NAG Library Function Document

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,
     const double x[], const double y[], const 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

nag_regress_confid_interval (g02cbc) 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

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

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

$df = \sum w_i - 2$ where **mean** = Nag_AboutMean

$df = \sum w_i - 1$ where **mean** = 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

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

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 you specify the probability $clp = 0.9(\alpha = 0.1)$ then the lower limit of this interval is

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

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 you specify the probability $clm = 0.6(\alpha = 0.4)$ then the lower limit of this interval is

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

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

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

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

4 References

Snedecor G W and Cochran W G (1967) *Statistical Methods* Iowa State University Press

5 Arguments

1: **mean** – Nag_SumSquare *Input*

On entry: indicates whether nag_regress_confid_interval (g02cbc) is to include a constant term in the regression.

mean = Nag_AboutMean
The constant term, a, is included.

mean = Nag_AboutZero
The constant term, a, is not included, i.e., $a = 0$.

Constraint: **mean** = Nag_AboutMean or Nag_AboutZero.

2: **n** – Integer *Input*

On entry: N, the number of observations.

Constraints:

if **mean** = Nag_AboutMean, $\mathbf{n} \geq 2$;
if **mean** = Nag_AboutZero, $\mathbf{n} \geq 1$.

3: **x**[**n**] – const double *Input*

On entry: observations on the independent variable, x.

Constraint: all the values of x must not be identical.

4: **y**[**n**] – const double *Input*

On entry: observations on the dependent variable, y.

5: **wt**[**n**] – const double *Input*

On entry: if weighted estimates are required then **wt** must contain the weights to be used in the weighted regression. Usually $\mathbf{wt}[i-1]$ 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 weights are not provided then **wt** must be set to **NULL** and the effective number of observations is **n**.

Constraint: if **wt** is not **NULL**, $\mathbf{wt}[i-1] = 0.0$, for $i = 1, 2, \ldots, n$.

6: **clm** – double *Input*

On entry: the confidence level for the confidence intervals for the mean.

Constraint: $0.0 < \mathbf{clm} < 1.0$.

7: **clp** – double *Input*

On entry: the confidence level for the prediction intervals.

Constraint: $0.0 < \mathbf{clp} < 1.0$.

8: **yhat**[**n**] – double *Output*

On exit: the fitted values, $\hat{\mathbf{y}}_i$.

9: **yml**[**n**] – double *Output*

On exit: $\mathbf{yml}[i-1]$ contains the lower limit of the confidence interval for the regression line at $\mathbf{x}[i-1]$.

10: **ymu**[**n**] – double *Output*

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

11: **yl**[**n**] – double *Output*

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

12: **yu**[**n**] – double *Output*

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

13: **h**[**n**] – double *Output*

On exit: the leverage of each observation on the regression.

14: **res**[**n**] – double *Output*

On exit: the residuals of the regression.

15: **rms** – double * *Output*

On exit: the residual mean square about the regression.

16: **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_BAD_PARAM

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

NE_INT_ARG_LT

On entry, **n** $= \langle value \rangle$.
Constraint: if **mean** = Nag_AboutMean, **n** ≥ 2.

On entry, **n** $= \langle value \rangle$.
Constraint: if **mean** = Nag_AboutZero, **n** ≥ 1.

NE_NEG_WEIGHT

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

NE_REAL_ARG_GE

On entry, **clm** must not be greater than or equal to 1.0: **clm** $= \langle value \rangle$.

On entry, **clp** must not be greater than or equal to 1.0: **clp** $= \langle value \rangle$.

NE_REAL_ARG_LE

On entry, **clm** must not be less than or equal to 0.0: **clm** $= \langle value \rangle$.

On entry, **clp** must not be less than or equal to 0.0: **clp** $= \langle value \rangle$.

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.

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.

NW_RMS_EQ_ZERO

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

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

nag_regress_confid_interval (g02cbc) is not threaded in any implementation.

9 Further Comments

None.

10 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.

10.1 Program Text

```
/* nag_regress_confid_interval (g02cbc) 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>

int main(void)
{
  Integer exit_status = 0, i, n;
  double clm, clp;
  double *h = 0, *res = 0, rms, *wt = 0, *x = 0, *y = 0, *yhat = 0;
  double *yl = 0, *yml = 0, *ymu = 0, *yu = 0;
  char nag_enum_arg[40];
  Nag_SumSquare mean;
  Nag_Boolean weight;
  NagError fail;

  INIT_FAIL(fail);
  printf("nag_regress_confid_interval (g02cbc) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "\n", &n);
#else
  scanf("%" NAG_IFMT "\n", &n);
#endif
#ifdef _WIN32
  scanf_s("%lf%lf\n", &clm, &clp);
#else
  scanf("%lf%lf\n", &clm, &clp);
#endif
#ifdef _WIN32
  scanf_s(" %39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf(" %39s", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  mean = (Nag_SumSquare) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s(" %39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf(" %39s", nag_enum_arg);
#endif
  weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
  if (n >= (mean == Nag_AboutMean ? 2 : 1)) {
    if (!(x = NAG_ALLOC(n, double)) ||
        !(y = NAG_ALLOC(n, double)) ||
        !(wt = NAG_ALLOC(n, double)) ||
        !(yhat = NAG_ALLOC(n, double)) ||
        !(yml = NAG_ALLOC(n, double)) ||
        !(ymu = NAG_ALLOC(n, double)) ||
        !(yl = NAG_ALLOC(n, double)) ||
        !(yu = NAG_ALLOC(n, double)) ||
        !(h = NAG_ALLOC(n, double)) || !(res = NAG_ALLOC(n, double))
          )
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  }
  else {
    printf("Invalid n.\n");
    exit_status = 1;
    return exit_status;
  }
  if (weight)
    for (i = 0; i < n; i++)
#ifdef _WIN32
      scanf_s("%lf%lf%lf\n", &x[i], &y[i], &wt[i]);
#else
      scanf("%lf%lf%lf\n", &x[i], &y[i], &wt[i]);
#endif
  else
    for (i = 0; i < n; ++i)
#ifdef _WIN32
      scanf_s("%lf%lf\n", &x[i], &y[i]);
#else
      scanf("%lf%lf\n", &x[i], &y[i]);
#endif

  /* nag_regress_confid_interval (g02cbc).
   * Simple linear regression confidence intervals for the
   * regression line and individual points
   */
  nag_regress_confid_interval(mean, n, x, y, wt, clm, clp, yhat, yml, ymu, yl,
                              yu, h, res, &rms, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_regress_confid_interval (g02cbc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\ni       yhat[i]    yml[i]    ymu[i]      yl[i]      yu[i]"
         " \n\n");
  for (i = 0; i < n; ++i) {
    printf("%" NAG_IFMT " %10.2f %10.2f", i, yhat[i], yml[i]);
    printf("%10.2f  %10.2f %10.2f\n", ymu[i], yl[i], yu[i]);
  }

END:
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(wt);
  NAG_FREE(yhat);
  NAG_FREE(yml);
  NAG_FREE(ymu);
  NAG_FREE(yl);
  NAG_FREE(yu);
  NAG_FREE(h);
  NAG_FREE(res);

  return exit_status;
}
```

10.2 Program Data

```
nag_regress_confid_interval (g02cbc) Example Program Data
9
0.95 0.95
Nag_AboutMean Nag_TRUE
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
```

10.3 Program Results

```
nag_regress_confid_interval (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
```