NAG Library Function Document
nag_deviates_chi_sq_vector (g01tcc)

1 Purpose

nag_deviates_chi_sq_vector (g01tcc) returns a number of deviates associated with the given probabilities of the $\chi^2$-distribution with real degrees of freedom.

2 Specification

```c
#include <nag.h>
#include <nagg01.h>
void nag_deviates_chi_sq_vector (Integer ltail,
     const Nag_TailProbability tail[], Integer lp, const double p[],
     Integer ldf, const double df[], double x[], Integer ivalid[],
     NagError *fail)
```

3 Description

The deviate, $x_{p_i}$, associated with the lower tail probability $p_i$ of the $\chi^2$-distribution with $\nu_i$ degrees of freedom is defined as the solution to

$$P(X_i \leq x_{p_i} ; \nu_i) = p_i = \frac{1}{2^{\nu_i/2} \Gamma(\nu_i/2)} \int_0^{x_{p_i}} e^{-X_i/2} X_i^{\nu_i/2 - 1} dX_i, \quad 0 \leq x_{p_i} < \infty; \nu_i > 0.$$ 

The required $x_{p_i}$ is found by using the relationship between a $\chi^2$-distribution and a gamma distribution, i.e., a $\chi^2$-distribution with $\nu_i$ degrees of freedom is equal to a gamma distribution with scale parameter 2 and shape parameter $\nu_i/2$.

For very large values of $\nu_i$, greater than $10^5$, Wilson and Hilferty’s Normal approximation to the $\chi^2$ is used; see Kendall and Stuart (1969).

The input arrays to this function are designed to allow maximum flexibility in the supply of vector arguments by re-using elements of any arrays that are shorter than the total number of evaluations required. See Section 2.6 in the g01 Chapter Introduction for further information.

4 References


5 Arguments

1:  Itail – Integer

   Input

   On entry: the length of the array tail.

   Constraint: Itail > 0.
2: \texttt{tail[ltail]} – \texttt{const Nag_TailProbability} \quad \textit{Input}

\textit{On entry:} indicates which tail the supplied probabilities represent. For \( j = (i - 1) \text{ mod } \text{ltail}, \) for \( i = 1, 2, \ldots, \text{max(ltail, lp, ldf)}: \)

\begin{itemize}
  \item \( \text{tail}[j] = \text{Nag_LowerTail} \)
    - The lower tail probability, i.e., \( p_i = P(X_i \leq x_{p_i} : \nu_i). \)
  \item \( \text{tail}[j] = \text{Nag_UpperTail} \)
    - The upper tail probability, i.e., \( p_i = P(X_i \geq x_{p_i} : \nu_i). \)
\end{itemize}

\textit{Constraint:} \( \text{tail}[j] = \text{Nag_LowerTail} \) or \( \text{Nag_UpperTail}, \) for \( j = 1, 2, \ldots, \text{ltail}. \)

3: \( \text{lp} \) – \texttt{Integer} \quad \textit{Input}

\textit{On entry:} the length of the array \( \text{p}. \)

\textit{Constraint:} \( \text{lp} > 0. \)

4: \( \text{p[lp]} \) – \texttt{const double} \quad \textit{Input}

\textit{On entry:} \( p_i, \) the probability of the required \( \chi^2\)-distribution as defined by \( \text{tail} \) with \( p_i = \text{p}[j], \) \( j = (i - 1) \text{ mod } \text{lp}. \)

\textit{Constraints:}

\begin{itemize}
  \item if \( \text{tail}[k] = \text{Nag_LowerTail}, \) \( 0.0 \leq \text{p}[j] < 1.0; \)
  \item otherwise \( 0.0 < \text{p}[j] \leq 1.0. \)
\end{itemize}

Where \( k = (i - 1) \text{ mod } \text{ltail} \) and \( j = (i - 1) \text{ mod } \text{lp}. \)

5: \( \text{ldf} \) – \texttt{Integer} \quad \textit{Input}

\textit{On entry:} the length of the array \( \text{df}. \)

\textit{Constraint:} \( \text{ldf} > 0. \)

6: \( \text{df[ldf]} \) – \texttt{const double} \quad \textit{Input}

\textit{On entry:} \( \nu_i, \) the degrees of freedom of the \( \chi^2\)-distribution with \( \nu_i = \text{df}[j], \) \( j = (i - 1) \text{ mod } \text{ldf}. \)

\textit{Constraint:} \( \text{df}[j] = 0.0, \) for \( j = 1, 2, \ldots, \text{ldf}. \)

7: \( \text{x[dim]} \) – \texttt{double} \quad \textit{Output}

\textit{Note:} the dimension, \( \text{dim}, \) of the array \( \text{x} \) must be at least \( \text{max(ltail, lp, ldf)}. \)

\textit{On exit:} \( x_{p_i}, \) the deviates for the \( \chi^2\)-distribution.

8: \( \text{ivalid[dim]} \) – \texttt{Integer} \quad \textit{Output}

\textit{Note:} the dimension, \( \text{dim}, \) of the array \( \text{ivalid} \) must be at least \( \text{max(ltail, lp, ldf)}. \)

\textit{On exit:} \( \text{ivalid}[i - 1] \) indicates any errors with the input arguments, with

\begin{itemize}
  \item \( \text{ivalid}[i - 1] = 0 \)
    - No error.
  \item \( \text{ivalid}[i - 1] = 1 \)
    - On entry, invalid value supplied in \( \text{tail} \) when calculating \( x_{p_i}. \)
  \item \( \text{ivalid}[i - 1] = 2 \)
    - On entry, invalid value for \( p_i. \)
  \item \( \text{ivalid}[i - 1] = 3 \)
    - On entry, \( \nu_i \leq 0.0. \)
ivalid[i - 1] = 4
  \( p_i \) is too close to 0.0 or 1.0 for the result to be calculated.

ivalid[i - 1] = 5
  The solution has failed to converge. The result should be a reasonable approximation.

9:  fail – NagError *
    The NAG error argument (see Section 3.6 in the Essential Introduction).

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_ARRAY_SIZE
  On entry, array size = \( \text{value} \).
  Constraint: \( \text{ldf} > 0 \).
  On entry, array size = \( \text{value} \).
  Constraint: \( \text{lp} > 0 \).
  On entry, array size = \( \text{value} \).
  Constraint: \( \text{ltail} > 0 \).

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

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.

NW_IVALID
  On entry, at least one value of tail, p or df was invalid, or the solution failed to converge.
  Check ivalid for more information.

7 Accuracy

The results should be accurate to five significant digits for most argument values. Some accuracy is lost for \( p_i \) close to 0.0 or 1.0.

8 Parallelism and Performance

Not applicable.

9 Further Comments

For higher accuracy the relationship described in Section 3 may be used and a direct call to nag_deviates_gamma_vector (g01tfc) made.
10 Example

This example reads lower tail probabilities for several \( \chi^2 \)-distributions, and calculates and prints the corresponding deviates.

10.1 Program Text

```c
/* nag_deviates_chi_sq_vector (g01tcc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 23, 2011.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
    Integer ltail, lp, ldf, i, lout;
    Integer *ivalid = 0;
    Integer exit_status = 0;

    /* NAG structures */
    NagError fail;
    Nag_TailProbability *tail = 0;

    /* Double scalar and array declarations */
    double *p = 0, *df = 0, *x = 0;

    /* Character scalar and array declarations */
    char ctail[40];

    /* Initialise the error structure to print out any error messages */
    INIT_FAIL(fail);

    printf("nag_deviates_chi_sq_vector (g01tcc) Example Program Results\n\n");

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

    /* Read in the input vectors */
    #ifdef _WIN32
        scanf_s("%NAG_IFMT"%f[\n] ", &ltail);
    #else
        scanf("%NAG_IFMT"%f[\n] ", &ltail);
    #endif
    if (!NAG_ALLOC(ltail, Nag_TailProbability)) {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (i = 0; i < ltail; i++) {
        scanf("%39s", ctail, _countof(ctail));
        tail[i] = (Nag_TailProbability) nag_enum_name_to_value(ctail);
    }
    printf("\n");
}
```


scanf("%*[\n] ");
#endif

#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n] ", &lp);
#else
    scanf("%"NAG_IFMT"%*[\n] ", &lp);
#endif
if (!(p = NAG_ALLOC(lp, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
for (i = 0; i < lp; i++)
#ifdef _WIN32
    scanf_s("%lf", &p[i]);
#else
    scanf("%lf", &p[i]);
#endif
#endif _WIN32
scanf_s("%*[\n] ");
#endif
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n] ", &ldf);
#else
    scanf("%"NAG_IFMT"%*[\n] ", &ldf);
#endif
if (!(df = NAG_ALLOC(ldf, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
for (i = 0; i < ldf; i++)
#ifdef _WIN32
    scanf_s("%lf", &df[i]);
#else
    scanf("%lf", &df[i]);
#endif
#endif _WIN32
scanf_s("%*[\n] ");
#endif

/* Allocate memory for output */
lout = MAX(ltail,MAX(lp,ldf));
if (! (x = NAG_ALLOC(lout, double)) ||
    !(ivalid = NAG_ALLOC(lout, Integer))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Calculate probability */
nag_deviates_chi_sq_vector(ltail, tail, lp, p, ldf, df, x, ivalid, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_deviates_chi_sq_vector (g01tcc)\n", fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID) goto END;
}

/* Display title */
printf(" tail p df x ivalid\n");
printf(" ------------------------------------------------------\n");
/* Display results */
for (i = 0; i < lout; i++)
    printf(" %15s %6.3f %6.1f %7.4f %3"NAG_IFMT"
           nag_enum_value_to_name(tail[i%ltail]), p[i%lp], df[i%ldf],
           x[i], ivalid[i]);

END:
NAG_FREE(tail);
NAG_FREE(p);
NAG_FREE(df);
NAG_FREE(x);
NAG_FREE(ivalid);

return(exit_status);
}

10.2 Program Data

nag_deviates_chi_sq_vector (g01tcc) Example Program Data

1 :: ltail
Nag_LowerTail :: tail
3 :: lp
0.010 0.428 0.869 :: p
3 :: ldf
20.0 7.5 45.0 :: df

10.3 Program Results

nag_deviates_chi_sq_vector (g01tcc) Example Program Results

<table>
<thead>
<tr>
<th>tail</th>
<th>p</th>
<th>df</th>
<th>x</th>
<th>ivalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nag_LowerTail</td>
<td>0.010</td>
<td>20.0</td>
<td>8.2604</td>
<td>0</td>
</tr>
<tr>
<td>Nag_LowerTail</td>
<td>0.428</td>
<td>7.5</td>
<td>6.2006</td>
<td>0</td>
</tr>
<tr>
<td>Nag_LowerTail</td>
<td>0.869</td>
<td>45.0</td>
<td>55.7381</td>
<td>0</td>
</tr>
</tbody>
</table>