Package ‘MiscMath’

January 21, 2025

Type Package

Title Miscellaneous Mathematical Tools

Version 1.0

Description Some basic math calculators for finding angles for triangles and for finding the greatest common divisor of two numbers and so on.

LazyLoad true

License GPL (>= 2)

Depends randomForest

NeedsCompilation yes

Author W.J. Braun [aut, cre]

Maintainer W.J. Braun <john.braun@ubc.ca>

Repository CRAN

Date/Publication 2025-01-21 15:30:06 UTC

Contents

MiscMath-package — *Miscellaneous Mathematical Tools and Facts*

Description

A random assortment of elementary mathematical formulas and calculators.

Examples

```
# Find the greatest common divisor of 57, 93 and 117.
gcd(c(57, 93, 117))
```

DecToBin — *Convert Decimal to Binary*

Description

Convert a given decimal constant in the interval (0, 1) to the corresponding binary representation.

Usage

```
DecToBin(x, m = 32, format = "character")
```

Arguments

`x`	a numeric vector of values in the interval (0, 1)
`m`	a numeric constant specifying the number of binary digits to use in the output
`format`	a character string constant specifying the form of the output

Details

Default format is as a character string. Alternatives are `plain` which prints results to the device, and `vector` which output a binary vector.

Value

a vector containing the binary representation

Examples

```
x <- c(.81, .57, .333)
DecToBin(x) # character output
DecToBin(x, format="vector") #  binary vector output
DecToBin(x, format="plain")
```

gcd *Greatest Common Divisor*

Description

Greatest common divisor or factor for all elements of a positive-integer-valued vector.

Usage

```
gcd(x)
```

Arguments

`x`	a numeric vector consisting of at least two positive integer values.

Details

The gcd is calculated using the Euclidean algorithm applied to successive pairs of the elements of x.

Value

a numeric constant containing the greatest common divisor.

Examples

```
x <- c(81, 57, 333)
gcd(x)
```

LawofCosines *Law of Cosines*

Description

Use of the ancient law of cosines to determine the angle between two sides of a triangle, given lengths of all three sides. This is a generalization of Pythagoras' Theorem.

Usage

```
LawofCosines(sides)
```

Arguments

`sides`	a numeric vector of length 3, containing the side lengths.

Value

a numeric constant giving the angle in between the sides corresponding to the first two components in `sides`. Result is expressed in degrees.

Examples

```
LawofCosines(c(3, 4, 5))
```

LawofSines *Law of Sines*

Description

Use of the ancient law of sines to determine the angle opposite one side of a triangle, given the length of the opposite side as well as the angle opposite another side whose length is also known. Alternatively, one can find the length of the side opposite a given angle.

Usage

```
LawofSines(sides, angles, findAngle)
```

Arguments

`sides`	a numeric vector of length 1 or 2, containing the side lengths.
`angles`	a numeric vector of length 1 or 2, containing the angles (in degrees).
`findAngle`	a logical constant

Details

If `findAngle` is `TRUE`, `sides` should be of length 2 and the function will compute angle opposite the side with length given by the second element of `sides`. Otherwise, `angles` should be of length 2, and the function will calculate the length of the side opposite the angle corresponding to the second element of `angles`.

Value

a numeric constant giving the angle opposite the given side, if `findAngle` is `TRUE`, or giving the length of the side opposite the given angle, if `findAngle` is `FALSE`.

Examples

```
LawofSines(c(3, 4), 50)  # find angle opposite the side of length 4.
LawofSines(3, c(70, 50), findAngle = FALSE) # find length of side opposite the 50 degree angle
```

MaxRunLength *Maximum Run Length*

Description

Calculate the maximum run length of of 0's in a binary vector.

Usage

```
MaxRunLength(x)
```

Arguments

x	a binary vector

Value

the maximum run length of 0's

Examples

```
x <- c(0L, 1L, 1L, 1L, 0L, 0L, 1L, 1L, 0L, 0L, 1L)
MaxRunLength(x) # should be 2
MaxRunLength(1L - x) # should be 3
# Test of Mersenne Twishter

RNGkind("Mers") # ensure that default generator is in use
N <- 10000
x <- runif(N)
y <- DecToBin(x, format = "vector", m = 40)
MaxHeadRunsM <- apply(y, 1, MaxRunLength) # 0 run lengths (Heads)
MaxTailRunsM <- apply(1-y, 1, MaxRunLength) # 1 run lengths (Tails)
# distributions of Max 0 run lengths and Max 1 run lengths should match
boxplot(MaxHeadRunsM, MaxTailRunsM, axes=FALSE, main="Maximum Run Length")
axis(side=1, at=c(1, 2), label=c("Heads", "Tails"))
axis(2)
box()

# Comparison with Wichmann-Hill Generator

RNGkind("Wich")
x <- runif(N)
y <- DecToBin(x, format = "vector", m = 40)
MaxHeadRunsW <- apply(y, 1, MaxRunLength)
MaxTailRunsW <- apply(1-y, 1, MaxRunLength)
boxplot(MaxHeadRunsW, MaxTailRunsW, axes=FALSE, main="Maximum Run Length")
axis(side=1, at=c(1, 2), label=c("Heads", "Tails"))
axis(2)
box()
RNGkind("Mers") # restore default generator
```

`microLASSO` *Simplest Case of LASSO Regression*

Description

Simple linear regression estimators for slope, intercept and noise standard deviation with absolute value penalty on slope.

Usage

```
microLASSO(x, y, lambda)
```

Arguments

`x`	a numeric vector of covariate values
`y`	a numeric vector of response values
`lambda`	a numeric constant which should be nonnegative

Value

a list consisting of

`Coefficients`	a numeric vector containing intercept and slope estimates
`RMSE`	a numeric constant containing the (penalized) maximum likelihood estimate of the noise standard deviation

Examples

```
x <- runif(30)
y <- x + rnorm(30)
microLASSO(x, y, lambda = 0.5)
```

`rAlias` *Alias Method for Generating Discrete Random Variates*

Description

Efficient method for generating discrete random variates from any distribution with a finite number (N) of states. The method is described in detail in Section 10.1 of the given reference.

Usage

```
rAlias(n, P)
```

Arguments

n	numeric, constant number of variates to be simulated
P	numeric, probability mass function, assuming states from 1 through N

Value

numeric vector of containing n simulated values from the given discrete distribution

References

S. Ross (1990) A Course in Simulation, MacMillan.

Examples

```
x <- rAlias(n = 100, P = c(1:5)/15)
table(x)/100
```

RNGtest — *Pseudorandom Number Testing via Random Forest*

Description

Given a sequence of pseudorandom numbers, this function constructs a random forest prediction model for successive values, based on previous values up to a given lag. The ability of the random forest model to predict future values is inversely related to the quality of the sequence as an approximation to locally random numbers.

Usage

```
RNGtest(u, m=5)
```

Arguments

u	numeric, a vector of pseudorandom numbers to test
m	numeric, number of lags to test

Value

Side effect is a two way layout of graphs showing effectiveness of prediction on a training and a testing subset of data. Good predictions indicate a poor quality sequence.

Author(s)

W. John Braun

Examples

```
x <- runif(200)
RNGtest(x, m = 4)
```

Index