```
> #
```

We will use some functions form the package `conf.design`, which should be loaded along with Stat5303libs. This package does manipulations of design generators to get the designs we need.

```
> gen1 <- t(c(1,1,0,1));gen1
```

We are going to need matrices with a column for each factor and a row for each generator. 1 means that the factor is in the generator, 0 means that the factor is not in. The row is for ABD.

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    1
> gen2 <- matrix(c(1,0,1,0,0,1,1,1),nrow=2);gen2
```

Two generators, ABD and CD.

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    1
[2,]    0    0    1    1
> gen3 <- rbind(c(1,1,0,1),c(0,0,1,1));gen3
```

Same thing another way.

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    1
[2,]    0    0    1    1
> gen4 <- rbind(c(1,1,0,1),c(1,1,1,1));gen4
```

This one is for ABD and ABCD.

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    1
[2,]    1    1    1    1

> conf.design(gen1,2)
```

There is a column for our two blocks, and then we see the eight factor level combinations in each block.

Note, for reasons I do not understand, this seems to throw an error occasionally. I have been able to force it to work via `as.data.frame(conf.design(gen1,2))`. If you get an error you can try this trick for other situations, too.

```
   Blocks T1 T2 T3 T4
1       0  0  0  0  0
2       0  1  1  0  0
3       0  0  0  1  0
4       0  1  1  1  0
5       0  1  0  0  1
6       0  0  1  0  1
7       0  1  0  1  1
8       0  0  1  1  1
9       1  1  0  0  0
10      1  0  1  0  0
11      1  1  0  1  0
12      1  0  1  1  0
13      1  0  0  0  1
14      1  1  1  0  1
15      1  0  0  1  1
16      1  1  1  1  1
```

```
> conf.design(c(1,1,1,1),2)
```

This is the usual blocking on ABCD. Note that one block has only even numbers of factors at the high level, and the other block has only odd numbers at the high level.

```
   Blocks T1 T2 T3 T4
1       0  0  0  0  0
2       0  1  1  0  0
3       0  1  0  1  0
4       0  0  1  1  0
5       0  1  0  0  1
6       0  0  1  0  1
7       0  0  0  1  1
8       0  1  1  1  1
9       1  1  0  0  0
10      1  0  1  0  0
11      1  0  0  1  0
12      1  1  1  1  0
13      1  0  0  0  1
14      1  1  1  0  1
15      1  1  0  1  1
16      1  0  1  1  1
```

```
> conf.design(gen4,2)
```

Now try with gen4, which had ABCD and ABD as generators. Blocks are now listed by a pair of 0/1 variables. Note that factor C is high in blocks 2 and 3 and low in blocks 1 and 4: C is confounded with blocks.

```
   Blocks T1 T2 T3 T4
1      00  0  0  0  0
2      00  1  1  0  0
3      00  1  0  0  1
4      00  0  1  0  1
5      01  0  0  1  0
6      01  1  1  1  0
7      01  1  0  1  1
8      01  0  1  1  1
9      10  1  0  1  0
10     10  0  1  1  0
11     10  0  0  1  1
12     10  1  1  1  1
13     11  1  0  0  0
14     11  0  1  0  0
15     11  0  0  0  1
16     11  1  1  0  1
```

```
> conf.set(gen4,2)
```

This function figures out the complete set of terms confounded with blocks. Here we see that C is also confounded.

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    1
[2,]    1    1    1    1
[3,]    0    0    1    0
```

```
> conf.set(gen2,2)
```

gen2 confounds ABD and CD, and thus ABC. That's better than what we got from gen4.

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    1
[2,]    0    0    1    1
[3,]    1    1    1    0
> conf.design(gen2,2)
```

Here are the blocks for gen2.

```
   Blocks T1 T2 T3 T4
1      00  0  0  0  0
2      00  1  1  0  0
3      00  1  0  1  1
4      00  0  1  1  1
5      01  0  0  1  0
6      01  1  1  1  0
7      01  1  0  0  1
8      01  0  1  0  1
9      10  1  0  0  0
10     10  0  1  0  0
11     10  0  0  1  1
12     10  1  1  1  1
13     11  1  0  1  0
14     11  0  1  1  0
15     11  0  0  0  1
16     11  1  1  0  1

> gen6 <- rbind(c(1,1,1,0,1,0,0,0),c(1,1,0,1,0,1,0,0),c(1,0,1,1,0,0,1,0),
  c(0,1,1,1,0,0,0,1));gen6
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    1    1    0    1    0    0    0
[2,]    1    1    0    1    0    1    0    0
[3,]    1    0    1    1    0    0    1    0
[4,]    0    1    1    1    0    0    0    1
```

```
> conf.set(gen6,2)
```

Suppose that you had to run a 2^8 design in 16 blocks of size 16. You would need four generators. This set means that the smallest confounded effect is a four factor interaction.

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
 [1,]    1    1    1    0    1    0    0    0
 [2,]    1    1    0    1    0    1    0    0
 [3,]    0    0    1    1    1    1    0    0
 [4,]    1    0    1    1    0    0    1    0
 [5,]    0    1    0    1    1    0    1    0
 [6,]    0    1    1    0    0    1    1    0
 [7,]    1    0    0    0    1    1    1    0
 [8,]    0    1    1    1    0    0    0    1
 [9,]    1    0    0    1    1    0    0    1
[10,]    1    0    1    0    0    1    0    1
[11,]    0    1    0    0    1    1    0    1
[12,]    1    1    0    0    0    0    1    1
[13,]    0    0    1    0    1    0    1    1
[14,]    0    0    0    1    0    1    1    1
[15,]    1    1    1    1    1    1    1    1
```

```
> conf.design(gen6,2)
    Blocks T1 T2 T3 T4 T5 T6 T7 T8
1     0000  0  0  0  0  0  0  0  0
2     0000  1  1  1  0  1  0  0  0
3     0000  1  1  0  1  0  1  0  0
4     0000  0  0  1  1  1  1  0  0
5     0000  1  0  1  1  0  0  1  0
6     0000  0  1  0  1  1  0  1  0
7     0000  0  1  1  0  0  1  1  0
8     0000  1  0  0  0  1  1  1  0
9     0000  0  1  1  1  0  0  0  1
10    0000  1  0  0  1  1  0  0  1
11    0000  1  0  1  0  0  1  0  1
12    0000  0  1  0  0  1  1  0  1
13    0000  1  1  0  0  0  0  1  1
14    0000  0  0  1  0  1  0  1  1
15    0000  0  0  0  1  0  1  1  1
16    0000  1  1  1  1  1  1  1  1
...
243   1111  0  0  1  0  0  1  0  0
244   1111  1  1  0  0  1  1  0  0
245   1111  0  1  0  0  0  0  1  0
246   1111  1  0  1  0  1  0  1  0
247   1111  1  0  0  1  0  1  1  0
248   1111  0  1  1  1  1  1  1  0
249   1111  1  0  0  0  0  0  0  1
250   1111  0  1  1  0  1  0  0  1
251   1111  0  1  0  1  0  1  0  1
252   1111  1  0  1  1  1  1  0  1
253   1111  0  0  1  1  0  0  1  1
254   1111  1  1  0  1  1  0  1  1
255   1111  1  1  1  0  0  1  1  1
256   1111  0  0  0  0  1  1  1  1
```

```
>
> dnpk <- read.table("dnpk.dat.txt",header=TRUE);dnpk
```

These data are from a 2^4 design replicated twice, blocked into four blocks of size 8 with dnpk confounded with blocks in both replicates. dnpk is thus completely confounded. Data are from Cochran and Cox.
Notice that in the first block of each replication, there are always an odd number of factors at the high level (either 1 or 3), whereas in block 2 of each replication there is always an even number of factors at the high level (0, 2, or 4).

```
   d n p k block rpl yield
1  1 1 2 1     1   1    45
2  1 1 1 2     1   1    55
3  2 1 1 1     1   1    53
4  1 2 2 2     1   1    36
5  2 2 1 2     1   1    41
6  2 2 2 1     1   1    48
7  2 1 2 2     1   1    55
8  1 2 1 1     1   1    42
9  2 1 2 1     2   1    50
10 1 2 1 2     2   1    44
11 2 1 1 2     2   1    43
12 1 1 2 2     2   1    51
13 2 2 2 2     2   1    44
14 1 1 1 1     2   1    58
15 2 2 1 1     2   1    41
16 1 2 2 1     2   1    50
17 1 1 2 1     1   2    39
18 1 1 1 2     1   2    50
19 2 1 1 1     1   2    42
20 1 2 2 2     1   2    43
21 2 2 1 2     1   2    34
22 2 2 2 1     1   2    52
23 2 1 2 2     1   2    44
24 1 2 1 1     1   2    47
25 2 1 2 1     2   2    52
26 1 2 1 2     2   2    43
27 2 1 1 2     2   2    52
28 1 1 2 2     2   2    56
29 2 2 2 2     2   2    54
30 1 1 1 1     2   2    57
31 2 2 1 1     2   2    42
32 1 2 2 1     2   2    39
> dnpk <- within(dnpk,{d <- as.factor(d);n <- as.factor(n);
    p <- as.factor(p);k <- as.factor(k)}))
> dnpk <- within(dnpk,{block <- as.factor(block);rpl<-as.factor(rpl)})
> fit1 <- lm(yield~rpl:block+d*n*p*k,data=dnpk);anova(fit1)
```

Here is the basic ANOVA. These data were set up with blocks numbered 1 and 2 in each replication, so the replication by block "interaction" actually enumerates all four blocks, with 3 degrees of freedom between the four blocks. We want treatments adjusted for blocks, and we did not quite get it here, because R wants to put two factor terms (in this case, all blocks is a two factor term) after main effects. In this case is does not matter, but in some strange cases it might. In those cases, we need to use the terms() function to get the terms in the order we want, or we need to make a single factor to enumerate all of the blocks.

Note that the four factor interaction dnpk does not even show up in this table. That is because it is confounded with blocks within each replication and has 0 degrees of freedom. It cannot be estimated because it is completely confounded with blocks.

```
Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq F value   Pr(>F)
d          1   2.00    2.00  0.0824 0.778258
n          1 325.12  325.12 13.3974 0.002572 **
p          1   6.12    6.12  0.2524 0.623205
k          1   4.50    4.50  0.1854 0.673303
rpl:block  3 126.38   42.13  1.7358 0.205538
d:n        1  32.00   32.00  1.3186 0.270083
d:p        1 242.00  242.00  9.9720 0.006982 **
n:p        1  78.13   78.13  3.2193 0.094393 .
d:k        1   6.13    6.13  0.2524 0.623205
n:k        1  32.00   32.00  1.3186 0.270083
p:k        1  24.50   24.50  1.0096 0.332058
d:n:p      1   2.00    2.00  0.0824 0.778258
d:n:k      1  10.13   10.13  0.4172 0.528774
d:p:k      1  15.13   15.13  0.6233 0.443007
n:p:k      1  32.00   32.00  1.3186 0.270083
Residuals 14 339.75   24.27
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> fit1
```

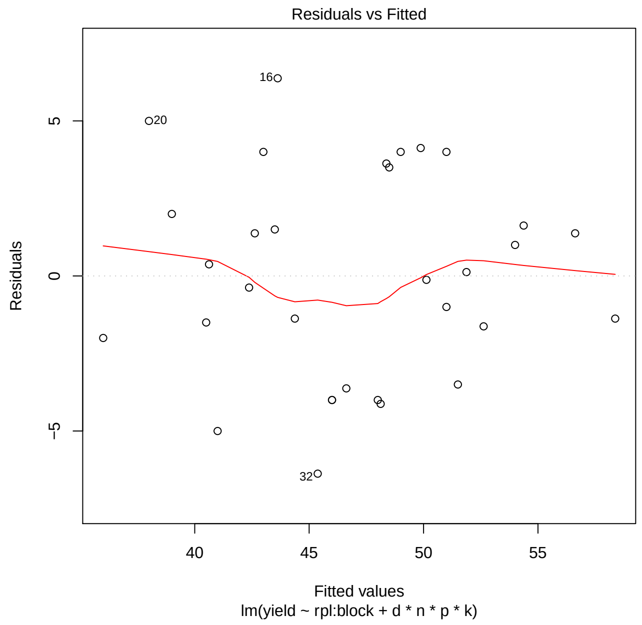
If you look at the coefficients you will see that we have missing for the four factor interaction.

```
Call:
lm.default(formula = yield ~ rpl:block + d * n * p * k)

Coefficients:
(Intercept)            d1            n1            p1            k1
    49.3750        0.2500        3.1875       -0.4375        0.3750
rpl1:block1  rpl2:block1  rpl1:block2  rpl2:block2          d1:n1
    -2.5000       -5.5000       -1.7500            NA        1.0000
      d1:p1         n1:p1         d1:k1         n1:k1         p1:k1
     2.7500        1.5625       -0.4375       -1.0000        0.8750
   d1:n1:p1      d1:n1:k1      d1:p1:k1      n1:p1:k1   d1:n1:p1:k1
    -0.2500       -0.5625        0.6875        1.0000            NA
```

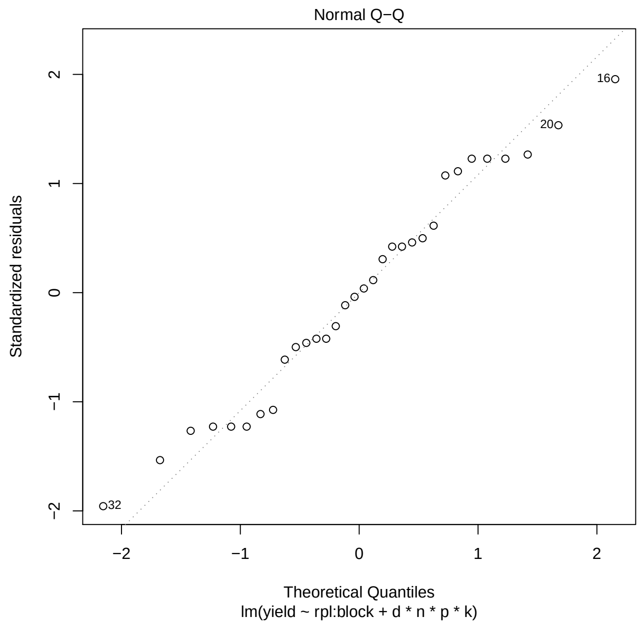
```
> plot(fit1,which=1)
```

Residuals don't look too bad. There is a bit of a tendency to decreasing errors, but no reasonable transformation helps.



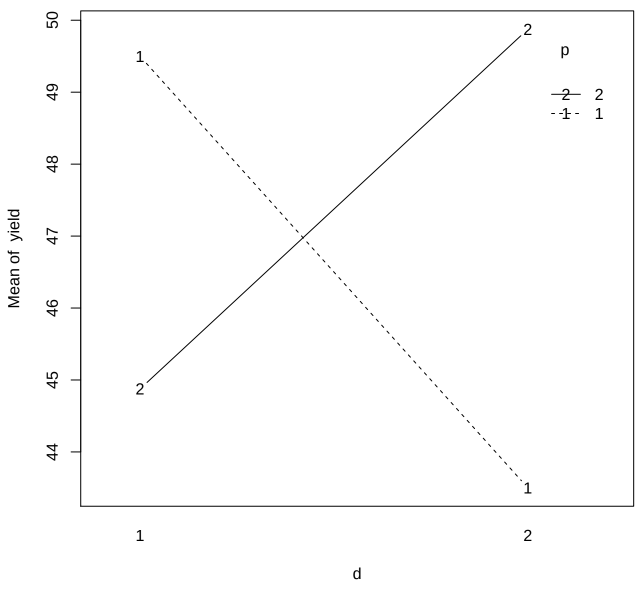
```
> plot(fit1,which=2)
```

Normality not bad either.



```
> with(dnpk,interactplot(d,p,yield))
```

Here is how we get a significant interaction without significant main effects.



```
> john <- read.table("john.dat.txt",header=TRUE);john
```

Data from John (1971). A 2^3 replicated four times and run in 8 blocks of 4. ABC, AB, AC, and BC are each confounded in one replication. Factors are sulfate of ammonia, sulfate of potash, and nitrogen; response is yield of potatoes in pounds per plot.

```
   a b c block yield
1  1 1 1     1   101
2  2 1 2     1   373
3  1 2 2     1   398
4  2 2 1     1   291
5  1 1 2     2   312
6  2 1 1     2   106
7  1 2 1     2   265
8  2 2 2     2   450
9  1 1 1     3   106
10 2 2 1     3   306
11 1 1 2     3   324
12 2 2 2     3   449
13 1 2 1     4   272
14 2 1 1     4    89
15 1 2 2     4   407
16 2 1 2     4   338
17 1 1 1     5    87
18 2 1 2     5   324
19 1 2 1     5   279
20 2 2 2     5   471
21 1 1 2     6   323
22 2 1 1     6   128
23 1 2 2     6   423
```

```
24 2 2 1     6   334
25 1 1 1     7   131
26 2 1 1     7   103
27 1 2 2     7   445
28 2 2 2     7   437
29 1 1 2     8   324
30 2 1 2     8   361
31 1 2 1     8   302
32 2 2 1     8   272
```

```
> john<-within(john,{a <- factor(a);b <- factor(b);c <- factor(c);block <- factor(block)})

> fit3 <- lm(yield~block+a*b*c,data=john)

> anova(fit3)
```

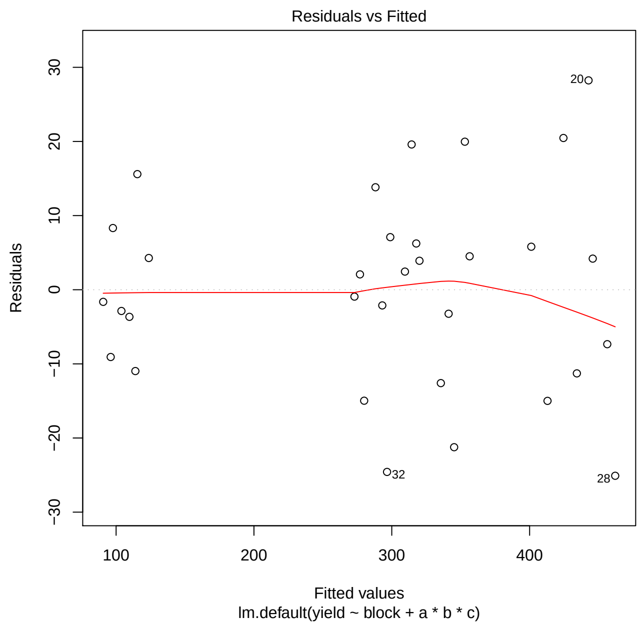
A, B, C and a couple of interactions are significant.

```
Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq  F value    Pr(>F)    
block      7   4499     643   2.0147  0.112834    
a          1   3465    3465  10.8624  0.004268 ** 
b          1 161170  161170 505.2090 4.404e-14 ***
c          1 278818  278818 873.9916 4.666e-16 ***
a:b        1     28      28   0.0883  0.769960    
a:c        1   1803    1803   5.6507  0.029457 *  
b:c        1  11528   11528  36.1366 1.402e-05 ***
a:b:c      1     45      45   0.1422  0.710737    
Residuals 17   5423     319                       
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

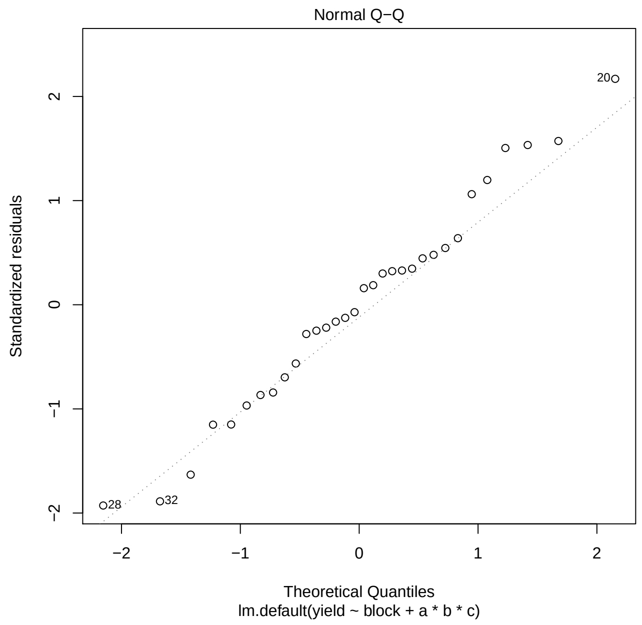
```
> plot(fit3,which=1)
```

A bit of increasing variance, but 1 is well within the Box-Cox interval.



```
> plot(fit3,which=2)
```

These are OK.



```
> summary(fit3)
```

Note that the standard errors for the A and AB effects are not the same. They would be the same in an RCB, for example. The difference is that A is never confounded, but AB is confounded in one of the four replications.

```
Call:
lm.default(formula = yield ~ block + a * b * c)

Residuals:
     Min       1Q   Median       3Q      Max
-25.0938  -9.5469   0.5729   6.4531  28.2396

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  291.594      3.157  92.352  < 2e-16 ***
block1        -2.219      9.115  -0.243  0.81059
block2        -6.969      9.115  -0.765  0.45501
block3         3.573      9.115   0.392  0.69993
block4       -14.010      9.115  -1.537  0.14267
block5       -10.010      9.115  -1.098  0.28740
block6        19.073      9.115   2.093  0.05170 .
block7         9.323      9.115   1.023  0.32072
a1           -10.406      3.157  -3.296  0.00427 **
b1           -70.969      3.157 -22.477 4.40e-14 ***
c1           -93.344      3.157 -29.563 4.67e-16 ***
a1:b1          1.083      3.646   0.297  0.76996
a1:c1          8.667      3.646   2.377  0.02946 *
b1:c1        -21.917      3.646  -6.011 1.40e-05 ***
a1:b1:c1       1.375      3.646   0.377  0.71074
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 17.86 on 17 degrees of freedom
Multiple R-squared: 0.9884,Adjusted R-squared: 0.9788
F-statistic: 103.3 on 14 and 17 DF,  p-value: 1.311e-13

> sqrt(4/3)*3.1574
```

Because AB is confounded in one of the replications, we have an effective samplesize of 3 instead of 4 when estimating AB effects. Thus the se for AB effects is a factor of $\sqrt{4/3}$ larger than that of the unconfounded effect.

```
(1)       3.6459
```

```
> lm(yield~block+a*b*c,data=john,subset=as.numeric(block) < 3)
```

The this model and the next one are not part of a standard analysis. They are merely presented to show that different terms are confounded in different replications. We confound ABC in the first replication (blocks 1 and 2).

```
Call:
lm.default(formula = yield ~ block + a * b * c, data = john,
    subset = as.numeric(block) < 3)

Coefficients:
(Intercept)       block1           a1           b1           c1        a1:b1        a1:c1
     287.00         3.75       -18.00       -64.00       -96.25         1.50        10.25
      b1:c1     a1:b1:c1
     -23.25           NA
```

```
> lm(yield~block+a*b*c,data=john,subset=as.numeric(block) > 6)
```

And we confound BC in the last replication (blocks 7 and 8).

```
Call:
lm.default(formula = yield ~ block + a * b * c, data = john,
    subset = as.numeric(block) > 6)

Coefficients:
(Intercept)       block1           a1           b1           c1        a1:b1        a1:c1
    296.875      -17.875        3.625      -67.125      -94.875       -5.875       10.875
      b1:c1     a1:b1:c1
         NA        5.375
```

```
> lm(yield~block+a*b*c,data=john,subset=as.numeric(block) > 2)
```

This model is fit to everything but the first replication. ABC is confounded in the first replication but not in the others. The estimate of ABC in the last three replications is the same as the estimate of ABC in the full model.

```
Call:
lm.default(formula = yield ~ block + a * b * c, data = john,
    subset = as.numeric(block) > 2)

Coefficients:
(Intercept)       block1       block2       block3       block4       block5           a1
    293.125        2.250      -15.750      -10.750       16.750        7.125       -7.875
         b1           c1        a1:b1        a1:c1        b1:c1     a1:b1:c1
    -73.292      -92.375        0.875        7.875      -21.250        1.375
```

```
> conf.design(c(1,1),3)
```

We can also use conf.design to confound a three series. Here we confound a 3^2 on A^1B^2.

```
  Blocks T1 T2
1      0  0  0
2      0  2  1
3      0  1  2
4      1  1  0
5      1  0  1
6      1  2  2
7      2  2  0
8      2  1  1
9      2  0  2
```

```
> gen9 <- rbind(c(1,0,2),c(1,1,0));gen9
```

Something a little bigger. Here we confound a 3^3 on A^1C^2 and A^1B^1.

```
     [,1] [,2] [,3]
[1,]    1    0    2
[2,]    1    1    0
> conf.set(gen9,3)
```

Full set of confounded effects.

```
     [,1] [,2] [,3]
[1,]    1    0    2
[2,]    1    1    0
[3,]    0    1    1
[4,]    1    2    1
> as.data.frame(conf.design(gen9,3))
```

Full design.

```
   Blocks T1 T2 T3
1      00  0  0  0
2      00  1  2  1
3      00  2  1  2
4      01  0  1  0
5      01  1  0  1
6      01  2  2  2
7      02  0  2  0
8      02  1  1  1
9      02  2  0  2
10     10  1  2  0
11     10  2  1  1
12     10  0  0  2
13     11  1  0  0
14     11  2  2  1
. . .
21     20  1  2  2
22     21  2  2  0
23     21  0  1  1
24     21  1  0  2
25     22  2  0  0
26     22  0  2  1
27     22  1  1  2
```