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 \( i \)th data point may have an associated weight \( w_i \). The values of \( a \) and \( b \) are estimated by minimizing \( \sum w_i e^2_i \) (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
\[ \hat{a} = y - \bar{b} \bar{x} \quad \hat{b} = \frac{\sum w_i (x_i - \bar{x})(y_i - \bar{y})}{\sum w_i (x_i - \bar{x})^2} \]

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

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

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

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

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

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

\[ y_{l_i} = y_i^* - t_{df}(0.95).\text{serr\_pred} \]

and the upper limit is

\[ y_{u_i} = y_i^* + t_{df}(0.95).\text{serr\_pred}. \]

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

\[ \text{serr\_arg} = \sqrt{\text{rms} \left( \frac{1}{\sum w_i} + \frac{(x_i - \bar{x})^2}{\sum w_i(x_i - \bar{x})^2} \right)} \]

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

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

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

\[ y_{ml_i} = \hat{y}_i - t_{df}(0.8).\text{serr\_arg} \]

and the upper limit is

\[ y_{mu_i} = \hat{y}_i + t_{df}(0.8).\text{serr\_arg}. \]

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

\[ \text{serr\_arg} = \sqrt{\text{rms} \left( h_i/w_i \right)} \]
\[ \text{serr\_pred} = \sqrt{\text{rms} \left( 1 + h_i/w_i \right)} \]

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

4 References

5 Arguments

1: mean – Nag_SumSquare  
   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  
   On entry: $N$, the number of observations.
   Constraints:
   \[
   \begin{align*}
   & \text{if mean} = \text{Nag}_\text{AboutMean}, n \geq 2; \\
   & \text{if mean} = \text{Nag}_\text{AboutZero}, n \geq 1.
   \end{align*}
   \]

3: x[n] – const double  
   On entry: observations on the independent variable, $x$.
   Constraint: all the values of $x$ must not be identical.

4: y[n] – const double  
   On entry: observations on the dependent variable, $y$.

5: wt[n] – const double  
   On entry: if weighted estimates are required then wt must contain the weights to be used in the weighted regression. Usually $wt[i - 1]$ will be an integral value corresponding to the number of observations associated with the $i$th data point, or zero if the $i$th 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, $wt[i - 1] = 0.0$, for $i = 1, 2, \ldots, n$.

6: clm – double  
   On entry: the confidence level for the confidence intervals for the mean.
   Constraint: $0.0 < \text{clm} < 1.0$.

7: clp – double  
   On entry: the confidence level for the prediction intervals.
   Constraint: $0.0 < \text{clp} < 1.0$.

8: yhat[n] – double  
   On exit: the fitted values, $\hat{y}_i$.

9: yml[n] – double  
   On exit: $yml[i - 1]$ contains the lower limit of the confidence interval for the regression line at $x[i - 1]$.
10:  \( \text{ymu}[n] \) – double  
    *Output*
    On exit: \( \text{ymu}[i - 1] \) contains the upper limit of the confidence interval for the regression line at \( x[i - 1] \).

11:  \( \text{yl}[n] \) – double  
    *Output*
    On exit: \( \text{yl}[i - 1] \) contains the lower limit of the confidence interval for the individual y value at \( x[i - 1] \).

12:  \( \text{yu}[n] \) – double  
    *Output*
    On exit: \( \text{yu}[i - 1] \) contains the upper limit of the confidence interval for the individual y value at \( x[i - 1] \).

13:  \( \text{h}[n] \) – double  
    *Output*
    On exit: the leverage of each observation on the regression.

14:  \( \text{res}[n] \) – double  
    *Output*
    On exit: the residuals of the regression.

15:  \( \text{rms} \) – double *  
    *Output*
    On exit: the residual mean square about the regression.

16:  \( \text{fail} \) – NagError *  
    *Input/Output*
    The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_BAD_PARAM**

On entry, argument \( \text{mean} \) had an illegal value.

**NE_INT_ARG_LT**

On entry, \( n \) = \( \langle \text{value} \rangle \).
Constraint: if \( \text{mean} = \text{Nag}\_\text{AboutMean} \), \( n \geq 2 \).

On entry, \( n \) = \( \langle \text{value} \rangle \).
Constraint: if \( \text{mean} = \text{Nag}\_\text{AboutZero} \), \( n \geq 1 \).

**NE_NEG_WEIGHT**

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

**NE_REAL_ARG_GE**

On entry, \( \text{clm} \) must not be greater than or equal to 1.0: \( \text{clm} = \langle \text{value} \rangle \).

On entry, \( \text{clp} \) must not be greater than or equal to 1.0: \( \text{clp} = \langle \text{value} \rangle \).

**NE_REAL_ARG_LE**

On entry, \( \text{clm} \) must not be less than or equal to 0.0: \( \text{clm} = \langle \text{value} \rangle \).

On entry, \( \text{clp} \) must not be less than or equal to 0.0: \( \text{clp} = \langle \text{value} \rangle \).

**NE_SW_LOW**

On entry, the sum of elements of \( \text{wt} \) must be greater than 1.0 if \( \text{mean} = \text{Nag}\_\text{AboutZero} \) and 2.0 if \( \text{mean} = \text{Nag}\_\text{AboutMean} \).
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
Not applicable.

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. *
 * Copyright 2014 Numerical Algorithms Group.
 */
#include <nag.h>
#include <stdio.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);
#endif
else
    scanf("%"NAG_IFMT"
", &n);
#endif
#ifdef _WIN32
    scanf_s("%lf%lf
", &clm, &clp);
#else
    scanf("%lf%lf
", &clm, &clp);
#endif
#ifdef _WIN32
    scanf_s(" %39s", nag_enum_arg, __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, __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;
}
#endif
for (i = 0; i < n; i++)
#ifdef _WIN32
    scanf_s("%lf%lf%lf
", &x[i], &y[i], &wt[i]);
#else
    scanf("%lf%lf%lf
", &x[i], &y[i], &wt[i]);
#endif
else
    for (i = 0; i < n; ++i)
#ifdef _WIN32
    scanf_s("%lf%lf
", &x[i], &y[i]);
#else
    scanf("%lf%lf
", &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,
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\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

<table>
<thead>
<tr>
<th>i</th>
<th>yhat[i]</th>
<th>yml[i]</th>
<th>ymu[i]</th>
<th>yl[i]</th>
<th>yu[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.47</td>
<td>1.76</td>
<td>5.18</td>
<td>-0.46</td>
<td>7.40</td>
</tr>
<tr>
<td>1</td>
<td>4.14</td>
<td>2.87</td>
<td>5.42</td>
<td>0.38</td>
<td>7.90</td>
</tr>
<tr>
<td>2</td>
<td>5.49</td>
<td>4.15</td>
<td>6.84</td>
<td>1.71</td>
<td>9.27</td>
</tr>
<tr>
<td>3</td>
<td>4.14</td>
<td>2.87</td>
<td>5.42</td>
<td>0.38</td>
<td>7.90</td>
</tr>
<tr>
<td>4</td>
<td>4.14</td>
<td>2.87</td>
<td>5.42</td>
<td>0.38</td>
<td>7.90</td>
</tr>
<tr>
<td>5</td>
<td>4.82</td>
<td>3.70</td>
<td>5.94</td>
<td>1.11</td>
<td>8.53</td>
</tr>
<tr>
<td>6</td>
<td>7.52</td>
<td>4.51</td>
<td>10.53</td>
<td>2.87</td>
<td>12.16</td>
</tr>
<tr>
<td>7</td>
<td>5.49</td>
<td>4.15</td>
<td>6.84</td>
<td>1.71</td>
<td>9.27</td>
</tr>
<tr>
<td>8</td>
<td>4.14</td>
<td>2.87</td>
<td>5.42</td>
<td>0.38</td>
<td>7.90</td>
</tr>
</tbody>
</table>