



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 DMZIP Procedure**

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).

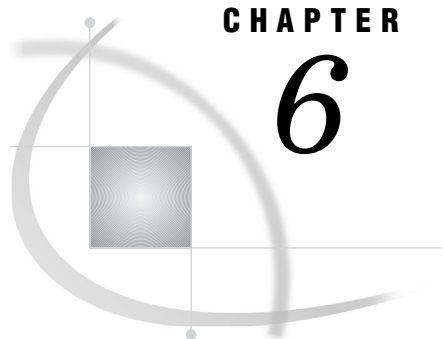
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

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

## 6

## The DMZIP Procedure (Experimental)

<i>Overview: DMZIP Procedure</i>	87
<i>Syntax: DMZIP Procedure</i>	87
<i>PROC DMZIP Statement</i>	88
<i>INPUT Statement</i>	88
<i>MAKE Statement</i>	93
<i>SCORE Statement</i>	93
<i>CODE Statement</i>	94
<i>Example: DMZIP Procedure</i>	95
<i>Example 1: PROC DMZIP</i>	95

### Overview: DMZIP Procedure

The DMZIP procedure is for testing the zip\* subsystem. It uses the SCORE statement to produce an output data set. The output data set contains the original data plus dummy variables, standardized variables, and transformed variables. You must run PROC DMDB to create a DMDB catalog before you run PROC DMZIP.

### Syntax: DMZIP Procedure

```

PROC DMZIP <option(s)>;
INPUT variables</option(s)>;
TARGET variable</option(s)>;
MAKE <option(s)>;
SCORE <option(s)>;
CODE <option(s)>;

```

The DMZIP procedure is interactive. Each statement is executed when it is submitted. The procedure does not terminate until you submit a QUIT statement or a new step. If an ERROR message is printed, submit a RUN statement to resume normal processing by PROC DMZIP.

To use PROC DMZIP:

- 1 Submit a PROC DMZIP statement.
- 2 Submit one or more INPUT or TARGET statements.

- 3 Submit a MAKE statement.
- 4 Submit zero or more SCORE statements.
- 5 Return to step (2) or QUIT.

*Note:* There is no FREQ statement. To use a FREQ variable, specify it with PROC DMDB.  $\Delta$

The following option can be used in any statement:

MESSAGE | MSG=SILENT | S | TERSE | T | VERBOSE | V  
The amount of information written to the log.

---

## PROC DMZIP Statement

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

### Options

The following option is required in the PROC DMZIP statement:

**DMDBCAT=***screenspec*

The DMDB catalog.

The following options can be used in the PROC DMZIP statement:

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

Specifies the input SAS data set containing the training data. The DATA= option is not required in the PROC statement.

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

**NOMINAL | NOM=GLM | REFERENCE | REF | DEVIATION | DEV**

The encoding used for nominal variables when no encoding is specified in a LEVEL= option. See the LEVEL= option for more information.

**Default:** GLM

**ORDINAL | ORD=COUNT | INDEX | RANK | THERMOMETER | THERM | THER | BATHTUB | BATH**

The encoding used for ordinal variables when no encoding is specified in a LEVEL= option. See the LEVEL= option for more information.

**Default:** RANK

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

Specifies the test data set.

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

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

Specifies the validation data set.

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

---

## INPUT Statement

**INPUT** *variables* </option(s)>;

The variables must have been specified in a CLASS or VAR statement in PROC DMDB. The variables can be character or numeric depending on the LEVEL= option as described below.

**LEVEL=NOMINAL | NOM | ORDINAL | ORD | INTERVAL | INT <(encoding)>**

The measurement level.

The combinations indicated by "Yes" in the following table are allowed.

DMDB Statement	Variable Type	Measurement Level		
		nominal	ordinal	interval
VAR	numeric	no	no	yes
CLASS	numeric	yes	yes	yes
	character	yes	yes	no

For nominal or ordinal variables, the measurement level can be followed by the encoding method in parentheses as an option. The following encoding methods can be used:

- LEVEL=NOMINAL(GLM)
- LEVEL=NOMINAL(REFERENCE | REF)
- LEVEL=NOMINAL(DEVIATION | DEV)
- LEVEL=ORDINAL(RANK)
- LEVEL=ORDINAL(INDEX)
- LEVEL=ORDINAL(THERMOMETER | THER)
- LEVEL=ORDINAL(BATHTUB | BATH)

Here are examples of different LEVEL= options:

```
data;
  input x $ @@;
  cards;
  d c b a d c b d c d
  ;
  LEVEL=NOMINAL (GLM)

  X      Xa   Xb   Xc   Xd
  -----
  a      1    0    0    0
  b      0    1    0    0
  c      0    0    1    0
  d      0    0    0    1
```

```
LEVEL=NOMINAL (REFERENCE)

  X      Xa   Xb   Xc
  -----
  a      1    0    0
  b      0    1    0
  c      0    0    1
  d      0    0    0
```

LEVEL=NOMINAL (DEVIATION)

X	Xa	Xb	Xc
a	1	0	0
b	0	1	0
c	0	0	1
d	-1	-1	-1

LEVEL=ORDINAL (RANK)

X	T_X
a	0.05
b	0.20
c	0.45
d	0.80

LEVEL=ORDINAL (INDEX)

X	T_X
a	0
b	0.333333
c	0.666667
d	1

LEVEL=ORDINAL (COUNT)

X	T_X
a	0
b	1
c	2
d	3

LEVEL=ORDINAL (THERMOMETER)

X	Xa	Xb	Xc
a	0	0	0
b	1	0	0
c	1	1	0
d	1	1	1

LEVEL=ORDINAL (BATHTUB)

X	Xa	Xb	Xc
a	-0.63246	-0.63246	-0.63246
b	0.63246	-0.63246	-0.63246
c	0.63246	0.63246	-0.63246
d	0.63246	0.63246	0.63246

**Default:** The default for DMDB VAR variables is LEVEL=INTERVAL. The default for DMDB CLASS variables is LEVEL=NOMINAL.

**MISSING | MISS=OMITCASE | OMIT | IGNORE | CATEGORY | CAT | MEAN | MIDRANGE | MID | MODE | MEDIAN | MED**

The method for handling missing values.

**OMITCASE | OMIT** Any case with any missing values for any of the listed variables is omitted from the OUT= data set. For the CODE statement, MISSING=OMITCASE is treated like MISSING=IGNORE. STDIZE=NONE must be used for all variables when MISSING=OMITCASE is specified for any variable. Statistics from the DMDB catalog are not adjusted for omitted cases.

**IGNORE** Missing values are ignored.

**CATEGORY | CAT** Missing values are treated as a valid category. This option cannot be used with LEVEL=INTERVAL.

**MEAN** Missing values are replaced by the mean.

**MIDRANGE | MID** Missing values are replaced by the midrange.

**MODE** Missing values are replaced by the mode. If there is no unique mode, the mean of all the modes is used. This option can be used only with DMDB CLASS variables.

**MEDIAN | MED** Missing values are replaced by the median. This option can be used only with DMDB CLASS variables.

**Default:** MISSING=IGNORE

**STANDARDIZE | STDIZE | STD=<std\_value>**

<std\_value> can be any of the following:

- NONE | NON | NO
- MEAN
- MEDIAN | MED
- SUM
- EUCLEN | EUC
- USTD
- STD
- RANGE | RAN
- MIDRANGE | MID
- MAXABS | MAX
- IQR
- MAD
- ABW<(p)> where  $p > 0$
- AHUBER<(p)> where  $p > 0$
- AWAVE<(p)> where  $p > 0$
- AGK(p) where  $0 < p < 1$
- SPACING(p) where  $0 < p < 1$
- LEASTP(p) | LP(p) | L(p) where  $p \geq 1$

Only STDIZE=NONE can be used when MISSING=OMITCASE is specified for any variable.

The method for standardizing expanded variables is as follows:

<b>STDIZE= option</b>	<b>Subtract</b>	<b>Divide By</b>
NONE   NON   NO	0	1
MEAN	mean	1
SUM	0	sum * dimension
EUCLEN   EUC	0	Sqrt(USS) * SQRT(dimension)
USTD	0	Sqrt(USS/N) * SQRT(dimension)
STD	mean	standard deviation * SQRT(dimension)
RANGE   RAN	minimum	range
MIDRANGE   MID	midrange	half-range
MAXABS   MAX	0	maximum absolute value

The following STDIZE= options work only for DMDB CLASS variables:

<b>STDIZE= option</b>	<b>Subtract</b>	<b>Divide By</b>
MEDIAN   MED	median	1
IQR	median	interquartile range * SQRT(dimension)
MAD	median	median absolute deviation from the median * SQRT(dimension)
ABW	biweight m est.	biweight A estimate * SQRT(dimension)
AHUBER	Huber m est.	Huber A estimate * SQRT(dimension)
AGK	mean	AGK estimate * SQRT(dimension)
SPACING	mid min-spacing	minimum spacing * SQRT(dimension)
LEASTP	L <sub>p</sub> location	L <sub>p</sub> scale * dimension**(1/p)

**Default:** STDIZE=STD

#### **UNRECOGNIZED | UNREC=OMITCASE | OMIT | IGNORE | MEAN | MODE | MEDIAN | MED**

The method for handling unrecognized categories (that is, categories that are not defined in the DMDB catalog) during scoring. The UNRECOGNIZED= options apply only to variables listed in a CLASS statement under PROC DMDB, and does not apply to variables specified as LEVEL=INTERVAL in PROC DMZIP.

**OMITCASE | OMIT** Any case with any unrecognized categories for any of the listed variables is omitted from the OUT= data set. For the CODE statement, UNRECOGNIZED=OMITCASE is treated like UNRECOGNIZED=IGNORE.

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 the modes is used.
MEDIAN   MED	Unrecognized categories are replaced by the median.
<b>Default:</b>	UNRECOGNIZED=OMIT

---

## MAKE Statement

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

### **OUTVAR=**data\_set

An output data set that contains a `_TYPE_` variable and all the expanded variables, all of which are character. There are three observations:

- `_TYPE_='VARIABLE'` contains the name of the original variable that corresponds to the expanded variable.
- `_TYPE_='FORMATTED'` contains the formatted value of the category that corresponds to the expanded variable for DMDB CLASS variables with multidimensional encodings. For DMDB CLASS variables with unidimensional encodings and for DMDB VAR variables, the value is blank.
- `_TYPE_='NORMALIZED'` contains the normalized value of the category that corresponds to the expanded variable for DMDB CLASS variables with multidimensional encodings. For DMDB CLASS variables with unidimensional encodings and for DMDB VAR variables, the value is blank.

---

## SCORE Statement

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

The following options can be used in the SCORE statement:

### **DATA=**data\_set

The data set to be scored, containing the input variables. If `DATA=` is not specified in the SCORE statement, the `DATA=` data set from the PROC statement is used. You must specify `DATA=` in the SCORE statement or PROC statement or both.

### **OUT=**data\_set

An output data set that contains the original data plus dummy variables, standardized variables, and transformed variables.

For the GLM, reference, deviation, thermometer, and bathtub encodings, dummy variables are created for each dimension and named by concatenating the original variable name with the category. Standardization and missing-value replacement are done in accordance with the `STDIZE=` and `MISSING=` options.

Otherwise, for any variable that is standardized, one transformed variable is created and named by concatenating "S\_" with the original variable name. Missing-value replacement is done in accordance with the `MISSING=` option.

Otherwise, for the rank and index encodings and for variables in which missing values are replaced, one transformed variable is created and named by concatenating "T\_" with the original variable name.

## CODE Statement

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

The `CODE` statement generates SAS DATA step code that generally mimics the computations done by the `SCORE` statement. However, by default, the code contains no reference to the target variable (to avoid notes or warnings that might confuse the user). Only if the `RESIDUAL` or `ERROR` option is specified should the code compute values that depend on the target variable. Some examples are the `R_*`, `E_*`, `F_*`, `CL_*`, `CP_*`, `BL_*`, `BP_*`, or `ROI_*` variables.

The recommended options for the `CODE` statement include the following:

### **CATALOG=screenspec**

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

### **DUMMIES|NODUMMIES**

Whether to keep dummy variables, standardized variables, or other transformed variables in the data set.

### **ERROR**

To compute the error function (`E_*`).

### **FILE=filename**

Where to write the code. This can be either of the following:

- A quoted string, the value of which is the name (including the extension, if any) of the file to be opened.
- An unquoted SAS name of no more than eight characters. If this name has been assigned as a fileref in a `FILENAME` statement, the file specified in the `FILENAME` statement is opened. The special filerefs `LOG` and `PRINT` are always assigned. If the specified name is not an assigned fileref, the specified value is concatenated with the extension ".txt" before opening. For example, if `FOO` is not an assigned fileref, `FILE=FOO` will cause `FOO.txt` to be opened. If the name has more than eight characters, an error message is printed.

### **FORMAT=format**

To format weights or other numeric values that don't have a format from the input data set.

**Default:** `BEST20`.

### **GROUP=name**

To support group processing. This name should be a valid SAS name of no more than 16 characters. It is used to construct array names and statement labels in the generated code.

### **LINESIZE|LS=int**

The size of a line for generated code. The permissible range is 64 to 254.

**Default:** 72

**LOOKUP=SELECT|LINEAR|LINFREQ|BINARY**

Specifies the algorithm for looking up CLASS levels:

SELECT	uses a SELECT statement.
LINEAR	uses a linear search with IF statements. Categories are in the order specified in the DMDB catalog. This process is slow if there are many categories.
LINFREQ	uses a linear search with IF statements. Categories are 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. This process is fast, but can produce incorrect results if you generate the code on an ASCII machine and execute the code on an EBCDIC machine or vice versa, and the normalized category values contain characters that collate in different orders on ASCII and EBCDIC. PROC CSCORE is unable to translate this code.
AUTO	selects the fastest method for each variable that is portable across ASCII and EBCDIC.

You cannot specify both FILE= and CATALOG=. If you specify neither, the code is written to the SAS log.

It is highly desirable for the data set produced by executing the generated code to match the data set that is produced by the SCORE statement as closely as possible so that PROC COMPARE will run cleanly.

**RESIDUAL**

To compute R\_\*, F\_\*, CL\_\*, CP\_\*, BL\_\*, BP\_\*, ROI\_\*.

---

## Example: DMZIP Procedure

---

### Example 1: PROC DMZIP

```
data nondm;
  input abc $ x z;
cards;
b 333 .
c 1 1
a . 2
. 22 3
b .A 4
c 22 5
. 1 6
. . 7
c 1 .
```

```

;
proc dmdb batch data=nondm out=dm dmdbcat=dm;
  class abc x;
  var z;
run;

proc dmzip data=dm dmdbcat=dm;
  input abc x / level=nominal stdize=no;
  input z / level=interval stdize=std;
  make;
  score out=out;
quit;
data both;
  merge out(rename=(abc=dmdb_abc x=dmdb_x z=dmdb_z)) nondm;
run;
proc print data=both;
run;

```

## Output

OBS	abca	abcb	abcc	x1	x22	x333	T_z	dmdb_z	dmdb_abc	dmdb_x	abc	x	z
1	0	1	0	0	0	1	.	.	0	0	b	333	.
2	0	0	1	1	0	0	-1.38873	1	1	1	c	1	1
3	1	0	0	.	.	.	-0.92582	2	2	2	a	.	2
4	.	.	.	0	1	0	-0.46291	3	3	3	.	22	3
5	0	1	0	.	.	.	0.00000	4	0	2	b	A	4
6	0	0	1	0	1	0	0.46291	5	1	3	c	22	5
7	.	.	.	1	0	0	0.92582	6	3	1	.	1	6
8	.	.	.	.	.	.	1.38873	7	3	2	.	7	.
9	0	0	1	1	0	0	.	.	1	1	c	1	.