%MktBal Macro

The %MktBal autocall macro creates factorial designs using an algorithm that ensures that the design is perfectly balanced, or when the number of levels of a factor does not divide the number of runs, as close to perfectly balanced as possible. Before using the %MktBal macro, you should try the %MktEx macro to see if it makes a design that is balanced enough for your needs. The %MktEx macro can directly create thousands of orthogonal and balanced designs that the %MktBal algorithm will never find. Even when the %MktEx macro cannot create an orthogonal and balanced design, it will usually find a nearly balanced design. Designs created with the %MktBal macro, while perfectly balanced, might be less efficient than designs found with the %MktEx macro, and for large problems, the %MktBal macro can be slow.

The %MktBal macro is *not* a full-featured experimental design generator. For example, you cannot specify interactions that you want to estimate or specify restrictions such as which levels may or may not appear together. You must use the %MktEx macro for that. The %MktBal macro builds a design by creating a balanced first factor, optimally (or nearly optimally) blocking it to create the second factor, then blocking the first two factors to create the third, and so on. Once it creates all factors, it refines each factor. Each factor is in turn removed from the design, and the rest of the design is reblocked, replacing the initial factor if the new design is more *D*-efficient.

The following steps provide a simple example of creating and evaluating a design with 2 two-level factors and 3 three-level factors in 18 runs:

```
%mktbal(2 2 3 3 3, n=18, seed=151)

%mkteval(data=design)
```

The %MktEval macro evaluates the results. This design is in fact optimal.

In all cases, the factors are named x1, x2, x3, and so on. You can use the %MktLab macro to conveniently change them.

This next example, at 120 runs and with factor levels greater than 5, is starting to get big, and by default, it will run slowly. You can use the maxstarts=, maxtries=, and maxiter= options to make the macro run more quickly. The following steps create the design, with and without these options:

```
%mktbal(2 3 4 5 6 7 8 9 10, n=120, options=progress, seed=17)

%mktbal(2 3 4 5 6 7 8 9 10, n=120, options=progress, seed=17,
        maxstarts=1, maxiter=1, maxtries=1)
```

The second example, with the options, runs much faster than the first.

%MktBal Macro Options

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

Option	Description
`list`	(positional) list of the numbers of levels
	(positional) "help" or "?" displays syntax summary
`init=`*SAS-data-set*	initial input experimental design
`iter=`*n*	maximum iterations (designs to create)
`maxinititer=`*n*	maximum initialization iterations
`maxiter=`*n*	maximum iterations (designs to create)
`maxstarts=`*n*	maximum number of random starts
`maxtries=`*n*	times to try refining each factor
`n=`*n*	number of runs in the design
`options=noprint`	suppress the final *D*-efficiency
`options=nosort`	do not sort the final design
`options=oa`	an orthogonal array is sought
`options=progress`	reports on macro progress
`options=sequential`	factors are added but not refined
`out=`*SAS-data set*	output experimental design
`seed=`*n*	random number seed

Help Option

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

```
%mktbal(help)
%mktbal(?)
```

list

specifies a list of the numbers of levels of all the factors. For example, for 3 two-level factors specify either `2 2 2` or `2 ** 3`. Lists of numbers, like `2 2 3 3 4 4` or a *levels**number of factors* syntax like: `2**2 3**2 4**2` can be used, or both can be combined: `2 2 3**4 5 6`. The specification `3**4` means 4 three-level factors. You must specify a list. Note that the factor list is a positional parameter. This means it must come first, and unlike all other parameters, it is not specified after a name and an equal sign.

n= *n*

specifies the number of runs in the design. You must specify `n=`. You can use the `%MktRuns` macro to get suggestions for values of `n=`.

out= *sas-data set*
specifies the output experimental design. The default is `out=design`.

These next options control some of the details of the `%MktBal` macro.

init= *sas-data set*
specifies the initial design. You can specify just the first few columns of the design, in this data set with column names `x1`, `x2`, ..., and these columns will never change. This is different from the `init=` data set in the `%MktEx` macro. It is often the case that the first few columns can be constructed combinatorially, and it is known that the optimal design will just add new columns to the initial few orthogonal columns. This option makes that process more efficient. By default, when `init=` is not specified, all columns are iteratively found and improved.

maxinititer= *n*
specifies the maximum number of random starts for each factor during the initialization stage. This is the number of iterations that PROC OPTEX performs for each factor during the initialization. With larger values, the macro tends to find slightly better designs at a cost of slower run times. The default is the value of `maxstarts=`.

maxiter= *n*
iter= *n*
specifies the maximum iterations (designs to create). By default, `maxiter=5`. This is the outer most set of iterations.

maxstarts= *n*
specifies the maximum number of random starts for each factor. This is the number of iterations that PROC OPTEX performs for each factor. With larger values, the macro tends to find slightly better designs at a cost of slower run times. The default is `maxstarts=10`.

maxtries= *n*
specifies the maximum number of times to try refining each factor after the initialization stage. The default is `maxtries=10`. Increasing the value of this option usually has little effect since the macro stops refining each design when the efficiency stabilizes.

options= *options-list*
specifies binary options. By default, none of these options are specified. Specify one or more of the following values after `options=`.

`noprint`
specifies that the final *D*-efficiency should not be displayed.

`nosort`
do not sort the final design.

`oa`
specifies that an orthogonal array is sought. Factors are added sequentially and they are acceptable only when D-efficiency reaches 100. With `options=oa`, you can use the `%MktBal` macro to search for small orthogonal arrays or in some cases augment existing orthogonal arrays.

`progress`
reports on the macro's progress. For large numbers of factors, a large number or runs, or when the number of levels is large, this macro is slow. The `options=progress` specification gives you information about which step is being executed.

`sequential`
sequentially adds factors to the design and does not go back and refine them.

seed= *n*
specifies the random number seed. By default, `seed=0`, and clock time is used to make the random number seed. By specifying a random number seed, results should be reproducible within a SAS release for a particular operating system and for a particular version of the macro. However, due to machine and macro differences, some results might not be exactly reproducible everywhere, although you would expect the efficiency differences to be slight.

%MktBal 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.