Package 'cat2cat'

March 27, 2021

Title Mapping of a Categorical Variable in a Panel Dataset

Version 0.2.1

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

Description There are offered automatic methods to map a categorical variable according to a specific encoding across different time points.
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 algorithm 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

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

Encoding UTF-8

Imports rlang, caret, progress, randomForest, MASS, tidyr, dplyr, data.table, assertthat

Suggests knitr, rmarkdown, pacman, testthat, magrittr, igraph

LazyData true

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Maciej Nasinski [aut, cre]

Repository CRAN

Date/Publication 2021-03-27 02:50:02 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, id_var = NULL, time_var = NULL,
    multiplier_var = NULL, freqs_df = NULL),
  mappings = list(trans = NULL, direction = NULL),
  ml = list(method = NULL, features = NULL, args = NULL)
)
```

Arguments

`data`	list with 4, 5, 6 or 7 named fields ‘old‘ ‘new‘ ‘cat_var‘ ‘time_var‘ and optional ‘id_var‘,‘multiplier_var‘,‘freq_df‘
`mappings`	list with 2 named fields ‘trans‘ ‘direction‘
`ml`	list with 3 named fields ‘method‘ ‘features‘ ‘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
- "time_var" character name of the time variable
- "id_var" 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" character name of the multiplier variable - number of replication needed to reproduce the population
- "freqs_df" only for advanced users - data.frame with 2 columns where first one is category name and second counts which will be used to assess the probabilities.

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.
- "direction" character direction - "backward" or "forward"

ml args

- "method" character vector - one or a few from "knn", "rf" and "lda" methods - "knn" k-NearestNeighbors, "lda" Linear Discrimination, "rf" Random Forest
- "features" character vector of features names where all have to be numeric or logical
- "args" 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. Method knn 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.

Examples

```
data(occup_small)
data(occup)
data(trans)

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

# default only simple frequencies

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

```
  mappings = list(trans = trans, direction = "forward")
)

# additionaly add probabilities from knn

occup_3 <- cat2cat(
  data = list(old = occup_old, new = occup_new, cat_var = "code", time_var = "year"),
  mappings = list(trans = trans, direction = "forward"),
  ml = list(
    method = "knn",
    features = c("age", "sex", "edu", "exp", "parttime", "salary"),
    args = list(k = 10)
  )
)
```

`cat2cat_agg` *Aggregate panel dataset - Manual mapping of a categorical variable according to a new encoding*

Description

Aggregate dataset - Manual mapping of a categorical variable according to a new encoding where user supplying transitions by 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

Details

data args

- "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

... equations where direction is set by ">","<","

Value

list of 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_man - could map in both directions at once although
## usually we want to have oold 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 object returned by get_mappings*

Description

applying frequencies to object returned by get_mappings

Usage

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

Arguments

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

Value

list

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

```
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 = "forward"),
  ml = list(
    method = c("knn", "rf"),
    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_rf_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"
)
```

`get_freqs` *Getting frequencies for a 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`	vector
`multiplier`	vector how many times to repeat certain value

Value

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

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 table of mappings to a list with keys*

Description

transforming table of mappings to list with keys where first column is assumed to be an old encoding.

Usage

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

Arguments

x	data.frame or matrix with 2 columns where first column is assumed to be an old 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

list with 2 fields ‘to_old‘ ‘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 5000 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), 5000)), ]
```

plot_c2c *Summary plots for cat2cat results.*

Description

This function help to understand properties of cat2cat resuls and possibly before further preprocessing.

Usage

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

Arguments

data	data.frame
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

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

prune_c2c(occup_2$old, method = "nonzero")
prune_c2c(occup_2$old, method = "highest")
prune_c2c(occup_2$old, method = "highest1")
prune_c2c(occup_2$old, method = "morethan", percent = 90)

prune_c2c(occup_2$old, column = "wei_knn_c2c", method = "nonzero")
```

---

summary_c2c *Summary for data.frame with replicated rows*

Description

transforming summary.lm object according to real number of d.f. 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(
    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