Response Surface Methods

Neil J. Hatfield

4/30/2021

In this tutorial, we are going to explore Response Surface Methods in R.

New Package

We are going to make use of one new package, `rsm`.

Context

We would like to explore improving boxed cake mix. In particular, we're interested in making our cake more palatable (appeal) . While there are many quantities we could manipulate (e.g., the amount of flour, sugar, number of eggs, amount of oil, water), we are going to focus on just two attributes: the baking temperature and baking time.

We're using these two options as they are both fairly easy for us to manipulate and a good starting point for improving. We are also going to keep things simple for the example and only work with two design variables (factors).

From the package, we're told to bake our cake at 350°F for 35 minutes. We will use this as our basis for our ***sequence*** of experiments.

Experiment 1

We are going to start with a relatively simple response surface design called *Central Composite*. We will have just three levels for each our factors: a "low", a "center", and a "high" value. For temperature, we will use 340°F, 350°F, and 360°F. For time, we'll use 33, 35, and 37 minutes.

We will *not* do a full crossing. Rather, we're going to only cross the low and high levels. We will keep the the central values together. Thus, our treatments (Temp-Time) will be Low-Low, Low-High, High-Low, High-High, and Center-Center. The first four treatments are *factorial points* as they are a whole step above or below center; the last treatment is called a *center point*.

We will also use imbalance to our advantage: we are going to bake three cakes at the Center and just one cake on the outer edges.

For each cake, we'll invite a group of tasters over to to eat cake, and provide scores. We will record the *SAM* of the scores for each cake. This means we will have just one score for each cake.

Data From Experiment 1

I'm going to create a data frame that contains the *uncoded* data values generated via our first experiment. Keep in mind that the appeal contains values of the *SAM* for each cake across all testers. Thus, our data frame's rows are the observations of a single cake.

```r
baking1 <- data.frame(
  time = c(33,37,33,37,35,35,35),
  temp = c(340,340,360,360,350,350,350),
  appeal = c(3.89, 6.36, 7.65, 6.79, 8.36, 7.63, 8.12)
)
```

Analyzing the Experiment 1 Data

Step 1–Code the Data

Remember that for Response Surface methods, we don't use the values of our data directly. Rather we use *coded* versions of data.

When your data come to you in *uncoded* form, our first step is to created coded versions. The `rsm` package has a function that can automatically do this for us: `coded.data`.

Unfortunately, this function will rename our factors to generic `x#`. Thus, we will want to fix those. Luckily, the order of columns is still the same as our original data frame.

```r
# Coding the data
coded1 <- rsm::coded.data(
  data = baking1
)

names(coded1) <- c("codedTime", "codedTemp", "appeal")
```

Step 2–Check What Models Work

We can look to see if a first order and/or second order model will work by attempting to create visualization for the scaled variance function. The details of the scaled variance function are beyond what we are going to concern ourselves with for this course.

For right now, we are mainly concerned with whether we can actually create the plot. If we can, then our model has sufficient information to be able to estimate key pieces of information. If we can't get the plot to generate, then that would indicate that we do not have sufficient information to properly fit the model.

```r
# Checking First Order
rsm::varfcn(
  design = coded1,
  formula = ~ rsm::FO(codedTime,codedTemp), # FO for First Order
  contour = TRUE,
  main = "Variance Function for 1st Experiment"
)
```

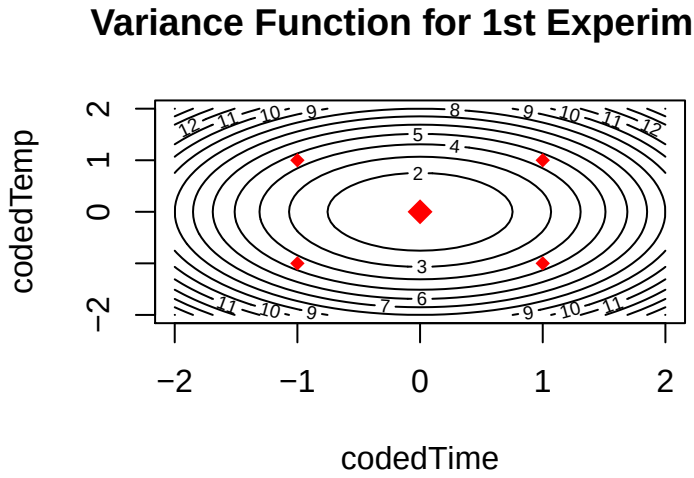

Figure 1: Does First Order Model Work?

```
# Check Second Order
rsm::varfcn(
  design = coded1,
  formula = ~ rsm::SO(codedTime,codedTemp), # SO for Second Order
  contour = TRUE,
  main = "Variance Function for 1st Experiment"
)
```

```
## Error in solve.default(t(mm) %*% mm): Lapack routine dgesv: system is exactly singular: U[6,6] = 0
```

Notice that we were able to get a plot for a First Order model (what we were aiming for), but we got an error message for the Second Order model. This indicates that we can continue to fitting a First Order model.

(Further interpretation of the plot is beyond this course.)

Step 3–Fit the First Order Model

We will now fit the first order model to our experiment data. For this, we'll use the `rsm` function from the `rsm` package. To help us specify that we want a First Order model, we will also use the `FO` function. This will ensure that you get all appropriate terms (and that R thinks about the data in the right way) that we need.

```
## Fitting the First Order Model
bakingFOModel <- rsm::rsm(
  formula = appeal ~ rsm::FO(codedTime, codedTemp),
  data = coded1
)
```

Step 4–Look at Plots

Let's look at a perspective plot for our model:

```
## Perspective Plot
persp(
  x = bakingFOModel,
  form = ~ codedTime + codedTemp
)
```

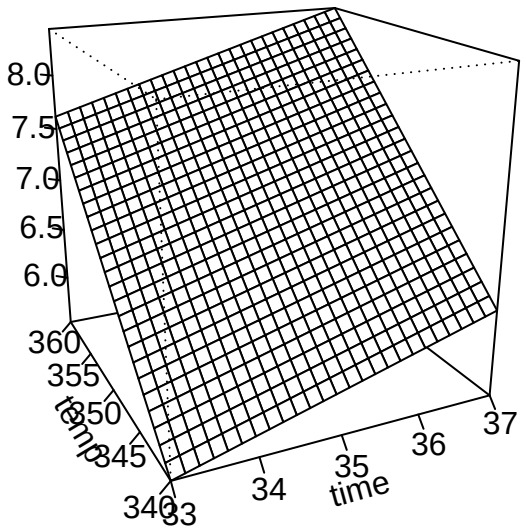

Figure 2: Perspective Plot for First Baking Experiment

From the perspective plot, we can see our linear response surface (a plane) and we can see that there does appear to be an increase in the cake's appeal as we increase baking time and temperature.

Let's look at a contour plot for our model:

```
## Contour Plot
contour(
  x = bakingFOModel,
  form = ~ codedTime + codedTemp
)
```

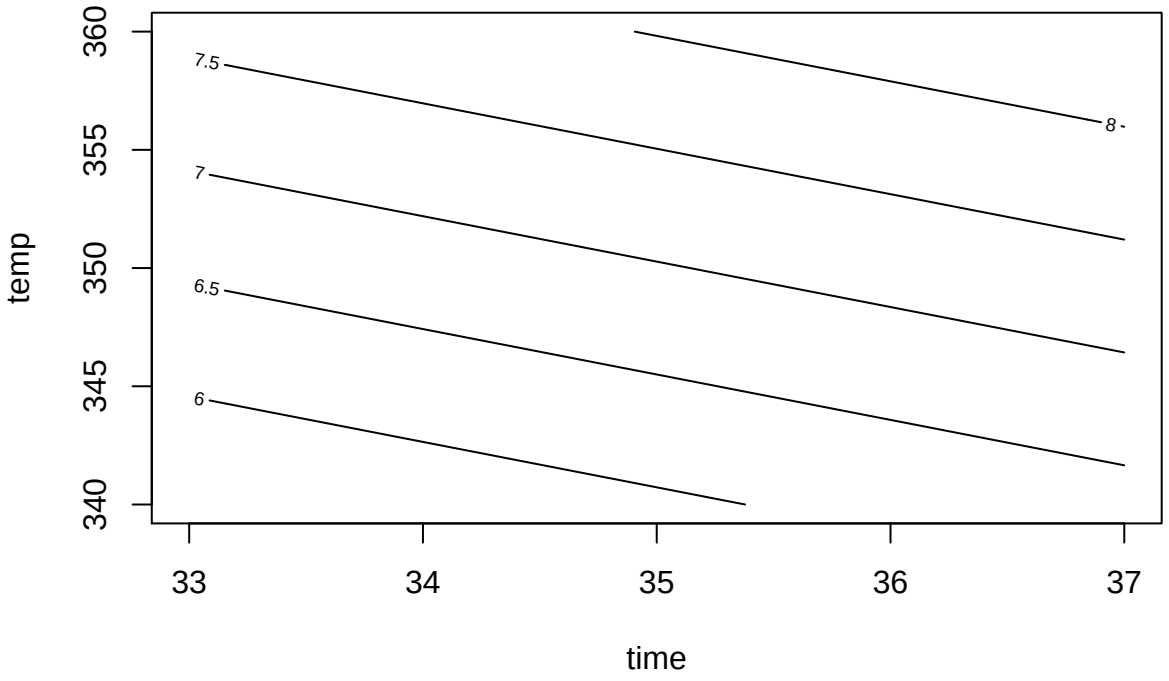

Figure 3: Contour Plot for First Baking Experiment

From the contour plot, we can see that as we increase the baking time and temperature, we end up with higher appeal values.

Step 5–Look at the Summary

For this, we will just concern ourselves with raw output.

```
summary(bakingFOModel)
```

```
##
## Call:
## rsm(formula = appeal ~ rsm::FO(codedTime, codedTemp), data = coded1)
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  6.97143    0.56715 12.2921 0.0002516 ***
## codedTime    0.40250    0.75027  0.5365 0.6200767
## codedTemp    1.04750    0.75027  1.3962 0.2351637
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.3587, Adjusted R-squared:  0.03801
## F-statistic: 1.119 on 2 and 4 DF,  p-value: 0.4113
##
## Analysis of Variance Table
##
## Response: appeal
##                                Df Sum Sq Mean Sq F value  Pr(>F)
## rsm::FO(codedTime, codedTemp)  2 5.0371  2.5185  1.1185 0.41130
## Residuals                      4 9.0064  2.2516
## Lack of fit                    2 8.7296  4.3648 31.5299 0.03074
## Pure error                     2 0.2769  0.1384
##
## Direction of steepest ascent (at radius 1):
## codedTime codedTemp
## 0.3586805 0.9334604
##
## Corresponding increment in original units:
##     time     temp
## 0.717361 9.334604
```

The t values and p-values for `codedTime` and `codedTemp` tell us that neither baking time nor baking temperature appear to impact the appeal of the cake.

If we look at the `Lack of fit` line ofthe ANOVA table, we can see that there is evidence that our First Order model is missing something.

Because we have a lack of fit, the last section for the direction of steepest ascent is not reliable.

IF we did not have a lack of fit, these values represent the rate of change of baking time (or baking temperature) with respect to appeal. This the inverse of what we usually have (rate of change of *appeal* with respect to *baking time*). You can use the multiplicative inverse to correct this

- $0.717361^{-1} = 1.3939983$ appeal units per minute
- $9.334604^{-1} = 0.1071283$ appeal units per degree

Experiment 2

We had a lack of fit of our first order model. Thus, we are going to a second experiment. To see about getting some improvements, we will add some *axial points*. Axial points will have center values for one factor and then a "key value" for the other factor. See Table 19.1 in Oehlert for coded key values.

In our second experiment we will bake 7 more cakes: one at each of 4 axial points and three more at the center point.

Once we bake the cakes, we will do the same procedure as in Experiment 1 (invite people, eat cake, score, calculate values of SAM). However, we will do something unique, we will **combine** the new data with the old data.

We will add a block to both the data frame and the model do denote which experiment each datum comes from.

Data From Experiment 2

For this portion, we're going to import the data in an already coded form:

```
baking2 <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/bake2.dat",
  header = TRUE
)
# Tell R to not treat our experiment numbers, the block, as numbers
baking2$block <- as.factor(baking2$block)
```

Notice that we used the `as.factor` function to tell R that the block column should not be treated as integers.

Analyze the Experiment 2

When you have already coded data, the process is a bit different. Essentially, we need to provide how to *uncode* the data:

Step 1–Code/Uncode the Data

When you load coded data, we don't have to tell R to code the data. However, you'll notice that R did tell us values in both coded and uncoded form. Thus, we do need to R how to uncode the data.

We need to give R a formula which will "uncode" the coded data. In essence, the formula has the form

$$coded = \frac{uncoded - \text{Center Value}}{\text{step size}}$$

where the step size is how far from the center value we are calling a single step.

```
coded2 <- rsm::as.coded.data(
  data = baking2,
  codedTime ~ (time - 35) / 2,
  codedTemp ~ (temp - 350) / 10,
  block = "block"
)
```

Step 2–Will a Second Order Model Work?

We will again see if we can generate a scaled variance plot. If so, we have the information to fit a Second Order model.

```
# Check Second Order
rsm::varfcn(
  design = coded2,
  formula = ~ rsm::SO(codedTime, codedTemp), # SO for Second Order
  contour = TRUE,
  main = "Variance Function for 2nd Experiment"
)
```

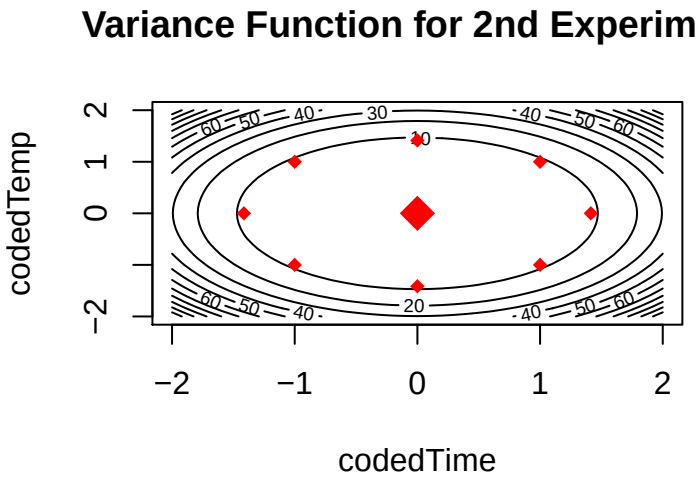

Figure 4: Does Second Order Model Work?

Success, we do.

Step 3–Fit the Second Order Model

We will now fit the second order model to our second experiment's data, **combined with** the data from the first experiment.

Notice that we will need to add the experiment number (our block) to the formula statement.

```
## Fitting the Second Order Model
bakingSOModel <- rsm::rsm(
  formula = appeal ~ block + rsm::SO(codedTime, codedTemp),
  data = coded2
)
```

Step 4–Look at Plots

Let's look at a perspective plot for our model:

```
## Perspective Plot
persp(
  x = bakingSOModel,
  form = ~ codedTime + codedTemp
)
```

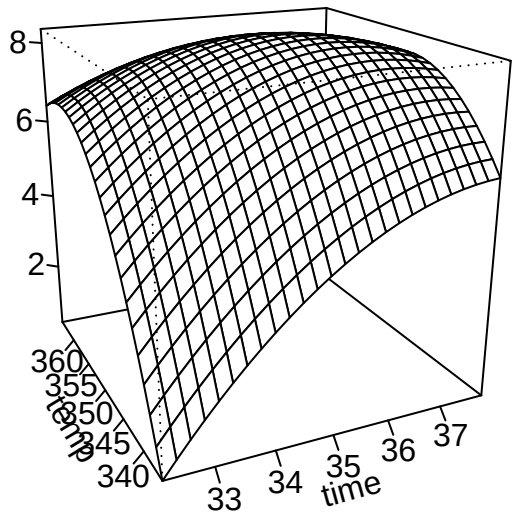

Figure 5: Perspective Plot for Second Baking Experiment

From the perspective plot, we can see our quadratic surface appears to have maximum point.

Let's look at a contour plot for our model:

```
## Contour Plot
contour(
  x = bakingSOModel,
  form = ~ codedTime + codedTemp
)
```

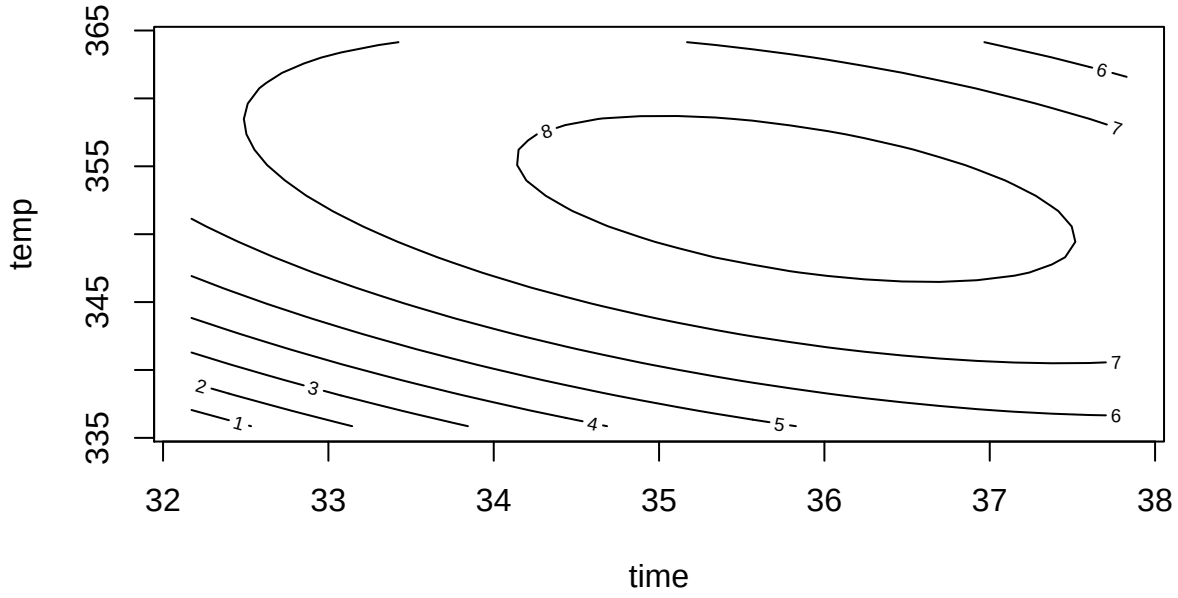

Figure 6: Contour Plot for Second Baking Experiment

In the contour plot, we have a concentric ellipses highlighting increasing levels of appeal. We can use these along with the axes to see where the highest appeals occur.

Step 5–Look at the Summary

For this, we will just concern ourselves with raw output.

```
summary(bakingSOModel)
```

```
##
## Call:
## rsm(formula = appeal ~ block + rsm::SO(codedTime, codedTemp),
##     data = coded2)
##
##                      Estimate Std. Error t value  Pr(>|t|)
## (Intercept)          8.070004   0.184207 43.8095 8.431e-10 ***
## block1              -0.056986   0.120592 -0.4725 0.6509127
## codedTime            0.735146   0.159540  4.6079 0.0024610 **
## codedTemp            0.964003   0.159540  6.0424 0.0005199 ***
## codedTime:codedTemp -0.832500   0.225606 -3.6901 0.0077560 **
## codedTime^2         -0.627555   0.166078 -3.7787 0.0069043 **
## codedTemp^2         -1.195226   0.166078 -7.1968 0.0001780 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Multiple R-squared:  0.9503, Adjusted R-squared:  0.9077
## F-statistic: 22.31 on 6 and 7 DF,  p-value: 0.000313
##
## Analysis of Variance Table
##
## Response: appeal
##                               Df  Sum Sq Mean Sq F value    Pr(>F)
## block                          1  0.0457  0.0457  0.2245 0.6500273
## rsm::FO(codedTime, codedTemp)  2 11.7562  5.8781 28.8717 0.0004156
## TWI(codedTime, codedTemp)      1  2.7722  2.7722 13.6165 0.0077560
## PQ(codedTime, codedTemp)       2 12.6763  6.3382 31.1315 0.0003282
## Residuals                      7  1.4252  0.2036
## Lack of fit                    3  0.9470  0.3157  2.6409 0.1857168
## Pure error                     4  0.4781  0.1195
##
## Stationary point of response surface:
## codedTime codedTemp
## 0.4138302 0.2591515
##
## Stationary point in original units:
##      time      temp
##  35.82766 352.59152
##
## Eigenanalysis:
## eigen() decomposition
## $values
## [1] -0.4075785 -1.4152030
##
## $vectors
##                 [,1]      [,2]
## codedTime -0.8841312 0.4672387
## codedTemp  0.4672387 0.8841312
```

Table of t Statistics The first portion of the output has our table for seeing which terms have a statistically significant impact on appeal. All first and second order terms are significant, so we will not remove any terms. Additionally, the estimates listed here are our coefficients for each term (i.e., the β values).

You'll notice that I skipped saying anything about the block. Just like RCBDs, we aren't interested in testing whether the block is statistically significant. Rather, this is an aspect that we jsut want to have accounted for.

Notice that the Adjusted R-squared value (i.e., ϵ^2) is approximately 91%! Our second order model explains essentially 91% of the variation in the SAM of appeal.

ANOVA Table The next table (the ANOVA table) has our lack of fit test, which shows us that the second order model appears to be doing well. That is, the second order model does a decent job of explaining the response surface given the times and temperatures we used.

Stationary Points and Eigenanalysis The last section of the summary points out our stationary points (either a maximum, a minimum, or saddle point). The coordinates are 35.83 minutes and 352.59°F. These are not far off from our Center Point (our original "best guess").

To decide what kind of stationary point we have, we need to look at the eigenvalues. (This would be equivalent to checking the second derivative when using calculus for an optimization problem.)

- All negative eigenvalues–Maximum
- All positive eigenvalues–Minimum
- Mixture–Saddle Point–a maximum in one direction, a minimum in the other direction

Since both of the eigenvalues (-0.4076 and -1.4152) are negative, our stationary point reflects a maximumization of the SAM of cake appeal.

A quick word of caution: the stationary points are *estimates* and the level of precision is unclear (we don't have any measure of standard error). You can use these values but don't treat them as perfect measures.

Code Appendix

```
# Setting Document Options
knitr::opts_chunk$set(
  echo = FALSE,
  warning = FALSE,
  message = FALSE,
  fig.align = "center"
)

packages <- c("tidyverse", "knitr", "kableExtra",
              "rsm")
lapply(packages, library, character.only = TRUE)

options(knitr.kable.NA = "")
options(contrasts = c("contr.sum", "contr.poly"))

source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")

baking1 <- data.frame(
  time = c(33,37,33,37,35,35,35),
  temp = c(340,340,360,360,350,350,350),
  appeal = c(3.89, 6.36, 7.65, 6.79, 8.36, 7.63, 8.12)
)

# Coding the data
coded1 <- rsm::coded.data(
  data = baking1
)

names(coded1) <- c("codedTime", "codedTemp", "appeal")

# Checking First Order
rsm::varfcn(
  design = coded1,
  formula = ~ rsm::FO(codedTime,codedTemp), # FO for First Order
  contour = TRUE,
  main = "Variance Function for 1st Experiment"
)

# Check Second Order
rsm::varfcn(
  design = coded1,
  formula = ~ rsm::SO(codedTime,codedTemp), # SO for Second Order
  contour = TRUE,
  main = "Variance Function for 1st Experiment"
)

## Fitting the First Order Model
bakingFOModel <- rsm::rsm(
  formula = appeal ~ rsm::FO(codedTime, codedTemp),
  data = coded1
)
```

```r
## Perspective Plot
persp(
  x = bakingFOModel,
  form = ~ codedTime + codedTemp
)

## Contour Plot
contour(
  x = bakingFOModel,
  form = ~ codedTime + codedTemp
)

summary(bakingFOModel)

baking2 <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/bake2.dat",
  header = TRUE
)
# Tell R to not treat our experiment numbers, the block, as numbers
baking2$block <- as.factor(baking2$block)

coded2 <- rsm::as.coded.data(
  data = baking2,
  codedTime ~ (time - 35) / 2,
  codedTemp ~ (temp - 350) / 10,
  block = "block"
)

# Check Second Order
rsm::varfcn(
  design = coded2,
  formula = ~ rsm::SO(codedTime, codedTemp), # SO for Second Order
  contour = TRUE,
  main = "Variance Function for 2nd Experiment"
)

## Fitting the Second Order Model
bakingSOModel <- rsm::rsm(
  formula = appeal ~ block + rsm::SO(codedTime, codedTemp),
  data = coded2
)

## Perspective Plot
persp(
  x = bakingSOModel,
  form = ~ codedTime + codedTemp
)

## Contour Plot
contour(
  x = bakingSOModel,
  form = ~ codedTime + codedTemp
)
```

```
summary(bakingSOModel)
```