Package ‘categoryEncodings’

Type Package
Title Category Variable Encodings
Encoding UTF-8
Version 1.4.3
Date 2020-1-30

BugReports https://github.com/JSzitas/categoryEncodings/issues

Description Simple, fast, and automatic encodings for category data using a data.table backend. Most of the methods are an implementation of "Sufficient Representation for Categorical Variables" by Johannemann, Hadad, Athey, Wager (2019) <arXiv:1908.09874>, particularly their mean, sparse principal component analysis, low rank representation, and multinomial logit encodings.

License GPL-3
RoxygenNote 6.1.1

Suggests testthat (>= 2.1.0), covr

URL https://github.com/JSzitas/categoryEncodings

Imports glmnet, sparsepca, data.table

NeedsCompilation no

Author Juraj Szitas [aut, cre]
Maintainer Juraj Szitas <szitas.juraj13@gmail.com>
Repository CRAN
Date/Publication 2020-03-02 11:20:06 UTC

R topics documented:

encode_categories ......................................................... 2
encode_deviation ............................................................ 3
encode_difference .......................................................... 4
encode_dummy ............................................................... 5
encode_helmert ............................................................. 6
**encode_categories**

Encode a given factor variable automatically

**Description**

TRANSFORMS THE ORIGINAL DESIGN MATRIX AUTOMATICALLY, USING THE APPROPRIATE ENCODING.

**Usage**

`encode_categories(X, Y = NULL, fact = NULL, method = NULL, keep = FALSE)`

**Arguments**

- **X**: The data.frame/data.table to transform.
- **Y**: Optional: The dependent variable to ignore in the transformation.
- **fact**: Optional: The factor variable(s) to encode by - either positive integer(s) specifying the column number, or the name(s) of the column. If left empty a heuristic is used to determine the factor variable(s), and a warning is written with the names of the variables converted.
- **method**: Optional: A character string indicating which encoding method to use, either of the following: "mean" * "median" * "deviation" * "lowrank" * "SPCA" * "mnl" * "dummy" * "difference" * "helmert" * "simple_effect" * "repeated_effect" If only a single method is specified, it is taken to encode either all of the variables supplied through *fact*, or variables which have been flagged as factors automatically. If multiple methods are specified, the number of methods must match the number of factor variables in *fact* - and these are applied in the order in which they were supplied. In case a mismatch occurs, an error is raised. If left empty, the appropriate method is selected on a case by case basis (and the selected methods are written out to console).
- **keep**: Whether to keep the original factor column(s), defaults to **FALSE**.

**Details**

Automatically selects the appropriate method given the number of anticipated newly created variables, based on the results in Johannemann et al.(2019) 'Sufficient Representations for Categorical Variables', and a simple heuristic - where
encode_deviation

Value

A new data.table X which contains the new columns and optionally the old factor(s).

Examples

design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
    sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"
encode_categories( design_mat, method = "mean" )

Description

Transforms the original design matrix using a deviation dummy encoding.

Usage

encode_deviation(X, fact, keep_factor = FALSE, encoding_only = FALSE)

Arguments

X
The data.frame/data.table to transform.
fact
The factor variable to encode by - either a positive integer specifying the column
number, or the name of the column.
keep_factor
Whether to keep the original factor column(defaults to **FALSE**).
encoding_only
Whether to return the full transformed dataset or only the new columns. Defaults
to FALSE and returns the full dataset.

Details

The deviation dummy variable encoding, with reference class level set to -1. The reference class is
always the last class observed.

Value

A new data.table X which contains the new columns and optionally the old factor.
Examples

design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
                     sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"

encode_deviation(X = design_mat, fact = "factor_var", keep_factor = FALSE)

---

encode_difference

Encode a given factor variable using difference encoding

Description

Transforms the original design matrix using a difference encoding.

Usage

encode_difference(X, fact, keep_factor = FALSE, encoding_only = FALSE)

Arguments

X
The data.frame/data.table to transform.

fact
The factor variable to encode by - either a positive integer specifying the column
number, or the name of the column.

keep_factor
Whether to keep the original factor column (defaults to **FALSE**).

encoding_only
Whether to return the full transformed dataset or only the new columns. Defaults
to FALSE and returns the full dataset.

Value

A new data.table X which contains the new columns and optionally the old factor.

Examples

design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
                     sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"

encode_difference(X = design_mat, fact = "factor_var", keep_factor = FALSE)
**Encode a given factor variable using dummy variables**

**Description**

Transforms the original design matrix using a dummy variable encoding.

**Usage**

```r
encode_dummy(X, fact, keep_factor = FALSE, encoding_only = FALSE)
```

**Arguments**

- **X**: The data.frame/data.table to transform.
- **fact**: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- **keep_factor**: Whether to keep the original factor column (defaults to **FALSE**).
- **encoding_only**: Whether to return the full transformed dataset or only the new columns. Defaults to **FALSE** and returns the full dataset.

**Details**

The basic dummy variable encoding, with reference class level set to 0. The reference class is always the first class observed.

**Value**

A new data.table X which contains the new columns and optionally the old factor.

**Examples**

```r
design_mat <- cbind( data.frame( matrix(rnorm(5*100), ncol = 5) ),
                      sample( sample(letters, 10), 100, replace = TRUE) )

colnames(design_mat)[6] <- "factor_var"

encode_dummy(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```
**encode_helmert**  
Encode a given factor variable using helmert encoding

**Description**
Transforms the original design matrix using a helmert (reverse difference) encoding.

**Usage**

```r
encode_helmert(X, fact, keep_factor = FALSE, encoding_only = FALSE)
```

**Arguments**

- `X`: The data.frame/data.table to transform.
- `fact`: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- `keep_factor`: Whether to keep the original factor column (defaults to **FALSE**).
- `encoding_only`: Whether to return the full transformed dataset or only the new columns. Defaults to **FALSE** and returns the full dataset.

**Value**
A new data.table X which contains the new columns and optionally the old factor.

**Examples**

```r
design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
                      sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"
encode_helmert(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```

---

**encode_lowrank**  
Encode a given factor variable using low rank encoding

**Description**
Transforms the original design matrix using a low rank encoding.

**Usage**

```r
encode_lowrank(X, fact, keep_factor = FALSE, encoding_only = FALSE)
```

**Examples**

```r
design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
                      sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"
encode_lowrank(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```
**encode_mean**

**Arguments**

- **X**: The data.frame/data.table to transform.
- **fact**: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- **keep_factor**: Whether to keep the original factor column (defaults to **FALSE**).
- **encoding_only**: Whether to return the full transformed dataset or only the new columns. Defaults to **FALSE** and returns the full dataset.

**Details**

Uses the method from Johannemann et al. (2019) 'Sufficient Representations for Categorical Variables' - Low rank.

**Value**

A new data.table X which contains the new columns and optionally the old factor.

**Examples**

```r
design_mat <- cbind(
data.frame( matrix(rnorm(5*100), ncol = 5) ),
  sample( sample(letters, 10), 100, replace = TRUE)
)
colnames(design_mat)[6] <- "factor_var"
encode_lowrank(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```

---

**encode_mean**

Encode a given factor variable using means encoding

**Description**

Transforms the original design matrix using a means encoding.

**Usage**

```r
encode_mean(X, fact, keep_factor = FALSE, encoding_only = FALSE)
```

**Arguments**

- **X**: The data.frame/data.table to transform.
- **fact**: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- **keep_factor**: Whether to keep the original factor column (defaults to **FALSE**).
- **encoding_only**: Whether to return the full transformed dataset or only the new columns. Defaults to **FALSE** and returns the full dataset.
**Details**

Uses the method from Johannemann et al. (2019) 'Sufficient Representations for Categorical Variables' - Means Encoding.

**Value**

A new data.table X which contains the new columns and optionally the old factor.

**Examples**

```r
design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
                      sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"
encode_mean(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```

**Description**

Encodes a given factor variable using median encoding.

**Usage**

```r
encode_median(X, fact, keep_factor = FALSE, encoding_only = FALSE)
```

**Arguments**

- **X**: The data.frame/data.table to transform.
- **fact**: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- **keep_factor**: Whether to keep the original factor column (defaults to `FALSE`).
- **encoding_only**: Whether to return the full transformed dataset or only the new columns. Defaults to `FALSE` and returns the full dataset.

**Details**

This might be somewhat lacking in theory (to the author’s best knowledge), but feel free to try it and publish the results if they turn out interesting on some particular problem.

**Value**

A new data.table X which contains the new columns and optionally the old factor.
encode_mnl

Examples

design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
                      sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"

encode_median(X = design_mat, fact = "factor_var", keep_factor = FALSE)

---

**encode_mnl**

*Encode a given factor variable using a multinomial logit representation*

**Description**

Transforms the original design matrix using a mnl encoding.

**Usage**

`encode_mnl(X, fact, keep_factor = FALSE, encoding_only = FALSE)`

**Arguments**

- **X**: The data.frame/data.table to transform.
- **fact**: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- **keep_factor**: Whether to keep the original factor column (defaults to `FALSE`).
- **encoding_only**: Whether to return the full transformed dataset or only the new columns. Defaults to `FALSE` and returns the full dataset.

**Details**

Uses the method from Johannemann et al.(2019) 'Sufficient Representations for Categorical Variables' - mnl.

**Value**

A new data.table X which contains the new columns and optionally the old factor.
encode_repeated_effect

Encode a given factor variable using a repeated effect encoding

Description
Transforms the original design matrix using a repeated effect encoding.

Usage
encode_repeated_effect(X, fact, keep_factor = FALSE, encoding_only = FALSE)

Arguments
- X: The data.frame/data.table to transform.
- fact: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- keep_factor: Whether to keep the original factor column (defaults to **FALSE**).
- encoding_only: Whether to return the full transformed dataset or only the new columns. Defaults to FALSE and returns the full dataset.

Value
A new data.table X which contains the new columns and optionally the old factor.

Examples
```r
design_mat <- cbind(data.frame(matrix(rnorm(5*100),ncol = 5),
                        sample(sample(letters, 10), 100, replace = TRUE)
                    )
colnames(design_mat)[6] <- "factor_var"
encode_mnl(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```
```r
design_mat <- cbind(data.frame(matrix(rnorm(5*100),ncol = 5),
                        sample(sample(letters, 10), 100, replace = TRUE)
                    )
colnames(design_mat)[6] <- "factor_var"
encode_repeated_effect(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```
**encode_simple_effect**

Encode a given factor variable using a simple effect encoding

Description

Transforms the original design matrix using a simple effect encoding.

Usage

```
encode_simple_effect(X, fact, keep_factor = FALSE,
                     encoding_only = FALSE)
```

Arguments

- **X**: The data.frame/data.table to transform.
- **fact**: The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.
- **keep_factor**: Whether to keep the original factor column (defaults to **FALSE**).
- **encoding_only**: Whether to return the full transformed dataset or only the new columns. Defaults to FALSE and returns the full dataset.

Value

A new data.table X which contains the new columns and optionally the old factor.

Examples

```
design_mat <- cbind( data.frame( matrix(rnorm(5*100),ncol = 5) ),
                      sample( sample(letters, 10), 100, replace = TRUE) )
colnames(design_mat)[6] <- "factor_var"
enc encode_simple_effect(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```

**encode_SPCA**

Encode a given factor variable using a sparse PCA representation

Description

Transforms the original design matrix using a sPCA encoding.

Usage

```
encode_SPCA(X, fact, keep_factor = FALSE, encoding_only = FALSE)
```
encode_SPCA

Arguments

X The data.frame/data.table to transform.

fact The factor variable to encode by - either a positive integer specifying the column number, or the name of the column.

keep_factor Whether to keep the original factor column (defaults to **FALSE**).

encoding_only Whether to return the full transformed dataset or only the new columns. Defaults to FALSE and returns the full dataset.

Details

Uses the method from Johannemann et al. (2019) 'Sufficient Representations for Categorical Variables' - sPCA.

Value

A new data.table X which contains the new columns and optionally the old factor.

Examples

design_mat <- cbind( data.frame( matrix(rnorm(5*100), ncol = 5) ),
                      sample( sample(letters, 10), 100, replace = TRUE) )

colnames(design_mat)[6] <- "factor_var"

encode_SPCA(X = design_mat, fact = "factor_var", keep_factor = FALSE)
Index

encode_categories, 2
encode_deviation, 3
encode_difference, 4
encode_dummy, 5
encode_helmert, 6
encode_lowrank, 6
encode_mean, 7
encode_median, 8
encode_mnl, 9
encode_repeated_effect, 10
encode_simple_effect, 11
encode_SPCA, 11