Modeling Social Data: Homework 2

Anshul Doshi

Anshul Doshi (ad3222) Modeling Social Data Assignment: 2

Problem 1: Modeling Scenarios

For the following, indicate whether you'd expect the performance of a flexible statistical model to perform better than an inflexible one.

To start this off, it is important to know what the difference between a flexible and inflexible statiscal learning method is. The basic paradigm we are working in is that we have some training data that we want to use to estimate f, some unknown function that demonstrates the relationship between input variables (predictors) and the response variables.

In trying to estimate f, we train on this observed data by applying some statistical learning method. This method would be flexible if our method can fit many different possible functional forms for f and inflexible otherwise. For example, a parametric model where you assume the functional form of f, for instance, to be linear, would be considered inflexible.

a.) The sample size n is extremely large, and the number of predictors p is small.

Better. Flexible methods generally need size n to be large as they fit the data closer to the true f so they can extract more information with a larger n. Moreover, the tradeoff that flexible methods ususally experience in which they overfit does not really come into play here as the number of predictors is small and we have a lot of samples to work with.

b.) The number of predictors p is extremely large, and the number of observations n is small.

Worse. Here, the problem of overfitting does come into play since we don't have as many oberservations and by using the large of number of predictors we are likely to follow the errors or noise too closely. Thus, an inflexible model where we reduce the problem of estimating f to estimating a set of parameters is preferred.

c.) The relationship between the predictors and response is highly non-linear.

Better. Since the true f is highly non-linear, our inflexible parametric methods that make assumptions of the functional form of f are probably going to be way off. Flexible methods will work better here as they are able to fit many possible forms of f.

d.) The variance of the error terms, i.e. Var(e), is extremely high.

Worse. High variance in errors indicates a lot of noise in the sample data. We know that flexible methods are susceptible to overfitting to this noise so we'd prefer to use an inflexible method.

Problem 2: Cross-validation for Polynomial Regression

```r
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
## Conflicts with tidy packages ----------------------------------------------
## filter(): dplyr, stats
## lag():    dplyr, stats
```

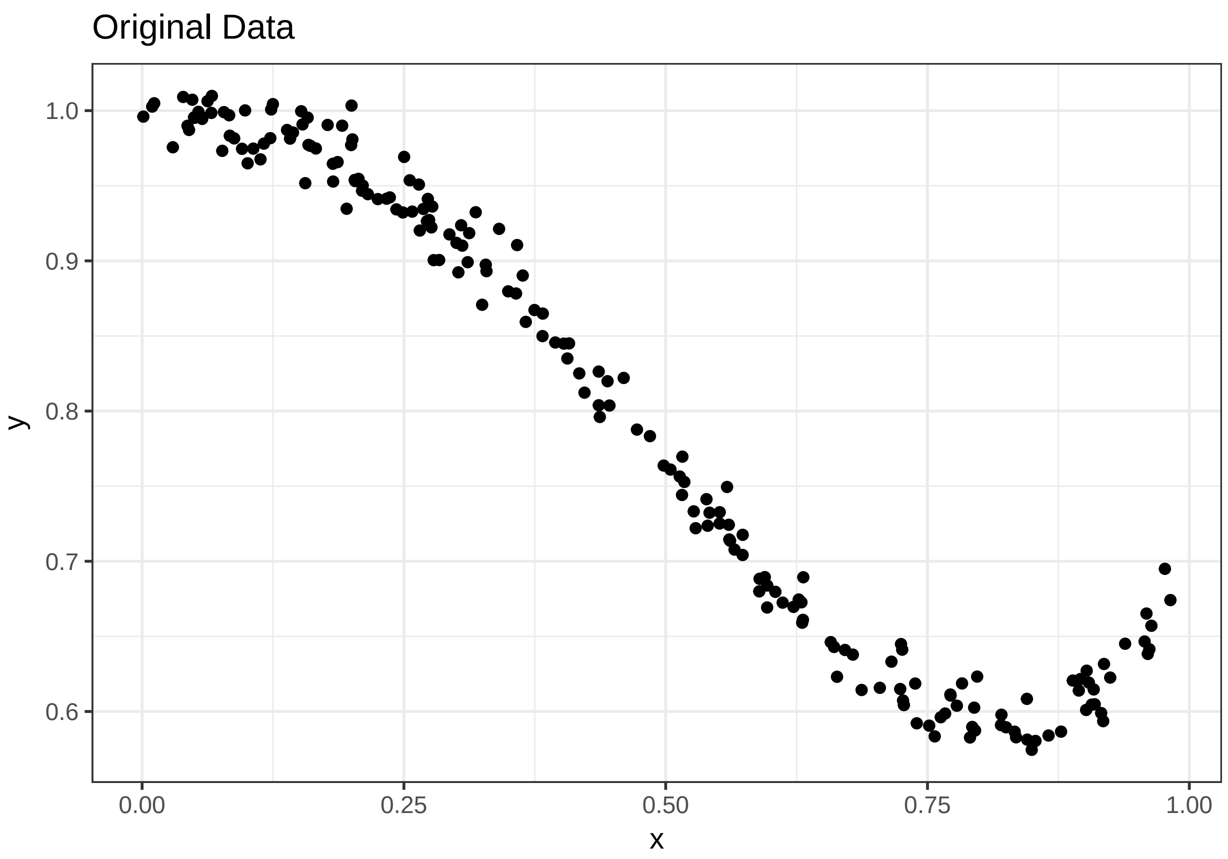
```r
library(modelr)

library(scales)
```

```
##
## Attaching package: 'scales'
## The following object is masked from 'package:purrr':
##
##     discard
## The following objects are masked from 'package:readr':
##
##     col_factor, col_numeric
```

```r
options(na.action = na.warn)

theme_set(theme_bw())
options(repr.plot.width=4, repr.plot.height=3)

setwd("~/Desktop/Spring2017/Modeling Social Data/HW2_ad3222")

# read data
data <- read_tsv('polyfit.tsv')
```

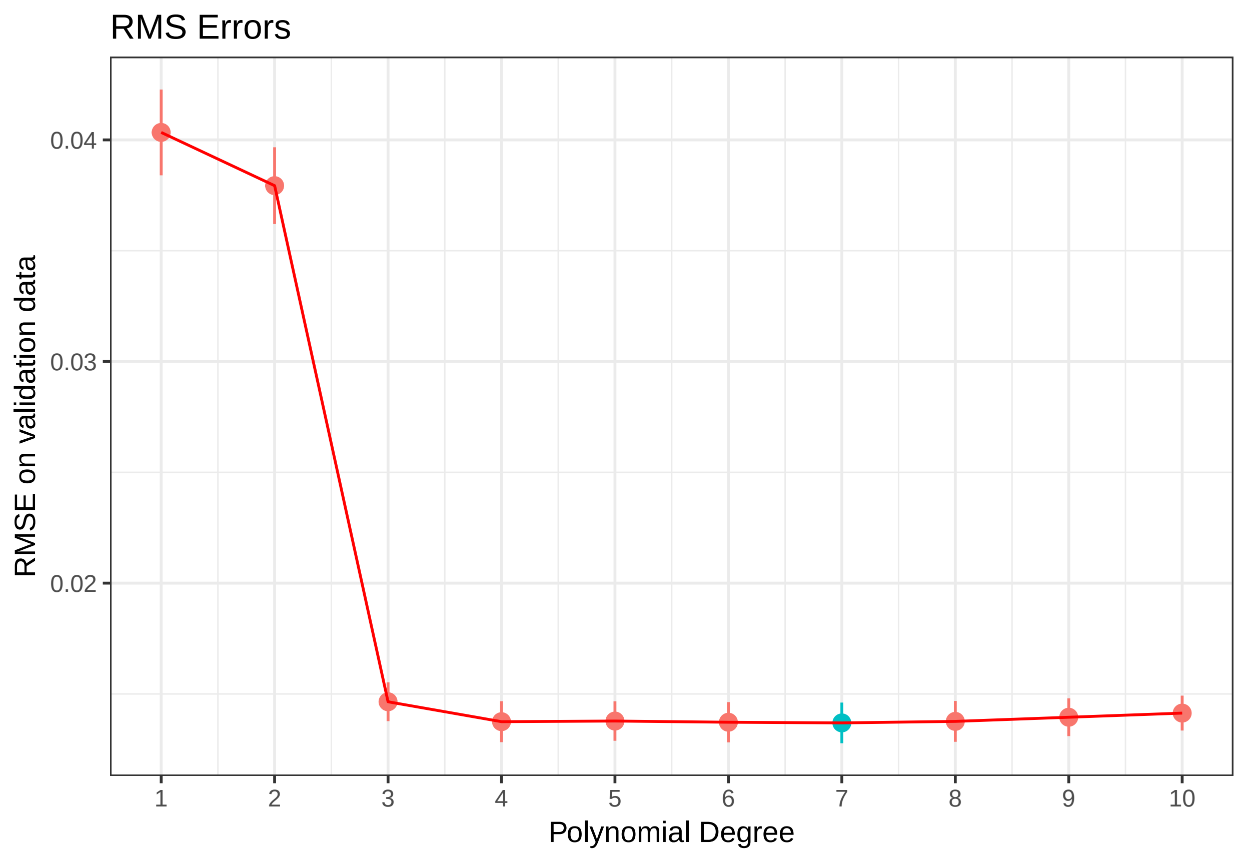
```
## Parsed with column specification:
## cols(
##   x = col_double(),
##   y = col_double()
## )
```

```r
#preliminary look at the data
ggplot(data, aes(x, y)) +
  geom_point() +
  ggtitle("Original Data")
```

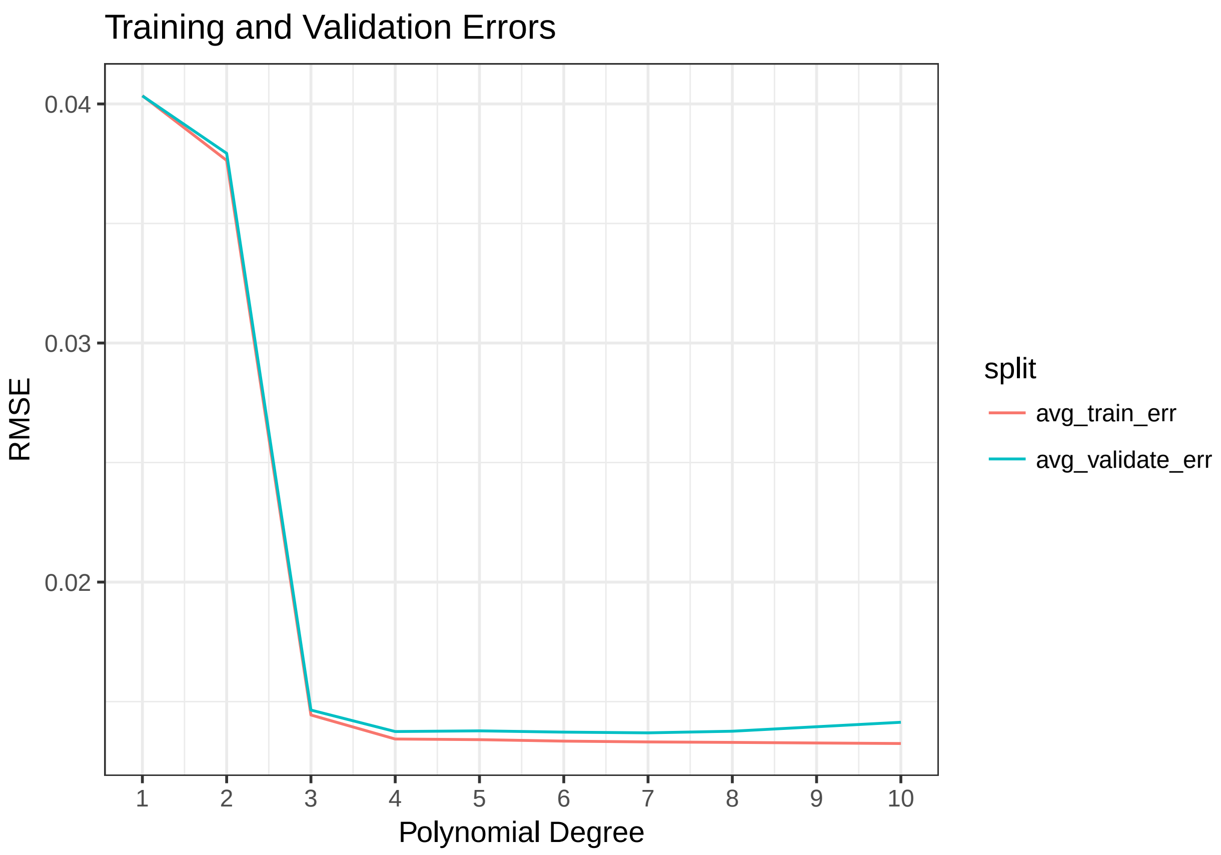


```r
#Relationship looks non-linear

# shuffle the data and assign each row to one of 5 different folds
# make sure to use this seed
set.seed(4990)

num_folds <- 5
num_row <- nrow(data)
frac_train <- .8
num_train <- floor(num_row*frac_train)

ndx <- sample(1:num_row, num_row, replace=F)

data <- data[ndx, ] %>%
  mutate(fold = (row_number() %% num_folds) + 1)

#Check to see if folds are even
# data %>%
#   group_by(fold) %>%
#   summarize(count=n())
# ungroup(data)

# implement 5-fold cross-validation to compute the average train / test error (RMSE) for each polynomia
K <- 1:10
avg_train_err <- c()
avg_validate_err <- c()
se_validate_err <- c()
```

```r
for(k in K) {
  train_err <- c()
  validate_err <- c()
  for (f in 1:num_folds) {
    # fit on the training data
    data_train <- filter(data, fold != f)
    model <- lm(y ~ poly(x, k, raw = T), data=data_train)
    train_err[f] <- sqrt(mean((predict(model, data_train) - data_train$y)^2))

    # evaluate on the validation data
    data_validate <- filter(data, fold == f)
    validate_err[f] <- sqrt(mean((predict(model, data_validate) - data_validate$y)^2))
  }
  # compute the average validation error across folds
  # and the standard error on this estimate
  avg_train_err[k] <- mean(train_err)
  avg_validate_err[k] <- mean(validate_err)
  se_validate_err[k] <- sd(validate_err) / sqrt(num_folds)
}

# plot the validate error, highlighting the value of k with the lowest average error
plot_data <- data.frame(K, avg_validate_err, se_validate_err)
ggplot(plot_data, aes(x=K, y=avg_validate_err)) +
  geom_pointrange(aes(ymin=avg_validate_err - se_validate_err,
                      ymax=avg_validate_err + se_validate_err,
                      color=avg_validate_err == min(avg_validate_err))) +
  geom_line(color = "red") +
  scale_x_continuous(breaks=1:12) +
  theme(legend.position="none") +
  xlab('Polynomial Degree') +
  ylab('RMSE on validation data') +
  ggtitle("RMS Errors")
```



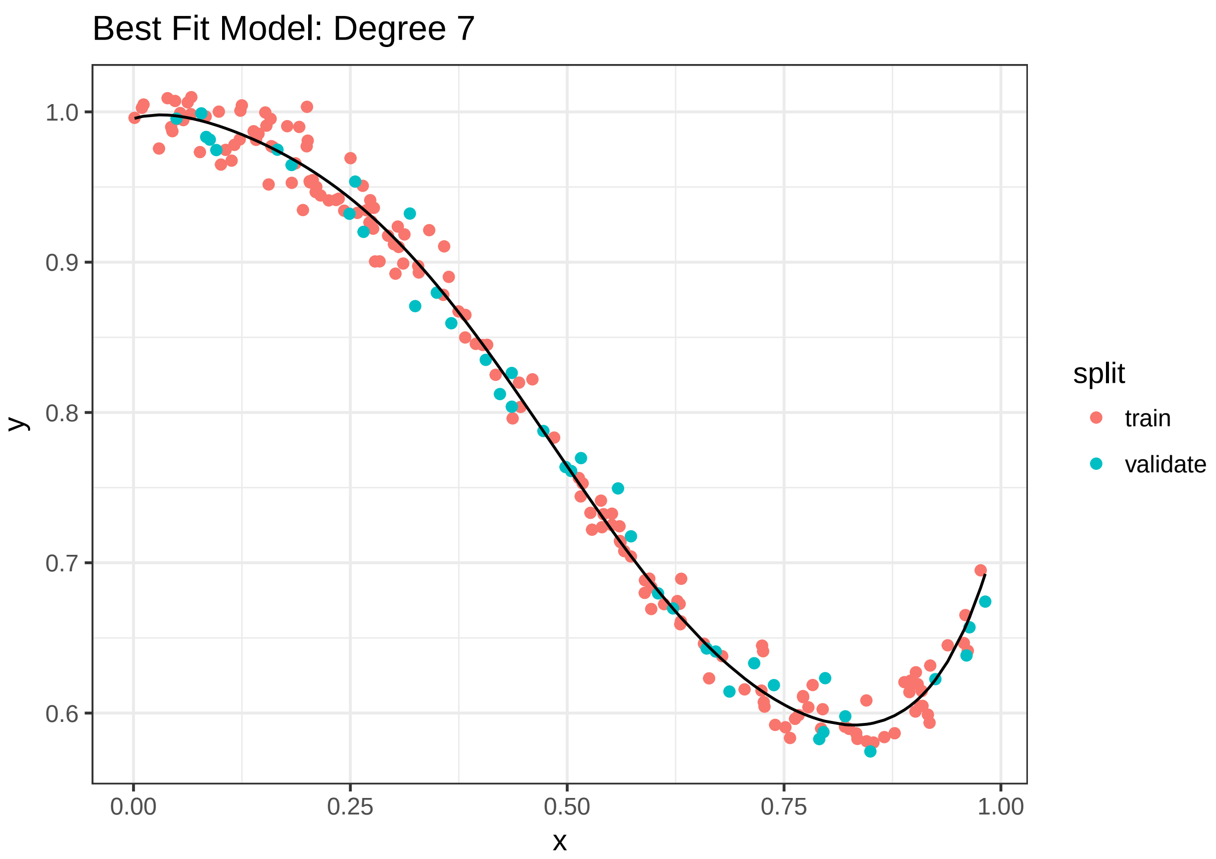
```r
# make a plot showing the how the train and test error vary with the polynomial degree
plot_data <- data.frame(K, avg_train_err, avg_validate_err) %>%
  gather("split", "error", -K)

ggplot(plot_data, aes(x=K, y=error, color=split)) +
  geom_line() +
  scale_x_continuous(breaks=K) +
  xlab('Polynomial Degree') +
  ylab('RMSE') +
  ggtitle("Training and Validation Errors")
```



```r
# select the degree with the lowest average test error
min_deg <- which.min(avg_validate_err)
cat("The minimum degree is:", min_deg)
```

```
## The minimum degree is: 7
```

```r
# fit a final model for this degree on all of the data
ndx <- sample(1:num_row, num_train, replace=F)
data_train <- data[ndx, ]
data_validate <- data[-ndx, ]
model <- lm(y ~ poly(x, min_deg, raw=T), data=data_train)

data_train <- data_train %>%
  add_predictions(model) %>%
  mutate(split = "train")
data_validate <- data_validate %>%
  add_predictions(model) %>%
  mutate(split = "validate")
plot_data <- bind_rows(data_train, data_validate)

# make a scatter plot of all of the data with the best-fit model overlayed as a line
ggplot(plot_data, aes(x, y)) +
  geom_point(aes(color = split)) +
  geom_line(aes(y = pred)) +
  xlab('x') +
  ylab('y') +
  scale_y_continuous() +
  ggtitle("Best Fit Model: Degree 7")
```



```r
# report the coefficients for the best-fit model
model
```

```
##
## Call:
## lm(formula = y ~ poly(x, min_deg, raw = T), data = data_train)
##
## Coefficients:
##                (Intercept)  poly(x, min_deg, raw = T)1
##                     0.9955                      0.1781
## poly(x, min_deg, raw = T)2  poly(x, min_deg, raw = T)3
##                    -3.6279                     18.5906
## poly(x, min_deg, raw = T)4  poly(x, min_deg, raw = T)5
##                   -62.9636                    106.3440
## poly(x, min_deg, raw = T)6  poly(x, min_deg, raw = T)7
##                   -85.2303                     26.4429
```

Problem 3: Sales Data

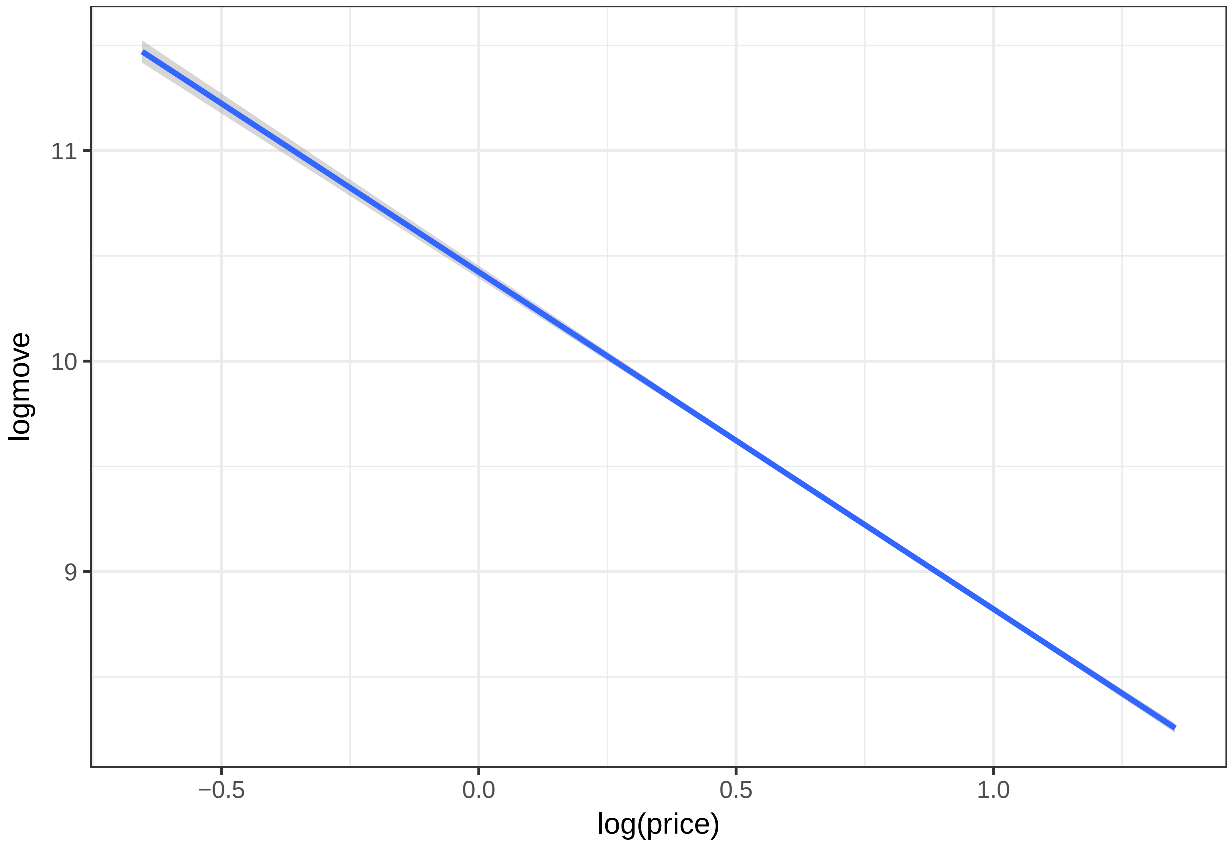
Part 1: Load oj.csv data

```r
library(tidyverse)
library(modelr)

setwd("~/Desktop/Spring2017/Modeling Social Data/HW2_ad3222")
```

```r
# load the oj data
oj <- read_csv('oj.csv')
```

```
## Parsed with column specification:
## cols(
##   store = col_integer(),
##   brand = col_character(),
##   week = col_integer(),
##   logmove = col_double(),
##   feat = col_integer(),
##   price = col_double(),
##   AGE60 = col_double(),
##   EDUC = col_double(),
##   ETHNIC = col_double(),
##   INCOME = col_double(),
##   HHLARGE = col_double(),
##   WORKWOM = col_double(),
##   HVAL150 = col_double(),
##   SSTRDIST = col_double(),
##   SSTRVOL = col_double(),
##   CPDIST5 = col_double(),
##   CPWVOL5 = col_double()
## )
```

```r
ggplot(oj, aes(x = log(price), y = logmove)) +
  geom_smooth(method = "lm")
```



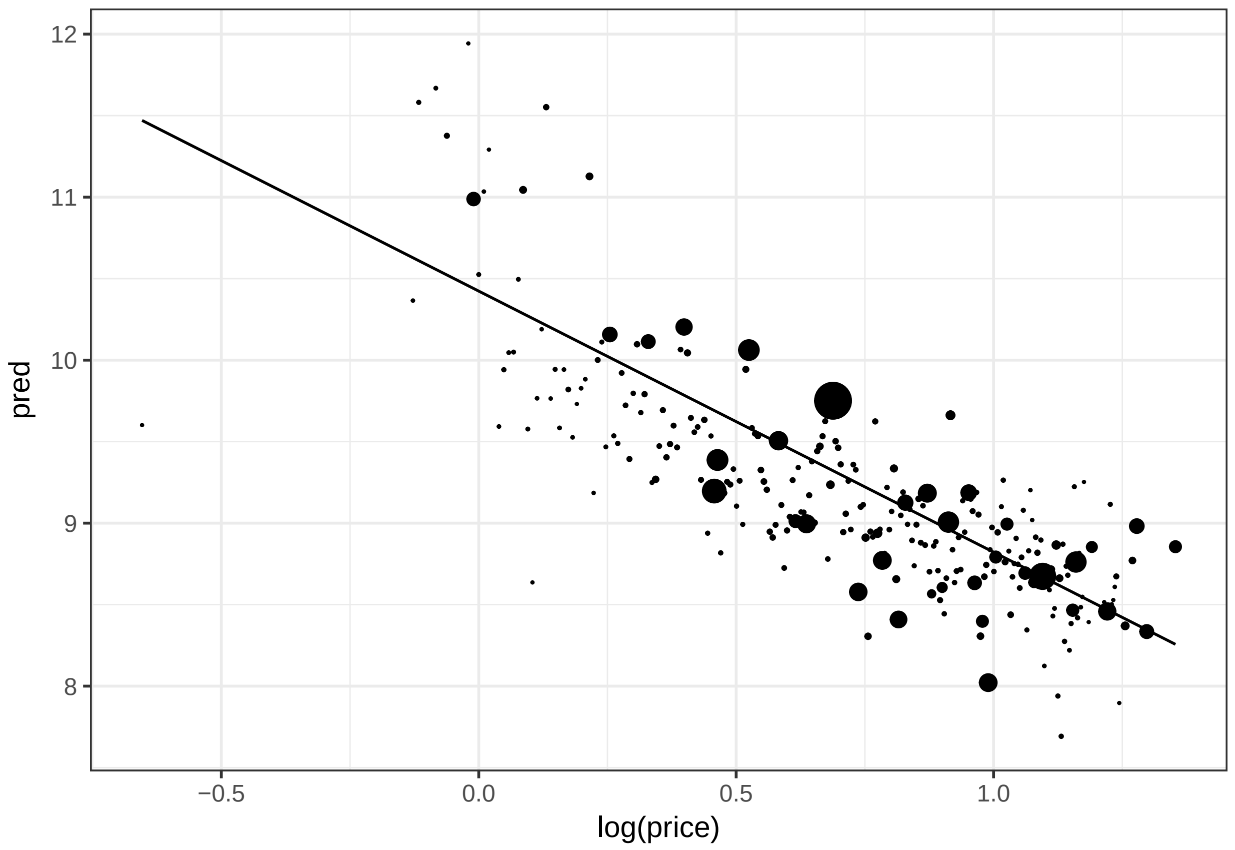
Part 2

Do regression on logmove and log(prices)

```r
########################################
# regress logmove on log(price)
########################################
model_data <- oj[, c("logmove", "price")]

#Lets do this with a log-log regression

model <- lm(logmove ~ log(price), model_data)
model
```

```
##
## Call:
## lm(formula = logmove ~ log(price), data = model_data)
##
## Coefficients:
## (Intercept)   log(price)
##      10.423       -1.601
```

```r
plot_data <- model_data %>%
  group_by(price) %>%
  summarize(count = n(),
            mean_logmov = mean(logmove)) %>%
  add_predictions(model) %>%
  mutate(pred = pred)

ggplot(plot_data, aes(x = log(price), y = pred)) +
  geom_line(aes(y = pred)) +
  geom_point(aes(y = mean_logmov, size = count)) +
  scale_size_area(guide = F)
```



a.) How much variance is explained?

```
pred_actual <- model_data %>%
  add_predictions(model) %>%
  mutate(actual = logmove)

pred_actual %>%
  summarize(rmse = sqrt(mean((pred - actual)^2)),
            cor = cor(pred,actual),
            cor_sq = cor^2)
```

```
## # A tibble: 1 × 3
##        rmse       cor    cor_sq
##       <dbl>     <dbl>     <dbl>
## 1 0.9070931 0.4562244 0.2081407
```

Variance is about 20% explained.

b.) What is the elasticity?

Since we have done log-log regression, the elasticity will just be the coefficent β_1

```
model$coefficients[2]
```

```
## log(price)
##  -1.601307
```

c.) Based on the model, if we were to increase the price at which we sell orange juice by 10%, by what percentage should we expect sales to drop?

Since elasticity is -1.6, increasing price would decrease sales by:

$$10 * 1.6e^{-1.6log(1.1)} = 13.7 \approx 14\%$$

Part 3

Run a new regression with interactions with brand

```
##########################################
# regress logmove on log(price) with brand interactions
##########################################
model_data <- oj[, c("logmove", "price", "brand")]
form <- as.formula(logmove ~ brand * (log(price)))
M <- model.matrix(form, model_data)

model <- lm(form, model_data)
```

a.) Report Elasticity for each Brand

We need to find slope of each line using the coefficients:

```
model
```

```
##
## Call:
## lm(formula = form, data = model_data)
##
## Coefficients:
##                  (Intercept)              brandminute.maid
##                     10.95468                       0.88825
##               brandtropicana                    log(price)
##                      0.96239                      -3.37753
## brandminute.maid:log(price)    brandtropicana:log(price)
##                      0.05679                       0.66576
```

Dominicks:

$$y = 10.95 - 3.377x \rightarrow -3.377$$

Minute Maid:

$$y = (10.95 + .888) + (-3.377 + .056)x \rightarrow -3.32$$

Tropicana:

$$y = (10.95 + .962) + (-3.377 + .666)x \rightarrow -2.711$$

b.) Compare above model with part 2

Compared to the model above, the elasticites we have here are more negative and are over double for two brands. This means that slight changes in price have a larger effect on sales

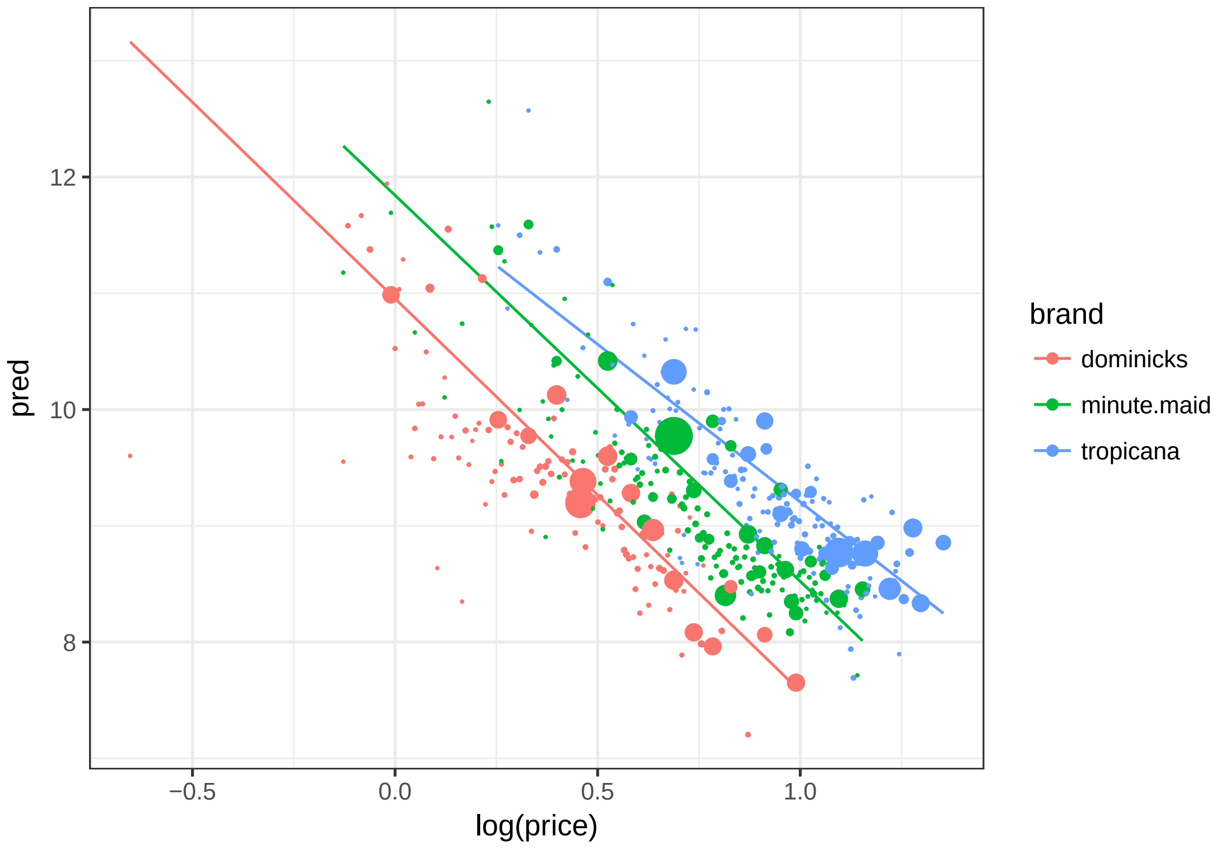
c.) Which brand is least sensitive to changes in price? What happens after a 10 increase on price with this model?

Under this model, tropicana is the least sensitive to change. If we increase price by 10% then tropicana sales would decrease by

$$10 * 2.7e^{-2.7log(1.1)} = 20.8 \approx 21\%$$

d.) Plot logmove vs log(price)

```r
plot_data <- model_data %>%
  group_by(price, brand) %>%
  summarize(count = n(),
            mean_logmov = mean(logmove)) %>%
  add_predictions(model) %>%
  mutate(pred = pred)

ggplot(plot_data, aes(x = log(price), y = pred, color = brand)) +
  geom_line(aes(y = pred)) +
  geom_point(aes(y = mean_logmov, size = count)) +
  scale_size_area(guide = F)
```



Part 4

Run a final regression that interacts log(price), brand, and feat (i.e., whether the product was featured in the store).

a.) Make a plot showing the fitted model with log(price) on the horizontal axis, logmove on the vertical, a different color line and panel for each brand, and a different line type for featured vs. non-featured products.

```r
##########################################
# regress logmove on log(price) interacted with brand and feat
##########################################
```

```
model_data <- oj[, c("logmove", "price", "brand", "feat")]
df <- data.frame(feat = c(0,1), new_feat = c("not featured", "featured"))
model_data <- left_join(model_data, df)
```

```
## Joining, by = "feat"
```

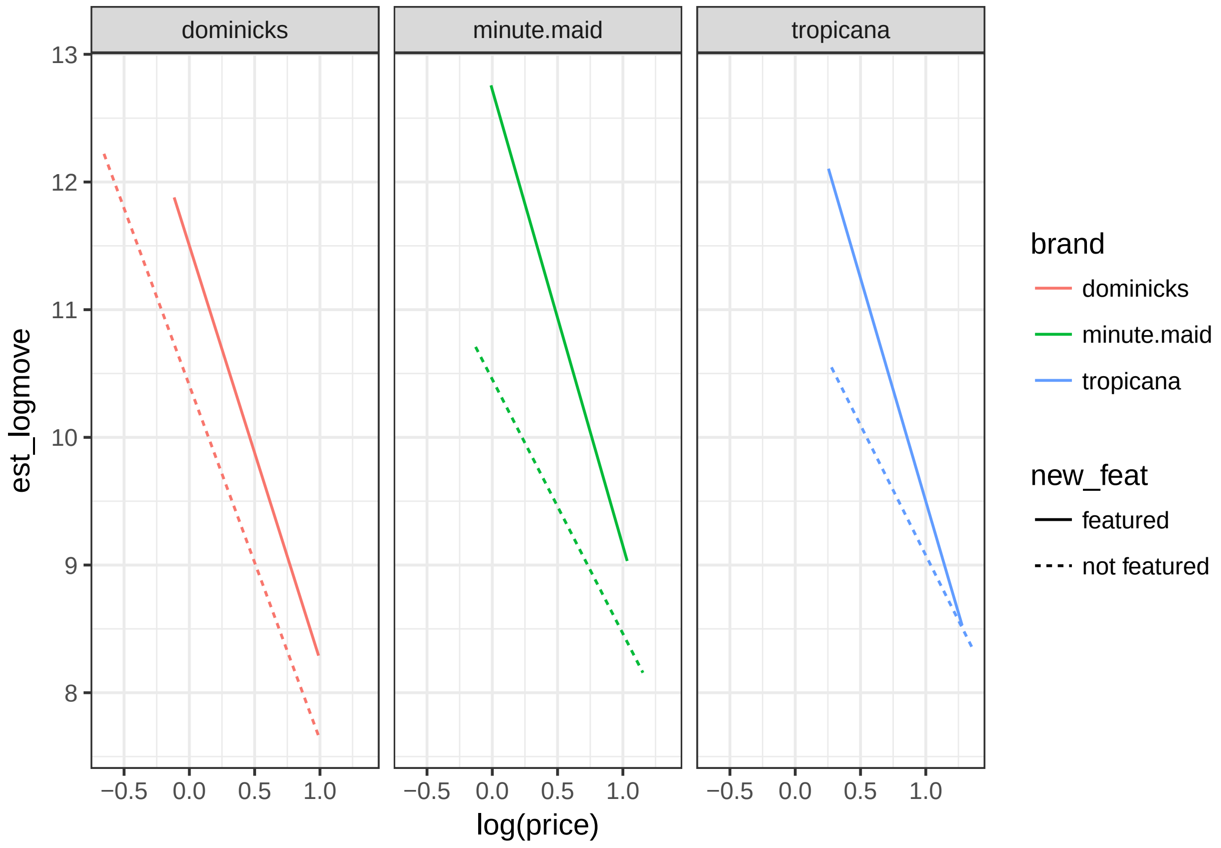
```
model_data
```

```
## # A tibble: 28,947 × 5
##     logmove price     brand  feat     new_feat
##       <dbl> <dbl>     <chr> <dbl>       <fctr>
## 1  9.018695  3.87 tropicana     0 not featured
## 2  8.723231  3.87 tropicana     0 not featured
## 3  8.253228  3.87 tropicana     0 not featured
## 4  8.987197  3.87 tropicana     0 not featured
## 5  9.093357  3.87 tropicana     0 not featured
## 6  8.877382  3.87 tropicana     0 not featured
## 7  9.294682  3.29 tropicana     0 not featured
## 8  8.954674  3.29 tropicana     0 not featured
## 9  9.049232  3.29 tropicana     0 not featured
## 10 8.613230  3.29 tropicana     0 not featured
## # ... with 28,937 more rows
```

```
form <- as.formula(logmove ~ brand * new_feat * log(price))
model <- lm(form, model_data)
model
```

```
##
## Call:
## lm(formula = form, data = model_data)
##
## Coefficients:
##                                      (Intercept)
##                                          11.5010
##                                brandminute.maid
##                                           1.2201
##                                  brandtropicana
##                                           1.4932
##                            new_featnot featured
##                                          -1.0944
##                                       log(price)
##                                          -3.2447
##            brandminute.maid:new_featnot featured
##                                          -1.1729
##              brandtropicana:new_featnot featured
##                                          -0.7853
##                      brandminute.maid:log(price)
##                                          -0.3263
##                        brandtropicana:log(price)
##                                          -0.2503
##                  new_featnot featured:log(price)
##                                           0.4706
## brandminute.maid:new_featnot featured:log(price)
##                                           1.1092
##   brandtropicana:new_featnot featured:log(price)
##                                                        0.9861
```

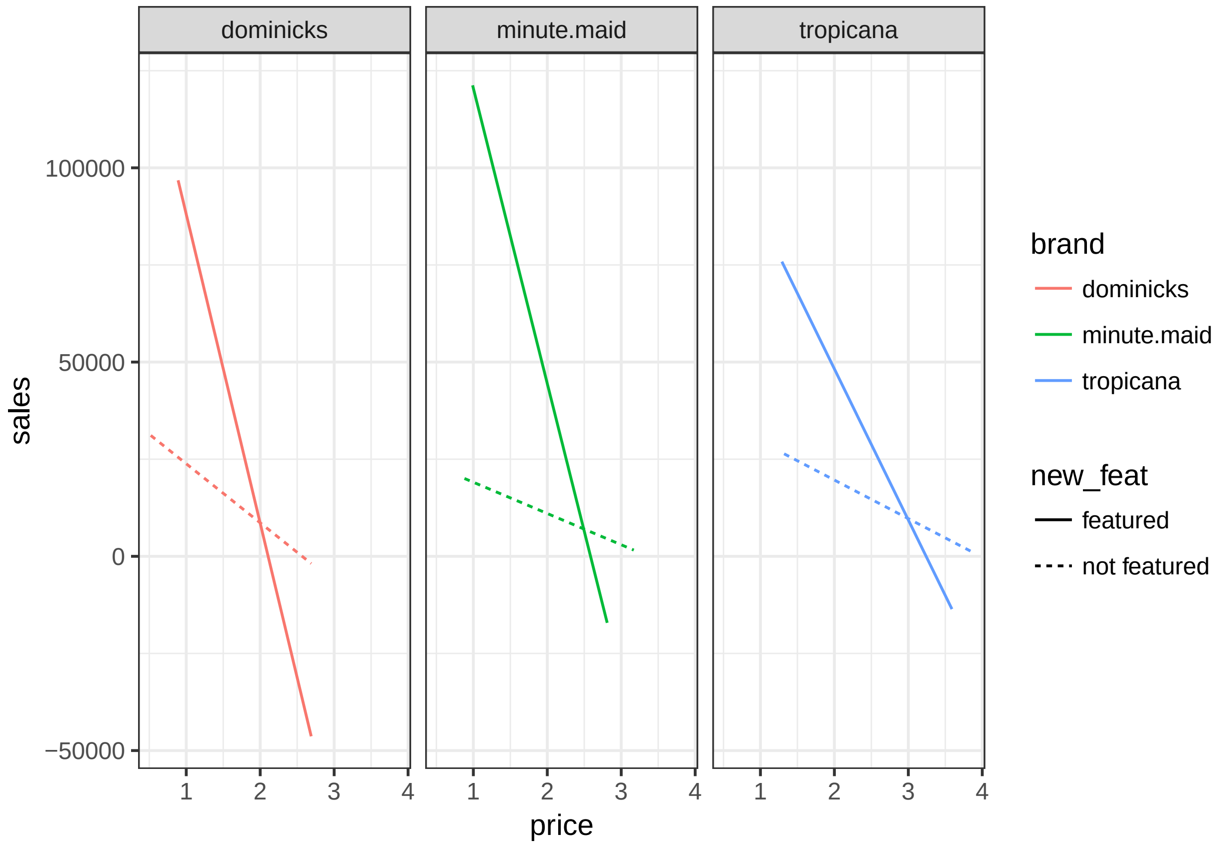
```
plot_data <- model_data %>%
  group_by(price, brand, new_feat) %>%
  summarize(count = n(),
            mean_logmov = mean(logmove)) %>%
  add_predictions(model) %>%
  mutate(est_logmove = pred)

ggplot(plot_data, aes(x = log(price), y = est_logmove, color = brand, linetype = new_feat)) +
  geom_line(aes(y = est_logmove)) +
  #geom_point(aes(y = mean_logmov, size = count)) +
  facet_wrap(~ brand)
```



b.) Repeat this plot, but transform the axes from log to linear space (using price and $e^{logmove}$ on the horizontal and vertical axes, respectively) for easier interpretation.

```
#Transofrm the axis
form <- as.formula(exp(logmove) ~ brand * new_feat * price)
model <- lm(form, model_data)

plot_data <- model_data %>%
  group_by(price, brand, new_feat) %>%
  summarize(count = n(),
            mean_logmov = mean(logmove)) %>%
  add_predictions(model) %>%
  mutate(sales = pred)
```

```r
ggplot(plot_data, aes(x = price, y = sales, color = brand, linetype = new_feat)) +
  geom_line(aes(y = sales)) +
  #geom_point(aes(y = mean_logmov, size = count)) +
  facet_wrap(~ brand)
```



c.) What are average sales of Tropicana when it's priced at 1.50 but not featured in the store?

```r
my_point <- data.frame(brand=c("tropicana"), new_feat=c("not featured"), price=c(1.50))
avg_tnf <- predict(model, my_point)
avg_tnf
```

```
##        1
## 24595.89
```

d.) Imagine we wanted to sell this same amount of Tropicana, but were willing to feature it in the store. How much could we charge?

```r
my_point <- data.frame(brand=c("tropicana"), new_feat=c("featured"),
                               price=model_data$price)
sales_vals <- predict(model, my_point)
my_price <- which.min(abs(sales_vals-avg_tnf))
model_data$price[my_price]
```

```
## [1] 2.61
```