ABSTRACT

In this tutorial we show how to use SAS® to design experiments which combine both blocking structure and split-plots.

The standard theory of incomplete block designs assumes no structure on its treatments (PROC PLAN). Standard fractional factorials come with very restrictive block sizes (PROC FACTEX). We meld these traditions in a rich and realistic manner.

Most industrial experiments are split-plots. Material from one (main-plot) experiment becomes the blocks for another (split-plot) experiment. We show how to simultaneously fractionate both the main-plot factors and the split-plot factors. Consider how you might design the following real experiment:

Study the effect of four two-level factors and their six two-factor interactions. Normally we would do this with a 2**4, but in this situation the material comes in blocks of size 3. We have enough material for 13 such blocks. These 13 blocks themselves consist of a 2**3 design with 5 replicated points.

The design strategies we describe have made significant contributions to the development of the company’s volleyball team schedule.

We will discuss and distribute the SAS code which generates these designs. The code uses data step programming, macro programming, and PROC OPTEX.

INTRODUCTION

Industry has a tremendous, on-going need to learn about complex and variable processes. Many industrial statisticians invest a significant proportion of their time teaching engineers how to design and analyze traditional fractional factorial experiments. Kahn and Bancroft (1992) describe a systematic way of teaching engineers how to do much of this work without needing to consult with a statistician. The underlying design methodology is to think carefully about the factors and interactions of interest, generate a full factorial design, give the OPTEX procedure these candidate points and the model of interest, replicate some of the design points to have an estimate of pure error, and produce attractive and fully randomized run sheets.

The beauty of D-optimal design software such as PROC OPTEX is its universality. Whatever your model, linear, quadratic, categorical, and arbitrary interactions, engineers can follow a single procedure to arrive at a reasonable design.

Here is code which will generate a design for a model with two class effects, three continuous variables, and selected quadratic and interaction effects. We have decided to use 5 points more than the saturated design. These additional points can be used in the analysis to check for lack of model fit:

```sas
data samp;
  do temp=100,125,150;
    do pres=20,30;
      do operator='Diccon', 'Bill';
        do machine='New', 'Old';
          do speed=1,3.5;
            cand_id+1;
            output;
          end;
        end;
      end;
    end;
  end;
run;
```

The candidate points are given to PROC OPTEX along with the model of interest.

```sas
proc OPTEX data=samp seed=8956325;
  class operator machine;
  id cand_id;
  model temp pres operator machine speed temp*temp temp*pres temp*speed pres*speed operator*machine operator*speed;
  output out=samp2;
  generate n=17 method=federov;
run;
```

We now select 5 points at random for replication. These are used in the analysis to estimate replicate variability, or pure error.

1276
Now rerandomize all the points twice, once to determine the sequence in which we make objects and once to determine their labels.

Now make attractive work sheets.

While this code and method is quite general, we are aware of four main limitations in using this specific D-optimal design methodology. First, the design produced is strongly model dependent. A small change in the model of interest can produce a quite different design. In practice the model of interest is not really fixed. In the analysis stage it is quite typical to have effects of only partial interest, which when they prove not to be important get pooled into the error term. In this reduced space the D-optimal design is likely no longer optimal. The classic designs, while often requiring many more points than needed for estimability, generally seem to achieve model robustness.

The second limitation is in generating designs with a blocking structure. To generate a blocked design with D-optimal software we typically include the blocking factor as a regular factor to be estimated. The algorithm then invests design points in estimating the blocking effects, even though the researcher has no interest in these effects. Indeed the traditional efficiency of standard blocked models is calculated with respect to the non-block effects only (Cochran and Cox). The D-optimal algorithm in PROC OPTEX does not do this.

Here is typical code used to generate a blocked design using PROC OPTEX.

The determination of the information matrix of the non D-optimal designs is 4/3 that of the D-optimal ones. We are surprised that the design excluding the (+1,+1,+1) point is not optimal, even though it splits evenly into 3 +1 and 3 -1 levels of each of the A, B and C factors.

The second limitation is in generating designs with a blocking structure. To generate a blocked design with D-optimal software we typically include the blocking factor as a regular factor to be estimated. The algorithm then invests design points in estimating the blocking effects, even though the researcher has no interest in these effects. Indeed the traditional efficiency of standard blocked models is calculated with respect to the non-block effects only (Cochran and Cox). The D-optimal algorithm in PROC OPTEX does not do this.

Here is typical code used to generate a blocked design using PROC OPTEX.
These first two limitations are connected. In the first we have only partial interest in some effects, in the second we have no interest in the block effects.

The 4 blocks in the previous example may have a structure on them. For example blocks 1 and 3 may be run on machine A while blocks 2 and 4 may be run on machine B. Placing this structure on the blocks has no effect on the efficiency of the design with respect to the temperature and operator effects. We simply are partitioning the 3 df block effect into a 1 df machine effect and a 2 df error term. The operator and temperature effects are still tested with respect to the same 1 df error. (Note that for an actual experiment we usually allocate more degrees of freedom to error than we show in this teaching example.)

We have just built a classic split-plot design. When we imposed a structure on the blocks, they became main-plots, sometimes called whole-plots. Temperature and operator are split-plot effects, machine is the main-plot effect.

In industrial experiments some factors are often very easy to change and thus can be easily randomized. Other factors, such as oven temperature, can be very slow to change and thus difficult to randomize over. Those factors difficult to change are used to block the design. Split-plot designs are very common.

The third and fourth limitations, related to plot size constraints and treatment balance, we will discuss later.

**SIMPLE SPLIT-PILOT DESIGNS**

Consider an experiment where we will study the effects of shift and oven. There are three shifts and each shift can run each of the three ovens. Because of production constraints, each shift can only run one oven of experimental material. A full factorial would generate 9 runs. We believe the shift by oven interaction is negligible and so decide to fractionate. The saturated design would use 5 runs. We will add a 6th df for lack-of-model estimation and a 7th df for an estimate of pure error.

```plaintext
OBS  SHIFT  OVEN DAY  PLOTID
 1     1    A    1    1
 2     2    C    1    3
 3     3    C    1    6
 4     1    B    2    2
 5     2    C    2    4
 6     3    A    2    7
 7     2    B    3    5
```

We have just built the main plot structure. We consider each of these plots as a block when we construct the split-plot structure. Let's say each oven run can bake multiple ingots. Each ingot has three two-level factors: size, material, and orientation. We think size may interact with material and also that it may interact with the main-plot factor oven. The saturated design has 13 ingots. Because of the cost of ingots we decide to use just 2 additional ingots for lack-of-fit df and 2 more for split-plot replication error.
How does one analyze the results from such an experiment? The nesting level of the factor effects is clear from the structure. It is possible to develop a practical analysis. Theoretical purity is harder to come by, unless the design is balanced. A starting point is to fit the main-plot effects, adjust for the blocking, and go on to split-plot effects. The split-plot analysis is correct, the main-plot one is at best suggestive.

The continuation of this analysis is a topic for another tutorial.

This model supports all the main effects and interactions for both the split-plot and main-plot main levels and interactions between the split and main plot effects that were specified in the PROC OPTEX model statement.

Here is the complete design. Note that for 15 runs in 7 blocks we expect at least one block to have three or more runs (plotid # 4). The replicated runs occur in plotids 6 and 7, as can be seen from the X values.
open to question. Better estimation of a main-by-splitplot factor interaction may require modifying the main-plot block assignments. We think that this is not a major difficulty. In many experiments the main-plots have less replication and greater variability than the split-plots. The design at the main-plot level is then of much greater concern than that at the split-plot level; this is the design addressed first in our method.

Because this is a tutorial, and because most people learn best with real examples we will design one more split-plot. Consider the design of a 2**4 fractional factorial where all 6 two-factor interactions are of interest. We have to run the experiment in blocks of size 3. We have only enough material for a total of 13 blocks. These 13 blocks are actually the main-plots making up a 2**3 with 5 replicated points. Here are the code and macros which generates the 39 points which make up this experiment.

* Stage 1 factors A B C, stage 2 D E F G;
* all 2 factor interactions within stages are of interest, as well as A*D B*D B*E;

```plaintext
title 'A good design at stage 1';
* robust: 2**3 half fraction + 1 other;
data stagldes;
do A= -1, 1;
do B:-1, 1;
do C=-1, 1;
sl_trial+l; mainplot+l; output;
if sltrial in ( 1, 4, 6, 7, 8) then 
do; mainplot+l; output: end:
end; end; end;
title 'The stage 2 trials';
proc factex;
factors D E F G;
model estimate=(D E F G @2):
output out=stag2cnd designrep=stagldes;
run;
* label the stage 2 treatment combinations;
proc sort data=stag2cnd;
by D E 'F G mainplot;
data stag2cnd; set stag2cnd;
by D E F G mainplot;
if first.G br first.F or first.E then trial+l;
run;
* search directly for a good design;
title 'TRY1 model mainplot DIEIFIG @2 A*D B*D B*E';
proc optex data=stag2cnd seed=214710043:
id trial C;
class mainplot;
model mainplot DIEIFIG @2 A*D B*D B*E;
generate n=39 initdesign=try1 method=M_Federov iter=20;
run;
************************************************************************************************
* either from tables or by search;
title2 'Block design TRY3 with model mainplot trial';
proc optex data=stag2cnd seed=214710043:
class mainplot trial;
model mainplot trial;
generate n=39 initdesign=try1 method=M_Federov iter=20:
output out=try2;
run;
* add in the factors A-G;
proc sort data=try2;
by trial mainplot;
data try3;
merge stag2cnd try2 (in=use);
by trial mainplot;
if use;
proc freq data=try3;
tables mainplot trial D E F G A*D B*D B*E/list nocum;
run;
* stage 2 as a pure block design;
data stag2des;
set try3;
drop a b c sltrial;
rename mainplot=block;
* master list of the 16 stage 2 treatments;
proc sort data=stag2des(drop=block);
output out=stag22 (rename=(trial=treat))
by trial;
data stag2t; set stag2t; by treat;
if first.treat;
run;
* random assignment of factor combinations;
title2 'Block design TRY3 with model mainplot DIEIFIG @2 A*D B*D B*E';
proc optex data=stag2cnd seed=214710043:
class mainplot;
model mainplot DIEIFIG @2 A*D B*D B*E;
generate n=39 initdesign=try3 method=sequential;
run;
* add in the factors A-G;
proc sort data=try2;
by trial mainplot;
data try3;
merge stag2cnd try2 (in=use);
by trial mainplot;
if use;
proc freq data=try3;
tables mainplot trial D E F G A*D B*D B*E/list nocum;
run;
* randomise blocks labels for main plots;
data tempmain;
set stagldes endzlastrec;
do i~1 to nn;
let seed=%eval(&init-'incr+& incr*'i):
randomise blocks labels for main plots:
data tempmain;
set stagldes end=lastrec;
retain seed &seed; drop seed;
call ranuni(seed.x);
if lastrec then call
symput('nextseed',trim(left{seed)));
proc sort data=tempmain;
by x;
data tempmain;
set tempmain;
drop x;
block+l;
* match split-plot blocks to main plots;
```

1280
data deslp213;
    merge stag2des (keep=block trial) tempmain; by block;
    if _n=1 then put
      * mesh &i seed &seed nextseed &nextseed*
    randomise treatment labels: split plots;
    data tempsplit;
    set stag2t (drop=treat);
    x=ranuni(&nextseed);
    proc sort data=tempsplt;
    by x;
    data tempsplt;
    set tempsplt;
    drop x;
    trial+1;
    * add in the split plot treatments;
    proc sort data=tempsplt; by trial;
    data deslp213;
    merge deslp213 tempsplt; by trial;
    run;
    * evaluate the combined design:
    title "mesh &i seed &seed , combining stages 1, 2"
    proc optex data=deslp213 seed:::153403012;
    class mainplot;
    model mainplot DIEIFIG @2 A*D B*D B*E;
    generate n=39 initdesign=deslp213
    run;
    %end;
    proc printto;
    run;
    * read the PROC OPTEX output back in:
    data idexam12;
    infile meshing;
    retain test n seed d a g std:
    drop test garbage;
    input @2 test $ @;
    if test =>'mesh' then
      input n garbage $ seed;
    if test :e'l' then do;
      run;
    %mend;
    input d a 9 std; output:
    %deslp2( 500, 225410043, 3214,s13bt.out);
    * examine the results:
    options pageno=l:
    filename meshsum 's13bt.lis';
    proc printto print=meshsum;
    title 'Random matchings ';
    title2
    'model mainplot DIEIFIG @2 A*D B*D B*E';
    proc sort data=idexam12:
    by descending d descending a descending g std:
    run;
    * print out the 100 top D designs;
    options obs=100;
    proc print data=idexam12;
    run;
    options obs=max:
    * reset the number of observations;
    * the D, A criteria depend on the
candidate set only via scaling;
    * but G and ave std.dev. of prediction
depend on it strongly;
    * look at the results;
    proc univariate data=idexam12 plot normal;
    id n;
    var d a;
    proc printto;
    run;

VOLLEYBALL TOURNAMENTS

The fourth limitation of D-optimal computer search algorithms is our difficulty in specifying certain kinds of
treatment balance, as in the model robustness paragraph
above. There is a large literature on incomplete block
designs. For certain values of the number of
(unstructured) treatments t, common block size k and
common treatment replication r there are particularly
attractive balanced incomplete block designs (BIBs).
These BIBs have the following balance property: the
number of times a treatment occurs in a block with
another is the same for all treatment pairs. These
designs are typically D-optimal with respect to an
additive block and treatment effect model. D-optimality
searches often miss these known designs, particularly
when the number of treatments t is large and the block
size k is small. In this case the number of blocks is large
and the search arduous because there are so many locally
optimal designs. We would like a general way of
generating designs close to these balanced designs for
any values of the design parameters.

In the Gore volleyball league we will have something like
15 teams and 5 courts. Each Thursday evening during
the summer three teams compete on each of the 5 courts.
They play a round robin on each court. We want a
schedule where each team plays each of the other 14
teams twice over a 14 Thursday season. Ideally the
schedule will place each team 3 times on 4 courts and
twice on the fifth court. Of course every team has to play
every week at 1 and only 1 court.

We could model this with a 15 level team effect, a 5 level
court effect, and a 14 level week effect. At any given
court on any night we expect three teams to show up, so
that there are a total of 3 teams * 5 courts * 14 nights or
210 runs in this design. To balance team with respect to
court we include the team·court interaction (56 degrees
of freedom). To have each team used each night we
model the team·night interaction (182 degrees of
freedom). At this stage we are modeling more degrees of
freedom than we have runs: PROC OPTEX will not
work. When we specify fewer terms in the model, the
designs PROC OPTEX finds are not satisfactory: teams
show up at two courts on the same night, or two teams
show up at one court and four at another. In this
particular case there are BIB arrangements for team
to court by night allocations. One is plan 11.24 (Cochran
and Cox) for 7 nights repeated twice. Each team plays
every other team once in plan 11.24. Allocation of
blocks to courts can be done in a moderately balanced
way through an ad-hoc search.
The relevance of this example: brute force is no substitute for knowledge and insight. For volleyball tournaments as we understand them, there is no general procedure. For each special case, go to the design literature in the references.

CONCLUSION

Efficient experimentation is a mainstay of competitive innovation and manufacturing. D-optimal algorithms give engineers high flexibility in specifying and designing experiments which address their specific needs. It is possible to use the technique in two stages to design split-plot experiments. We have supplied code which demonstrates use of PROC OPTEX in those situations. The D-optimal algorithm has four main limitations. First, they fail to provide model robustness, producing models which are marginally better for the stated model but significantly worse for closely related models which are also of interest. Second, they do not have the ability to discount the information being learned about the blocking factors, despite our lack of interest. Third, they are unable to explicitly support block size constraints which are common restrictions in experimentation. And fourth, they produce designs which estimate effects, not contrasts whereas many experiments have as their main interest prespecified contrasts. Volleyball tournaments are an example of such designs. The standard literature on BIBs is still required to produce good tournament designs.

REFERENCES


The authors can be reached at W. L. Gore & Associates, Inc., 297 Blue Ball Road, Elkton, MD, 21922-1130.

SAS is a registered trademark of SAS Institute, Inc. in the USA and other countries. © indicates USA registration.