Package ‘CryptRndTest’

January 20, 2025

Type Package

Title Statistical Tests for Cryptographic Randomness

Version 1.2.7

Date 2022-04-20

Author Haydar Demirhan (<https://orcid.org/0000-0002-8565-4710>)

Maintainer Haydar Demirhan <haydar.demirhan@rmit.edu.au>

Description Performs cryptographic randomness tests on a sequence of random integers or bits. Included tests are greatest common divisor, birthday spacings, book stack, adaptive chi-square, topological binary, and three random walk tests (Ryabko and Monarev, 2005) <doi:10.1016/j.jspi.2004.02.010>. Tests except greatest common divisor and birthday spacings are not covered by standard test suites. In addition to the chi-square goodness-of-fit test, results of Anderson-Darling, Kolmogorov-Smirnov, and Jarque-Bera tests are also generated by some of the cryptographic randomness tests.

Depends kSamples, sfsmisc, Rmpfr, parallel

Imports LambertW, gmp, tseries, methods

Suggests R.rsp, bbotk

VignetteBuilder R.rsp

License GPL-3

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-22 08:40:14 UTC

Contents

CryptRndTest-package 2
adaptive.chi.square 4
birthday.spacings 6
book.stack 7
CryptRndTest-internal 9

GCD 9
GCD.big 10
GCD.q 11
GCD.test 12
print.CryptRndTest 14
random.walk.tests 14
Strlng2 17
TBT.criticalValue 18
toBaseTen 20
toBaseTwo 20
topological.binary 21

Index **23**

`CryptRndTest-package` *Statistical Tests for Cryptographic Randomness*

Description

Performs cryptographic randomness tests on a sequence of random integers or bits. Included tests are greatest common divisor, birthday spacings, book stack, adaptive chi-square, topological binary, and three random walk tests. Tests except greatest common divisor and birthday spacings are not covered by standard test suites. In addition to the chi-square goodness-of-fit test, results of Anderson-Darling, Kolmogorov-Smirnov, and Jarque-Bera tests are also generated by some of the cryptographic randomness tests. Additionally, it includes functions for the calculation of greatest common divisor, the Stirling numbers of the second kind, critical value of the topological binary test, and base conversions from base 2 to 10 and vice versa.

Details

Package:	CryptRndTest
Type:	Package
Version:	1.2.7
Date:	2022-04-20
License:	GPL-3

To test statistical randomness of cryptographic randomness use functions `birthday.spacings` and `GCD.test` for testing sequences of integers, functions `adaptive.chi.square` and `book.stack` for testing sequences of integers or bits, and use functions `random.walk.tests` and `topological.binary` for testing sequences of bits. The function `random.walk.tests` performs random walk-excursion, random walk-expansion, and random walk-height tests.

Additionally, use the function `GCD.q` to compute greatest common divisor (GCD), the number of iterations required to find GCD, and the sequence of partial quotients for two integers. Use the function `GCD` to compute GCD and the number iterations required to find GCD, recursively. Use the function `GCD.big` to compute GCD, the number iterations required to find GCD, and the sequence of partial quotients for two big integers. Use the function `Strlng2` to compute the Stirling numbers of the second kind in an approximate manner when the inputs are large. Use the function `TBT.criticalValue` to compute the critical value for the topological binary test at a given level of significance. Use the function `toBaseTwo` to convert integers (including big integers) from base 10 to 2. Use the function `toBaseTen` to convert binary sequences (including long binary sequences) from base 2 to 10.

Note

Acknowledgement: The package **CryptRndTest** is based upon work supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. 114F249 of ARDEB-3001 grant.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

References

Alcover, P.M., Guillamon, A., Ruiz, M.C., A new randomness test for bit sequences. Informatica (2013), 24(3), 339–356.

Bleick, W.W., Wang, P.C.C., Asymptotics of Stirling Numbers of the Second Kind. Proceedings of the American Mathematical Society (1974), 42(2), 575–580.

Doganaksoy, A., Calik, C., Sulak, F., Turan, M.S., New randomness tests using random walk, In: National Cryptology Symposium II, (2006), Ankara, Turkey.

Marsaglia, G., Tsang, W.W., Some Difficult-to-pass tests of randomness. Journal of Statistical Software (2002), 7(3).

Ryabko, B.Ya., Monarev, V.A., Using information theory approach to randomness testing. Journal of Statistical Planning and Inference (2005), 133, 95–110.

Ryabko, B.Ya., Stognienko, V.S., Shokin Yu.I., A new test for randomness and its application to some cryptographic problems. Journal of Statistical Planning and Inference (2004), 123, 365–376.

Temme, N.M., Asymptotic estimates of Stirling numbers. Studies in Applied Mathematics (1993), 89, 233–243.

See Also

`adaptive.chi.square`, `birthday.spacings`, `book.stack`, `GCD.test`, `GCD`, `GCD.q`, `GCD.big`, `random.walk.tests`, `topological.binary`, `Strlng2`, `Stirling2`

Examples

```
# ----- General settings ---
RNGkind(kind = "Super-Duper")
B=8                 # Bit length is 8.
k=2000              # Generate 20000 integers.
alpha=0.05

# ----- Adaptive chi-square -----
```

```
A=0
A=round(runif(k,0,(2^B-1)))
S=2                 # Divide alphabet to two sub-sets.
test1=adaptive.chi.square(x=A, B, S, alpha, bit = FALSE)
print(test1)

# ----- Birthday Spacings -----
m=16                  # Number of birthdays is 16.
n=2^B                 # Length of year is 256.
lambda=(m^3)/(4*n)
x=round(runif(k,0,(2^B-1)))
test2=birthday.spacings(x, m, n, alpha, lambda, num.class=10)
print(test2)

# ----- Book Stack -----
n=B*(2^(B/2))         # Number of required bits.
N=n/B                 # Number of integers to be generated.
A=0
A=round(runif(N,0,(2^B-1)))
K=2                   # Divide alphabet to two sub-sets.
test3=book.stack(x=A, B, k = K, alpha, bit = FALSE)
print(test3)

# ----- Toplogical Binary Test -----

dat=round(runif(k,0,(2^B-1)))
x=sfsmisc::digitsBase(dat, base= 2, B) #Convert to base 2
critical.value=150                     #Obtained for B=8
test4=topological.binary(x, B, alpha, critical.value)
print(test4)

# ----- Other Functions -----
# ----- GCD -----
result=GCD(45,2)
print(result)

result=GCD(321235,25521)
print(result)

# ----- Striling 2 -----
Strlng2(1500,410,log=TRUE)  # Large values of n and k
```

`adaptive.chi.square` *Adaptive Chi-Square Test*

Description

Performs Adaptive Chi-Square test of Ryabko et al.(2004) to evaluate the randomness of an RNG.

Usage

```
adaptive.chi.square(x, B, S, alpha = 0.05, prop=0.5, bit=FALSE)
```

Arguments

`x`	a vector or matrix that includes random data. See details for further information.
`B`	the length of words (B-bit) that the chippered file will be divided into.
`S`	the number of subsets where letters of an alphabet are combined, and $S \geq 2$
`alpha`	a predetermined value of significance level with the default value of 0.05.
`prop`	a predetermined value of proportion of training data set.
`bit`	if `x` contains a sequence of bits, `bit` is set `TRUE`. Otherwise, a sequence of integers is entered and `bit` is set `FALSE`.

Details

It is possible to apply adaptive Chi-Square to smaller samples than that required for the regular chi-square test.

If `x` contains a sequence of bits, then `x` should be a matrix of Bxk, where k is the number of words (integers) generated by the RNG of interest. Otherwise, `x` is a kx1 vector of the words. Because bits will be converted to base 10 before application of the test, implementation time will be shorter with integer input.

The degrees of freedom of the resulting chi-square test is `S-1`. The value of `S` should be much less than 2^B.

Value

`statistic`	calculated value of the test statistic.
`p.value`	p-value of the test.
`result.acsq`	returns 0 if H0 is rejected and 1 otherwise.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Ryabko, B.Ya., Stognienko, V.S., Shokin Yu.I., A new test for randomness and its application to some cryptographic problems. Journal of Statistical Planning and Inference (2004), 123, 365–376.

Examples

```
RNGkind(kind = ”Super-Duper”)
B=16                 # Bit length is 16.
k=5000               # Generate 5000 integers.
x=0
x=round(runif(k,0,(2^B-1)))
S=2                  # Divide alphabet to two subsets.
alpha = 0.05
test=adaptive.chi.square(x, B, S, alpha, bit =FALSE)
print(test)
```

birthday.spacings *Birthday Spacings Test*

Description

Performs Birthday Spacings test of Marsaglia and Tsang (2002) to evaluate the randomness of an RNG. The Kolmogorov-Smirnov, Anderson-Darling, and Chi-Square tests are applied as goodness-of-fit tests.

Usage

```
birthday.spacings(x, m = 128, n = 2^16, alpha = 0.05, lambda, num.class = 10)
```

Arguments

`x` a vector that includes random integers.

`m` the number of birthdays.

`n` the length of year.

`alpha` a predetermined value of significance level with the default value of 0.05.

`lambda` mean of Poisson distribution that constitutes theoretical cumulative distribution function in goodness-of-fit tests. See Details section.

`num.class` number of classes in the constructed frequency table for goodness-of-fit testing.

Details

This is one of the "difficult to pass tests" that RNG’s that are able to pass this set of tests possibly pass most of the tests included in the Diehard Battery of Tests.

To conduct the test, m birthdays are randomly chosen from a year composed of n days. When the birthdays are sorted, asymptotic distribution of the number of duplicated values among the spacings between birthdays is Poisson with mean $\lambda = m^3/(4n)$. For most of the cases, this formula for `lambda` is useful. However, user should check suitability of the value entered for `lambda`. Note that some suitable values for m and n are given by Marsaglia and Tsang (2002).

The argument `num.class` should be increased along with increasing bit-length. It can be set to 5 for testing with 8-bit and to 10 for testing with 16-bit and higher.

Value

`AD.statistic` calculated value of the test statistic of Anderson-Darling goodness-of-fit test.

`AD.pvalue` p-value of the test of Anderson-Darling goodness-of-fit test.

`AD.result` returns 0 if H0 is rejected and 1 otherwise in Anderson-Darling goodness-of-fit test.

`KS.statistic` calculated value of the test statistic of Kolmogorov-Smirnov goodness-of-fit test.

`KS.pvalue` p-value of the test of Kolmogorov-Smirnov goodness-of-fit test.

`KS.result` returns 0 if H0 is rejected and 1 otherwise in Kolmogorov-Smirnov goodness-of-fit test.

`CS.statistic` calculated value of the test statistic of Chi-Square goodness-of-fit test.

`CS.pvalue` p-value of the test of Chi-Square goodness-of-fit test.

`CS.result` returns 0 if H0 is rejected and 1 otherwise in Chi-Square goodness-of-fit test.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Marsaglia, G., Tsang, W.W., Some Difficult-to-pass tests of randomness. Journal of Statistical Software (2002), 7(3).

Examples

```
RNGkind(kind = "L'Ecuyer-CMRG")
B=16                 # Bit length is 16.
m=32                 # Number of birthdays is 64.
n=2^B                # Length of year is 65536.
lambda=(m^3)/(4*n)
k=5000               # Generate 5000 integers.
x=round(runif(k,0,(2^B-1)))
alpha = 0.05
test=birthday.spacings(x, m, n, alpha, lambda, num.class=10)
print(test)
```

`book.stack` *Book Stack Test*

Description

Performs Book Stack test of Ryabko and Monarev (2005) to evaluate the randomness of an RNG. The Chi-Square test is applied as the goodness-of-fit test.

Usage

```
book.stack(x, B, k=2, alpha=0.05, bit=FALSE)
```

Arguments

`x`	a vector or matrix that includes random data. See details for further information.
`B`	the length of words (B-bit) that the chippered file will be divided.
`k`	the number of subsets that the alphabet will be divided. It should be chosen to ensure $|x|/k$ will be an integer.
`alpha`	a predetermined value of significance level with the default value of 0.05.
`bit`	if `x` contains a sequence of bits, `bit` is set `TRUE`. Otherwise, a sequence of integers is entered and `bit` is set `FALSE`.

Details

If `x` contains a sequence of bits, then `x` should be a matrix of BxN, where N is the number of words (integers) generated by the RNG of interest. Otherwise, `x` is an Nx1 vector of the words. Because bits will be converted to base-10 before application of the test, implementation time will be shorter with integer input. Optimal value of N, which also represents the length of sample that is composed of B-bit words, is obtained by the optimal length of sample composed of bits (n) that is given by Ryabko and Monarev (2005) as $n = B(2^(B/2))$. For example, if $B = 16$, then $n = 4096$ and the legth of alphabet is 65536. In this case, we need to enter 4096 bits or $N = 4096/16 = 256$ integers. However, under the setting $B = 32$, the length of alphabet is 2^32 and we need to enter 65536. Note that it is hard to implement the test for $B > 32$ due to the memory overflows. Therefore, this test is applicable for smaller values of B. In this test, because there is no asymptotic theoretical distribution introduced, only chi-square test is applied as goodness-of-fit test.

Value

`statistic`	calculated value of the test statistic.
`p.value`	p-value of the Chi-Square test.
`BS.result`	returns 0 if H0 is rejected and 1 otherwise.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Ryabko, B.Ya., Monarev, V.A., Using information theory approach to randomness testing. Journal of Statistical Planning and Inference (2005), 133, 95–110.

Examples

```
RNGkind(kind = ”L'Ecuyer-CMRG”)
B=8                 # Bit length is 8.
n=B*(2^(B/2))       # Number of required bits.
N=n/B               # Number of integers to be generated.
x=round(runif(N,0,(2^B-1)))
k=2                   # Divide alphabet to two sub-sets.
alpha=0.05
test=book.stack(x, B, k, alpha, bit = FALSE)
print(test)
```

`CryptRndTest-internal` *Functions for internal use only*

Description

Contains functions designed for internal use only.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

See Also

`adaptive.chi.square`, `birthday.spacings`, `book.stack`, `GCD.test`, `GCD`, `GCD.q`, `GCD.big`, `random.walk.tests`, `topological.binary`, `Strlng2`, `Stirling2`

`GCD` *Greatest Common Divisor*

Description

Finds the greatest common divisor (GCD) of two integers using a recursive approximation. In addition to the value of GCD, it generates the number of required iterations to find GCD.

Usage

```
GCD(x, y, k = 0)
```

Arguments

`x`	the first integer greater than zero.
`y`	the second integer greater than zero.
`k`	initial value for counting the number of steps. It must be set zero.

Value

k	the number of required iterations to find GCD.
g	the value of greatest common divisor.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

Examples

```
result=GCD(4535,2451)
print(result)

result=GCD(35,2)
print(result)
```

GCD.big *Greatest Common Divisor for Large Integers*

Description

Finds the greatest common divisor (GCD) of two large integers. It utilizes multiple precision floating point numbers along with the package Rmpfr. In addition to the value of GCD, it generates the number of required iterations to find GCD and the sequence of partial quotients.

Usage

```
GCD.big(x, y, B)
```

Arguments

x	the first integer greater than zero.
y	the second integer greater than zero.
B	default precision in bits.

Value

k	the number of required iterations to find GCD.
q	the sequence of partial quotients.
g	the value of greatest common divisor.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

Examples

```
result=GCD.big(14532710900972355716,4463510164971546043,64)
print(result)
```

GCD.q *Greatest Common Divisor*

Description

Finds the greatest common divisor (GCD) of two integers using the Euclidean algorithm. In addition to the value of GCD, it generates the number of required iterations to find GCD and the sequence of partial quotients.

Usage

```
GCD.q(x, y)
```

Arguments

x	the first integer greater than zero.
y	the second integer greater than zero.

Value

k	the number of required iterations to find GCD.
q	the sequence of partial quotients.
g	the value of greatest common divisor.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

Examples

```
result=GCD.q(4535,2451)
print(result)

result=GCD.q(35,2)
print(result)
```

GCD.test *GCD Test*

Description

Performs Greatest Common Divisor (GCD) test of Marsaglia and Tsang (2002) to evaluate the randomness of an RNG. Randomness tests are conducted over two outputs of greatest common divisor operation, namely the number of required iterations and the value of greatest common divisor. The Kolmogorov-Smirnov, Anderson-Darling, Jarque-Bera, and Chi-Square tests are applied as goodness-of-fit tests when the test is conducted over the number of required iterations. The Kolmogorov-Smirnov and Chi-Square tests are applied as goodness-of-fit tests when the test is conducted over the value of greatest common divisor.

Usage

```
GCD.test(x, B = 32, KS = TRUE, CSQ = TRUE, AD = TRUE, JB = TRUE,
         test.k = TRUE, test.g = TRUE, mu, sd, alpha = 0.05)
```

Arguments

x	an $N * 2$ matrix of integers that includes random data. See details for further information.
B	the length of words (B-bit).
KS	if TRUE, Kolmogorov-Smirnov goodness-of-fit test is applied.
CSQ	if TRUE, Chi-Square goodness-of-fit test is applied.
AD	if TRUE, Anderson-Darling goodness-of-fit test is applied.
JB	if TRUE, Jarque-Bera goodness-of-fit test is applied.
test.k	if TRUE, randomness test is applied over the number of required iterations of the GCD operation.
test.g	if TRUE, randomness test is applied over the value of greatest common divisor.
mu	the mean of theoretical normal distribution that the number of required iterations follows.
sd	the standard deviation of theoretical normal distribution that the number of required iterations follows.
alpha	a predetermined value of significance level with the default value of 0.05.

Details

Total number of integers to be tested is divided into two sets and entered as `x`. The GCD operation is applied to each row of `x`.

The number of required iterations follows a normal distribution with parameters `mu` and `sd`. Values of `mu` and `sd` are obtained by Monte Carlo simulation and given by Marsaglia and Tsang (2002) for 32-bit setting. We obtained values of `mu` and `sd` for other bit settings as `mu=4.2503, sd=1.650673` for 8-bits, `mu=8.8772, sd=2.38282` for 16-bits, ...for 24-bits,...

Value

`sig.value.k` a $4x1$ vector of p-values. Elements of `sig.value.k` include p-value of Kolmogorov-Smirnov and Chi-Square tests, respectively.

`sig.value.g` a $2x1$ vector of p-values. Elements of `sig.value.g` include p-value of Kolmogorov-Smirnov, Chi-Square, Jarque-Bera, and Anderson-Darling tests, respectively.

`KS.result.k` returns 0 if H0 is rejected and 1 otherwise in Kolmogorov-Smirnov goodness-of-fit test conducted over the number of required iterations.

`CSQ.result.k` returns 0 if H0 is rejected and 1 otherwise in Chi-Square goodness-of-fit test conducted over the number of required iterations.

`JB.result.k` returns 0 if H0 is rejected and 1 otherwise in Jarque-Bera goodness-of-fit test conducted over the number of required iterations.

`AD.result.k` returns 0 if H0 is rejected and 1 otherwise in Anderson-Darling goodness-of-fit test conducted over the number of required iterations.

`KS.result.g` returns 0 if H0 is rejected and 1 otherwise in Kolmogorov-Smirnov goodness-of-fit test conducted over the value of greatest common divisor.

`CSQ.result.g` returns 0 if H0 is rejected and 1 otherwise in Chi-Square goodness-of-fit test conducted over the value of greatest common divisor.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Marsaglia, G., Tsang, W.W., Some Difficult-to-pass tests of randomness. Journal of Statistical Software (2002), 7(3).

See Also

See the function `GCD` that provides detailed results for the greatest common divisor operation.

Examples

```
RNGkind(kind = ”L'Ecuyer-CMRG”)
B=16                 # Bit length is 16.
k=250                # Generate 250 integers.
x=array(0,dim=c(k,2))
x[,1]=round(runif(k,0,(2^B-1)))
x[,2]=round(runif(k,0,(2^B-1)))
mu=8.8772
sd=2.38282
alpha = 0.05
test=GCD.test(x,B=B,KS=TRUE,CSQ=TRUE,AD=TRUE,JB=TRUE,
              test.k=TRUE,test.g=TRUE,mu=mu,sd=sd,alpha=alpha)
print(test)
```

print.CryptRndTest *Print Test Results*

Description

Prints a summary of test results.

Usage

```
## S3 method for class 'CryptRndTest'
print(x,...)
```

Arguments

x	an object including information to be printed.
...	other arguments.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

random.walk.tests *Random Walk Tests*

Description

Performs random walk tests of Doganaksoy et al. (2006) to evaluate the randomness of an RNG. It runs Random Walk Excursion, Random Walk Expansion, and Random Walk Height tests.

Usage

```
random.walk.tests(x, B = 64, Excursion = TRUE, Expansion = TRUE,
                  Height = TRUE, alpha = 0.05)
```

Arguments

x	a matrix that includes random data in base-2 format. See details for further information.
B	the length of words (B-bit). See Details section.
Excursion	if TRUE, Random Walk Excursion test is applied.
Expansion	if TRUE, Random Walk Expansion test is applied.
Height	if TRUE, Random Walk Height test is applied.
alpha	a predetermined value of significance level with the default value of 0.05.

Details

Argument `x` should be entered as a matrix of bits of dimension Bxk, where k is the number of words (integers) generated by the RNG of interest.

If `Excursion` is TRUE, B takes the values 16, 32, 64, 128, and 256. If `Height` is TRUE, B takes 64, 128, 256, 512, and 1024. If `Expansion` is TRUE, B takes 32, 64, and 128. Because theoretical cumulative distribution functions for the other word lengths, `random.walk.tests()` performs tests under given bit settings. If one of the tests is not applied, all the results related with that test in output are set to -1.

Value

`AD.statistic.Excursion`
value of test statistic of Anderson-Darling goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`KS.statistic.Excursion`
value of test statistic of Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`CS.statistic.Excursion`
value of test statistic of Chi-Square goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`AD.pvalue.Excursion`
p-value of Anderson-Darling goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`KS.pvalue.Excursion`
p-value of Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`CS.pvalue.Excursion`
p-value of Chi-Square goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`AD.result.Excursion`
returns 0 if H0 is rejected and 1 otherwise in Anderson-Darling goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`KS.result.Excursion`
returns 0 if H0 is rejected and 1 otherwise in Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`CS.result.Excursion`
returns 0 if H0 is rejected and 1 otherwise in Chi-Square goodness-of-fit test conducted after application of Random Walk Excursion procedure.

`AD.statistic.Expansion`
value of test statistic of Anderson-Darling goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`KS.statistic.Expansion`
value of test statistic of Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`CS.statistic.Expansion`
value of test statistic of Chi-Square goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`AD.pvalue.Expansion`
: p-value of Anderson-Darling goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`KS.pvalue.Expansion`
: p-value of Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`CS.pvalue.Expansion`
: p-value of Chi-Square goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`AD.result.Expansion`
: returns 0 if H0 is rejected and 1 otherwise in Anderson-Darling goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`KS.result.Expansion`
: returns 0 if H0 is rejected and 1 otherwise in Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`CS.result.Expansion`
: returns 0 if H0 is rejected and 1 otherwise in Chi-Square goodness-of-fit test conducted after application of Random Walk Expansion procedure.

`AD.statistic.Height`
: value of test statistic of Anderson-Darling goodness-of-fit test conducted after application of Random Walk Height procedure.

`KS.statistic.Height`
: value of test statistic of Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Height procedure.

`CS.statistic.Height`
: value of test statistic of Chi-Square goodness-of-fit test conducted after application of Random Walk Height procedure.

`AD.pvalue.Height`
: p-value of Anderson-Darling goodness-of-fit test conducted after application of Random Walk Height procedure.

`KS.pvalue.Height`
: p-value of Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Height procedure.

`CS.pvalue.Height`
: p-value of Chi-Square goodness-of-fit test conducted after application of Random Walk Height procedure.

`AD.result.Height`
: returns 0 if H0 is rejected and 1 otherwise in Anderson-Darling goodness-of-fit test conducted after application of Random Walk Height procedure.

`KS.result.Height`
: returns 0 if H0 is rejected and 1 otherwise in Kolmogorov-Smirnov goodness-of-fit test conducted after application of Random Walk Height procedure.

`CS.result.Height`
: returns 0 if H0 is rejected and 1 otherwise in Chi-Square goodness-of-fit test conducted after application of Random Walk Height procedure.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Doganaksoy, A., Calik, C., Sulak, F., Turan, M.S., New randomness tests using random walk, In: National Cryptology Symposium II, (2006), Ankara, Turkey.

Examples

```
RNGkind(kind = "Super-Duper")
B=64                        # Bit length is 64.
k=500                       # Generate 500 integers.
dat=round(runif(k,0,(2^B-1)))
x=sfsmisc::digitsBase(dat, base= 2, B) #Convert to base 2
alpha = 0.05
test=random.walk.tests(x, B, Excursion = TRUE, Expansion = TRUE, Height = TRUE, alpha)
print(test)
```

`Strlng2` *Stirling Number of The Second Kind*

Description

Asymptotically computes natural logarithm of Stirling numbers of the second kind for large values of inputs by the approach of Bleick and Wang (1954) and Temme (1993). For small or moderate values of inputs, this function is not as precise as available functions.

Usage

```
Strlng2(n, k, log = TRUE)
```

Arguments

`n`	positive integer greater than zero.
`k`	positive integer between 1 and `n`.
`log`	if `TRUE`, natural logarithm of the Stirling numbers of the second kind is returned.

Details

Due to the overflows in the calculation of large factorials, an asymptotic calculation of the Stirling numbers of the second kind is required. This function makes use of Lambert W function to calculate the Stirling numbers of the second kind with large values of `n` and `k`.

Value

`Stirling.num` the corresponding Stirling number of the second kind to the pair `(n,k)`.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Bleick, W.W., Wang, P.C.C., Asymptotics of Stirling Numbers of the Second Kind. Proceedings of the American Mathematical Society (1974), 42(2), 575–580.

Temme, N.M., Asymptotic estimates of Stirling numbers. Studies in Applied Mathematics (1993), 89, 233–243.

See Also

See also `Stirling2` function from the package **copula**.

Examples

```
# When n = 10 and k = 4, exact value is 34105
gmp::Stirling2(10,4)
Strlng2(10,4,log=FALSE)
# ---- Moderate values of n and k ----
# When n = 30 and k = 20, exact value is 581535955088511150
log(581535955088511150)-log(gmp::Stirling2(30,20))
log(581535955088511150)-Strlng2(30,20,log=TRUE)
# ---- Large values of n and k ----
gmp::Stirling2(50,10)
Strlng2(50,10,log=FALSE)
```

`TBT.criticalValue` *Critical value for Topological Binary Test*

Description

Approximately computes cumulative distribution function of the test statistic of the Topological Binary Test of Alcover et al. (2013) and finds the required critical value for the test.

Usage

```
TBT.criticalValue(m, k, alpha = 0.01, cdf = FALSE, exact = TRUE)
```

Arguments

`m` the length of words (B-bit) in Topological Binary Test.

`k` the number of words (integers) generated by the RNG of interest that will be tested.

`alpha` a predetermined value of type-I error with the default value of 0.05.

`cdf` if `TRUE`, the cumulative distribution function of the test statistic is stored and printed.

`exact` if `TRUE`, the function `Stirling2` from the package **gmp** is used to calculate the Stirling numbers of the second kind in the case that the function `Strlng2` from the package **CryptRndTest** returns a `NaN`. Otherwise, nothing is done for `NaN`'s generated by `Strlng2`.

Details

The function `TBT.criticalValue` lists the cumulative probabilities greater than zero if `cdf` is set to `TRUE`.

A correction factor is applied to improve accuracy of the the function `Strlng2` in the computation of probabilities. Accuracy of the computations decreases with increasing value of `m`.

Value

`prob` a vector containing the cumulative probabilities corresponding to the values in `value`.

`value` a vector containing the values of the test statistic.

`critical.value` critical value of the test statistic corresponding to `alpha`.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Alcover, P.M., Guillamon, A., Ruiz, M.C., A new randomness test for bit sequences. Informatica (2013), 24(3), 339–356.

Examples

```
# Critical values for the Topological Binary Test at 0.01 and 0.05 levels of significance.

TBT.criticalValue(m=8, k=256, alpha=0.01, cdf=FALSE, exact=FALSE)
TBT.criticalValue(m=8, k=256, alpha=0.05, cdf=FALSE, exact=FALSE)
```

toBaseTen *Convert form Base 2 to 10*

Description

Converts large integers form base 2 to base 10 using mpfr numbers by Pmpfr package.

Usage

```
toBaseTen(x, m = 128, prec = 256, toFile = FALSE, file)
```

Arguments

`x`	an m-by-k binary matrix including the data in base 2.
`m`	desired bit length in the output.
`prec`	precision of the calculations.
`toFile`	if `TRUE`, the resulting numbers are written on a file.
`file`	the path of the file to which the output is written.

Value

`dat`	an m-by-k matrix that contains the input data in base 10 format.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

toBaseTwo *Convert form Base 10 to 2*

Description

Converts large integers form base 10 to base 2 using mpfr numbers by Pmpfr package.

Usage

```
toBaseTwo(x, m = 128, prec = 512, num.CPU = 4)
```

Arguments

`x`	an mpfr vector including the data in base 10.
`m`	desired bit length in the output.
`prec`	precision of the calculations.
`num.CPU`	the number of CPUs that will be used in parallel computing.

Details

The function toBaseTwo utilizes the package parallel to make calculation utilizing parallel computing.

Value

`r.bit`	a list of mpfr numbers that contains the input data in base 2 format.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

`topological.binary` *Topological Binary Test*

Description

Performs Topological Binary Test of Alcover et al. (2013) to evaluate the randomness of an RNG. No additional goodness-of-fit test is applied after calculation of test statistic of Topological Binary Test.

Usage

```
topological.binary(x, B, alpha = 0.05, critical.value)
```

Arguments

`x`	a matrix that includes random data in base-2 format. See details for further information.
`B`	the length of words (B-bit).
`alpha`	a predetermined value of significance level with the default value of 0.05.
`critical.value`	a value used to decide whether to reject the null hypothesis at the significance level of `alpha`. See details for further information.

Details

The argument `x` should be entered as a matrix of bits of dimension Bxk, where k is the number of words (integers) generated by the RNG of interest.

The argument `critical.value` should be calculated regarding the value of `B`. For $B = 8, ..., 16$, values of `critical.value` are tabulated by Alcover et al. (2013) and calculation procedure of `critical.value` for the values greater than 16 is described therein. The tabulated values can be used if the number of words (k) is equal to 2^B. Otherwise, it should be calculated over the given cumulative distribution function by Alcover et al. (2013). For example, if $k = 10^4$, then `critical.value`$= 9245$ and if $k = 2 * 10^4$, then `critical.value`$= 19999$.

Topological binary test is itself constitutes a goodness-of-fit test based on the number of different B-bit patterns among the non-overlapping B-bit blocks composed of the input sequence of bits.

Value

`statistic` calculated value of the test statistic.

`result.TBT` returns 0 if H0 is rejected and 1 otherwise.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydarde@hacettepe.edu.tr>

References

Alcover, P.M., Guillamon, A., Ruiz, M.C., A new randomness test for bit sequences. Informatica (2013), 24(3), 339–356.

Examples

```
RNGkind(kind = "Super-Duper")
B=16                          # Bit length is 16.
k=5000                        # Generate 5000 integers.
dat=round(runif(k,0,(2^B-1)))
x=sfsmisc::digitsBase(dat, base= 2, B) #Convert to base 2
alpha = 0.05
critical.value=9245           #Obtained for B = 16
test=topological.binary(x, B, alpha, critical.value)
print(test)
```

Index

∗ **Anderson-Darling**
birthday.spacings, 6
CryptRndTest-package, 2
GCD.test, 12
random.walk.tests, 14
∗ **Chi-Square**
birthday.spacings, 6
book.stack, 7
GCD.test, 12
random.walk.tests, 14
∗ **Jarque-Bera**
CryptRndTest-package, 2
GCD.test, 12
∗ **Kolmogorov-Smirnov**
birthday.spacings, 6
CryptRndTest-package, 2
GCD.test, 12
random.walk.tests, 14
∗ **Stirling numbers of the second kind**
CryptRndTest-package, 2
∗ **approximation**
Strlng2, 17
∗ **chi-Square**
CryptRndTest-package, 2
∗ **critical value**
TBT.criticalValue, 18
∗ **goodness-of-fit test**
adaptive.chi.square, 4
birthday.spacings, 6
book.stack, 7
CryptRndTest-package, 2
GCD.test, 12
random.walk.tests, 14
∗ **greatest common divisor**
CryptRndTest-package, 2
∗ **large factorials**
Strlng2, 17
∗ **nonparametric**
adaptive.chi.square, 4
birthday.spacings, 6
book.stack, 7
CryptRndTest-package, 2
GCD.test, 12
random.walk.tests, 14
topological.binary, 21
∗ **randomness test**
adaptive.chi.square, 4
birthday.spacings, 6
book.stack, 7
CryptRndTest-package, 2
GCD.test, 12
random.walk.tests, 14
topological.binary, 21

adaptive.chi.square, *3*, 4, *9*
adaptive.chi.square.default *(CryptRndTest-internal)*, 9
adaptive.chi.square.main *(CryptRndTest-internal)*, 9

birthday.spacings, *3*, 6, *9*
birthday.spacings.default *(CryptRndTest-internal)*, 9
birthday.spacings.main *(CryptRndTest-internal)*, 9
book.stack, *3*, 7, *9*
book.stack.default *(CryptRndTest-internal)*, 9
book.stack.main *(CryptRndTest-internal)*, 9

check *(CryptRndTest-internal)*, 9
CryptRndTest *(CryptRndTest-package)*, 2
CryptRndTest-internal, 9
CryptRndTest-package, 2

dogumGunuAraliklari *(CryptRndTest-internal)*, 9

GCD, *3*, *9*, 9, *13*

GCD.big, *3*, *9*, 10
GCD.q, *3*, *9*, 11
GCD.test, *3*, *9*, 12
GCD.test.default
(CryptRndTest-internal), 9
GCD.test.main (CryptRndTest-internal), 9

KSADCHRY (CryptRndTest-internal), 9
KSADdga (CryptRndTest-internal), 9

print.CryptRndTest, 14

Random.walk.D (CryptRndTest-internal), 9
Random.walk.G (CryptRndTest-internal), 9
random.walk.tests, *3*, *9*, 14
random.walk.tests.default
(CryptRndTest-internal), 9
random.walk.tests.main
(CryptRndTest-internal), 9
Random.walk.Y (CryptRndTest-internal), 9

Stirling2, *3*, *9*
Strlng2, *3*, *9*, 17

TBT.criticalValue, 18
toBaseTen, 20
toBaseTwo, 20
topological.binary, *3*, *9*, 21
topological.binary.default
(CryptRndTest-internal), 9
topological.binary.main
(CryptRndTest-internal), 9