Package: cat2cat (via r-universe)

September 18, 2024

Title Handling an Inconsistently Coded Categorical Variable in a Longitudinal Dataset

Version 0.4.7

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

Description Unifying 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 assigned to a few categories. Then using frequencies or statistical methods to approximate the probabilities of being assigned 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) | file LICENSE

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, dplyr, forcats, knitr, magrittr, randomForest, rmarkdown, testthat (>= 3.0.0), tidyr

LazyData true

VignetteBuilder knitr

RoxygenNote 7.2.3

Config/testthat/edition 3

Repository https://polkas.r-universe.dev

RemoteUrl https://github.com/polkas/cat2cat

RemoteRef HEAD

RemoteSha 6c4eb9f6103d6c2bdd9e6b5f5d1af5ab39ebf939

Contents

`cat2cat` *Automatic mapping in a panel dataset*

Description

The objective is to unify an inconsistently coded categorical variable in a panel dataset according to a mapping (transition) table. The mapping (transition) table is the core element of the process. The function has a modular design with three arguments 'data', 'mappings', and 'ml'. Each of these arguments is of a 'list' type, wherein the 'ml' argument is optional. Arguments are separated to identify the core elements of the 'cat2cat' procedure. Although this function seems complex initially, it is built to offer a wide range of applications for complex tasks. The function contains many validation checks to prevent incorrect usage. The function has to be applied iteratively for each two neighboring periods of a panel dataset. The `prune_c2c` function could be needed to limit growing number of replications.

Usage

```
cat2cat(
  data = list(old = NULL, new = NULL, time_var = NULL, cat_var = NULL, cat_var_old =
     NULL, cat_var_new = NULL, id_var = NULL, multiplier_var = NULL),
   mappings = list(trans = NULL, direction = NULL, freqs_df = 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‘.
`mappings`	‘named list‘ with 3 fields ‘trans‘, ‘direction‘ and optional ‘freqs_df‘.
`ml`	‘named list‘ (optional) 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

"time_var" character(1) name of the time variable.

"cat_var" character(1) name of the categorical variable.

"cat_var_old" Optional character(1) name of the categorical variable in the older time point. Default ‘cat_var‘.

"cat_var_new" Optional character(1) name of the categorical variable in the newer time point. Default ‘cat_var‘.

"id_var" Optional character(1) name of the unique identifier variable - if this is specified then for subjects observed in both periods, the direct mapping is applied.

"multiplier_var" Optional character(1) name of the multiplier variable - number of replication needed to reproduce the population

"freqs_df" Only for the backward compatibility check the definition in the description of the mappings argument

mappings args

"trans" data.frame with 2 columns - mapping (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 mapping (transition) table should to have a candidate for each category from the targeted for an update period.

"direction" character(1) direction - "backward" or "forward"

"freqs_df" Optional - data.frame with 2 columns where first one is category name (base period) and second counts. If It is not provided then is assessed automatically. Artificial counts for each variable level in the base period. It is optional nevertheless will be often needed, as gives more control. It will be used to assess the probabilities. The multiplier variable is omitted so sb has to apply it in this table.

Optional ml args

"data" data.frame - dataset with features and the ‘cat_var‘.

"cat_var" character(1) - 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 and 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 the changes to one direction. The new columns are added instead of the additional metadata as we are working with new datasets where observations could be replicated. For the transparency the probability and number of replications are part of each observation in the 'data.frame'.

Note

'trans' arg columns and the 'cat_var' column have to be of the same type. The mapping (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", package = "cat2cat")
data("occup", package = "cat2cat")
data("trans", package = "cat2cat")

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 a 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")
)
```

```
mappings <- list(trans = trans, direction = "backward")

ml_setup <- list(
    data = occup_small[occup_small$year >= 2010, ],
    cat_var = "code",
    method = "knn",
    features = c("age", "sex", "edu", "exp", "parttime", "salary"),
    args = list(k = 10)
)

# ml model performance check
print(cat2cat_ml_run(mappings, ml_setup))

# additional probabilities from knn
occup_ml <- cat2cat(
  data = list(
    old = occup_old, new = occup_new, cat_var = "code", time_var = "year"
  ),
  mappings = mappings,
  ml = ml_setup
)

## End(Not run)
```

`cat2cat_agg` *Manual mapping for an aggregated panel dataset*

Description

Manual mapping of an inconsistently coded categorical variable according to the user provided mappings (equations).

Usage

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

Arguments

`data`	list with 5 named fields 'old', 'new', 'cat_var', 'time_var', 'freq_var'.
`...`	mapping 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 - deprecated - name of the categorical variable

"cat_var_old" character name of the categorical variable in the old period

"cat_var_new" character name of the categorical variable in the new period

"time_var" character name of time variable

"freq_var" character name of frequency variable

Value

'named list' with 2 fields old and new - 2 data.frames. There will be added additional columns to each. The new columns are added instead of the additional metadata as we are working with new datasets where observations could be replicated. For the transparency the probability and number of replications are part of each observation in the 'data.frame'.

Note

All mapping equations have to be valid ones.

Examples

```
data("verticals", package = "cat2cat")
agg_old <- verticals[verticals$v_date == "2020-04-01", ]
agg_new <- verticals[verticals$v_date == "2020-05-01", ]

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

agg <- cat2cat_agg(
  data = list(
    old = agg_old,
    new = agg_new,
    cat_var_old = "vertical",
    cat_var_new = "vertical",
    time_var = "v_date",
    freq_var = "counts"
  ),
  Automotive %<% c(Automotive1, Automotive2),
  c(Kids1, Kids2) %>% c(Kids),
  Home %>% c(Home, Supermarket)
)

## possible processing
library("dplyr")
agg %>%
  bind_rows() %>%
```

```
  group_by(v_date, vertical) %>%
  summarise(
    sales = sum(sales * prop_c2c),
    counts = sum(counts * prop_c2c),
    v_date = first(v_date)
  )
```

`cat2cat_ml_run` *Function to check cat2cat ml models performance*

Description

ml and mappings arguments in `cat2cat` function can be used to run cross validation across all groups in ml data.

Usage

```
cat2cat_ml_run(mappings, ml, ...)

## S3 method for class 'cat2cat_ml_run'
print(x, ...)
```

Arguments

`mappings`	'named list' with 3 fields 'trans', 'direction' and optional 'freqs_df'.
`ml`	'named list' (optional) with up to 5 fields 'data', 'cat_var', 'method', 'features' and optional 'args'.
`...`	other arguments
`x`	cat2cat_ml_run instance created with `cat2cat_ml_run` function.

Value

argument x invisibly

See Also

`cat2cat`

Examples

```
## Not run:
library("cat2cat")
data("occup", package = "cat2cat")
data("trans", package = "cat2cat")

occup_2006 <- occup[occup$year == 2006, ]
occup_2008 <- occup[occup$year == 2008, ]
occup_2010 <- occup[occup$year == 2010, ]
```

```
occup_2012 <- occup[occup$year == 2012, ]

library("caret")
ml_setup <- list(
  data = rbind(occup_2010, occup_2012),
  cat_var = "code",
  method = c("knn", "rf", "lda"),
  features = c("age", "sex", "edu", "exp", "parttime", "salary"),
  args = list(k = 10, ntree = 50)
)
data <- list(
  old = occup_2008, new = occup_2010,
  cat_var_old = "code", cat_var_new = "code", time_var = "year"
)
mappings <- list(trans = trans, direction = "backward")
res <- cat2cat_ml_run(mappings, ml_setup, test_prop = 0.2)
res

## End(Not run)
```

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

Description

applying frequencies to the object returned by the 'get_mappings' function. We will get a symmetric object to the one returned by the '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`	'data.frame' object like the one returned by the 'get_freqs' function.

Value

a 'list' with a structure like 'to_x' object but with probabilities for each category.

Note

'freqs' arg first column (keys) and the to_x arg values have to be of the same type. The uniform distribution (outcomes are equally likely) is assumed for no match for all possible categories.

Examples

```
data("trans", package = "cat2cat")
data("occup", package = "cat2cat")

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` *Make a combination of weights from different methods*

Description

adding the additional column which is a mix of weights columns by each row. Ensemble of a few methods usually produces more accurate solutions than a single model would.

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‘ like result of the ‘cat2cat‘ function for a specific period.
`cols`	‘character‘ vector default all columns under the regex "wei_.*_c2c".

weis	'numeric' vector weighs for columns in the 'cols' argument. By default a vector of the same length as 'cols' argument and with equally spaced probability (summing to 1).
na.rm	'logical(1)' if 'NA' values should be omitted, default TRUE.

Value

'data.frame' with the additional column 'wei_cross_c2c'.

Examples

```
## Not run:
data("occup_small", package = "cat2cat")
data("occup", package = "cat2cat")
data("trans", package = "cat2cat")

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 obs
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(1)‘ a categorical variable name.
ml	‘character‘ vector of ml models applied, any of ‘c("knn", "rf", "lda")‘.

Value

the provided ‘data.frame‘ with additional ‘cat2cat‘ like columns.

Examples

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

data("occup_small", package = "cat2cat")
occup_old <- occup_small[occup_small$year == 2008, ]
dummy_c2c(occup_old, "code")
dummy_c2c(occup_old, "code", "knn")

## End(Not run)
```

---

get_freqs *Getting frequencies from a vector with an optional multiplier*

Description

getting frequencies for a vector with an optional multiplier.

Usage

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

Arguments

x	‘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", package = "cat2cat")

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 mapping (transition) table 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 mapping (transition) table is used to build such associative lists.

Usage

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

Arguments

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

Value

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

Examples

```
data("trans", package = "cat2cat")

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", package = "cat2cat")
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(1)‘ - name of a certain wei_*_c2c column, added by cat2cat function. Default ‘wei_freq_c2c‘.
type	‘character(1)‘ - 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", package = "cat2cat")
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 *Pruning which could be useful after the mapping process*

Description

user could specify one of four methods to prune replications created in the cat2cat procedure.

Usage

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

Arguments

df	'data.frame' like result of the 'cat2cat' function for a specific period.
index	'character(1)' a column name with the 'cat2cat' identifier. Should not be updated in most cases. Default 'index_c2c'.
column	'character(1)' a column name with weights, default 'wei_freq_c2c'.
method	'character(1)' one of four available methods: "nonzero" (default), "highest", "highest1" or "morethan".
percent	'integer(1)' from 0 to 99

Details

method - specify a 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' with the same structure and possibly reduced number of rows

Examples

```
## Not run:
data("occup_small", package = "cat2cat")
data("occup", package = "cat2cat")
data("trans", package = "cat2cat")

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 observations.

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", package = "cat2cat")
data("trans", package = "cat2cat")

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(
  formula = I(log(salary)) ~ age + sex + factor(edu) + parttime + exp,
  data = occup_2$old,
  weights = multiplier * wei_freq_c2c
)

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

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

Description

trans dataset containing mappings (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

mapping (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