NAG Library Function Document

nag_frequency_table (g01aec)

1 Purpose

nag_frequency_table (g01aec) constructs a frequency distribution of a variable, according to either user-supplied, or function-calculated class boundary values.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
```

```
void nag_frequency_table (Integer n, const double x[], Integer num_class,
     Nag_ClassBoundary classb, double cint[], Integer ifreq[], double *xmin,
     double *xmax, NagError *fail)
```

3 Description

The data consists of a sample of n observations of a continuous variable, denoted by x_i, for $i = 1, 2, \ldots, n$. Let $a = \min(x_1, \ldots, x_n)$ and $b = \max(x_1, \ldots, x_n)$.

nag_frequency_table (g01aec) constructs a frequency distribution with $k(> 1)$ classes denoted by f_i, for $i = 1, 2, \ldots, k$.

The boundary values may be either user-supplied, or function-calculated, and are denoted by y_j, for $j = 1, 2, \ldots, k - 1$.

If the boundary values of the classes are to be function-calculated, then they are determined in one of the following ways:

(a) if $k > 2$, the range of x values is divided into $k - 2$ intervals of equal length, and two extreme intervals, defined by the class boundary values $y_1, y_2, \ldots, y_{k-1}$;

(b) if $k = 2$, $y_1 = \frac{1}{2}(a + b)$.

However formed, the values $y_1, \ldots, y_{k-1}$ are assumed to be in ascending order. The class frequencies are formed with

$$f_1 = \text{ the number of } x \text{ values in the interval } (-\infty, y_1)$$

$$f_i = \text{ the number of } x \text{ values in the interval } [y_{i-1}, y_i), \quad i = 2, \ldots, k - 1$$

$$f_k = \text{ the number of } x \text{ values in the interval } [y_{k-1}, \infty),$$

where [means inclusive, and) means exclusive. If the class boundary values are function-calculated and $k > 2$, then $f_1 = f_k = 0$, and y_1 and y_{k-1} are chosen so that $y_1 < a$ and $y_{k-1} > b$.

If a frequency distribution is required for a discrete variable, then it is suggested that you supply the class boundary values; function-calculated boundary values may be slightly imprecise (due to the adjustment of y_1 and y_{k-1} outlined above) and cause values very close to a class boundary to be assigned to the wrong class.

4 References

None.

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of observations.

Constraint: $\mathbf{n} \geq 1$.

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

On entry: the sample of observations of the variable for which the frequency distribution is required, x_i, for $i = 1, 2, \ldots, n$. The values may be in any order.

3: **num_class** – Integer *Input*

On entry: k, the number of classes desired in the frequency distribution. Whether or not class boundary values are user-supplied, **num_class** must include the two extreme classes which stretch to $\pm\infty$.

Constraint: **num_class** ≥ 2.

4: **classb** – Nag_ClassBoundary *Input*

On entry: indicates whether class boundary values are to be calculated within nag_frequency_table (g01aec), or are supplied by you.

If **classb** = Nag_ClassBoundaryComp, then the class boundary values are to be calculated within the function.

If **classb** = Nag_ClassBoundaryUser, they are user-supplied.

Constraint: **classb** = Nag_ClassBoundaryComp or Nag_ClassBoundaryUser.

5: **cint**[**num_class**] – double *Input/Output*

On entry: if **classb** = Nag_ClassBoundaryComp, then the elements of **cint** need not be assigned values, as nag_frequency_table (g01aec) calculates $k - 1$ class boundary values.

If **classb** = Nag_ClassBoundaryUser, the first $k - 1$ elements of **cint** must contain the class boundary values you supplied, in ascending order.

On exit: the first $k - 1$ elements of **cint** contain the class boundary values in ascending order.

Constraint: if **classb** = Nag_ClassBoundaryUser, **cint**$[i - 1] <$ **cint**$[i]$, for $i = 1, 2, \ldots, k - 2$.

6: **ifreq**[**num_class**] – Integer *Output*

On exit: the elements of **ifreq** contain the frequencies in each class, f_i, for $i = 1, 2, \ldots, k$. In particular **ifreq**[0] contains the frequency of the class up to **cint**[0], f_1, and **ifreq**$[k - 1]$ contains the frequency of the class greater than **cint**$[k - 2]$, f_k.

7: **xmin** – double * *Output*

On exit: the smallest value in the sample, a.

8: **xmax** – double * *Output*

On exit: the largest value in the sample, b.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

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

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

On entry, **num_class** = ⟨*value*⟩.
Constraint: **num_class** ≥ 2.

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.

NE_NOT_STRICTLY_INCREASING

On entry, **cint**[⟨*value*⟩] = ⟨*value*⟩ and **cint**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **cint**[⟨*value*⟩] < **cint**[⟨*value*⟩].

7 Accuracy

The method used is believed to be stable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_frequency_table (g01aec) increases with **num_class** and **n**. It also depends on the distribution of the sample observations.

10 Example

This example summarises a number of datasets. For each dataset the sample observations and optionally class boundary values are read. nag_frequency_table (g01aec) is then called and the frequency distribution and largest and smallest observations printed.

10.1 Program Text

```
/* nag_frequency_table (g01aec) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * Mark 6a revised, 2001.
 */

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

int main(void)
{
  Integer           exit_status = 0, i, j, *jfreq = 0, n, nprob, num_class;
  char              nag_enum_arg[40];
  Nag_ClassBoundary class;
  double            *a = 0, *c = 0, xmax, xmin;
```

```
  NagError          fail;

  INIT_FAIL(fail);

  printf("nag_frequency_table (g01aec) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld", &nprob);
  for (i = 1; i <= nprob; ++i)
    {
      scanf("%ld %39s %ld", &n, nag_enum_arg, &num_class);
      /* nag_enum_name_to_value (x04nac).
       * Converts NAG enum member name to value
       */
      class = (Nag_ClassBoundary) nag_enum_name_to_value(nag_enum_arg);
      if (!(a = NAG_ALLOC(n, double))
         || !(c = NAG_ALLOC(num_class-1, double))
         || !(jfreq = NAG_ALLOC(num_class, Integer)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      for (j = 1; j <= n; ++j)
        scanf("%lf", &a[j - 1]);
      printf("Problem %ld\n", i);
      printf("Number of cases %ld\n", n);
      printf(
              "Number of classes, including extreme classes %ld\n",
              num_class);
      if (class != Nag_ClassBoundaryUser)
        printf("Routine-supplied class boundaries\n\n");
      else
        {
          for (j = 1; j <= num_class-1; ++j)
            scanf("%lf", &c[j - 1]);
          printf("User-supplied class boundaries\n");
        }
      /* nag_frequency_table (g01aec).
       * Frequency table from raw data
       */
      nag_frequency_table(n, a, num_class, class, c, jfreq, &xmin, &xmax,
                          &fail);
      if (fail.code == NE_NOERROR)
        {
          printf("Successful call of "
                  "nag_frequency_table (g01aec)\n\n");
          printf("*** Frequency  distribution ***\n\n");
          printf("       Class            Frequency\n\n");
          printf("   Up to    %8.2f %11ld\n", c[0], jfreq[0]);
          if (num_class-1 > 1)
            {
              for (j = 2; j <= num_class-1; ++j)
                printf("%8.2f to %8.2f %11ld\n", c[j - 2],
                        c[j - 1], jfreq[j - 1]);
            }
          printf("%8.2f     and over  %9ld\n\n",
                  c[num_class - 2], jfreq[num_class-1]);
          printf("Total frequency = %ld\n", n);
          printf("Minimum = %9.2f\n", xmin);
          printf("Maximum = %9.2f\n", xmax);
        }
      else
        {
          printf("Error from nag_frequency_table (g01aec).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      NAG_FREE(a);
```

```
      NAG_FREE(c);
      NAG_FREE(jfreq);
    }

 END:
  NAG_FREE(a);
  NAG_FREE(c);
  NAG_FREE(jfreq);

  return exit_status;
}
```

10.2 Program Data

```
nag_frequency_table (g01aec) Example Program Data
 1
 70  Nag_ClassBoundaryComp  7
22.3   21.6   22.6   22.4   22.4   22.4   22.1   21.9   23.1   23.4
23.4   22.6   22.5   22.5   22.1   22.6   22.3   22.4   21.8   22.3
22.1   23.6   20.8   22.2   23.1   21.1   21.7   21.4   21.6   22.5
21.2   22.6   22.2   22.2   21.4   21.7   23.2   23.1   22.3   22.3
21.1   21.4   21.5   21.8   22.8   21.4   20.7   21.6   23.2   23.6
22.7   21.7   23.0   21.9   22.6   22.1   22.2   23.4   21.5   23.0
22.8   21.4   23.2   21.8   21.2   22.0   22.4   22.8   23.2   23.6
```

10.3 Program Results

```
nag_frequency_table (g01aec) Example Program Results

Problem 1
Number of cases 70
Number of classes, including extreme classes 7
Routine-supplied class boundaries

Successful call of nag_frequency_table (g01aec)

*** Frequency  distribution ***

       Class            Frequency

   Up to       20.70          0
   20.70 to    21.28          6
   21.28 to    21.86         16
   21.86 to    22.44         21
   22.44 to    23.02         14
   23.02 to    23.60         13
   23.60     and over         0

Total frequency = 70
Minimum =     20.70
Maximum =     23.60
```