

NAG Library Function Document

nag_mv_z_scores (g03zac)

1 Purpose

nag_mv_z_scores (g03zac) produces standardized values (z-scores) for a data matrix.

2 Specification

```
#include <nag.h>
#include <nagg03.h>

void nag_mv_z_scores (Integer n, Integer m, const double x[], Integer tdx,
    Integer nvar, const Integer isx[], const double s[], const double e[],
    double z[], Integer tdz, NagError *fail)
```

3 Description

For a data matrix, X , consisting of n observations on p variables, with elements x_{ij} , nag_mv_z_scores (g03zac) computes a matrix, Z , with elements z_{ij} such that:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \quad i = 1, 2, \dots, n; j = 1, 2, \dots, p,$$

where μ_j is a location shift and σ_j is a scaling factor. Typically, μ_j will be the mean and σ_j will be the standard deviation of the j th variable and therefore the elements in column j of Z will have zero mean and unit variance.

4 References

None.

5 Arguments

- | | | |
|----|--|--------------|
| 1: | n – Integer | <i>Input</i> |
| | <i>On entry:</i> the number of observations in the data matrix, n . | |
| | <i>Constraint:</i> $n \geq 1$. | |
| 2: | m – Integer | <i>Input</i> |
| | <i>On entry:</i> the number of variables in the data array x . | |
| | <i>Constraint:</i> $m \geq \text{nvar}$. | |
| 3: | x [$n \times \text{tdx}$] – const double | <i>Input</i> |
| | <i>On entry:</i> x [($i - 1$) \times tdx + $j - 1$] must contain the i th sample point for the j th variable x_{ij} , for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$. | |
| 4: | tdx – Integer | <i>Input</i> |
| | <i>On entry:</i> the stride separating matrix column elements in the array x . | |
| | <i>Constraint:</i> $\text{tdx} \geq m$. | |

- 5: **nvar** – Integer *Input*
On entry: the number of variables to be standardized, p .
Constraint: **nvar** ≥ 1 .
- 6: **isx[m]** – const Integer *Input*
On entry: **isx**[$j - 1$] indicates whether or not the observations on the j th variable are included in the matrix of standardized values.
 If **isx**[$j - 1$] $\neq 0$, then the observations from the j th variable are included.
 If **isx**[$j - 1$] = 0, then the observations from the j th variable are not included.
Constraint: **isx**[$j - 1$] $\neq 0$ for **nvar** values of j .
- 7: **s[m]** – const double *Input*
On entry: if **isx**[$j - 1$] $\neq 0$, then **s**[$j - 1$] must contain the scaling (standard deviation), σ_j , for the j th variable.
 If **isx**[$j - 1$] = 0, then **s**[$j - 1$] is not referenced.
Constraint: if **isx**[$j - 1$] $\neq 0$, **s**[$j - 1$] > 0.0 , for $j = 1, 2, \dots, m$.
- 8: **e[m]** – const double *Input*
On entry: if **isx**[$j - 1$] $\neq 0$, then **e**[$j - 1$] must contain the location shift (mean), μ_j , for the j th variable.
 If **isx**[$j - 1$] = 0, then **e**[$j - 1$] is not referenced.
- 9: **z[n \times tdz]** – double *Output*
Note: the (i, j) th element of the matrix Z is stored in **z**[($i - 1$) \times **tdz** + $j - 1$].
On exit: the matrix of standardized values (z -scores), Z .
- 10: **tdz** – Integer *Input*
On entry: the stride separating matrix column elements in the array **z**.
Constraint: **tdz** \geq **nvar**.
- 11: **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_2_INT_ARG_LT

On entry, **m** = $\langle value \rangle$ while **nvar** = $\langle value \rangle$. These arguments must satisfy **m** \geq **nvar**.

On entry, **tdx** = $\langle value \rangle$ while **m** = $\langle value \rangle$. These arguments must satisfy **tdx** \geq **m**.

On entry, **tdz** = $\langle value \rangle$ while **nvar** = $\langle value \rangle$. These arguments must satisfy **tdz** \geq **nvar**.

NE_INT_ARG_LT

On entry, **n** = $\langle value \rangle$.

Constraint: **n** ≥ 1 .

On entry, **nvar** = $\langle value \rangle$.

Constraint: **nvar** ≥ 1 .

NE_INTARR_REALARR

On entry, $\mathbf{isx}[\langle value \rangle] = \langle value \rangle$, $\mathbf{s}[\langle value \rangle] = \langle value \rangle$.

Constraint: if $\mathbf{isx}[j - 1] = 0$, $\mathbf{s}[j - 1] > 0.0$, for $j = 1, 2, \dots, m$.

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_VAR_INCL_INDICATED

The number of variables, **nvar** in the analysis = $\langle value \rangle$, while number of variables included in the analysis via array **isx** = $\langle value \rangle$.

Constraint: these two numbers must be the same.

7 Accuracy

Standard accuracy is achieved.

8 Parallelism and Performance

nag_mv_z_scores (g03zac) is not threaded in any implementation.

9 Further Comments

Means and standard deviations may be obtained using nag_summary_stats_onevar (g01atc) or nag_corr_cov (g02bxc).

10 Example

A 4 by 3 data matrix is input along with location and scaling values. The first and third columns are scaled and the results printed.

10.1 Program Text

```
/* nag_mv_z_scores (g03zac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg03.h>

#define X(I, J) x[(I) *tdx + J]
#define Z(I, J) z[(I) *tdz + J]
int main(void)
{
    Integer exit_status = 0, i, *isx = 0, j, m, n, nvar, tdx, tdz;
    NagError fail;
    double *e = 0, *s = 0, *x = 0, *z = 0;

    INIT_FAIL(fail);

    printf("nag_mv_z_scores (g03zac) Example Program Results\n\n");

    /* Skip headings 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
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " ", &m);
#else
    scanf("%" NAG_IFMT " ", &m);
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " ", &nvar);
#else
    scanf("%" NAG_IFMT " ", &nvar);
#endif

    if (n >= 1 && nvar >= 1 && m >= nvar) {
        if (!(e = NAG_ALLOC(m, double)) ||
            !(s = NAG_ALLOC(m, double)) ||
            !(x = NAG_ALLOC((n) * (m), double)) ||
            !(z = NAG_ALLOC((n) * (nvar), double)) ||
            !(isx = NAG_ALLOC(m, Integer)))
        {
            printf("Allocation failure\\n");
            exit_status = -1;
            goto END;
        }
        tdx = m;
        tdz = nvar;
    }
    else {
        printf("Invalid n or nvar.\\n");
        exit_status = 1;
        return exit_status;
    }
    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j)
#ifdef _WIN32
            scanf_s("%lf", &X(i, j));
#else
            scanf("%lf", &X(i, j));
#endif
    }
    for (j = 0; j < m; ++j)
#ifdef _WIN32
        scanf_s("%" NAG_IFMT " ", &isx[j]);
#else
        scanf("%" NAG_IFMT " ", &isx[j]);
#endif

    for (j = 0; j < m; ++j)
#ifdef _WIN32
        scanf_s("%lf", &e[j]);
#else
        scanf("%lf", &e[j]);
#endif

    for (j = 0; j < m; ++j)
#ifdef _WIN32
        scanf_s("%lf", &s[j]);
#else
        scanf("%lf", &s[j]);
#endif

    /* nag_mv_z_scores (g03zac).
     * Standardize values of a data matrix

```

```

    */
    nag_mv_z_scores(n, m, x, tdx, nvar, isx, s, e, z, tdz, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_mv_z_scores (g03zac).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    printf("\nStandardized Values\n\n");
    for (i = 0; i < n; ++i) {
        for (j = 0; j < nvar; ++j)
            printf("%8.3f", z(i, j));
        printf("\n");
    }
END:
    NAG_FREE(e);
    NAG_FREE(s);
    NAG_FREE(x);
    NAG_FREE(z);
    NAG_FREE(isx);
    return exit_status;
}

```

10.2 Program Data

```

nag_mv_z_scores (g03zac) Example Program Data
4 3 2
15.0 0.0 1500.0
12.0 1.0 1000.0
18.0 2.0 1200.0
14.0 3.0 500.0
1      0      1
14.75 0.0 1050.0
2.50 0.0 420.3

```

10.3 Program Results

```

nag_mv_z_scores (g03zac) Example Program Results

```

Standardized Values

```

0.100 1.071
-1.100 -0.119
1.300 0.357
-0.300 -1.309

```
