Package: categoryEncodings (via r-universe)

October 24, 2024

Type Package

Title Category Variable Encodings

Encoding UTF-8

Version 1.5.0

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 Johannemann, Hadad, Athey, Wager (2019) <arXiv:1908.09874>, particularly their 'means', ``sPCA'', ``low-rank'' and ``multinomial logit''.

License GPL-3

RoxygenNote 7.1.2

Suggests testthat (>= 2.1.0), covr, tibble

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

Imports glmnet, sparsepca, data.table

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

RemoteUrl https://github.com/jszitas/categoryencodings

RemoteRef HEAD

RemoteSha 77448086806c750b788d04b9253b648d47afaf65

Contents

`encoder` *Train an encoder*

Description

Make your own encoder to be used in a pipeline

Usage

```
encoder(
  X,
  Y = NULL,
  fact = NULL,
  method = NULL,
  custom_encoding_assignment = NULL,
  ...
)
```

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 to correspond in the order in which they were supplied. In case a missmatch occurs, an error is raised. If left empty, the appriopriate method is selected on a case by case basis (and the selected methods are written out to console).

custom_encoding_assignment
: **experimental** A function which takes two arguments (**X** and **fact**) denoting the data and the factors, respectivelly, and assigns a valid encoding **method** to each factor in **fact**.

...
: Not implemented.

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

Value

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

---

encode_categories *Encode a given factor variable automatically*

Description

[deprecated: use encoder()] 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 to correspond in the order in which they were supplied. In case a missmatch occurs, an error is raised. If left empty, the appriopriate 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

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" )
```

`encode_deviation` *Encode a given factor variable using deviation encoding*

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_dummy *Encode a given factor variable using dummy variables*

Description

Transforms the original design matrix using a dummy variable encoding.

Usage

```
encode_dummy(
  X,
  fact,
  keep_factor = FALSE,
  encoding_only = FALSE,
  use_reference = TRUE,
  reference_value = 0
)
```

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.
use_reference	Whether to include a reference level (i.e. whether the new encoding contains an **intercept-like** constant term). Defaults to **TRUE**.
reference_value	What the reference value should be if **use_reference** is set to **TRUE**. Defaults to 0.

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

```
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

```
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

```
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

```
encode_lowrank(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' - Low rank.

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_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

```
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

```
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)
```

encode_median *Encode a given factor variable using median encoding*

Description

Transforms the original design matrix using a median encoding.

Usage

```
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.

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.

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_mnl(X = design_mat, fact = "factor_var", keep_factor = FALSE)
```

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

```
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"

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, ...)
```

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.
...	Additional parameters to pass to spca.

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