Package 'cat2cat'

August 5, 2022

Title Handling an Inconsistently Coded Categorical Variable in a Panel Dataset

Version 0.4.2

Maintainer Maciej Nasinski <nasinski.maciej@gmail.com>

Description Unifying of an inconsistently coded categorical variable between two different time points in accordance with a mapping table.
The main rule is to replicate the observation if it could be assign to a few categories.
Then using simple frequencies or statistical methods to approximate probabilities of being assign to each of them.
This procedure was invented and implemented in the paper by (Nasinski, Majchrowska and Broniatowska (2020) <doi:10.24425/cejeme.2020.134747>).

Depends R (>= 3.6)

License GPL (>= 2)

URL https://github.com/Polkas/cat2cat,
https://polkas.github.io/cat2cat/

BugReports https://github.com/Polkas/cat2cat/issues

Encoding UTF-8

Imports MASS

Suggests caret, randomForest, knitr, rmarkdown, pacman, testthat, magrittr, dplyr

LazyData true

VignetteBuilder knitr

RoxygenNote 7.2.1

NeedsCompilation no

Author Maciej Nasinski [aut, cre]

Repository CRAN

Date/Publication 2022-08-05 15:50:02 UTC

R topics documented:

`cat2cat` *Automatic mapping of a categorical variable in a panel dataset according to a new encoding*

Description

This function is built to work for two time points at once. Thus for more periods some recursion will be needed. The `prune_c2c` might be needed when we have many interactions to limit growing number of replications. This function might seems to be a complex at the first glance though it is built to offer a wide range of applications for complex tasks.

Usage

```
cat2cat(
 data = list(old = NULL, new = NULL, cat_var = NULL, cat_var_old = NULL, cat_var_new =
   NULL, id_var = NULL, time_var = NULL, multiplier_var = NULL, freqs_df = NULL),
  mappings = list(trans = NULL, direction = NULL),
 ml = list(data = NULL, cat_var = NULL, method = NULL, features = NULL, args = NULL)
)
```

Arguments

`data`	'named list' with fields 'old', 'new', 'cat_var' (or 'cat_var_old' and 'cat_var_new'), 'time_var' and optional 'id_var','multiplier_var','freq_df'.
`mappings`	'named list' with 2 fields 'trans' and 'direction'.
`ml`	'named list' with up to 5 fields 'data', 'cat_var', 'method', 'features' and optional 'args'.

Details

data args

- "old" data.frame older time point in a panel
- "new" data.frame more recent time point in a panel
- "cat_var" character name of the categorical variable.
- "cat_var_old" optional character name of the categorical variable in the older time point. Default ‘cat_var‘.
- "cat_var_new" optional character name of the categorical variable in the newer time point. Default ‘cat_var‘.
- "time_var" character name of the time variable.
- "id_var" optional character name of the unique identifier variable - if this is specified then for subjects observe in both periods the direct mapping is applied.
- "multiplier_var" optional character name of the multiplier variable - number of replication needed to reproduce the population
- "freqs_df" optional - data.frame with 2 columns where first one is category name and second counts. It is optional nevertheless will be very often needed, as give more control. It will be used to assess the probabilities. The multiplier variable is omit so sb has to apply it in this table.

mappings args

- "trans" data.frame with 2 columns - transition table - all categories for cat_var in old and new datasets have to be included. First column contains an old encoding and second a new one. The transition table should to have a candidate for each category from the targeted for an update period.
- "direction" character direction - "backward" or "forward"

optional ml args

- "data" data.frame - dataset with features and the ‘cat_var‘.
- "cat_var" character - the dependent variable name.
- "method" character vector - one or a few from "knn", "rf" and "lda" methods - "knn" k-NearestNeighbors, "lda" Linear Discrimination Analysis, "rf" Random Forest
- "features" character vector of features names where all have to be numeric or logical
- "args" optional - list parameters: knn: k ; rf: ntree

Without ml section only simple frequencies are assessed. When ml model is broken then weights from simple frequencies are taken. ‘knn‘ method is recommended for smaller datasets.

Value

named list with 2 fields old an new - 2 data.frames. There will be added additional columns like index_c2c, g_new_c2c, wei_freq_c2c, rep_c2c, wei_(ml method name)_c2c. Additional columns will be informative only for a one data.frame as we always make a changes to one direction.

Note

The transition table should to have a candidate for each category from the targeted for an update period. The observation from targeted for an updated period without a matched category from base period is removed. If you want to leave NA values add ‘c(NA, NA)‘ row to the ‘trans‘ table. Please check the vignette for more information.

Examples

```
## Not run:
data(occup_small)
data(occup)
data(trans)

occup_old <- occup_small[occup_small$year == 2008, ]
occup_new <- occup_small[occup_small$year == 2010, ]

# Adding the dummy level to the mapping table for levels without the candidate
# The best to fill them manually with proper candidates, if possible
# In this case it is only needed for forward mapping, to suppress warnings
trans2 <- rbind(
  trans,
  data.frame(
    old = "no_cat",
    new = setdiff(c(occup_new$code), trans$new)
  )
)

# default only simple frequencies
occup_simple <- cat2cat(
  data = list(old = occup_old, new = occup_new, cat_var = "code", time_var = "year"),
  mappings = list(trans = trans2, direction = "forward")
)

# additional probabilities from knn
occup_ml <- cat2cat(
  data = list(old = occup_old, new = occup_new, cat_var = "code", time_var = "year"),
  mappings = list(trans = trans, direction = "backward"),
  ml = list(
    data = occup_old,
    cat_var = "code",
    method = "knn",
    features = c("age", "sex", "edu", "exp", "parttime", "salary"),
    args = list(k = 10)
  )
)

## End(Not run)
```

cat2cat_agg *Manual mapping of a categorical variable according to a new encoding for aggregated panel dataset*

Description

Aggregate panel dataset - Manual mapping of a categorical variable according to a new encoding where user providing transitions with equations.

Usage

```
cat2cat_agg(
 data = list(old = NULL, new = NULL, cat_var = NULL, time_var = NULL, freq_var = NULL),
  ...
)
```

Arguments

data	list with 5 named fields ‘old‘, ‘new‘, ‘cat_var‘, ‘time_var‘, ‘freq_var‘.
...	equations where direction is set with any of ‘>‘, ‘<‘, ‘%>%‘, ‘%<%‘.

Details

data argument - list with fields

- "old" data.frame older time point in the panel
- "new" data.frame more recent time point in the panel
- "cat_var" character name of the categorical variable
- "time_var" character name of time variable
- "freq_var" character name of frequency variable

Value

list of two data.frame objects.

Examples

```
data(verticals)
agg_old <- verticals[verticals$v_date == "2020-04-01", ]
agg_new <- verticals[verticals$v_date == "2020-05-01", ]

## cat2cat_agg - could map in both directions at once although
## usually we want to have old or new representation

agg <- cat2cat_agg(
  data = list(
    old = agg_old,
```

```
    new = agg_new,
    cat_var = "vertical",
    time_var = "v_date",
    freq_var = "counts"
  ),
  Automotive %<% c(Automotive1, Automotive2),
  c(Kids1, Kids2) %>% c(Kids),
  Home %>% c(Home, Supermarket)
)
```

`cat_apply_freq` *Applying frequencies to the object returned by get_mappings function*

Description

applying frequencies to the object returned by get_mappings. We will get a symmetric object to returned one by get_mappings function, nevertheless categories are replaced with frequencies. Frequencies for each category/key are sum to 1, so could be interpreted as probabilities.

Usage

```
cat_apply_freq(to_x, freqs)
```

Arguments

`to_x`	list object returned by get_mappings.
`freqs`	vector object returned by get_freqs.

Value

a list with 2 named lists ‘to_old‘ and ‘to_new‘.

Examples

```
data(trans)
data(occup)

mappings <- get_mappings(trans)

mappings$to_old[1:4]
mappings$to_new[1:4]

mapp_p <- cat_apply_freq(
  mappings$to_old,
  get_freqs(
    occup$code[occup$year == "2008"],
    occup$multiplier[occup$year == "2008"]
  )
)
```

```
head(data.frame(I(mappings$to_old), I(mapp_p)))
mapp_p <- cat_apply_freq(
  mappings$to_new,
  get_freqs(
    occup$code[occup$year == "2010"],
    occup$multiplier[occup$year == "2010"]
  )
)
head(data.frame(I(mappings$to_new), I(mapp_p)))
```

`cross_c2c` *a function to make a combination of weights from different methods by each row*

Description

adding additional column which is a mix of weights columns by each row

Usage

```
cross_c2c(
  df,
  cols = colnames(df)[grepl("^wei_.*_c2c$", colnames(df))],
  weis = rep(1/length(cols), length(cols)),
  na.rm = TRUE
)
```

Arguments

`df`	data.frame
`cols`	character vector default all columns follow regex like "wei_.*_c2c"
`weis`	numeric vector Default vector the same length as cols and with equally spaced values summing to 1.
`na.rm`	logical if NA should be skipped, default TRUE

Value

data.frame with an additional column wei_cross_c2c

Examples

```
## Not run:
data(occup_small)
data(occup)
data(trans)

occup_old <- occup_small[occup_small$year == 2008, ]
occup_new <- occup_small[occup_small$year == 2010, ]
```

```
# mix of methods - forward direction, try out backward too
occup_mix <- cat2cat(
  data = list(old = occup_old, new = occup_new, cat_var = "code", time_var = "year"),
  mappings = list(trans = trans, direction = "backward"),
  ml = list(
    data = occup_new,
    cat_var = "code",
    method = c("knn"),
    features = c("age", "sex", "edu", "exp", "parttime", "salary"),
    args = list(k = 10, ntree = 20)
  )
)
# correlation between ml model
occup_mix_old <- occup_mix$old
cor(occup_mix_old[occup_mix_old$rep_c2c != 1, c("wei_knn_c2c", "wei_freq_c2c")])
# cross all methods and subset one highest probability category for each subject
occup_old_highest1_mix <- prune_c2c(cross_c2c(occup_mix$old),
  column = "wei_cross_c2c", method = "highest1"
)

## End(Not run)
```

---

dummy_c2c *Add default cat2cat columns to a 'data.frame'*

Description

a utils function to add default cat2cat columns to a 'data.frame'. It will be useful e.g. for a boarder periods which will not have additional 'cat2cat' columns.

Usage

```
dummy_c2c(df, cat_var, ml = NULL)
```

Arguments

df	'data.frame'
cat_var	'character' a categorical variable name.
ml	'character' vector of ml models applied, any of 'c("knn", "rf", "lda")'.

Examples

```
## Not run:
dummy_c2c(airquality, "Month")

data(occup_small)#'
occup_old <- occup_small[occup_small$year == 2008, ]
```

```
dummy_c2c(occup_old, "code")

## End(Not run)
```

`get_freqs` *Getting frequencies from a ‘character‘ vector with an optional multiplier argument*

Description

getting frequencies for a vector with an optional multiplier argument

Usage

```
get_freqs(x, multiplier = NULL)
```

Arguments

`x`	character vector categorical variable to summarize.
`multiplier`	numeric vector how many times to repeat certain value, additional weights.

Value

data.frame with two columns ‘input‘ ‘Freq‘

Note

without multiplier variable it is a basic ‘table‘ function wrapped with the ‘as.data.frame‘ function. The ‘table‘ function is used with the ‘useNA = "ifany"‘ argument.

Examples

```
data(occup)

head(get_freqs(occup$code[occup$year == "2008"]))
head(get_freqs(occup$code[occup$year == "2010"]))

head(get_freqs(occup$code[occup$year == "2008"], occup$multiplier[occup$year == "2008"]))
head(get_freqs(occup$code[occup$year == "2010"], occup$multiplier[occup$year == "2010"]))
```

get_mappings *Transforming a transition table with mappings to two associative lists*

Description

to rearrange the one classification encoding into another, an associative list that maps keys to values is used. More precisely, an association list is used which is a linked list in which each list element consists of a key and value or values. An association list where unique categories codes are keys and matching categories from next or previous time point are values. A transition table is used to build such associative lists.

Usage

```
get_mappings(x = data.frame())
```

Arguments

x	data.frame or matrix - transition table with 2 columns where first column is assumed to be the older encoding.

Details

the named list will be a more efficient solution than hash map as we are not expecting more than a few thousand keys.

Value

a list with 2 named lists ‘to_old‘ and ‘to_new‘.

Examples

```
data(trans)

mappings <- get_mappings(trans)
mappings$to_old[1:4]
mappings$to_new[1:4]
```

occup *Occupational dataset*

Description

Occupational dataset

Usage

```
occup
```

Format

A data frame with around 70000 observations and 12 variables.

id integer id

age numeric age of a subject

sex numeric sex of a subject

edu integer edu level of education of a subject where lower means higher - 1 for at least master degree

exp numeric exp number of experience years for a subject

district integer district

parttime numeric contract type regards time where 1 mean full-time (work a whole week)

salary numeric salary per year

code character code - occupational code

multiplier numeric multiplier for the subject to reproduce a population - how many of such subjects in population

year integer year

code4 character code - occupational code - first 4 digits

Details

occup dataset is an example of unbalance panel dataset. This is a simulated data although there are applied a real world characteristics from national statistical office survey. The original survey is anonymous and take place every two years. It is presenting a characteristics from randomly selected company and then using k step procedure employees are chosen.

occupational dataset

`occup_small` *Occupational dataset - small one*

Description

Occupational dataset - small one

Usage

```
occup_small
```

Format

A data frame with around 8000 observations and 12 variables.

id integer id

age numeric age of a subject

sex numeric sex of a subject

edu integer edu level of education of a subject where lower means higher - 1 for at least master degree

exp numeric exp number of experience years for a subject

district integer district

parttime numeric contract type regards time where 1 mean full-time (work a whole week)

salary numeric salary per year

code character code - occupational code

multiplier numeric multiplier for the subject to reproduce a population - how many of such subjects in population

year integer year

code4 character code - occupational code - first 4 digits

Details

occup dataset is an example of unbalance panel dataset. This is a simulated data although there are applied a real world characteristics from national statistical office survey. The original survey is anonymous and take place every two years. It is presenting a characteristics from randomly selected company and then using k step procedure employees are chosen.

occupational dataset

Examples

```
set.seed(1234)
data(occup)
occup_small <- occup[sort(sample(nrow(occup), 8000)), ]
```

---

plot_c2c *Summary plots for cat2cat results*

Description

This function help to understand properties of cat2cat results. It is recommended to run it before further processing, like next iterations.

Usage

```
plot_c2c(data, weis = "wei_freq_c2c", type = c("both", "hist", "bar"))
```

Arguments

data	data.frame - one of the data.frames returned by the ‘cat2cat‘ function.
weis	character - name of a certain wei_*_c2c column, added by cat2cat function. Default wei_freq_c2c
type	character - one of 3 types ‘"both"‘, ‘"hist"‘, ‘"bar"‘.

Value

base plot graphics

Note

It will work only for data.frame produced by cat2cat function.

Examples

```
data(occup_small)
occup_old <- occup_small[occup_small$year == 2008, ]
occup_new <- occup_small[occup_small$year == 2010, ]

occup_2 <- cat2cat(
  data = list(old = occup_old, new = occup_new, cat_var = "code", time_var = "year"),
  mappings = list(trans = trans, direction = "backward")
)

plot_c2c(occup_2$old, type = c("both"))
plot_c2c(occup_2$old, type = c("hist"))
plot_c2c(occup_2$old, type = c("bar"))
```

---

prune_c2c *A set of prune methods which will be useful after transition process*

Description

user could specify one from four methods to prune replications

Usage

```
prune_c2c(
  df,
  index = "index_c2c",
  column = "wei_freq_c2c",
  method = "nonzero",
  percent = 50
)
```

Arguments

df	data.frame
index	character default wei_freq_c2c
column	character default index_c2c
method	character one of four available methods: default "nonzero", "highest", "highest1", "morethan"
percent	integer from 0 to 99

Details

method - specify method to reduce number of replications

- "nonzero" remove nonzero probabilities
- "highest" leave only highest probabilities for each subject- accepting ties
- "highest1" leave only highest probabilities for each subject- not accepting ties so always one is returned
- "morethan" leave rows where a probability is higher than value specify by percent argument

Value

data.frame

Examples

```
## Not run:
data(occup_small)
data(occup)
data(trans)

occup_old <- occup_small[occup_small$year == 2008, ]
occup_new <- occup_small[occup_small$year == 2010, ]

occup_ml <- cat2cat(
  data = list(old = occup_old, new = occup_new, cat_var = "code", time_var = "year"),
  mappings = list(trans = trans, direction = "backward"),
  ml = list(
    data = occup_new,
    cat_var = "code",
    method = "knn",
    features = c("age", "sex", "edu", "exp", "parttime", "salary"),
    args = list(k = 10)
  )
)

prune_c2c(occup_ml$old, method = "nonzero")
prune_c2c(occup_ml$old, method = "highest")
prune_c2c(occup_ml$old, method = "highest1")
prune_c2c(occup_ml$old, method = "morethan", percent = 90)
```

```
prune_c2c(occup_ml$old, column = "wei_knn_c2c", method = "nonzero")

## End(Not run)
```

---

summary_c2c *Adjusted summary for linear regression when based on replicated dataset*

Description

adjusting lm object results according to original number of degree of freedom. The standard errors, t statistics and p values have to be adjusted because of replicated rows.

Usage

```
summary_c2c(x, df_old, df_new = x$df.residual)
```

Arguments

x	lm object
df_old	integer number of d.f in original dataset. For bigger datasets ‘nrow‘ should be sufficient.
df_new	integer number of d.f in dataset with replicated rows, Default: x$df.residual

Details

The size of the correction is equal to sqrt(df_new / df_old). Where standard errors are multiplied and t statistics divided by it. In most cases the default df_new value should be used.

Value

data.frame with additional columns over a regular summary.lm output, like correct and statistics adjusted by it.

Examples

```
data(occup_small)
data(trans)

occup_old <- occup_small[occup_small$year == 2008, ]
occup_new <- occup_small[occup_small$year == 2010, ]

occup_2 <- cat2cat(
  data = list(old = occup_old, new = occup_new, cat_var = "code", time_var = "year"),
  mappings = list(trans = trans, direction = "backward"),
  ml = list(
    data = occup_new,
    cat_var = "code",
```

```
    method = "knn",
    features = c("age", "sex", "edu", "exp", "parttime", "salary"),
    args = list(k = 10)
  )
)

# Regression
# we have to adjust size of std as we artificialy enlarge degrees of freedom
lms <- lm(I(log(salary)) ~ age + sex + factor(edu) + parttime + exp, occup_2$old,
  weights = multiplier * wei_freq_c2c
)

summary_c2c(lms, df_old = nrow(occup_old))
```

`trans` *trans dataset containing transitions between old (2008) and new (2010) occupational codes. This table could be used to map encodings in both directions.*

Description

trans dataset containing transitions between old (2008) and new (2010) occupational codes. This table could be used to map encodings in both directions.

Usage

```
trans
```

Format

A data frame with 2693 observations and 2 variables.

old character an old encoding of a certain occupation

new character a new encoding of a certain occupation

Details

transition table for occupations where first column contains old encodings and second one a new encoding

verticals *verticals dataset*

Description

verticals dataset

Usage

```
verticals
```

Format

A data frame with 21 observations and 4 variables.

vertical character an certain sales vertical

sales numeric a size of sale

counts integer counts size

v_date character Date

Details

random data - aggregate sales across e-commerce verticals

Examples

```
set.seed(1234)
agg_old <- data.frame(
  vertical = c(
    "Electronics", "Kids1", "Kids2", "Automotive", "Books",
    "Clothes", "Home", "Fashion", "Health", "Sport"
  ),
  sales = rnorm(10, 100, 10),
  counts = rgeom(10, 0.0001),
  v_date = rep("2020-04-01", 10), stringsAsFactors = FALSE
)

agg_new <- data.frame(
  vertical = c(
    "Electronics", "Supermarket", "Kids", "Automotive1",
    "Automotive2", "Books", "Clothes", "Home", "Fashion", "Health", "Sport"
  ),
  sales = rnorm(11, 100, 10),
  counts = rgeom(11, 0.0001),
  v_date = rep("2020-05-01", 11), stringsAsFactors = FALSE
)
verticals <- rbind(agg_old, agg_new)
```

verticals2 *verticals2 dataset*

Description

verticals2 dataset

Usage

```
verticals2
```

Format

A data frame with 202 observations and 4 variables.

ean product ean

vertical character an certain sales vertical

sales numeric a size of sale

v_date character Date

Details

random data - single products sales across e-commerce verticals

Examples

```
set.seed(1234)
vert_old <- data.frame(
  ean = 90000001:90000020,
  vertical = sample(c(
    "Electronics", "Kids1", "Kids2", "Automotive", "Books",
    "Clothes", "Home", "Fashion", "Health", "Sport"
  ), 20, replace = TRUE),
  sales = rnorm(20, 100, 10),
  v_date = rep("2020-04-01", 20), stringsAsFactors = FALSE
)

vert_old2 <- data.frame(
  ean = 90000021:90000100,
  vertical = sample(c(
    "Electronics", "Kids1", "Kids2", "Automotive", "Books",
    "Clothes", "Home", "Fashion", "Health", "Sport"
  ), 80, replace = TRUE),
  sales = rnorm(80, 100, 10),
  v_date = rep("2020-04-01", 80), stringsAsFactors = FALSE
)

vert_new <- vert_old2
vert_new$sales <- rnorm(nrow(vert_new), 80, 10)
```

```
vert_new$v_date <- "2020-05-01"
vert_new$vertical[vert_new$vertical %in% c("Kids1", "Kids2")] <- "Kids"
vert_new$vertical[vert_new$vertical %in% c("Automotive")] <-
  sample(c("Automotive1", "Automotive2"), sum(vert_new$vertical %in% c("Automotive")),
    replace = TRUE
  )
vert_new$vertical[vert_new$vertical %in% c("Home")] <-
  sample(c("Home", "Supermarket"), sum(vert_new$vertical %in% c("Home")), replace = TRUE)

vert_new2 <- data.frame(
  ean = 90000101:90000120,
  vertical = sample(c(
    "Electronics", "Supermarket", "Kids", "Automotive1",
    "Automotive2", "Books", "Clothes", "Home",
    "Fashion", "Health", "Sport"
  ), 20,
  replace = TRUE
  ),
  sales = rnorm(20, 100, 10),
  v_date = rep("2020-05-01", 20), stringsAsFactors = FALSE
)

verticals2 <- rbind(
  rbind(vert_old, vert_old2),
  rbind(vert_new, vert_new2)
)
verticals2$vertical <- as.character(verticals2$vertical)
```

Index