NAG Library Function Document

nag_durbin_watson_stat (g02fcc)

1 Purpose

nag_durbin_watson_stat (g02fcc) calculates the Durbin–Watson statistic, for a set of residuals, and the upper and lower bounds for its significance.

2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_durbin_watson_stat (Integer n, Integer p, const double res[],
     double *d, double *pdl, double *pdu, NagError *fail)
```

3 Description

For the general linear regression model

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

where y is a vector of length n of the dependent variable,

X is a 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.

The residuals are given by

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

and the fitted values, $\hat{y} = X\hat{\beta}$, can be written as Hy for a n by n matrix H. Note that when a mean term is included in the model the sum of the residuals is zero. If the observations have been taken serially, that is $y_1, y_2, \ldots, y_n$ can be considered as a time series, the Durbin–Watson test can be used to test for serial correlation in the ϵ_i, see Durbin and Watson (1950), Durbin and Watson (1951) and Durbin and Watson (1971).

The Durbin–Watson statistic is

$$d = \frac{\sum_{i=1}^{n-1} (r_{i+1} - r_i)^2}{\sum_{i=1}^{n} r_i^2}.$$

Positive serial correlation in the ϵ_i will lead to a small value of d while for independent errors d will be close to 2. Durbin and Watson show that the exact distribution of d depends on the eigenvalues of the matrix HA where the matrix A is such that d can be written as

$$d = \frac{r^{\mathrm{T}} A r}{r^{\mathrm{T}} r}$$

and the eigenvalues of the matrix A are $\lambda_j = (1 - \cos(\pi j/n))$, for $j = 1, 2, \ldots, n - 1$.

However bounds on the distribution can be obtained, the lower bound being

$$d_{\mathrm{l}} = \frac{\sum_{i=1}^{n-p} \lambda_i u_i^2}{\sum_{i=1}^{n-p} u_i^2}$$

and the upper bound being

$$d_{\mathrm{u}} = \frac{\sum_{i=1}^{n-p} \lambda_{i-1+p} u_i^2}{\sum_{i=1}^{n-p} u_i^2},$$

where the u_i are independent standard Normal variables. The lower tail probabilities associated with these bounds, p_{l} and p_{u}, are computed by nag_prob_durbin_watson (g01epc). The interpretation of the bounds is that, for a test of size (significance) α, if $p_l \leq \alpha$ the test is significant, if $p_u > \alpha$ the test is not significant, while if $p_{\mathrm{l}} > \alpha$ and $p_{\mathrm{u}} \leq \alpha$ no conclusion can be reached.

The above probabilities are for the usual test of positive auto-correlation. If the alternative of negative auto-correlation is required, then a call to nag_prob_durbin_watson (g01epc) should be made with the argument **d** taking the value of $4 - d$; see Newbold (1988).

4 References

Durbin J and Watson G S (1950) Testing for serial correlation in least squares regression. I *Biometrika* **37** 409–428

Durbin J and Watson G S (1951) Testing for serial correlation in least squares regression. II *Biometrika* **38** 159–178

Durbin J and Watson G S (1971) Testing for serial correlation in least squares regression. III *Biometrika* **58** 1–19

Granger C W J and Newbold P (1986) *Forecasting Economic Time Series* (2nd Edition) Academic Press

Newbold P (1988) *Statistics for Business and Economics* Prentice–Hall

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of residuals.

Constraint: **n** $>$ **p**.

2: **p** – Integer *Input*

On entry: p, the number of independent variables in the regression model, including the mean.

Constraint: **p** ≥ 1.

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

On entry: the residuals, $r_1, r_2, \ldots, r_n$.

Constraint: the mean of the residuals $\leq \sqrt{\epsilon}$, where $\epsilon =$ ***machine precision***.

4: **d** – double * *Output*

On exit: the Durbin–Watson statistic, d.

5: **pdl** – double * *Output*

On exit: lower bound for the significance of the Durbin–Watson statistic, p_{l}.

6: **pdu** – double * *Output*

On exit: upper bound for the significance of the Durbin–Watson statistic, p_{u}.

7: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **p** $= \langle value \rangle$.
Constraint: **p** ≥ 1.

NE_INT_2

On entry, **n** $= \langle value \rangle$ and **p** $= \langle value \rangle$.
Constraint: **n** $>$ **p**.

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.

NE_RESID_IDEN

On entry, all residuals are identical.

NE_RESID_MEAN

On entry, the mean of **res** is not approximately 0.0, mean $= \langle value \rangle$.

7 Accuracy

The probabilities are computed to an accuracy of at least 4 decimal places.

8 Parallelism and Performance

Not applicable.

9 Further Comments

If the exact probabilities are required, then the first $n - p$ eigenvalues of HA can be computed and nag_prob_lin_chi_sq (g01jdc) used to compute the required probabilities with the argument **c** set to 0.0 and the argument **d** set to the Durbin–Watson statistic d.

10 Example

A set of 10 residuals are read in and the Durbin–Watson statistic along with the probability bounds are computed and printed.

10.1 Program Text

```
/* nag_durbin_watson_stat (g02fcc) Example Program.
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

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

int main(void)
{
  /* Scalars */
  double   d, pdl, pdu;
  Integer  exit_status, i, p, n;
  NagError fail;

  /* Arrays */
  double   *res = 0;

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_durbin_watson_stat (g02fcc) Example Program Results\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &p);
  n = 10;

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

  for (i = 1; i <= n; ++i)
    scanf("%lf", &res[i - 1]);
  scanf("%*[^\n] ");

  /* nag_durbin_watson_stat (g02fcc).
   * Computes Durbin-Watson test statistic
   */
  nag_durbin_watson_stat(n, p, res, &d, &pdl, &pdu, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_durbin_watson_stat (g02fcc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\n");
  printf(" Durbin-Watson statistic %10.4f\n\n", d);
  printf(" Lower and upper bound %10.4f%10.4f\n", pdl, pdu);
 END:
  NAG_FREE(res);
  return exit_status;
```

```
}
```

10.2 Program Data

```
nag_durbin_watson_stat (g02fcc) Example Program Data
 2
3.735719 0.912755 0.683626 0.416693 1.9902
-0.444816 -1.283088 -3.666035 -0.426357 -1.918697
```

10.3 Program Results

```
nag_durbin_watson_stat (g02fcc) Example Program Results

 Durbin-Watson statistic     0.9238

 Lower and upper bound     0.0610    0.0060
```