Nearest Neighbours: nagdmc_kdtree

Purpose

nagdmc_kdtree computes a k-d tree for use with the nearest neighbour functions **nagdmc_knnc** and **nagdmc_knnp**.

Declaration

```
#include <nagdmc.h>
void nagdmc_kdtree(long rec1, long nvar, long nrec, long dblk, double data[],
                   long nxvar, long xvar[], long yvar, long ng, long nig[],
                   long *iproot, int *info);
```

Parameters

1: **rec1** – `long` *Input*

On entry: the index in the data of the first data record used in the analysis.

Constraint: **rec1** ≥ 0.

2: **nvar** – `long` *Input*

On entry: the number of variables in the data.

Constraint: **nvar** > 1.

3: **nrec** – `long` *Input*

On entry: the number of consecutive records, beginning at **rec1**, used in the analysis.

Constraint: **nrec** > 1.

4: **dblk** – `long` *Input*

On entry: the total number of records in the data block.

Constraint: **dblk** $\geq$ **rec1** $+$ **nrec**.

5: **data**[**dblk** $*$ **nvar**] – `double` *Input*

On entry: the data values for the jth variable (for $j = 0, 1, \ldots,$ **nvar**-1) are stored in **data**[$i*$**nvar**$+j$], for $i = 0, 1, \ldots,$ **dblk** $- 1$.

6: **nxvar** – `long` *Input*

On entry: the number of independent variables. If **nxvar** $= 0$ then all variables in the data, excluding **yvar**, are treated as independent variables.

Constraint: $0 \leq$ **nxvar** $<$ **nvar**.

7: **xvar**[**nxvar**] – `long` *Input*

On entry: the indices indicating the position in **data** in which values of the independent variables are stored. If **nxvar** $= 0$ then **xvar** must be 0, and the indices of independent variables are given by $j = 0, 1, \ldots,$ **nvar** $- 1$; $j \neq$ **yvar**.

Constraints: if **nxvar** > 0, $0 \leq$ **xvar**$[i] <$ **nvar**, for $i = 0, 1, \ldots,$ **nxvar** $- 1$; otherwise **xvar** must be 0.

8: **yvar** – `long` *Input*

On entry: the index in **data** in which values of the dependent variable are stored.

Constraints: $0 \leq$ **yvar** $<$ **nvar**; if **nxvar** > 0, **yvar** $\neq$ **xvar**$[i]$, for $i = 0, 1, \ldots,$ **nxvar** $- 1$.

9: **ng** – `long` *Input*

On entry: the number of groups in the dependent variable. If the dependent variable is continuous, set **ng** equal to -1.

Constraint: if **ng** $\neq -1$, **ng** > 1.

10: **nig[ng]** – `long` *Input*

On entry: if **ng** $\neq -1$, the number of data records in each group on the dependent variable; otherwise **nig** must be 0.

Constraints: **nig**$[i] \geq 1$, for $i = 0, 1, \ldots,$ **ng** $- 1$; and the elements of **nig** must sum equal to **nrec**.

11: **iproot** – `long *` *Output*

On exit: an integer cast of the root of the k-d tree.

12: **info** – `int *` *Output*

On exit: **info** gives information on the success of the function call:

0: the function successfully completed its task.

i; $i = 1, 2, 3, 4, 6, 7, \ldots, 10$: the specification of the ith formal parameter was incorrect.

99: the function failed to allocate enough memory.

100: an internal error occurred during the execution of the function.

Notation

nrec the number of data records, n.
data the set of data records, X.
yvar the dependent variable in the data, y.

Description

A k-d tree is a lattice that represents k-dimensional data values in a binary tree (see Figure 1).

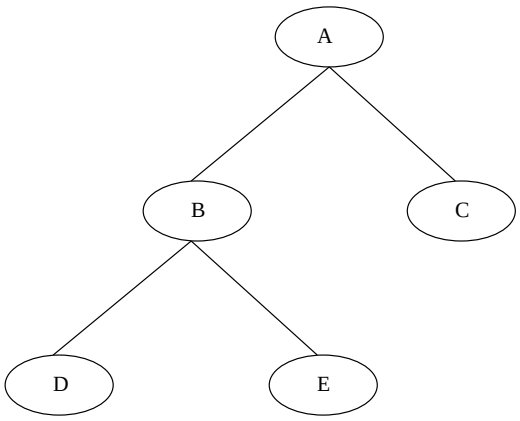

Figure 1: *Example of a binary tree lattice. Ellipses are used to represent nodes in the tree and parent nodes are linked by line segments to their child nodes in the lattice. Nodes without children are known as leaf nodes (nodes C, D and E in the figure). The single node without a parent node is (node A in the figure) is called the root node.*

Let X be a set of n data records x_i for $i = 1, 2, \ldots, n$ that we wish to represent in a binary tree lattice. Each node in the lattice is associated with a set of data records. In general, let Z_i be the set of data records associated with node k. If node k is not a leaf node it is associated with a test that partitions Z_k, i.e., sends each member of Z_k to either its left or right child node; otherwise it stores values of the dependent variable, y, in Z_k.

The process of calculating a k-d tree is initiated by associating a root node with the n data reocrds in X and is halted when Z_k cannot be partitioned, i.e., $|Z_k| = 1$ or all members of Z_k are the same.

The test used to partition data records at node k is the condition:

$$x_{ij} \leq u,$$

where x_{ij} is the value of variable j for a data record belonging to Z_k; u is the median of variable j in Z_k; and j is the variable with the highest range for data records belonging to Z_k.

References and Further Reading

Bentley J L (1975) Multi-dimensional binary search trees used for associative searching *Communications of the ACM* **18**(9) 509–517.

Storer J A and Cohn M (1993) Algorithms for fast vector quantization *Proc. Data Compression Conference* 381–390 IEEE Computer Society Press.

See Also

nagdmc_knnc.pdf	calculates nearest neighbour classifications.
nagdmc_knnp.pdf	calculates nearest neighbour approximations.
knnc_ex.c	the example calling program for a categorical dependent variable.
knnp_ex.c	the example calling program for a continuous dependent variable.