://CRAN.R-project.org/package=basecamb,
    https://github.com/codeblue-team/basecamb

BugReports https://github.com/codeblue-team/basecamb/issues

NeedsCompilation no

Author J. Peter Marquardt [aut] (<https://orcid.org/0000-0002-5596-1357>),
  Till D. Best [aut, cre] (<https://orcid.org/0000-0001-7323-827X>)

Maintainer Till D. Best <till-best@gmx.de>

Repository CRAN

Date/Publication 2022-02-06 21:00:05 UTC

R topics documented:

  .scale_variable .......................................................... 2
  apply_data_dictionary .................................................... 3
  apply_function_to_imputed_data ....................................... 4
.scale_variable

Scaling a variable

Description
A helper function to scale a variable in a dataframe. Divides 'variable' by 'scaling_denominator'.

Usage
.scale_variable(data, variable, scaling_denominator)

Arguments
- data : data.frame
- variable : a char indicating the variable to be scaled
- scaling_denominator : a numeric indicating the scaling. The variable is divided by the scaling_denominator.

Value
the input dataframe with the newly scaled 'variable'
apply_data_dictionary

Clean column names, types and levels

Description

Use a data dictionary data.frame to apply the following tidying steps to your data.frame:

- Remove superfluous columns
- Rename columns
- Ensure/coerce correct data type for each column
- Assign factorial levels, including renaming and grouping

Usage

apply_data_dictionary(
  data,
  data_dictionary,
  na_action_default = "keep_NA",
  print_coerced_NA = TRUE
)

Arguments

data data.frame to be cleaned
data_dictionary data.frame with the following columns:
- old_column_name : character with the old column name
- new_data_type : character denoting the tidy data type. Supported types are:
  - character
  - integer
  - float
  - factor
  - date
- new_column_name : tidy column name. Can be left blank to keep the old column name
- coding (factor and date columns only):
  - factor columns: character denoting old value (key) and new value (value) in a standardised fashion:
    * key-value pairs are separated from other key-value-pairs by a comma (",")
    * key and value of the same pair are separated by an equal sign ("=")
    * quotations around individual keys and values are recommended for clarity, but do not affect functionality.
apply_function_to_imputed_data

* all values will be coerced to type character, with the exception of "NA" being parsed as type NA
* using "default" as a key will assign the specified value to all current values that do not match any of the specified keys, excluding NA
  * using "NA" as a key will assign the specified value to all current NA values
* example coding: '"key1' = 'val1', 'key2' = 'val2', 'default' = 'Other', 'NA' = NA"
* if no coding is specified for a column, the coding remains unchanged
  * date columns: character denoting coding (see format argument in as.Date)
• Optional other columns (do not affect behaviour)

na_action_default
character: Specify what to do with NA values. Defaults to 'keep_NA'. Options are:
  • 'keep_NA' NA values remain NA values
  • 'assign_default' NA values are assigned the value specified as 'default'. Requires a 'default' value to be specified Can be overwritten for individual columns by specifying a value for key 'NA'

print_coerced_NA
logical indicating whether a message specifying the location of NAs that are introduced by apply_data_dictionary() to data should be printed.

Value
clean data.frame

Author(s)
J. Peter Marquardt

---

apply_function_to_imputed_data

Apply function to dataframes in a mice object

Description
Wrapper function to apply a function on each dataframe in an imputed dataset created with mice::mice().

Usage
apply_function_to_imputed_data(mice_data, fun, ...)

---
assign_factorial_levels

Assign custom values for key levels in factorial columns

Arguments

mice_data a mids object generated by mice::mice().
fun the function to apply to each dataframe. May only take one positional argument of type data.frame.
... other arguments passed to fun()

Value

a mids object with transformed data.

Author(s)

J. Peter Marquardt

Description

Use a named vector of keys (current value) and values for factorial columns to assign meaningful levels and/or group levels

Usage

assign_factorial_levels(
  data,
  factor_keys_values,
  na_action_default = "keep_NA"
)

Arguments

data data.frame to modify
factor_keys_values named list with:
  • Keys: Names of factor columns
  • values: Named vectors with
    – keys: current value (string representation)
    – values: new value to be assigned
    – if a 'default' key is passed, all existing values not conforming to the new scheme will be converted to the 'default' value
    – if a 'NA' key is passed, all NA values will be converted to the value specified here. Overwrites na_action_default for the specified column.
assign_types_names

Assign tidy types and names to a data.frame

Description

Verbosely assign tidy name and data type for each column of a data.frame and get rid of superfluous columns. Uses a .csv file for assignments to encourage a data dictionary based workflow. CAVE! Requires 'Date' type columns to already be read in as Date.

Usage

assign_types_names(data, meta_data)

Arguments

data data.frame to be tidied. Dates must already be of type date.

meta_data data.frame specifying old column names, new column names and datatypes of data. Has the following columns:

• old_column_name : character with the old column name.
• new_data_type : character denoting the tidy data type. Supported types are:
  – character (will be coerced using as.character()).
  – integer (will be coerced using as.integer()).
build_model_formula

- float (will be coerced using as.double()).
- factor (will be coerced using as.factor()). Will result in a warning if
  the new factor variable will have more than 10 levels.
- date (can only confirm correct datatype assignment or coerce characters
  with format '%Y-%m-%d').

- new_column_name : tidy column name. Can be left blank to keep the old
  column name.
- Optional other columns (do not affect behavior).

Value

  clean data.frame

Author(s)

  J. Peter Marquardt

build_model_formula  Build formula for statistical models

Description

  Build formula used in statistical models from vectors of strings with the option to specify an envi-
  ronment.

Usage

  build_model_formula(
    outcome,       # character denoting the column with the outcome.
    predictors,    # vector of characters denoting the columns with the predictors.
    censor_event = NULL,    # character denoting the column with the censoring event, for use in Survival-type
    env = parent.frame()    # models.
  )

Arguments

  outcome       # environment to be used in formula creation
  predictors
  censor_event
  env

Value

  formula for use in statistical models
Author(s)

J. Peter Marquardt

Examples

```r
build_model_formula("outcome", c("pred_1", "pred_2"))
build_model_formula("outcome", c("pred_1", "pred_2"), censor_event = "cens_event")
```

cox.zph.mids

Test cox proportional odds assumption on models using multiple imputation.

Description

Constructs a model and conducts a cox.zph test for each imputation of the data set.

Usage

```r
cox.zph.mids(
    model,      # cox proportional model to be evaluated
    imputations, # mids object containing imputations
    p_level = 0.05, # value below which violation of proportional odds assumption is assumed. Defaults to .05
    global_only = TRUE, # return global p-value only. Implies p_only to be TRUE
    return_raw = FALSE, # return cox.zph objects in a list. If TRUE, function will not return anything else
    p_only = TRUE, # returns p-values of test only. If FALSE returns Chi² and degrees of freedom as well
    verbose = TRUE # Set to FALSE to deactivate messages
)
```

Arguments

- **model**: cox proportional model to be evaluated
- **imputations**: mids object containing imputations
- **p_level**: value below which violation of proportional odds assumption is assumed. Defaults to .05
- **global_only**: return global p-value only. Implies p_only to be TRUE
- **return_raw**: return cox.zph objects in a list. If TRUE, function will not return anything else
- **p_only**: returns p-values of test only. If FALSE returns Chi² and degrees of freedom as well
- **verbose**: Set to FALSE to deactivate messages

Value

depending on specified options, this function can return

- default: A vector of global p-values
- global_only = FALSE: a data.frame with p-values for all variables plus the global
- return_raw = TRUE: list of cox.zph objects
Author(s)
J. Peter Marquardt

Examples
```r
data <- data.frame(time = 101:200, status = rep(c(0,1), 50), pred = rep(c(1:9, NA), 10))
imputed_data <- mice::mice(data)
cox_mod <- Hmisc::fit.mult.impute(survival::Surv(time, status) ~ pred,
fitter = rms::cph, xtrans = imputed_data)
cox.zph.mids(cox_mod, imputed_data)
```
Description

Filter a dataframe for the nth entry of each subject in it. A typical use cases would be to filter a dataset for the first or last measurement of a subject.

Usage

```r
filter_nth_entry(data, ID_column, entry_column, n = 1, reverse_order = FALSE)
```

Arguments

- `data`: the data.frame to filter
- `ID_column`: character column identifying subjects
- `entry_column`: character column identifying order of entries. That column can by of types Date, numeric, or any other type suitable for `order()`
- `n`: integer number of entry to keep after ordering
- `reverse_order`: logical when TRUE sorts entries last to first before filtering

Value

data.frame with <= 1 entry per subject

Author(s)

J. Peter Marquardt

Examples

```r
data <- data.frame(list(ID = rep(1:5, 3), encounter = rep(1:3, each=5), value = rep(4:6, each=5)))
filter_nth_entry(data, 'ID', 'encounter')
filter_nth_entry(data, 'ID', 'encounter', n = 2)
filter_nth_entry(data, 'ID', 'encounter', reverse_order = TRUE)
```
fit_mult_impute_obs_outcome

Fit a model on multiply imputed data using only observations with non-missing outcome(s)

Description

This function is a wrapper for fitting models with Hmisc::fit.mult.impute() on a multiply imputed dataset generated with mice::mice(). Cases with a missing outcome in the original dataset are removed from the mids object by using the "subset" argument in Hmisc::fit.mult.impute().

Usage

fit_mult_impute_obs_outcome(mids, formula, fitter, ...)

Arguments

mids a mids object, i.e. the imputed dataset.
formula a formula that describes the model to be fit. The outcome (y variable) in the formula will be used to remove missing cases.
fitter a modeling function (not in quotes) that is compatible with Hmisc::fit.mult.impute().
... additional arguments to Hmisc::fit.mult.impute().

Value

mod a fit.mult.impute object.

Author(s)

Till D. Best

Examples

# create an imputed dataset
imputed_data <- mice::mice(airquality)

fit_mult_impute_obs_outcome(mids = imputed_data, formula = Ozone ~ Solar.R + Wind, fitter = glm)
or_model_summary

Summarise a logistic regression model on the odds ratio scale

Description

This function summarises regression models that return data on the log-odds scale and returns a dataframe with estimates and confidence intervals as odds ratios. P value are also provided. Additionally, intercepts can be removed from the summary. This comes in handy when ordinal logistic regression models are fit. Ordinal regression models (such as proportional odds models) usually result in many intercepts that are not really of interest. This function is also compatible with models obtained from multiply imputed datasets, for example models fitted with \texttt{Hmisc::fit.mult.impute()}. 

Usage

\begin{verbatim}
or_model_summary(
    model, 
    conf_int = 1.96, 
    print_intercept = FALSE, 
    round_est = 3, 
    round_p = 4 
)
\end{verbatim}

Arguments

model          a model object with estimates on the log-odds scale. 
conf_int       a numeric used to calculate the confidence intervals. The default of 1.96 gives the 95\% confidence interval. 
print_intercept a logical flag indicating whether intercepts shall be removed. All variables that start with "y>=" will be removed. If there is a variable matching this pattern, it will also be removed! 
round_est      the number of decimals returned for estimates (odds ratios) and confidence intervals. 
round_p        the number of decimals provided for p-values. 

Details

CAVE! The function does not check whether your estimates are on the log-odds scale. It will do the transformation no matter what! 

Value

a dataframe with the adjusted odds ratio, confidence intervals and p-values. 

Author(s)

Till D. Best
parse_date_columns

Examples

# fit a logistic model
mod <- glm(formula = am ~ mpg + cyl, data = mtcars, family = binomial())

or_model_summary(model = mod)

parse_date_columns

Parse values in date columns as Dates

Description

Parse date columns in a data.frame as Date. Use a named list to specify each date column (key) and the format (value) it is coded in.

Usage

parse_date_columns(data, date_formats)

Arguments

data  data.frame to modify
date_formats  named list with:

• Keys: Names of date columns
• values: character specifying the format

Value

data.frame with date columns in Date type

Author(s)

J. Peter Marquardt

Examples

data <- data.frame(date = rep('01/23/4567', 5))
data <- parse_date_columns(data, list(date = '%m/%d/%Y'))
remove_missing_from_mids

Remove missing cases from a mids object

Description

Deprecated, use `apply_function_to_imputed_data` instead.

Usage

```r
remove_missing_from_mids(mids, var)
```

Arguments

- `mids`: mids objects that is filtered.
- `var`: a string or vector of strings specifying the variable(s). All cases (i.e. rows) for which there are missing values are removed.

Details

Remove_missing_from_mids is used to filter a mids object for missing cases in the original dataset in the variable `var`. This is useful for situations where you want to use as many observations as possible for imputation but only fit your model on a subset of these. Or, if you want to create one large imputed datset from which multiple analyses with multiple outcomes are derived.

Value

a mids object filtered for observed cases of `var`.

Author(s)

Till D. Best

See Also

`apply_function_to_imputed_data`
scale_continuous_predictors

Scale continuous predictors

Description
This function linearly scales variables in data objects according to a data dictionary. The data dictionary has at least two columns, "variable" and "scaling_denominator". "Variable" is divided by "scaling_denominator".

Usage
scale_continuous_predictors(data, scaling_dictionary)

Arguments
- data: a data object with variables.
- scaling_dictionary: a data.frame with two columns that are called "variable" and "scaling_denominator".

Value
The data with the newly scaled 'variables'.

Author(s)
Till D. Best
Index

.scale_variable, 2
apply_data_dictionary, 3
apply_function_to_imputed_data, 4, 14
assign_factorial_levels, 5
assign_types_names, 6
build_model_formula, 7
cox.zph.mids, 8
deconstruct_formula, 9
filter_nth_entry, 10
fit_mult_impute_obs_outcome, 11
or_model_summary, 12
parse_date_columns, 13
remove_missing_from_mids, 14
scale_continuous_predictors, 15