NAG Library Function Document

nag_tsa_mean_range (g13auc)

1 Purpose

nag_tsa_mean_range (g13auc) calculates the range (or standard deviation) and the mean for groups of successive time series values. It is intended for use in the construction of range-mean plots.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
```

```
void nag_tsa_mean_range (Integer n, const double z[], Integer m,
     Nag_RangeStat rs, double y[], double mean[], NagError *fail)
```

3 Description

Let $Z_1, Z_2, \ldots, Z_n$ denote n successive observations in a time series. The series may be divided into groups of m successive values and for each group the range or standard deviation (depending on a user-supplied option) and the mean are calculated. If n is not a multiple of m then groups of equal size m are found starting from the end of the series of observations provided, and any remaining observations at the start of the series are ignored. The number of groups used, k, is the integer part of n/m. If you wish to ensure that no observations are ignored then the number of observations, n, should be chosen so that n is divisible by m.

The mean, M_i, the range, R_i, and the standard deviation, S_i, for the ith group are defined as

$$M_i = \frac{1}{m}\sum_{j=1}^{m} Z_{l+m(i-1)+j}$$

$$R_i = \max{}_{1 \leq j \leq m}\left\{Z_{l+m(i-1)+j}\right\} - \min{}_{1 \leq j \leq m}\left\{Z_{l+m(i-1)+j}\right\}$$

and

$$S_i = \sqrt{\left(\frac{1}{m-1}\right)\sum_{j=1}^{m}\left(Z_{l+m(i-1)+j} - M_i\right)^2}$$

where $l = n - km$, the number of observations ignored.

For seasonal data it is recommended that m should be equal to the seasonal period. For non-seasonal data the recommended group size is 8.

A plot of range against mean or of standard deviation against mean is useful for finding a transformation of the series which makes the variance constant. If the plot appears random or the range (or standard deviation) seems to be constant irrespective of the mean level then this suggests that no transformation of the time series is called for. On the other hand an approximate linear relationship between range (or standard deviation) and mean would indicate that a log transformation is appropriate. Further details may be found in either Jenkins (1979) or McLeod (1982).

You have the choice of whether to use the range or the standard deviation as a measure of variability. If the group size is small they are both equally good but if the group size is fairly large (e.g., $m = 12$ for monthly data) then the range may not be as good an estimate of variability as the standard deviation.

4 References

Jenkins G M (1979) *Practical Experiences with Modelling and Forecasting Time Series* GJP Publications, Lancaster

McLeod G (1982) *Box–Jenkins in Practice. 1: Univariate Stochastic and Single Output Transfer Function/Noise Analysis* GJP Publications, Lancaster

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of observations in the time series.

Constraint: **n** $\geq$ **m**.

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

On entry: **z**$[t-1]$ must contain the tth observation Z_t, for $t = 1, 2, \dots, n$.

3: **m** – Integer *Input*

On entry: m, the group size.

Constraint: **m** ≥ 2.

4: **rs** – Nag_RangeStat *Input*

On entry: indicates whether ranges or standard deviations are to be calculated.

rs = Nag_UseRange
Ranges are calculated.

rs = Nag_UseSD
Standard deviations are calculated.

Constraint: **rs** = Nag_UseRange or Nag_UseSD.

5: **y**[**int**(**n**/**m**)] – double *Output*

On exit: **y**$[i-1]$ contains the range or standard deviation, as determined by **rs**, of the ith group of observations, for $i = 1, 2, \dots, k$.

6: **mean**[**int**(**n**/**m**)] – double *Output*

On exit: **mean**$[i-1]$ contains the mean of the ith group of observations, for $i = 1, 2, \dots, k$.

7: **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, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} \geq 2$.

NE_INT_2

On entry, $\mathbf{n} = \langle value \rangle$ and $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq \mathbf{m}$.

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.

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

The computations are believed to be stable.

8 Parallelism and Performance

nag_tsa_mean_range (g13auc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_tsa_mean_range (g13auc) is approximately proportional to n.

10 Example

The following program produces the statistics for a range-mean plot for a series of 100 observations divided into groups of 8.

10.1 Program Text

```
/* nag_tsa_mean_range (g13auc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg13.h>

int main(void)
{
  /* Scalars */
  Integer exit_status, i, ngrps, m, n;

  /* Arrays */
  double *mean = 0, *range = 0, *z = 0;
  NagError fail;
```

```
  INIT_FAIL(fail);

  exit_status = 0;

  printf("nag_tsa_mean_range (g13auc) 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, &m);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &n, &m);
#endif
  if (n >= m && m >= 1) {
    ngrps = n / m;

    /* Allocate arrays */
    if (!(mean = NAG_ALLOC(ngrps, double)) ||
        !(range = NAG_ALLOC(ngrps, double)) || !(z = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

    for (i = 1; i <= n; ++i)
#ifdef _WIN32
      scanf_s("%lf", &z[i - 1]);
#else
      scanf("%lf", &z[i - 1]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif

    printf("\n");

    /* nag_tsa_mean_range (g13auc).
     * Computes quantities needed for range-mean or standard
     * deviation-mean plot
     */
    nag_tsa_mean_range(n, z, m, Nag_UseRange, range, mean, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_tsa_mean_range (g13auc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

    printf("   Range      Mean\n");
    for (i = 1; i <= ngrps; i++)
      printf("%8.3f  %8.3f\n", range[i - 1], mean[i - 1]);
  }

END:
  NAG_FREE(mean);
  NAG_FREE(range);
  NAG_FREE(z);

  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_mean_range (g13auc) Example Program Data
100 8  : n, no. of obs in time series, m, no. of obs in each group
 101  82  66  35  31   6  20  90 154 125
  85  68  38  23  10  24  83 133 131 118
  90  67  60  47  41  21  16   6   4   7
  14  34  45  43  49  42  28  10   5   2
   0   1   3  12  14  35  47  41  30  24
  16   7   4   2   8  13  36  50  62  67
  72  48  29   8  13  57 122 139 103  86
  63  37  26  11  15  40  62  98 124  96
  65  64  54  39  21   7   4  23  53  94
  96  77  59  44  47  30  16   7  37  74 : End of time series
```

10.3 Program Results

```
nag_tsa_mean_range (g13auc) Example Program Results

   Range      Mean
 148.000    72.375
 123.000    70.000
  84.000    43.500
  45.000    29.750
  28.000     7.625
  40.000    26.750
  65.000    30.250
 131.000    61.000
  92.000    47.625
  85.000    75.250
  92.000    46.875
  67.000    39.250
```