Problem_Set_3_Carlos_Sathler

Carlos Sathler

2/12/2019

Contents

Problem 1

(a)

```
#scatterplotMatrix(~ WT18 + WT2 + WT9, data=BGSgirls, regLine = T, smooth = F)
BGSgirls.lm = lm(WT18 ~ WT2 + WT9, data=BGSgirls)
tidy(BGSgirls.lm)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic       p.value
##   <chr>          <dbl>     <dbl>     <dbl>         <dbl>
## 1 (Intercept)   34.3       6.55       5.23 0.00000180
## 2 WT2           -0.974     0.701     -1.39 0.169
## 3 WT9            1.20      0.179      6.70 0.00000000509
```

There relationship between girl's weight at age 18 and weights at age 2 and 9 is the following:
$WT18 = 34.26 - 0.97 * WT2 + 1.2 * WT9$

(b)

Actually, the model suggests that girls who have the **same weight at age 9**, and are heavier than average at age 2, should NOT be heavier than average at age 18. The negative sign in the model means that if two girls have the same weight at age 9, their difference in weight at age 18 will be **inversely proportional** to their weight at 2. More exactly, according to the model, a difference in weight of +1kg at age 2, will equate to a difference in weight of -0.97kg at age 18, on average (again, assuming the girls have the same weight at age 9).

(c)

That is incorrect because according to the model any prediction of weight at age 18 should take into account a girl's weight not only at age 9, but also at age 2. In other words, according to the model $\hat{E}(WT18 \mid WT2, WT9) = 34.26 - (0.97)WT2 + (1.2)WT9$. So, in order to predict difference in weight at age 18, we need to consider difference in weight at age 2, in addition to difference of weight at age 9. It is true, however, that if two girls have a one pound difference in weight at age 9 **and they had the same weight at age 2**, then they will have a 1.2 difference in weitght at age 18, on average.

Problem 2

```
#scatterplotMatrix(~ log(acrePrice) + year, data = MinnLand, smooth=F, regLine=F)
MinnLand$year.factor = factor(MinnLand$year)
MinnLand$log.acrePrice = log(MinnLand$acrePrice)
gg = ggplot(data=MinnLand, aes(x=year.factor, y=log.acrePrice)) + geom_boxplot(outlier.alpha = 0.3)
gg = gg + ylab("log(acrePrice)") + xlab("year as factor")
#gg
```

(a)

```
model_a = lm(log(acrePrice) ~ year, data=MinnLand)
tidy(model_a)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) -194.      3.98        -48.7       0
## 2 year           0.100   0.00199      50.6       0
```

Regression equation:
log(acrePrice) = -193.88 + 0.1 * year

(b)

```
model_b = lm(log(acrePrice) ~ year.factor, data=MinnLand)
tidy(model_b)
```

```
## # A tibble: 10 x 5
##    term            estimate std.error statistic   p.value
##    <chr>              <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)      7.27       0.0285  255.     0.
##  2 year.factor2003 -0.00155    0.0321   -0.0484 9.61e-  1
##  3 year.factor2004  0.148      0.0315    4.69   2.76e-  6
##  4 year.factor2005  0.360      0.0318   11.3    1.01e- 29
##  5 year.factor2006  0.394      0.0320   12.3    8.66e- 35
##  6 year.factor2007  0.477      0.0319   15.0    2.43e- 50
##  7 year.factor2008  0.684      0.0316   21.6    2.07e-102
##  8 year.factor2009  0.714      0.0336   21.3    2.43e- 99
##  9 year.factor2010  0.757      0.0326   23.2    1.04e-117
## 10 year.factor2011  0.721      0.0353   20.4    7.94e- 92
```

Note: In the formula I'll use the "indicator variable" notation, from the book An Introduction to Statistical Learing with Applications in R, by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. In the notation I(logical_expression) = 1, if logical_expression = True, and I(logical_expression) = 0 if logical_expression = False.

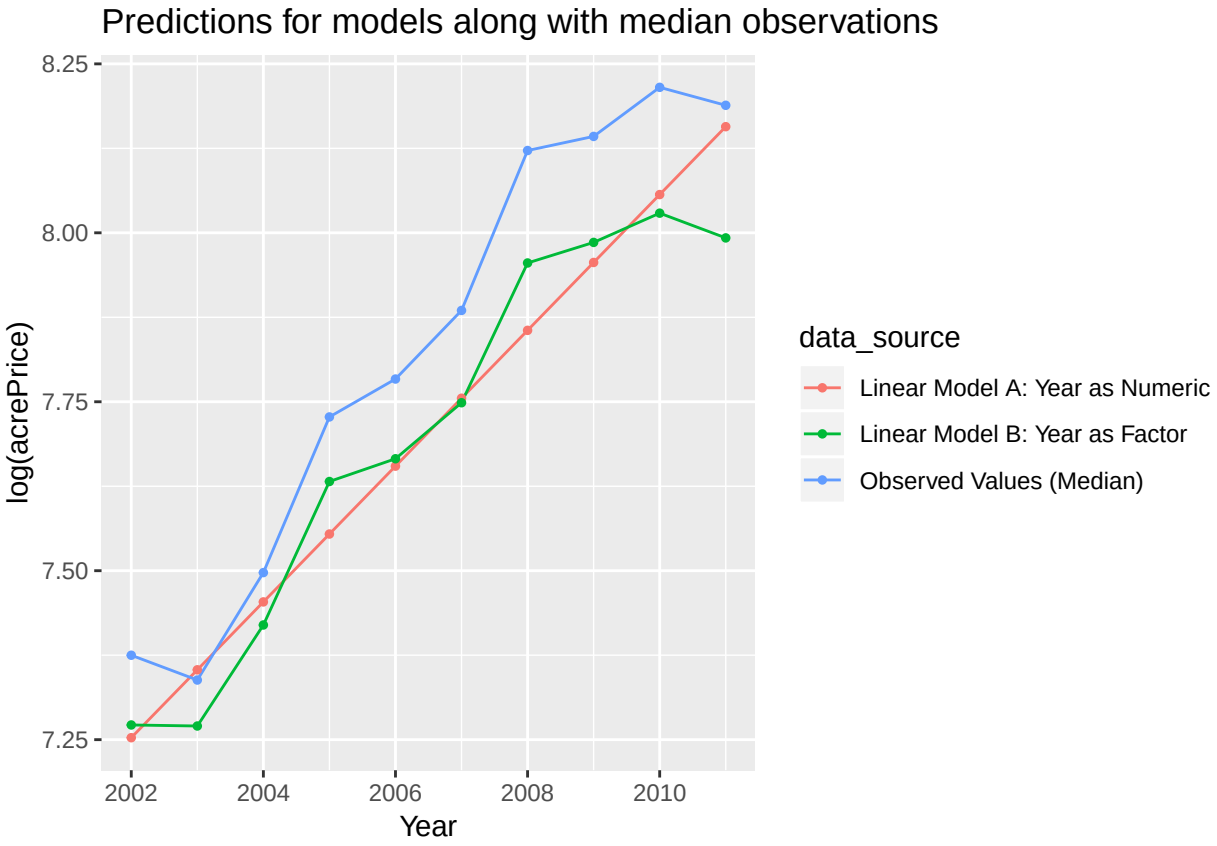
log(acrePrice) = 7.272 - 0.002 * I(year=2003) + 0.148 * I(year=2004) + 0.360 * I(year=2005) + + 0.394 * I(year=2006) + 0.477 * I(year=2007) + 0.684 * I(year=2008) + + 0.714 * I(year=2009) + 0.757 * I(year=2010) + 0.721 * I(year=2011)

Coeficient for the year 2008 is 0.684. It means that prediction for year 2008 is this coeficient plus the baseline, as seen below: log(acrePrice) for year 2008 = 7.272 + 0.684 = 7.956

We note that the baseline 7.272 is the prediction of log(acrePrice) when year = 2002.

(c)

```
set_colnames = function(df) {
  names(df) = c('year','log_of_acrePrice','data_source')
  return(df)
}
# predictions
new.data.year.n = data.frame(year = c(2002:2011))
new.data.year.f = data.frame(year.factor = factor(c(2002:2011)))
model_a.predictions = predict(object = model_a, newdata = new.data.year.n,
                              interval = "prediction", level = 0.95)
model_b.predictions = predict(object = model_b, newdata = new.data.year.f,
                              interval = "prediction", level = 0.95)
# plotting
years = c(2002:2011)
duration = length(years)
model_a.values.df = data.frame(years, model_a.predictions[,1],
                               rep('Linear Model A: Year as Numeric', duration))
model_b.values.df = data.frame(years, model_b.predictions[,1],
                               rep('Linear Model B: Year as Factor', duration))
observed.values.df = cbind(aggregate(log(acrePrice) ~ year, data=MinnLand, median),
                      rep('Observed Values (Median)', duration))
model_a.values.df = set_colnames(model_a.values.df)
model_b.values.df = set_colnames(model_b.values.df)
observed.values.df = set_colnames(observed.values.df)
all_data.df = rbind(model_a.values.df, model_b.values.df, observed.values.df)
gg = ggplot(all_data.df, aes(x=year, y=log_of_acrePrice, col=data_source)) + geom_line() + geom_point(s
gg = gg + ylab('log(acrePrice)') + xlab('Year')
gg = gg + ggtitle('Predictions for models along with median observations')
gg
```

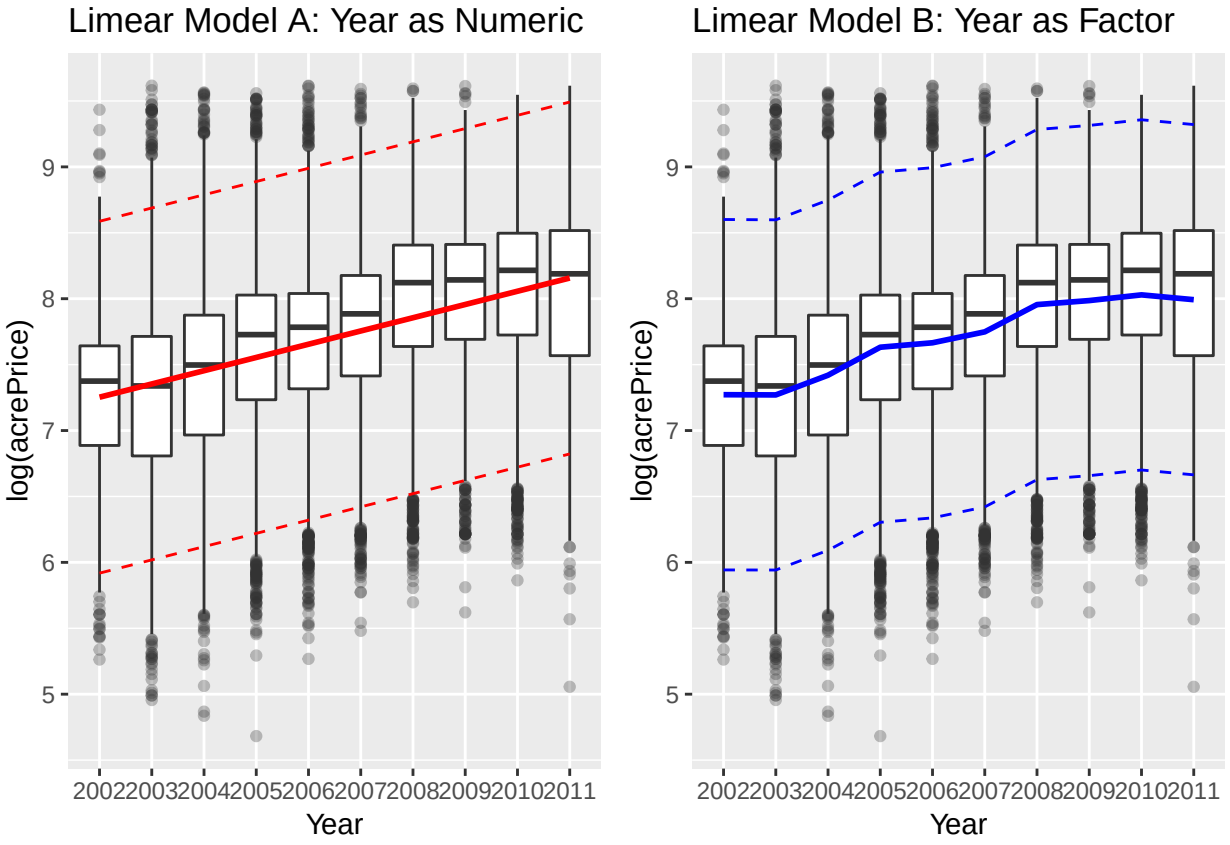


Median of observed values for each year appear in blue. Linear Model A (year as numeric, red) is the best **straight line** that could fit the entire dataset. The point predictions for Model B (year as a factor, green) are the mean values of observations for each year in the range. As seen on the plot, Linear Model B predictions reflect variations in the yearly data, going up and down as the median of the observed data goes up and down.

```
gg1 = ggplot(data=MinnLand, aes(x=year.factor, y=log.acrePrice)) + geom_boxplot(outlier.alpha = 0.3)
gg1 = gg1 + ylab("log(acrePrice)") + xlab("Year") + ggtitle("Limear Model A: Year as Numeric")
gg1 = gg1 + geom_line(data = data.frame(year.factor = factor(c(2002:2011)),
                      log.acrePrice = model_a.predictions[,1]), group=1, lwd=1, color='red')
gg1 = gg1 + geom_line(data = data.frame(year.factor = factor(c(2002:2011)),
                      log.acrePrice = model_a.predictions[,2]), group=1, lty=2, color='red')
gg1 = gg1 + geom_line(data = data.frame(year.factor = factor(c(2002:2011)),
                      log.acrePrice = model_a.predictions[,3]), group=1, lty=2, color='red')

gg2 = ggplot(data=MinnLand, aes(x=year.factor, y=log.acrePrice)) + geom_boxplot(outlier.alpha = 0.3)
gg2 = gg2 + ylab("log(acrePrice)") + xlab("Year") + ggtitle("Limear Model B: Year as Factor")
gg2 = gg2 + geom_line(data = data.frame(year.factor = factor(c(2002:2011)),
                      log.acrePrice = model_b.predictions[,1]), group=1, lwd=1, color='blue')
gg2 = gg2 + geom_line(data = data.frame(year.factor = factor(c(2002:2011)),
                      log.acrePrice = model_b.predictions[,2]), group=1, lty=2, color='blue')
gg2 = gg2 + geom_line(data = data.frame(year.factor = factor(c(2002:2011)),
                      log.acrePrice = model_b.predictions[,3]), group=1, lty=2, color='blue')

ggarrange(gg1, gg2, nrow = 1, ncol = 2)
```



The above plots highlight that predictions for year as a factor are more sensitive to variations of log(acrePrice), going up and down, year by year, as log(acrePrice) increases and decreases.

(d)

```
# Comparing adjusted R-squared for two models
summary(model_a)$adj.r.squared
```

```
## [1] 0.120392
```

```
summary(model_b)$adj.r.squared
```

```
## [1] 0.1289185
```

We see that the adjusted R-squared value for Model B (year as factor) is slightly better.

```
# Comparing Akaike information criterion for two models
AIC(model_a)
```

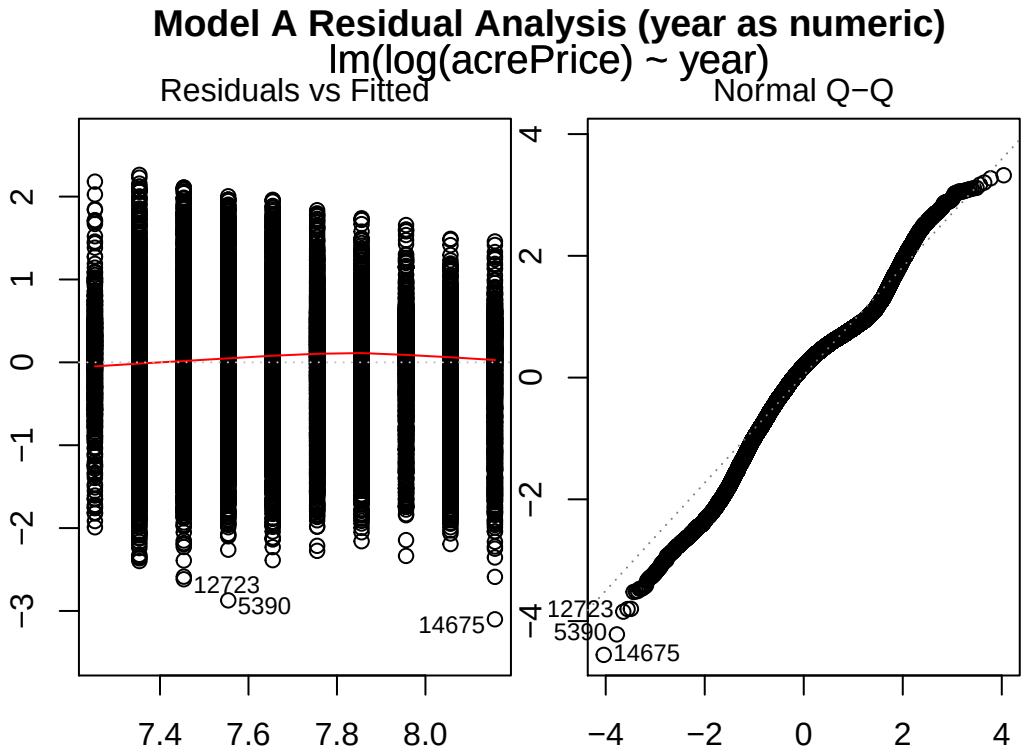
```
## [1] 38693.84
```

```
AIC(model_b)
```

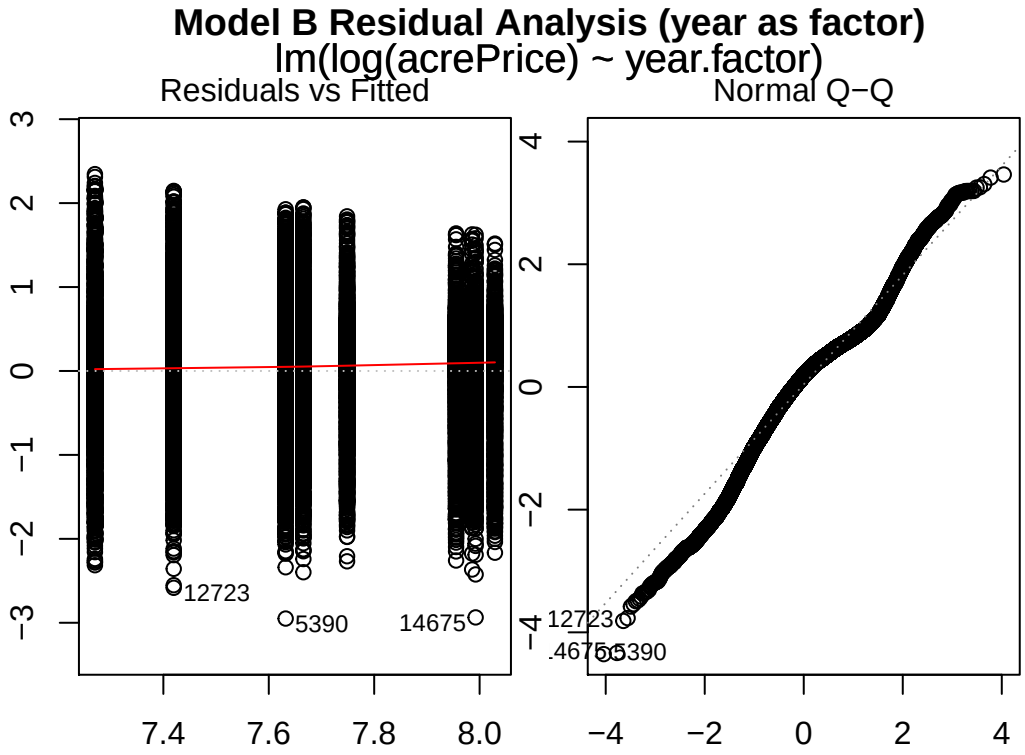
```
## [1] 38519.69
```

Uning the Akaike information criterion we see that, again, Model B offer slightly better result (smaller AIC).

```
par(mfrow=c(1,2), mar = c(1, 1, 1, 1), oma = c(3, 3, 3, 3) )
plot(model_a, which = 1)
plot(model_a, which = 2)
title("Model A Residual Analysis (year as numeric)", outer = TRUE)
```




```
par(mfrow=c(1,2), mar = c(1, 1, 1, 1), oma = c(3, 3, 3, 3) )
plot(model_b, which = 1)
plot(model_b, which = 2)
title("Model B Residual Analysis (year as factor)", outer = TRUE)
```



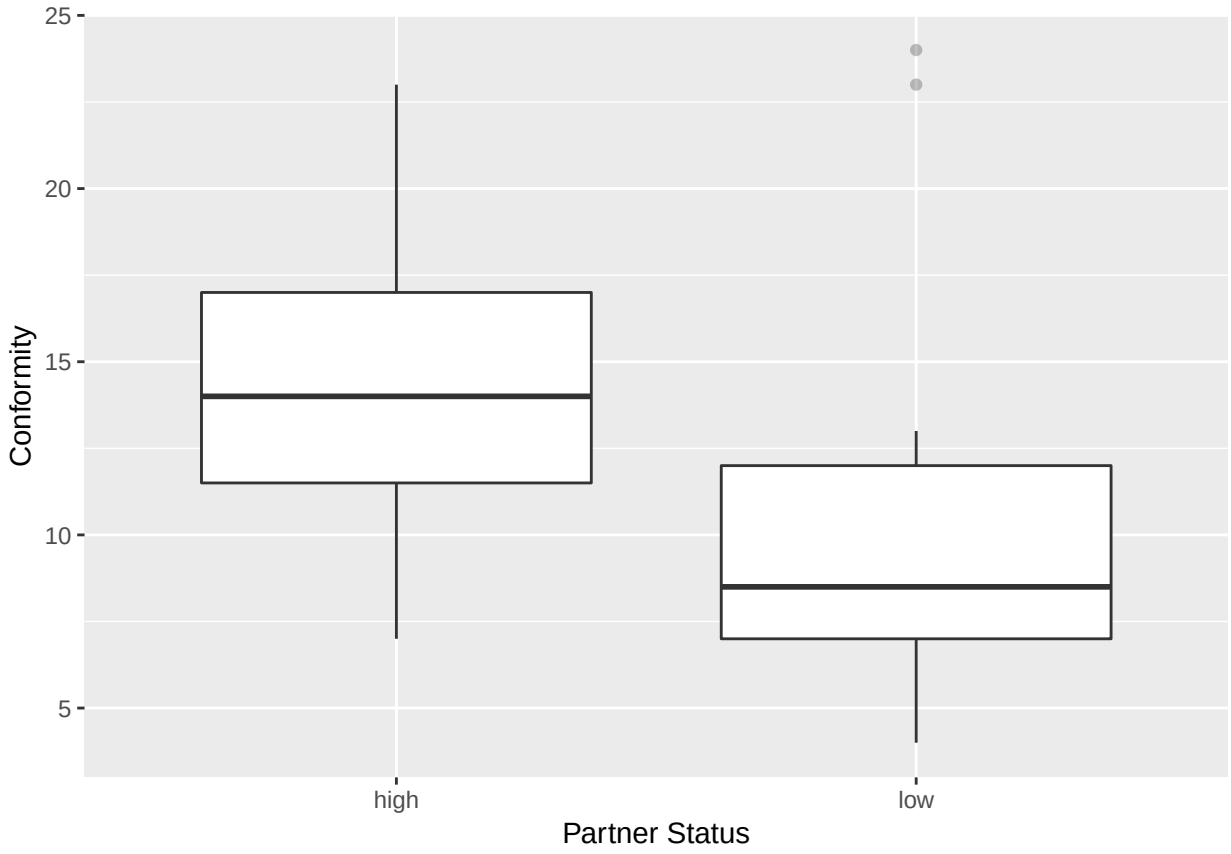
Both models show heterodasticity and non-normal residual distribution, in a comparable way.

Conclusion: Model B is a better fit because residual distribution is not too different between models and Model B has better adjusted R-squared and AIC scores.

Problem 3

(a)

```
#summary(Moore)
gg = ggplot(data=Moore, aes(x=partner.status, y=conformity)) + geom_boxplot(outlier.alpha = 0.3)
gg = gg + xlab("Partner Status") + ylab("Conformity")
gg
```




```
wilcox_test(conformity ~ partner.status, data=Moore, exact=T)
```

```
##
##  Asymptotic Wilcoxon-Mann-Whitney Test
##
## data:  conformity by partner.status (high, low)
## Z = 3.1646, p-value = 0.001553
## alternative hypothesis: true mu is not equal to 0
```

We have evidence to reject the null (alpha = 5%) that the confirmity scores come from the same distribution regarless of partner status. So, there is evidence that partner status affects conformity.

(b)

```
model = lm(conformity ~ fscore + partner.status + fscore:partner.status, data=Moore)
tidy(model)
```

```
## # A tibble: 4 x 5
##   term                      estimate std.error statistic     p.value
##   <chr>                        <dbl>     <dbl>     <dbl>       <dbl>
## 1 (Intercept)                 20.8      3.26       6.37 0.000000127
## 2 fscore                      -0.151    0.0717    -2.11 0.0413
## 3 partner.statuslow          -15.5      4.40      -3.53 0.00104
## 4 fscore:partner.statuslow     0.261    0.0970     2.69 0.0102
```

The answer is yes. The greater than zero coefficient for the interaction term status=low:fscore means that when status changes from high to low, a new term containing score will be added to the formula to calculate conformity. See below:

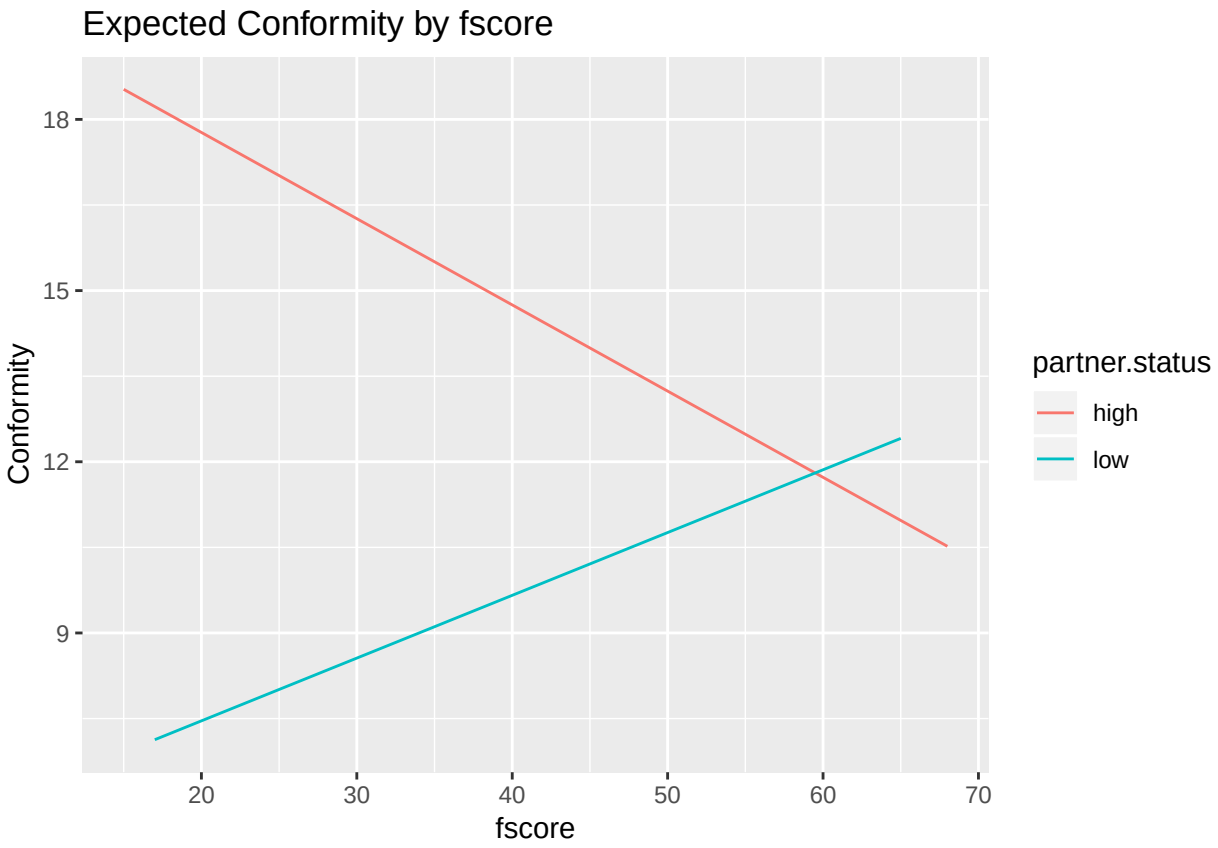
If **status = High** this is the prediction formula for conformity: conformity $= 20.793 - 0.151 * fscore$

If **status = Low** this is the prediction formula for conformity: conformity $= 20.793 - 0.151 * fscore - 15.534 + 0.261 * fscore$ conformity $= 5.259 + 0.11 * fscore$

Assuming the assumptions for the linear model hold reasonably well, all p-values show evidence (alpha = 5%) that we can reject the null that the coefficients are zero, including the coefficient for the interaction term status=low:fscore. That is the coefficient that resulted in the final 0.11 multiplier for fscore in the formula for status=low, versus the 0.151 multiplier for fscore in the formula for status=high. Because the interaction term coefficient is not zero, the effect of status differs for people with different fscores.

(c)

```
model.df = augment(model)
gg = ggplot(model.df, aes(x = fscore, y = .fitted, group=partner.status, color=partner.status)) + geom_l
#gg = gg + facet_wrap(~ partner.status, labeller = label_both)
gg = gg + xlab("fscore") + ylab("Conformity") + ggtitle("Expected Conformity by fscore")
gg
```



As expected from our calculations in (c), when partner status is high, conformity decreases as authoritarianism scores increase; conversely, when status is low, conformity increases as authoritarianism scores increase. The rate of decrease when status is high is more pronouced than the rate of increase when status is low.

Problem 4

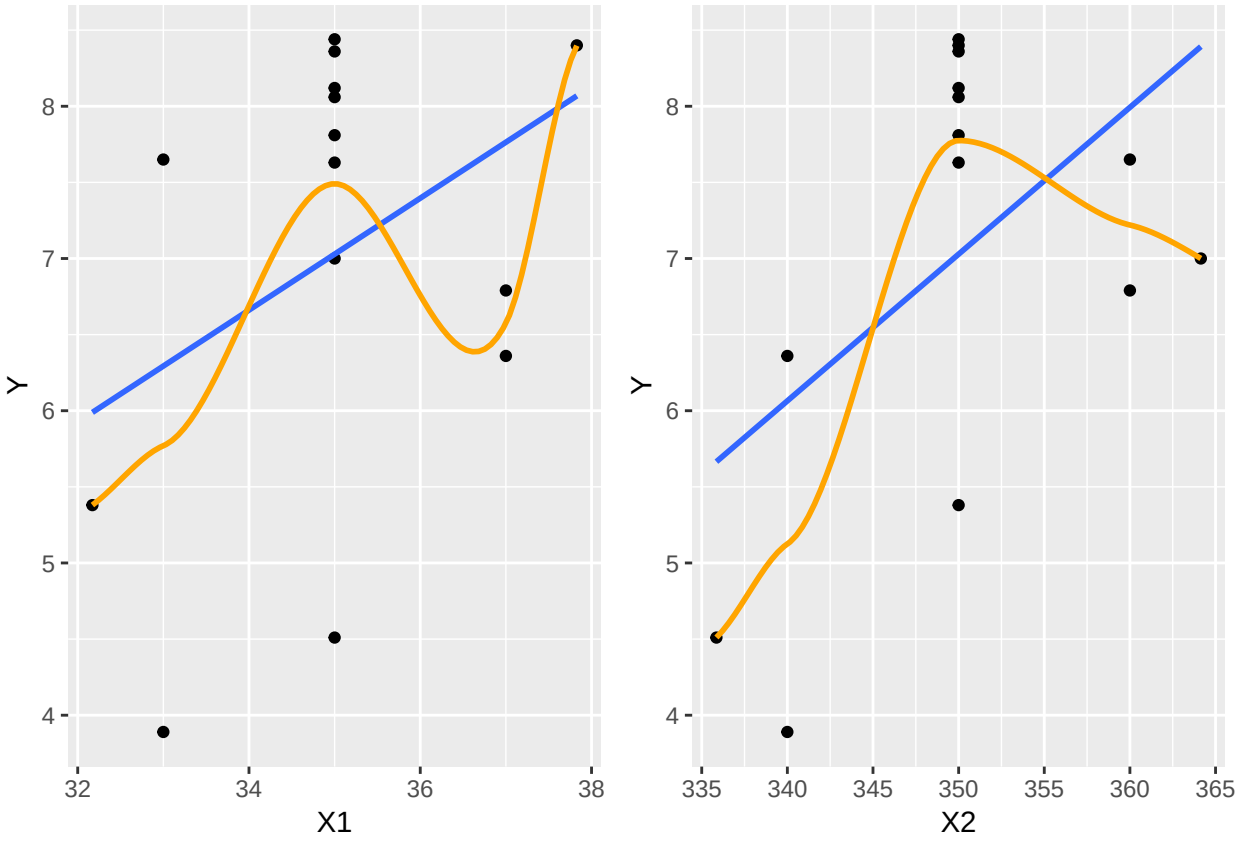
(a)

```
gg1 = ggplot(cakes, aes(y = Y, x = X1)) + geom_point()
gg1 = gg1 + geom_smooth(method='lm', se = F) + geom_smooth(se=F, color ='orange')

gg2 = ggplot(cakes, aes(y = Y, x = X2)) + geom_point()
gg2 = gg2 + geom_smooth(method='lm', se = F) + geom_smooth(se=F, color ='orange')

ggarrange(gg1, gg2, nrow = 1, ncol = 2)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



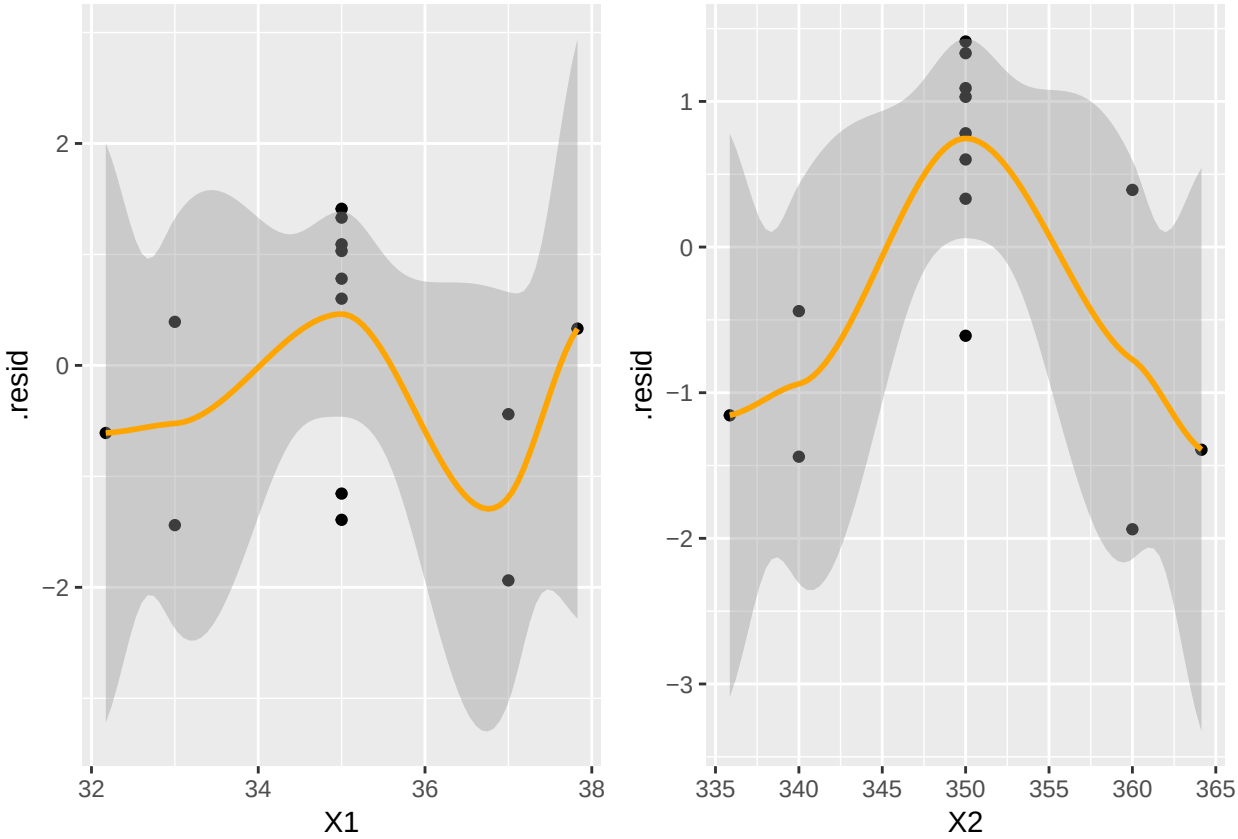
The plots show that the relationships between Y and X1 and Y and X2. Clearly, both are non-linear relationships, which is expected. Of course, taste will be bad if food is not cooked long enough, or if it is cooked at low temperatures. Taste will improve progressively as food cooks long enough, or at high enough temperatures, but if cooked for too long, or at temperatures that are too high, food will overcook or even burn, and taste will be bad.

```
model = lm(Y ~ X1 + X2, data=cakes)
model.df = augment(model)

gg1 = ggplot(model.df, aes(y = .resid, x = X1)) + geom_point() + geom_smooth(color='orange')
gg2 = ggplot(model.df, aes(y = .resid, x = X2)) + geom_point() + geom_smooth(color='orange')

ggarrange(gg1, gg2, nrow = 1, ncol = 2)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Fitting a linear model to express palatability score as a linear function of X1 and X2 results in poor fit, as confirmed by the above residual plots for the model Y = b0 + b1 * X1 + b2 * X2. The residual plots show curvy line around zero, reflecting the poor fit of the model. Clearly, if the relationship betwee Y and X1 and Y and X2 is non-linear, we should not expect a linear function of the 2 explanatory variables combined would work. And we should have known this as well, since low values for both X1 and X2 taken together generate poor testing food; the taste will only improve until both time and temperature reach values in an optimal "zone", after which longer cooking time and/or temperature will cause taste to deteriorate.

(b)

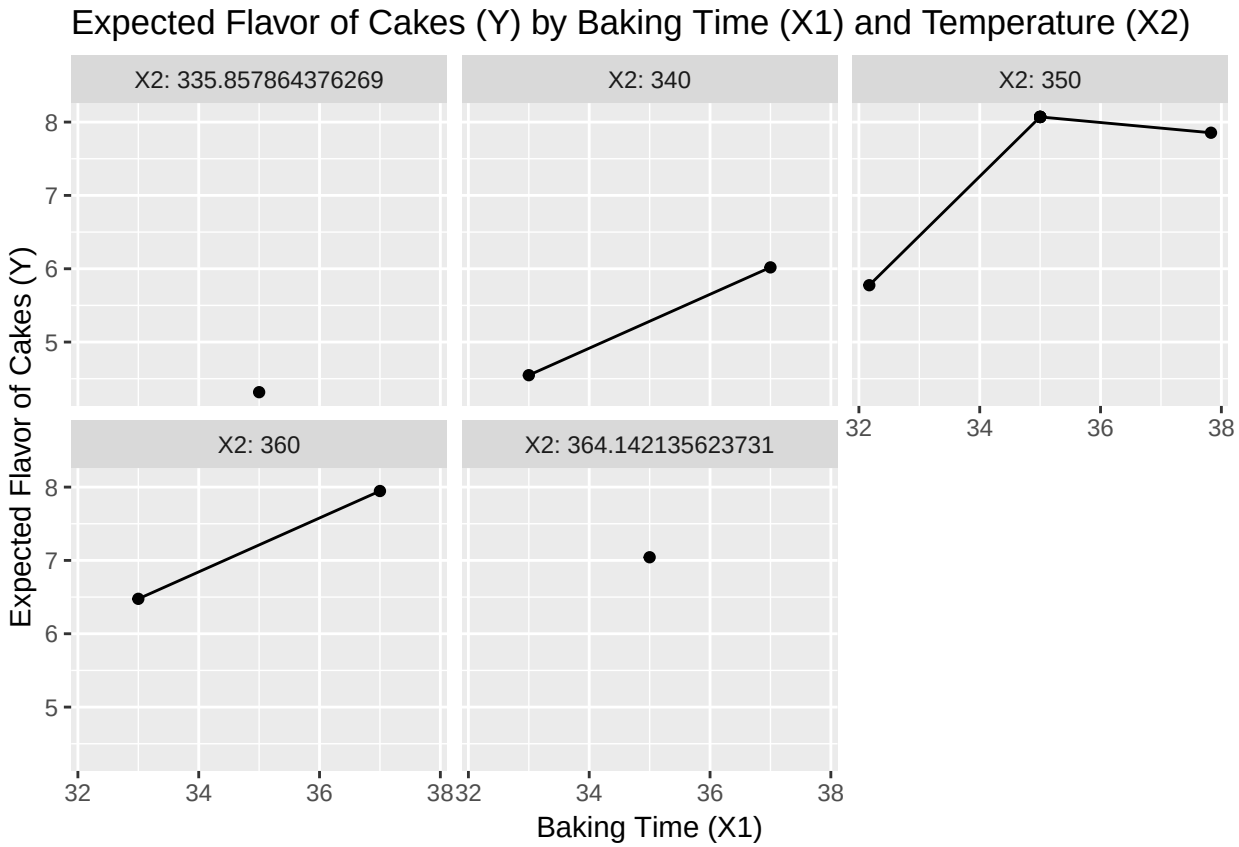
```
model = lm(Y ~ X1 + X2 + I(X1^2) + I(X2^2), data=cakes)
model.df = augment(model)

gg1 = ggplot(model.df, aes(y = .resid, x = X1)) + geom_point() + geom_smooth(color='orange')
gg2 = ggplot(model.df, aes(y = .resid, x = X2)) + geom_point() + geom_smooth(color='orange')

#ggarrange(gg1, gg2, nrow = 1, ncol = 2)
```

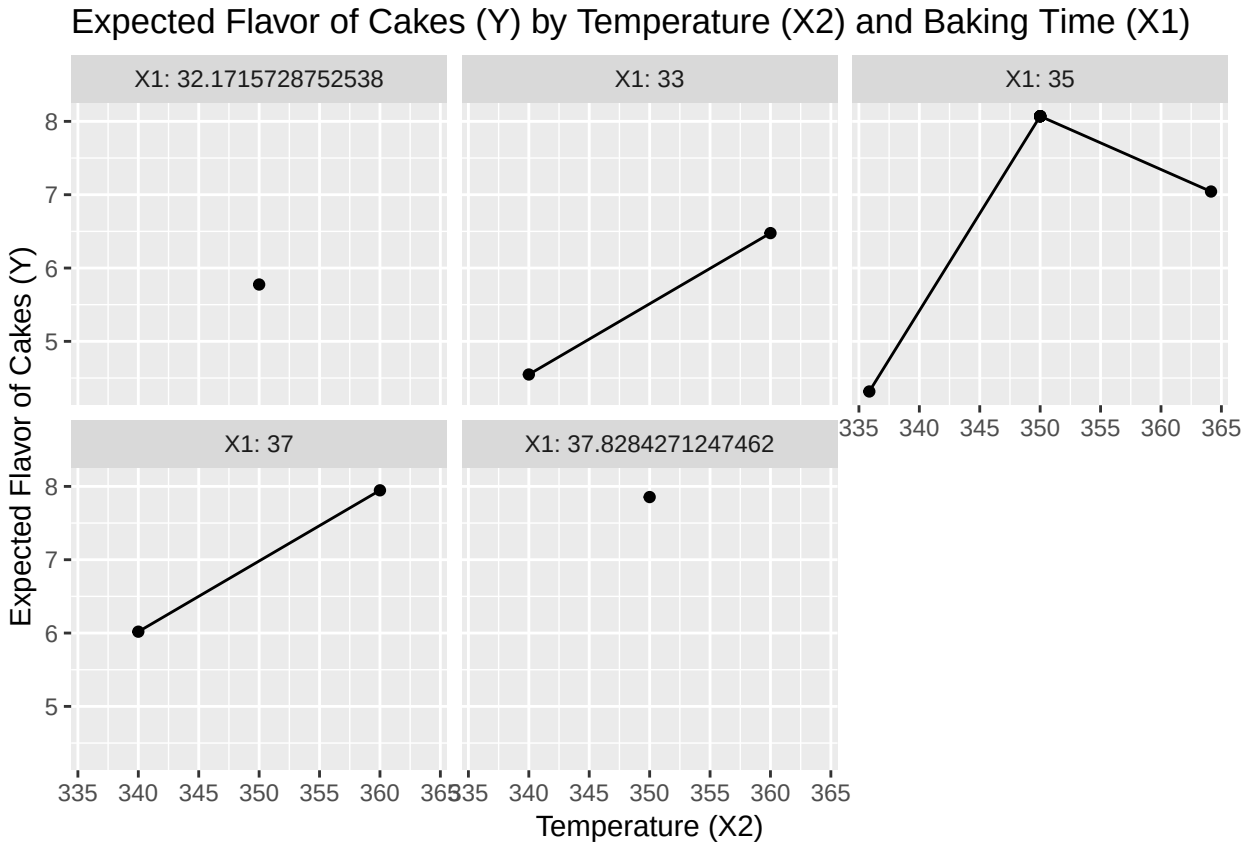
```
gg = ggplot(model.df, aes(y = .fitted, x = X1)) + geom_point() + geom_line()
gg = gg + facet_wrap(~ X2, labeller = label_both)
gg = gg + ggtitle('Expected Flavor of Cakes (Y) by Baking Time (X1) and Temperature (X2)')
gg = gg + ylab('Expected Flavor of Cakes (Y)') + xlab('Baking Time (X1)')
gg
```

```
## geom_path: Each group consists of only one observation. Do you need to
## adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to
## adjust the group aesthetic?
```




```
gg = ggplot(model.df, aes(y = .fitted, x = X2)) + geom_point() + geom_line()
gg = gg + facet_wrap(~ X1, labeller = label_both)
gg = gg + ggtitle('Expected Flavor of Cakes (Y) by Temperature (X2) and Baking Time (X1)')
gg = gg + ylab('Expected Flavor of Cakes (Y)') + xlab('Temperature (X2)')
gg
```

```
## geom_path: Each group consists of only one observation. Do you need to
## adjust the group aesthetic?
## geom_path: Each group consists of only one observation. Do you need to
## adjust the group aesthetic?
```



(c)

```
tidy(model)
```

```
## # A tibble: 5 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) -1695.     325.          -5.22 0.000550
## 2 X1             11.3      4.42         2.57 0.0304
## 3 X2              8.46     1.77         4.78 0.000996
## 4 I(X1^2)        -0.157    0.0632      -2.48 0.0348
## 5 I(X2^2)        -0.0120   0.00253     -4.73 0.00107
```

Formula for model: $Y = -1694.579 + 11.349 * X_1 + 8.461 * X_2 - 0.157 * X_1^2 - 0.012 * X_2^2$

$Max = -b/2a$

$Max\ X_1 = -11.349/(2 * -0.157) = 36.143$

$Max\ X_2 = -8.461/(2 * -0.012) = 352.542$

Maximum predicted palatalability happens with baking time = 36.143 minutes and temperature = 352.542 Fahrenheit.

Problem 5

(a)

```
MinnLand$log_acrePrice = log(MinnLand$acrePrice)
unique(MinnLand$region)
```

```
## [1] Northwest     Central       West Central  South Central South West
## [6] South East
## 6 Levels: Northwest West Central Central South West ... South East
```

```
#summary(MinnLand)
#scatterplotMatrix(~ log_acrePrice + crpPct + region, data=MinnLand, regLine = FALSE, smooth = FALSE)
model_a = lm(log_acrePrice ~ crpPct + region, data=MinnLand)
tidy(model_a)
```

```
## # A tibble: 7 x 5
##   term                estimate std.error statistic  p.value
##   <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)          6.88     0.00926      744.  0.
## 2 crpPct              -0.00487  0.000239     -20.4 2.33e-91
## 3 regionWest Central   0.757    0.0131        57.9 0.
## 4 regionCentral        1.05     0.0125        83.9 0.
## 5 regionSouth West     1.03     0.0141        73.4 0.
## 6 regionSouth Central  1.26     0.0138        91.2 0.
## 7 regionSouth East     1.26     0.0153        82.6 0.
```

Coefficient for crpPct = -0.005

Given any region, if crpPct (the "percentage of all farm acres enrolled in CRP") increases by one unit, we expect a decrease of 0.005 in the log of the sale price in dollars per acre, on average, in that region.

(b)

```
model_b = lm(log_acrePrice ~ crpPct + region + crpPct:region, data=MinnLand)
tidy(model_b)
```

```
## # A tibble: 12 x 5
##    term                         estimate std.error statistic  p.value
##    <chr>                           <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)                  6.88      0.00952    723.    0.
##  2 crpPct                      -0.00481   0.000312   -15.4   1.91e-53
##  3 regionWest Central           0.753     0.0138      54.8   0.
##  4 regionCentral                1.05      0.0128      81.9   0.
##  5 regionSouth West             1.04      0.0145      71.7   0.
##  6 regionSouth Central          1.26      0.0141      89.6   0.
##  7 regionSouth East             1.26      0.0156      80.9   0.
##  8 crpPct:regionWest Central    0.000713  0.000611     1.17  2.43e- 1
##  9 crpPct:regionCentral        -0.000469  0.000939    -0.499 6.18e- 1
## 10 crpPct:regionSouth West     -0.00284   0.000882    -3.22  1.29e- 3
## 11 crpPct:regionSouth Central -0.000285  0.00148     -0.192 8.48e- 1
## 12 crpPct:regionSouth East     0.00324   0.00156      2.07  3.80e- 2
```

According to this model, the relationship between crpPct and log(acrePrice) is no longer the same for all regions. In the new model, for a given region, a one unit increase in crpPct should generate a change in

log(acrePrice) based on the regions, as follows:

If region = 'Northwest' => log(acrePrice) decreases by 0.005 (-0.0048146) If region = 'West Central' => log(acrePrice) decreases by 0.004 (-0.0048146 + 0.0007133) If region = 'Central' => log(acrePrice) decreases by 0.005 (-0.0048146 - 0.0004686) If region = 'South West' => log(acrePrice) decreases by 0.007 (-0.0048146 - 0.0028393) If region = 'South Central' => log(acrePrice) decreases by 0.005 (-0.0048146) If region = 'South East' => log(acrePrice) decreases by 0.005 (-0.0048146)

(c)

```
anova(model_a, model_b)
```

```
## Analysis of Variance Table
##
## Model 1: log_acrePrice ~ crpPct + region
## Model 2: log_acrePrice ~ crpPct + region + crpPct:region
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1  18693 5581.7
## 2  18688 5576.2  5    5.4969 3.6844 0.002469 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

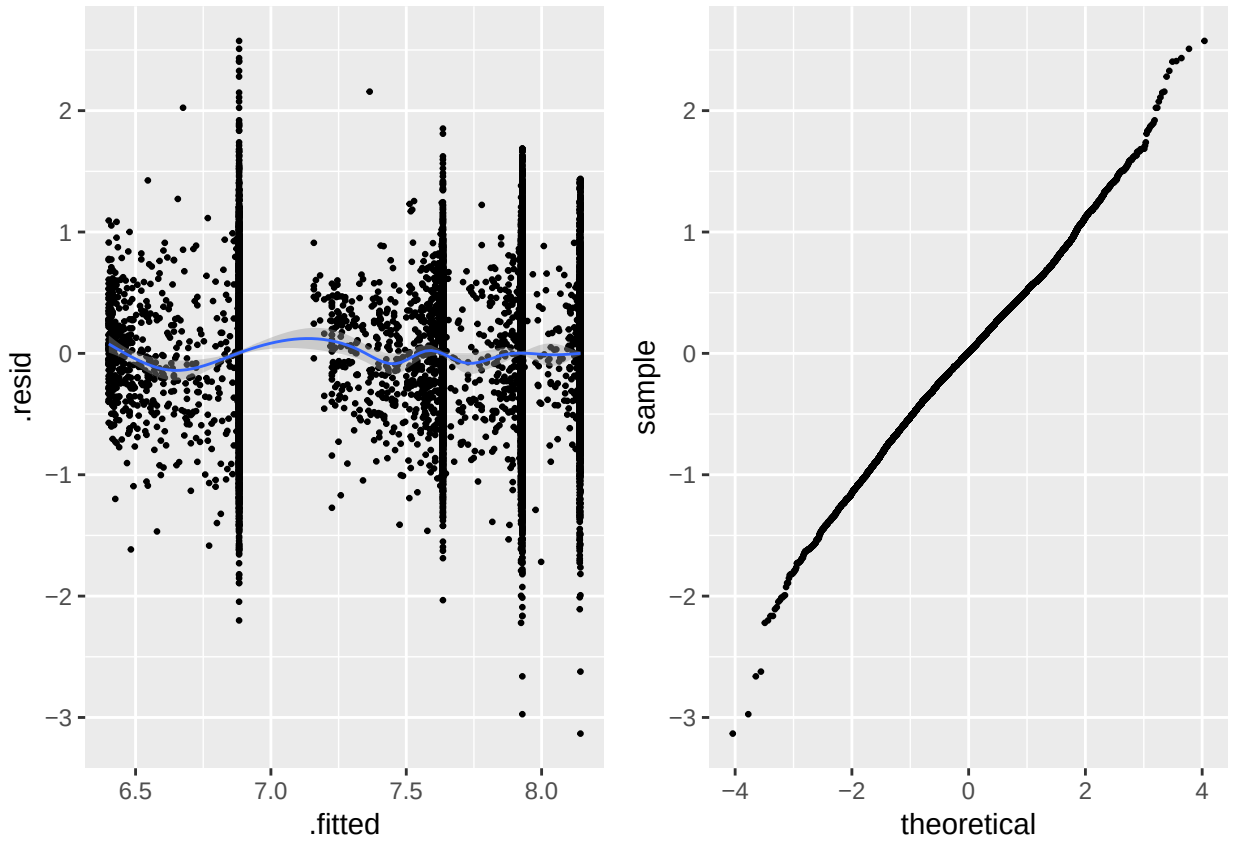
We test for the following: H0 : The reduced model (first model) defines the relationship between response and regressors H1 : The full model (second model) defines the relationship between response and regressors

P-value = 0.002469, therefore we have evidence to reject the null hypotheses that the interaction term from the second model has not effect in the model, for alpha = 5%. In other words, there is statistical evidence to support the conclusion that adding the interaction term to the model, helps explain changes in the response.

(d)

```
model_b.df = augment(model_b)
g1 = ggplot(model_b.df, aes(x = .fitted, y = .resid)) + geom_point(size=0.5) + geom_smooth(size=0.5)
g2 = ggplot(model_b.df, aes(x = .fitted, y = .resid)) + geom_point(size=0.1) + geom_smooth(size=0.5)
g2 = g2 + facet_wrap(~region, 3)
g3= ggplot(model_b.df, aes(sample = .resid)) + stat_qq(size=0.5)
g4= ggplot(model_b.df, aes(sample = .resid)) + stat_qq(size=0.1) + facet_wrap(~region, 3)
#ggarrange(g1, g2, g3, g4, nrow = 2, ncol = 2)
ggarrange(g1, g3, nrow = 1, ncol = 2)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
#
ncvTest(model_b)
```

```
## Non-constant Variance Score Test
## Variance formula: ~fitted.values
## Chisquare = 198.1959, Df = 1, p = < 2.22e-16
```

We learned from the class lectures that homoskedasticity is the main assumption of analysis of variance. Technically, the test of nonconstant variance ncvTest() shows evidence of heteroskedasticity, since the p-value very close to zero measn significant evidence that we can reject the null that the distribution of residuals is homoskedastic.

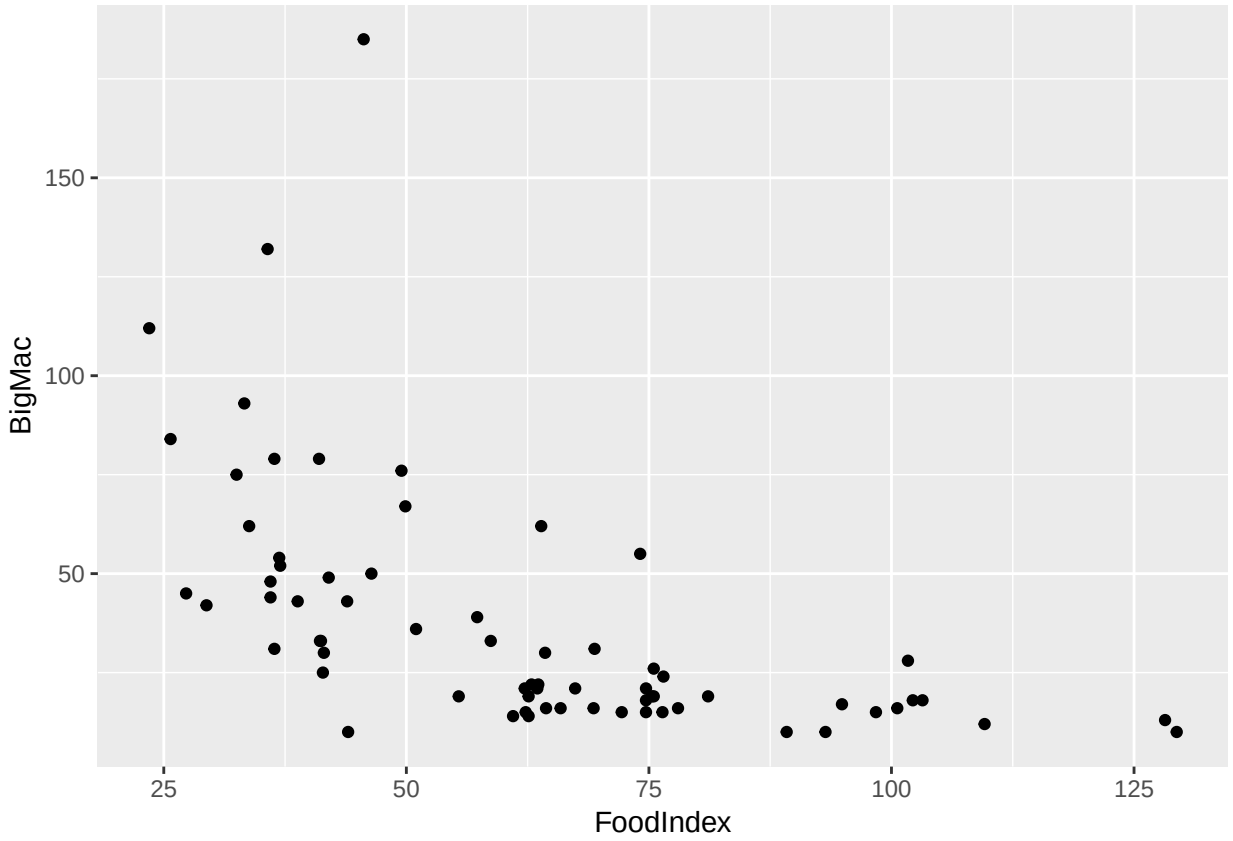
However, even though the loess smother is a bit curvy in the residuals plot on the left, even though the variance of residulas seem to increase as fitted values increase, and even though the qq plot shows residuals are not perfectly normally distributed (see extreme values), we believe the assumptions for ANOVA are close to satisfied.

Problem 6

(a)

We first look at the plot with no transformation.

```
ggplot(BigMac2003, aes(x=FoodIndex, y=BigMac)) + geom_point()
```



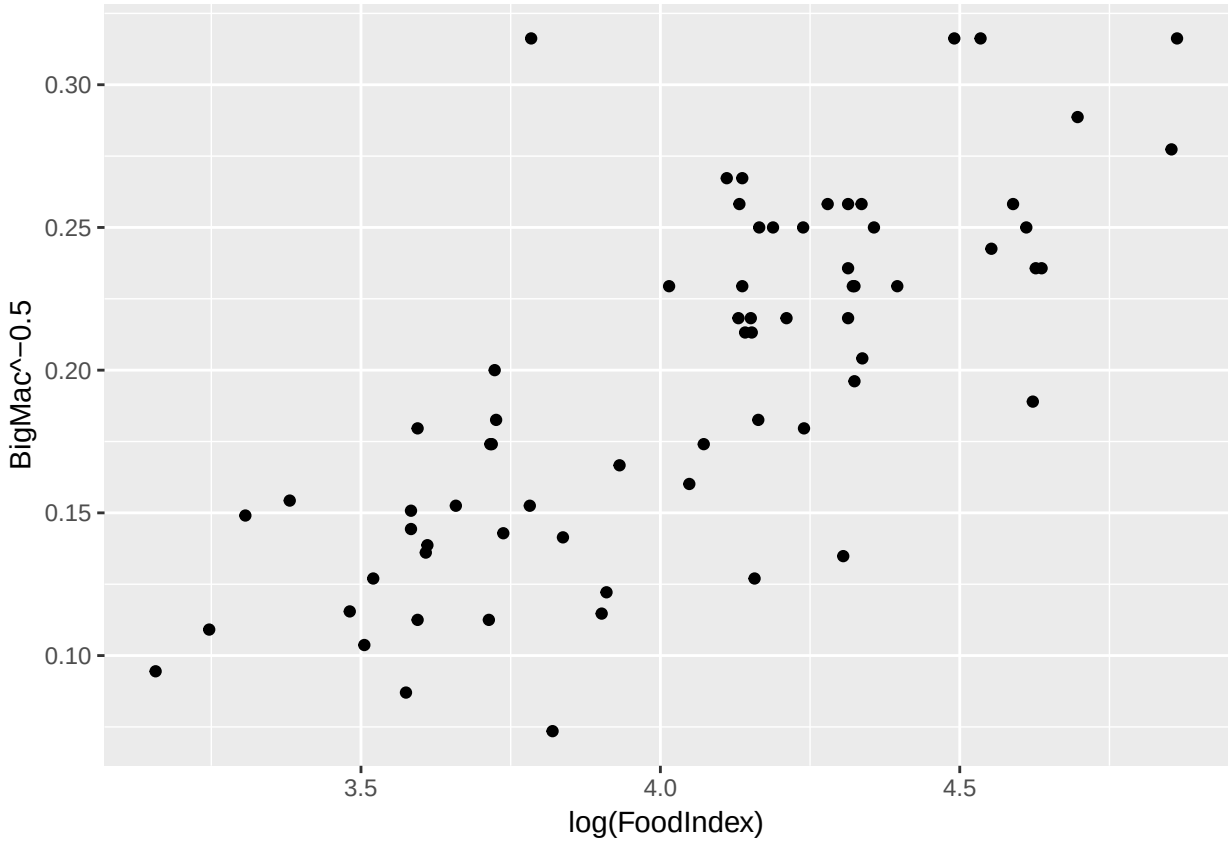
We'll use the powerTransform function to find lambda for the Box-Cox power family transformations

```
summary(powerTransform(cbind(BigMac, FoodIndex) ~ 1, BigMac2003))
```

```
## bcPower Transformations to Multinormality
##           Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## BigMac      -0.3803        -0.5      -0.6714      -0.0892
## FoodIndex    0.0851         0.0      -0.3958       0.5661
##
## Likelihood ratio test that transformation parameters are equal to 0
##  (all log transformations)
##                               LRT df     pval
## LR test, lambda = (0 0) 6.905702  2 0.031655
##
## Likelihood ratio test that no transformations are needed
##                               LRT df       pval
## LR test, lambda = (1 1) 106.9817  2 < 2.22e-16
```

Lambda for BigMac is -0.5, and for FoodIndex = 0, or log.

```
ggplot(BigMac2003, aes(x=log(FoodIndex), y=BigMac^-0.5)) + geom_point()
```



So, the transformations are: BigMac -> 1/sqrt(BigMac) (response) FoodIndex -> log(FoodIndex) (explanatory variable)

The resulting plot of the relationship of the variables after these transformations shows linearity.

(b)

First, I'll find transformations for all variables (removed TaxRate because of negative values)

```
summary(powerTransform(cbind(FoodIndex, Bread, Rice, Bus, Apt, TeachGI, TeachNI, TeachHours) ~ 1, BigMa
```

```
## bcPower Transformations to Multinormality
##             Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## FoodIndex      0.0864        0.00      -0.3295       0.5022
## Bread         -0.1237        0.00      -0.3989       0.1515
## Rice          -0.2268        0.00      -0.4896       0.0360
## Bus            0.1739        0.00      -0.0297       0.3775
## Apt            0.3726        0.50       0.1286       0.6165
## TeachGI        0.1611        0.16       0.0459       0.2763
## TeachNI        0.1380        0.14       0.0202       0.2559
## TeachHours     1.4661        1.00       0.5780       2.3543
##
## Likelihood ratio test that transformation parameters are equal to 0
##  (all log transformations)
##                                           LRT df       pval
## LR test, lambda = (0 0 0 0 0 0 0 0) 31.07999  8 0.00013597
```

```
##
## Likelihood ratio test that no transformations are needed
##                                          LRT df       pval
## LR test, lambda = (1 1 1 1 1 1 1 1) 398.1536  8 < 2.22e-16
```

I will try to improve the model by considering all the other variables, with recommended transformation.

```
BigMac2003$BigMac_trans = BigMac2003$BigMac^-0.5
BigMac2003.subsets = regsubsets(BigMac_trans ~ log(FoodIndex) + log(FoodIndex):Bread + log(FoodIndex):R
summary(BigMac2003.subsets)
```

```
## Subset selection object
## Call: regsubsets.formula(BigMac_trans ~ log(FoodIndex) + log(FoodIndex):Bread +
##     log(FoodIndex):Rice + log(FoodIndex):Bus + log(FoodIndex):Apt +
##     log(FoodIndex):TeachGI + log(FoodIndex):TeachNI + log(FoodIndex):TeachHours,
##     data = BigMac2003)
## 8 Variables  (and intercept)
##                            Forced in Forced out
## log(FoodIndex)                 FALSE      FALSE
## log(FoodIndex):Bread           FALSE      FALSE
## log(FoodIndex):Rice            FALSE      FALSE
## log(FoodIndex):Bus             FALSE      FALSE
## log(FoodIndex):Apt             FALSE      FALSE
## log(FoodIndex):TeachGI         FALSE      FALSE
## log(FoodIndex):TeachNI         FALSE      FALSE
## log(FoodIndex):TeachHours      FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          log(FoodIndex) log(FoodIndex):Bread log(FoodIndex):Rice
## 1  ( 1 ) " "            " "                  " "
## 2  ( 1 ) "*"            "*"                  " "
## 3  ( 1 ) "*"            "*"                  "*"
## 4  ( 1 ) "*"            "*"                  "*"
## 5  ( 1 ) "*"            "*"                  "*"
## 6  ( 1 ) "*"            "*"                  "*"
## 7  ( 1 ) "*"            "*"                  "*"
## 8  ( 1 ) "*"            "*"                  "*"
##          log(FoodIndex):Bus log(FoodIndex):Apt log(FoodIndex):TeachGI
## 1  ( 1 ) " "                " "                "*"
## 2  ( 1 ) " "                " "                " "
## 3  ( 1 ) " "                " "                " "
## 4  ( 1 ) " "                " "                "*"
## 5  ( 1 ) "*"                " "                " "
## 6  ( 1 ) "*"                " "                " "
## 7  ( 1 ) "*"                "*"                " "
## 8  ( 1 ) "*"                "*"                "*"
##          log(FoodIndex):TeachNI log(FoodIndex):TeachHours
## 1  ( 1 ) " "                    " "
## 2  ( 1 ) " "                    " "
## 3  ( 1 ) " "                    " "
## 4  ( 1 ) " "                    " "
## 5  ( 1 ) "*"                    " "
## 6  ( 1 ) "*"                    "*"
## 7  ( 1 ) "*"                    "*"
## 8  ( 1 ) "*"                    "*"
```

```r
# Fitting one model with log(FoodIndex)
model_1 = lm(BigMac_trans ~ log(FoodIndex), data = BigMac2003)
# adjusted r-squared
summary(model_1)$adj.r.squared
```

```
## [1] 0.5576697
```

```r
# AIC
AIC(model_1)
```

```
## [1] -240.3644
```

```r
# Fitting one model with log(FoodIndex) and interaction variable log(FoodIndex):Bread
model_2 = lm(BigMac_trans ~ log(FoodIndex) + log(FoodIndex):Bread, data = BigMac2003)
# adjusted r-squared
summary(model_2)$adj.r.squared
```

```
## [1] 0.7099342
```

```r
# AIC
AIC(model_2)
```

```
## [1] -268.5165
```

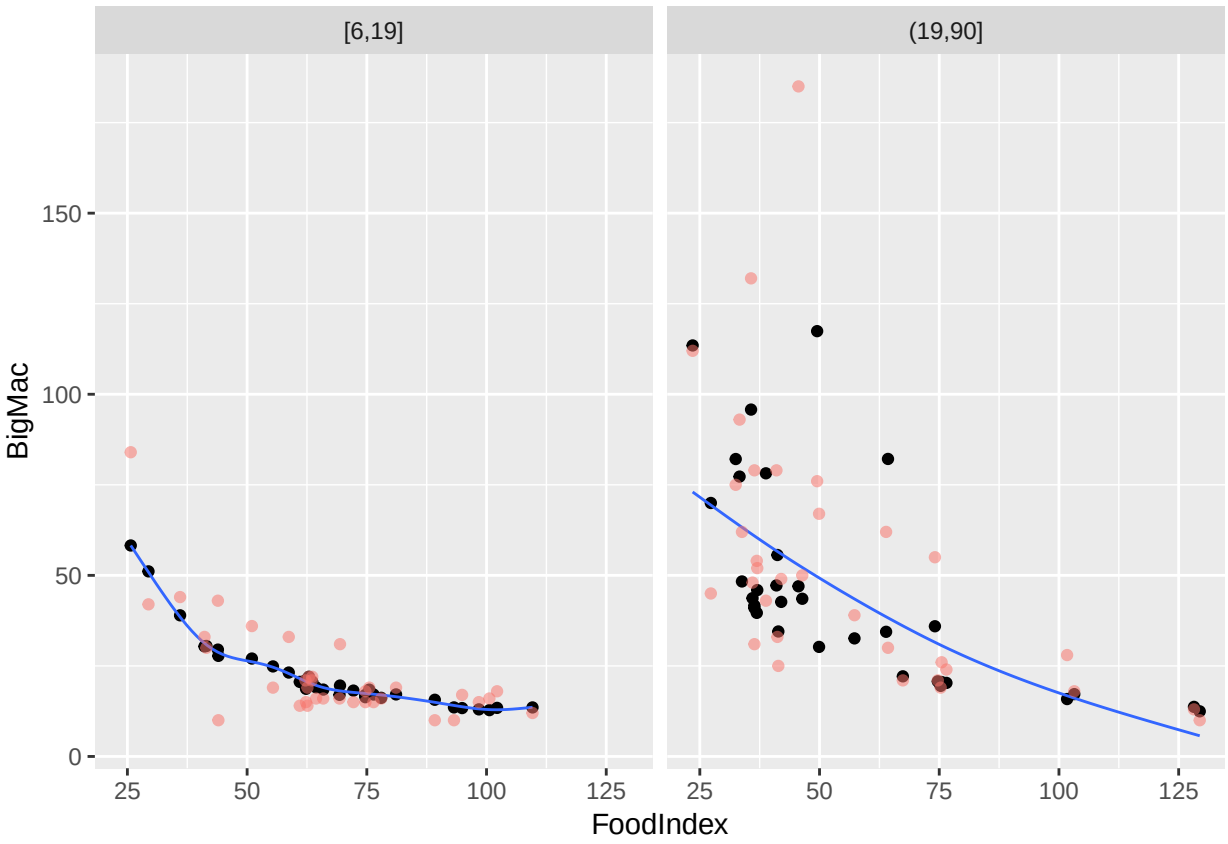
The second model has adjusted R-squared = 0.7099, which is better than the first (R-squared for the first is 0.5577). Additionally, the Akaike information criterion for the second model is better, confirming the second model with the interaction term is indeed the better model.

(c)

```r
tidy(model_2)
```

```
## # A tibble: 3 x 5
##   term                  estimate std.error statistic  p.value
##   <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)          -0.199    0.0425        -4.69 1.44e- 5
## 2 log(FoodIndex)        0.106    0.0101        10.5  1.10e-15
## 3 log(FoodIndex):Bread -0.000355 0.0000591     -6.01 8.76e- 8
```

```r
model_2.df = augment(model_2)
gg1 = ggplot(model_2.df, aes(x = exp(log.FoodIndex.), y = (.fitted)^-2))
gg1 = gg1 + geom_point() + geom_smooth(method='gam', formula = y ~ s(x), se=F, size=0.5)
gg1 = gg1 + facet_wrap(~ cut_number(Bread, n=2))
gg1 = gg1 + xlab("FoodIndex") + ylab("BigMac")
gg2 = gg1 + geom_point(aes(y=(BigMac_trans)^-2, col='red', alpha=0.1), show.legend = FALSE)
#gg1
gg2
```



The plot of fitted values (black dots are fitted values, red dots are observed values) shows that for bread prices below the median of bread prices, BigMac decreases fast for FoodIndex < 50, and then decreases slowly until it plateaus around FoodIndex values greater than 90. For values of bread price above the median of bread prices, we notice much higher prices of BigMac for FoodIndex values between 25 and 75 and a less sharp decrease in BigMac pricess as FoodIndex increases. BigMac prices decrease significantly when FoodIndex is higher than 75. Additionally, we note that high bread prices (above median) in conjunction with high FoodIndex (greater than 100) are associated with low BigMac prices.

The formula for the final model is: 1/sqrt(BigMac) = -0.1991 + 0.1063 * log(FoodIndex) - 0.0004 * log(FoodIndex) * Bread => BigMac = (-0.1991 + 0.1063 * log(FoodIndex) - 0.0004 * log(FoodIndex) * Bread) ^ -2 => BigMac = (-0.1991 + log(FoodIndex) * (0.1063 - 0.004 * Bread)) ^ -2

$$BigMac = \frac{1}{[-0.1991 + log(FoodIndex) * (0.1063 - 0.004 * Bread)]^2}$$

Problem 7

```
start.time = Sys.time()
run.time.min = 5

all.vars = c('Sex','WT2','HT2','WT9','HT9','LG9','ST9')

# create interaction variables
for (i in 1:length(all.vars)) {
  x = t(combn(all.vars,i))
  for (j in 1:dim(x)[1]) {
    if (i == 1) {
      comb.vars = gsub(", ", ":", toString(x[j,]))
    } else {
      comb.vars = c(comb.vars, gsub(", ", ":", toString(x[j,])))
    }
  }
}
all.vars = c(all.vars, comb.vars)
print(length(all.vars))
```

```
## [1] 128
```

```
please_break = FALSE
for (i in 1:length(all.vars)) {
  x = t(combn(all.vars,i))
  for (j in 1:dim(x)[1]) {
    if (difftime(Sys.time(), start.time, units='mins') > run.time.min) {
      please_break = TRUE
      break
    }
    expl.vars = gsub(",", "+", toString(x[j,]))
    aic = AIC(lm(formula(paste('HT18 ~' , expl.vars)), data=BGSall))
    if (aic < 708.1) {
      print(expl.vars)
      print(aic)
    }
  }
  if (please_break) {
    break
  }
}
```

```
## [1] "HT9+ Sex:WT9+ WT9:HT9:ST9"
## [1] 707.9812
## [1] "HT9+ Sex:WT9+ HT2:WT9:HT9:ST9"
## [1] 708.0956
## [1] "HT9+ WT9:HT9+ Sex:HT9:LG9"
## [1] 708.0664
```

We'll use model with HT9, Sex:WT9, and WT9:HT9:ST9, which yielded AIC = 707.9812

```
model = lm(HT18 ~ HT9+ Sex:WT9+ WT9:HT9:ST9, data=BGSall)
summary(model)
```

```
##
```

```
## Call:
## lm(formula = HT18 ~ HT9 + Sex:WT9 + WT9:HT9:ST9, data = BGSall)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9497 -1.8761 -0.0289  1.8398  7.7710
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.575e+01  8.861e+00   1.777   0.0778 .
## HT9          1.216e+00  6.982e-02  17.413   <2e-16 ***
## Sex:WT9     -3.679e-01  1.690e-02 -21.768   <2e-16 ***
## HT9:WT9:ST9 -6.673e-06  3.000e-06  -2.224   0.0278 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.197 on 132 degrees of freedom
## Multiple R-squared:  0.8723, Adjusted R-squared:  0.8694
## F-statistic: 300.5 on 3 and 132 DF,  p-value: < 2.2e-16
```

```
BGSall.sample = sample_n(BGSall, 500, replace = T)
preds = predict(model, newdata = BGSall.sample, interval = "prediction", level=0.9)
apply(preds, MARGIN=2, mean)
```

```
##      fit      lwr      upr
## 172.8938 167.5287 178.2589
```

I generated a sample of 500 observations from the original dataset. The predictions from the model average 172.35, with a 90% confidence interval for the prediction in the range [166.9833, 177.7190]. So, the prediction errors for individuals similar to those in this datase, would be, for this confidence level, plus or minus 5.3678kg on average.

We could also use the residuals from the model to estimate expected error for predictions in similar population. Residuals range from -7.9497 to 7.7710, so we should expect that on average, prediction errors should be no higher than plus of minus 8kg.