SAS/QC® 13.2 User's Guide

The FACTEX Procedure

This document is an individual chapter from *SAS/QC® 13.2 User's Guide*.

The correct bibliographic citation for the complete manual is as follows: SAS Institute Inc. 2014. *SAS/QC® 13.2 User's Guide*. Cary, NC: SAS Institute Inc.

Copyright © 2014, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

For a hard-copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a) and DFAR 227.7202-4 and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

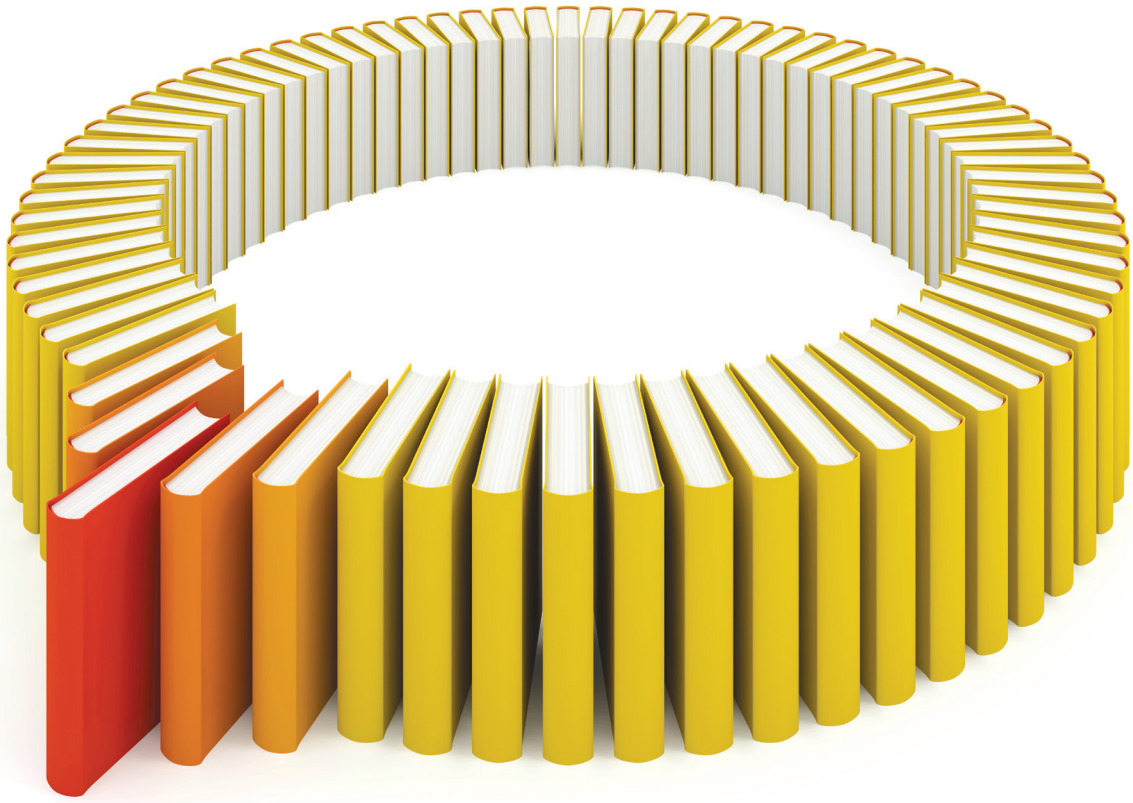
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

August 2014

SAS provides a complete selection of books and electronic products to help customers use SAS® software to its fullest potential. For more information about our offerings, visit **support.sas.com/bookstore** or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.

sas
THE POWER TO KNOW®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613

Chapter 7
The FACTEX Procedure

Contents

Introduction **608**
Overview 608
Features 609
Learning about the FACTEX Procedure 610
Getting Started 610
Example of a Two-Level Full Factorial Design 610
Example of a Full Factorial Design in Two Blocks 612
Example of a Half-Fraction Factorial Design 614
Using the FACTEX Procedure Interactively 616
Details of the FACTEX Procedure **617**
Syntax 617
Summary of Functions 617
Summary of Designs 619
Statement Descriptions 621
PROC FACTEX Statement 621
BLOCKS Statement 622
UNITEFFECT Statement 624
EXAMINE Statement 625
FACTORS Statement 626
MODEL Statement 627
OUTPUT Statement 629
SIZE Statement 633
Advanced Examples 634
Example 7.1: Completely Randomized Design 634
Example 7.2: Resolution 4 Augmented Design 635
Example 7.3: Factorial Design with Center Points 638
Example 7.4: Fold-Over Design 639
Example 7.5: Randomized Complete Block Design 641
Example 7.6: Two-Level Design with Design Replication and Point Replication . . . 642
Example 7.7: Mixed-Level Design Using Design Replication and Point Replication . 645
Example 7.8: Mixed-Level Design Using Pseudo-Factors 647
Example 7.9: Mixed-Level Design by Collapsing Factors 648
Example 7.10: Hyper-Graeco-Latin Square Design 649
Example 7.11: Resolution 4 Design with Minimum Aberration 651
Example 7.12: Replicated Blocked Design with Partial Confounding 654
Example 7.13: Incomplete Block Design 657

Example 7.14: Design with Inner Array and Outer Array . . . 660
Example 7.15: Fractional Factorial Split-Plot Designs . . . 665
Example 7.16: A Design for a Three-Step Process . . . 669
Example 7.17: A Strip-Split-Split-Plot Design . . . 672
Example 7.18: Design and Analysis of a Complete Factorial Experiment . . . 674
Computational Details . . . 676
Types of Factors . . . 676
Specifying Effects in the MODEL Statement . . . 677
Factor Variable Characteristics in the Output Data Set . . . 678
Statistical Details . . . 678
Resolution . . . 678
Randomization . . . 679
Replication . . . 681
Confounding Rules . . . 683
Alias Structure . . . 684
Minimum Aberration . . . 684
MaxClear Designs . . . 685
Split-Plot Designs . . . 686
Output . . . 686
ODS Tables . . . 687
Theory of Orthogonal Designs . . . **687**
Overview . . . 687
Structure of General Factorial Designs . . . 687
Suitable Confounding Rules . . . 688
Design Factors . . . 688
Block Factors . . . 689
General Criteria . . . 690
Searching for Confounding Rules . . . 690
Speeding Up the Search . . . 691
General Recommendations . . . 692
References . . . **692**

Introduction to the FACTEX Procedure

Overview

The FACTEX procedure constructs orthogonal factorial experimental designs. These designs can be either full or fractional factorial designs, and they can be with or without blocks. You can also construct designs for experiments with multiple stages, such as split-plot (Huang, Chen, and Voelkel 1998) and split-lot designs (Butler 2004). After you have constructed a design by using the FACTEX procedure and run the experiment, you can analyze the results with a variety of SAS procedures including the GLM and REG procedures.

Factorial experiments are useful for studying the effects of various factors on a response. Texts that discuss experimental design include Box, Hunter, and Hunter (1978), Cochran and Cox (1957), Montgomery (1991), and Wu and Hamada (2000). For details about the general mathematical theory of orthogonal factorial designs, refer to Bose (1947).

NOTE: For two-level designs, instead of using PROC FACTEX directly, a more appropriate tool might be the ADX Interface for Design of Experiments. The ADX Interface is designed primarily for engineers and researchers who require a point-and-click solution for the entire experimental process, from building the designs through determining significant effects to optimization and reporting. ADX gives you most of the two-level designs provided by the FACTEX procedure in a system that integrates construction and analysis of designs, without the need for programming. In addition to two-level designs for standard models (with and without blocking), ADX makes it easy to use PROC FACTEX to construct designs for estimating particular effects of interest. Moreover, ADX also uses the OPTEX procedure to construct two-level designs of nonstandard sizes. For more information, see *Getting Started with the SAS ADX Interface for Design of Experiments*.

Features

There is no inherent limit to the number of factors and the size of the design that you can construct with the FACTEX procedure. Instead of looking up designs in an internal table, the FACTEX procedure uses a general algorithm to search for the construction rules for a specified design.

You can use the FACTEX procedure to generate designs such as the following:

- factorial designs, such as 2^3 designs, with and without blocking
- fractional factorial designs, such as 2^{4-1}_{IV}, with and without blocking
- split-plot and fractional split-plot designs
- three-level designs, with and without blocking
- mixed-level factorial designs, such as 4×3 designs, with and without blocking
- randomized complete block design
- factorial designs with outer arrays
- hyper-Graeco-Latin square designs

You can also create more complex designs, such as incomplete block designs, by using the FACTEX procedure in conjunction with the DATA step.

You can save the design constructed by the FACTEX procedure in a SAS data set. After you have run your experiment, you can add the values of the response variable and use the GLM procedure to perform analysis of variance and study significance of effects.

The FACTEX procedure is an interactive procedure. After specifying an initial design, you can submit additional statements without reinvoking the procedure. After you have constructed a design, you can do the following:

- print the design points

- examine the alias structure for the design
- modify the design by changing its size, changing the use of blocking, or specifying the effects of interest in the model again
- output the design to a data set
- examine the confounding rules that generate the design
- randomize the design
- replicate the design
- recode the design from standard values (such as ± 1) to values appropriate for your situation
- find another design

Learning about the FACTEX Procedure

To learn the basic syntax of the FACTEX procedure, read the section "Getting Started" on page 610, which contains some simple introductory examples. The summary tables in the section "Summary of Functions" on page 617 provide an overview of the syntax. The section "Summary of Designs" on page 619 shows simple ways to construct full factorial designs and fractional factorial designs. The section "Advanced Examples" on page 634 illustrates construction of complex designs.

Getting Started

The following introductory examples illustrate the capabilities of the FACTEX procedure. See the section "Advanced Examples" on page 634 for illustrations of complex features.

Example of a Two-Level Full Factorial Design

NOTE: See *Two-Level Full Factorial Design* in the SAS/QC Sample Library.

This example introduces the basic syntax used with the FACTEX procedure.

An experimenter is interested in studying the effects of three factors—cutting speed (Speed), feed rate (FeedRate), and tool angle (Angle)—on the surface finish of a metallic part and decides to run a complete factorial experiment with two levels for each factor as follows:

Factor	Low Level	High Level
Cutting speed	300	500
Feed rate	20	30
Tool angle	6	8

This is a 2^3 factorial design—in other words, a complete factorial experiment with three factors, each at two levels. Hence there are eight runs in the experiment. Since complete factorial designs have full resolution, all of the main effects and interaction terms can be estimated. For a definition of the design resolution, see the section "Resolution" on page 678.

You can use the following statements to create the required design:

```
proc factex;
   factors Speed FeedRate Angle;
   examine design;
run;
```

These statements invoke the FACTEX procedure, list factor names, and display the generated design points. By default, the FACTEX procedure assumes that the size of the design is a full factorial and that each factor has only two levels.

After you submit the preceding statements, you see the following messages in the SAS log:

```
NOTE: No design size specified.
      Default is a full replicate in 8 runs.
NOTE: Design has 8 runs, full resolution.
```

The output is shown in Figure 7.1. The two factor levels are represented by the coded values –1 and +1.

Figure 7.1 2^3 Factorial Design

The FACTEX Procedure

Design Points			
Experiment Number	Speed	FeedRate	Angle
1	-1	-1	-1
2	-1	-1	1
3	-1	1	-1
4	-1	1	1
5	1	-1	-1
6	1	-1	1
7	1	1	-1
8	1	1	1

If you prefer to work with the actual (decoded) values of the factors, you can specify these values in an OUTPUT OUT= statement, as follows:

```
proc factex;
   factors Speed FeedRate Angle;
   output out=SavedDesign
          Speed    nvals=(300 500)
          FeedRate nvals=(20 30)
          Angle    nvals=(6 8);
run;
proc print;
run;
```

The OUTPUT statement in PROC FACTEX recodes the factor levels and saves the constructed design in the SavedDesign data set. Since the levels in this example are of numeric type, you use the NVALS= option to list the factor levels. Optionally, you can use the CVALS= option for levels of character type (see the section "Example of a Full Factorial Design in Two Blocks" on page 612). The design is saved in a user-specified output data set (SavedDesign). This is verified by the following message in the SAS log:

```
NOTE: The data set WORK.SAVEDDESIGN has 8 observations
      and 3 variables.
```

Figure 7.2 shows a listing of the data set SavedDesign.

Figure 7.2 2^3 Factorial Design after Decoding

Obs	Speed	FeedRate	Angle
1	300	20	6
2	300	20	8
3	300	30	6
4	300	30	8
5	500	20	6
6	500	20	8
7	500	30	6
8	500	30	8

Although small complete factorial designs are not difficult to create manually, you can easily extend this example to construct a design with many factors.

Example of a Full Factorial Design in Two Blocks

NOTE: See *Full Factorial Design in Two Blocks* in the SAS/QC Sample Library.

The previous example illustrates a complete factorial experiment that involves eight runs and three factors: cutting speed (Speed), feed rate (FeedRate), and tool angle (Angle).

Now, suppose two machines (A and B) are used to complete the experiment, with four runs being performed on each machine. As there is the possibility that the machine affects the part finish, you should consider machine as a block factor and account for the block effect in assigning the runs to machines.

The following statements construct a blocked design:

```
proc factex;
   factors Speed FeedRate Angle;
   blocks nblocks=2;
   model resolution=max;
   examine design;
run;
```

The FACTORS statement in PROC FACTEX specifies three factors of a 2^3 factorial. The BLOCKS statement specifies that the number of blocks is 2. The RESOLUTION=MAX option in the MODEL statement specifies a design with the highest resolution—that is, the best design in a general sense. Optionally, if you know the resolution of the design, you can replace RESOLUTION=MAX with RESOLUTION=*r*, where *r* is the resolution number. For information about resolution, see the section "Resolution" on page 678.

By default, the FACTEX procedure assumes that the size of the design is a full factorial and that each factor is at two levels.

After you submit the preceding statements, you see the following messages in the SAS log:

```
NOTE: No design size specified.
      Default is a full replicate in 8 runs.
NOTE: Design has 8 runs in 2 blocks of size 4,
      resolution = 6.
```

The output is shown in Figure 7.3. Note that, by default, the name for the block variable is BLOCK and its levels are 1 and 2. Also, note that the default factor levels for a two-level design are –1 and 1.

Figure 7.3 2^3 Factorial Design in Two Blocks before Decoding

The FACTEX Procedure

Design Points				
Experiment Number	Speed	FeedRate	Angle	Block
1	-1	-1	-1	1
2	-1	-1	1	2
3	-1	1	-1	2
4	-1	1	1	1
5	1	-1	-1	2
6	1	-1	1	1
7	1	1	-1	1
8	1	1	1	2

You can rename the block variable and use actual levels for the block variable appropriate for your situation as follows:

```
proc factex;
   factors Speed FeedRate Angle;
   blocks nblocks=2;
   model resolution=max;
   output out=BlockDesign
     Speed    nvals=(300 500)
     FeedRate nvals=(20 30)
     Angle    nvals=(6 8)
     blockname=Machine cvals=('A' 'B');
run;

proc print;
run;
```

Figure 7.4 shows the listing of the design saved in the data set BlockDesign.

Figure 7.4 2^3 Factorial Design in Two Blocks after Decoding

Obs	Machine	Speed	FeedRate	Angle
1	A	300	20	6
2	A	300	30	8
3	A	500	20	8
4	A	500	30	6
5	B	300	20	8
6	B	300	30	6
7	B	500	20	6
8	B	500	30	8

Example of a Half-Fraction Factorial Design

NOTE: See *Half-Fraction Factorial Design* in the SAS/QC Sample Library.

Often you do not have the resources for a full factorial design. In this case, a fractional factorial design is a reasonable alternative, provided that the effects of interest can be estimated.

Box, Hunter, and Hunter (1978) describe a fractional factorial design for studying a chemical reaction to determine what percentage of the chemicals responded in a reactor. The researchers identified the following five treatment factors that were thought to influence the percentage of reactant:

- the feed rate of the chemicals (FeedRate), ranging from 10 to 15 liters per minute
- the percentage of the catalyst (Catalyst), ranging from 1% to 2%
- the agitation rate of the reactor (AgitRate), ranging from 100 to 120 revolutions per minute
- the temperature (Temperature), ranging from 140 to 180 degrees Celsius
- the concentration (Concentration), ranging from 3% to 6%

The complete 2^5 factorial design requires 32 runs, but the experimenter can afford only 16 runs.

Suppose that all main effects and two-factor interactions are to be estimated. An appropriate design for this situation is a design of resolution 5 (denoted as 2^{5-1}_{V}), in which no main effect or two-factor interaction is aliased with any other main effect or two-factor interaction but in which two-factor interactions are aliased with three-factor interactions. This design loses the ability to estimate interactions between three or more factors, but this is usually not a serious loss. For more information about resolution, see the section "Resolution" on page 678.

You can use the following statements to construct a 16-run factorial design that has five factors and resolution 5:

```
proc factex;
   factors FeedRate Catalyst AgitRate Temperature Concentration;
   size design=16;
   model resolution=5;
   output out=Reaction FeedRate      nvals=(10 15)
                       Catalyst      nvals=(1 2)
                       AgitRate      nvals=(100 120)
                       Temperature   nvals=(140 180)
                       Concentration nvals=(3 6);
run;
proc print;
run;
```

The design saved in the Reaction data set is listed in Figure 7.5.

Figure 7.5 Half-Fraction of a 2^5 Design for Reactors

Obs	FeedRate	Catalyst	AgitRate	Temperature	Concentration
1	10	1	100	140	6
2	10	1	100	180	3
3	10	1	120	140	3
4	10	1	120	180	6
5	10	2	100	140	3
6	10	2	100	180	6
7	10	2	120	140	6
8	10	2	120	180	3
9	15	1	100	140	3
10	15	1	100	180	6
11	15	1	120	140	6
12	15	1	120	180	3
13	15	2	100	140	6
14	15	2	100	180	3
15	15	2	120	140	3
16	15	2	120	180	6

The use of a half-fraction causes some interaction terms to be confounded with each other. You can use the EXAMINE statement with the ALIASING option to determine which interaction terms are aliased, as follows:

```
proc factex;
   factors FeedRate Catalyst AgitRate Temperature Concentration;
   size design=16;
   model resolution=5;
   examine aliasing;
run;
```

The alias structure summarizes the estimability of all main effects and two- and three-factor interactions. Figure 7.6 indicates that each of the three-factor interactions is confounded with a two-factor interaction. Thus, if a particular three-factor interaction is believed to be significant, the aliased two-factor interaction cannot be estimated with this half-fraction design.

Figure 7.6 Alias Structure of Reactor Design

The FACTEX Procedure

Aliasing Structure
FeedRate
Catalyst
AgitRate
Temperature
Concentration
FeedRate*Catalyst = AgitRate*Temperature*Concentration
FeedRate*AgitRate = Catalyst*Temperature*Concentration
FeedRate*Temperature = Catalyst*AgitRate*Concentration
FeedRate*Concentration = Catalyst*AgitRate*Temperature
Catalyst*AgitRate = FeedRate*Temperature*Concentration
Catalyst*Temperature = FeedRate*AgitRate*Concentration
Catalyst*Concentration = FeedRate*AgitRate*Temperature
AgitRate*Temperature = FeedRate*Catalyst*Concentration
AgitRate*Concentration = FeedRate*Catalyst*Temperature
Temperature*Concentration = FeedRate*Catalyst*AgitRate

When you submit the preceding statements, the following message is displayed in the SAS log:

```
NOTE: Design has 16 runs, resolution = 5.
```

This message confirms that the design exists. If you specify a factorial design that does *not* exist, an error message is displayed in the SAS log. For instance, suppose that you replaced the MODEL statement in the preceding example with the following statement:

```
model resolution=6;
```

Since the maximum resolution of a 2^{5-1} design is 5, the following message appears in the SAS log:

```
ERROR: No such design exists.
```

In general, it is good practice to check the SAS log to see if a design exists.

Using the FACTEX Procedure Interactively

By using the FACTEX procedure interactively, you can quickly explore many design possibilities. The following steps provide one strategy for interactive use:

1. Invoke the procedure by using the PROC FACTEX statement, and use a FACTORS statement to identify factors in the design.
2. For a design that involves blocking, use the BLOCKS and MODEL statements. You might want to use the optimization features for the BLOCKS statement.
3. For a fractional replicate of a design, use the SIZE and MODEL statements to specify the characteristics of the design. If the design involves blocking, use a BLOCKS statement as well. If you are unsure of the size of the design or of the number of blocks, use the optimization features for either the BLOCKS or SIZE statement.

4 Enter a RUN statement and check the SAS log to see if the design exists. If a design exists, go on to the next step; otherwise, modify the characteristics given in the SIZE, BLOCKS, and MODEL statements.

5 Examine the alias structure of the design. If it is not appropriate for your situation, go back to step 2 and search for another design.

6 After you have repeated steps 2, 3, and 4 and found an acceptable design, use the OUTPUT statement to save the design. You can optionally recode factor values, recode and rename the block factor, and create new factors by using output-value settings.

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*) > > < **MAXCLEAR** < (*d*) > > > **;**
BLOCKS *block-specification* **;**
UNITEFFECT *uniteffect* / < **WHOLE=()** > < **SUB=()** > **;**
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 7.1 to Table 7.4 classify the statements and options in PROC FACTEX by function.

Table 7.1 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

Table 7.1 *continued*

Function	Statement	Option
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 nonnegligible 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 7.2 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 7.3 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*

Table 7.3 *continued*

Function	Statement	Option
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 7.4 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 7.5 summarizes basic design types that you can construct with the FACTEX procedure by providing example code for each type.

Table 7.5 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 Temperature Time;`
	`  examine design;`
	`run;`

Table 7.5 *continued*

Design Type	Example Statements

A full factorial design in three factors, each at three levels coded as –1, 0, and +1.	`proc factex;`
	`  factors Pressure Temperature 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 Temperature Time;`
	`  output out= SavedDesign designrep= 2`
	`  Pressure cvals=('low' 'high')`
	`  Temperature 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 Temperature Time;`
	`  output out= SavedDesign`
	`  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 Temperature Time;`
	`  output out  =+ SavedDesign`
	`  pointrep=+ OutArray;`
	`run;`

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 Temperature Time;`
	`  blocks nblocks= 2;`
	`  output out= SavedDesign;`
	`run;`

Table 7.5 *continued*

Design Type	Example Statements

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 Temperature Time;`
	`  blocks size= 4;`
	`  output out= SavedDesign`
	`  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 Temperature 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 Temperature 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  = SavedDesign;`
	`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 information about the NOCHECK option, see the section "Speeding Up the Search" on page 691.

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 blocks or split-plot units 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 as follows:

- 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 seven 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 the section "Block Size Restrictions" on page 624 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 information about pseudo-factors, see the section "Types of Factors" on page 676.

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 the section "Block Size Restrictions" on page 624 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 the section "Example of a Full Factorial Design in Two Blocks" on page 612 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 the section "SIZE Statement" on page 633 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).

UNITS=(*units-specifications***)**

specifies unit factors that index the runs of the experiment. Each *unit-specification* has the form

unitfactor **=** *number-of-levels*

where the number of levels for each unit factor must be a power of the number of levels specified in the FACTORS statement (2 by default). You can give multiple *unit-specifications* in the UNITS= option; the product of their numbers of levels must be less than the size of the experiment, as specified in the SIZE statement.

Unit factors are not involved in the model structure of the design. Instead, you use a UNITS=() blocks specification in conjunction with one or more UNITEFFECT statements to constrain how the factor levels can change across the runs of the experiment.

The following statement illustrates how to specify two unit factors:

```
blocks units=(Unit1=4 Unit2=8);
```

See the section "Split-Plot Designs" on page 686 for details about how to use the UNITS= option and the UNITEFFECT statement to construct split-plot designs.

Equivalent BLOCK 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 as follows:

```
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 676 and "Theory of Orthogonal Designs" on page 687.

UNITEFFECT Statement

UNITEFFECT *uniteffect* / < **WHOLE=(***wholeuniteffects***)** > < **SUB=(***subuniteffects***)** > **;**

You use the UNITEFFECT statement to specify constraints on how the factor levels can change across the runs of the experiment. Such constraints are known as randomization restrictions. UNITEFFECT statements are used in conjunction with a UNITS specification in the BLOCKS statement, where unit factors are defined that index the runs of the experiment.

The *uniteffect* is an interaction between *unitfactors* defined in the BLOCKS UNITS=() specification:

*unitfactor * . . . * unitfactor*

It defines a partition of the runs on which to apply whole-unit and subunit effects of the factors named in the FACTORS statement.

The *wholeuniteffects* specified by the WHOLE=() option typically define a necessary feature of how the experiment must be designed, and are thus known as "design constraints." In contrast, the SUB=() option indicates which unit mean contrasts will be used to compute the *subuniteffects* and which random error terms will be used to test them, and are thus known as "model constraints." For both *wholeuniteffects* and *subuniteffects*, the effects listed must be enclosed within parentheses, as in the MODEL statement. See the section "Specifying Effects in the MODEL Statement" on page 677 for details.

If you have specified units in the BLOCK statement as

```
blocks units=(WholePlot=4);
```

then the following statement illustrates how to specify unit effects corresponding to these units:

```
uniteffect WholePlot / whole=(x1-x3) sub=(x4-x6);
```

See the section "Split-Plot Designs" on page 686 for details about how to use the UNITS= option and the UNITEFFECT statement to construct split-plot designs.

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.

ABERRATION

AB

displays the aberration vector for the design, summarizing the confounded interactions. See the section "Minimum Aberration" on page 684 for more information.

ALIASING < (< *d* > < UNITS < = ONE | ALL > >) >

A < (< *d* > < UNITS < = ONE | ALL > >) >

lists the alias structure of the design, which identifies effects that are confounded with one another and are thus, indistinguishable.

You can specify *d* in parentheses 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 the following, where main effects have order 1, two-factor interactions have order 2, and so on:
  - one plus the largest order of an effect to be estimated
  - the largest order of an effect considered to be nonnegligible

If your design involves unit effects, then you can also specify UNITS in parentheses immediately after the ALIASING option to display for each treatment effect alias string the top unit effect with which it is aliased. Using the UNITS option can give you information about which error stratum can be used to

estimate the background error variance for each estimable treatment effect. If unit effects are nested, as they typically are in complex split-plot designs, then treatment effects can be aliased with more than one unit effect. In this case, specify (UNITS=ALL) to see all unit effects with which each treatment effect is aliased.

For details about aliasing, see the section "Alias Structure" on page 684.

CONFOUNDING

C

lists the confounding rules used to construct the design. For the definition of confounding rules, see the sections "Confounding Rules" on page 683 and "Suitable Confounding Rules" on page 688.

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 the section "OUTPUT Statement" on page 629.

SUMMARY < (< *d* >) >

S < (< *d* >) >

displays the modeling summary for the design, summarizing how many interactions of each order are estimable and how many are clearly estimable (that is, unaliased with any other interactions of interest).

You can specify *d* in parentheses immediately after the SUMMARY option to display the modeling summary accounting for effects up to and including order *d*. The default value for *d* is determined automatically from the model as it is for the ALIASING option.

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: 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 the section "Types of Factors" on page 676 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 the section "Structure of General Factorial Designs" on page 687.

MODEL Statement

MODEL *model-specification* < / < **MINABS** < (*d*) > > < **MAXCLEAR** < (*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.

NOTE: 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 the section "Specifying Effects in the MODEL Statement" on page 677 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 ESTIMATE= 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 7.8 for an example that uses the ESTIMATE= option. See "Theory of Orthogonal Designs" on page 687 for details about 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 information about design resolution, see the section "Resolution" on page 678. For an example of *model specification* that uses the RESOLUTION=*r* option, see the section "Example of a Half-Fraction Factorial Design" on page 614. For an example of the RESOLUTION=MAX option, see the section "Example of a Full Factorial Design in Two Blocks" on page 612.

The following options for the MODEL statement are available:

MAXCLEAR <(*d*)>
requests a search for a design that maximizes the number of clear interactions. Clear interactions are those which are not aliased with any other effects that are either required to be estimable or assumed to be nonnegligible. Specifying (*d*) immediately after the MAXCLEAR option requests a search for a maximum-clarity design that involves interactions up to order *d*. The default value for *d* is determined automatically from the model, as it is for the ALIASING option in the EXAMINE statement, 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 the following, where main effects have order 1, two-factor interactions have order 2, and so on:
  - one plus the largest order of an effect to be estimated
  - the largest order of an effect considered to be nonnegligible

For details about MaxClear designs, see the section "MaxClear Designs" on page 685.

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 that involves interactions up to order *d*. 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 $d = r + 2$.
- If you specify the model with a list of effects in the MODEL statement, then *d* is the larger of the following, where main effects have order 1, two-factor interactions have order 2, and so on:
  - three plus twice the largest order of an effect to be estimated
  - one plus twice the largest order of an effect considered to be nonnegligible

  See the section "Minimum Aberration" on page 684 for more information. For an example of the MINABS option, see Example 7.11.

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 7.6 lists equivalent ways to specify common models.

Table 7.6 Equivalent of Model Specifications

RES= option	EST= and NONNEG= options
`model res=3`	`model est=(x1-x+f);`
`model res=4`	`model est=(x1-x+ f) nonneg=(x1|x2|x3|+...+|x+f+@2);`
`model res=5`	`model est=(x1|x2|x3|+...+|x+ f+@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 that uses the RESOLUTION= option rather than listing the effects. For more on interpreting the resolution number, see the section "Resolution" on page 678.

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 (Temperature) 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)
                  Temperature 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 information about recoding a factor, see "Factor Variable Characteristics in the Output Data Set" on page 678.

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 the 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 the section "Recode Design Factors" on page 629.

CVALS= lists new character levels for the block factor. For details, see the section "Recode Design Factors" on page 629.

Note that you can simply rename the block variable by 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 in addition to 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 the section "Replication" on page 681. For illustrations of the difference between the DESIGNREP= and POINTREP= options, see Example 7.6 and Example 7.7.

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 in addition to 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 the section "Replication" on page 681. For illustrations of the difference between the DESIGNREP= and POINTREP= options, see Example 7.6 and Example 7.7.

Randomize Design

RANDOMIZE < (*u*) > < NOVALRAN >

randomizes the design. See the section "Randomization" on page 679 for details. The following *options* are available:

(*u*)

specifies an integer used to start the pseudo-random number generator for randomizing the design. The value of *u* must be enclosed in parentheses immediately after the keyword RANDOMIZE. If you do not specify a seed, or if you specify a value less than or equal to zero, by default the seed is generated from reading the time of day from the computer's clock.

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 7.8 for an example. See the section "Structure of General Factorial Designs" on page 687 for information about 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 the section "Recode Design Factors" on page 629 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 2-level factors (A1 and A2) into one 4-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.) 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 Example 7.1.

FRACTION=*h*
specifies the fraction of one full replication of all possible combinations of the factors. For instance, FRACTION=2 specifies a half-fraction, 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 7.4.

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 about run-indexing factors, see "Types of Factors" on page 676 and "Structure of General Factorial Designs" on page 687.

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).

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 7.1: Completely Randomized Design

NOTE: See *A Completely Randomized Design* 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=SurfaceExperiment randomize(713)
      Speed nvals=(100 125 150 175 200);
run;
proc print data=SurfaceExperiment;
run;
```

The RANDOMIZE option in the OUTPUT statement randomizes the run order; the random seed, 713 here, is optional. The design saved in the data set SurfaceExperiment is displayed in Output 7.1.1.

Output 7.1.1 A Completely Randomized Design

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

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 632.

Example 7.2: Resolution 4 Augmented Design

NOTE: See *Resolution IV Augmented Design* in the SAS/QC Sample Library.

Box, Hunter, and Hunter (1978) describe an injection molding experiment that involves eight 2-level factors: mold temperature (Temp), moisture content (Moisture), holding pressure (HoldPress), cavity thickness (Thick), booster pressure (BoostPress), 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 7.2.1, with the following statements:

```
proc factex;
   factors Temp        Moisture HoldPress Thick
           BoostPress  Time     Speed     Gate;
   size design=16;
   model resolution=4;
   examine design aliasing;
run;
```

The FACTORS statement lists factor names. The option DESIGN=16 of the SIZE statement specifies the design size. The RESOLUTION=4 specifies the resolution of the design. The EXAMINE statement lists points and aliasing.

Output 7.2.1 A 2^{8-4}_{IV} Design

The FACTEX Procedure

Design Points								
Experiment Number	Temp	Moisture	HoldPress	Thick	BoostPress	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 7.2.2.

Output 7.2.2 Alias Structure for a 2^{8-4}_{IV} Design

Aliasing Structure
Temp
Moisture
HoldPress
Thick
BoostPress
Time
Speed
Gate
Temp*Moisture = HoldPress*Gate = Thick*Speed = BoostPress*Time
Temp*HoldPress = Moisture*Gate = Thick*Time = BoostPress*Speed
Temp*Thick = Moisture*Speed = HoldPress*Time = BoostPress*Gate
Temp*BoostPress = Moisture*Time = HoldPress*Speed = Thick*Gate
Temp*Time = Moisture*BoostPress = HoldPress*Thick = Speed*Gate
Temp*Speed = Moisture*Thick = HoldPress*BoostPress = Time*Gate
Temp*Gate = Moisture*HoldPress = Thick*BoostPress = Time*Speed

Subsequent analysis of the data collected for the design suggests that HoldPress and BoostPress have statistically significant effects. There also seems to be significant effect associated with the sum of the aliased two-factor interactions Temp*BoostPress, Moisture*Time, HoldPress*Speed, and Thick*Gate. This chain of confounded interactions is identified in Output 7.2.2.

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 these four effects are estimable, regardless of all other main effects and interactions. For example, the following statements generate a suitable set of runs:

```
proc factex nocheck;
   factors Temp        Moisture HoldPress Thick
           BoostPress  Time     Speed     Gate;
   model estimate=(Temp*BoostPress
                   Moisture*Time
                   HoldPress*Speed
                   Thick*Gate);
   size design=min;
   examine design aliasing(2);
run;
```

The DESIGN=MIN option directs FACTEX to search for the smallest design that allows all four interactions to be estimated. Eight runs are required: see Output 7.2.3.

Output 7.2.3 Additional Runs to Resolve Ambiguities

The FACTEX Procedure

	Design Points							
Experiment Number	Temp	Moisture	HoldPress	Thick	BoostPress	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

Output 7.2.4 shows the alias structure of the additional eight runs. Note that the alias chain of interest

```
Temp*BoostPress=Moisture*Time=HoldPress*Speed=Thick*Gate
```

from the original design is broken: in this new set of runs, these four interactions are aliased with main effects and with other two-factor interactions, but they are unaliased with each other. Therefore, when these four runs are added to the original 16 runs, the main effects of the eight factors plus the four two-factor interactions that were originally aliased with each other can all be estimated with the 20 runs.

Output 7.2.4 Alias Structure of the Additional Experiment

Aliasing Structure
0 = Thick*BoostPress = Thick*Time = Thick*Speed = BoostPress*Time = BoostPress*Speed = Time*Speed
Temp = Thick*Gate = BoostPress*Gate = Time*Gate = Speed*Gate
Moisture = HoldPress*Gate
HoldPress = Moisture*Gate
Thick = BoostPress = Time = Speed = Temp*Gate
Gate = Temp*Thick = Temp*BoostPress = Temp*Time = Temp*Speed = Moisture*HoldPress
Temp*Moisture = HoldPress*Thick = HoldPress*BoostPress = HoldPress*Time = HoldPress*Speed
Temp*HoldPress = Moisture*Thick = Moisture*BoostPress = Moisture*Time = Moisture*Speed

Example 7.3: Factorial Design with Center Points

NOTE: See *A Factorial Design with Center Points* in the SAS/QC Sample Library.

Factorial designs that involve 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 by 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 2-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=Factorial
          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 CPDesign;
   set Factorial Center;
run;
```

```
proc print data=CPDesign;
run;
```

The design saved in the data set CPDesign is displayed in Output 7.3.1. Observations 1 to 8 are the factorial points, and observations 9 to 12 are the center points.

Output 7.3.1 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 7.4: Fold-Over Design

NOTE: See *A Fold-Over Design* 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 $r + 1$. [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.

Consider a $\frac{1}{8}$ 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;
   model resolution=3;
   examine aliasing;
   output out=Original;
run;

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

The option FRACTION=8 of the SIZE statement specifies a $\frac{1}{8}$ fraction of a complete factorial—that is, 8 ($=\frac{1}{8}2^6$). The design, which is saved in the data set Original, is displayed in Output 7.4.1.

[1] This is not true if the original design has even resolution.

Output 7.4.1 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 7.4.2 indicates that all the main effects are confounded with the two-factor interactions.

Output 7.4.2 Alias Structure for a 2^{6-3}_{III} Design

The FACTEX Procedure

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 by using the following DATA step:

```
data FoldOver;
   set Original;
   A=-A; B=-B; C=-C;
   D=-D; E=-E; F=-F;
run;
title 'Fold-Over Design';
proc print data=FoldOver;
run;
```

Here, the DATA step creates the fold-over fraction by reversing the signs of the values of the factors in the original fraction. The fold-over design is displayed in Output 7.4.3.

Output 7.4.3 A 2_{III}^{6-3} 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 7.5: Randomized Complete Block Design

NOTE: See *A Randomized Complete Block Design* 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 7.5.1:

```
proc factex;
   factors Block / nlev=4;
   output out=Blocks Block nvals=(1 2 3 4) randomize(12345);
run;
   factors Treatment / nlev=6;
   output out=RCBD
          designrep=Blocks
          randomize(54321)
          Treatment cvals=('A' 'B' 'C' 'D' 'E' 'F');
run;
quit;
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 7.5.1 A Randomized Complete Block Design

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

Example 7.6: Two-Level Design with Design Replication and Point Replication

NOTE: See *A Two-Level Design with Replication* 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 with 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 (Temperature), 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 Temperature Time Velocity;
   size fraction=2;
   model res=max;
   output out=Unreplicated;
run;
proc print data=Unreplicated;
run;
```

The design, saved in the data set Unreplicated), is shown in Output 7.6.1.

Output 7.6.1 Unreplicated Design

Obs	Pressure	Temperature	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 Temperature Time Velocity;
   size fraction=2;
   model res=max;
   output out=Replicated designrep=3;
run;
proc print data=Replicated;
run;
```

The design, saved in the data set Replicated, is displayed in Output 7.6.2.

Output 7.6.2 Design Replication

Obs	Pressure	Temperature	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 Temperature Time Velocity;
   size fraction=2;
   model res=max;
   output out=PointReplicated pointrep=3;
run;
proc print data=PointReplicated;
run;
```

The design, saved in the data set PointReplicated, is displayed in Output 7.6.3. 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 7.6.3 Point Replication

Obs	Pressure	Temperature	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 by using design replication (Output 7.6.2) and point replication (Output 7.6.3). 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 the section "Replication" on page 681 for more information on design replication.

Example 7.7: Mixed-Level Design Using Design Replication and Point Replication

NOTE: See *A Mixed-Level Design Using Replication* 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 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 crossproduct 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 7.14.) All estimable effects in each of the component

designs, in addition to all generalized interactions between estimable effects in different component designs, are estimable in the crossproduct; refer to Section 3 of Chakravarti (1956).

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

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

```
proc factex;
   factors A B;
   output out=ab;
run;
   factors C / nlev=3;
   output out=DesignReplicated designrep=ab;
run;
proc print data=DesignReplicated;
run;
```

Output 7.7.1 lists the mixed-level design saved in the data set DesignReplicated.

Output 7.7.1 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 by using the point replication feature of the FACTEX procedure. The following SAS statements produce a complete $2^2 \times 3$ factorial design by using point replication:

```
proc factex;
   factors A B;
   output out=ab;
run;
   factors C / nlev=3;
   output out=PointReplicated pointrep=ab;
run;
proc print data=PointReplicated;
run;
```

Output 7.7.2 lists the mixed-level design saved in the data set PointReplicated.

Output 7.7.2 $2^2 \times 3$ 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 7.7.1 and Output 7.7.2. In design replication, the mixed-level design is given by AB ⊗ C, while for point replication the mixed-level design is given by C ⊗ AB, where ⊗ 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 7.8: Mixed-Level Design Using Pseudo-Factors

NOTE: See *Mixed-Level Designs Using Pseudo-Factors* 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 by 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 the section "Types of Factors" on page 676 for details.

For example, the following statements create a design for one 4-level factor (A) and three 2-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');
run;
proc print;
   var A B C D;
run;
```

The levels of two 2-level pseudo-factors (A1 and A2) are used to represent the four levels of A. Hence the three degrees of freedom associated with A are 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, in addition to 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 4-level factor A. The data set DesignA is listed in Output 7.8.1.

Output 7.8.1 4×2^3 Design of Resolution 4 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 7.9: Mixed-Level Design by Collapsing Factors

NOTE: See *Mixed-Level Design with Collapsing Factors* 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 2-level factors (x1 and x2) and two 3-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=MixedLevel x1 nvals=(-1 1 -1)
                         x2 nvals=(-1 1 -1);
run;
proc print data=MixedLevel;
run;
```

The mixed-level design is a three-quarter fraction with resolution 5; refer to Margolin (1967). The design is displayed in Output 7.9.1.

Output 7.9.1 $2^2 \times 3^2$ 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 7.10: Hyper-Graeco-Latin Square Design

NOTE: See *Hyper-Graeco-Latin Square* 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 example, 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 4-level factors in 16 runs.

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

data _null_;
   array t{3} $ t1-t3;
   array s{4} $ s1-s4; /* Buffer for holding each row      */
   file print;         /* Direct printing to output screen */
   do square=1 to 3;
      put "Square " square ":";
      n = 1;
      do r=1 to 4;
         do c=1 to 4;
            set OrthArray point=n; n=n+1;
            s{c}=t{square};
         end;
         put "       " s1-s4;
      end;
      put;
   end;
   stop;
run;
```

In most cases, the form that appears in the output data set OrthArray is most useful. The form that usually appears in textbooks is displayed in Output 7.10.1, which can be produced by using a simple DATA step (not shown here).

Output 7.10.1 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 7.11: Resolution 4 Design with Minimum Aberration

NOTE: See *A Res IV Design with Minimum Aberration* in the SAS/QC Sample Library.

If a design has resolution 4, then you can simultaneously estimate all main effects and *some* two-factor interactions. However, not all resolution 4 designs are equivalent; you might be able to estimate more two-factor interactions with some than with others. Among all resolution 4 designs, a design with the maximum number of estimable two-factor interactions is said to have *minimum aberration*.

For example, if you use the FACTEX procedure to generate a resolution 4 2-level design in 32 runs for seven factors, you can 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 7.11.1:

```
proc factex;
   factors A B C D E F G;
   model resolution=4;
   size design=32;
   examine aliasing;
run;
```

Output 7.11.1 Alias Structure for Default 2_{IV}^{7-2} Design

The FACTEX Procedure

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 contrast, the resolution 4 design given in Table 12.15 of Box, Hunter, and Hunter (1978) is a minimum aberration design that permits 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 in the MODEL statement, as in the following statements:

```
proc factex;
   factors A B C D E F G;
   model resolution=4 / minabs;
   size design=32;
   examine aliasing;
run;
```

The alias structure for the resulting design is shown in Output 7.11.2.

Output 7.11.2 Alias Structure for Minimum Aberration 2_{IV}^{7-2} Design

The FACTEX Procedure

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_{III}^{10-6}, 2_{IV}^{7-2}, 2_{IV}^{8-3}, 2_{IV}^{9-4}, and 2_{V}^{10-3}), the MINABS option is required to construct a design with minimum aberration, and in two cases (2_{III}^{9-5}, 2_{IV}^{9-3}), the NOCHECK option is also required. If the FACTEX procedure is given sufficient time to run, specifying both the MINABS and the NOCHECK options always results in a minimum aberration design. However, with the default search time of 60 seconds, there are three cases (2_{IV}^{10-5}, 2_{IV}^{10-4}, and 2_{IV}^{11-5}) for which the FACTEX procedure is unable to find the minimum aberration design, even with both the MINABS and NOCHECK options specified.

Example 7.12: Replicated Blocked Design with Partial Confounding

NOTE: See *Replicated Blocked Design with Confounding* 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 replicates 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 7.12.1. The highest-order interaction A*B*C is confounded with the block effect. The design, with recoded block levels, is saved in a data set named REP1.

Output 7.12.1 Confounding Rule and Alias Structure for Replicate 1

The FACTEX Procedure

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 7.12.2. 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 7.12.2 Confounding Rule and Alias Structure for Replicate 2

The FACTEX Procedure

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 7.12.3. The interaction B*C is free of aliases, but the interaction A*C is confounded with the block effect.

Output 7.12.3 Confounding Rule and Alias Structure for Replicate 3

The FACTEX Procedure

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 7.12.4.

Output 7.12.4 Confounding Rule and Alias Structure for Replicate 4

The FACTEX Procedure

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 7.12.5.

Output 7.12.5 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 7.13: Incomplete Block Design

NOTE: See *Incomplete Block Design* 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 of balanced lattices of Yates (1936); refer to page 396 of Cochran and Cox (1957). In a balanced lattice, 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 4-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 by 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 that have the Repth factor equal to Block–1. These run numbers are the treatments that go into the particular block.

The design is printed by 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;
   file print;
   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 ARRAY statement creates a buffer for holding each block, and the FILE statement directs the printing to output screen. The design is displayed in Output 7.13.1.

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 / noprint;
   output data=b out=c;
run;
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 7.13.2.

Output 7.13.1 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 7.13.2 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 7.14: Design with Inner Array and Outer Array

NOTE: See *A Problem In Quality Improvement* 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 crossproduct 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:

- interference (Interference), defined as the difference between the outer width of the tubing and the inner width of the connector
- connector wall thickness (ConnectorWall)
- depth of insertion (InsertDepth) of the tubing into the connector
- amount of adhesive (Glue) in the connector pre-dip

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

- amount of time (Time) allowed for assembly
- temperature (Temperature))
- 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 Temperature Humidity;
   output out=OuterArray Time        nvals=(  24  120)
                         Temperature nvals=(  72  150)
                         Humidity    nvals=(0.25 0.75);
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 Design.

```
proc factex;
   factors Interference ConnectorWall InsertDepth Glue /
      nlev=3;
   size design=9;
   model resolution=3;
   output out=Design pointrep=OuterArray
        Interference  cvals=('Low'    'Medium' 'High'  )
        ConnectorWall cvals=('Thin'   'Medium' 'Thick' )
        InsertDepth   cvals=('Shallow' 'Deep' 'Medium')
        Glue          cvals=('Low'    'High'   'Medium');
run;
```

The final design is listed in Output 7.14.1. Main effects of each factor can be estimated free of each other but are confounded with two-factor interactions.

Output 7.14.1 Design for Control Factor and Noise Factors

Obs	Interference	ConnectorWall	InsertDepth	Glue	Time	Temperature	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
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

Output 7.14.1 *continued*

Obs	Interference	ConnectorWall	InsertDepth	Glue	Time	Temperature	Humidity
49	High	Thin	Deep	Medium	24	72	0.25
50	High	Thin	Deep	Medium	24	72	0.75

Obs	Interference	ConnectorWall	InsertDepth	Glue	Time	Temperature	Humidity
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 InsertDepth 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, the 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 in this example; 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 interesting interactions exist 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 remain confounded with one another, but they are 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 Interference ConnectorWall InsertDepth 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 3-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 Interference ConnectorWall InsertDepth 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 7.15: Fractional Factorial Split-Plot Designs

NOTE: See *Fractional Factorial Split-Plot Design* in the SAS/QC Sample Library.

In split-plot designs, not all factor levels can change from plot to plot. In the simplest split-plot structure, runs are grouped into whole plots; certain factors are applied to all plots in the whole plot, while others are applied to individual plots within a whole plot. The two types of factors are termed *whole-plot factors* and *subplot factors*, respectively. Split-plot designs are very common in chemical and process industries, where factors of interest are often applied at different stages of the production process and the final measurements of interest are made on the finished product. In this case, the different stages of production might give rise to multiple whole-plot effects.

Suppose you are designing an experiment to measure six factors that affect characteristics of metal wires sheathed with a certain material. Three of the factors (W1, W2, W3) apply to how the wires themselves are made, and the other three (S1, S2, S3) apply to the sheathing material. You propose to first prepare eight different batches of wire, making two wires from each batch, and then to prepare the sheathing material for each wire individually. This describes a standard split-plot experiment with batches of wires forming whole plots and sheathed wires forming subplots. The following code constructs a resolution 4 design for this experiment, specifying the Wire unit effect in the BLOCKS statement, and then in the UNITEFFECT statement specifying that W1, W2, and W3 should be constant within Wire and that S1, S2, and S3 should change within Wire. The resulting design is printed, sorted by Wire.

```
proc factex;
   factors W1 W2 W3
           S1 S2 S3;
   size design=16;
   blocks units=(Wire=8);
   model r=4;
   uniteffect Wire / whole=(W1 W2 W3)
                     sub  =(S1 S2 S3);
   examine aliasing(units);
   output out=WireExperiment1;
run;

proc sort data=WireExperiment1;
   by Wire W1-W3 S1-S3;
run;
proc print data=WireExperiment1;
run;
```

Output 7.15.1 shows the aliasing structure for the design, which indicates that the main effects of the wire factors are indeed estimated on the Wire whole plots and the main effects of the sheath factors are estimated on the subplots. Interestingly, some of the sheath factor interactions are also confounded with whole plots.

Output 7.15.1 A Split-Plot Design

The FACTEX Procedure

Aliasing Structure	
Units	
Wire	W1
Wire	W2
Wire	W3
Wire	W1*W2 = S1*S2
Wire	W1*W3 = S1*S3
Wire	W2*W3 = S2*S3
Residual	S1
Residual	S2
Residual	S3
Residual	W1*S1 = W2*S2 = W3*S3
Residual	W1*S2 = W2*S1
Residual	W1*S3 = W3*S1
Residual	W2*S3 = W3*S2

The final design is listed in Output 7.15.2. Notice that the factors W1, W2, and W3 are constant within Wire, while S1, S2, and S3 change within Wire.

Output 7.15.2 A Split-Plot Design

Obs	_1_	_2_	_3_	_4_	W1	W2	W3	S1	S2	S3	Wire
1	-1	-1	-1	1	-1	1	1	-1	1	1	1
2	-1	-1	-1	-1	-1	1	1	1	-1	-1	1
3	-1	-1	1	-1	1	-1	-1	-1	1	1	2
4	-1	-1	1	1	1	-1	-1	1	-1	-1	2
5	-1	1	-1	-1	1	-1	1	-1	1	-1	3
6	-1	1	-1	1	1	-1	1	1	-1	1	3
7	-1	1	1	1	-1	1	-1	-1	1	-1	4
8	-1	1	1	-1	-1	1	-1	1	-1	1	4
9	1	-1	-1	-1	1	1	-1	-1	-1	1	5
10	1	-1	-1	1	1	1	-1	1	1	-1	5
11	1	-1	1	1	-1	-1	1	-1	-1	1	6
12	1	-1	1	-1	-1	-1	1	1	1	-1	6
13	1	1	-1	1	-1	-1	-1	-1	-1	-1	7
14	1	1	-1	-1	-1	-1	-1	1	1	1	7
15	1	1	1	-1	1	1	1	-1	-1	-1	8
16	1	1	1	1	1	1	1	1	1	1	8

To see why the Wire factors are constant within wire and the sheath factors change, examine the confounding rules for the design. The following statements produce the table of confounding rules listed in Output 7.15.3:

```
proc factex;
   factors W1 W2 W3
           S1 S2 S3;
   size design=16;
```

```
   blocks units=(Wire=8);
   model r=4;
   uniteffect Wire / whole=(W1 W2 W3)
                     sub  =(S1 S2 S3);
   examine confounding;
run;
```

Output 7.15.3 Split-Plot Confounding Rules

The FACTEX Procedure

Factor Confounding Rules
W1 = [1]*[2]*[3]
W2 = [2]*[3]
W3 = [1]*[3]
S1 = [1]*[2]*[3]*[4]
S2 = [2]*[3]*[4]
S3 = [1]*[3]*[4]

The terms [i] on the right-hand side of these rules denote plot-indexing pseudo-factors, as discussed in the section "Split-Plot Designs" on page 686. Note that the wire factors W1, W2, and W3 are confounded only with interactions between the first three pseudo-factors, the ones identified with the eight levels of the Wire unitfactor. This guarantees that these factors are constant within levels of Wire. By contrast, the confounding rules for the sheath factors S1, S2, and S3 each involve the fourth pseudo-factor, so they must change within levels of Wire.

There are only eight different combinations of the sheath factors, but the previous design requires you to produce batches of sheath material 16 times, once for each of the two wires to be made from each wire batch. If instead you propose to make just four batches of sheath material and apply part of each batch to parts of different batches of wires, the design becomes a row-column design instead of a split-plot design. Furthermore, suppose that the number of batches rather than the size of each batch is the main cost, so that you can prepare eight batches of wire and four batches of sheathing material in sufficient quantity to make 64 different sheathed wires. Since there can be only four different combinations of the three sheathing factors, each sheathing factor interaction is aliased with a main effect, and thus the design necessarily has resolution 3. All other interactions are estimable free of main effects. The following FACTEX statements create the design and display the two unit effects with their respective wholeunit factor levels:

```
proc factex;
   factors W1 W2 W3
           S1 S2 S3;
   size design=64;
   blocks units=(Wire=8 Sheath=4);
   model r=3;
   uniteffect Wire   / whole=(W1 W2 W3);
   uniteffect Sheath / whole=(S1 S2 S3);
   examine aliasing(units);
   output out=WireExperiment2;
proc freq data=WireExperiment2;
   table Wire  *W1*W2*W3 / list nocum nopct;
   table Sheath*S1*S2*S3 / list nocum nopct;
run;
```

Output 7.15.4 A Split-Lot Design: Wire Units

The FREQ Procedure

Wire	W1	W2	W3	Frequency
1	-1	1	1	8
2	1	-1	-1	8
3	1	-1	1	8
4	-1	1	-1	8
5	1	1	-1	8
6	-1	-1	1	8
7	-1	-1	-1	8
8	1	1	1	8

Output 7.15.5 A Split-Lot Design: Sheath Units

The FREQ Procedure

Sheath	S1	S2	S3	Frequency
1	1	-1	-1	16
2	-1	1	-1	16
3	-1	-1	1	16
4	1	1	1	16

The results, listed in Output 7.15.4 and Output 7.15.5, indicate that W1, W2, and W3 are constant within Wire and S1, S2, and S3 are constant within Sheath.

Example 7.16: A Design for a Three-Step Process

NOTE: See *A Design for a Three-Step Process* in the SAS/QC Sample Library.

Ramirez and Weisz (2009) discuss an experiment on a multistep milling process with 16 processing factors, with a single factor applied at the first stage, seven more factors at the second stage, and eight more at the final stage. The experiment involves eight first-stage runs, eight second-stage runs within each of those, and again, two to four third-stage runs within each of those, for a total of 128 to 256 total experimental units. This example explores several different ways to design this experiment, depending on what kinds of effects are most important.

The following statements request a design of maximum resolution for this split-plot structure. The factors are listed in macro variables, for ease in specifying them in UNITEFFECT statements. The BLOCK statement defines the unit factors for the first two processing stages, with eight runs of each. The two UNITEFFECT statements then use these unit factors to specify which unit effects correspond to which factors. Finally, the EXAMINE statement requests that the aliasing structure and the overall modeling summary be displayed to see how many effects of different orders are estimable and clear. The UNITS suboption of the ALIASING option includes the unit effect confounding for each alias string in the alias structure.

```
%let F1 = Z;
%let F2 = A B C D E F G;
%let F3 = P Q R S T U V W;
proc factex;
   factors &F1 &F2 &F3;
   model r=max;
   size design=128;
   blocks units=(Step1=8 Step2=8);
   uniteffect Step1       / whole=(&F1) sub=(&F2 &F3);
   uniteffect Step1*Step2 / whole=(&F2) sub=(    &F3);
   examine aliasing(units) summary;
quit;
```

Output 7.16.1 Aliasing for Default 128-Run Three-Step Design

The FACTEX Procedure

Aliasing Structure	
Units	
Step1	Z
Step1	A*B = C*D = E*F = P*Q = R*S = T*U = V*W
Step1	A*C = B*D = E*G = P*R = Q*S = T*V = U*W
Step1	A*D = B*C = F*G = P*S = Q*R = T*W = U*V
Step1*Step2	A
Step1*Step2	B
Step1*Step2	C
Step1*Step2	D
Step1*Step2	E
Step1*Step2	F
Step1*Step2	G
Step1*Step2	Z*A
Step1*Step2	Z*B
Step1*Step2	Z*C
Step1*Step2	Z*D
Step1*Step2	Z*E
Step1*Step2	Z*F
Step1*Step2	Z*G
Step1*Step2	A*E = B*F = C*G = P*T = Q*U = R*V = S*W
Step1*Step2	A*F = B*E = D*G = P*U = Q*T = R*W = S*V
Step1*Step2	A*G = C*E = D*F = P*V = Q*W = R*T = S*U
Step1*Step2	B*G = C*F = D*E = P*W = Q*V = R*U = S*T
Residual	P
Residual	Q
Residual	R
Residual	S
Residual	T
Residual	U
Residual	V
Residual	W
Residual	Z*P
Residual	Z*Q
Residual	Z*R
Residual	Z*S
Residual	Z*T
Residual	Z*U
Residual	Z*V
Residual	Z*W
Residual	A*P = B*Q = C*R = D*S = E*T = F*U = G*V
Residual	A*Q = B*P = C*S = D*R = E*U = F*T = G*W
Residual	A*R = B*S = C*P = D*Q = E*V = F*W = G*T
Residual	A*S = B*R = C*Q = D*P = E*W = F*V = G*U
Residual	A*T = B*U = C*V = D*W = E*P = F*Q = G*R
Residual	A*U = B*T = C*W = D*V = E*Q = F*P = G*S

Output 7.16.1 *continued*

The FACTEX Procedure

Aliasing Structure	
Units	
Residual	A*V = B*W = C*T = D*U = E*R = F*S = G*P
Residual	A*W = B*V = C*U = D*T = E*S = F*R = G*Q

Output 7.16.2 Modeling Summary for Default 128-Run Three-Step Design

Modeling Summary		
	Effects	
	Main	2FI
Total	16	120
Estimable	16	30
Clear	16	15

The resulting design has resolution 4, which means that main effects are clear of two-factor interactions but interactions are aliased with each other. Output 7.16.1 shows which interactions are aliased and also shows which units are used to estimate them. Note that several interactions between Step2 and Step3 factors are estimated with Step2 units.

As Output 7.16.2 shows, only 30/120=25% of the two-factor interactions (2FI) are estimable and only 15/120=13% of them are clear. If simply protecting the main-effects estimates against potential two-factor interactions is sufficient, then this design suffices. However, if you want to estimate as many of the two-factor interactions as possible, then you should look for a MaxClear design. The following statements use the MAXCLEAR option in the MODEL statement to request a MaxClear design, and they also use the ORDER=RANDOM(RESTART) option in the PROC FACTEX statement to improve the chances that the best design is found. For more details about MaxClear designs, see the section "MaxClear Designs" on page 685.

```
%let F1 = Z;
%let F2 = A B C D E F G;
%let F3 = P Q R S T U V W;
proc factex order=random(restart seed=1);
   factors &F1 &F2 &F3;
   model r=max / maxclear;
   size design=128;
   blocks units=(Step1=8 Step2=8);
   uniteffect Step1       / whole=(&F1) sub=(&F2 &F3);
   uniteffect Step1*Step2 / whole=(&F2) sub=(    &F3);
   examine summary;
quit;
```

Output 7.16.3 Modeling Summary for MaxClear 128-Run Three-Step Design

The FACTEX Procedure

Modeling Summary		
	Effects	
	Main	2FI
Total	16	120
Estimable	16	87
Clear	16	69

The modeling summary results for the MaxClear design are shown in Output 7.16.3. Now 87/120=73% of the 2FI are estimable and 69/120=58% of them clear. This is a great improvement over the default design, but if complete estimability of all two-factor interactions is required, more than 128 runs are necessary. The following statements construct a design in 256 runs, effectively doubling the number of third-stage runs from two to four:

```
%let F1 = Z;
%let F2 = A B C D E F G;
%let F3 = P Q R S T U V W;
proc factex;
   factors &F1 &F2 &F3;
   model r=max;
   size design=256;
   blocks units=(Step1=8 Step2=8);
   uniteffect Step1       / whole=(&F1) sub=(&F2 &F3);
   uniteffect Step1*Step2 / whole=(&F2) sub=(    &F3);
   examine aliasing(units);
quit;
```

The aliasing structure (not shown) shows that the resulting design has resolution 5, which means that all main effects and two factor interactions are estimable free of each other. Even though the required 256 runs means this is a relatively large experiment, it is still only a tiny fraction of the 65,536 runs required for a complete factorial design.

Example 7.17: A Strip-Split-Split-Plot Design

NOTE: See *A Strip-Split-Split-Plot Design* in the SAS/QC Sample Library.

Suppose you are designing an experiment for a three-step process running on different machines. One way to model this is with a row×column strip-split-split-plot structure, with one type of unit, Machine, crossed with a process that has a split-split-plot structure. The following statements create a resolution 4 design in 11 factors for this situation, with one Machine factor MSetting and three, three, and five whole plot, split-plot, and split-split-plot process factors, respectively. The statements also specify that the design's aliasing structure and modeling summary be displayed, with the unit effect confounding for each alias string included in the alias structure.

```
%let FR = X11-X13;
%let FC = X21-X23;
%let FX = X31-X35;
proc factex;
   factors MSetting &FR &FC &FX;
   model r=4;
   blocks units=(Machine=2 Step1=8 Step2=4 Step3=2);
   uniteffect Machine           / whole=(MSetting);
   uniteffect Step1             / whole=(&FR) sub=(&FC &FX);
   uniteffect Step1*Step2       / whole=(&FC) sub=(    &FX);
   uniteffect Step1*Step2*Step3 / whole=(&FX);
   size design=128;
   examine aliasing(units) summary;
run;
```

The UNITEFFECT statements define a triply nested split-plot structure for the process on each machine, including the Step1*Step2*Step3 split-split units for the process, in order to ensure that process effects are crossed with Machine.

As Output 7.17.1 shows, 36/66=55% of the 2FI are estimable and 21/66=32% of them are clear. The aliasing structure (not shown) indicates that the main effect of MSetting is the only thing that is estimated with the Machine units; all interactions between MSetting and the process factors are estimated with the experimental units, labeled "Residual" in the alias structure.

Output 7.17.1 A Strip-Split-Split-Plot Design

The FACTEX Procedure

Modeling Summary		
	Effects	
	Main	2FI
Total	12	66
Estimable	12	36
Clear	12	21

If simply protecting the main-effects estimates against potential two-factor interactions is the reason for requiring a resolution 4 design, then the design of Output 7.17.1 suffices. However, if you want to estimate as many of the two-factor interactions as possible, then you should use the MAXCLEAR option in the MODEL statement to construct a MaxClear design, as shown in the following statements:

```
%let FR = X11-X13;
%let FC = X21-X23;
%let FX = X31-X35;
proc factex order=random(restart seed=230501);
   factors MSetting &FR &FC &FX;
   model r=4 / maxclear;
   blocks units=(Machine=2 Step1=8 Step2=4 Step3=2);
   uniteffect Machine           / whole=(MSetting);
   uniteffect Step1             / whole=(&FR) sub=(&FC &FX);
   uniteffect Step1*Step2       / whole=(&FC) sub=(    &FX);
   uniteffect Step1*Step2*Step3 / whole=(&FX);
   size design=128;
   examine summary;
run;
```

As Output 7.17.2 shows, now 55/66=83% of the 2FI are estimable and 45/66=68% of them are clear—more than twice as many clear interactions as before.

Output 7.17.2 A Strip-Split-Split-Plot Design

The FACTEX Procedure

Modeling Summary		
	Effects	
	Main	2FI
Total	12	66
Estimable	12	55
Clear	12	45

For details about MaxClear designs, see the section "MaxClear Designs" on page 685.

Example 7.18: Design and Analysis of a Complete Factorial Experiment

NOTE: See *Complete Factorial Experiment* 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 has four factors: cathode power (Power), gas flow (Flow), reactor chamber pressure (Pressure), and electrode gap (Gap). A single replicate of a 2^4 design is run, and the etch rate (Rate) is 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=EtcherDesign
          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 EtcherDesign. The experiment that uses the 16-run design is performed, and the etch rate is measured. The following DATA step updates the data set EtcherDesign with the values of Rate:

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

title 'Nitride Etch Process Experiment';
proc print;
run;
```

The data set DESGNDAT is listed in Output 7.18.1.

Output 7.18.1 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=EtcherDesign;
   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 by 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 7.18.2. 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 7.18.2 Analysis of Variance for the Nitride Etch Process Experiment

Nitride Etch Process Experiment

The GLM Procedure

Dependent Variable: Rate

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 2-level factors is restricted to no more than two blocks. Instead, the values of several 2-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 2-level factors (P_1, P_2, and P_3) can be used to index the values of an 8-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 7.7 summarizes the different types of factors discussed in this section.

Table 7.7 Types of Factors

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*[2] 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:

[2]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.

- 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 might 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 are available, you should use a resolution 5 design so that all main effects and two-factor interactions are 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 Example 7.2 for an example.

Many 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 2-level factors in 16 runs that permits estimation of all main effects and two-factor interactions. This chapter uses Arabic numerals for resolution since these correspond directly to what you specify 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 might 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 2-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:

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

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 two steps:

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

   $$\begin{aligned}\text{Block 1 runs: } \{1,2,3,4\} &\rightarrow \{4,1,2,3\}\\ \text{Block 2 runs: } \{5,6,7,8\} &\rightarrow \{8,7,6,5\}\end{aligned}$$

2. Randomly permute the order of the blocks:

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

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 7.7.

Figure 7.7 Four-Run Design Replicated Using the POINTREP= Option

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

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 7.8.

Figure 7.8 Four-Run Design Replicated Using the DESIGNREP= Option

	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

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 might consist of all possible different settings for the associated factors, or they might 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 7.14 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 676 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 2-level factors, the rules are displayed in a multiplicative notation that uses 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 always has 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 3-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 `[B i]` on the left-hand side.

For details about how confounding rules are constructed, see "Suitable Confounding Rules" on page 688.

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 because 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,

```
Temperature*Moisture=HoldPress*Gage=Thickness*Screw=BoostPress*Time
```

is an alias chain that shows the relationship between four 2-factor interactions. If you want separate estimates of Temperature*Moisture and Thickness*Screw, for example, 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 ALIASING option, or you can use the default calculated by the procedure. Alias chains that are confounded with blocks are displayed with [B] on the left-hand side.

Minimum Aberration

As discussed in the section "Speeding Up the Search" on page 691, 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 might 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 2-level factors in 32 runs (2^{7-2}_{IV}) discussed in Example 7.11.

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 by 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 *i*th-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 in the section "EXAMINE Statement" on page 625). You can set d to any level by specifying (d) immediately after the MINABS option.

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 *i*th-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

MaxClear Designs

As discussed in the section "Alias Structure" on page 684, the alias structure for a factorial design can tell you important information about which effects are confounded and hence cannot be estimated separately from one another. In some cases, you cannot avoid the fact that some potentially active effects are aliased; for example, in resolution 4 designs, some two-factor interactions are aliased with each other and hence cannot be jointly estimated. In this case, you might want a design that has as many two-factor interactions as possible unaliased with any other interaction—that is, as many *clear* two-factor interactions as possible. This is known as the *MaxClear* design, and you can use the MAXCLEAR option in the MODEL statement to request it.

To explore how well a given design performs on the MaxClear criterion, you can use the ALIASING option in the EXAMINE statement to examine the alias structure. Clear interactions are those that are displayed by themselves, with no other interactions in their alias chain. Alternatively, the SUMMARY option in the EXAMINE statement displays a summary count of how many interactions there are in total up to a certain order d, how many of those are unaliased with interactions of lower order and are thus in a sense estimable, and how many are unaliased with any interactions of order d or lower and are thus clear.

Obviously, whether an interaction is clear depends on what other effects are considered to be potentially of interest. For a given design, the default order d for considering interaction clarity is the same as the

default order d of interactions included in the alias structure. As with the alias structure, you can specify an alternative value of d in the MAXCLEAR option in the MODEL statement or in the SUMMARY option in the EXAMINE statement.

Split-Plot Designs

As discussed in the section "Structure of General Factorial Designs" on page 687, for a design with q-level factors in q^m runs, the FACTEX procedure usually treats the first m factors of the design as the run-indexing factors, and computes the levels of all other factors as linear combinations of these over the Galois field of order q. However, when you restrict the design's randomization by using the BLOCKS UNITS=() option and UNITEFFECT statements to specify unitfactors and uniteffects, PROC FACTEX instead computes the levels of all factors (including the first m) in terms of underlying plot-indexing pseudo-factors that are distinct from those named in the FACTORS statement. These plot-indexing pseudo-factors are denoted [i], for i=1, ..., m, and they are associated with unitfactors as follows: If the BLOCK UNIT=() specification has the form

block units=(Stage1=n_1 **Stage2=**$n_2 \ldots$**);**

where $n_1 = q^{k_1}$, $n_2 = q^{k_2}$, ..., then the first unitfactor, Stage1, is identified with all possible interactions between the first k_1 plot-indexing pseudo-factors, the second with the next k_2 pseudo-factors, and so on. If you save a split-plot design to a data set by using the OUTPUT statement, then the plot-indexing pseudo-factors are also included in the data set with names _1_, _2_, ..., up to the base-q logarithm of the number of runs.

The whole-plot and subplot constraints specified by the UNITEFFECT statement define the relation between the plot-indexing pseudo-factors that correspond to the specified *uniteffect* and the factor effects specified in the WHOLE=() and SUB=() options. In particular, with a BLOCK UNIT=() specification of the previous form, a UNITEFFECT statement of the form

uniteffect Stage1 / whole=(*Stage-1-effects***);**

means that the *Stage-1-effects* should be aliased only with interactions between the first k_1 plot-indexing pseudo-factors, while

uniteffect Stage1*Stage2 / sub=(*Stage-2-effects***);**

means that the *Stage-2-effects* should not be aliased with interactions between the first $k_1 + k_2$ plot-indexing pseudo-factors.

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 following:

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

The DESIGN option in the EXAMINE statement displays the coded runs in the design that uses standard values, as described in the section "OUTPUT Statement" on page 629. The CONFOUNDING option in this

EXAMINE statement displays the confounding rules used to construct the design. The ALIAS option in this statement displays the aliasing 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 7.8 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

Theory of Orthogonal Designs

Overview

This chapter provides the mathematical and statistical background for designs that are constructed by the FACTEX procedure; it also outlines the search algorithm that is used to find suitable construction rules. The material in this chapter is general and theoretical; you do not need to read this chapter to use the procedure for constructing most common experimental designs. On the other hand, you might want to read this chapter for the following reasons:

- to understand the general structure of designs that can be constructed with the FACTEX procedure
- to construct designs for factors with more than two levels, especially if interactions are involved
- to improve the search used by the procedure when constructing complicated designs involving many factors

Structure of General Factorial Designs

The FACTEX procedure constructs a fractional design for q-level factors by using the *Galois field* (or *finite field*) of size q. This is a system with q elements and two operations + and $\times$, which satisfy the usual mathematical axioms for addition and multiplication. When q is a prime number, finite field arithmetic is

equivalent to regular integer arithmetic modulo q. When $q = 2$, addition of the two elements of the finite field is equivalent to multiplication of the integers +1 and –1. Since designs for factors with levels +1 and –1 are the factorial designs most commonly covered in textbooks, the arithmetic for fractional factorial designs is usually shown in multiplicative form. However, throughout this section a more general notation is used.

A design for q-level factors in q^m runs constructed by the FACTEX procedure has the following general form. The first m factors are taken to index the runs in the design, with one run for each different combination of the levels of these factors, where the levels run from 0 to $q - 1$. These factors are called *run-indexing factors*. For a particular run, the value F of any other factor in the design is derived from the levels $P_1, P_2, \ldots, P_m$ of the run-indexing factors by means of *confounding rules*. These rules are of the general form

$$F = r_1 P_1 + r_2 P_2 + \ldots + r_m P_m$$

where all the arithmetic is performed in the finite field of size q. The linear combination on the right-hand side of the preceding equation is called a *generalized interaction* between the run-indexing factors. A generalized interaction is part of the statistical interaction between the factors with nonzero coefficients in the linear combination. The factor F is said to be *confounded* or *aliased* with this generalized interaction; two terms are confounded when the levels they take in the design yield identical partitions of the runs, so that their effects cannot be distinguished. The confounding rules characterize the design, and the problem of constructing the design reduces to finding suitable confounding rules.

Suitable Confounding Rules

Design Factors

This section explains how the criteria for a design can be reduced to prescribing that certain generalized interactions are *not* to be "confounded with zero."

Suitable confounding rules depend on the effects you want to estimate with the design. For example, if you want to estimate the main effects of both A and B, the following rule is inappropriate:

$$A = B$$

With this rule, the levels of A and B are the same in every run of the design, and the main effects of the two factors cannot be estimated independently of one another. Thus, the first criterion for a suitable confounding rule is that no two effects you want to estimate should be confounded with each other.

Furthermore, an effect you want to estimate should not be confounded with an effect that is nonnegligible. For example, if the interaction between C and D is nonnegligible and you want to estimate the main effect of A, the following confounding rule is inappropriate:

$$A = C + D$$

(Recall that this section uses a general linear form for confounding rules instead of the usual multiplicative form. For factors with levels +1 and –1, the preceding rule is equivalent to $A = C * D$.)

Another kind of confounding involves *confounding with zero*. If a factor or a generalized interaction F has the same value in every run of the design, then F is *confounded with zero*. Such confounding is denoted as

$$0 = F$$

Interactions are estimable with the design if and only if they are not confounded with zero. Consequently, another criterion for a suitable confounding rule is that no effect that you want to estimate can be confounded with zero. The confounding rule for two main effects

$$A = B$$

can be written as a generalized interaction confounded with zero:

$$0 = -A + B$$

The right-hand side of the preceding equation is part of the interaction between A and B. Thus, for any two effects to be unconfounded, it is equivalent to prescribe that no part of their generalized interaction be confounded with zero.

It is not enough to make sure that only the confounding rules themselves satisfy these restrictions. The consequences of the confounding rules must also satisfy the restrictions. For example, suppose you want to make sure that main effects are not confounded with two-factor interactions, and suppose that the confounding rule for factor E is

$$E = A + B + C + D$$

Then the following rule cannot be used for factor F:

$$F = A + B + C$$

Even though the rule for F does not confound F with a two-factor interaction, this rule forces a generalized interaction between E and F to be aliased with the main effect of D, since

$$E - F = (A + B + C + D) - (A + B + C) = D$$

Block Factors

If your design involves blocks, additional confounding criteria need to be considered. Blocks are introduced into designs by means of *block pseudo-factors*. (See "Types of Factors" on page 676 for details.) A design for q-level factors in q^s blocks contains s block pseudo-factors. Denoting the levels of these factors for any given run by $B_1, B_2, \ldots, B_s$, the index of the block in which the run occurs is given by

$$B_1 + qB_2 + q^2B_3 + \ldots + q^{s-1}B_s$$

For each block to occur in the design, every possible combination of block pseudo-factors must occur. This can happen only if all main effects and interactions between the block factors are estimable, which leads to yet another criterion for the confounding rules. Moreover, the effects you want to estimate cannot be confounded with blocks. In general:

- no generalized block pseudo-factors can be confounded with zero
- no generalized interactions between block pseudo-factors and effects you want to estimate can be confounded with zero

General Criteria

The criteria for an orthogonally confounded q^k design reduce to requiring that no generalized interactions in a certain set $\mathcal{M}$ can be confounded with zero. (See the section "Structure of General Factorial Designs" on page 687 for a definition of *generalized interaction*.) This section presents the general definition of $\mathcal{M}$. First, define three sets, as follows:

$\mathcal{E}$	the set of effects that you want to estimate
$\mathcal{N}$	the set of effects that you do not want to estimate but that have unknown nonzero magnitudes (referred to as *nonnegligible* effects)
$\mathcal{B}$	the set of all generalized interactions between block pseudo-factors

Furthermore, for any two sets of effects $\mathcal{A}$ and $\mathcal{B}$, denote by $\mathcal{A} \times \mathcal{B}$ the set of all generalized interactions between the effects in $\mathcal{A}$ and the effects in $\mathcal{B}$.

Then the general rules for creating the set of effects $\mathcal{M}$ that are not to be confounded with zero are as follows:

- Put $\mathcal{E}$ in $\mathcal{M}$. This ensures that all effects in $\mathcal{E}$ are estimable.
- Put $\mathcal{E} \times \mathcal{E}$ in $\mathcal{M}$. This ensures that all pairs of effects in $\mathcal{E}$ are unconfounded with each other.
- Put $\mathcal{E} \times \mathcal{N}$ in $\mathcal{M}$. This ensures that effects in $\mathcal{E}$ are unconfounded with effects in $\mathcal{N}$.
- Put $\mathcal{B}$ in $\mathcal{M}$. This ensures that all q^s blocks occur in the design.
- Put $\mathcal{E} \times \mathcal{B}$ in $\mathcal{M}$. This ensures that effects in $\mathcal{E}$ are unconfounded with blocks.

Searching for Confounding Rules

The goal in constructing a design, then, is to find confounding rules that do not confound with zero any of the effects in the set $\mathcal{M}$ defined previously. This section describes the sequential search performed by the FACTEX procedure to accomplish this goal.

First, construct the set C_1 of candidates for the first confounding rule, taking into account the set $\mathcal{M}$ of effects not to be confounded with zero. If C_1 is empty, then no design is possible; otherwise, choose one of the candidates $r_1 \in C_1$ for the first confounding rule and construct the set C_2 of candidates for the second confounding rule, taking both $\mathcal{M}$ and r_1 into account. If C_2 is empty, choose another candidate from C_1; otherwise, choose one of the candidates rules $r_2 \in C_2$ and go on to the third rule. The search continues either until it succeeds in finding a rule for every non-run-indexing factor or until the search fails because the set C_1 is exhausted.

The algorithm used by the FACTEX procedure to select confounding rules is essentially a depth-first tree search. Imagine a tree structure in which the branches connected to the root node correspond to the candidates

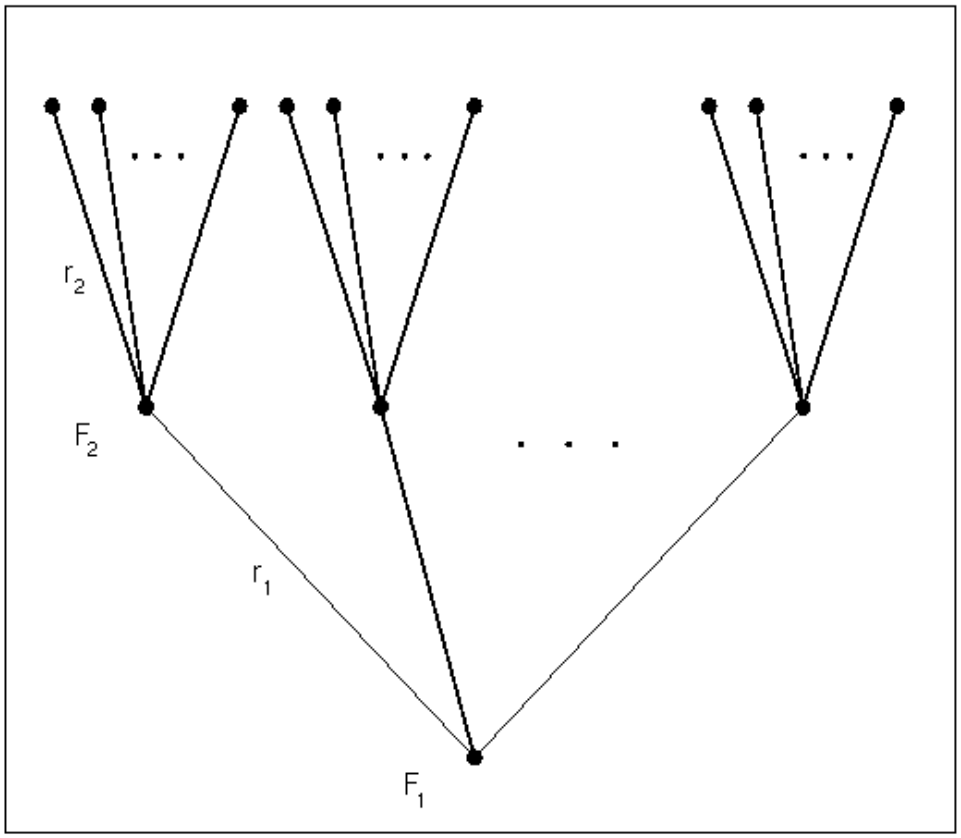
C_1. Traversing one of these branches corresponds to choosing the corresponding rule r_1 from C_1. The branches attached to the node at the next level correspond to the candidates for the second rule given r_1. In general, each node at level i of the tree corresponds to a set of feasible choices for rules $r_1, \ldots, r_i$, and the rest of the tree above this node corresponds to the set of all possible feasible choices for the rest of the rules.

Speeding Up the Search

For designs with many factors or blocks, the tree of candidate confounding rules can be very large and the search can take a very long time. In these cases, the FACTEX procedure spends a lot of time exploring sets of rules that are essentially the same and that all result in failure. A technique for pruning the search tree (see Figure 7.9) is as follows. Suppose that for some selection r_i for rule i, all the branches for the next rule eventually result in failure. Then any other selection r_i' is immediately declared a failure if the resulting number of candidates is the same as for the failed rule r_i. The search goes on to the next selection for rule i.

This method of pruning is not perfect; it might prune a branch of the search tree that would have resulted in a success. In mathematical terms, candidate sets C_i are not necessarily isomorphic because they have the same size. You can use the NOCHECK option in the PROC FACTEX statement to turn off the pruning. With the NOCHECK option, the FACTEX procedure searches the entire tree of feasible confounding rules, and if given enough time will find a design if one exists. The default argument for the TIME= option in the PROC FACTEX statement limits the search time to one minute.

Figure 7.9 Search Tree



On the other hand, the NOCHECK option is rarely needed to produce a design with a given resolution. For example, consider all possible blocked and unblocked two-level designs with minimum resolution for 20 or fewer factors and 128 or fewer runs. Of the nearly 400 different designs, the NOCHECK option is required to find a design in only nine cases. In one case (seven factors in 128 runs and blocks of size 2), NOCHECK is actually unable to find a design in the default time of 60 seconds, whereas the default search has no trouble finding a design.

General Recommendations

Choosing appropriate confounding rules can be difficult, especially if the set $\mathcal{M}$ is complicated. Even if a design is found that satisfies the model specification, it is a good idea to examine the alias structure to make sure that you understand the alias structure generated by the confounding rules. To do so, use the ALIAS option in the EXAMINE statement.

For more details about the general mathematical theory of orthogonal factorial designs, refer to Bose (1947).

References

Addelman, S. (1962), "Orthogonal Main-Effects Plans for Asymmetrical Factorial Experiments," *Technometrics*, 4, 21–46.

Bose, R. C. (1947), "Mathematical Theory of the Symmetrical Factorial Design," *Sankhyā*, 8, 107–166.

Box, G. E. P., Hunter, W. G., and Hunter, J. S. (1978), *Statistics for Experimenters*, New York: John Wiley & Sons.

Butler, N. A. (2004), "Construction of Two-Level Split-Plot Fractional Factorial Designs for Multistage Processes," *Technometrics*, 46, 445–451.

Byrne, D. M. and Taguchi, S. (1986), "The Taguchi Approach to Parameter Designs," *Quality Congress Transactions*, 177, 168–177.

Chakravarti, I. M. (1956), "Fractional Replication in Asymmetrical Factorial Designs and Partially Balanced Arrays," *Sankhyā*, 17, 143–164.

Cochran, W. G. and Cox, G. M. (1957), *Experimental Designs*, 2nd Edition, New York: John Wiley & Sons.

Dehnad, K., ed. (1989), *Quality Control, Robust Design, and Taguchi Method*, Pacific Grove, CA: Wadsworth & Brooks/Cole.

Fries, A. and Hunter, W. G. (1980), "Minimum Aberration 2^{k-p} Designs," *Technometrics*, 22, 601–608.

Huang, P., Chen, D., and Voelkel, J. O. (1998), "Minimum-Aberration Two-Level Split-Plot Designs," *Technometrics*, 40, 314–326.

Kempthorne, O. (1975), *The Design and Analysis of Experiments*, Huntington, NY: Robert E. Krieger Publishing.

Margolin, B. H. (1967), "Systematic Methods of Analyzing $2^n \times 3^m$ Factorial Experiments with Applications," *Technometrics*, 11, 431–444.

Montgomery, D. C. (1991), *Design and Analysis of Experiments*, 3rd Edition, New York: John Wiley & Sons.

Phadke, M. (1989), *Quality Engineering Using Robust Design*, Englewood Cliffs, NJ: Prentice-Hall.

Ramirez, J. G. and Weisz, J. T. (2009), "Designing Multi-step Fractional Factorial Split-Plots: A Combined JMP and SAS User Application," in *Proceedings of the SAS Global Forum 2009 Conference*, Cary, NC: SAS Institute Inc.

Searle, S. R. (1971), *Linear Models*, New York: John Wiley & Sons.

Williams, E. J. (1949), "Experimental Designs Balanced for the Estimation of Residual Effects of Treatments," *Australian Journal of Scientific Research, Series A*, 2, 149–168.

Wu, C. F. J. and Hamada, M. (2000), *Experiments: Planning, Analysis, and Parameter Design Optimization*, New York: John Wiley & Sons.

Yates, F. (1936), "Incomplete Randomized Blocks," *Annals of Eugenics*, 7, 121–140.

Yin, G. Z. and Jillie, D. W. (1987), "Orthogonal Design for Process Optimization and Its Application in Plasma Etching," *Solid State Technology*, 30, 127–132.

Subject Index

aberration of a design, *see* minimum aberration
alias structure
    breaking links, example, 639, 641
    details, 684
    example, 635, 636, 638, 654–656
    syntax, 625
analysis of variance, 675
augment, factorial design
    example, 635, 639

balanced lattice, 657
block designs
    balanced lattice, examples, 657
    randomized complete, examples, 641
block specification, FACTEX procedure
    block pseudo-factors, 622
    block size restrictions, 624
    number of blocks, 623
    runs per block, 623
blocking, FACTEX procedure
    block pseudo-factor, 689
    blocking factor, 689
    example, 664
    incomplete block design, example, 657
    randomization, 679
    rename block variable, 630

center points, example, 638
coding, FACTEX procedure
    block factor, 630
    design factor, 629
collapsing factors, example, 648
confounding rules
    compare with alias structure, 684
    design factors, 688
    details, 683
    example, 654
    MaxClear designs, 685
    minimum aberration, 684
    notation, 683
    orthogonally confounded, 690
    partial confounding, example, 654
    run-indexing factors, 688
    searching, 690
    split-plot designs, 686
    syntax, 625, 626
    unconfounded effects, 689
control factor design, 683
control factors, 683
control factors, example, 660
curvature, check for, example, 638

derived factors, FACTEX procedure
    creating, 632
    example, 647
design characteristics, FACTEX procedure
    alias structure, 683
    confounding rules, 683
    design listing, 626
design size specification, FACTEX procedure
    fraction, 633
    minimum runs, 634
    number of runs, 633
    run indexing factors, 633
    syntax, 633
design, factorial, *see* factorial design

effect length, FACTEX procedure
    limit, 622
examine design, FACTEX procedure, *see* design characteristics, FACTEX procedure
examples, FACTEX procedure
    advanced, 634
    alias links breaking, 635
    center points, 638
    collapsing factors, 648
    completely randomized, 634
    derived factors, 647
    design replication, 642, 645
    fold-over design, 639
    full factorial, 610
    full factorial in blocks, 612
    getting started, 610
    half-fraction factorial, 614
    hyper-Graeco-Latin square, 649
    incomplete block design, 657
    minimum aberration, 651
    mixed-level, 645, 647
    partial confounding, 654
    point replication, 642, 645
    pseudo-factors, 647
    randomized complete block design, 641
    RCBD, 641
    replication, 642, 645
    resolution 3 design, 639
    resolution 4, 651
    resolution 4, augmented, 635
    resolution III design, 639

resolution IV, 651
resolution IV, augmented, 635
sequential construction, 654

FACTEX procedure
block specification, 622
block specification options, summary, 617
design factor levels, 626
design size options, summary, 617
design size specification, 633
design specification options, summary, 617
examining design characteristics, 625
factor specification options, summary, 617
features, 609
getting started examples, 610
invoking, 621
learning about FACTEX, 610
listing design factors, 626
model specification, 627
model specification options, summary, 617
output, 629
overview, 608
randomization, 632
replication, 631
resolution, 627
split-plot designs, 686
statement descriptions, 621
summary of functions, 617
syntax, 617
uniteffect specification, 624
units specification, 623
using interactively, 616
factorial designs, *see* examples, See examples, FACTEX procedure
balanced lattice, 657, 658
efficiency, 629
fractional factorial, MaxClear designs, 685
fractional factorial, minimum aberration, 684
fractional factorial, theory, 687
mixed-level, 632
orthogonal, 645
replicate, 631
resolution, 627
split-plot designs, 686
factors, FACTEX procedure
block factor, 676, 689
block pseudo-factor, 677, 683, 689
derived factor, 677
design factor, 676
design factor coding, 629
design factor levels, 626
design factor names, 626
pseudo-factor, 677
run-indexing factor, 677, 683, 688
types, 676
fold-over design, example, 639

GLM procedure, 675, 676
Graeco-Latin square, 650

hyper-Graeco-Latin square, example, 649

independent estimate of error, examples, 638, 642
inner array, 660, 683
interaction, FACTEX procedure
alias structure, 684
between control and noise factors, 663
confounding, 688
examples, 654, 674, 675
generalized, 645, 688, 690
minimum aberration, 684
minimum aberration, example, 651
nonnegligible, 688
resolution, 678
specify terms, 627, 677

main effect, 677, 678, 688, 689
main effect, examples, 654–656, 674, 675
MaxClear designs, 685
minimum aberration
aberration vector, 684
blocked design, 685
example, 651
limitation, 653
minimum aberration, 684
mixed-level, factorial design
construction, examples, 645–649
derived factors, 632
model specification, FACTEX procedure
directly, 627
estimated effects, 627
indirectly, 627
maximum clarity, 628
minimum aberration, 628
nonnegligible effects, 627
resolution, 627
resolution, maximum, 627
specifying effects, 677
mutually orthogonal Latin square, 650, 657

noise factors, 660, 683

ODS tables
FACTEX procedure, 687
orthogonal confounding, 676, 677
orthogonal design
theory, 687
outer array, 660, 683
output, FACTEX procedure

code design factor levels, 629
decode block factor levels, 630
decode design factor levels, 629
details, 686
options, 629
output data set, 629, 687
rename block variable, 630

partial confounding, example, 654
PLAN procedure, 659
pseudo-factors, example, 647

randomization, FACTEX procedure
blocking, 679
details, 679
example, 634, 641
prevent, 632, 681
seed, 632, 641
randomized complete block, example, 641
randomized treatments, example, 641
replication, FACTEX procedure
data set, 631
design point, 631
design replication, 681, 683
details, 681
entire design, 631
example, 642, 645
fixed number of times, 681
inner array, 683
number of times, 631
outer array, 683
point replication, 681, 683
resolution, FACTEX procedure
comparison, 679
definition, 678
example, 614, 635, 651
MaxClear designs, 685
minimum aberration, 684
number, 679
numbering scheme, 679
syntax, 627
response, factorial design, 675, 676

search design, FACTEX procedure
confounding rules, 690
limit, 622
maximum time, 622
speeding, 691
signal-to-noise ratio, 660
size specification, FACTEX procedure, *see* design size specification
split-plot designs, 665, 686

Type I sum of squares, 675

Syntax Index

BLOCKS statement, FACTEX procedure, *see* FACTEX procedure, BLOCKS statement
  syntax, 622

EXAMINE statement, FACTEX procedure, *see* FACTEX procedure, EXAMINE statement
  syntax, 625

FACTEX procedure, 617
  getting started, 610
  learning about FACTEX, 610
  overview, 608
  summary of functions, 617
  syntax, 617
FACTEX procedure, BLOCKS statement
  NBLKFACS= option, 622
  NBLKFACS=MAXIMUM option, 623
  NBLOCKS= option, 623
  NBLOCKS= option, examples, 612, 654
  NBLOCKS=MAXIMUM option, 623
  SIZE= option, 623
  SIZE=MINIMUM option, 623
  UNITS= option, 623
FACTEX procedure, EXAMINE statement
  ALIASING option, 625
  ALIASING option, example, 615
  CONFOUNDING option, 625, 626
  DESIGN option, 626
  DESIGN option, example, 610
  SUMMARY option, 626
FACTEX procedure, FACTORS statement
  example, 610
  NLEV= option, 626
FACTEX procedure, MODEL statement
  ESTIMATE= option, 627
  ESTIMATE= option, examples, 637, 654
  MAXCLEAR option, 628
  MINABS option, 628, 685
  MINABS option, example, 652
  MINABS option, limitation, 653
  NONNEGLIGIBLE= option, 627
  RESOLUTION= option, 627
  RESOLUTION= option, examples, 614, 635, 639
  RESOLUTION=MAX option, 627
  RESOLUTION=MAX option, examples, 612, 643, 644
FACTEX procedure, OUTPUT statement
  CVALS= option, 630–632, 678
  CVALS= option, example, 641
  decode design factors, 629
  derived factors, 632
  derived factors, examples, 647, 649
  DESIGNREP= option, 631
  DESIGNREP= option, examples, 642–647
  NOVALRAN option, 632
  NVALS= option, 630–632, 678
  NVALS= option, example, 641
  OUT= option, 629
  OUT= option, example, 641
  POINTREP= option, 631
  POINTREP= option, examples, 642–647
  RANDOMIZE= option, 632
  RANDOMIZE= option, examples, 634, 641
  RANDOMIZE= option, NOVALRAN option, 632
  RANDOMIZE= option, seed, 632
  recode block factor, 630
  recode block factor levels, examples, 613, 641
  recode design factor levels, examples, 611, 615, 641
FACTEX procedure, PROC FACTEX statement
  example, 610
  NAMELEN option, 622
  NOCHECK option, 622, 653, 691
  ODS tables, 687
  SECONDS= option, 622
  TIME= option, 622, 653
FACTEX procedure, SIZE statement
  DESIGN= option, 633
  DESIGN= option, examples, 615, 635
  DESIGN=MINIMUM option, 634
  FRACTION= option, 633
  FRACTION=MAXIMUM option, 634
  NRUNFACS= option, 633
  NRUNFACS=MINIMUM option, 634
FACTORS statement, FACTEX procedure, *see* FACTEX procedure, FACTORS statement
  syntax, 626

MODEL statement, FACTEX procedure, *see* FACTEX procedure, MODEL statement
  syntax, 627

OUTPUT statement, FACTEX procedure, *see* FACTEX procedure, OUTPUT statement
  syntax, 629

PROC FACTEX statement, *see* FACTEX procedure, PROC FACTEX statement

syntax, 621

SIZE statement, FACTEX procedure, *see* FACTEX procedure, SIZE statement
syntax, 633

UNITEFFECT statement, FACTEX procedure, *see* FACTEX procedure, UNITEFFECTS statement
syntax, 624