Assignment 5

Max Troeger

2025-03-28

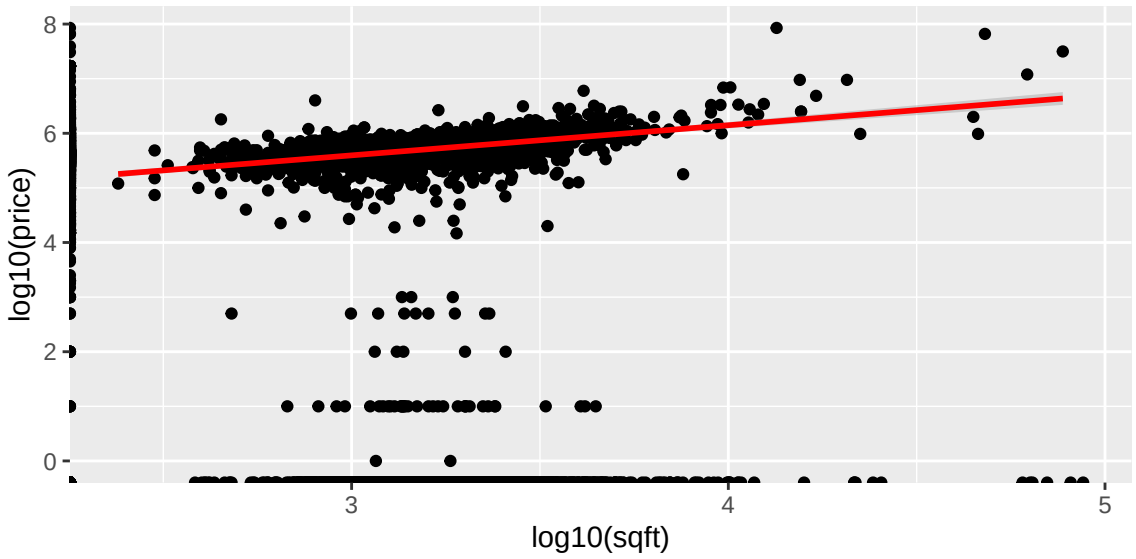
(1) Derived Dataset - Staten Island

(1a)

For the first derived dataset I chose Staten Island. I'm looking for the interaction among the gross square footage, total number of units, and sale price terms. Because many sale prices are 0 or are ridiculously high/low, I'll need to constrain the prices to eliminate outliers for the whole dataset. ## (1b) Now we need to eliminate the outliers present here, before doing so we demonstrate their predominance:

```
statenisland <- subset(nycdata,BOROUGH=="STATEN ISLAND")
statenisland$sqft <- as.numeric(statenisland$`GROSS SQUARE FEET`)
statenisland$price <- as.numeric(statenisland$`SALE PRICE`)


ggplot(statenisland, aes(x = log10(sqft), y = log10(price))) +
  geom_point() +
  stat_smooth(method = "lm",color="red")
```

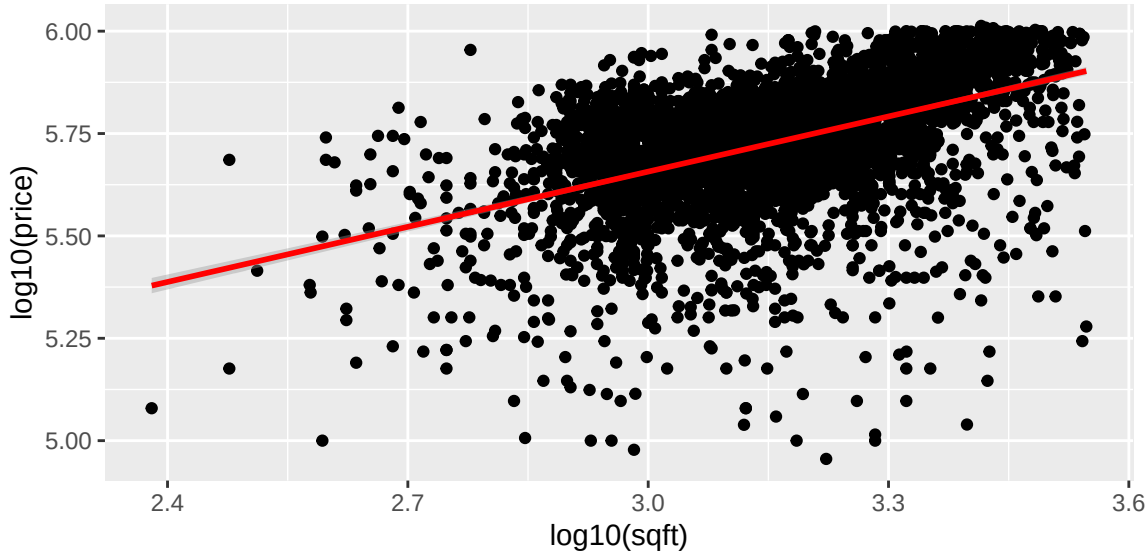


Clearly we have price and square footage outliers. The procedure outlined below is as follows: remove clearly erroneous data points (i.e., where the square footage or price are 0) and then remove data points that fall outside of the first and fourth quartiles of the Staten Island data set. Since we do not want to fit "noise as signal" we only care about the internal structure of the data, which is what permits us to cut off the tails in this way. We now remove the outliers:

```
statenisland <- subset(statenisland,sqft!=0)
quartiles <- quantile(statenisland$sqft, probs=c(.25, .75), na.rm = TRUE)
IQR <- IQR(statenisland$sqft)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
statenisland <- subset(statenisland, statenisland$sqft > Lower &
                                     statenisland$sqft < Upper)


statenisland <- subset(statenisland,price!=0)
quartiles <- quantile(statenisland$price, probs=c(.25, .75), na.rm = TRUE)
IQR <- IQR(statenisland$price)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
statenisland <- subset(statenisland, statenisland$price > Lower &
                                     statenisland$price < Upper)
ggplot(statenisland, aes(x = log10(sqft), y = log10(price))) +
```

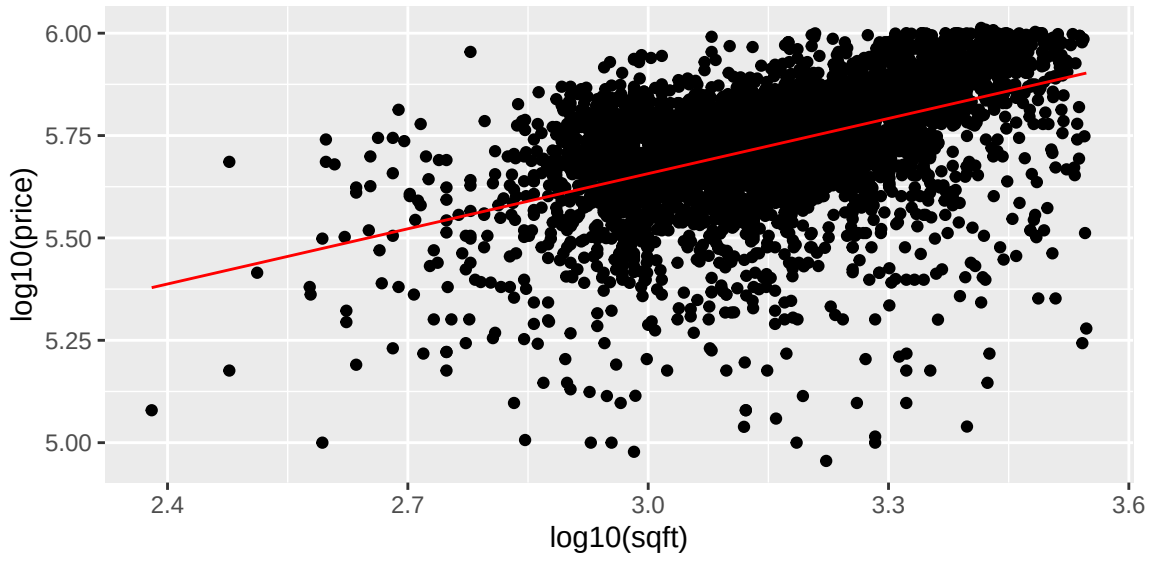
```
  geom_point() +
  stat_smooth(method = "lm",color="red")
```



(1c)

We now attempt to fit a linear model to `log10(price)`. Although we suspect `log10(sqft)` would give a strong model, we only realize $\bar{R}^2 = 0.24$ with $F = 1401$.

```
fit_initial <- lm(log10(price)~log10(sqft),statenisland)
pred <- predict(fit_initial,statenisland)
pred_initial <- data.frame(orig=log10(statenisland$sqft),pred=pred)
ggplot(statenisland, aes(x = log10(sqft), y = log10(price))) +
  geom_point() +
  geom_line(color='red',data = pred_initial, aes(x=orig,y=pred))
```



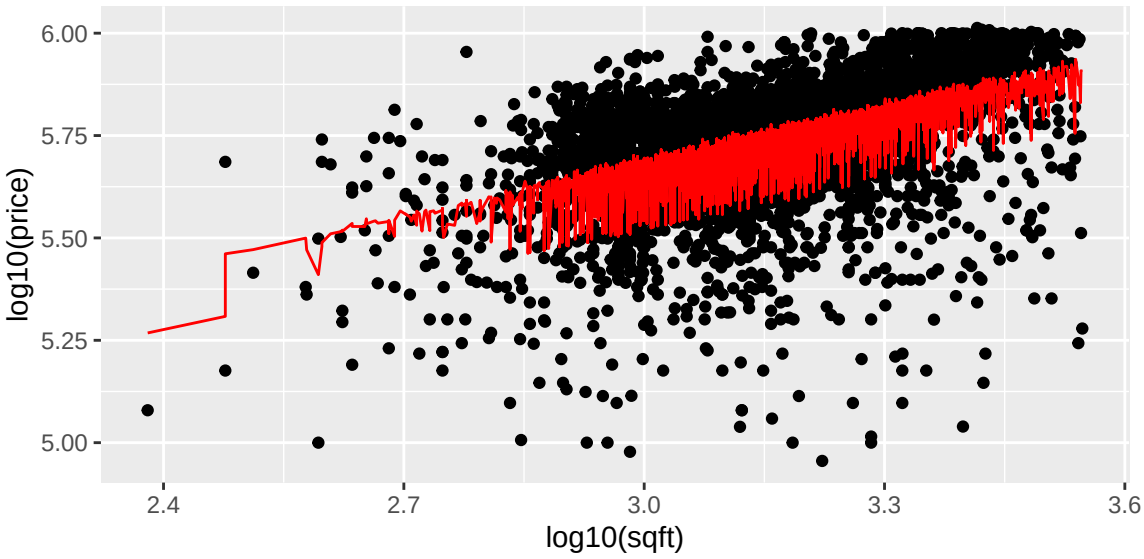

```
# summary(fit_initial)
```

For better fit, we construct a model by regressing `log10(price)` on `log10(sqft)` as before, but with the addition of dummy variables and other variables. We then run backwards selection on this model to enhance parsimony, arriving at $\bar{R}^2 = 0.30$ with $F = 137.9$:

```
fit <- lm(log10(price)~log10(sqft)
                      +as.factor(`ZIP CODE`)
                      +`YEAR BUILT`
                      +`TOTAL UNITS`
                      ,data=statenisland) %>%  stepAIC(direction='backward',trace = FALSE)
summary(fit)$adj.r.squared
```

```
## [1] 0.299785
```

```
pred <- predict(fit,statenisland)
pred_aic <- data.frame(orig=log10(statenisland$sqft),pred=pred)
ggplot(statenisland, aes(x = log10(sqft), y = log10(price))) +
  geom_point() +
  geom_line(color='red',data = pred_aic, aes(x=orig,y=pred))
```




```
# summary(fit)

# produces long output
```

This is only a marginal improvement, but including other variables would significantly overfit the data (i.e. including `BLOCK` as a factor).

(1d)

We now train KNN, Random Forest, and SVM models on the Staten Island data: ### (1di) KNN By trial and error we arrive at a model of the specification listed below. Through several trials we identify $k = 1$ as giving the strongest testing accuracy. In order to convert `NEIGHBORHOOD` into an appropriate factor, we first make it into a type `factor` and convert it into a unique number. This is necessary because `knn` will fail to process the dataset otherwise. From our trimmed dataset we also omit all `NA`s for the same reason. From a confusion matrix, we arrive at the following testing accuracy:

```
set.seed(1)

statenisland_trim <- statenisland %>% select(price,sqft)

statenisland_trim$neighborhood <- as.numeric(factor(statenisland$NEIGHBORHOOD))

statenisland_trim$zip <- as.factor(statenisland$`ZIP CODE`)

#statenisland_trim <- statenisland %>% select(!BOROUGH) %>%

#                                      select(!`EASE-MENT`) %>%

#                                      select(!`APARTMENT NUMBER`) %>%

#                                      select(!`Census Tract 2020`) %>%
```

```
#                                       select(!`NTA Code`) %>%
#                                       select(!Latitude) %>%
#                                       select(!Longitude) %>%
#                                       select(!ADDRESS) %>%
#                                       select(!NTA) %>%
#                                       select(!`BUILDING CLASS CATEGORY`) %>%
#                                       select(!`SALE DATE`) %>%
#                                       select(!`SALE PRICE`) %>%
#                                       select(!`NEIGHBORHOOD`)
statenisland_trim <- statenisland_trim %>% na.omit
indices=c(1 :nrow(statenisland_trim))
train_pct =.7
train_size = floor(nrow(statenisland_trim) * train_pct)
train_idx <- sample(indices,train_size)
train = rep(FALSE ,nrow(statenisland_trim))
for(i  in  train_idx){
    train[i]= TRUE
}
train_data <- statenisland_trim[train,]
test_data <- statenisland_trim[!train,]
test_price = as.matrix(statenisland_trim$price[!train])
train_price = as.matrix(statenisland_trim$price[train ])
pred.knn <- knn(train_data,test_data,train_price,k =1)
c_matrix <- table(pred.knn,statenisland_trim$price[!train])
# testing accuracy
100*sum(diag(c_matrix))/sum(c_matrix)
```

```
## [1] 0.1488095
```

(1dii) Random Forest

We now use a random forest to see if we can arrive at a better fit:

```
fit.rf= randomForest(price~sqft,data=train_data)
pred  <-  predict(fit.rf,test_data)
probs  <-  pred
pred.rf <- rep(0,length(probs))
pred.rf[probs>0.5] <- 1
c_matrix <- table(pred.rf, test_price)
100*sum(diag(c_matrix))/sum(c_matrix)
```

```
## [1] 0.07440476
```

The forecasting accuracy of the random forest method is considerably worse than KNN.

(1diii) SVM

We now try a support vector machine:

```
fit.svm <- svm(price ~ sqft, data=train_data, kernel = 'radial')
pred.svm  <-  predict(fit.svm,test_data)
c_matrix <- table(pred.svm, test_price)
100*sum(diag(c_matrix))/sum(c_matrix)
```

```
## [1] 0.3720238
```

The forecasting accuracy of the SVM is leaps and bounds better than both the KNN and random forest on the Staten Island dataset.

(2) Derived Dataset - Manhattan

For the second derived dataset I chose Manhattan. Although units in Manhattan include entries of the form `BOROUGH=["1","MANHATTAN"]`, choosing just the former gives a sufficiently large sample size for our purposes (~96,000 observations). Again, I'm looking for the interaction among the gross square footage, total number of units, and sale price terms with the same comment about outliers.

(2a)

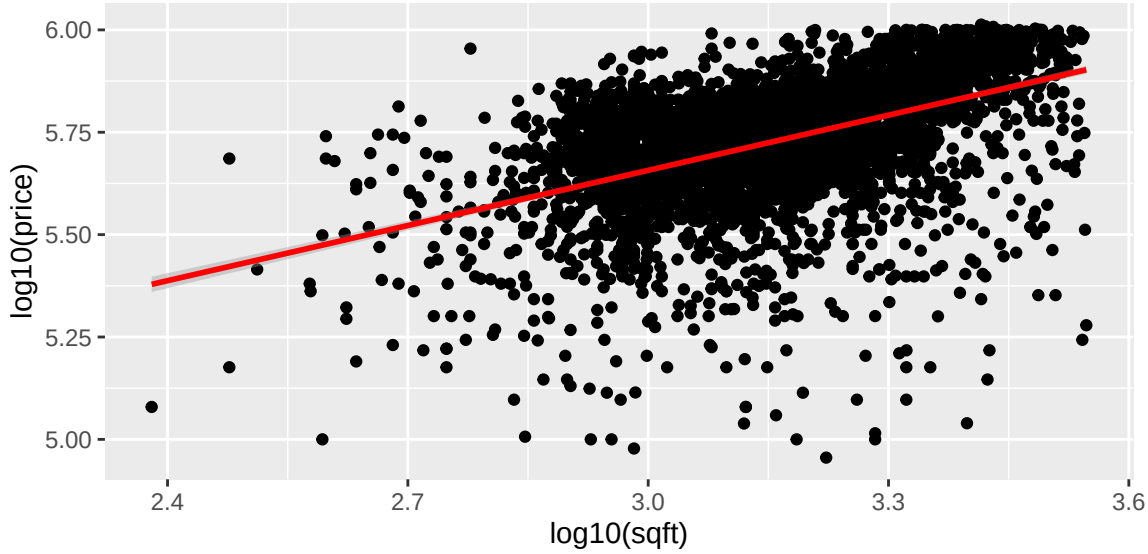
```
manhattan <- subset(nycdata,BOROUGH=="1")
manhattan$sqft <- as.numeric(manhattan$`GROSS SQUARE FEET`)
```

```
## Warning: NAs introduced by coercion
```

```
manhattan$price <- as.numeric(manhattan$`SALE PRICE`)
manhattan <- subset(manhattan,sqft!=0)


ggplot(statenisland, aes(x = log10(sqft), y = log10(price))) +
  geom_point() +
  stat_smooth(method = "lm",color="red")
```

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



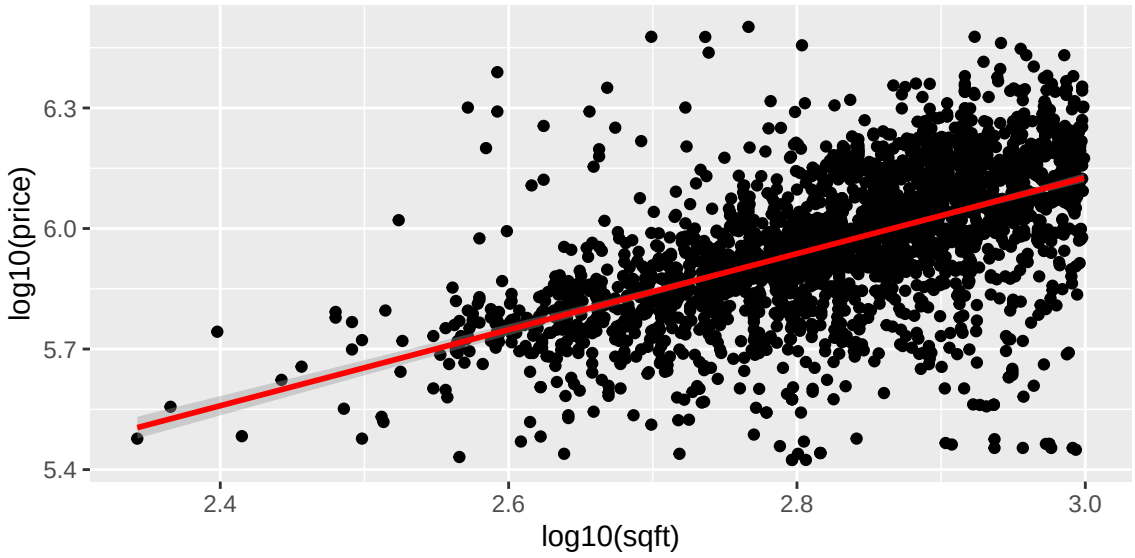
As before, we have price and square footage outliers. The procedure outlined below is as follows: remove clearly erroneous data points (i.e., where the square footage or price are 0) and then remove data points that fall outside of the first and fourth quartiles of the Staten Island data set. Since we do not want to fit "noise as signal" we only care about the internal structure of the data, which is what permits us to cut off the tails in this way. We again now remove the outliers, but because they are more extreme we remove outliers after adjusting for heteroskedasticity (i.e., we adjust `log10(price)`):

```
quartiles <- quantile(manhattan$sqft, probs=c(.25, .75), na.rm = TRUE)
IQR <- IQR(manhattan$sqft)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
manhattan <- subset(manhattan, manhattan$sqft > Lower &
                                    manhattan$sqft < Upper)
```

```
manhattan <- subset(manhattan,price!=0)
quartiles <- quantile(log10(manhattan$price), probs=c(.25, .75), na.rm = TRUE)
IQR <- IQR(log10(manhattan$price))
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
manhattan <- subset(manhattan, log10(manhattan$price) > Lower &
                                    log10(manhattan$price) < Upper)

ggplot(manhattan, aes(x = log10(sqft), y = log10(price))) +
  geom_point() +
  stat_smooth(method = "lm",color="red")
```

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

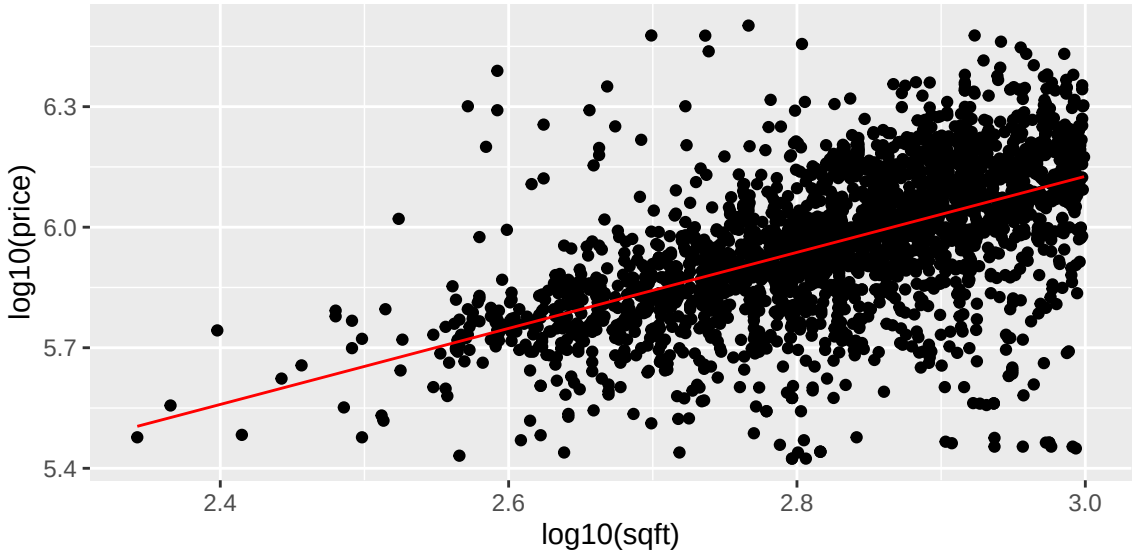


(2b)

We now attempt to fit a linear model to `log10(price)`. As before, we suspect `log10(sqft)` may or may not give strong model, and we realize $\bar{R}^2 = 0.36$ with $F = 1344$ (this is still stronger than for Staten Island).

```
fit_initial <- lm(log10(price)~log10(sqft),manhattan)
pred <- predict(fit_initial,manhattan)
pred_initial <- data.frame(orig=log10(manhattan$sqft),pred=pred)
ggplot(manhattan, aes(x = log10(sqft), y = log10(price))) +
```

```
  geom_point() +
  geom_line(color='red',data = pred_initial, aes(x=orig,y=pred))
```



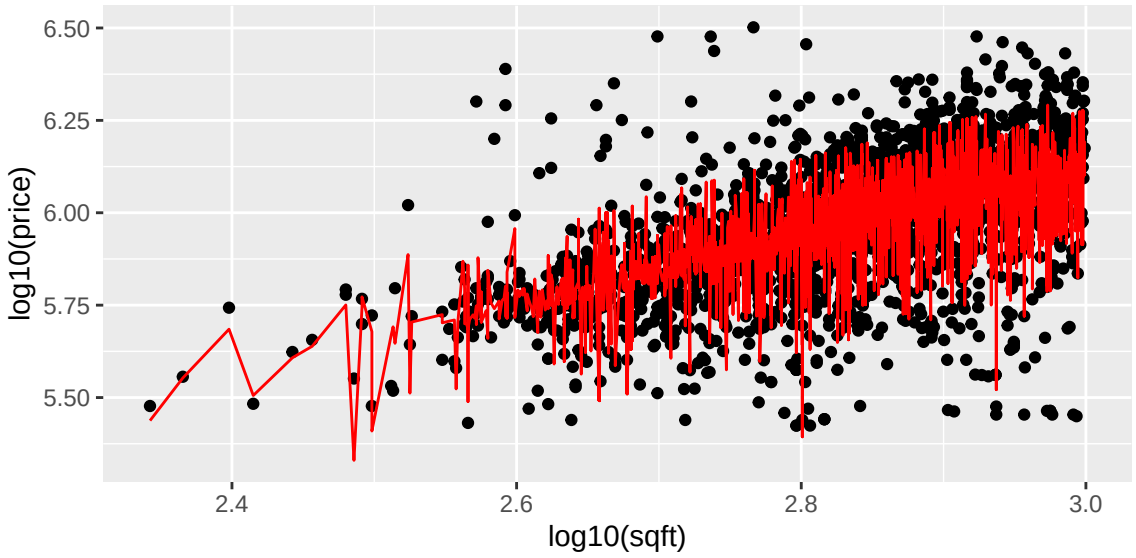

```
summary(fit_initial)
```

```
##
## Call:
## lm(formula = log10(price) ~ log10(sqft), data = manhattan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67091 -0.06768  0.00371  0.08246  0.64843
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.28810    0.07851   41.88   <2e-16 ***
## log10(sqft)  0.94617    0.02774   34.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.151 on 2512 degrees of freedom
## Multiple R-squared:  0.3166, Adjusted R-squared:  0.3163
## F-statistic:  1164 on 1 and 2512 DF,  p-value: < 2.2e-16
```

```
fit <- lm(log10(price)~log10(sqft)
                       +as.factor(`ZIP CODE`)
                       +`YEAR BUILT`
                       +`TOTAL UNITS`
                       ,data=manhattan) %>%  stepAIC(direction='backward',trace = FALSE)
summary(fit)$adj.r.squared
```

```
## [1] 0.5615459
```

```
pred <- predict(fit,manhattan)
pred_aic <- data.frame(orig=log10(manhattan$sqft),pred=pred)
ggplot(manhattan, aes(x = log10(sqft), y = log10(price))) +
  geom_point() +
  geom_line(color='red',data = pred_aic, aes(x=orig,y=pred))
```



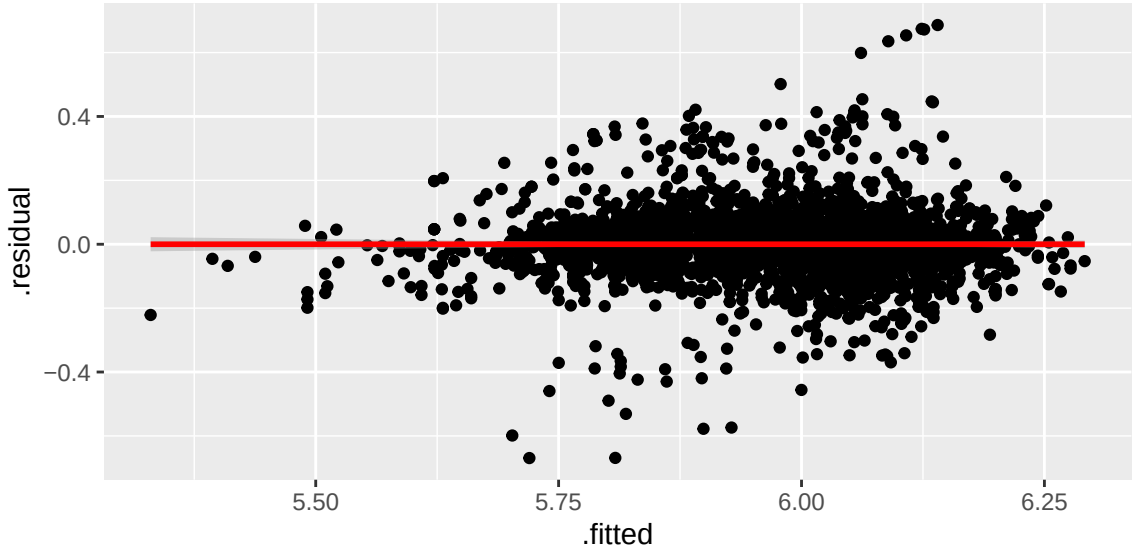
Taking the best model from (1c), we arrive at $\bar{R}^2 = 0.60$ with $F = 78.4$. Clearly we are over-fitting, but including ZIP and UNIT factors gives some strongly significant regressors.

Given that these fits are stronger for the Manhattan data than for the Staten Island data—despite the shared specification—we can conclude that the model generalizes well. We now plot the residuals (because we took the logarithm, we compare the `log10` residual on price):

```
resid <-pred-log10(manhattan$price)
resid.out <- data.frame(.fitted=pred, .residual=resid)
```

```
ggplot(resid.out, aes(x = .fitted, y = .residual)) +
  geom_point() +
  stat_smooth(method = "lm",color="red")
```

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



(2b)

We now reuse the KNN, Random Forest, and SVM model specifications generated on the Staten Island data on the Manhattan data and compare accuracy:

(2bi) KNN

```
set.seed(1)
manhattan_trim <- manhattan %>% select(price,sqft)
manhattan_trim$neighborhood <- as.numeric(factor(manhattan$NEIGHBORHOOD))
manhattan_trim$zip <- as.factor(manhattan$`ZIP CODE`)
manhattan_trim <- manhattan_trim %>% na.omit
indices=c(1 :nrow(manhattan_trim))
train_pct =.7
train_size = floor(nrow(manhattan_trim) * train_pct)
train_idx <- sample(indices,train_size)
train = rep(FALSE ,nrow(manhattan_trim))
for(i  in  train_idx){
```

```
    train[i]= TRUE
}
train_data <- manhattan_trim[train,]
test_data <-  manhattan_trim[!train,]
test_price =  as.matrix(manhattan_trim$price[!train])
train_price = as.matrix(manhattan_trim$price[train ])
pred.knn <- knn(train_data,test_data,train_price,k =1)
c_matrix <- table(pred.knn,manhattan_trim$price[!train])
# testing accuracy
100*sum(diag(c_matrix))/sum(c_matrix)
```

```
## [1] 0.6622517
```

Despite using the same specification as the Staten Island KNN model, KNN performs considerably better. This is likely because the size of a sold property in Manhattan captures a greater portion of the variation of the final sale price than in Staten Island.

(2bii) Random Forest

```
fit.rf= randomForest(price~sqft+zip+neighborhood,data=train_data)
pred  <-  predict(fit.rf,test_data)
probs  <-  pred
pred.rf <- rep(0,length(probs))
pred.rf[probs>0.5] <- 1
c_matrix <- table(pred.rf, test_price)
100*sum(diag(c_matrix))/sum(c_matrix)
```

```
## [1] 0.1324503
```

We find considerably stronger forecasting accuracy here than in Staten Island likely for the same reason as mentioned above.

(2biii) SVM

```
fit.svm <- svm(price ~ sqft+zip+neighborhood, data=train_data, kernel = 'radial')
pred.svm  <-  predict(fit.svm,test_data)
```

```
c_matrix <- table(pred.svm, test_price)
100*sum(diag(c_matrix))/sum(c_matrix)
```

```
## [1] 0.2649007
```

The SVM here does worse than in Staten Island. It is worth mentioning that removing `zip` and `neighborhood` improves the accuracy precipitously (to approximately 0.53). Because these two datasets differ, we could improve the accuracy by *tuning*. The size of this dataset, however, makes tuning infeasible.

(2c)

We observe a much greater overall price in housing in Manhattan compared to Staten Island. The square footage is also *much* lower, the highest in Staten Island being 3523 sqft and in Manhattan, 998 sqft. Manhattan also only has 36 neighborhoods, whereas Staten Island has 54. The overall greater variety in the housing sales in Staten Island may very well be responsible for the weaker model correlation and testing accuracies.

(3) Conclusions

We clearly demonstrate that as model flexibility increases, so does variance; moreover, as we add dummy variables to our linear regressions the overall significance of our regressors decrease despite R^2 and $\bar{R}^2$ increasing. We also find that "more" flexible models, like KNN, may better suit certain datasets than others. In particular, we find that KNN on the Staten Island dataset underperforms relative to the forecasting accuracy of a linear model, whereas the same KNN model overperforms the forecasting accuracy of the linear model for the Manhattan dataset. Nevertheless, we find that an SVM model performed similarly (~20-30% forecasting accuracy) for both datasets. Removing some features from each model led to different, sometimes stronger, forecasting accuracies. This implies that, despite sharing a common host dataset, that individual boroughs of New York City have features that are more pronounced in the determination of (and explain more of the variance in) final sale prices. In particular, the square footage of sold housing was a stronger indicator of the final sale price in Manhattan than in Staten Island.

(4)

Thank you for the extension. This semester has been absolutely slammed and I **really appreciate** your flexibility. Thank you, Professor!