Chapter 15
Details of the FACTEX Procedure

Chapter Table of Contents

Chapter 15
Details of the FACTEX Procedure

Syntax

You can specify the following statements with the FACTEX procedure. Items within the brackets <> are optional.

PROC FACTEX <*options*> **;**
FACTORS *factor-names* < **/** *option*> **;**
SIZE *size-specification* **;**
MODEL *model-specification* <**MINABS**<*(d)*>>**;**
BLOCKS *block-specification* **;**
EXAMINE <*options*> **;**
OUTPUT OUT=*SAS-data-set* <*options*> **;**

To generate a design and save it in a data set, you use at least the PROC FACTEX, FACTORS, and OUTPUT statements. The FACTORS statement should immediately follow the PROC FACTEX statement. You use the MODEL and SIZE statements for designs that are less than a full replicate (for example, fractional factorial designs). You can use the BLOCKS statement for designs that involve blocking. The EXAMINE statement can be used as needed.

Summary of Functions

Table 15.1 to Table 15.4 classify the FACTEX statements and options by function.

Table 15.1. Summary of Options for Specifying the Design

Function	Statement	Option
Factor Specification		
Factor names	FACTORS	$factor_1 \ldots factor_f$
Number of levels	FACTORS	$factor_1 \ldots factor_f$ / NLEV=q
Design Size Specification		
(one of the following)		
Number of runs	SIZE	DESIGN=n
Fraction of one full replicate	SIZE	FRACTION=h
Number of *run indexing factors*	SIZE	NRUNFACS=m
Minimum number of runs	SIZE	DESIGN=MINIMUM or FRACTION=MAXIMUM or NRUNFACS=MINIMUM
Block Specification		
(one of the following)		
Number of blocks	BLOCKS	NBLOCKS=b
Block size	BLOCKS	SIZE=k
Number of *block pseudo-factors*	BLOCKS	NBLKFACS=s
Minimum block size	BLOCKS	NBLOCKS=MAXIMUM or SIZE=MINIMUM or NBLKFACS=MAXIMUM
Model Specification		
(one of the following)		
Estimated effects	MODEL	ESTIMATE=(*effects*)
Estimated effects and non-negligible effects	MODEL	ESTIMATE=(*effects*) NONNEG=(*nonnegligible-effects*)
Design resolution number	MODEL	RESOLUTION=r
Design with highest resolution	MODEL	RESOLUTION=MAXIMUM
Minimum aberration design (up to d^{th} order interactions)	MODEL	EST=(. . .) <NONNEG=(. . .)> or RES=. . . / MINABS<(d)>

Table 15.2. Summary of Options for Searching the Design

Function	Statement	Option
Search for the Design		
Allow maximum time of t seconds	PROC FACTEX	SECONDS=t or TIME=t
Limit the design searches	PROC FACTEX	NOCHECK

Table 15.3. Summary of Options for Replicating and Randomizing the Design

Function	Statement	Option
Replication		
Replicate entire design c times	OUTPUT OUT=*SAS-data-set*	DESIGNREP=c
Replicate design for each point in the data set	OUTPUT OUT=*SAS-data-set*	DESIGNREP=*SAS-data-set*
Replicate each point in design p times	OUTPUT OUT=*SAS-data-set*	POINTREP=p
Replicate data set for each point in the design	OUTPUT OUT=*SAS-data-set*	POINTREP=*SAS-data-set*
Randomization		
Randomize the design	OUTPUT OUT=*SAS-data-set*	RANDOMIZE
Randomize the design but not the assignment of factor levels	OUTPUT OUT=*SAS-data-set*	RANDOMIZE NOVALRAN
Specify seed number	OUTPUT OUT=*SAS-data-set*	RANDOMIZE (u)

Table 15.4. Summary of Options for Examining and Saving the Design

Function	Statement	Option
List the Design		
Coded factor and block levels	EXAMINE	DESIGN
List the Design Characteristics		
Alias structure (up to d^{th} order interactions)	EXAMINE	ALIASING<(d)>
Confounding rules	EXAMINE	CONFOUNDING
Save the Design		
Coded factor levels	OUTPUT OUT=*SAS-data-set*	
Decoded factor levels (numeric type)	OUTPUT OUT=*SAS-data-set*	*factor-name* NVALS=(*level1 . . . levelq*)
Decoded factor levels (character type)	OUTPUT OUT=*SAS-data-set*	*factor-name* CVALS=(*'level1' . . . 'levelq'*)
Block variable name	OUTPUT OUT=*SAS-data-set*	BLOCKNAME=*block-name*
Decoded block levels (numeric type)	OUTPUT OUT=*SAS-data-set*	BLOCKNAME=*block-name* NVALS=(*level1 . . . levelb*)
Decoded block levels (character type)	OUTPUT OUT=*SAS-data-set*	BLOCKNAME=*block-name* CVALS=(*'level1' . . . 'levelb'*)

Summary of Designs

Table 15.5 summarizes basic design types that you can construct with the FACTEX procedure by providing example code for each type.

Table 15.5. Basic Designs Constructed by the FACTEX Procedure

Design Type	Example Statements

A full factorial design in three factors, each at two levels coded as -1 and $+1$.	`proc factex;`
	`  factors` *pressure temp time* `;`
	`  examine design;`
	`run;`

A full factorial design in three factors, each at three levels coded as -1, 0, and $+1$.	`proc factex;`
	`  factors` *pressure temp time* `/nlev=` *3* `;`
	`  examine design;`
	`run;`

A full factorial design in three factors, each at two levels. The entire design is replicated twice, and the design with recoded factor levels is saved in a SAS data set.	`proc factex;`
	`  factors` *pressure temp time* `;`
	`  output out=` *savdesgn* `designrep=` *2*
	pressure `cvals=(` *'low' 'high'* `)`
	temp `nvals=(` *200 300* `)`
	time `nvals=(` *10 20* `);`
	`run;`

A full factorial design in three factors, each at two levels coded as -1 and $+1$. Each run in the design is replicated three times, and the replicated design is randomized and saved in a SAS data set.	`proc factex;`
	`  factors` *pressure temp time* `;`
	`  output out=` *savdesgn*
	`pointrep=`*3* `randomize;`
	`run;`

A full factorial design in three control factors, each at two levels coded as -1 and $+1$. A noise factor design (*outer array*) read from a SAS data set is replicated for each run in the control factor design (*inner array*), and the product design is saved in a SAS data set.	`proc factex;`
	`  factors` *pressure temp time* `;`
	`  output out =` *savdesgn*
	`pointrep=` *outarray* `;`
	`run;`

Design Type	Example Statements
A full factorial blocked design in three factors, each at two levels coded as -1 and $+1$. The design is arranged in two blocks and saved in a SAS data set. By default, the block variable is named BLOCK and the two block levels are numbered 1 and 2.	**proc factex;** **factors** *pressure temp time* **;** **blocks nblocks=** *2* **;** **output out=** *savdesgn* **;** **run;**
A full factorial blocked design in three factors, each at two levels coded as -1 and $+1$. Each block contains four runs; the block variable is renamed and the block levels of character type are recoded. The design is saved in a SAS data set.	**proc factex;** **factors** *pressure temp time* **;** **blocks size=** *4* **;** **output out=** *savdesgn* **blockname=** *machine* **cvals(** *'A' 'B'* **);** **run;**
A fractional factorial design of resolution 4 in four factors, each at two levels coded as -1 and $+1$. The size of the design is eight runs.	**proc factex;** **factors** *pressure temp time catalyst* **;** **size design=** *8* **;** **model resolution=** *4* **;** **examine design;** **run;**
A one-half fraction of a factorial design in four factors, each at two levels coded as -1 and $+1$. The design is of maximum resolution. The design points, the alias structure, and the confounding rules are listed.	**proc factex;** **factors** *pressure temp time catalyst* **;** **size fraction=** *2* **;** **model resolution=maximum;** **examine design aliasing confounding;** **run;**
A one-quarter fraction of a factorial design in six factors, each at two levels coded as -1 and $+1$. Main effects are estimated, and some two-factor interactions are considered nonnegligible. The design is saved in a SAS data set.	**proc factex;** **factors** *x1-x6* **;** **size fraction=** *4* **;** **model estimate=(** *x1 x2 x3 x4 x5 x6* **)** **nonneg =(** *x1*x5 x1*x6 x5*x6* **);** **output out =** *savdesgn* **;** **run;**

Statement Descriptions

This section provides detailed syntax information for the FACTEX procedure statements, beginning with the PROC FACTEX statement. The remaining statements are presented in alphabetical order.

PROC FACTEX Statement

PROC FACTEX <*options*> **;**

You use the PROC FACTEX statement to invoke the FACTEX procedure. The following *options* are available:

NAMELEN
specifies the length of effect names in tables and output data sets to be n characters long, where n is a value between 20 and 200 characters. The default length is 20 characters.

NOCHECK
suppresses a technique for limiting the amount of search required to find a design. The technique dramatically reduces the search time by pruning branches of the search tree that are unlikely to contain the specified design, but in rare cases it can keep the FACTEX procedure from finding a design that does, in fact, exist. The NOCHECK option turns off this technique at the potential cost of an increase in run time. Note, however, that the run time is always bounded by the TIME= option or its default value. For more on the NOCHECK option, see "Speeding up the Search" on page 507.

TIME=t
SECONDS=t
specifies the maximum number of seconds to spend on the search. The default is 60 seconds.

BLOCKS Statement

BLOCKS *block-specification* **;**
You use the BLOCKS statement to specify the number of blocks in the design or the size of each block in the design. By default, the FACTEX procedure constructs designs that do not contain blocks. If you use the BLOCKS statement, you also need to use the MODEL statement or SIZE statement. In particular, if you use the BLOCKS statement and your design is a fractional factorial design, you must use the MODEL statement.

The two simplest explicit *block-specifications* that you can use are

- NBLOCKS=*b*, which specifies the number of blocks (*b*) in the design
- SIZE=*k*, which specifies the number of runs (*k*) in each block

Use only one of these two options. In all, there are six mutually exclusive *block-specifications* that you can use, as described by the following list:

NBLKFACS=s

specifies the number of *block pseudo-factors* for the design. The design contains a different block for each possible combination of the levels of the block pseudo-factors. Values of s are the integers 1, 2, and so on. See "Block Size Restrictions" on page 448 for details.

If each factor in the design has q levels, then NBLKFACS=s specifies a design with q^s blocks. The size of each block depends on the number of runs in the design, as specified in the SIZE statement. If the design has n runs, then each block has n/q^s runs.

The following statement illustrates how to request a two-level factorial design arranged in eight (2^3) blocks:

```
blocks nblkfacs=3;
```

For more on pseudo-factors, see "Types of Factors" on page 488.

NBLOCKS=b

specifies the number of blocks in the design. The values of b must be a power of q, the number of levels of each factor in the design. See "Block Size Restrictions" on page 448 for details. The size of each block depends on the number of runs in the design, as specified in the SIZE statement. If the design has n runs, then each block has n/b runs. See page 433 for an illustration of this option.

The following statement illustrates how to specify a design arranged in four blocks:

```
blocks nblocks=4;
```

SIZE=k

specifies the number of runs per block in the design. The value k must be a power of q, the number of levels for each factor in the design. The number of blocks depends on the number of runs in the design, as specified in the SIZE statement. If the design has n runs, then it has n/k blocks.

CAUTION: Do not confuse the SIZE= option in the BLOCKS statement with the SIZE statement, which you use to specify the overall size of the design. See page 455 for details of the SIZE statement.

The following statement illustrates how to specify blocks of size two:

```
blocks size=2;
```

NBLKFACS=MAXIMUM
NBLOCKS=MAXIMUM
SIZE=MINIMUM

constructs a blocked design with the minimum number of runs per block, given all the other characteristics of the design. In other words, the block size is optimized. You cannot specify this option if you specify any of the design size optimization options in the SIZE statement (see DESIGN=MINIMUM on page 456).

Equivalence of Specifications

The three explicit *block-specifications* are related to each other, as demonstrated by the following example.

Suppose you want to construct a design for 11 two-level factors in 128 runs in blocks of size 8. Since $128/2^4 = 128/16 = 8$, three equivalent block specifications are

```
blocks nblkfacs=4;
blocks nblocks=16;
blocks size=8;
```

Block Size Restrictions

The number of blocks and the number of runs in each block must be less than the total number of runs in the design. Hence, there are some restrictions on the block size.

- If you use SIZE=k or NBLOCKS=b, the numbers you specify for k and b must be less than or equal to the size of the design, as specified in the SIZE statement. Or, if you do not use a SIZE statement, k and b must be less than or equal to the number of runs for a full replication of all possible combinations of the factors.

  For example, for a 2^3 design you cannot specify a design arranged in 8 blocks (NBLOCKS=8). Likewise, you cannot construct a design with block size greater than 8 (SIZE=8).

- If you use NBLKFACS=s, the value of s can be no greater than the number of *run-indexing factors*, which give the number of runs needed to index the design. For details, see "Types of Factors" on page 488 and Chapter 16, "Theory of Orthogonal Designs" on page 503.

EXAMINE Statement

EXAMINE <*options*> **;**

You use the EXAMINE statement to specify the characteristics of the design that are to be listed in the output.

The *options* are remembered by the procedure; once specified, they remain in effect until you submit a new EXAMINE statement with different options or until you turn off all EXAMINE options by submitting just

```
examine;
```

The following *options* are available.

ALIASING$< (d) >$
A$< (d) >$
lists the alias structure of the design, which identifies effects that are confounded with one another and are thus indistinguishable.

You can specify (d) immediately after the ALIASING option for a listing of the alias structure with effects up to and including order d. For example, the following statement requests aliases for up to fourth-order effects (for example, A*B*C*D):

```
examine aliasing(4);
```

Each line of the alias structure is listed in the form

effect=*effect*= . . . =*effect*

for as many effects as are aliased with one another.

The default value for d is determined automatically from the model as follows:

- If you specify the model with a resolution number r in the MODEL statement, then d is the integer part of $(r + 1)/2$.
- If you specify the model with a list of effects in the MODEL statement, then d is the larger of
  - one plus the largest order of an effect to be estimated
  - the largest order of an effect considered to be nonnegligible

where main effects have order 1, two-factor interactions have order 2, and so on. For details on aliasing, see "Alias Structure" on page 496.

CONFOUNDING
C

lists the confounding rules used to construct the design. For the definition of confounding rules, see "Confounding Rules" on page 496 and "Suitable Confounding Rules" on page 504.

DESIGN
D

lists the points in the design in standard order with the factor levels coded. For a description of the randomization and coding rules, see "OUTPUT Statement" on page 452.

FACTORS Statement

FACTORS *factor-names* < / *option*> **;**

You use the FACTORS statement to start the construction of a new design by naming the factors in the design. The FACTORS statement clears all previous specifications for the design (number of runs, block size, and so on). Use it when you want to start a new design. **Note that the FACTORS statement should be the first statement following the PROC FACTEX statement.**

In the FACTORS statement,

factor-names
lists names for the factors in the design. These names must be valid SAS variable names. See "Types of Factors" on page 488 for details.

The following *option* is available:

NLEV=q
specifies the number of levels for each factor in the design. The value of q must be an integer greater than or equal to 2. **The default value for q is 2.** In order to construct a design that involves either fractionation or blocking, q must be either a prime number or an integer power of a prime number. For the reason behind this restriction, see "Structure of General Factorial Designs" on page 503.

MODEL Statement

MODEL *model-specification* $<$**MINABS** $<$ (d) $>>$ **;**

You use the MODEL statement to provide the model for the construction of the factorial design. The model can be specified either directly by specifying the effects to be estimated with the ESTIMATE= option or indirectly by specifying the resolution of the design with the RESOLUTION= option. **If you create a fractional factorial design or if you create a design that involves blocking, the MODEL statement is required.**

The two *model-specifications* are described as follows:

ESTIMATE=(*effects***)** $<$*option*$>$
identifies the *effects* that you want to estimate with the design. To specify *effects*, simply list the names of main effects, and join terms in interactions with asterisks. The *effects* listed must be enclosed within parentheses. See "Specifying Effects in the MODEL Statement" on page 489 for details. You can use EST or E for the keyword ESTIMATE.

After the ESTIMATE= *option*, you can specify the following *option*:

NONNEGLIGIBLE=(*nonnegligible-effects***)**
identifies nonnegligible effects. These are the effects whose magnitudes are unknown, but you do not necessarily want to estimate them with the design. If you do not want certain effects to be aliased with ESTIMATE= effects, then list them in the NONNEGLIGIBLE= effects. The *nonnegligible-effects* listed must be enclosed within parentheses.

You can use NONNEG or N for the keyword NONNEGLIGIBLE.

For example, suppose that you want to construct a fraction of a 2^4 design in order to estimate the main effects of the four factors. To specify the model, simply list the main effects with the EFFECTS= option, since these are the effects of interest. Furthermore, if you consider the two-factor interactions to be significant but are not interested in estimating them, then list these interactions with the NONNEGLIGIBLE= option.

See Example 15.8 on page 468 for an example using the ESTIMATE= option. See page 503 for details on how the FACTEX procedure interprets the model and derives an appropriate confounding scheme.

RESOLUTION=r
RESOLUTION=MAXIMUM

specifies the resolution of the design. The resolution number r must be a positive integer greater than or equal to 3. The interpretation of r is as follows:

- If r is odd, then the effects of interest are taken to be those of order $(r-1)/2$ or less.
- If r is even, then the effects of interest are taken to be those of order $(r-2)/2$ or less, and the nonnegligible effects are taken to be those of order $r/2$ or less.

If you specify RESOLUTION=MAXIMUM, the FACTEX procedure searches for a design with the highest resolution that satisfies the SIZE statement requirements.

You can use RES or R for the keyword RESOLUTION and MAX for MAXIMUM.

For more on design resolution, see "Resolution" on page 491. For an example of *model specification* using the RESOLUTION=r option, see page 435. For an example of the RESOLUTION=MAX option, see page 433.

MINABS $<(d)>$

requests a search for a design that has minimum aberration. Specifying (d) immediately after the MINABS option requests a search for a minimum aberration design involving interactions up to order d. The default value for d is the same as for the ALIASING option in the EXAMINE statement. See "Minimum Aberration" on page 497 for more information. For an example of the MINABS option, see Example 15.11 on page 472.

Examples of the MODEL Statement

Suppose you specify a design with the following FACTORS statement, where the number of factors f can be replaced with a number:

```
factors x1-xf;
```

Then Table 15.6 lists equivalent ways to specify common models.

Table 15.6. Equivalent of Model Specifications

RES= option	EST= and NONNEG= options
`model res=3`	`model est=(x1-xf);`
`model res=4`	`model est=(x1-xf) nonneg=(x1|x2|x3|...|xf@2);`
`model res=5`	`model est=(x1|x2|x3|...|xf@2);`

The resolution specification is more concise than the effects specification and is also more efficient in an algorithmic sense. To decrease the time required to find a design, particularly for designs with a large number of factors, you should specify your model using the RESOLUTION= option rather than listing the effects. For more information on interpreting the resolution number, see "Resolution" on page 491.

OUTPUT Statement

OUTPUT OUT= *SAS-data-set* <*options*> **;**

You use the OUTPUT statement to save a design in an output data set. Optionally, you can use the OUTPUT statement to modify the design by specifying values to be output for factors, creating new factors, randomizing the design, and replicating the design. You specify the output data set as follows:

OUT=*SAS-data-set*
gives the name of the output data set in which the design is saved. Note that OUT= is required.

options
You can use the *options* to

- recode the values for design factors
- recode the values for the block variable
- replicate the entire design
- replicate each point of the design
- randomize the design
- create derived factors based on the original factors

The following list describes the preceding *options*:

Recode Design Factors

By default, the output data set contains a variable for each factor in the design coded with standard values, as follows:

- For factors with 2 levels ($q = 2$), the values are -1 and +1.
- For factors with 3 levels ($q = 3$), the values are -1, 0, and +1.
- For factors with q levels ($q > 3$), the values are $0, 1, 2, \ldots q - 1$.

You can recode the levels of the factor from the standard levels to levels appropriate for your situation.

For example, suppose that you want to recode a three-level factorial design from the standard levels -1, 0, and +1 to the actual levels. Suppose the factors are pressure (PRESSURE) with character levels, agitation rate (RATE) with numeric levels, and temperature (TEMP) with numeric levels. You can use the following statement to recode the factor levels and save the design in a SAS data set named RECODE:

```
output out=recode pressure cvals=('low' 'medium' 'high')
                  rate     nvals=(20    40       60    )
                  temp     cvals=(100   150      200   );
```

The general form of *options* to recode factors is as follows:

factor-name **NVALS= (***level1 level2 . . . levelq***)**
or

factor-name **CVALS= (***'level1' 'level2' . . . 'levelq'***)**
where

factor-name	gives the name of the design factor.
NVALS=	lists new numeric levels for design factors.
CVALS=	lists new character levels for design factors. Each string can be up to 40 characters long.

When recoding a factor, the NVALS= and CVALS= options map the first value listed to the lowest value for the factor, the second value listed to the next lowest value, and so on. If you rename and recode a factor, the type and length of the new variable are determined by whether you use the CVALS= option (character variable with length equal to the longest string) or the NVALS= option (numeric variable). For more on recoding a factor, see "Factor Variable Characteristics in the Output Data Set" on page 490.

Recode Block Factor

If the design uses blocking, the output data set automatically contains a block variable named BLOCK, and for a design with b blocks, the default values of the block variable are $1, 2, \ldots b$. You can rename the block variable and optionally recode the block levels from the default levels to levels appropriate for your situation.

For example, for a design arranged in four blocks, suppose that the block variable is day of the week (DAY) and that the four block levels of character type are *Mon*, *Tue*, *Wed*, and *Thu*. You can use the following statement to rename the block variable, recode the block levels, and save the design in a SAS data set named RECODE:

```
output out=recode
       blockname=day cvals=('Mon' 'Tue' 'Wed' 'Thu');
```

The general form of *options* to change the block variable name or change the block levels is as follows:

BLOCKNAME= *block-name* <**NVALS= (***level1 level2 . . . levelb***)**>
or
BLOCKNAME= *block-name* <**CVALS= (***'level1' 'level2' . . . 'levelb'***)**>

where

block-name	gives a new name for the block factor.
NVALS=	lists new numeric levels for the block factor. For details, see "Recode Design Factors" on page 452.
CVALS=	lists new character levels for the block factor. For details, see "Recode Design Factors" on page 452.

Note that you can simply rename the block variable using only the BLOCKNAME= option, without using the NVALS= and CVALS= options.

Replicate Entire Design

DESIGNREP=*c*

DESIGNREP=*SAS-data-set*

replicates the entire design. Specify DESIGNREP=*c* to replicate the design c times, where c is an integer. Alternatively, you can specify a SAS data set with the DESIGNREP option. In this case, the design is replicated once for each point in the DESIGNREP= data set, and the OUT= data set contains the variables in the DESIGNREP= data set as well as the design variables.

In mathematical notation, the OUT= data set is the direct product of the DESIGNREP= data set and the design. If the design is A and the DESIGNREP= data set is B, then the OUT= data set is B $\otimes$ A, where $\otimes$ denotes the direct product.

For details, see "Replication" on page 494. For illustrations of the difference between the DESIGNREP= and POINTREP= options, see Example 15.6 on page 464 and Example 15.7 on page 467.

Replicate Design Point

POINTREP=*p*

POINTREP=*SAS-data-set*

replicates each point of the design. Specify POINTREP=*p* to replicate each design point p times, where p is an integer. Alternatively, you can specify a SAS data set with the POINTREP= option. In this case, the POINTREP= data set is replicated once for each point in the design and the OUT= data set contains the variables in the POINTREP= data set as well as the design variables.

In mathematical notation, the OUT= data set is the direct product of the design and the POINT= data set. If the design is A and the POINTREP= data set is B, then the OUT= data set is A $\otimes$ B, where $\otimes$ denotes the direct product.

For details, see "Replication" on page 494. For illustrations of the difference between the DESIGNREP= and POINTREP= options, see Example 15.6 on page 464 and Example 15.7 on page 467.

Randomize Design

RANDOMIZE $<(u)>$ $<$ **NOVALRAN** $>$

randomizes the design. See "Randomization" on page 492 for details. The following *options* are available:

(u)

specifies a number u to start the pseudo-random number generator. The value of u must be enclosed in parentheses immediately after the keyword RANDOMIZE, and it can be any positive integer up to $2^{31} - 1$. The default value of u is generated from the time of day.

NOVALRAN

prevents the randomization of theoretical factor levels to actual levels. The randomization of run order is still performed.

Create Derived Factors

You can create *derived factors* based on the joint values of a set of the design factors. Each distinct combination of levels of the design factors corresponds to a single level for the derived factor. Thus, when you create a derived factor from k design factors, each with q levels, the derived factor has q^k levels. Derived factors are useful when you create mixed-level designs; see Example 15.8 on page 468 for an example. See "Structure of General Factorial Designs" on page 503 for information on how the levels of design factors are mapped into levels of the derived factor. The general form of the *option* for creating derived factors is

[*design-factors*]**=** *derived-factor* < **NVALS= (***list-of-numbers***)**>

or

[*design-factors*]**=** *derived-factor* < **CVALS= (***'string1' 'string2' . . . 'stringn'***)**>

where

design-factors gives names of factors currently in the design. These factors are combined to create the new derived factor.

derived-factor gives a name to the new derived factor. This name must not be used in the design.

NVALS= lists new numeric levels for the derived factor.

CVALS= lists new character levels for the derived factor. See "Recode Design Factors" on page 452 for details.

If you create a derived factor and do not use the NVALS= or CVALS= option to assign levels to the derived factor, the FACTEX procedure assigns the values $0, 1, \ldots, q^k - 1$, where the derived factor is created from k design factors, each with q levels. In general, the CVALS= or NVALS= list for a derived factor must contain q^k values.

The following statement gives an example of creating a derived factor and then renaming the levels of the factor:

```
output out=new [a1 a2]=a cvals=('A' 'B' 'C' 'D');
```

This statement converts two two-level factors (A1 and A2) into one four-level factor (A), which has the levels A, B, C, and D.

SIZE Statement

SIZE *size-specification* **;**

You use the SIZE statement to specify the size of the design, which is the number of runs in the design. The SIZE statement is required for designs of less than a full replicate (for example, fractional factorial designs). By default, the design consists of one full replication of all possible combinations of the factors.

The two simplest explicit *size-specifications* that you can use are

- DESIGN=n, which specifies the number of runs (n) in the design
- FRACTION=h, which specifies $1/h$ of one full replicate

Use only one of these two options. In all, there are six mutually exclusive *size-specifications* that you can use, as described by the following list:

DESIGN=n
specifies the actual number of runs in the design. The number of runs must be a power of the number of levels q for the factors in the design. (See the NLEV= option on page 450). If the last FACTORS statement does not contain the NLEV= option, then $q = 2$ by default, and as a result, n must be a power of 2. For an example, see page 457.

FRACTION=h
specifies the fraction of one full replication of all possible combinations of the factors. For instance, FRACTION=2 specifies a half-fraction, and FRACTION=4 specifies a quarter-fraction, and so on. In general, FRACTION=h specifies a design with 1/h of the runs in a full replicate. If the design has f factors, each with q levels, then the size of the design is q^f/h. If you use FRACTION=h, h must be a power of q. See Example 15.4 on page 461.

NRUNFACS=m
specifies the number of *run-indexing factors* in the design. The design contains one run for each possible combination of the levels of the run-indexing factors. Run-indexing factors are the first m factors for a design in q^m runs. All possible combinations of the levels of the run-indexing factors occur in the design. As a result, if each factor has q levels, the number of runs in the design is q^m. For details on run-indexing factors, see "Types of Factors" on page 488 and "Structure of General Factorial Designs" on page 503.

DESIGN=MINIMUM
FRACTION=MAXIMUM
NRUNFACS=MINIMUM
constructs a design with the minimum number of runs (no larger than one full replicate) given all of the other characteristics of the design. In other words, the design size is optimized. You cannot specify this option if you specify any of the block size optimization features in the BLOCKS statement (see NBLKFACS=MAXIMUM on page 447).

Equivalence of Specifications

The three explicit *size-specifications* are related to each other, as demonstrated by the following example. Suppose you want to construct a design for 11 two-level factors in 128 runs. Since $128 = 2^{11}/16 = 2^7$, three equivalent size specifications for this design are

```
size design=128;
size fraction=16;
size nrunfacs=7;
```

Advanced Examples

Example 15.1. Completely Randomized Design

See FACTEX8 in the SAS/QC Sample Library

An experimenter wants to study the effect of cutting speed (SPEED) on the surface finish of a component. He considers testing the components at five levels of cutting speed (100, 125, 150, 175, and 200) and decides to test five components at each level.

The design used is a single-factor *completely randomized design* with five levels and 25 runs. The following statements generate the required design:

```
proc factex;
   factors speed / nlev=5;
   size design=25;
   output out=surfexpt randomize        /* Randomly assign run order */
      speed nvals=(100 125 150 175 200);
run;

proc print data=surfexpt;
run;
```

The design saved in the data set SURFEXPT is displayed in Output 15.1.1.

Output 15.1.1. A Completely Randomized Design

Obs	speed
1	150
2	200
3	150
4	125
5	125
6	175
7	200
8	125
9	100
10	175
11	100
12	100
13	100
14	150
15	125
16	200
17	150
18	150
19	175
20	100
21	175
22	200
23	125
24	175
25	200

If you are working through this example on your computer, you might find a different run order in your output. This is due to the difference in the seed value of the random number generator. You can specify a seed value with the RANDOMIZE option. For syntax, see "Randomize Design" on page 454.

Example 15.2. Resolution IV Augmented Design

See RCBD in the SAS/QC Sample Library

Box, Hunter, and Hunter (1978) describe an injection molding experiment involving eight two-level factors: mold temperature (TEMP), moisture content (MOIST), holding pressure (HOLDPR), cavity thickness (THICK), booster pressure (BOOSTPR), cycle time (TIME), screw speed (SPEED), and gate size (GATE).

The design used has 16 runs and is of resolution 4; it is often denoted as 2^{8-4}_{IV}. You can generate this design, shown in Output 15.2.1, with the following statements:

```
proc factex;
   factors temp    moist holdpr thick    /* List factor names        */
           boostpr time  speed  gate;
   size design=16;                       /* Construct 16-run design  */
   model resolution=4;                   /*    of resolution 4       */
   examine design aliasing;              /* List points and aliasing */
run;
```

Output 15.2.1. A 2^{8-4}_{IV} Design

Design Points

Experiment Number	temp	moist	holdpr	thick	boostpr	time	speed	gate
1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	1	1	1	1	-1
3	-1	-1	1	-1	1	1	-1	1
4	-1	-1	1	1	-1	-1	1	1
5	-1	1	-1	-1	1	-1	1	1
6	-1	1	-1	1	-1	1	-1	1
7	-1	1	1	-1	-1	1	1	-1
8	-1	1	1	1	1	-1	-1	-1
9	1	-1	-1	-1	-1	1	1	1
10	1	-1	-1	1	1	-1	-1	1
11	1	-1	1	-1	1	-1	1	-1
12	1	-1	1	1	-1	1	-1	-1
13	1	1	-1	-1	1	1	-1	-1
14	1	1	-1	1	-1	-1	1	-1
15	1	1	1	-1	-1	-1	-1	1
16	1	1	1	1	1	1	1	1

The alias structure is shown in Output 15.2.2.

Output 15.2.2. Alias Structure for a 2^{8-4}_{IV} Design

```
Aliasing Structure

 temp
 moist
 holdpr
 thick
 boostpr
 time
 speed
 gate
 temp*moist = holdpr*gate = thick*speed = boostpr*time
 temp*holdpr = moist*gate = thick*time = boostpr*speed
 temp*thick = moist*speed = holdpr*time = boostpr*gate
 temp*boostpr = moist*time = holdpr*speed = thick*gate
 temp*time = moist*boostpr = holdpr*thick = speed*gate
 temp*speed = moist*thick = holdpr*boostpr = time*gate
 temp*gate = moist*holdpr = thick*boostpr = time*speed
```

Subsequent analysis of the data collected for the design suggests that HOLDPR and BOOSTPR have statistically significant effects. There also seems to be significant effect associated with the sum of the aliased two-factor interactions TEMP*BOOSTPR, MOIST*TIME, HOLDPR*SPEED, and THICK*GATE. This chain of confounded interactions is identified in Output 15.2.2.

A few runs can be added to the design to distinguish between the effects due to these four interactions. You simply need a design in which any three of these effects are estimable, regardless of all other main effects and interactions. For example, the following statements generate a suitable set of runs (see Output 15.2.3):

```
proc factex;
   factors temp    moist holdpr thick
           boostpr time  speed  gate;
   model estimate=(moist*time
                   holdpr*speed
                   thick*gate  );
   size design=4;
   examine design aliasing(2);
run;
```

Output 15.2.3. Additional Runs to Resolve Ambiguities

Design Points

Experiment Number	temp	moist	holdpr	thick	boostpr	time	speed	gate
1	-1	-1	1	1	1	1	-1	1
2	-1	1	-1	-1	-1	-1	-1	1
3	1	-1	-1	-1	-1	-1	1	1
4	1	1	1	1	1	1	1	1

Output 15.2.4 shows the alias structure of the additional four-run experiment. Note that the alias link

TEMP*BOOSTPR = MOIST*TIME = HOLDPR*SPEED = THICK*GATE

found in the original design is broken. When these four runs are added to the original 16 runs, the four two-factor interactions can be estimated separately with the 20 runs.

Output 15.2.4. Alias Structure of the Additional Experiment

```
Aliasing Structure

 0 = gate = temp*speed = holdpr*thick = holdpr*boostprs = holdpr*time
   = thick*boostprs = thick*time = boostprs*time
 temp = speed = temp*gate = moist*holdpr = moist*thick = moist*boostprs
      = moist*time = speed*gate
 moist = temp*holdpr = temp*thick = temp*boostprs = temp*time = moist*gate
       = holdpr*speed = thick*speed = boostprs*speed = time*speed
 holdpr = thick = boostprs = time = temp*moist = moist*speed = holdpr*gate
        = thick*gate = boostprs*gate = time*gate
```

Example 15.3. Factorial Design with Center Points

See FACTEX9 in the SAS/QC Sample Library

Factorial designs involving two levels are the most popular experimental designs. For two-level designs, it is assumed that the response is close to linear over the range of the factor levels. To check for curvature and to obtain an independent estimate of error, you can replicate points at the center of a two-level design. Adding center points to the design does not affect the estimates of factorial effects.

To construct a design with center points, you first create a data set with factorial points using the FACTEX procedure and then augment it with center points by using a simple DATA step. The following example illustrates this technique.

A researcher is studying the effect of three two-level factors—current (CURRENT), voltage (VOLTAGE), and time (TIME)—by conducting an experiment using a complete factorial design. The researcher is interested in studying the overall *curvature* over the range of factor levels by adding four center points.

You can construct this design in two stages. First, create the basic 2^3 design with the following statements:

```
proc factex;
   factors current voltage time;
   output out=factdat
          current nvals=(12  28 )
          voltage nvals=(100 200)
          time    nvals=(50  60 );
run;
```

Next, create the center points and append to the basic design as follows:

```
data center(drop=i);
   do i = 1 to 4;
      current = 20;
      voltage = 150;
      time    = 55;
      output;
   end;
data cpdesgn;
   set factdat center;
run;

proc print data=cpdesgn;
run;
```

The design saved in the data set CPDESIGN is displayed in Output 15.3.1. Observations 1 to 8 are the factorial points, and observations 9 to 12 are the center points.

Output 15.3.1. A 2^3 Design with Four Center Points

Obs	current	voltage	time
1	12	100	50
2	12	100	60
3	12	200	50
4	12	200	60
5	28	100	50
6	28	100	60
7	28	200	50
8	28	200	60
9	20	150	55
10	20	150	55
11	20	150	55
12	20	150	55

Example 15.4. Fold-Over Design

See FACTEX10 in the SAS/QC Sample Library

Folding over a fractional factorial design is a method for breaking the links between aliased effects in a design. Folding over a design means adding a new fraction identical to the original fraction except that the signs of all the factors are reversed. The new fraction is called a *fold-over* design. Combining a fold-over design with the original fraction converts a design of odd resolution r into a design of resolution $r + 1$.* For example, folding over a resolution 3 design yields a resolution 4 design. You can use the FACTEX procedure to construct the original design fraction and a DATA step to generate the fold-over design.

*This is not true if the original design has even resolution.

Consider a 1/8 fraction of a 2^6 factorial design with factors A, B, C, D, E, and F. The following statements construct a 2^{6-3}_{III} design:

```
proc factex;
   factors a b c d e f;
   size fraction=8;           /* Specify 1/8 fraction design */
   model resolution=3;        /*     of resolution 3         */
   examine aliasing;
   output out=original;
run;

title 'Original Design';
proc print data=original;
run;
```

The design, which is saved in the data set ORIGINAL, is displayed in Output 15.4.1.

Output 15.4.1. A 2^{6-3}_{III} Design

Original Design

Obs	a	b	c	d	e	f
1	-1	-1	-1	-1	1	1
2	-1	-1	1	1	-1	-1
3	-1	1	-1	1	-1	1
4	-1	1	1	-1	1	-1
5	1	-1	-1	1	1	-1
6	1	-1	1	-1	-1	1
7	1	1	-1	-1	-1	-1
8	1	1	1	1	1	1

Since the design is of resolution 3, the alias structure in Output 15.4.2 indicates that all the main effects are confounded with the two-factor interactions.

Output 15.4.2. Alias Structure for a 2^{6-3}_{III} Design

```
Aliasing Structure

 a = c*f = d*e
 b = c*e = d*f
 c = a*f = b*e
 d = a*e = b*f
 e = a*d = b*c
 f = a*c = b*d
 a*b = c*d = e*f
```

To separate the main effects and the two-factor interactions, augment the original design with a 1/8 fraction in which the signs of all the factors are reversed. The combined design (original design and fold-over design) of resolution 4 breaks the alias links between the main effects and the two-factor interactions. The fold-over design can be created using the following DATA step:

```
data foldover;                  /* Create the fold-over design with */
   set original;                /*    the factor signs reversed     */
   a=-a; b=-b; c=-c;
   d=-d; e=-e; f=-f;
run;

title 'Fold-Over Design';
proc print data=foldover;
run;
```

The fold-over design is displayed in Output 15.4.3.

Output 15.4.3. A 2^{6-3}_{III} Design with Signs Reversed

Fold-Over Design

Obs	a	b	c	d	e	f
1	1	1	1	1	-1	-1
2	1	1	-1	-1	1	1
3	1	-1	1	-1	1	-1
4	1	-1	-1	1	-1	1
5	-1	1	1	-1	-1	1
6	-1	1	-1	1	1	-1
7	-1	-1	1	1	1	1
8	-1	-1	-1	-1	-1	-1

Example 15.5. Randomized Complete Block Design

See FACTEX11 in the SAS/QC Sample Library

In a randomized complete block design (RCBD), each level of a "treatment" appears once in each block, and each block contains all the treatments. The order of treatments is randomized separately for each block. You can create RCBDs with the FACTEX procedure.

Suppose you want to construct an RCBD with six treatments in four blocks. To test each treatment once in each block, you need 24 experimental units. The following statements construct the randomized complete block design shown in Output 15.5.1:

```
proc factex;
   factors block / nlev=4;
   output out=blocks
          block nvals=(1 2 3 4);
run;
   factors trt / nlev=6;
   output out=rcbd
          designrep=blocks
          randomize (101)
          trt cvals=('A' 'B' 'C'
                     'D' 'E' 'F');
run;

proc print data=rcbd;
run;
```

Note that the order of the runs within each block is randomized and that the blocks (1, 2, 3, and 4) are in a random order.

Output 15.5.1. A Randomized Complete Block Design

Obs	blocks	trt
1	1	B
2	1	A
3	1	D
4	1	E
5	1	F
6	1	C
7	4	B
8	4	D
9	4	C
10	4	A
11	4	E
12	4	F
13	2	B
14	2	C
15	2	F
16	2	E
17	2	D
18	2	A
19	3	C
20	3	A
21	3	B
22	3	E
23	3	F
24	3	D

Example 15.6. Two-Level Design with Design Replication and Point Replication

See FACTEX12 in the SAS/QC Sample Library

You can replicate a design to obtain an independent estimate of experimental error or to estimate effects more precisely. There are two ways you can replicate a design using the FACTEX procedure: you can replicate the entire design with the DESIGNREP= option, or you can replicate each point in the design with the POINTREP= option. The following example illustrates the difference.

A process engineer is conducting an experiment to study the shrinkage of an injection-molded plastic component. The engineer chooses to determine the effect of the following four factors, each at two levels: holding pressure (PRESSURE), molding temperature (TEMP), cooling time (TIME), and injection velocity (VELOCITY).

The design used is a half-fraction of a 2^4 factorial design, denoted as 2^{4-1}_{IV}. The following statements construct the design:

```
proc factex;
   factors pressure temp time velocity;
   size fraction=2;
   model res=max;
   output out=savunrep;
run;

proc print data=savunrep;
run;
```

The design, saved in the data set SAVUNREP, is shown in Output 15.6.1.

Output 15.6.1. Unreplicated Design

Obs	pressure	temp	time	velocity
1	-1	-1	-1	-1
2	-1	-1	1	1
3	-1	1	-1	1
4	-1	1	1	-1
5	1	-1	-1	1
6	1	-1	1	-1
7	1	1	-1	-1
8	1	1	1	1

To obtain a more precise estimate of the experimental error, the engineer decides to replicate the entire design three times. The following statements generate a 2^{4-1}_{IV} design with three replicates in 24 runs:

```
proc factex;
   factors pressure temp time velocity;
   size fraction=2;
   model res=max;
   output out=savedrep designrep=3;
run;

proc print data=savedrep;
run;
```

The design, saved in the data set SAVEDREP, is displayed in Output 15.6.2.

Output 15.6.2. Design Replication

Obs	pressure	temp	time	velocity
1	-1	-1	-1	-1
2	-1	-1	1	1
3	-1	1	-1	1
4	-1	1	1	-1
5	1	-1	-1	1
6	1	-1	1	-1
7	1	1	-1	-1
8	1	1	1	1
9	-1	-1	-1	-1
10	-1	-1	1	1
11	-1	1	-1	1
12	-1	1	1	-1
13	1	-1	-1	1
14	1	-1	1	-1
15	1	1	-1	-1
16	1	1	1	1
17	-1	-1	-1	-1
18	-1	-1	1	1
19	-1	1	-1	1
20	-1	1	1	-1
21	1	-1	-1	1
22	1	-1	1	-1
23	1	1	-1	-1
24	1	1	1	1

The first replicate comprises observations 1 to 8, the second replicate comprises observations 9 to 16, and the third replicate comprises observations 17 to 24.

Now, instead of replicating the entire design, suppose the engineer decides to replicate each run in the design three times. The following statements construct a 2^{4-1}_{IV} design in 24 runs with point replication:

```
proc factex;
   factors pressure temp time velocity;
   size fraction=2;
   model res=max;
   output out=saveprep pointrep=3;
run;

proc print data=saveprep;
run;
```

The design, saved in the data set SAVEPREP, is displayed in Output 15.6.3. The first design point is replicated three times (observations 1–3), the second design point is replicated three times (observations 4–6), and so on.

Output 15.6.3. Point Replication

Obs	pressure	temp	time	velocity
1	-1	-1	-1	-1
2	-1	-1	-1	-1
3	-1	-1	-1	-1
4	-1	-1	1	1
5	-1	-1	1	1
6	-1	-1	1	1
7	-1	1	-1	1
8	-1	1	-1	1
9	-1	1	-1	1
10	-1	1	1	-1
11	-1	1	1	-1
12	-1	1	1	-1
13	1	-1	-1	1
14	1	-1	-1	1
15	1	-1	-1	1
16	1	-1	1	-1
17	1	-1	1	-1
18	1	-1	1	-1
19	1	1	-1	-1
20	1	1	-1	-1
21	1	1	-1	-1
22	1	1	1	1
23	1	1	1	1
24	1	1	1	1

Note the difference in the arrangement of the designs created using design replication (Output 15.6.2) and point replication (Output 15.6.3). In design replication, the original design is replicated a specified number of times; but in point replication, each run in the original design is replicated a specified number of times. See page 494 for more information on design replication.

Example 15.7. Mixed-Level Design Using Design Replication and Point Replication

See FACTEX13 in the SAS/QC Sample Library

Orthogonal factorial designs are most commonly used at the initial stages of experimentation. At these stages, it is best to experiment with as few levels of each factor as possible in order to minimize the number of runs required. Thus, these designs usually involve only two levels of each factor. Occasionally some factors will naturally have more than two levels of interest—different types of seed, for instance.

You can create designs for factors with different numbers of levels simply by taking the cross product of component designs in which the factors all have the same numbers of levels, that is, replicating every run of one design for each run of the other. (See Example 15.14 on page 482.) All estimable effects in each of the component designs, as well as all generalized interactions between estimable effects in different component designs, are estimable in the cross-product; refer to Section 3 of Chakravarti (1956).

This example illustrates how you can construct a mixed level design using the OUTPUT statement with the POINTREP= option or the DESIGNREP= option to take the cross product between two designs.

Suppose you want to construct a mixed-level factorial design for two two-level factors (A and B) and one three-level factor (C) with 12 runs. The following SAS statements produce a complete 3×2^2 factorial design using design replication:

```
proc factex;
   factors a b;
   output out=ab;
run;
   factors c / nlev=3;
   output out=drepdesn
          designrep=ab;
run;

proc print data=drepdesn;
run;
```

Output 15.7.1 lists the mixed-level design saved in the data set DREPDESN.

Output 15.7.1. 3×2^2 Mixed-Level Design Using Design Replication

Obs	a	b	c
1	-1	-1	-1
2	-1	-1	0
3	-1	-1	1
4	-1	1	-1
5	-1	1	0
6	-1	1	1
7	1	-1	-1
8	1	-1	0
9	1	-1	1
10	1	1	-1
11	1	1	0
12	1	1	1

You can also create a mixed-level design for the preceding factors using the point replication feature of the FACTEX procedure. The following SAS statements produce a complete $2^2 \times 3$ factorial design using point replication:

```
proc factex;
   factors a b;
   output out=ab;
run;
   factors c / nlev=3;
   output out=prepdesn
          pointrep=ab;
run;

proc print data=prepdesn;
run;
```

Output 15.7.2 lists the mixed-level design saved in the data set PREPDESN.

Output 15.7.2. $2^2 \times 3$ Mixed-Level Design Using Point Replication

Obs	c	a	b
1	-1	-1	-1
2	-1	-1	1
3	-1	1	-1
4	-1	1	1
5	0	-1	-1
6	0	-1	1
7	0	1	-1
8	0	1	1
9	1	-1	-1
10	1	-1	1
11	1	1	-1
12	1	1	1

Note the difference between the designs in Output 15.7.1 and Output 15.7.2. In design replication, the mixed-level design is given by AB $\otimes$ C, while for point replication the mixed-level design is given by C $\otimes$ AB, where $\otimes$ denotes the direct product. In design replication, you can view the DESIGNREP= data set as nested *outside* the design, while in point replication, you can view the POINTREP= data set as nested *inside* the design.

Example 15.8. Mixed-Level Design Using Pseudo-Factors

See FACTEX6A in the SAS/QC Sample Library

If the numbers of levels for the factors of the mixed-level design are all powers of the same prime power q, you can construct the design using *pseudo-factors*, where the levels of k q-level pseudo-factors are associated with the levels of a single *derived factor* with q^k levels. Refer to Section 5 of Chakravarti (1956) and see "Types of Factors" on page 488 for details.

For example, the following statements create a design for one four-level factor (A) and three two-level factors (B, C, and D) in 16 runs (a half replicate):

```
proc factex;
   factors a1 a2 b c d;
   model estimate     =(b c d   a1|a2                  )
         nonnegligible=(b|c|d@2 a1|a2|b a1|a2|c a1|a2|d);
   size design=16;
   output out=designa [a1 a2]=a cvals = ('A' 'B' 'C' 'D');
proc print;
  var a b c d;
run;
```

The levels of two two-level pseudo-factors (A1 and A2) are used to represent the four levels of A. Hence the three degrees of freedom associated with A will be given by the main effects of A1 and A2 and their interaction A1*A2, and you can thus refer to (A1|A2) as the main effect of A.

The MODEL statement specifies that the main effects of all factors are to be estimable, and that all of the two-factor interactions between B, C, and D, as well as the interactions between each of these and (A1|A2), are to be nonnegligible. As a result, the mixed-level design has resolution 4. The design is saved in the data set DESIGNA, combining the levels of the two pseudo-factors, A1 and A2, to obtain the levels of the four-level factor A. The data set DESIGNA is listed in Output 15.8.1.

Output 15.8.1. 4×2^3 Design of Resolution IV in 16 Runs

Obs	a	b	c	d
1	A	-1	-1	1
2	A	-1	1	-1
3	A	1	-1	-1
4	A	1	1	1
5	C	-1	-1	-1
6	C	-1	1	1
7	C	1	-1	1
8	C	1	1	-1
9	B	-1	-1	-1
10	B	-1	1	1
11	B	1	-1	1
12	B	1	1	-1
13	D	-1	-1	1
14	D	-1	1	-1
15	D	1	-1	-1
16	D	1	1	1

Example 15.9. Mixed-Level Design by Collapsing Factors

See FACTEX6C in the SAS/QC Sample Library

You can construct a mixed-level design by *collapsing* factors, that is, by replacing a factor with n levels by a factor with m levels, where $m < n$. Orthogonality is retained in the sense that estimates of different effects are uncorrelated, although not all estimates have equal variance; refer to Section 6 of Chakravarti (1956). This method has been used by Addelman (1962) to derive main effects plans for factors with mixed numbers of levels and by Margolin (1967) to construct plans that consider two-factor interactions.

You can use the value specification in the OUTPUT statement as a convenient tool for collapsing factors. For example, the following statements create a 27-run design for two two-level factors (X1 and X2) and two three-level factors (X3 and X4) such that all main effects and two-factor interactions are uncorrelated:

```
proc factex;
   factors x1-x4 / nlev = 3;
   size design=27;
   model r=4;
   output out=savmixed x1 nvals=(-1 1 -1)
                       x2 nvals=(-1 1 -1);
proc print data=savmixed;
run;
```

The mixed-level design is a three-quarter fraction with resolution 5; refer to Margolin (1967). The design is displayed in Output 15.9.1.

Output 15.9.1. $2^2 \times 3^2$ Design of Resolution V in 27 Runs

Obs	x1	x2	x3	x4
1	-1	-1	-1	-1
2	-1	-1	0	1
3	-1	-1	1	0
4	-1	1	-1	1
5	-1	1	0	0
6	-1	1	1	-1
7	-1	-1	-1	0
8	-1	-1	0	-1
9	-1	-1	1	1
10	1	-1	-1	1
11	1	-1	0	0
12	1	-1	1	-1
13	1	1	-1	0
14	1	1	0	-1
15	1	1	1	1
16	1	-1	-1	-1
17	1	-1	0	1
18	1	-1	1	0
19	-1	-1	-1	0
20	-1	-1	0	-1
21	-1	-1	1	1
22	-1	1	-1	-1
23	-1	1	0	1
24	-1	1	1	0
25	-1	-1	-1	1
26	-1	-1	0	0
27	-1	-1	1	-1

Example 15.10. Hyper-Graeco-Latin Square Design

See FACTEX7A in the SAS/QC Sample Library

A $q \times q$ Latin square is an arrangement of q symbols, each repeated q times, in a square of side q such that each symbol appears exactly once in each row and in each column. Such arrangements are useful as designs for *row-and-column* experiments, where it is necessary to balance the effects of two q-level factors simultaneously.

A Graeco-Latin square is actually a pair of Latin squares; when superimposed, each symbol in one square occurs exactly once with each symbol in the other square. The following is an example of a 5×5 Graeco-Latin square, where Latin letters are used for the symbols of one square and Greek letters are used for the symbols of the other:

$$\begin{array}{ccccc} A\alpha & B\beta & C\gamma & D\delta & E\epsilon \\ B\gamma & C\delta & D\epsilon & E\alpha & A\beta \\ C\epsilon & D\alpha & E\beta & A\gamma & B\delta \\ D\beta & E\gamma & A\delta & B\epsilon & C\alpha \\ E\delta & A\epsilon & B\alpha & C\beta & D\gamma \end{array}$$

Whenever q is a power of a prime number, you can construct up to $q - 1$ squares, each with q symbols that are balanced over all the other factors. The result is called a *hyper-Graeco-Latin Square* or a complete set of *mutually orthogonal* Latin squares. Such arrangements can be useful as designs (refer to Williams 1949), or they can be used to construct other designs.

When q is a prime power, hyper-Graeco-Latin squares are straightforward to construct with the FACTEX procedure. This is because *a complete set of $q - 1$ mutually orthogonal $q \times q$ Latin squares is equivalent to a resolution 3 design for $q + 1$ q-level factors in q^2 runs, where two of the factors index rows and columns and each of the remaining factors indexes the treatments of one of the squares.*

For instance, the following statements generate a complete set of three mutually orthogonal 4×4 Latin squares, with rows indexed by the factor ROW, columns indexed by the factor COLUMN, and the treatment factors in the respective squares indexed by T1, T2, and T3. The first step is to construct a resolution 3 design for five four-level factors in 16 runs.

```
proc factex;
   factors row column t1-t3 / nlev=4;
   size design=16;
   model resolution=3;
   output out=graeco t1 cvals=('A' 'B' 'C' 'D')
                     t2 cvals=('A' 'B' 'C' 'D')
                     t3 cvals=('A' 'B' 'C' 'D');
run;
```

In most cases, the form that appears in the output data set GRAECO is most useful. The form that usually appears in textbooks is displayed in Output 15.10.1, which can be produced using a simple DATA step (not shown here).

Output 15.10.1. Hyper-Graeco-Latin Square

```
Square 1 :
      A D B C
      D A C B
      B C A D
      C B D A

Square 2 :
      A D B C
      C B D A
      D A C B
      B C A D

Square 3 :
      A D B C
      B C A D
      C B D A
      D A C B
```

Example 15.11. Resolution IV Design with Minimum Aberration

See FACTEX14 in the SAS/QC Sample Library

If a design has resolution IV, then you can simultaneously estimate all main effects and *some* two-factor interactions. However, not all resolution IV designs are equivalent; you may be able to estimate more two-factor interactions with some than with others. Among all resolution IV designs, a design that allows you to estimate the maximum number of two-factor interactions is said to have *minimum aberration*.

For example, if you use the FACTEX procedure to generate a resolution IV two-level design in 32 runs for seven factors, you will be able to estimate all main effects and 15 of the 21 two-factor interactions with the design that is created by default. The following statements create this design and display its alias structure in Output 15.11.1:

```
proc factex;
   factors a b c d e f g;
   model resolution=4;
   size design=32;
   examine aliasing;
run;
```

Output 15.11.1. Alias Structure for Default 2^{7-2}_{IV} Design

```
Aliasing Structure

a
b
c
d
e
f
g
a*b = f*g
a*c
a*d
a*e
a*f = b*g
a*g = b*f
b*c
b*d
b*e
c*d = e*g
c*e = d*g
c*f
c*g = d*e
d*f
e*f
```

In constrast, the resolution 4 design given in Table 12.15 of Box, Hunter, and Hunter (1978) is a minimum aberration design that allows estimation of 18 two-factor interactions, three more than can be estimated with the default design. The FACTEX procedure constructs the minimum aberration design if you specify the MINABS option to the MODEL statement, as in the following statements:

```
proc factex;
   factors a b c d e f g;
   model resolution=4 / minab
   size design=32;
   examine aliasing;
run;
```

The alias structure for the resulting design is shown in Output 15.11.2.

Output 15.11.2. Alias Structure for Minimum Aberration 2^{7-2}_{IV} Design

```
Aliasing Structure

a
b
c
d
e
f
g
a*b
a*c
a*d
a*e
a*f
a*g
b*c
b*d
b*e
b*f
b*g
c*d = e*f
c*e = d*f
c*f = d*e
c*g
d*g
e*g
f*g
```

All of the designs listed in Table 12.15 of Box, Hunter, and Hunter (1978) have minimum aberration. For most of these cases, the default design constructed by the FACTEX procedure has minimum aberration—that is, the MINABS option is not required. This is important because the MINABS option forces the FACTEX procedure to check many more designs, and the search can, therefore, take longer to run. You can limit the search time with the TIME= option in the PROC FACTEX statement. In five of the cases (2^{10-6}_{III}, 2^{7-2}_{IV}, 2^{8-3}_{IV}, 2^{9-4}_{IV}, and 2^{10-3}_{V}), the MINABS option is required to construct a design with minimum aberration, and in two cases (2^{9-5}_{III}, 2^{9-3}_{IV}), the NOCHECK option is required as well. If the FACTEX procedure is given a sufficiently large amount of time to run, specifying both the MINABS and the NOCHECK options will always result in a minimum aberration design. However, with the default search time of 60 seconds, there are three cases (2^{10-5}_{IV}, 2^{10-4}_{IV}, and 2^{11-5}_{IV}) for which the FACTEX procedure is unable to find the minimum aberration design, even with both the MINABS and NOCHECK options specified.

Example 15.12. Replicated Blocked Design with Partial Confounding

See FACTEX15 in the SAS/QC Sample Library

In an unreplicated blocked design, the interaction effect that is confounded with the block effect cannot be estimated. You can replicate the experiment so that a different interaction effect is confounded in each replicate. This enables you to obtain information about an interaction effect from the replicate(s) in which it is not confounded.

For example, consider a 2^3 design with factors A, B, and C arranged in two blocks. Suppose you decide to run four replicates of the design. By constructing the design sequentially, you can choose the effects to be estimated in each replicate depending on the interaction confounded with the block effect in the other replicates.

In the first replicate, you specify only that the main effects are to be estimable. The following statements generate an eight-run two-level design arranged in two blocks:

```
proc factex;
   factors a b c;
   blocks nblocks=2;
   model est=(a b c);
   examine confounding aliasing;
   output out=rep1 blockname=block nvals=(1 2);
run;
```

The alias structure and the confounding scheme are listed in Output 15.12.1. The highest order interaction A*B*C is confounded with the block effect. The design, with recoded block levels, is saved in a dataset named REP1.

Output 15.12.1. Confounding Rule and Alias Structure for Replicate 1

```
                Block Pseudo-factor Confounding Rules

                          [B1] = a*b*c

Aliasing Structure

      a
      b
      c
      a*b
      a*c
      b*c
```

If you were to analyze this replicate by itself, you could not determine whether an effect is due to A*B*C or the block effect. You can construct a second replicate that confounds a different interaction effect with the block effect. Since the FACTEX procedure is interactive, simply submit the following statements to generate the second replicate:

```
   model est=(a b c a*b*c);
   output out=rep2
      blockname=block nvals=(3 4);
run;
```

The alias structure and the confounding scheme for the second replicate are listed in Output 15.12.2. The interaction A*B*C is free of any aliases, but now the two-factor interaction B*C is confounded with the block effect.

Output 15.12.2. Confounding Rule and Alias Structure for Replicate 2

```
                    Block Pseudo-factor Confounding Rules

                              [B1] = b*c

Aliasing Structure

       a
       b
       c
       a*b
       a*c
 [B] = b*c
       a*b*c
```

To estimate the interaction B*C with the third replicate, submit the following statements (immediately after the preceding statements):

```
   model est=(a b c a*b*c b*c);
   output out=rep3 blockname=block nvals=(5 6);
run;
```

The alias structure and confounding rules are shown in Output 15.12.3. The interaction B*C is free of aliases, but the interaction A*C is confounded with the block effect.

Output 15.12.3. Confounding Rule and Alias Structure for Replicate 3

```
                    Block Pseudo-factor Confounding Rules

                              [B1] = a*c

Aliasing Structure

       a
       b
       c
       a*b
 [B] = a*c
       b*c
       a*b*c
```

Finally, to estimate the interaction effect A*C with the fourth replicate, submit the following statements:

```
   model est=(a b c a*b*c b*c a*c);
   output out=rep4 blockname=block nvals=(7 8);
run;
```

The alias structure and confounding rules are displayed in Output 15.12.4.

Output 15.12.4. Confounding Rule and Alias Structure for Replicate 4

```
                    Block Pseudo-factor Confounding Rules

                              [B1] = a*b

Aliasing Structure

       a
       b
       c
 [B] = a*b
       a*c
       b*c
       a*b*c
```

When combined, these four replicates give full information on the main effects and three-quarter information on each of the interactions. The following statements combine the four replicates:

```
data combine;
  set rep1 rep2 rep3 rep4;
run;

proc print data=combine;
run;
```

The final design is saved in the data set COMBINE. A partial listing of this data set is shown in Output 15.12.5.

Output 15.12.5. Combined Design

Obs	block	a	b	c
1	1	-1	-1	-1
2	1	-1	1	1
3	1	1	-1	1
4	1	1	1	-1
5	2	-1	-1	1
6	2	-1	1	-1
7	2	1	-1	-1
8	2	1	1	1
9	3	-1	-1	1
10	3	-1	1	-1
11	3	1	-1	1
12	3	1	1	-1
13	4	-1	-1	-1
14	4	-1	1	1
15	4	1	-1	-1
16	4	1	1	1
17	5	-1	-1	1
18	5	-1	1	1
19	5	1	-1	-1
20	5	1	1	-1
21	6	-1	-1	-1
22	6	-1	1	-1
23	6	1	-1	1
24	6	1	1	1
25	7	-1	1	-1
26	7	-1	1	1
27	7	1	-1	-1
28	7	1	-1	1
29	8	-1	-1	-1
30	8	-1	-1	1
31	8	1	1	-1
32	8	1	1	1

Example 15.13. Incomplete Block Design

See FACTEX7B in the SAS/QC Sample Library

Several important series of balanced incomplete block designs can be derived from orthogonal factorial designs. One is the series on *balanced lattice* of Yates (1936); refer to page 396 of Cochran and Cox (1957). In this situation, the number of treatments v must be the square of a power of a prime number: $v = q^2, \quad q = p^k$ where p is a prime number. These designs are based on a complete set of $q - 1$ mutually orthogonal $q \times q$ Latin squares, which is equivalent to a resolution 3 design for $q + 1$ q-level factors in q^2 runs.

The balanced lattice designs include $q + 1$ replicates of the treatments. They are constructed by associating each treatment with a run in the factorial design, each replicate with one of the factors, and each block with one of the q values of that factor. For example, the treatments in Block 3 within Replicate 2 are those treatments that are associated with runs for which factor 2 is set at value 3.

The following statements use this method to construct a balanced lattice design for 16 treatments in five replicates of four blocks each. The construction procedure is based on a resolution 3 design for five four-level factors in 16 runs.

```
proc factex;
   factors x1-x5 / nlev=4;
   size design=16;
   model r=3;
   output out=a;
run;
```

In the following DATA step, the incomplete block design is built using the design saved in the data set A by the FACTEX procedure:

```
data b;
   keep rep block plot t;
   array x{5} x1-x5;
   do rep = 1 to 5;
      do block = 1 to 4;
         plot = 0;
         do n = 1 to 16;
            set a point=n;
            if (x{rep}=block-1) then do;
               t = n;
               plot = plot + 1;
               output;
               end;
            end;
         end;
      end;
   stop;
run;
```

For each block within each replicate, the program loops through the run numbers in the factorial design and chooses those which have the REPth factor equal to BLOCK-1. These run numbers are the treatments that go into the particular block.

The design is printed using a DATA step. Each block of each replicate is built into the variables S1, S2, S3, and S4, and each block is printed with a PUT statement.

```
data _null_;
   array s{4} s1-s4;            /* Buffer for holding each block    */
   file print;                  /* Direct printing to output screen */
   n = 1;
   do r = 1 to 5;
      put "Replication " r 1.0 ":";
      do b = 1 to 4;
         do p = 1 to 4;
            set b point=n;
            s{plot} = t;
            n = n+1;
            end;
         put "    Block " b 1.0 ":" (s1-s4) (3.0);
         end;
      put;
      end;
   stop;
run;
```

The design is displayed in Output 15.13.1.

You can use the PLAN procedure to randomize the block design, as shown by the following statements:

```
proc plan seed=54321;
   factors rep=5 block=4 plot=4;
   output data=b out=c;

proc sort;
   by rep block plot;
run;
```

The variable PLOT indexes the plots within each block. Refer to the *SAS/STAT User's Guide* for a general discussion of randomizing block designs.

Finally, substitute **`set c`** for **`set b`** in the preceding DATA step. Running this DATA step creates the randomized design displayed in Output 15.13.2.

Output 15.13.1. A Balanced Lattice

```
Replication 1:
     Block 1:  1  2  3  4
     Block 2:  5  6  7  8
     Block 3:  9 10 11 12
     Block 4: 13 14 15 16

Replication 2:
     Block 1:  1  5  9 13
     Block 2:  2  6 10 14
     Block 3:  3  7 11 15
     Block 4:  4  8 12 16

Replication 3:
     Block 1:  1  6 11 16
     Block 2:  3  8  9 14
     Block 3:  4  7 10 13
     Block 4:  2  5 12 15

Replication 4:
     Block 1:  1  8 10 15
     Block 2:  3  6 12 13
     Block 3:  4  5 11 14
     Block 4:  2  7  9 16

Replication 5:
     Block 1:  1  7 12 14
     Block 2:  3  5 10 16
     Block 3:  4  6  9 15
     Block 4:  2  8 11 13
```

Output 15.13.2. Randomized Design

```
Replication 1:
     Block 1: 15  5  2 12
     Block 2:  3  8  9 14
     Block 3: 16  1 11  6
     Block 4:  7 10 13  4

Replication 2:
     Block 1:  2  4  3  1
     Block 2:  5  7  8  6
     Block 3:  9 11 10 12
     Block 4: 15 16 13 14

Replication 3:
     Block 1:  2 13  8 11
     Block 2: 14 12  7  1
     Block 3: 15  4  9  6
     Block 4:  5 16  3 10

Replication 4:
     Block 1: 13  1  5  9
     Block 2: 14  2 10  6
     Block 3: 11 15  3  7
     Block 4: 16 12  4  8

Replication 5:
     Block 1:  2 16  7  9
     Block 2: 15 10  8  1
     Block 3:  3 12  6 13
     Block 4:  5 11 14  4
```

Example 15.14. Design with Inner Array and Outer Array

See FACTEX4 in the SAS/QC Sample Library

Byrne and Taguchi (1986) report the use of a fractional factorial design to investigate fitting an elastomeric connector to a nylon tube as tightly as possible. Their experiment applies the design philosophy of Genichi Taguchi, which distinguishes between *control factors* and *noise factors*. Control factors are typically those that the engineer is able to set under real conditions, while noise factors vary uncontrollably in practice (though within a predictable range).

The experimental layout consists of two designs, one for the control factors and one for the noise factors. The design for the control factors is called the *inner array*, and the design for noise factors is called the *outer array*. The outer array is replicated for each of the runs in the inner array, and a performance measure ("signal-to-noise ratio") is computed over the replicate. The performance measure thus reflects variation due to changes in the noise factors. You can construct such a cross-product design with the replication options in the OUTPUT statement of the FACTEX procedure, as shown in this example.

Researchers identified the following four control factors that were thought to influence the amount of force required to pull the connector off the tube:

- the interference (INTERFER), defined as the difference between the outer width of the tubing and the inner width of the connector
- the connector wall thickness (CONNWALL)
- the depth of insertion (IDEPTH) of the tubing into the connector
- the amount of adhesive (GLUE) in the connector pre-dip

Researchers also identified the following three noise factors related to the assembly:

- the amount of time (TIME) allowed for assembly
- the temperature (TEMPERAT)
- the relative humidity (HUMIDITY)

Three levels were selected for each of the control factors, and two levels were selected for each of the noise factors.

The following statements construct the 72-run design used by Byrne and Taguchi (1986). First, an eight-run outer array for the three noise factors is created and saved in the data set OUTERARY.

```
proc factex;
   factors time temperat humidity;
   output out=outerary time     nvals=( 24 120)
                        temperat nvals=( 72 150)
                        humidity nvals=(.25 .75);
run;
```

Next, a nine-run inner array (design of resolution 3) is chosen for the control factors. The POINTREP option in the OUTPUT statement replicates the eight-run outer array in the data set OUTERARY for each of the nine runs in the inner array and saves the final design containing 72 runs in the data set SAVEDESN.

```
proc factex;
   factors interfer connwall idepth glue / nlev=3;
   size design=9;
   model resolution=3;
   output out=savedesn pointrep=outerary
        interfer cvals=('Low'     'Medium' 'High'  )
        connwall cvals=('Thin'    'Medium' 'Thick' )
        idepth   cvals=('Shallow' 'Deep'   'Medium')
        glue     cvals=('Low'     'High'   'Medium');
run;

proc print data=savedesn;
run;
```

The final design is listed in Output 15.14.1. Main effects of each factor can be estimated free of each other but are confounded with two-factor interactions.

Output 15.14.1. Design for Control Factor and Noise Factors

Obs	interfer	connwall	idepth	glue	time	temperat	humidity
1	Low	Thin	Shallow	Low	24	72	0.25
2	Low	Thin	Shallow	Low	24	72	0.75
3	Low	Thin	Shallow	Low	24	150	0.25
4	Low	Thin	Shallow	Low	24	150	0.75
5	Low	Thin	Shallow	Low	120	72	0.25
6	Low	Thin	Shallow	Low	120	72	0.75
7	Low	Thin	Shallow	Low	120	150	0.25
8	Low	Thin	Shallow	Low	120	150	0.75
9	Low	Medium	Medium	Medium	24	72	0.25
10	Low	Medium	Medium	Medium	24	72	0.75
11	Low	Medium	Medium	Medium	24	150	0.25
12	Low	Medium	Medium	Medium	24	150	0.75
13	Low	Medium	Medium	Medium	120	72	0.25
14	Low	Medium	Medium	Medium	120	72	0.75
15	Low	Medium	Medium	Medium	120	150	0.25
16	Low	Medium	Medium	Medium	120	150	0.75
17	Low	Thick	Deep	High	24	72	0.25
18	Low	Thick	Deep	High	24	72	0.75
19	Low	Thick	Deep	High	24	150	0.25
20	Low	Thick	Deep	High	24	150	0.75
21	Low	Thick	Deep	High	120	72	0.25
22	Low	Thick	Deep	High	120	72	0.75
23	Low	Thick	Deep	High	120	150	0.25
24	Low	Thick	Deep	High	120	150	0.75
25	Medium	Thin	Medium	High	24	72	0.25
26	Medium	Thin	Medium	High	24	72	0.75
27	Medium	Thin	Medium	High	24	150	0.25
28	Medium	Thin	Medium	High	24	150	0.75
29	Medium	Thin	Medium	High	120	72	0.25
30	Medium	Thin	Medium	High	120	72	0.75

Output 15.14.1. (continued)

Obs	interfer	connwall	idepth	glue	time	temperat	humidity
31	Medium	Thin	Medium	High	120	150	0.25
32	Medium	Thin	Medium	High	120	150	0.75
33	Medium	Medium	Deep	Low	24	72	0.25
34	Medium	Medium	Deep	Low	24	72	0.75
35	Medium	Medium	Deep	Low	24	150	0.25
36	Medium	Medium	Deep	Low	24	150	0.75
37	Medium	Medium	Deep	Low	120	72	0.25
38	Medium	Medium	Deep	Low	120	72	0.75
39	Medium	Medium	Deep	Low	120	150	0.25
40	Medium	Medium	Deep	Low	120	150	0.75
41	Medium	Thick	Shallow	Medium	24	72	0.25
42	Medium	Thick	Shallow	Medium	24	72	0.75
43	Medium	Thick	Shallow	Medium	24	150	0.25
44	Medium	Thick	Shallow	Medium	24	150	0.75
45	Medium	Thick	Shallow	Medium	120	72	0.25
46	Medium	Thick	Shallow	Medium	120	72	0.75
47	Medium	Thick	Shallow	Medium	120	150	0.25
48	Medium	Thick	Shallow	Medium	120	150	0.75
49	High	Thin	Deep	Medium	24	72	0.25
50	High	Thin	Deep	Medium	24	72	0.75
51	High	Thin	Deep	Medium	24	150	0.25
52	High	Thin	Deep	Medium	24	150	0.75
53	High	Thin	Deep	Medium	120	72	0.25
54	High	Thin	Deep	Medium	120	72	0.75
55	High	Thin	Deep	Medium	120	150	0.25
56	High	Thin	Deep	Medium	120	150	0.75
57	High	Medium	Shallow	High	24	72	0.25
58	High	Medium	Shallow	High	24	72	0.75
59	High	Medium	Shallow	High	24	150	0.25
60	High	Medium	Shallow	High	24	150	0.75
61	High	Medium	Shallow	High	120	72	0.25
62	High	Medium	Shallow	High	120	72	0.75
63	High	Medium	Shallow	High	120	150	0.25
64	High	Medium	Shallow	High	120	150	0.75
65	High	Thick	Medium	Low	24	72	0.25
66	High	Thick	Medium	Low	24	72	0.75
67	High	Thick	Medium	Low	24	150	0.25
68	High	Thick	Medium	Low	24	150	0.75
69	High	Thick	Medium	Low	120	72	0.25
70	High	Thick	Medium	Low	120	72	0.75
71	High	Thick	Medium	Low	120	150	0.25
72	High	Thick	Medium	Low	120	150	0.75

Note that the levels of IDEPTH and GLUE are listed in the OUTPUT statement in a nonstandard order so that the design produced by the FACTEX procedure matches the design of Byrne and Taguchi (1986). The order of assignment of levels does not affect the properties of the resulting design. Furthermore, design can be randomized with the RANDOMIZE option in the OUTPUT statement.

Byrne and Taguchi (1986) indicate that a smaller outer array with only four runs would have been sufficient. You can generate this design (not shown here) by modifying the statements on page 482; specifically, add the following SIZE and MODEL statements:

```
size design=4;
model resolution=3;
```

In their analysis of the data from the experiment based on the smaller design, Byrne and Taguchi (1986) note several interesting interactions between control and noise factors. However, since the inner array is of resolution 3, it is impossible to say whether or not there exist interesting interactions between the control factors. In other words, you cannot determine whether an effect is due to an interaction or to the main effect with which it is confounded.

One alternative is to begin with a design of resolution 4. Two-factor interactions will remain confounded with one another, but they will be free of main effects. Moreover, further experimentation can be carried out to distinguish between confounded interactions that seem important. To determine the optimal size of this design, submit the following statements interactively:

```
proc factex;
   factors interfer connwall idepth glue / nlev=3;
   model resolution=4;
   size design=minimum;
run;
```

This causes the following message to appear in the SAS log:

```
NOTE: Design has 27 runs, resolution = 4.
```

In other words, the smallest resolution 4 design for four three-level factors has 27 runs, which together with the eight-run outer array requires 216 runs. Even the smaller four-run outer array requires 108 runs. Both of these designs are substantially larger than the design originally reported, but the larger designs protect against the effects of unsuspected interactions.

A second alternative is to begin with only two levels of the control factors. Further experimentation can then be directed toward exploring the effects of factors determined to be important in this initial stage of experimentation. Note that NLEV=2 is the default in the FACTORS statement. Submit the following additional statements:

```
   factors interfer connwall idepth glue;
   model resolution=4;
   size design=minimum;
run;
```

This causes the following message to appear in the SAS log:

```
NOTE: Design has 8 runs, resolution = 4.
```

Thus, as few as eight runs can be used for the inner array. This design is amenable to blocking, whereas the proposed nine-run design is not. Blocking is an important consideration whenever experimental conditions can vary over the course of conducting the experiment.

Now, submit the following statements:

```
   size design=8;
   blocks size=minimum;
run;
```

This causes the following message to appear in the SAS log:

```
NOTE: Design has 8 runs in 4 blocks of size 2,
      resolution = 4.
```

Thus the experiment can be run in blocks as small as two runs.

Example 15.15. Design and Analysis of a Complete Factorial Experiment

See FACTEX16 in the SAS/QC Sample Library

Yin and Jillie (1987) describe an experiment on a nitride etch process for a single wafer plasma etcher. The experiment was run using four factors: cathode power (POWER), gas flow (FLOW), reactor chamber pressure (PRESSURE), and electrode gap (GAP). A single replicate of a 2^4 design was run, and the etch rate (RATE) was measured.

You can use the following statements to construct a 16-run design in the four factors:

```
proc factex;
   factors power flow pressure gap;
   output out=desgndat
          power    nvals=(0.80 1.20)
          flow     nvals=(4.50 550 )
          pressure nvals=(125  200 )
          gap      nvals=(275  325 );
run;
```

The design with the actual (decoded) factor levels is saved in the data set DESGNDAT. The experiment using the 16-run design is performed, and the etch rate is measured. The following DATA step updates the data set DESGNDAT with the values of RATE:

```
data desgndat;
   set desgndat;
   input rate @@;
   datalines;
   550   669  604   650  633   642  601   635
   1037  749  1052  868  1075  860  1063  729
   ;
```

The data set DESGNDAT is listed in Output 15.15.1.

Output 15.15.1. A 2^4 Design with Responses

Nitride Etch Process Experiment

Obs	power	flow	pressure	gap	rate
1	0.8	4.5	125	275	550
2	0.8	4.5	125	325	669
3	0.8	4.5	200	275	604
4	0.8	4.5	200	325	650
5	0.8	550.0	125	275	633
6	0.8	550.0	125	325	642
7	0.8	550.0	200	275	601
8	0.8	550.0	200	325	635
9	1.2	4.5	125	275	1037
10	1.2	4.5	125	325	749
11	1.2	4.5	200	275	1052
12	1.2	4.5	200	325	868
13	1.2	550.0	125	275	1075
14	1.2	550.0	125	325	860
15	1.2	550.0	200	275	1063
16	1.2	550.0	200	325	729

To perform an analysis of variance on the responses, you can use the GLM procedure, as follows:

```
proc glm data=desgndat;
   class power flow pressure gap;
   model rate=power|flow|pressure|gap@2 / ss1;
run;
```

The factors are listed in both the CLASS and MODEL statements, and the response as a function of the factors is modeled using the MODEL statement. The MODEL statement requests Type I sum of squares (SS1) and lists all effects that contain two or fewer factors. It is assumed that three-factor and higher interactions are not significant.

Part of the output from the GLM procedure is shown in Output 15.15.2. The main effect of the factors POWER and GAP and the interaction between POWER and GAP are significant (their p-values are less than 0.01).

Output 15.15.2. Analysis of Variance for the Nitride Etch Process Experiment

General Linear Models Procedure

Source	DF	Type I SS	Mean Square	F Value	Pr > F
power	1	374850.0625	374850.0625	183.99	<.0001
flow	1	217.5625	217.5625	0.11	0.7571
power*flow	1	18.0625	18.0625	0.01	0.9286
pressure	1	10.5625	10.5625	0.01	0.9454
power*pressure	1	1.5625	1.5625	0.00	0.9790
flow*pressure	1	7700.0625	7700.0625	3.78	0.1095
gap	1	41310.5625	41310.5625	20.28	0.0064
power*gap	1	94402.5625	94402.5625	46.34	0.0010
flow*gap	1	2475.0625	2475.0625	1.21	0.3206
pressure*gap	1	248.0625	248.0625	0.12	0.7414

Computational Details

Types of Factors

The *factors* of a design are variables that an experimenter can set at several values. In general, experiments are performed to study the effects of different levels of the factors on the *response* of interest. For example, consider an experiment to maximize the percentage of raw material that responds to a chemical reaction. The factors might include the reaction temperature and the feed rate of the chemicals, while the response is the yield rate. Factors of different types are used in different ways in constructing a design. This section defines the different types of factors.

Block factors are unavoidable factors that are known to affect the response, but in a relatively uninteresting way. For example, in the chemical experiment, the technician operating the equipment might have a noticeable effect on the yield of the process. The operator effect might be unavoidable, but it is usually not very interesting. On the other hand, factors whose effects are directly of interest are called *design factors*. One goal in designing an experiment is to avoid getting the effects of the design factors mixed up, or *confounded*, with the effects of any block factors.

When constructing a design by orthogonal confounding, all factors formally have the same number of levels q, where q is a prime number or a power of a prime number. Usually, q is two, and the factor levels are chosen to represent high and low values.

However, this does not mean, for example, that a design for two-level factors is restricted to no more than two blocks. Instead, the values of several two-level factors can be used to index the values of a single factor with more than two levels. As an example, the values of three two-level factors (P_1, P_2, and P_3) can be used to index the values of an eight-level factor(F), as follows:

P_1	P_2	P_3	F
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

The factors P_i are used only to derive the levels of the factor F; thus, they are called *pseudo-factors*, and F is called a *derived factor*. In general, k q-level pseudo-factors give rise to a single q^k-level derived factor. Block factors can be derived factors, and their associated formal factors (the P_i factors) are called *block pseudo-factors*.

The method for constructing an orthogonally confounded design for q-level factors in q^m runs distinguishes between the first m factors and the remaining factors. Each of the q^m different combinations of the first m factors occurs once in the design in an order similar to the preceding table. For this reason, the first m factors are called the *run-indexing factors*.

Table 15.7 summarizes the different types of factors discussed in this section.

Table 15.7. Types of Factors

Term	Definition
Block factor	Unavoidable factor whose effect is not of direct interest
Block pseudo-factor	Pseudo-factor used to derive levels of a block factor
Derived factor	Factor whose levels are derived from pseudo-factors
Design factor	Factor whose effect is of direct interest
Pseudo-factor	Formal factor combined to derive the levels of a real factor
Run-indexing factors	The first m design factors, whose q^m combinations index the runs in the design

Specifying Effects in the MODEL Statement

The FACTEX procedure accepts models that contain terms for main effects and interactions. *Main effects* are specified by writing variable names by themselves.

```
A B C
```

Interactions are specified by joining variable names with asterisks.

```
A*B B*C A*B*C
```

In addition, the *bar operator* (`|`) simplifies specification for interactions. The `@` *operator*, used in combination with the bar operator, further simplifies specification of interactions. For example, two ways of writing the complete set of effects for a model with up to three-factor interactions are

```
model estimate=(a b c a*b a*c b*c a*b*c);
```

and

```
model estimate=(a|b|c);
```

When the bar (|) is used, the right- and left-hand sides become effects, and their cross becomes an interaction effect. Multiple bars are permitted. The expressions are expanded from left to right, using rules given by Searle (1971). For example, A|B|C is evaluated as follows:

```
A|B|C  → {A|B}|C
       → {A B A*B}|C
       → A B A*B C A*C B*C A*B*C
```

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by specifying the number, preceded by an @ sign, at the end of the bar effect. For example, the specification A|B|C@2 results in only those effects that contain two or fewer factors. In this case, the effects A, B, A*B, C, A*C, and B*C are generated.

Factor Variable Characteristics in the Output Data Set

When you use the OUTPUT statement to save a design in a data set, and you rename and recode a factor, the type and length of the new variable are determined by whether you use the NVALS= or CVALS= option. A factor variable whose values are coded with the NVALS= specification is of numeric type. A factor variable whose values are coded with the CVALS= option is of character type, and the length of the variable is set to the length of the longest character string; shorter strings are padded with trailing blanks.

For example, in the specifications

```
cvals=('String 1' 'A longer string')
cvals=('String 1' 'String 2')
```

the first value in the first CVALS= specification is padded with seven trailing blanks. One consequence is that it no longer matches the 'String 1' of the second specification. To match two such values (for example, when merging two designs), use the TRIM function in the DATA step (see *SAS Language Reference: Dictionary* for details).

Statistical Details

Resolution

The resolution of a design indicates which effects can be estimated free of other effects. The resolution of a design is generally defined as the smallest *order*[*] of the interactions that are confounded with zero. Since having an effect of order $n + m$ confounded with zero is equivalent to having an effect of order n confounded with an effect of order m, the resolution can be interpreted as follows:

- If r is odd, then effects of order $e = (r - 1)/2$ or less can be estimated free of each other. However, at least some of the effects of order e are confounded with interactions of order $e + 1$. A design of odd resolution is appropriate when effects of interest are those of order e or less, while those of order $e + 1$ or higher are all negligible.
- If r is even, then effects of order $e = (r - 2)/2$ or less can be estimated free of each other and are also free of interactions of order $e + 1$. A design of even resolution is appropriate when effects of order e or less are of interest, effects of order $e + 1$ are not negligible, and effects of order $e + 2$ or higher are negligible. If the design uses blocking, interactions of order $e + 1$ or higher may be confounded with blocks.

In particular, for resolution 5 designs, all main effects and two-factor interactions can be estimated free of each other. For resolution 4 designs, all main effects can be estimated free of each other and free of two-factor interactions, but some two-factor interactions are confounded with each other and/or with blocks. For resolution 3 designs, all main effects can be estimated free of each other, but some of them are confounded with two-factor interactions.

In general, higher resolutions require larger designs. Resolution 3 designs are popular because they handle relatively many factors in a minimal number of runs. However, they offer no protection against interactions. If resources allow, you should use a resolution 5 design so that all main effects and two-factor interactions will be independently estimable. If a resolution 5 design is too large, you should use a design of resolution 4, which ensures estimability of main effects free of any two-factor interactions. In this case, if data from the initial design reveal significant effects associated with confounded two-factor interactions, further experiments can be run to distinguish between effects that are confounded with each other in the design. See page 458 for an example.

[*]The order of an effect is the number of factors involved in it. For example, main effects have order one, two-factor interactions have order two, and so on.

Note that most references on fractional factorial designs use Roman numerals to denote resolution of a design—III, IV, V, and so on. A common notation for an orthogonally confounded design of resolution r for k q-level factors in q^{k-p} runs is

$$q_r^{k-p}$$

For example, 2_{V}^{5-1} denotes a design for five two-level factors in 16 runs that allows estimation of all main effects and two-factor interactions. This chapter uses Arabic numerals for resolution since these are specified with the RESOLUTION= option in the MODEL statement.

Randomization

In many experiments, proper randomization is crucial to the validity of the conclusions. Randomization neutralizes the effects of systematic biases that may be involved in implementing the design and provides a basis for the assumptions underlying the analysis. Refer to Kempthorne (1975) for a discussion.

The way in which randomization is handled depends on whether the design involves blocking.

- For designs without block factors, proper randomization consists of randomly permuting the overall order of the runs and randomly assigning the actual levels of each factor to the theoretical levels it has for the purpose of constructing the design.
- For designs with block factors, proper randomization calls for first performing separate random permutations for the runs within each block, and then randomly permuting the order in which the blocks are run.

For example, suppose you generate a full factorial design for three two-level factors A, B, and C in eight runs. The following steps are involved in randomizing this design:

1. Randomly permute the order of the runs.

   $$\text{Runs: } \{1,2,3,4,5,6,7,8\} \quad \rightarrow \quad \{3,8,1,2,4,7,6,5\}$$

2. Randomly assign the actual levels to the theoretical levels for each factor.

   $$\begin{aligned} \text{Factor A levels: } \{0,1\} &\quad \rightarrow \quad \{1,-1\} \\ \text{Factor B levels: } \{0,1\} &\quad \rightarrow \quad \{1,-1\} \\ \text{Factor C levels: } \{0,1\} &\quad \rightarrow \quad \{-1,1\} \end{aligned}$$

Thus, the effect of the randomization is to transform the original design, as follows:

Run	A	B	C
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

$\longrightarrow$

Run	A	B	C
3	1	-1	-1
8	-1	-1	1
1	1	1	-1
2	1	1	1
4	1	-1	1
7	-1	-1	-1
6	-1	1	1
5	-1	1	-1

If the original design is in two blocks, then the first step is replaced with the following:

1. Randomly permute the order of the runs within each block.

   Block 1 runs: $\{1, 2, 3, 4\} \rightarrow \{4, 1, 2, 3\}$
   Block 2 runs: $\{5, 6, 7, 8\} \rightarrow \{8, 7, 6, 5\}$

2. Randomly permute the order of the blocks.

   Block levels: $\{1, 2\} \rightarrow \{2, 1\}$

The resulting transformation is shown in the following:

Run	Block	A	B	C
1	1	0	0	0
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0
5	2	0	0	1
6	2	0	1	0
7	2	1	0	0
8	2	1	1	1

$\longrightarrow$

Run	Block	A	B	C
8	2	-1	-1	1
7	2	-1	1	-1
6	2	1	-1	-1
5	2	1	1	1
4	1	-1	-1	-1
1	1	1	1	-1
2	1	1	-1	1
3	1	-1	1	1

If you use the RANDOMIZE option in the OUTPUT statement, the output data set contains a randomized design. In some cases, it is appropriate to randomize the run order but not the assignment of theoretical factor levels to actual levels. In these cases, specify both the NOVALRAN and RANDOMIZE options in the OUTPUT statement.

Replication

In quality improvement applications, it is often important to analyze both the mean response of a process and the variability around the mean. To study variability with an experimental design, you must take several measurements of the response for each different combination of the factors of interest; that is, you must *replicate* the design runs.

Replicating a Fixed Number of Times

A simple method of replication is to take a given number of measurements for each combination of factor levels in the basic design. You can replicate runs in the design by specifying numbers for the POINTREP= and DESIGNREP= options in the OUTPUT statement. For example, the following code constructs a full 2^2 design and uses both of these options to replicate the design three times:

```
proc factex;
   factors a b;
   output out=one pointrep =3;
run;
   output out=two designrep=3;
run;
```

The output data sets ONE and TWO have the same 12 runs, but they are in different orders. in the data set ONE, the POINTREP= option causes all three replications of each run to occur together, as shown in Figure 15.1.

OBS	A	B	
1	-1	-1	3 replicates of run 1
2	-1	-1	
3	-1	-1	
4	-1	1	3 replicates of run 2
5	-1	1	
6	-1	1	
7	1	-1	3 replicates of run 3
8	1	-1	
9	1	-1	
10	1	1	3 replicates of run 4
11	1	1	
12	1	1	

Figure 15.1. Four-Run Design Replicated Using the POINTREP= Option

On the other hand, in the data set TWO, the DESIGNREP= option causes all four runs of the design to occur together three times, as shown in Figure 15.2.

	OBS	A	B
Replicate 1	1	-1	-1
	2	-1	1
	3	1	-1
	4	1	1
Replicate 2	5	-1	-1
	6	-1	1
	7	1	-1
	8	1	1
Replicate 3	9	-1	-1
	10	-1	1
	11	1	-1
	12	1	1

Figure 15.2. Four-Run Design Replicated Using the DESIGNREP= Option

Replicating with an Outer Array

Another method of design replication considers the range of environmental conditions over which the process should maintain consistency. This method distinguishes between *control factors* and *noise factors*. Control factors are factors that are under the control of the designer or the process engineer. Noise factors cause the performance of a product to vary when the nominal values of the control variables are fixed (noise factors are controllable for the purposes of experimenting with the process). Typical noise factors are variations in the manufacturing environment or the customer's environment due to temperature or humidity. The object of experimentation is to find the best settings for the control factors for a variety of settings for the noise factors. In other words, the goal is to develop a process that runs well in a variety of environments. Refer to Dehnad (1989) and Phadke (1989) for further discussion.

To achieve this goal, a collection of environmental conditions (settings for the noise factors) is determined. This collection is called the *outer array*. Each run in the control factor design (*inner array*) is replicated within each of these environments. The mean and variance of the process over the outer array are computed for each run in the inner array. Either the outer array or the inner array may consist of all possible different settings for the associated factors, or they may be fractions of all possible settings.

You can replicate designs in this way by using data set names for the POINTREP= and DESIGNREP= options in the OUTPUT statement. If you construct a design for your control factors and you want to run a noise factor design for each run in the control factor design, specify the data set that holds the noise factor design (that is, the *outer array*) with the POINTREP= option in the OUTPUT statement. See Example 15.14 on page 482 for an example.

Confounding Rules

Confounding rules give the values of factors in terms of the values of the *run-indexing factors* for a design. (See "Types of Factors" on page 488 for a discussion of run-indexing factors.) The FACTEX procedure uses these rules to construct designs. The confounding rules also determine the alias structure of the design. To display the confounding rules for a design, use the CONFOUNDING option in the EXAMINE statement.

For two-level factors, the rules are displayed in a multiplicative notation using the default values of -1 and $+1$ for the factors. For example, the confounding rule

```
X8 = X1*X2*X3*X4*X5*X6*X7
```

means that the level of factor X8 is derived as the product of the levels of factors X1 through X7 for each run in the design. X8 will always have a value of +1 or -1 since these are the values of X1 through X7. For factors with $q > 2$ levels, confounding rules are printed in an additive notation, and the arithmetic is performed in the Galois field of size q. For example, in a design for three-level factors, the confounding rule

```
F = B + (2*C) + D + (2*E)
```

means that the level of factor F is computed by adding the levels of B and D and two times the levels of C and E, all modulo 3. Note that if q is not a prime number, Galois field arithmetic is not equivalent to arithmetic modulo q.

Blocks are introduced into designs by using *block pseudo-factors*. The confounding rule for the ith block pseudo-factor has `[Bi]` on the left-hand side.

For details on how confounding rules are constructed, see "Suitable Confounding Rules" on page 504.

Alias Structure

The alias structure of a design identifies which effects are confounded (or aliased) with each other in the design. Note the difference between alias structure and confounding rules: the confounding rules are used to construct the design, and the alias structure is a result of using a given set of confounding rules. To display the alias structure for a design, use the ALIAS option in the EXAMINE statement.

Examining the alias structure is important since aliased effects cannot be estimated separately from one another. When several effects are listed as equal, the effects are all jointly aliased with one another and form an *alias chain* or *alias string*. For example,

```
TEMP*MOIST = HPRESS*GATE = THICK*SCREW = BPRESS*TIME
```

is an alias chain that shows the relationship between four two-factor interactions. If you want separate estimates of TEMP*MOIST and THICK*SCREW, for instance, a design with this alias chain would not be acceptable. Designs of even resolution $2k$ contain one or more such chains of confounded k-factor interactions.

By default, the FACTEX procedure displays alias chains with effects up to a certain order d, where main effects are order 1, two-factor interactions are order 2, and so on. The value of d can be specified in the ALIAS option , or you can use the default calculated by the procedure ; see page 448 for details. Alias chains that are confounded with blocks are displayed with [B] on the left-hand side.

Minimum Aberration

As discussed in "Speeding up the Search" on page 507, the FACTEX procedure uses a tree search algorithm to find the confounding rules of a design that matches the size and resolution you specify. There may be more than one solution set of confounding rules, and usually the FACTEX procedure chooses the first one it finds. However, there can still be important differences between designs with the same resolution; to deal with these differences, Fries and Hunter (1980) introduced the concept of *aberration* in confounded fractional factorial designs. This section defines aberration and discusses how to request minimum aberration designs with the FACTEX procedure.

Recall that a design has resolution r if r is the smallest order of the interactions that are confounded with zero. The idea behind minimum aberration is that a resolution r design that confounds *as few* rth-order interactions as possible is preferable. Technically, the aberration of a design is the vector $\mathbf{k} = \{k_1, k_2, \ldots\}$, where k_i is the number of ith-order interactions that are confounded with zero. A design with aberration $\mathbf{k}$ has *minimum aberration* if $\mathbf{k} \leq \mathbf{k}'$ for any other design with aberration $\mathbf{k}'$, in the sense that $k_i < k_i'$ for the first i for which $k_i \neq k_i'$.

For example, consider the resolution 4 design for seven two-level factors in 32 runs (2^{7-2}_{IV}) discussed in Example 15.11 on page 472.

By specifying 5 for the order d for the ALIASING option, you can see how many fourth- and fifth-order interactions are confounded with zero. The default design constructed by the FACTEX procedure confounds two fourth-order interactions and no fifth-order interactions with zero.

```
0 = A*B*F*G = C*D*E*G
```

Thus, part of the aberration for this design is

$$\{k_3, k_4, k_5, \ldots\} \quad = \quad \{0, 2, 0, \ldots\}$$

On the other hand, the design constructed using the MINABS option confounds only one fourth-order interaction and two fifth-order interactions with zero.

```
0 = C*D*E*F = A*B*C*F*G = A*B*D*E*G
```

Thus, part of the aberration for this design is

$$\{k_3', k_4', k_5', \ldots\} \quad = \quad \{0, 1, 2, \ldots\}$$

Since the two aberrations first differ for k_4 and k_4', and since $k_4' < k_4$, the aberration for the second design is less than the aberration for the first design.

The definition of aberration requires evaluating the number of ith-order interactions that are confounded with zero for all $i \leq k$, where k is the number of factors. Since there are q^k generalized interactions between k q-level factors, this evaluation can be prohibitive if there are many factors. Moreover, it is unnecessary if, as is usually the case, you are interested only in small-order interactions. Therefore, when you specify the MINABS option, by default the FACTEX procedure evaluates the aberration only up to order d, where d is the same as the default maximum order for listing the aliasing (see the specifications for the EXAMINE statement on page 448). You can set d to any level by specifying (d) immediately after the MINABS option ; see page 451 for details.

The discussion so far has dealt only with fractional unblocked designs, but one more point to consider is the definition of aberration for block designs. Define a vector $\mathbf{b} = b_1, b_2, \ldots$ similar to the aberration vector $\mathbf{k}$, except that b_i is the number of ith-order interactions that are confounded with blocks. A block design with $\mathbf{k}$ and $\mathbf{b}$ has minimum aberration if

- $\mathbf{k}$ is minimum
- among all designs with minimum $\mathbf{k}$, $\mathbf{b}$ is minimum

Output

By default, the FACTEX procedure does not display any output. For each design that it constructs, the procedure displays a message in the SAS log that provides

- the number of runs in the design
- the number of blocks and the block size, if appropriate
- the maximum resolution of the design

If you use the DESIGN option in an EXAMINE statement, the procedure displays the coded runs in the design using standard values, as described in the "OUTPUT Statement" section on page 452. If you use the CONFOUNDING option in an EXAMINE statement, the procedure displays the confounding rules used to construct the design. If you use the ALIAS option in an EXAMINE statement, the procedure displays the alias structure for the design.

The FACTEX procedure also creates output data sets with the OUTPUT statement. Since the procedure is interactive, you can use many OUTPUT statements in a given run of the FACTEX procedure to produce many output data sets if you separate them with **`run;`** statements.

ODS Tables

The following table summarizes the ODS tables that you can request with the PROC FACTEX statement.

Table 15.8. ODS Tables Produced in PROC FACTEX

ODS Table Name	Description	Statement	Option
DesignPoints	Design points	EXAMINE	DESIGN
FactorRules	Treatment factor confounding rules	EXAMINE	CONFOUNDING
BlockRules	Block factor confounding rules	EXAMINE	CONFOUNDING
Aliasing	Alias structure	EXAMINE	ALIASING

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/QC® User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. 1994 pp.

SAS/QC® User's Guide, Version 8

ISBN 1–58025–493–4

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

1st printing, October 1999