%MktMDiff Macro

The `%MktMDiff` autocall macro analyzes MaxDiff (maximum difference or best-worst) data (Louviere 1991, Finn and Louviere 1992). The result of the analysis is a scaling of the attributes on a preference or importance scale. In a MaxDiff study, subjects are shown sets of messages or product attributes and are asked to choose the best (or most important) from each set as well as the worst (or least important). The design consists of:

- t attributes
- b sets (or blocks) of attributes with
- k attributes in each set

These are the parameters of a balanced incomplete block design or BIBD, which can be constructed from the `%MktBIBD` macro. Note, however, that you can use designs that are produced by the `%MktBIBD` macro but do not meet the strict requirements for a BIBD. See page 1111 for more information about this. To illustrate, a researcher is interested in preference for cell phones based on the attributes of the phones. The attributes are as follows:

Camera
Flip
Hands Free
Games
Internet
Free Replacement
Battery Life
Large Letters
Applications

Subjects are shown subsets of these 9 attributes and asked to pick which is the most important when they make a cell phone choice and which is the least important. We can use the `%MktBSize` macro as follows to get ideas about how many blocks to use and how many to show at one time:

```
%mktbsize(nattrs=9, setsize=2 to 9, nsets=1 to 20)
```

The results of this step are as follows:

t Number of Attributes	k Set Size	b Number of Sets	r Attribute Frequency	Lambda Pairwise Frequencies	n Total Sample Size
9	3	12	4	1	36
9	4	18	8	3	72
9	5	18	10	5	90
9	6	12	8	5	72
9	8	9	8	7	72

With 9 attributes, there are five sizes that meet the necessary but not sufficient conditions for the existence of a BIBD. Of the candidates (3, 4, 5, 6, and 8), 4 or 5 seem like good choices. (Three seems

a bit small and more than 5 seems a bit big given that we only have 9 attributes.) The following step creates a BIBD with $t = 9$ attributes, shown in $b = 18$ sets of size $k = 5$:

```
%mktbibd(nattrs=9, setsize=5, nsets=18, seed=377, out=sasuser.bibd)
```

The following table, produced by the macro, shows that the design is in fact a BIBD:

Attribute by Attribute Frequencies

	1	2	3	4	5	6	7	8	9
1	10	5	5	5	5	5	5	5	5
2		10	5	5	5	5	5	5	5
3			10	5	5	5	5	5	5
4				10	5	5	5	5	5
5					10	5	5	5	5
6						10	5	5	5
7							10	5	5
8								10	5
9									10

This is the attribute by attribute frequency matrix. The diagonal elements of the matrix show how often each attribute occurs. Each of the $t = 9$ attributes occurs the same number of times (10 times). Furthermore, each of the $t = 9$ attributes occurs with each of the remaining 8 attributes exactly 5 times. These two constant values, one one the diagonal and one off, show that the design is a BIBD. The design is as follows:

Balanced Incomplete Block Design

x1	x2	x3	x4	x5
1	2	4	7	9
6	2	9	8	4
5	8	2	1	7
7	9	1	6	3
9	7	6	5	2
3	1	5	6	8
4	6	3	8	2
6	7	4	1	8
7	5	2	3	4
1	3	8	2	7
3	4	1	5	9
2	9	6	3	1
8	3	9	2	5
9	8	7	4	3
5	4	8	9	1

```
2     1     5     4     6
8     5     7     9     6
4     6     3     7     5
```

The first set or consists of attributes 1, 2, 4, 7, and 9, which are as follows:

Camera
Flip
Games
Battery Life
Applications

Subjects will choose the best and worst from this and every other set.

The `%MktMDiff` macro reads the experimental design and the data, combines them, arranges them in the right form for analysis, and performs the analysis using a multinomial logit model. The data are arrayed so that each original MaxDiff set forms two choice sets in the analysis: one positively weighted set for the best choice and one negatively weighted set for the worst choice. The input data can come in one of eight forms:

`bw`
best then worst (e.g. `b1-b18 w1-w18`), and the data are attribute numbers (range from 1 to `nattrs=`t). If the `groups=` option is specified, for example with `groups=3`, the variables are `b1-b6 w1-w6` and three observations provide the information about all of the rows in the block design.

`wb`
worst then best (e.g. `w1-w18 b1-b18`), and the data are attribute numbers (range from 1 to `nattrs=`t). If the `groups=` option is specified, for example with `groups=3`, the variables are `w1-w6 b1-b6` and three observations provide the information about all of the rows in the block design.

`bwalt`
best then worst and alternating (e.g. `b1 w1 b2 w2 ... b18 w18`), and the data are attribute numbers (range from 1 to `nattrs=`t). If the `groups=` option is specified, for example with `groups=3`, the variables are `b1 w1 b2 w2 ... b6 w6` and three observations provide the information about all of the rows in the block design.

`wbalt`
worst then best and alternating (e.g. `w1 b1 w2 b2 ... w18 b18`), and the data are attribute numbers (range from 1 to `nattrs=`t). If the `groups=` option is specified, for example with `groups=3`, the variables are `w1 b1 w2 b2 ... w6 b6` and three observations provide the information about all of the rows in the block design.

`bwpos`
best then worst (e.g. `b1-b18 w1-w18`), and the data are positions (range from 1 to `setsize=`k). If the `groups=` option is specified, for example with `groups=3`, the variables are `b1-b6 w1-w6` and three observations provide the information about all of the rows in the block design.

`wbpos`
worst then best (e.g. `w1-w18 b1-b18`), and the data are positions (range from 1 to `setsize=`k). If the `groups=` option is specified, for example with `groups=3`, the variables are `w1-w6 b1-b6` and three observations provide the information about all of the rows in the block design.

`bwaltpos`
best then worst and alternating (e.g. `b1 w1 b2 w2 ... b18 w18`), and the data are positions (range from 1 to `setsize=`k). If the `groups=` option is specified, for example with `groups=3`, the variables are `b1 w1 b2 w2 ... b6 w6` and three observations provide the information about all of the rows in the block design.

`wbaltpos`
worst then best and alternating (e.g. `w1 b1 w2 b2 ... w18 b18`), and the data are positions (range from 1 to `setsize=`k). If the `groups=` option is specified, for example with `groups=3`, the variables are `w1 b1 w2 b2 ... w6 b6` and three observations provide the information about all of the rows in the block design.

Note that in all cases, any variable names can be used. For example, the variables could be `x1-x36`. In the case of `bwalt`, the odd numbered variables will correspond to best picks and the even numbered variables will correspond to worst picks.

The following example uses the design created previously:

```
title 'Best Worst Example with Cell Phone Attributes';

data bestworst;
   input Sub $ @4 (b1-b18 w1-w18) (1.);
   datalines;
 1 18866188439934965394195534221293549 4
 2 76535887389138849392267364433659555 4
 3 78212628289284856499599344721393565 5
 4 48136326424639918716212541535128145 3
 5 78716886381187817522599538235229329 3
 6 78765886789187866749596544231334545 3
 7 78817186773688818746539534439334445 3
 8 78877186771188868744535344535333525 4
 9 18877888789687868742532344324234445 4
10 78877888781688868744535344434334445 4
11 78777887781687866744235334334323545 3
12 78777838771189866742532144325354459 4
13 76766828579198868744195136423223419 4
14 18716887774187868744532344521633445 3
15 78817648711698867549239331433957945 7
16 26766561573388467744219334234954556 4
17 48119186781393826614795624413958459 4
18 72577881483258518514119364329694446 7
19 18867886381127926344512334425393459 6
20 72869861271928148326575528585194459 7
;
```

```
%let attrlist=Camera,Flip,Hands Free,Games,Internet
,Free Replacement,Battery Life,Large Letters,Applications;

%phchoice( on )

%mktmdiff(bw, nattrs=9, nsets=18, setsize=5, attrs=attrlist,
          data=bestworst, design=sasuser.bibd)
```

The DATA step reads 36 variables with data and a subject variable, which is ignored. The descriptions of each of the t attributes are listed in comma-delimited form and stored in a macro variable. (Note that when the list is split across lines, care is taken to ensure that the next attribute description, "Free Replacement" immediately follows the comma so that it will not begin with a leading blank.) The `%PHChoice` macro is used to customize the output from PROC PHREG, which the `%MktMDiff` macro calls, to look like the output from a discrete choice procedure instead of a survival analysis procedure. The `%MktMDiff` macro begins with a positional parameter that specifies the layout of the data. Positional parameters do not begin with a keyword and an equal sign. Because of the initial `bw` specification, the data are best then worst. The variables do not alternate. The data are attribute numbers not positions. The design has `nsets=18` sets, `nattrs=9` attributes, and `setsize=5` are shown at a time. The `attrlist` macro variable contains the list of the attributes. Note that a variable name and not the value of the variable are specified. In other words, `attrlist` not `&attrlist` is specified. The SAS data set with the data is called `bestworst`, and the SAS data set with the design is called `sasuser.bibd`.

The `%MktMDiff` macro begins by displaying the following summary of the input:

```
Var Order:   Best then Worst
Alternating: Variables Do Not Alternate
Data:        Attribute Numbers (Not Positions)
Best Vars:   b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15 b16 b17 b18
Worst Vars:  w1 w2 w3 w4 w5 w6 w7 w8 w9 w10 w11 w12 w13 w14 w15 w16 w17 w18
Attributes:  Camera
             Flip
             Hands Free
             Games
             Internet
             Free Replacement
             Battery Life
             Large Letters
             Applications
```

The data consists of the best variables then the worst variables. The best variables are `b1-b18,` and the worst variables are `w1-w18`. The data are attribute numbers (1 to $t = 9$) not positions (1 to $k = 5$). Finally, the descriptions of each of the t attributes are listed.

The following table is of interest as a check of the integrity of the input data:

Summary of Subjects, Sets, and Chosen and Unchosen Alternatives

Pattern	Number of Choices	Number of Alternatives	Chosen Alternatives	Not Chosen
1	36	100	20	80

In the aggregate data set, there is one pattern of input and it occurs 36 times (18 best choices per subject plus 18 worst choices). There are 100 alternatives (5 attributes in a set examined by 20 individuals), each of the 20 subjects chose 1 ($20 \times 1 = 20$) and did not choose 4 ($20 \times 4 = 80$).

The final table of results is as follows:

Best Worst Example with Cell Phone Attributes
Multinomial Logit Parameter Estimates

	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq
Large Letters	0	0	.	.	.
Battery Life	1	-0.52009	0.17088	9.2635	0.0023
Free Replacement	1	-1.01229	0.17686	32.7598	<.0001
Camera	1	-1.30851	0.18232	51.5096	<.0001
Applications	1	-1.93107	0.18741	106.1680	<.0001
Flip	1	-2.00892	0.18788	114.3334	<.0001
Internet	1	-2.44592	0.18731	170.5156	<.0001
Hands Free	1	-2.47063	0.18781	173.0569	<.0001
Games	1	-2.86329	0.18802	231.9065	<.0001

The parameter estimates are arranged from most preferred to least preferred. These results are also available in a SAS data set called `parmest`.

You could change the reference level using the `classopts=` option as follows:

```
%mktmdiff(bw, nattrs=9, nsets=18, setsize=5,
          attrs=attrlist, classopts=zero='Internet',
          data=bestworst, design=sasuser.bibd)
```

The new parameter estimates table is as follows:

Best Worst Example with Cell Phone Attributes
Multinomial Logit Parameter Estimates

	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq
Large Letters	1	2.44592	0.18731	170.5156	<.0001
Battery Life	1	1.92583	0.18533	107.9789	<.0001
Free Replacement	1	1.43363	0.18658	59.0424	<.0001
Camera	1	1.13741	0.18484	37.8650	<.0001
Applications	1	0.51485	0.18056	8.1305	0.0044
Flip	1	0.43700	0.18045	5.8650	0.0154
Hands Free	1	-0.02471	0.17655	0.0196	0.8887
Games	1	-0.41737	0.17375	5.7702	0.0163

Note that the parameter estimates all change by a constant amount. If you take the original estimate for the internet parameter and subtract it from all of the original estimates, you get the new estimates.

Experimental Design for a MaxDiff Study

The experimental design for a MaxDiff study is a block design. A set of t attributes are organized into b blocks or sets of size k. Ideally, we prefer to use a balanced incomplete block design or BIBD. In a BIBD, each of the t attributes appears the same number of times, and each of the t attributes appears with every other attribute the same number of times. However, BIBDs do not exist for every combination of b, t, and k that might be of interest. Hence, in some situations we are forced to use unbalanced block designs in which each of the t attributes appears the same number of times, but not all of the $t(t-1)/2$ pairs of attributes appear the same number of times. Fortunately, while a BIBD is nice, it is not required for a MaxDiff analysis.

There is a set of necessary but not sufficient conditions for the existence of a BIBD. See page 971. You can use the `%MktBSize` macro to get lists of designs that meet these conditions for ranges of b, t, and k, for example, as follows.

```
%mktbsize(nattrs=1 to 20, setsize=2 to 0.5 * t, nsets=t to 100)
```

This step also shows that you can restrict the sizes being considered using the values of b, t, and k. For example, this step will never consider a set size more than one half the number attributes. Nor will it consider designs with fewer sets than attributes. The results of this step are not shown.

Then you can use the `%MktBIBD` macro to find either a BIBD or an unbalanced block design, for example, as follows:

```
%mktbibd(nattrs=12, setsize=6, nsets=22, seed=104)
```

There is never a guarantee that the `%MktBIBD` macro will find a BIBD, even when one exists. It does a computerized search using PROC OPTEX. However, it does a good job of finding small BIBDs and

coming very close for larger ones—certainly close enough for most MaxDiff studies. However, just because you run the `%MktBIBD` macro and get something back does not mean it is suitable for use. Like any design, you need to examine its properties and ensure that it meets your needs. Consider, for example, the following step, which creates a design with 20 attributes shown in 16 sets of size 5:

```
%mktbibd(nattrs=20, setsize=5, nsets=16, seed=292, out=sasuser.maxdiffdes)
```

The attribute by attribute frequency matrix is as follows:

Attribute by Attribute Frequencies

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	4	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1
2		4	1	1	1	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1
3			4	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0
4				4	1	1	0	1	1	1	1	1	1	0	0	1	1	1	1	1
5					4	1	1	1	1	1	1	0	1	1	1	0	1	0	1	1
6						4	1	1	1	1	0	1	1	1	1	1	0	1	1	1
7							4	1	1	1	1	1	1	0	0	1	1	1	1	1
8								4	0	0	1	1	1	1	1	1	1	1	1	1
9									4	0	1	1	1	1	1	1	1	1	1	1
10										4	1	1	1	1	1	1	1	1	1	1
11											4	1	1	1	1	1	0	1	1	1
12												4	1	1	1	0	1	0	1	1
13													4	1	1	1	1	1	0	0
14														4	0	1	1	1	1	1
15															4	1	1	1	1	1
16																4	1	0	1	1
17																	4	1	1	1
18																		4	1	1
19																			4	0
20																				4

The design (not shown) has $bk = 16 \times 5 = 80$ entries, and each of the 20 attributes appears exactly 4 times. However, the off-diagonal attribute by attribute frequencies are not all the same, so this is not a BIBD. Furthermore, some of the attributes never appear with other attributes, so this design is not a good candidate for a MaxDiff study.

The `%MktBIBD` macro also tries to optimize the attribute by position frequencies. For this design, they are as follows:

Attribute by Position Frequencies

	1	2	3	4	5
1	1	1	0	1	1
2	0	1	1	1	1
3	1	1	0	1	1
4	1	1	1	1	0
5	1	0	1	1	1
6	1	1	1	0	1
7	1	1	0	1	1
8	1	1	1	0	1
9	1	0	1	1	1
10	0	1	1	1	1
11	1	1	1	1	0
12	1	1	1	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	0	1	1	1	1
16	1	1	1	1	0
17	1	1	1	1	0
18	1	0	1	1	1
19	1	0	1	1	1
20	0	1	1	1	1

Given the b, t, and k specifications, these frequencies are optimal. There are no 2's, and there is exactly one zero in each row, that is, each attribute appears in all but one position.

The following step requests 20 blocks or sets instead of 16:

```
%mktbibd(nattrs=20, setsize=5, nsets=20, seed=292, out=sasuser.maxdiffdes)
```

The number of sets is increased by 4 since $4k = t = 20$. You have to increase b by multiples of 4 (given this k and t) so that each attribute can appear an equal number of times. The attribute by attribute frequency matrix is as follows:

Attribute by Attribute Frequencies

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	5	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1
2		5	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1
3			5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1
4				5	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1
5					5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
6						5	1	0	1	1	1	1	1	1	1	2	1	1	2	1
7							5	1	1	1	1	1	1	1	1	1	1	1	1	1
8								5	1	1	1	1	1	1	1	2	1	1	2	1
9									5	1	1	1	1	1	2	1	1	1	1	1
10										5	1	1	2	1	1	1	1	1	1	1
11											5	1	1	1	1	1	2	1	1	1
12												5	1	1	1	1	1	1	1	1
13													5	1	1	1	1	1	1	1
14														5	1	1	1	1	1	1
15															5	1	1	1	1	1
16																5	1	1	0	1
17																	5	1	1	1
18																		5	1	1
19																			5	1
20																				5

Now each attribute appears 5 times, and the pairwise frequencies are all 1's and 2's with two 0's. Given the b, t, and k specifications, these frequencies are not quite what we hoped. There are more 2's than 0's, so ideally we would like the 0's to go away. The macro finds designs with this pattern of frequencies over and over, so it is likely that a design of this size with no zero frequencies is impossible. Block designs are subject to all sorts of Sudoku-like rules—if this treatment occurs with that treatment in this block, then it cannot occur with this other treatment in that other block. Every time PROC OPTEX tries to change a 0 frequency to a 1, it changes a 1 to a 0 rather than a 2 to a 1.

The attribute by position frequencies are as follows:

Attribute by Position Frequencies

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	1	1	1
9	1	1	1	1	1
10	1	1	1	1	1
11	1	1	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1
16	1	1	1	1	1
17	1	1	1	1	1
18	1	1	1	1	1
19	1	1	1	1	1
20	1	1	1	1	1

These are perfect. Each attribute appears in every position exactly once.

The following step requests 24 blocks or sets instead of 16 or 20:

```
%mktbibd(nattrs=20, setsize=5, nsets=24, seed=292, out=sasuser.maxdiffdes)
```

The attribute by attribute frequency matrix is as follows:

Attribute by Attribute Frequencies

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6	1	1	1	1	1	1	1	1	2	2	1	2	1	2	1	1	1	2	1
2		6	2	1	1	1	2	1	1	1	1	2	1	1	1	2	2	1	1	1
3			6	1	2	2	1	2	1	1	1	1	1	1	1	1	1	2	1	1
4				6	2	1	1	1	2	2	1	2	1	1	1	1	1	1	2	1
5					6	1	2	1	1	1	1	1	2	1	1	1	1	1	1	2
6						6	1	1	1	1	1	1	1	1	1	2	2	1	2	2
7							6	1	2	2	1	1	1	2	1	1	1	1	1	1
8								6	1	1	2	1	1	2	1	2	2	1	1	1
9									6	1	1	1	2	1	1	1	1	2	1	2
10										6	1	1	1	1	1	2	2	1	1	1
11											6	2	1	1	1	1	1	2	1	2
12												6	1	1	2	1	1	1	2	1
13													6	2	1	1	2	1	1	1
14														6	2	1	1	1	2	1
15															6	1	1	2	1	2
16																6	1	2	1	1
17																	6	1	1	1
18																		6	1	1
19																			6	1
20																				6

Now each attribute appears 6 times, and the pairwise frequencies are all 1's and 2's with no 0's.

The attribute by position frequencies are as follows:

Attribute by Position Frequencies

	1	2	3	4	5
1	1	1	2	1	1
2	1	2	1	1	1
3	1	1	1	2	1
4	1	1	1	1	2
5	2	1	1	1	1
6	1	1	2	1	1
7	1	1	1	2	1
8	1	1	2	1	1
9	1	2	1	1	1
10	1	2	1	1	1

11	1	1	1	2	1
12	1	1	1	1	2
13	2	1	1	1	1
14	1	2	1	1	1
15	2	1	1	1	1
16	1	1	1	2	1
17	2	1	1	1	1
18	1	1	1	1	2
19	1	1	1	1	2
20	1	1	2	1	1

Again, given the b, t, and k specifications, these frequencies are optimal. There are no 0's or 3's, and there is exactly one 2 in each row, that is, each attribute appears in all positions but one more than the others. Often times, we are content to use a design with patterns of frequencies like we see here, not perfect, but optimal given the design specifications.

You could run the following step to see if a BIBD is possible if you change the block size:

```
%mktbsize(nattrs=20, setsize=5, nsets=t to 500)
```

The results are as follows:

---

t Number of Attributes	k Set Size	b Number of Sets	r Attribute Frequency	Lambda Pairwise Frequencies	n Total Sample Size
20	5	76	19	4	380

You would need 76 sets before you would have any hope of finding a BIBD. If you are willing to change the number of messages or the number of messages shown at one time, then you have more possibilities. The following step investigates:

```
%mktbsize(nattrs=18 to 22, setsize=4 to 6, nsets=t to 500)
```

The results are as follows:

---

t Number of Attributes	k Set Size	b Number of Sets	r Attribute Frequency	Lambda Pairwise Frequencies	n Total Sample Size
18	4	153	34	6	612
18	5	306	85	20	1530
18	6	51	17	5	306

19	4	57	12	2	228
19	5	171	45	10	855
19	6	57	18	5	342
20	4	95	19	3	380
20	5	76	19	4	380
20	6	190	57	15	1140
21	4	105	20	3	420
21	5	21	5	1	105
21	6	28	8	2	168
22	4	77	14	2	308
22	5	462	105	20	2310
22	6	77	21	5	462

We can try to find a BIBD with 21 attributes, and 21 sets of size 5 as follows:

```
%mktbibd(nattrs=21, setsize=5, nsets=21, seed=292, out=sasuser.maxdiffdes)
```

The attribute by attribute frequency matrix is as follows:

Attribute by Attribute Frequencies

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2		5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3			5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4				5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5					5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6						5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7							5	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8								5	1	1	1	1	1	1	1	1	1	1	1	1	1
9									5	1	1	1	1	1	1	1	1	1	1	1	1
10										5	1	1	1	1	1	1	1	1	1	1	1
11											5	1	1	1	1	1	1	1	1	1	1
12												5	1	1	1	1	1	1	1	1	1
13													5	1	1	1	1	1	1	1	1
14														5	1	1	1	1	1	1	1
15															5	1	1	1	1	1	1
16																5	1	1	1	1	1
17																	5	1	1	1	1
18																		5	1	1	1
19																			5	1	1
20																				5	1
21																					5

The attribute by position frequencies are as follows:

Attribute by Position Frequencies

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	1	1	1
9	1	1	1	1	1
10	1	1	1	1	1
11	1	1	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1
16	1	1	1	1	1
17	1	1	1	1	1
18	1	1	1	1	1
19	1	1	1	1	1
20	1	1	1	1	1
21	1	1	1	1	1

These results are perfect. The art of design is determining what ranges of parameters work for you and then finding the best design that works for what you need.

%MktMDiff Macro Options

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

Option	Description
`layout`	(positional) choice data set structure
	(positional) "help" or "?" displays syntax summary
`attrs=`*macro-variable*	macro variable with attribute descriptions
`b=`*b*	number of sets (alias for `nsets=`)
`classopts=`*options-list*	class variable options
`data=`*SAS-data-set*	input data set
`design=`*SAS-data-set*	design data set
`group=`*k*	number of groups in the `design=` data set
`k=`*k*	set size (alias for `setsize=`)
`nattrs=`*t*	number of attributes (alias for `t=`)

Option	Description
`nsets=`*b*	number of sets (alias for `b=`)
`options=nocode`	just create the OUT= data set
`options=noanalysis`	just create the OUT= and OUTCODED= data sets
`options=nosort`	do not sort the parameter estimates table
`options=nodrop`	do not drop the variable or parameter column
`options=rescale`	do `rescale=` even when dubious
`out=`*SAS-data-set*	merged data and design data set
`outcoded=`*SAS-data-set*	coded data set
`outparm=`*SAS-data-set*	parameter estimates data set
`rescale=`*value*	table of rescaled parameter estimates
`setsize=`*k*	set size (alias for `k=`)
`t=`*t*	number of attributes or messages (alias for `nattrs=`)
`vars=`*variable-list*	`data=` data set variables

Help Option

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

```
%mktmdiff(help)
%mktmdiff(?)
```

Required Options

The `layout`, `b=` or `nsets=`, `k=` or `setsize=`, `t=` or `nattrs=`, and the `design=` options must be specified. The `layout` option is a positional parameter and must be specified first.

layout

specifies a nonoptional positional parameter that indicates the structure of the choice data. Just specify a value, not `layout=`. The value has several components, you can use any case, and spaces between the pieces are optional. Values include `b`, `w`, `alt`, and `pos`. Both a `b` and a `w` must be specified. When `b` comes before `w`, the best variables come before the worst variables. Otherwise, when `w` comes before `b`, the worst variables come before the best variables. You can optionally specify `alt` as well which means that the variables alternate (e.g. `b1 w1 b2 w2 b3 w3` where the `b` variables are the best variables and the `w` variables are the worst variables). Otherwise it is expected that all of one type appear then all of the other type. By default, the data are assumed to be the numbers of the attributes that were chosen (e.g. 1 to 6 when there are a total of `nattrs=6` attributes). If instead (say with `setsize=3`), the data are 1 to 3 indicating the position of the chosen attribute, then you must specify `pos` as well.

Examples:

`bw`
(or `b w`)
best then worst, e.g. `b1-b6 w1-w6` or even `x1-x12` (if $2 \times b = 12$). While the values need to be best then worst, the variable names do not need to reflect this. The data are attribute numbers.

`wb`
(or `w b`)
worst then best, e.g. `w1-w6 b1-b6` or even `x1-x12` (if $2 \times b = 12$). While the values need to be worst then best, the variable names do not need to reflect this. The data are attribute numbers.

`bwalt`
(or `b w alt` or `alt b w` and so on)
best then worst and alternating, e.g. `b1 w1 b2 w2 ...` or even `x1-x12` (if $2 \times b = 12$). While the values need to alternate best then worst, the variable names do not need to reflect this. The data are attribute numbers.

`wbalt`
(or `w b alt` or `alt w b` and so on)
worst then best and alternating, e.g. `w1 b1 w2 b2 ...` or even `x1-x12` (if $2 \times b = 12$). While the values need to alternate worst then best, the variable names do not need to reflect this. The data are attribute numbers.

`bwpos`
(or `b w alt` or `b pos w` and so on)
best then worst, and the data are positions.

`wbpos`
(or `w b pos` or `w pos b` and so on)
worst then best, and the data are positions.

`bwaltpos`
(or `b w alt pos` or `alt b pos w` and so on)
best then worst and alternating, and the data are positions.

`wbaltpos`
(or `w b alt pos` or `alt w pos b` and so on)
worst then best and alternating, and the data are positions.

design= *SAS-data-set*

specifies the data set with the design. It is typically a BIBD in the format produced by the `out=` option of the `%MktBIBD` macro. You must specify this option. All numeric variables are assumed to contain the BIBD, unless there is exactly one extra variable, and the first or last variable is called `Group`. In that case, the group variable is ignored.

nattrs= *t*
t= *t*

specifies the number of attributes or messages. The `nattrs=` and `t=` options are aliases. This option (in one of its two forms) must be specified. The "`t`" in `t=` stands for treatments and corresponds to typical BIBD notation.

nsets= *b*
b= *b*

specifies the number of sets. The `nsets=` and `b=` options are aliases. This option (in one of its two forms) must be specified. The "`b`" in `b=` stands for blocks and corresponds to typical BIBD notation.

setsize= *k*
k= *k*
specifies the number of attributes or messages shown at one time (in a set). The `setsize=` and `k=` options are aliases. This option (in one of its two forms) must be specified. The `k=` option is named using typical BIBD notation.

Data Set Options

The `design=` option is a required data set option that is listed in the previous section. The following are optional data set options.

data= *SAS-data-set*
specifies the input data set. By default, the last data set created is used.

out= *SAS-data-set*
specifies the output data set in a form ready for coding. The default is `out=maxdiff`.

outcoded= *SAS-data-set*
specifies the output data set that has been coded by PROC TRANSREG. The default is `outcoded=coded`.

outparm= *SAS-data-set* specifies the output data set with the parameter estimates table. The default is `outparm=parmest`. If the `rescale=` option is specified, the requested results are added to this data set.

Other Options

attrs= *macro-variable*
specifies the name of a macro variable that contains a comma delimited list of attribute descriptions. Alternatively, after the name you can specify a single nonblank delimiter character. Examples: `attrs=myattrlist`, `attrs=myattrs -`, (using a dash as a delimiter). If you do not specify this option, the default attribute names are "A1", "A2", and so on. The macro variable must have a name that does not match any of the macro parameter names (so you cannot specify `attrs=attrs` for example). The name also must not match any macro names used internally by the macro. In the unlikely event you specify an invalid name, an error is displayed. Most names, including any name that contains '`att`' that is not '`attrs`' or '`nattrs`' is always going to work.

classopts= *options-list*
specifies options to apply to the class variable, for example: `classopts=effects`. The class variable in the final analysis data set contains the descriptions of the attributes that are specified in the `attrs=` macro variable. The default is `classopts=zero=none`. A reference cell coding is used, and the reference level is displayed with a parameter estimate of zero. If you specify a null value (`classopts=`,) then the default reference cell coding is used and the reference level is not displayed. You can control the reference level by specifying `classopts=zero=`*'reference-level'* and naming the appropriate reference level. The reference level will exactly match one of the descriptions in the `attrs=` macro variable.

group= *g*

specifies the number of blocks of choice sets. By default, when `group=` is not specified, it is assumed that the design consists of one big group. Otherwise, with n subjects in each group and `group=`g there are $n \times g$ rows of data each consisting of b/g best values and b/g worst values. The number of subjects in each group must be the same. It is also assumed that the design is sorted by the group variable. If the design is made by the `%MktBIBD` macro, this will always be the case. If the `design=` data set has $k + 1$ variables and either the first variable or the last variable is called `Group` (case is ignored), then that variable is ignored and not treated as part of the design. Otherwise, the `design=` data set is required to have k variables.

options= *options-list*

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

`nocode`
just create the OUT= data set but do not code or do the analysis.

`noanalysis`
just create the OUT= and OUTCODED= data sets but do not do the analysis.

`nosort`
do not sort the parameter estimates table by the parameter estimates.

`nodrop`
do not drop the variable or parameter column from the parameter estimates table.

`rescale`
ignore the usual restriction that `rescale=` option can only be used in the context of the default `classopts=`. If you use this option, you must ensure that you are only using the `zero=` option to change the reference level or are otherwise doing something that will not change the coding from reference cell to something else.

rescale= *value*

specifies various ways to rescale the parameter estimates. You can specify any of the following values:

`default`
ordinary parameter estimates.

`center`
parameter estimates are centered.

`p`
parameter estimates are scaled to probabilities:

$$\frac{\exp(\hat{\beta}_i)}{\sum_{j=1}^{m} \exp(\hat{\beta}_j)}$$

`p100`
parameter estimates are scaled to probabilities then multiplied by 100:

$$\frac{100 \exp(\hat{\beta}_i)}{\sum_{j=1}^{m} \exp(\hat{\beta}_j)}$$

`adjusted`
`adj`
parameter estimates are adjusted by the number of attributes in each set. First, $\hat{\boldsymbol{\beta}}$ is centered then scaled as follows:

$$\frac{\exp(\hat{\beta}_i)}{\exp(\hat{\beta}_i) + k - 1}$$

Finally, the adjusted values are rescaled to sum to 1.

`adjusted100`
`adj100`
parameter estimates are adjusted by the number of attributes in each set. First, $\hat{\boldsymbol{\beta}}$ is centered then scaled as follows:

$$\frac{\exp(\hat{\beta}_i)}{\exp(\hat{\beta}_i) + k - 1}$$

Finally, the adjusted values are rescaled to sum to 100.

`all`
the default and all rescaled values are reported.

Note that `rescale=` and `rescale=default` are equivalent. When this option is specified (or any option besides `rescale=default` is specified), an additional table is displayed with the rescaled parameter estimates. You can specify multiple values and get multiple columns in the results table. Example: `rescale=default adj100`. All specified rescalings are added to the `outparm=` data set. These rescalings were suggested by Sawtooth Software (2005, 2007).

Do not use both the `rescale=` option and the `classopts=` option unless you are only changing details of the reference cell coding. If you use these options together, you must ensure that you are only using the `zero=` option to change the reference level or are otherwise doing something that will not change the coding from reference cell to something else. In that case, you can specify `options=rescale` to allow the analysis to proceed.

vars= *variable-list*
specifies the variables in the `data=` data set that contain the data. There must be $2 \times b$ variables in this list (from `nsets=`b). The default is all numeric variables in the data set.

%MktMDiff 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.