NAG Library Function Document

nag_binary_factor_service (g11sbc)

1 Purpose

nag_binary_factor_service (g11sbc) is a service function which may be used prior to calling nag_binary_factor (g11sac) to calculate the frequency distribution of a set of dichotomous score patterns.

2 Specification

```
#include <nag.h>
#include <nagg11.h>
void nag_binary_factor_service (Nag_OrderType order, Integer p, Integer n,
     Integer *ns, Nag_Boolean x[], Integer pdx, Integer irl[],
     NagError *fail)
```

3 Description

When each of n individuals responds to each of p dichotomous variables the data assumes the form of the matrix X defined below

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} = \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_n \end{bmatrix},$$

where the x take the value of 0 or 1 and $\underline{x}_l = \left(x_{l1}, x_{l2}, \dots, x_{lp}\right)$, for $l = 1, 2, \dots, n$, denotes the score pattern of the lth individual. nag_binary_factor_service (g11sbc) calculates the number of different score patterns, s, and the frequency with which each occurs. This information can then be passed to nag_binary_factor (g11sac).

4 References

None.

5 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **p** – Integer *Input*

On entry: p, the number of dichotomous variables.

Constraint: **p** ≥ 3.

3: **n** – Integer *Input*

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

Constraint: **n** ≥ 7.

4: **ns** – Integer * *Output*

On exit: the number of different score patterns, s.

5: **x**[dim] – Nag_Boolean *Input/Output*

Note: the dimension, *dim*, of the array **x** must be at least

max(1, **pdx** × **p**) when **order** = Nag_ColMajor;
max(1, **n** × **pdx**) when **order** = Nag_RowMajor.

Where **X**(i, j) appears in this document, it refers to the array element

x$[(j-1) \times$ **pdx** $+ i - 1]$ when **order** = Nag_ColMajor;
x$[(i-1) \times$ **pdx** $+ j - 1]$ when **order** = Nag_RowMajor.

On entry: **X**(i, j) must be set equal to Nag_TRUE if $x_{ij} = 1$, and Nag_FALSE if $x_{ij} = 0$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$.

On exit: the first s rows of **x** contain the s different score patterns.

6: **pdx** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **x**.

Constraints:

if **order** = Nag_ColMajor, **pdx** $\geq$ **n**;
if **order** = Nag_RowMajor, **pdx** $\geq$ **p**.

7: **irl**[**n**] – Integer *Output*

On exit: the frequency with which the lth row of **x** occurs, for $l = 1, 2, \ldots, s$.

8: **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_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 7.

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

On entry, **pdx** = ⟨*value*⟩.
Constraint: **pdx** > 0.

NE_INT_2

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

On entry, **pdx** = ⟨*value*⟩ and **p** = ⟨*value*⟩.
Constraint: **pdx** ≥ **p**.

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

Exact.

8 Parallelism and Performance

nag_binary_factor_service (g11sbc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_binary_factor_service (g11sbc) is small and increases with n.

10 Example

This example counts the frequencies of different score patterns in the following list:

Score Patterns
000
010
111
000
001
000
000
110
001
011

10.1 Program Text

```
/* nag_binary_factor_service (g11sbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
```

```
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg11.h>

int main(void)
{
  /* Scalars */
  Integer exit_status, i, p, ns, j, n, nrx, pdx;
  /* Arrays */
  char nag_enum_arg[40];
  Integer *irl = 0;
  Nag_Boolean *x = 0;
  Nag_OrderType order;
  NagError fail;

#ifdef NAG_COLUMN_MAJOR
#define X(I, J) x[(J-1)*pdx + I - 1]
  order = Nag_ColMajor;
#else
#define X(I, J) x[(I-1)*pdx + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_binary_factor_service (g11sbc) Example Program Results\n");

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

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &n, &p);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &n, &p);
#endif

  if (n > 0 && p > 0) {
    /* Allocate arrays */
    nrx = n;
    if (!(irl = NAG_ALLOC(n, Integer)) ||
        !(x = NAG_ALLOC(nrx * p, Nag_Boolean)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

    if (order == Nag_ColMajor)
      pdx = nrx;
    else
      pdx = p;

    for (i = 1; i <= n; ++i) {
      for (j = 1; j <= p; ++j) {
#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
         */
        X(i, j) = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
      }
```

```
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }

    /* nag_binary_factor_service (g11sbc).
     * Frequency count for nag_binary_factor (g11sac)
     */
    nag_binary_factor_service(order, p, n, &ns, x, pdx, irl, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_binary_factor_service (g11sbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

    printf("\n");
    printf("Frequency                 Score pattern\n");
    printf("\n");
    for (i = 1; i <= ns; ++i) {
      printf("%5" NAG_IFMT "            ", irl[i - 1]);
      for (j = 1; j <= p; ++j)
        printf("%-9s  ", nag_enum_value_to_name(X(i, j)));

      printf("\n");
    }
  }

END:
  NAG_FREE(irl);
  NAG_FREE(x);

  return exit_status;
}
```

10.2 Program Data

```
nag_binary_factor_service (g11sbc) Example Program Data
10 3
Nag_FALSE  Nag_FALSE  Nag_FALSE
Nag_FALSE  Nag_TRUE   Nag_FALSE
Nag_TRUE   Nag_TRUE   Nag_TRUE
Nag_FALSE  Nag_FALSE  Nag_FALSE
Nag_FALSE  Nag_FALSE  Nag_TRUE
Nag_FALSE  Nag_FALSE  Nag_FALSE
Nag_FALSE  Nag_FALSE  Nag_FALSE
Nag_TRUE   Nag_TRUE   Nag_FALSE
Nag_FALSE  Nag_FALSE  Nag_TRUE
Nag_FALSE  Nag_TRUE   Nag_TRUE
```

10.3 Program Results

```
nag_binary_factor_service (g11sbc) Example Program Results

Frequency                 Score pattern

    4             Nag_FALSE  Nag_FALSE  Nag_FALSE
    1             Nag_FALSE  Nag_TRUE   Nag_FALSE
    1             Nag_TRUE   Nag_TRUE   Nag_TRUE
    2             Nag_FALSE  Nag_FALSE  Nag_TRUE
    1             Nag_TRUE   Nag_TRUE   Nag_FALSE
    1             Nag_FALSE  Nag_TRUE   Nag_TRUE
```