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

These are data from a (more or less) central composite design. The factors are amount of detergent, wash temperature, and agitation time; the response is whiteness of cotton clothes after washing. The "more or less" comes from the fact that they didn't quite get the design centered properly.

```
      d  w  a whiteness
1  0.09 29  7     50.97
2  0.21 29  7     82.73
3  0.09 29 13     53.55
4  0.21 29 13     92.03
5  0.09 52  7     67.40
6  0.21 52  7     97.28
7  0.09 52 13     76.00
8  0.21 52 13     96.86
9  0.05 41 10     46.95
10 0.25 41 10     92.64
11 0.15 41  5     77.91
12 0.15 41 15     90.68
13 0.15 21 10     70.53
14 0.15 60 10     83.79
15 0.15 41 10     85.20
16 0.15 41 10     83.35
17 0.15 41 10     85.55
18 0.15 41 10     87.34
19 0.15 41 10     92.09
20 0.15 41 10     86.85
> attach(wash)
> cd <- (d-.15)/.06;cd
```

It's usually best to work with "coded variables," which we get by subtracting out the center and dividing by the step.

```
 [1] -1.000000  1.000000 -1.000000  1.000000 -1.000000  1.000000 -1.000000
 [8]  1.000000 -1.666667  1.666667  0.000000  0.000000  0.000000  0.000000
[15]  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
> ca <- (a-10)/3;ca
 [1] -1.000000 -1.000000  1.000000  1.000000 -1.000000 -1.000000  1.000000
 [8]  1.000000  0.000000  0.000000 -1.666667  1.666667  0.000000  0.000000
[15]  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
> cw <- (w-40.5)/11.5;cw
```

Wash temperature was not quite correctly centered.

```
 [1] -1.00000000 -1.00000000 -1.00000000 -1.00000000  1.00000000
 [6]  1.00000000  1.00000000  1.00000000  0.04347826  0.04347826
[11]  0.04347826  0.04347826 -1.69565217  1.69565217  0.04347826
[16]  0.04347826  0.04347826  0.04347826  0.04347826  0.04347826

> cd2<-cd^2;cw2<-cw^2;ca2<-ca^2;cdca<-cd*ca;cdcw<-cd*cw;cacw<-ca*cw
```

Compute the second order terms.

```
> fit1 <- lm(whiteness~ cd+cw+ca+cd2+cw2+ca2+cdca+cdcw+cacw)
```

Fit the second order model.

```
> fit2 <- lm(whiteness~poly(cd,cw,ca,degree=2))
```

Fit the second order model a different way, using orthogonal polynomials.

```
> fit1
```

The coefficients for the two models are different, because the polynomials are somewhat different.

```
Call:
lm.default(formula = whiteness ~ cd + cw + ca + cd2 + cw2 + ca2 +     cdca + cdcw + cacw)

Coefficients:
(Intercept)           cd           cw           ca          cd2
    86.4510      14.5862       5.8666       3.0577      -5.9142
        cw2          ca2         cdca         cdcw         cacw
    -3.0713      -0.6942      -0.2875      -2.4608      -0.4386

> fit2

Call:
lm.default(formula = whiteness ~ poly(cd, cw, ca, degree = 2))

Coefficients:
                      (Intercept)  poly(cd, cw, ca, degree = 2)1.0.0
                           79.985                             53.506
poly(cd, cw, ca, degree = 2)2.0.0  poly(cd, cw, ca, degree = 2)0.1.0
                          -22.321                             22.009
poly(cd, cw, ca, degree = 2)1.1.0  poly(cd, cw, ca, degree = 2)0.2.0
                          -33.608                            -11.914
poly(cd, cw, ca, degree = 2)0.0.1  poly(cd, cw, ca, degree = 2)1.0.1
                           11.223                             -3.897
poly(cd, cw, ca, degree = 2)0.1.1  poly(cd, cw, ca, degree = 2)0.0.2
                           -5.990                             -2.620

> anova(fit1)
```

Inference for the two models is the same. I'll probably use this form rather than the orthogonal polynomial form, because it's a little easier to understand what we're working with.

Detergent and wash temperature are quadratic with an interaction. Agitation time has a linear effect, but does not appear to have a quadratic effect or to interact with the other variables. Agitation also seems to have a smaller effect than the other two.

```
Analysis of Variance Table

Response: whiteness
          Df  Sum Sq Mean Sq  F value    Pr(>F)
cd         1 2866.74 2866.74 225.6497 3.448e-08 ***
cw         1  488.58  488.58  38.4574 0.0001013 ***
ca         1  126.09  126.09   9.9252 0.0103254 *
cd2        1  442.08  442.08  34.7977 0.0001513 ***
cw2        1  134.74  134.74  10.6057 0.0086230 **
ca2        1    6.75    6.75   0.5313 0.4827718
cdca       1    0.66    0.66   0.0520 0.8241329
cdcw       1   48.48   48.48   3.8162 0.0792855 .
cacw       1    1.54    1.54   0.1212 0.7349360
Residuals 10  127.04   12.70
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> summary(fit2)

Call:
lm.default(formula = whiteness ~ poly(cd, cw, ca, degree = 2))

Residuals:
     Min       1Q   Median       3Q      Max
-4.09141 -1.77238 -0.02806  0.87618  5.38975

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                         79.985      0.797 100.357 2.36e-16 ***
poly(cd, cw, ca, degree = 2)1.0.0   53.506      3.564  15.011 3.47e-08 ***
poly(cd, cw, ca, degree = 2)2.0.0  -22.321      3.594  -6.210 0.000100 ***
poly(cd, cw, ca, degree = 2)0.1.0   22.009      3.564   6.175 0.000105 ***
poly(cd, cw, ca, degree = 2)1.1.0  -33.608     17.204  -1.954 0.079285 .
poly(cd, cw, ca, degree = 2)0.2.0  -11.914      3.596  -3.313 0.007844 **
poly(cd, cw, ca, degree = 2)0.0.1   11.223      3.564   3.149 0.010357 *
poly(cd, cw, ca, degree = 2)1.0.1   -3.897     17.082  -0.228 0.824133
poly(cd, cw, ca, degree = 2)0.1.1   -5.990     17.204  -0.348 0.734936
poly(cd, cw, ca, degree = 2)0.0.2   -2.620      3.594  -0.729 0.482772
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.564 on 10 degrees of freedom
Multiple R-squared: 0.9701,Adjusted R-squared: 0.9431
F-statistic:    36 on 9 and 10 DF,  p-value: 1.883e-06

> fit3 <- lm(whiteness~d+w+a+I(d^2)+I(w^2)+I(a^2)+I(d*w)+I(d*a)+I(w*a))
```

You can work with the original variables, but if they are not well-centered you can get very different p-values when testing due to colinearity in the predictors.

```
> summary(fit3)

Call:
lm.default(formula = whiteness ~ d + w + a + I(d^2) + I(w^2) +
    I(a^2) + I(d * w) + I(d * a) + I(w * a))

Residuals:
     Min       1Q   Median       3Q      Max
-4.09141 -1.77238 -0.02806  0.87618  5.38975

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.285e+01  2.773e+01  -3.348 0.007386 **
d            8.964e+02  1.303e+02   6.878 4.31e-05 ***
w            3.053e+00  7.319e-01   4.172 0.001913 **
a            3.316e+00  2.810e+00   1.180 0.265200
I(d^2)      -1.643e+03  2.645e+02  -6.210 0.000100 ***
I(w^2)      -2.322e-02  7.011e-03  -3.313 0.007844 **
I(a^2)      -7.713e-02  1.058e-01  -0.729 0.482772
I(d * w)    -3.566e+00  1.826e+00  -1.954 0.079285 .
I(d * a)    -1.597e+00  7.001e+00  -0.228 0.824133
I(w * a)    -1.271e-02  3.651e-02  -0.348 0.734936
```

```
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.564 on 10 degrees of freedom
Multiple R-squared: 0.9701,Adjusted R-squared: 0.9431
F-statistic:    36 on 9 and 10 DF,  p-value: 1.883e-06

> #
```

Residual plots and Box-Cox all show no problems.

```
> library(rsm)
```

Now that we've done all that work, here is another way to do it all that is probably easier (although it hides a few of the bits that you should know about). It uses the rsm library, which automatically does many of the things that you want to do.

```
> CD <- coded.data(wash,cw~(w-40.5)/11.5,ca~(a-10)/3,cd~(d-.15)/.06)
```

First we make up a coded data set. You need a data frame and the various formulas.

```
> CD
```

Just like we did by hand before.

```
          cd          cw        ca whiteness
1  -1.000000 -1.00000000 -1.000000     50.97
2   1.000000 -1.00000000 -1.000000     82.73
3  -1.000000 -1.00000000  1.000000     53.55
4   1.000000 -1.00000000  1.000000     92.03
5  -1.000000  1.00000000 -1.000000     67.40
6   1.000000  1.00000000 -1.000000     97.28
7  -1.000000  1.00000000  1.000000     76.00
8   1.000000  1.00000000  1.000000     96.86
9  -1.666667  0.04347826  0.000000     46.95
10  1.666667  0.04347826  0.000000     92.64
11  0.000000  0.04347826 -1.666667     77.91
12  0.000000  0.04347826  1.666667     90.68
13  0.000000 -1.69565217  0.000000     70.53
14  0.000000  1.69565217  0.000000     83.79
15  0.000000  0.04347826  0.000000     85.20
16  0.000000  0.04347826  0.000000     83.35
17  0.000000  0.04347826  0.000000     85.55
18  0.000000  0.04347826  0.000000     87.34
19  0.000000  0.04347826  0.000000     92.09
20  0.000000  0.04347826  0.000000     86.85

Variable codings ...
cw ~ (w - 40.5)/11.5
ca ~ (a - 10)/3
cd ~ (d - 0.15)/0.06
> fit4 <- rsm(whiteness ~ SO(cw,ca,cd),data=CD)
```

The rsm function does a standard response surface analysis. SO() expands to linear terms, squared terms, and cross product terms.

```
> summary(fit4)
```

Here is the summary in its full glory. It does the estimated effects with their se's, it gives an anova split by order and including a lack of fit test, and it does the canonical analysis including eigenvectors, eigenvalues, and the stationary point in both coded and original units. It's your complete entertainment experience.

We see three negative eigenvalues, indicating a surface with a maximum. The estimated maximum is somewhat outside our range of experimentation.

The first canonical variable is almost exactly the agitation variable. This might not be surprising, because the agitation variable was the least important and did not interact with the other two. The other two canonical variables are mostly combinations of detergent or wash temperature. Both canonical variables 2 and 3 are mostly one of detergent or temperature, with a bit of the other thrown in. The significant interaction observed above suggests that we should see these two mixed together in the canonical variables.

```
Call:
rsm(formula = whiteness ~ SO(cw, ca, cd), data = CD)

Residuals:
     Min       1Q   Median       3Q      Max
-4.09141 -1.77238 -0.02806  0.87618  5.38975

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  86.4510     1.4549  59.419 4.43e-14 ***
cw            5.8666     0.9611   6.104 0.000115 ***
ca            3.0577     0.9684   3.158 0.010199 *
cd           14.5862     0.9684  15.063 3.36e-08 ***
cw:ca        -0.4386     1.2597  -0.348 0.734936
cw:cd        -2.4608     1.2597  -1.954 0.079285 .
ca:cd        -0.2875     1.2602  -0.228 0.824133
cw^2         -3.0713     0.9271  -3.313 0.007844 **
ca^2         -0.6942     0.9524  -0.729 0.482772
cd^2         -5.9142     0.9524  -6.210 0.000100 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.564 on 10 degrees of freedom
Multiple R-squared: 0.9701,Adjusted R-squared: 0.9431
F-statistic:    36 on 9 and 10 DF,  p-value: 1.883e-06

Analysis of Variance Table

Response: whiteness
                Df Sum Sq Mean Sq F value    Pr(>F)
FO(cw, ca, cd)   3 3481.4 1160.47 91.3441 1.443e-07
TWI(cw, ca, cd)  3   50.7   16.89  1.3298  0.318985
PQ(cw, ca, cd)   3  583.6  194.52 15.3116  0.000456
Residuals       10  127.0   12.70
Lack of fit      5   82.8   16.55  1.8695  0.254436
Pure error       5   44.3    8.85

Stationary point of response surface:
       cw        ca        cd
0.3783154 1.8531410 1.1094070
```

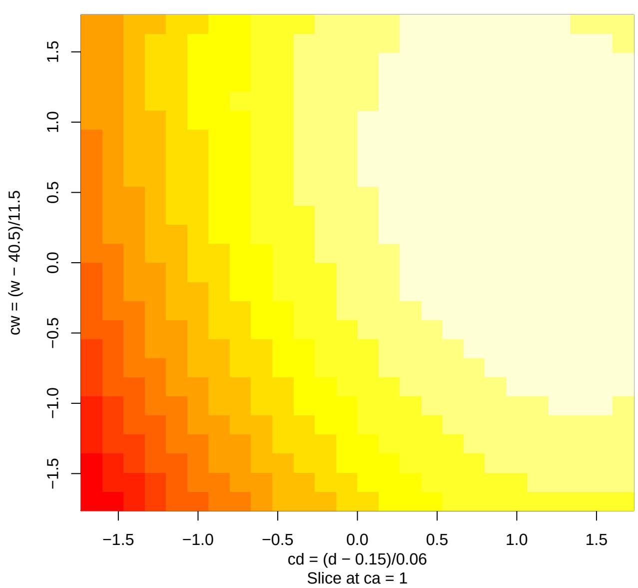
```
Stationary point in original units:
         w          a          d
44.8506268 15.5594230  0.2165644

Eigenanalysis:
$values
[1] -0.673919 -2.625170 -6.380608

$vectors
             [,1]       [,2]        [,3]
[1,]  0.087653547  0.9324826 -0.35041838
[2,] -0.996128187  0.0796676 -0.03717135
[3,]  0.006744651 -0.3523198 -0.93585536
```

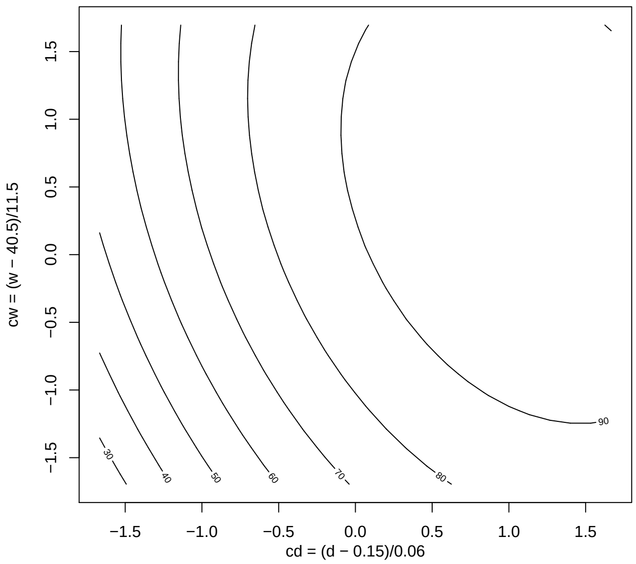
> **image(fit4,cw~cd,at=list(ca=1))**

There are several plotting functions. This is a simple example, and there are about a million optional arguments. This plots the the response as a function of cw (vertical) and cd (horizontal) with ca fixed at 1.



> **contour(fit4,cw~cd,at=list(ca=1))**

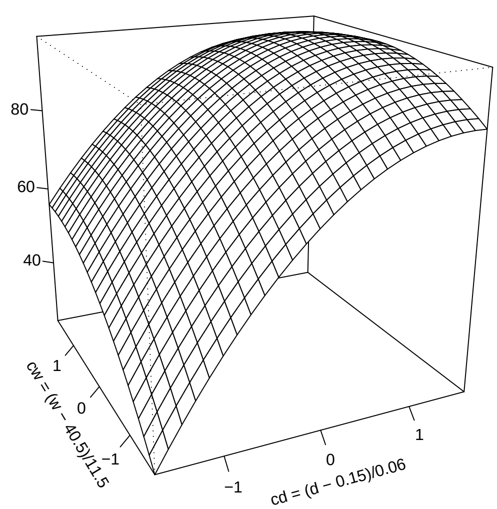
This time contours.



Slice at ca = 1

```
> persp(fit4,cw~cd,at=list(ca=1))
```

And now a perspective plot.



Slice at ca = 1

```
> solve.QP(-2*fit4$B,fit4$b,matrix(c(1,0,0),ncol=1),-100)
```

Optimizing a quadratic function subject to linear equality and/or inequality constraints is called quadratic programming. If you want to maximize the function, use -2 times matrix of second order terms and the linear terms as the first two arguments. If you want to minimize, use 2B and -b instead.
The constraints are "greater than or equal to" constraints. The third argument has a column of coefficients for every constraint, and the four argument has the lower bound. Here we're saying that the first argument must be at least -100, which seems safe enough. The constrained solution is the same as the unconstrained solution, because the unconstrained solution meets the constraint.

```
$solution
[1] 0.3783154 1.8531410 1.1094070

$value
[1] -12.03396

$unconstrained.solution
[1] 0.3783154 1.8531410 1.1094070

$iterations
[1] 1 0

$Lagrangian
[1] 0

$iact
[1] 0

> solve.QP(-2*fit4$B,fit4$b,matrix(c(1,0,0),ncol=1),-1,meq=1)
```

The first meq of the constraints are equality constraints. It defaults to zero, but here we say that the first variable must equal –1.

```
$solution
[1] -1.000000  2.231063  1.386971

$value
[1] -6.784151

$unconstrained.solution
[1] 0.3783154 1.8531410 1.1094070

$iterations
[1] 2 0

$Lagrangian
[1] 7.617714

$iact
[1] 1

> solve.QP(-2*fit4$B,fit4$b,matrix(c(1,0,0),ncol=1),.5)
```

Here we say that the first variable must be at least .5. The unconstrained solution has that variable at .378, so the constrained solution will be right on the boundary.

```
$solution
```

```
[1] 0.500000 1.819776 1.084902

$value
[1] -11.99304

$unconstrained.solution
[1] 0.3783154 1.8531410 1.1094070

$iterations
[1] 2 0

$Lagrangian
[1] 0.6725302

$iact
[1] 1

> solve.QP(-2*fit4$B,fit4$b,matrix(c(1,-1,0),ncol=1),.5)
```

You can use more interesting constraints like the first variable minus the second variable must be at least .5.

```
$solution
[1] 0.8256995 0.3256995 1.0534576

$value
[1] -10.11785

$unconstrained.solution
[1] 0.3783154 1.8531410 1.1094070

$iterations
[1] 2 0

$Lagrangian
[1] 1.940538

$iact
[1] 1
```