



THE  
POWER  
TO KNOW.

**SAS<sup>®</sup> Enterprise Miner<sup>™</sup> and  
SAS<sup>®</sup> Text Miner Procedures  
Reference for SAS<sup>®</sup> 9.1.3  
The DMVQ Procedure  
(Book Excerpt)**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® *Enterprise Miner™* and SAS® *Text Miner Procedures: Reference for SAS® 9.1.3*, Cary, NC: SAS Institute Inc.

**SAS® Enterprise Miner™ and SAS® Text Miner Procedures: Reference for SAS 9.1.3**

Copyright © 2008 by SAS Institute Inc., Cary, NC, USA.

All rights reserved. Produced in the United States of America.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

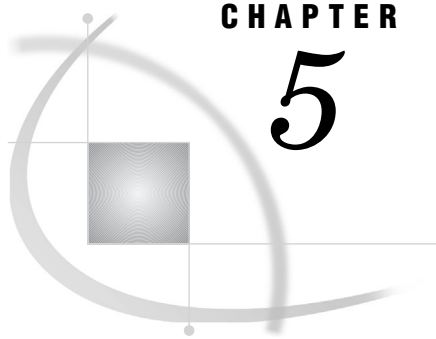
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, October 2008

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/publishing](http://support.sas.com/publishing) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.



## CHAPTER

## 5

## The DMVQ Procedure (Experimental)

<i>Overview: DMVQ Procedure</i>	63
<i>Syntax: DMVQ Procedure</i>	63
<i>PROC DMVQ STATEMENT</i>	64
<i>INPUT Statement</i>	67
<i>TARGET Statement</i>	74
<i>VQ Statement</i>	74
<i>SOM Statement</i>	74
<i>MAKE Statement</i>	75
<i>INITIAL Statement</i>	76
<i>TRAIN Statement</i>	77
<i>CODE Statement</i>	81
<i>Details: DMVQ Procedure</i>	83
<i>Comparing DMVQ and FASTCLUS Procedures</i>	83
<i>References</i>	85

### Overview: DMVQ Procedure

The DMVQ procedure performs data mining vector quantization. Data mining vector quantization is used to form disjoint clusters of observations based on some criteria.

### Syntax: DMVQ Procedure

```

PROC DMVQ
  DMDBCAT <option(s)>;
  INPUT variables / <option(s)>;
  TARGET variables / <option(s)>;
  VQ <option(s)> | SOM <option(s)> ;
  <MAKEoption(s)>;
  <INITIAL <option(s)>;>
  <TRAIN <option(s)>;>
  <SCORE <option(s)>;>
  <CODE <option(s)>;>

```

The DMVQ procedure does not have a **FREQ** statement. You must use the **DMDB** procedure if you want to specify a **FREQ** variable. DMVQ is an interactive procedure. Each statement is executed when it is submitted. The order in which the statements

are submitted is important. The procedure does not terminate until you submit a QUIT statement or a new step. If an ERROR message halts the process, submit a RUN statement to resume normal processing by the DMVQ procedure.

#### Outline of DMVQ Procedure Use

- 1 Submit a DMVQ procedure statement.
- 2 Submit one or more INPUT or TARGET statements.
- 3 Submit a VQ or SOM statement.
- 4 Optionally, submit an INITIAL statement.
- 5 Optionally, submit a TRAIN statement.
- 6 Optionally, submit any number of SAVE, SCORE, and CODE statements.
- 7 Quit.

---

## PROC DMVQ STATEMENT

Invokes the DMVQ procedure.

**PROC DMVQ** *<option(s)>*;

### Required Arguments

The following argument is required by the DMVQ procedure:

**DMDBCAT=** *<libref.> SAS-catalog*

specifies the data mining database (DMDB) catalog that the DMVQ procedure will use. The DMDB catalog pre-summarizes the data, including continuous statistics such as mean, standard deviation, and skew. It also counts all discrete values of class variables. The DMDB is maintained as a SAS data set. The metadata information that is associated with the DMDB is maintained in a SAS catalog.

### Options

The following options can appear in the DMVQ procedure statement:

**DATA=** *<libref.> SAS-data-set*

specifies the input SAS data set that contains the training data. There is no default setting for the DATA= option. The DATA= option is not required in the PROC DMVQ statement.

**MESSAGE | MSG=**

specifies the amount of information written to the SAS log. The following settings are available for the MESSAGE option:

**SILENT | S**

writes no process information to the SAS log.

**TERSE | T**

writes a subset of the available process information to the SAS log.

**VERBOSE | V**

writes all available process information to the SAS log.

**MINMAX | NOMINMAX**

MINMAX requests that the minimum and maximum values of each variable in each cluster be written to the OUTSTAT= data set. NOMINMAX saves memory by not computing the statistics.

**Default:** MINMAX

**NOMINAL | NOM=**

specifies the encoding setting that is used for nominal variables when no encoding is specified by using a LEVEL= option. See the LEVEL= option for additional information.

The following encoding settings are available for the NOMINAL option:

GLM

generalized linear model encoding

REFERENCE | REF

reference encoding

DEVIATION | DEV

deviation encoding

THERMOMETER | THERM | THER

thermometer encoding

BATHTUB | BATH

bathtub encoding

**Default:** GLM

**ORDINAL | ORD=**

specifies the encoding setting that is used for ordinal variables when no encoding is specified by using a LEVEL= option. See the LEVEL= option for additional information.

The following encoding settings are available for the ORDINAL option:

COUNT

count encoding

INDEX

index encoding

RANK

rank encoding

BATHTUB | BATH

bathtub encoding

**Default:** RANK

**STANDARDIZE | STDIZE | STD= *method-to-standardize-expanded-variables***

specifies the standardization methods to use when computing the location and scale measures. These measures are then used to standardize expanded (dummy and transformed) variables. Expanded variables are standardized by subtracting the location and dividing by the scale measure. If no method is specified, the STD (standard deviation) method is used. The following STDIZE= methods are available to standardize expanded variables:

NONE | NON | NO

specifies that no standardization is performed. Only STDIZE=NONE can be used if MISSING=OMITCASE is specified for any variable.

MEAN

standardizes expanded variables by subtracting the mean and dividing by one.

**MEDIAN | MED**

standardizes expanded DMDB class variables only. Standardizes the variables by subtracting the median.

**SUM**

standardizes expanded variables by dividing by the product of (sum \* dimension).

**EUCLLEN | EUC**

standardizes expanded variables by dividing by the Euclidean length. Euclidean length is calculated as the product of ((SQRT(Uncorrected Sum of Squares) \* (SQRT(dimension))).

**USTD**

standardizes expanded variables by dividing by the uncorrected standard deviation about the origin, which is the product of ((SQRT(Uncorrected Sum of Squares/Number of Observations)) \* (SQRT(dimension))).

**STD**

standardizes expanded variables by subtracting the mean and dividing by the product of (Standard deviation \* (SQRT(dimension))).

**RANGE | RAN**

standardizes expanded variables by subtracting the minimum variable value and dividing by the uncorrected standard deviation, which is the product of ((SQRT(Uncorrected Sum of Squares/Number of Observations)) \* (SQRT(dimension))).

**MIDRANGE | MID**

standardizes expanded variables by subtracting the midrange variable value and dividing by the half-range.

**MAXABS | MAX**

standardizes expanded variables by dividing by the maximum absolute value.

**IQR**

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the median and dividing by the product of (Interquartile range \* (SQRT(dimension))).

**MAD**

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the median and dividing by the median absolute deviation.

**ABW <p> | where p > 0**

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the biweight  $M$  estimator and dividing by the product of (biweight  $A$  estimate \*(SQRT(dimension))). The  $p$  argument is the tuning constant value where  $p > 0$ .

**AHUBER <p> | where p > 0**

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the Huber  $M$  estimator and dividing by the product of (Huber  $A$  estimate \*(SQRT(dimension))). The  $p$  argument is the tuning constant value where  $p > 0$ .

**AWAVE <p> | where p > 0**

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the Wave  $M$  estimator and dividing by the product of (Wave  $A$  estimate \*(SQRT(dimension))). The  $p$  argument is the tuning constant value where  $p > 0$ .

**AGK (p)**

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the mean and dividing by the product of (AGK estimate  $\times$  (SQRT(dimension))). The  $p$  argument is a numeric value where  $0 < p < 1$ , and specifies the proportion of pairs to be included in the estimate of the within-cluster variances.

**SPACING** ( $p$ )

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the mid-minimum spacing and dividing by the product of minimum spacing  $\times$  (SQRT(dimension)). The  $p$  argument is a numeric value where  $0 > p > 1$ , and specifies the proportion of data to be contained in the spacing.

**LEASSTP** ( $p$ ) | **LP** ( $p$ ) | **L** ( $p$ )

standardizes expanded DMDB class variables only. It standardizes the variables by subtracting the  $L(p)$  location and dividing by the product of ( $L(p)$  scale $\times$  dimension $^{1/p}$ ). The  $p$  argument is a numeric value that specifies the power to which differences are to be raised in computing an  $L(p)$  or Minkowski metric.

**TESTDATA** | **TEST=** *<libref> SAS-data-set*

specifies the test data set.

**Default:** There is no default setting for the TESTDATA=option.

**VALIDDATA** | **VALIDATA** | **VALID=** *<libref> SAS-data-set*

specifies the validation data set.

**Default:** There is no default setting for the VALIDDATA=option.

---

## INPUT Statement

The INPUT statement specifies the input variables and options.

**INPUT** *variable-name(s)* / *<option(s)>* ;

### Required Argument

***variable-name(s)***

The variables must have been specified in a CLASS or VAR statement in the required DMDB procedure statement. The variables can be character variables or numeric variables, depending on how the LEVEL= option is configured.

INPUT variables are used to compute clusters. Variables that are listed in an INPUT statement can be rejected from use in cluster computations, if the information in the DMDB catalog shows any of the following to be true about the variable:

- A VAR variable has zero variance.
- A CLASS variable has fewer than two usable categories. A CLASS variable category is usable if it has a nonmissing value, or if the MISSING=CATEGORY option is specified.

Even if an INPUT variable is rejected, missing values for that variable can be replaced in an OUT= data set, provided that a nonmissing replacement value is available.

## Options

The following options can be used in an INPUT statement:

### **ID= *name***

specifies the identifier for the input layer. The name must be a valid SAS name with no more than eight characters.

### **LEVEL= *measurement-level* <(encoding)>**

specifies the measurement level for a variable. The combinations indicated in the table below are allowed:

**Table 5.1** Allowable Variable Measurement Levels by DMDB Statement Type and Variable Type

Variable Measurement Levels				
DMDB Statement	Variable Type	LEVEL= Nominal	LEVEL= Ordinal	LEVEL= Interval
VAR	numeric	no	no	yes
CLASS	numeric	yes	yes	yes
CLASS	character	yes	yes	no

The following measurement levels can be used with the LEVEL= option:

### NOMINAL | NOM

nominal variable measurement level. LEVEL=NOMINAL is the default setting for DMDB CLASS variables. You can specify LEVEL=NOMINAL for both numeric and character variables in a DMDB CLASS statement. You cannot specify LEVEL=NOMINAL for variables in a DMDB VAR statement.

### ORDINAL | ORD

ordinal variable measurement level. You can specify LEVEL=ORDINAL for both numeric and character variables in a DMDB CLASS statement. You cannot specify LEVEL=ORDINAL for variables in a DMDB VAR statement.

### INTERVAL | INT

interval variable measurement level. LEVEL=INTERVAL is the default setting for DMDB VAR variables. You can specify LEVEL=INTERVAL for numeric variables in DMDB VAR and DMDB CLASS statements. You cannot specify LEVEL=INTERVAL for character variables.

### (encoding) (optional with LEVEL= specification)

For nominal variables and ordinal variables, the measurement level can be followed by the encoding method in parentheses as an option. The encoding method specifies how to compute one or more quantitative variables to be used for computing distances. These quantitative variables are similar to the dummy variables that are used in linear models. The number of quantitative variables computed for an encoding is called the “dimension” of the encoding.

Each encoding method below is accompanied by an example matrix. The example matrix is populated from the data set that is generated by the following SAS code:

```
data;
input x $ @@;
cards;
```

d c b a d c b d c d;

The following encoding methods are available:

**LEVEL=NOMINAL (GLM)**

The dimension is the number of categories.

**Table 5.2** Ordinal-Class Variable Encoding Matrix Using GLM Encoding

<b>X</b>	<b>Xa</b>	<b>Xb</b>	<b>Xc</b>	<b>Xd</b>
<b>a</b>	1	0	0	0
<b>b</b>	0	1	0	0
<b>c</b>	0	0	1	0
<b>d</b>	0	0	0	1

**LEVEL=NOMINAL (REFERENCE | REF)**

The dimension is the number of categories minus one.

**Table 5.3** Ordinal-Class Variable Encoding Matrix Using Reference Encoding

<b>X</b>	<b>Xa</b>	<b>Xb</b>	<b>Xc</b>	<b>Xd</b>
<b>a</b>	1	0	0	0
<b>b</b>	0	1	0	0
<b>c</b>	0	0	1	0
<b>d</b>	0	0	0	0

**LEVEL=NOMINAL (DEVIATION | DEV)**

The dimension is the number of categories minus one.

**Table 5.4** Ordinal-Class Variable Encoding Matrix Using Deviation Encoding

<b>X</b>	<b>Xa</b>	<b>Xb</b>	<b>Xc</b>
<b>a</b>	1	0	0
<b>b</b>	0	1	0
<b>c</b>	0	0	1
<b>d</b>	-1	-1	-1

**LEVEL=ORDINAL (RANK)**

The dimension is one.

**Table 5.5** Ordinal-Class Variable Encoding Matrix Using Rank Encoding

<b>X</b>	<b>T_X</b>
<b>a</b>	0.05
<b>b</b>	0.20

<b>X</b>	<b>T_X</b>
<b>c</b>	0.45
<b>d</b>	0.80

**LEVEL=ORDINAL (INDEX)**

The dimension is one.

**Table 5.6** Ordinal-Class Variable Encoding Matrix Using Index Encoding

<b>X</b>	<b>T_X</b>
<b>a</b>	0.0
<b>b</b>	0.333333333
<b>c</b>	0.666666667
<b>d</b>	1

**LEVEL=ORDINAL (THERMOMETER | THERM)**

The dimension is the number of categories minus one.

**Table 5.7** Ordinal-Class Variable Encoding Matrix Using Thermometer Encoding

<b>X</b>	<b>Xa</b>	<b>Xb</b>	<b>Xc</b>
<b>a</b>	1	0	0
<b>b</b>	0	1	0
<b>c</b>	0	0	1
<b>d</b>	1	1	1

**LEVEL=ORDINAL (BATHTUB | BATH)**

The dimension of the number of categories minus one.

**Table 5.8** Ordinal-Class Variable Encoding Matrix Using Bathtub Encoding

<b>X</b>	<b>Xa</b>	<b>Xb</b>	<b>Xc</b>
<b>a</b>	-0.63246	-0.63246	-0.63246
<b>b</b>	0.63246	-0.63246	-0.63246
<b>c</b>	0.63246	0.63246	-0.63246
<b>d</b>	0.63246	0.63246	0.63246

**MISSING | MISS= *method(s)***

specifies the method or methods used to handle missing values during clustering and scoring.

In scored data sets, missing values in the original variables are not replaced. When the MISSING= option calls for replacement of missing values in a given variable, a new variable is created. The new variable name is formed by concatenating the prefix IM\_ with the name of the original variable. The new

variable contains all of the nonmissing values of the original variable as well as the replaced values for any cases where the original variable is missing.

The following missing-value methods are available:

#### OMITCASE | OMIT

Any case that has missing values for any of the listed variables is omitted from the cluster analysis. In scoring, cases with missing values are assigned a missing value for the distance and cluster number, and no missing values are replaced. The standardization setting for all variables must be set to STDIZE=NONE when MISSING=OMITCASE is specified for any variable. Statistics from the DMDB catalog are not adjusted for omitted cases.

#### IGNORE

Missing values are ignored during clustering and scoring. For cases with ignored missing values, distance from the cluster seed is computed as follows:

Distance for nonmissing variables \*  $\text{SQRT}(\text{Sum of variances for nonmissing variables} / \text{Sum of variances for all nonrejected variables})$  where the variances are obtained from the DMDB catalog and include the effect of the STDIZE= option setting.

#### CATEGORY | CAT

Missing values are treated as a valid category during clustering and scoring. You cannot use the method MISSING=CATEGORY with the option LEVEL=INTERVAL.

#### MEAN

Missing variable values are replaced with the mean of the known variable values during clustering and scoring.

#### MIDRANGE | MID

Missing values are replaced by the midrange during clustering and scoring.

#### MODE

Missing values are replaced by the mode during clustering and scoring. If there is no unique mode, the mean of all the modes is used. You can use the method MISSING=MODE only with DMDB CLASS variables.

#### MEDIAN | MED

Missing values are replaced by the median value during clustering and scoring. You can use the method MISSING=MEDIAN only with DMDB CLASS variables.

#### *(initial\_method imputation\_method)*

specifies separate methods for handling missing values during cluster initiation (*initial\_method*) and for imputing missing values during scoring (*imputation\_method*). During estimation, missing values are ignored except where otherwise indicated below.

The *initial\_method* can be one of the following:

#### OMITCASE | OMIT

Omit cases with missing values during cluster initialization and estimation, but not during scoring.

#### IGNORE

Ignore missing values during cluster initialization.

#### MEAN

Replace missing values with the mean value during cluster initialization.

#### MIDRANGE | MID

Replace missing values with the midrange during cluster initialization.

**MODE**

Replace missing values with the mode during cluster initialization.

**MEDIAN | MED**

Replace missing values with the median value during cluster initialization.

The *imputation\_method* can be one of the following:

**NEARSEED | NS**

Missing values are replaced with the seed of the nearest cluster. You cannot use the *imputation\_method* NEARSEED with target variables because targets have no seeds.

The following table lists the allowable combinations of **MISSING=** and **LEVEL=**:

**Table 5.9** Allowable Combinations of **MISSING=** and **LEVEL=**

<b>MISSING= method</b>	<b>LEVEL=</b>	<b>LEVEL=</b>	<b>LEVEL=</b>
	<b>NOMINAL</b>	<b>ORDINAL</b>	<b>INTERVAL</b>
<b>IGNORE</b>	yes	yes	yes
<b>MEAN</b>	yes [1]	yes [4]	yes [5]
<b>MEDIAN</b>	no	yes	yes, CLASS only
<b>MIDRANGE</b>	no	no	yes [5]
<b>MODE</b>	yes [2]	yes	yes, CLASS only
<b>OMITCASE</b>	yes	yes	yes

- 1 When a variable is missing, the expanded (imputed and dummy) variables are set to their respective mean values. The imputed value is the category that has the largest proportion in the DMDB data set; that is, it has the same value as the imputed value that would be produced by setting **MISSING=MODE**.
- 2 When a variable is missing, the expanded variables are set to the expanded values of the modal category.
- 3 When a variable is missing, each expanded variable is set to the seed of the cluster that the expanded variable case is assigned to. The imputed value is the category with the largest proportion in the cluster.
- 4 Means are generally not suitable for use with ordinal variables. However, with RANK encoding, the mean is equivalent to the median. Therefore, it is reasonable to use **MISSING=MEAN** with **ORDINAL(RANK)** variables.
- 5 Not recommended for use with CLASS variables, because there might not be a category that corresponds exactly to the computed replacement value. When this problem occurs, the category that is closest to the computed replacement value is used to replace the missing value.

*Note:*

- When a variable is missing, the expanded variables (the dummy and transformed variables) are set to the means of the expanded variables. The imputed value is the category with the largest proportion in the DMDB data set, which is also the imputed value produced by **MISSING=MODE**.
- When a variable is missing, the expanded variables are set to the expanded values of the modal category.

- $\square$  When a variable is missing, the expanded variables are set to the seeds of the expanded variables for the cluster category with the largest proportion in the cluster.
- $\square$  Means are generally not suitable for use with ordinal variables. However, with RANK encoding, the mean is equivalent to the median. Hence, using MISSING=MEAN is reasonable with ORDINAL (RANK) variables.
- $\square$  Not recommended for use with CLASS variables, since there might not be a category that corresponds exactly to the computed replacement value. In such a case, the category that is closest to the computed replacement value is used to replace the missing value.

$\Delta$

**SMOOTH= *number* where *number* > 0.**

specifies the smoothing parameter to be used with IMPUTE=CONDMEAN, MULTMEAN.

The effect of SMOOTH=*s* is to multiply the estimated common within-cluster standard deviation by *s*.

If SMOOTH= is not specified, but INSTAT= is, any smoothing parameters are read from the INSTAT= data set. If neither SMOOTH= nor INSTAT= is specified, then the smoothing parameter is set to any of the following:

- $\square$  For IMPUTE=CONDMEAN, a separate smoothing parameter is estimated for each variable, if SMOOTH= is not specified.
- $\square$  For IMPUTE=MULTMEAN, the default is SMOOTH=2.
- $\square$  For IMPUTE=MULTSTOC, the default is SMOOTH=1.

**UNRECOGNIZED | UNREC= *method***

specifies the method you want to use to handle unrecognized categories (categories that are not defined in the DMDB catalog) during scoring. The UNRECOGNIZED= options apply only to variables that are listed in a CLASS statement that belongs to a DMDB procedure. It does not apply to variables that are specified as LEVEL=INTERVAL in the DMVQ procedure. If an unrecognized category is found in the training data set, training is ended abruptly. The following are valid methods for the UNRECOGNIZED= option:

**OMITCASE | OMIT**

Any case that has unrecognized categories for any of the listed variables is assigned a missing value for the distance and cluster number.

**IGNORE**

Unrecognized categories are ignored.

**MEAN**

Unrecognized categories are replaced by the mean.

**MODE**

Unrecognized categories are replaced by the mode. If there is no unique mode, the mean of all of the modes is used.

**MEDIAN | MED**

Unrecognized categories are replaced by the median.

**NEARSEED | NS**

Unrecognized categories are replaced by the seed of the nearest cluster. You cannot use UNRECOGNIZED=NEARSEED with target variables because targets have no seeds.

---

## TARGET Statement

The TARGET statement specifies the target variable (or variables) and options. Even though target variables are not used during clustering algorithms, when a data set is scored, statistics for target variables are output.

**TARGET** *variable-name(s) / <option(s)>* ;

The TARGET statement uses the same syntax as the INPUT statement.

---

## VQ Statement

The VQ statement specifies vector quantization options.

**VQ** *<option(s)>* ;

### Options

The following options can be used in a VQ statement:

**CLUSNAME=** *variable-name*

specifies the name of the variable that is used to identify clusters.

**CLUSLABEL=** *variable-name | quoted\_string*

specifies the label of the variable that is used to identify clusters.

**DISTNAME=** *variable-name*

specifies the name of the variable that gives the distance from each case to the cluster seed.

**DISTLABEL=** *variable-name | quoted\_string*

specifies the label of the variable that gives the distance from each case to the cluster seed.

**MAXCLUSTERS | MAXCLUS | MAXC | MAX=** *integer*

specifies the maximum number of clusters. The integer must be greater than zero.

---

## SOM Statement

Specifies parameters for the self-organizing map.

**SOM** *<option(s)>* ;

### Options

The following options can be used with the SOM statement:

**CLUSNAME=** *name*

specifies the name of the variable used to identify clusters.

**CLUSLABEL=** *name* | *quoted\_string*

specifies the label of the variable used to identify clusters.

**COLNAME=** *name*

specifies the name of the variable that is used to identify columns in the SOM.

**COLLABEL=** *name* | *quoted\_string*

specifies the label for the variable used to identify columns in the SOM.

**COLUMNS** | **COLS=** *integer*

specifies the number of columns in the SOM. The integer must be greater than zero. In the SOM statement, either the ROWS= or the COLUMNS= option must be specified with a value that is greater than one.

**Default:** 1

**DISTNAME=** *name*

specifies the name of the variable that gives the distance from each case to the cluster seed.

**DISTLABEL=** *name* | *quoted\_string*

specifies the label of the variable that gives the distance from each case to the cluster seed.

**ROWNAME=** *variable-name*

specifies the name of the variable that is used to identify rows in the SOM.

**ROWLABEL=** *name* | *quoted\_string*

specifies the label for the variable used to identify rows in the SOM.

**ROWS=** *integer*

specifies the number of rows in the SOM. The integer must be greater than zero. In the SOM statement, either the ROWS= or the COLUMNS= option must be specified with a value that is greater than one.

**Default:** 1

**SOMNAME=** *name*

specifies the name of the character variable that contains the row and column in the SOM.

**SOMLABEL=** *name* | *quoted\_string*

specifies the label of the character variable that contains the row and column in the SOM.

---

## MAKE Statement

Specifies output data set options.

**MAKE** *<option(s)>* ;

### Options

The following options can be used in the MAKE statement:

**OUTSTAT=** *outstat-data-set-name*

specifies the name of an output data set that contains statistics that are computed from the DMDB catalog. The OUTSTAT data set does not contain any clustering results.

**OUTVAR= *outvar-data-set-name***

specifies the name of the OUTVAR output data set.

The OUTVAR data set contains four observations:

**\_TYPE\_='FORMATTED'**

contains the formatted value of the category that corresponds to the dummy variable for each DMDB class variable with multidimensional encodings. Other variables are blank.

**\_TYPE\_='NORMALIZED'**

contains the normalized value of the category that corresponds to the dummy variable for each DMDB CLASS variable with multidimensional encodings. Other variables are blank.

**\_TYPE\_='TYPE'**

contains information on up to six variable types:

**'ORIGINAL'**

original, untransformed variables.

**'TRANSFORMED'**

transformed (standardized or missing-value replaced) variables.

**'DUMMY'**

dummy variables.

**'IMPUTED'**

imputed variables.

**'MIV'**

missing indicator variables.

**'OMITTED'**

variables that were omitted from clustering because all values are missing, or because there is only one category.

**\_TYPE\_='VARIABLE'**

contains the names of the original variables.

## INITIAL Statement

**Specifies initial seed options.**

**INITIAL** <options> ;

### Options

The following options can be used with the INITIAL statement:

**INITIAL | INIT= *initial-seed-selection-method***

specifies the method to be used to provide initial cluster-seed selection. Cluster-seed selection method descriptions use the following notations:

- r* = number of rows in a SOM
- c* = number of columns in a SOM
- s* = number of seeds =  $r * c$  for a SOM

The following methods are available to select initial cluster seeds:

**FIRST**

selects the first *s* complete cases as initial seeds.

**MACQUEEN**

uses MacQueens' *k*-means algorithm to compute the initial seeds.

**SEPARATE**

selects initial seeds that are well-separated using the partial replacement FASTCLUS algorithm.

**OUTLIER**

selects initial seeds that are very well-separated using the full replacement FASTCLUS algorithm.

**PRINCOMP**

selects initial seeds on an evenly spaced grid in the plane of the first two principal components. If the value of ROWS= is less than or equal to the value of COLUMNS=, the first principal component is oriented to vary with the column number. The second principal component is oriented to vary with the row number. If the value of ROWS= is greater than the value of COLUMNS=, the first principal component is oriented to vary with the row number. The second principal component is oriented to vary with the column number. The maximum absolute value in the grid (along with each component) is the square root of the corresponding eigenvalue.

If you specify VQ instead of SOM, the initial seeds are evenly spaced along the first principal component.

**INSTAT= *DMVQ-OUTSTAT-data-set***

reads a SAS data set that was previously created by the DMVQ procedure using the OUTSTAT= option.

**OUT= *output-data-set-name***

specifies the name of an output data set that contains the original data, plus cluster membership and distance variables.

**OUTMEAN= *output-data-set-name***

specifies the name of an output data set that contains the means of the expanded (imputed and dummy) variables.

**OUTSTAT= *output-data-set-name***

specifies the name of an output data set that contains all of the statistics that are computed by the analysis.

**RADIUS | R= *number where number ≥ 0***

specifies the initial seed radius.

---

## TRAIN Statement

Specifies training options such as number of iterations or steps, convergence criteria, learning rates and steps, neighborhoods, cluster seed write frequencies, and output data sets.

**TRAIN** <*option(s)*> ;

### Options

The following options can be used in the TRAIN statement:

**FCONVERGE | FCONV= *number***

specifies the F-convergence criterion.

**Default:** The default setting for the F-convergence criterion for SOMs is -1.

Otherwise, the default setting is 0.0001.

**JIGGLE= *number* where  $0 \leq \text{number} < 1$** 

specifies the (Zeger et al. [1992]) code-vector jiggling parameter. The FCONVERGE= option does not apply when jiggling is used.

**KERNEL= *integer* where *integer*  $\geq 0$** 

specifies the kernel shape.

**Table 5.10** Kernel Shapes by KERNEL= Value

Value	Kernel Shape
0	uniform
1	Epanechnikov
2	biweight
3	triweight

**Default:** KERNEL=1

**KMETRIC= *integer* where *integer*  $\geq 0$** 

specifies the L<sub>p</sub> kernel metric.

**Table 5.11** L<sub>p</sub> Kernel Metrics by KMETRIC= Value

Value	L <sub>p</sub> Kernel Metric
0	max
1	cityblock
2	Euclidean

**LEARN= *number* where  $0 < \text{number} \leq 1$** 

specifies the learning rate for Kohonen training. The value that is specified for LEARN= becomes the default value for LEARNINITIAL= and LEARNFINAL=.

**LEARNINITIAL | LI= *number* where  $0 < \text{number} \leq 1$** 

specifies the initial learning rate for Kohonen training.

**Default:** If LEARN= is specified, the default value for LEARNINITIAL= is the value of LEARN=. If LEARN= is not specified, the default value for LEARNINITIAL= is 0.9 for a SOM, or 0.5 for VQ.

**LEARNFINAL | LF= *number* where  $0 < \text{number} \leq 1$** 

specifies the final learning rate for Kohonen training.

**Default:** If LEARN= is specified, the default value for LEARNFINAL= is the value of LEARN=. If LEARN= is not specified, the default value for LEARNFINAL= is 0.02.

**LEARNSTEPS | LS= *integer* where *integer*  $\geq 1$** 

specifies the step number at which the learning rate reaches the value of LEARNFINAL.

**Default:** 1000

**MAXITER | MAXIT= *integer* where *integer*  $\geq 0$**

specifies the maximum number of training iterations allowed.

**Default:** For TECH=KOHONEN, the default setting is 100. Otherwise, the default setting is 10.

**MAXSTEPS | MAXSTEP= *integer where integer*  $\geq 0$**

specifies the maximum number of steps allowed for Kohonen training. Kohonen training is ended when either the MAXSTEPS= or MAXITER= limit is reached.

**Default:** MAX(1000, LEARNSTEPS, 500\*ROWS\*COLUMNS)

**NEIGHRESET | NR= *integer where integer*  $\geq 1$**

specifies the number of steps after which the neighborhood size and kernel are reset.

**Default:** 1000

**NEIGHBORHOOD | NEIGH= *number where number*  $\geq 0$**

specifies the neighborhood size. The value that is specified for NEIGHBORHOOD= becomes the default value for NEIGHINITIAL= and NEIGHFINAL=.

**NEIGHINITIAL | NI= *number where number*  $\geq 0$**

specifies the initial neighborhood size.

**Default:** If NEIGHBORHOOD= is specified, the default value for NEIGHINITIAL= is the value of NEIGHBORHOOD=. If NEIGHBORHOOD= is not specified, the default value for NEIGHINITIAL= is MAX(5, MAX(ROWS,COLUMNS)/2).

**NEIGHFINAL | NF= *number where number*  $\geq 0$**

specifies the final neighborhood size.

**Default:** If NEIGHBORHOOD= is specified, the default value for NEIGHFINAL= is the value of NEIGHBORHOOD=. If NEIGHBORHOOD= is not specified, the default value for NEIGHFINAL= is 0.

**NEIGHSTEPS | NS= *integer where integer*  $\geq 1$**

specifies the step number at which the neighborhood size reaches the value of NEIGHFINAL= during Kohonen SOM training.

**Default:** 1000

**NEIGHITER | NIT= *integer where integer*  $\geq 0$**

specifies the iteration number at which the neighborhood size reaches the value of NEIGHFINAL= during batch SOM training.

**Default:** MIN( 10, MAX( 3, MAXITER/2) )

**OUT= *SAS-data-set***

specifies an output data set that contains all of the original data plus cluster membership and distance variables.

**OUTMEAN= *SAS-data-set***

specifies the output data set that contains the means of the expanded (imputed and dummy) variables.

**OUTSEED= *SAS-data-set***

specifies the name of the output data set that contains cluster seeds.

**OUTSTAT= *SAS-data-set***

specifies an output data set that contains all of the statistics that were computed in the analysis. The OUTSTAT= data set provides the following statistics:

DMDB\_FREQ

the sum of frequencies (observations) for nonmissing values of each variable.

DMDB\_WEIGHT

the sum of weights for nonmissing values of each variable.

DMDB\_MEAN

the mean value statistic that is computed from all nonmissing values of each variable.

**DMDB\_STD**

the standard deviation statistic computed from all nonmissing values of each variable.

**DMDB\_MIN**

the minimum of all nonmissing values for each variable.

**DMDB\_MAX**

the maximum of all nonmissing values for each variable.

**LOCATION**

the **STDIZE**= location measure that incorporates replaced missing values.

**SCALE**

the **STDIZE**= scale measure that incorporates replaced missing values.

**CRITERION**

the clustering criterion value that is used for all variables, as calculated by the Clustering Method property specified in the Cluster node Properties Panel.

**PSEUDO\_F**

the Pseudo F statistic, summarized across all input variables  $\left[ \frac{R^2}{c-1} \right] / \left[ \frac{(1-R^2)}{n-c} \right]$  where  $R^2$  is the observed overall  $R^2$ ,  $c$  is the number of clusters, and  $n$  is the number of observations.

**ERSQ**

the approximate expected value of the squared multiple correlation  $R$ , which is the proportion of variance accounted for by the clusters, under a uniform null hypothesis. If the number of clusters is greater than one-fifth of the number of observations, **ERSQ** and **CCC** are given missing values.

**CCC**

the cubic clustering criterion value, which is useful in determining the number of clusters in the data. **CCC** values that are greater than 2 or 3 indicate good clusters. Values between 0 and 2 indicate potential clusters (but they should be considered with caution). Large negative values can indicate outliers. If the number of clusters is greater than one-fifth of the number of observations, **CCC** and **ERSQ** are given missing values.

**TOTAL\_STD**

the total standard deviation of each variable.

**WITHIN\_STD**

the pooled within-cluster standard deviation of each variable.

**RSQ**

the squared multiple correlation  $R$ , which is the proportion of variance accounted for by the clusters.

**RSQ\_RATIO**

the ratio of between-cluster variance to within-cluster variance  $(R^2/(1-R^2))$ .

**SEED**

seeds for each cluster and variable.

**CLUS\_MEAN**

the mean of each variable within each cluster.

**CLUS\_STD**

the standard deviations of each variable within each cluster.

**CLUS\_MIN**

the minimum of each variable within each cluster.

**CLUS\_MAX**

the maximum of each variable within each cluster.

**CLUS\_FREQ**

the sum of frequencies for nonmissing values of each variable within each cluster.

**SEEDITER= *integer* where *integer*  $\geq 1$** 

specifies the frequency with which seeds are written to the OUTSEED= data set during training. If SEEDITER=10, then seeds are written to the OUTSEED= data set every 10th iteration. If you specify SEEDITER=0, only the final seeds are written.

**Default:** 0**TECH= *training-technique***

specifies the training technique to be used. The following are the available training techniques:

**FORGY | LLOYD**

specifies either the FORGY or the generalized Lloyd I training method.

**JANCEY**

specifies Jancey training.

**KOHONEN**

specifies Kohonen training.

**NWSOM | NW**

specifies Nadaraya-Watson batch SOM training.

**LLSOM | LL**

specifies local-linear batch SOM training.

**XCONVERGE | XCONV= *number* where *number*  $\geq 0$** 

specifies the X-convergence criterion.

**Default:** 0.0001

---

## CODE Statement

If you want to score a data set, you can use a CODE statement to write SAS DATA step code to a file or catalog entry. This code can then be included into a DATA step that reads (using a SET statement) the data set to be scored.

**CODE FILE=** *codefile-name* <*code-option(s)*> ;

### Required Arguments

The following argument is required by the CODE statement:

**FILE=** *codefile-name*

specifies where to write the code. When enclosed in a quoted string, the FILE= argument specifies the path for writing the code to an external file. Here is an example:

```
FILE="c:\mydir\scorecode.sas".
```

FILE= can also use unquoted SAS filenames of no more than eight characters. If the filename is assigned as a fileref in a FILENAME statement, the file specified in

the FILENAME statement is opened. The special filerefs LOG and PRING are always assigned. If the specified name is not an assigned fileref, the specified value is concatenated with a .txt extension before opening. For example, if FOO is not an assigned fileref, FILE=FOO would cause FOO.txt to be opened. If the name has more than eight characters, an error message is printed.

## Options

### CATALOG | CAT | C= *library.catalog.entry.type*

specifies where to write the code in the form of library.catalog.entry.type. The compound name can have one to four levels. The default library is determined by the SAS system option USER=, usually WORK. The default entry is SASCODE, and the default type is SOURCE.

### DUMMIES | NODUMMIES

Use DUMMIES | NODUMMIES to specify whether to keep dummy variables, standardized variables, or other transformed variables in the data set.

**Default:** NODUMMIES

### ERROR | NOERROR

Specifies whether the error function E\_\* is to be computed.

### FORMAT= *format*

Use FORMAT= to format weights or other numeric values that don't have a format from the input data set.

**Default:** BEST20.

### GROUP= *group identifier*

Use GROUP= to specify the group identifier for group processing. The identifier should be a valid SAS name of no more than 16 characters, which is used to construct array names and statement labels in the generated code.

### LINESIZE | LS=*integer where integer* $\geq 0$

Use LINESIZE= to specify the line size for generated code. The permissible range is 64 to 254.

**Default:** 72

### LOOKUP= *lookup-algorithm*

Use LOOKUP= to specify the algorithm that you want to use to look up CLASS levels. The following lookup algorithms are available for use with LOOKUP:

#### SELECT

SELECT uses a SELECT statement to perform lookups. SELECT can be slow if there are large numbers of categories.

#### LINEAR

LINEAR uses a linear search with IF statements with categories in the order specified in the DMDB catalog. This process is slow if there are many categories.

#### LINFREQ

LINFREQ uses a linear search with IF statements with categories in descending order of frequency as given in the DMDB catalog. This process is fast if the distribution of class frequencies is highly uneven, but is slow if there are many categories, all with approximately equal frequencies.

#### BINARY

uses a binary search. Although fast, this process can produce incorrect results depending on which character-encoding system the machine uses. Some characters

collate in different orders in ASCII and EBCDIC. So if you generate the code on an ASCII machine and execute the code on an EBCDIC machine—or vice versa, PROC CSCORE cannot translate the code.

#### AUTO

AUTO selects the fastest method for each variable that is portable across ASCII and EBCDIC.

**Default:** AUTO

#### PMML | XML

produces scoring code in Predictive Modeling Markup Language, an XML-based standard for representing data mining results. For more information, see the PMML Support in Enterprise Miner section in the Enterprise Miner 5.3 Java Help.

#### RESIDUAL | NORESIDUAL

Use RESIDUAL to generate residual values for the variables R\_\*, F\_\*, CL\_\*, CP\_\*, BL\_\*, BP\_\*, and ROI\_\*. If you request code for residuals and then score a data set that does not contain target values, the residuals will have missing values.

**Default:** NORESIDUAL

---

## Details: DMVQ Procedure

---

### Comparing DMVQ and FASTCLUS Procedures

The following section discusses some of the operational differences that exist between the DMVQ and FASTCLUS procedures.

- FASTCLUS does not standardize by default. DMVQ uses STDIZE=STD.
- To get the FASTCLUS procedure's default VQ training configuration using the DMVQ procedure, use the TECH=FORGY option.
- To get the FASTCLUS procedure's default SOM training configuration using the DMVQ procedure, you need to submit three TRAIN statements with TECH=KOHONEN, TECH=NWSOM, and TECH=LLSOM, respectively.

**Table 5.12** FASTCLUS Statements and Equivalent DMVQ Settings

FASTCLUS Statement	DMVQ Statement	DMVQ Option
VAR	INPUT	LEVEL=INTERVAL
ID	not Applicable	
FREQ	from DMDB	
WEIGHT	from DMDB	
BY	not Applicable	

Table 5.13 FASTCLUS Options and Equivalent DMVQ Options

<b>FASTCLUS Option</b>	<b>DMVQ Statement</b>	<b>DMVQ Option</b>
<b>BINS=</b>	not applicable	
<b>CLUSLABEL=</b>	VQ	CLUSLABEL=
<b>CLUSTER=</b>	VQ	CLUSNAME
<b>COLUMNS=</b>	SOM	COLUMNS=
<b>COLLABEL=</b>	SOM	COLLABEL=
<b>CONVERGE=</b>	TRAIN	XCONVERGE=
<b>DATA=</b>	PROC DMVQ	DATA=
<b>DELETE=</b>	not applicable	
<b>DISTANCE</b>	not applicable	
<b>DRIFT REPLACE=NONE</b>	INITIAL	MACQUEEN
<b>HC=</b>	not applicable	
<b>HP=</b>	not applicable	
<b>IMPUTE=NEARSEED</b>	PROC DMVQ	MISSING=(IGNORE NEARSEED)
<b>INSTAT=</b>	INITIAL	INSTAT=
<b>IRLS</b>	not applicable	
<b>KERNEL=</b>	TRAIN	KERNEL=
<b>KMETRIC=</b>	TRAIN	KMETRIC=
<b>KOHONEN</b>	TRAIN	TECH=KOHONEN
<b>KOITER=</b>	TRAIN	TECH=KOHONEN MAXITER=
<b>KOCONV=</b>	TRAIN	TECH=KOHONEN CONVERGE=
<b>LEAST=</b>	not applicable	
<b>LEARN=</b>	TRAIN	LEARN=
<b>LEARNINITIAL=</b>	TRAIN	LEARNINITIAL=
<b>LEARNFINAL=</b>	TRAIN	LEARNFINAL=
<b>LEARNSTEPS=</b>	TRAIN	LEARNSTEPS=
<b>LIST</b>	not applicable	
<b>LLSOM=</b>	TRAIN	TECH=LLSOM
<b>LLITER=</b>	TRAIN	TECH=LLSOM MAXITER=
<b>LLCONV=</b>	TRAIN	TECH=LLSOM CONVERGE=
<b>MAXCLUSTERS=</b>	VQ	MAXCLUSTERS=
<b>MAXITER=</b>	TRAIN	MAXITER=
<b>MEAN=</b>	SCORE/SAVE	OUTMEAN=
<b>NEIGHBORHOOD=</b>	TRAIN	NEIGHBORHOOD=
<b>NEIGHINITIAL=</b>	TRAIN	NEIGHINITIAL=
<b>NEIGHFINAL=</b>	TRAIN	NEIGHFINAL=

<b>FASTCLUS</b> <b>Option</b>	<b>DMVQ</b> <b>Statement</b>	<b>DMVQ</b> <b>Option</b>
<b>NEIGHSTEPS=</b>	TRAIN	NEIGHSTEPS=
<b>NEIGHRESET=</b>	TRAIN	NEIGHRESET=
<b>NOMISS</b>	INPUT	MISSING=OMITCASE
<b>NOPRINT</b>	not applicable	
<b>NWSOM</b>	TRAIN	TECH=NWSOM
<b>NWITER=</b>	TRAIN	TECH=NWSOM MAXITER=
<b>NWCONV=</b>	TRAIN	TECH=NWSOM CONVERGE=
<b>OUT=</b>	SCORE	OUT=
<b>OUTITER</b>	TRAIN	SEEDITER= OUTSEED=
<b>OUTSEED=</b>	SCORE/SAVE	OUTMEAN=
<b>OUTSTAT</b>	SCORE/SAVE	OUTSTAT=
<b>RADIUS=</b>	INITIAL	RADIUS=
<b>REPLACE=NONE</b>	INITIAL	FIRST
<b>REPLACE=PART</b>	INITIAL	SARLE
<b>REPLACE=FULL</b>	INITIAL	OUTLIER
<b>REPLACE=PC</b>	INITIAL	PRINCOMP
<b>ROWLABEL=</b>	SOM	ROWLABEL=
<b>ROWS=</b>	SOM	ROWS=
<b>SEED=</b>	INITIAL	DATA= (original vars) or INSTAT= (expanded vars)
<b>SHORT</b>	not applicable	
<b>SOMIDLABEL</b>	SOM	SOMLABEL=
<b>STD=</b>	PROC DMVQ	STDIZE=
<b>STRICT</b>	not applicable	
<b>SUMMARY</b>	not applicable	
<b>VARDEF</b>	DMDB	

---

## References

Zeger, K., J. Vaisey, and A. Gersho. 1992. "Globally Optimal Vector Quantizer Design by Stochastic Relaxation," *IEEE Transactions on Signal Processing*, 40, 310-322.