NAG C Library Function Document

nag_cp_stat (g02ecc)

1 Purpose
nag_cp_stat (g02ecc) calculates $R^2$ and $C_p$-values from the residual sums of squares for a series of linear regression models.

2 Specification

```c
void nag_cp_stat (Nag_IncludeMean mean, Integer n, double sigsq, double tss,
                  Integer nmod, const Integer nterms[], const double rss[], double rsq[],
                  double cp[], NagError *fail)
```

3 Description

When selecting a linear regression model for a set of $n$ observations a balance has to be found between the number of independent variables in the model and fit as measured by the residual sum of squares. The more variables included the smaller will be the residual sum of squares. Two statistics can help in selecting the best model.

(a) $R^2$ represents the proportion of variation in the dependent variable that is explained by the independent variables.

$$R^2 = \frac{\text{Regression Sum of Squares}}{\text{Total Sum of Squares}},$$

where Total sum of squares = \( tss = \sum (y - \bar{y})^2 \) (if mean is fitted, otherwise \( tss = \sum y^2 \)) and Regression sum of squares = \( \text{RegSS} = tss - \text{rss} \), where \( \text{rss} = \text{residual sum of squares} = \sum (y - \hat{y})^2 \).

The $R^2$-values can be examined to find a model with a high $R^2$-value but with small number of independent variables.

(b) $C_p$ statistic.

$$C_p = \frac{\text{rss}^2}{\hat{\sigma}^2} - (n - 2p),$$

where $p$ is the number of parameters (including the mean) in the model and $\hat{\sigma}^2$ is an estimate of the true variance of the errors. This can often be obtained from fitting the full model.

A well fitting model will have $C_p \approx p$. $C_p$ is often plotted against $p$ to see which models are closest to the $C_p = p$ line.

nag_cp_stat (g02ecc) may be called after nag_all_regsn (g02eac) which calculates the residual sums of squares for all possible linear regression models.

4 References


5 Parameters

1:  mean – Nag_IncludeMean

   Input

   On entry: indicates if a mean term is to be included.
If \( \text{mean} = \text{Nag\_MeanInclude} \) (Mean), a mean term, intercept, has been included in the model.

If \( \text{mean} = \text{Nag\_MeanZero} \) (Zero), the model passes through the origin, zero-point.

\[ \text{Constraint: mean} = \text{Nag\_MeanInclude} \text{ or } \text{Nag\_MeanZero}. \]

2: \[ n \] – Integer  
\[ \text{Input} \]

On entry: the number of observations used in the regression model, \( n \).

\[ \text{Constraint: } n \text{ must be greater than } 2 \times p_{\text{max}}, \text{ where } p_{\text{max}} \text{ is the largest number of independent variables fitted (including the mean if fitted)}. \]

3: \[ \text{sigsq} \] – double  
\[ \text{Input} \]

On entry: the best estimate of true variance of the errors, \( \sigma^2 \).

\[ \text{Constraint: sigsq} > 0.0. \]

4: \[ \text{tss} \] – double  
\[ \text{Input} \]

On entry: the total sum of squares for the regression model.

\[ \text{Constraint: tss} > 0.0. \]

5: \[ \text{nmod} \] – Integer  
\[ \text{Input} \]

On entry: the number of regression models.

\[ \text{Constraint: nmod} > 0. \]

6: \[ \text{nterms}[\text{nmod}] \] – const Integer  
\[ \text{Input} \]

On entry: \( \text{nterms}[i - 1] \) must contain the number of independent variables (not counting the mean) fitted to the \( i \)th model, for \( i = 1, 2, \ldots, \text{nmod} \).

7: \[ \text{rss}[\text{nmod}] \] – const double  
\[ \text{Input} \]

On entry: \( \text{rss}[i - 1] \) must contain the residual sum of squares for the \( i \)th model.

\[ \text{Constraint: rss}[i - 1] \leq \text{tss} \text{ for } i = 1, 2, \ldots, \text{nmod}. \]

8: \[ \text{rsq}[\text{nmod}] \] – double  
\[ \text{Output} \]

On exit: \( \text{rsq}[i - 1] \) contains the \( R^2 \)-value for the \( i \)th model, for \( i = 1, 2, \ldots, \text{nmod} \).

9: \[ \text{cp}[\text{nmod}] \] – double  
\[ \text{Output} \]

On exit: \( \text{cp}[i - 1] \) contains the \( C_p \)-value for the \( i \)th model, for \( i = 1, 2, \ldots, \text{nmod} \).

10: \[ \text{fail} \] – NagError *  
\[ \text{Input/Output} \]

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE\_INT

On entry, \( \text{nmod} = \langle \text{value} \rangle. \)

Constraint: \( \text{nmod} > 0. \)

NE\_MODEL\_PARAMETERS

On entry, number of parameters for model \( \langle \text{value} \rangle \) is too large for \( n \). \( n = \langle \text{value} \rangle \) number of parameters = \( \langle \text{value} \rangle. \)
NE_REAL
  On entry, \( tss = \langle value \rangle \).
  Constraint: \( tss > 0.0 \).
  On entry, \( sigsq = \langle value \rangle \).
  Constraint: \( sigsq > 0.0 \).

NE_REAL_ARRAY_ELEM_CONS
  On entry, value of \( rss > tss: rss[\langle value \rangle] = \langle value \rangle, tss = \langle value \rangle \).
  Value of \( cp < 0.0: cp[\langle value \rangle] = \langle value \rangle \).

NE_BAD_PARAM
  On entry, parameter \( \langle value \rangle \) had an illegal value.

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 consult NAG for assistance.

7 Accuracy
Accuracy is sufficient for all practical purposes.

8 Further Comments
None.

9 Example
The data, from an oxygen uptake experiment, is given by Weisberg (1985). The independent and
dependent variables are read and the residual sums of squares for all possible models computed using
nag_all_regsn (g02eac). The values of \( R^2 \) and \( C_p \) are then computed and printed along with the names of
variables in the models.

9.1 Program Text
/* nag_cp_stat (g02ecc) Example Program. */
/* Copyright 2002 Numerical Algorithms Group. */
/* Mark 7, 2002. */
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>
int main(void)
{
  /* Scalars */
  double sigsq, tss;
  Integer exit_status, num_models, i, ii, j, m, max_mod, n, nmod, pdx;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  const char **model=0;
  ...
double *cp=0, *rsq=0, *rss=0, *wtptr=0, *x=0, *y=0;
Integer *sx=0, *mrank=0, *nterms=0;
const char *var_names[] = { "DAY", "BOD", "TKN", "TS", "TVS", "COD" };

/* For iteration over model */
Integer model_index=0;

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

INIT_FAIL(fail);
exit_status = 0;
Vprintf("g02ecc Example Program Results\n");

/* Skip heading in data file */
Vscanf("%*[\n] ");
Vscanf("%ld%ld%*[\n] ", &n, &m);
max_mod = 2 <<m;

/* Allocate memory */
if ( !(x = NAG_ALLOC(n * m, double)) ||
    !(y = NAG_ALLOC(n, double)) ||
    !(sx = NAG_ALLOC(m, Integer)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

#ifdef NAG_COLUMN_MAJOR
pdx = n;
#else
pdx = m;
#endif
order = Nag_RowMajor;

for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= m; ++j)
        Vscanf("%lf", &x(i,j));
    Vscanf("%lf%*[\n] ", &x[i - 1]);
}

num_models = 1;
for (j = 1; j <= m; ++j)
{
    Vscanf("%ld", &sx[j - 1]);
    if (sx[j - 1] == 1)
        num_models <<= 1;
}
Vscanf("%*[\n] ");

/* Allocate memory */
if ( !(model = NAG_ALLOC(num_models*m, const char *)) ||
    !(cp = NAG_ALLOC(num_models, double)) ||
    !(rsq = NAG_ALLOC(num_models, double)) ||
    !(rss = NAG_ALLOC(num_models, double)) ||
    !(mrank = NAG_ALLOC(num_models, Integer)) ||
    !(nterms = NAG_ALLOC(num_models, Integer)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Calculate residual sums of squares using g02eac */

g02eac(order, Nag_MeanInclude, n, m, x, pdx, var_names, sx, y, wtptr,
&nmod, model, rss, nterms, mrank, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g02eac.\n\n", fail.message);
    exit_status = 1;
    goto END;
}

tss = rss[0];
sigsq = rss[nmod - 1] / (n - nterms[nmod - 1] - 1);
g02ecc(Nag_MeanInclude, n, sigsq, tss, nmod, nterms, rss, rsq, cp, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g02ecc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}

Vprintf("\n");
Vprintf("Number of CP RSQ MODEL\n");
Vprintf("parameters\n");
for (i = 1; i <= nmod; ++i)
{
    ii = nterms[i - 1];
    Vprintf(" %7ld%11.2f%8.4f ", ii, cp[i - 1], rsq[i - 1]);
    for (j = 1; j <= ii; ++j)
    {
        Vprintf("%-3.3s %s", model[model_index++],
        j%5 == 0 || j == 5 ?"\n": "");
    }
    Vprintf("\n");
}

END:
if (model) NAG_FREE(model);
if (cp) NAG_FREE(cp);
if (rsq) NAG_FREE(rsq);
if (rss) NAG_FREE(rss);
if (x) NAG_FREE(x);
if (y) NAG_FREE(y);
if (sx) NAG_FREE(sx);
if (mrank) NAG_FREE(mrank);
if (nterms) NAG_FREE(nterms);

return exit_status;

9.2 Program Data

g02ecc Example Program Data

20 6
  0. 1125.0 232.0 7160.0 85.9 8905.0 1.5563
  7. 920.0 268.0 8804.0 86.5 7388.0 0.8976
  15. 835.0 271.0 8108.0 85.2 5348.0 0.7482
  22. 1000.0 237.0 6370.0 83.8 8056.0 0.7160
  29. 1150.0 192.0 6441.0 82.1 6960.0 0.3010
  37. 990.0 202.0 5154.0 79.2 5690.0 0.3617
  44. 840.0 184.0 5896.0 81.2 6932.0 0.1139
  58. 650.0 200.0 5336.0 80.6 5400.0 0.1139
  65. 640.0 180.0 5041.0 78.4 3177.0 -0.2218
  72. 831.0 165.0 5012.0 79.3 4461.0 -0.1549
  80. 570.0 151.0 4825.0 78.7 3901.0 0.0000
  86. 570.0 171.0 4391.0 78.0 5002.0 0.0000
  93. 510.0 243.0 4320.0 72.3 4665.0 -0.0969
 100. 555.0 147.0 3709.0 74.9 4642.0 -0.2218
 107. 460.0 286.0 3969.0 74.4 4840.0 -0.3979
 122. 275.0 198.0 3558.0 72.5 4479.0 -0.1549
 129. 510.0 196.0 4361.0 57.7 4200.0 -0.2218
 151. 165.0 210.0 3301.0 71.8 3410.0 -0.3979
### 9.3 Program Results

**g02ecc Example Program Results**

<table>
<thead>
<tr>
<th>Number of parameters</th>
<th>CP</th>
<th>RSQ</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>55.45</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>56.84</td>
<td>0.0082</td>
<td>TKN</td>
</tr>
<tr>
<td>1</td>
<td>20.33</td>
<td>0.5054</td>
<td>TVS</td>
</tr>
<tr>
<td>1</td>
<td>13.50</td>
<td>0.5983</td>
<td>BOD</td>
</tr>
<tr>
<td>1</td>
<td>6.57</td>
<td>0.6926</td>
<td>COD</td>
</tr>
<tr>
<td>1</td>
<td>6.29</td>
<td>0.6965</td>
<td>TS</td>
</tr>
<tr>
<td>2</td>
<td>21.36</td>
<td>0.5185</td>
<td>TKN TVS</td>
</tr>
<tr>
<td>2</td>
<td>11.33</td>
<td>0.6551</td>
<td>BOD TVS</td>
</tr>
<tr>
<td>2</td>
<td>9.09</td>
<td>0.6856</td>
<td>BOD TKN</td>
</tr>
<tr>
<td>2</td>
<td>7.70</td>
<td>0.7045</td>
<td>BOD COD</td>
</tr>
<tr>
<td>2</td>
<td>7.33</td>
<td>0.7095</td>
<td>TKN TS</td>
</tr>
<tr>
<td>2</td>
<td>7.16</td>
<td>0.7119</td>
<td>TS TVS</td>
</tr>
<tr>
<td>2</td>
<td>6.88</td>
<td>0.7157</td>
<td>BOD TS</td>
</tr>
<tr>
<td>2</td>
<td>6.87</td>
<td>0.7158</td>
<td>TKN COD</td>
</tr>
<tr>
<td>2</td>
<td>5.27</td>
<td>0.7376</td>
<td>TVS COD</td>
</tr>
<tr>
<td>2</td>
<td>1.74</td>
<td>0.7857</td>
<td>TS COD</td>
</tr>
<tr>
<td>3</td>
<td>8.68</td>
<td>0.7184</td>
<td>BOD TKN TVS</td>
</tr>
<tr>
<td>3</td>
<td>8.16</td>
<td>0.7255</td>
<td>TKN TS TVS</td>
</tr>
<tr>
<td>3</td>
<td>8.15</td>
<td>0.7256</td>
<td>BOD TS TVS</td>
</tr>
<tr>
<td>3</td>
<td>7.15</td>
<td>0.7392</td>
<td>BOD TVS COD</td>
</tr>
<tr>
<td>3</td>
<td>6.51</td>
<td>0.7479</td>
<td>BOD TKN COD</td>
</tr>
<tr>
<td>3</td>
<td>6.25</td>
<td>0.7515</td>
<td>BOD TKN TS</td>
</tr>
<tr>
<td>3</td>
<td>5.67</td>
<td>0.7595</td>
<td>TKN TVS COD</td>
</tr>
<tr>
<td>3</td>
<td>3.44</td>
<td>0.7898</td>
<td>BOD TS COD</td>
</tr>
<tr>
<td>3</td>
<td>3.42</td>
<td>0.7900</td>
<td>TS TVS COD</td>
</tr>
<tr>
<td>3</td>
<td>2.32</td>
<td>0.8050</td>
<td>TKN TS COD</td>
</tr>
<tr>
<td>4</td>
<td>7.70</td>
<td>0.7591</td>
<td>BOD TKN TS TVS</td>
</tr>
<tr>
<td>4</td>
<td>6.78</td>
<td>0.7716</td>
<td>BOD TKN TVS COD</td>
</tr>
<tr>
<td>4</td>
<td>5.07</td>
<td>0.7948</td>
<td>BOD TS TVS COD</td>
</tr>
<tr>
<td>4</td>
<td>4.32</td>
<td>0.8050</td>
<td>BOD TKN TS COD</td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td>0.8094</td>
<td>TKN TS TVS COD</td>
</tr>
<tr>
<td>5</td>
<td>6.00</td>
<td>0.8094</td>
<td>BOD TKN TS TVS COD</td>
</tr>
</tbody>
</table>