Homework 8 - AGST 5014

Igor Kuivjogi Fernandes and Ashmita Upadhyay

2023-04-27

1. The following experiment comes from a central composite design with 4 factors in 2 blocks. Conduct the proper analysis (including graphs, interpretation, etc).

```r
q1 <- read.csv("HW8_Q1.csv")
q1 <- transform(q1, block = factor(block), logSD = NULL)
str(q1)
```

```
## 'data.frame':    30 obs. of  6 variables:
##  $ block: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ x1   : int  -1 1 -1 1 -1 1 -1 1 -1 1 ...
##  $ x2   : int  -1 -1 1 1 -1 -1 1 1 -1 -1 ...
##  $ x3   : int  -1 -1 -1 -1 1 1 1 1 -1 -1 ...
##  $ x4   : int  -1 -1 -1 -1 -1 -1 -1 -1 1 1 ...
##  $ ave  : int  367 369 374 370 372 355 397 377 350 373 ...
```

Let's fit some Response Surface Methodology models.

First, we can start with a simple first order model:

```r
library(rsm)

mod1 <- rsm(ave ~ block + FO(x1, x2, x3, x4), data = q1)
summary(mod1)
```

```
##
## Call:
## rsm(formula = ave ~ block + FO(x1, x2, x3, x4), data = q1)
##
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 367.111111   1.883588 194.8999 < 2.2e-16 ***
## block2       -1.527778   2.978214  -0.5130  0.612652
## x1           -0.083333   1.631235  -0.0511  0.959680
## x2            5.083333   1.631235   3.1162  0.004701 **
## x3            0.250000   1.631235   0.1533  0.879476
## x4           -6.083333   1.631235  -3.7293  0.001041 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.499,  Adjusted R-squared:  0.3947
## F-statistic: 4.782 on 5 and 24 DF,  p-value: 0.003577
##
```

```
## Analysis of Variance Table
##
## Response: ave
##                     Df  Sum Sq Mean Sq F value   Pr(>F)
## block                1   16.81   16.81  0.2632 0.612652
## FO(x1, x2, x3, x4)   4 1510.00  377.50  5.9112 0.001856
## Residuals           24 1532.69   63.86
## Lack of fit         20 1521.94   76.10 28.3152 0.002594
## Pure error           4   10.75    2.69
##
## Direction of steepest ascent (at radius 1):
##          x1          x2          x3          x4
## -0.01050596  0.64086379  0.03151789 -0.76693536
##
## Corresponding increment in original units:
##          x1          x2          x3          x4
## -0.01050596  0.64086379  0.03151789 -0.76693536
```

The lack of fit is significant (p-value < 0.05), so we should include more complex terms.

Now a second-order model:

```
mod2 <- rsm(ave ~ block + SO(x1, x2, x3, x4), data = q1)
summary(mod2)
```

```
##
## Call:
## rsm(formula = ave ~ block + SO(x1, x2, x3, x4), data = q1)
##
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 372.800000   1.506375 247.4815 < 2.2e-16 ***
## block2       -2.950000   1.207787  -2.4425 0.0284522 *
## x1           -0.083333   0.636560  -0.1309 0.8977075
## x2            5.083333   0.636560   7.9856 1.398e-06 ***
## x3            0.250000   0.636560   0.3927 0.7004292
## x4           -6.083333   0.636560  -9.5566 1.633e-07 ***
## x1:x2        -2.875000   0.779623  -3.6877 0.0024360 **
## x1:x3        -3.750000   0.779623  -4.8100 0.0002773 ***
## x1:x4         4.375000   0.779623   5.6117 6.412e-05 ***
## x2:x3         4.625000   0.779623   5.9324 3.657e-05 ***
## x2:x4        -1.500000   0.779623  -1.9240 0.0749257 .
## x3:x4        -2.125000   0.779623  -2.7257 0.0164099 *
## x1^2         -2.037500   0.603894  -3.3739 0.0045424 **
## x2^2         -1.662500   0.603894  -2.7530 0.0155541 *
## x3^2         -2.537500   0.603894  -4.2019 0.0008873 ***
## x4^2         -0.162500   0.603894  -0.2691 0.7917877
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.9555, Adjusted R-squared:  0.9078
## F-statistic: 20.04 on 15 and 14 DF,  p-value: 6.54e-07
##
## Analysis of Variance Table
##
```

```
## Response: ave
##                      Df  Sum Sq Mean Sq F value    Pr(>F)
## block                 1   16.81   16.81  1.7281  0.209786
## FO(x1, x2, x3, x4)    4 1510.00  377.50 38.8175 1.965e-07
## TWI(x1, x2, x3, x4)   6 1114.00  185.67 19.0917 5.355e-06
## PQ(x1, x2, x3, x4)    4  282.54   70.64  7.2634  0.002201
## Residuals            14  136.15    9.72
## Lack of fit          10  125.40   12.54  4.6660  0.075500
## Pure error            4   10.75    2.69
##
## Stationary point of response surface:
##         x1         x2         x3         x4
##  0.8607107 -0.3307115 -0.8394866 -0.1161465
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1]  3.258222 -1.198324 -3.807935 -4.651963
##
## $vectors
##          [,1]       [,2]       [,3]        [,4]
## x1  0.5177048 0.04099358  0.7608371 -0.38913772
## x2 -0.4504231 0.58176202  0.5056034  0.45059647
## x3 -0.4517232 0.37582195 -0.1219894 -0.79988915
## x4  0.5701289 0.72015994 -0.3880860  0.07557783
```

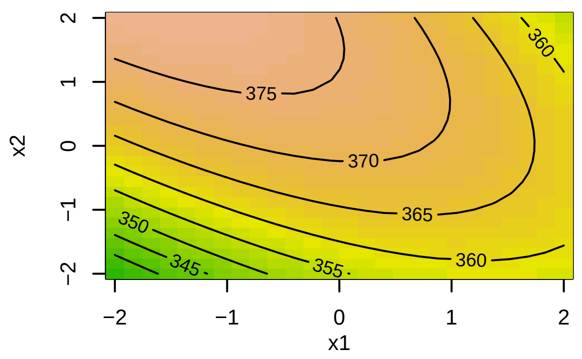
There are positive and negatives eigenvalues, which means we have a saddle point.
The adj.$R^2 = 0.9078$, and the lack of fit is not significant, so we can stick with this model.

The optimal experimental points are:

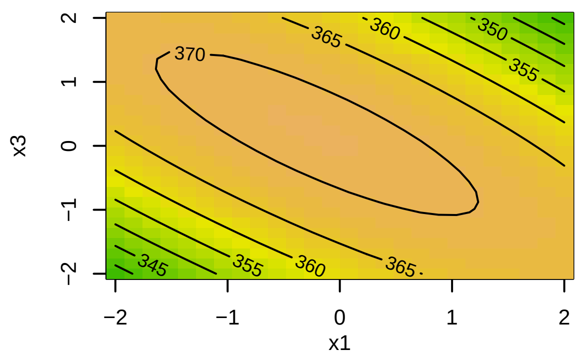
```
summary(mod2)$canonical$xs
```

```
##         x1         x2         x3         x4
##  0.8607107 -0.3307115 -0.8394866 -0.1161465
```

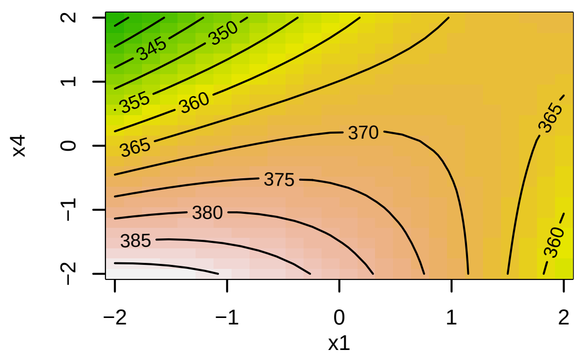
```
par(mfrow = c(3, 2))
contour(
  mod2,
  ~ x1 + x2 + x3 + x4,
  image = TRUE
)
```



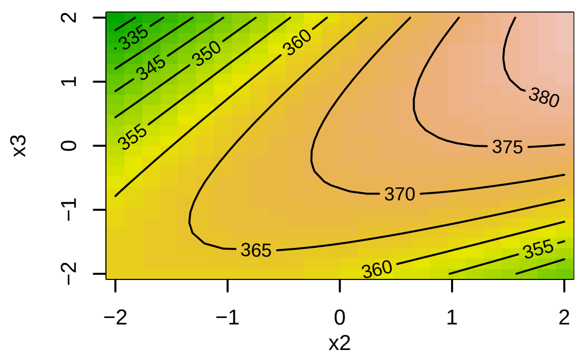
Slice at x3 = 0, x4 = 0



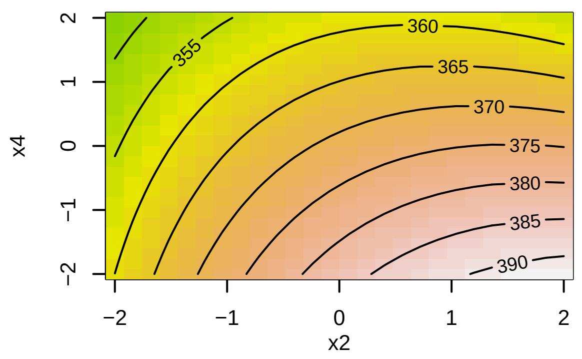
Slice at x2 = 0, x4 = 0



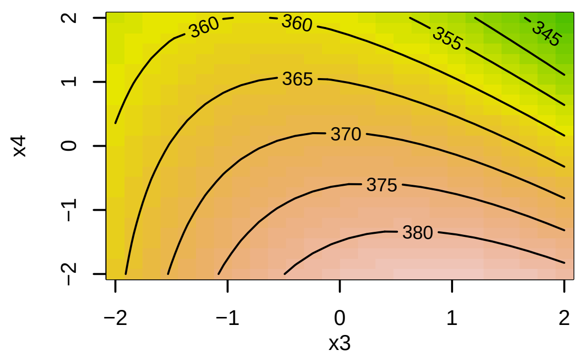
Slice at x2 = 0, x3 = 0



Slice at x1 = 0, x4 = 0



Slice at x1 = 0, x3 = 0



Slice at x1 = 0, x2 = 0

We have 6 plots because there are 4 two-way interactions (`x1:x2`, `x1:x3`, `x1:x4`, `x2:x3`, `x2:x4`, and `x3:x4`). From the plots we can see that the maximum point reached lies roughly in the interval 380-390.
We can check the distribution of the fitted values:

```
summary(mod2$fitted.values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   346.8   359.4   368.7   366.5   372.2   397.1
```

The lowest point was 346.8, whereas the maximum point was 397.1.

Now we can run a steepest-ascent algorithm to search for a better solution:

```r
steep <- steepest(mod2)
```

```
## Path of steepest ascent from ridge analysis:
```

```r
steep
```

```
##    dist     x1    x2    x3     x4 |    yhat
## 1   0.0  0.000 0.000 0.000  0.000 | 372.800
## 2   0.5 -0.127 0.288 0.116 -0.371 | 377.106
## 3   1.0 -0.351 0.538 0.312 -0.700 | 382.675
## 4   1.5 -0.595 0.775 0.526 -1.009 | 389.783
## 5   2.0 -0.846 1.007 0.745 -1.309 | 398.485
## 6   2.5 -1.101 1.237 0.966 -1.605 | 408.819
## 7   3.0 -1.356 1.465 1.189 -1.897 | 420.740
## 8   3.5 -1.613 1.693 1.413 -2.188 | 434.322
## 9   4.0 -1.870 1.920 1.637 -2.477 | 449.497
## 10  4.5 -2.127 2.147 1.862 -2.766 | 466.323
## 11  5.0 -2.385 2.373 2.086 -3.054 | 484.750
```

The optimal solution points now are:

```r
opt_points <- steep[which.max(steep$yhat), ]
opt_points
```

```
##    dist     x1    x2    x3     x4 |   yhat
## 11    5 -2.385 2.373 2.086 -3.054 | 484.75
```

These are the new points we would use in the process to get the higher response values.
If we do predictions using these new coefficients, we get the maximum predicted response found by the steepest-ascent algorithm above:

```r
grid <- expand.grid(
  block = unique(q1$block),
  x1 = opt_points$x1,
  x2 = opt_points$x2,
  x3 = opt_points$x3,
  x4 = opt_points$x4
)
predict(mod2, grid)
```

```
##        1        2
## 484.7497 481.7997
```

As we have 2 blocks, the model did two predictions. The first matches with the steepest-ascent algorithm.

2. The design below presents the yield of different common bean cultivars. There was a variable stand count in each plot. Conduct the proper analysis.

```
q2 <- read.csv("HW8_Q2.csv")
q2 <- transform(q2, block = factor(block), cv = factor(cv))
str(q2)
```

```
## 'data.frame':    28 obs. of  4 variables:
##  $ block: Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 2 2 2 ...
##  $ cv   : Factor w/ 7 levels "CNFP8018","CNFP8019",..: 1 2 3 4 5 6 7 1 2 3 ...
##  $ stand: int  95 124 106 92 101 98 103 103 114 116 ...
##  $ yield: num  1588 2012 1975 1838 1825 ...
```

```
table(q2$block)
```

```
## 
## 1 2 3 4 
## 7 7 7 7
```

The blocks are equally frequent.

The `stand` variable is a covariable, hence we can use ANCOVA to analyse the yield of different cultivars.

First step is to check whether `stand` is independent of the treatment `cv`.

```
check <- aov(stand ~ block + cv, data = q2)
summary(check)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)  
## block        3 1066.0   355.3   4.082 0.0225 *
## cv           6  790.7   131.8   1.514 0.2298  
## Residuals   18 1567.0    87.1                 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `cv` is not significant, so we expect `stand` and `cv` to not be related.

Next, we run ANCOVA with interaction. For ANCOVA, we should use Type III sum of squares.

```
check_inter <- lm(
  yield ~ block + cv * stand,
  contrasts = list(cv = contr.sum),
  data = q2
)
car::Anova(check_inter, type = 'III')
```

```
## Anova Table (Type III tests)
## 
## Response: yield
##             Sum Sq Df F value Pr(>F)
## (Intercept)  11688  1  0.2633 0.6180
## block         75844  3  0.5694 0.6466
## cv           488399  6  1.8334 0.1816
## stand        140453  1  3.1636 0.1029
## cv:stand     441062  6  1.6557 0.2217
## Residuals    488368 11
```

The interaction `cv:stand` is not significant, so we can go further and fit ANCOVA without the interaction:

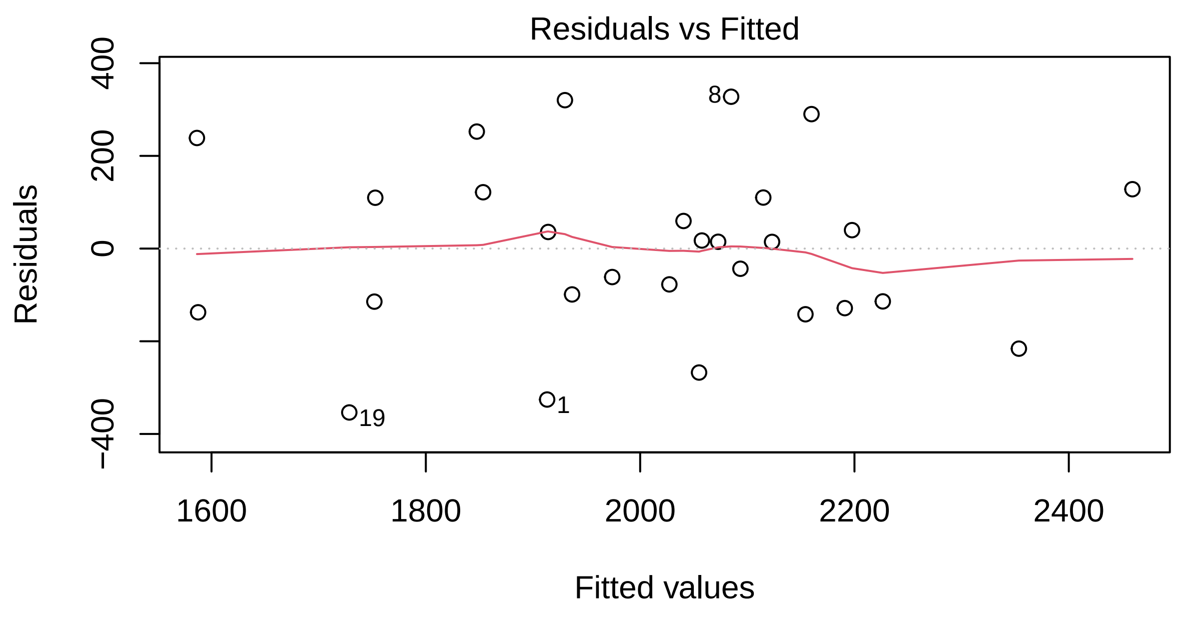
```
ancova <- lm(
  yield ~ block + cv + stand,
  contrasts = list(cv = contr.sum),
  data = q2
)
car::Anova(ancova, type = 'III')
```

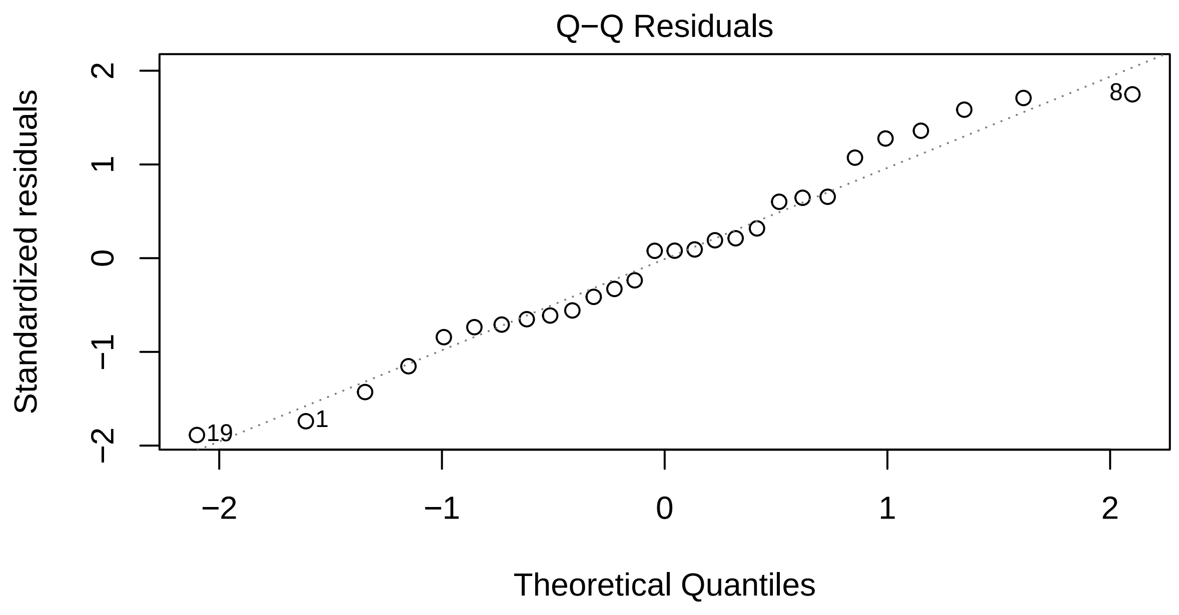
```
## Anova Table (Type III tests)
##
## Response: yield
##             Sum Sq Df F value  Pr(>F)
## (Intercept)  24622  1  0.4504 0.51118
## block       116917  3  0.7128 0.55769
## cv          899421  6  2.7419 0.04741 *
## stand       348326  1  6.3712 0.02184 *
## Residuals   929430 17
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `cv` is indeed significant, using a significance level of $\alpha = 0.05$.

Now let's check the usual ANOVA assumptions:

```
par(mfrow= c(2, 1))
plot(ancova, which = 1)
plot(ancova, which = 2)
```

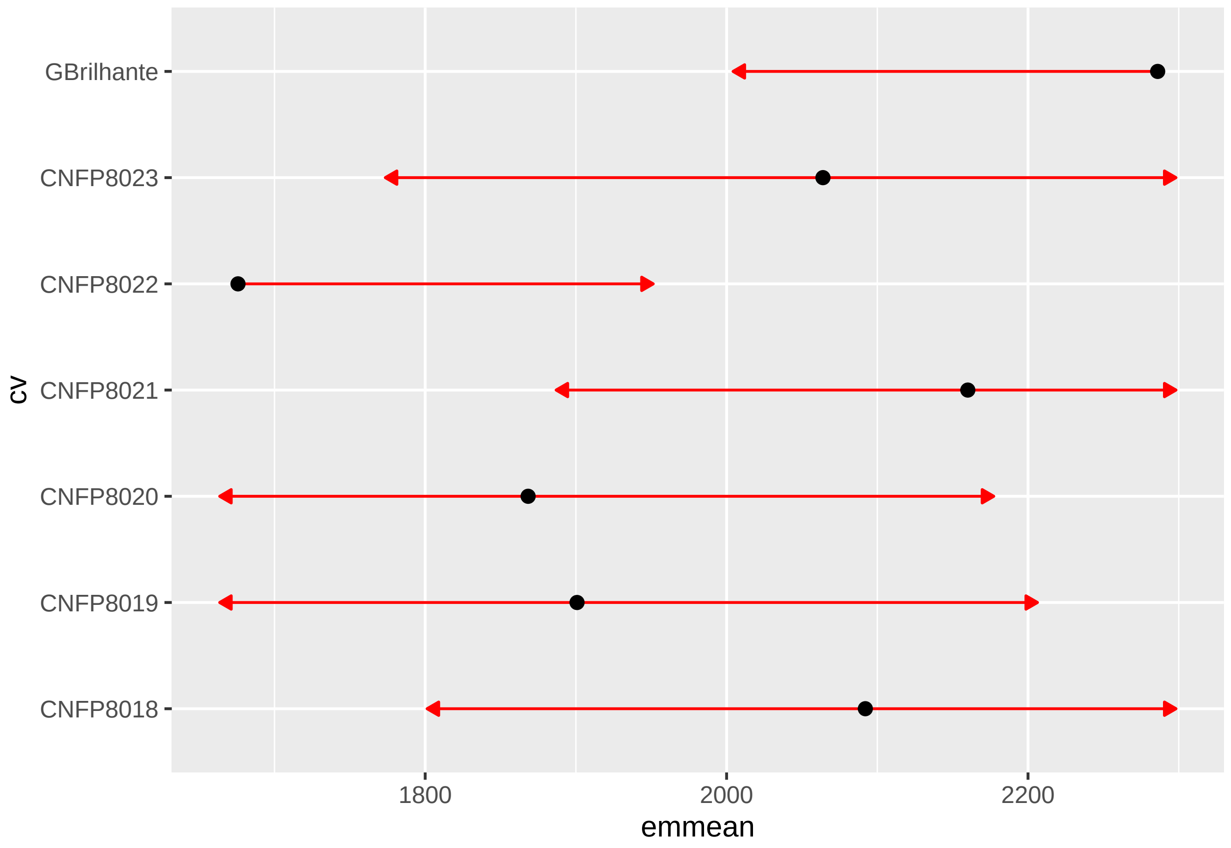




We have homogeneous variance across the fitted values and the residuals seems to be normally distributed.

Which cultivar was the best?

```r
plot(
  emmeans::emmeans(ancova, pairwise ~ cv, adjust = 'tukey'),
  interval = F, comparisons = T
)
```



We can see that cultivar `GBrilhante` had better performance than `CNFP8022`, but `GBrilhante` is not different from others.

3. Design a proper experiment to identify the best dose of Nitrogen and amount of water to maximize yield (choose what values you would use).

Nitrogen:

- Levels: 100 and 200
- Center: $\frac{100+200}{2} = 150$
- Range $= 200 - 100 = 100$

Water:

- Levels: 10 and 30
- Center: $\frac{10+30}{2} = 20$
- Range: $30 - 10 = 20$

```
nitro_levels <- c(100, 200)
nitro_center <- mean(nitro_levels)
```

```r
nitro_range <- max(nitro_levels) - min(nitro_levels)
nitro_center; nitro_range
```

```
## [1] 150
```

```
## [1] 100
```

```r
water_levels <- c(10, 30)
water_center <- mean(water_levels)
water_range <- max(water_levels) - min(water_levels)
water_center; water_range
```

```
## [1] 20
```

```
## [1] 20
```

```r
set.seed(2023)
q3 <- ccd(
    basis = 2,  # 2 factors (nitrogen and water)
    n0 = c(2, 2),  # 2 central point reps for each block
    alpha = 'rotatable',
    coding = list(
      x1 ~ (nitrogen - nitro_center) / (0.5 * nitro_range),
      x2 ~ (water - water_center) / (0.5 * water_range)
    )
)
q3
```

```
##    run.order std.order  nitrogen     water Block
## 1          1         2 200.00000 10.000000     1
## 2          2         5 150.00000 20.000000     1
## 3          3         4 200.00000 30.000000     1
## 4          4         3 100.00000 30.000000     1
## 5          5         1 100.00000 10.000000     1
## 6          6         6 150.00000 20.000000     1
## 7          1         6 150.00000 20.000000     2
## 8          2         3 150.00000  5.857864     2
## 9          3         4 150.00000 34.142136     2
## 10         4         5 150.00000 20.000000     2
## 11         5         2 220.71068 20.000000     2
## 12         6         1  79.28932 20.000000     2
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ (nitrogen - nitro_center)/(0.5 * nitro_range)
## x2 ~ (water - water_center)/(0.5 * water_range)
```

Convert to data.frame and generate yield:

```r
set.seed(2023)
q3y <- as.data.frame(q3)
q3y$yield <- rnorm(nrow(q3y), 10, 1)
q3y[(q3y$x1 == 0) & (q3y$x2 <= 1.3), 'yield'] <- (
  q3y[(q3y$x1 == 0) & (q3y$x2 <= 1.3), 'yield'] + rnorm(5, 4, 1)
)
q3y
```

```
##    run.order std.order        x1        x2 Block     yield
## 1          1         2  1.000000 -1.000000     1  9.916216
## 2          2         5  0.000000  0.000000     1 13.579093
## 3          3         4  1.000000  1.000000     1  8.124933
## 4          4         3 -1.000000  1.000000     1  9.813855
## 5          5         1 -1.000000 -1.000000     1  9.366514
## 6          6         6  0.000000  0.000000     1 15.754156
## 7          1         6  0.000000  0.000000     2 12.483375
## 8          2         3  0.000000 -1.414214     2 15.700017
## 9          3         4  0.000000  1.414214     2  9.600733
## 10         4         5  0.000000  0.000000     2 14.127723
## 11         5         2  1.414214  0.000000     2 10.326962
## 12         6         1 -1.414214  0.000000     2  9.587253
```

Now we can fit Response Surface Methodology models:

First, a first-order model:

```r
mod1 <- rsm(yield ~ FO(x1, x2), data = q3y)
summary(mod1)
```

```
##
## Call:
## rsm(formula = yield ~ FO(x1, x2), data = q3y)
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 11.531736   0.778361 14.8154 1.256e-07 ***
## x1          -0.011639   0.953294 -0.0122    0.9905
## x2          -1.246204   0.953294 -1.3073    0.2235
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.1596, Adjusted R-squared:  -0.02717
## F-statistic: 0.8545 on 2 and 9 DF,  p-value: 0.4573
##
## Analysis of Variance Table
##
## Response: yield
##             Df Sum Sq Mean Sq F value Pr(>F)
## FO(x1, x2)   2 12.425  6.2126  0.8545 0.4573
## Residuals    9 65.431  7.2701
## Lack of fit  6 59.861  9.9769  5.3736 0.0979
## Pure error   3  5.570  1.8566
##
## Direction of steepest ascent (at radius 1):
```

```
##          x1          x2
## -0.00933944 -0.99995639
##
## Corresponding increment in original units:
##          x1          x2
## -0.00933944 -0.99995639
```

Despite we don't have a significant lack of fit (at significance level of $\alpha = 0.05$), we have a bad adj. R^2. We can try a more complex model such as second-order model:

```
mod2 <- rsm(yield ~ SO(x1, x2), data = q3y)
summary(mod2)
```

```
##
## Call:
## rsm(formula = yield ~ SO(x1, x2), data = q3y)
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 13.986087   0.930541 15.0301 5.466e-06 ***
## x1          -0.011639   0.657992 -0.0177   0.98646
## x2          -1.246204   0.657992 -1.8940   0.10706
## x1:x2       -0.559656   0.930541 -0.6014   0.56956
## x1^2        -2.514080   0.735657 -3.4175   0.01419 *
## x2^2        -1.167446   0.735657 -1.5869   0.16362
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.7331, Adjusted R-squared:  0.5106
## F-statistic: 3.296 on 5 and 6 DF,  p-value: 0.08941
##
## Analysis of Variance Table
##
## Response: yield
##             Df Sum Sq Mean Sq F value  Pr(>F)
## FO(x1, x2)   2 12.425  6.2126  1.7937 0.24511
## TWI(x1, x2)  1  1.253  1.2529  0.3617 0.56956
## PQ(x1, x2)   2 43.397 21.6984  6.2646 0.03395
## Residuals    6 20.782  3.4636
## Lack of fit  3 15.212  5.0706  2.7311 0.21560
## Pure error   3  5.570  1.8566
##
## Stationary point of response surface:
##         x1         x2
##  0.0586566 -0.5477903
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1] -1.111613 -2.569913
##
## $vectors
##          [,1]       [,2]
## x1  0.1956688 -0.9806700
## x2 -0.9806700 -0.1956688
```

This model is better, because the adj. $R^2 \approx 51$ and the lack of fit is not significant.
The negatives eigenvalues indicate a maximum point.

The optimal experimental points are:

```
opt_exp_points <- summary(mod2)$canonical$xs
opt_exp_points
```

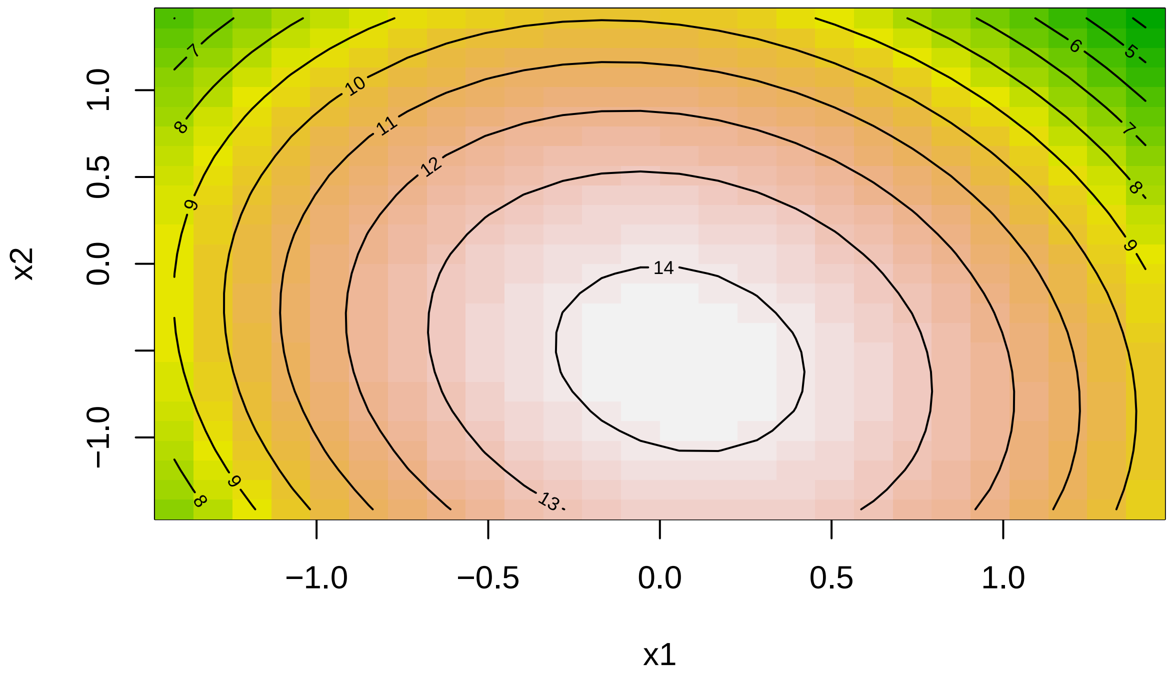
```
##          x1         x2
##   0.0586566 -0.5477903
```

In the original scale we have to back transform the coded variables:

```
opt_orig_points <- c()
opt_orig_points[1] <- ((0.5 * nitro_range) * opt_exp_points[1]) + nitro_center
opt_orig_points[2] <- ((0.5 * water_range) * opt_exp_points[2]) + water_center
names(opt_orig_points) <- c('nitrogen', 'water')
opt_orig_points
```

```
## nitrogen    water
## 152.9328  14.5221
```

```
contour(
  mod2,
  ~ x1 + x2,
  image = TRUE
)
```



The maximum seems to be close to the range $x1 = 0$ and $x2 < 0$.
We can check the distribution of the fitted values:

```
summary(mod2$fitted.values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.487   9.466  11.551  11.532  13.986  13.986
```

The maximum yield found was 13.978.

Maybe we can find a highest point using steepest-ascent method:

```
steep <- steepest(mod2)
```

```
## Path of steepest ascent from ridge analysis:
```

```
steep
```

```
##    dist    x1     x2 |   yhat
## 1   0.0 0.000  0.000 | 13.986
## 2   0.5 0.051 -0.497 | 14.324
## 3   1.0 0.135 -0.991 | 14.102
## 4   1.5 0.226 -1.483 | 13.323
## 5   2.0 0.320 -1.974 | 11.989
## 6   2.5 0.415 -2.465 | 10.099
## 7   3.0 0.511 -2.956 |  7.652
## 8   3.5 0.608 -3.447 |  4.647
## 9   4.0 0.705 -3.937 |  1.093
## 10  4.5 0.802 -4.428 | -3.025
## 11  5.0 0.899 -4.919 | -7.699
```

Maximum point:

```
df_opt_points <- steep[which.max(steep$yhat), ]
df_opt_points
```

```
##   dist    x1     x2 |   yhat
## 2  0.5 0.051 -0.497 | 14.324
```

The maximum yield was found using $x = 0.051$ and $x2 = -0.497$.
If we do a predict using these points we can reach to the same maximum yield:

```
predict(mod2, df_opt_points[, c('x1', 'x2')])
```

```
##        2
## 14.32413
```

We can back transform and check what are the best treatment levels to reach the optimum yield:

```r
opt_points <- c()
opt_points[1] <- ((0.5 * nitro_range) * df_opt_points$x1) + nitro_center
opt_points[2] <- ((0.5 * water_range) * df_opt_points$x2) + water_center
names(opt_points) <- c('nitrogen', 'water')
opt_points
```

```
## nitrogen    water
##   152.55    15.03
```