Library dox
User's Manual

Version 2.0

October 2003

Insightful, Inc.
Seattle, Washington

The correct bibliographic reference for this document is as follows:
Insightful. Library dox User's Manual, Version 2.0, Seattle: Insightful Corp., 2003.

Contents

How to Use This Book

The *Library dox User's Manual* describes how to use the S-PLUS library dox, and includes detailed descriptions of the principal library dox functions.

In this document, you will learn how to perform the following tasks with library dox:

- Identify the most important factors affecting a product or process using *fractional factorial designs.*
- Analyze the response to changes in these key factors using *response surface methods.*
- Find the settings of "control" factors for which a response is minimally sensitive to "noise" factors, using *robust design methods.*

Note: This manual does *not* describe *full* or *replicated* factorial designs in detail, although library dox handles these designs. These designs are described in the chapters Designed Experiments and Analysis of Variance and Further Topics in Analysis of Variance in the S-PLUS *Guide to Statistics.*

Intended Audience

This manual, like the library dox is intended for industrial engineers, scientists, and statisticians—anyone involved in the design and analysis of industrial experiments.

Typographic Conventions

This manual obeys the following typographic conventions:

- The *italic font* is used for emphasis, and also for user-supplied variables within UNIX, Windows, and S-PLUS commands. For example,

  All objects have implicit, defining, and optional *attributes*.

- The **bold font** is used for UNIX and Windows commands and filenames, as well as for chapter and section headings. For example,

  setenv S_PRINT_ORIENTATION portrait
  SET SHOME=C:\SPLUS

  In this font, both **“** and **”** represent the double-quote key on your keyboard (").

- The `typewriter font` is used for S-PLUS functions and examples of S-PLUS sessions. For example,

```
> plot(corn.rain)
```

  Displayed S-PLUS commands are shown with the S-PLUS prompt `>`. Commands that require more than one line of input are displayed with the S-PLUS continuation prompts `+` and `Continue string:`.

- Boxed text represents either keys from the workstation keyboard or mouse buttons. For example,

  To delete a character, press the Delete key.

Warning: When you see the “dangerous bend” sign followed by the word **Warning**, you are seeing a warning about S-PLUS behavior. Read these warnings carefully.

⟹ **Hint:** When you see the right arrow followed by the word **Hint**, you are getting a peek ahead into more sophisticated use of S-PLUS.

Note: Points of interest that are neither warnings nor hints are preceded by the word **Note**.

Related Books

For users familiar with S-PLUS, the *Library dox User's Manual* contains all the information most users need to begin making productive use of library dox.

The S-PLUS *User's Manual* provides complete procedures to basic S-PLUS operation, including graphics manipulation, customization, and data input and output.

The S-PLUS *Guide to Statistics* describe how to analyze data using a variety of statistical and mathematical techniques, including classical statistical inference, linear regression, ANOVA models, generalized linear and generalized additive models, loess models, nonlinear regression, and regression and classification trees.

Comments?

We want our documentation to be useful, and we want it to address your needs. If you have any comments on this or any S-PLUS-related document, please send electronic mail to the following address:

```
doc@insightful.com
```

We'd love to hear from you.

Acknowledgements

Library dox was originally developed and and sold as the S+DOX module. It was jointly developed by Deborah J. Donnell of MathSoft, Inc. (now Insightful Corp.), and Perry D. Haaland and Michael A. O'Connell of the Becton Dickinson Research Center.

Some material in this manual is adapted from P. Haaland, *Experimental Design in Biotechnology*, Marcel Dekker, New York, 1989, and used by permission of the publisher. *Experimental Design in Biotechnology* may be ordered directly from the publisher by calling 1-800-228-1160

This manual was written by Deborah J. Donnell, Perry D. Haaland, and Michael A. O'Connell, with editing and formatting support from Richard Calaway, Jeff Cline, and Kjrsten Henriksen of MathSoft, Inc. (now Insightful Corp.).

We thank Marcel Dekker, Inc., Van Nostrand Reinhold, Inc., The Royal Statistical Society (*Applied Statistics*), William Keating, and Becton Dickinson for permission to use experimental design examples.

Chapter 1

Overview of the dox Library

Library dox is the S-PLUS library for design and analysis of industrial experiments. It provides a powerful, easy-to-use set of functions to simplify the design of experiments and to facilitate graphical analysis of the results. It is intended for use by industrial engineers, scientists, and statisticians. The functionality of library dox includes three major areas; design and analysis of fractional factorial experiments, response surface/optimization experiments, and robust design experiments (including Taguchi methods).

1.1 Why Design Experiments?

The competitive position of an industry or company depends on introducing new products quickly and efficiently, and on making continuous improvements to existing products and processes. To maintain a competitive position, industrial engineers and scientists must be able to efficiently collect and analyze data to develop and/or improve products. Statistical experimental design and analysis is a proven methodology for conducting and analyzing experiments efficiently and with reliable results.

Consider the development of a new chemical process. The developers seek operating conditions that produce the highest yield, acceptable purity, lowest cost, and

consistent performance over a wide range of conditions. The factors that affect the chemical process may include pH, amounts and types of catalysts, purity of the starting materials, reaction times and temperatures, amounts of various additives, etc. Optimal settings for each of these factors must be found to achieve the highest yield. If experiments are conducted by varying each factor in isolation, critical interactions are likely to be missed; for example, different catalysts may work more efficiently at different pH levels. On the other hand, a study of all possible combinations of the experimental factors would be impossible, as it would require a great number of experimental trials. Such an optimization problem appears daunting to an engineer or scientist untrained in statistical methods, so there is a great tendency to address only the obvious possibilities and hope for the best.

Industrial prosperity, even survival, demands higher quality than "change one variable at a time" or "try only the obvious" approaches can provide. What is needed is the application of statistical experimental design and analysis techniques. In this approach experimental trials are carefully selected in such a way that the effects of each of the experimental factors and their critical interactions are clearly delineated. Graphical methods are used to carry out an informative and helpful analysis. Consequently, experimenters move rapidly through a series of small experiments to optimize even extremely complex and costly processes.

It is no accident that today's leaders in research and development encourage the use of statistically designed experiments at all stages of the development cycle. The library dox software for industrial experimental design provides a resource that facilitates the use of statistical methods in industrial applications. Industrial engineers and scientists who use designed experiments can expect to make higher quality products—products with both high return and consistent performance. In addition, because of the efficient use of resources, these products can be brought to the market more quickly.

1.2 Approach and Philosophy

library dox was originally developed to support the consulting practice of industrial statisticians at the Becton Dickinson Research Center. library dox has a very practical, focused outlook and provides for the most common activities of industrial experimenters. The most commonly used experimental designs are readily available, together with straightforward, state-of-the art tools for data analysis. The analysis is primarily graphical, so experimental results are easily visualized and presented to others. Much of the graphical approach adopted in library dox is described in the book *Experimental Design in Biotechnology* by P. Haaland (Haa89). This reference is recommended as a practical guide for scientists and engineers who want to get started using statistical design to solve real problems. Experimenters are also referred to the excellent book by Box, Hunter and Hunter, *Statistics for Experimenters* (BHH78).

1.2.1 Integrating the Needs of Statisticians, Engineers, and Scientists

library dox has been designed to meet the needs of both the statistician and the practicing engineer/scientist. library dox provides an integrated resource for a modern industrial facility where engineers and scientists are expected to be largely responsible for their own experimental design and analysis, with statisticians available for special assistance. For scientists with less statistical training, standard analyses are simple to use and interpret. For statisticians, library dox and S-PLUS offer a rich resource for statistical methods. With library dox, the same software resource supports both high and low end users.

1.2.2 Object Oriented Structure Gives Simplicity and Guidance

All of the functions in library dox are object oriented. This means that the values returned by a function are self describing; for example an experimental design "knows" its design type, properties and experimental factors. Similarly, when a model is fit to experimental results the returned model result retains information about the original design, the factor names, various estimates of error, the use of transformations, etc. As a result, graphics and summary functions in library dox need only be given an analysis object for an appropriate graph or table to be created. This gives library dox a pleasantly simple structure with only a few commands.

library dox uses sensible, *context-sensitive* defaults. As the functions are object-oriented, the defaults can depend on the specific analysis. In most cases, a sophisticated graphical analysis can be completed using default parameters generated by library dox. In this way, library dox provides simple but very functional guidance to the practitioner.

1.3 Overview of Specific Capabilities

library dox may be sectioned into three closely related areas for the design and analysis of experiments: fractional factorial designs, response surface methods and robust designs. By *fractional factorial designs* we mean designs with all factors at two levels or just one factor at three levels. We use the term *response surface methods* to describe traditional second order designs with more than one factor at more than two levels. The term *robust designs* is used in reference to design analysis techniques for the joint analysis of mean and variance (signal and noise). Included in this is the Taguchi approach to design and analysis.

1.3.1 Fractional Factorial Designs

Fractional factorial designs are typically used by engineers and scientists to screen many experimental factors (or process variables) and, as a result, to find the important factors affecting the process responses. These designs provide the greatest amount of information in the fewest number of runs.

Special problems are encountered in the analysis and interpretation of these designs because they are often *unreplicated* and *effect saturated*. Unreplicated experiments have only one observation per set of experimental conditions so there is no independent estimate of error. Effect saturated experiments have as many independent model terms as there are observations. These designs present similar difficulties because there is no objective, classical method for testing significance of effects. library dox provides a state-of-the-art solution to this problem by implementing model selection methods based on the pseudo standard error (PSE) and other robust estimates of scale (Len89; HO94).

All of the standard two-level fractional factorial, Plackett-Burman, and orthogonal array designs are available in library dox. Designs can be generated using a simple naming convention. Specific designs can be obtained by providing the generating fraction. Blocking, center points, and randomization are available as appropriate for each design. A number of interesting and useful mixed level designs with two and three level factors are also provided.

The analysis of fractional factorial designs provided by library dox is primarily graphical. The graphical analysis consists of Pareto plots, normal and half-normal plots, active contrast (Bayes) plots, and interaction plots. In the unreplicated, effect saturated case, tests and approximate critical values are provided based on the pseudo standard error (PSE) and other robust estimates of scale (HO94). The classical ANOVA table may be obtained when there are degrees of freedom available to estimate the error. Transformation analysis of the response is simple to understand via the graphical interpretation provided. Classical statistical model fitting and diagnostics are also available.

1.3.2 Response Surface Methods

Response surface methods are used by engineers and scientists to optimize a process or product response. They typically involve three or four experimental factors, and often follow after a fractional factorial screening experiment has identified these few factors. Points are chosen in a "space-filling" design, so that a smooth surface can be fit to the response. From this surface, engineers can identify settings where the response is optimized.

library dox provides central composite designs, including face-centered cube designs, Box-Behnken designs and full matrix designs, with or without added center points.

Some or all of the factors can be specified on a log scale. By default, full quadratic surfaces are fitted, though the user can specify simpler models. Contour, surface and image plots are easily displayed and the user can specify slices of the surface using a straightforward syntax. Both constrained and unconstrained searches for optimal settings are available. Multiple responses can be overlaid on a single plot, allowing the scientist to visualize trade-offs between responses.

1.3.3 Robust Design

The robust design approach aims to develop processes and products that are insensitive to sources of "uncontrollable" noise. A common objective of a robust design is to make a process perform on target with minimum variability. For example, an engineer may wish to make a machine fill a container with a target volume and to rarely be over or under by more than 0.01%. These kinds of studies are often a key component in quality improvement work.

library dox provides both the Taguchi approach to robust design and analysis and a more standard statistical approach. In the Taguchi approach, the engineer uses inner (control) and outer (noise) arrays as elements of the design, and signal-to-noise ratios as the response in analysis. By maximizing an appropriate signal-to-noise ratio, the mean and variance are optimized together. This approach has been somewhat controversial and indeed has inherent problems as pointed out by (Box88) and others. The Taguchi approach has, however, been successful and certainly has drawn much needed attention to variance modeling and optimization.

The more standard approach replaces signal-to-noise ratios with transformations of the mean and standard deviation. Robust designs are simple to generate and analyze, and graphical methods are heavily emphasized. Regardless of the response(s) chosen for analysis, library dox uses the same graphical methods as for fractional factorial designs to identify important effects and recommend factor settings for optimization of the process mean and variance.

library dox provides significance testing based on robust estimates of scale as described by (HO94).

Chapter 2

Fractional Factorial Designs

Fractional factorial designs provide an efficient and reliable method for identifying important factors affecting a process or product. A typical application is a complex industrial process, where there are many factors that may affect the process, but only limited resources for investigation.

An example is taken from a company developing an enzyme immunoassay as a medical diagnostic test kit. The test kit is a complex product involving many separate components. Because components of the test kit lose their potency over time, the kit has a limited shelf life. The company is interested in developing a storage buffer solution that enhances the stability of the components, and hence increases the shelf life of the kit.

Different combinations of storage buffer components need to be tested in order to find a buffer with good stability for storage. There are many possible factors that might affect stability: pH, type of buffer, ionic strength, buffer salts, addition of proteins, chelators, and antimicrobial agents (Haa89, Chapter 4).

This is a typical situation where a fractional factorial experiment is used—there are many factors you want to test, but making up and testing new buffers is time consuming and expensive, so you have a limit on the number of candidate buffer solutions you can try. A fractional factorial experiment is an efficient way to screen several factors with just a few storage solutions.

The main aim of a factorial experiment is to find out if the factors are important or not. To do this, you only need to use two levels for each factor—this is enough to see whether changing the factor setting affects the response. For example, a factorial design with three factors at two levels requires eight *experimental runs* to study all combinations of the factor settings:

	pH	azide	thimer
1	−	−	−
2	−	−	+
3	−	+	−
4	−	+	+
5	+	−	−
6	+	−	+
7	+	+	−
8	+	+	+

Each row represents the settings of the factors for one storage solution. For `pH`, the "−" indicates a low setting and the "+" a high setting. For `azide` and `thimer` (thimerosal) which are both antimicrobial agents, the "−" means "absent" and the "+" means "present". The specific settings are determined by the scientists previous experience with the process. To carry out the experiment, eight different storage buffers are prepared and their storage stability is measured (a response value) for each storage buffer (run).

Notice the symmetry in the pattern of the levels in the experiment above—the levels are chosen so the effect of each factor (`pH, azide, thimer`) can be calculated independently. For instance, suppose changing `pH` from high to low increases the response by about 4 but the presence of `azide` decreases the response by about 2. The *effect* of `pH` can be computed by comparing all the responses at high `pH` against all those at low `pH`: the effect is the difference of the average response at each level. Note that `azide` is present in two and absent in two of the high `pH` responses, so that the decreasing effect of `azide` cancels out—similarly at low `pH`. So the `pH` effect is independent of the `azide` effect. Designs with this property have effects that are called *orthogonal*.

The above design is a *full* factorial, which has all combinations of all factor levels. With a full factorial the effects of all the factors and their interactions can be computed, but the number of experimental runs needed increases by a factor of two for each factor added—so for 5 factors $2^5 = 32$ runs are needed.

A *fractional* factorial design contains only a (small) fraction of the runs required for the full factorial—usually an even fraction, like 1/2 or 1/4. If the runs in the fraction are chosen carefully, all the *main* effects can still be estimated, that is, the effect of changing between levels for `pH`, `azide`, and `thimer`. However not all *interactions* can be estimated. An interaction is an effect that indicates a dependency between two or more factors, for instance if `azide` has a different effect at high `pH` than low `pH` there is a two-factor `azide:pH` interaction.

An example of a fractional factorial is a half fraction of the above eight run experiment:

	pH	azide	thimer
1	−	−	−
4	−	+	+
6	+	−	+
7	+	+	−

With this design the main effects of `pH`, `azide`, and `thimer` can still be estimated.

Once the experimental responses have been collected, the aim of the analysis in library dox is to determine which experimental factors are important. To decide which factors "significantly" affect the response you need to estimate the natural variability of the response, that is, the change in the response measured several times with exactly the same factor settings. This natural variation is the experimental error. In an unreplicated factorial experiment you do not have a direct measure of the experimental error (because you never use exactly the same settings twice), but you can estimate the error based on the variability of the unimportant effects. In library dox there are several alternative methods for calculating an approximate error, and we give some guidance on when to use each method. The estimated error is then used as a guide for assessing which effects are large.

The analysis of the fractional factorial experiment reveals which factors are important. But for continuous factors, you can only assess the *direction* of the best settings from the experiment. For example, a single storage buffer experiment as described above could establish that high pH is better than low, but could not identify the optimal high pH setting. Finding optimal settings for the experimental factors would be the aim of a subsequent experiment, possibly a *response surface experiment*. See chapter 3, Response Surface Methods and Process Optimization.

The analysis of fractional factorials usually follows a simple initial sequence:

1. Read in or create the design and response;
2. Take an initial look at the data with some simple exploratory plots;
3. Fit a saturated model;
4. Examine a Pareto plot of the effects to identify the most important factors and interactions.

The effects are explored further using a main effects plot, two-factor interaction plots, and a half normal plot. A full normal plot is used as a simple check for a single outlier. If it is difficult to decide whether a factor is significant, further analysis using active contrast plots and empirical Bayes methods can be helpful. Sometimes transforming the response can lead to a much simpler model; the Box-Cox log-likelihood transformation analysis evaluates the best transformation.

In summary, a fractional factorial is a screening experiment used to find the most important factors with the fewest number of experimental runs. Fractional factorial experiments are both efficient and cost-effective first steps in investigating a process. The software and procedures described in this chapter are especially helpful for *unreplicated, effect-saturated* designs for which there is no estimate of error. The techniques can also be used for unsaturated and replicated designs.

2.1 Starting out with Fractional Factorial Designs

The most important stage in designing an experiment is outside the scope of this software: defining the goal of the experiment, choosing a response that will reflect that goal, and deciding on the experimental factors and their levels. This is an iterative process requiring careful thought about the particular problem.

Once the experimental objectives are decided, the response variables have been selected and possible experimental factors are identified, an experimental design can be selected. Choosing an experimental design involves compromising between the number of factors you want to include, the number of runs you can afford, and the numbers of interactions you want to be able to estimate. With 4 factors, if you can afford a full factorial with $2^4 = 16$ runs, you will be able to estimate all main effects and two and three way interactions. If you use 8 runs (half fraction) of the full design, you can still estimate all main effects but two way interactions are *confounded* with each other. When two effects are confounded they cannot be estimated independently. Often in screening experiments you are willing to assume three-way interactions are unimportant, and sometimes that main effects are more important than two-way interactions, in exchange for using fewer experimental runs.

Generate the eight run design with four two-level factors with either of these calls:

```
> half.four <- design.digest(rep(2,4), fraction = 1/2)
```

or

```
> half.four <- design.digest("ff0408")

> half.four
Design Name: ff0408

  A B C D
1 - - - -
2 - - + +
```

```
3 - + - +
4 - + + -
5 + - - +
6 + - + -
7 + + - -
8 + + + +

Fraction:  ~ A:B:C:D
```

The character string provides a shorthand notation for designs in the `design.digest`: in `"ff0408"`, `ff` stands for *fractional factorial*, the next two digits stand for *4 factors*, and the last two digits stand for *8 runs*.

The *confounding pattern* of this design shows which terms are confounded. This information is in the summary of the design:

```
> summary(half.four)
Design Name: ff0408

Generating Fraction:  ~  A:B:C:D
Fractional Number of Runs: 1 / 2
Resolution: IV

Confounding Pattern:
 Main effects are not confounded with two-factor
interactions. Two-factor interactions are confounded
with other two-factor interactions. Only one two-factor
interaction from each line listed below can be estimated.

A:B + C:D
A:C + B:D
B:C + A:D

Variable summaries:
   A     B     C     D
 -:4   -:4   -:4   -:4
 +:4   +:4   +:4   +:4
```

The confounding pattern shows the terms that are *completely confounded* (or equivalently, *aliased*) with other model terms. Terms that are completely confounded are actually indistinguishable from each other. The above design allows us to estimate the main effects of each factor. However, the two-factor interactions are confounded with each other, e.g., `A:B`[1] is confounded with `C:D`, so you will not be able to tell whether a significant interaction is due to a `A:B`, `C:D`, or a combination of the two.

[1] `A:B` is the notation for the two-factor interaction between `A` and `B`.

Fractional factorial designs are characterized by their confounding patterns. In general, you always want at least main effects to be independent of each other (resolution III designs). In resolution III designs, two-way interactions are confounded with main effects. Designs like the above example, that have main effects independent of all two-way interactions are called resolution IV designs. Designs with all main effects and all two-way interactions independent of each other are called resolution V designs. The different resolution designs are suited to different experimental applications.

The following table summarizes the highest resolution available for 8 and 16 run experiments for the given number of factors:

	Runs	
Number of factors	8	16
3	full	
4	IV	full
5	III	V
6	III	IV
7	III	IV

By default, fractions with highest resolution are returned by `design.digest`, and the `summary` function gives the confounding pattern of the returned design.

For early screening experiments with many factors, even if you can afford more runs, a good strategy is to start with a minimal design (resolution III). Then, after analyzing the results of this first experiment, you can decide whether to complete the fraction, or—if you have eliminated some factors—to use the extra runs to refine your analysis. For intermediate screening experiments where there are not so many factors and some information regarding interactions is desirable, resolution IV designs are appropriate. Resolution V designs are generally used for late screening experiments when most of the unimportant factors have been eliminated and interactions between the remaining factors are all possible.

2.1.1 Generating a Design

The initial analysis of a factorial design usually follows a straightforward sequence of steps: getting the data into library dox, plotting the data, fitting a saturated model, and examining the estimated effects.

The actual fractional factorial screening experiment that was conducted to study storage buffer stability used five experimental factors: pH of the buffer; the presence or absence of three antimicrobial agents: gentamicin, thimerosal, and azide; and the presence or absence of a chelating agent which scavenges free electrons from the buffer: chelex.

A design of 16 runs is selected. The data frame is already available in library dox with the name `buffer.df`, but you could create the design data frame as follows:

```
> buffer.design <- design.digest("ff0516",
+   c("pH","chelex","azide","gent","thimer"))
```

or

```
> buffer.design <- design.digest(rep(2,5), c("pH",
+   "chelex","azide","gent","thimer"), fraction = 1/2)
```

The first argument to `design.digest` gives either the name of the design (`"ff0516"`, a fractional factorial design with five factors and 16 runs) or the number of levels in each factor. The second names the factors. The default levels for two level factors are "−" and "+", representing low and high levels. The default fraction is chosen by `design.digest` from a list of standard fractions[2].

2.1.2 Randomizing the Design

In experimental designs it is extremely important that the order of the runs is randomized when the experiment is conducted. This guards against effects from time of day or temperature of the laboratory getting mixed up with the effect of experimental factors (which could happen if all of the runs with the low level of a factor were prepared together, followed by all of the high level). To make this easy, library dox provides two functions, `randomize.design` and `sort.design`, to randomize and then resort the design.

Before the experiment, create a randomized version of `buffer.design`, then print the design using `worksheet` for use in the laboratory.

```
> buffer.rdes <- randomize.design(buffer.design)
> worksheet(buffer.rdes, response = "rate", graphics = T)
```

The worksheet in figure 2.1 appears in the graphics window with an empty column for the single response `rate`.

[2]Note that the S-PLUS function `design.digest` uses a default row ordering different from that of `fac.design`, with the leftmost factor varying slowest. For example, to generate the same design with `fac.design`, reverse the order of the factors: `fac.design((rep(2,5), c("thimer","gent","azide","chelex", "pH"), fraction = 1/2)`

	pH	chelex	azide	gent	thimer	rate
1	-	+	-	+	+	
2	+	-	+	+	-	
3	-	+	-	-	-	
4	+	+	-	-	+	
5	+	-	-	-	-	
6	+	-	+	-	+	
7	-	+	+	-	+	
8	-	-	+	-	-	
9	-	-	-	+	-	
10	+	+	+	+	+	
11	+	+	+	-	-	
12	-	-	+	+	+	
13	+	+	-	+	-	
14	-	+	+	+	-	
15	-	-	-	-	+	
16	+	-	-	+	+	

Figure 2.1: A worksheet of the randomized design for the storage buffer experiment.

2.1.3 Entering the Data

The 16 experimental buffers were made up and reagents stored in each of the buffers for one month. The stability response is measured as the amount or rate of degradation (determined by assay) based on the amount of active component remaining. The aim is to achieve a low rate of degradation.

Enter the response, degradation rate, in the order corresponding to the randomized design. Add it to the data frame, then resort the design[3]:

```
> rate <- c(7.04, 1.73, 8.42, 1.55, 2.34, 1.29, 7.21, 8.78,
+   9.81, 1.36, 1.81, 7.92, 1.68, 9.96, 6.9, 1.22)
> buffer.rdf <- cbind(buffer.rdes, rate)
> buffer.df <- sort.design(buffer.rdf)
> buffer.df
```

[3]If the response has already been reordered to match the design in *standard* order, omit the reordering step and `cbind` the response to the original design frame

```
Design Name: ff0516

   pH chelex azide gent thimer rate
 1  -      -     -    -      + 6.90
 2  -      -     -    +      - 9.81
 3  -      -     +    -      - 8.78
 4  -      -     +    +      + 7.92
 5  -      +     -    -      - 8.42
 6  -      +     -    +      + 7.04
 7  -      +     +    -      + 7.21
 8  -      +     +    +      - 9.96
 9  +      -     -    -      - 2.34
10  +      -     -    +      + 1.22
11  +      -     +    -      + 1.29
12  +      -     +    +      - 1.73
13  +      +     -    -      + 1.55
14  +      +     -    +      - 1.68
15  +      +     +    -      - 1.81
16  +      +     +    +      + 1.36

Fraction:  ~ pH:chelex:azide:gent:thimer
```

If the experiment is in an ASCII file **buffer.dat** with the column labeled at the top of the file, create the fractional factorial data frame in library dox[4] by:

```
> buffer.rate <- read.table("buffer.dat", header = T),
> buffer.rate <- as.fac.design(buffer.rate, factors = 1:5)
```

2.1.4 Fitting a Model to the Data

The default data plot splits the response by each factor:

```
> plot(buffer.df)
```

Figure 2.2 is clearly dominated by a large `pH` effect.

Now fit an *effect saturated* model to the data using `fac.aov` and print the model summary:

[4] `design.digest` returns an object of class `fac.design`. The properties of `fac.design` objects are used extensively by library dox. The function `as.fac.design` is required to convert the data frame created by `read.table` to this class.

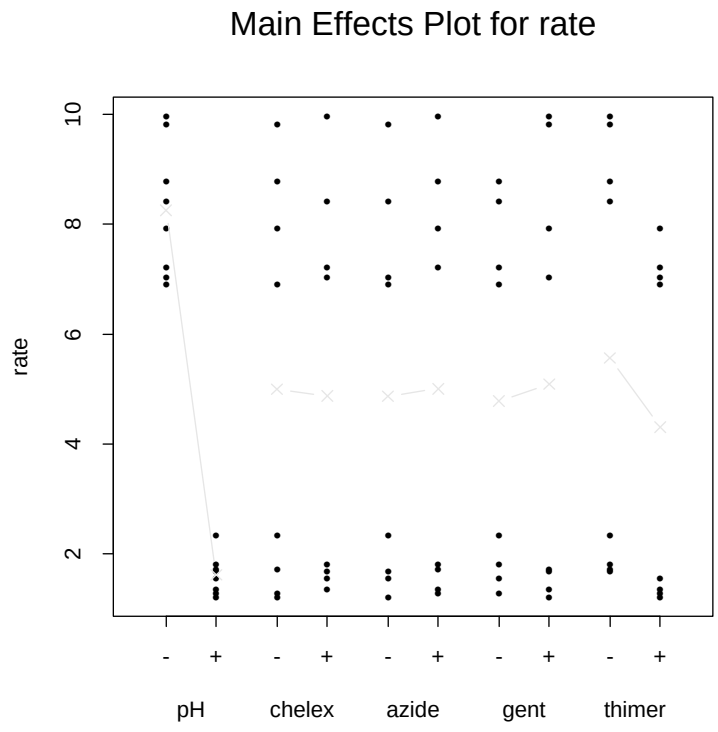

Figure 2.2: An exploratory plot of the buffer experiment with the response plotted by each experimental factor.

```
> summary(buffer.fac)
Estimated Effects for Response: rate

Design Name: ff0516
              Effect Std. Error |t(PSE)| P-value
          pH -6.6300 0.216      30.700   <.001
      chelex -0.1200 0.216       0.555   >0.3
       azide  0.1370 0.216       0.635   >0.3
        gent  0.3030 0.216       1.400   0.165
      thimer -1.2600 0.216       5.800   0.009
   pH:chelex  0.0750 0.216       0.347   >0.3
    pH:azide -0.2870 0.216       1.330   0.184
 chelex:azide  0.2750 0.216       1.270   0.201
     pH:gent -0.5530 0.216       2.550   0.031
  chelex:gent -0.0400 0.216       0.185   >0.3
   azide:gent  0.1670 0.216       0.774   >0.3
    pH:thimer  0.7200 0.216       3.330   0.015
chelex:thimer  0.0775 0.216       0.358   >0.3
```

```
azide:thimer  0.1300 0.216       0.601   >0.3
 gent:thimer -0.1550 0.216       0.716   >0.3

 Pseudo Standard Error of the Effects =  0.216
 Approximate P-values calculated based on an
   empirical distribution for the PSE obtained
   via simulation from an appropriate null
```

An effect saturated model has no residual degrees of freedom. The estimated effects are the average change in response from `-` to `+` level of a factor, e.g., on average, the rate of degradation decreases by 6.6% when `pH` is changed from low to high. The standard errors given in the summary are approximate and based on the pseudo standard error (Len89) which basically reflects the variance of the smallest effects. The p-values are based on the empirical distribution of the statistic t(PSE) which is equal to the absolute value of an estimated effect divided by the pseudo standard error (HO94). The pseudo standard error and alternative error methods are explained in section 2.2.3.

Important in the analysis of saturated designs is the so-called Pareto principle: the idea that usually only a few of the factors are responsible for most of the variability in the response. The Pareto plot is a graphical display of the effects ordered by size:

```
> pareto(buffer.fac)
```

The Pareto plot shown in figure 2.3 displays the ordered absolute effects, where the sign of the effect is indicated by shading. Often, as in this plot, there are some obviously large effects and some obviously small effects, and a few lying in-between. The standard error is needed to assess these in-between effects. The significance line gives a 5% critical value based on the pseudo standard error: effects larger than this are possibly significant. In this case, `pH` and `thimer` are clearly significant, and the two interactions `pH:thimer` and `pH:gent` are probably also important.

A second view of the effects is given by the half-normal plot (based on the quantile-quantile or QQ-plot, hence the function name):

```
> qqnorm(buffer.fac)
```

In a half-normal plot the ordered absolute values of the effects are plotted against ordered normal quantiles. If none of the factors affect the response, the data should look like a random sample from a normal distribution, and the points on the half-normal plot lie close to a straight line, with slope equal to the reciprocal standard error of the effects. If there *are* significant factors, their effects will be much larger than expected in a normal sample, and they will lie far below and to the right of this line. The non-significant effects will still lie close to the line.

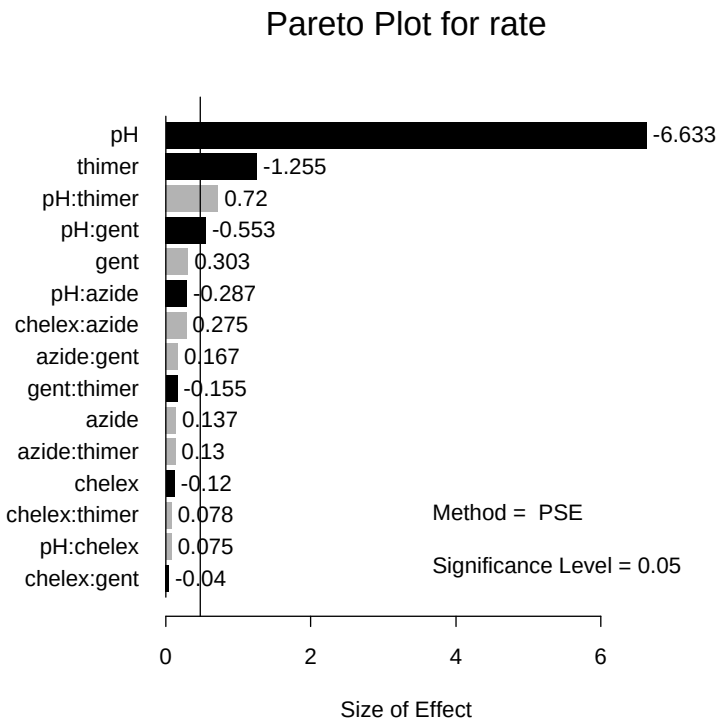

Figure 2.3: In this Pareto plot of factorial effects for the buffer experiment, the effects of `pH` and `thimer` are clearly important, and the two-way interactions `pH:thimer` and `pH:gent` are possibly important.

In figure 2.4 the largest four effects seem to "fall off" the pseudo standard error line through the remaining small effects. See section 2.2.3 for further use of the half-normal plot. From this plot and the Pareto plot, it seems likely that the two largest interactions are significant.

You can easily interpret these interactions with the help of two-factor interaction plots:

```
> tfiplot(buffer.fac,~ pH:thimer + pH:gent)
```

The second argument to `tfiplot` is an S-PLUS `formula` that specifies the two factor interactions you want to plot. It is usually better to put the biggest effect (here, `pH`) on the x-axis, by naming it first in each formula term.

Recall that a slow degradation rate is better, so clearly high `pH` is preferable. Figure 2.5 also shows that while the presence of thimerosal at low pH has a large effect

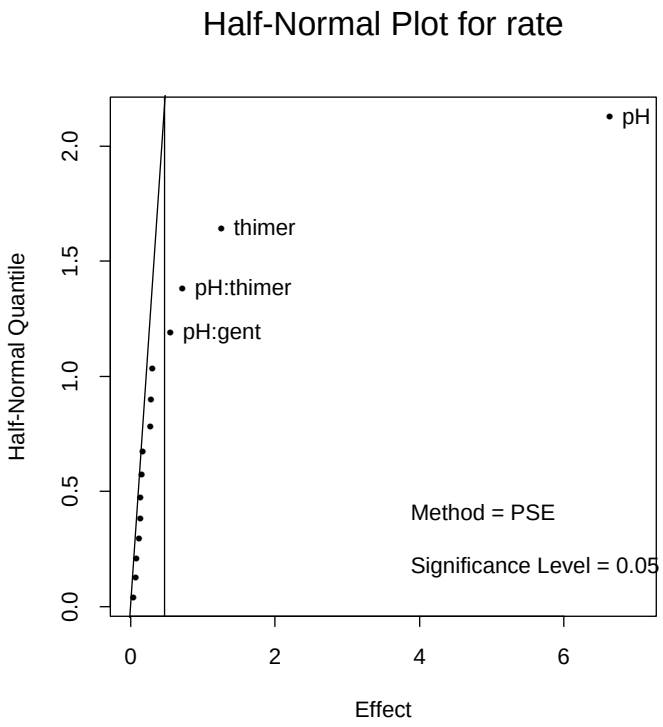

Figure 2.4: The half-normal plot of effects are shown for the buffer experiment, with the line of slope equal to the pseudo standard error. Possibly significant effects are labeled.

in reducing the degradation rate, at high pH it makes less difference. This type of interaction is often referred to as a proportional interaction, and can sometimes be removed by transforming the response; see section 2.4. The `pH:gent` interaction is often referred to as an inconsistent interaction because at low pH degradation rate is lower without gentamicin while at high pH (though there is only a small difference) degradation rate seems lower *with* gentamicin.

With an effect-saturated model, there are as many terms in the model as there are total degrees of freedom, so the data "fit perfectly", and in general it is difficult to check for outliers. However, a single outlier will change the estimates of all effects; in particular, the near-zero effects will be shifted away from zero as the outlying observation contributes either by addition or subtraction to every effect. This creates a shift in the effects at zero, so a full-normal (QQ) plot of the effects will show a gap at zero (Dan76). A full-normal plot is similar to the half-normal plot, but the *signed* effects are plotted, rather than their absolute values.

Use a full-normal plot to check for an outlier in the buffer experiment:

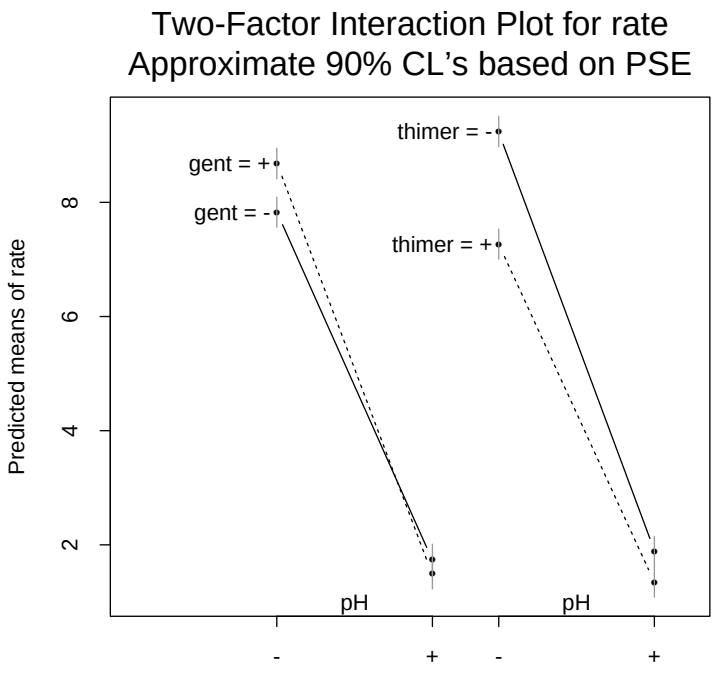

Figure 2.5: Two-factor interactions are plotted for the effects `pH:thimer` and `pH:gent`.

```
> qqnorm(buffer.fac, full = T)
```

No outlier is apparent in figure 2.6.

To illustrate the plot when there *is* an outlier, we create an outlier in this data and display the full-normal:

```
> buffer.outlier <- buffer.df
> # create an outlier in the first observation
> buffer.outlier[1,"rate"] <- buffer.outlier[1,"rate"] - 4
> buffer.outfac <- fac.aov(buffer.outlier)
> qqnorm(buffer.outfac, full = T, omit = 1)
```

The argument `omit = 1` omits the largest (absolute) effect simply to reduce the distortion in the plot due to the large `pH` effect. You can clearly see a gap in the effects at zero in figure 2.7: this feature usually indicates a single outlier in the data.

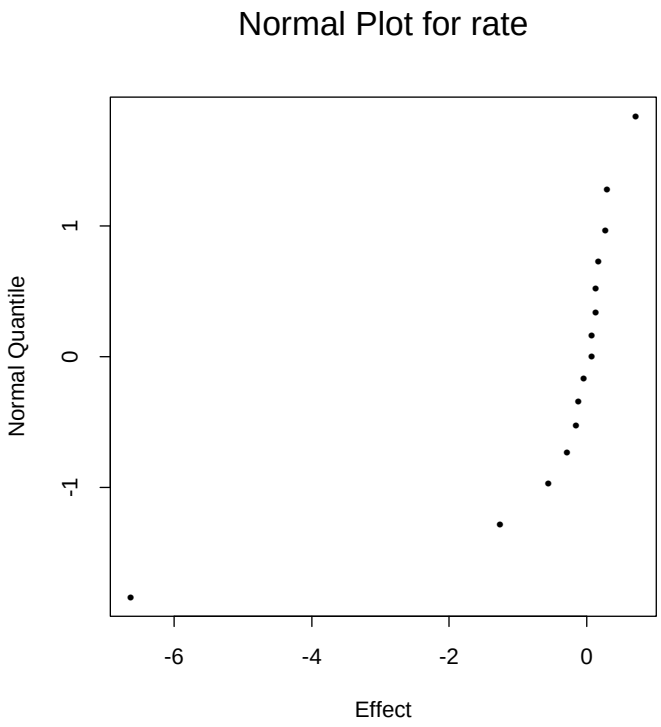

Figure 2.6: Quantiles of the fitted effects are plotted against Normal quantiles for the buffer experiment fit.

Thus far, you can eliminate azide and chelex as relatively unimportant influences in the buffer solution. If the high pH, high gentamicin condition has not produced a desirable degradation rate, further experiment could concentrate on a new high pH range and investigate the effect of thimerosal and gentamicin in that range.

2.2 Options in Analyzing the Data

This section gives more details regarding the analysis techniques and illustrates some of the options available in the factorial analysis functions. This includes more information on the use of formulas to specify terms included in the model, aliasing of effects, selecting a method for estimating the error, and Bayesian methods for identifying important effects.

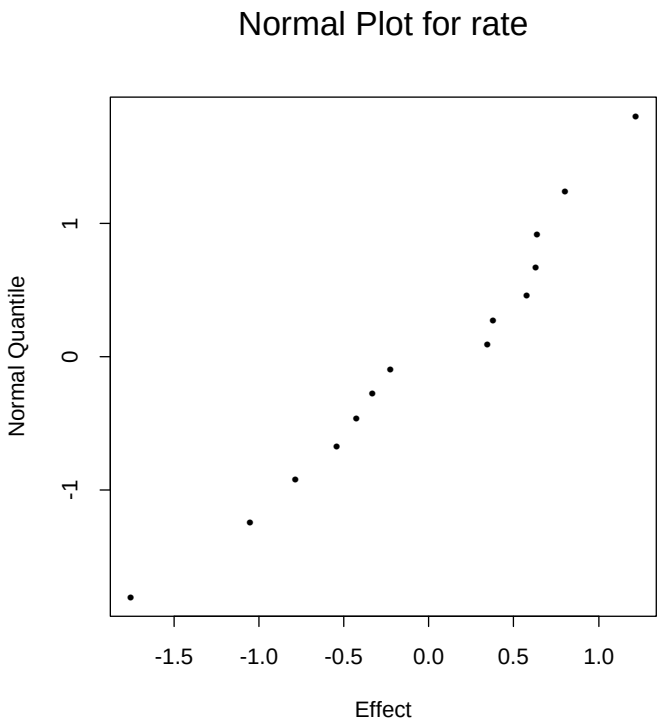

Figure 2.7: Quantiles of the fitted effects are plotted against Normal quantiles, where the data have an outlier. The gap in the effects at zero indicates the presence of an outlier. The largest (pH) effect has been omitted from the plot to make the outlier effect easier to see.

2.2.1 Fitting Models

The default method for fractional factorials fits an effect-saturated model to the data, using the default formula[5]:

```
> formula(buffer.df)
rate ~ pH * chelex * azide * gent * thimer
```

Because there are only 16 data points and 15 degrees of freedom available, only the first 15 terms implied by the formula are fitted, giving the effect-saturated

[5]When there are 7 or more factors, the default formula is simplified to a third order model, `y ~ (.)^3`. In general the default model will have enough terms to ensure that the full effect-saturated model will be obtained but not so many terms as to cause excessively long times to fit the model.

analysis of the previous section. Of course, you may choose to use only some of the factors in the data frame by giving the formula explicitly, in the usual S-PLUS style. (Formulas are discussed thoroughly in the S-PLUS *Guide to Statistics*.)

To show how formulas work, suppose you have decided that `chelex` is not a significant factor and that it has no important interactions with other factors. Redo the analysis excluding `chelex`:

```
> buffer.newfac <- fac.aov(rate ~ pH * azide * gent *
+ thimer, data = buffer.df)
> pareto(buffer.newfac)
```

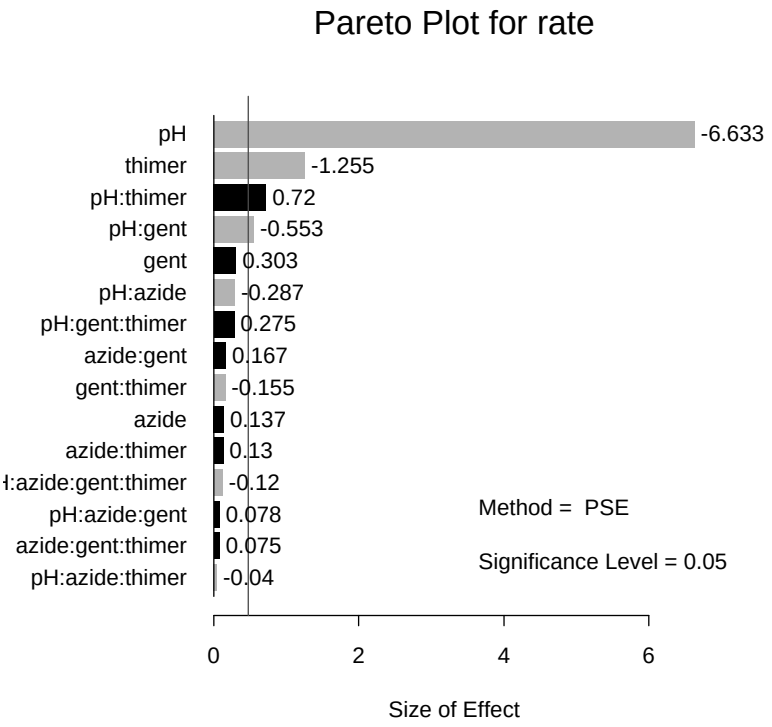

Figure 2.8: The Pareto plot for the effects of the model `rate ~ pH * azide * gent * thimer`.

Comparing the Pareto plots figures 2.8 and 2.3 you get the same values for the estimated effects, but the effects have different names! The terms that were previously associated with `chelex` are now labeled as three and four-way interaction effects: these are effects that were confounded with the `chelex` effects in the original model. If you were willing to ignore the high order interactions, you could specify a second order model (only main effects and two-way interactions) by:

```
> buffer.newfac2 <- update(buffer.newfac, . ~ (pH + azide
+        + gent + thimer)^2)
> summary(buffer.newfac2)
Estimated Effects for Response: rate

Design Name: ff0516 (reduced model)
             Effect Std. Error |t(MSE)| P-value
         pH -6.630 0.144      46.200   <.001
      azide  0.137 0.144       0.957   0.383
       gent  0.303 0.144       2.110   0.089
     thimer -1.260 0.144       8.730   <.001
   pH:azide -0.287 0.144       2.000   0.102
    pH:gent -0.553 0.144       3.840   0.012
  pH:thimer  0.720 0.144       5.010   0.004
 azide:gent  0.167 0.144       1.170   0.296
azide:thimer 0.130 0.144       0.905   0.407
 gent:thimer -0.155 0.144      1.080   0.33

  Mean Standard Error of Effects =  0.144
         with  5  degrees of freedom.
  Experimental Error (RMSE) = 0.287
  R-squared =  0.998
  P-values calculated using the t-distribution.
```

The omitted effects allow estimation of the experimental error with the usual root mean squared error of the standard ANOVA, and the mean standard error of the effects is based on this.

2.2.2 Aliased Terms

Effects that are completely confounded or aliased are indistinguishable, so when interactions are aliased the name of the interaction is completely arbitrary. For instance, suppose you fit a saturated model to the following half fraction of a four factor experiment on latching mechanism performance (Dia81, p. 113).

	tension	angle	slength	swidth	fail
1	−	−	−	−	30
2	−	−	+	+	8
3	−	+	−	+	12
4	−	+	+	−	0
5	+	−	−	+	32
6	+	−	+	−	8
7	+	+	−	−	8
8	+	+	+	+	4

The data frame `latch.df` is available in library dox, but you could create it with the following commmands:

```
> latch.design <- design.digest("ff0408",
+   c("tension", "angle", "slength", "swidth")
> latch.fail <- c(30, 8, 12, 0, 32, 8, 8, 4)
> latch.df <- cbind(latch.design, fail=latch.fail)
```

The confounding pattern among the two-way interactions is:

```
> summary(latch.df)

Design Name: ff0408

Generating Fraction:  ~  tension:angle:slength:swidth
Fractional Number of Runs: 1 / 2
Resolution: IV

Confounding Pattern:
 Main effects are not confounded with two-factor
interactions. Two-factor interactions are confounded
with other two-factor interactions. Only one two-factor
interaction from each line listed below can be estimated.

A:B + C:D
A:C + B:D
B:C + A:D

Variable summaries:
 tension   angle   slength   swidth         fail
 -:4       -:4     -:4       -:4      Min.   : 0.00
 +:4       +:4     +:4       +:4      1st Qu.: 7.00
                                      Median : 8.00
                                      Mean   :12.75
                                      3rd Qu.:16.50
                                      Max.   :32.00
```

The main effects are independent, but the two-way interactions are confounded. This is a resolution IV design. In particular, `tension:angle (A:B)` is confounded with `slength:swidth (C:D)`, `tension:slength (A:C)` is confounded with `angle:swidth (B:D)` etc. You can look at two different "interpretations" of the saturated model effects, by changing the order of the factors in the formula:

```
> summary(fac.aov(latch.df))
Estimated Effects for Response: fail

Design Name: ff0408
                Effect Std. Error |t(PSE)| P-value
        tension   0.5  2.17       0.231    >0.3
          angle -13.5  2.17       6.230    0.01
        slength -15.5  2.17       7.150    0.01
         swidth   2.5  2.17       1.150    0.229
  tension:angle  -0.5  2.17       0.231    >0.3
tension:slength   1.5  2.17       0.692    >0.3
  angle:slength   7.5  2.17       3.460    0.027

  Pseudo Standard Error of the Effects =  2.17
  Approximate P-values calculated based on an
    empirical distribution for the PSE obtained
    via simulation from an appropriate null
> summary(fac.aov(fail ~
+   swidth*slength*tension*angle, latch.df))
Estimated Effects for Response: fail

Design Name: ff0408
                Effect Std. Error |t(PSE)| P-value
         swidth   2.5  2.17       1.150    0.229
        slength -15.5  2.17       7.150    0.01
        tension   0.5  2.17       0.231    >0.3
          angle -13.5  2.17       6.230    0.01
 swidth:slength  -0.5  2.17       0.231    >0.3
 swidth:tension   7.5  2.17       3.460    0.027
slength:tension   1.5  2.17       0.692    >0.3

  Pseudo Standard Error of the Effects =  2.17
  Approximate P-values calculated based on an
    empirical distribution for the PSE obtained
    via simulation from an appropriate null
```

Each of the interaction terms has been given a different label—the large interaction of 7.5 is called `angle:slength` in the first model, and `tension:swidth` in the second. Note that interpreting the interaction as `angle:slength` is the most likely and consistent explanation of the data, because both `slength` and `angle` have large main effects. Thus even though a resolution IV design was used the large interaction may be given a "most likely" interpretation.

2.2.3 Selecting an Error Method

In effect-saturated models, there is no direct measurement of error: the tests you use to identify the significant effects are based on methods that approximate the error. The error approximations all assume that only some of the fitted effects are really active; the other smaller effects are inactive and reflect the error of the response. The methods differ in how the "small" effects are defined and the type of scale estimate used.

The default method is the pseudo standard error (PSE), which is a robust scale estimate calculated from the trimmed effects[6]. The following definition of the pseudo standard error is similar to that given by Lenth (Len89) but has been adapted according to Haaland and O'Connell(HO94):

$$\begin{aligned} s_0 &= 1.5 \times \text{median}(\|T_i\|) \\ \widehat{\sigma}_{\text{PSE}} &= a_{\text{PSE}} \times \text{median}(\|T_i\| : \|T_i\| \leq 2.5 s_0) \end{aligned}$$

where $\|T_i\|, i = 1, \ldots, k$ are the absolute values of the estimated effects and a_{PSE} is a consistency constant that depends on the total number of effects.

Simulation studies have shown that the PSE is a good all-around method that works well under a variety of conditions (HO94). Two additional estimates of the standard error of the effects that are available are the trimmed standard error (TSE) and the adaptive standard error (ASE).

The adaptive standard error (Don93) is based on trimmed effects as per the pseudo standard error but it follows up with an efficient scale estimate namely,

$$\hat{\sigma}_{\text{ASE}} = a_{\text{ASE}} \times \sqrt{\frac{\sum_{\|T_i\| \leq 2.5 s_0} T_i^2}{m}}$$

where m is the number of $\|T_i\|$ in the summation and a_{ASE} is a consistency constant that depends on the total number of effects. The ASE is a very efficient estimate of scale when there are only a few significant effects, but it breaks down when there are many significant effects.

The trimmed standard error is similar to an ANOVA-based method proposed by Berk and Picard (BP92), namely,

$$\hat{\sigma}_{\text{TSE}} = a_{\text{TSE}} \times \sqrt{\frac{\sum_{i=1}^{m} \|T\|_{(i)}^2}{m}}$$

where $\|T\|_{(1)} \leq \|T\|_{(2)} \leq ... \leq \|T\|_{(k)}$ are the order statistics of the absolute values of the effects, $m = \lfloor .6k \rfloor$, the smallest integer less than $.6k$, and a_{TSE} is a consistency constant that depends on the total number of effects.

[6] *Trimmed effects* are those remaining after the largest effects are removed according to some rule.

The TSE is equivalent to using the standard error from an ANOVA with the smallest 60% of the effects included in the error.

There are two situations where you might want to consider an alternative to the default method PSE. The first case is if you have a strong *a priori* belief that there are only a few significant effects. In this case, the adaptive standard error ASE seems to fare well because of its efficient (second stage) scale estimate. The TSE may also be useful in this case. The second case is where possibly as many as half of the effects may be significant. In this case, the PSE has the best chance of providing a good estimate of the error, but an estimate with a higher breakdown point may be required. Note that the PSE and ASE both use the median to provide a (first stage) estimate of scale, so if half of the effects are of even moderate size these estimates of scale will be too large. For $m = 0.6$ this argument also applies to the TSE. Development of a useful scale estimate is an active area of research, and we hope to have additional methods available for this case in the future. The best solution for now is to closely examine the Pareto plot and the half-normal plot and to use your common sense in deciding which estimated effects are of possible *practical* significance. It is, in fact, important to understand both the statistical and practical significance of effects in all applications.

Note that the test statistics `t(PSE)`, `t(ASE)`, and `t(TSE)` do not follow a known distribution, that is, you can't look up initial values in a Student's t-table. The critical values and p-values used in library dox are based on simulation results (HO94).

The results from the different error estimates are easily compared by asking for a Pareto plot with `all` error methods:

```
> pareto(buffer.fac, method = "all")
```

In figure 2.9, all but the trimmed standard error agree that there are four significant effects at the 5% level.

A further informal check is to examine the half-normal plot with the significant effects omitted; the points should look like a sample from a normal distribution, and a line through the origin with a slope given by the error estimate should fit the data. For this example, omit the four largest effects and look at the half-normal plot of the remaining effects:

```
> qqnorm(buffer.fac, method = "all", omit = 4)
```

Figure 2.10 indicates that σ_{PSE} estimates standard error of small effects well.

In many cases, the results are the same for all error estimates. Where the errors change dramatically, common sense will often indicate which error term is most appropriate. Active contrast plots (see section 2.2.4) and further analysis of a reduced model (see section 2.3) can also be used to help assess the consistency of the results for a given error method.

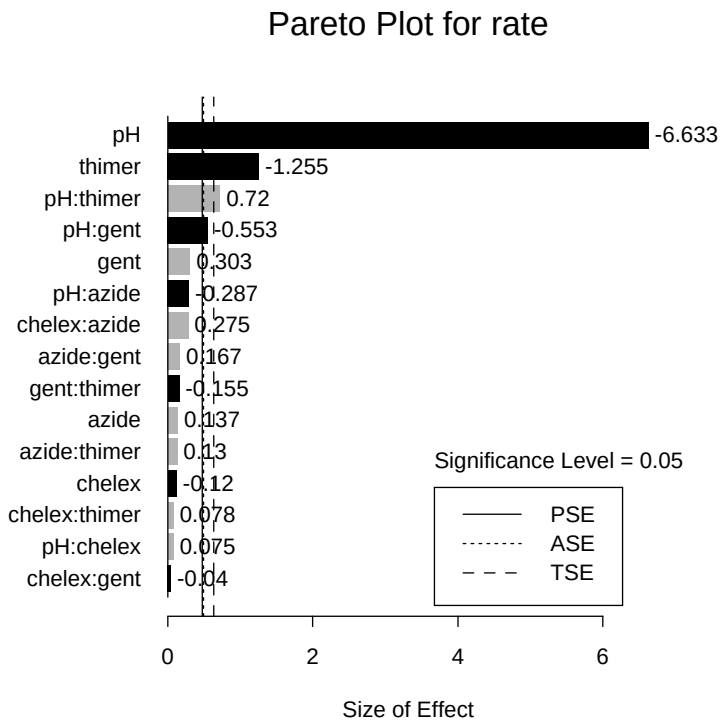

Figure 2.9: This Pareto plot compares different error methods for the degradation rate in the buffer experiment.

2.2.4 Bayesian Analysis

In the analysis of fractional factorial designs, you are often willing to accept the Pareto principle and assume that most of the effects are small, and that only a few of the factors (or interactions) are responsible for most of the observed variability in the response. This can be made into an explicit prior assumption; for example, you may believe that 40% of the true effects are nonzero. Then the remaining effects look like normal observations with some variance, σ^2. This assumption means that the observed effects would look similar to data where 40% of the observations have a large variance, $k^2\sigma^2$, and the rest are normal, $\mathrm{N}(0, \sigma^2)$, i.e., the effects may be thought to arise from a contaminated normal distribution:

$$(1 - \alpha)\mathrm{N}(0, \sigma^2) + \alpha\mathrm{N}(0, k^2\sigma^2).$$

where $\alpha(= 0.4)$ is the probability that an individual effect is "active," and k is a scale factor for the active effects (BM86).

If you know both α and k, you can use Bayesian methods to calculate the probability of any effect being "active" (nonzero), based on the data observed. The active

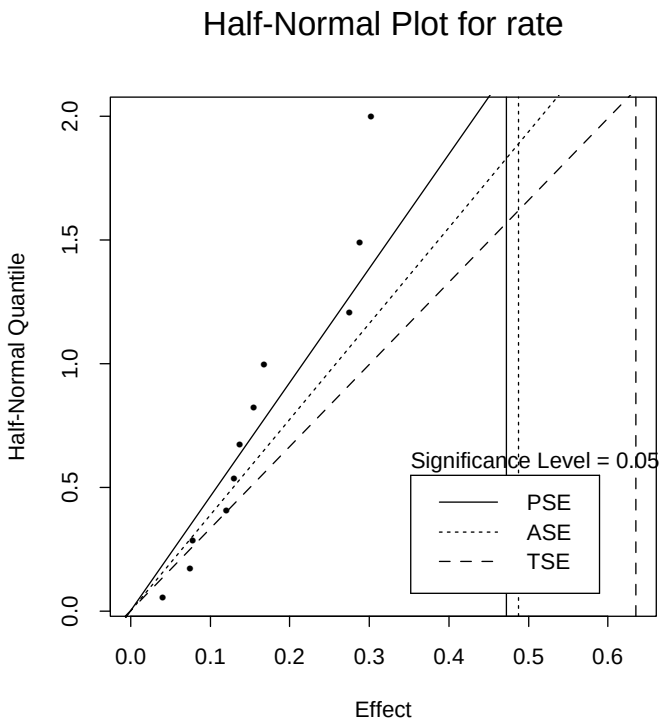

Figure 2.10: Half-normal plot of the "non-significant" effects, with slopes based on each of the error estimates. This plot illustrates the estimate of the error relative to the smallest effects.

contrast plot displays the posterior probability of an effect being active for a given α and k. Our recommended default values are $\alpha = 0.4$ and $k = 5$ (Haa89). Another useful set of values, given by Box and Meyer, is $\alpha = 0.2$ and $k = 10$ (BM86). The default settings are appropriate for the case of "many small effects" while the other are appropriate for the case of "a few large effects". These two cases could be interpreted as "intermediate to late" and "early" screening, respectively.

The active contrast plot for the storage buffer data is displayed by:

```
> acplot(buffer.fac)
```

Effects with probabilities greater than 0.5 are usually interpreted as active. So figure 2.11 indicates only the two main effects `pH` and `thimer` are active for $\alpha = 0.4$ and $k = 5$. In this case, the active contrast plot for the Box and Meyer settings $\alpha = 0.2$ and $k = 10$ is quite similar.

An alternative way to compute active contrast probabilities is the empirical Bayes

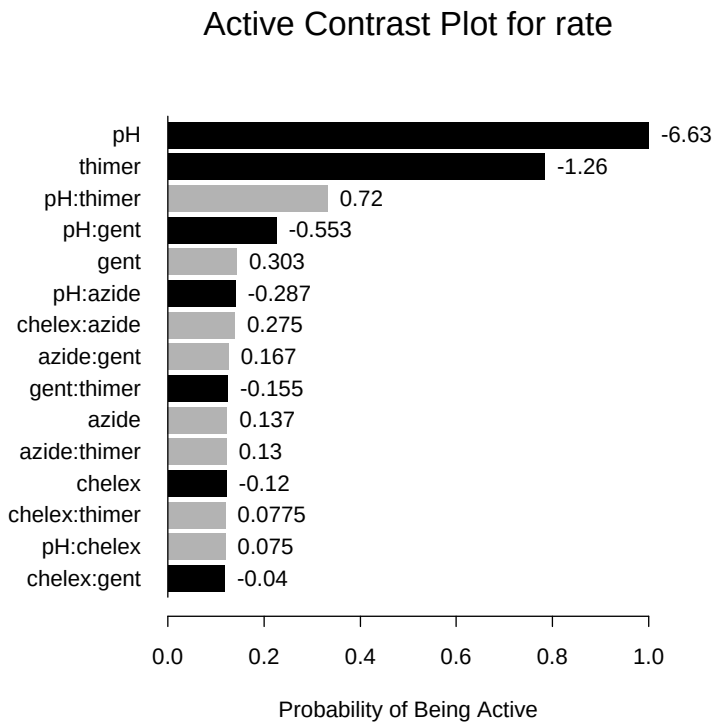

Figure 2.11: Active contrast plot for the storage buffer problem. For each effect the probability of that effect being active is plotted, as calculated from the parameters α and k. Typically, an effect is considered to be active if this probability is greater than 0.5.

method. If you haven't had enough experience with the process to guess appropriate values for α and k, the empirical Bayes method can provide (empirical) values of both α and k from the observed data. Another alternative is to specify how many effects you think are active based on your analysis so far—this determines α—and to let the empirical Bayes method estimate k based on the resulting model.

The default for the empirical Bayes procedure is to use the effects that are significant from the PSE method in a reduced model and to then estimate k from the overall F-ratio of this model. Since the summary output for the storage buffer data shows four effects are significant at the 95% confidence level based on the PSE method, this is the same as setting α=4/15=0.27. The estimated k is 18.69 in this case (this is the square root of the F-ratio for a reduced model containing these four effects). Compute and display the empirical Bayes probabilities:

```
> ebplot(buffer.fac)
```

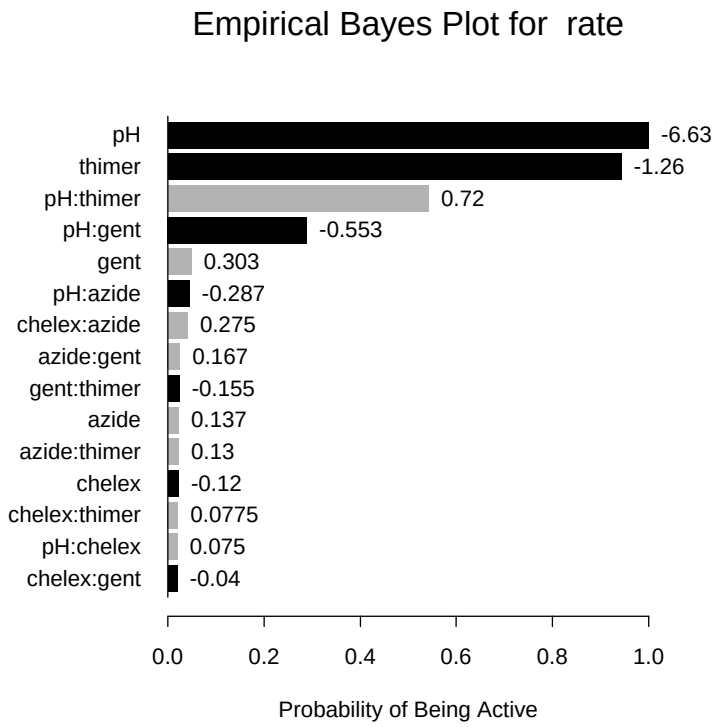

Figure 2.12: Empirical Bayes plot for effects selected by the PSE method at the 5% significance level. Like the active contrast plot, the probability of an effect being active is plotted. Empirical Bayes methods are used to compute values for the parameters α and k.

The empirical Bayes plot given in figure 2.12 lends support to the belief that `pH`, `thimer` and `pH:thimer` are significant. The effect `pH:gent` stands out from the rest of the effects but is not clearly significant.

The active contrast plot can be used to corroborate the tentative conclusions drawn from the Pareto, half-normal, and two-factor interaction plots. If `alpha` and `k` are specified, the posterior probabilities are independent of the error estimates in the analysis. In cases where it is not clear if certain effects are significant, it is often helpful to look at the active contrast plots with the standard settings described above along with any values for `alpha` and `k` which seem reasonable based on your experience. Occasionally it is hard to find values for α and k that gives the clear separation shown, for example, in figure 2.12. This can occur when there are very many significant effects or when there is not a very clear distinction between the null effects and the smallest of the possibly significant effects.

2.3 Fitting a Reduced Model

Based on the results of the analysis so far, you can usually identify which effects are significant. A reduced model containing just these effects provides a useful summary of the experiment. This empirical model may also be used to predict the response. Prediction of the response should only be carried out within the space defined by the levels of the experimental factors. However, if the target response is not achieved withing this space, the reduced (empirical) model may be used, with care, to suggest a region for further experimental investigation.

If the reduced model is to be used for prediction, it is important to examine the diagnostic plot.

The reduced model also provides an alternative estimate of the experimental error.

2.3.1 Specifying the Reduced Model

An extensive graphical analysis has already been conducted for the degradation rate data. One way to summarize the results of this analysis is that the `pH` and `thimer` effects are clearly significant, the `pH:thimer` effect is probably significant, and the `pH:gent` effect is possibly significant. You now may fit *reduced* model with just these factors and their interactions:

```
> buffer.redmod <- update(buffer.fac,
+   . ~ pH * thimer + pH * gent)
> summary(buffer.redmod)
Estimated Effects for Response: rate

Design Name: ff0516 (reduced model)
          Effect Std. Error |t(MSE)| P-value
       pH -6.630 0.165      40.10    <.001
   thimer -1.260 0.165       7.58    <.001
     gent  0.303 0.165       1.83    0.097
pH:thimer  0.720 0.165       4.35    0.001
  pH:gent -0.553 0.165       3.34    0.008

  Mean Standard Error of Effects =  0.165
         with  10  degrees of freedom.
  Experimental Error (RMSE) = 0.331
  R-squared =  0.994
  P-values calculated using the t-distribution.
```

The "." on the left hand side of the formula implies that the same response is to be used in the updated model as in the original `buffer.fac`.

First note that in addition to the largest four effects, the model contains also contains a fifth effect, `gent`, so that the model is "hierarchical." That is, if a two-factor interaction term is included in the model, both corresponding main effects are also included.

Because you have only five effects in the model, the remaining ten effects can be used to provide an alternate estimate of the experimental error and the standard error of the estimated effects. The estimated mean standard error (MSE) of the effects is 0.165, which seems in good agreement with the pseudo standard error estimate of 0.216. The experimental error, which is the usual root mean square error from an ANOVA, provides an estimate of the standard deviation of the response variable. This value should seem reasonable based on previous experience with the process. The percentage of variability in the response explained by the statistical model, or R-squared, is quite high at 99.4%, indicating that the other non-significant effects explain a very small portion of the total variability in the response. As expected, all of the terms included in the model are highly significant (except for `gent` which was included to obtain a hierarchical model).

As a final view of the reduced model you look at the Pareto plot with all of the effects and a new significance line drawn using the mean standard error from the reduced model:

```
> pareto(buffer.redmod)
```

Figure 2.13 shows that the significance lines for the PSE and MSE estimates agree quite closely. This is a good indication that the reduced model is consistent with the conclusions of the previous analysis. Note that you could have generated the reduced model with the function `fac.aov` but we chose to use the function `update`. Using `update`, instead of refitting the model with `fac.aov`, has the advantage that the standard error estimates PSE, TSE, and ASE from the saturated model are retained by the returned model. It also keeps the estimates of the effects from the saturated model so that meaningful Pareto and half-normal plots can be generated. These estimates from the saturated model, not present in the reduced model, are parenthesized in the Pareto plot.

2.3.2 Diagnostic Plots

Now that you have fit a reduced model, the model is no longer saturated, hence there are *residuals*. The residuals are the observed values of the response minus the predicted values. Our tests for significant effects are valid if the residuals approximately follow a normal distribution and do not show any unusual patterns. Diagnostic plots evaluate these assumptions and also provide assurance for using the submodel to predict the response over the space defined by the factor levels.

```
> plot(buffer.redmod)
```

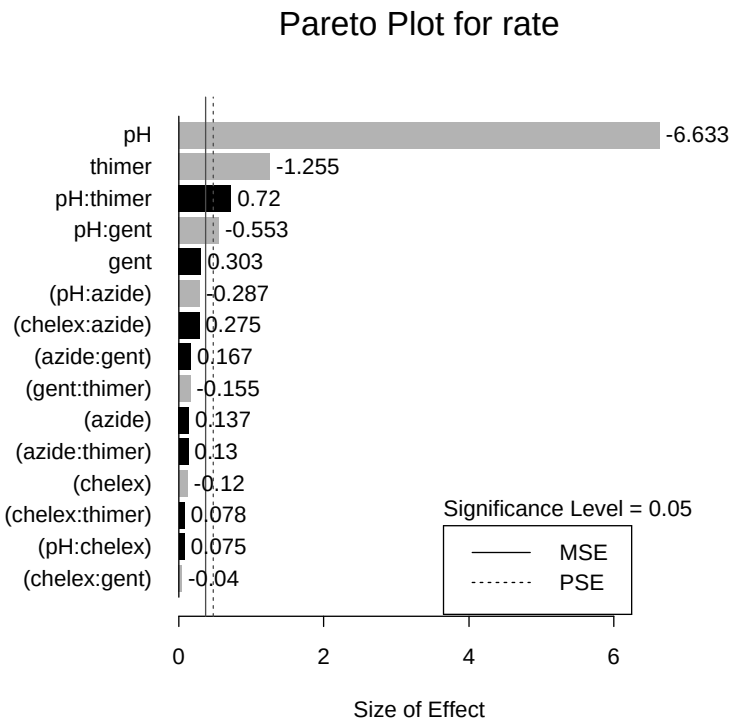

Figure 2.13: The Pareto chart for the reduced model showing significance lines for both the PSE and MSE estimates.

In the first of the diagnostic plots shown in figure 2.14 the observed degradation rates are plotted against the values predicted from the reduced model. The points lie close to a diagonal line indicating that the reduced model fits the observed data well over the range of the response. It is interesting to note that the responses are separated into two groups. The values with higher responses correspond to low pH and the values with lower responses correspond to high pH.

The second diagnostic plot (residuals versus run order) is useful in detecting any source of systematic bias in the response values. For example, suppose the measurement of degradation took a long time and the samples sat on the laboratory bench during waiting to be assayed. If additional degradation could be taking place while the samples are waiting to be measured, you would expect a downward drift in the residuals versus run order. In this example, the values were listed in a standard order rather than in run order so this plot is not very meaningful.

The bottom left hand plot in figure 2.14 shows the standardized errors versus the predicted values. If there is a "V" shaped pattern going from left to right, it suggests that the variance of the residuals is an increasing function of the response. There

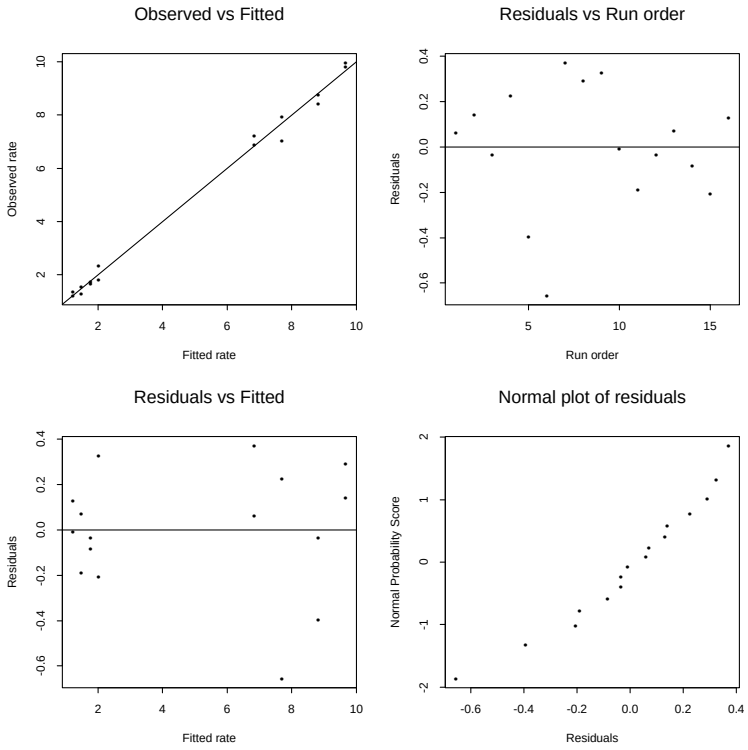

Figure 2.14: Diagnostic plots to evaluate the assumptions of the reduced model. The upper left plot shows the predicted versus observed values. The upper right hand plot shows the residuals versus run order. The lower left hand plot shows standardized residuals versus predicted values. The lower right hand plot shows a normal (QQ) plot of the residuals.

is some suggestion of increasing variance of the residuals with the mean in this example. A logarithmic or square root transformation may be useful in satisfying the assumptions of the analysis in this case; namely, that the residuals have constant variance. When there are four or more residual degrees of freedom, the standardized residuals follow the usual definition; namely,

$$r_i^s = \frac{r_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}$$

otherwise the ordinary residuals are simply divided by the standard error. See section 2.4, Transformation Analysis, for how to carry out a formal transformation analysis.

The last plot in figure 2.14 is a full-normal (QQ) plot of the residuals. Ideally, the residuals should all fall along a straight line. Patterns to watch out for are generally

"S" shaped curves which suggest that there may be one or more unusually large residual. Such residuals may correspond to outliers and should be identified and closely studied.

2.4 Transformation Analysis

It is not uncommon for a response to be much more variable at the high level settings than the low level settings. This tends to happen for counts and percentages, or when the response is related to duration. In some of these cases, it is appropriate to transform the response. The response on the transformed scale will often better satisfy the assumption that the error variance of the response is constant over the design. Another reason to use a transformation is that after a transformation of the response variable, the fitted model may be more parsimonious or simpler to interpret. For example, it may contain only main effects of factors and no interactions (see figure 2.5).

The Box-Cox transformation analysis summarizes the models for *power* transforms for the response, $y^{(\lambda)} = (y^\lambda - 1)/y^\lambda$, for a range of values of λ, where $\lambda = 0$ corresponds to the logarithm transform. Typically a range from $\lambda = -1$ (the reciprocal) to $\lambda = 1$ (the identity) is considered.

Notice in the first exploratory plot of the storage buffer data, figure 2.2, the rate seems to be a little more variable when it is higher. This pattern was seen again in figure 2.14. Transformation of the response may lead to a more constant variance. Results of a Box-Cox analysis for $-1 \leq \lambda \leq 1$ are given in figure 2.15:

```
> buffer.bc <- boxcox(buffer.fac, n.effects = 4)
> plot(buffer.bc)
```

For the transformation analysis, a reduced model must be specified, as given by the second argument, `n.effects = 4`, which specifies a model with the four largest terms in the untransformed (current) model. Alternatively, a model formula can be supplied. The mean square error from this reduced model is used to compute t-statistics for each of the terms, as displayed in the lower plot. Maximizing the log likelihood is equivalent to minimizing the residual sum of squares for the model: the first plot shows the log likelihood for $-1 \leq \lambda \leq 1$, indicating a 95% confidence interval for the optimal λ between about 0 and 0.75. Usually, you choose convenient transforms from this interval: in this analysis, possibilities are a log transform, at $\lambda = 0$ and square root transform, at $\lambda = 0.5$. The second plot shows the t-statistic of the 4 largest terms in the analysis of the transformed response, and the 5% significance band for the t-statistic. In this plot, you see that the t-statistic for `pH:thimer` lies inside the confidence band, so the model for `log(rate)` does not have a significant `pH:thimer` interaction.

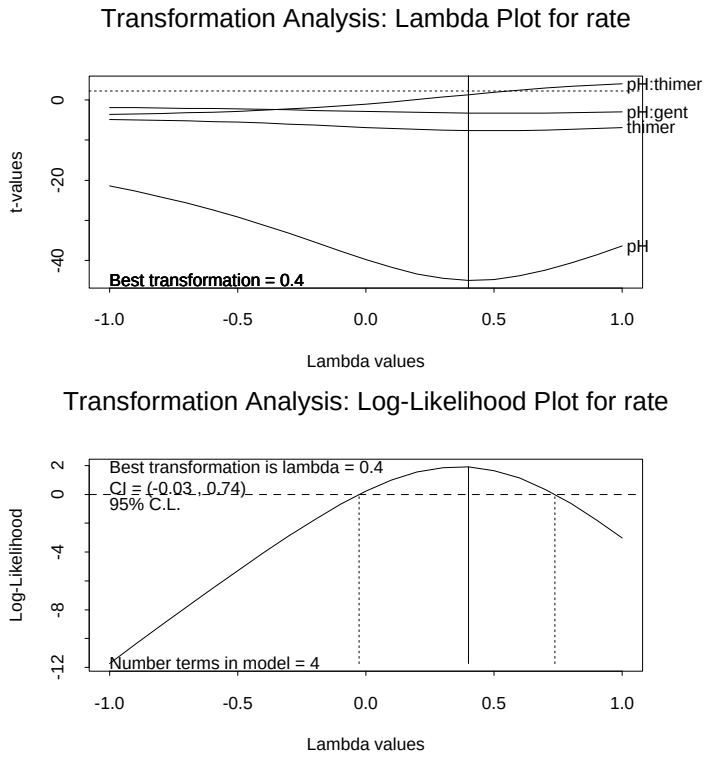

Figure 2.15: Transformation analysis of the degradation rate. The upper plot gives the log likelihood values, the lower, t-statistics from the analysis on the transformed scale.

Thus a choice of a square root or logarithmic transformation may achieve constant variance and/or provide a simpler model. To verify this, refit the reduced model on the transformed scale. Since the best transformation is closest to the standard square root transformation, consider it first:

```
> buffer.sqrt <-fac.aov(sqrt(rate)~pH*thimer+pH*gent,
+ buffer.df)
> summary(buffer.sqrt)
Estimated Effects for Response: sqrt(rate)

Design Name: ff0516
           Effect Std. Error |t(MSE)| P-value
       pH -1.6000 0.0367     43.60    <.001
   thimer -0.2760 0.0367      7.53    <.001
     gent  0.0256 0.0367      0.70    0.5
pH:thimer  0.0673 0.0367      1.84    0.096
  pH:gent -0.1190 0.0367      3.25    0.009
```

```
Mean Standard Error of Effects =  0.0367
       with  10  degrees of freedom.
Experimental Error (RMSE) = 0.0733
R-squared =  0.995
P-values calculated using the t-distribution.
```

Notice that in this model the `pH:thimer` interaction is not significant at the 5% level.

Now look at the same two-factor interactions as examined in the untransformed models:

```
> tfiplot(buffer.sqrt,~ pH:thimer + pH:gent)
```

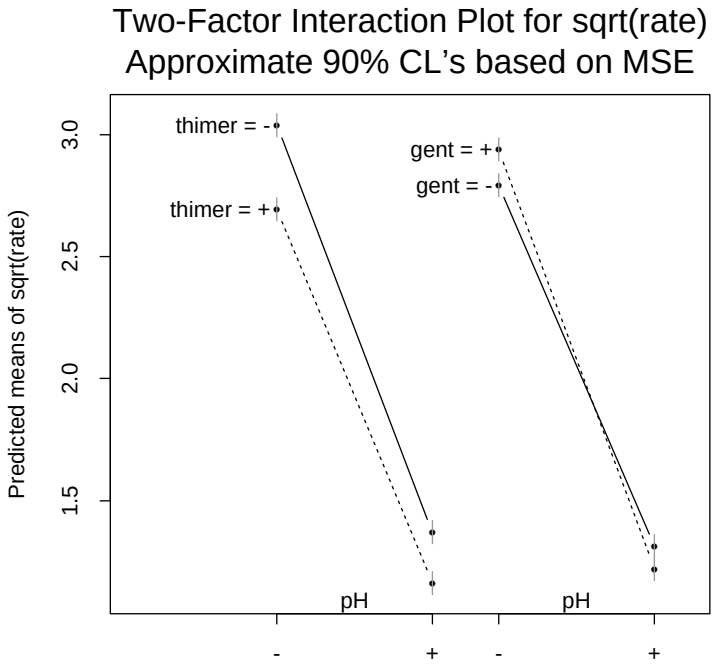

Figure 2.16: Two-factor interaction plots for the model of the square root of rate in the buffer experiment.

The two-factor interaction plot using the square root transformation shows that the lines for `thimer=-` and `thimer=+` are nearly parallel. As discussed above the

`pH:gent` is not transformable and hasn't been greatly affected by the transformation. Thus the square root transformation can be used to obtain a simpler, model with one fewer interaction.

Now, fit a model without `pH:thimer` and examine the diagnostic plots again:

```
> buffer.sqrtrm <- update(buffer.sqrt,
+ . ~ pH + thimer + pH * gent, buffer.df)
> plot(buffer.sqrtrm)
```

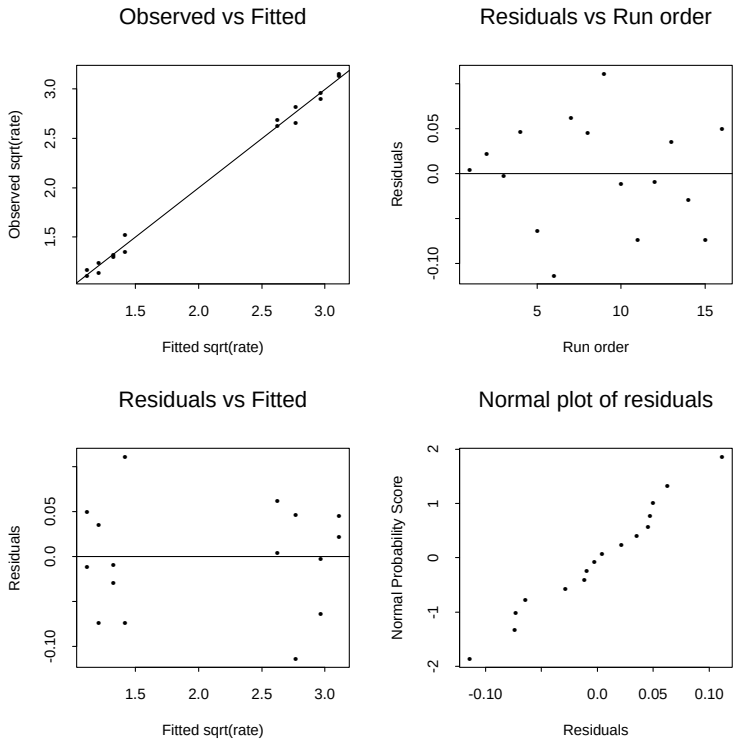

Figure 2.17: Diagnostic plots for square root response.

The residuals on the square root scale for the reduced model without the `pH:thimer` interaction are shown in figure 2.17. Most of the plots look similar to those shown in figure 2.14. The plot of standardized residuals versus predicted values shows that the variance is more stable across the range of the response. Thus the analysis on the transformed scale better satisfies the model assumptions.

As a final note, there is not much difference in the analysis between the log and the square root analysis. The log transformation makes the `pH:thimer` interaction even

less significant, but it tends to overcorrect for the nonconstant variance. The important point of the transformation is that it highlights the *proportional* nature of the `pH:thimer` interaction in that `thimer` slightly modifies the `pH` effect in proportion to the size of response obtained at each level of `pH`.

2.5 Conclusions

Fractional factorial designs provide a powerful way to screen many factors while using only a small number of experimental runs. Because of the way the runs are selected it is often possible to estimate all main effects and two-factor interactions independently. When fewer runs are chosen some confounding may be introduced, but it is known in advance which effects are confounded. Thus the experimenter may make a reasoned trade-off between experimental objectives and cost. The dox library provides a number of functions that make it easy to design fractional factorial experiments.

The analysis of fractional factorials provided in library dox is straightforward and primarily graphical. This allows you to quickly visualize the effects of the experimental factors on the response and provides useful material for presentation of the experimental results to others. The graphical methods include Pareto plots, normal plots, and two-factor interaction plots. Active contrast (Bayes) plots and transformation analysis are also available. An additional summary is provided by available by fitting reduced models and examining diagnostic plots. By combining this informative analysis with carefully selected designs and clear thinking about experimental objectives, the experimenter can quickly and reliably solve complex problems with limited resources.

Chapter 3

Response Surface Methods and Process Optimization

Response surface methods are used to study changes in a process response in terms of a few key experimental factors. Often response surface methods are used after an initial screening experiment. The screening experiment identifies the important factors affecting the response, and a follow-up response surface experiment studies the response over these important factors in more detail. The aim is to fit a second order polynomial surface to the response in terms of the experimental factors. The surface may then be used to reveal combinations of factor settings at which, say, the maximum yield of the process occurs. Also, if multiple responses are measured, regions of factor settings which, for example, yield is maximized *and* cost is minimized may be readily identified. For responses with a specific target value, the surface may be used to indicate a region of suitable operating conditions. From this region, manufacturing conditions can be chosen to give cheaper or faster production. Box, in (BD87), provides a comprehensive treatise on response surface methodology. In addition, an excellent review of response surface methods is given by Myers *et al.* (MKC89).

A diagnostic assay that includes a method for amplifying DNA known as Strand Displacement Amplification (SDA), developed at Becton Dickinson Research Center, uses two enzymes, Hinc II and Exo-Klenow. Of these, Hinc II is the more

expensive. In preparing the diagnostic assay kit, a response surface experiment is used to try to reduce the cost of the assay, possibly by reducing the amount of Hinc II used ($KDN^{+}93$).

This experiment is well-suited to response surface methods, since there are three relevant factors, time and the two enzymes, and much is already known about the assay. This knowledge includes a desired range of values for the response, and operating levels for the factors, that indicate satisfactory assay performance.

The response surface is constructed using a low-order polynomial, typically a second order, or quadratic model. A second order model can only capture simple curvature, but this is often sufficient as an approximation to the response near its optimum. The advantage of a quadratic model is that only a few settings are needed for each factor—typically only three to five different settings. This means we do not need a large number of runs. For example, a response surface for a three factor experiment requires only 14 runs, plus between two and four center points.

Most popular RSM designs are constructed by augmenting a fractional factorial design ('cube'), which may be used to estimate a linear surface, with design points that allow estimation of the quadratic terms. Center points are added to provide an estimate of the response variance at the center of the design. The default response surface design is known as a *central composite* design and includes a 'star' of points outside the cube placed in line with the center of each face. Each factor then has five different settings. By placing the star points actually on the cube one may reduce this to three settings per factor. This design is called a *face-centered cube* design.

For the strand displacement amplication (SDA) example, only three levels of each factor were considered and a face-centered cube design was used. Prior to this work the operational settings for SDA were 150 units of Hinc II, 5 units of Exo-Klenow, and a test time of 2 hours at 37 C. Since the aim was to reduce the amount of enzymes, the enzyme amounts given above were used as the high levels of the factors, and two lower levels were chosen. It was thought that if the amount of either enzyme was reduced, the test time may need to be increased to maintain the same signal, so 2 hours was chosen as the shortest time. The 18 runs used in the design are as follows:

	Hinc.II	Exo.Klenow	time
1	50	1	2.0
2	50	1	3.0
3	50	5	2.0
4	50	5	3.0
5	150	1	2.0
6	150	1	3.0
7	150	5	2.0
8	150	5	3.0
9	50	3	2.5
10	150	3	2.5
11	100	1	2.5
12	100	5	2.5
13	100	3	2.0
14	100	3	3.0
15	100	3	2.5
16	100	3	2.5
17	100	3	2.5
18	100	3	2.5

The first eight runs are a "cube" on the high and low levels of each factor, the next six are points at the center of each "face", and the last four, points in the center.

With this design, we can fit the default quadratic model in library dox:

$$y = \mu + a_1x_1 + a_2x_2 + a_3x_3 + b_1x_1^2 + b_2x_2^2 + b_3x_3^2 + c_1x_1x_2 + c_2x_2x_3 + c_3x_1x_3$$

where x_i are the three experimental factors, and a_i, b_i and c_i are coefficients of the model.

After the experiment has been run, the data are entered into library dox, ready for analysis. Briefly, the standard analysis procedure is to first fit the quadratic model and examine the residual plot for lack of fit. If the model seems adequate it may then be used to construct a response surface. Also, the fitted coefficients are studied using Pareto plots and printed summaries. The Pareto plot gives a quick visualization of the important factors and is useful in deciding which factors to use in the surface plots.

The surface is visualized using contour, surface, and image plots. The contour and image plots provide useful two dimensional views of the surface classified by two factors. The surface plot gives a three dimensional view of the response in terms of two factors. In each of these functions, by default, additional factors are set to their center value, but they may also be set to any other values as appropriate.

Lastly, numerical optimization is used to obtain an estimate of the optimum of the surface and the corresponding settings of the factors. This optimization may be done over the entire factor space or over a subspace defined by constraints on one or more factors. Also, joint optimization of more than one response in terms of

any two of the factors can be assessed informally by overlaying contour plots of two different responses.

3.1 Response Surface Designs

Four types of response surface designs are provided: central composite, face-centered cube, 3^k, and Box-Behnken designs.

The 3^k designs are not as efficient as the other designs, as they require many more runs. Any of the face-centered cube, Box-Behnken, and central composite designs are good candidates for a response surface experiment. The central composite designs[1] with star points placed on the sphere containing all the cube points, have the advantage that variances of the predictions from the model are functions only of the distance from the center of the design, due to design rotatability. Hence, for example, the variance of predictions at the design points are equal. In library dox all three of these preferred designs, by default, include replication of the center point which may be used to provide an estimate of the pure (experimental) error.

Note that response surface designs are used only with continuous factors, where it makes sense to smoothly interpolate between settings. For nominal factors, such as present and absent, the best levels are usually discovered in a screening experiment.

The SDA data set is available under the name `sda.df`. The face-centered cube design for the SDA experiment may also be generated using `rsm.design`:

```
> sda.design <- rsm.design(3, type = "fc", alpha = 1,
+   factor = list(Hinc.II = c(50, 150),
+       Exo.Klenow = c(1, 5), time=c(2,3)))
```

The first argument is the number of factors, `type="fc"` specifies a face-centered cube, `factor` gives the names and levels of the cube points of the design.

A worksheet with space for recording two response measurements resulting from each of the factor combinations run in the laboratory experiment can be generated as follows:

```
> worksheet(sda.design, responses = c("opt.dens", "vis"))
```

[1] Sometimes called star designs, because of the "star" shaped pattern of points at the largest and smallest factor settings.

3.2 Adding the Response

In the SDA experiment, two different responses are measured to assess the assay results: a visual score between 1 and 4, and a measure of optical density. In both cases, a higher value indicates better DNA amplification and a more promising assay.

The two responses, visual score (`vis`) and optical density (`opt.dens`) are added to the design:

```
> sda.vis <- c(1.5,2.5,1.5,3,2,3.5,2.5,2.5,1.5,3,
+     2.5,2.5,2.5,3.5,3,3.5,2.5,2.5)
> sda.od  <- c(10.45,23.24,7.00,43.00,28.62,72.66,35.02,
+     96.87,7.56,50.88,37.28,37.94,29.01,36.64,39.32,
+     63.36,28.88,50.87)
> sda.df <- cbind(sda.design, vis = sda.vis,
+      opt.dens = sda.od)
```

Alternatively, if the experimental design and response measurements are already available in a file, read in these data and convert them to a response surface design, specifying which columns are factors:

```
> sda.df <- read.table("sda.dat")
> sda.df <- as.rsm.design(sda.df, rsm.factors = 1:3)
```

Plot the response data in terms of the experimental factors:

```
> plot(sda.df)
```

The resulting plots are given in Figure 3.1. The optical density increases with both enzyme amounts and time indicating a possible useful tradeoff. In particular, the visual score clearly improves with `time` and `Hinc.II`, but is low at the low level of `Hinc.II` and `time`.

3.3 Coding of the Factors

The convention in fitting response surface models is to fit the surface using standardized factors. This involves recoding the experimental factors to the *standard coding*, with the center point 0, and cube points 1 and -1. For the factor `Hinc.II`, for example, the standard coding is:

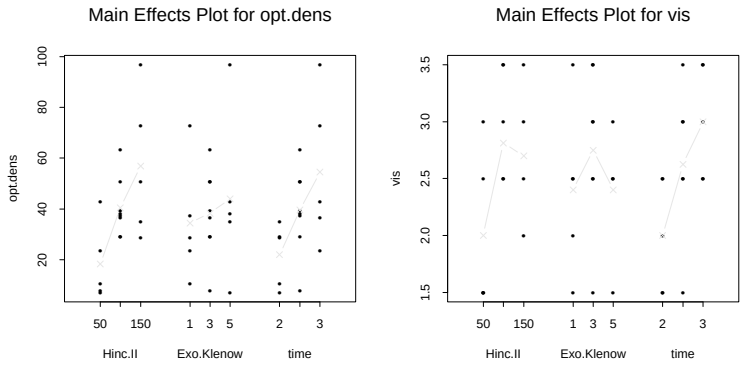

Figure 3.1: Plots of the individual and mean response for the experimental factors `Hinc.II`, `Exo.Klenow` and `time`.

```
Hinc.II  Standard Coding
     50   -1
     50   -1
     50   -1
     50   -1
    150    1
    150    1
    150    1
    150    1
     50   -1
    150    1
    100    0
    100    0
    100    0
    100    0
    100    0
    100    0
    100    0
    100    0
```

The reason for this is that on the standardized scale the columns of the design matrix are orthogonal, so the coefficients of the model are independent. As a result, each of the statistics in the summary may be interpreted independently, and if terms are dropped from the model, the remaining coefficients are unchanged. Translating the coefficients of the standardized model to the original factor levels involves a simple linear transformation.[2]

[2]For the users' convenience, `print` prints coefficients for both the original and coded (scaled) factors, and `summary` prints coefficients for the coded factors.

3.4 Fit the Response Surface

The response surface is fitted to the response optical density via a linear model function `rsm.lm`

```
> sda.od.rsm <- rsm.lm(sda.df, response = opt.dens)
```

By default, this call fits the full quadratic model, as generated by the following formula:

```
> formula(sda.df, response = opt.dens)
opt.dens ~ (Hinc.II + Exo.Klenow + time)^2 + Hinc.II^2 +
        Exo.Klenow^2 + time^2
```

First, we check the fit of the model by looking at the diagnostic plots of the residuals. A residual is the difference between the observed response and the fit of the quadratic surface at a given design point. If the fit is poor, the residual is large:

```
> plot(sda.od.rsm, run.order =  c(7, 6, 10, 11, 14,
+ 8, 4, 13, 9, 18, 16, 5, 15, 1, 17, 3, 12, 2))
```

While the model does not fit the data very closely, there are no systematic departures and no major problems regarding the usual second order assumptions of the model.

Now summarize the model:

```
> summary(sda.od.rsm)
Estimated Effects for Response: opt.dens

Summary for the scaled coefficients
                    Coef.   Std. Error t_value  P-value
       (Intercept)  42.00     6.6       6.300    <.001
           Hinc.II  11.00     5.3       2.000    0.08
         Exo.Klenow  -4.00     5.3      -0.740    0.48
              time   7.50     5.3       1.400    0.20
          Hinc.II^2  -8.70    10.0      -0.850    0.42
      Exo.Klenow^2  -0.28    10.0      -0.027    0.98
           time^2  -5.10    10.0      -0.490    0.63
Hinc.II:Exo.Klenow  -9.10     6.0      -1.500    0.17
      Hinc.II:time  -3.80     6.0      -0.630    0.55
   Exo.Klenow:time  -5.80     6.0      -0.970    0.36
```

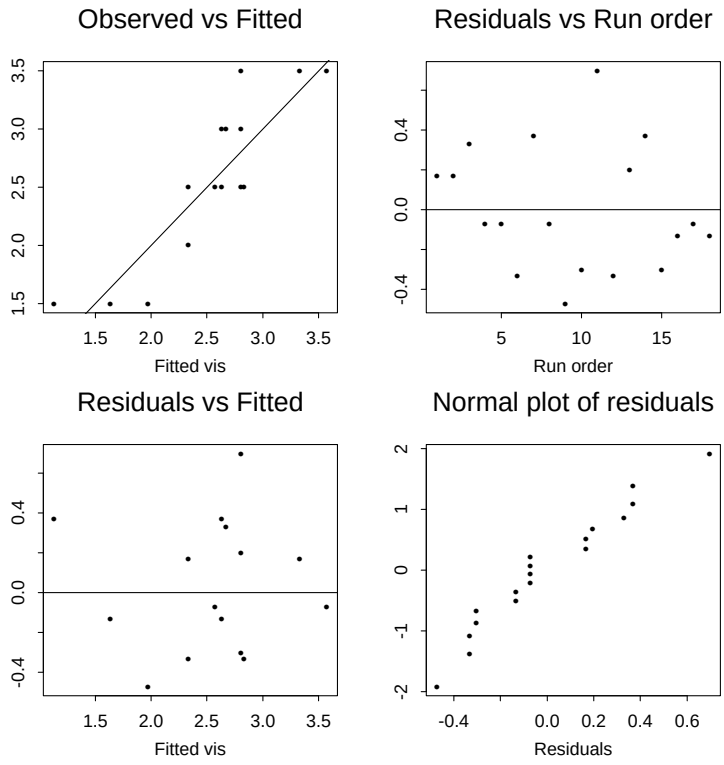

Figure 3.2: Residual plots for the quadratic model fitted to the response Optical Density.

```
Experimental Error (RMSE) = 17 on 8 degrees of freedom
R-Squared = 0.61
```

The Pareto plot displays the relative size and sign of the t-statistics given in the summary table and is useful for quick visualization of the order of importance of the effects:

```
> pareto(sda.od.rsm)
```

The vertical line in the Pareto plot, Figure 3.3, indicates the 10% significance level cutoff for the t-statistic based on the residual mean square estimate of experimental error. Although no factors appear to have a dramatic effect on optical density, `Hinc.II` and `time` appear to be most important. Also, the sizes of the two interactions with `Exo.Klenow` indicate that the effects of both these factors are modified somewhat by `Exo.Klenow`.

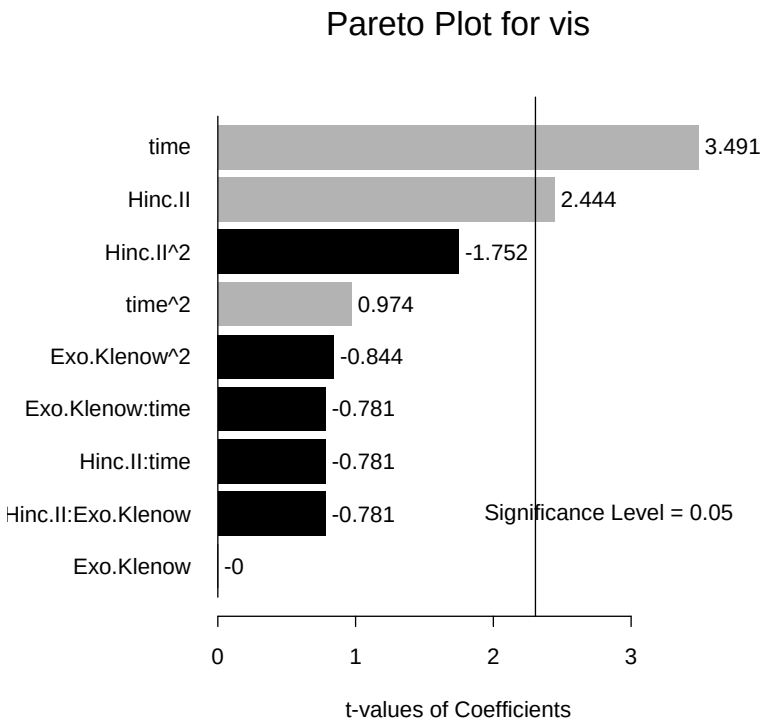

Figure 3.3: The Pareto plot for optical density, displaying the t-statistics for the model.

Contour, surface, and image plots help to visualize the response surface. These plots display the fitted surface in terms of two experimental factors. The Pareto plot (Figure 3.3) indicates `Hinc.II` and `time` are the most important factors, so fitted surface plots with the value of `Exo.Klenow` fixed at its center value, 3, are first constructed.

```
> contour(sda.od.rsm, vary=list(Hinc.II="v",time="v",
+ Exo.Klenow = 3))
> surface(sda.od.rsm, vary=list(Hinc.II="v",time="v",
+ Exo.Klenow = 3))
> image(sda.od.rsm, vary=list(Hinc.II="v",time="v",
+ Exo.Klenow = 3))
```

These plots are given in Figure 3.4 and show that by increasing the time to three hours, we can dramatically reduce the amount of Hinc II, without compromising the optical density response of the assay. For example, the contour plot shows that an `Opt.dens` response of 35 units may be obtained by using about 140 units of

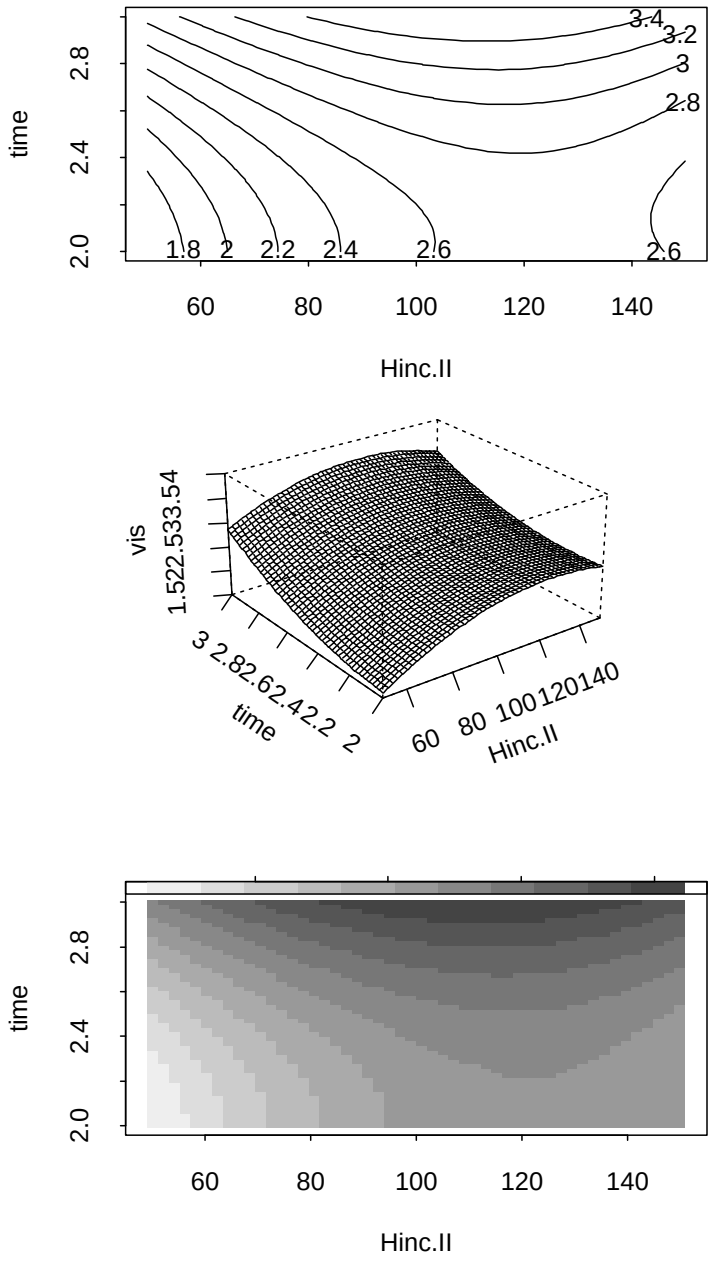

Figure 3.4: Contour and surface plots of the surface fitted to optical density, with Exo-Klenow held at 3 units.

Hinc II and a reaction time of about 2 hours, or by using just 70 units of Hinc II and a reaction time of 3 hours.

Now the reader can repeat the same analysis sequence on the visual score response. Fit the surface:

```
> sda.vis.rsm <- rsm.lm(sda.df, response = vis)
```

The reader can verify, using diagnostic plots, that the model is adequate. Again, the Pareto plot shows that `Hinc.II` and `time` are important effects on the visual score, as was the case for optical density.[3] We again plot the response surface in terms of `Hinc.II` and `time`. To examine the effect of `Exo.Klenow` along with this, we construct a sequence of contour plots at three different levels of `Exo.Klenow`, keeping the same contour lines in all plots. This simultaneous investigation of the factor effects is also recommended for the `opt.dens` response given the possible interactions of `Exo.Klenow` with the other factors suggested in Figure 3.3.

```
> lvls <- seq(1.6,3.6,.2)
> contour(sda.vis.rsm, vary = list("Hinc.II" = "v",
+     time = "v", Exo.Klenow = 1), levels = lvls)
> contour(sda.vis.rsm, vary = list("Hinc.II" = "v",
+     time = "v", Exo.Klenow = 3), levels = lvls)
> contour(sda.vis.rsm, vary = list("Hinc.II" = "v",
+     time = "v", Exo.Klenow = 5), levels = lvls)
```

As noted in Figure 3.4, the visual score response may be maintained for a reduced amount of `Hinc.II` by increasing `time` (Figure 3.5). Also, Figure 3.5 shows the surface is steepest for `Exo.Klenow = 1`, as the contour lines for the first plot are closer together than the plots for `Exo.Klenow = 3` and `5`. This is not a dramatic effect however, and it appears that the `Exo.Klenow` setting may be comfortably reduced. In the next Section, `optim` is used to estimate the maximum of the visual score.

3.5 Optimization

Frequently, response surface methods are used to find "optimal" operating conditions, i.e., settings of the factors for which the response reaches a maximum or minimum. Contour and surface plots allow visualization of the location of such an optimum. The function `optim` computes where the minimum or maximum value of the response surface occurs. For example, the maximum visual score is obtained as

[3] Even though the visual score is discrete, we treat it as a continuous response in this analysis and keep this in mind when interpreting residual plots and the like.

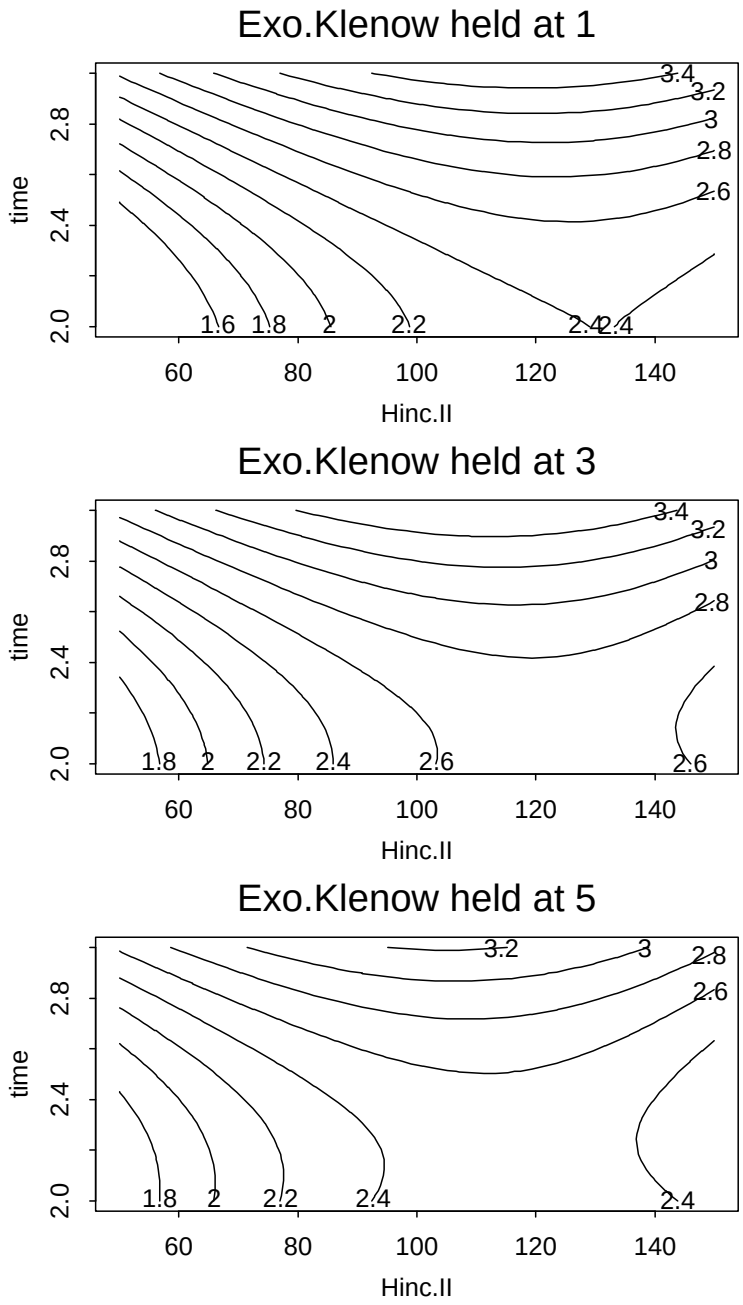

Figure 3.5: Contour plots of the visibility index surface for values of Exo-Klenow of 1, 3, 5.

```
> optim(sda.vis.rsm)
Convergence has been reached

  Hinc.II Exo.Klenow time    vis
 113.8967   2.311882    3 3.6242
( relative function convergence )
```

Notice the maximum of 3.6 occurs on the boundary, `time = 3`. By default, the optimization is confined to the range of the experimental factors. Also the value or range of any of the factors can be further constrained in the optimization. For example, if the amount of `Hinc.II` was restricted to be less than 80, `optim` can then compute the new constrained maximum:

```
> optim(sda.vis.rsm, constrained = list(Hinc.II = c(0,80)))
Convergence has been reached

 Hinc.II Exo.Klenow time      vis
      80   2.676924    3 3.410343
( relative function convergence )

Variables were constrained as follows:
         Hinc.II  = [ 0 , 80 ]
```

While the maximum of the fitted visual score occurs for a relatively high value of `Hinc.II`, the contour plot shows the function is fairly flat near the maximum at a time of 3 hours. It seems we could use less `Hinc.II`, and still achieve a visual score of around 3.4.

3.5.1 Optimization of Multiple Responses

The two responses in the above example measure a similar performance criterion, namely the SDA assay signal. In some applications multiple responses may measure two quite different response attributes of a process. In the above example, increased time may incur cost, so a possible response would measure the cost of the assay in addition to its performance. The goal of the joint analysis may then be to minimize cost and maximize signal, or at least to minimize cost for a fixed desirable signal. A graphical solution to this joint optimization problem is to overlay contour plots of different responses at the same factor settings:

```
> contour(sda.vis.rsm, vary = list("Hinc.II" = "v",
+   time = "v", Exo.Klenow = 2.3), main = "")
> contour(sda.od.rsm, vary = list("Hinc.II" = "v",
```

```
+   time = "v", Exo.Klenow = 2.3), add = T, lty = 2,
+    main =
      "Contour plot for Visual Score and Optical Density")
> legend(par("usr")[1],par("usr")[4], lty=c(1,2),
+ legend.names = c("Visibility", "Optical Density")
```

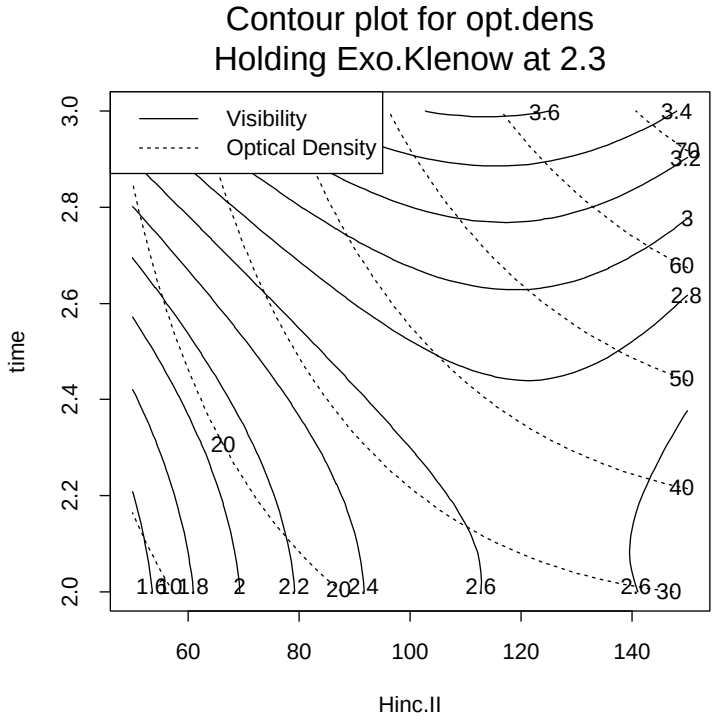

Figure 3.6: Contour plot with the fits for visual score and optical density overlaid.

In this case the two responses both measure the assay signal. Figure 3.6 shows the visual score is flat across a wide range of `Hinc.II` values for `time` close to 3 hours. A similar reasonable `opt.dens` response may be obtained for `Hinc.II` at around 140 units and `time` of 2 hours, and for `Hinc.II` at around just 70 units and `time` of 3 hours. Thus both responses indicate satisfactory assay signal at around 70 units of `Hinc.II` for a 3 hour assay `time`. As noted above, this type of plot may be much more informative for multiple responses measuring quite different process performance criteria.

3.6 RSM Problems with Factors on the Log Scale

In many industrial applications it is natural to consider experimental factors on the log scale. For example the concentration of a chemical additive may be considered as a solution percentage and this may be most appropriately studied on the log scale. An example of this application is currently awaiting publication approval and will be included in the main release.

3.7 Adding New Design Points to Fractional Factorial Screening Design

In industries where the raw materials for experiments are expensive, it may not be possible to repeat runs from an earlier screening design. Instead, the response surface may be modeled by adding "star" and center points to an earlier fractional factorial design. In library dox the fractional factorial is converted to a response surface design by:

```
> convert.to.rsm(inhib.df[,c("pH","gent","thimer")],
+ numeric.levels = list(raddos=c(100,200), gent = c(5,10),
+ thimer = c(100, 200)))
```

The design returned is a central composite design. The "cube" portion is the factors of the factorial design converted into numeric variables using the numeric levels given. Center and star points are added to the cube design, as in `rsm.design`.

3.8 Summary

Response surface designs are used to study the effects of a few key factors on the response(s). such designs typically have factors with between three ad five levels and allow for fitting of a second order polynomial model or response surface. Based on the response surface model, contour and surface plots can be used to study the behavior of the response. The optimum value of the response and the corresponding factor settings can be determined by numerical methods. Response surfaces fro two different responses can be compared by overlaying their contour plots. Thus, response surfaces provide a powerful means of optimizing and understanding process response over key manufacturing factors.

Chapter 4

Robust Design Methods

Like fractional factorial designs, robust design methods[1] are used to improve product and process performance in industrial applications. Robust design methods focus on improving the consistency of product performance as well as the average performance. In a robust design approach, the process mean and variance are studied simultaneously to find settings of the experimental factors that produce minimal variance at the desired process mean. In contrast, traditional fractional factorial design and analysis assume that variation is the same over the range of factor settings, and study only the process mean. library dox provides design and analysis tools for implementing robust design methods including the Taguchi method. Excellent reviews of the Taguchi method are given by both Kacker and Phadke (Kac85; Pha89).

The goal of industrial research is to develop a product that is robust to the conditions experienced in manufacturing and customer environments. Using robust design allows technology to be transferred from research to development, from there to manufacturing, and finally to the customer, with little change in performance.

The fundamental principle of robust design is to improve the quality of a product by

[1]The ideas of this chapter relate to those popularized by Taguchi's work in industrial design. However, because we only loosely follow his approach, we prefer to use the term "robust design" to describe the analysis methods we present.

minimizing the effect of causes of variation without actually eliminating the causes. This is achieved by finding settings of "control" factors for which the process is minimally sensitive, or robust, to the effects of "noise" factors. A control factor is an experimental factor that can be freely controlled by the experimenter, e.g., the speed of injection of a polymer in an injection molding process. A noise factor is a factor that cannot be controlled easily or economically in the customer's environment, e.g., ambient temperature in the manufacturing facility. As a result of a robust design experiment, the control factors may be set at levels where the process response is consistent over the space defined by the noise factors. At these settings the process is more stable and consequently the quality of the product is improved: since the response changes very little over a range of operating conditions, it will always be close to target.

In the following example a robust design was used to study the factors affecting shrinkage of products made by injection molding as reported by Engel in (Eng92).Seven control factors were chosen:

A : cycle time
B : mold temperature
C : holding pressure
D : cavity thickness
E : holding time
G : injection speed
H : gate time

Each of these factors may affect the amount of shrinkage, and their settings are easily altered within an injection molding cycle.

Three noise factors were selected: these are difficult to control in the manufacturing cycle, so we want the process to be insensitive to their values:

M : percentage regrind
N : moisture content
O : ambient temperature

A robust design experiment aims to identify settings of the control factors where the response is both "good" *and* unaffected by the noise factors. For manufacturing of the molded parts, the goal is to stabilize the shrinkage at 2.5% across the settings of percentage regrind, moisture content of the injected material and ambient temperature.

Robust design methods often use *orthogonal arrays*. The important property of an orthogonal array is that main effect estimates are all independent of each other. The fractional factorial designs discussed in chapter 2 are all orthogonal arrays. Taguchi also introduced orthogonal arrays that allow several factors to be studied at three levels. These designs employ a minimal number of runs for estimating effects of a relatively large number of control factors and may be obtained via the function `oa.design`. For example, the popular L_{18} orthogonal array, which has one factor

at two levels and seven factors at three levels may be obtained as follows:

```
> oa.design(c(2,rep(3,7)))
   A B C D E G H I
 1 - - - - - - - -
 2 - - 0 0 0 0 0 0
 3 - - + + + + + +
 4 - 0 - - 0 0 + +
 5 - 0 0 0 + + - -
 6 - 0 + + - - 0 0
 7 - + - 0 - + 0 +
 8 - + 0 + 0 - + -
 9 - + + - + 0 - 0
10 + - - + + 0 0 -
11 + - 0 - - + + 0
12 + - + 0 0 - - +
13 + 0 - 0 + - + 0
14 + 0 0 + - 0 - +
15 + 0 + - 0 + 0 -
16 + + - + 0 + - 0
17 + + 0 - + - 0 +
18 + + + 0 - 0 + -

Orthogonal array design with 2 residual df.
Using columns 1, 2, 3, 4, 5, 6, 7, 8
    from design oa.18.2p1x3p7
```

Constructing a robust design in library dox involves choosing a design for both the control and noise factors. Typically an orthogonal array is chosen for the control factors, and a (fractional) factorial for the noise factors. In this setting a robust design is constructed by combining each point in the noise array with each combination of the control array factors. While this involves a lot of experimental runs, it may provide a thorough understanding as to how variability due to the noise factor is affected by settings of the control factors. Under this approach, the design on the control factors is often called the "inner" array, and the design on the noise factors the "outer" array. The robust design method allows us to study the variance of the response over the noise array at each point of the control array.

The 32-run design chosen for the shrinkage experiment is:

							−	+	+	−	M
							−	+	−	+	N
A	B	C	D	E	G	H	−	−	+	+	O
−	−	−	−	−	−	−					
−	−	+	−	+	+	+					
−	+	−	+	+	−	+					
−	+	+	+	−	+	−					
+	−	−	+	−	+	+					
+	−	+	+	+	−	−					
+	+	−	−	+	+	−					
+	+	+	−	−	−	+					

Once the response data have been collected, library dox is used to view and analyze the data. The goal of the statistical analysis is to identify those control factors that affect the mean response, and those that affect the variance of the response. Settings of the control factors that satisfy the dual requirements of achieving optimal response and small variance over the noise factors may then be obtained. Selecting good settings is not a cut and dried procedure: often there are several possible settings, and cost and convenience determine the final choice. We will illustrate some approaches to selecting suitable control factor settings in the following sections.

The first step in the statistical analysis is to summarize the experimental data over the noise array at each point of the control design. These summaries are often referred to as performance criteria and they characterize the mean and/or variance of the response over the noise array. There is ongoing debate regarding the choice of suitable performance criteria. Taguchi methods advocate using *signal-to-noise* ratios as performance criteria and different signal-to-noise ratios are recommended depending on whether the goal is to maximize, minimize or achieve a certain target value for the response (see section 4.1.3, Signal-to-Noise Ratios Explained, for further details). Box, in (Box88), presents some criticisms regarding the use of signal-to-noise ratios and recommends analysis of the mean and standard deviation of the response, often on some transformed scale. In order to facilitate each type of analysis in library dox, we construct both the Taguchi signal-to-noise ratios and the mean and standard deviation for the response on both raw and transformed scales.

In library dox, each performance criterion is treated as a response in an analysis of control factor effects. We use the graphical fractional factorial methods of chapter 2, Fractional Factorial Designs, to identify the factors that affect each response. Using this analysis approach, factor settings are selected that achieve the maximal, minimal or target response, as the case may be.

In summary, the analysis of robust designs in library dox usually involves the following steps: first create control and noise designs and combine them into a single robust design frame. Enter the experimental response and add them to the design frame. These can then be examined graphically, displayed by each factor in turn. Next, compute and plot performance criteria for the analysis: by default library dox computes the mean and standard deviation on raw and transformed scales, and an

appropriate signal-to-noise ratio. For each performance criterion then conduct a graphical analysis: compute the effects, examine the Pareto plot, half normal plots and active contrast plots as described in chapter 2. Based on these plots, the best settings for the process may then be selected.

4.1 Choosing a Robust Design

The most important part of designing an experiment is actually outside the scope of this software. This is the process of selecting informative response variables, identifying important control and noise factors, and choosing good settings for the factors. These choices, which ultimately determine the success of the experiment, can only be guided by experience and knowledge of the process or product. A good summary of the issues affecting these choices is found in Phadke (Pha89) and Coleman and Montgomery (CM93).

To construct a robust design using `robust.design`, choose a fractional factorial or orthogonal array for both the control and noise factors[2]. The number of runs in a robust design experiment tends to become large quickly, so minimal designs (where only main effects can be estimated) are usually chosen. These may be generated by `fac.design`, `design.digest` and `oa.design`. The control and noise designs are then combined to form the robust design.

For the injection molding experiment, the combined seven-factor control design and the three-factor noise design are available in the data set named `mold.df` but these can also be generated and combined using `robust.design`:

```
> cont.des <- oa.design(rep(2,7), min.resid.df=0)
> nois.des <- fac.design(rep(2,3), c("M","N","O"),
+   fraction=1/2)
> mold.des <- robust.design(control = cont.des,
+   noise = nois.des)
```

This gives the 32-run experiment of the previous section—eight runs in the control design, each repeated at the four points of the noise design:

```
> summary(mold.des)
Pattern of noise over control factors

                  - + + - M
```

[2]Technically, the noise array is chosen to allow consistent estimation of the sensitivity of the response to the noise factors at the control factor settings. The points in the noise space are often chosen to follow a fractional factorial design but may also be chosen to simply cover the noise space.

```
                - + - + N
  A B C D E G H - - + + O
1 - - - - - - - -
2 - - + - + + +
3 - + - + + - +
4 - + + + - + -
5 + - - + - + +
6 + - + + + - -
7 + + - - + + -
8 + + + - - - +
```

4.1.1 Randomizing the Design

In any experimental design, it is important to randomize the order of the runs in the design, so that, for example, time of day and position effects are not confused with the effects of the experimental factors. In robust design it may not always be feasible to completely randomize the design: for instance, if it is difficult to change the levels of the noise factors, the randomization is within the levels of the noise design. The function `randomize.design` is used with `restrict` before the design is printed as a worksheet for recording the results.

```
> mold.rdes <- randomize.design(mold.des,
+   restrict = c("M","N","O"))
> worksheet(mold.rdes, response = "Shrinkage", graphics = T)
```

A work sheet appears in the graphics window, as in figure 4.1, with the randomization restricted to be within levels of the three noise factors.

Once the experiment has been run at the 32 different settings, enter the measured values obtained for the response and add them to the design (here we assume the responses are in the experimental run order):

```
> response <- c(2.2, 4, 2.1, 0.5, 2, 2, 3, 0.3, 2.1, 1.9,
+   3.1, 1.9, 1.9, 2.5, 4.2, 3.1, 4.6, 3, 1.9, 0.4, 2.3,
+   2.7, 1, 1.8, 2, 0.3, 2.3, 3.1, 1.8, 2.2, 2.8, 3)
> mold.rdf <- cbind(mold.rdes, shrink = response)
> mold.df <- sort.design(mold.rdf)
> summary(mold.df)
Response:  shrink

                -   +   +   -   M
                -   +   -   +   N
  A B C D E G H -   -   +   +   O
1 - - - - - - - 2.2 2.3 2.3 2.1
```

	A	B	C	D	E	G	H	M	N	O	Shrinkage
1	-	-	-	-	-	-	-	-	-	-	
2	+	+	-	-	+	+	-	-	-	-	
3	+	-	+	+	+	-	-	-	-	-	
4	-	+	-	+	+	-	+	-	-	-	
5	+	+	+	-	-	-	+	-	-	-	
6	-	+	+	+	-	+	-	-	-	-	
7	+	-	-	+	-	+	+	-	-	-	
8	-	-	+	-	+	+	+	-	-	-	
9	-	-	-	-	-	-	-	-	+	+	
10	+	+	-	-	+	+	-	-	+	+	
11	+	-	-	+	-	+	+	-	+	+	
12	+	+	+	-	-	-	+	-	+	+	
13	-	+	+	+	-	+	-	-	+	+	
14	-	-	+	-	+	+	+	-	+	+	
15	+	-	+	+	+	-	-	-	+	+	
16	-	+	-	+	+	-	+	-	+	+	
17	+	+	-	-	+	+	-	+	-	+	
18	+	-	-	+	-	+	+	+	-	+	
19	+	+	+	-	-	-	+	+	-	+	
20	-	+	-	+	+	-	+	+	-	+	
21	-	-	-	-	-	-	-	+	-	+	
22	-	-	+	-	+	+	+	+	-	+	
23	+	-	+	+	+	-	-	+	-	+	
24	-	+	+	+	-	+	-	+	-	+	
25	-	+	+	+	-	+	-	+	+	-	
26	-	-	+	-	+	+	+	+	+	-	
27	-	-	-	-	-	-	-	+	+	-	
28	+	-	+	+	+	-	-	+	+	-	
29	+	+	+	-	-	-	+	+	+	-	
30	+	+	-	-	+	+	-	+	+	-	
31	-	+	-	+	+	-	+	+	+	-	
32	+	-	-	+	-	+	+	+	+	-	

Figure 4.1: A worksheet for the molding experiment.

```
2 - - + - + + + 0.3 0.3 2.7 2.5
3 - + - + + - + 0.5 2.8 0.4 3.1
4 - + + + - + - 2   2   1.8 1.9
5 + - - + - + + 3   3   3   3.1
6 + - + + + - - 2.1 3.1 1   4.2
7 + + - - + + - 4   2.2 4.6 1.9
8 + + + - - - + 2   1.8 1.9 1.9
```

4.1.2 Computing Performance Criteria

Compute the performance criteria over the noise array at each control point. The function `robust.sn` returns the fractional factorial control array and the five dif-

ferent performance criteria. These performance criteria are subsequently analyzed using the graphical techniques presented in chapter 2, Fractional Factorial Designs.

```
> mold.sn <- robust.sn(mold.df, snratio = "target")
> mold.sn
  A B C D E G H shrink.mean shrink.sd shrink.meanl
1 - - - - - - -        2.23    0.0957        0.799
2 - - + - + + +        1.45    1.3300       -0.125
3 - + - + + - +        1.70    1.4500        0.138
4 - + + + - + -        1.92    0.0957        0.654
5 + - - + - + +        3.02    0.0500        1.110
6 + - + + + - -        2.60    1.3700        0.827
7 + + - - + + -        3.18    1.3300        1.090
8 + + + - - - +        1.90    0.0816        0.641
  shrink.lsdl shrink.target
1      -3.140        27.300
2       0.221         0.748
3       0.089         1.390
4      -2.990        26.100
5      -4.110        35.600
6      -0.478         5.570
7      -0.831         7.570
8      -3.150        27.300

 Summary statistics are calculated over the noise design
  M N O
1 - - -
2 + + -
3 + - +
4 - + +

Fraction:  ~ M:N:O
```

In the injection molding experiment, the aim is to achieve a shrinkage percentage close to the target of 2.5% so the `"target"` signal-to-noise ratio is specified (other choices are `"small"` and `"large"`).

By default five summary statistics are computed for each control factor combination:

`mean`	mean response
`sd`	standard deviation response
`meanl`	mean log response
`lsdln`	log standard deviation of the log response
`target`	target signal-to-noise ratio (`small`, `large` are other options)

Each performance criterion is computed over the noise design for each combination of control settings. For instance, the mean response for the first combination of

control factors (where all factors are at their low level) is computed as

$$\frac{2.2 + 2.3 + 2.3 + 2.1}{4} = 2.225$$

Choice of an appropriate signal-to-noise ratio depends on the type of response measured in the experiment, that is, whether the aim is to maximize, minimize, or achieve a fixed target. Problems are usually classified into three types:

smaller-is-better: want to minimize the response, e.g., number of defects.

larger-is-better: want to maximize the response, e.g., yield.

nominal-is-best: the response has some ideal target value, and the closer to the target (nominal) value the better, e.g., % shrinkage.

4.1.3 Signal-to-Noise Ratios Explained

The robust design approach aims to increase the quality of products and processes. Taguchi defines ideal quality as a product that gives on-target performance always for its entire life. Deviations from this result in a loss of quality. Quality loss is then defined as the total loss to society for the life of the product. With increased quality as the goal of the experiment, performance criteria are chosen to reflect this loss of quality. Commonly, simple quadratic loss is used:

$$L = k(y - m)^2$$

where y is the response, m is the target value, and k is a quality loss coefficient related to costs and limits of the response.

In smaller-is-better problems, the target value is $m = 0$ so $L = ky^2$ is an appropriate measure of loss. By analogy, for larger-is-better problems, $L = k(1/y^2)$ is an intuitively reasonable measure of loss. The signal-to-noise (SN) ratios essentially rescale the quadratic loss functions so that maximizing the SN ratio leads to a better process. The signal-to-noise ratios recommended by Taguchi, defined for each point in the control design, are as follows:

smaller-is-better : $SN_{S_j} = -10\log(\frac{1}{n}\sum_{i=1}^{n} y_{ij}^2)$

larger-is-better : $SN_{L_j} = -10\log(\frac{1}{n}\sum_{i=1}^{n} \frac{1}{(y_{ij})^2})$

nominal-is-best : $SN_{T_j} = -10\log(\frac{\bar{y}_j^2}{s^2})$

where y_{ij} $i = 1, n$ are response observations for the n points in the noise array at the jth run of the control array.

In each case a larger SN ratio indicates a better process. The larger-is-better and smaller-is-better SN ratios are derived directly from the simple quadratic loss functions above. The derivation of the nominal-is-best SN ratio is a little more complex and essentially assumes that the standard deviation of the response is proportional to the mean over the noise factors (Box88; Pha89). There is, in fact, on going debate in the statistical literature regarding appropriate performance criteria for use in the nominal-is-best problem. An alternative to SN_T, based on a Taylor series expansion for the variance of the log response, is to use the standard deviation of the log response, sd $(\log y)$, or as suggested by Box (Box88), log(sd $(\log y)$). (Note, however, that difficulties may arise if the target is near zero). If the standard deviation is *not* proportional to the mean, then we should simply minimize var(Y) (equivalently, sd(Y)).

Notice that the nominal-is-best SN ratio is not a function of the target value, m, even though the object is to bring the mean to target while minimizing variability in the response.For nominal-is-best problems the optimization has two stages (Pha89). The first stage involves finding the control factor settings that maximize the signal-to-noise ratio, or minimize the standard deviation of the response, over the noise array. This gives settings where the variance is small. The response is then adjusted to target. In practice it is hoped that this can be done by using a "scaling" factor that does not affect the SN ratio.

In what follows we restrict attention to the standard Taguchi analysis of the mean and signal-to-noise ratio as performance criteria for a robust design analysis. We analyze a nominal-is-best example since this involves a more detailed analysis than the larger-is-better and smaller-is-better cases. We emphasize that there may be several advantages to an alternative analysis of the mean and standard deviation of the response on the raw or transformed scale, and we refer the reader to Box (Box88). For an interesting approach to joint modeling of the mean and variance of the response, see Engel (Eng92).

4.2 Analyzing the Data

First plot the data:

```
> plot(mold.sn, y = "shrink.mean")
> plot(mold.sn, y = "shrink.target")
```

Figure 4.2 is a plot of the main effects for each factor. Factors `A`, `C` and `H` appear to have the biggest effect on the mean; `E` has a big effect on the signal-to-noise ratio.

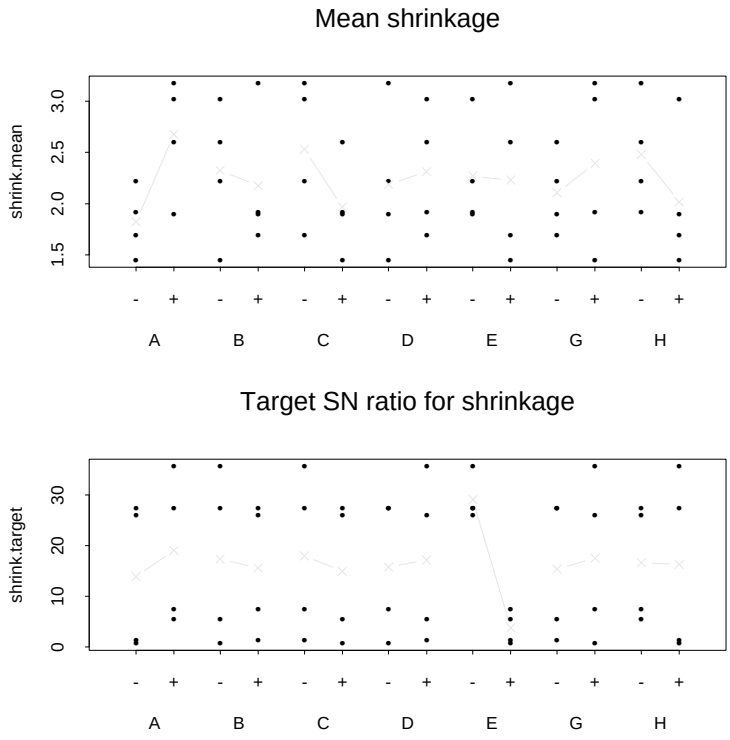

Figure 4.2: Plots of the main effects for the summary responses mean shrinkage and the target SN ratio.

Now analyze the responses more formally. First estimate the effects of the control factors on the response `shrink.mean`:

```
> moldmean.fac <- fac.aov(mold.sn, response = shrink.mean)
> moldmean.fac
Estimated Effects for Response: shrink.mean
   Effect
A  0.8500
B -0.1500
C -0.5625
D  0.1250
E -0.0375
G  0.2875
H -0.4625

Pseudo Standard Error of Effects = 0.4155
   Mean Standard Error = not available
```

Now look at the effects using a `pareto` plot.

```
> pareto(moldmean.fac)
```

Look at the fitted means:

```
> fitted(moldmean.fac)
     1    2   3     4     5   6     7   8
 2.225 1.45 1.7 1.925 3.025 2.6 3.175 1.9
```

These easily encompass the target of 2.5%. In figure 4.3 the `A`, `C` and `H` effects are the largest, although none are judged significant at the 5% level based on the pseudo standard error estimate of scale. In this case with just eight experimental runs, and with four of the seven estimated effects being at least moderate in size, the PSE estimate of scale breaks down and does not provide a meaningful cutoff for identifying significant effects (HO94).

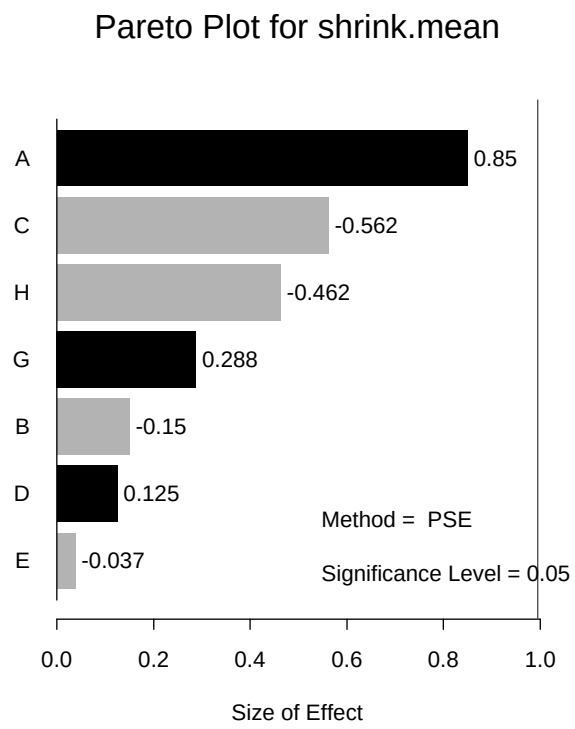

Figure 4.3: The Pareto plot of effects for response mean shrinkage.

Repeat the same sequence for the target SN ratio SN_T:

```
> moldtarget.fac <- fac.aov(mold.sn, response =
+   shrink.target)
> moldtarget.fac
Estimated Effects for Response: shrink.target
  Effect
A   5.15
B  -1.73
C  -3.05
D   1.42
E -25.27
G   2.10
H  -0.36

Pseudo Standard Error of Effects = 2.768
   Mean Standard Error = not available
> pareto(moldtarget.fac)
```

In figure 4.4 the factor `E` has a large effect on the SN ratio: since the estimated effect is negative we know that the maximum SN ratio occurs at the low level of E.

4.3 Selecting the Best Settings

Often there is no *best* setting of the factors but rather a range of possible choices. In such cases cost or convenience may determine the final choice. Compromises often have to be made, especially when there is more than one response variable being analyzed.

4.3.1 Selecting the Best Settings for Nominal-is-Best

We now give a two-stage procedure for choosing the control factor settings that minimize variability over the noise array subject to bringing the process mean to its target value. Justification of this procedure is given in Phadke (Pha89, pp. 281–283). First, maximize the target signal-to-noise ratio (or alternatively minimize `lsdl` = log(sd(log(Y)))). Then find a scaling factor (a factor that does not affect the variance, but does affect the mean) and adjust the response to target.

To find settings that maximize the signal-to-noise ratio, simply choose appropriate levels of factors that have an important effect on SN_T. For two level factors, if the estimated effect is negative, choose the low level; if positive, choose the high level.

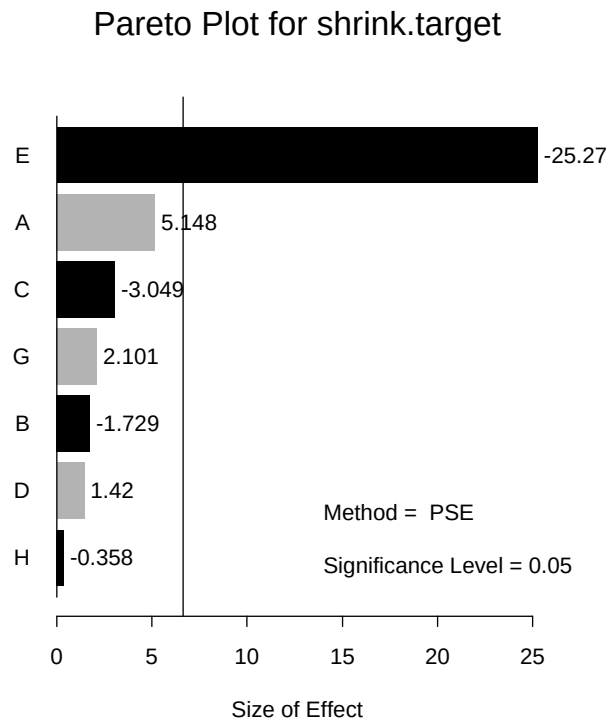

Figure 4.4: The Pareto plot of effects for response target SN ratio.

From the analysis held in `moldtarget.fac`, `E` is the only important factor affecting the target SN ratio, and its effect is negative, so `E` should be set at its low level.

We now want to adjust the mean to the target value 2.5%. To do this, control factors that affect the mean response but *don't* affect the target SN ratio are used, if they are available. That way, the SN ratio remains high *and* we achieve the target. In this case `A`, `C` and `H` affect the mean, and none of them affect SN_T, so there are several combinations of factor settings that we may use to adjust the mean to target without affecting SN_T.

First fit a model comprising these three factors and the factor affecting SN_T, i.e., factor `E`:

```
> moldmean.facs <- update(moldmean.fac, . ~ A + C + E + H)
```

Since we now want to use this submodel for predicting and adjusting the process mean to target we must first examine the relevant diagnostic plots.

```
> plot(moldmean.facs)
```

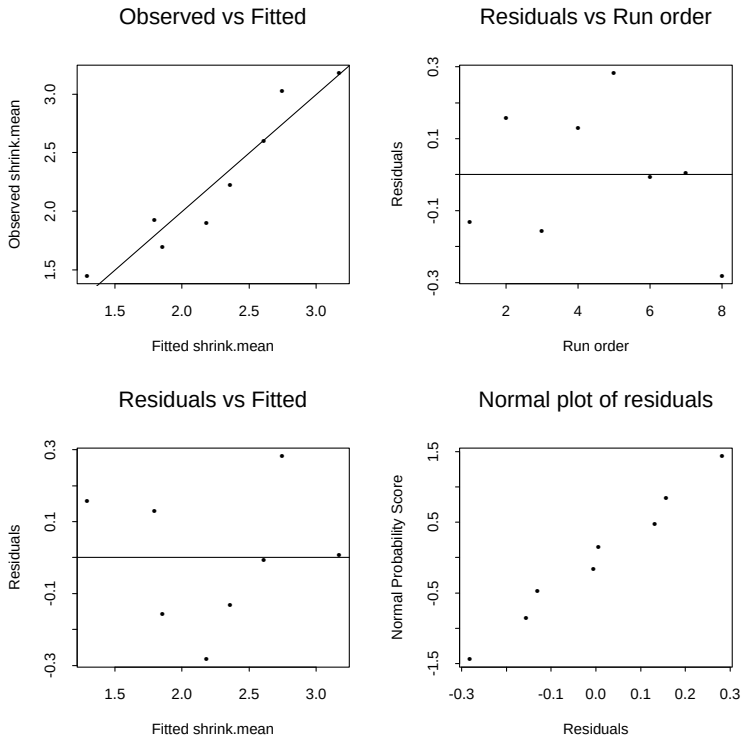

Figure 4.5: Diagnostic plots for the submodel `shrink.mean ~ A + C + E + H`.

Figure 4.5 shows that the submodel predicts the mean shrinkage well across the entire factor space and that the second order assumptions of the submodel are satisfied. Prediction and adjustment of the mean shrinkage from this model may then proceed.

Contour plots of the response in terms of two factors, with the other factors fixed, are now examined. These may be used to identify combinations of factor settings for which the response is predicted at the target value. Factors `A` and `C` have the largest effects in the full model, so these are obvious candidates. In the contour plot construction, the factor levels are coerced into integers, so that `"-"` becomes `1` and `"+"` becomes `2`. This essentially converts the factors to continuous variables and thus allows location of the target response in terms of intermediate settings of the factors.[3] Since factor `E` is known to maximize the signal-to-noise ratio when set to `"-"`, both contour plots given below include `E = "-"`.

[3] Clearly this will only make sense when the factors are levels of some intrinsically continuous variable.

```
> contour(moldmean.facs, vary = list(A = "v", C = "v",
+   E = "-", H = "-"), levels = seq(1.5,3,.25))
> contour( moldmean.facs, vary = list(A = "v", C = "v",
+   E = "-", H = "+"), levels = seq(1.5,3,.25))
```

In the first plot, `A` and `C` vary over their (coerced) ranges 1 to 2 and `H` is fixed at its low level[4]. In the second plot `H` is fixed at its high level. The contour plots in figure 4.6 show a range of possible settings for which the target value is obtained. Settings can be picked that are most convenient: for instance, if the low levels of C and `H` are the usual operating levels, it may be convenient to choose A to adjust the target. From the leftmost plot in figure 4.6, we can read off an approximate value for `A` of 1.2, with `C` and `H` at `"-"`. Thus factor `A` may be set at 20% of the distance between its low and high settings, and in combination with `C` and `H` at their low levels, the process mean response should be approximately brought to target.

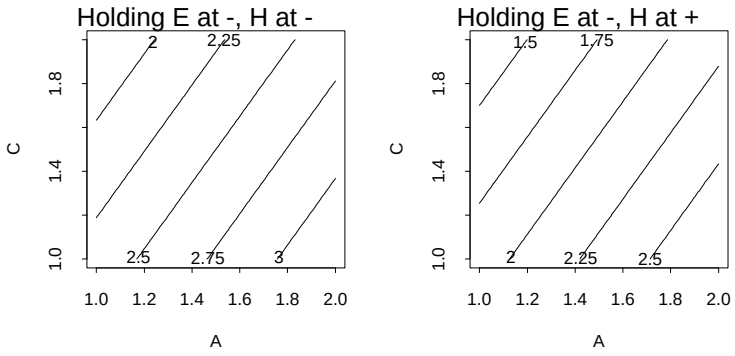

Figure 4.6: Contour plots for the reduced mean shrinkage model: `shrink.mean  A + C + E + H`.

The settings of factors `A`, `C` and `H` for which the target values are achieved can be explored further, as explained in section 4.3.2, Fine Tuning the Response to Target. In practice, the approximate method given above, combined with knowledge of the cost and use of each factor, produces an equally effective strategy.

4.3.2 Fine Tuning the Response to Target

Now that some possible settings have been chosen, we can check the results by predicting values for the mean response at these settings. In accordance with the

[4] `A` and `C` have levels `"-"` and `"+"` in the design. In drawing a contour plot, we are using a tacit assumption that `A` and `C` are continuous, thus `"-"` and `"+"` are mapped to 1 and 2 respectively. The contour function also allows you to give numeric levels for `A` and `C`.

first graph in figure 4.6, try two different settings for `A` and `C` of (1.2,1) and (1.8,2), with `E` and `H` at `"-"`.

```
> newdata <- data.frame(A = c(1.2, 1.8),
+   C = c(1, 2), E = rep("-", 2), H = rep("-", 2))
> predm <- predict(moldmean.facs, newdata, numeric.levels =
+   list(A = c(1,2), C = c(1,2)))
> cbind(newdata, predm)
    A C E H       X
1 1.2 1 - - 2.52625
2 1.8 2 - - 2.47375
```

Both of these combinations of factor settings closely satisfy our goal: a high signal-to-noise ratio and a mean of 2.5% shrinkage.

To keep track of the target signal-to-noise ratio we may also like to construct a model for SN_T. Note that SN_T is really affected only by factor `E`, so an appropriate submodel is fit as follows:

```
> moldtarget.facs <- update(moldtarget.fac, . ~ E)
> newdata.t <- data.frame(E = c("-","+"))
> predt <- predict(moldtarget.facs, newdata.t)
> cbind(newdata.t, predt)
  E         X
1 - 29.090520
2 +  3.820328
```

Note the huge effect of factor `E` on SN_T. You should examine the diagnostic plot for this model. Note that a model for SN_T containing additional terms for factors `A`, `C` and `H` produces similar predictions to the simple model for SN_T above. This is because these factors have little effect on SN_T. Thus for any settings that bring the mean close to target SN_T has a similar value of around 29, provided factor `E` is at its low setting.

```
> moldtarget.facs <- update(
+   moldtarget.fac, . ~ A + C + E + H)
> predt <- predict(moldtarget.facs, newdata,
+   numeric.levels = list(A = c(1,2), C = c(1,2)))
> cbind(newdata, predt)
    A C E H        X
1 1.2 1 - - 29.24969
2 1.8 2 - - 29.28944
```

Alternatively, rather than having to read approximate factor settings off the contour plots, we can use values returned by the `contour` function. `contour` can return a

vector of values corresponding to the target line on the contour plot. These can then be used as values in a prediction data set. We obtain and save combinations of `A` and `C` that produce predictions of 2.5% as follows:

```
> contour.save <- contour(moldmean.facs, vary =
+   list(A='v', C='v', E='-', H='-'), levels = 2.5,
+   save= T)
```

The output `contour.save` is a list containing a structure `"2.5"` with vectors `x` and `y` containing values for `A` and `C`, respectively, that achieve the target value 2.5%. Predictions for these values of `A` and `C` in combination with the chosen levels for the other factors, or predictions for any other set of factor combinations, may be obtained using `predict`. In this example there are 82 pairs of `(A,C)` settings saved in `contour.save` that achieve the target value 2.5%. You may check that these combinations all produce a prediction of 2.5% mean shrinkage as follows:

```
> Avals <- contour.save$"2.5"$x
> Cvals <- contour.save$"2.5"$y
> ACpairs <- data.frame(A=Avals, C=Cvals,
+   E=rep('-',length(Avals)), H=rep('-',length(Avals)))
> predm.all <- predict(moldmean.facs, ACpairs,
+   numeric.levels = list(A = c(1,2), C = c(1,2)))
> cbind(ACpairs, predm.all)
```

4.3.3 Verification

A further small experiment with the control factors set at some of these combinations is then recommended to verify and document the on-target response. The improvement in signal-to-noise ratio, or alternatively in standard deviation of the response, may also be verified and documented in this experiment by considering each of the combinations of control factors at each of the noise array settings studied in the robust design experiment described above. With three combinations of the control factors `A` and `C` from the above list, each examined at four noise array settings, a twelve-run verification experiment could be conducted. Factor `E` will be set at its low level, `"-"`, in the verification experiment, and the other factors may be set at their most convenient and/or inexpensive levels.

If the verification experiment confirms the findings of the initial robust design experiment, the process is then set to target, is robust to known sources of noise and furthermore is set to run as cheaply and easily as it can.

Appendix A

Design Digest

The purpose of the design digest and of the function `design.digest` is to facilitate the selection and creation of commonly used fractional factorial designs. This appendix includes information about how to generate designs and about the properties of these designs. Center points, blocking, replication, and randomization are also addressed. In addition, information is provided on irregular fractions, partial confounding, fold-over designs, mixed-level designs, and orthogonal arrays. For more information on these topics, see (Haa89) and (BHH78) listed in the Bibliography. At the end of this appendix are descriptions of each of the designs that can be generated by name using `design.digest`. Each description includes a listing of the design and information about its properties.

A.1 Designs Available in the Design Digest

The function `design.digest` can generate standard designs using a simple naming convention. This collection of designs is called the *design digest*. A design name is of the form *aakkrr*, where

aa = `ff` for a standard two-level fractional factorial

= `mf` for a mixed factorial with one factor having three levels and the rest of the factors having 2 levels

= `if` for an irregular two-level fractional factorial design

= `pb` for a two-level Plackett-Burman design

kk = number of experimental factors, $kk = 03, 04, 05, \ldots, 11$

rr = number of runs, $rr = 08, 12, 16, 24, 32, 64$

Table A.1 lists the names of the designs that are included in the design digest.

Table A.1: Designs available in the design digest.

Number of Factors	**8-Run**	**12-Run**	**16-Run**	**24-Run**	**32-Run**	**64-Run**
3	`ff0308`	`mf0312`	*	*	*	*
4	`ff0408`	`if0412`	`ff0416`	`mf0424`	*	*
		`mf0412`				
5	`ff0508`	`if0512`	`ff0516`	`mf0524`	`ff0532`	*
		`mf0512`				
6	`ff0608`	`if0612`	`ff0616`	`if0624`	`ff0632`	`ff0664`
7	`ff0708`	`pb0712`	`ff0716`	`pb0724`	`ff0732`	`ff0764`
8	+	`pb0812`	`ff0816`	`pb0824`	`ff0832`	`ff0864`
9	+	`pb0912`	`ff0916`	`pb0924`	`ff0932`	`ff0964`
10	+	`pb1012`	`ff1016`	`pb1024`	`ff1032`	`ff1064`
11	+	`pb1112`	`ff1116`	`pb1124`	`ff1132`	`ff1164`

* Replicate a smaller design to achieve this sample size.
+ No design exists for this combination of factors and runs.

The designs listed in table A.1 are frequently used fractional factorial designs. Any of these designs can be generated via the function `design.digest` using the design name as follows:

```
> design.digest("ff0308")

Design Name: ff0308

  A B C
1 - - -
2 - - +
3 - + -
```

```
4 - + +
5 + - -
6 + - +
7 + + -
8 + + +
```

The same design can be generated with customized factor names and levels as follows:

```
> fac.names <- list(time=c(30,60),temp=c(25,35),
+     catalyst=c("A","B"))
> design.digest("ff0308",factor.names=fac.names)

Design Name: ff0308

  time temp catalyst
1   30   25        A
2   30   25        B
3   30   35        A
4   30   35        B
5   60   25        A
6   60   25        B
7   60   35        A
8   60   35        B
```

In this case, `fac.names` is a list providing names and levels for the corresponding factors. As long as the second argument to `design.digest` is the value of `factor.names`, the argument name doesn't have to be explicitly given. For example, a design with custom factor names but having the default factor levels of `-` and `+` can be generated as follows:

```
> design.digest("ff0308",c("time","temp","catalyst"))

Design Name: ff0308

  time temp catalyst
1    -    -        -
2    -    -        +
3    -    +        -
4    -    +        +
5    +    -        -
6    +    -        +
7    +    +        -
8    +    +        +
```

The expression `factor.names(design)` returns a list containing the factor names and levels. It can also be used to assign factor names and levels to an existing design. For example, another way to produce the design above is as follows:

```
> fac.names <- list(time=c(30,60),temp=c(25,35),
+ catalyst=c("A","B"))
> a.design <- design.digest("ff0308")
> factor.names(a.design) <- fac.names
```

The names of the rows can be changed similarly by using the function `row.names`, for example:

```
> row.names(a.design) <- paste("Run",1:nrow(a.design))
```

Many other kinds of designs can be generated using the functions `fac.design` and `oa.design`. There are also other methods for creating designs using `design.digest`. These methods are covered in section A.10, More on generating designs.

A.2 Design Resolution

One of the important properties of fractional factorial designs is *resolution*. A resolution III design is one in which two-factor interactions may be confounded with each other and with main effects. Two effects are confounded if they cannot be independently estimated. A resolution IV design does not have confounding between two-factor interactions and main effects, but does have confounding among two-factor interactions. A resolution V design does not have confounding among main effects and two-factor interactions or among two-factor interactions. Thus all of the terms up to two-factor interactions can be independently estimated in a resolution V design. There are resolutions higher than V, but typically you are not interested in them since the allowable confounding involves three-way interactions which are generally assumed to be negligible.

One way to remember the different levels of resolution is to count up the number of factors involved in the allowable confounding. For example, a main effect confounded with a two-factor interaction has one + two = three factors involved in the confounding relationship and so is classed as resolution III. Two two-factor interactions involve four factors corresponding to a resolution IV design. Finally, a resolution V design allows at most confounding between a two-factor interaction and a three-factor interaction.

In general, when choosing an experimental design you should use the highest resolution design that you can afford because of the protection offered against misinterpretation of two-factor interactions. However, higher resolutions require larger

sample sizes so there is a trade-off between this protection and the cost of running the experiment. Table A.2 provides information about the sample sizes required to get resolution III, IV, and V designs for standard two-level fractional factorial designs.

Table A.2: Sample sizes for standard fractional factorial designs.

Number of Factors	**Resolution III**	**Resolution IV**	**Resolution V**
3	4	*	8
4	*	8	16
5	8	*	16
6	8	16	32
7	8	16	64
8	*	16	64
9	16	32	128^{+}
10	16	32	128^{+}
11	16	32	128^{+}

* There is no standard two-level fractional factorial design for this combination.
$^{+}$ This design is not available in library dox. The maximum number of runs is 64.

To find out the resolution of a design, use the `summary` function as follows:

```
> summary(design.digest("ff0308"))

Design Name: ff0308

Fractional Number of Runs: Full Factorial
Resolution: >V

 No confounding.
All main effects and two-factor interactions can be estimated.

Variable summaries:
   A     B     C
 -:4   -:4   -:4
 +:4   +:4   +:4
```

Design `ff0308` is a full factorial design with resolution greater than V. Thus there is no confounding between main effects and two-factor interactions or among two-factor interactions. Further interpretation of the information provided by the `summary` function is provided in the following sections.

A.3 Confounding

For designs with less than resolution V, you need to be aware of which effects are confounded with each other. For example, from table A.2, design `ff0408` is a resolution IV design so two-factor interactions may be confounded with each other. The function `summary` lists the *confounding pattern* of the design so that you can properly interpret the results of the analysis; namely,

```
> summary(design.digest("ff0408"))

Design Name: ff0408

Generating Fraction:  ~  A:B:C:D
Fractional Number of Runs: 1 / 2
Resolution: IV

Confounding Pattern:
 Main effects are not confounded with two-factor
interactions. Two-factor interactions are confounded
with other two-factor interactions. Only one two-factor
interaction from each line listed below can be estimated.

A:B + C:D
A:C + B:D
B:C + A:D

Variable summaries:
   A     B     C     D
 -:4   -:4   -:4   -:4
 +:4   +:4   +:4   +:4
```

The first line in the confounding pattern is `A:B + C:D`. This means that the two-factor interaction `A:B` is confounded with the two-factor interaction `C:D`. Thus `A:B` and `C:D` can not be distinguished from each other. Only the sum (or difference) between the two interactions can be estimated.

If you rank your experimental factors in order of importance based on your prior belief, then it is often reasonable to assume that two-factor interactions among more important factors are more likely to be significant than interactions among less important factors. The default saturated models fit to designs in the design digest label the interactions terms as interactions of the factors leftmost in the design frame. Hence when generating designs, put the factors in order of importance from left to right in the design (most important first). For example, consider the confounding pattern for the design `ff0508`:

```
> summary(design.digest("ff0508"))

Design Name: ff0508

Generating Fraction:  ~   - A:D:E - B:C:E
Fractional Number of Runs: 1 / 4
Resolution: III

Confounding Pattern:
 Main effects are confounded with two-factor interactions.
Only one effect from each line listed in the confounding
pattern given below may be estimated.

A + D:E
B + C:E
C + B:E
D + A:E
E + B:C + A:D
A:B + C:D
A:C + B:D

Variable summaries:
   A     B     C     D     E
 -:4   -:4   -:4   -:4   -:4
 +:4   +:4   +:4   +:4   +:4
```

If the design is arranged so that `A` is assigned to the most important and `E` to the least important, then `D:E` may be assumed to be the least likely interaction. In this design it is confounded with the factor `A` which is mostly likely to be important. Similarly, the `A:B` interaction may be assumed to be more likely to be important than the `C:D` interaction. Therefore, a significant `A:B + C:D` interaction would typically be interpreted as being due simply to `A:B`.

While this is not a foolproof rule, it does make it worthwhile to rank the experimental factors in order of presumed importance before assigning them to factors in a design. In this way, you will take best advantage of the confounding properties of the designs in the design digest.

In addition to the `summary` function, the `alias` function also provides information about the confounding pattern of a design. Although the information provided by `alias` is in a slightly more difficult form to interpret, it is more complete in that information about confounding of higher level interactions is also present. Compare for example, the results of the following two commands:

```
> summary(design.digest("ff0508"))
> alias(design.digest("ff0508"))
```

In the output from `alias` there is a one in the intersection between the column labeled `A` and the row labeled `D:E` which indicates that these two effects are confounded with each other. This same information is provided in the confounding pattern.

A.4 Replication

When a design is replicated, each of its runs is repeated one or more times so that there are multiple, independent response values for each run. Replication is a good practice when there is quite a bit of experimental noise and the cost of repeating runs is small compared to the overall cost of setting up the experiment. A design can be replicated as follows:

```
> design.digest("ff0308",rep=2)

Design Name: ff0308

   A B C
 1 - - -
 2 - - -
 3 - - +
 4 - - +
 5 - + -
 6 - + -
 7 - + +
 8 - + +
 9 + - -
10 + - -
11 + - +
12 + - +
13 + + -
14 + + -
15 + + +
16 + + +
```

Note that there are 16 runs in the new design; each of the original eight runs appears twice. One advantage of replication is that it provides an independent measure of the experimental error. This is especially useful for significance testing. Replication also increases the power of the experiment; that is, it makes it easier (more likely) to detect smaller effects.

A caution to be aware of in replication is that simply repeatedly measuring the same sample is not the same as making independent runs of the experiment. As a general rule, making repeated measurements is desirable when the cost of measurement is low but the measurements are noisy. Repeated measurement are then averaged, and the averages analyzed as an unreplicated experiment. The experiment should *not* be treated as if it is replicated.

A.5 Center Points

A *center point* is a run made with the each of the factors set at the midpoint of its range. Center points are used to provide information about experimental error and about the value of the response in the middle of the experimental region. Center points are difficult to interpret if one or more of the experimental factors is qualitative.

To create a design with center points use the arguments `n.cp` to specify how many center points, `cp.values` to give the values for the center points, and `cp.place` to specify where in the design the center points should be placed. The possible values for `cp.place` are `"first"`, `"last"`, `"ends"`, `"spaced"`, and `"random"`. If the design is blocked the placement occurs within each block (see below). When `"ends"` is used, half of the center points are placed at the beginning and half at the end. When `"spaced"` is used, center points are evenly divided between the beginning, middle, and end of the design (block). `n.cp` is adjusted if not divisible by two or three, respectively. The default is `"last"`. For example,

```
> factor.names <- list(temp=c(25,45),time=c(10,30),pH=c(7,8))
> design.digest("ff0308",factor.names,n.cp=2,
+   cp.values=c(30,20,7.5),cp.place="ends")

Design Name: ff0308

   temp time  pH
 1   30   20 7.5
 2   25   10   7
 3   25   10   8
 4   25   30   7
 5   25   30   8
 6   45   10   7
 7   45   10   8
 8   45   30   7
 9   45   30   8
10   30   20 7.5

> design.digest("ff0308",factor.names,n.cp=3,
```

```
+   cp.values=c(30,20,7.5),cp.place="spaced")

Design Name: ff0308

   temp time  pH
 1   30   20 7.5
 2   25   10   7
 3   25   10   8
 4   25   30   7
 5   25   30   8
 6   30   20 7.5
 7   45   10   7
 8   45   10   8
 9   45   30   7
10   45   30   8
11   30   20 7.5
```

Center points may be added to an existing design using the function `addcp`.

If a design includes center points, then `fac.aov` automatically computes an estimate of the curvature or quadratic effect. If this term is significant, it means that the mean value of the center points is significantly higher or lower than the average of the other design points.

A.6 Randomization

Systematic changes in experimental conditions can lead to confusion and misinterpretation of experimental results. For example, suppose that samples to be tested are left on the lab bench during the day and that there is degradation in the samples during the course of measurement. If all of the runs with high settings of factor `A` are run first and all of the runs with low settings are run last, then the degradation in the samples will be confounded with the effect of factor `A`.

The best strategy is to think carefully about which conditions might change during the course of an experiment and take proper precautions to minimize or eliminate their effects. Sometimes changes are unavoidable; for example two batches of raw materials may need to be used to complete the experiment. In this case the experiment should be *blocked* to account for the planned change. (See section A.7, Blocking.)

Randomization can prevent you from misinterpreting a degradation effect as a treatment effect as described in the example above. An experiment is randomized by randomly rearranging the order of the runs. For example, to randomize an experiment in library dox, use the function `randomize.design` as follows:

```
> randomize.design(design.digest("ff0408"))

Design Name: ff0408

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

Fraction:  ~ A:B:C:D
```

To return a design to its original order, use the function `sort.design(design)`. Sometimes it may not be possible to completely randomize a design, either due to expense or difficulty of changing factor settings. In this case, you can use the argument `restrict` to restrict the randomization as follows:

```
> randomize.design(design.digest("ff0408"),restrict=c("A","B"))

Design Name: ff0408

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

Fraction:  ~ A:B:C:D
```

The function `randomize.design` randomizes within blocks if a design is already blocked. If a design has center points, their placement is preserved unless a new value is given for the argument `cp.place`, for example `cp.place="random"` would randomize the location of the center points regardless of there previous placement. The allocation of center points to blocks is always preserved.

A.7 Blocking

Blocking is used when it is known in advance that experimental conditions will change during the course of an experiment and it is desired to keep these changes from influencing the results of the experiment. For example, if more than one batch of raw materials or more than one technician is required to complete the experiment, then this source of variation should be controlled by blocking.

If, for example, two batches of raw materials will be used, then half of the experiment should be done with each batch. When an experiment is blocked, the runs are divided up in such a way that the difference between batches of raw materials is independent of the main effects (and often of the two-factor interactions). This is done by associating the difference in raw materials with one of the model terms, typically a higher-order interaction.

For example, consider the problem of blocking the design `ff0516`. Because it is a resolution V design, all of the two-factor interactions can be estimated. If factors are listed in order from most to least important, it is reasonable to assume that `D:E` is the interaction that is least likely to be important. Thus, it makes sense to associate the affect of raw material batch with the `D:E` interaction. Generate this design as follows:

```
> design.digest("ff0516",n.blocks=2)

Design Name: ff0516

   Blocks A B C D E
 1      1 - - + - -
 2      1 - - + + +
 3      1 - + - - -
 4      1 - + - + +
 5      1 + - - - -
 6      1 + - - + +
 7      1 + + + - -
 8      1 + + + + +
 9      2 - - - - +
10      2 - - - + -
11      2 - + + - +
12      2 - + + + -
13      2 + - + - +
14      2 + - + + -
15      2 + + - - +
16      2 + + - + -

Fraction:  ~  - A:B:C:D:E
```

Notice that the runs in Block #1 and Block #2 correspond to the product of columns `D` and `E` being `+` and `-`, respectively. More information about the blocked design can be obtained using the `summary` function as follows:

```
> summary(design.digest("ff0516",n.blocks=2))

Design Name: ff0516

Blocking: 2 Blocks of Size 8
Blocks are confounded with at least one two-factor interaction.
Blocks Generated by:  ~  D:E

Generating Fraction:  ~   - A:B:C:D:E
Fractional Number of Runs: 1 / 2
Resolution: V

 No confounding.
All main effects and two-factor interactions can be estimated.

Variable summaries:
 Blocks     A     B     C     D     E
 1:8      -:8   -:8   -:8   -:8   -:8
 2:8      +:8   +:8   +:8   +:8   +:8
```

Since the blocks were generated by the `D:E` interaction, the estimate of block effect is confounded with the estimated `D:E` interaction. Thus, an estimated block effect replaces an estimated `D:E` interaction in the analysis.

Blocking has been implemented in library dox following the guidelines presented by Box, Hunter, and Hunter (BHH78). Tables A.3 and A.4 show the blocking arrangements available in library dox.

There are three arguments to `design.digest` that control blocking. These are `block.size`, the number of runs in each block, `n.blocks`, the number of blocks, and `block.gen`, a model formula giving one or more defining contrasts (e.g., `A:B:D + B:C:E`) to be used to generate blocks. If either `block.size` or `n.blocks` is given, then `design.digest` looks up `block.gen` in the table of blocking patterns shown in table A.4. For example, the three calls

```
> design.digest("ff0308",n.blocks=2)
> design.digest("ff0308",block.size=4)
> design.digest("ff0308",block.gen=~A:B:C)
```

are equivalent. However, giving a defining contrast other than one shown in table A.4 generates a different blocking pattern from the default.

Table A.3: Blocking patterns for factorial designs: 1-5 factors.

Design Name	Number of Blocks	Block Size	Block Generator	Interactions Confounded with Blocks
ff0308	2	4	A:B:C	A:B:C
	4	2	B:C, A:C	B:C, A:C, A:B
ff0312	2	6	A:B:C	A:B:C
ff0408	2	4	B:C	B:C
if0412	2	6	C:D	C:D
mf0412	2	6	C:D	C:D
ff0416	2	8	A:B:C:D	A:B:C:D
	4	4	A:C:D, A:B:D	A:C:D, A:B:D, B:C
	8	2	C:D, B:C, A:D	C:D, B:C, A:D, B:D, A:C, A:B:C:D, A:B
mf0424	2	12	A:B:C:D	A:B:C:D
ff0508	2	4	A:C	A:C
if0512	2	6	C:E	C:E
mf0512	2	6	B:D	B:D
ff0516	2	8	D:E	D:E
mf0524	2	12	A:C:D	A:C:D
ff0532	2	16	A:B:C:D:E	A:B:C:D:E
	4	8	C:D:E, A:B:C	C:D:E, A:B:C, A:B:D:E
	8	4	A:D:E, A:C:D, A:B:C	A:D:E, A:C:D, A:B:C, A:C:E, B:C:D:E, B:D, B:E
	16	2	D:E, C:E, B:C, A:B	all 2fis and 4fis

[+]2fi, 4fi, etc. are abbreviations for two-factor interaction, four-factor interaction, etc.

Table A.4: Blocking patterns for factorial designs: 6-11 factors.

Design Name	Number of Blocks	Block Size	Block Generator	Interactions Confounded with Blocks
ff0608	2	4	A:B	A:B
if0612	2	6	C:E	C:E
ff0616	2	8	A:B:D	A:B:D
ff0632	2	16	D:E:G	D:E:G
ff0664	2	32	A:B:C:D:E:G	A:B:C:D:E:G
ff0664	4	16	A:D:E:G, A:B:C:D	no 2fis[+]
ff0664	8	8	B:D:G, A:B:E:G, C:D:E:G	no 2fis
ff0664	16	2	E:G, D:E, C:D, B:C, A:B	all 2fis, 4fis and 6fis
ff0716	2	8	A:B:C	A:B:C
pb0724	2	12	A:C:E	A:C:E
ff0732	2	16	C:D:G	C:D:G
ff0764	2	32	A:C:E:H, B:C:G:H, D:E:G:H	4fis only
ff0816	2	8	C:D	C:D
pb0824	2	12	A:B:C	A:B:C
ff0832	2	16	A:E:I	A:E:I
ff0864	4	32	D:G:I, A:E:G	D:G:I, A:E:G, A:D:E:I
ff0916	2	8	A:E	A:E
ff0932	2	16	A:E:I	A:E:I
ff1016	2	8	B:D	B:D
ff1032	2	16	A:E:I	A:E:I
ff1116	2	8	A:D	A:D
ff1132	2	16	A:D:L	A:D:L

[+]2fi, 4fi, etc. are abbreviations for two-factor interaction, four-factor interaction, etc.

A.8 Irregular Fractions and Placket-Burman Designs

The standard fractional factorials designs of 8, 16, and 32 runs are used extensively and their properties are well known. However, there are times when an eight-run design doesn't provide high enough resolution while a sixteen-run design is too expensive. The design digest includes a number of twelve-run designs for this purpose; namely, `if0412`, `if0512`, `if0612`, `pb0712`, `pb0812`, `pb0912`, `pb1012`, and `pb1112`. There are also compromise designs of 24 runs that provide an alternative between 16 and 32 run designs; namely `if0624`, `pb0724`, `pb0824`, `pb0924`, `pb1024`, and `pb1124`.

The `if` series of designs represent a special type of fractional factorial called irregular fractions (Joh69; Haa89). For example, the design `if0412` is a 3/4 fraction of the full factorial `ff0416`. It is interesting to note from the summary of design `if0412` that it is a *nearly* resolution V design. By nearly resolution V, we mean that there is some partial confounding or correlation among the main effects and two factor interactions. To investigate this further consider the following:

```
> if.design <- design.digest("if0412")
> alias(design.digest("if0412"))
Model
 ~ A * B * C * D

Complete
        (Intercept)  A  B  C  D A:B A:C B:C A:D B:D C:D B:C:D
A:B:C:D  1           1                                    -1
  A:C:D                 1        1                 -1
  A:B:D                    1         1          -1
  A:B:C                       1         -1   1

Partial
            (I)  A B C D A:B A:C B:C A:D B:D C:D B:C:D
(Intercept)      1                          -1
          A                                 -1
          B              1                -1
          C                 1          -1
          D                    -1   1
        A:B                             -1
        A:C                          -1
        B:C                       -1
        A:D
        B:D
        C:D
      B:C:D
```

```
Notes:
$"Max. Abs. Corr.":
[1] 0.5
```

Notice that the effect `A:B` is partially confounded with `B`. This means that if we look at the model matrix used to estimate the effects, the correlation between the columns labeled `B` and `A:B` is nonzero. We can look at these terms directly as follows:

```
> attach(if.design)
> a.b <- fac2num(A,numeric.levels=c(-1,1))*
+ fac2num(B,numeric.levels=c(-1,1))
> a.b
 [1]  1  1  1  1 -1 -1 -1 -1 -1 -1  1  1
> b <- fac2num(B,numeric.levels=c(-1,1))
> b
 [1] -1 -1 -1 -1  1  1  1  1 -1 -1  1  1
> cor(a.b,b)
[1] -0.3333333
```

Thus, the correlation between the `B` and `A:B` columns in the model matrix is -0.33. This is different from the correlation reported by `alias` because the `alias` function reports the correlation between the *coefficients*. For the purposes of evaluating an experimental design, we generally prefer to look at the correlation between columns in the design matrix. We can look at all of the correlations between main effects and two-factor interactions as follows:

```
> if.fac <- fac.aov(y~.^2,cbind(if.design,y=rnorm(12)))
> round(cor(if.fac$x)[-1,-1],2) #  omit intercept
    A     B     C     D   A:B   A:C   A:D  B:C  B:D  C:D
  A 1  0.00  0.00  0.00  0.00  0.00  0.00 0.00 0.00 0.00
  B 0  1.00  0.00  0.00 -0.33  0.00  0.00 0.00 0.00 0.33
  C 0  0.00  1.00  0.00  0.00 -0.33  0.00 0.00 0.33 0.00
  D 0  0.00  0.00  1.00  0.00  0.00 -0.33 0.33 0.00 0.00
A:B 0 -0.33  0.00  0.00  1.00  0.00  0.00 0.00 0.00 0.33
A:C 0  0.00 -0.33  0.00  0.00  1.00  0.00 0.00 0.33 0.00
A:D 0  0.00  0.00 -0.33  0.00  0.00  1.00 0.33 0.00 0.00
B:C 0  0.00  0.00  0.33  0.00  0.00  0.33 1.00 0.00 0.00
B:D 0  0.00  0.33  0.00  0.00  0.33  0.00 0.00 1.00 0.00
```

Thus, the correlations among model terms are at most ± 0.33. This level of partial confounding is often an acceptable trade-off against the cost of doing the extra 4 runs necessary to get the true resolution V design (`ff0416`).

Note: In the above code, it was necessary to create a dummy `fac.aov` object in order to get the model matrix. This was done by adding a dummy response to the design consisting of 12 uniform random normal deviates. The formula `y~.^2` specifies all main effects and all interactions up to order 2. The resulting model matrix is the component `x` of the resulting object `if.fac`. In order to have a more readable output, the correlation result was rounded to 2 decimal places and the row and column corresponding to the correlation with the intercept were omitted.

One interesting note about the design `if0412` is that it is the only design in the design digest that is not balanced. When running this design, select the levels of factor A so that the low level (with eight runs) is of more interest than the high level (with only four runs).

The `pb` series of designs is based on the well known Plackett-Burman design (PB46; BHH78; Haa89). The Plackett-Burman designs also have interesting confounding patterns. By executing the following commands, you can see that in the twelve-run Plackett-Burman designs the main effects are unconfounded with other main effects but the two-factor interactions are partially confounded with each other (correlation $= \pm 0.33$).

```
> pb.design <- design.digest('pb1112')
> pb.fac <- fac.aov(y~.^2,cbind(pb.design,y=rnorm(12)))
> round(cor(pb.fac$x),2)
```

Note that the design digest produces a design with 11 columns when you request the design `pb0712`. This is because you can include the extra columns as dummy factors in the analysis. If any of the dummy factors turn up as being significant, it suggests the presence of two-factor interactions. (See (Haa89) for more details.)

The 24-run Plackett-Burman designs are called *fold-over* designs (BW51). Main effects in these designs are clear of other main effects and two-factor interactions. Two-factor interactions are partially confounded (correlation $= \pm 0.33$) with other two-factor interactions that do not have a letter in common. For example, `A:B` is partially confounded with `C:D` but is independent of `A:C`. Thus the estimable two-factor interactions in these designs are those that involve factor `A`. For example, in `pb1124` there is no partial confounding among the effects `A, B, C, ... , L, A:B, A:C, ... , A:L`.

A.9 Mixed-Level Designs

The designs `mf0412`, `mf0424`, `mf0512`, and `mf0524` are mixed-level fractional factorial designs (LH88; Haa89). Each of these designs allows for one factor to have three levels while all the rest of the factors must have two levels.

- Design `mf0412` is a 1/2 fraction of a 3×2^3 factorial.

- Design `mf0424` is a full 3×2^3 factorial.
- Design `mf0512` is a $1/4$ fraction of a 3×2^4 factorial.
- Design `mf0524` is a $1/2$ fraction of a 3×2^4 factorial.

Designs `mf0412` and `mf0524` are nearly resolution V while `mf0412` is nearly resolution IV. In each of these designs except `mf0524` the interactions between the quadratic term (three-level factor) and other two-level factors and two-factor interactions involving only two-level factors are confounded. There is also some other partial confounding (correlation = ± 0.33) but it can be ignored in the analysis.

A.10 More on Generating Designs

The design names given in table A.1 provide a shortcut for generating designs. However, in general a design can be specified by giving a vector of levels for the factors and by specifying a fraction as follows:

```
> design.digest(rep(2,4),fraction=1/2)
```

This gives the same design as `design.digest("ff0408")`. Consider the following ways of generating a one-fourth fraction of a two-level, five factor design:

```
> design.digest("ff0508")
> design.digest(rep(2,5),fraction=1/4)
> design.digest(rep(2,5),fraction=~- A:D:E - B:C:E)
> design.digest(rep(2,5),fraction=~- B:D:E - A:C:E)
```

The first three calls produce the same result because the design `ff0508` is a $1/4$ fraction of the full factorial generated by the defining contrasts `~ - A:D:E - B:C:E`. (See (BHH78) for more information on defining contrasts.) However, the last design which was generated with a different defining contrast is different from the first three.

The function `fac.design` is similar to `design.digest` except that it doesn't follow the naming conventions of the design digest (so design names can't be used). If the same fraction is given, the both functions return the same design although there are differences in how the design is sorted. For example, the following are generally equivalent for any design:

```
> design.digest(rep(2,4),fraction=~ A:B:C:D)
> sort.design(fac.design(rep(2,4),fraction=~ A:B:C:D))
```

Although `design.digest` can only generate designs with one factor at 3 levels and the rest at two levels, `fac.design` can generate designs with arbitrary numbers of levels for each factor. For example, a useful design with three factors at three levels can be generated as follows:

```
fac.design(rep(3,3))
```

However, `fac.design` cannot fractionate designs unless all of the factors have two-levels. See (CH92) for more information about `fac.design`.

A.11 Generating Standard Orthogonal Arrays

In the application of robust design or Taguchi methods, orthogonal array designs are often used (Pha89). The common two-level orthogonal arrays are called `L4, L8, L12, L16`, and `L32`. The common three-level orthogonal arrays are called `L9, L18 and L27`. In many cases, these correspond to standard designs that are available in the design digest. In other cases, specific orthogonal arrays can be generated using the function `oa.design`.

For example, the two-level designs of size 8, 16, and 32 are equivalent to standard fractional factorial designs so you can use `ff0516` instead of `L16(5)` for the case of five two-level factors. (For a design with only two-level or only three-level factors, we use the notation `Ln(k)` to indicate that there are `n` runs and `k` factors. For designs that can have both two and three-level factors, we use the notation `Ln(`k_1, `,` k_2`)` to indicate that there are k_1 factors with two levels and k_2 factors with 3 levels.) As an alternative to the `L12` designs, you can use the `if` and `pb` although they are not exactly equivalent. The following are a few examples of how to generate selected orthogonal arrays:

```
> oa.design(rep(2,7))                    # L12(7)
> oa.design(rep(2,11),min.resid.df=0)    # L12(11)
> oa.design(rep(3,4))                    # L9(4)
> oa.design(c(2,rep(3,6)))               # L18(1,6)
```

The designs `L9` and `L18` are especially useful because they allow several factors to have three levels each.

Bibliography

G.E.P. Box and D.R. Cox. An analysis of transformations (with discussion). *Journal of the Royal Statistical Society, Series B*, 26:211–246, 1964.

G.E.P. Box and N. R. Draper. *Empirical Model Building and Response Surfaces*. Wiley, 1987.

G.E.P. Box and C. A. Fung. Some considerations in estimating data transformations. MRC Report 2609, University of Wisconsin-Madison, 1983.

G.E.P. Box and C. A. Fung. Studies in quality improvement: Minimizing transmitted variation by parameter design. Report 8, University of Wisconsin-Madison, 1986.

G.E.P. Box, W.G. Hunter, and J.S. Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis and Model Building*. John Wiley, New York, 1978.

G.E.P. Box and R. D. Meyer. An analysis for unreplicated fractional factorials. *Technometrics*, 28:1–18, 1986.

G.E.P. Box. Signal-to-noise ratios, performance criteria, and transformations. *Technometrics*, pages 1–17, 1988.

K. N. Berk and R. R. Picard. Significance tests for saturated orthogonal arrays. *Journal of Quality Technology*, 23:79–89, 1992.

G.E.P. Box and K.B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society*, Series B, 13, 1, 1951.

J.M. Chambers and T.J. Hastie. *Statistical Models in S*. Wadsworth and Brooks Cole Advanced Books and Software, Pacific Grove, CA, 1992.

D. E. Coleman and D. C. Montgomery. A systematic approach to planning for a designed industrial experiment (with discussion). *Technometrics*, 35:1–27, 1993.

C. Daniel. Use of half-normal plots in interpreting factorial two-level experiments. *Technometrics*, 1:311–341, 1959.

C. Daniel. *Applications of Statistics to Industrial Experimentation*. Wiley, New York, 1976.

P. Diamond. *Practical Experiment Designs for Engineers and Scientists*. Life-time Learning Publications, Belmont, CA, 1981.

A. Dong. On the identification of active contrasts in unreplicated fractional factorials. *Statistica Sinica*, 3:209–217, 1993.

J. Engel. Modeling variation in industrial experiments. *Applied Statistics*, 41:579–593, 1992.

P. Haaland. *Experimental Design in Biotechnology*. Marcel Dekker, New York, 1989.

P. Haaland and M. A. O'Connell. Inference for effect saturated fractional factorials. Submitted to Technometrics, 1994.

P.W.M. John. Some non-orthogonal fractions of 2^n designs. *Journal of the Royal Statistical Society*, Series B, 31:270–275, 1969.

Kacker. Off-line quality control, parameter design, and the Taguchi method (with discussion). *Journal of Quality Technology*, 17:176–209, 1985.

W. Keating, C. Donahue, C. Nycz, J. Shram, J. Weng, P. Haaland, S. Jurgensen, J. Nadeau, and M. Little. Enhanced performance of a sensitive rapid method for the detection of *mycobacterium tuberculosis* using strand displacement amplification. In *Proceedings of the 93rd meeting of the American Society for Microbiology*, 1993.

R. V. Lenth. Quick and easy analysis of unreplicated factorials. *Technometrics*, 31:469–473, 1989.

R.F. Liddle and P.D. Haaland. Efficient nonstandard screening designs. In *Annual Meetings of the American Statistical Association*, New Orleans, LA, 1988.

R. H. Myers, A. I. Khuri, and W. H. Carter, Jr. Response surface methodology: 1966–1988. *Technometrics*, 31:137–158, 1989.

V. J. Nair. On the behavior of some estimators from probability plots. *Journal of the American Statistical Association*, 79:469–473, 1984.

R.L. Plackett and J.P. Burman. The design of optimum multifactorial experiments. *Biometrika*, 33:305–325, 1946.

M. Phadke. *Quality Engineering using Robust Design*. Prentice Hall, New Jersey, 1989.

W.R. Stephenson, F. L. Hulting, and K. Moore. Posterior probabilities for identifying active effects in unreplicated experiments. *Journal of Quality Technology*, 21:202–212, 1989.

G Taguchi. *Introduction to Quality Engineering: Designing quality into productes and processes*. Asian Productivity Organization, Tokyo, 1986.

S. Weisberg. *Applied Linear Regression*. John Wiley, New York, second edition, 1985.

D. A. Zahn. An empirical study of the half-normal plot. *Technometrics*, 17:201–211, 1975.