

NAG 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

```
#include <nag.h>
#include <nagg02.h>

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 = $\mathbf{tss} = \sum (y - \bar{y})^2$ (if mean is fitted, otherwise $\mathbf{tss} = \sum y^2$) and

Regression Sum of Squares = $\text{RegSS} = \mathbf{tss} - \mathbf{rss}$, where

\mathbf{rss} = 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.

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

where p is the number of arguments (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 $C_p \simeq p$. 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

- Draper N R and Smith H (1985) *Applied Regression Analysis* (2nd Edition) Wiley
 Weisberg S (1985) *Applied Linear Regression* Wiley

5 Arguments

- 1: **mean** – Nag_IncludeMean *Input*
On entry: indicates if a mean term is to be included.
mean = Nag_MeanInclude
 A mean term, intercept, will be included in the model.
mean = Nag_MeanZero
 The model will pass through the origin, zero-point.
Constraint: **mean** = Nag_MeanInclude or Nag_MeanZero.
- 2: **n** – Integer *Input*
On entry: n , the number of observations used in the regression model.
Constraint: **n** must be greater than $2 \times p_{\max}$, where p_{\max} is the largest number of independent variables fitted (including the mean if fitted).
- 3: **sigsq** – double *Input*
On entry: the best estimate of true variance of the errors, $\hat{\sigma}^2$.
Constraint: **sigsq** > 0.0.
- 4: **tss** – double *Input*
On entry: the total sum of squares for the regression model.
Constraint: **tss** > 0.0.
- 5: **nmod** – Integer *Input*
On entry: the number of regression models.
Constraint: **nmod** > 0.
- 6: **nterms[nmod]** – const Integer *Input*
On entry: **nterms**[$i - 1$] must contain the number of independent variables (not counting the mean) fitted to the i th model, for $i = 1, 2, \dots, \mathbf{nmod}$.
- 7: **rss[nmod]** – const double *Input*
On entry: **rss**[$i - 1$] must contain the residual sum of squares for the i th model.
Constraint: **rss**[$i - 1$] ≤ **tss**, for $i = 1, 2, \dots, \mathbf{nmod}$.
- 8: **rsq[nmod]** – double *Output*
On exit: **rsq**[$i - 1$] contains the R^2 -value for the i th model, for $i = 1, 2, \dots, \mathbf{nmod}$.
- 9: **cp[nmod]** – double *Output*
On exit: **cp**[$i - 1$] contains the C_p -value for the i th model, for $i = 1, 2, \dots, \mathbf{nmod}$.
- 10: **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 $\langle value \rangle$ had an illegal value.

NE_INT

On entry, $nmod = \langle value \rangle$.

Constraint: $nmod > 0$.

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_MODEL_PARAMETERS

On entry, number of parameters for model $\langle value \rangle$ is too large for n . $n = \langle value \rangle$, number of parameters = $\langle value \rangle$.

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.

NE_REAL

On entry, $sig\mathit{sq} = \langle value \rangle$.

Constraint: $sig\mathit{sq} > 0.0$.

On entry, $t\mathit{ss} = \langle value \rangle$.

Constraint: $t\mathit{ss} > 0.0$.

NE_REAL_ARRAY_ELEM_CONS

On entry, $cp[\langle value \rangle] = \langle value \rangle$.

Constraint: $cp[i] \geq 0.0$, for all i .

On entry, $r\mathit{ss}[\langle value \rangle] = \langle value \rangle$ and $t\mathit{ss} = \langle value \rangle$.

Constraint: $r\mathit{ss}[i] \leq t\mathit{ss}$, for all i .

7 Accuracy

Accuracy is sufficient for all practical purposes.

8 Parallelism and Performance

`nag_cp_stat` (g02ecc) is not threaded in any implementation.

9 Further Comments

None.

10 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_regn` (`g02eac`). The values of R^2 and C_p are then computed and printed along with the names of variables in the models.

10.1 Program Text

```

/* nag_cp_stat (g02ecc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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, n, nmod, pdx;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    char **model = 0;
    double *cp = 0, *rsq = 0, *rss = 0, *wtptr = 0, *x = 0, *y = 0;
    Integer *sx = 0, *mrnk = 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

    INIT_FAIL(fail);

    exit_status = 0;
    printf("nag_cp_stat (g02ecc) 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

    /* Allocate memory */
    if (!(x = NAG_ALLOC(n * m, double)) ||

```

```

        !(y = NAG_ALLOC(n, double)) || !(sx = NAG_ALLOC(m, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

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

    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= m; ++j)
#ifdef _WIN32
            scanf_s("%lf", &X(i, j));
#else
            scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
            scanf_s("%lf%*[\n] ", &y[i - 1]);
#else
            scanf("%lf%*[\n] ", &y[i - 1]);
#endif
        }

        num_models = 1;
        for (j = 1; j <= m; ++j) {
#ifdef _WIN32
            scanf_s("%" NAG_IFMT " ", &sx[j - 1]);
#else
            scanf("%" NAG_IFMT " ", &sx[j - 1]);
#endif
            if (sx[j - 1] == 1)
                num_models <<= 1;
        }
#ifdef _WIN32
        scanf_s("%*[\n] ");
#else
        scanf("%*[\n] ");
#endif

    /* Allocate memory */
    if (!(model = NAG_ALLOC(num_models * m, 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)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Calculate residual sums of squares using nag_all_regsn (g02eac) */

    /* nag_all_regsn (g02eac).
     * Computes residual sums of squares for all possible linear
     * regressions for a set of independent variables
     */
    nag_all_regsn(order, Nag_MeanInclude, n, m, x, pdx, var_names, sx, y, wtptr,
                  &nmod, (const char **) model, rss, nterms, mrank, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_all_regsn (g02eac).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
}

```

```

tss = rss[0];
sigsg = rss[nmod - 1] / (n - nterms[nmod - 1] - 1);
/* nag_cp_stat (g02ecc).
 * Calculates R^2 and C_P values from residual sums of
 * squares
 */
nag_cp_stat(Nag_MeanInclude, n, sigsg, tss, nmod, nterms, rss, rsq, cp,
            &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_cp_stat (g02ecc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

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

return exit_status;
}

```

10.2 Program Data

nag_cp_stat (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.  583.0 165.0 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
171. 244.0 327.0 2964.0 72.5 3360.0 -0.5229
220.  79.0 334.0 2777.0 71.9 2599.0 -0.0458
0    1    1    1    1    1

```

10.3 Program Results

nag_cp_stat (g02ecc) Example Program Results

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