%MktLab Macro

The `%MktLab` autocall macro processes an experimental design, usually created by the `%MktEx` macro, and assigns the final variable names and levels. See the following pages for examples of using this macro in the design chapter: 81, 85, 109, 112, 166 and 201. Also see the following pages for examples of using this macro in the discrete choice chapter: 333, 353, 501, 538, 564, 567, 596 and 602. Additional examples appear throughout this chapter. For example, say you used the `%MktEx` macro to create a design with 11 two-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;
```

The design is as follows:

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

Either the `%MktEx` macro or the `%MktLab` macro can be used to assign levels of –1 and 1 and add an intercept. You can do it directly with the `%MktEx` macro, using `levels=i int`. The value of `i` specifies centered integer levels, and `int` adds the intercept. The following step illustrates this:

```
%mktex(n=12,                    /* 12 runs                              */
       options=nosort,          /* do not sort design                   */
       levels=i                 /* -1 and 1 instead of 1 and 2          */
              int)              /* add an intercept to the design       */
```

However, if you want to change the factor names, and for more complicated relabeling of the levels, you need to use the `%MktLab` macro. This is illustrated in the following step:

```
%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 a Hadamard matrix (although, of course, the Hadamard matrix can have any set of variable names). The resulting Hadamard matrix is as follows:

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 `key=` data set that follows 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;
```

The `key=` data set is as follows:

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 step uses this data set to make the final design:

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

The Hadamard matrix from this step (not shown) is exactly the same as the one just shown previously.

The `key=` data set contains all of the variables that you want in the design and all of their levels. This information is applied to the design, by default the one stored in a data set called RANDOMIZED, which is the default `outr=` data set name from the `%MktEx` macro. The results are stored in a new data set, FINAL, with the desired factor names and levels.

Consider the consumer food product example from page 255. The following step reads 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 created by the `%MktEx` macro always have factor names `x1`, `x2`, ..., and so on, and the levels are consecutive integers beginning with 1 (1, 2 for two-level factors; 1, 2, 3 for three-level factors; and so on). The `%MktLab` macro provides you with 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 step creates a `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    .     .   .    .    .     .
;

%mktlab(data=randomized, key=key)

proc sort; by shelftalker; run;

proc print; by shelftalker; run;
```

The variable `Client` with 4 levels is made from `x1`, `ClientLineExtension` with 4 levels is made from `x2`, `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 four levels are filled in with ordinary missing values. The `%MktLab` macro takes the default `data=randomized` data set from `%MktEx` and uses the rules in the `key=key` data set, to create the information in the `out=final` data set, which is shown next, sorted by the shelf talker variable.

Some of the design is as follows:

```
----------------------------- ShelfTalker=No -----------------------------

                Client
                 Line      Client                          Private   National
Obs   Client   Extension   Micro    Regional   Private     Micro     Label

  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
  .
  .
  .

----------------------------- ShelfTalker=Yes ----------------------------

                Client
                 Line      Client                          Private   National
Obs   Client   Extension   Micro    Regional   Private     Micro     Label

 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
  .
  .
  .
```

This macro creates the `out=` data set by repeatedly reading and rereading the `key=` data set, one datum 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, ..., 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. Unlike previous releases, the `data=` data is not required to have the default `levels=` specification (integer values beginning with 1). Other numeric values are fine and are converted to consecutive positive integers before performing the final mapping.

The following steps create the L_{36}, changes the names of the two-level factors to `two1-two11`, assigns them values the –1, 1, changes the names of the three-level factors to `thr1-thr12`, and assigns 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;
```

The Key data set is as follows:

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
.	.	.	.	.	.	.	.	.	.	.

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

The first five rows of the design are as follows:

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

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

This following steps create a design and block it. This example shows that it is okay if not all of 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 steps create the design:

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

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

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

%mktlab(data=blocked, key=key)

proc print; run;
```

The results are as follows:

Block	Run	Brand	Price	Size	x4
1	1	Acme	$1.49	12	2
	2	Acme	$1.79	6	3
	3	Acme	$1.79	8	2
	4	Acme	$1.99	8	1
	5	Acme	$1.99	12	1
	6	Apex	$1.49	8	3
	7	Apex	$1.49	12	3
	8	Apex	$1.79	6	1
	9	Apex	$1.99	6	2
2	1	Acme	$1.49	6	1
	2	Acme	$1.49	6	2
	3	Acme	$1.79	8	3
	4	Acme	$1.99	12	3

```
5     Apex     $1.49      8      1
6     Apex     $1.79     12      1
7     Apex     $1.79     12      2
8     Apex     $1.99      6      3
9     Apex     $1.99      8      2
```

This next example illustrates using the `labels=` option. This option is more typically used with `values=` input, rather than when you construct the `key=` data set yourself, but it can be used either way. This example is from a vacation choice example. The following steps create the design:

```
%mktex(3 ** 15,                    /* 15 three-level factors              */
       n=36,                       /* 36 runs                             */
       seed=17,                    /* random number seed                  */
       maxtime=0)                  /* no more than 1 iter in each phase   */

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

%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;
```

```
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
;

%mktlab(data=blocked, key=key, labels=lab)

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

The variable name, label, and format information are as follows:

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		

%MktLab Macro Options

The following options can be used with the `%MktLab` macro:

Option	Description
`help`	(positional) "help" or "?" displays syntax summary
`cfill=`*character-string*	character fill value
`data=`*SAS-data-set*	input design data set
`dolist=`*do-list*	new values using a do-list syntax
`int=`*variable-list*	name of an intercept variable
`key=`*SAS-data-set*	`Key` data set
`labels=`*macro-name*	macro that provides labels and formats
`nfill=`*number*	numeric fill value
`options=noprint`	suppress the display of the variable mappings
`out=`*SAS-data-set*	output data set with recoded design
`prefix=`*variable-prefix*	prefix for naming variables
`statements=`*SAS-code*	add extra statements
`values=`*value-list*	the new values for all of the variables
`vars=`*variable-list*	list of variable names

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

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

cfill= *character-string*

specifies the fill value in the `key=` data set for character variables. See the `nfill=` option for more information about fill values. The default is `cfill=' '`.

data= *SAS-data-set*

specifies the input data set with the experimental design, usually created by the `%MktEx` macro. The default is `data=Randomized`. The factor levels in the `data=` data set must be consecutive integers beginning with 1.

dolist= *do-list*

specifies the new values, using a do-list syntax (`n TO m <BY p>`), for example: `dolist=1 to 10` or `dolist=0 to 9`. With asymmetric designs (not all factors have the same levels), specify the levels for the largest number of levels. For example, with two-level and three-level factors and `dolist= 0 to 2`, the two-level factors is assigned levels 0 and 1, and the three-level factors is assigned levels 0, 1, and 2. Do not specify both `values=` and `dolist=`. By default, when `key=`, `values=`, and `dolist=` are all not specified, the default value list comes from `dolist=1 to 100`.

int= *variable-list*

specifies the name of an intercept variable (column of ones), if you want an intercept added to the `out=`

data set. You can also specify a variable list instead of a variable name if you would like to make a list of variables with values all one. This can be useful, for example, for creating flag variables for generic choice models when the design is going to be used as a candidate set for the `%ChoicEff` macro.

key= *SAS-data-set*

specifies the input data set with the key to recoding the design. When `values=` or `dolist=` is specified, this data set is made for you. By default, when `key=`, `values=`, and `dolist=` are all not 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 additional information to the `key=` data set. For a simple format specification, it is easier to use `statements=`. For more involved specifications, use `labels=`. Note that you specify just the macro name, no percents on the `labels=` option. This option is illustrated in the following step:

```
%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 two-level factors should have a value of `nfill=`, which by default is ordinary missing. If the macro tries to access one of these values, it displays a warning. If you would like ordinary missing (.) to be a legitimate level, specify a different `nfill=` 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. Specify values after `options=`.

`noprint`
suppresses the display of the variable mappings.

out= *SAS-data-set*

specifies the output data set with the final, recoded design. The default is `out=final`. Often, you will want to specify a two-level name to create a permanent SAS data set so the design is available later for analysis.

prefix= *variable-prefix*

specifies a prefix for naming variables when `values=` is specified. For example, `prefix=Var` creates variables `Var1`, `Var2`, and so on. By default, the variables are `x1`, `x2`, ... This option is ignored when `vars=` is specified.

statements= *SAS-code*

is an alternative to `labels=` that you can use to add extra statements to the `key=` data set. For a simple format specification, it is easier to use `statements=`. For more involved specifications, use `labels=`. This option is illustrated in the following step:

```
%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; 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
.				
.				
.				

values= *value-list*

specifies the new values for all of the variables. If all variables will have the same value, it is easier to specify `values=` or `dolist=` than `key=`. When you specify `values=`, 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. With asymmetric designs (not all factors have the same levels) specify the levels for the largest number of levels. For example, with two-level and three-level factors and `values=a b c`, the two-level factors are assigned levels `'a'` and `'b'`, and the three-level factors are assigned levels `'a'`, `'b'`, and `'c'`. Do not specify both `values=` and `dolist=`. By default, when `key=`, `values=`, and `dolist=` are all not specified, the default value list comes from `dolist=1 to 100`.

vars= *variable-list*
specifies a list of variable names when `values=` or `dolist=` is specified. If `vars=` is not specified with `values=`, then `prefix=` is used.

%MktLab Macro Notes

This macro specifies `options nonotes` throughout most of its execution. If you want to see all of the notes, submit the statement `%let mktopts = notes;` before running the macro. To see the macro version, submit the statement `%let mktopts = version;` before running the macro.