NAG C Library Function Document

nag_kernel_density_estim (g10bac)

1 Purpose
nag_kernel_density_estim (g10bac) performs kernel density estimation using a Gaussian kernel.

2 Specification
#include <nag.h>
#include <nagl0.h>

void nag_kernel_density_estim(Integer n, const double x[], double window,
   double low, double high, Integer ns, double smooth[], double t[],
   NagError *fail)

3 Description
Given a sample of \( n \) observations, \( x_1, x_2, \ldots, x_n \), from a distribution with unknown density function, \( f(x) \), an estimate of the density function, \( \hat{f}(x) \), may be required. The simplest form of density estimator is the histogram. This may be defined by:

\[
\hat{f}(x) = \frac{1}{nh} n_j; \quad a + (j - 1)h < x < a +jh, \quad j = 1, 2, \ldots, n_s,
\]

where \( n_j \) is the number of observations falling in the interval \( a + (j - 1)h \) to \( a + jh \), \( a \) is the lower bound to the histogram and \( b = n_s h \) is the upper bound. The value \( h \) is known as the window width. To produce a smoother density estimate a kernel method can be used. A kernel function, \( K(t) \), satisfies the conditions:

\[
\int_{-\infty}^{\infty} K(t) dt = 1 \quad \text{and} \quad K(t) \geq 0.
\]

The kernel density estimator is then defined as:

\[
\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K \left( \frac{x - x_i}{h} \right).
\]

The choice of \( K \) is usually not important but to ease the computational burden use can be made of the Gaussian kernel defined as:

\[
K(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}.
\]

The smoothness of the estimator depends on the window width \( h \). The larger the value of \( h \) the smoother the density estimate. The value of \( h \) can be chosen by examining plots of the smoothed density for different values of \( h \) or by using cross-validation methods (Silverman (1990)).

Silverman (1982) and Silverman (1990) show how the Gaussian kernel density estimator can be computed using a fast Fourier transform (FFT). In order to compute the kernel density estimate over the range \( a \) to \( b \) the following steps are required:

1. discretize the data to give \( n_s \) equally spaced points \( t_l \) with weights \( \xi_l \) (see Jones and Lotwick (1984));
2. compute the FFT of the weights \( \xi_l \) to give \( Y_l \);
3. compute \( \xi_l = e^{-\frac{1}{2}h^2 s_l^2} Y_l \) where \( s_l = 2\pi l/(b - a) \);
4. find the inverse FFT of \( \xi_l \) to give \( \hat{f}(x) \).
4 Parameters

1: \( n \) – Integer \hspace{1cm} \text{Input}
   
   \text{On entry: the number of observations in the sample, } n. 
   
   \text{Constraint: } n > 0.

2: \( x[n] \) – const double \hspace{1cm} \text{Input}
   
   \text{On entry: the } n \text{ observations, } x_i, \text{ for } i = 1, 2, \ldots, n.

3: \text{window} – double \hspace{1cm} \text{Input}
   
   \text{On entry: the window width, } h. 
   
   \text{Constraint: window[ ] > 0.0.}

4: \text{low} – double \hspace{1cm} \text{Input}
   
   \text{On entry: the lower limit of the interval on which the estimate is calculated, } a. \text{ For most applications low[ ] should be at least three window widths below the lowest data point.}
   
   \text{Constraint: low[ ] < high[ ].}

5: \text{high} – double \hspace{1cm} \text{Input}
   
   \text{On entry: the upper limit of the interval on which the estimate is calculated, } b. \text{ For most applications high[ ] should be at least three window widths above the highest data point.}

6: \text{ns} – Integer \hspace{1cm} \text{Input}
   
   \text{On entry: the number of points at which the estimate is calculated, } n_s.
   
   \text{Constraints: } \text{ns[ ]} \geq 2.
   
   \text{The largest prime factor of } \text{ns[ ]} \text{ must not exceed 19, and the total number of prime factors of } \text{ns[ ]}, \text{ counting repetitions, must not exceed 20.}

7: \text{smooth[ns]} – double \hspace{1cm} \text{Output}
   
   \text{On exit: the } n_s \text{ values of the density estimate, } \hat{f}(t_l), \text{ for } l = 1, 2, \ldots, n_s.

8: \text{t[ns]} – double \hspace{1cm} \text{Output}
   
   \text{On exit: the points at which the estimate is calculated, } t_l, \text{ for } l = 1, 2, \ldots, n_s.

9: \text{fail} – NagError * \hspace{1cm} \text{Input/Output}
   
   \text{The NAG error parameter (see the Essential Introduction).}

5 Error Indicators and Warnings

\textbf{NE_INT_ARG_LE}

\text{On entry, } n[ ] \text{ must not be less than or equal to 0: } n[ ] = \text{<value>.}

\textbf{NE_INT_ARG_LT}

\text{On entry, } \text{ns[ ]} \text{ must not be less than 2: } n_s[ ] = \text{<value>.}

\textbf{NE_REAL_ARG_LE}

\text{On entry, } \text{window[ ]} \text{ must not be less than or equal to 0.0: } \text{window[ ]} = \text{<value>.}
NE_2_REAL_ARG_LE
On entry, high[] = <value> while low[] = <value>.
These parameters must satisfy high[] > low[].

NE_C06_FACTORS
At least one of the prime factors of ns[] is greater than 19 or ns[] has more than 20 prime factors.

NE_G10BA_INTERVAL
On entry, the interval given by low[] to high[] does not extend beyond three window[] widths at
either extreme of the data set. This may distort the density estimate in some cases.

NE_ALLOC_FAIL
Memory allocation failed.

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 consult NAG for assistance.

6 Further Comments
The time for computing the weights of the discretized data is of order \( n \) while the time for computing the
FFT is of order \( n \log(n) \) as is the time for computing the inverse of the FFT.

6.1 Accuracy
See Jones and Lotwick (1984) for a discussion of the accuracy of this method.

6.2 References
120–122
Statist. 31 93–99

7 See Also
None.

8 Example
A sample of 1000 standard Normal (0,1) variates are generated using nag_random_normal (g05ddc) and
the density estimated on 100 points with a window width of 0.1.
8.1 Program Text

/* nag_kernel_density_estim (g10bac) Example Program. */
* Copyright 2000 Numerical Algorithms Group.
* Mark 6, 2000. */

#include <stdio.h>
#include <nag.h>
#include <nag.stdlib.h>
#include <nag01.h>
#include <nag05.h>
#include <nag10.h>

int main(void)
{
    Integer i, init, increment, j, n, ns;
    Integer exit_status=0;
    double enda, endb, *s=0, high, low, *smooth=0, window, *x=0;
    Integer ifail, *isort=0;
    Boolean usefft;
    NagError fail;

    INIT_FAIL(fail);
    Vprintf("g10bac Example Program Results\n");

    /* Skip heading in data file */
    Vscanf("%*[\n "");

    Vscanf("%lf ", &window);
    Vscanf("%lf , %lf", &low, &high);

    /* Generate Normal (0,1) Distribution */
    n = 1000;
    ns = 100;
    if (!((x = NAG_ALLOC(n, double))
        || !s = NAG_ALLOC(ns, double))
        || smooth = NAG_ALLOC(ns, double))
        || !isort = NAG_ALLOC(ns, Integer))
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    init = 0;
g05cbc(init);
enda = 0.0;
endb = 1.0;
for (i = 0; i < n; i++)
    x[i] = g05ddc(enda, endb);

    /* Perform kernel density estimation */
    usefft = FALSE;
    ifail = 0;
g10bac(n, x, window, low, high, ns, smooth, s, &fail);
    if (fail.code != NE_NOERROR)
8.2 Program Data

g10bac Example Program Data

0.1
-4.0, 4.0

8.3 Program Results

g10bac Example Program Results

<table>
<thead>
<tr>
<th>Points</th>
<th>Density</th>
<th>Points</th>
<th>Value</th>
<th>Density</th>
<th>Points</th>
<th>Value</th>
<th>Density</th>
<th>Points</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.9600</td>
<td>0.0000</td>
<td>-1.9600</td>
<td>0.0508</td>
<td>0.0400</td>
<td>0.3698</td>
<td>2.0400</td>
<td>0.0464</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.8800</td>
<td>0.0011</td>
<td>-1.8800</td>
<td>0.0573</td>
<td>0.1200</td>
<td>0.3614</td>
<td>2.1200</td>
<td>0.0361</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.8000</td>
<td>0.0011</td>
<td>-1.8000</td>
<td>0.0763</td>
<td>0.2000</td>
<td>0.3393</td>
<td>2.2000</td>
<td>0.0344</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.7200</td>
<td>0.0037</td>
<td>-1.7200</td>
<td>0.0763</td>
<td>0.2800</td>
<td>0.3346</td>
<td>2.2800</td>
<td>0.0307</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.6400</td>
<td>0.0049</td>
<td>-1.6400</td>
<td>0.0719</td>
<td>0.3600</td>
<td>0.3618</td>
<td>2.3600</td>
<td>0.0207</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.5600</td>
<td>0.0023</td>
<td>-1.5600</td>
<td>0.0942</td>
<td>0.4400</td>
<td>0.3553</td>
<td>2.4400</td>
<td>0.0096</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.4800</td>
<td>0.0003</td>
<td>-1.4800</td>
<td>0.1292</td>
<td>0.5200</td>
<td>0.3312</td>
<td>2.5200</td>
<td>0.0071</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.4000</td>
<td>0.0003</td>
<td>-1.4000</td>
<td>0.1440</td>
<td>0.6000</td>
<td>0.3356</td>
<td>2.6000</td>
<td>0.0133</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.3200</td>
<td>0.0003</td>
<td>-1.3200</td>
<td>0.1659</td>
<td>0.6800</td>
<td>0.3496</td>
<td>2.6800</td>
<td>0.0162</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.2400</td>
<td>0.0021</td>
<td>-1.2400</td>
<td>0.2181</td>
<td>0.7600</td>
<td>0.3310</td>
<td>2.7600</td>
<td>0.0117</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.1600</td>
<td>0.0047</td>
<td>-1.1600</td>
<td>0.2511</td>
<td>0.8400</td>
<td>0.2922</td>
<td>2.8400</td>
<td>0.0074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.0800</td>
<td>0.0039</td>
<td>-1.0800</td>
<td>0.2443</td>
<td>0.9200</td>
<td>0.2812</td>
<td>2.9200</td>
<td>0.0077</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.0000</td>
<td>0.0015</td>
<td>-1.0000</td>
<td>0.2443</td>
<td>1.0000</td>
<td>0.3011</td>
<td>3.0000</td>
<td>0.0073</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.9200</td>
<td>0.0012</td>
<td>-0.9200</td>
<td>0.2415</td>
<td>1.0780</td>
<td>0.2872</td>
<td>3.0800</td>
<td>0.0040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.8400</td>
<td>0.0038</td>
<td>-0.8400</td>
<td>0.2565</td>
<td>1.1600</td>
<td>0.2134</td>
<td>3.1600</td>
<td>0.0011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.7600</td>
<td>0.0062</td>
<td>-0.7600</td>
<td>0.2970</td>
<td>1.2400</td>
<td>0.1577</td>
<td>3.2400</td>
<td>0.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.6800</td>
<td>0.0115</td>
<td>-0.6800</td>
<td>0.3435</td>
<td>1.3200</td>
<td>0.1395</td>
<td>3.3200</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.6000</td>
<td>0.0218</td>
<td>-0.6000</td>
<td>0.3642</td>
<td>1.4000</td>
<td>0.1370</td>
<td>3.4000</td>
<td>0.0004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.5200</td>
<td>0.0231</td>
<td>-0.5200</td>
<td>0.3822</td>
<td>1.4800</td>
<td>0.1315</td>
<td>3.4800</td>
<td>0.0029</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.4400</td>
<td>0.0191</td>
<td>-0.4400</td>
<td>0.4081</td>
<td>1.5600</td>
<td>0.1295</td>
<td>3.5600</td>
<td>0.0055</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.3600</td>
<td>0.0230</td>
<td>-0.3600</td>
<td>0.4051</td>
<td>1.6400</td>
<td>0.1270</td>
<td>3.6400</td>
<td>0.0031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.2800</td>
<td>0.0297</td>
<td>-0.2800</td>
<td>0.3843</td>
<td>1.7200</td>
<td>0.1109</td>
<td>3.7200</td>
<td>0.0006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.2000</td>
<td>0.0316</td>
<td>-0.2000</td>
<td>0.3447</td>
<td>1.8000</td>
<td>0.0947</td>
<td>3.8000</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.1200</td>
<td>0.0417</td>
<td>-0.1200</td>
<td>0.3214</td>
<td>1.8800</td>
<td>0.0847</td>
<td>3.8800</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.0400</td>
<td>0.0536</td>
<td>-0.0400</td>
<td>0.3474</td>
<td>1.9600</td>
<td>0.0655</td>
<td>3.9600</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>