NAG DMC

User Guide

1 Introduction

This document provides a guide to the functionality available in NAG DMC, and, for convenience and clarity, is divided into sections based on a need to accomplish common tasks (as listed in Section 2). A brief summary of the contents of this document now follows.

Section 3.1 describes methods for data imputation, and Section 3.2 describes a method to find outliers in continuous data values. Section 4 covers methods used to transform data values, and Section 5 describes methods used for cluster analysis. Classification and regression models are described in Section 6 and Section 7, respectively. Section 8 describes the method of association rules, and, finally, Section 9 describes the utility functions available in NAG DMC.

2 Task Guide

This section provides some general guidance on the selection of suitable NAG DMC functions to solve common analytical tasks. If there is a choice of functions for a particular task, further guidance is given in the form of a table; otherwise the reader is directed to the relevant section in this document which describes the methods and available functions.

The common analytical tasks are defined as:

(a) replace missing values in data; see Section 3.1;
(b) reduce the number of variables while retaining most of the information; see Section 4.2;
(c) reduce the number of data records while retaining most of the information; see Section 5;
(d) find unusual data records in a set of data; see Section 3.2;
(e) find groups in data; see Section 5;
(f) assign data records to one or more groups; see Section 6;
(g) predict continuous values in data, e.g., a monetary cost; see Section 7;
(h) compute rules for associations between data values; see Section 8.

3 Data Cleaning

3.1 Data Imputation

One approach to handling data containing values missing at random is to replace each missing value by a suitable value. This task is known as data imputation, for which there are three fundamental approaches.

The first approach, the simplest of the three, is to replace missing values on a continuous variable by the mean of the variable and to replace missing values on a categorical variable by the mode of the variable.

The second approach is to find the data record that is most similar to the one with the missing value; this record is called the donor. If there are more than one possible donors then the donor used can be selected in different ways, either by choosing one at random or by selecting the first in the list. In these approaches the skill is in selecting the best variables to match the recipient and the donor. An effective method used to select donors is to use a set of distance criteria for a set of variables. For continuous variables, the choices of distance criteria include the Euclidean, Manhattan, regression and threshold distances; for categorical variables simple (Boolean) matching and scaled rank differences are common choices. The user may also supply distance tables for categorical variables.

The third approach is to use mathematical models to predict the missing value. In general, an iterative approach can be used to fit a model by using data containing missing values. Starting

with guesses for the missing values, a model is fitted to the data and then used to produce better estimates of the missing values. This procedure is then iterated until estimates of missing values converge. If the fitting is by maximum likelihood and the prediction by taking the expected value then the model is the EM algorithm for imputation.

3.1.1 Table

Method	Most Intensive:		Categorical Variable?
	Memory	Computation	
Simple			Yes
Distance	$\star$	$\star$	Yes
EM			No

Table 1: *Table of functions for data imputation tasks. If an analysis includes data on categorical variables, the EM method for data imputation should not be used. See the function documents described in Section 3.1.2 for further detail on these methods.*

3.1.2 Functions

The function **nagdmc_impute_simp** replaces missing data values with either the mean or the mode of variables for continuous and categorical variables, respectively.

The function **nagdmc_impute_dist** replaces missing data values by using a range of distance criteria for variables.

The function **nagdmc_impute_em** replaces missing data values by using an EM algorithm.

Memory allocated by the data imputation functions may be returned to the operating system by using **nagdmc_free_impute**.

3.2 Outlier Detection

Outlier detection concerns finding suspect data records in a set of data. Data records are identified as suspect if they seem not to be drawn from an assumed distribution. In NAG DMC a multivariate Normal distribution is assumed and, therefore, data values are assumed to be on continuous variables.

The first step is to search for a subset of data records assumed to be "typical". This search can be based on either the Mahalanobis distance from the mean values or distances from medians of p variables.

The method then grows the initial "typical" subset by including all data records at Mahalanobis distances from the mean values less than the value of a corrected χ^2 distribution with p degrees of freedom, for a given rejection level, α.

When the size of the "typical" subset no longer changes, the data records not included in the "typical" subset are deemed to be outliers.

If the Mahalanobis distance is selected to choose the initial "typical" subset, the EM data imputation function (see Section 3.1) is used to impute any missing values in the data.

3.2.1 Functions

The function **nagdmc_bacon** computes a feed-forward outlier search for a given a initial value for the number of "good" data records and a rejection level for the χ^2 distribution. The name BACON is due to the acronym: Blocked Adaptive Computationally-efficient Outlier Nominator; and is contrived in honour of Sir Francis Bacon who in 1620 wrote:

"Whoever knows the ways of Nature will more easily notice her deviations; and, on the other hand, whoever knows her deviations will more accurately describe her ways."

4 Data Transformations

4.1 Scaling Data

The contribution to a distance computation by data values on a continuous variable depends on the range of its values. Thus, for functions which do not including scaling as an option, it is often

preferable to transform all data values on continuous variables to be on the same scale. The usual method of scaling continuous variables is to subtract the mean value from data values and divide by the standard deviation. This results in data on continuous variables having zero mean and unit variance.

4.2 Principal Component Analysis

Principal component analysis (PCA) is a tool for reducing the number of variables needed in an analysis. PCA derives a set of orthogonal, i.e., uncorrelated, variables that contain most of the information (measured in terms of variance or sums of squares) in the original data. The new variables, called principal components, are calculated as linear transformations of the original data.

The choice of the number of principal components used can be based either on the number of principal components needed to explain a given percentage of information in the data or by carrying out statistical tests. There is also the question of what type of standardisation to use. If all the variables are of a similar magnitude and variation, no standardisation is required. If the variables differ in magnitude then standardisation should be applied. If the standard deviation is used then this is equivalent to performing PCA on the correlation matrix.

4.3 Functions

4.3.1 Scaling

The function **nagdmc_scale** scales data values on continuous variables to have zero mean and unit variance.

4.3.2 Principal Components Analysis

The function **nagdmc_pca** computes a principal component analysis. If standardisation is required the options are: standardisation by standard deviation, and user-supplied scalings; otherwise the options are sums of squares or variance. The function returns the coefficients that are used to compute the new variables and information on the percentage of variation contained in each component. The user has the option of supplying the means and, if required, the standard deviations (if these are available) to avoid them being re-computed.

The function **nagdmc_pca_score** computes the transformed values of a data record, given a value for the number of components to use and information returned by nagdmc_pca.

5 Cluster Analysis

Cluster analysis is the statistical name for techniques that aim to find groups of similar data records in a study. For example, if there are data on customers who purchased a number of product lines, the retailer may wish to group together customers with similar purchasing patterns. Given these groups, further analysis can be carried out, e.g., are there other key characteristics that define the group?

Cluster analysis starts by working out how similar or how different two individuals are. Having encoded and appropriately scaled data, the next step is to group together data records that are close together. There are two main approaches, namely k-means clustering and hierarchical clustering.

5.1 k-means Clustering

In k-means clustering the user decides how many groups or clusters there are in the data. Unless there is prior knowledge of the number of groups in the data, it is wise to experiment with different values of k to determine which is best suited to their data. The method then allocates each data record to one of k groups. The approach is to start with an initial allocation and then move data records around between groups until the grouping with the smallest total within-group distance is found.

The initial allocation can be found in a number of different ways. One approach is to select k individuals at random and use then as the group centres. Then allocate the remainder to the group with the nearest group centre. Other approaches to finding initial clusters include using prior information to give the group centres or using key individuals as group centres.

5.2 Hierarchical Clustering

Hierarchical clustering usually starts from the collection of data records and agglomerates them step-by-step until there is only one group. The advantage of this approach is that by looking at the way the groups join together there is an indication of how many groups there are in the data and how well defined they are. There is also no requirement to provide a starting point. There are six common methods for hierarchical clustering: single link, complete link, group average, centroid, median and minimum variance. The difference between the methods is in the way distances between groups are calculated from distances between individuals. This choice affects the properties of the clustering method.

The standard approach to carrying out hierarchical clustering is first to compute the distance matrix for all individuals then update this matrix as the analysis proceeds. This approach is suitable for at most a few thousand individuals. However, of the six common methods of hierarchical clustering, an alternative approach exists for two of them, namely the group average and minimum variance methods. These two methods produce more "rounded" clusters than single link clustering. Both of these two methods have the property that the group members can be replaced by the group centroid or mean when calculating the between-group distances, so only the group means need to be stored at any one time. This approach is efficient computationally since as many mergers as possible are computed in parallel.

5.3 Table

Method	Most Intensive:		Number of Groups Known?
	Memory	Computation	
k-means Clustering			Yes
Hierarchical Clustering	⋆	⋆	No

Table 2: *Table of cluster analysis functions. You must give a value for the number of groups in a data set for a k-means cluster analysis. See the function documents described in Section 5.4 for further detail on these methods.*

5.4 Functions

5.4.1 k-means Clustering

The utility function **nagdmc_rints** can be used to randomly select data records used to form cluster centres, whereas **nagdmc_nrgp** can be used to allocate data records to the nearest cluster.

The function **nagdmc_kmeans** computes a k-means cluster analysis for Euclidean distances.

Given a clustering, the function **nagdmc_wcss** can be used for computing the within-cluster sums of squares.

5.4.2 Hierarchical Clustering

The function **nagdmc_hclust** provides either group average or minimum variance hierarchical cluster analysis and returns the history of the agglomeration.

The function **nagdmc_cind** takes the results of an hierarchical cluster analysis and allocates the data records to the required number of groups.

6 Classification

6.1 Decision Trees

Decision trees use recursive partitioning to associate data records with a set of leaf nodes. Figure 1 shows an example of a simple binary decision tree. In the figure a test at root node A partitions data into two child nodes: the internal node B; and the leaf node C. Node B partitions the data associated with it into two leaf nodes, node D and node E. Since each parent node has two child nodes, this tree is known as a binary tree. In general, decision trees in which parent nodes can be associated with any number (greater than one) of child nodes are known as n-ary decision trees.

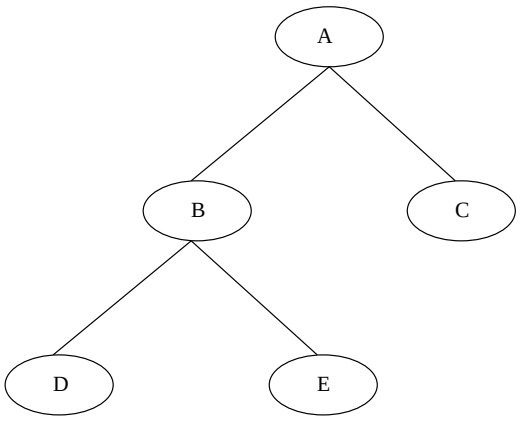

Figure 1: *Example of a binary decision tree. Ellipses are used to represent nodes in the tree and parent nodes are linked by line segments to their child nodes in the tree.*

A partition of data is found by evaluating the result of tests on data at a node and retaining the best test according to a criterion. For the decision tree classifiers in NAG DMC this criterion is either the Gini index of impurity or minimum entropy. Summary statistics, such as the mean or mode of data, are used to label each leaf node with a value for the dependent variable in the data.

NAG DMC includes functions to compute binary and n-ary decision trees for classification.

6.2 Generalised Linear Models

Generalised linear models allow a wide range of models to be fitted. These included logistic and probit regression models for binary data, and log-linear models for contingency tables. A generalised linear model consists of:

(a) A distribution for the dependent variable, Y;
(b) A model, known as a linear predictor, defined by a linear combination of independent variables;
(c) A link function between the expected value of Y and the linear predictor.

In NAG DMC the following distributions are available:

(a) binomial distribution, typically used for binary classification tasks, the available link functions are:
  (i) logistic link;
  (ii) probit link;
  (iii) complementary log-log.
(b) Poisson distribution, typically used for count data, the available link functions are:
  (i) exponent link;
  (ii) identity link;
  (iii) log link;
  (iv) square root link;
  (v) reciprocal link.

In each case maximum likelihood estimation estimates for the free parameters in a model are found by using an iterative weighted least-squares procedure.

6.3 Multi-layer Perceptron Neural Networks

See Section 7.3 for an introduction to multi-layer perceptrons (MLP) models.

If a classification tasks has two classes, due to the nature of optimisation, one class should be coded as zero and the other as one; otherwise for $c > 2$ classes c binary dependent variables should be constructed that take the value one to indicate class membership and else are set to zero.

6.4 Nearest Neighbour Models

Given a data record x with an unknown value associated with its dependent variable, a k-nearest neighbour model finds in a set of training data the k data records most similar to x. The measure of similarity between x and training data records is a distance function computed over the independent

variables. The value of the dependent variable is then set equal to the class value with the highest probability amongst the k-nearest neighbours. Optionally, prior probabilities may be set for each class value.

The measure of similarity used when comparing data records can be one of two distance functions. Firstly, the Euclidean distance squared which calculates a measure of distance by evaluating the sum of squared differences between data values. Secondly, the ℓ_1-norm, or Manhattan distance, which calculates a measure of distance by evaluating the sum of absolute differences between data values.

6.5 Table

Method	Most Intensive:		More Than	Unequal Class
	Memory	Computation	Two Classes?	Proportions?
Decision Trees			Yes	Yes
Generlised Linear Models			No	Yes
Multi-layer Perceptron		$\star$	Yes	No
Nearest Neighbour Models	$\star$		Yes	Yes

Table 3: *Table of functions used to classify data. MLP models should be applied to data sets with (roughly) equal numbers of data records in each category. See the function documents described in Section 6.6 for further detail on these methods.*

6.6 Functions

6.6.1 Decision Trees

The function **nagdmc_gini_tree** computes a binary decision tree by using a criterion based on the Gini index of impurity. The computed decision tree can be saved to and loaded from a binary file by using the functions **nagdmc_save_gini_tree** and **nagdmc_load_gini_tree**, respectively. The memory containing a Gini index decision tree can be returned to the operating system by using the function **nagdmc_free_gini_tree**.

The function **nagdmc_predict_gini_tree** can be used to compute predictions for new data records given a decision tree computed by nagdmc_gini_tree.

The function **nagdmc_entropy_tree** computes a n-ary decision tree by using the minimum entropy criterion. The accuracy of an entropy decision tree on new data may be increased by pessimistic error pruning of the fitted model. The function **nagdmc_prune_entropy_tree** can be used for this purpose.

The fitted entropy decision tree can be saved to and loaded from a binary file by using the functions **nagdmc_save_entropy_tree** and **nagdmc_load_entropy_tree**, respectively.

The memory containing an entropy decision tree can be returned to the operating system by using **nagdmc_free_entropy_tree**.

The function **nagdmc_predict_entropy_tree** can be used to compute predictions for new data records given a decision tree computed by nagdmc_entropy_tree.

6.6.2 Generalised Linear Models

The function **nagdmc_binomial_reg** computes a generalised linear model with binomial errors.

The function **nagdmc_logit_reg** provides an easier to use interface for the computation of logistic regression models, i.e., generalised linear models with binomial errors and the logistic link function.

The function **nagdmc_probit_reg** provides an easier to use interface for the computation of generalised linear models with binomial errors and the probit link function.

The function **nagdmc_poisson_reg** computes a generalised linear model with Poisson errors.

The function **nagdmc_loglinear_reg** provides an easier to use interface for the computation of generalised linear models with Poisson errors and the log-linear link function.

Information on fitted regression models is available by using **nagdmc_extr_reg**, and **nagdmc_predict_reg** computes predictions for new data on independent variables.

A fitted generalised linear model can be saved to disk by using **nagdmc_save_reg** and read into memory by using **nagdmc_load_reg**. The function **nagdmc_free_reg** returns to the operating system memory allocated to a generalised linear model.

6.6.3 Multi-layer Perceptron Neural Networks

The function **nagdmc_mlp** is used to compute an MLP model. The function can be made to try a number of initial conditions by using different random values for the free parameters in which case the function uses the best of these initial optimisations as a starting point for the main optimisation. The user controls the number of initial optimisations and the number of iterations of the optimisation.

The function **nagdmc_predict_mlp** can be used to compute predictions for new data records given MLP model information as returned by **nagdmc_mlp**.

6.6.4 Nearest Neighbour Models

In NAG DMC, nearest neighbour approximations are found by using a two-stage process. Firstly, a k-d tree is computed by using **nagdmc_kdtree**. This tree can be saved into a binary file for future use with **nagdmc_save_kdtree**, and loaded into memory from the file by **nagdmc_load_kdtree**. The function **nagdmc_free_kdtree** returns the memory containing a k-d tree to the operating system.

Secondly, nearest neighbour approximations can be computed using prior probabilities of class membership by **nagdmc_knnc**.

7 Regression

7.1 Decision Trees

The two decision trees available in NAG DMC for regression tasks are both binary trees, see Section 6.1. Each decision tree uses as its criterion the minimum sum of squares about the mean for data at a node. However, one of the decision trees uses a robust estimate of the mean of the dependent variable at a node, whereas the other uses the simple average.

7.2 Linear Regression

Linear regression models can be used to predict an outcome y from a number of independent variables. The predictive model is a linear combination of independent variables and a constant term. The coefficients or weights in a model are estimated by using a training set of data with known values of y and fitted using least squares approximation. The sign and magnitude of the coefficients may be used interpret a fitted model. The ratio of a coefficient value to its standard error measures its statistical importance; this ratio would generally be greater than two for the variable to be considered worth keeping in a model.

Pre-processing of data values on categorical independent variables may be required. If a categorical variables has more than two levels (for example, a variable could take the values red, blue or green), dummy variables should be calculated and these used in a model; otherwise the two categories should be encoded as zero and one.

7.3 Multi-layer Perceptron Neural Networks

Multi-layer perceptrons (MLPs) are flexible non-linear models that may be represented by a directed graph, as shown in Figure 2.

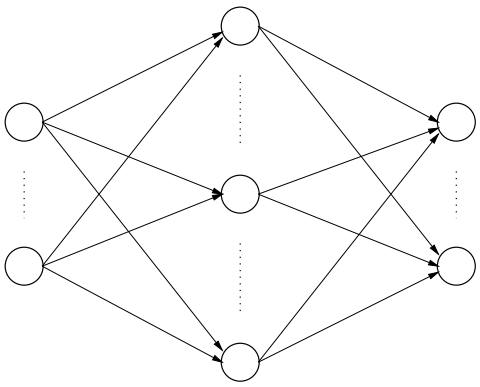

Figure 2: *Diagram of a feed-forward MLP with a single hidden layer. Computational nodes in the MLP are represented by circles; line segments connecting circles represent weights in the MLP. The input layer consists of as many nodes as there are independent variables in the analysis. The output layer consists of as many nodes as there are dependent variables in the analysis. You select the number of nodes in the hidden layer, and, for a given number of independent and dependent variables, this determines the number of free parameters in an MLP model.*

A node in the input layer of an MLP takes the value of its independent variable and multiplies this by a weight value. A node in either the hidden or output layer sums the contributions from nodes it is connected to in the previous layer, and applies a transfer function to this sum. There are several possibilities for transfer functions, including the sigmoidal (logistic) function, the hyperbolic tangent function, and the linear function.

The process of optimising values of the free parameters in an MLP is known as training. Training involves minimising the sum of squared error function between MLP predictions and training data values. Typically, the error surface of such an optimisation is highly complex. Thus training an MLP can be an intensive computational task and, therefore, MLPs are not generally suitable for large data sets.

It is recommended that data values on continuous variables are scaled before running a multi-layer perceptron as this makes MLPs easier to train successfully and tends to produce results with a higher accuracy.

Even if it were possible to find a global minimum when optimising parameter values in a multi-layer perceptron, it is unlikely that this would be desirable due to the problem of over-fitting a model. In order to avoid over-fitting, the optimisation is usually halted prematurely by use of a set of validation data. Hence the initial values for the free parameters in the MLP can have a significant influence on the accuracy of approximation results. For this reason several optimisations should be tried each with different initial conditions.

7.4 Nearest Neighbour Models

Given a data record x with an unknown value associated with its dependent variable, a k-nearest neighbour model finds in a set of training data the k data records most similar to x. The measure of similarity between x and training data records is a distance function computed over the independent variables in x. The value of the dependent variable is then set equal to the mean value of the dependent variable for the k-nearest neighbours.

The measure of similarity used when comparing data records can be one of two distance functions. Firstly, the Euclidean distance squared which calculates a measure of distance by evaluating the sum of squared differences between data values. Secondly, the ℓ_1-norm, or Manhattan distance, which calculates a measure of distance by evaluating the sum of absolute differences between data values.

7.5 Radial Basis Function Models

A radial basis function (RBF) computes a scalar function of the distance from its centre location to the values of independent variables in a data record, where distance is measured using the Euclidean norm. The common choices of RBF include the linear, cubic, thin plate spline, Gaussian,

multiquadric, inverse multiquadric and Cauchy functions. Centre locations of RBFs may be chosen based on domain knowledge of the task or by cluster analysis, see Section 5.

An RBF model is a linear combination of the outputs of RBFs, the weights in which are given by a least squares fit of the model to training data records.

As the computation of an RBF model involves distance calculations, any categorical variables with more than two levels should be replaced by dummy variables in the analysis; values on categorical variables with two levels should be labelled zero and one.

7.6 Table

Method	Most Intensive:		Non-linear Task?
	Memory	Computation	
Regression Decision Tree			Yes
Linear Regression			No
Multi-layer Perceptron		⋆	Yes
Nearest Neighbour Models	⋆		Yes
Radial Basis Function Model			Yes

Table 4: *Table of functions for regression tasks. Linear regression models may not give accurate predictions for non-linear tasks (i.e., for tasks which cannot be adequately described by a straight line). See the function documents described in Section 7.7 for further detail on these methods.*

7.7 Functions

7.7.1 Decision Trees

The function **nagdmc_reg_tree** computes binary decision trees using a criterion that partitions data at a node so that the sum of squares about the mean is minimised. A computed decision tree can be saved to a binary file by using **nagdmc_save_reg_tree** and subsequently loaded into memory by **nagdmc_load_reg_tree**. The function **nagdmc_free_reg_tree** returns to the operating system memory used to store a decision tree computed by nagdmc_reg_tree.

The function **nagdmc_predict_reg_tree** can be used to compute predictions for new data records given a decision tree computed by nagdmc_reg_tree.

The function **nagdmc_waid** computes binary decision trees using a criterion that partitions data at a node so that the sum of squares about a robust estimate of the mean is minimised. A computed decision tree can be saved to a binary file by using **nagdmc_save_waid** and subsequently loaded into memory by **nagdmc_load_waid**. The function **nagdmc_free_waid** returns to the operating system memory used to store a decision tree computed by nagdmc_waid.

The function **nagdmc_predict_waid** can be used to compute predictions for new data records given a decision tree computed by nagdmc_waid.

7.7.2 Linear Regression

The function **nagdmc_linear_reg** computes the coefficients and returns information on the fit of the model, e.g., the R^2 value that gives the amount of variation in the dependent variable explained by a fitted model.

The function **nagdmc_stepwise_reg** can be used as a automatic tool for model selection, i.e., by selecting a combination of independent variables from a given set for inclusion in a linear regression model.

Results from a fitted linear regression model can be extracted by using **nagdmc_extr_reg**, and **nagdmc_predict_reg** computes predictions for new data on independent variables.

A fitted regression model can be saved to disk by using **nagdmc_save_reg** and read into memory by using **nagdmc_load_reg**. The function **nagdmc_free_reg** returns to the operating system memory allocated to a regression model.

7.7.3 Multi-layer Perceptron Neural Networks

The function **nagdmc_mlp** is used to compute an MLP model. The function can be made to try a number of initial conditions by using different random values for the free parameters in which

case the function uses the best of these initial optimisations as a starting point for the main optimisation. The user controls the number of initial optimisations and the number of iterations of the optimisation.

The function **nagdmc_predict_mlp** can be used to compute predictions for new data records given MLP model information as returned by **nagdmc_mlp**.

7.7.4 Nearest Neighbour Models

In NAG DMC, nearest neighbour approximations are found by using a two-stage process. Firstly, a k-d tree is computed by using **nagdmc_kdtree**. This tree can be saved into a binary file for future use with **nagdmc_save_kdtree**, and loaded into memory from the file by **nagdmc_load_kdtree**. The function **nagdmc_free_kdtree** returns the memory containing a k-d tree to the operating system.

Secondly, nearest neighbour approximations can be computed by using **nagdmc_knnp**.

7.7.5 Radial Basis Function Models

The function **nagdmc_rbf** computes a least squares fit of a given number of RBFs of a chosen type to training data records.

The function **nagdmc_predict_rbf** computes RBF model predictions given a fitted model for new data records.

8 Association Rules

The goal of association analysis is to determine relationships between nominal data values. The relationships take the form of rules and are expressed as:

$$\text{Antecedent} \rightarrow \text{Consequent}.$$

For example, consider the rule:

$$\{1, 41, 6\} \rightarrow 19,$$

which, in the case of market basket analysis, could be interpreted as:

"Customers who purchase the items with identifiers 1, 41 and 6 are likely to buy item 19."

In general, for large data sets the number of available rules can be very high. For example, given a supermarket product range of several thousand items there may be billions of rules. However, the number of rules generated can be reduced by limiting the search to those rules with a high probability.

Data for the association rules generator in NAG DMC must be sorted in ascending order for each transaction.

8.1 Functions

The utility functions can be used to sort transaction data (see Section 8).

Once sorted, transaction data can be read into an array by using **nagdmc_assoc_data**. The function **nagdmc_assoc** computes association rules for the sorted data. Association rules can be printed by using **nagdmc_assoc_print**.

9 Utility Functions

The utility functions are designed to support the main functions described above, and to help with prototyping.

Utility functions are included which:

(a) compute random real and integer numbers with or without replacement. These functions can be used, e.g., to compute random subsets of data records;
(b) rank and sort arrays of real and integer numbers;
(c) compute summary statistics, e.g., mean and variance for continuous variables, for a set of data records;
(d) cross-tabulate results following a classification.

9.1 Functions

nagdmc_rng is the basic random number generator, returning a random number between 0 and 1 and using a starting point set by **nagdmc_srs**, optionally using the system clock.

The function **nagdmc_rints** uses the basic random number generator to produce a sample of random integers in a given range that can be used to select data records from data set. The sample may either include repeated numbers or exclude repeats.

Functions used to rank real and integer data are **nagdmc_rank_real** and **nagdmc_rank_long**, respectively. The function **nagdmc_index** converts a rank order into an index order.

Given a rank or index order, the functions used to reorder real-valued data and integer-valued data are **nagdmc_order_real** and **nagdmc_order_long**, respectively.

nagdmc_dsu updates means and sums of squares about the mean for a set of data. It can be used to compute the sample mean and variance by processing chunks of data at a time.

The function **nagdmc_tab2** is a simple two-way tabulation that can be used to compare two-way classifications.
