Package: misc (via r-universe)

November 14, 2024

Title Miscellaneous Useful R Functions

Version 0.4.0

Date 2024-10-15

Description Miscellaneous Useful R Functions.

Depends R (>= 3.5.0), MASS, stats, utils, foreach, doSNOW, snow, parallel, compiler

License MIT

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.0

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

RemoteUrl https://github.com/thierrymoudiki/misc

RemoteRef HEAD

RemoteSha 15aaeb626dc7e3c90441bdd1c17b6fcbd68d2aba

Contents

`debug_print` *Debug print*

Description

Debug print

Usage

```
debug_print(x)
```

Arguments

`x`	An object to be printed

Examples

```
misc::debug_print(1:10)
misc::debug_print("Hello, world!")
```

`fit_param_dist` *Fit multiple parametric distributions, compute KL divergence, simulate best fit*

Description

Fit multiple parametric distributions, compute KL divergence, simulate best fit

Usage

```
fit_param_dist(vector, num_bins = 30, verbose = TRUE)
```

Arguments

`vector`	Numeric vector of data to fit
`num_bins`	Number of bins for the empirical histogram
`verbose`	Logical indicating whether to print results

Value

Function to simulate data from the best-fitting distribution

Examples

```
set.seed(123)
n <- 1000
vector <- rnorm(n)

start <- proc.time()[3]
simulate_function <- fit_param_dist(vector)
end <- proc.time()[3]
print(paste("Time taken:", end - start))
simulated_data <- simulate_function(n)  # Generate 100 samples from the best-fit distribution
par(mfrow = c(1, 2))
hist(vector, main = "Original Data", xlab = "Value", ylab = "Frequency")
hist(simulated_data, main = "Simulated Data", xlab = "Value", ylab = "Frequency")
```

---

is_package_available *Check if a package is available*

Description

Check if a package is available

Usage

```
is_package_available(pkg_name)
```

Arguments

pkg_name	A package name

Value

A logical value

Examples

```
misc::is_package_available("dplyr")
```

is_wholenumber *Check if a number is a whole number*

Description

Check if a number is a whole number

Usage

```
is_wholenumber(x, tol = .Machine$double.eps^0.5)
```

Arguments

x	A number
tol	A tolerance level

Value

A logical value

Examples

```
is_wholenumber(1)
is_wholenumber(1.1)
is_wholenumber(1L)
```

KL_divergence_hist *Function to calculate KL divergence for continuous distributions using histograms*

Description

Function to calculate KL divergence for continuous distributions using histograms

Usage

```
KL_divergence_hist(P, Q)
```

Arguments

P	Numeric vector representing the empirical distribution
Q	Numeric vector representing the theoretical distribution

Value

KL divergence between P and Q

Examples

```
P <- c(0.2, 0.3, 0.5)
Q <- c(0.1, 0.4, 0.5)
misc::KL_divergence_hist(P, Q)
```

`one_hot_encode` *One-hot encoding*

Description

One-hot encoding

Usage

```
one_hot_encode(y)
```

Arguments

`y`	A vector of class labels
`n_classes`	The number of classes

Value

A matrix of one-hot encoded labels

Examples

```
y <- as.factor(c(1, 2, 1, 1, 2))
misc::one_hot_encode(y)
```

`parfor` *Sequential or parallel for loop.*

Description

Sequential or parallel for loop.

Usage

```
parfor(
  what,
  args,
  cl = NULL,
  combine = c,
  errorhandling = c("stop", "remove", "pass"),
  verbose = FALSE,
  show_progress = TRUE,
  export = NULL,
  ...
)
```

Arguments

what	A function.
args	A list of arguments.
cl	Number of cores to use. If NULL, the loop will be sequential. It -1, the number of cores will be detected automatically.
combine	A function to combine the results.
errorhandling	A character string specifying how to handle errors. Possible values are "stop", "remove", and "pass".
verbose	A logical indicating whether to print progress.
show_progress	A logical indicating whether to show a progress bar.
export	A list of objects to export to the workers.
...	Additional arguments to pass to what for foreach::foreach (excludind .combine, .errorhandling, .options.snow, .verbose, and .export).

Value

A list of results.

Examples

```
# Sequential
print(misc::parfor(function(x) x^2, 1:10))

# Parallel
print(misc::parfor(function(x) x^2, 1:10, cl = 2))
```

rm_zero_cols *Removing columns containing only zeros*

Description

Removing columns containing only zeros

Usage

```
rm_zero_cols(X)
```

Arguments

X	A matrix or data frame

Value

A matrix or data frame

Examples

```
X <- matrix(c(1, 0, 3, 0, 5, 0, 0, 0), nrow = 2)
print(misc::rm_zero_cols(X))
```

scale_matrix *Scale matrix*

Description

Scale matrix

Usage

```
scale_matrix(X, X_mean = NULL, X_sd = NULL)
```

Arguments

X	A matrix
X_mean	Mean of each column
X_sd	Standard deviation of each column

Value

A list containing the scaled matrix, mean of each column, and standard deviation of each column

Examples

```
X <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
(X_scaled <- misc::scale_matrix(X))
(X_scaled <- misc::scale_matrix(X, X_mean = colMeans(X), X_sd = apply(X, 2, stats::sd)))
print(colMeans(X_scaled$X))
print(apply(X_scaled$X, 2, stats::sd))
```

`sort_df` *Sort data frame*

Description

Sort data frame

Usage

```
sort_df(df, by, decreasing = FALSE)
```

Arguments

`df`	data frame
`by`	column to sort by
`decreasing`	logical. Should sorting be decreasing?

Value

A sorted data frame

Examples

```
df <- data.frame(a = c(2, 4, 3), b = c(3, 5, 1))
misc::sort_df(df, "a")
misc::sort_df(df, "b", decreasing = TRUE)
```

splitts *Partition a time series object*

Description

Partition a time series object

Usage

```
splitts(y, split_prob = 0.5, return_indices = FALSE)
```

Arguments

y	A time series object
split_prob	Splitting ratio
return_indices	if TRUE, returns series' indices, otherwise, time series objects

Examples

```
misc::splitts(ts(1:10))
```

split_data *Split a dataset*

Description

Split a dataset

Usage

```
split_data(y, p = 0.5, seed = 123, type_split = c("stratify", "sequential"))
```

Arguments

y	A vector of labels
p	A proportion of the dataset to split
seed	An integer to set the seed
type_split	A character string specifying the type of split

Value

A vector of indices

Examples

```
set.seed(123)
(y <- rnorm(10))
misc::split_data(y, 0.5)
misc::split_data(y, 0.5, type_split = "sequential")
```

`timeit` *Timing an expression*

Description

Timing an expression

Usage

```
timeit(expr, times = 1, ...)
```

Arguments

`expr`	an R expression
`times`	number of repetitions
`...`	additional arguments passed to `base::eval`

Value

the elapsed time in seconds

Examples

```
timeit(1 + 1)
timeit(1 + 1, times = 10)
```

`vlookup` *VLOOKUP*

Description

A simple implementation similar to the VLOOKUP function in Excel.

Usage

```
vlookup(this, df, key, value)
```

Arguments

this	The value to look up
df	A data frame
key	The column to look up
value	The column to return

Value

The value in the value column corresponding to the key column

Examples

```
df <- data.frame(key = c("a", "b", "c"), value = c(1, 2, 3))
print(misc::vlookup("b", df, "key", "value"))
```

---

winkler_score *Winkler score for probabilistic forecasts*

Description

Winkler score for probabilistic forecasts

Usage

```
winkler_score(actual, lower, upper, level = 95, scale = FALSE)
```

Arguments

actual	numeric vector of actual values
lower	numeric vector of lower bounds
upper	numeric vector of upper bounds
level	numeric level of confidence
scale	logical, if TRUE, the score is scaled by the range of the bounds

Value

numeric score

Examples

```
actual <- c(1, 2, 3, 4, 5)
lower <- c(0, 1, 2, 3, 4)
upper <- c(2, 3, 4, 5, 6)
winkler_score(actual, lower, upper)
winkler_score(actual, lower, upper, scale = TRUE)
winkler_score(actual, lower, upper, level = 99)
winkler_score(actual, lower, upper, level = 99, scale = TRUE)
```

Index