Design of Experiments Final Project

Elmer Camargo

2023-11-15

Abstract

Provided a goal to maximize the response of a simulated surface, I followed a strategy to start small and advance the region of interest to find the maximized response of Y being approximate to 630.98 when A = 3927.15, B = 400.8652, C = 273.6, D = 70.9 , E = 1319.71, and F = 56.807. The methodologies used consisted of fractional factorial design, aliasing, coding and uncoding functions, exploratory data analysis, first order and second order modeling, curvature testing, backwards selection, residual analysis, eigenvalue evaluation, ridge analysis, steepest ascent, and iterative line search. The workflow of the project follows a setup for the experiment, modeling, and evaluation.

Experiment Type	Trials Executed
Experiment 1	19
Line Search 1	7
Experiment 2	11
Line Search 2	3
Experiment 3	11
Line Search 3	3
Total	54

Background

The final project for ISE 525 Design of Experiments is to use a simulated lab software to study a response surface by running experiments in order to achieve a provided goal. The goal for my experiment is below.

Goal

```r
print(readLines("experiment_0.txt"))
```

```
##  [1] "Student ID# 9275"
##  [2] ""
##  [3] " Your objective is to MAXIMIZE the response."
##  [4] ""
##  [5] " Region of Operability    Current Operating Point"
##  [6] "   3500 < A < 5000              A = 4341.0"
##  [7] "    230 < B <  450              B =  255.3"
##  [8] "    170 < C <  390              C =  273.6"
##  [9] "     50 < D <  100              D =   70.9"
## [10] "   1000 < E < 1700              E = 1540.0"
## [11] "     30 < F <   90              F =   41.3"
```

Current Operating Positions (COP)

Given the constraints to where I can operate within my experiment using 6 variables, I'd like to have a lay of the land and know where the current operating positions are the variable boundaries.

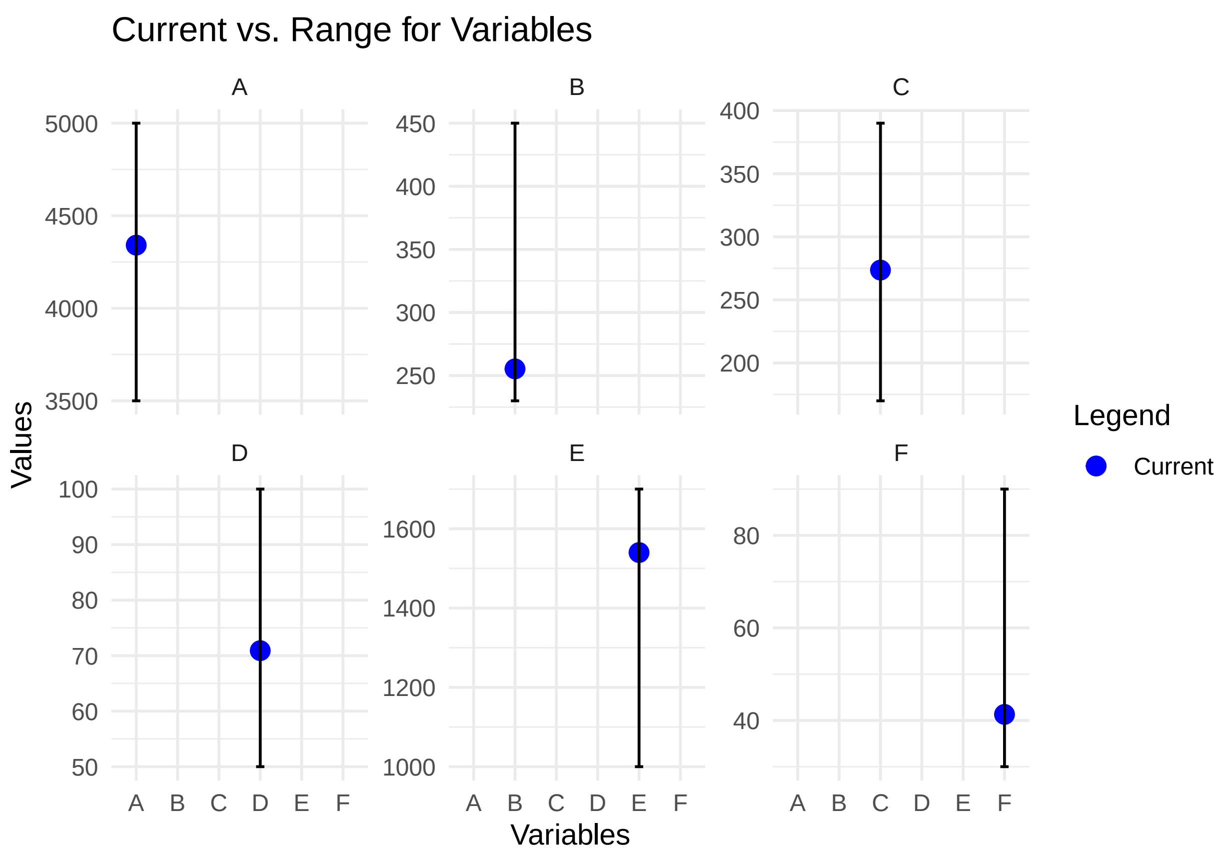
```r
cop <- data.frame(variables=c("A", "B", "C", "D", "E", "F"),
                  current=c(4341.0, 255.3, 273.6, 70.9, 1540.0, 41.3),
                  lower=c(3500,230,170,50,1000,30),
                  upper=c(5000,450,390,100,1700,90))
cop
```

```
##   variables current lower upper
## 1         A  4341.0  3500  5000
## 2         B   255.3   230   450
## 3         C   273.6   170   390
## 4         D    70.9    50   100
## 5         E  1540.0  1000  1700
## 6         F    41.3    30    90
```

Below is a visual of the range I have for the settings I am to explore during the project

```r
ggplot(cop, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 3) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
```

```r
  labs(title = "Current vs. Range for Variables",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = "blue") +
  facet_wrap(~variables, scales = "free_y")
```



Experiment 1

Setup

High and Low Assignment

To conduct any kind of experiment I need to be able to assign values between 1 (high) and -1 (low). I decided to use an increase and decrease of 15% of the range as my scaling factor. I also apply a min and a max function to ensure my output for highs and lows are within the upper and lower bounds possible for each settings. The position of settings I am planning to use for the first experiment are below.

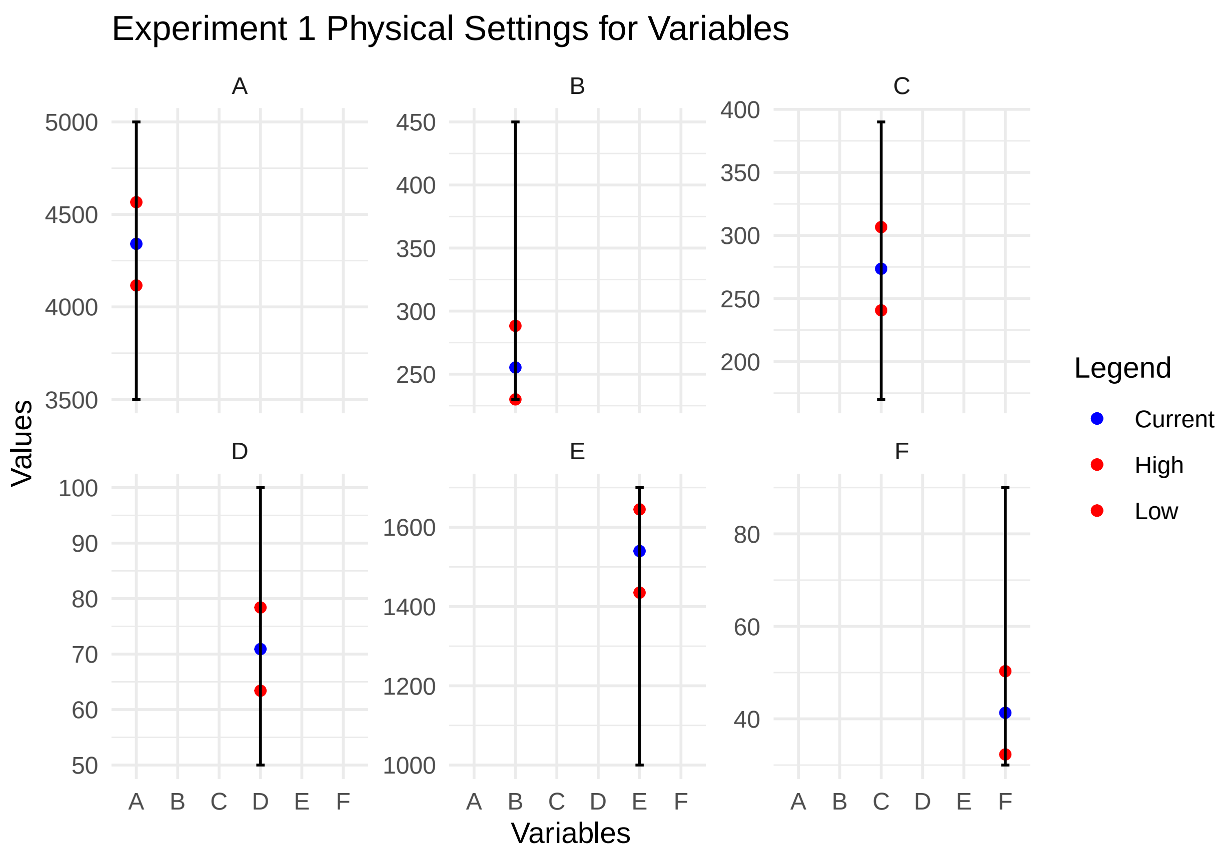
```r
cop$range <- cop$upper - cop$lower
cop$change <- cop$range * .15
cop$increase <- cop$current + cop$change
cop$decrease <- cop$current - cop$change
```

```r
# applying a min and max to avoid going above or below a limit
cop$high <- apply(cop[, c("upper","increase")], 1, min)
cop$low <- apply(cop[, c("lower","decrease")], 1, max)

cop[,c("variables", "high", "low")]
```

```
##   variables   high    low
## 1         A 4566.0 4116.0
## 2         B  288.3  230.0
## 3         C  306.6  240.6
## 4         D   78.4   63.4
## 5         E 1645.0 1435.0
## 6         F   50.3   32.3
```

Below is a visualization for where I am setting the high and low settings for my experiment

```r
ggplot(cop, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  labs(title = "Experiment 1 Physical Settings for Variables",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("blue", "red", "red")) +
  facet_wrap(~variables, scales = "free_y")
```



Fractional Design

I am using a 2^{6-2} fractional factorial design with 3 center points to begin my analysis. The generators used for this design are E = ABC and F = BCD. An alias structure is included in the appendix along with the words provided by the decision of the generators. My decision to go with a 2^{6-2} is to see if interactions occur in the process and to more easily be able to identify the variables responsible.

```
quarter_design <- FrF2(nfactors = 6,
                       nruns = 2^(6-2),
                       generators=c("ABC","BCD"),
                       ncenter = 3,
                       randomize=FALSE)
quarter_design
```

```
##      A  B  C  D  E  F
## 1   -1 -1 -1 -1 -1 -1
## 2    1 -1 -1 -1  1 -1
## 3   -1  1 -1 -1  1  1
## 4    1  1 -1 -1 -1  1
## 5   -1 -1  1 -1  1  1
## 6    1 -1  1 -1 -1  1
## 7   -1  1  1 -1 -1 -1
## 8    1  1  1 -1  1 -1
## 9   -1 -1 -1  1 -1  1
## 10   1 -1 -1  1  1  1
```

```
## 11 -1  1 -1  1  1 -1
## 12  1  1 -1  1 -1 -1
## 13 -1 -1  1  1  1 -1
## 14  1 -1  1  1 -1 -1
## 15 -1  1  1  1 -1  1
## 16  1  1  1  1  1  1
## 17  0  0  0  0  0  0
## 18  0  0  0  0  0  0
## 19  0  0  0  0  0  0
## class=design, type= FrF2.generators.center
```

```r
aliasprint(quarter_design)
```

```
## $legend
## [1] A=A B=B C=C D=D E=E F=F
##
## $main
## character(0)
##
## $fi2
## [1] AB=CE    AC=BE    AD=EF    AE=BC=DF AF=DE    BD=CF    BF=CD
```

Design Encoding

To be able to run the experiment I convert the coded numbers (-1/+1) into the natural numbers that the simulation program expects. To do so I create an coding function to convert the coded design into one with physical settings.

```r
calculate_midpoint <- function(vector){
  low <- min(vector)
  hi <- max(vector)
  return ((hi+low)/2)
}

calculate_midrange <- function(vector){
  low <- min(vector)
  hi <- max(vector)
  return ((hi-low)/2)
}

uncode_variable <- function(design, t_cop, variable_name) {
  # Calculate the midrange using the
  #"high" and "low" columns from transposed verion of cop
  midrange <- calculate_midrange(t_cop[c("high", "low"), variable_name])
  midpoint <- calculate_midpoint(t_cop[c("high","low"), variable_name])
  # Create a new uncoded column in the design dataframe
  design[[paste0(variable_name, "_uncoded")]] <- design[[variable_name]] * midrange +
    midpoint

  return(design)
}
```

```r
# transposing cop for easier data manipulation
t_cop <- data.frame(t(cop[,-1]))
colnames(t_cop) <- cop$variables

# I write the design to csv because R uses factor
# values of 1 and 2 underneath the hood.
# I write to csv and read back to interpret the variables as -1 and 1.
write.csv(as.data.frame(quarter_design), "design.csv", row.names = F)
deisgn <- read.csv("design.csv")

for(i in colnames(deisgn)){
  deisgn <- uncode_variable(deisgn, t_cop, i)
}

deisgn_coded <- deisgn[,1:6]
deisgn_uncoded <- deisgn[,7:12]
head(deisgn_uncoded)
```

```
##   A_uncoded B_uncoded C_uncoded D_uncoded E_uncoded F_uncoded
## 1      4116     230.0     240.6      63.4      1435      32.3
## 2      4566     230.0     240.6      63.4      1645      32.3
## 3      4116     288.3     240.6      63.4      1645      50.3
## 4      4566     288.3     240.6      63.4      1435      50.3
## 5      4116     230.0     306.6      63.4      1645      50.3
## 6      4566     230.0     306.6      63.4      1435      50.3
```

```r
write.csv(deisgn_uncoded, "exp_1.csv", row.names = F)
write.csv(deisgn_coded, "exp_1_coded.csv", row.names = F)
```

Results

Exploratory Data Analysis

I ran the experiment with design and received this output which I then manually transferred onto a written csv of the coded design. For all simulation outputs in order of execution, see appendix.

```r
print(readLines("exp_results_1.txt"))
```

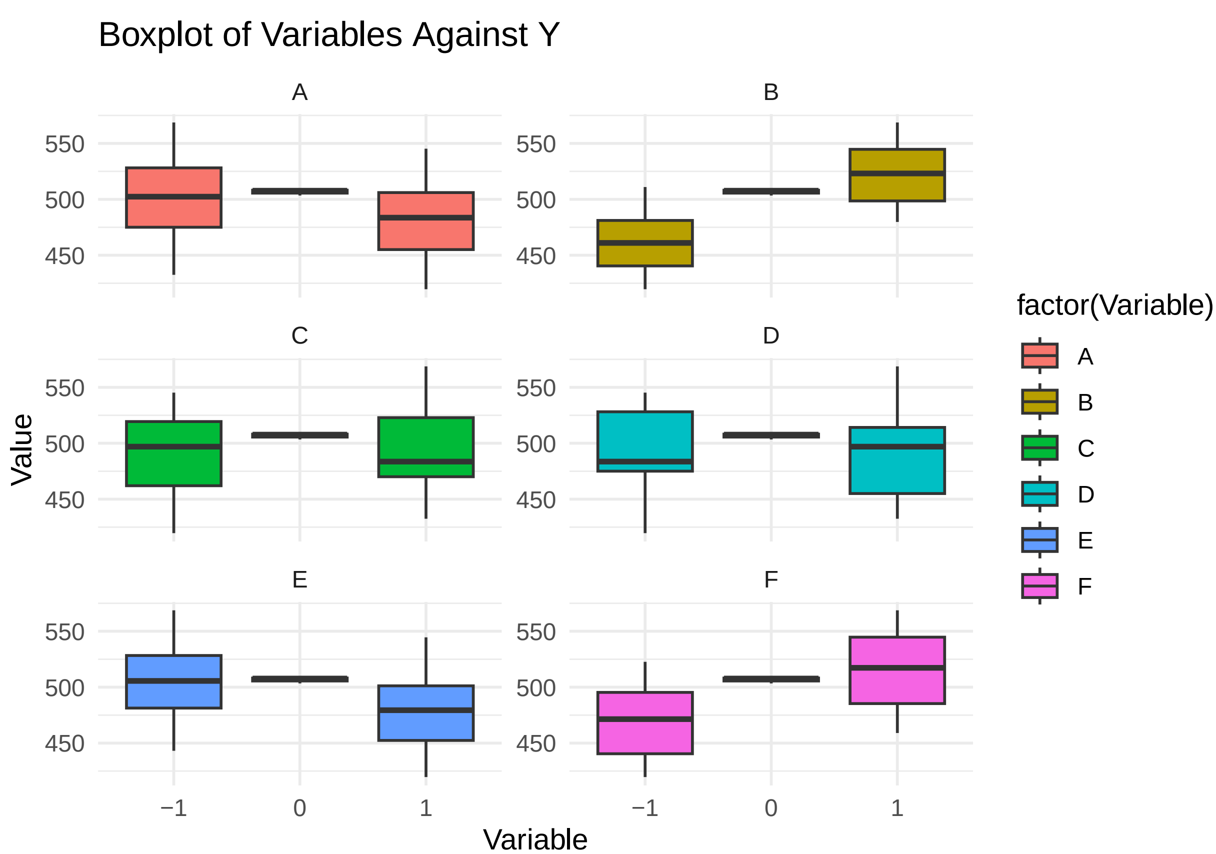
```
##  [1] "Student ID# 9275"
##  [2] ""
##  [3] " RUN     A       B       C       D       E       F       Y"
##  [4] "   1 4116.00  230.00  240.60   63.40 1435.00   32.30  463.04"
##  [5] "   2 4566.00  230.00  240.60   63.40 1645.00   32.30  419.63"
##  [6] "   3 4116.00  288.30  240.60   63.40 1645.00   50.30  544.54"
##  [7] "   4 4566.00  288.30  240.60   63.40 1435.00   50.30  545.25"
##  [8] "   5 4116.00  230.00  306.60   63.40 1645.00   50.30  479.03"
##  [9] "   6 4566.00  230.00  306.60   63.40 1435.00   50.30  487.41"
## [10] "   7 4116.00  288.30  306.60   63.40 1435.00   32.30  522.66"
## [11] "   8 4566.00  288.30  306.60   63.40 1645.00   32.30  479.79"
## [12] "   9 4116.00  230.00  240.60   78.40 1435.00   50.30  511.00"
## [13] "  10 4566.00  230.00  240.60   78.40 1645.00   50.30  459.02"
```

```
## [14] "  11 4116.00  288.30  240.60   78.40 1645.00   32.30  493.73"
## [15] "  12 4566.00  288.30  240.60   78.40 1435.00   32.30  500.17"
## [16] "  13 4116.00  230.00  306.60   78.40 1645.00   32.30  432.50"
## [17] "  14 4566.00  230.00  306.60   78.40 1435.00   32.30  443.13"
## [18] "  15 4116.00  288.30  306.60   78.40 1435.00   50.30  568.72"
## [19] "  16 4566.00  288.30  306.60   78.40 1645.00   50.30  523.59"
## [20] "  17 4341.00  259.15  273.60   70.90 1540.00   41.30  508.60"
## [21] "  18 4341.00  259.15  273.60   70.90 1540.00   41.30  507.71"
## [22] "  19 4341.00  259.15  273.60   70.90 1540.00   41.30  503.38"
```

```
results_coded <- read.csv("exp_1_results_coded.csv")
```

I make a boxplot of the variables to get a general sense of what the results are. From visual analysis it appears B and F in the high provides a large response. E and A in the low seem to perform better than in the high.

```
df_long <- gather(results_coded, key = "Variable", value = "Value", -Y)

ggplot(df_long, aes(x = factor(Value), y = Y)) +
  geom_boxplot(aes(fill = factor(Variable)), position = "dodge") +
  labs(title = "Boxplot of Variables Against Y",
       x = "Variable",
       y = "Value") +
  facet_wrap(~ Variable, scales = "free_y", ncol = 2) +
  theme_minimal()
```



Modeling

I use a backwards selection modeling strategy to find a suitable first order model.

```r
mod1 <- lm(Y~A+B+C+D+E+F, data=results_coded)
summary(mod1)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + C + D + E + F, data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9232 -4.2700 -0.9769  0.4218 14.2368
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 494.36316    1.66595 296.745  < 2e-16 ***
## A            -9.82687    1.81543  -5.413 0.000157 ***
## B            30.23063    1.81543  16.652 1.17e-09 ***
## C             0.02813    1.81543   0.015 0.987894
## D            -0.59312    1.81543  -0.327 0.749513
## E           -13.09687    1.81543  -7.214 1.07e-05 ***
## F            22.74438    1.81543  12.528 2.99e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.262 on 12 degrees of freedom
## Multiple R-squared:  0.9773, Adjusted R-squared:  0.9659
## F-statistic: 85.95 on 6 and 12 DF,  p-value: 3.722e-09
```

Remove C

```r
mod2 <- lm(Y~A+B+D+E+F, data=results_coded)
summary(mod2)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + D + E + F, data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9513 -4.2982 -0.9488  0.4218 14.2368
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 494.3632     1.6006 308.859  < 2e-16 ***
## A            -9.8269     1.7442  -5.634 8.15e-05 ***
## B            30.2306     1.7442  17.332 2.29e-10 ***
## D            -0.5931     1.7442  -0.340    0.739
## E           -13.0969     1.7442  -7.509 4.44e-06 ***
## F            22.7444     1.7442  13.040 7.67e-09 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.977 on 13 degrees of freedom
## Multiple R-squared:  0.9773, Adjusted R-squared:  0.9685
## F-statistic: 111.7 on 5 and 13 DF,  p-value: 3.281e-10
```

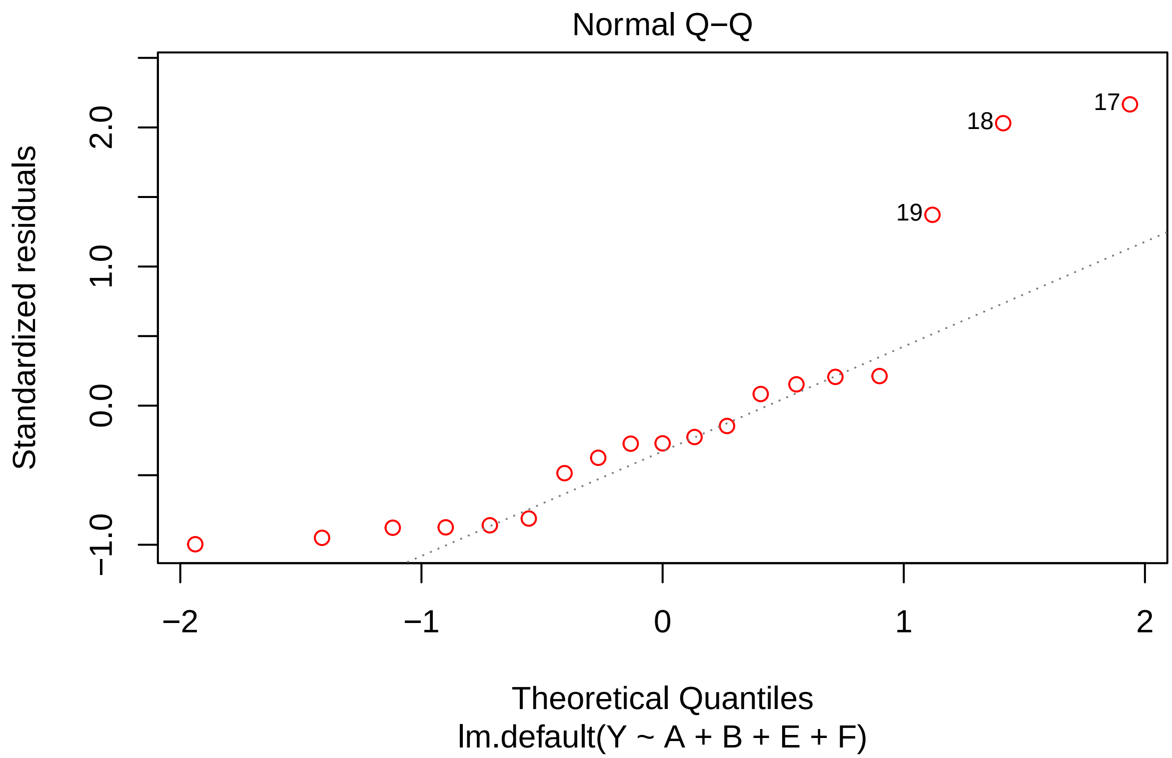
Remove D

```r
mod3 <- lm(Y~A+B+E+F, data=results_coded)
summary(mod3)
```

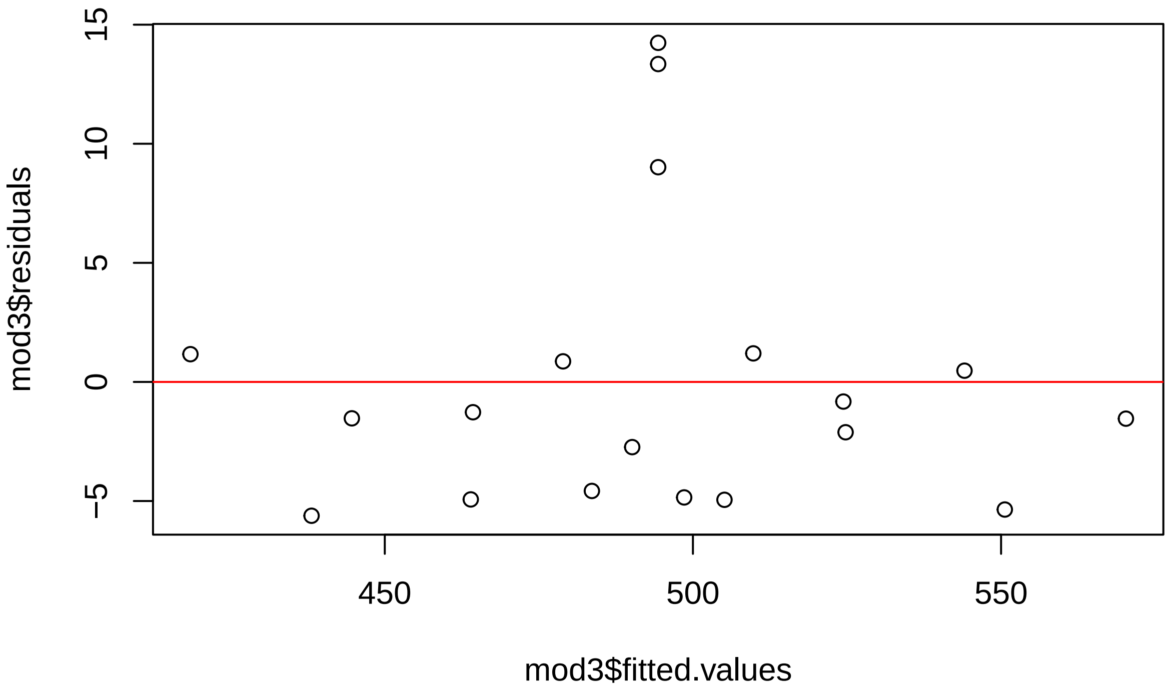
```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F, data = results_coded)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.618 -4.713 -1.528  1.015 14.237
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  494.363      1.549 319.102  < 2e-16 ***
## A             -9.827      1.688  -5.821 4.44e-05 ***
## B             30.231      1.688  17.907 4.79e-11 ***
## E            -13.097      1.688  -7.758 1.96e-06 ***
## F             22.744      1.688  13.472 2.09e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.753 on 14 degrees of freedom
## Multiple R-squared:  0.9771, Adjusted R-squared:  0.9705
## F-statistic: 149.1 on 4 and 14 DF,  p-value: 2.623e-11
```

Linear modeling using variables A, B, E, and F provides all significant variables, however through residual analysis it appears there is a violation of regression assumptions - Normality of residuals.

```r
plot(mod3, which=2, col=c("red"))
```




```
plot(mod3$fitted.values,mod3$residuals)
abline(0,0, col="red")
```



The violation of regression assumptions can potentially be fixed by a second order model. I then check the results of the center points to test for curvature.

```
data <- results_coded[results_coded$A != 0,]
center <- results_coded[results_coded$A == 0,]

yf <- mean(data$Y)
yc <- mean(center$Y)
curve_sse <- sum(center$Y^2) - (sum(center$Y)^2/nrow(center))
dfe <- (nrow(center)-1)
curve_mse <- curve_sse/dfe

sscurve <- (((yf-yc)^2)*nrow(data)*nrow(center))/(nrow(data)+nrow(center))
dfc <- 1

mscurve <- sscurve/dfc
fcurve <- mscurve/curve_mse
f_crit <- qf(p=.05, df1=dfc, df2=dfe, lower.tail=FALSE)
fcurve
```

```
## [1] 67.99716
```

```
f_crit
```

```
## [1] 18.51282
```

```r
dfc
```

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

```r
dfe
```

```
## [1] 2
```

```r
fcurve > f_crit
```

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

From calculating the F statistic using the center points the null hypothesis of no curvature is rejected and a second order model is recommended.

I first check to see if including interactions help in providing a suitable modeling.

```r
# from alias struct: AB=CE    AC=BE    AE=BC=DF
mod4 <- lm(Y~A+
             B+
             E+
             F+
             A*B+
             A*E+
             A*F+
             B*E+
             B*F+
             E*F, data=results_coded)
summary(mod4)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F + A * B + A * E + A *
##     F + B * E + B * F + E * F, data = results_coded)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.802 -2.653 -2.224 -1.111 14.237
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 494.3632     1.9167 257.926  < 2e-16 ***
## A            -9.8269     2.0887  -4.705  0.00153 **
## B            30.2306     2.0887  14.474 5.08e-07 ***
## E           -13.0969     2.0887  -6.270  0.00024 ***
## F            22.7444     2.0887  10.889 4.48e-06 ***
## A:B          -0.2794     2.0887  -0.134  0.89690
## A:E           1.3556     2.0887   0.649  0.53450
## A:F          -1.1756     2.0887  -0.563  0.58895
## B:E           1.2031     2.0887   0.576  0.58044
## B:F           0.4744     2.0887   0.227  0.82603
## E:F          -0.1781     2.0887  -0.085  0.93413
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.355 on 8 degrees of freedom
## Multiple R-squared:  0.9799, Adjusted R-squared:  0.9548
## F-statistic: 39.07 on 10 and 8 DF,  p-value: 1.064e-05
```

Interactions dont seem to help so I include a squared variable to continue testing

```r
mod5 <- lm(Y~A+
             B+
             E+
             F+
             A*B+
             A*E+
             A*F+
             B*E+
             B*F+
             E*F+
             I(A^2), data=results_coded)
summary(mod5)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F + A * B + A * E + A *
##     F + B * E + B * F + E * F + I(A^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1833 -0.6119  0.0306  1.0211  2.0367
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 506.5633     1.1576 437.592  < 2e-16 ***
## A            -9.8269     0.5013 -19.604 2.24e-07 ***
## B            30.2306     0.5013  60.309 9.05e-11 ***
## E           -13.0969     0.5013 -26.128 3.08e-08 ***
## F            22.7444     0.5013  45.374 6.60e-10 ***
## I(A^2)      -14.4877     1.2615 -11.485 8.53e-06 ***
## A:B          -0.2794     0.5013  -0.557   0.5947
## A:E           1.3556     0.5013   2.704   0.0304 *
## A:F          -1.1756     0.5013  -2.345   0.0514 .
## B:E           1.2031     0.5013   2.400   0.0475 *
## B:F           0.4744     0.5013   0.946   0.3755
## E:F          -0.1781     0.5013  -0.355   0.7328
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.005 on 7 degrees of freedom
## Multiple R-squared:  0.999,  Adjusted R-squared:  0.9974
## F-statistic: 628.6 on 11 and 7 DF,  p-value: 2.17e-09
```

Including the squared term turns out to be significant and impacts the interaction terms AE (aliased with BC and DF), AF (aliased with DE), and BE (aliased CE). Given the aliasing, it is unlikely that the non statistically significant variables are responsible for the significance, and more likely that the significant variables are responsible for significance. I run an additional model to double check if including the variables C and D change anything.

```
mod6 <- lm(Y~A+
             B+
             C+
             D+
             E+
             F+
             A*B+
             A*E+
             A*F+
             B*E+
             B*F+
             E*F+
             I(A^2), data=results_coded)
summary(mod6)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + C + D + E + F + A * B + A *
##     E + A * F + B * E + B * F + E * F + I(A^2), data = results_coded)
##
## Residuals:
##       1       2       3       4       5       6       7       8       9      10
## -0.9494  0.6281  0.5956 -0.2744 -0.5956  0.2744  0.9494 -0.6281  0.9494 -0.6281
##      11      12      13      14      15      16      17      18      19
## -0.5956  0.2744  0.5956 -0.2744 -0.9494  0.6281  2.0367  1.1467 -3.1833
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 506.56333    1.22475 413.606 1.57e-12 ***
## A            -9.82687    0.53033 -18.530 8.42e-06 ***
## B            30.23063    0.53033  57.003 3.14e-08 ***
## C             0.02813    0.53033   0.053 0.959759
## D            -0.59312    0.53033  -1.118 0.314228
## E           -13.09687    0.53033 -24.696 2.03e-06 ***
## F            22.74438    0.53033  42.887 1.30e-07 ***
## I(A^2)      -14.48771    1.33464 -10.855 0.000115 ***
## A:B          -0.27938    0.53033  -0.527 0.620863
## A:E           1.35562    0.53033   2.556 0.050882 .
## A:F          -1.17563    0.53033  -2.217 0.077442 .
## B:E           1.20312    0.53033   2.269 0.072563 .
## B:F           0.47437    0.53033   0.894 0.412045
## E:F          -0.17813    0.53033  -0.336 0.750601
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.121 on 5 degrees of freedom
## Multiple R-squared:  0.9992, Adjusted R-squared:  0.9971
## F-statistic: 475.3 on 13 and 5 DF,  p-value: 7.81e-07
```

Including them did not change anything, so I assume that the interaction is unlikely due to C and D and more likely to the statistically significant variables and continue iterating through the squared terms. I do not have enough degrees of freedom to use all squared terms so for now I am selecting the one that provides a more significant impact on the response.

```
mod7 <- lm(Y~A+
             B+
             E+
             F+
             A*E+
             A*F+
             B*E+
             I(B^2), data=results_coded)
```

```
mod8 <- lm(Y~A+
             B+
             E+
             F+
             A*E+
             A*F+
             B*E+
             I(E^2), data=results_coded)
```

```
mod9 <- lm(Y~A+
             B+
             E+
             F+
             A*E+
             A*F+
             B*E+
             I(F^2), data=results_coded)
```

```
mod10 <- lm(Y~A+
             B+
             E+
             F+
             A*E+
             A*F+
             B*E+
             I(A^2), data=results_coded)
```

```
summary(mod10)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F + A * E + A * F + B *
##     E + I(A^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1833 -0.7444 -0.2475  1.0127  2.1250
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 506.5633     1.0567 479.382  < 2e-16 ***
## A            -9.8269     0.4576 -21.476 1.07e-09 ***
## B            30.2306     0.4576  66.069 1.54e-14 ***
## E           -13.0969     0.4576 -28.623 6.31e-11 ***
## F            22.7444     0.4576  49.707 2.62e-13 ***
## I(A^2)      -14.4877     1.1515 -12.581 1.87e-07 ***
## A:E           1.3556     0.4576   2.963   0.0142 *
## A:F          -1.1756     0.4576  -2.569   0.0279 *
## B:E           1.2031     0.4576   2.629   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.83 on 10 degrees of freedom
## Multiple R-squared:  0.9988, Adjusted R-squared:  0.9978
## F-statistic:  1037 on 8 and 10 DF,  p-value: 1.411e-13
```

```r
summary(mod7)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F + A * E + A * F + B *
##     E + I(B^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1833 -0.7444 -0.2475  1.0127  2.1250
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 506.5633     1.0567 479.382  < 2e-16 ***
## A            -9.8269     0.4576 -21.476 1.07e-09 ***
## B            30.2306     0.4576  66.069 1.54e-14 ***
## E           -13.0969     0.4576 -28.623 6.31e-11 ***
## F            22.7444     0.4576  49.707 2.62e-13 ***
## I(B^2)      -14.4877     1.1515 -12.581 1.87e-07 ***
## A:E           1.3556     0.4576   2.963   0.0142 *
## A:F          -1.1756     0.4576  -2.569   0.0279 *
## B:E           1.2031     0.4576   2.629   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.83 on 10 degrees of freedom
## Multiple R-squared:  0.9988, Adjusted R-squared:  0.9978
## F-statistic:  1037 on 8 and 10 DF,  p-value: 1.411e-13
```

```r
summary(mod8)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F + A * E + A * F + B *
##     E + I(E^2), data = results_coded)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1833 -0.7444 -0.2475  1.0127  2.1250
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 506.5633     1.0567 479.382  < 2e-16 ***
## A            -9.8269     0.4576 -21.476 1.07e-09 ***
## B            30.2306     0.4576  66.069 1.54e-14 ***
## E           -13.0969     0.4576 -28.623 6.31e-11 ***
## F            22.7444     0.4576  49.707 2.62e-13 ***
## I(E^2)      -14.4877     1.1515 -12.581 1.87e-07 ***
## A:E           1.3556     0.4576   2.963   0.0142 *
## A:F          -1.1756     0.4576  -2.569   0.0279 *
## B:E           1.2031     0.4576   2.629   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.83 on 10 degrees of freedom
## Multiple R-squared:  0.9988, Adjusted R-squared:  0.9978
## F-statistic:  1037 on 8 and 10 DF,  p-value: 1.411e-13
```

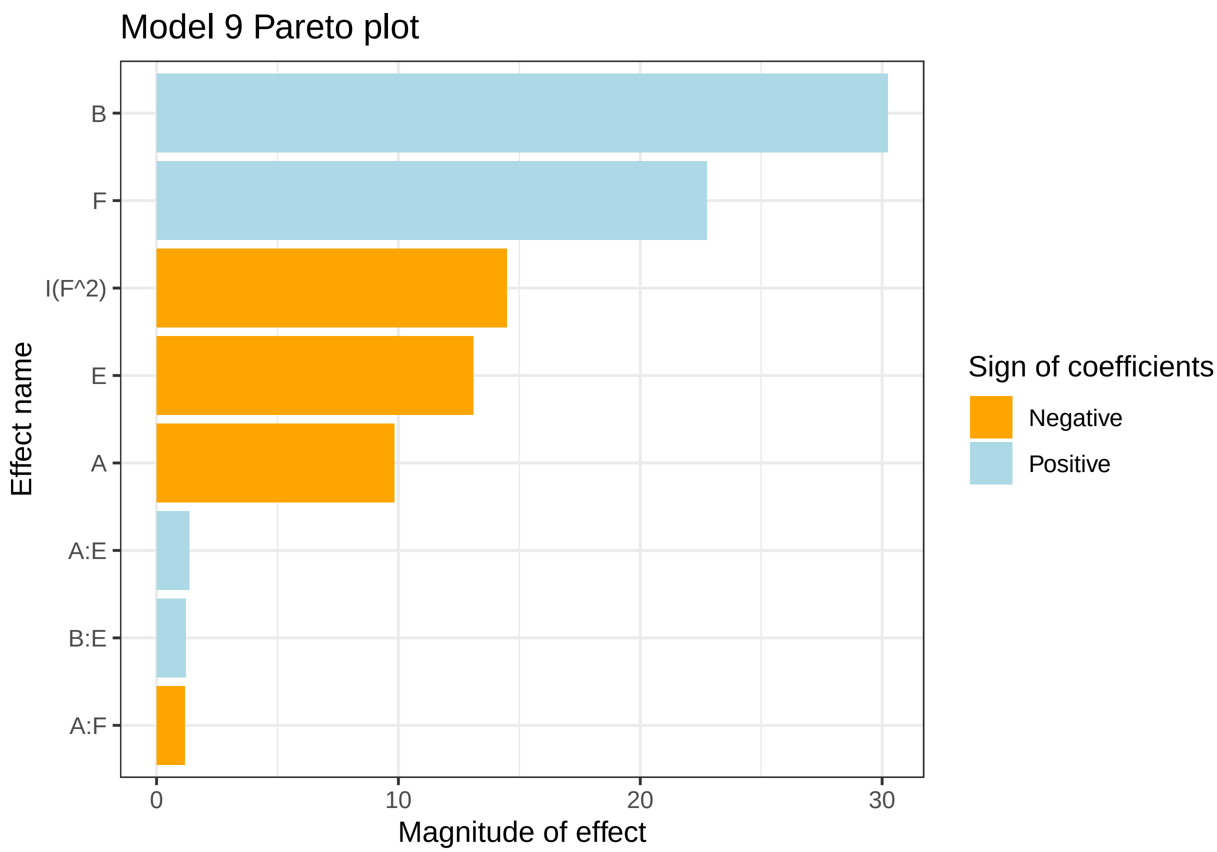
```r
summary(mod9)
```

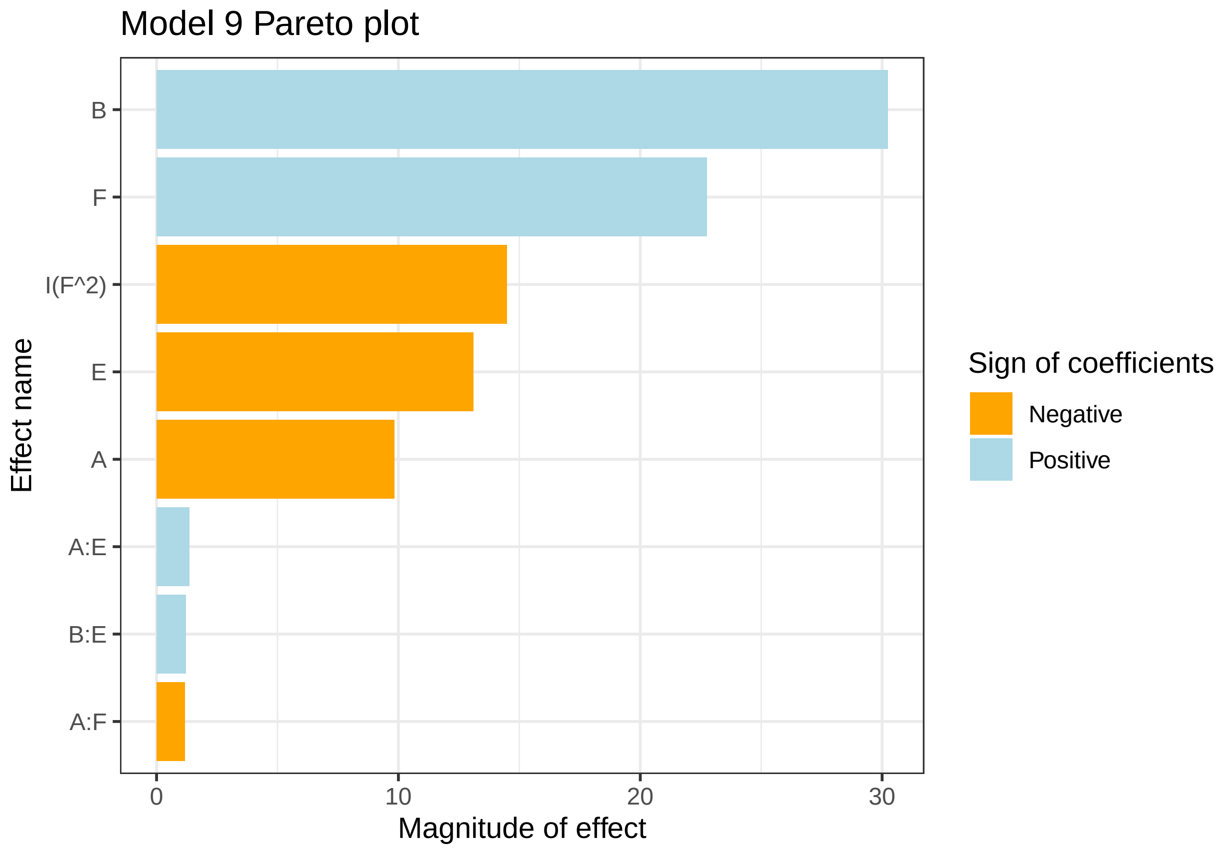
```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F + A * E + A * F + B *
##     E + I(F^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1833 -0.7444 -0.2475  1.0127  2.1250
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 506.5633     1.0567 479.382  < 2e-16 ***
## A            -9.8269     0.4576 -21.476 1.07e-09 ***
## B            30.2306     0.4576  66.069 1.54e-14 ***
## E           -13.0969     0.4576 -28.623 6.31e-11 ***
## F            22.7444     0.4576  49.707 2.62e-13 ***
## I(F^2)      -14.4877     1.1515 -12.581 1.87e-07 ***
## A:E           1.3556     0.4576   2.963   0.0142 *
## A:F          -1.1756     0.4576  -2.569   0.0279 *
## B:E           1.2031     0.4576   2.629   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.83 on 10 degrees of freedom
## Multiple R-squared:  0.9988, Adjusted R-squared:  0.9978
## F-statistic:  1037 on 8 and 10 DF,  p-value: 1.411e-13
```

All models perform similarly so I choose Model 9 as my final model. Below is the Pareto Plot and the residuals plots that have solved our issue of normality.

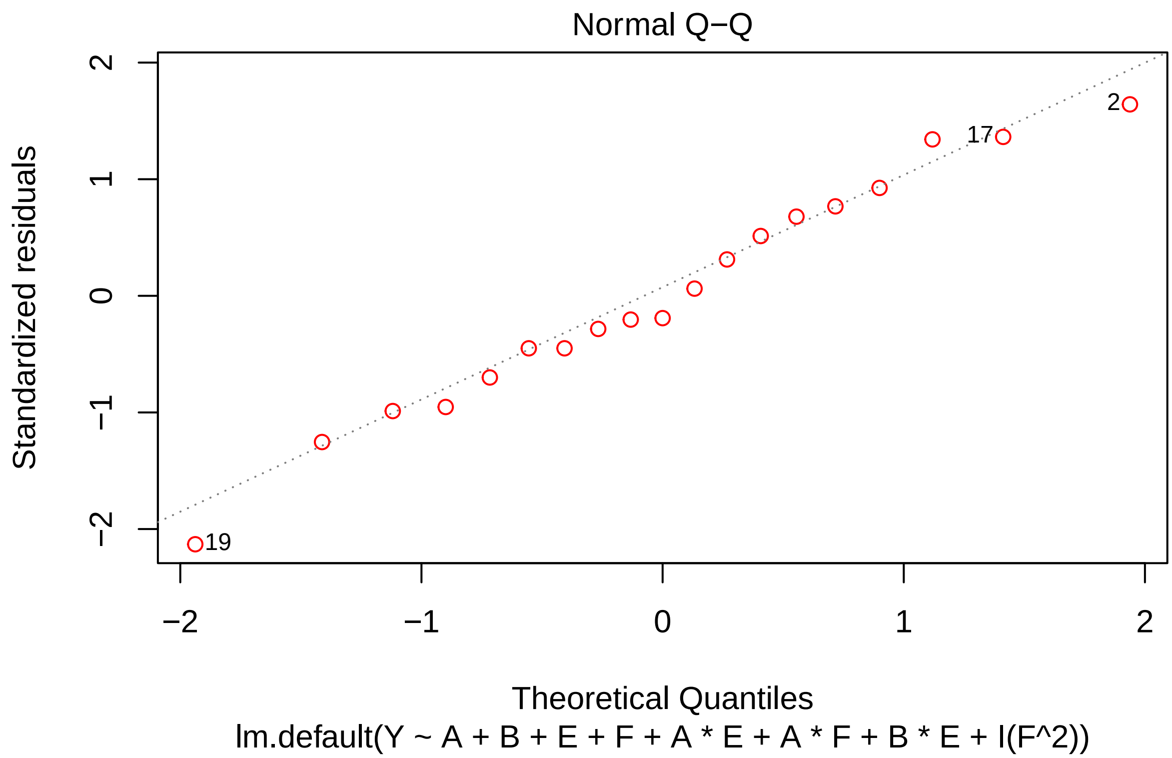
I am not sure why, but the pareto plot gets duplicated when the document is compiled. This happens throughout the pdf. Please ignore the duplicate.

```
paretoPlot(mod9, xlab="Effect name", ylab="Magnitude of effect",
           main="Model 9 Pareto plot", legendtitle="Sign of coefficients",
           negative=c("Negative", "orange"),
           positive=c("Positive", "lightblue"))
```

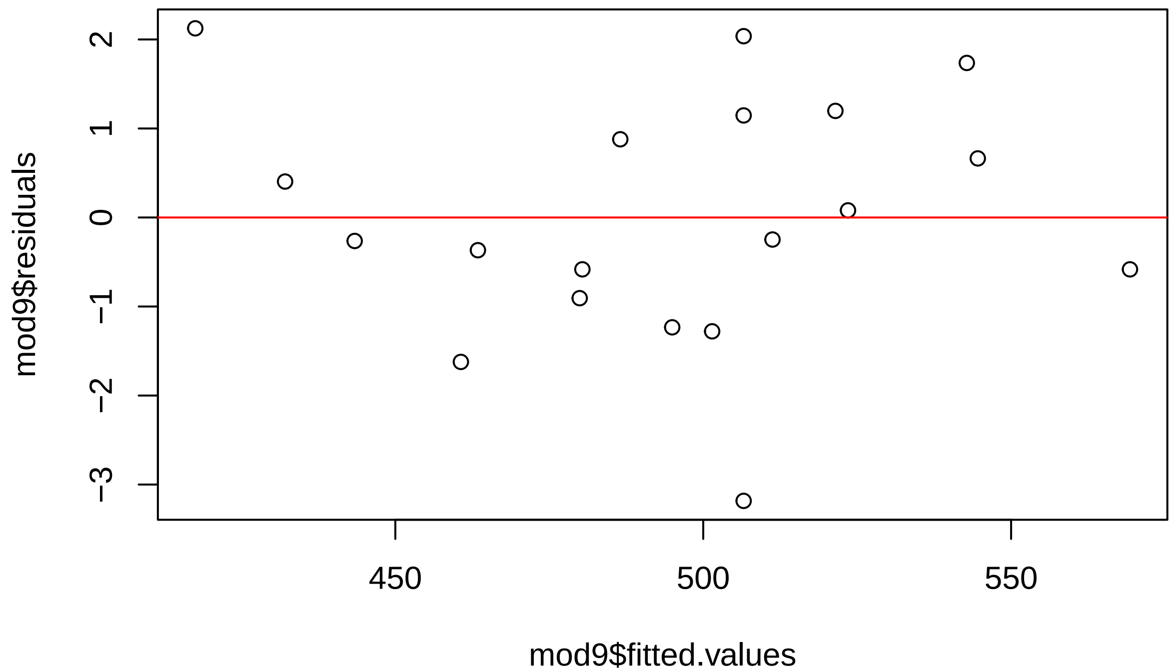





```r
plot(mod9, which=2, col=c("red"))
```




```
plot(mod9$fitted.values,mod9$residuals)
abline(0,0, col="red")
```



RSM

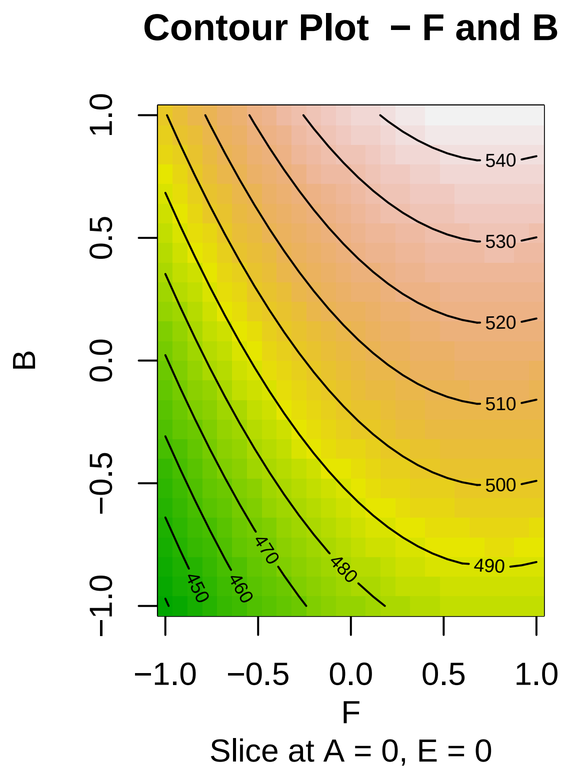
I recreate the model using an `rsm` module that is helpful in providing response surface modeling specific calculations (eigenvalues, eigenvectors, steepest ascent) and visualizations. From the output of the model, the eigenvalues have a mix of positive and negative values signaling a saddle system surface.

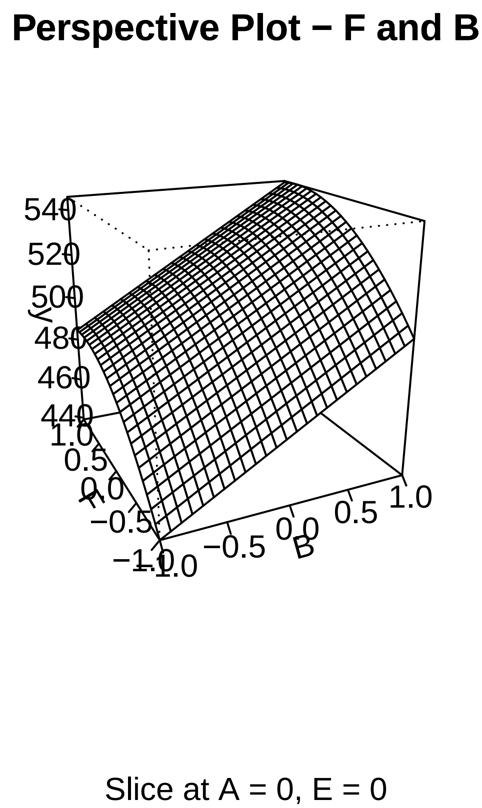
```
rsm_mod <- rsm(Y~FO(A,
                    B,
                    E,
                    F)+
                 TWI(A, E)+
                 TWI(A, F)+
                 TWI(B, E)+
                 PQ(F)
               ,data=results_coded)
summary(rsm_mod)
```

```
##
## Call:
## rsm(formula = Y ~ FO(A, B, E, F) + TWI(A, E) + TWI(A, F) + TWI(B,
##     E) + PQ(F), data = results_coded)
##
##                Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 506.56333    1.05670 479.3822 < 2.2e-16 ***
## A            -9.82687    0.45756 -21.4765 1.068e-09 ***
```

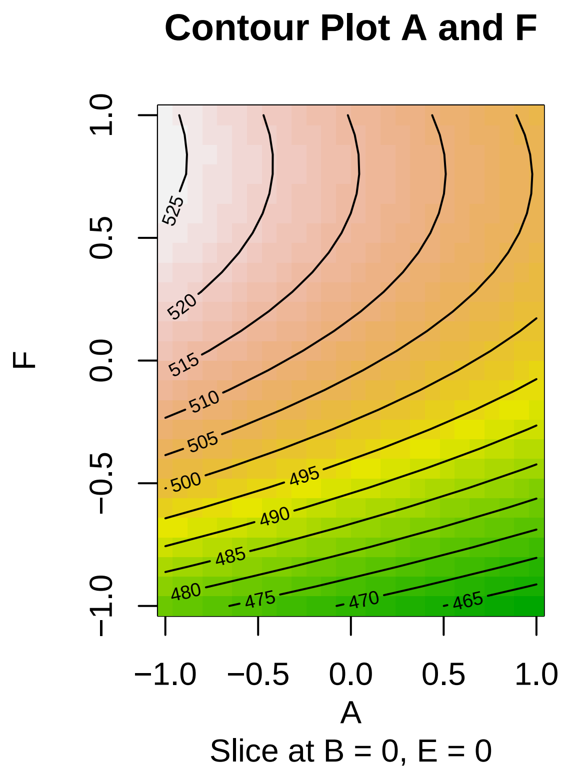
```
## B            30.23063    0.45756  66.0685 1.537e-14 ***
## E           -13.09687    0.45756 -28.6230 6.307e-11 ***
## F            22.74438    0.45756  49.7075 2.623e-13 ***
## A:E           1.35562    0.45756   2.9627   0.01422 *
## A:F          -1.17563    0.45756  -2.5693   0.02792 *
## B:E           1.20312    0.45756   2.6294   0.02519 *
## F^2         -14.48771    1.15151 -12.5815 1.870e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.9988, Adjusted R-squared:  0.9978
## F-statistic:  1037 on 8 and 10 DF,  p-value: 1.411e-13
##
## Analysis of Variance Table
##
## Response: Y
##                 Df  Sum Sq Mean Sq   F value    Pr(>F)
## FO(A, B, E, F)   4 27188.7  6797.2 2029.0993 1.691e-14
## TWI(A, E)        1    29.4    29.4    8.7776   0.01422
## TWI(A, F)        1    22.1    22.1    6.6013   0.02792
## TWI(B, E)        1    23.2    23.2    6.9138   0.02519
## PQ(F)            1   530.3   530.3  158.2931 1.870e-07
## Residuals       10    33.5     3.3
## Lack of fit      8    17.9     2.2    0.2870   0.91843
## Pure error       2    15.6     7.8
##
## Stationary point of response surface:
##           A           B           E           F
##   939.48049 -1047.67697   -25.12675   -37.33276
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1]   0.91262619   0.01049893  -0.89926277 -14.51157069
##
## $vectors
##          [,1]        [,2]        [,3]          [,4]
## A -0.53638765 -0.66337602  0.52016853  4.056171e-02
## B -0.46434393  0.74768883  0.47470635  7.867281e-05
## E -0.70445288  0.01304924 -0.70962826 -1.897834e-03
## F  0.02047328  0.02689579 -0.02250158  9.991752e-01
```

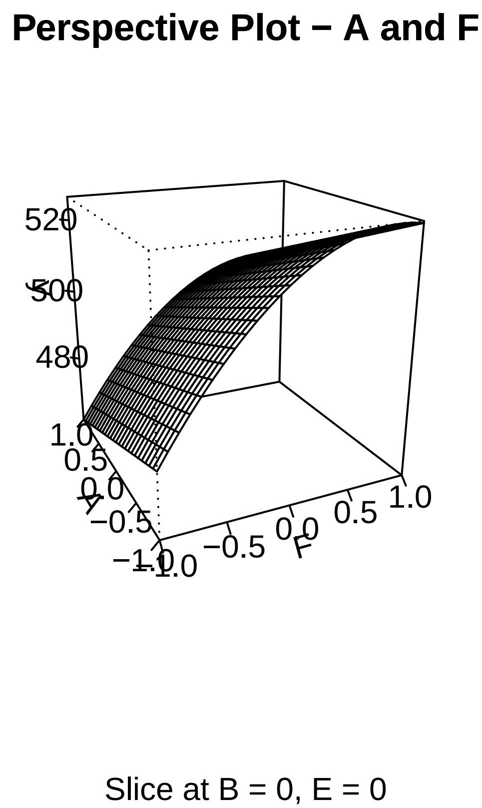
```r
par(mfrow = c(1, 2))

contour(rsm_mod, ~ F + B, image = TRUE, main="Contour Plot  - F and B")
persp(rsm_mod, F~B, zlab = "y", main="Perspective Plot - F and B")
```






```
contour(rsm_mod, ~ A + F, image = TRUE, main="Contour Plot A and F")
persp(rsm_mod, A~F, zlab = "y", main="Perspective Plot - A and F")
```






```r
par(mfrow = c(1, 1))
```

Testing and Evaluation

Steepest Ascent

Given the clues of being on a saddle system, ridge analysis is necessary to determine the method of steepest ascent. Below is the output from the ridge analysis in determining steepest ascent for maximizing Y.

```r
line_search_df <- steepest(rsm_mod)
```

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

```r
for(i in c("A", "B", "E", "F")){
  line_search_df <- uncode_variable(line_search_df, t_cop, i)
}
head(line_search_df)
```

```
##   dist      A     B      E     F |    yhat  A_uncoded  B_uncoded  E_uncoded
## 1  0.0  0.000 0.000  0.000 0.000 | 506.563  4341.000  259.1500  1540.000
## 2  0.5 -0.136 0.395 -0.168 0.218 | 526.297  4310.400  270.6642  1522.360
## 3  1.0 -0.294 0.819 -0.344 0.353 | 544.860  4274.850  283.0238  1503.880
## 4  1.5 -0.467 1.252 -0.517 0.443 | 562.797  4235.925  295.6458  1485.715
```

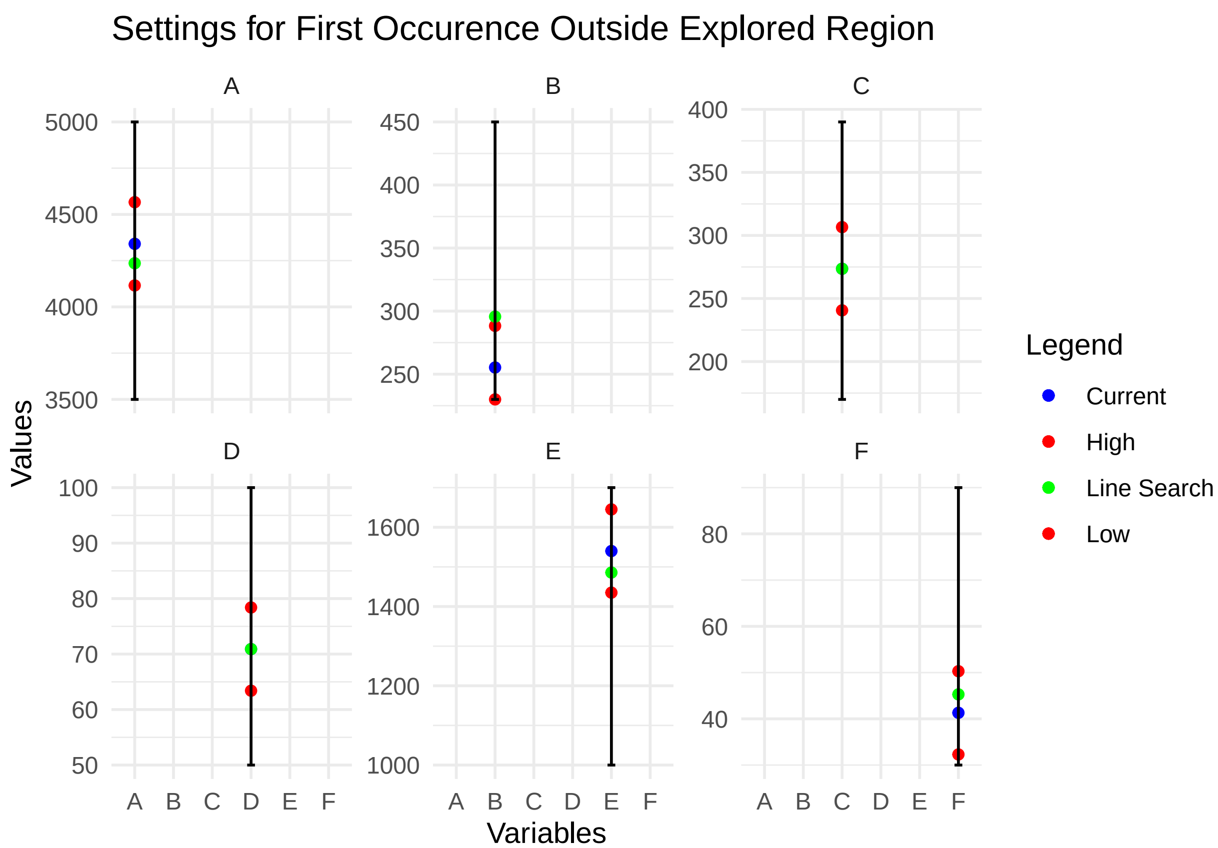
```
## 5  2.0 -0.652 1.688 -0.686 0.507 | 580.393  4194.300  308.3552  1467.970
## 6  2.5 -0.845 2.123 -0.849 0.555 | 597.682  4150.875  321.0354  1450.855
##   F_uncoded
## 1    41.300
## 2    43.262
## 3    44.477
## 4    45.287
## 5    45.863
## 6    46.295
```

In order to perform the next test runs for the line search I need to provide settings for factors C and D. I reuse the same setting as before since they did not impact the response.

```
actual_line_search_df <- line_search_df[,c("A_uncoded","B_uncoded")]
actual_line_search_df$C_uncoded <- t_cop$C[1]
actual_line_search_df$D_uncoded <- t_cop$D[1]
actual_line_search_df$E_uncoded <- line_search_df$E_uncoded
actual_line_search_df$F_uncoded <- line_search_df$F_uncoded
```

Below is a visualization of the first run that will get me out of the explored region. For the actual experiment I plan to run the settings one by one until the expected response is no longer suitable.

```
ggplot(cop, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_point(aes(y = t(actual_line_search_df)[,4], color = "Line Search"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  # Add legend
  labs(title = "Settings for First Occurence Outside Explored Region",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("High" = "red",
                                "Low" = "red",
                                "Current" = "blue",
                                "Line Search" = "green")) +
  facet_wrap(~variables, scales = "free_y")
```




```
actual_line_search_df$expected_y <- line_search_df$yhat
write.csv(actual_line_search_df, "linesearch1.csv", row.names = F)
```

Line Search Results

Please see appendix for line search outputs. The line search results was promising - leading to an 18% increase in Y. The line search ended after 7 runs because the actual result were becoming farther and farther away from the expected results.

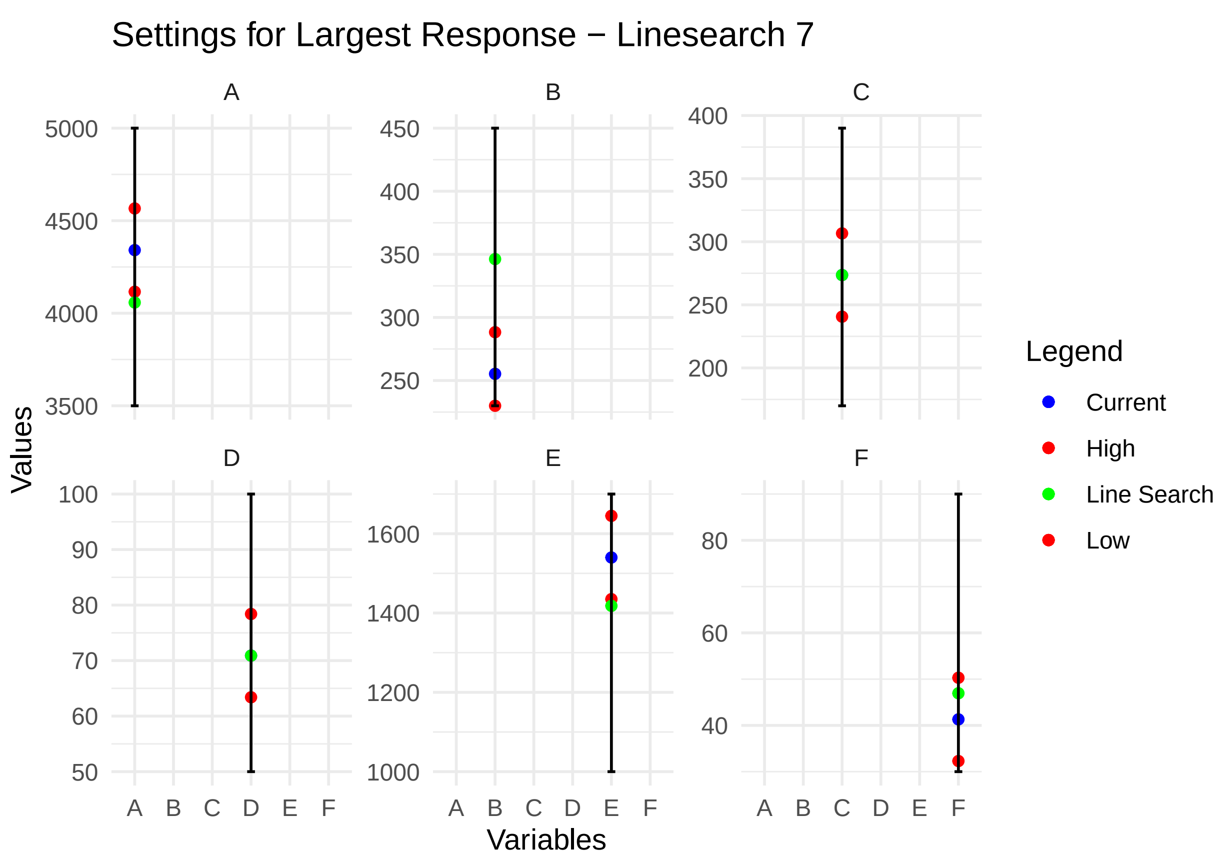
```
linesearch_results <- read.csv("linesearch1_results.csv")
linesearch_results[8,]
```

```
##   A_uncoded B_uncoded C_uncoded D_uncoded E_uncoded F_uncoded expected_y
## 8   4057.95  346.2502     273.6      70.9  1418.095    46.943    631.759
##   actual_y percent_diff
## 8   595.32  0.007718871
```

```
(max(linesearch_results$actual_y, na.rm = TRUE) -
    min(linesearch_results$actual_y, na.rm = TRUE))/
  min(linesearch_results$actual_y, na.rm = TRUE)
```

```
## [1] 0.1825513
```

```r
ggplot(cop, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_point(aes(y = t(actual_line_search_df)[1:6,8], color = "Line Search"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  # Add legend
  labs(title = "Settings for Largest Response - Linesearch 7",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("High" = "red",
                                "Low" = "red",
                                "Current" = "blue",
                                "Line Search" = "green")) +
  facet_wrap(~variables, scales = "free_y")
```



Experiment 2

Setup

High and Low Assignment

With the results of the final line search trial, I can use that as the new current operating position and apply the same 15% increase and decrease for the high and low settings for the next experiment.

```
t(linesearch_results[8,c("A_uncoded",
                              "B_uncoded",
                              "C_uncoded",
                              "D_uncoded",
                              "E_uncoded",
                              "F_uncoded")])
```

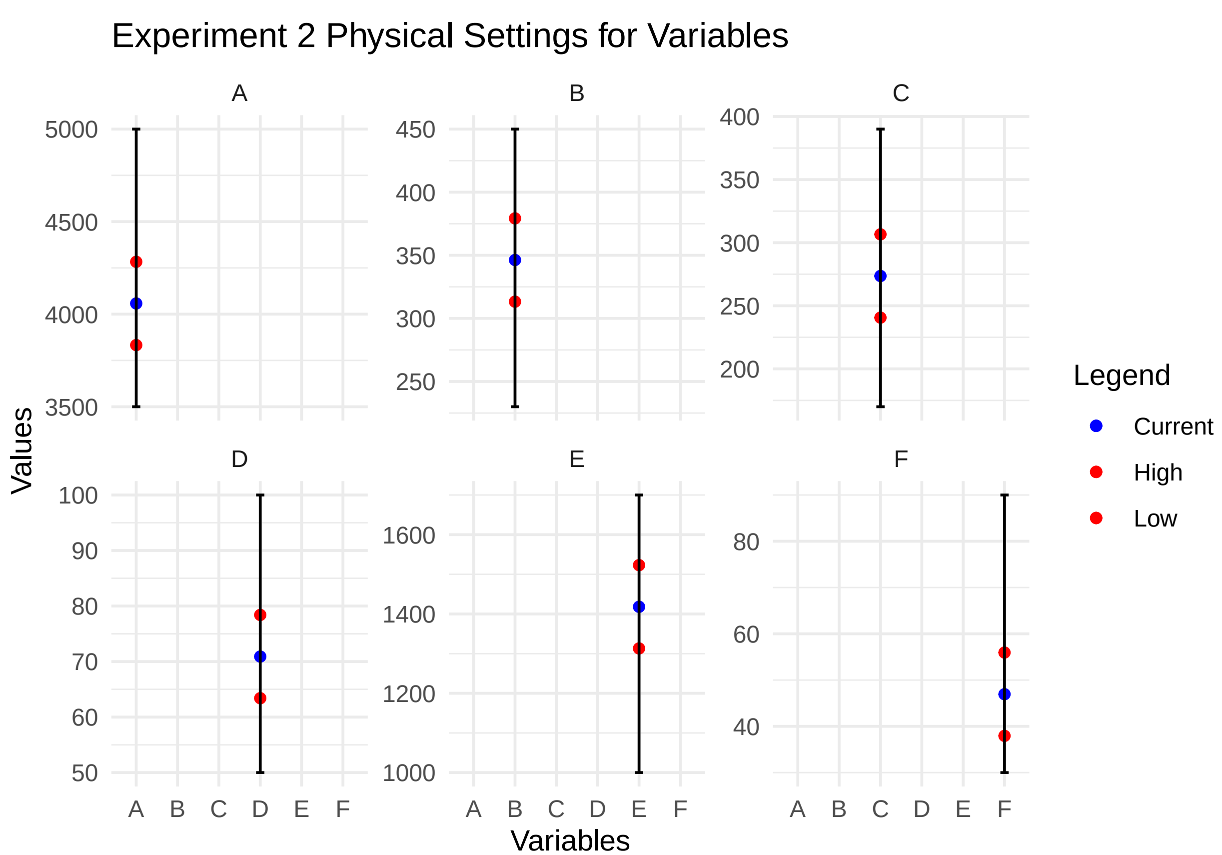
```
##                   8
## A_uncoded 4057.9500
## B_uncoded  346.2502
## C_uncoded  273.6000
## D_uncoded   70.9000
## E_uncoded 1418.0950
## F_uncoded   46.9430
```

```
cop2 <- data.frame(variables=c("A", "B", "C", "D", "E", "F"),
                  current=c(t(linesearch_results[8,c("A_uncoded",
                              "B_uncoded",
                              "C_uncoded",
                              "D_uncoded",
                              "E_uncoded",
                              "F_uncoded")])),
                  lower=c(3500,230,170,50,1000,30),
                  upper=c(5000,450,390,100,1700,90))

cop2$range <- cop2$upper - cop2$lower
cop2$change <- cop2$range * .15
cop2$increase <- cop2$current + cop2$change
cop2$decrease <- cop2$current - cop2$change

# applying a min and max to avoid going above or below a limit
cop2$high <- apply(cop2[, c("upper","increase")], 1, min)
cop2$low <- apply(cop2[, c("lower","decrease")], 1, max)

cop2
```

```
##   variables   current lower upper range change  increase  decrease      high
## 1         A 4057.9500  3500  5000  1500  225.0 4282.9500 3832.9500 4282.9500
## 2         B  346.2502   230   450   220   33.0  379.2502  313.2502  379.2502
## 3         C  273.6000   170   390   220   33.0  306.6000  240.6000  306.6000
## 4         D   70.9000    50   100    50    7.5   78.4000   63.4000   78.4000
## 5         E 1418.0950  1000  1700   700  105.0 1523.0950 1313.0950 1523.0950
## 6         F   46.9430    30    90    60    9.0   55.9430   37.9430   55.9430
```

```
##          low
## 1 3832.9500
## 2  313.2502
## 3  240.6000
## 4   63.4000
## 5 1313.0950
## 6   37.9430
```

```
ggplot(cop2, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  labs(title = "Experiment 2 Physical Settings for Variables",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("blue", "red", "red")) +
  facet_wrap(~variables, scales = "free_y")
```



Fractional Design

For the next experiment , I decide to use 2^{6-3} fractional factorial since I have a good idea of the interaction effects that are important.

```r
eight_design <- FrF2(nfactors = 6,
                     nruns = 2^(6-3),
                     generators=c("AB","AC","BC"),
                     ncenter = 3,
                     randomize=FALSE)

eight_design
```

```
##     A  B  C  D  E  F
## 1  -1 -1 -1  1  1  1
## 2   1 -1 -1 -1 -1  1
## 3  -1  1 -1 -1  1 -1
## 4   1  1 -1  1 -1 -1
## 5  -1 -1  1  1 -1 -1
## 6   1 -1  1 -1  1 -1
## 7  -1  1  1 -1 -1  1
## 8   1  1  1  1  1  1
## 9   0  0  0  0  0  0
## 10  0  0  0  0  0  0
## 11  0  0  0  0  0  0
## class=design, type= FrF2.generators.center
```

```r
aliasprint(eight_design)
```

```
## $legend
## [1] A=A B=B C=C D=D E=E F=F
##
## $main
## [1] A=BD=CE B=AD=CF C=AE=BF D=AB=EF E=AC=DF F=BC=DE
##
## $fi2
## [1] AF=BE=CD
```

Design Encoding

```r
t_cop <- data.frame(t(cop2[,-1]))
colnames(t_cop) <- cop2$variables

write.csv(as.data.frame(eight_design), "design.csv", row.names = F)
deisgn <- read.csv("design.csv")

for(i in colnames(deisgn)){
  deisgn <- uncode_variable(deisgn, t_cop, i)
}

deisgn_coded <- deisgn[,1:6]
```

```r
deisgn_uncoded <- deisgn[,7:12]
head(deisgn_uncoded)
```

```
##   A_uncoded B_uncoded C_uncoded D_uncoded E_uncoded F_uncoded
## 1   3832.95  313.2502     240.6      78.4  1523.095    55.943
## 2   4282.95  313.2502     240.6      63.4  1313.095    55.943
## 3   3832.95  379.2502     240.6      63.4  1523.095    37.943
## 4   4282.95  379.2502     240.6      78.4  1313.095    37.943
## 5   3832.95  313.2502     306.6      78.4  1313.095    37.943
## 6   4282.95  313.2502     306.6      63.4  1523.095    37.943
```

```r
write.csv(deisgn_uncoded, "exp_2.csv", row.names = F)
write.csv(deisgn_coded, "exp_2_coded.csv", row.names = F)
```

Results

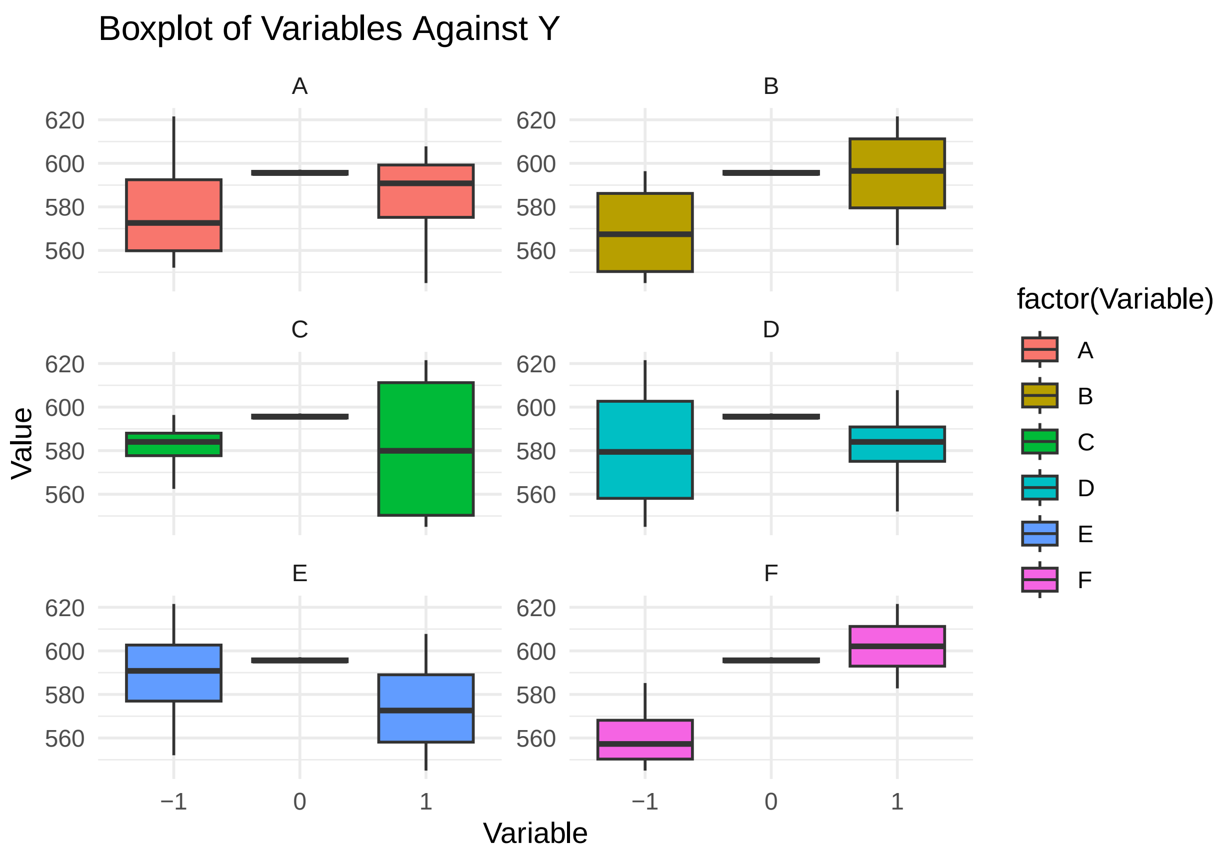
Exploratory Data Analysis

Below are the results of experiment 2 and the results visualized as box plots again.

```r
results_coded <- read.csv("exp_2_results_coded.csv")
head(results_coded)
```

```
##    A  B  C  D  E  F      Y
## 1 -1 -1 -1  1  1  1 582.78
## 2  1 -1 -1 -1 -1  1 596.39
## 3 -1  1 -1 -1  1 -1 562.45
## 4  1  1 -1  1 -1 -1 585.24
## 5 -1 -1  1  1 -1 -1 552.08
## 6  1 -1  1 -1  1 -1 545.03
```

```r
df_long <- gather(results_coded, key = "Variable", value = "Value", -Y)

ggplot(df_long, aes(x = factor(Value), y = Y)) +
  geom_boxplot(aes(fill = factor(Variable)), position = "dodge") +
  labs(title = "Boxplot of Variables Against Y",
       x = "Variable",
       y = "Value") +
  facet_wrap(~ Variable, scales = "free_y", ncol = 2) +
  theme_minimal()
```



Comparing to the previous box plots, it seems F is still a fair contributor. B is still better in the high position but looks like it is not as impactful as before. C and D look suspicious and can potentially be experimental error, though hard to tell. A performs worse in the low setting and E performs worse in the high setting.

Modeling

I use a backwards selection modeling technique again to provide a suitable first order model.

```r
mod1 <- lm(Y~A+B+C+D+E+F, data=results_coded)
summary(mod1)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + C + D + E + F, data = results_coded)
##
## Residuals:
##      1      2      3      4      5      6      7      8      9     10     11
## -1.842 -5.814 -5.814 -1.842 -5.814 -1.842 -1.842 -5.814  8.981 10.051 11.591
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 585.48909    3.25020 180.139  5.7e-09 ***
## A             1.94875    3.81120   0.511  0.63606
## B            12.59125    3.81120   3.304  0.02983 *
## C            -0.05375    3.81120  -0.014  0.98942
```

```
## D             0.30875    3.81120   0.081  0.93932
## E            -7.15125    3.81120  -1.876  0.13384
## F            20.46125    3.81120   5.369  0.00581 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.78 on 4 degrees of freedom
## Multiple R-squared:  0.9158, Adjusted R-squared:  0.7896
## F-statistic: 7.254 on 6 and 4 DF,  p-value: 0.03788
```

Remove C

```r
mod2 <- lm(Y~A+B+D+E+F, data=results_coded)
summary(mod2)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + D + E + F, data = results_coded)
##
## Residuals:
##      1      2      3      4      5      6      7      8      9     10     11
## -1.788 -5.760 -5.760 -1.788 -5.868 -1.895 -1.895 -5.868  8.981 10.051 11.591
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 585.4891     2.9071 201.397 5.73e-11 ***
## A             1.9487     3.4089   0.572  0.59229
## B            12.5912     3.4089   3.694  0.01409 *
## D             0.3088     3.4089   0.091  0.93135
## E            -7.1512     3.4089  -2.098  0.09000 .
## F            20.4612     3.4089   6.002  0.00184 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.642 on 5 degrees of freedom
## Multiple R-squared:  0.9158, Adjusted R-squared:  0.8317
## F-statistic: 10.88 on 5 and 5 DF,  p-value: 0.01017
```

Remove D

```r
mod3 <- lm(Y~A+B+E+F, data=results_coded)
summary(mod3)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F, data = results_coded)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.069 -5.559 -2.204  3.751 11.591
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  585.489      2.656 220.439 5.88e-13 ***
## A              1.949      3.114   0.626 0.554544
## B             12.591      3.114   4.043 0.006781 **
## E             -7.151      3.114  -2.296 0.061424 .
## F             20.461      3.114   6.570 0.000596 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.809 on 6 degrees of freedom
## Multiple R-squared:  0.9157, Adjusted R-squared:  0.8595
## F-statistic: 16.29 on 4 and 6 DF,  p-value: 0.002245
```

Remove A

```
mod4 <- lm(Y~B+E+F, data=results_coded)
summary(mod4)
```

```
##
## Call:
## lm.default(formula = Y ~ B + E + F, data = results_coded)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.018 -4.137 -3.428  4.725 11.591
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  585.489      2.538 230.693 7.59e-15 ***
## B             12.591      2.976   4.231 0.003884 **
## E             -7.151      2.976  -2.403 0.047260 *
## F             20.461      2.976   6.875 0.000237 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.417 on 7 degrees of freedom
## Multiple R-squared:  0.9102, Adjusted R-squared:  0.8717
## F-statistic: 23.65 on 3 and 7 DF,  p-value: 0.0004874
```

I am assuming the curvature test performed previous still holds true and move forward with interactions and squared terms with the model using B, E, and F as variables.

```
mod5 <- lm(Y~B+
             E+
             F+
             B*E+
             B*F+
             E*F+
             I(F^2), data=results_coded)
summary(mod5)
```

```
##
```

```
## Call:
## lm.default(formula = Y ~ B + E + F + B * E + B * F + E * F +
##     I(F^2), data = results_coded)
##
## Residuals:
##       1       2       3       4       5       6       7       8       9      10
## -1.9487  1.9487 -1.9487  1.9487 -1.9487  1.9487 -1.9487  1.9487 -1.2267 -0.1567
##      11
##  1.3833
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 595.69667    1.93861 307.280  7.6e-08 ***
## B            12.59125    1.18715  10.606 0.001791 **
## E            -7.15125    1.18715  -6.024 0.009170 **
## F            20.46125    1.18715  17.236 0.000426 ***
## I(F^2)      -14.03542    2.27322  -6.174 0.008554 **
## B:E          -1.98625    1.18715  -1.673 0.192897
## B:F          -0.05375    1.18715  -0.045 0.966732
## E:F           0.30875    1.18715   0.260 0.811633
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.358 on 3 degrees of freedom
## Multiple R-squared:  0.9939, Adjusted R-squared:  0.9796
## F-statistic: 69.55 on 7 and 3 DF,  p-value: 0.00258
```

Interesting enough, the interactions lose significance and get dropped.

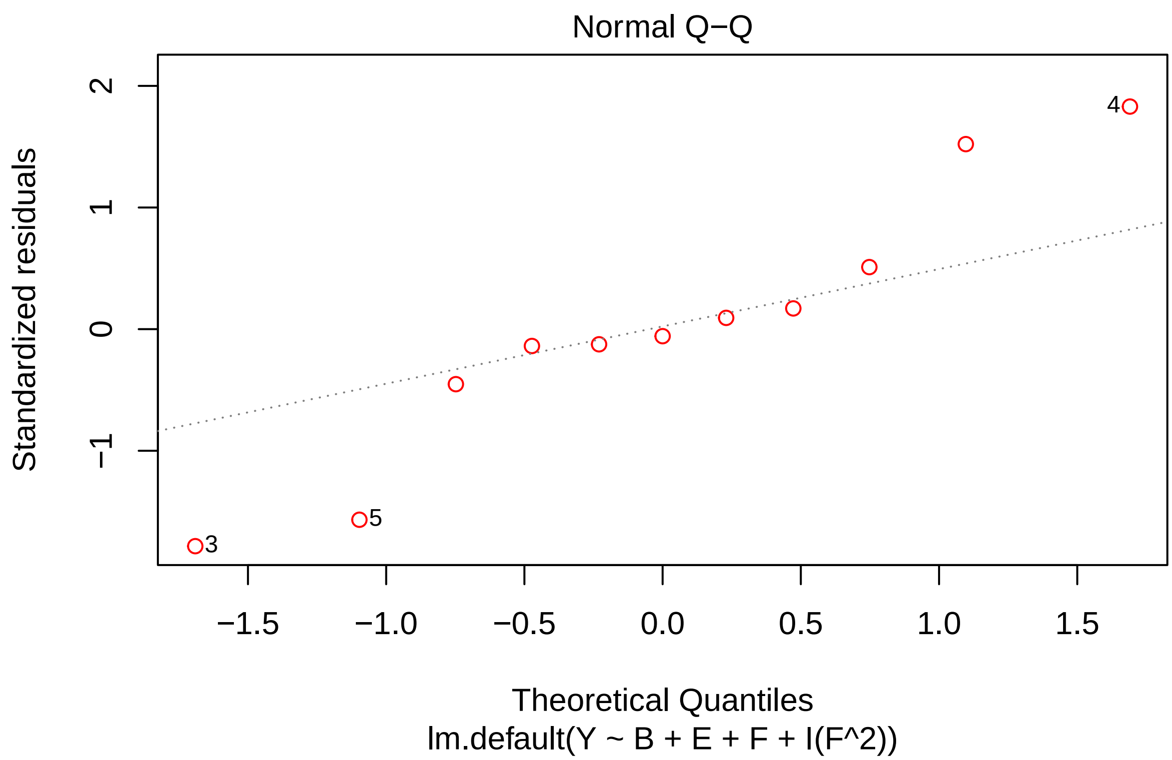
```
mod6 <- lm(Y~B+
             E+
             F+
             I(F^2), data=results_coded)
summary(mod6)
```

```
##
## Call:
## lm.default(formula = Y ~ B + E + F + I(F^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1900 -0.7758 -0.1567  0.8917  4.2975
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  595.697      1.917 310.690 7.50e-14 ***
## B             12.591      1.174  10.724 3.88e-05 ***
## E             -7.151      1.174  -6.091 0.000891 ***
## F             20.461      1.174  17.427 2.29e-06 ***
## I(F^2)       -14.035      2.248  -6.243 0.000783 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 3.321 on 6 degrees of freedom
## Multiple R-squared:  0.988,  Adjusted R-squared:   0.98
## F-statistic: 123.7 on 4 and 6 DF,  p-value: 6.818e-06
```

The residuals qqplot for the model provided by backwards elimination process looks to have a violation of normality.

```
plot(mod6, which=2, col=c("red"))
```



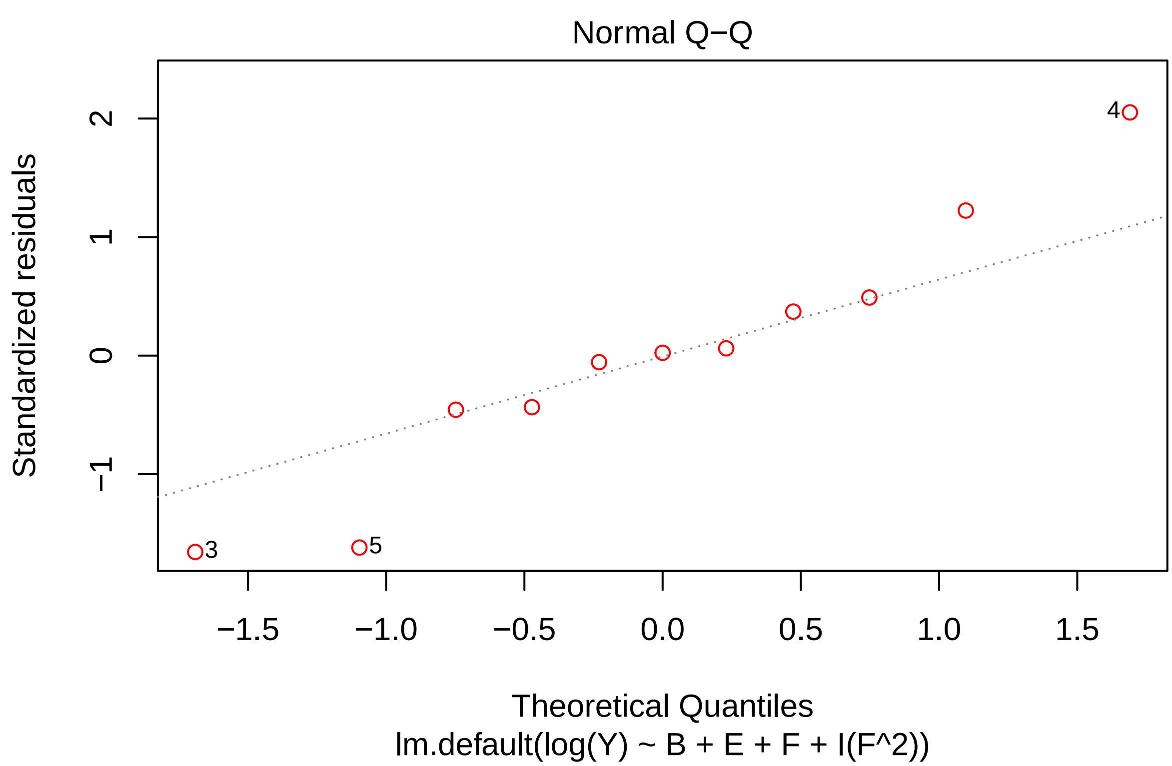
I apply a log transformation to the Y term to try and fix the issue. After applying the transformation, the residuals still show that same “S” pattern in the qqplot. I try applying an exponential transform to F smaller than 2, but received strange errors that I believe could be stemming from the result of using a smaller fractional experiment and not enough degrees of freedom. (See appendix for attempt).

```
mod7 <- lm(log(Y)~B+
             E+
             F+
             I(F^2), data=results_coded)
summary(mod7)
```

```
##
## Call:
## lm.default(formula = log(Y) ~ B + E + F + I(F^2), data = results_coded)
##
## Residuals:
```

```
##        Min         1Q     Median         3Q        Max
## -0.0067941 -0.0019659  0.0000972  0.0019222  0.0084146
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  6.389730   0.003349 1908.161  < 2e-16 ***
## B            0.021637   0.002051   10.552 4.26e-05 ***
## E           -0.012256   0.002051   -5.977 0.000985 ***
## F            0.035232   0.002051   17.181 2.49e-06 ***
## I(F^2)      -0.024783   0.003927   -6.312 0.000738 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0058 on 6 degrees of freedom
## Multiple R-squared:  0.9877, Adjusted R-squared:  0.9795
## F-statistic: 120.5 on 4 and 6 DF,  p-value: 7.362e-06
```

```r
plot(mod7, which=2, col=c("red"))
```



I also test if changing the squared term to B provided any different results. It doesnt.

```r
mod7b <- lm(log(Y)~B+
             E+
             F+
             I(B^2), data=results_coded)
summary(mod7b)
```

```
##
## Call:
## lm.default(formula = log(Y) ~ B + E + F + I(B^2), data = results_coded)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0067941 -0.0019659  0.0000972  0.0019222  0.0084146
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  6.389730   0.003349 1908.161  < 2e-16 ***
## B            0.021637   0.002051   10.552 4.26e-05 ***
## E           -0.012256   0.002051   -5.977 0.000985 ***
## F            0.035232   0.002051   17.181 2.49e-06 ***
## I(B^2)      -0.024783   0.003927   -6.312 0.000738 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0058 on 6 degrees of freedom
## Multiple R-squared:  0.9877, Adjusted R-squared:  0.9795
## F-statistic: 120.5 on 4 and 6 DF,  p-value: 7.362e-06
```

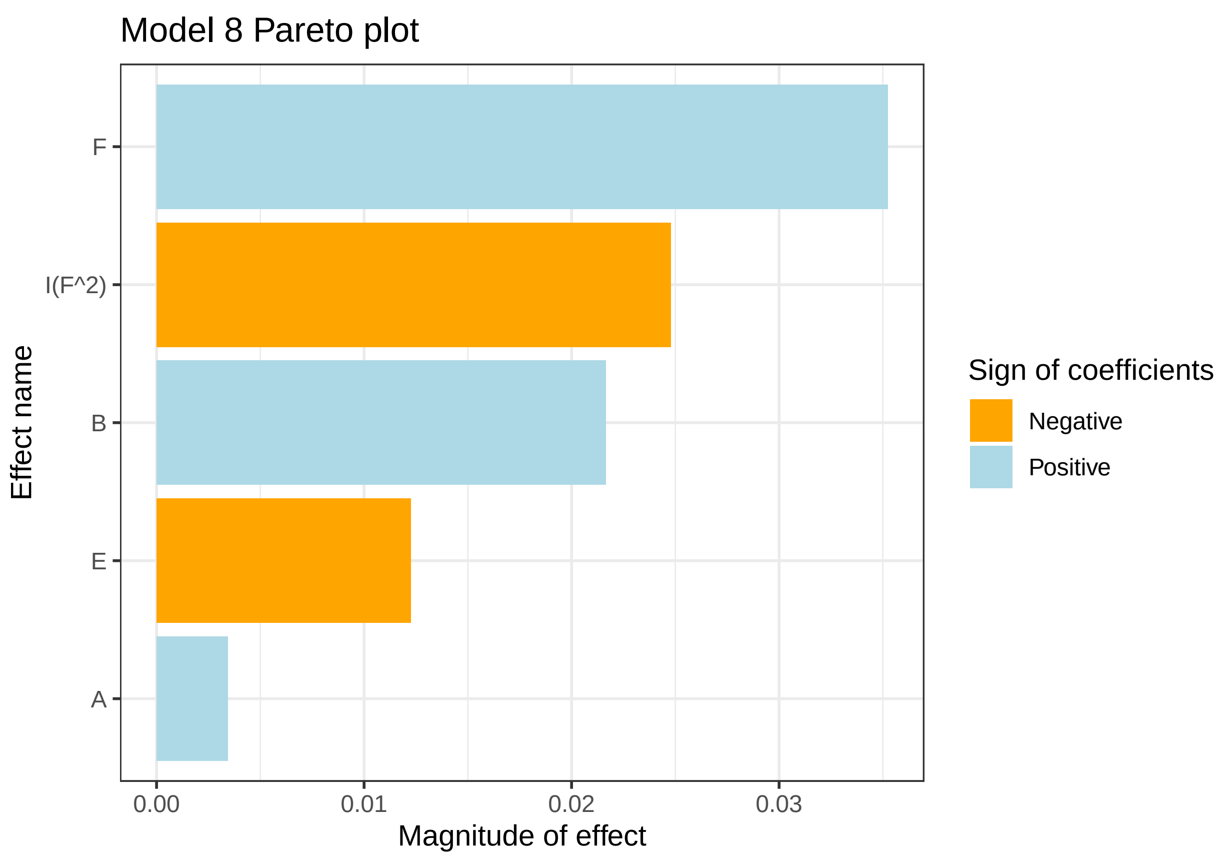
I decide to try adding back in A and find marginal significance and decide to use this as the final model.

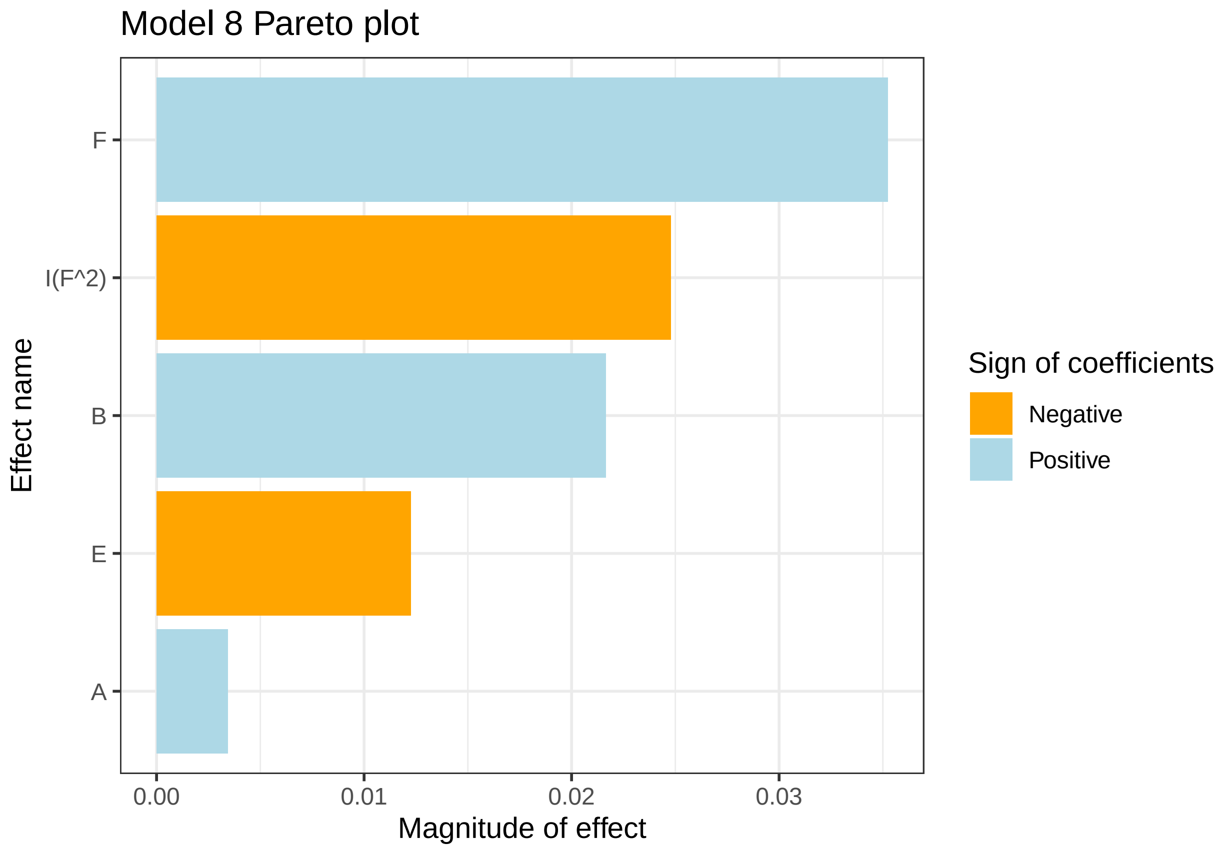
```
mod8 <- lm(log(Y)~
             A+
             B+
             E+
             F+
             I(F^2), data=results_coded)
summary(mod8)
```

```
##
## Call:
## lm.default(formula = log(Y) ~ A + B + E + F + I(F^2), data = results_coded)
##
## Residuals:
##          1          2          3          4          5          6          7
##  0.0049689 -0.0033485 -0.0033485  0.0049689 -0.0031940  0.0015736  0.0015736
##          8          9         10         11
## -0.0031940 -0.0020597 -0.0002614  0.0023211
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  6.389730   0.002669 2393.997 2.41e-16 ***
## A            0.003446   0.001634    2.108 0.088831 .
## B            0.021637   0.001634   13.238 4.40e-05 ***
## E           -0.012256   0.001634   -7.498 0.000667 ***
## F            0.035232   0.001634   21.556 3.99e-06 ***
## I(F^2)      -0.024783   0.003130   -7.919 0.000517 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.004623 on 5 degrees of freedom
## Multiple R-squared:  0.9935, Adjusted R-squared:  0.987
## F-statistic: 152.7 on 5 and 5 DF,  p-value: 1.843e-05
```

```r
paretoPlot(mod8, xlab="Effect name", ylab="Magnitude of effect",
           main="Model 8 Pareto plot", legendtitle="Sign of coefficients",
           negative=c("Negative", "orange"),
           positive=c("Positive", "lightblue"))
```





Given that the model provide negative effects when squared and that the magnitude of the effects are low, I suspect that I will see small returns in the coming line search and that I am approaching the steepest part of the response surface. Furthermore, I note that I should keep an eye on adjusting variables B and F since I believe that both have negative effects as input increases and that there will be a tipping point where I see a drop in response as they get larger.

RSM

I again use the **rsm** module to recreate the model and from looking at the results - a notification of a near stationary point is provided further giving me indications that I am approaching a peak on the surface.

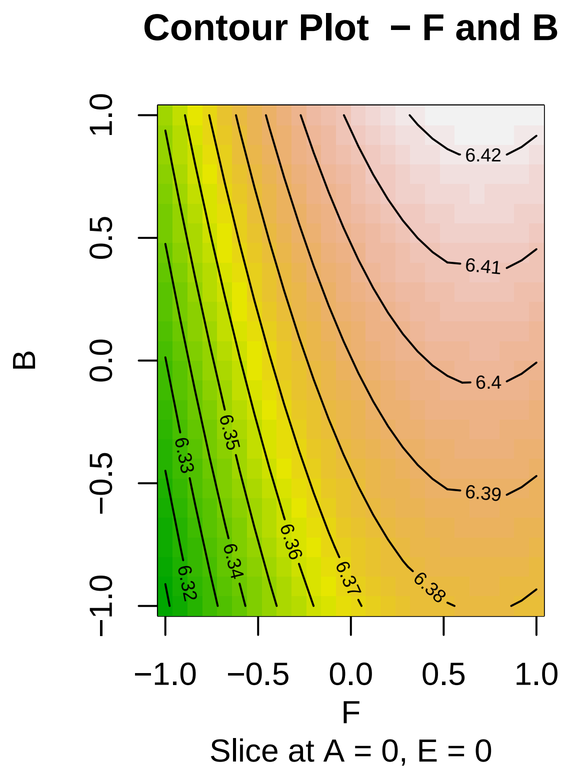
```
rsm_mod <- rsm(log(Y)~FO(A,
                    B,
                    E,
                    F)+
                 PQ(F)
               ,data=results_coded)
summary(rsm_mod)
```

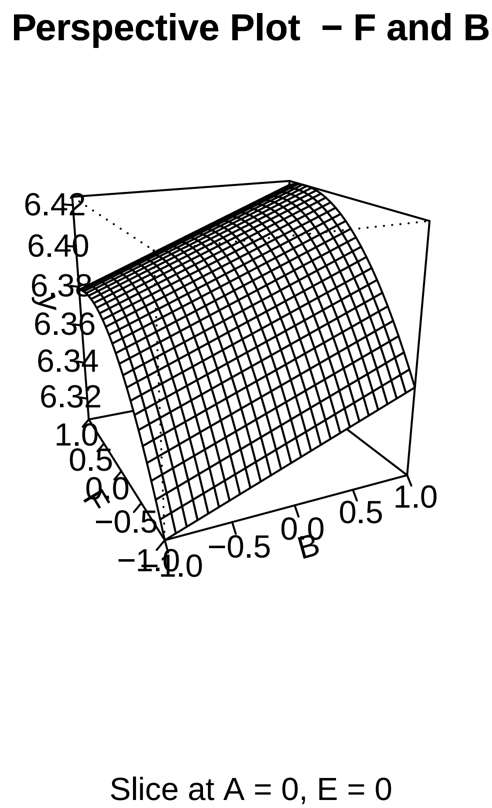
```
## Near-stationary-ridge situation detected -- stationary point altered
##  Change 'threshold' if this is not what you intend
```

```
##
## Call:
## rsm(formula = log(Y) ~ FO(A, B, E, F) + PQ(F), data = results_coded)
```

```
##
##              Estimate Std. Error   t value  Pr(>|t|)
## (Intercept)  6.3897300  0.0026691 2393.9971 2.414e-16 ***
## A            0.0034457  0.0016345    2.1081 0.0888308 .
## B            0.0216374  0.0016345   13.2382 4.395e-05 ***
## E           -0.0122555  0.0016345   -7.4982 0.0006670 ***
## F            0.0352325  0.0016345   21.5560 3.986e-06 ***
## F^2         -0.0247830  0.0031298   -7.9185 0.0005171 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.9935, Adjusted R-squared:  0.987
## F-statistic: 152.7 on 5 and 5 DF,  p-value: 1.843e-05
##
## Analysis of Variance Table
##
## Response: log(Y)
##                 Df    Sum Sq   Mean Sq  F value    Pr(>F)
## FO(A, B, E, F)   4 0.0149726 0.0037431 175.1451 1.472e-05
## PQ(F)            1 0.0013401 0.0013401  62.7028 0.0005171
## Residuals        5 0.0001069 0.0000214
## Lack of fit      3 0.0000972 0.0000324   6.6787 0.1330023
## Pure error       2 0.0000097 0.0000048
##
## Stationary point of response surface:
##         A         B         E         F
## 0.0000000 0.0000000 0.0000000 0.7108197
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1]  0.00000000  0.00000000  0.00000000 -0.02478299
##
## $vectors
##   [,1] [,2] [,3] [,4]
## A    1    0    0    0
## B    0    0    1    0
## E    0    1    0    0
## F    0    0    0    1
```

```r
par(mfrow = c(1, 2))
contour(rsm_mod, ~ F + B, image = TRUE, main="Contour Plot  - F and B")
persp(rsm_mod, F~B, zlab = "y", main="Perspective Plot  - F and B")
```






```r
par(mfrow = c(1, 1))
```

Testing and Evaluation

Steepest Ascent

I again follow through with steepest ascent and write to csv the trials to run.

```r
line_search_df2 <- steepest(rsm_mod)
```

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

```r
line_search_df2$yhat <- exp(line_search_df2$yhat)

for(i in c("A", "B", "E", "F")){
  line_search_df2 <- uncode_variable(line_search_df2, t_cop, i)
}
head(line_search_df2)
```

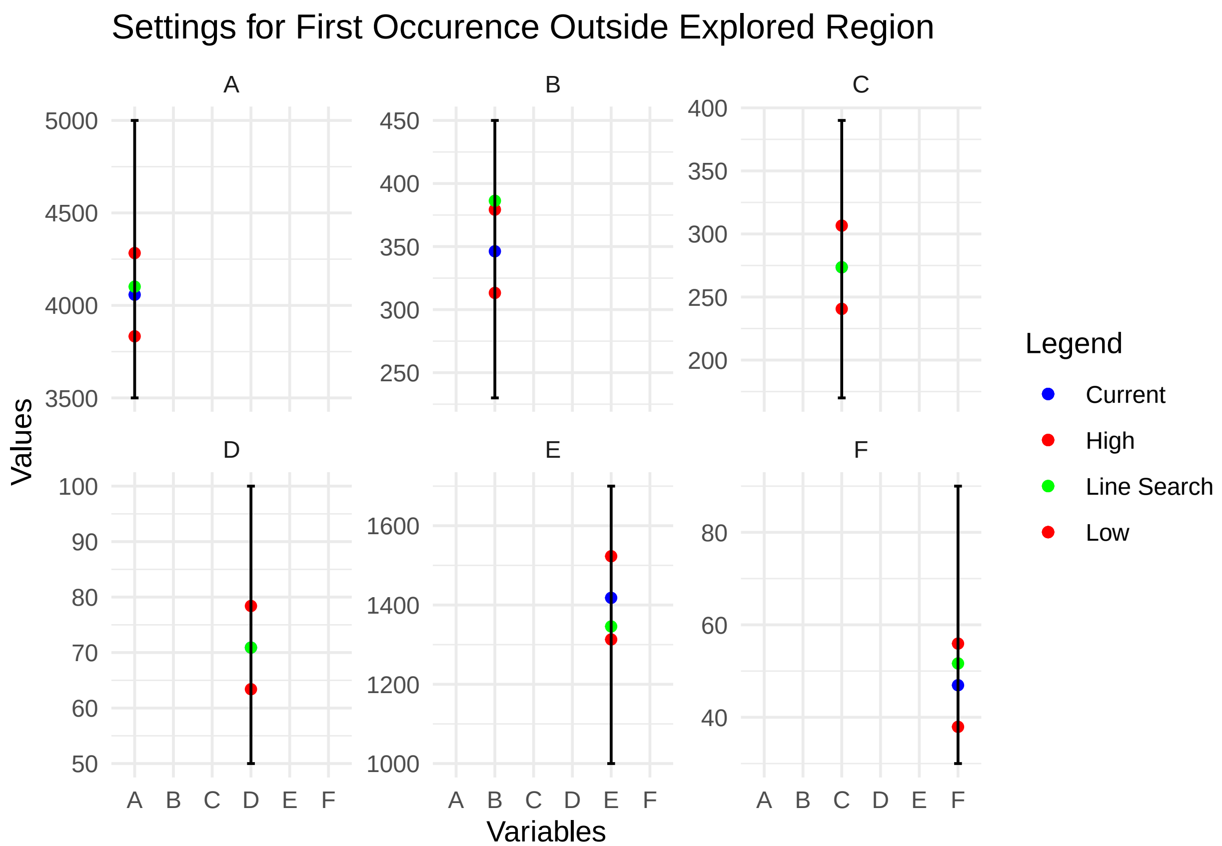
```
##   dist     A     B      E     F |     yhat A_uncoded B_uncoded E_uncoded
## 1  0.0 0.000 0.000  0.000 0.000 | 595.8566  4057.950  346.2502  1418.095
## 2  0.5 0.054 0.338 -0.191 0.310 | 606.6791  4070.100  357.4042  1398.040
## 3  1.0 0.122 0.768 -0.435 0.453 | 615.8479  4085.400  371.5942  1372.420
```

```
## 4  1.5 0.193 1.214 -0.687 0.523 | 624.5304  4101.375  386.3122  1345.960
## 5  2.0 0.264 1.655 -0.937 0.562 | 632.7023  4117.350  400.8652  1319.710
## 6  2.5 0.333 2.091 -1.184 0.588 | 640.9811  4132.875  415.2532  1293.775
##   F_uncoded
## 1    46.943
## 2    49.733
## 3    51.020
## 4    51.650
## 5    52.001
## 6    52.235
```

```
actual_line_search_df2 <- line_search_df2[,c("A_uncoded","B_uncoded")]
actual_line_search_df2$C_uncoded <- t_cop$C[1]
actual_line_search_df2$D_uncoded <- t_cop$D[1]
actual_line_search_df2$E_uncoded <- line_search_df2$E_uncoded
actual_line_search_df2$F_uncoded <- line_search_df2$F_uncoded

# applying limits
# Apply the upper limits using a for loop
for (col_name in colnames(t_cop)) {
  actual_line_search_df2[, paste0(col_name,"_uncoded")] <- pmin(
    actual_line_search_df2[,
                           paste0(col_name,"_uncoded")],
    t_cop[3,col_name])
}

# Apply the lower limits using a for loop
for (col_name in colnames(t_cop)) {
  actual_line_search_df2[, paste0(col_name,"_uncoded")] <- pmax(
    actual_line_search_df2[,
                           paste0(col_name,"_uncoded")],
    t_cop[2,col_name])
}
```

```
ggplot(cop2, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_point(aes(y = t(actual_line_search_df2)[,4], color = "Line Search"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  # Add legend
  labs(title = "Settings for First Occurence Outside Explored Region",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("High" = "red",
                                "Low" = "red",
```

```
                                       "Current" = "blue",
                                       "Line Search" = "green")) +
  facet_wrap(~variables, scales = "free_y")
```




```
actual_line_search_df2$expected_y <- line_search_df2$yhat
write.csv(actual_line_search_df2, "linesearch2.csv", row.names = F)
```

Line Search Results 2

To save on resources, instead of testing each run provided by the line search that had an incremental change of .5, I performed every other trial to test a delta of 1.

The line search results was promising - leading to a 4% increase of Y from the previous settings (Y=619.81) and further leading me to believe I was experiencing diminishing returns due being closer to the peak. I ended the line search fter the third trial when I saw that my response went down.

```
linesearch_results <- read.csv("linesearch2_results.csv")
linesearch_results[c(3,5,7),]
```

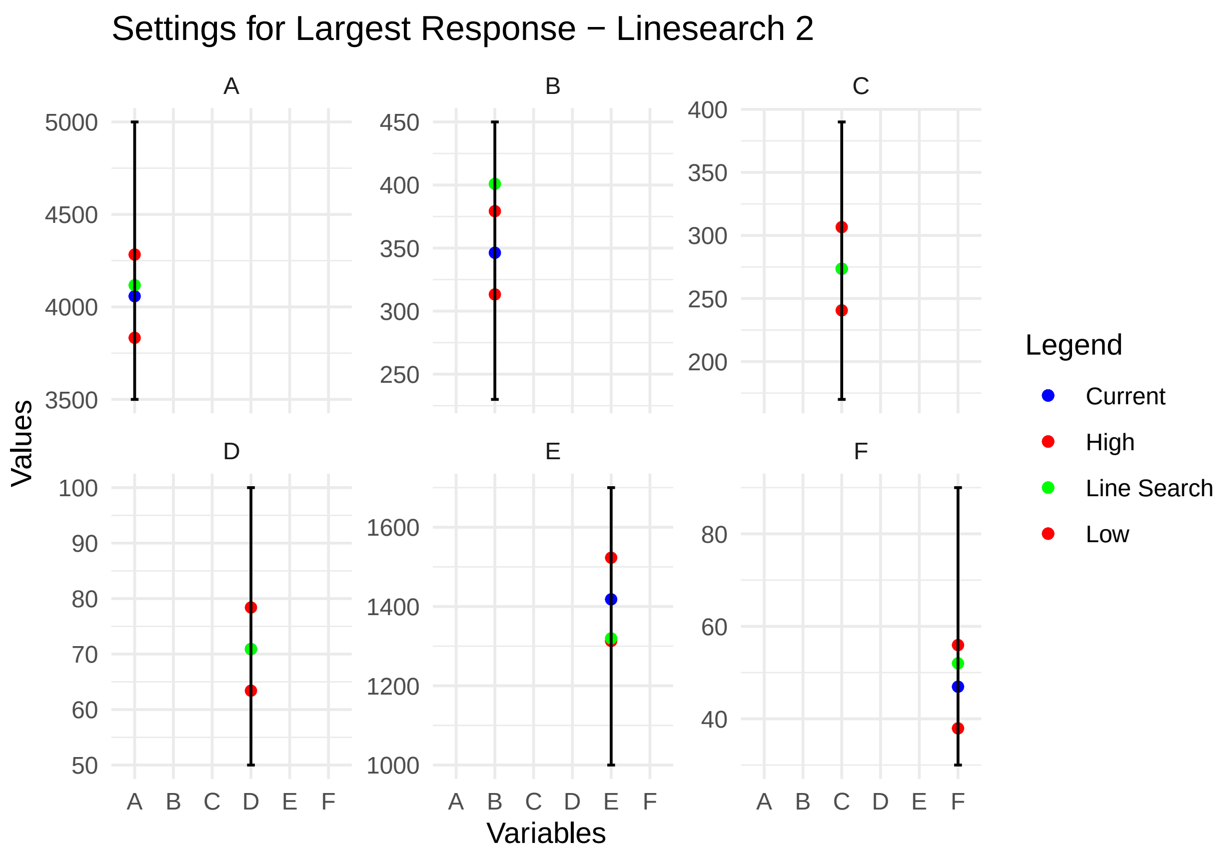
```
##   A_uncoded B_uncoded C_uncoded D_uncoded E_uncoded F_uncoded expected_y
## 3   4085.40  371.5942     273.6      70.9   1372.42    51.020   615.8479
## 5   4117.35  400.8652     273.6      70.9   1319.71    52.001   632.7023
## 7   4149.30  430.3012     273.6      70.9   1266.58    52.406   649.3683
##   actual_y
```

```
## 3   610.87
## 5   619.81
## 7   615.21
```

```
(619.81 - 595.32)/619.81
```

```
## [1] 0.03951211
```

```
ggplot(cop2, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_point(aes(y = t(actual_line_search_df2)[1:6,5], color = "Line Search"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  labs(title = "Settings for Largest Response - Linesearch 2",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("High" = "red",
                                "Low" = "red",
                                "Current" = "blue",
                                "Line Search" = "green")) +
  facet_wrap(~variables, scales = "free_y")
```



Experiment 3

Setup

High and Low Assignment

With the results of the most successful line search trial, I use that as the new current operating position and run a final experiment to test whether or not I am on the highest point of the surface. The only change I am making to my strategy is to use a 10% increase and decrease instead of 15% since I believe I am very close to the point that maximizes Y.

```
t(linesearch_results[5,c("A_uncoded",
                              "B_uncoded",
                              "C_uncoded",
                              "D_uncoded",
                              "E_uncoded",
                              "F_uncoded")])
```

```
##                    5
## A_uncoded 4117.3500
## B_uncoded  400.8652
## C_uncoded  273.6000
## D_uncoded   70.9000
```

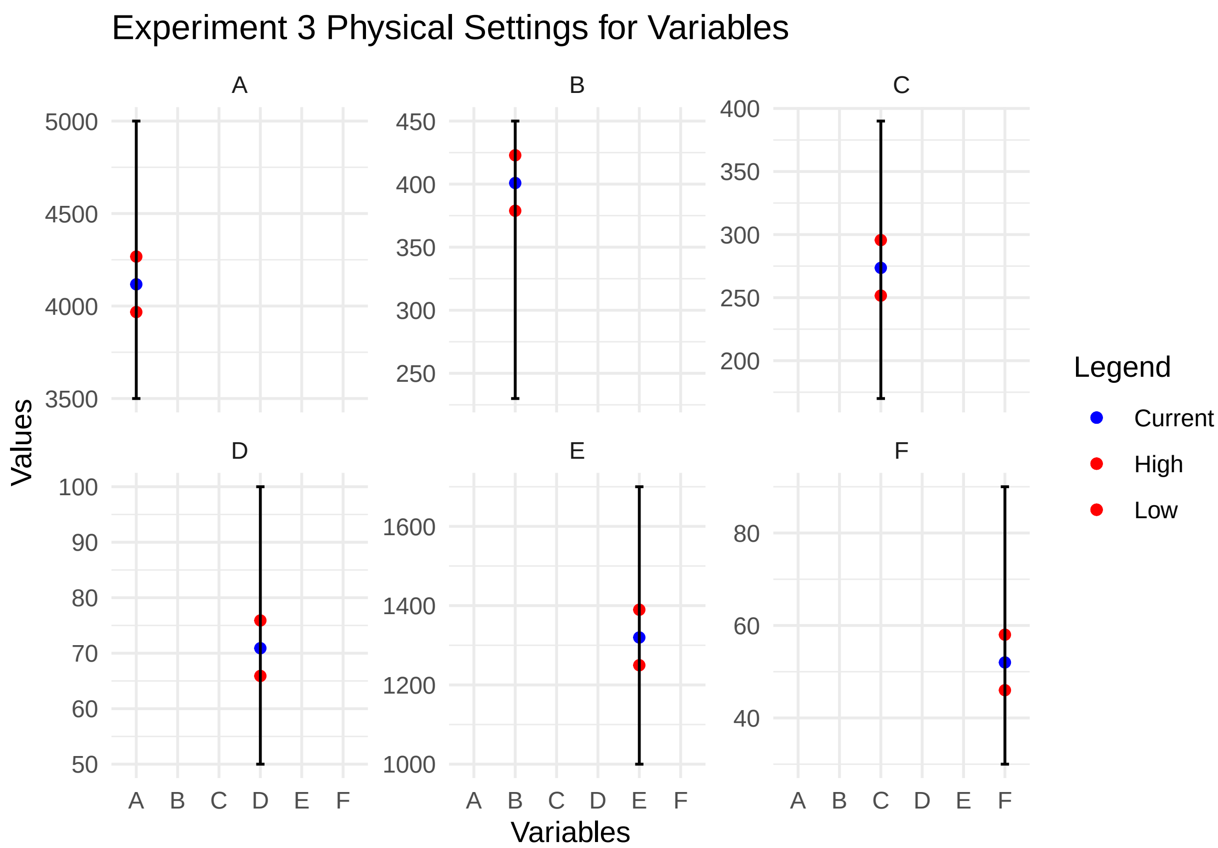
```
## E_uncoded 1319.7100
## F_uncoded   52.0010
```

```
cop3 <- data.frame(variables=c("A", "B", "C", "D", "E", "F"),
                  current=c(t(linesearch_results[5,c("A_uncoded",
                              "B_uncoded",
                              "C_uncoded",
                              "D_uncoded",
                              "E_uncoded",
                              "F_uncoded")])),
                  lower=c(3500,230,170,50,1000,30),
                  upper=c(5000,450,390,100,1700,90))

cop3$range <- cop3$upper - cop3$lower
cop3$change <- cop3$range * .10
cop3$increase <- cop3$current + cop3$change
cop3$decrease <- cop3$current - cop3$change

# applying a min and max to avoid going above or below a limit
cop3$high <- apply(cop3[, c("upper","increase")], 1, min)
cop3$low <- apply(cop3[, c("lower","decrease")], 1, max)

cop3
```

```
##   variables   current lower upper range change  increase  decrease      high
## 1         A 4117.3500  3500  5000  1500    150 4267.3500 3967.3500 4267.3500
## 2         B  400.8652   230   450   220     22  422.8652  378.8652  422.8652
## 3         C  273.6000   170   390   220     22  295.6000  251.6000  295.6000
## 4         D   70.9000    50   100    50      5   75.9000   65.9000   75.9000
## 5         E 1319.7100  1000  1700   700     70 1389.7100 1249.7100 1389.7100
## 6         F   52.0010    30    90    60      6   58.0010   46.0010   58.0010
##         low
## 1 3967.3500
## 2  378.8652
## 3  251.6000
## 4   65.9000
## 5 1249.7100
## 6   46.0010
```

```
ggplot(cop3, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  labs(title = "Experiment 3 Physical Settings for Variables",
       x = "Variables",
       y = "Values",
```

```r
      color = "Legend") +
  scale_color_manual(values = c("blue", "red", "red")) +
  facet_wrap(~variables, scales = "free_y")
```



Fractional Design

I again decide to go with an 2^{6-3}$ design to minimize the number of trials to run. I write the design to csv and continue with the test.

```r
eight_design <- FrF2(nfactors = 6,
                     nruns = 2^(6-3),
                     generators=c("AB","AC","BC"),
                     ncenter = 3,
                     randomize=FALSE)

eight_design
```

```
##    A  B  C  D  E  F
## 1 -1 -1 -1  1  1  1
## 2  1 -1 -1 -1 -1  1
## 3 -1  1 -1 -1  1 -1
## 4  1  1 -1  1 -1 -1
## 5 -1 -1  1  1 -1 -1
## 6  1 -1  1 -1  1 -1
```

```
## 7  -1  1  1 -1 -1  1
## 8   1  1  1  1  1  1
## 9   0  0  0  0  0  0
## 10  0  0  0  0  0  0
## 11  0  0  0  0  0  0
## class=design, type= FrF2.generators.center
```

```r
aliasprint(eight_design)
```

```
## $legend
## [1] A=A B=B C=C D=D E=E F=F
##
## $main
## [1] A=BD=CE B=AD=CF C=AE=BF D=AB=EF E=AC=DF F=BC=DE
##
## $fi2
## [1] AF=BE=CD
```

Design Encoding

```r
t_cop <- data.frame(t(cop3[,-1]))
colnames(t_cop) <- cop3$variables

write.csv(as.data.frame(eight_design), "design.csv", row.names = F)
deisgn <- read.csv("design.csv")

for(i in colnames(deisgn)){
  deisgn <- uncode_variable(deisgn, t_cop, i)
}

deisgn_coded <- deisgn[,1:6]
deisgn_uncoded <- deisgn[,7:12]
deisgn_uncoded
```

```
##    A_uncoded B_uncoded C_uncoded D_uncoded E_uncoded F_uncoded
## 1    3967.35  378.8652     251.6      75.9   1389.71    58.001
## 2    4267.35  378.8652     251.6      65.9   1249.71    58.001
## 3    3967.35  422.8652     251.6      65.9   1389.71    46.001
## 4    4267.35  422.8652     251.6      75.9   1249.71    46.001
## 5    3967.35  378.8652     295.6      75.9   1249.71    46.001
## 6    4267.35  378.8652     295.6      65.9   1389.71    46.001
## 7    3967.35  422.8652     295.6      65.9   1249.71    58.001
## 8    4267.35  422.8652     295.6      75.9   1389.71    58.001
## 9    4117.35  400.8652     273.6      70.9   1319.71    52.001
## 10   4117.35  400.8652     273.6      70.9   1319.71    52.001
## 11   4117.35  400.8652     273.6      70.9   1319.71    52.001
```

```r
write.csv(deisgn_uncoded, "exp_3.csv", row.names = F)
write.csv(deisgn_coded, "exp_3_coded.csv", row.names = F)
```

Results

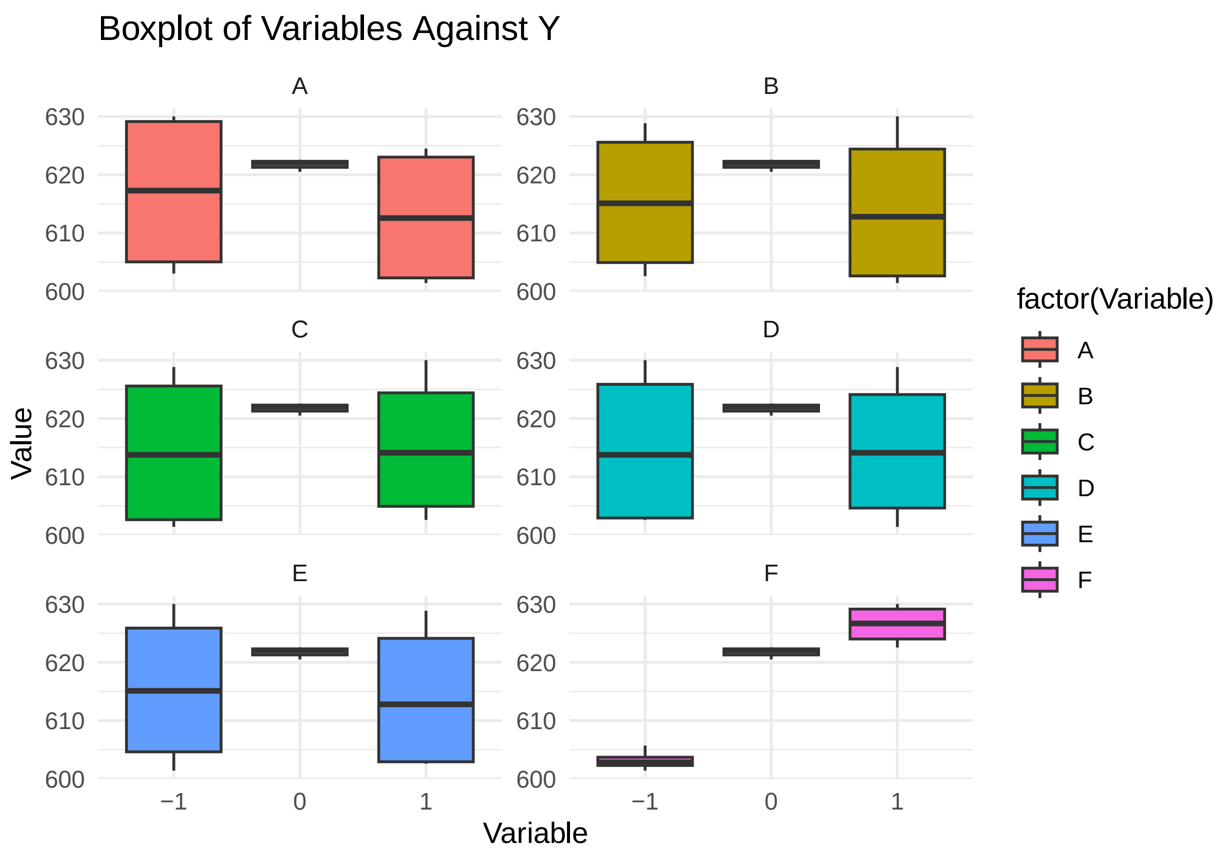
Below are the results of experiment 3 and an exploratory analysis.

```
results_coded <- read.csv("exp_3_results_coded.csv")
head(results_coded)
```

```
##    A  B  C  D  E  F      Y
## 1 -1 -1 -1  1  1  1 628.85
## 2  1 -1 -1 -1 -1  1 624.49
## 3 -1  1 -1 -1  1 -1 603.03
## 4  1  1 -1  1 -1 -1 601.40
## 5 -1 -1  1  1 -1 -1 605.70
## 6  1 -1  1 -1  1 -1 602.58
```

Exploratory Data Analysis

```
df_long <- gather(results_coded, key = "Variable", value = "Value", -Y)

ggplot(df_long, aes(x = factor(Value), y = Y)) +
  geom_boxplot(aes(fill = factor(Variable)), position = "dodge") +
  labs(title = "Boxplot of Variables Against Y",
       x = "Variable",
       y = "Value") +
  facet_wrap(~ Variable, scales = "free_y", ncol = 2) +
  theme_minimal()
```



From the box plots it is evident that F is likely to be the only variable that can be adjusted. A, B, C, D, and E all look to have better results in the current (0) setting than high or low.

Modeling

I use backwards selection again to provide a suitable second order model. In the final iteration of the backwards scheduling I see that the F and A variable are significant variable along with a marginally significant interaction

```
mod1 <- lm(Y~A+B+C+D+E+F
           +I(F^2), data=results_coded)
summary(mod1)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + C + D + E + F + I(F^2), data = results_coded)
##
## Residuals:
##       1       2       3       4       5       6       7       8       9      10
##  0.8863 -0.8863 -0.8863  0.8863 -0.8863  0.8863  0.8863 -0.8863  0.8733 -1.2067
##      11
##  0.3333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 621.7067     0.9783 635.506 8.59e-09 ***
## A            -2.0738     0.5991  -3.462 0.040593 *
## B            -0.5788     0.5991  -0.966 0.405270
## C             0.3838     0.5991   0.641 0.567373
## D            -0.2037     0.5991  -0.340 0.756197
## E            -0.5763     0.5991  -0.962 0.407059
## F            11.6487     0.5991  19.445 0.000297 ***
## I(F^2)       -6.8804     1.1471  -5.998 0.009282 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.694 on 3 degrees of freedom
## Multiple R-squared:  0.993,  Adjusted R-squared:  0.9768
## F-statistic:  61.2 on 7 and 3 DF,  p-value: 0.003117
```

Remove D

```
mod2 <- lm(Y~A+B+C+E+F
           +I(F^2), data=results_coded)
summary(mod2)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + C + E + F + I(F^2), data = results_coded)
##
## Residuals:
##       1       2       3       4       5       6       7       8       9      10
##  0.6825 -0.6825 -0.6825  0.6825 -1.0900  1.0900  1.0900 -1.0900  0.8733 -1.2067
##      11
##  0.3333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 621.7067     0.8634 720.069 2.23e-11 ***
## A            -2.0738     0.5287  -3.922  0.01722 *
## B            -0.5788     0.5287  -1.095  0.33517
## C             0.3838     0.5287   0.726  0.50815
## E            -0.5763     0.5287  -1.090  0.33702
## F            11.6487     0.5287  22.032 2.51e-05 ***
## I(F^2)       -6.8804     1.0124  -6.796  0.00245 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.495 on 4 degrees of freedom
## Multiple R-squared:  0.9928, Adjusted R-squared:  0.9819
## F-statistic: 91.65 on 6 and 4 DF,  p-value: 0.0003099
```

Remove C

```
mod3 <- lm(Y~A+B+E+F
           +I(F^2), data=results_coded)
summary(mod3)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + E + F + I(F^2), data = results_coded)
##
## Residuals:
##       1       2       3       4       5       6       7       8       9      10
##  0.2987 -1.0663 -1.0663  0.2987 -0.7062  1.4738  1.4738 -0.7062  0.8733 -1.2067
##      11
##  0.3333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 621.7067     0.8215 756.769 7.65e-14 ***
## A            -2.0738     0.5031  -4.122 0.009155 **
## B            -0.5788     0.5031  -1.150 0.302004
## E            -0.5763     0.5031  -1.145 0.303874
## F            11.6487     0.5031  23.155 2.80e-06 ***
## I(F^2)       -6.8804     0.9633  -7.142 0.000836 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.423 on 5 degrees of freedom
## Multiple R-squared:  0.9918, Adjusted R-squared:  0.9837
## F-statistic: 121.4 on 5 and 5 DF,  p-value: 3.252e-05
```

Remove E

```
mod4 <- lm(Y~A+B+F
           +I(F^2), data=results_coded)
summary(mod4)
```

```
##
## Call:
## lm.default(formula = Y ~ A + B + F + I(F^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6425 -0.8483 -0.1300  0.8742  2.0500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 621.7067     0.8426 737.827 4.18e-16 ***
## A            -2.0738     0.5160  -4.019 0.006967 **
## B            -0.5788     0.5160  -1.122 0.304897
## F            11.6487     0.5160  22.575 4.94e-07 ***
## I(F^2)       -6.8804     0.9881  -6.964 0.000436 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.459 on 6 degrees of freedom
## Multiple R-squared:  0.9897, Adjusted R-squared:  0.9828
## F-statistic: 143.9 on 4 and 6 DF,  p-value: 4.359e-06
```

Remove B

```r
mod5 <- lm(Y~A+F
           +I(F^2), data=results_coded)
summary(mod5)
```

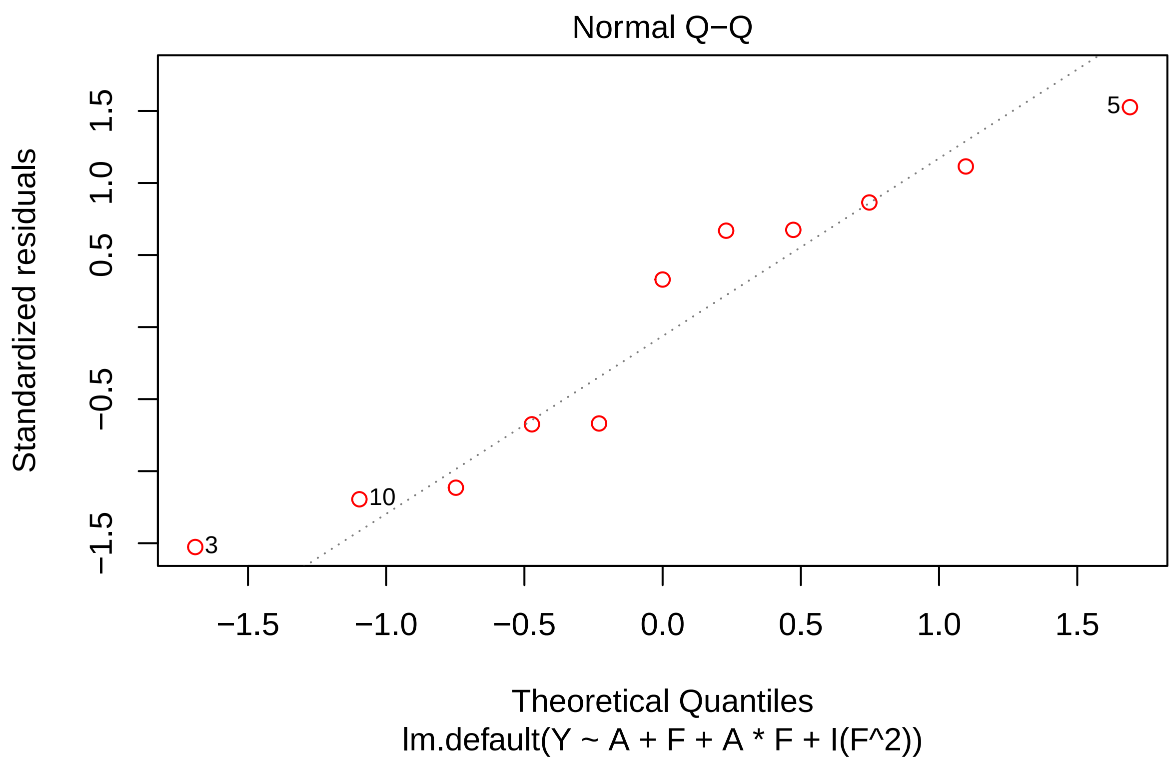
```
##
## Call:
## lm.default(formula = Y ~ A + F + I(F^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2212 -0.5590  0.3013  0.6610  1.4763
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 621.7067     0.8580 724.593  < 2e-16 ***
## A            -2.0738     0.5254  -3.947 0.005554 **
## F            11.6487     0.5254  22.170  9.6e-08 ***
## I(F^2)       -6.8804     1.0061  -6.839 0.000244 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.486 on 7 degrees of freedom
## Multiple R-squared:  0.9875, Adjusted R-squared:  0.9822
## F-statistic: 184.6 on 3 and 7 DF,  p-value: 5.032e-07
```

```r
mod6 <- lm(Y~A+F+
             A*F+
           I(F^2), data=results_coded)
summary(mod6)
```

```
##
## Call:
## lm.default(formula = Y ~ A + F + A * F + I(F^2), data = results_coded)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3350 -0.7825  0.3333  0.7317  1.3350
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 621.7067     0.7140 870.743  < 2e-16 ***
## A            -2.0738     0.4372  -4.743 0.003182 **
## F            11.6487     0.4372  26.642 1.85e-07 ***
## I(F^2)       -6.8804     0.8372  -8.218 0.000175 ***
## A:F          -0.8863     0.4372  -2.027 0.089038 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.237 on 6 degrees of freedom
## Multiple R-squared:  0.9926, Adjusted R-squared:  0.9877
## F-statistic:   201 on 4 and 6 DF,  p-value: 1.617e-06
```

The residuals for this model don't look perfect but much better than the previous experiment.

```r
plot(mod6, which=2, col=c("red"))
```



RSM

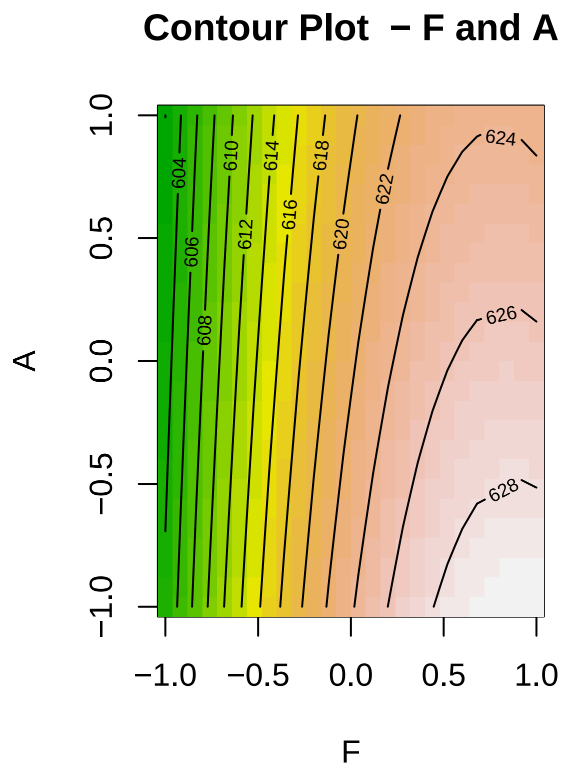
The model is acceptable and from the contour plot it appears the peak is within range and can be reached in the next steepest ascent.

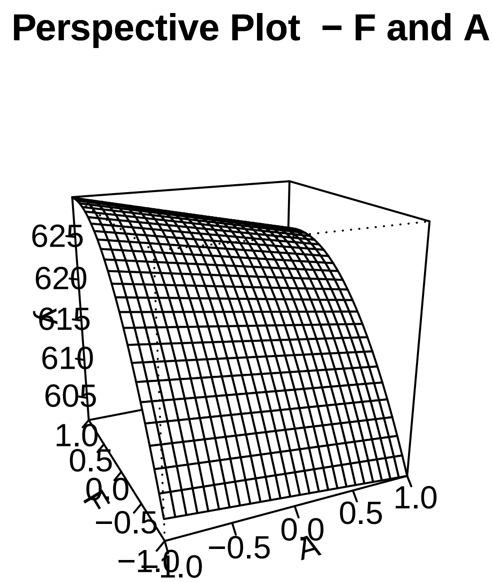
```r
rsm_mod <- rsm(Y~FO(F,A)+TWI(F,A)+PQ(F),
               data=results_coded)
summary(rsm_mod)
```

```
##
## Call:
## rsm(formula = Y ~ FO(F, A) + TWI(F, A) + PQ(F), data = results_coded)
##
##              Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 621.70667    0.71400 870.7429 < 2.2e-16 ***
## F            11.64875    0.43723  26.6421 1.846e-07 ***
## A            -2.07375    0.43723  -4.7429 0.0031819 **
## F:A          -0.88625    0.43723  -2.0270 0.0890380 .
## F^2          -6.88042    0.83723  -8.2180 0.0001752 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Multiple R-squared:  0.9926, Adjusted R-squared:  0.9877
## F-statistic:   201 on 4 and 6 DF,  p-value: 1.617e-06
##
## Analysis of Variance Table
##
## Response: Y
##             Df  Sum Sq Mean Sq  F value    Pr(>F)
## FO(F, A)     2 1119.95     560 366.1478 5.367e-07
## TWI(F, A)    1    6.28       6   4.1086 0.0890380
## PQ(F)        1  103.29     103  67.5360 0.0001752
## Residuals    6    9.18       2
## Lack of fit  0    0.00     Inf
## Pure error   6    9.18       2
##
## Stationary point of response surface:
##         F         A
## -2.339915 49.475809
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1]  0.02842153 -6.90883820
##
## $vectors
##          [,1]        [,2]
## F  0.06400734 -0.99794943
## A -0.99794943 -0.06400734
```

```
par(mfrow = c(1, 2))
contour(rsm_mod, ~ F + A, image = TRUE, main="Contour Plot  - F and A")
persp(rsm_mod, F~A, zlab = "y", main="Perspective Plot  - F and A")
```






```r
par(mfrow = c(1, 1))
```

Testing and Evaluation

Steepest Ascent

```r
line_search_df3 <- steepest(rsm_mod)
```

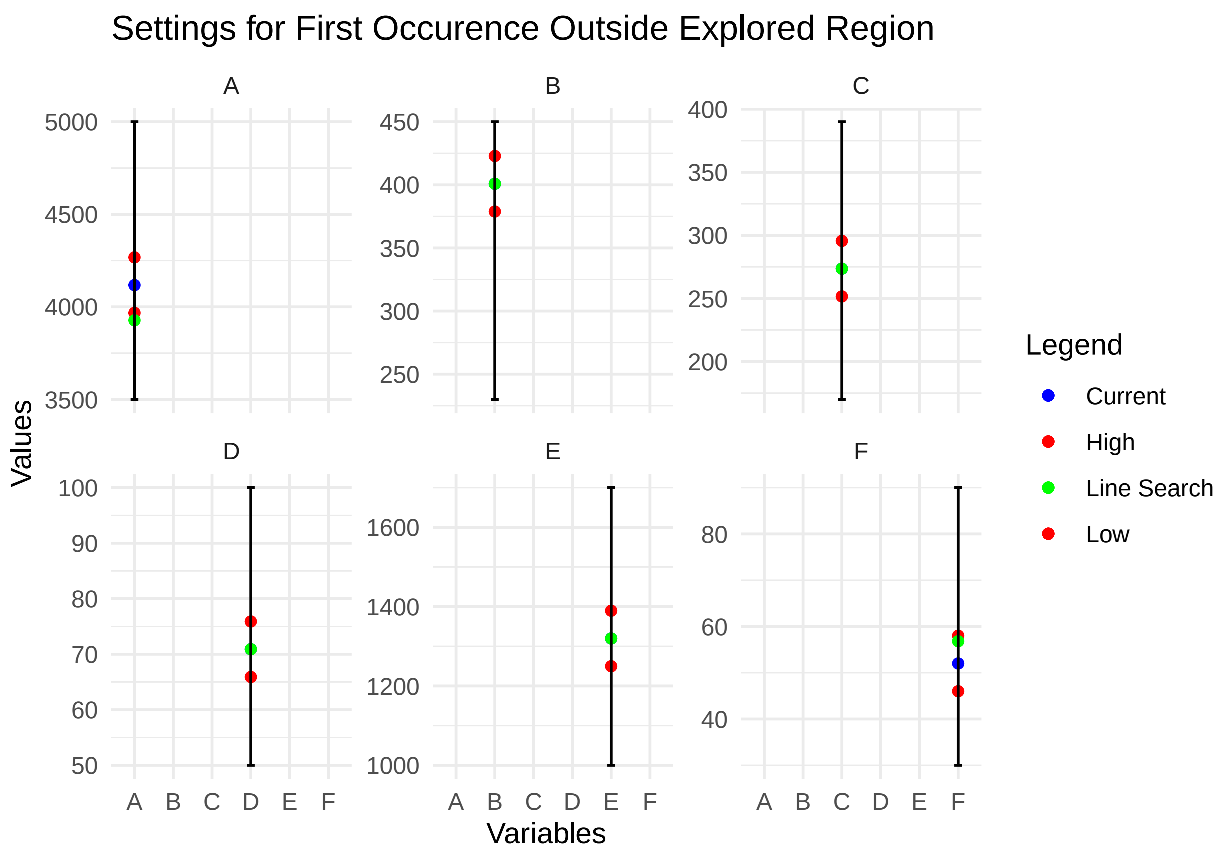
```
## Path of steepest ascent from ridge analysis:
```

```r
for(i in c("A","F")){
  line_search_df3 <- uncode_variable(line_search_df3, t_cop, i)
}
head(line_search_df3)
```

```
##   dist     F      A |    yhat A_uncoded F_uncoded
## 1  0.0 0.000  0.000 | 621.707   4117.35    52.001
## 2  0.5 0.457 -0.204 | 626.099   4086.75    54.743
## 3  1.0 0.700 -0.714 | 628.413   4010.25    56.201
## 4  1.5 0.801 -1.268 | 630.152   3927.15    56.807
## 5  2.0 0.864 -1.804 | 631.757   3846.75    57.185
## 6  2.5 0.914 -2.327 | 633.316   3768.30    57.485
```

```r
actual_line_search_df3 <- data.frame(A_uncoded=line_search_df3$A_uncoded)
actual_line_search_df3$B_uncoded <- t_cop$B[1]
actual_line_search_df3$C_uncoded <- t_cop$C[1]
actual_line_search_df3$D_uncoded <- t_cop$D[1]
actual_line_search_df3$E_uncoded <- t_cop$E[1]
actual_line_search_df3$F_uncoded <- line_search_df3$F_uncoded

# applying limits
# Apply the upper limits using a for loop
for (col_name in colnames(t_cop)) {
  actual_line_search_df3[, paste0(col_name,"_uncoded")] <- pmin(
    actual_line_search_df3[,
                           paste0(col_name,"_uncoded")],
    t_cop[3,col_name])
}

# Apply the lower limits using a for loop
for (col_name in colnames(t_cop)) {
  actual_line_search_df3[, paste0(col_name,"_uncoded")] <- pmax(
    actual_line_search_df3[,
                           paste0(col_name,"_uncoded")],
    t_cop[2,col_name])
}
```

```r
ggplot(cop3, aes(x = variables, y = current)) +
  geom_point(aes(color = "Current"),
             size = 1.5) +
  geom_point(aes(y = high,color = "High"),
             size = 1.5) +
  geom_point(aes(y = low, color = "Low"),
             size = 1.5) +
  geom_point(aes(y = t(actual_line_search_df3)[,4], color = "Line Search"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  # Add legend
  labs(title = "Settings for First Occurence Outside Explored Region",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("High" = "red",
                                "Low" = "red",
                                "Current" = "blue",
                                "Line Search" = "green")) +
  facet_wrap(~variables, scales = "free_y")
```




```
actual_line_search_df3$expected_y <- line_search_df3$yhat
write.csv(actual_line_search_df3, "linesearch3.csv", row.names = F)
```

Line Search Results 3

Again to save on resources, instead of testing each run provided by the line search that had an incremental change of .5, I performed every other trial to test a delta of 1.

After hitting a decrease in Y, I performed the iteration of the line search previous to that one and determined it to be the settings that maximized Y to 630.27 (give or take a few points to account for experimental errors).

From the previous settings, the response improved by about 2%. The response was overall increased from 508.6 to 630.98 roughly 19% improvement. The final settings can be seen below

```
linesearch_results <- read.csv("linesearch3_results.csv")
linesearch_results[c(3,4,5),]
```

```
##   A_uncoded B_uncoded C_uncoded D_uncoded E_uncoded F_uncoded expected_y
## 3   4010.25  400.8652     273.6      70.9   1319.71    56.201    628.413
## 4   3927.15  400.8652     273.6      70.9   1319.71    56.807    630.152
## 5   3846.75  400.8652     273.6      70.9   1319.71    57.185    631.757
##   actual_y
## 3   629.08
## 4   630.98
## 5   627.27
```

```r
(630.98 - 619.81)/619.81
```

```
## [1] 0.01802165
```

```r
(630.98 - 508.6)/630.98
```

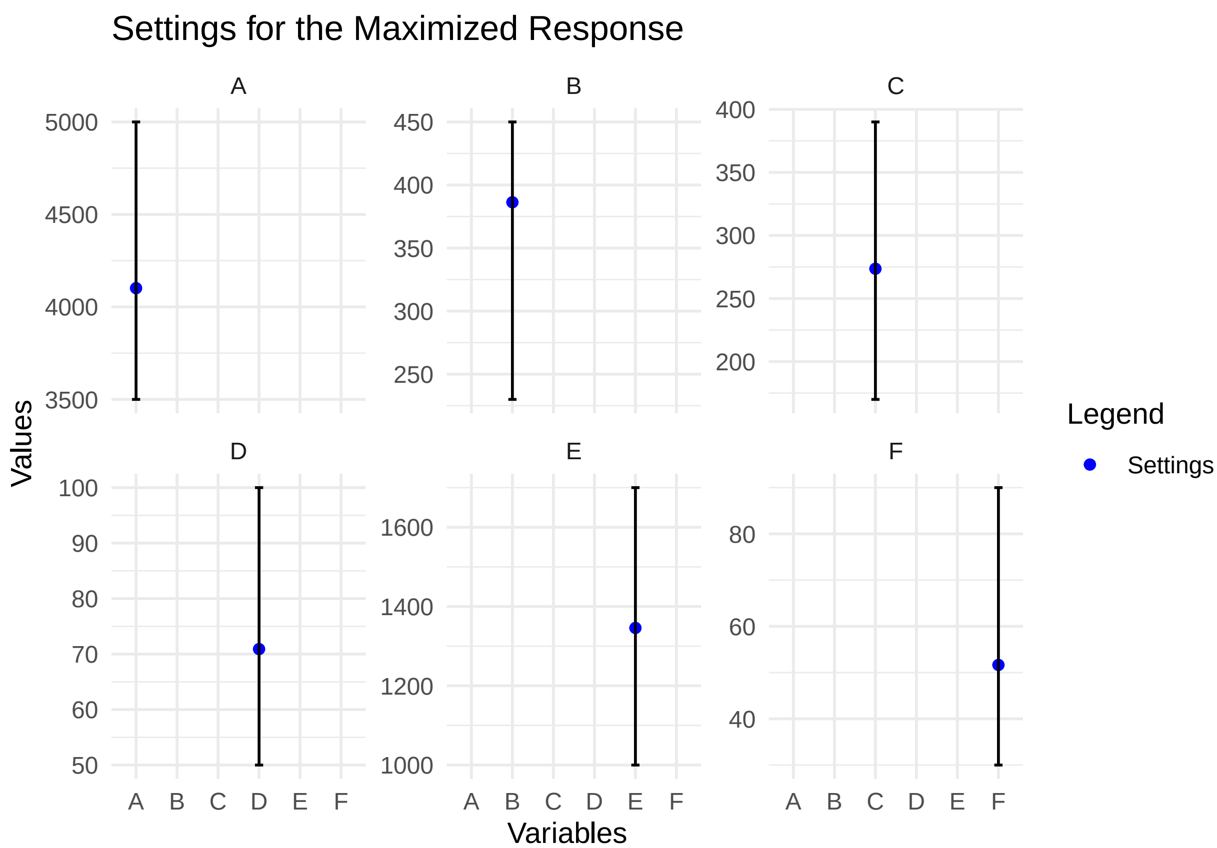
```
## [1] 0.1939523
```

Final Recommendation

```r
linesearch_results[c(4),1:8]
```

```
##   A_uncoded B_uncoded C_uncoded D_uncoded E_uncoded F_uncoded expected_y
## 4   3927.15  400.8652     273.6      70.9   1319.71    56.807    630.152
##   actual_y
## 4   630.98
```

```r
ggplot(cop3, aes(x = variables, y = current)) +
  geom_point(aes(y = t(actual_line_search_df2)[1:6,4], color = "Settings"),
             size = 1.5) +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                width = 0.2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  labs(title = "Settings for the Maximized Response",
       x = "Variables",
       y = "Values",
       color = "Legend") +
  scale_color_manual(values = c("Settings" = "blue")) +
  facet_wrap(~variables, scales = "free_y")
```



Post Experiment Reflection

- I don't think center points were needed after the initial experiment since there was curvature.
- I could have included axial points to my fractional design to be able to do second order modeling, but didnt think of it until after running the experiments.
- In experiment 2, after deciding to include factor A I should have retried the interaction terms again.

Appendix

Alias Structure for Quarter Fractional Factorial

```
diff_strings_add_if_not_present <- function(string1, vector_of_strings) {
  chars1 <- strsplit(string1, '')[[1]]

  subtract_chars <- function(string2) {
    chars2 <- strsplit(string2, '')[[1]]
    result <- setdiff(chars2, chars1)
    result <- c(result, setdiff(chars1, chars2))
    result <- sort(result)
    return(paste(result, collapse = ""))
```

```r
  }

  result_vector <- sapply(vector_of_strings, subtract_chars)

  return(result_vector)
}
create_words <- function(base_generators_words){
  all_words <- c()
  for(i in base_generators_words){
    words <- diff_strings_add_if_not_present(i,setdiff(base_generators_words,i))
    all_words <- union(all_words, words)
  }
  print(all_words)
}
generate_combinations <- function(str_vector) {
  result_list <- list()

  for (length_combinations in 1:length(str_vector)) {
    current_combinations <- combn(str_vector, length_combinations, simplify = FALSE)

    result_list <- c(result_list,
                     sapply(current_combinations,
                            function(comb) paste(comb, collapse = "")))
  }

  result_vector <- unlist(result_list)

  return(result_vector)
}
```

```r
# words
# e = abc -> I = abce
# f = bcd -> I = bcdf
create_words(c("abce","bcdf"))
```

```
## [1] "adef"
```

```r
words <- create_words(c("abce","bcdf", "adef"))
```

```
## [1] "adef" "bcdf" "abce"
```

```r
factors <- c("a", "b", "c", "d", "e", "f")
combinations <- generate_combinations(factors)

alias_list <- list()
for(i in combinations){
  alias_list[[i]] <- diff_strings_add_if_not_present(i, words)
}

alias_structure <- t(data.frame(alias_list))
alias_structure
```

```
##         adef     bcdf     abce
## a       "def"    "abcdf"  "bce"
## b       "abdef"  "cdf"    "ace"
## c       "acdef"  "bdf"    "abe"
## d       "aef"    "bcf"    "abcde"
## e       "adf"    "bcdef"  "abc"
## f       "ade"    "bcd"    "abcef"
## ab      "bdef"   "acdf"   "ce"
## ac      "cdef"   "abdf"   "be"
## ad      "ef"     "abcf"   "bcde"
## ae      "df"     "abcdef" "bc"
## af      "de"     "abcd"   "bcef"
## bc      "abcdef" "df"     "ae"
## bd      "abef"   "cf"     "acde"
## be      "abdf"   "cdef"   "ac"
## bf      "abde"   "cd"     "acef"
## cd      "acef"   "bf"     "abde"
## ce      "acdf"   "bdef"   "ab"
## cf      "acde"   "bd"     "abef"
## de      "af"     "bcef"   "abcd"
## df      "ae"     "bc"     "abcdef"
## ef      "ad"     "bcde"   "abcf"
## abc     "bcdef"  "adf"    "e"
## abd     "bef"    "acf"    "cde"
## abe     "bdf"    "acdef"  "c"
## abf     "bde"    "acd"    "cef"
## acd     "cef"    "abf"    "bde"
## ace     "cdf"    "abdef"  "b"
## acf     "cde"    "abd"    "bef"
## ade     "f"      "abcef"  "bcd"
## adf     "e"      "abc"    "bcdef"
## aef     "d"      "abcde"  "bcf"
## bcd     "abcef"  "f"      "ade"
## bce     "abcdf"  "def"    "a"
## bcf     "abcde"  "d"      "aef"
## bde     "abf"    "cef"    "acd"
## bdf     "abe"    "c"      "acdef"
## bef     "abd"    "cde"    "acf"
## cde     "acf"    "bef"    "abd"
## cdf     "ace"    "b"      "abdef"
## cef     "acd"    "bde"    "abf"
## def     "a"      "bce"    "abcdf"
## abcd    "bcef"   "af"     "de"
## abce    "bcdf"   "adef"   ""
## abcf    "bcde"   "ad"     "ef"
## abde    "bf"     "acef"   "cd"
## abdf    "be"     "ac"     "cdef"
## abef    "bd"     "acde"   "cf"
## acde    "cf"     "abef"   "bd"
## acdf    "ce"     "ab"     "bdef"
## acef    "cd"     "abde"   "bf"
## adef    ""       "abce"   "bcdf"
## bcde    "abcf"   "ef"     "ad"
## bcdf    "abce"   ""       "adef"
```

```
## bcef   "abcd"   "de"     "af"
## bdef   "ab"     "ce"     "acdf"
## cdef   "ac"     "be"     "abdf"
## abcde  "bcf"    "aef"    "d"
## abcdf  "bce"    "a"      "def"
## abcef  "bcd"    "ade"    "f"
## abdef  "b"      "ace"    "cdf"
## acdef  "c"      "abe"    "bdf"
## bcdef  "abc"    "e"      "adf"
## abcdef "bc"     "ae"     "df"
```

Alias Structure for Eight Fractional Factorial

```r
# words
# d = ab -> I = abd
# e = ac -> I = ace
# f = bc -> I = fbc
create_words(c("abd","ace", "fbc"))
```

```
## [1] "bcde" "acdf" "abef"
```

```r
words <- create_words(c("abd","ace", "fbc","bcde", "acdf", "abef"))
```

```
## [1] "bcde" "acdf" "ace"  "bcf"  "def"  "abef" "abd"
```

```r
factors <- c("a", "b", "c", "d", "e", "f")
combinations <- generate_combinations(factors)

alias_list <- list()
for(i in combinations){
  alias_list[[i]] <- diff_strings_add_if_not_present(i, words)
}

alias_structure <- t(data.frame(alias_list))
alias_structure
```

```
##         bcde     acdf     ace      bcf      def      abef     abd
## a       "abcde"  "cdf"    "ce"     "abcf"   "adef"   "bef"    "bd"
## b       "cde"    "abcdf"  "abce"   "cf"     "bdef"   "aef"    "ad"
## c       "bde"    "adf"    "ae"     "bf"     "cdef"   "abcef"  "abcd"
## d       "bce"    "acf"    "acde"   "bcdf"   "ef"     "abdef"  "ab"
## e       "bcd"    "acdef"  "ac"     "bcef"   "df"     "abf"    "abde"
## f       "bcdef"  "acd"    "acef"   "bc"     "de"     "abe"    "abdf"
## ab      "acde"   "bcdf"   "bce"    "acf"    "abdef"  "ef"     "d"
## ac      "abde"   "df"     "e"      "abf"    "acdef"  "bcef"   "bcd"
## ad      "abce"   "cf"     "cde"    "abcdf"  "aef"    "bdef"   "b"
## ae      "abcd"   "cdef"   "c"      "abcef"  "adf"    "bf"     "bde"
## af      "abcdef" "cd"     "cef"    "abc"    "ade"    "be"     "bdf"
## bc      "de"     "abdf"   "abe"    "f"      "bcdef"  "acef"   "acd"
## bd      "ce"     "abcf"   "abcde"  "cdf"    "bef"    "adef"   "a"
```

```
## be     "cd"     "abcdef" "abc"    "cef"    "bdf"    "af"     "ade"
## bf     "cdef"   "abcd"   "abcef"  "c"      "bde"    "ae"     "adf"
## cd     "be"     "af"     "ade"    "bdf"    "cef"    "abcdef" "abc"
## ce     "bd"     "adef"   "a"      "bef"    "cdf"    "abcf"   "abcde"
## cf     "bdef"   "ad"     "aef"    "b"      "cde"    "abce"   "abcdf"
## de     "bc"     "acef"   "acd"    "bcdef"  "f"      "abdf"   "abe"
## df     "bcef"   "ac"     "acdef"  "bcd"    "e"      "abde"   "abf"
## ef     "bcdf"   "acde"   "acf"    "bce"    "d"      "ab"     "abdef"
## abc    "ade"    "bdf"    "be"     "af"     "abcdef" "cef"    "cd"
## abd    "ace"    "bcf"    "bcde"   "acdf"   "abef"   "def"    ""
## abe    "acd"    "bcdef"  "bc"     "acef"   "abdf"   "f"      "de"
## abf    "acdef"  "bcd"    "bcef"   "ac"     "abde"   "e"      "df"
## acd    "abe"    "f"      "de"     "abdf"   "acef"   "bcdef"  "bc"
## ace    "abd"    "def"    ""       "abef"   "acdf"   "bcf"    "bcde"
## acf    "abdef"  "d"      "ef"     "ab"     "acde"   "bce"    "bcdf"
## ade    "abc"    "cef"    "cd"     "abcdef" "af"     "bdf"    "be"
## adf    "abcef"  "c"      "cdef"   "abcd"   "ae"     "bde"    "bf"
## aef    "abcdf"  "cde"    "cf"     "abce"   "ad"     "b"      "bdef"
## bcd    "e"      "abf"    "abde"   "df"     "bcef"   "acdef"  "ac"
## bce    "d"      "abdef"  "ab"     "ef"     "bcdf"   "acf"    "acde"
## bcf    "def"    "abd"    "abef"   ""       "bcde"   "ace"    "acdf"
## bde    "c"      "abcef"  "abcd"   "cdef"   "bf"     "adf"    "ae"
## bdf    "cef"    "abc"    "abcdef" "cd"     "be"     "ade"    "af"
## bef    "cdf"    "abcde"  "abcf"   "ce"     "bd"     "a"      "adef"
## cde    "b"      "aef"    "ad"     "bdef"   "cf"     "abcdf"  "abce"
## cdf    "bef"    "a"      "adef"   "bd"     "ce"     "abcde"  "abcf"
## cef    "bdf"    "ade"    "af"     "be"     "cd"     "abc"    "abcdef"
## def    "bcf"    "ace"    "acdf"   "bcde"   ""       "abd"    "abef"
## abcd   "ae"     "bf"     "bde"    "adf"    "abcef"  "cdef"   "c"
## abce   "ad"     "bdef"   "b"      "aef"    "abcdf"  "cf"     "cde"
## abcf   "adef"   "bd"     "bef"    "a"      "abcde"  "ce"     "cdf"
## abde   "ac"     "bcef"   "bcd"    "acdef"  "abf"    "df"     "e"
## abdf   "acef"   "bc"     "bcdef"  "acd"    "abe"    "de"     "f"
## abef   "acdf"   "bcde"   "bcf"    "ace"    "abd"    ""       "def"
## acde   "ab"     "ef"     "d"      "abdef"  "acf"    "bcdf"   "bce"
## acdf   "abef"   ""       "def"    "abd"    "ace"    "bcde"   "bcf"
## acef   "abdf"   "de"     "f"      "abe"    "acd"    "bc"     "bcdef"
## adef   "abcf"   "ce"     "cdf"    "abcde"  "a"      "bd"     "bef"
## bcde   ""       "abef"   "abd"    "def"    "bcf"    "acdf"   "ace"
## bcdf   "ef"     "ab"     "abdef"  "d"      "bce"    "acde"   "acf"
## bcef   "df"     "abde"   "abf"    "e"      "bcd"    "ac"     "acdef"
## bdef   "cf"     "abce"   "abcdf"  "cde"    "b"      "ad"     "aef"
## cdef   "bf"     "ae"     "adf"    "bde"    "c"      "abcd"   "abcef"
## abcde  "a"      "bef"    "bd"     "adef"   "abcf"   "cdf"    "ce"
## abcdf  "aef"    "b"      "bdef"   "ad"     "abce"   "cde"    "cf"
## abcef  "adf"    "bde"    "bf"     "ae"     "abcd"   "c"      "cdef"
## abdef  "acf"    "bce"    "bcdf"   "acde"   "ab"     "d"      "ef"
## acdef  "abf"    "e"      "df"     "abde"   "ac"     "bcd"    "bcef"
## bcdef  "f"      "abe"    "abdf"   "de"     "bc"     "acd"    "acef"
## abcdef "af"     "be"     "bdf"    "ade"    "abc"    "cd"     "cef"
```

Smaller Exponent Attempt

```
results_coded <- read.csv("exp_2_results_coded.csv")
mod8 <- lm(log(Y)~B+
             E+
             F+
             I(F^1.5), data=results_coded)
summary(mod8)
```

```
##
## Call:
## lm.default(formula = log(Y) ~ B + E + F + I(F^1.5), data = results_coded)
##
## Residuals:
##          1          2          7          8          9         10         11
## -0.0001744  0.0001744 -0.0001744  0.0001744 -0.0020597 -0.0002614  0.0023211
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  6.3897300  0.0010446 6117.080 9.63e-12 ***
## B            0.0208272  0.0009046   23.023 0.000179 ***
## E           -0.0113681  0.0009046  -12.567 0.001086 **
## F            0.0104495  0.0013818    7.562 0.004796 **
## I(F^1.5)            NA         NA       NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.001809 on 3 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.996,  Adjusted R-squared:  0.992
## F-statistic: 248.4 on 3 and 3 DF,  p-value: 0.0004305
```

Simulation Output in Order

```
print(readLines("experiment_0.txt"))
```

```
##  [1] "Student ID# 9275"
##  [2] ""
##  [3] " Your objective is to MAXIMIZE the response."
##  [4] ""
##  [5] " Region of Operability    Current Operating Point"
##  [6] "   3500 < A < 5000              A = 4341.0"
##  [7] "    230 < B <  450              B =  255.3"
##  [8] "    170 < C <  390              C =  273.6"
##  [9] "     50 < D <  100              D =   70.9"
## [10] "   1000 < E < 1700              E = 1540.0"
## [11] "     30 < F <   90              F =   41.3"
```

```r
print(readLines("exp_results_1.txt"))
```

```
##  [1] "Student ID# 9275"
##  [2] ""
##  [3] " RUN     A       B       C       D       E       F       Y"
##  [4] "   1 4116.00  230.00  240.60   63.40 1435.00   32.30  463.04"
##  [5] "   2 4566.00  230.00  240.60   63.40 1645.00   32.30  419.63"
##  [6] "   3 4116.00  288.30  240.60   63.40 1645.00   50.30  544.54"
##  [7] "   4 4566.00  288.30  240.60   63.40 1435.00   50.30  545.25"
##  [8] "   5 4116.00  230.00  306.60   63.40 1645.00   50.30  479.03"
##  [9] "   6 4566.00  230.00  306.60   63.40 1435.00   50.30  487.41"
## [10] "   7 4116.00  288.30  306.60   63.40 1435.00   32.30  522.66"
## [11] "   8 4566.00  288.30  306.60   63.40 1645.00   32.30  479.79"
## [12] "   9 4116.00  230.00  240.60   78.40 1435.00   50.30  511.00"
## [13] "  10 4566.00  230.00  240.60   78.40 1645.00   50.30  459.02"
## [14] "  11 4116.00  288.30  240.60   78.40 1645.00   32.30  493.73"
## [15] "  12 4566.00  288.30  240.60   78.40 1435.00   32.30  500.17"
## [16] "  13 4116.00  230.00  306.60   78.40 1645.00   32.30  432.50"
## [17] "  14 4566.00  230.00  306.60   78.40 1435.00   32.30  443.13"
## [18] "  15 4116.00  288.30  306.60   78.40 1435.00   50.30  568.72"
## [19] "  16 4566.00  288.30  306.60   78.40 1645.00   50.30  523.59"
## [20] "  17 4341.00  259.15  273.60   70.90 1540.00   41.30  508.60"
## [21] "  18 4341.00  259.15  273.60   70.90 1540.00   41.30  507.71"
## [22] "  19 4341.00  259.15  273.60   70.90 1540.00   41.30  503.38"
```

```r
print(readLines("linesearch0.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN     A       B       C       D       E       F       Y"
## [4] "   1 4341.00  259.15  273.60   70.90 1540.00   41.30  503.42"
```

```r
print(readLines("linesearch1.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN     A       B       C       D       E       F       Y"
## [4] "   1 4310.40  270.66  273.60   70.90 1522.36   43.26  522.42"
```

```r
print(readLines("linesearch2.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN     A       B       C       D       E       F       Y"
## [4] "   1 4274.85  283.02  273.60   70.90 1503.88   44.48  544.80"
```

```r
print(readLines("linesearch3.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN     A       B       C       D       E       F       Y"
## [4] "   1 4235.92  295.65  273.60   70.90 1485.71   45.29  558.98"
```

```r
print(readLines("linesearch4.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4194.30   308.36   273.60   70.90 1467.97   45.86   571.22"
```

```r
print(readLines("linesearch5.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4150.88   321.04   273.60   70.90 1450.85   46.29   581.39"
```

```r
print(readLines("linesearch6.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4105.20   333.66   273.60   70.90 1434.16   46.65   590.76"
```

```r
print(readLines("linesearch7.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4057.95   346.25   273.60   70.90 1418.09   46.94   595.32"
```

```r
print(readLines("exp_results_2.txt"))
```

```
##  [1] "Student ID# 9275"
##  [2] ""
##  [3] " RUN      A        B        C        D        E        F        Y"
##  [4] "    1 3832.95   313.25   240.60   78.40 1523.09   55.94   582.78"
##  [5] "    2 4282.95   313.25   240.60   63.40 1313.09   55.94   596.39"
##  [6] "    3 3832.95   379.25   240.60   63.40 1523.09   37.94   562.45"
##  [7] "    4 4282.95   379.25   240.60   78.40 1313.09   37.94   585.24"
##  [8] "    5 3832.95   313.25   306.60   78.40 1313.09   37.94   552.08"
##  [9] "    6 4282.95   313.25   306.60   63.40 1523.09   37.94   545.03"
## [10] "    7 3832.95   379.25   306.60   63.40 1313.09   55.94   621.54"
## [11] "    8 4282.95   379.25   306.60   78.40 1523.09   55.94   607.78"
## [12] "    9 4057.95   346.25   273.60   70.90 1418.09   46.94   594.47"
## [13] "   10 4057.95   346.25   273.60   70.90 1418.09   46.94   595.54"
## [14] "   11 4057.95   346.25   273.60   70.90 1418.09   46.94   597.08"
```

```r
print(readLines("linesearch2_0.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4085.40   371.59   273.60   70.90 1372.42   51.02   610.87"
```

```r
print(readLines("linesearch2_1.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4117.35   400.86   273.60    70.90 1319.71    52.00   619.81"
```

```r
print(readLines("linesearch2_2.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4149.30   430.30   273.60    70.90 1266.58    52.41   615.21"
```

```r
print(readLines("exp_results_3.txt"))
```

```
##  [1] "Student ID# 9275"
##  [2] ""
##  [3] " RUN      A        B        C        D        E        F        Y"
##  [4] "    1 3967.35   378.87   251.60    75.90 1389.71    58.00   628.85"
##  [5] "    2 4267.35   378.87   251.60    65.90 1249.71    58.00   624.49"
##  [6] "    3 3967.35   422.87   251.60    65.90 1389.71    46.00   603.03"
##  [7] "    4 4267.35   422.87   251.60    75.90 1249.71    46.00   601.40"
##  [8] "    5 3967.35   378.87   295.60    75.90 1249.71    46.00   605.70"
##  [9] "    6 4267.35   378.87   295.60    65.90 1389.71    46.00   602.58"
## [10] "    7 3967.35   422.87   295.60    65.90 1249.71    58.00   630.02"
## [11] "    8 4267.35   422.87   295.60    75.90 1389.71    58.00   622.54"
## [12] "    9 4117.35   400.87   273.60    70.90 1319.71    52.00   622.58"
## [13] "   10 4117.35   400.87   273.60    70.90 1319.71    52.00   620.50"
## [14] "   11 4117.35   400.87   273.60    70.90 1319.71    52.00   622.04"
```

```r
print(readLines("linesearch3_0.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 4010.25   400.86   273.60    70.90 1319.71    56.20   629.08"
```

```r
print(readLines("linesearch3_1.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 3846.75   400.87   273.60    70.90 1319.71    57.19   627.27"
```

```r
print(readLines("linesearch3_2.txt"))
```

```
## [1] "Student ID# 9275"
## [2] ""
## [3] " RUN      A        B        C        D        E        F        Y"
## [4] "    1 3927.15   400.87   273.60    70.90 1319.71    56.81   630.98"
```