NAG Library Function Document

nag_regsn_std_resid_influence (g02fac)

1 Purpose

nag_regsn_std_resid_influence (g02fac) calculates two types of standardized residuals and two measures of influence for a linear regression.

2 Specification

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

```
void nag_regsn_std_resid_influence (Integer n, Integer ip, Integer nres,
     const double res[], const double h[], double rms, double sres[],
     NagError *fail)
```

3 Description

For the general linear regression model is defined by

$$y = X\beta + \epsilon$$

where y is a vector of length n of the dependent variable,
X is an n by p matrix of the independent variables,
β is a vector of length p of unknown arguments,
and ϵ is a vector of length n of unknown random errors such that $\operatorname{var}\epsilon = \sigma^2 I$.

The residuals are given by

$$r = y - \hat{y} = y - X\hat{\beta}.$$

The fitted values, $\hat{y} = X\hat{\beta}$, can be written as Hy for an n by n matrix H. The ith diagonal element of H, h_i, gives a measure of the influence of the ith value of the independent variables on the fitted regression model. The values of r and the h_i are returned by nag_regsn_mult_linear (g02dac).

nag_regsn_std_resid_influence (g02fac) calculates statistics which help to indicate if an observation is extreme and having an undue influence on the fit of the regression model. Two types of standardized residual are calculated:

(a) The ith residual is standardized by its variance when the estimate of σ^2, s^2, is calculated from all the data; known as internal studentization.

$$RI_i = \frac{r_i}{s\sqrt{1-h_i}}.$$

(b) The ith residual is standardized by its variance when the estimate of σ^2, s_{-i}^2 is calculated from the data excluding the ith observation; known as external studentization.

$$RE_i = \frac{r_i}{s_{-i}\sqrt{1-h_i}} = r_i\sqrt{\frac{n-p-1}{n-p-RI_i^2}}.$$

The two measures of influence are:

(a) Cook's D

$$D_i = \frac{1}{p} RE_i^2 \frac{h_i}{1 - h_i}$$

(b) Atkinson's T

$$T_i = |RE_i| \sqrt{\left(\frac{n-p}{p}\right)\left(\frac{h_i}{1-h_i}\right)}.$$

4 References

Atkinson A C (1981) Two graphical displays for outlying and influential observations in regression *Biometrika* **68** 13–20

Cook R D and Weisberg S (1982) *Residuals and Influence in Regression* Chapman and Hall

5 Arguments

1: **n** – Integer *Input*

On entry: number of observations included in the regression, n.

Constraint: $\mathbf{n} > \mathbf{ip} + 1$.

2: **ip** – Integer *Input*

On entry: the number of linear arguments estimated in the regression model, p.

Constraint: $\mathbf{ip} \geq 1$.

3: **nres** – Integer *Input*

On entry: the number of residuals.

Constraint: $1 \leq \mathbf{nres} \leq \mathbf{n}$.

4: **res**[**nres**] – const double *Input*

On entry: the residuals, r_i.

5: **h**[**nres**] – const double *Input*

On entry: the diagonal elements of H, h_i, corresponding to the residuals in **res**.

Constraint: $0.0 < \mathbf{h}[i] < 1.0$, for $i = 0, 1, \ldots, \mathbf{nres} - 1$.

6: **rms** – double *Input*

On entry: the estimate of σ^2 based on all n observations, s^2, i.e., the residual mean square.

Constraint: $\mathbf{rms} > 0.0$.

7: **sres**[**nres** × **4**] – double *Output*

On exit: the standardized residuals and influence statistics.

For the observation with residual given in **res**[i]:

$\mathbf{sres}[(i) \times 4]$ is the internally studentized residual

$\mathbf{sres}[(i) \times 4 + 1]$ is the externally studentized residual

$\mathbf{sres}[(i) \times 4 + 2]$ is Cook's D statistic

$\mathbf{sres}[(i) \times 4 + 3]$ is Atkinson's T statistic.

8: **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_2_INT_ARG_GT

On entry, **nres** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These arguments must satisfy **nres** ≥ **n**.

NE_2_INT_ARG_LE

On entry, **n** = ⟨*value*⟩ while **ip** + 1 = ⟨*value*⟩. These arguments must satisfy **n** > **ip** + 1.

NE_INT_ARG_LT

On entry, **ip** = ⟨*value*⟩.
Constraint: **ip** ≥ 1.

On entry, **nres** = ⟨*value*⟩.
Constraint: **nres** ≥ 1.

NE_REAL_ARG_GE

On entry, **h**[⟨*value*⟩] must not be greater than or equal to 1.0: **h**[⟨*value*⟩] = ⟨*value*⟩.

NE_REAL_ARG_LE

On entry, **h**[⟨*value*⟩] must not be less than or equal to 0.0: **h**[⟨*value*⟩] = ⟨*value*⟩.

On entry, **rms** must not be less than or equal to 0.0: **rms** = ⟨*value*⟩.

NE_RESID_LARG

On entry, the value of a residual is too large for the given value of **rms**: **res**[⟨*value*⟩] = ⟨*value*⟩, **rms** = ⟨*value*⟩.

7 Accuracy

Accuracy is sufficient for all practical purposes.

8 Parallelism and Performance

nag_regsn_std_resid_influence (g02fac) is not threaded in any implementation.

9 Further Comments

None.

10 Example

A set of 24 residuals and h_i values from an 11 argument model fitted to the cloud seeding data considered in Cook and Weisberg (1982) are input and the standardized residuals etc calculated and printed for the first 10 observations.

10.1 Program Text

```
/* nag_regsn_std_resid_influence (g02fac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define SRES(I, J) sres[(I) *tdsres + J]
int main(void)
{
  Integer exit_status = 0, i, ip, j, n, nres, tdsres;
  double *h = 0, *res = 0, rms, *sres = 0;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_regsn_std_resid_influence (g02fac) 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 " %" NAG_IFMT " %lf", &n, &ip, &nres,
          &rms);
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT " %" NAG_IFMT " %lf", &n, &ip, &nres,
        &rms);
#endif
  if (nres <= n && n > ip + 1) {
    if (!(h = NAG_ALLOC(n, double)) ||
        !(res = NAG_ALLOC(n, double)) || !(sres = NAG_ALLOC(n * 4, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    tdsres = 4;
  }
  else {
    printf("Invalid nres or n or ip.\n");
    exit_status = 1;
    return exit_status;
  }
  for (i = 0; i < n; i++)
#ifdef _WIN32
    scanf_s("%lf%lf", &res[i], &h[i]);
#else
    scanf("%lf%lf", &res[i], &h[i]);
#endif
  /* nag_regsn_std_resid_influence (g02fac).
   * Calculates standardized residuals and influence
   * statistics
   */
  nag_regsn_std_resid_influence(n, ip, nres, res, h, rms, sres, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_regsn_std_resid_influence (g02fac).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }
```

```
  printf("\n");
  printf("         Internally     Externally\n");
  printf(" Obs.    studentized    studentized    Cook's D    Atkinson's T\n");
  printf("          residuals      residuals\n\n");
  for (i = 0; i < nres; i++) {
    printf("%2" NAG_IFMT "", i + 1);
    for (j = 0; j < 4; j++)
      printf("%14.3f", SRES(i, j));
    printf("\n");
  }
END:
  NAG_FREE(h);
  NAG_FREE(res);
  NAG_FREE(sres);
  return exit_status;
}
```

10.2 Program Data

```
nag_regsn_std_resid_influence (g02fac) Example Program Data
 24 11 10 .5798
      0.2660        0.5519
     -0.1387        0.9746
     -0.2971        0.6256
      0.5926        0.3144
     -0.4013        0.4106
      0.1396        0.6268
     -1.3173        0.5479
      1.1226        0.2325
      0.0321        0.4115
     -0.7111        0.3577
      0.3439        0.3342
     -0.4379        0.1673
      0.0633        0.3874
     -0.0936        0.1705
      0.9968        0.3466
      0.0209        0.3743
     -0.4056        0.7527
      0.1396        0.9069
      0.0327        0.2610
      0.2970        0.6256
     -0.2277        0.2485
      0.5180        0.3072
      0.5301        0.5848
     -1.0650        0.4794
```

10.3 Program Results

```
nag_regsn_std_resid_influence (g02fac) Example Program Results

         Internally     Externally
 Obs.    studentized    studentized    Cook's D    Atkinson's T
          residuals      residuals

 1         0.522          0.507         0.030         0.611
 2        -1.143         -1.158         4.557        -7.797
 3        -0.638         -0.622         0.062        -0.875
 4         0.940          0.935         0.037         0.689
 5        -0.686         -0.672         0.030        -0.610
 6         0.300          0.289         0.014         0.408
 7        -2.573         -3.529         0.729        -4.223
 8         1.683          1.828         0.078         1.094
 9         0.055          0.053         0.000         0.048
10        -1.165         -1.183         0.069        -0.960
```