Homework 11 Boulder housing sales for 2024

Jared Andreatta

2025-04-17

```r
# setwd("~/Dropbox/Home/Teaching/MATH531/MATH-531/MATH531S2024/Homework")
suppressMessages(library(fields))
knitr::opts_chunk$set(echo = TRUE)
```

Background – Boulder County housing sales for 2024

The goal of this assignment is to come up a defensible model for predicting the selling price of a house in Boulder County based on some covariables. In particular we expect the square footage (SF) of the house will be useful along with some other measures. There is also the particular town/city where the house is located and this might also be a factor. The real estate wisdom is that the three most important features determining price are location, location and location!

These data were assembled after much munging and merging. The sources are publically available through the Boulder County .gov web site.

In subsetting these data to something that looked like a family house I made these restrictions, among others that

```
ind5<- work$SF<= 6000 & work$SF >=300 &
  work$nbrBedRoom <=6 & work$nbrFullBaths <=6 &
  work$nbrBedRoom >0 &
  work$nbrFullBaths > 0
ind6<- work$price>=1e4 & work$price<= 2e7 # before dividing by 1000
work$price <- work$price/1000
```

Load the Boulder sales data and the mystery house.

```r
load("sales2024.rda")
```

I found that log10 price seems to make the most sense for the assumptions of a linear model and to make the numbers easier ** price is in 1000's of dollars **. Below are some basic plots just to show some about these data. The alpha function adds some transparency to the plotted symbols.

```r
head( sales2024)
```

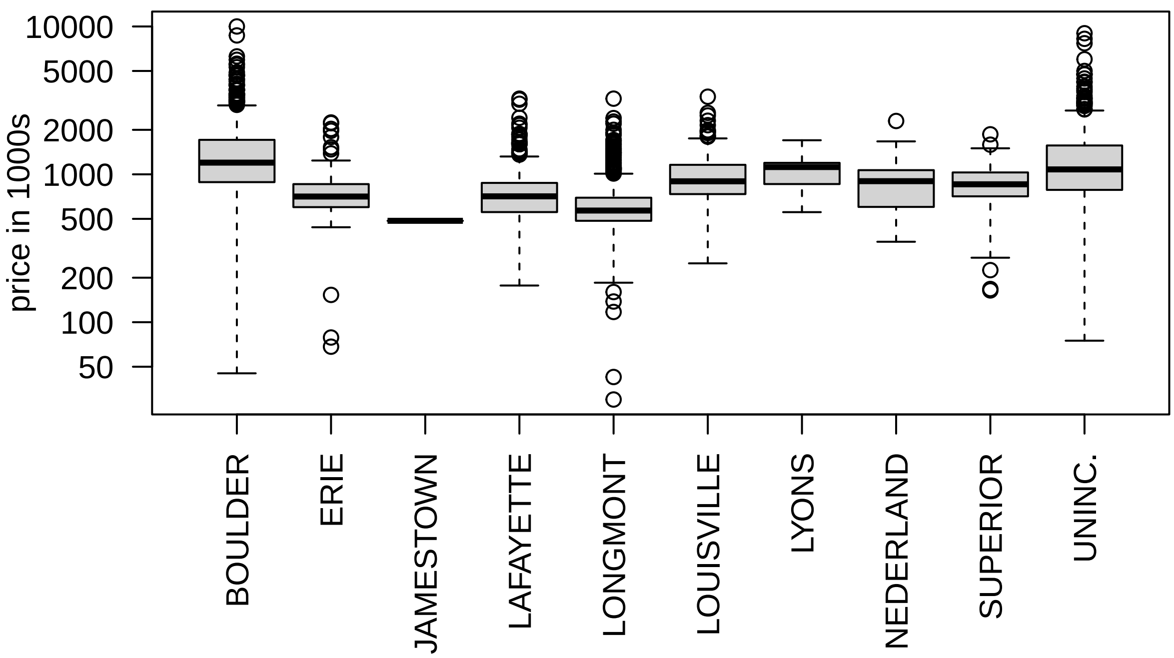
```
##          strap price year month    city builtYear mainfloorSF nbrBedRoom
## 16565 R0000064  1800 2024     3 BOULDER      1974        1397          3
## 16642 R0000078  1300 2024     9 BOULDER      1974        1222          3
## 17458 R0000244  2200 2024     2 BOULDER      1889        1588          4
## 17752 R0000310   950 2024     6 BOULDER      1920        1140          3
## 17949 R0000347  1200 2024     2 BOULDER      1950        1031          2
## 18022 R0000361   720 2024    11 BOULDER      1925         735          2
##       nbrFullBaths totalActualVal landAssessedVal bldAssessedVal   SF
## 16565            1        2047400           94597          38894 2625
## 16642            2        1623900           81793          23323 1222
```

```
## 17458              2        1969200          59408          68843 2080
## 17752              2        1553700          68173          32240 1592
## 17949              2        1530400          71027          27825 1031
## 18022              1        1004900          56910           6733  735
```

```
library( fields)
library( scales) # loaded for the alpha function
bubblePlot( sales2024$SF,
            sales2024$price,
            sales2024$builtYear, log="y", size=.5,
            col=alpha(turbo(256),.5) )
title(" Price/ SF based on age of house")
```


```
bubblePlot( sales2024$SF,
            sales2024$price,
            sales2024$nbrBedRoom, xlim=c(500,6000),
            col=alpha(turbo(6),.5),
            log="y", size=.5,
            xlab="square footage",
            ylab= "price in 1000s")

title(" Price vs SF based on number of bedroom")
```


```
par( mar=c( 10,5,2,1))
boxplot( sales2024$price ~ sales2024$city,
         log="y", las=2, xlab="", ylab="price in 1000s")
```



And here is an example of a fitted linear model – with no guarantee this is the best one. The combination of a continuous variable (SF) and a factor (city) is known historically as the *analysis of covariance*.

```
fit<- lm( log10(price)~ SF + city , data=sales2024)
summary( fit)
```

```
##
## Call:
## lm(formula = log10(price) ~ SF + city, data = sales2024)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.31673 -0.06055  0.00573  0.07015  0.83482
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.788e+00  7.952e-03 350.545  < 2e-16 ***
## SF              1.698e-04  3.175e-06  53.478  < 2e-16 ***
## cityERIE       -3.251e-01  9.584e-03 -33.918  < 2e-16 ***
## cityJAMESTOWN  -4.619e-01  1.432e-01  -3.226  0.00127 **
## cityLAFAYETTE  -2.258e-01  9.033e-03 -24.998  < 2e-16 ***
## cityLONGMONT   -3.096e-01  6.966e-03 -44.450  < 2e-16 ***
## cityLOUISVILLE -1.358e-01  1.099e-02 -12.361  < 2e-16 ***
## cityLYONS      -1.367e-01  3.329e-02  -4.108 4.08e-05 ***
## cityNEDERLAND  -1.775e-01  2.971e-02  -5.974 2.55e-09 ***
## citySUPERIOR   -2.255e-01  1.134e-02 -19.881  < 2e-16 ***
## cityUNINC.     -1.131e-01  8.557e-03 -13.223  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1431 on 3474 degrees of freedom
## Multiple R-squared:  0.6206, Adjusted R-squared:  0.6195
## F-statistic: 568.2 on 10 and 3474 DF,  p-value: < 2.2e-16
```

Problem 1

- Create the regression X matrix for this model and redo the computation "by hand" using linear algebra. Check that you get the same OLS parameter estimates as the call to **lm**.

```r
X <- model.matrix(log10(price) ~ SF + city, data=sales2024)
Y <- log10(sales2024$price)

betahat <- solve(t(X) %*% X) %*% t(X) %*% Y
yhat <- X%*%betahat

compare_beta <- data.frame(betahat, coef(fit))
compare_fits <- data.frame(yhat, fitted.values(fit))

compare_beta
```

```
##                      betahat     coef.fit.
## (Intercept)     2.7875192615  2.7875192615
## SF              0.0001698014  0.0001698014
## cityERIE       -0.3250720388 -0.3250720388
## cityJAMESTOWN  -0.4619262595 -0.4619262595
## cityLAFAYETTE  -0.2257996697 -0.2257996697
## cityLONGMONT   -0.3096373583 -0.3096373583
## cityLOUISVILLE -0.1357940406 -0.1357940406
## cityLYONS      -0.1367485822 -0.1367485822
## cityNEDERLAND  -0.1775097616 -0.1775097616
```

```
## citySUPERIOR   -0.2254782378 -0.2254782378
## cityUNINC.     -0.1131408927 -0.1131408927
```

```r
head(compare_fits)
```

```
##            yhat fitted.values.fit.
## 16565 3.233248           3.233248
## 16642 2.995017           2.995017
## 17458 3.140706           3.140706
## 17752 3.057843           3.057843
## 17949 2.962584           2.962584
## 18022 2.912323           2.912323
```

- For this model what is the estimated effect of the log price for those houses sold in the city of Louisville? How about Boulder?

If the house is in Boulder, we expect a 2.788 log10 home price, or $10^{2.788} = \$613,083$ home price, for a house in Boulder for a 0 square foot house. Clearly, this is unfeasible, but this serves as a baseline for a house in Boulder when all other predictors are 0 and it is a benchmark for homes in other cities. For a house in Louisville, we expect a -.137 decrease in the log10 prices from a house in Boulder, holding square footage constant. If we exponentiate $10^{-.137} = .731$, we can interpret this as a house in Louisville being 73.1% in price from a house in Boulder with the same square footage, on average.

```r
10^(fit$coefficients[1])
```

```
## (Intercept)
##     613.083
```

```r
10^(fit$coefficients[7])
```

```
## cityLOUISVILLE
##      0.7314859
```

Problem 2

- Take a look at

```r
MysteryInfo
```

```
##          strap    city builtYear mainfloorSF nbrBedRoom nbrFullBaths
## 16238 R0009567 BOULDER      1965        1867          6            3
##       TotalFinishedSF totalActualVal landAssessedVal bldAssessedVal
## 16238            2325        3416200          102623         122577
```

This is a house in my neighborhood that recently went up for sale. Also see **MysteryHouse.pdf** (The house was almost totally rebuilt in 2018 and the county data is misleading. I think what was left from 1965 was a single wall with the chimney.) Using the model in problem 1 work out "by hand" the OLS prediction for the mean log sale price for this house. Use the covariates from the R object to make your prediction and also from the real estate ad.

- Give a 95% CI for this prediction assuming normal errors. Now transform your interval to price instead of log price to make it easier to interpret.

$CI_{.95} = [\$1483136\ ,\ \$1561112]$

```r
MysteryInfo.data <- data.frame(SF=MysteryInfo$TotalFinishedSF, city="BOULDER")
# fit from problem 1
fit <- lm(log10(price) ~ SF + city, data = sales2024)
X <- model.matrix(fit)

# Manually constructing row of MysteryInfo data
x0 <- setNames(rep(0, ncol(X)), colnames(X))
x0["(Intercept)"] <- 1
x0["SF"]          <- MysteryInfo$TotalFinishedSF
x0 <- matrix(x0, nrow = 1)

# x0^T betahat
beta_hat <- coef(fit)
x0_betahat <- as.numeric(x0 %*% beta_hat)

# Variance-covariance matrix for standard error
vcov_mat <- vcov(fit)
SE_mean <- sqrt(x0 %*% vcov_mat %*% t(x0))

# Degrees of freedom and critical value
df_res <- fit$df.residual
crit <- qt(.975, df = df_res)

# 95% CI for log10(price)
ci_log <- c(x0_betahat - crit*SE_mean,
               x0_betahat + crit*SE_mean)

# 95% CI for real price
ci_price_mean <- 10^ci_log * 1000

cat("95% CI on log10(price): [",
    ci_log[1], ",", ci_log[2], "]\n")
```

```
## 95% CI on log10(price): [ 3.171181 , 3.193434 ]
```

```r
# Sanity check
pred.CI.log <- predict(fit, MysteryInfo.data, interval = "confidence", level=.95)
print(pred.CI.log)
```

```
##        fit      lwr      upr
## 1 3.182307 3.171181 3.193434
```

```r
cat("95% CI on price: [",
    ci_price_mean[1], ",", ci_price_mean[2], "]\n")
```

```
## 95% CI on price: [ 1483136 , 1561112 ]
```

```r
# Sanity Check
print(10^pred.CI.log*1000)
```

```
##       fit     lwr     upr
## 1 1521624 1483136 1561112
```

- Redo your confidence interval where now you are predicting a specific house (aka a new observation) rather than the mean.

$PI_{.95} = [\$797107\ ,\ \$2904681]$

```r
# Everything remains the same, but we add sigma^2 to the term inside the sqrt
# in the SE expression to get SE of a prediction instead of the mean
sigma2 <- summary(fit)$sigma^2

# Standard error for predicted value
SE_pred <- sqrt( sigma2 + x0 %*% vcov_mat %*% t(x0) )
crit <- qt(.975, df = df_res)

# Prediction interval
pi_log <- c( x0_betahat - crit*SE_pred,
               x0_betahat + crit*SE_pred )

# 95% PI for log10(price)
cat("95% PI on log10(price): [",
    pi_log[1], ",", pi_log[2], "]\n")
```

```
## 95% PI on log10(price): [ 2.901517 , 3.463098 ]
```

```r
# Sanity Check
pred.PI.log <- predict(fit, MysteryInfo.data, interval = "prediction", level = .95)
print(pred.PI.log)
```

```
##        fit      lwr      upr
## 1 3.182307 2.901517 3.463098
```

```r
# Converted back to price
pi_price <- 10^pi_log * 1000

# 95% PI for actual price
cat("95% PI on price ($): [",
    pi_price[1], ",", round(pi_price[2]), "]\n")
```

```
## 95% PI on price ($): [ 797106.8 , 2904681 ]
```

```r
# Sanity Check
print(10^pred.PI.log*1000)
```

```
##       fit      lwr     upr
## 1 1521624 797106.8 2904681
```

Problem 3

- Using additional variables create a better model for these data and use an F-test to established that it is significantly better over the model in problem 1.

I added the **nbrFullBaths, nbrBedRoom,** and **totalActualVal** variables to the model. I calculated the F-test manually we get an F-statistic of 401.4 and a p-value close to 0. This indicates that the model with the new variables is significantly better than the model in the first.

```r
library(GGally)

sales_cleaned <- subset(sales2024, select=c(-strap,-year,-month))
# ggpairs(sales_cleaned)
```

```r
# Fit from problem 1
X <- model.matrix(log10(price) ~ SF + city, data=sales2024)
```

```r
Y <- log10(sales2024$price)
betahat <- solve(t(X) %*% X) %*% t(X) %*% Y
yhat <- X%*%betahat

# New fit
X_new <- model.matrix(log10(price) ~ SF+city+nbrFullBaths+nbrBedRoom+totalActualVal, data=sales2024)
Y_new <- log10(sales2024$price)
betahat_new <- solve(t(X_new) %*% X_new) %*% t(X_new) %*% Y_new
yhat_new <- X_new%*%betahat_new

SSE <- sum((Y-yhat)^2) # Old SSE
SSE_new <- sum((Y_new-yhat_new)^2) # New SSE
q <- 3 # q = k - p (difference between # predictors)
k <- 6 # 6 params in new model
n <- length(Y) # Sample size

# F-stat and p-val
F_test <- ((SSE - SSE_new) / q) / (SSE_new / (n-k))
p_val  <- pf(F_test, df1 = q, df2 = n-k, lower.tail = FALSE)

cat("F-stat: ", F_test, "\n")
```

```
## F-stat:  401.3579
```

```r
cat("p-val: ", p_val, "\n")
```

```
## p-val:  7.071502e-224
```

```r
# Sanity check
fit<- lm( log10(price)~ SF + city , data=sales2024)
fit2 <- lm(log10(price) ~ SF+city+nbrFullBaths+nbrBedRoom+totalActualVal, data=sales2024)

anova(fit, fit2)
```

```
## Analysis of Variance Table
##
## Model 1: log10(price) ~ SF + city
## Model 2: log10(price) ~ SF + city + nbrFullBaths + nbrBedRoom + totalActualVal
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   3474 71.140
## 2   3471 52.849  3    18.291 400.43 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Problem 4

- For the model in problem 1 use 10 fold cross-validation to evaluate the prediction error. (You can use lm here if you don't want to do the computations by hand.)

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
# Seed for reproducibiity
set.seed(1122)
```

```r
# This is doing random 10-fold CV
train_control <- trainControl(
                  method = "cv",
                  number = 10,
                  savePredictions = "final")

cv_model <- train(log10(price) ~ SF + city,
                  data = sales2024,
                  method = "lm",
                  trControl = train_control)

print(cv_model)
```

```
## Linear Regression
##
## 3485 samples
##    2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3137, 3136, 3136, 3136, 3136, 3137, ...
## Resampling results:
##
##   RMSE       Rsquared  MAE
##   0.1432135  0.61894   0.09572419
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
results <- cv_model$results

# RMSE/MAE in actual price
cat("Price RMSE: ", 10^(results$RMSE)*1000, "\n")
```

```
## Price RMSE:  1390.636
```

```r
cat("Price MAE: ", 10^(results$MAE)*1000)
```

```
## Price MAE:  1246.592
```

- For your out-of-sample predictions find the prediction errors, divide them by the prediction standard error (based on predicting a specific house.) and summarize this distribution. Note that if the linear model is appropriate these standardized prediction errors will be distributed N(0,1).

The distribution is approximately normal with $\mu = -0.000744$ and $\sigma = 1.000772$. However, based off of the qqplot, there seems to be fat tails of the distribution that need to be considered.

```r
library(caret)

# Final cv model
fm <- cv_model$finalModel
sigma_hat <- summary(fm)$sigma # sigma

pred <- cv_model$pred # Prediction vector
nd <- sales2024[pred$rowIndex, ] # Hold-out

dv <- dummyVars( ~ SF + city, data = sales2024 ) # Dummy vars
```

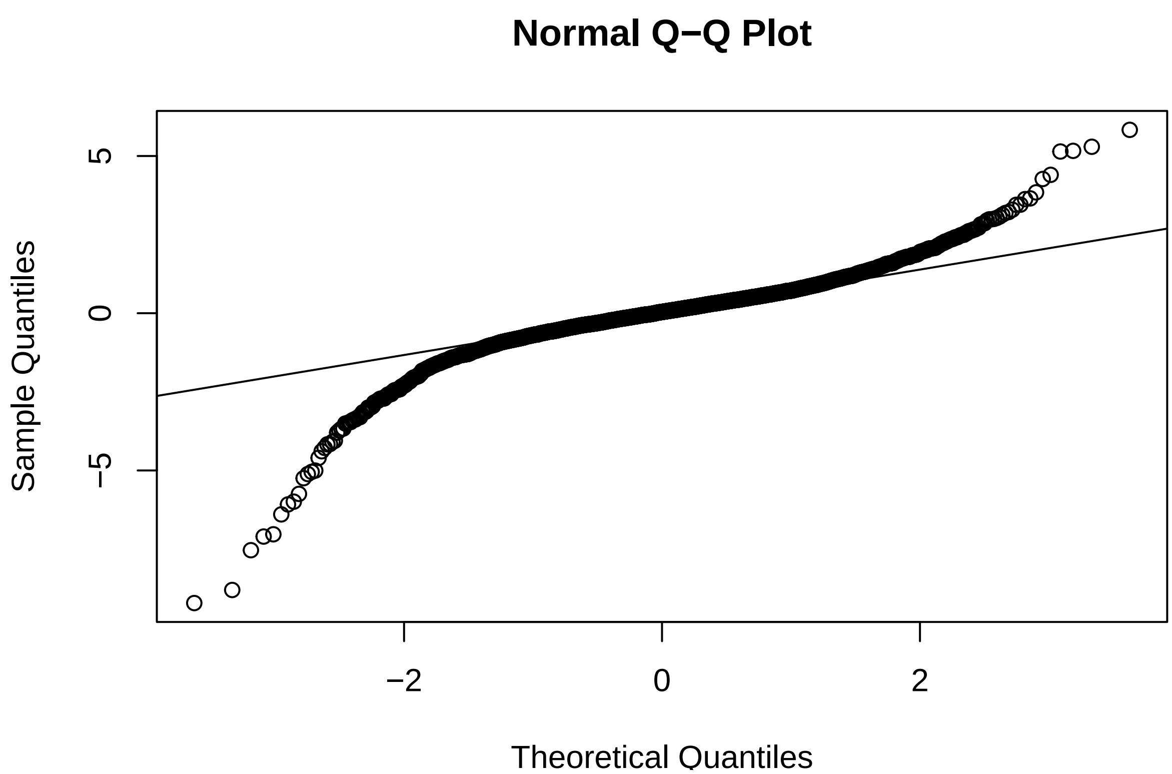
```r
nd_dum <- predict(dv, newdata = nd)
nd_df <- as.data.frame(nd_dum)

# Predictions and SE from final model
pf <- predict(fm, newdata = nd_df, se.fit = TRUE)

# Prediction SE and standardized error
pred$se.pred <- sqrt(pf$se.fit^2 + sigma_hat^2)
pred$std_pred_err <- (pred$obs - pred$pred) / pred$se.pred

# Summary stats
summary(pred$std_pred_err)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -9.216460 -0.428327  0.042140 -0.000744  0.487626  5.831300
```

```r
sd(pred$std_pred_err)
```

```
## [1] 1.000772
```

```r
# QQ Plot
qqnorm(pred$std_pred_err)
qqline(pred$std_pred_err)
```



Problem 5 EXTRA CREDIT

Apply the 10 cross-validation exercise to your (better) model in problem 3. Use the same cross-validation folds and compare the out-of-sample prediction errors to those from problem 4. Is your model still preforming better?

Based off of the cv_model summary (RMSE and R^2), this model is performing better than the model in problem 1. However, I can't seem to get the same code to work for this model for analyzing the prediction errors.

```
library(caret)

set.seed(1122)

train_control <- trainControl(method = "cv", number = 10)

cv_model <- train(log10(price) ~ SF+city+nbrFullBaths+nbrBedRoom+totalActualVal,
                  data = sales2024,
                  method = "lm",
                  trControl = train_control)
```

```
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
```

```
print(cv_model)
```

```
## Linear Regression
##
## 3485 samples
##    5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3137, 3136, 3136, 3136, 3136, 3137, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1230328  0.7168956  0.0755786
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
### Can't get this to work ###

# fm  <- cv_model$finalModel # Final cv model
# sigma_hat <- summary(fm)$sigma # sigma
#
# pd <- cv_model$pred # Prediction vector
# nd <- sales2024[pd$rowIndex, ] # Hold-out
#
# dv <- dummyVars( ~ SF + city + nbrFullBaths + nbrBedRoom + totalActualVal, data = sales2024 ) # Dummy
# nd_dum <- predict(dv, newdata = nd)
# nd_df <- as.data.frame(nd_dum)
#
# pf <- predict(fm, newdata = sales2024, se.fit = TRUE) # Predictions and SE from final model
#
# pd$se.pred <- sqrt(pf$se.fit^2 + sigma_hat^2) # Prediction SE
# pd$std_pred_err <- (pd$obs - pd$pred) / pd$se.pred # Standardized Pred error
#
# summary(pd$std_pred_err) # Summary Stats
# sd(pd$std_pred_err) # Std Dev
#
# # QQ Plot
```

```r
# qqnorm(pd$std_pred_err)
# qqline(pd$std_pred_err)
```