NAG Library Function Document

nag_1_sample_ks_test (g08cbc)

1 Purpose

nag_1_sample_ks_test (g08cbc) performs the one sample Kolmogorov–Smirnov test, using one of the distributions provided.

2 Specification

```
#include <nag.h>
#include <nagg08.h>
```

```
void nag_1_sample_ks_test (Integer n, const double x[],
     Nag_Distributions dist, double par[], Nag_ParaEstimates estima,
     Nag_TestStatistics ntype, double *d, double *z, double *p,
     NagError *fail)
```

3 Description

The data consist of a single sample of n observations denoted by $x_1, x_2, \ldots, x_n$. Let $S_n\left(x_{(i)}\right)$ and $F_0\left(x_{(i)}\right)$ represent the sample cumulative distribution function and the theoretical (null) cumulative distribution function respectively at the point $x_{(i)}$ where $x_{(i)}$ is the ith smallest sample observation.

The Kolmogorov–Smirnov test provides a test of the null hypothesis H_0: the data are a random sample of observations from a theoretical distribution specified by you against one of the following alternative hypotheses:

(i) H_1: the data cannot be considered to be a random sample from the specified null distribution.

(ii) H_2: the data arise from a distribution which dominates the specified null distribution. In practical terms, this would be demonstrated if the values of the sample cumulative distribution function $S_n(x)$ tended to exceed the corresponding values of the theoretical cumulative distribution function $F_0(x)$.

(iii) H_3: the data arise from a distribution which is dominated by the specified null distribution. In practical terms, this would be demonstrated if the values of the theoretical cumulative distribution function $F_0(x)$ tended to exceed the corresponding values of the sample cumulative distribution function $S_n(x)$.

One of the following test statistics is computed depending on the particular alternative null hypothesis specified (see the description of the argument **ntype** in Section 5).

For the alternative hypothesis H_1.

D_n – the largest absolute deviation between the sample cumulative distribution function and the theoretical cumulative distribution function. Formally $D_n = \max\left\{D_n^+, D_n^-\right\}$.

For the alternative hypothesis H_2.

D_n^+ – the largest positive deviation between the sample cumulative distribution function and the theoretical cumulative distribution function. Formally $D_n^+ = \max\left\{S_n\left(x_{(i)}\right) - F_0\left(x_{(i)}\right), 0\right\}$ for both discrete and continuous null distributions.

For the alternative hypothesis H_3.

D_n^- – the largest positive deviation between the theoretical cumulative distribution function and the sample cumulative distribution function. Formally if the null distribution is discrete then $D_n^- = \max\left\{F_0\left(x_{(i)}\right) - S_n\left(x_{(i)}\right), 0\right\}$ and if the null distribution is continuous then $D_n^- = \max\left\{F_0\left(x_{(i)}\right) - S_n\left(x_{(i-1)}\right), 0\right\}$.

The standardized statistic $Z = D \times \sqrt{n}$ is also computed where D may be D_n, D_n^+ or D_n^- depending on the choice of the alternative hypothesis. This is the standardized value of D with no correction for continuity applied and the distribution of Z converges asymptotically to a limiting distribution, first derived by Kolmogorov (1933), and then tabulated by Smirnov (1948). The asymptotic distributions for the one-sided statistics were obtained by Smirnov (1933).

The probability, under the null hypothesis, of obtaining a value of the test statistic as extreme as that observed, is computed. If $n \leq 100$ an exact method given by Conover (1980), is used. Note that the method used is only exact for continuous theoretical distributions and does not include Conover's modification for discrete distributions. This method computes the one-sided probabilities. The two-sided probabilities are estimated by doubling the one-sided probability. This is a good estimate for small p, that is $p \leq 0.10$, but it becomes very poor for larger p. If $n > 100$ then p is computed using the Kolmogorov–Smirnov limiting distributions, see Feller (1948), Kendall and Stuart (1973), Kolmogorov (1933), Smirnov (1933) and Smirnov (1948).

4 References

Conover W J (1980) *Practical Nonparametric Statistics* Wiley

Feller W (1948) On the Kolmogorov–Smirnov limit theorems for empirical distributions *Ann. Math. Statist.* **19** 179–181

Kendall M G and Stuart A (1973) *The Advanced Theory of Statistics (Volume 2)* (3rd Edition) Griffin

Kolmogorov A N (1933) Sulla determinazione empirica di una legge di distribuzione *Giornale dell' Istituto Italiano degli Attuari* **4** 83–91

Siegel S (1956) *Non-parametric Statistics for the Behavioral Sciences* McGraw–Hill

Smirnov N (1933) Estimate of deviation between empirical distribution functions in two independent samples *Bull. Moscow Univ.* **2(2)** 3–16

Smirnov N (1948) Table for estimating the goodness of fit of empirical distributions *Ann. Math. Statist.* **19** 279–281

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of observations in the sample.

Constraint: **n** ≥ 3.

2: **x**[**n**] – const double *Input*

On entry: the sample observations $x_1, x_2, \ldots, x_n$.

Constraint: the sample observations supplied must be consistent, in the usual manner, with the null distribution chosen, as specified by the arguments **dist** and **par**. For further details see Section 9.

3: **dist** – Nag_Distributions *Input*

On entry: the theoretical (null) distribution from which it is suspected the data may arise.

dist = Nag_Uniform
The uniform distribution over (a, b).

dist = Nag_Normal
The Normal distribution with mean μ and variance σ^2.

dist = Nag_Gamma
The gamma distribution with shape parameterα and scale parameter β, where the mean $= \alpha\beta$.

dist = Nag_Beta
The beta distribution with shape parameters α and β, where the mean $= \alpha/(\alpha + \beta)$.

dist = Nag_Binomial
The binomial distribution with the number of trials, m, and the probability of a success, p.

dist = Nag_Exponential
The exponential distribution with parameter λ, where the mean $= 1/\lambda$.

dist = Nag_Poisson
The Poisson distribution with parameter μ, where the mean $= \mu$.

dist = Nag_NegBinomial
The negative binomial distribution with the number of trials, m, and the probability of success, p.

dist = Nag_GenPareto
The generalized Pareto distribution with shape parameter ξ and scale β.

Constraint: **dist** = Nag_Uniform, Nag_Normal, Nag_Gamma, Nag_Beta, Nag_Binomial, Nag_Exponential, Nag_Poisson, Nag_NegBinomial or Nag_GenPareto.

4: **par**[**2**] – double *Input/Output*

On entry: if **estima** = Nag_ParaSupplied, **par** must contain the known values of the parameter(s) of the null distribution as follows.

If a uniform distribution is used, then **par**[0] and **par**[1] must contain the boundaries a and b respectively.

If a Normal distribution is used, then **par**[0] and **par**[1] must contain the mean, μ, and the variance, σ^2, respectively.

If a gamma distribution is used, then **par**[0] and **par**[1] must contain the parameters α and β respectively.

If a beta distribution is used, then **par**[0] and **par**[1] must contain the parameters α and β respectively.

If a binomial distribution is used, then **par**[0] and **par**[1] must contain the parameters m and p respectively.

If an exponential distribution is used, then **par**[0] must contain the parameter λ.

If a Poisson distribution is used, then **par**[0] must contain the parameter μ.

If a negative binomial distribution is used, **par**[0] and **par**[1] must contain the parameters m and p respectively.

If a generalized Pareto distribution is used, **par**[0] and **par**[1] must contain the parameters ξ and β respectively.

If **estima** = Nag_ParaEstimated, **par** need not be set except when the null distribution requested is either the binomial or the negative binomial distribution in which case **par**[0] must contain the parameter m.

On exit: if **estima** = Nag_ParaSupplied, **par** is unchanged; if **estima** = Nag_ParaEstimated, and **dist** = Nag_Binomial or **dist** = Nag_NegBinomial then **par**[1] is estimated from the data; otherwise **par**[0] and **par**[1] are estimated from the data.

Constraints:

if **dist** = Nag_Uniform, **par**[0] $<$ **par**[1];
if **dist** = Nag_Normal, **par**[1] > 0.0;
if **dist** = Nag_Gamma, **par**[0] > 0.0 and **par**[1] > 0.0;
if **dist** = Nag_Beta, **par**[0] > 0.0 and **par**[1] > 0.0 and **par**[0] $\leq 10^6$ and **par**[1] $\leq 10^6$;

if **dist** = Nag_Binomial, **par**[0] $\geq$ 1.0 and 0.0 < **par**[1] < 1.0 and
par[0] $\times$ **par**[1] $\times$ (1.0 − **par**[1]) $\leq 10^6$ and **par**[0] < 1/eps, where
eps = ***machine precision***, see nag_machine_precision (X02AJC);
if **dist** = Nag_Exponential, **par**[0] > 0.0;
if **dist** = Nag_Poisson, **par**[0] > 0.0 and **par**[0] $\leq 10^6$;
if **dist** = Nag_NegBinomial, **par**[0] $\geq$ 1.0 and 0.0 < **par**[1] < 1.0 and
par[0] $\times$ (1.0 − **par**[1])/(**par**[1] $\times$ **par**[1]) $\leq 10^6$ and **par**[0] < 1/eps, where
eps = ***machine precision***, see nag_machine_precision (X02AJC);
if **dist** = Nag_GenPareto, **par**[1] > 0.

5: **estima** – Nag_ParaEstimates *Input*

On entry: **estima** must specify whether values of the parameters of the null distribution are known or are to be estimated from the data.

estima = Nag_ParaSupplied
Values of the parameters will be supplied in the array **par** described above.

estima = Nag_ParaEstimated
Parameters are to be estimated from the data except when the null distribution requested is the binomial distribution or the negative binomial distribution in which case the first parameter, m, must be supplied in **par**[0] and only the second parameter, p, is estimated from the data.

Constraint: **estima** = Nag_ParaSupplied or Nag_ParaEstimated.

6: **ntype** – Nag_TestStatistics *Input*

On entry: the test statistic to be calculated, i.e., the choice of alternative hypothesis.

ntype = Nag_TestStatisticsDAbs
Computes D_n, to test H_0 against H_1,

ntype = Nag_TestStatisticsDPos
Computes D_n^+, to test H_0 against H_2,

ntype = Nag_TestStatisticsDNeg
Computes D_n^-, to test H_0 against H_3.

Constraint: **ntype** = Nag_TestStatisticsDAbs, Nag_TestStatisticsDPos or Nag_TestStatisticsDNeg.

7: **d** – double * *Output*

On exit: the Kolmogorov–Smirnov test statistic (D_n, D_n^+ or D_n^- according to the value of **ntype**).

8: **z** – double * *Output*

On exit: a standardized value, Z, of the test statistic, D, without any correction for continuity.

9: **p** – double * *Output*

On exit: the probability, p, associated with the observed value of D where D may be D_n, D_n^+ or D_n^- depending on the value of **ntype** (see Section 3).

10: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_G08CB_DATA

On entry, **dist** $=$ Nag_Beta and at least one observation is illegal.
Constraint: $0 \leq \mathbf{x}[i-1] \leq 1$, for $i = 1, 2, \ldots, \mathbf{n}$.

On entry, **dist** $=$ Nag_Binomial and all observations are zero or m.
Constraint: at least one $0.0 < \mathbf{x}[i-1] < \mathbf{par}[0]$, for $i = 1, 2, \ldots, \mathbf{n}$.

On entry, **dist** $=$ Nag_Binomial and at least one observation is illegal.
Constraint: $0 \leq \mathbf{x}[i-1] \leq \mathbf{par}[0]$, for $i = 1, 2, \ldots, \mathbf{n}$.

On entry, **dist** $=$ Nag_Exponential or Nag_Poisson and all observations are zero.
Constraint: at least one $\mathbf{x}[i-1] > 0$, for $i = 1, 2, \ldots, \mathbf{n}$.

On entry, **dist** $=$ Nag_Gamma, Nag_Exponential, Nag_Poisson, Nag_NegBinomial or Nag_GenPareto and at least one observation is negative.
Constraint: $\mathbf{x}[i-1] \geq 0$, for $i = 1, 2, \ldots, \mathbf{n}$.

On entry, **dist** $=$ Nag_Uniform and at least one observation is illegal.
Constraint: $\mathbf{par}[0] \leq \mathbf{x}[i-1] \leq \mathbf{par}[1]$, for $i = 1, 2, \ldots, \mathbf{n}$.

NE_G08CB_PARAM

On entry, **dist** $=$ Nag_Binomial and $m = \mathbf{par}[0] = \langle value \rangle$.
Note that m must always be supplied.
Constraint: for the binomial distribution, $1 \leq \mathbf{par}[0] < 1/\text{eps}$, where eps $=$ ***machine precision***, see nag_machine_precision (X02AJC).

On entry, **dist** $=$ Nag_GenPareto and **estima** $=$ Nag_ParaEstimated.
The parameter estimates are invalid; the data may not be from the generalized Pareto distribution.

On entry, **dist** $=$ Nag_NegBinomial and $m = \mathbf{par}[0] = \langle value \rangle$.
Note that m must always be supplied.
Constraint: for the negative binomial distribution, $1 \leq \mathbf{par}[0] < 1/\text{eps}$, where eps $=$ ***machine precision***, see nag_machine_precision (X02AJC).

On entry, **estima** $=$ Nag_ParaSupplied and $\mathbf{par}[0] = \langle value \rangle$; $\mathbf{par}[1] = \langle value \rangle$.
Constraint: for the beta distribution, $0 < \mathbf{par}[0]$ and $\mathbf{par}[1] \leq 1000000$.

On entry, **estima** $=$ Nag_ParaSupplied and $\mathbf{par}[0] = \langle value \rangle$; $\mathbf{par}[1] = \langle value \rangle$.
Constraint: for the gamma distribution, $\mathbf{par}[0]$ and $\mathbf{par}[1] > 0$.

On entry, **estima** $=$ Nag_ParaSupplied and $\mathbf{par}[0] = \langle value \rangle$; $\mathbf{par}[1] = \langle value \rangle$.
Constraint: for the generalized Pareto distribution with $\mathbf{par}[0] < 0$, $0 \leq \mathbf{x}[i-1] \leq -\mathbf{par}[1]/\mathbf{par}[0]$, for $i = 1, 2, \ldots, \mathbf{n}$.

On entry, **estima** $=$ Nag_ParaSupplied and $\mathbf{par}[0] = \langle value \rangle$; $\mathbf{par}[1] = \langle value \rangle$.
Constraint: for the uniform distribution, $\mathbf{par}[0] < \mathbf{par}[1]$.

On entry, **estima** $=$ Nag_ParaSupplied and $\mathbf{par}[0] = \langle value \rangle$.
Constraint: for the exponential distribution, $\mathbf{par}[0] > 0$.

On entry, **estima** $=$ Nag_ParaSupplied and $\mathbf{par}[0] = \langle value \rangle$.
Constraint: for the Poisson distribution, $0 < \mathbf{par}[0] < 1000000$.

On entry, **estima** = Nag_ParaSupplied and **par**[1] = ⟨*value*⟩.
Constraint: for the binomial distribution, $0 <$ **par**[1] < 1.

On entry, **estima** = Nag_ParaSupplied and **par**[1] = ⟨*value*⟩.
Constraint: for the generalized Pareto distribution, **par**[1] > 0.

On entry, **estima** = Nag_ParaSupplied and **par**[1] = ⟨*value*⟩.
Constraint: for the negative binomial distribution, $0 <$ **par**[1] < 1.

On entry, **estima** = Nag_ParaSupplied and **par**[1] = ⟨*value*⟩.
Constraint: for the Normal distribution, **par**[1] > 0.

NE_G08CB_SAMPLE

On entry, **dist** = Nag_Uniform, Nag_Normal, Nag_Gamma, Nag_Beta or Nag_GenPareto, **estima** = Nag_ParaEstimated and the whole sample is constant. Thus the variance is zero.

NE_G08CB_VARIANCE

On entry, **dist** = Nag_Binomial, **par**[0] = ⟨*value*⟩, **par**[1] = ⟨*value*⟩.
The variance **par**[0] × **par**[1] × (1 − **par**[1]) exceeds 1000000.

On entry, **dist** = Nag_NegBinomial, **par**[0] = ⟨*value*⟩, **par**[1] = ⟨*value*⟩.
The variance **par**[0] × (1 − **par**[1])/(**par**[1] × **par**[1]) exceeds 1000000.

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 3.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The approximation for p, given when $n > 100$, has a relative error of at most 2.5% for most cases. The two-sided probability is approximated by doubling the one-sided probability. This is only good for small p, i.e., $p < 0.10$ but very poor for large p. The error is always on the conservative side, that is the tail probability, p, is over estimated.

8 Parallelism and Performance

nag_1_sample_ks_test (g08cbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by nag_1_sample_ks_test (g08cbc) increases with n until $n > 100$ at which point it drops and then increases slowly with n. The time may also depend on the choice of null distribution and on whether or not the parameters are to be estimated.

The data supplied in the argument **x** must be consistent with the chosen null distribution as follows:

when **dist** = Nag_Uniform, then **par**$[0] \leq x_i \leq$ **par**$[1]$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_Normal, then there are no constraints on the x_i's;

when **dist** = Nag_Gamma, then $x_i \geq 0.0$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_Beta, then $0.0 \leq x_i \leq 1.0$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_Binomial, then $0.0 \leq x_i \leq$ **par**$[0]$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_Exponential, then $x_i \geq 0.0$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_Poisson, then $x_i \geq 0.0$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_NegBinomial, then $x_i \geq 0.0$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_GenPareto and **par**$[0] \geq 0.0$, then $x_i \geq 0.0$, for $i = 1, 2, \ldots, n$;

when **dist** = Nag_GenPareto and **par**$[0] < 0.0$, then $0.0 \leq x_i \leq -$**par**$[1]/$**par**$[0]$, for $i = 1, 2, \ldots, n$.

10 Example

The following example program reads in a set of data consisting of 30 observations. The Kolmogorov–Smirnov test is then applied twice, firstly to test whether the sample is taken from a uniform distribution, $U(0, 2)$, and secondly to test whether the sample is taken from a Normal distribution where the mean and variance are estimated from the data. In both cases we are testing against H_1; that is, we are doing a two tailed test. The values of **d**, **z** and **p** are printed for each case.

10.1 Program Text

```
/* nag_1_sample_ks_test (g08cbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg08.h>

int main(void)
{
  Integer exit_status = 0;
  Integer i, n, np;
  double d, p, *par = 0, *x = 0, z;
  char nag_enum_arg[40];
  Nag_TestStatistics ntype;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_1_sample_ks_test (g08cbc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
```

```
#else
  scanf("%*[^\n]");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &n);
#else
  scanf("%" NAG_IFMT "", &n);
#endif
  x = NAG_ALLOC(n, double);

  printf("\n");
  for (i = 1; i <= n; ++i)
#ifdef _WIN32
    scanf_s("%lf", &x[i - 1]);
#else
    scanf("%lf", &x[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &np);
#else
  scanf("%" NAG_IFMT "", &np);
#endif
  if (!(par = NAG_ALLOC(np, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  for (i = 1; i <= np; ++i)
#ifdef _WIN32
    scanf_s("%lf", &par[i - 1]);
#else
    scanf("%lf", &par[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  ntype = (Nag_TestStatistics) nag_enum_name_to_value(nag_enum_arg);

  /* nag_1_sample_ks_test (g08cbc).
   * Performs the one-sample Kolmogorov-Smirnov test for
   * standard distributions
   */
  nag_1_sample_ks_test(n, x, Nag_Uniform, par, Nag_ParaSupplied, ntype, &d,
                       &z, &p, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_1_sample_ks_test (g08cbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  printf("Test against uniform distribution on (0,2)\n");
  printf("\n");
  printf("Test statistic D = %8.4f\n", d);
  printf("Z statistic      = %8.4f\n", z);
  printf("Tail probability = %8.4f\n", p);
  printf("\n");
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &np);
#else
  scanf("%" NAG_IFMT "", &np);
#endif
  for (i = 1; i <= np; ++i)
#ifdef _WIN32
    scanf_s("%lf", &par[i - 1]);
```

```
#else
    scanf("%lf", &par[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  ntype = (Nag_TestStatistics) nag_enum_name_to_value(nag_enum_arg);

  /* nag_1_sample_ks_test (g08cbc), see above. */
  nag_1_sample_ks_test(n, x, Nag_Normal, par, Nag_ParaEstimated, ntype, &d,
                       &z, &p, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_1_sample_ks_test (g08cbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("Test against Normal distribution with parameters estimated"
         " from the data\n\n");
  printf("Mean = %6.4f  and variance = %6.4f\n", par[0], par[1]);
  printf("Test statistic D = %8.4f\n", d);
  printf("Z statistic      = %8.4f\n", z);
  printf("Tail probability = %8.4f\n", p);

END:
  NAG_FREE(x);
  NAG_FREE(par);

  return exit_status;
}
```

10.2 Program Data

```
nag_1_sample_ks_test (g08cbc) Example Program Data
 30
 0.01 0.30 0.20 0.90 1.20 0.09 1.30 0.18 0.90 0.48
 1.98 0.03 0.50 0.07 0.70 0.60 0.95 1.00 0.31 1.45
 1.04 1.25 0.15 0.75 0.85 0.22 1.56 0.81 0.57 0.55
 2  0.0  2.0  Nag_TestStatisticsDAbs
 2  0.0  1.0  Nag_TestStatisticsDAbs
```

10.3 Program Results

```
nag_1_sample_ks_test (g08cbc) Example Program Results

Test against uniform distribution on (0,2)

Test statistic D =   0.2800
Z statistic      =   1.5336
Tail probability =   0.0143

Test against Normal distribution with parameters estimated from the data

Mean = 0.6967  and variance = 0.2564
Test statistic D =   0.1108
Z statistic      =   0.6068
Tail probability =   0.8925
```