Package 'piecenorms'

July 29, 2024

Title Calculate a Piecewise Normalised Score Using Class Intervals

Version 1.1.0

Author David Hammond [aut, cre]

URL https://github.com/david-hammond/piecenorms

BugReports https://github.com/david-hammond/piecenorms/issues

Maintainer David Hammond <anotherdavidhammond@gmail.com>

Description Provides an implementation of piecewise normalisation techniques useful when dealing with the communication of skewed and highly skewed data. It also provides utilities that recommends a normalisation technique based on the distribution of the data.

License MIT + file LICENSE

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, rlang, scales, R6, classInt, univariateML, COINr, stats, vdiffr

Depends R (>= 2.10)

NeedsCompilation no

Repository CRAN

Date/Publication 2024-07-29 17:20:08 UTC

Contents

piecenorms-package	*piecenorms: Calculate a Piecewise Normalised Score Using Class Intervals*

Description

`piecenorms` has been built to calculate normalised data piecewise using class intervals. This is useful in communication of highly skewed data.

Details

For highly skewed data, the package `classInt` provides a series of options for selecting class intervals. The `classInts` can be used as the breaks for calculating the piecewise normalisation function `piecenorm`. The function also allows the user to select their own breaks manually.

For any call to `piecenorm`, the user provides a vector of observations, a vector of breaks and a direction for the normalisation. The data is then cut into classes and normalised within its class.

Number of Bins:
$$n = \text{length}(\text{brks}) - 1$$

Normalisation Class Intervals:
$$\left(\frac{i-1}{n}, \frac{i}{n} \right] \forall i \in \{1 : n\}$$

In cases where there is only one bin defined as `c(min(obs), max(obs))`**, the function** `piecenorm` **resolves to standard minmax normalisation**.

The `piecenorms` package also provides a `normalisr` R6 class that

- Classifies data into a likely distribution family
- Provides a recommendation of an appropriate normalisation technique
- Provides functionality to apply this normalisation technique to a new data set

This is useful when the user would like to analyse how distributions have changed over time.

Note

As with any non-linear transformation, piecewise normalization preserves *ordinal invariance* within each class but does not preserve *global relative magnitudes*. However, it does maintain *relative magnitudes within each class*. On the other hand, more standard techniques like *min-max* normalization preserves both *ordinal invariance* and *global relative magnitudes*.

Definitions of each are as follows:

- **Ordinal Invariance:** The property that the order of the data points is preserved. If one normalized value is larger than another, it reflects the same order as in the original data.
- **Non-Preservation of Relative Magnitudes (Global):** This refers to the loss of the proportionality of the original data values when normalized. If one value is twice as large as another in the original data, this relationship might not be preserved in the normalized data.
- **Ordinal Invariance:** The property that the order of the data points is preserved. If one normalized value is larger than another, it reflects the same order as in the original data.

Author(s)

Maintainer: David Hammond <anotherdavidhammond@gmail.com>

See Also

Useful links:

- <https://github.com/david-hammond/piecenorms>
- Report bugs at <https://github.com/david-hammond/piecenorms/issues>

---

normalisr	*Creates a recommended classInt based on the type of distribution.*

Description

Creates a recommended classInt based on the type of distribution.

Creates a recommended classInt based on the type of distribution.

Details

Creates a normalisr R6 class for recommending a classInt based on the shape of the distribution of the observed data

Public fields

data (numeric())
    Original observations

outliers (logical())
    Logical vector indicating is observations are outliers

quantiles (numeric())
    Vector of quantiles

fitted_distribution (character())
    Suggested distribution

normalisation (character())
    Recommended class interval style based on distribution

breaks (numeric())
    Recommended breaks for classes

number_of_classes (numeric())
    Number of classes identified

normalised_data (numeric())
    Normalised values based on recommendations

polarity (numeric(1))
    Which direction should the normalisation occur

percentiles (numeric())
    Observation percentiles

fittedmodel (character())
    Fitted univariate model

model (univariateML())
    Fitted univariate model parameters

Methods

Public methods:

- normalisr$new()
- normalisr$print()
- normalisr$plot()
- normalisr$hist()
- normalisr$setManualBreaks()
- normalisr$applyto()
- normalisr$as.data.frame()
- normalisr$clone()

Method new(): Creates a new instance of this R6 class.

Create a new normalisr object.

Usage:
```
normalisr$new(
  x,
  polarity = 1,
  classint_preference = "jenks",
  num_classes = NULL,
  potential_distrs = c("unif", "power", "norm", "lnorm", "weibull", "pareto", "exp")
)
```

Arguments:

x A numeric vector of observations

polarity Which direction should the normalisation occur, defaults to 1 but can either be:

- **1:**: Lowest value is normalised to 0, highest value is normalised to 1
- **-1:** Highest value is normalised to 0, lowest value is normalised to 1

classint_preference Preference for classInt breaks (see ?classInt::classIntervals)

num_classes Preference for number of classInt breaks, defaults to Sturges number (see
    ?grDevices::nclass.Sturges)

potential_distrs The types of distributions to fit, defaults to c("unif", "power", "norm",
    "lnorm", "weibull", "pareto", "exp")

Returns: A new normalisr object.

Method print(): Prints the normalisr

Usage:
```
normalisr$print()
```

Method `plot()`: Plots the normalised values against the original

Usage:
```
normalisr$plot()
```

Method `hist()`: Histogram of normalised values against the original

Usage:
```
normalisr$hist()
```

Method `setManualBreaks()`: Allows user to set manual breaks

Usage:
```
normalisr$setManualBreaks(brks)
```

Arguments:
brks User Defined Breaks

Method `applyto()`: Applies the normalisation model to new data

Usage:
```
normalisr$applyto(x)
```

Arguments:
x A numeric vector of observations

Method `as.data.frame()`: Returns a data frame of the normalisation

Usage:
```
normalisr$as.data.frame()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
```
normalisr$clone(deep = FALSE)
```

Arguments:
deep Whether to make a deep clone.

Examples

```
set.seed(12345)

# Binary distribution test
x <- sample(c(0,1), 100, replace = TRUE)
y <- sample(c(0,1), 100, replace = TRUE)
mdl <- normalisr$new(x)
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)

# Uniform distribution test
x <- runif(100)
```

```
y <- runif(100)
mdl <- normalisr$new(x)
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)


# Normal distribution tests
x <- rnorm(100)
y <- rnorm(100)
mdl <- normalisr$new(x)
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)

# Lognormal distribution tests
x <- rlnorm(100)
y <- rlnorm(100)
mdl <- normalisr$new(x)
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)

# Lognormal distribution tests with 5 classes
x <- rlnorm(100)
y <- rlnorm(100)
mdl <- normalisr$new(x, num_classes = 5)
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)

# Exponential distribution test
x <- exp(1:100)
y <- exp(1:100)
mdl <- normalisr$new(x)
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)

# Poisson distribution test
x <- rpois(100, lambda = 0.5)
y <- rpois(100, lambda = 0.5)
mdl <- normalisr$new(x)
```

```
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)

# Weibull distribution test
x <- rweibull(100, shape = 0.5)
y <- rweibull(100, shape = 0.5)
mdl <- normalisr$new(x)
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)

# Set user defined breaks
mdl$setManualBreaks(c(5,10))
print(mdl)
mdl$plot()
mdl$hist()
head(mdl$as.data.frame())
mdl$applyto(y)
```

---

piecenorm   *Get piecewse normalised values from a vector of observations*

Description

Get piecewse normalised values from a vector of observations

Usage

```
piecenorm(obs, breaks, polarity = 1)
```

Arguments

obs	A vector of observations.
breaks	The breaks to normalise to.
polarity	Which direction should the normalisation occur.

Value

Vector of normalised observations

Examples

```
obs <- exp(1:10)
breaks <- c(min(obs), 8, 20, 100, 1000, 25000)
y <- piecenorm(obs, breaks)
plot(obs, y, type = 'l',
xlab = "Original Values",
ylab = "Normalised Values")
```

Index