Package 'cleandata'

December 1, 2018

Type Package

Title To Inspect and Manipulate Data; and to Keep Track of This Process

Version 0.3.0

Author Sherry Zhao

Maintainer Sherry Zhao <sxzhao@gwu.edu>

Description Functions to work with data frames to prepare data for further analysis. The functions for imputation, encoding, partitioning, and other manipulation can produce log files to keep track of process.

BugReports https://github.com/sherrisherry/cleandata/issues

URL https://github.com/sherrisherry/cleandata

Depends R (>= 3.0.0)

Imports stats

Suggests R.rsp

License MIT + file LICENSE

Encoding UTF-8

VignetteBuilder R.rsp

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2018-12-01 05:10:02 UTC

R topics documented:

---

encode *List of Encoders*

Description

The return value of `inspect_map` can be used to create inputs for the following fuctions. Refer to vignettes for examples.

`encode_ordinal`: Encode Ordinal Data Into Sequential Integers

`encode_binary`: Encode Binary Data Into 0 and 1

`encode_onehot`: Encode categorical data by One-hot encoding

---

encode_binary *Encode Binary Data Into 0 and 1*

Description

Encodes binary data into 0 and 1. Optionally records the result into a log file.

Usage

```
encode_binary(x, out.int=FALSE, full_print=TRUE, log = eval.parent(in_log_default))
```

Arguments

`x`	The data frame
`out.int`	Whether to convert encoded `x` to integers. Only set to `TRUE` when no `NA` in `x` because `NA`s in `x` causes error when converting to integers. By default, the encoded `x` is factorial.
`full_print`	When set to `FALSE`, only print minimum information. A full output includes summary of `x` before and after encoding.
`log`	Controls log files. To produce log files, assign it or the `log_arg` variable in the parent environment (dynamic scope) a list of arguments for `sink()`, such as `file`, `append`, and `split`.

Value

An encoded data frame.

Warning

x can only be a data frame. Don't pass a vector to it.

See Also

inspect_map, sink

Examples

```
# refer to vignettes if you want to use log files
message('refer to vignettes if you want to use log files')

# building a data frame
A <- as.factor(c('x', 'y', 'x'))
B <- as.factor(c('y', 'x', 'y'))
C <- as.factor(c('i', 'j', 'i'))
df <- data.frame(A, B, C)

# encoding
df <- encode_binary(df)
print(df)
```

---

encode_onehot *One-Hot Encoding*

Description

Encodes categorical data by One-hot encoding. Optionally records the result into a log file.

Usage

```
encode_onehot(x, colname.sep = '_', drop1st = FALSE,
    full_print=TRUE, log = eval.parent(in_log_default))
```

Arguments

x	The data frame
colname.sep	A character or string that acts as an divider in the names of the columns of encoding results.
drop1st	Whether drop the 1st level of every encoded column. The 1st level refers to the level that corresponds to 1 in a factor.
full_print	When set to FALSE, only print minimum information. A full output includes summary of x before and after encoding.
log	Controls log files. To produce log files, assign it or the log_arg variable in the parent environment (dynamic scope) a list of arguments for sink(), such as file, append, and split.

Value

An encoded data frame.

Warning

x can only be a data frame. Don't pass a vector to it.

See Also

inspect_map, sink

Examples

```
# refer to vignettes if you want to use log files
message('refer to vignettes if you want to use log files')

# building a data frame
A <- as.factor(c('x', 'y', 'x'))
B <- as.factor(c('i', 'j', 'k'))
df <- data.frame(A, B)

# encoding
df0 <- encode_onehot(df)
df0 <- cbind(df, df0)
print(df0)
df0 <- encode_onehot(df, colname.sep = '-', drop1st = TRUE)
df0 <- cbind(df, df0)
rm(df)
print(df0)
```

---

encode_ordinal *Encode Ordinal Data Into Integers*

Description

Encodes ordinal data into sequential integers by a given order. Optionally records the result into a log file.

Usage

```
encode_ordinal(x, order, none='', out.int=FALSE,
    full_print=TRUE, log = eval.parent(in_log_default))
```

Arguments

x	The data frame
order	a vector of the ordered labels from low to high.
none	The 'none'-but-not-'NA' level, which is always encoded to 0.
out.int	Whether to convert encoded x to integers. Only set to TRUE when no NA in x because NAs in x causes error when converting to integers. By default, the encoded x is factorial.
full_print	When set to FALSE, only print minimum information. A full output includes summary of x before and after encoding.
log	Controls log files. To produce log files, assign it or the log_arg variable in the parent environment (dynamic scope) a list of arguments for sink(), such as file, append, and split.

Value

An encoded data frame.

Warning

x can only be a data frame. Don't pass a vector to it.

See Also

inspect_map, sink

Examples

```
# refer to vignettes if you want to use log files
message('refer to vignettes if you want to use log files')

# building a data frame
A <- as.factor(c('y', 'z', 'x', 'y', 'z'))
B <- as.factor(c('y', 'x', 'z', 'z', 'x'))
C <- as.factor(c('k', 'i', 'i', 'j', 'k'))
df <- data.frame(A, B, C)

# encoding
df[, 1:2] <- encode_ordinal(df[,1:2], order = c('z', 'x', 'y'))
df[, 3] <- encode_ordinal(df[, 3, drop = FALSE], order = c('k', 'j', 'i'))
print(df)
```

impute *Impute Missing Values*

Description

impute_mode: Impute NAs by the modes of their corresponding columns.

impute_median: Impute NAs by the medians of their corresponding columns.

impute_mean: Impute NAs by the means of their corresponding columns.

Usage

```
impute_mode(x,cols=colnames(x),idx=row.names(x),log = eval.parent(in_log_default))

impute_median(x,cols=colnames(x),idx=row.names(x),log = eval.parent(in_log_default))

impute_mean(x,cols=colnames(x),idx=row.names(x),log = eval.parent(in_log_default))
```

Arguments

x	The data frame to be imputed.
cols	The index of columns of x to be imputed.
idx	The index of rows of x to be used to calculate the values to impute NAs. Use this parameter to prevent leakage.
log	Controls log files. To produce log files, assign it or the log_arg variable in the parent environment (dynamic scope) a list of arguments for sink(), such as file, append, and split.

Value

An imputed data frame.

See Also

inspect_map, sink

Examples

```
# refer to vignettes if you want to use log files
message('refer to vignettes if you want to use log files')

# building a data frame
A <- as.factor(c('y', 'x', 'x', 'y', 'z'))
B <- c(6, 3:6)
C <- 1:5
df <- data.frame(A, B, C)
df[3, 1] <- NA; df[2, 2] <- NA; df [5, 3] <- NA
print(df)
```

```
# imputation
df0 <- impute_mode(df, cols = 1:3)
print(df0)
df0 <- impute_mode(df, cols = 1:3, idx = 1:3)
print(df0)
df0 <- impute_median(df, cols = 2:3)
print(df0)
df0 <- impute_mean(df, cols = 2:3)
print(df0)
```

`inspect_map` *Classify The Columns of A Data Frame*

Description

Provide a map for imputation and encoding.

Usage

```
inspect_map(x, common = 0, message = TRUE)
```

Arguments

`x`	The data frame
`common`	a non-negative numerical parameter, if 2 factorial columns share more than 'common' levels, they share the same scheme. 0 means all the levels should be the same for 2 factorial columns to share the same scheme.
`message`	Whether print the process.

Value

A list of `factor_cols` (list), `factor_levels` (list), `num_cols` (vector), `char_cols` (vector), `ordered_cols` (vector), and `other_cols` (vector).

`factor_cols`	a list, in which each member is a vector of the names of the factorial columns that share the same scheme. The name of a vector is the same as its 1st member. Refer to the argument `common` for more information about scheme.
`factor_levels`	a list, in which each member is a scheme of the factorial columns. The name of a scheme is the same as its corresponding vector in `factor_cols`.
`num_cols`	a vector, in which are the names of the numerical columns.
`char_cols`	a vector, in which are the names of the string columns.
`ordered_cols`	a vector, in which are the names of the ordered factorial columns.
`other_cols`	a vector, in which are the names of the other columns.

See Also

encode, impute

Examples

```
# building a data frame
A <- as.factor(c('x', 'y', 'z'))
B <- as.ordered(c('z', 'x', 'y'))
C <- as.factor(c('y', 'z', 'x'))
D <- as.factor(c('i', 'j', 'k'))
E <- 5:7
df <- data.frame(A, B, C, D, E)

# inspection
dmap <- inspect_map(df)
summary(dmap)
print(dmap)
```

---

inspect_na *Find Out Which Columns Have Most NAs*

Description

Return the names and numbers of NAs of columns that have top # (refer to argument top) most NAs.

Usage

```
inspect_na(x, top=ncol(x))
```

Arguments

x	The data frame
top	The value of #.

Value

A named vector.

Examples

```
# building a data frame
A <- as.factor(c('y', 'x', 'x', 'y', 'z'))
B <- c(6, 3:6)
C <- 1:5
df <- data.frame(A, B, C)
df[3, 1] <- NA; df[2, 2] <- NA; df[4, 2] <- NA; df [5, 3] <- NA
print(df)
```

```
# inspection
a <- inspect_na(df)
print(a)
```

`inspect_smap` *Simply Classify The Columns of A Data Frame*

Description

A simplified thus faster version of `inspect_map`.

Usage

```
inspect_smap(x, message = TRUE)
```

Arguments

`x`	The data frame
`message`	Whether print the process.

Value

A list of `factor_cols` (vector), `num_cols` (vector), `char_cols` (vector), `ordered_cols` (vector), and `other_cols` (vector).

`factor_cols`	a vector, in which are the names of the factorial columns.
`num_cols`	a vector, in which are the names of the numerical columns.
`char_cols`	a vector, in which are the names of the string columns.
`ordered_cols`	a vector, in which are the names of the ordered factorial columns.
`other_cols`	a vector, in which are the names of the other columns.

See Also

`inspect_map`

Examples

```
# building a data frame
A <- as.factor(c('x', 'y', 'z'))
B <- as.ordered(c('z', 'x', 'y'))
C <- as.factor(c('y', 'z', 'x'))
D <- as.factor(c('i', 'j', 'k'))
E <- 5:7
df <- data.frame(A, B, C, D, E)

# inspection
dmap <- inspect_smap(df)
summary(dmap)
print(dmap)
```

partition_random *Partitioning A Dataset Randomly*

Description

Designed to create a validation column. Optionally records the result into a log file.

Usage

```
partition_random(x, name = 'Partition', train,
    val = 10^ceiling(log10(train))-train, test = TRUE,
seed = FALSE, log = eval.parent(in_log_default))
```

Arguments

x	The data frame
name	The name of the validation column.
train	The proportion of the training set.
val	The proportion of the validation set. If not given, a default value is calculated by assuming the sum of train and val is a nth power of 10.
test	Whether to have test set. If TURE, a default value is calculated by assuming the sum of train and val is a nth power of 10.
seed	Whether to set a random seed. If you want a reproducible result, pass a number to seed as the random seed.
log	Controls log files. To produce log files, assign it or the log_arg variable in the parent environment (dynamic scope) a list of arguments for sink(), such as file, append, and split.

Value

A partitioned column.

Warning

x can only be a data frame. Don't pass a vector to it.

See Also

sink

Examples

```
# refer to vignettes if you want to use log files
message('refer to vignettes if you want to use log files')

# building a data frame
A <- 2:16
B <- letters[12:26]
df <- data.frame(A, B)

# partitioning
df0 <- partition_random(df, train = 7)
df0 <- cbind(df, df0)
print(df0)
df0 <- partition_random(df, train = 7, val = 2)
df0 <- cbind(df, df0)
print(df0)
```

---

wh_dict *Create Data Dictionary from Data Warehouse*

Description

Stacks part of a data frame and repeat the other columns to fit the result of stacking. Optionally records the result into a log file.

Usage

```
wh_dict(x, attr, value)
```

Arguments

x	The data frame
attr	The index of the column in x to be explained.
value	The index of the column in x as the explanation in the Keys column of the dictionary.

Value

A 2-column data frame, in which the Keys column stores the explanation of the values in x[, attr].

Warning

x can only be a data frame. Don't pass a vector to it.

See Also

inspect_map, encode

Examples

```
# refer to vignettes if you want to use log files
message('refer to vignettes if you want to use log files')

# building a data frame
A <- c('i', 'j', 'i', 'k', 'j')
B <- as.factor(c('x', 'y', 'x', 'z', 'y'))
C <- 1:5
df <- data.frame(A, B, C)
print(df)

# encoding
dict <- wh_dict(df, attr = 'B', value = 'A')
print(dict)
```

Index