NAG Library Function Document

nag_prob_normal (g01eac)

1 Purpose

nag_prob_normal (g01eac) returns a one or two tail probability for the standard Normal distribution.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
double nag_prob_normal (Nag_TailProbability tail, double x, NagError *fail)
```

3 Description

The lower tail probability for the standard Normal distribution, $P(X \leq x)$ is defined by:

$$P(X \leq x) = \int_{-\infty}^{x} Z(X)\, dX,$$

where

$$Z(X) = \frac{1}{\sqrt{2\pi}} e^{-X^2/2}, -\infty < X < \infty.$$

The relationship

$$P(X \leq x) = \tfrac{1}{2}\text{erfc}\left(\frac{-x}{\sqrt{2}}\right)$$

is used, where erfc is the complementary error function, and is computed using nag_erfc (s15adc). For the upper tail probability the relationship $P(X \geq x) = P(X \leq -x)$ is used and for the two tail significance level probability twice the probability obtained from the absolute value of x is returned.

When the two tail confidence probability is required the relationship

$$P(X \leq |x|) - P(X \leq -|x|) = \text{erf}\left(\frac{|x|}{\sqrt{2}}\right),$$

is used, where erf is the error function, and is computed using nag_erf (s15aec).

4 References

Abramowitz M and Stegun I A (1972) *Handbook of Mathematical Functions* (3rd Edition) Dover Publications

Hastings N A J and Peacock J B (1975) *Statistical Distributions* Butterworth

5 Arguments

1: **tail** – Nag_TailProbability *Input*

On entry: indicates which tail the returned probability should represent.

tail = Nag_LowerTail
The lower tail probability is returned, i.e., $P(X \leq x)$.

tail = Nag_UpperTail
The upper tail probability is returned, i.e., $P(X \geq x)$.

tail = Nag_TwoTailSignif
The two tail (significance level) probability is returned, i.e., $P(X \geq |x|) + P(X \leq -|x|)$.

tail = Nag_TwoTailConfid
The two tail (confidence interval) probability is returned, i.e., $P(X \leq |x|) - P(X \leq -|x|)$.

Constraint: **tail** = Nag_LowerTail, Nag_UpperTail, Nag_TwoTailSignif or Nag_TwoTailConfid.

2: **x** – double *Input*

On entry: x, the value of the standard Normal variate.

3: **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 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

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 2.7.6 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 2.7.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Accuracy is limited by ***machine precision***. For detailed error analysis see nag_erfc (s15adc) and nag_erf (s15aec).

8 Parallelism and Performance

nag_prob_normal (g01eac) is not threaded in any implementation.

9 Further Comments

None.

10 Example

Four values of **tail** and **x** are input and the probabilities calculated and printed.

10.1 Program Text

```
/* nag_prob_normal (g01eac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <nagg01.h>

int main(void)
{
  Integer exit_status = 0;
  double prob;
  double x;
  Integer i;
  char nag_enum_arg[40];
  Nag_TailProbability tail;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_prob_normal (g01eac) Example Program Results\n");

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

  printf("\n     Tail               X      Probability \n\n");
  for (i = 1; i <= 4; ++i) {
#ifdef _WIN32
    scanf_s("%39s %lf ", nag_enum_arg, (unsigned)_countof(nag_enum_arg),
            &x);
#else
    scanf("%39s %lf ", nag_enum_arg, &x);
#endif
    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    tail = (Nag_TailProbability) nag_enum_name_to_value(nag_enum_arg);

    /* nag_prob_normal (g01eac).
     * Probabilities for the standard Normal distribution
     */
    prob = nag_prob_normal(tail, x, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_prob_normal (g01eac).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
    printf(" %-17s    %4.2f       %6.4f\n", nag_enum_arg, x, prob);
  }

END:
  return exit_status;
}
```

10.2 Program Data

```
nag_prob_normal (g01eac) Example Program Data
Nag_LowerTail      1.96
Nag_UpperTail      1.96
Nag_TwoTailConfid  1.96
Nag_TwoTailSignif  1.96
```

10.3 Program Results

```
nag_prob_normal (g01eac) Example Program Results

     Tail               X      Probability

 Nag_LowerTail        1.96       0.9750
 Nag_UpperTail        1.96       0.0250
 Nag_TwoTailConfid    1.96       0.9500
 Nag_TwoTailSignif    1.96       0.0500
```