RSM tutorial

2023-01-18

Introduction

- This RMarkdown document is adapted from the package author's publication (https://cran.r-project.org/web/packages/rsm/vignettes/rsm.pdf) on `rsm`
- `rsm` has built in functions for the generation of central composite and Box-Behnken designs
- It provides functionality for the analysis for experimental data, estimation of response surface, testing the model's lack of fit, data visualization of contour plots of the fitted surface, and more
- `rsm` covers standard first order (linear) and second order (quadratic) designs for one response variable
- Functions provided by package:
    - functions/datatypes for coding and decoding factor levels
    - Functions to generate central composite and Box-Behnken designs (CCD and BBD, respectively)
    - Expands on base-R's `lm` function to be compatible with standard response-surface models
    - Functions for data visualization
    - Provides guidance for further experimentation (i.e. path of steepest ascent)

Coding of data

Install package and load library:

```
install.packages("rsm", repos = "https://cran.us.r-project.org") #install package
```

```
library(rsm)
```

Load and view sample dataframe:

```
ChemReact1
```

```
##   Time Temp Yield
## 1   80  170  80.5
## 2   80  180  81.5
## 3   90  170  82.0
## 4   90  180  83.5
## 5   85  175  83.9
## 6   85  175  84.3
## 7   85  175  84.0
```

- This dataframe has easily understandable parameters
    - Time and Temperature are continuous independent variables and Yield is a continuous dependent variable
    - We will use `coded.data` to code them as x1 and x2, respectively:

```
CR1 <- coded.data(ChemReact1, x1 ~ (Time - 85)/5, #name of the coded variable ~ linear expres
sion fir the uncoded variable
                              x2 ~ (Temp - 175)/5)
CR1
```

```
##   Time Temp Yield
## 1   80  170  80.5
## 2   80  180  81.5
## 3   90  170  82.0
## 4   90  180  83.5
## 5   85  175  83.9
## 6   85  175  84.3
## 7   85  175  84.0
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ (Time - 85)/5
## x2 ~ (Temp - 175)/5
```

- The dataframe looks very similar to how they were presented in the original, ChemReact1 dataframe, but see they are internally Time and Temp are internally understood by R as x1 and x2. This can be witnessed if CR1 is coerced into a standard dataframe:

```
as.data.frame(CR1)
```

```
##   x1 x2 Yield
## 1 -1 -1  80.5
## 2 -1  1  81.5
## 3  1 -1  82.0
## 4  1  1  83.5
## 5  0  0  83.9
## 6  0  0  84.3
## 7  0  0  84.0
```

- This is important so that the dataframe is compatible with downstream `rsm` package functions

Generating a design

- Function for creating a Box-Behnken design - `bbd`
- Function for creating a central composite design - `ccd`

Box-Behnken design

Create a 3-factor Box-Behnken design with two center points:

```
bbd(3, n0 = 2, coding =list(x1 ~ (Force - 20)/3, x2 ~ (Rate - 50)/10, x3 ~ Polish - 4))
```

```
##    run.order std.order Force Rate Polish
## 1          1        14    20   50      4
## 2          2         5    17   50      3
## 3          3         6    23   50      3
## 4          4         7    17   50      5
## 5          5        13    20   50      4
## 6          6         9    20   40      3
## 7          7         8    23   50      5
## 8          8         4    23   60      4
## 9          9         2    23   40      4
## 10        10         1    17   40      4
## 11        11        10    20   60      3
## 12        12        12    20   60      5
## 13        13         3    17   60      4
## 14        14        11    20   40      5
## 
## Data are stored in coded form using these coding formulas ...
## x1 ~ (Force - 20)/3
## x2 ~ (Rate - 50)/10
## x3 ~ Polish - 4
```

- This example of an experiment has 3 variables (x1, x2, and x3). The experiment is randomized.
- If there were a 4th or 5th variable, then the design would be blocked (default setting) and the blocks would be individually randomized

Central composite design

- CCD is a popular design because it allows the experimenter to iterizatively improve a system through optimization experiments
- One could first experiment with a single block for a first-order model and then add more block(s) if necessray to perform a second-order model fit
- There are two types of CCD blocks
    - Cube block - has design points from a two-level factorial or fractional factorial design plus center points
        - best predictions are found within the confines of these points
        - Used to estimate main effects and two factor interactions
    - Star block - contains axis points and center points
        - Used to estimate quadratic effects

Fitting a response-surface model

- The `rsm` function must make use of the following arguments:
    - `FO()` - first order
    - `TWI()` - two-way intereaction
    - `PQ()` - pure quadratic
    - `SO()` - second order

Fit first order response-surface model to the data in the first block:

```
CR1.rsm <- rsm(Yield ~ FO(x1, x2), data = CR1)
summary(CR1.rsm)
```

```
##
## Call:
## rsm(formula = Yield ~ FO(x1, x2), data = CR1)
##
##             Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 82.81429    0.54719 151.3456 1.143e-08 ***
## x1           0.87500    0.72386   1.2088    0.2933
## x2           0.62500    0.72386   0.8634    0.4366
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.3555, Adjusted R-squared:  0.0333
## F-statistic: 1.103 on 2 and 4 DF,  p-value: 0.4153
##
## Analysis of Variance Table
##
## Response: Yield
##             Df Sum Sq Mean Sq F value  Pr(>F)
## FO(x1, x2)   2 4.6250  2.3125  1.1033 0.41534
## Residuals    4 8.3836  2.0959
## Lack of fit  2 8.2969  4.1485 95.7335 0.01034
## Pure error   2 0.0867  0.0433
##
## Direction of steepest ascent (at radius 1):
##        x1        x2
## 0.8137335 0.5812382
##
## Corresponding increment in original units:
##     Time     Temp
## 4.068667 2.906191
```

- The summary of this model indicates that there is a significant lack of fit ($p = 0.01034$)
- This suggests that it may be a good idea to test a higher order model

Model with two-way interactions:

```
CR1.rsmi <- update(CR1.rsm, . ~ . + TWI(x1, x2))
summary(CR1.rsmi)
```

```
##
## Call:
## rsm(formula = Yield ~ FO(x1, x2) + TWI(x1, x2), data = CR1)
##
##             Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 82.81429    0.62948 131.5604 9.683e-07 ***
## x1           0.87500    0.83272   1.0508    0.3705
## x2           0.62500    0.83272   0.7506    0.5074
## x1:x2        0.12500    0.83272   0.1501    0.8902
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.3603, Adjusted R-squared:  -0.2793
## F-statistic: 0.5633 on 3 and 3 DF,  p-value: 0.6755
##
## Analysis of Variance Table
##
## Response: Yield
##             Df Sum Sq Mean Sq  F value   Pr(>F)
## FO(x1, x2)   2 4.6250  2.3125   0.8337 0.515302
## TWI(x1, x2)  1 0.0625  0.0625   0.0225 0.890202
## Residuals    3 8.3211  2.7737
## Lack of fit  1 8.2344  8.2344 190.0247 0.005221
## Pure error   2 0.0867  0.0433
##
## Stationary point of response surface:
## x1 x2
## -5 -7
##
## Stationary point in original units:
## Time Temp
##   60  140
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1]  0.0625 -0.0625
##
## $vectors
##         [,1]       [,2]
## x1 0.7071068 -0.7071068
## x2 0.7071068  0.7071068
```

The p-value for the lack of fit test is still very small (p = 0.005221). To investigate further, more data is needed so we can combine two blocks of the greater ChemReact experiment (ChemReact1 + ChemReact2):

```
( CR2 <- djoin(CR1, ChemReact2) )
```

```
##     Time   Temp Yield Block
## 1  80.00 170.00  80.5     1
## 2  80.00 180.00  81.5     1
## 3  90.00 170.00  82.0     1
## 4  90.00 180.00  83.5     1
## 5  85.00 175.00  83.9     1
## 6  85.00 175.00  84.3     1
## 7  85.00 175.00  84.0     1
## 8  85.00 175.00  79.7     2
## 9  85.00 175.00  79.8     2
## 10 85.00 175.00  79.5     2
## 11 92.07 175.00  78.4     2
## 12 77.93 175.00  75.6     2
## 13 85.00 182.07  78.5     2
## 14 85.00 167.93  77.0     2
##
## Data are stored in coded form using these coding formulas ...
## x1 ~ (Time - 85)/5
## x2 ~ (Temp - 175)/5
```

The larger dataset allows us to fit a full second-order model to the data. This can be accomplished by using the following code:

```
CR2.rsm <- rsm(Yield ~ Block + SO(x1, x2), data = CR2)
summary(CR2.rsm)
```

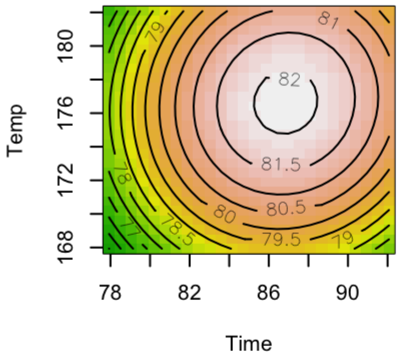
```
##
## Call:
## rsm(formula = Yield ~ Block + SO(x1, x2), data = CR2)
##
##              Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 84.095427   0.079631 1056.067 < 2.2e-16 ***
## Block2      -4.457530   0.087226  -51.103 2.877e-10 ***
## x1           0.932541   0.057699   16.162 8.444e-07 ***
## x2           0.577712   0.057699   10.012 2.122e-05 ***
## x1:x2        0.125000   0.081592    1.532    0.1694
## x1^2        -1.308555   0.060064  -21.786 1.083e-07 ***
## x2^2        -0.933442   0.060064  -15.541 1.104e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.9981, Adjusted R-squared:  0.9964
## F-statistic: 607.2 on 6 and 7 DF,  p-value: 3.811e-09
##
## Analysis of Variance Table
##
## Response: Yield
##             Df Sum Sq Mean Sq   F value    Pr(>F)
## Block        1 69.531  69.531 2611.0950 2.879e-10
## FO(x1, x2)   2  9.626   4.813  180.7341 9.450e-07
## TWI(x1, x2)  1  0.063   0.063    2.3470    0.1694
## PQ(x1, x2)   2 17.791   8.896  334.0539 1.135e-07
## Residuals    7  0.186   0.027
## Lack of fit  3  0.053   0.018    0.5307    0.6851
## Pure error   4  0.133   0.033
##
## Stationary point of response surface:
##        x1        x2
## 0.3722954 0.3343802
##
## Stationary point in original units:
##      Time      Temp
##  86.86148 176.67190
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1] -0.9233027 -1.3186949
##
## $vectors
##          [,1]       [,2]
## x1 -0.1601375 -0.9870947
## x2 -0.9870947  0.1601375
```

- In this model, the lack of fit is not significant (p = 0.6851)
- The stationary point values are like the coordinates for the center of the plot
- In the Eigenanalysis, both eigenvalues are negative
    - This is indicative that the stationary point is a maximum
    - This is an ideal situation in which the optimal conditions for this system are in close range of the stationary point
    - The next step would be to collect more data around this estimated optimum point when

- Time = 86.86148 min
- Temp = 176.67190 C

Visualization with a contour plot:

```
par(mfrow = c(2,3))
contour(CR2.rsm, ~ x1 + x2, image = TRUE, at = summary(CR2.rsm)$canonical$xs)
```



Another dataset example is a paper-helicopter experiment in which different variables such as wing area (A), wing shape (R), body width (W), and body length (L) influence how long a paper helicopter can fly. Each observation (row) represents the results of 10 replicated flights at each experimental condition.

This model studies average flight time with the variable name ‘ave’ using a second-order surface:

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

```
##
## Call:
## rsm(formula = ave ~ block + SO(x1, x2, x3, x4), data = heli)
##
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 372.800000   1.506375 247.4815 < 2.2e-16 ***
## block2       -2.950000   1.207787  -2.4425 0.0284522 *
## x1           -0.083333   0.636560  -0.1309 0.8977075
## x2            5.083333   0.636560   7.9856 1.398e-06 ***
## x3            0.250000   0.636560   0.3927 0.7004292
## x4           -6.083333   0.636560  -9.5566 1.633e-07 ***
## x1:x2        -2.875000   0.779623  -3.6877 0.0024360 **
## x1:x3        -3.750000   0.779623  -4.8100 0.0002773 ***
## x1:x4         4.375000   0.779623   5.6117 6.412e-05 ***
## x2:x3         4.625000   0.779623   5.9324 3.657e-05 ***
## x2:x4        -1.500000   0.779623  -1.9240 0.0749257 .
## x3:x4        -2.125000   0.779623  -2.7257 0.0164099 *
## x1^2         -2.037500   0.603894  -3.3739 0.0045424 **
## x2^2         -1.662500   0.603894  -2.7530 0.0155541 *
## x3^2         -2.537500   0.603894  -4.2019 0.0008873 ***
## x4^2         -0.162500   0.603894  -0.2691 0.7917877
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.9555, Adjusted R-squared:  0.9078
## F-statistic: 20.04 on 15 and 14 DF,  p-value: 6.54e-07
##
## Analysis of Variance Table
##
## Response: ave
##                      Df  Sum Sq Mean Sq F value    Pr(>F)
## block                 1   16.81   16.81  1.7281  0.209786
## FO(x1, x2, x3, x4)    4 1510.00  377.50 38.8175 1.965e-07
## TWI(x1, x2, x3, x4)   6 1114.00  185.67 19.0917 5.355e-06
## PQ(x1, x2, x3, x4)    4  282.54   70.64  7.2634  0.002201
## Residuals            14  136.15    9.72
## Lack of fit          10  125.40   12.54  4.6660  0.075500
## Pure error            4   10.75    2.69
##
## Stationary point of response surface:
##         x1         x2         x3         x4
##  0.8607107 -0.3307115 -0.8394866 -0.1161465
##
## Stationary point in original units:
##         A         R         W         L
## 12.916426  2.434015  1.040128  1.941927
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1]  3.258222 -1.198324 -3.807935 -4.651963
##
## $vectors
##          [,1]       [,2]       [,3]        [,4]
## x1  0.5177048 0.04099358  0.7608371 -0.38913772
```

```
## x2 -0.4504231 0.58176202  0.5056034  0.45059647
## x3 -0.4517232 0.37582195 -0.1219894 -0.79988915
## x4  0.5701289 0.72015994 -0.3880860  0.07557783
```

Predict the response variable at the stationary point:

```
max <- data.frame(x1 = 0.8607107, x2 = -0.3307115, x3 = -0.8394866, x4 = -0.1161465, block = 
"1") #use the coded values
predict(heli.rsm, newdata = max) #predict the response variable using this model at these par
ameters
```

```
##        1 
## 372.1719
```

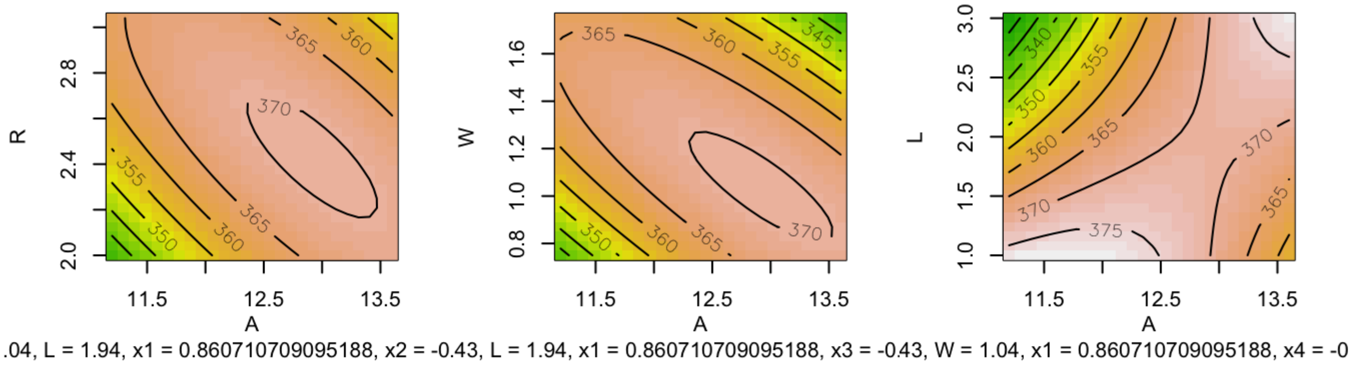
- In the ANOVA table, we can see that many of the second order terms (i.e. a:b, ab^x terms) have statistically significant (low) p-values
- This means that they contribute significantly to the model and that this ‘canonical’ analysis is relevant
- The stationary point seems to be close to the optimized region, but since the eigenvalues are of mixed sign so this stationary point is a saddle point (neither a minimum or maximum). Further analysis is requried

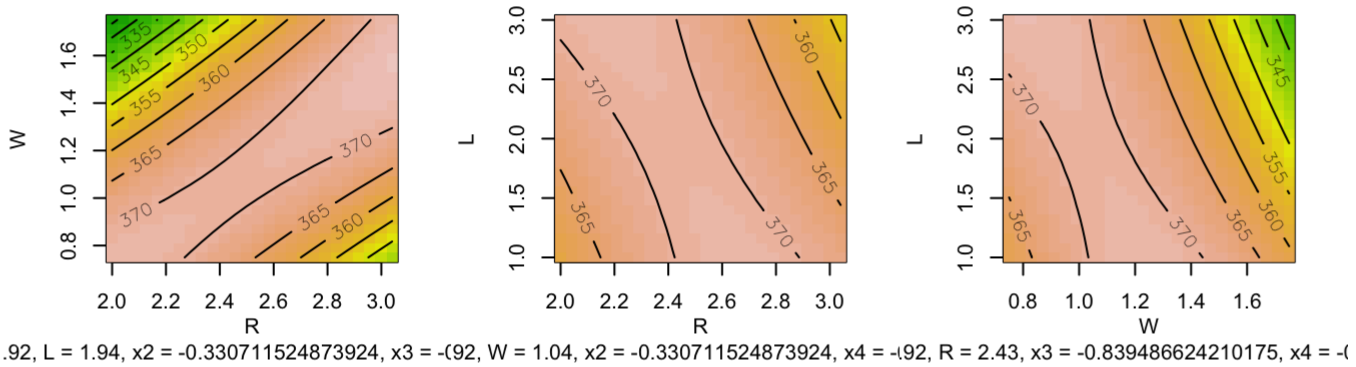
Displaying a response surface

- The canonical analyses and the breakdown of their tables are important for understanding the behavior of a second order response surface
- However, effective graphs tend to be easier to interpret
- `rsm` includes functions for data visualization
- In the paper helicopter example, since there are four predictor variables, there are six pairs of predictors

Visualize the behavior of the fitted surface around the stationary point:

```
 par(mfrow = c(2, 3))
contour(heli.rsm, ~ x1 + x2 + x3 + x4,
        image = TRUE, #fills spaces between the contour lines with color; color levels are co
nsistent across plots
   at = summary(heli.rsm)$canonical$xs) #location of stationary point in the coded units
```





Direction for further experimentation

- In many first-order and some second-order cases, we find that the saddle point or stationary point is far from the optimal experimental region
- If this is the case then the most practical the next is to determine where in which direction to investigate further
- This is a facet of RSM called direction of steepest ascent
- The `rsm` summary table provides information about this concept and we can zoom into that using the `steepest` function:

```
steepest(CR1.rsm, #dataframe to access
         dist = c(0,0.5,1)) #distance
```

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

```
##   dist    x1    x2 |   Time    Temp |   yhat
## 1  0.0 0.000 0.000 | 85.000 175.000 | 82.814
## 2  0.5 0.407 0.291 | 87.035 176.455 | 83.352
## 3  1.0 0.814 0.581 | 89.070 177.905 | 83.890
```

- yhat is the predicted value of the response variable in the model
- any set of distances can be factored in using the dist argument
- However, while the fitted values are displayed, remember that these are only predictions
- As the distance along the path increases, the predictions become less reliable
- The real use case of the path of steepest ascent is to determine the next set of experiments to get close to the optimal experimental conditions

- For a second-order model, the `steepest` function works, but it employs the 'ridge analysis method' which is a type of analog of the steepest ascent
- In this method, for a specified distance, *d*, a point at which the predicted response is a maximum among all predictor combinations at radius *d* can be determined
- It is sensible to use this method when the stationary point is somewhat far away

```
steepest(heli.rsm, #dataframe to access
         dist = c(0,0.5,1)) #distance
```

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

```
##   dist     x1    x2    x3     x4 |       A       R     W      L |    yhat
## 1  0.0  0.000 0.000 0.000  0.000 | 12.4000 2.52000 1.250 2.0000 | 372.800
## 2  0.5 -0.127 0.288 0.116 -0.371 | 12.3238 2.59488 1.279 1.8145 | 377.106
## 3  1.0 -0.351 0.538 0.312 -0.700 | 12.1894 2.65988 1.328 1.6500 | 382.675
```