The %MktLab Macro

Introduction

The %MktLab autocall macro processes an experimental design (usually created by the %MktEx macro) and assigns the final variable names and levels.

%MktLab Macro Syntax

%MktLab(< *optional arguments* >**)**

Optional Arguments

CFILL=*character-string*

specifies the fill value in the KEY= data set for character variables. For more information about fill values, see the NFILL= argument. By default, CFILL=' '.

DATA=*SAS-data-set*

specifies the input data set that contains the experimental design (usually created by the %MktEx macro). By default, DATA=Randomized. The factor levels in the DATA= data set must be consecutive integers, beginning with 1.

DOLIST=*do-list*

specifies the new values by using a do-list syntax (*n* to *m* <by *p*>); for example, **DOLIST=1 to 10** or **DOLIST=0 to 9**. You specify the levels for the largest number of levels for asymmetric designs (not all factors have the same levels). For example, with 2-level and 3-level factors and **DOLIST= 0 to 2**, the 2-level factors are assigned levels 0 and 1, and the 3-level factors are assigned levels 0, 1, and 2. Do not specify both the VALUES= and DOLIST= arguments. By default, when the KEY=, VALUES=, and DOLIST= arguments are not all specified, the default value list comes from **DOLIST=1 to 100**.

INT=*variable-list*

specifies the name of an intercept variable (column of ones) to be added to the OUT= data set. You can also specify a variable list instead of a variable name if you want to make a list of variables that have values that are all equal to 1. This can be useful for creating flag variables for generic choice models when the design is to be used as a candidate set for the %ChoicEff macro.

KEY=*SAS-data-set*

specifies the input data set that contains the key to recoding the design. This data set is created for you when you specify the VALUES= or DOLIST= argument. When the KEY=, VALUES=, and DOLIST= arguments are not all specified, the default value list comes from **DOLIST=1 to 100**.

LABELS=*macro-name*

specifies the name of a macro that provides labels, formats, or other information to the KEY= data set. You specify just the macro name without a percent sign (%) in the LABELS= argument.

It is easier to specify the STATEMENTS= argument for a simple format specification. For more complicated specifications, you can use LABELS=. The following statements demonstrate how you might use this argument:

```
%mktex(3 ** 4, n=18, seed=205)

%macro labs;
   label x1 = 'Sploosh' x2 = 'Plumbob'
         x3 = 'Platter' x4 = 'Moosey';
   format x1-x4 dollar5.2;
%mend;

%mktlab(data=randomized, VALUES=1.49 1.99 2.49, LABELS=labs)

proc print label;
run;
```

The first part of the design is as follows:

Obs	Sploosh	Plumbob	Platter	Moosey
1	$2.49	$2.49	$2.49	$1.49
2	$2.49	$2.49	$1.99	$1.99
3	$1.49	$1.49	$1.49	$1.49
4	$1.99	$1.99	$2.49	$2.49
.				
.				
.				

NFILL=*number*

specifies the fill value in the KEY= data set for numeric variables. For example, when the maximum number of levels is three, the last value in the KEY= data set for numeric 2-level factors should match the value that is specified in the NFILL= argument, which by default is ordinary missing. If the macro tries to access one of these values, it displays a warning. If you want ordinary missing (.) to be a legitimate level, specify a different NFILL= argument value and use it for the extra places in the KEY= data set.

OPTIONS=*options-list*

specifies binary options. By default, none of these options are specified. You can specify the following values:

NOPRINT

suppresses the display of the variable mappings.

OUT=*SAS-data-set*
specifies the output data set that contains the final, recoded design. By default, OUT=Final.

PREFIX=*variable-prefix*
specifies a prefix for naming variables when the VALUES= argument is specified. For example, PREFIX=Var creates the variables Var1, Var2, and so on. By default, the variables are X1, X2, and so on. This argument is ignored when the VARS= argument is specified.

STATEMENTS=*SAS-code*
is an alternative to the LABELS= argument that you can use to add extra statements to the KEY= data set. For a simple format specification, it is easier to use the STATEMENTS= argument. For more complicated specifications, use the LABELS= argument. The following statements demonstrate how you might use this argument:

```
%mktex(3 ** 4, n=18, seed=205)

%mktlab(data=randomized, VALUES=1.49 1.99 2.49,
        vars=Sploosh Plumbob Platter Moosey,
        STATEMENTS=format Sploosh Plumbob Platter Moosey dollar5.2)
proc print data=final(obs=4);
run;
```

Figure 1 shows the first four observations of the design.

Figure 1 First Four Observations of Final Data Set

Obs	Sploosh	Plumbob	Platter	Moosey
1	$2.49	$1.49	$2.49	$1.49
2	$2.49	$1.49	$1.49	$2.49
3	$1.49	$1.99	$2.49	$1.99
4	$1.99	$2.49	$1.99	$1.49

VALUES=*value-list*
specifies the new values for all the variables. If all the variables are to have the same value, it is easier to specify the VALUES= or DOLIST= argument than the KEY= argument. When you specify the VALUES= argument, the KEY= data set is created for you. Specify a list of levels, separated by blanks. If your levels contain blanks, separate them with two blanks. You specify the levels for the largest number of levels for asymmetric designs (not all factors have the same levels). For example, for 2-level and 3-level factors and VALUES=a b c, the 2-level factors are assigned levels `'a'` and `'b'`, and the 3-level factors are assigned levels `'a'`, `'b'`, and `'c'`. Do not specify both the VALUES= and DOLIST= arguments. By default, when the KEY=, VALUES=, and DOLIST= arguments are not all specified, the default value list comes from **`DOLIST=1 to 100`**.

VARS=*variable-list*
specifies a list of variable names when the VALUES= or DOLIST= argument is specified. If the VARS= argument is not specified along with the VALUES= argument, then the PREFIX= argument is used.

Help Option

You can specify either of the following to display the option names and simple examples of the macro syntax:

```
%mktlab(help)
%mktlab(?)
```

%MktLab Macro Notes

This macro specifies **options nonotes** throughout most of its execution. If you want to see all the notes, submit the following statement before running the macro:

```
%let mktopts = notes;
```

To see the macro version, submit the following statement before running the macro:

```
%let mktopts = version;
```

Example 1: Adding an Intercept and Relabeling the Levels to a Design

Suppose you used the %MktEx macro to create a design that has eleven 2-level factors (with default levels of 1 and 2). The following steps create and display the design:

```
%mktex(n=12, OPTIONS=nosort)

proc print noobs;
run;
```

Figure 2 shows the design.

Figure 2 Experimental Design with Eleven 2-Level Factors

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11
2	2	2	2	2	2	2	2	2	2	2
1	2	1	2	2	2	1	1	1	2	1
1	1	2	1	2	2	2	1	1	1	2
2	1	1	2	1	2	2	2	1	1	1
1	2	1	1	2	1	2	2	2	1	1
1	1	2	1	1	2	1	2	2	2	1
1	1	1	2	1	1	2	1	2	2	2
2	1	1	1	2	1	1	2	1	2	2
2	2	1	1	1	2	1	1	2	1	2
2	2	2	1	1	1	2	1	1	2	1
1	2	2	2	1	1	1	2	1	1	2
2	1	2	2	2	1	1	1	2	1	1

You can use either the %MktEx macro or the %MktLab macro to assign levels of –1 and 1 and add an intercept. You do it directly with the %MktEx macro by specifying LEVELS=I INT. The value of **I** specifies centered integer levels, and **INT** adds the intercept. The following statements illustrate this method:

```
%mktex(n=12, OPTIONS=nosort, levels=i int)
```

However, if you want to change the factor names, or if you require a more complicated relabeling of the levels, you can use the %MktLab macro. The following statements demonstrate this method:

```
%mktex(n=12, OPTIONS=nosort)

%mktlab(data=design, VALUES=1 -1, int=Had0, prefix=Had)

proc print noobs;
run;
```

The %MktLab macro assigns levels of –1 and 1, adds an intercept named Had0, and changes the variable name prefixes from X to Had. This creates the Hadamard matrix shown in Figure 3.

Figure 3 A Hadamard Matrix

Had0	Had1	Had2	Had3	Had4	Had5	Had6	Had7	Had8	Had9	Had10	Had11
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	1	-1	-1	-1	1	1	1	-1	1
1	1	1	-1	1	-1	-1	-1	1	1	1	-1
1	-1	1	1	-1	1	-1	-1	-1	1	1	1
1	1	-1	1	1	-1	1	-1	-1	-1	1	1
1	1	1	-1	1	1	-1	1	-1	-1	-1	1
1	1	1	1	-1	1	1	-1	1	-1	-1	-1
1	-1	1	1	1	-1	1	1	-1	1	-1	-1
1	-1	-1	1	1	1	-1	1	1	-1	1	-1
1	-1	-1	-1	1	1	1	-1	1	1	-1	1
1	1	-1	-1	-1	1	1	1	-1	1	1	-1
1	-1	1	-1	-1	-1	1	1	1	-1	1	1

Alternatively, you can use the following KEY= data set to do the same thing:

```
data key;
  array Had[11];
  input Had1 @@;
  do i = 2 to 11; Had[i] = Had1; end;
  drop i;
  datalines;
1 -1
;

proc print data=key;
run;
```

Figure 4 shows the Key data set.

Figure 4 Key Data Set

Obs	Had1	Had2	Had3	Had4	Had5	Had6	Had7	Had8	Had9	Had10	Had11
1	1	1	1	1	1	1	1	1	1	1	1
2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

The following statements use this data set to make the final design:

```
%mktlab(data=design, KEY=key, int=Had0)

proc print data=Final;
run;
```

The KEY= argument specifies the data set that contains all the variables that you want in the design and all their levels. This information is applied to the design. The DATA= argument specifies the data set that contains the design. If you do not specify the DATA= argument, the %MktLab macro uses the OUTR= data set name from the %MktEx macro, which is named Randomized by default. The %MktLab macro saves the recoded design in an output data set that is named Final by default.

Figure 5 shows that the Hadamard matrix from this operation is exactly the same as the previous one.

Figure 5 Key Data Set

Obs	Had0	Had1	Had2	Had3	Had4	Had5	Had6	Had7	Had8	Had9	Had10	Had11
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	1	1	-1	1	-1	-1	-1	1	1	1	-1	1
3	1	1	1	-1	1	-1	-1	-1	1	1	1	-1
4	1	-1	1	1	-1	1	-1	-1	-1	1	1	1
5	1	1	-1	1	1	-1	1	-1	-1	-1	1	1
6	1	1	1	-1	1	1	-1	1	-1	-1	-1	1
7	1	1	1	1	-1	1	1	-1	1	-1	-1	-1
8	1	-1	1	1	1	-1	1	1	-1	1	-1	-1
9	1	-1	-1	1	1	1	-1	1	1	-1	1	-1
10	1	-1	-1	-1	1	1	1	-1	1	1	-1	1
11	1	1	-1	-1	-1	1	1	1	-1	1	1	-1
12	1	-1	1	-1	-1	-1	1	1	1	-1	1	1

Example 2: Renaming Variables and Labeling Factor Levels

This example considers consumer food products. The following statements read a possible design:

```
data randomized;
  input x1-x8 @@;
  datalines;
4 2 1 1 1 2 2 2 2 1 1 2 1 3 1 3 3 4 2 2 1 3 2 3 4 3 2 1 3 2 2 3 4 1 2 1
1 1 1 1 2 4 1 2 1 2 1 1 1 2 1 2 3 3 2 1 2 2 2 2 2 2 2 3 1 4 2 1 1 2 2 2
3 2 2 1 3 1 2 1 1 4 1 2 2 3 1 2 1 3 2 2 1 3 1 1 3 2 1 2 2 1 2 3 3 4 1 1
3 1 1 3 4 1 2 2 2 1 2 1 2 3 2 1 2 3 2 2 2 1 2 1 3 3 1 3 4 2 2 2 1 3 1 2
2 4 2 2 3 1 1 2 3 1 2 2 3 2 1 2 3 3 1 1 2 3 1 1 4 4 2 1 2 2 1 3 1 1 1 1
```

```
3 2 1 2 4 3 1 2 3 3 2 2 1 2 2 1 2 1 1 3 1 3 1 1 1 1 2 3
;
```

Designs that you create by using the %MktEx macro always have factor names X1, X2, and so on, and the levels are consecutive integers that begin with 1 (1, 2 for 2-level factors; 1, 2, 3 for 3-level factors; and so on). The %MktLab macro provides a convenient way to change the names and levels to more meaningful values. The data set Key contains the variable names and levels that you ultimately want. The following statements create the Key data set:

```
data key;
  missing N;
  input Client ClientLineExtension ClientMicro $ ShelfTalker $
        Regional Private PrivateMicro $ NationalLabel;
  format _numeric_ dollar5.2;
  datalines;
1.29 1.39 micro Yes 1.99 1.49 micro 1.99
1.69 1.89 stove No  2.49 2.29 stove 2.39
2.09 2.39 .     .   N     N   .     N
N    N    .     .   .     .   .     .
;
```

The variable Client with 4 levels is made from X1, the variable ClientLineExtension with 4 levels is made from X2, and the variable ClientMicro with 2 levels is made from X3. The **N** (for "not available") is treated as a special missing value. The Key data set has four rows because the maximum number of levels is four. Factors with fewer than 4 levels are filled in with ordinary missing values.

Now you can use the %MktLab macro to recode the design. This macro takes the default DATA=RANDOMIZED data set from the %MktEx macro and uses the rules in the KEY=Key data set to create the information in the OUT=Final data set, as follows:

```
%mktlab(data=randomized, KEY=key)

proc sort data=final out=final;
   by shelftalker;
run;

proc print data=final;
   by shelftalker;
run;
```

Figure 6 shows the recoded design.

Figure 6 Final Data Set

ShelfTalker=No

Obs	Client	ClientLineExtension	ClientMicro	Regional	Private	PrivateMicro	NationalLabel
1	$1.69	$1.39	micro	$1.99	N	micro	N
2	$2.09	N	stove	$1.99	N	stove	N
3	$1.69	N	micro	$1.99	$2.29	micro	$1.99
4	$1.29	$1.89	micro	N	N	stove	$1.99
5	$1.69	$1.89	stove	$2.49	$2.29	stove	N
6	$1.29	N	micro	$2.49	N	micro	$2.39
7	$1.29	$2.39	stove	$1.99	N	micro	$1.99
8	$2.09	$1.89	micro	$2.49	$1.49	stove	N
9	N	$1.39	stove	$2.49	$1.49	stove	$1.99
10	N	$1.89	stove	$1.99	N	micro	$2.39
11	$1.69	N	stove	N	$1.49	micro	$2.39
12	$2.09	$1.39	stove	N	$2.29	micro	$2.39
13	N	$2.39	micro	N	N	stove	$2.39

ShelfTalker=Yes

Obs	Client	ClientLineExtension	ClientMicro	Regional	Private	PrivateMicro	NationalLabel
14	N	$1.89	micro	$1.99	$2.29	stove	$2.39
15	N	$2.39	stove	N	$2.29	stove	N
16	N	$1.39	stove	$1.99	$1.49	micro	$1.99
17	$1.29	N	stove	$1.99	$2.29	stove	$2.39
18	$2.09	$1.89	stove	N	$1.49	stove	$1.99
19	$2.09	N	micro	N	$1.49	micro	N
20	$1.69	$2.39	stove	$2.49	N	stove	$2.39
21	$1.69	$1.39	stove	N	N	micro	N
22	$2.09	$2.39	micro	$2.49	N	micro	$1.99
23	N	N	stove	$2.49	$2.29	micro	N
24	$1.29	$1.39	micro	N	$2.29	micro	$2.39
25	$1.29	$1.89	stove	$2.49	$1.49	micro	N
26	$1.29	$2.39	micro	$1.99	$1.49	stove	N

The %MktLab macro creates the OUT= data set by repeatedly reading the KEY= data set, one data item at a time, using the information in the DATA= data set to determine which levels to read from the KEY= data set. In this example, for the first observation, X1 = 4, so the fourth value of the first KEY= variable is read; then X2 = 2, so the second value of the second KEY= variable is read; then X3 = 1, so the first value of the third KEY= variable is read; and so on; then X8 = 2, so the second value of the eighth KEY= variable is read; then the first observation is output. This continues for all observations.

Example 3: Relabeling an Orthogonal Array

This example shows how to use the %MktLab macro to relabel an orthogonal array. The following statements create the L_{36} orthogonal array ($3^{12}2^{11}$ in 36 rows); change the names of the 2-level factors to Two1–Two11 and assign them values the –1, 1; and change the names of the 3-level factors to Thr1–Thr12 and assign them the values –1, 0, 1:

```
%mktex(n=36, seed=420)

data key;
  array x[23] two1-two11 thr1-thr12;
  input two1 thr1;
  do i =  2 to 11; x[i] = two1; end;
  do i = 13 to 23; x[i] = thr1; end;
  drop i;
  datalines;
-1 -1
 1  0
 .  1
;

%mktlab(data=randomized, KEY=key)

proc print data=key noobs;
   var two:;
run;

proc print data=key noobs;
   var thr:;
run;

proc print data=final(obs=5) noobs;
   var two:;
run;

proc print data=final(obs=5) noobs;
   var thr:;
run;
```

Figure 7 and Figure 8 show the Key data set.

Figure 7 Key Data Set: Two Variables

two1	two2	two3	two4	two5	two6	two7	two8	two9	two10	two11
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	1	1	1	1	1
.	.	.	.	.	.	.	.	.	.	.

Figure 8 Key Data Set: Thr Variables

thr1	thr2	thr3	thr4	thr5	thr6	thr7	thr8	thr9	thr10	thr11	thr12
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1

Figure 9 and Figure 10 show the Final data set.

Figure 9 Final Data Set: Two Variables

two1	two2	two3	two4	two5	two6	two7	two8	two9	two10	two11
-1	-1	1	1	1	1	1	1	1	1	-1
1	1	1	1	-1	1	1	-1	-1	1	1
-1	-1	-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	1	1	1	-1	-1	-1	1
-1	1	-1	-1	-1	-1	1	1	1	1	1

Figure 10 Final Data Set: Thr Variables

thr1	thr2	thr3	thr4	thr5	thr6	thr7	thr8	thr9	thr10	thr11	thr12
0	0	0	0	-1	-1	0	0	-1	0	0	-1
1	-1	-1	1	-1	0	0	1	1	0	-1	-1
0	-1	1	0	0	0	1	0	1	1	1	-1
0	1	0	-1	0	1	-1	-1	1	0	-1	-1
1	-1	0	-1	0	-1	1	1	0	-1	0	-1

Example 4: Relabeling a Blocked Design

This example creates a design and blocks it. It shows that it is OK if not all the variables in the input design are used. The variables Block, Run, and X4 are just copied from the input to the output. The following statements create the design and block it:

```
%mktex(n=18, seed=396)

%mktblock(data=design, nblocks=2, factors=x1-x4, seed=292)
```

The %MktEx macro saves the design in the output data set Design, and the %MktBlock macro saves the blocked design in the output data set Blocked.

The following DATA step creates the Key data set:

```
data key;
   input Brand $ Price Size;
   format price dollar5.2;
   datalines;
Acme 1.49   6
Apex 1.79   8
.    1.99  12
;
```

Now you use the %MktLab macro and the Blocked and Key data sets to recode the design:

```
%mktlab(data=blocked, KEY=key)

proc print;
run;
```

Figure 10 shows the results.

Figure 11 Recoded Blocked Design

Obs	Brand	Price	Size	Block	Run	x4
1	Acme	$1.49	12	1	1	2
2	Acme	$1.79	6	1	2	3
3	Acme	$1.79	8	1	3	2
4	Acme	$1.99	8	1	4	1
5	Apex	$1.49	8	1	5	3
6	Apex	$1.49	12	1	6	3
7	Apex	$1.79	6	1	7	1
8	Apex	$1.99	6	1	8	2
9	Apex	$1.99	12	1	9	1
10	Acme	$1.49	6	2	1	1
11	Acme	$1.49	8	2	2	1
12	Acme	$1.79	12	2	3	2
13	Acme	$1.99	6	2	4	3
14	Acme	$1.99	12	2	5	3
15	Apex	$1.49	6	2	6	2
16	Apex	$1.79	8	2	7	3
17	Apex	$1.79	12	2	8	1
18	Apex	$1.99	8	2	9	2

Example 5: Using the LABELS= Argument

This example demonstrates how to use the %MktLab macro and the LABELS= argument for a vacation choice design. You usually use the LABELS= argument with the VALUES= argument rather than constructing the KEY= data set yourself, but you can use it either way. The following statements create the design:

```
%mktex(3 ** 15, n=36, seed=17, maxtime=0)

%mktblock(data=randomized, nblocks=2, factors=x1-x15, seed=448)
```

Next you write a macro that creates the labels:

```
%macro lab;
   label X1  = 'Hawaii, Accommodations'
         X2  = 'Alaska, Accommodations'
         X3  = 'Mexico, Accommodations'
         X4  = 'California, Accommodations'
         X5  = 'Maine, Accommodations'
         X6  = 'Hawaii, Scenery'
         X7  = 'Alaska, Scenery'
         X8  = 'Mexico, Scenery'
         X9  = 'California, Scenery'
         X10 = 'Maine, Scenery'
         X11 = 'Hawaii, Price'
         X12 = 'Alaska, Price'
         X13 = 'Mexico, Price'
         X14 = 'California, Price'
```

```
         X15 = 'Maine, Price';

   format x11-x15 dollar5.;
%mend;
```

Then you create the Key data set:

```
data key;
   length x1-x5 $ 16 x6-x10 $ 8 x11-x15 8;
   input x1 & $ x6 $ x11;
   x2  = x1;    x3 = x1;    x4 = x1;    x5 = x1;
   x7  = x6;    x8 = x6;    x9 = x6;   x10 = x6;
   x12 = x11;  x13 = x11;  x14 = x11;  x15 = x11;
   datalines;
Cabin            Mountains   999
Bed & Breakfast  Lake       1249
Hotel            Beach      1499
;
```

Finally, you invoke the %MktLab macro, and you specify the LABELS= argument and supply the name of the macro (without the % sign) that defines the labels.

```
%mktlab(data=blocked, KEY=key, LABELS=lab)
```

You can use PROC CONTENTS as follows to see how the labels have been applied to the design:

```
proc contents p;
ods select position;
run;
```

Figure 12 shows the variable name, label, and format information.

Figure 12 Variables in Creation Order

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	x1	Char	16		Hawaii, Accommodations
2	x2	Char	16		Alaska, Accommodations
3	x3	Char	16		Mexico, Accommodations
4	x4	Char	16		California, Accommodations
5	x5	Char	16		Maine, Accommodations
6	x6	Char	8		Hawaii, Scenery
7	x7	Char	8		Alaska, Scenery
8	x8	Char	8		Mexico, Scenery
9	x9	Char	8		California, Scenery
10	x10	Char	8		Maine, Scenery
11	x11	Num	8	DOLLAR5.	Hawaii, Price
12	x12	Num	8	DOLLAR5.	Alaska, Price
13	x13	Num	8	DOLLAR5.	Mexico, Price
14	x14	Num	8	DOLLAR5.	California, Price
15	x15	Num	8	DOLLAR5.	Maine, Price
16	Block	Num	8		
17	Run	Num	8		