Chapter 8

Experimental Designs for Fitting Response Surfaces — II

8.1 Evaluating design efficiencies

Here are some ideas for evaluating efficiencies of designs. Let's start with a single rep of a 2^3 design.

```
#### 8.1
df.8.1 <- read.table(text="
x1 x2 x3
-1 -1 -1
 1 -1 -1
-1  1 -1
 1  1 -1
-1 -1  1
 1 -1  1
-1  1  1
 1  1  1
", header=TRUE)
str(df.8.1)
```

```
## 'data.frame': 8 obs. of  3 variables:
##  $ x1: int  -1 1 -1 1 -1 1 -1 1
##  $ x2: int  -1 -1 1 1 -1 -1 1 1
##  $ x3: int  -1 -1 -1 -1 1 1 1 1
```

You can use the package `AlgDesign` function `eval.design()` to evaluate the D and A efficiencies of the design. The efficiency depends on the model function you specify.

```
library(AlgDesign)
eval.design(~ x1 + x2 + x3, df.8.1)
```

```
## $determinant
## [1] 1
##
## $A
## [1] 1
##
## $diagonality
## [1] 1
##
## $gmean.variances
## [1] 1
```

```
# these can be assigned to variables
D.eff.lin <-     eval.design(~ x1 + x2 + x3, df.8.1)$determinant
# note that A-efficiency is 1/A given from eval.design()
A.eff.lin <- 1 / eval.design(~ x1 + x2 + x3, df.8.1)$A
c(D.eff.lin, A.eff.lin)
```

```
## [1] 1 1
```

Now we calculate the same quantities after adding 3 center points.

```
df.8.1c <- rbind(df.8.1, data.frame(x1 = rep(0,3), x2 = rep(0,3), x3 = rep(0,3)))
df.8.1c
```

```
##    x1 x2 x3
## 1  -1 -1 -1
## 2   1 -1 -1
## 3  -1  1 -1
## 4   1  1 -1
## 5  -1 -1  1
## 6   1 -1  1
## 7  -1  1  1
## 8   1  1  1
## 9   0  0  0
## 10  0  0  0
## 11  0  0  0
```

```
library(AlgDesign)
eval.design(~ x1 + x2 + x3, df.8.1c)
```

```
## $determinant
## [1] 0.7875
##
## $A
## [1] 1.281
##
## $diagonality
## [1] 1
##
## $gmean.variances
## [1] 1.375
```

```
D.eff.lin <-     eval.design(~ x1 + x2 + x3, df.8.1c)$determinant
A.eff.lin <- 1 / eval.design(~ x1 + x2 + x3, df.8.1c)$A
c(D.eff.lin, A.eff.lin)
```

```
## [1] 0.7875 0.7805
```

8.2 Augmenting experimental designs

Sometimes it is desirable to augment a design in an optimal way. Suppose it is expected that a linear model will suffice for the experiment and so the design in `df.8.2` is run.

```
#### 8.2
df.8.2 <- read.table(text="
x1 x2
-1 -1
 1 -1
-1  1
 0  0
 0  0
", header=TRUE)
str(df.8.2)
```

```
## 'data.frame': 5 obs. of  2 variables:
##  $ x1: int  -1 1 -1 0 0
##  $ x2: int  -1 -1 1 0 0
```

Later, the experimenters realized that a second-order design would be more appropriate. They constructed a sensible list of design points to include (based on a face-centered CCD).

Then they evaluate the star points at a number of values (0.5, 1.0, 1.5, 2.0), and find the D-optimal augmentation at each.

```
for (i.alpha in c(0.5, 1.0, 1.5, 2.0)) {
  # augment design candidate points
  D.candidate.points <- i.alpha * data.frame(x1 = c( 1, -1,  0,  0)
                                           , x2 = c( 0,  0,  1, -1))

  # combine original points with candidate points
  D.all <- rbind(df.8.2, D.candidate.points)

  # keep the original points and augment the design with 4 points
  library(AlgDesign)
  D.aug <- optFederov(~ quad(.), D.all, nTrials = dim(df.8.2)[1] + 4, rows = 1:dim(df.8.2)[1]
                      , augment = TRUE, criterion = "D", maxIteration = 1e4, nRepeats = 1e2)
  ## same as:
  #D.aug <- optFederov(~ x1 + x2 + x1:x2 + x1^2 + x2^2
  #                    , D.all, nTrials = dim(df.8.2)[1] + 4, rows = 1:dim(df.8.2)[1]
  #                    , augment = TRUE, criterion = "D", maxIteration = 1e4, nRepeats = 1e2

  # Added points
  print(D.aug$design[(dim(df.8.2)[1]+1):dim(D.aug$design)[1],])

  library(AlgDesign)
  # quadratic
  #rm(D.eff.lin, A.eff.lin)
```

```
  D.eff.lin <-     eval.design(~ quad(.), D.aug$design)$determinant
  A.eff.lin <- 1 / eval.design(~ quad(.), D.aug$design)$A
  print(c(i.alpha, D.eff.lin, A.eff.lin))
}
```

```
##     x1   x2
## 6  0.5  0.0
## 7 -0.5  0.0
## 8  0.0  0.5
## 9  0.0 -0.5
## [1] 0.50000 0.18812 0.05287
##   x1 x2
## 6  1  0
## 7 -1  0
## 8  0  1
## 9  0 -1
## [1] 1.0000 0.3813 0.2556
##     x1   x2
## 6  1.5  0.0
## 7 -1.5  0.0
## 8  0.0  1.5
## 9  0.0 -1.5
## [1] 1.5000 0.6584 0.4209
##   x1 x2
## 6  2  0
## 7 -2  0
## 8  0  2
## 9  0 -2
## [1] 2.0000 1.1079 0.6066
```

Funding comes through and they don't have to just add the four axial points, but can afford to add any 12 points they like to the initial run. They look at a range of combinations of points for x_1 and x_2 and find the D-optimal augmentation.

```
# augment design candidate points
D.candidate.points <- expand.grid(x1 = seq(-2, 2, by = 0.5)
                                  , x2 = seq(-2, 2, by = 0.5))
str(D.candidate.points)
```

```
## 'data.frame': 81 obs. of  2 variables:
##  $ x1: num  -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 -2 ...
##  $ x2: num  -2 -2 -2 -2 -2 -2 -2 -2 -2 -1.5 ...
##  - attr(*, "out.attrs")=List of 2
##   ..$ dim     : Named int  9 9
##   .. ..- attr(*, "names")= chr  "x1" "x2"
##   ..$ dimnames:List of 2
##   .. ..$ x1: chr  "x1=-2.0" "x1=-1.5" "x1=-1.0" "x1=-0.5" ...
##   .. ..$ x2: chr  "x2=-2.0" "x2=-1.5" "x2=-1.0" "x2=-0.5" ...
```

```
head(D.candidate.points)
```

```
##     x1 x2
```

```
## 1 -2.0 -2
## 2 -1.5 -2
## 3 -1.0 -2
## 4 -0.5 -2
## 5  0.0 -2
## 6  0.5 -2
```

```
tail(D.candidate.points)
```

```
##      x1 x2
## 76 -0.5  2
## 77  0.0  2
## 78  0.5  2
## 79  1.0  2
## 80  1.5  2
## 81  2.0  2
```

```
# combine original points with candidate points
D.all <- rbind(df.8.2, D.candidate.points)

# keep the original points and augment the design with 4 points
library(AlgDesign)
D.aug <- optFederov(~ quad(.), D.all, nTrials = dim(df.8.2)[1] + 12, rows = 1:dim(df.8.2)[1]
                    , augment = TRUE, criterion = "D", maxIteration = 1e4, nRepeats = 1e2)

# Added points
print(D.aug$design[(dim(df.8.2)[1]+1):dim(D.aug$design)[1],])
```

```
##      x1   x2
## 6  -2.0 -2.0
## 7  -1.5 -2.0
## 13  1.5 -2.0
## 14  2.0 -2.0
## 42 -2.0  0.0
## 50  2.0  0.0
## 69 -2.0  1.5
## 77  2.0  1.5
## 78 -2.0  2.0
## 82  0.0  2.0
## 83  0.5  2.0
## 86  2.0  2.0
```

```
library(AlgDesign)
# quadratic
#rm(D.eff.lin, A.eff.lin)
D.eff.lin <-     eval.design(~ quad(.), D.aug$design)$determinant
A.eff.lin <- 1 / eval.design(~ quad(.), D.aug$design)$A
print(c(i.alpha, D.eff.lin, A.eff.lin))
```

```
## [1] 2.000 2.517 1.026
```

```
# full design
library(plyr)
D.aug.ordered <- arrange(D.aug$design, x1, x2)
D.aug.ordered
```

```
##      x1   x2
```

```
## 1  -2.0 -2.0
## 2  -2.0  0.0
## 3  -2.0  1.5
## 4  -2.0  2.0
## 5  -1.5 -2.0
## 6  -1.0 -1.0
## 7  -1.0  1.0
## 8   0.0  0.0
## 9   0.0  0.0
## 10  0.0  2.0
## 11  0.5  2.0
## 12  1.0 -1.0
## 13  1.5 -2.0
## 14  2.0 -2.0
## 15  2.0  0.0
## 16  2.0  1.5
## 17  2.0  2.0
```

8.3 Example 8.10, p. 390

The `AlgDesign` package can be used to create computer-generated designs, providing a list of permitted/suggested design points.

D-optimal $3 \times 2 \times 2$ for $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{11} x_1^2 + \varepsilon$.

```
#### 8.10
# design candidate values  3 * 2 * 2
D.candidate.points <- expand.grid(x1 = seq(-1, 1, by = 1)
                                  , x2 = c(-1, 1)
                                  , x3 = c(-1, 1))
D.candidate.points
```

```
##    x1 x2 x3
## 1  -1 -1 -1
## 2   0 -1 -1
## 3   1 -1 -1
## 4  -1  1 -1
## 5   0  1 -1
## 6   1  1 -1
## 7  -1 -1  1
## 8   0 -1  1
## 9   1 -1  1
## 10 -1  1  1
## 11  0  1  1
## 12  1  1  1
```

```
# choose the 12 points based on D-criterion
library(AlgDesign)
D.gen <- optFederov(~ x1 + x2 + x3 + x1^2, D.candidate.points, nTrials = 12
                    , criterion = "D", evaluateI = TRUE, maxIteration = 1e4, nRepeats = 1e2)
D.gen
```

```
## $D
## [1] 0.9036
##
## $A
## [1] 1.125
##
## $I
## [1] 4
##
## $Ge
## [1] 0.889
##
## $Dea
## [1] 0.882
##
## $design
##    x1 x2 x3
## 1  -1 -1 -1
## 2   0 -1 -1
## 3   1 -1 -1
```

```
## 4  -1  1 -1
## 5   0  1 -1
## 6   1  1 -1
## 7  -1 -1  1
## 8   0 -1  1
## 9   1 -1  1
## 10 -1  1  1
## 11  0  1  1
## 12  1  1  1
##
## $rows
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```
A.eff <- 1/D.gen$A # A efficiency is the reciprocal of £A

# easier to read when ordered
library(plyr)
D.gen.ordered <- arrange(D.gen$design, x1, x2, x3)
D.gen.ordered
```

```
##    x1 x2 x3
## 1  -1 -1 -1
## 2  -1 -1  1
## 3  -1  1 -1
## 4  -1  1  1
## 5   0 -1 -1
## 6   0 -1  1
## 7   0  1 -1
## 8   0  1  1
## 9   1 -1 -1
## 10  1 -1  1
## 11  1  1 -1
## 12  1  1  1
```

D-optimal $5 \times 2 \times 2$ for $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{11} x_1^2 + \varepsilon$.

```
#### 8.10
# design candidate values  5 * 2 * 2
D.candidate.points <- expand.grid(x1 = seq(-1, 1, by = 0.5)
                                  , x2 = c(-1, 1)
                                  , x3 = c(-1, 1))

D.candidate.points
```

```
##      x1 x2 x3
## 1  -1.0 -1 -1
## 2  -0.5 -1 -1
## 3   0.0 -1 -1
## 4   0.5 -1 -1
## 5   1.0 -1 -1
## 6  -1.0  1 -1
## 7  -0.5  1 -1
## 8   0.0  1 -1
## 9   0.5  1 -1
## 10  1.0  1 -1
## 11 -1.0 -1  1
```

```
## 12 -0.5 -1  1
## 13  0.0 -1  1
## 14  0.5 -1  1
## 15  1.0 -1  1
## 16 -1.0  1  1
## 17 -0.5  1  1
## 18  0.0  1  1
## 19  0.5  1  1
## 20  1.0  1  1
```

```
# choose the 12 points based on D-criterion
library(AlgDesign)
D.gen <- optFederov(~ x1 + x2 + x3 + x1^2, D.candidate.points, nTrials = 12
                    , criterion = "D", evaluateI = TRUE, maxIteration = 1e4, nRepeats = 1e2)
D.gen
```

```
## $D
## [1] 0.9306
##
## $A
## [1] 1.083
##
## $I
## [1] 3.667
##
## $Ge
## [1] 0.923
##
## $Dea
## [1] 0.92
##
## $design
##      x1 x2 x3
## 1  -1.0 -1 -1
## 4   0.5 -1 -1
## 5   1.0 -1 -1
## 6  -1.0  1 -1
## 7  -0.5  1 -1
## 10  1.0  1 -1
## 11 -1.0 -1  1
## 12 -0.5 -1  1
## 15  1.0 -1  1
## 16 -1.0  1  1
## 19  0.5  1  1
## 20  1.0  1  1
##
## $rows
##  [1]  1  4  5  6  7 10 11 12 15 16 19 20
```

```
A.eff <- 1/D.gen$A # A efficiency is the reciprocal of £A

# easier to read when ordered
library(plyr)
D.gen.ordered <- arrange(D.gen$design, x1, x2, x3)
```

```
D.gen.ordered
```

```
##      x1 x2 x3
## 1  -1.0 -1 -1
## 2  -1.0 -1  1
## 3  -1.0  1 -1
## 4  -1.0  1  1
## 5  -0.5 -1  1
## 6  -0.5  1 -1
## 7   0.5 -1 -1
## 8   0.5  1  1
## 9   1.0 -1 -1
## 10  1.0 -1  1
## 11  1.0  1 -1
## 12  1.0  1  1
```

I-optimal $5 \times 2 \times 2$ for $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{11} x_1^2 + \varepsilon$.

```
#### 8.10
# design candidate values  5 * 2 * 2
D.candidate.points <- expand.grid(x1 = seq(-1, 1, by = 0.5)
                                  , x2 = c(-1, 1)
                                  , x3 = c(-1, 1))
D.candidate.points
```

```
##      x1 x2 x3
## 1  -1.0 -1 -1
## 2  -0.5 -1 -1
## 3   0.0 -1 -1
## 4   0.5 -1 -1
## 5   1.0 -1 -1
## 6  -1.0  1 -1
## 7  -0.5  1 -1
## 8   0.0  1 -1
## 9   0.5  1 -1
## 10  1.0  1 -1
## 11 -1.0 -1  1
## 12 -0.5 -1  1
## 13  0.0 -1  1
## 14  0.5 -1  1
## 15  1.0 -1  1
## 16 -1.0  1  1
## 17 -0.5  1  1
## 18  0.0  1  1
## 19  0.5  1  1
## 20  1.0  1  1
```

```
# choose the 12 points based on D-criterion
library(AlgDesign)
D.gen <- optFederov(~ x1 + x2 + x3 + x1^2, D.candidate.points, nTrials = 12
                    , criterion = "I", evaluateI = TRUE, maxIteration = 1e4, nRepeats = 1e2)
D.gen
```

```
## $D
## [1] 0.9306
##
```

```
## $A
## [1] 1.083
##
## $I
## [1] 3.667
##
## $Ge
## [1] 0.923
##
## $Dea
## [1] 0.92
##
## $design
##      x1 x2 x3
## 1  -1.0 -1 -1
## 4   0.5 -1 -1
## 5   1.0 -1 -1
## 6  -1.0  1 -1
## 7  -0.5  1 -1
## 10  1.0  1 -1
## 11 -1.0 -1  1
## 12 -0.5 -1  1
## 15  1.0 -1  1
## 16 -1.0  1  1
## 19  0.5  1  1
## 20  1.0  1  1
##
## $rows
##  [1]  1  4  5  6  7 10 11 12 15 16 19 20
```

```
A.eff <- 1/D.gen$A # A efficiency is the reciprocal of £A

# easier to read when ordered
library(plyr)
D.gen.ordered <- arrange(D.gen$design, x1, x2, x3)
D.gen.ordered
```

```
##      x1 x2 x3
## 1  -1.0 -1 -1
## 2  -1.0 -1  1
## 3  -1.0  1 -1
## 4  -1.0  1  1
## 5  -0.5 -1  1
## 6  -0.5  1 -1
## 7   0.5 -1 -1
## 8   0.5  1  1
## 9   1.0 -1 -1
## 10  1.0 -1  1
## 11  1.0  1 -1
## 12  1.0  1  1
```

A-optimal $5 \times 2 \times 2$ for $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{11} x_1^2 + \varepsilon$.

```r
#### 8.10
# design candidate values
D.candidate.points <- expand.grid(x1 = seq(-1, 1, by = 0.5)
                                  , x2 = c(-1, 1)
                                  , x3 = c(-1, 1))
#D.candidate.points

# choose the 12 points based on A-criterion
library(AlgDesign)
D.gen <- optFederov(~ x1 + x2 + x3 + x1^2, D.candidate.points, nTrials = 12
                    , criterion = "A", evaluateI = TRUE, maxIteration = 1e4, nRepeats = 1e2)
D.gen
```

```
## $D
## [1] 0.9306
##
## $A
## [1] 1.083
##
## $I
## [1] 3.667
##
## $Ge
## [1] 0.923
##
## $Dea
## [1] 0.92
##
## $design
##      x1 x2 x3
## 1  -1.0 -1 -1
## 2  -0.5 -1 -1
## 5   1.0 -1 -1
## 6  -1.0  1 -1
## 9   0.5  1 -1
## 10  1.0  1 -1
## 11 -1.0 -1  1
## 14  0.5 -1  1
## 15  1.0 -1  1
## 16 -1.0  1  1
## 17 -0.5  1  1
## 20  1.0  1  1
##
## $rows
##  [1]  1  2  5  6  9 10 11 14 15 16 17 20
```

```r
A.eff <- 1/D.gen$A # A efficiency is the reciprocal of £A

# easier to read when ordered
library(plyr)
D.gen.ordered <- arrange(D.gen$design, x1, x2, x3)
D.gen.ordered
```

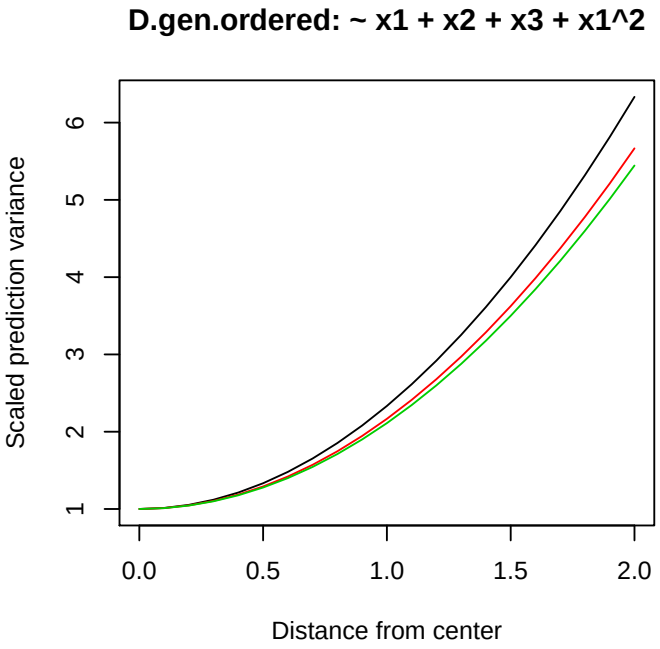
```
##      x1 x2 x3
```

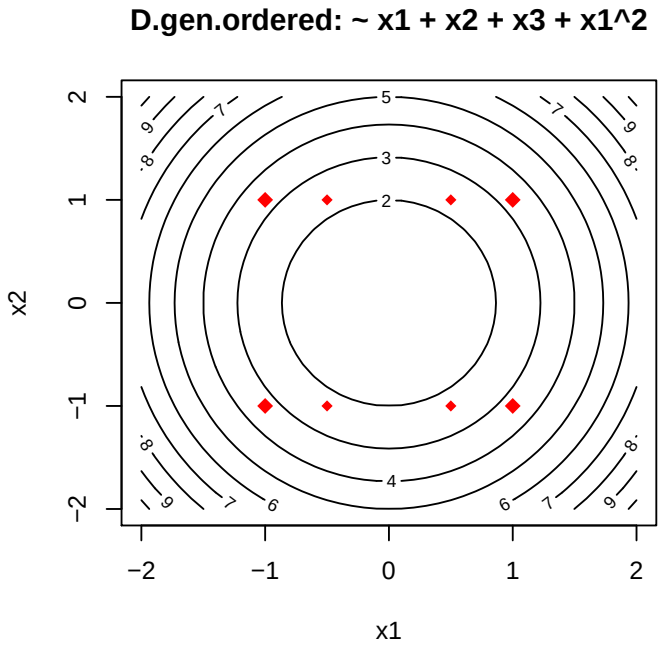
```
## 1  -1.0 -1 -1
## 2  -1.0 -1  1
## 3  -1.0  1 -1
## 4  -1.0  1  1
## 5  -0.5 -1 -1
## 6  -0.5  1  1
## 7   0.5 -1  1
## 8   0.5  1 -1
## 9   1.0 -1 -1
## 10  1.0 -1  1
## 11  1.0  1 -1
## 12  1.0  1  1
```

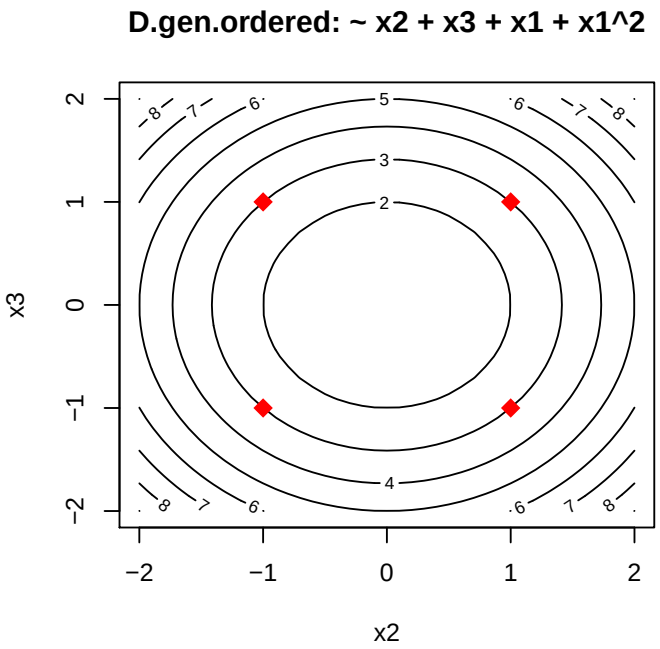
All of these choose the same designs.

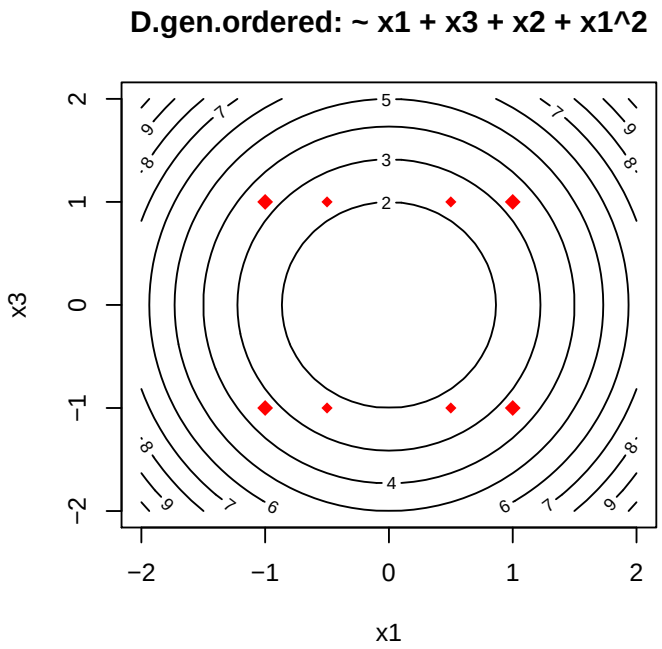
The function `varfcn()` computes the scaled variance function for a design, based on a specified model, and plots it either as a function of the radius in a specified direction or as a contour plot.

```
library(rsm)
par(mfrow = c(2,2))
varfcn(D.gen.ordered, ~ x1 + x2 + x3 + x1^2)
# the contour plots will plot the first two variables in the formula
varfcn(D.gen.ordered, ~ x1 + x2 + x3 + x1^2, contour = TRUE)
varfcn(D.gen.ordered, ~ x2 + x3 + x1 + x1^2, contour = TRUE)
varfcn(D.gen.ordered, ~ x1 + x3 + x2 + x1^2, contour = TRUE)
```


D.gen.ordered: ~ x1 + x2 + x3 + x1^2
Scaled prediction variance
Distance from center
0.0
0.5
1.0
1.5
2.0
1
2
3
4
5
6


D.gen.ordered: ~ x1 + x2 + x3 + x1^2
x2
x1
−2
−1
0
1
2
2
3
4
5
6
7
8
9


D.gen.ordered: ~ x2 + x3 + x1 + x1^2
x3
x2
−2
−1
0
1
2
2
3
4
5
6
7
8


D.gen.ordered: ~ x1 + x3 + x2 + x1^2
x3
x1
−2
−1
0
1
2
2
3
4
5
6
7
8
9