nag_regress_confid_interval (g02cbc)

1. Purpose

nag_regress_confid_interval (g02cbc) performs a simple linear regression with or without a constant term. The data is optionally weighted, and confidence intervals are calculated for the predicted and average values of y at a given x.

2. Specification

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

void nag_regress_confid_interval(Nag_SumSquare mean, Integer n,
   double x[], double y[], double wt[], double clm, double clp,
   double yhat[], double yml[], double ymu[], double yl[], double yu[],
   double h[], double res[], double *rms, NagError *fail)
```

3. Description

This function fits a straight line model of the form,

$$E(y) = a + bx$$

where $E(y)$ is the expected value of the variable y, to the data points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

such that

$$y_i = a + bx_i + e_i, \quad i = 1, 2, \ldots, n,$$

where the e_i values are independent random errors. The ith data point may have an associated weight w_i. The values of a and b are estimated by minimizing $\sum w_i e_i^2$ (if the weights option is not selected then $w_i = 1.0$).
The fitted values $\hat{y}_i$ are calculated using

$$\hat{y}_i = \hat{a} + \hat{b}x_i$$

where

$$\hat{a} = \bar{y} - b\bar{x} \qquad \hat{b} = \frac{\sum w_i(x_i - \bar{x})(y_i - \bar{y})}{\sum w_i(x_i - \bar{x})^2}$$

and the weighted means $\bar{x}$ and $\bar{y}$ are given by

$$\bar{y} = \frac{\sum w_i y_i}{\sum w_i} \quad \text{and} \quad \bar{x} = \frac{\sum w_i x_i}{\sum w_i}.$$

The residuals of the regression are calculated using

$$res_i = y_i - \hat{y}_i$$

and the residual mean square about the regression rms, is determined using

$$rms = \frac{\sum w_i(y_i - \hat{y}_i)^2}{df}$$

where df (the number of degrees of freedom) has the following values

$$df = \sum w_i - 2 \quad \text{where } \mathbf{mean} = \mathbf{Nag\_AboutMean}$$
$$df = \sum w_i - 1 \quad \text{where } \mathbf{mean} = \mathbf{Nag\_AboutZero}.$$

Note: the weights should be scaled to give the required degrees of freedom.
The function calculates predicted y estimates for a value of x, x_i^*, is given by

$$y_i^* = \hat{a} + \hat{b}x_i^*$$

this prediction has a standard error

$$serr\_pred = \sqrt{rms}\sqrt{1 + \frac{1}{\sum w_i} + \frac{(x_i^* - \bar{x})^2}{\sum w_i(x_i - \bar{x})^2}}.$$

The $(1-\alpha)$ confidence interval for this estimation of y is given by

$$y_i^* \pm t_{df}(1-\alpha/2).serr\_pred$$

where $t_{df}(1-\alpha/2)$ refers to the $(1-\alpha/2)$ point of the t distribution with df degrees of freedom (e.g. when $df = 20$ and $\alpha = 0.1$, $t_{20}(0.95) = 2.086$). If the user specifies the probability $clp = 0.9(\alpha = 0.1)$ then the lower limit of this interval is

$$yl_i = y_i^* - t_{df}(0.95).serr\_pred$$

and the upper limit is

$$yu_i = y_i^* + t_{df}(0.95).serr\_pred.$$

The mean value of y at x_i is estimated by the fitted value $\hat{y}_i$. This has a standard error of

$$serr\_arg = \sqrt{rms}\sqrt{\frac{1}{\sum w_i} + \frac{(x_i - \bar{x})^2}{\sum w_i(x_i - \bar{x})^2}}$$

and a $(1-\alpha)$ confidence interval is given by

$$\hat{y}_i \pm t_{df}(1-\alpha/2).serr\_arg.$$

For example, if the user specifies the probability $clm = 0.6(\alpha = 0.4)$ then the lower limit of this interval is

$$yml_i = \hat{y}_i - t_{df}(0.8).serr\_arg$$

and the upper limit is

$$ymu_i = \hat{y}_i + t_{df}(0.8).serr\_arg.$$

The leverage, h_i, is a measure of the influence a value x_i has on the fitted line at that point, $\hat{y}_i$. The leverage is given by

$$h_i = \frac{w_i}{\sum w_i} + \frac{w_i(x_i - \bar{x})^2}{\sum w_i(x_i - \bar{x})^2}$$

so it can be seen that

$$\begin{aligned} serr\_arg &= \sqrt{rms}\sqrt{h_i/w_i} \\ \text{and} \quad serr\_pred &= \sqrt{rms}\sqrt{1 + h_i/w_i} \end{aligned}$$

Similar formulae can be derived for the case when the line goes through the origin, that is $a = 0$.

4. Parameters

mean

Input: indicates whether nag_regress_confid_interval is to include a constant term in the regression.
If **mean** = **Nag_AboutMean**, the constant term, a, is included.
If **mean** = **Nag_AboutZero**, the constant term, a, is not included, i.e., $a = 0$.
Constraint: **mean** = **Nag_AboutMean** or **Nag_AboutZero**.

n

Input: the number of observations, **n**.
Constraint: if **mean** = **Nag_AboutMean** **n** ≥ 2. if **mean** = **Nag_AboutZero** **n** ≥ 1.

x[n]

Input: observations on the independent variable, x.
Constraint: all the values of x must not be identical.

y[n]
: Input: observations on the dependent variable, y.

wt[n]
: Input: if weighted estimates are required then **wt** must contain the weights to be used in the weighted regression. Otherwise **wt** need not be defined and may be set to the null pointer **NULL**, i.e.(double *)0.
Usually **wt**$[i-1]$ will be an integral value corresponding to the number of observations associated with the ith data point, or zero if the ith data point is to be ignored. The sum of the weights therefore represents the effective total number of observations used to create the regression line.
If **wt** = **NULL**, then the effective number of observations is n.
Constraint: **wt** = **NULL** or **wt**$[i-1] \geq 0.0$, for $i = 1, \ldots, n$.

clm
: Input: the confidence level for the confidence intervals for the mean.
Constraint: $0.0 <$ **clm** < 1.0.

clp
: Input: the confidence level for the prediction intervals.
Constraint: $0.0 <$ **clp** < 1.0.

yhat[n]
: Output: the fitted values, $\hat{\mathbf{y}}_\mathbf{i}$.

yml[n]
: Output: **yml**$[i-1]$ contains the lower limit of the confidence interval for the regression line at **x**$[i-1]$.

ymu[n]
: Output: **ymu**$[i-1]$ contains the upper limit of the confidence interval for the regression line at **x**$[i-1]$.

yl[n]
: Output: **yl**$[i-1]$ contains the lower limit of the confidence interval for the individual y value at **x**$[i-1]$.

yu[n]
: Output: **yu**$[i-1]$ contains the upper limit of the confidence interval for the individual y value at **x**$[i-1]$.

h[n]
: Output: the leverage of each observation on the regression.

res[n]
: Output: the residuals of the regression.

rms
: Output: the residual mean square about the regression.

fail
: The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_BAD_PARAM
: On entry, parameter **mean** had an illegal value.

NE_INT_ARG_LT
: On entry, **n** must not be less than 1: **n** = ⟨*value*⟩
if **mean** = **Nag_AboutZero**.
On entry, **n** must not be less than 2: **n** = ⟨*value*⟩
if **mean** = **Nag_AboutMean**.

NE_REAL_ARG_GE
: On entry, **clm** must not be greater than or equal to 1.0: **clm** = ⟨*value*⟩.
On entry, **clp** must not be greater than or equal to 1.0: **clp** = ⟨*value*⟩.

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

NE_NEG_WEIGHT
On entry, at least one of the weights is negative.

NE_WT_LOW
On entry, **wt** must contain at least 1 positive element if **mean** = **Nag_AboutZero** or at least 2 positive elements if **mean** = **Nag_AboutMean**.

NE_X_IDEN
On entry, all elements of **x** are equal.

NE_SW_LOW
On entry, the sum of elements of **wt** must be greater than 1.0 if **mean** = **Nag_AboutZero** and 2.0 if **mean** = **Nag_AboutMean**.

NW_RMS_EQ_ZERO
Residual mean sum of squares is zero, i.e., a perfect fit was obtained.

6. Further Comments

None.

6.1. Accuracy

The computations are believed to be stable.

6.2. References

Snedecor G W and Cochran W G (1967) *Statistical Methods.* (6th Edn) Iowa State University Press.

7. See Also

nag_simple_linear_regression (g02cac)

8. Example

A program to calculate the fitted value of y and the upper and lower limits of the confidence interval for the regression line as well as the individual y values.

8.1. Program Text

```
/* nag_regress_confid_interval(g02cbc) Example Program
 *
 * Copyright 1994 Numerical Algorithms Group.
 *
 * Mark 3, 1994.
 */

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

#define NMAX 10
```

```
main()
{
  Nag_SumSquare mean;
  Integer n;
  double x[NMAX], y[NMAX], wt[NMAX];
  double clm, clp;
  double yhat[NMAX], yml[NMAX], ymu[NMAX], yl[NMAX], yu[NMAX], h[NMAX],
  res[NMAX], rms;
  Integer i;
  char m, w;

  Vprintf("g02cbc Example Program Results\n");
  /*  Skip heading in data file */
  Vscanf("%*[^\n]");
  Vscanf("%ld\n",&n);
  if (n>=1 && n<= NMAX)
    {
      Vscanf("%lf%lf\n",&clm,&clp);
      Vscanf(" %c %c\n",&m,&w);
      if (m == 'm' || m == 'M')
        mean = Nag_AboutMean;
      else if (m == 'z'|| m == 'Z')
        mean = Nag_AboutZero;
      if (w == 'w' || w == 'W')
        for (i=0; i<n; i++)
          Vscanf("%lf%lf%lf\n",&x[i],&y[i],&wt[i]);
      else
        for (i = 0; i < n; ++i)
          Vscanf("%lf%lf\n",&x[i],&y[i]);

      g02cbc(mean, n, x, y, wt, clm, clp, yhat, yml, ymu, yl, yu, h, res,
             &rms, NAGERR_DEFAULT);

      Vprintf ("\ni       yhat[i]    yml[i]    ymu[i]      yl[i]      yu[i]\
 \n\n");
      for (i=0; i < n; ++i) {
        Vprintf("%ld %10.2f %10.2f", i, yhat[i], yml[i]);
        Vprintf("%10.2f  %10.2f %10.2f\n",ymu[i], yl[i], yu[i]);
      }
    }
  else
    {
      Vfprintf(stderr, "n is out of range:\
 n = %-3ld\n",n);
      exit(EXIT_FAILURE);
    }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
g02cbc Example Program Data
9
0.95 0.95
mw
1.0 4.0 1.0
2.0 4.0 2.0
4.0 5.1 1.0
2.0 4.0 1.0
2.0 6.0 1.0
3.0 5.2 1.0
7.0 9.1 1.0
4.0 2.0 1.0
2.0 4.1 1.0
```

8.3. Program Results

```
g02cbc Example Program Results

i       yhat[i]     yml[i]     ymu[i]      yl[i]      yu[i]

0       3.47        1.76       5.18       -0.46       7.40
1       4.14        2.87       5.42        0.38       7.90
2       5.49        4.15       6.84        1.71       9.27
3       4.14        2.87       5.42        0.38       7.90
4       4.14        2.87       5.42        0.38       7.90
5       4.82        3.70       5.94        1.11       8.53
6       7.52        4.51      10.53        2.87      12.16
7       5.49        4.15       6.84        1.71       9.27
8       4.14        2.87       5.42        0.38       7.90
```