IMT 573 Lab: Simple Linear Regression

Naga Soundari Balamurugan

November 13th, 2018

{Don't forget to list the full names of your collaborators!}

Collaborators: Jayashree Raman

Instructions:

1. Download the `week8a_lab.Rmd` file from Canvas. Open `week8a_lab.Rmd` in RStudio and supply your solutions to the assignment by editing `week8a_lab.Rmd`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment.
3. Be sure to include well-documented (e.g. commented) code chucks, figures and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text. If you are using more than just a standard function that you found from another source, please credit the source in the comments.
4. Collaboration on labs is encouraged, but students must turn in an individual assignments. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
5. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF` or `Knit Word`, rename the R Markdown file to `YourLastName_YourFirstName_Lab8a.Rmd`, knit a PDF or DOC and submit both the PDF/DOC and the Rmd file on Canvas.

In this lab, you will need access to the following R packages (you may need to install the package "caTools" in your console before using the `library()` function):

```
# Load some helpful libraries
library(tidyverse)
library(caTools)
library(dplyr)
```

1. Load the kc_house_data.csv data available in the labs folder on Canvas and call it "housing". This data was retrieved from (and you can find further information about it at): https://www.kaggle.com/harlfoxem/housesalesprediction/version/1

```
#Read the data
housing <- read.csv("kc_house_data.csv")
```

Split the data into a training set and a testing set (75% training 25% testing). I have provided code for one of many options for doing a train/test split:

```
# code adapted from https://rpubs.com/ID_Tech/S1 AND https://stackoverflow.com/a/31634462

# Set seed for reproducibility
set.seed(1695)
# splits the data in the ratio mentioned in SplitRatio. After splitting marks these rows as logical
# TRUE and the the remaining are marked as logical FALSE
sample <- sample.split(housing$price, SplitRatio = .75)
```

```r
# creates a training dataset named train with rows which are marked as TRUE
train <- subset(housing, sample == TRUE)
# creates a training dataset named test with rows which are marked as FALSE
test <- subset(housing, sample == FALSE)
```

2. Use the `price` variable as your dependent variable (also known as your response or output variable). Run several multiple regression models (you choose the independent (or predictor, input) variables) on the *training* data and compare p-values and r-squared values. When you have run a handful of different models, report back on which one seemed to be the best of all the ones you ran. If you run into problems (messy data, null values, missing values) consider your options for overcoming them and write out your thought process here.

```r
model1 <- lm(price ~ sqft_living + yr_built, data = train)
summary(model1)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + yr_built, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1796773  -140215   -21038   106614  3960723
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.766e+06  1.406e+05   33.89   <2e-16 ***
## sqft_living  3.144e+02  2.251e+00  139.67   <2e-16 ***
## yr_built    -2.474e+03  7.207e+01  -34.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 260600 on 16953 degrees of freedom
## Multiple R-squared:  0.5367, Adjusted R-squared:  0.5366
## F-statistic:  9819 on 2 and 16953 DF,  p-value: < 2.2e-16
```

```r
model2 <- lm(price ~ sqft_living + sqft_lot + yr_built, data = train)
summary(model2)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + sqft_lot + yr_built, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1737241  -140120   -20842   106659  3947377
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.770e+06  1.405e+05  33.953  < 2e-16 ***
## sqft_living  3.167e+02  2.280e+00 138.912  < 2e-16 ***
## sqft_lot    -2.944e-01  4.787e-02  -6.149 7.95e-10 ***
## yr_built    -2.476e+03  7.199e+01 -34.397  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 260300 on 16952 degrees of freedom
## Multiple R-squared:  0.5377, Adjusted R-squared:  0.5376
## F-statistic:  6573 on 3 and 16952 DF,  p-value: < 2.2e-16
```

```r
model3 <- lm(price ~ bedrooms + bathrooms + zipcode, data = train)
summary(model3)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + zipcode, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1616424  -187147   -42014   112164  5905959
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.128e+07  4.646e+06  -8.886  < 2e-16 ***
## bedrooms     2.170e+04  3.095e+03   7.012 2.43e-12 ***
## bathrooms    2.547e+05  3.764e+03  67.685  < 2e-16 ***
## zipcode      4.202e+02  4.735e+01   8.875  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 322700 on 16952 degrees of freedom
## Multiple R-squared:  0.2897, Adjusted R-squared:  0.2896
## F-statistic:  2305 on 3 and 16952 DF,  p-value: < 2.2e-16
```

```r
model4 <- lm(price ~ bedrooms + bathrooms, data = train)
summary(model4)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1552976  -188861   -43178   114029  5889327
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -51462       9606  -5.357 8.57e-08 ***
## bedrooms       20071       3096   6.482 9.31e-11 ***
## bathrooms     249840       3731  66.956  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 323400 on 16953 degrees of freedom
## Multiple R-squared:  0.2864, Adjusted R-squared:  0.2864
## F-statistic:  3403 on 2 and 16953 DF,  p-value: < 2.2e-16
```

```r
model5 <- lm(price ~ sqft_living + sqft_lot + zipcode, data = train)
summary(model5)
```

```
##
## Call:
```

```
## lm(formula = price ~ sqft_living + sqft_lot + zipcode, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1607666  -147369   -22910   106501  4195298
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.357e+07  3.856e+06 -16.488  < 2e-16 ***
## sqft_living  2.986e+02  2.256e+00 132.328  < 2e-16 ***
## sqft_lot    -2.082e-01  4.935e-02  -4.219 2.47e-05 ***
## zipcode      6.474e+02  3.930e+01  16.472  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 267100 on 16952 degrees of freedom
## Multiple R-squared:  0.5133, Adjusted R-squared:  0.5132
## F-statistic:  5958 on 3 and 16952 DF,  p-value: < 2.2e-16
```

```
model6 <- lm(price ~ waterfront + zipcode, data = train)
summary(model6)
```

```
##
## Call:
## lm(formula = price ~ waterfront + zipcode, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1463423  -216023   -82711   110562  7174070
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.637e+07  5.161e+06   8.985   <2e-16 ***
## waterfront   1.208e+06  3.160e+04  38.215   <2e-16 ***
## zipcode     -4.673e+02  5.262e+01  -8.880   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 366800 on 16953 degrees of freedom
## Multiple R-squared:  0.08217,    Adjusted R-squared:  0.08206
## F-statistic: 758.8 on 2 and 16953 DF,  p-value: < 2.2e-16
```

```
model7 <- lm(price ~ sqft_living + sqft_lot + yr_built + bedrooms + bathrooms, data = train)
summary(model7)
```

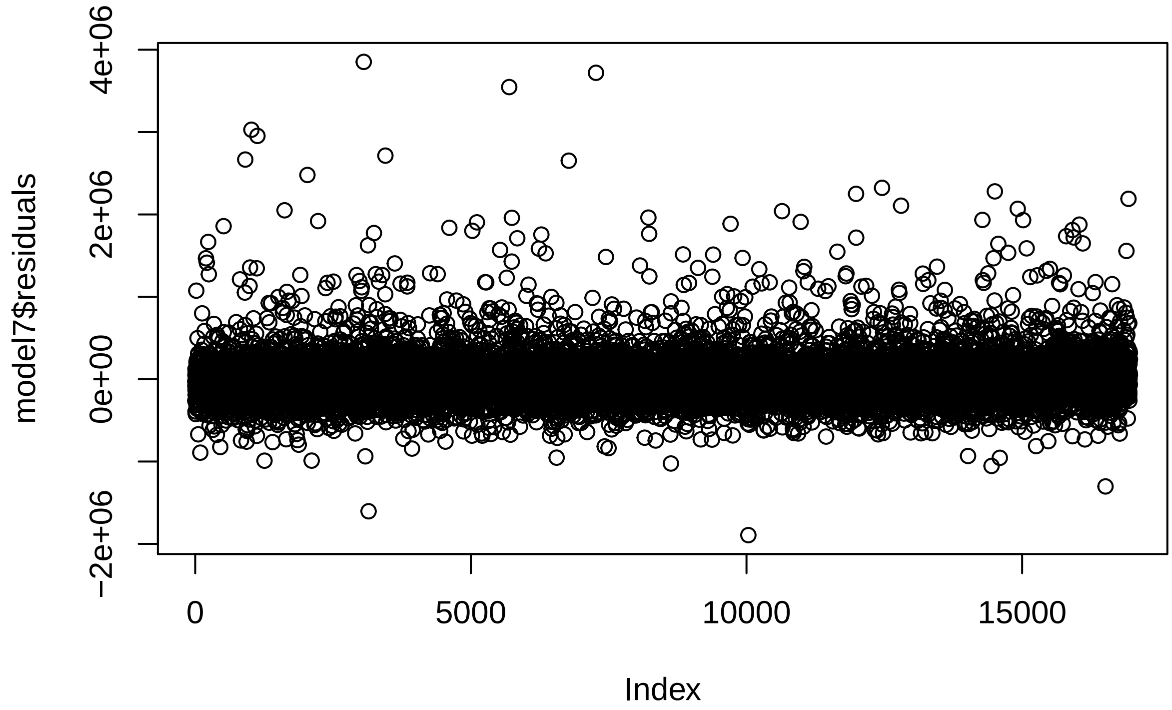
```
##
## Call:
## lm(formula = price ~ sqft_living + sqft_lot + yr_built + bedrooms +
##     bathrooms, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1893641  -133975   -17858   100312  3852125
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.214e+06  1.510e+05  41.158  < 2e-16 ***
## sqft_living  3.139e+02  3.432e+00  91.467  < 2e-16 ***
## sqft_lot    -3.348e-01  4.683e-02  -7.151 8.99e-13 ***
## yr_built    -3.171e+03  7.780e+01 -40.758  < 2e-16 ***
## bedrooms    -7.191e+04  2.586e+03 -27.810  < 2e-16 ***
## bathrooms    8.169e+04  4.323e+03  18.898  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 253200 on 16950 degrees of freedom
## Multiple R-squared:  0.5628, Adjusted R-squared:  0.5627
## F-statistic:  4365 on 5 and 16950 DF,  p-value: < 2.2e-16
```

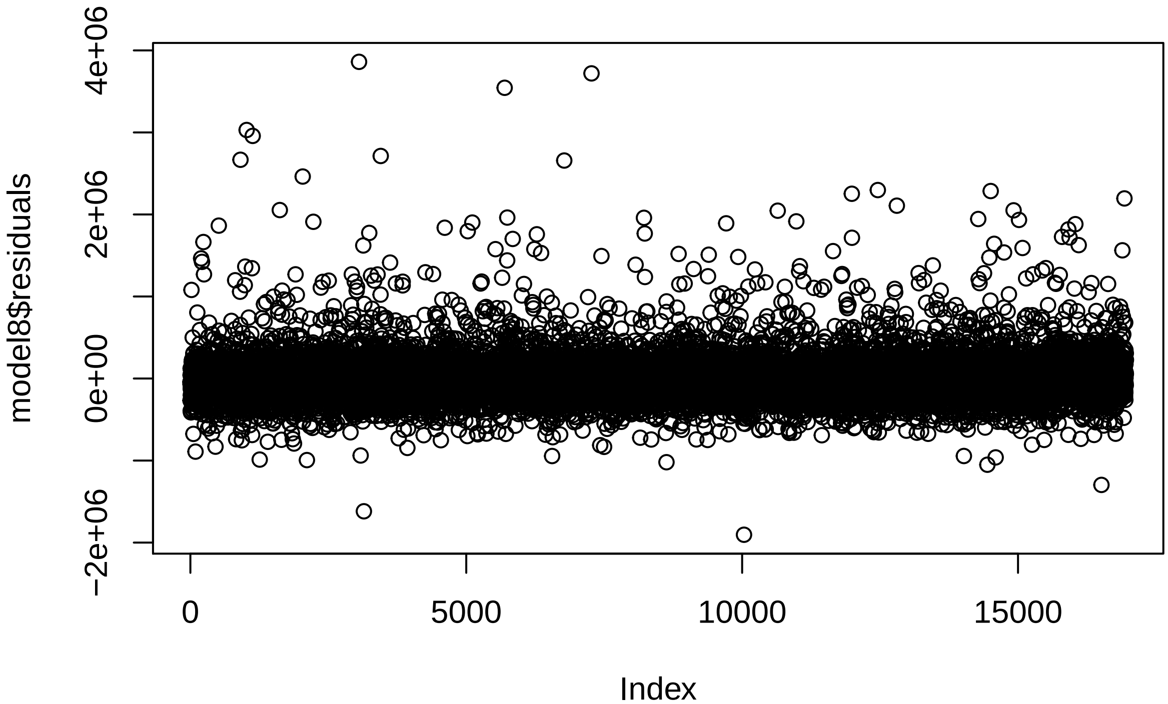
```r
model8 <- lm(price ~ sqft_living + sqft_lot + yr_built + bedrooms + bathrooms + zipcode, data = train)
summary(model8)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + sqft_lot + yr_built + bedrooms +
##     bathrooms + zipcode, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1904434  -133804   -17448   100878  3860370
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.807e+06  3.910e+06  -2.252 0.024308 *
## sqft_living  3.148e+02  3.438e+00  91.562  < 2e-16 ***
## sqft_lot    -3.159e-01  4.707e-02  -6.711 1.99e-11 ***
## yr_built    -3.074e+03  8.173e+01 -37.610  < 2e-16 ***
## bedrooms    -7.110e+04  2.593e+03 -27.418  < 2e-16 ***
## bathrooms    8.060e+04  4.330e+03  18.615  < 2e-16 ***
## zipcode      1.512e+02  3.932e+01   3.845 0.000121 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 253100 on 16949 degrees of freedom
## Multiple R-squared:  0.5632, Adjusted R-squared:  0.5631
## F-statistic:  3643 on 6 and 16949 DF,  p-value: < 2.2e-16
```

```r
plot(model7$residuals)
```




```r
plot(model8$residuals)
```



I had first created the models model1 till model 6. Among them, model 3, 4 and 6 has a low R-squared values and shows no significant impact on the price. Thus, I feel only the number of bedrooms, bathrooms, waterfront and zipcode does not affect the price. The models 1, 2 and 5 shows a good fit with a high R-squared value which means, sqft_living, sqft_lot, year built and zipcode affects the price significantly.

Thus, I created model 7 and 8 which takes all the above variables as input to predict the price. From the residual plots of model 7 and 8, we can see there data points are evenly spread. though there are more data points on the above the 0 line, the number is negligible. From all the analysis, Model 8 fits well with a higher R-squared than all the other models(sqft_living, sqft_lot, yr_built, bedrooms, bathrooms, zipcode are good predictors of price together).

3. Run a correlation test on the variables from your best fitting model. Does anything stand out to you?

```
columns_needed <- c("price", "sqft_living", "sqft_lot", "yr_built", "bedrooms", "bathrooms", "zipcode")
housing_data <- train[columns_needed]

cor_test_predictors <- cor(housing_data)
cor_test_predictors
```

```
##                   price sqft_living    sqft_lot    yr_built    bedrooms
## price        1.00000000   0.7102682  0.09225609  0.05952115  0.31264041
## sqft_living  0.71026819   1.0000000  0.17330781  0.32295429  0.57524429
## sqft_lot     0.09225609   0.1733078  1.00000000  0.05181870  0.03113696
## yr_built     0.05952115   0.3229543  0.05181870  1.00000000  0.16173812
## bedrooms     0.31264041   0.5752443  0.03113696  0.16173812  1.00000000
## bathrooms    0.53354431   0.7593865  0.08372773  0.50661072  0.51857754
## zipcode     -0.05568397  -0.2036783 -0.12878090 -0.34981116 -0.15599034
```

```
##               bathrooms     zipcode
## price        0.53354431 -0.05568397
## sqft_living  0.75938649 -0.20367831
## sqft_lot     0.08372773 -0.12878090
## yr_built     0.50661072 -0.34981116
## bedrooms     0.51857754 -0.15599034
## bathrooms    1.00000000 -0.20498436
## zipcode     -0.20498436  1.00000000
```

From the correlation matrix, We can see that price has a high positive correlation with sqft_living followed by no of bathrooms. Also sqft_living is highly correlated positively with number of bathrooms and bedrooms which is inituitive and obvious. Though addition of zipcode made a better fit(comparing model 7 and model 8), the correlation matrix does not show any significant correlation of zipcode with any other variable.

4. Create a binary classification for housing price: Mansion(1)/Not Mansion(0), on the full dataset and re-run your train/test split. (You decide the threshold price for a mansion versus not mansion.) Now create some logistic regression models using the binary classification as the output/dependent variable.

```
thresholdPrice <- 1000000
housing$Mansion <- ifelse(housing$price > thresholdPrice, 1, 0)
```

```
# code adapted from https://rpubs.com/ID_Tech/S1 AND https://stackoverflow.com/a/31634462

# Set seed for reproducibility
set.seed(1128)
# splits the data in the ratio mentioned in SplitRatio. After splitting marks these rows as logical
# TRUE and the the remaining are marked as logical FALSE
sample <- sample.split(housing$price, SplitRatio = .75)
# creates a training dataset named train with rows which are marked as TRUE
train_mansion <- subset(housing, sample == TRUE)
# creates a training dataset named test with rows which are marked as FALSE
test_mansion <- subset(housing, sample == FALSE)


#Logistic model
model_glm <- glm(Mansion ~ price, family = "binomial", data = train_mansion)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_glm)
```

```
##
## Call:
## glm(formula = Mansion ~ price, family = "binomial", data = train_mansion)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.01157   0.00000   0.00000   0.00000   0.04722
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.189e+04  3.207e+04  -0.683    0.495
## price        2.188e-02  3.206e-02   0.683    0.495
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8.7767e+03  on 16955  degrees of freedom
## Residual deviance: 5.1348e-03  on 16954  degrees of freedom
## AIC: 4.0051
##
## Number of Fisher Scoring iterations: 25
```

```
#Predictions
predictions <- predict(model_glm, test_mansion, type = "response")
predictions$Mansion <- ifelse(predictions > 0.5, 1, 0)
```

```
## Warning in predictions$Mansion <- ifelse(predictions > 0.5, 1, 0): Coercing
## LHS to a list
```

```
predictionList <- unlist(predictions$Mansion)

#Accuracy calculation
accuracyList <- as.data.frame(ifelse(test_mansion$Mansion == predictionList, TRUE, FALSE))
colnames(accuracyList) <- c("Accuracy")

no_of_correct <- accuracyList %>% filter(Accuracy == TRUE) %>% count()
acc_percent <- (no_of_correct$n/nrow(accuracyList)) * 100
acc_percent
```

```
## [1] 100
```

We got an accuracy percent of 100.