NAG Library Function Document

nag_5pt_summary_stats (g01alc)

1 Purpose

nag_5pt_summary_stats (g01alc) calculates a five-point summary for a single sample.

2 Specification

```c
#include <nag.h>
#include <nagg01.h>
void nag_5pt_summary_stats (Integer n, const double x[], double res[], NagError *fail)
```

3 Description

nag_5pt_summary_stats (g01alc) calculates the minimum, lower hinge, median, upper hinge and the maximum of a sample of \( n \) observations.

The data consist of a single sample of \( n \) observations denoted by \( x_i \), for \( i = 1, 2, \ldots, n \), represent the sample observations sorted into ascending order.

Let \( m = \frac{n}{2} \) if \( n \) is even and \( \frac{(n+1)}{2} \) if \( n \) is odd,

and \( k = \frac{m}{2} \) if \( m \) is even and \( \frac{(m+1)}{2} \) if \( m \) is odd.

Then we have

- Minimum = \( z_1 \),
- Maximum = \( z_n \),
- Median = \( z_m \) if \( n \) is odd,
  = \( \frac{z_m + z_{m+1}}{2} \) if \( n \) is even,
- Lower hinge = \( z_k \) if \( m \) is odd,
  = \( \frac{z_k + z_{k+1}}{2} \) if \( m \) is even,
- Upper hinge = \( z_{n-k+1} \) if \( m \) is odd,
  = \( \frac{z_{n-k} + z_{n-k+1}}{2} \) if \( m \) is even.

4 References

Tukey J W (1977) *Exploratory Data Analysis* Addison-Wesley

5 Arguments

1: \( n \) – Integer

   *Input*

   *On entry:* \( n \), number of observations in the sample.

   *Constraint:* \( n \geq 5 \).
2:  \textbf{x[n]} \text{ – const double}\hfill \textit{Input}\hfill
\textit{On entry:} the sample observations, \( x_1, x_2, \ldots, x_n \).

3:  \textbf{res[5]} \text{ – double}\hfill \textit{Output}\hfill
\textit{On exit:} \textbf{res} contains the five-point summary.
\begin{itemize}
  \item \textbf{res[0]}\hfill The minimum.
  \item \textbf{res[1]}\hfill The lower hinge.
  \item \textbf{res[2]}\hfill The median.
  \item \textbf{res[3]}\hfill The upper hinge.
  \item \textbf{res[4]}\hfill The maximum.
\end{itemize}

4:  \textbf{fail} \text{ – NagError *}\hfill \textit{Input/Output}\hfill
\textit{The NAG error argument (see Section 3.6 in the Essential Introduction).}

6  \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}
On entry, argument \textit{value} had an illegal value.

\textbf{NE_INT_ARG_LT}
On entry, \textbf{n} = \textit{value}.
Constraint: \textbf{n} \geq 5.

\textbf{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 3.6.6 in the Essential Introduction for further information.

\textbf{NE_NO_LICENCE}
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

7  \textbf{Accuracy}
The computations are stable.

8  \textbf{Parallelism and Performance}
Not applicable.
9 Further Comments

The time taken by nag_5pt_summary_stats (g01alc) is proportional to \( n \).

10 Example

This example calculates a five-point summary for a sample of 12 observations.

10.1 Program Text

/* nag_5pt_summary_stats (g01alc) Example Program.
* Copyright 2014 Numerical Algorithms Group.
* Mark 4, 1996.
*/
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
  Integer exit_status = 0, i, n;
  NagError fail;
  double *res = 0, *x = 0;
  INIT_FAIL(fail);
  printf("nag_5pt_summary_stats (g01alc) Example Program Results\n");
  /* Skip heading 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
  if (n >= 5)
  {
    if (!(x = NAG_ALLOC(n, double)) ||
       !(res = NAG_ALLOC(5, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  }
  else
  {
    printf("Invalid n.\n");
    exit_status = 1;
    return exit_status;
  }
  for (i = 1; i <= n; ++i)
  {#ifdef _WIN32
    scanf_s("%lf ", &x[i - 1]);
        #else
    scanf("%lf ", &x[i - 1]);
  #endif
  /* nag_5pt_summary_stats (g01alc).
   * Five-point summary (median, hinges and extremes)
/*
nag_5pt_summary_stats(n, x, res, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_5pt_summary_stats (g01alc).\n\ns\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\n");
printf(" Maximum %16.4f\n", res[4]);
printf(" Upper Hinge %16.4f\n", res[3]);
printf(" Median %16.4f\n", res[2]);
printf(" Lower Hinge %16.4f\n", res[1]);
printf(" Minimum %16.4f\n", res[0]);
END:
NAG_FREE(x);
NAG_FREE(res);
    return exit_status;
}

10.2 Program Data

nag_5pt_summary_stats (g01alc) Example Program Data
12
12.0 9.0 2.0 5.0 6.0 8.0 2.0 7.0 3.0 1.0 11.0 10.0

10.3 Program Results

nag_5pt_summary_stats (g01alc) Example Program Results

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>12.0000</td>
</tr>
<tr>
<td>Upper Hinge</td>
<td>9.5000</td>
</tr>
<tr>
<td>Median</td>
<td>6.5000</td>
</tr>
<tr>
<td>Lower Hinge</td>
<td>2.5000</td>
</tr>
<tr>
<td>Minimum</td>
<td>1.0000</td>
</tr>
</tbody>
</table>