NAG Library Function Document

nag_anova_confid_interval (g04dbc)

1 Purpose

nag_anova_confid_interval (g04dbc) computes simultaneous confidence intervals for the differences between means. It is intended for use after nag_anova_random (g04bbc) or nag_anova_row_col (g04bcc).

2 Specification

```
#include <nag.h>
#include <nagg04.h>
```

```
void nag_anova_confid_interval (Nag_IntervalType type, Integer nt,
     const double tmean[], double rdf, const double c[], Integer tdc,
     double clevel, double cil[], double ciu[], Integer isig[],
     NagError *fail)
```

3 Description

In the computation of analysis of a designed experiment the first stage is to compute the basic analysis of variance table, the estimate of the error variance (the residual or error mean square), $\hat{\sigma}^2$, the residual degrees of freedom, ν, and the (variance ratio) F-statistic for the t treatments. The second stage of the analysis is to compare the treatment means. If the treatments have no structure, for example the treatments are different varieties, rather than being structured, for example a set of different temperatures, then a multiple comparison procedure can be used.

A multiple comparison procedure looks at all possible pairs of means and either computes confidence intervals for the difference in means or performs a suitable test on the difference. If there are t treatments then there are $t(t-1)/2$ comparisons to be considered. In tests the type 1 error or significance level is the probability that the result is considered to be significant when there is no difference in the means. If the usual t-test is used with, say, a five percent significance level then the type 1 error for all $k = t(t-1)/2$ tests will be much higher. If the tests were independent then if each test is carried out at the 100α percent level then the overall type 1 error would be $\alpha^* = 1 - (1-\alpha)^k \simeq k\alpha$. In order to provide an overall protection the individual tests, or confidence intervals, would have to be carried out at a value of α such that α^* is the required significance level, e.g., five percent.

The $100(1-\alpha)$ percent confidence interval for the difference in two treatment means, $\hat{\tau}_i$ and $\hat{\tau}_j$ is given by

$$\left(\hat{\tau}_i - \hat{\tau}_j\right) \pm T^*_{(\alpha,\nu,t)} se\left(\hat{\tau}_i - \hat{\tau}_j\right),$$

where $se()$ denotes the standard error of the difference in means and $T^*_{(\alpha,\nu,t)}$ is an appropriate percentage point from a distribution. There are several possible choices for $T^*_{(\alpha,\nu,t)}$. These are:

(a) $\frac{1}{2}q_{(1-\alpha,\nu,t)}$, the studentized range statistic. It is the appropriate statistic to compare the largest mean with the smallest mean. This is known as Tukey–Kramer method.

(b) $t_{(\alpha/k,\nu)}$, this is the Bonferroni method.

(c) $t_{(\alpha_0,\nu)}$, where $\alpha_0 = 1 - (1-\alpha)^{1/k}$, this is known as the Dunn–Sidak method.

(d) $t_{(\alpha,\nu)}$, this is known as Fisher's LSD (least significant difference) method. It should only be used if the overall F-test is significant, the number of treatment comparisons is small and were planned before the analysis.

(e) $\sqrt{(k-1)F_{1-\alpha,k-1,\nu}}$ where $F_{1-\alpha,k-1,\nu}$ is the deviate corresponding to a lower tail probability of $1-\alpha$ from an F-distribution with $k-1$ and ν degrees of freedom. This is Scheffe's method.

In cases (b), (c) and (d), $t_{(\alpha,\nu)}$ denotes the α two-tail significance level for the Student's t-distribution with ν degrees of freedom, see nag_deviates_students_t (g01fbc).

The Scheffe method is the most conservative, followed closely by the Dunn–Sidak and Tukey–Kramer methods.

To compute a test for the difference between two means the statistic,

$$\frac{\hat{\tau}_i - \hat{\tau}_j}{se(\hat{\tau}_i - \hat{\tau}_j)}$$

is compared with the appropriate value of $T^*_{(\alpha,\nu,t)}$.

4 References

Kotz S and Johnson N L (ed.) (1985a) Multiple range and associated test procedures *Encyclopedia of Statistical Sciences* **5** Wiley, New York

Kotz S and Johnson N L (ed.) (1985b) Multiple comparison *Encyclopedia of Statistical Sciences* **5** Wiley, New York

Winer B J (1970) *Statistical Principles in Experimental Design* McGraw–Hill

5 Arguments

1: **type** – Nag_IntervalType *Input*

On entry: indicates which method is to be used.

type = Nag_TukeyInterval
The Tukey–Kramer method is used.

type = Nag_BonferroniInterval
The Bonferroni method is used.

type = Nag_DunnInterval
The Dunn–Sidak method is used.

type = Nag_FisherInterval
The Fisher LSD method is used.

type = Nag_ScheffeInterval
The Scheffe's method is used.

Constraint: **type** = Nag_TukeyInterval, Nag_BonferroniInterval, Nag_DunnInterval, Nag_FisherInterval or Nag_ScheffeInterval.

2: **nt** – Integer *Input*

On entry: the number of treatment means, t.

Constraint: **nt** ≥ 2.

3: **tmean**[**nt**] – const double *Input*

On entry: **tmean**$[i-1]$ contains the treatment means, $\hat{\tau}_i$, $i = 1,2,\ldots,t$.

4: **rdf** – double *Input*

On entry: the residual degrees of freedom, ν.

Constraint: **rdf** ≥ 1.0.

5: **c**[**nt** $\times$ **tdc**] – const double *Input*

On entry: the strictly lower triangular part of **c** must contain the standard errors of the differences between the means as returned by nag_anova_random (g04bbc) and nag_anova_row_col (g04bcc). That is **c**[$(i-1) \times$ **tdc** $+ j - 1$], $i > j$, contains the standard error of the difference between the ith and jth mean in **tmean**.

Constraint: **c**[$(i-1) \times$ **tdc** $+ j - 1$] > 0.0, for $i = 2, 3, \ldots, t$ and $j = 1, 2, \ldots, i - 1$.

6: **tdc** – Integer *Input*

On entry: the stride separating matrix column elements in the array **c**.

Constraint: **tdc** $\geq$ **nt**.

7: **clevel** – double *Input*

On entry: the required confidence level for the computed intervals, $(1 - \alpha)$.

Constraint: $0.0 <$ **clevel** < 1.0.

8: **cil**[**nt** $\times$ (**nt** $-$ **1**)/**2**] – double *Output*

On exit: **cil**[$(i-1)(i-2)/2 + j - 1$] contains the lower limit to the confidence interval for the difference between ith and jth means in **tmean**, for $i = 2, 3, \ldots, t$ and $j = 1, 2, \ldots, i - 1$.

9: **ciu**[**nt** $\times$ (**nt** $-$ **1**)/**2**] – double *Output*

On exit: **ciu**[$(i-1)(i-2)/2 + j - 1$] contains the upper limit to the confidence interval for the difference between ith and jth means in **tmean**, for $i = 2, 3, \ldots, t$ and $j = 1, 2, \ldots, i - 1$.

10: **isig**[**nt** $\times$ (**nt** $-$ **1**)/**2**] – Integer *Output*

On exit: **isig**[$(i-1)(i-2)/2 + j - 1$] indicates if the difference between ith and jth means in **tmean** is significant, for $i = 2, 3, \ldots, t$ and $j = 1, 2, \ldots, i - 1$. If the difference is significant then the returned value is 1; otherwise the returned value is 0.

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

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

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

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

NE_2D_REAL_ARRAY_CONS

On entry, **c**[$(\langle value \rangle) \times$ **tdc** $+ \langle value \rangle$] $= \langle value \rangle$.
Constraint: **c**[$(i) \times$ **tdc** $+ j$] > 0.0, for $i = 1, 2, \ldots,$ **nt** $- 1$ and $j = 0, 1, \ldots, i - 1$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument **type** had an illegal value.

NE_INT_ARG_LT

On entry, **nt** $= \langle value \rangle$.
Constraint: **nt** ≥ 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_REAL

On entry, **clevel** = ⟨*value*⟩.
Constraint: $0.0 <$ **clevel** < 1.0.

NE_REAL_ARG_LT

On entry, **rdf** must not be less than 1.0: **rdf** = ⟨*value*⟩.

NE_STUDENTIZED_STAT

There has been a failure in the computation of the studentized range statistic. Try using a smaller value of **clevel**.

7 Accuracy

For the accuracy of the percentage point statistics see nag_deviates_students_t (g01fbc).

8 Parallelism and Performance

Not applicable.

9 Further Comments

An alternative approach to one used in nag_anova_confid_interval (g04dbc) is the sequential testing of the Student–Newman–Keuls procedure. This, in effect, uses the Tukey–Kramer method but first ordering the treatment means and examining only subsets of the treatment means in which the largest and smallest are significantly different. At each stage the third argument of the Studentized range statistic is the number of means in the subset rather than the total number of means.

10 Example

In the example taken from Winer (1970) a completely randomized design with unequal treatment replication is analysed using nag_anova_random (g04bbc) and then confidence intervals are computed by nag_anova_confid_interval (g04dbc) using the Tukey–Kramer method.

10.1 Program Text

```
/* nag_anova_confid_interval (g04dbc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg04.h>

int main(void)
{

  Integer          exit_status = 0, nt, i, ij, irdf, j, n, nblock;
  Integer          *irep = 0, *isig = 0, *it = 0;
  const char       *fmt_99998[] = { "", "   %3.0f  ", "%10.1f  ", "%10.1f  ",
                                  "%10.3f  ", "%9.4f" };
  char             star[1*2+1];
  char             nag_enum_arg[40];
  double           clevel, gmean, rdf, tol;
  double           *bmean = 0, *c = 0, *cil = 0, *ciu = 0, *ef = 0, *r = 0;
  double           *table = 0, *tmean = 0, *y = 0;
  Nag_IntervalType type;
  NagError         fail;

  #define TABLE(I, J) table[((I) -1)*5 + (J) -1]

  INIT_FAIL(fail);

  printf(
          "nag_anova_confid_interval (g04dbc) 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, &nt);
#else
  scanf("%"NAG_IFMT" %"NAG_IFMT" ", &n, &nt);
#endif
  if (!(y = NAG_ALLOC(n, double))
      || !(it = NAG_ALLOC(n, Integer))
      || !(tmean = NAG_ALLOC(nt, double))
      || !(table = NAG_ALLOC(4*5, double))
      || !(c = NAG_ALLOC(nt*nt, double))
      || !(irep = NAG_ALLOC(nt, Integer))
      || !(r = NAG_ALLOC(n, double))
      || !(ef = NAG_ALLOC(nt, double))
      || !(isig = NAG_ALLOC(nt*(nt-1)/2, Integer))
      || !(cil = NAG_ALLOC(nt*(nt-1)/2, double))
      || !(ciu = NAG_ALLOC(nt*(nt-1)/2, double)))
    {
      printf("Allocation failure\n");
      exit_status = 1;
      goto END;
    }


  for (i = 1; i <= n; ++i)
#ifdef _WIN32
    scanf_s("%lf ", &y[i - 1]);
#else
    scanf("%lf ", &y[i - 1]);
#endif
  for (i = 1; i <= n; ++i)
#ifdef _WIN32
    scanf_s("%"NAG_IFMT" ", &it[i - 1]);
#else
    scanf("%"NAG_IFMT" ", &it[i - 1]);
#endif
  tol = 5e-6;
  irdf = 0;
  nblock = 1;
  if (!(bmean = NAG_ALLOC(nblock, double)))
    {
      exit_status = -1;
      printf("Allocation failure\n");
      goto END;
    }
  /* nag_anova_random (g04bbc).
   * General block design or completely randomized design
   */
  nag_anova_random(n, y, Nag_NoBlocks, nblock, nt, it, &gmean, bmean, tmean,
                   table, c, nt, irep, r, ef, tol, irdf, &fail);
  if (fail.code != NE_NOERROR)
```

```
    {
      printf("Error from nag_anova_random (g04bbc).\n%s\n",
              fail.message);
      exit_status = -1;
      goto END;
    }

  printf("\n%s\n\n", "ANOVA table");
  printf("%s\n\n",
         "  Source        df          SS           MS           F        Prob");
  printf(" Treatments");
  for (j = 1; j <= 5; ++j)
    printf(fmt_99998[j], TABLE(2, j));
  printf("\n");
  printf(" Residual  ");
  for (j = 1; j <= 3; ++j)
    printf(fmt_99998[j], TABLE(3, j));
  printf("\n");
  printf(" Total     ");
  for (j = 1; j <= 2; ++j)
    printf(fmt_99998[j], TABLE(4, j));
  printf("\n");
  printf("\n Treatment means\n");
  printf("\n");
  for (j = 1; j <= nt; ++j)
    printf("%8.3f%s", tmean[j - 1], j%8?"":"\n");
  printf("\n");
  printf("\n Simultaneous Confidence Intervals\n\n");
  rdf = TABLE(3, 1);
#ifdef _WIN32
  scanf_s(" %39s %lf", nag_enum_arg, _countof(nag_enum_arg), &clevel);
#else
  scanf(" %39s %lf", nag_enum_arg, &clevel);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  type = (Nag_IntervalType) nag_enum_name_to_value(nag_enum_arg);

  /* nag_anova_confid_interval (g04dbc).
   * Computes confidence intervals for differences between
   * means computed by nag_anova_random (g04bbc) or
   * nag_anova_row_col (g04bcc)
   */
  nag_anova_confid_interval(type, nt, tmean, rdf, c, nt, clevel, cil, ciu,
                            isig, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_anova_confid_interval (g04dbc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  star[1] = '*';
  star[0] = ' ';
  star[2] = '\setminus 0';
  ij = 0;
  for (i = 1; i <= nt; ++i)
    {
      for (j = 1; j <= i - 1; ++j)
        {
          ++ij;
          printf("  %2"NAG_IFMT"%2"NAG_IFMT"   %10.3f  %10.3f  %c\n",
                  i, j, cil[ij - 1], ciu[ij - 1], star[isig[ij - 1]]);
        }
    }

 END:
  NAG_FREE(y);
  NAG_FREE(it);
  NAG_FREE(tmean);
  NAG_FREE(table);
  NAG_FREE(c);
  NAG_FREE(irep);
  NAG_FREE(r);
  NAG_FREE(ef);
  NAG_FREE(isig);
  NAG_FREE(cil);
  NAG_FREE(ciu);
  NAG_FREE(bmean);

  return exit_status;
}
```

10.2 Program Data

```
nag_anova_confid_interval (g04dbc) Example Program Data

26 4

 3  2  4  3  1  5
 7  8  4 10  6
 3  2  1  2  4  2  3  1
10 12  8  5 12 10  9

1 1 1 1 1 1
2 2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4

Nag_TukeyInterval .95
```

10.3 Program Results

```
nag_anova_confid_interval (g04dbc) Example Program Results

ANOVA table

  Source        df         SS          MS          F        Prob

 Treatments     3       239.9        80.0      24.029     0.0000
 Residual      22        73.2         3.3
 Total         25       313.1

 Treatment means

   3.000   7.000   2.250   9.429

 Simultaneous Confidence Intervals

   2 1        0.933       7.067  *
   3 1       -3.486       1.986
   3 2       -7.638      -1.862  *
   4 1        3.610       9.247  *
   4 2       -0.538       5.395
   4 3        4.557       9.800  *
```