NAG Library Function Document

nag_estim_gen_pareto (g07bfc)

1 Purpose
nag_estim_gen_pareto (g07bfc) estimates parameter values for the generalized Pareto distribution by using either moments or maximum likelihood.

2 Specification
#include <nag.h>
#include <nagg07.h>

void nag_estim_gen_pareto (Integer n, const double y[], Nag_OptimOpt optopt,
                           double *xi, double *beta, double asvc[], double obsvc[], double *ll,
                           NagError *fail)

3 Description
Let the distribution function of a set of \( n \) observations \( y_i, \quad i = 1, 2, \ldots, n \) be given by the generalized Pareto distribution:

\[
F(y) = \begin{cases} 
1 - \left(1 + \frac{\xi y}{\beta}\right)^{-1/\xi}, & \xi \neq 0 \\
1 - e^{-\frac{y}{\beta}}, & \xi = 0;
\end{cases}
\]

where
- \( \beta > 0 \)
- \( y \geq 0 \), when \( \xi \geq 0 \);
- \( 0 \leq y \leq -\frac{\beta}{\xi} \), when \( \xi < 0 \).

Estimates \( \hat{\xi} \) and \( \hat{\beta} \) of the parameters \( \xi \) and \( \beta \) are calculated by using one of:
- method of moments (MOM);
- probability-weighted moments (PWM);
- maximum likelihood estimates (MLE) that seek to maximize the log-likelihood:

\[
L = -n \ln \hat{\beta} - \left(1 + \frac{1}{\xi}\right) \sum_{i=1}^{n} \ln \left(1 + \frac{\hat{\xi} y_i}{\beta}\right).
\]

The variances and covariance of the asymptotic Normal distribution of parameter estimates \( \hat{\xi} \) and \( \hat{\beta} \) are returned if \( \hat{\xi} \) satisfies:
- \( \hat{\xi} < \frac{1}{4} \) for the MOM;
- \( \hat{\xi} < \frac{1}{2} \) for the PWM method;
- \( \hat{\xi} < -\frac{1}{2} \) for the MLE method.

If the MLE option is exercised, the observed variances and covariance of \( \hat{\xi} \) and \( \hat{\beta} \) is returned, given by the negative inverse Hessian of \( L \).
4 References

5 Arguments
1: \( n \) – Integer
   \textit{Input}
   \textit{On entry:} the number of observations.
   \textit{Constraint:} \( n > 1 \).

2: \( y[n] \) – const double
   \textit{Input}
   \textit{On entry:} the \( n \) observations \( y_i \), for \( i = 1, 2, \ldots, n \), assumed to follow a generalized Pareto distribution.
   \textit{Constraints:}
   \[
   y[i - 1] \geq 0.0; \\
   \sum_{i=1}^{n} y[i - 1] > 0.0.
   \]

3: \( \texttt{optopt} \) – Nag_OptimOpt
   \textit{Input}
   \textit{On entry:} determines the method of estimation, set:
   \begin{align*}
   \texttt{optopt} &= \texttt{Nag\_PWM} & \text{For the method of probability-weighted moments.} \\
   \texttt{optopt} &= \texttt{Nag\_MOM} & \text{For the method of moments.} \\
   \texttt{optopt} &= \texttt{Nag\_MOMMLE} & \text{For maximum likelihood with starting values given by the method of moments estimates.} \\
   \texttt{optopt} &= \texttt{Nag\_PWMMLE} & \text{For maximum likelihood with starting values given by the method of probability-weighted moments.}
   \end{align*}
   \textit{Constraint:} \( \texttt{optopt} = \texttt{Nag\_PWM, Nag\_MOM, Nag\_MOMMLE or Nag\_PWMMLE} \).

4: \( x_i \) – double *
   \textit{Output}
   \textit{On exit:} the parameter estimate \( \hat{\xi} \).

5: \( \texttt{beta} \) – double *
   \textit{Output}
   \textit{On exit:} the parameter estimate \( \hat{\beta} \).

6: \( \texttt{asvc[4]} \) – double
   \textit{Output}
   \textit{On exit:} the variance-covariance of the asymptotic Normal distribution of \( \hat{\xi} \) and \( \hat{\beta} \). \( \texttt{asvc[0]} \) contains the variance of \( \hat{\xi} \); \( \texttt{asvc[3]} \) contains the variance of \( \hat{\beta} \); \( \texttt{asvc[1]} \) and \( \texttt{asvc[2]} \) contain the covariance of \( \hat{\xi} \) and \( \hat{\beta} \).

7: \( \texttt{obsvc[4]} \) – double
   \textit{Output}
   \textit{On exit:} if maximum likelihood estimates are requested, the observed variance-covariance of \( \hat{\xi} \) and \( \hat{\beta} \). \( \texttt{obsvc[0]} \) contains the variance of \( \hat{\xi} \); \( \texttt{obsvc[3]} \) contains the variance of \( \hat{\beta} \); \( \texttt{obsvc[1]} \) and \( \texttt{obsvc[2]} \) contain the covariance of \( \hat{\xi} \) and \( \hat{\beta} \).
6 Error Indicators and Warnings

NE_ALLOC_FAIL
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
On entry, argument <value> had an illegal value.

NE_INT
On entry, \( n = \langle value \rangle \).
Constraint: \( n > 1 \).

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.
An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_OPTIMIZE
The optimization of log-likelihood failed to converge; no maximum likelihood estimates are returned. Try using the other maximum likelihood option by resetting \texttt{optopt}. If this also fails, moments-based estimates can be returned by an appropriate setting of \texttt{optopt}.
Variance of data in \( y \) is too low for method of moments optimization.

NE_REAL_ARRAY
On entry, \( y[\langle value \rangle] = \langle value \rangle \).
Constraint: \( y[i - 1] \geq 0.0 \) for all \( i \).

NE_ZERO_SUM
The sum of \( y \) is zero within machine precision.

NW_PARAM_DIST
The asymptotic distribution of parameter estimates is invalid and the distribution of maximum likelihood estimates cannot be calculated for the returned parameter estimates because the Hessian matrix could not be inverted.

NW_PARAM_DIST_ASYM
The asymptotic distribution is not available for the returned parameter estimates.
The distribution of maximum likelihood estimates cannot be calculated for the returned parameter estimates because the Hessian matrix could not be inverted.

7 Accuracy
Not applicable.

8 Parallelism and Performance

nag_estim_gen_pareto (g07bfc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

The search for maximum likelihood parameter estimates is further restricted by requiring

$$1 + \frac{\xi \beta}{B} > 0,$$

as this avoids the possibility of making the log-likelihood $L$ arbitrarily high.

10 Example

This example calculates parameter estimates for 23 observations assumed to be drawn from a generalized Pareto distribution.

10.1 Program Text

/* nag_estim_gen_pareto (g07bfc) Example Program. * * Copyright 2014 Numerical Algorithms Group. * * Mark 9, 2009. */ /* Pre-processor includes */ #include <stdio.h> #include <math.h> #include <nag.h> #include <nag_stdlib.h> #include <nag07.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, n;

    /* Double scalar and array declarations */
    double asvc[4], beta, ll, obsvc[4], xi, *y = 0;

    /* Character scalar and array declarations */
    char soptopt[12];

    /* NAG types */
    NagError fail;
    Nag_OptimOpt optopt;
/* Initialise the error structure */
INIT_FAIL(fail);

printf("nag_estim_gen_pareto (g07bfc) Example Program Results\n\n");

/* Skip header in data file */
#ifdef _WIN32
    scanf_s("%*[\n "");
#else
    scanf("%*[\n "");
#endif

/* Read parameter values */
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%11s%*[\n "]", &n, soptopt, _countof(soptopt));
#else
    scanf("%"NAG_IFMT"%11s%*[\n "]", &n, soptopt);
#endif

optopt = (Nag_OptimOpt) nag_enum_name_to_value(soptopt);

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

/* Read data values */
for (i = 1; i <= n; i++)
#ifdef _WIN32
    scanf_s("%lf", &y[i - 1]);
#else
    scanf("%lf", &y[i - 1]);
#endif

/* Calculate the GPD parameter estimates */
nag_estim_gen_pareto(n, y, optopt, &xi, &beta, asvc, obsvc, &ll, &fail);

/* Print parameter estimates */
switch (fail.code)
{
    case NE_NOERROR:
    case NW_PARAM_DIST:
    case NW_PARAM_DIST_ASYM:
    case NW_PARAM_DIST_OBS:
        printf(" Parameter estimates\n");
        printf( " %-12s%12.6e\n %-12s%12.6e\n", "xi", xi, "beta", beta);
        break;
    default:
        printf("Error from nag_estim_gen_pareto (g07bfc).\n\n", fail.message);
        exit_status = -1;
        goto END;
}

/* Print parameter distribution */
if (optopt == Nag_MOMMLE || optopt == Nag_PWMMLE)
{
    switch (fail.code)
    {
        case NW_PARAM_DIST:
        case NW_PARAM_DIST_OBS:
            printf("%s\n", fail.message);
            exit_status = -1;
    }
break;
default:
    printf("\n Observed distribution\n");
    printf(" %-20s%12.6e\n %-20s%12.6e\n %-20s%12.6e\n", "Var(xi)", obsvc[0], "Var(beta)", obsvc[3], "Covar(xi,beta)", obsvc[1]);
    printf("\n Final log-likelihood: %12.6e\n", ll);
}
}
else
{
    switch (fail.code)
    {
        case NW_PARAM_DIST:
        case NW_PARAM_DIST_ASYM:
            printf(" %s\n", fail.message);
            exit_status = -1;
        default:
            printf("\n Asymptotic distribution\n");
            printf(" %-20s%12.6e\n %-20s%12.6e\n %-20s%12.6e\n", "Var(xi)", asvc[0], "Var(beta)", asvc[3], "Covar(xi,beta)", asvc[1]);
    }
}
END:
NAG_FREE(y);
return exit_status;
}

10.2 Program Data

nag_estim_gen_pareto (g07bfc) Example Program Data
23 Nag_PWMMLE
1.5800 0.1390 2.3624 2.9435 0.1363 0.9688
0.6585 2.8011 0.9880 1.7887 0.0630 0.3862
1.5130 0.0669 1.3659 0.4256 0.3485 27.8760
5.2503 1.1028 0.5273 1.3189 0.6490

10.3 Program Results

nag_estim_gen_pareto (g07bfc) Example Program Results

Parameter estimates
xi 5.404394e-01
beta 1.040549e+00

Observed distribution
Var(xi) 7.993204e-02
Var(beta) 1.198720e-01
Covar(xi,beta) -4.550923e-02

Final log-likelihood: -3.634433e+01