NAG Library Function Document

nag_kernel_density_gauss (g10bbc)

1 Purpose

nag_kernel_density_gauss (g10bbc) performs kernel density estimation using a Gaussian kernel.

2 Specification

```
#include <nag.h>
#include <nagg10.h>
void nag_kernel_density_gauss (Integer n, const double x[],
     Nag_WindowType wtype, double *window, double *slo, double *shi,
     Integer ns, double smooth[], double t[], Nag_Boolean fcall,
     double rcomm[], 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, $b = n_s h$ is the upper bound and n_s is the total number of intervals. 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 (see 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.

(i) Discretize the data to give n_s equally spaced points t_l with weights ξ_l (see Jones and Lotwick (1984)).

(ii) Compute the FFT of the weights ξ_l to give Y_l.

(iii) Compute $\zeta_l = e^{-\frac{1}{2}h^2 s_l^2} Y_l$ where $s_l = 2\pi l/(b-a)$.

(iv) Find the inverse FFT of ζ_l to give $\hat{f}(x)$.

To compute the kernel density estimate for further values of h only steps (iii) and (iv) need be repeated.

4 References

Jones M C and Lotwick H W (1984) Remark AS R50. A remark on algorithm AS 176. Kernel density estimation using the Fast Fourier Transform *Appl. Statist.* **33** 120–122

Silverman B W (1982) Algorithm AS 176. Kernel density estimation using the fast Fourier transform *Appl. Statist.* **31** 93–99

Silverman B W (1990) *Density Estimation* Chapman and Hall

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of observations in the sample.

If **fcall** = Nag_FALSE, **n** must be unchanged since the last call to nag_kernel_density_gauss (g10bbc).

Constraint: **n** > 0.

2: **x**[**n**] – const double *Input*

On entry: x_i, for $i = 1, 2, \ldots, n$.

If **fcall** = Nag_FALSE, **x** must be unchanged since the last call to nag_kernel_density_gauss (g10bbc).

3: **wtype** – Nag_WindowType *Input*

On entry: how the window width, h, is to be calculated:

wtype = Nag_WindowSupplied
h is supplied in **window**.

wtype = Nag_RuleOfThumb
h is to be calculated from the data, with

$$h = m \times \left(\frac{0.9 \times \min(q_{75} - q_{25}, \sigma)}{n^{0.2}} \right)$$

where $q_{75} - q_{25}$ is the inter-quartile range and σ the standard deviation of the sample, x, and m is a multipler supplied in **window**. The 25% and 75% quartiles, q_{25} and q_{75}, are calculated using nag_double_quantiles (g01amc). This is the "rule-of-thumb" suggested by Silverman (1990).

Suggested value: **wtype** = Nag_RuleOfThumb and **window** = 1.0.

Constraint: **wtype** = Nag_WindowSupplied or Nag_RuleOfThumb.

4: **window** – double * *Input/Output*

On entry: if **wtype** = Nag_WindowSupplied, then h, the window width. Otherwise, m, the multiplier used in the calculation of h.

Suggested value: **window** = 1.0 and **wtype** = Nag_RuleOfThumb.

On exit: h, the window width actually used.

Constraint: **window** > 0.0.

5: **slo** – double * *Input/Output*

On entry: if **slo** $<$ **shi** then a, the lower limit of the interval on which the estimate is calculated. Otherwise, a and b, the lower and upper limits of the interval, are calculated as follows:

$$\begin{aligned} a &= \min_i\{x_i\} - \textbf{slo} \times h \\ b &= \max_i\{x_i\} + \textbf{slo} \times h \end{aligned}$$

where h is the window width.

For most applications a should be at least three window widths below the lowest data point.

If **fcall** = Nag_FALSE, **slo** must be unchanged since the last call to nag_kernel_density_gauss (g10bbc).

Suggested value: **slo** = 3.0 and **shi** = 0.0 which would cause a and b to be set 3 window widths below and above the lowest and highest data points respectively.

On exit: a, the lower limit actually used.

6: **shi** – double * *Input/Output*

On entry: if **slo** < **shi** then b, the upper limit of the interval on which the estimate is calculated. Otherwise a value for b is calculated from the data as stated in the description of **slo** and the value supplied in **shi** is not used.

For most applications b should be at least three window widths above the highest data point.

If **fcall** = Nag_FALSE, **shi** must be unchanged since the last call to nag_kernel_density_gauss (g10bbc).

On exit: b, the upper limit actually used.

7: **ns** – Integer *Input*

On entry: n_s, the number of points at which the estimate is calculated.

If **fcall** = Nag_FALSE, **ns** must be unchanged since the last call to nag_kernel_density_gauss (g10bbc).

Suggested value: **ns** = 512.

Constraint: **ns** ≥ 2.

8: **smooth**[**ns**] – double *Output*

On exit: $\hat{f}(t_l)$, for $l = 1, 2, \ldots, n_s$, the n_s values of the density estimate.

9: **t**[**ns**] – double *Output*

On exit: t_l, for $l = 1, 2, \ldots, n_s$, the points at which the estimate is calculated.

10: **fcall** – Nag_Boolean *Input*

On entry: if **fcall** = Nag_TRUE then the values of Y_l are to be calculated by this call to nag_kernel_density_gauss (g10bbc), otherwise it is assumed that the values of Y_l were calculated by a previous call to this routine and the relevant information is stored in **rcomm**.

11: **rcomm**[**ns** + **20**] – double *Communication Array*

On entry: communication array, used to store information between calls to nag_kernel_density_gauss (g10bbc).

If **fcall** = Nag_FALSE, **rcomm** must be unchanged since the last call to nag_kernel_density_gauss (g10bbc).

On exit: the last **ns** elements of **rcomm** contain the fast Fourier transform of the weights of the discretized data, that is **rcomm**$[l + 19] = Y_l$, for $l = 1, 2, \ldots, n_s$.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_ILLEGAL_COMM

rcomm has been corrupted between calls.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** > 0.

On entry, **ns** = ⟨*value*⟩.
Constraint: **ns** ≥ 2.

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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_PREV_CALL

On entry, **n** = ⟨*value*⟩.
On entry at previous call, **n** = ⟨*value*⟩.
Constraint: if **fcall** = Nag_FALSE, **n** must be unchanged since previous call.

On entry, **ns** = ⟨*value*⟩.
On entry at previous call, **ns** = ⟨*value*⟩.
Constraint: if **fcall** = Nag_FALSE, **ns** must be unchanged since previous call.

On entry, **shi** = ⟨*value*⟩.
On exit from previous call, **shi** = ⟨*value*⟩.
Constraint: if **fcall** = Nag_FALSE, **shi** must be unchanged since previous call.

On entry, **slo** = ⟨*value*⟩.
On exit from previous call, **slo** = ⟨*value*⟩.
Constraint: if **fcall** = Nag_FALSE, **slo** must be unchanged since previous call.

NE_REAL

On entry, **window** = ⟨*value*⟩.
Constraint: **window** > 0.0.

NW_POTENTIAL_PROBLEM

On entry, **slo** = ⟨*value*⟩ and **shi** = ⟨*value*⟩.
On entry, min(**x**) = ⟨*value*⟩ and max(**x**) = ⟨*value*⟩.
Expected values of at least ⟨*value*⟩ and ⟨*value*⟩ for **slo** and **shi**.
All output values have been returned.

7 Accuracy

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

8 Parallelism and Performance

nag_kernel_density_gauss (g10bbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_kernel_density_gauss (g10bbc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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 time for computing the weights of the discretized data is of order n, while the time for computing the FFT is of order $n_s \log(n_s)$, as is the time for computing the inverse of the FFT.

10 Example

Data is read from a file and the density estimated. The first 20 values are then printed.

10.1 Program Text

```
/* nag_kernel_density_gauss (g10bbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>
#include <nagg10.h>

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

  /* Nag Types */
  NagError fail;
  Nag_Boolean fcall;
  Nag_WindowType wtype;

  /* Double scalar and array declarations */
  double shi, slo, window;
  double *rcomm = 0, *smooth = 0, *t = 0, *x = 0;
```

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

  /* Initialize the error structure */
  INIT_FAIL(fail);

  printf("nag_kernel_density_gauss (g10bbc) Example Program Results\n\n");

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

  /* Read in density estimation information */
#ifdef _WIN32
  scanf_s("%39s %lf %lf %lf %" NAG_IFMT "%*[^\n] ", cwtype,
          (unsigned)_countof(cwtype), &window, &slo, &shi, &ns);
#else
  scanf("%39s %lf %lf %lf %" NAG_IFMT "%*[^\n] ", cwtype, &window, &slo, &shi,
        &ns);
#endif
  wtype = (Nag_WindowType) nag_enum_name_to_value(cwtype);

  /* Read in the size of the dataset */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &n);
#endif

  if (!(smooth = NAG_ALLOC(ns, double)) ||
      !(t = NAG_ALLOC(ns, double)) ||
      !(rcomm = NAG_ALLOC(ns + 20, double)) || !(x = NAG_ALLOC(n, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Only calling the routine once */
  fcall = Nag_TRUE;

  /* Read in data */
  for (i = 0; i < n; i++) {
#ifdef _WIN32
    scanf_s("%lf", &x[i]);
#else
    scanf("%lf", &x[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Call nag_kernel_density_gauss (g10bbc) to perform kernel
   * density estimation
   */
  nag_kernel_density_gauss(n, x, wtype, &window, &slo, &shi, ns, smooth, t,
                           fcall, rcomm, &fail);
  if (fail.code != NE_NOERROR && fail.code != NW_POTENTIAL_PROBLEM) {
    printf("Error from nag_kernel_density_gauss (g10bbc).\n%s\n",
           fail.message);
    exit_status = -1;
    goto END;
  }
```

```
  /* Display the summary of results */
  printf("Window Width Used = %13.4e\n", window);
  printf("Interval = (%13.4e,%13.4e)\n", slo, shi);
  printf("\n");
  printf("First %" NAG_IFMT " output values:\n", MIN(ns, 20));
  printf("\n");
  printf("      Time        Density\n");
  printf("      Point       Estimate\n");
  printf(" -------------------------\n");
  for (i = 0; i < MIN(20, ns); i++)
    printf(" %13.3e %13.3e\n", t[i], smooth[i]);

END:
  NAG_FREE(smooth);
  NAG_FREE(t);
  NAG_FREE(rcomm);
  NAG_FREE(x);

  return exit_status;
}
```

10.2 Program Data

```
nag_kernel_density_gauss (g10bbc) Example Program Data
Nag_RuleOfThumb  1.0  3.0  0.0  512 :: wtype,window,slo,shi,ns
100                                  :: n
 0.114 -0.232 -0.570  1.853 -0.994
-0.374 -1.028  0.509  0.881 -0.453
 0.588 -0.625 -1.622 -0.567  0.421
-0.475  0.054  0.817  1.015  0.608
-1.353 -0.912 -1.136  1.067  0.121
-0.075 -0.745  1.217 -1.058 -0.894
 1.026 -0.967 -1.065  0.513  0.969
 0.582 -0.985  0.097  0.416 -0.514
 0.898 -0.154  0.617 -0.436 -1.212
-1.571  0.210 -1.101  1.018 -1.702
-2.230 -0.648 -0.350  0.446 -2.667
 0.094 -0.380 -2.852 -0.888 -1.481
-0.359 -0.554  1.531  0.052 -1.715
 1.255 -0.540  0.362 -0.654 -0.272
-1.810  0.269 -1.918  0.001  1.240
-0.368 -0.647 -2.282  0.498  0.001
-3.059 -1.171  0.566  0.948  0.925
 0.825  0.130  0.930  0.523  0.443
-0.649  0.554 -2.823  0.158 -1.180
 0.610  0.877  0.791 -0.078  1.412   :: End of x
```

10.3 Program Results

```
nag_kernel_density_gauss (g10bbc) Example Program Results

Window Width Used =    3.7638e-01
Interval = (  -4.1882e+00,   2.9822e+00)

First 20 output values:

      Time        Density
      Point       Estimate
 -------------------------
    -4.181e+00     3.828e-06
    -4.167e+00     4.031e-06
    -4.153e+00     4.423e-06
    -4.139e+00     5.021e-06
    -4.125e+00     5.846e-06
    -4.111e+00     6.928e-06
    -4.097e+00     8.305e-06
    -4.083e+00     1.002e-05
    -4.069e+00     1.215e-05
    -4.055e+00     1.474e-05
    -4.041e+00     1.788e-05
```

```
-4.027e+00     2.168e-05
-4.013e+00     2.624e-05
-3.999e+00     3.170e-05
-3.985e+00     3.821e-05
-3.971e+00     4.596e-05
-3.957e+00     5.514e-05
-3.943e+00     6.599e-05
-3.929e+00     7.877e-05
-3.915e+00     9.380e-05
```

This plot shows the estimated density function for the example data for several window widths.