Package 'oneclust'

September 1, 2020

Type Package

Title Maximum Homogeneity Clustering for Univariate Data

Version 0.2.1

Maintainer Nan Xiao <me@nanx.me>

Description Maximum homogeneity clustering algorithm for one-dimensional data described in W. D. Fisher (1958) <doi:10.1080/01621459.1958.10501479> via dynamic programming.

License GPL-3

URL https://nanx.me/oneclust/, https://github.com/nanxstats/oneclust

Encoding UTF-8

LazyData true

VignetteBuilder knitr

BugReports https://github.com/nanxstats/oneclust/issues

LinkingTo Rcpp

Imports Rcpp, magrittr

Suggests genlasso, knitr, rmarkdown

RoxygenNote 7.1.1

NeedsCompilation yes

Author Nan Xiao [aut, cre] (<https://orcid.org/0000-0002-0250-5673>)

Repository CRAN

Date/Publication 2020-09-01 08:50:02 UTC

R topics documented:

cud *Masataka Okabe and Kei Ito's Color Universal Design palette*

Description

Masataka Okabe and Kei Ito's Color Universal Design palette

Usage

```
cud(x, shift = TRUE, reverse = FALSE)
```

Arguments

x	vector, color index
shift	start from the second color in the CUD palette?
reverse	reverse the order?

Value

a vector of color hex values

Examples

```
barplot(rep(1, 7), col = cud(1:7))
barplot(rep(1, 8), col = cud(1:8, shift = FALSE))
barplot(rep(1, 8), col = cud(1:8, shift = FALSE, reverse = TRUE))
```

oneclust *Maximum homogeneity clustering for one-dimensional data*

Description

Maximum homogeneity clustering for one-dimensional data

Usage

```
oneclust(x, k, w = NULL, sort = TRUE)
```

Arguments

x	numeric vector, samples to be clustered
k	integer, number of clusters
w	numeric vector, sample weights (optional)
sort	should we sort x (and w) before clustering? Default is TRUE. Otherwise the order of the data is respected.

Value

a list containing:

- `cluster` - cluster id of each sample
- `cut` - index of the optimal cut points

References

Fisher, Walter D. 1958. On Grouping for Maximum Homogeneity. *Journal of the American Statistical Association* 53 (284): 789–98.

Examples

```
set.seed(42)
x <- sample(c(
  rnorm(50, sd = 0.2),
  rnorm(50, mean = 1, sd = 0.3),
  rnorm(100, mean = -1, sd = 0.25)
))
oneclust(x, 3)
```

`sim_postcode_levels` *Simulate the levels and their sizes in a high-cardinality feature*

Description

Simulate the levels and their sizes in a high-cardinality feature

Usage

```
sim_postcode_levels(nlevels = 100L, seed = 1001)
```

Arguments

`nlevels`	number of levels to generate
`seed`	random seed

Value

a data frame of postal codes and sizes

Note

The code is derived from the example described in the "rare levels" vignette in the `vtreat` package.

Examples

```
df_levels <- sim_postcode_levels(nlevels = 500, seed = 42)
head(df_levels)
```

`sim_postcode_samples` *Simulate a high-cardinality feature and a binary response*

Description

Simulate a high-cardinality feature and a binary response

Usage

```
sim_postcode_samples(
  df_levels,
  n = 2000L,
  threshold = 1000,
  prob = c(0.3, 0.1),
  seed = 1001
)
```

Arguments

`df_levels`	number of levels
`n`	number of samples
`threshold`	the threshold for determining if a postal code is rare
`prob`	occurrence probability vector of the class 1 event in rare and non-rare postal codes
`seed`	random seed

Value

a data frame of samples with postal codes, response labels, and level rarity status

Note

The code is derived from the example described in the "rare levels" vignette in the `vtreat` package.

Examples

```
df_levels <- sim_postcode_levels(nlevels = 500, seed = 42)
df_postcode <- sim_postcode_samples(
  df_levels,
  n = 10000, threshold = 3000, prob = c(0.2, 0.1), seed = 43
)
head(df_postcode)
```

Index