




# MMA867 Assignment 1

## Technical Report

Jacqueline Mak  
Student #: 20311028

1540	Jacquelinemak		0.13626	6	now
<b>Your Best Entry</b> 					
Your submission scored 0.13626, which is an improvement of your previous score of 0.15088. Great job!					
 <a href="#">Tweet this!</a>					

## Overview

This report describes the methods used to handle missing data in regression and the approach to build a regression model for house price prediction. Two datasets are provided which include the training set and the test set. There are 79 explanatory variables describing the features of residential homes in Ames, Iowa and the goal is to predict the final price of each home.

## Data Cleaning & Missing Data

I begin looking at the dimensions and the structure of the train and test set.

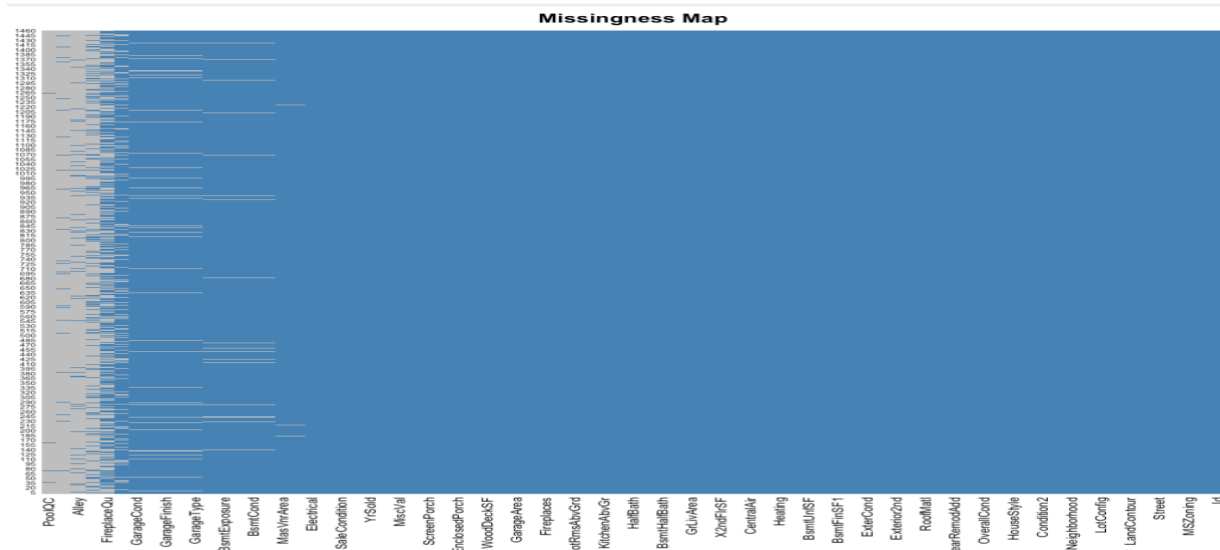
The train set consists of 81 variables with 1460 observations.

```
'data.frame': 1460 obs. of 81 variables:
 $ Id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ MSSubClass : int  60 20 60 70 60 50 20 60 50 190 ...
 $ MSZoning  : Factor w/ 5 levels "C (all)","FV",...: 4 4 4 4 4 4 4 4 4 5 4 ...
 $ LotFrontage : int  65 80 68 60 84 85 75 NA 51 50 ...
 $ LotArea    : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
```

The test set consists 80 variables with 1459 observations. The SalePrice column is missing, so I added a SalePrice column as NA which would make both datasets with the same number of variables.

```
'data.frame': 1459 obs. of 80 variables:
 $ Id      : int  1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 ...
 $ MSSubClass : int  20 20 60 60 120 60 20 60 20 20 ...
 $ MSZoning  : Factor w/ 5 levels "C (all)","FV",...: 3 4 4 4 4 4 4 4 4 4 ...
 $ LotFrontage : int  80 81 74 78 43 75 NA 63 85 70 ...
 $ LotArea    : int  11622 14267 13830 9978 5005 10000 7980 8402 10176 8400 ...
```

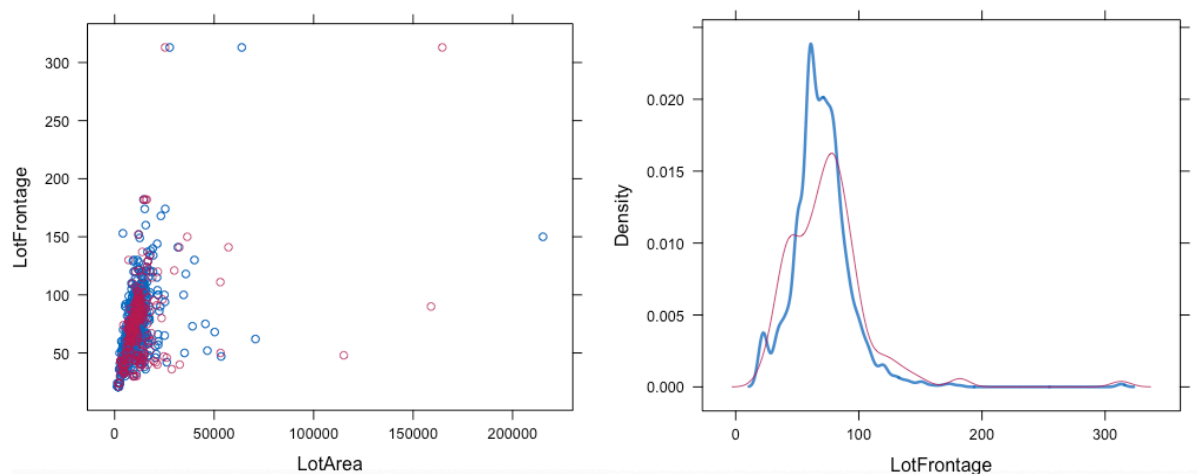
Before we start to build a regression model for house price prediction, it is important to fill in the missing data. I used the mice package to do the data imputation on both the train and test set. First, I want to visualize a high-level view of the missing data in the train set. In the Amelia package, this was accomplished using the missmap function indicating that 6% of data is missing.



I also want to look at the number of data missing under each variable. As you can see, PoolQC, MiscFeature, Alley and Fence are missing the majority of their data, so I believe it's reasonable to remove these variables. The remaining variables with missing data such as FireplaceQu is missing 47% of the observations, LotFrontage is missing 259, GarageType is missing 81 and etc. I will use the mice package to impute the missing data. The method I chose for the mice function is 'cart' because it works for all of the variable types including numeral, logical and factor, which makes it a lot easier to impute missing data.

PoolQC	MiscFeature	Alley	Fence	FireplaceQu	LotFrontage	GarageType
1453	1406	1369	1179	690	259	81
GarageYrBlt	GarageFinish	GarageQual	GarageCond	BsmtExposure	BsmtFinType2	BsmtQual
81	81	81	81	38	38	37
BsmtCond	BsmtFinType1	MasVnrType	MasVnrArea	Electrical	Id	MSSubClass
37	37	8	8	1	0	0

I then checked the imputed data to ensure the data is reasonably correct. Looking at the imputed data for LotFrontage, I used a scatter plot for visualization purposes and compared it to LotArea. The imputed data for LotFrontage highly correlated with LotArea, which make sense. Furthermore, I used a density plot to compare the imputed data for LotFrontage and it has a similar distribution as the original LotFrontage data. I then merged the imputed data to the original train set and lastly check there are no more missing values.



For the test set, I applied the same approach as I did for the train set to handle the missing data. As mentioned previously, since the test set doesn't have the SalePrice column, I added the column to the test set and then combined the train and test set using the rbind() function. I then dropped the Id column as it is unnecessary for the prediction process and dropped the Utilities column as it has zero variance. There are no missing data now except for the SalePrice column as NA.

SalePrice	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape
1459	0	0	0	0	0	0	0
LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	HouseStyle
0	0	0	0	0	0	0	0

## Predictive Modeling

### Simple linear regression

My first approach is to start off with a simple linear regression. Before I begin, I divided the data set back into train (row 1 to 1460) and test set (row 1461 to 2919). I ran the standard linear regression to predict house prices using only one variable OverallCond.

Output of the model:

```
Call:
lm(formula = SalePrice ~ OverallCond, data = training_houses_data)

Residuals:
    Min       1Q   Median       3Q      Max
-159924  -49459  -16590   30881  576439

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  211910     10597   19.997 < 0.0000000000000002 ***
OverallCond   -5558       1864   -2.982    0.00291 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

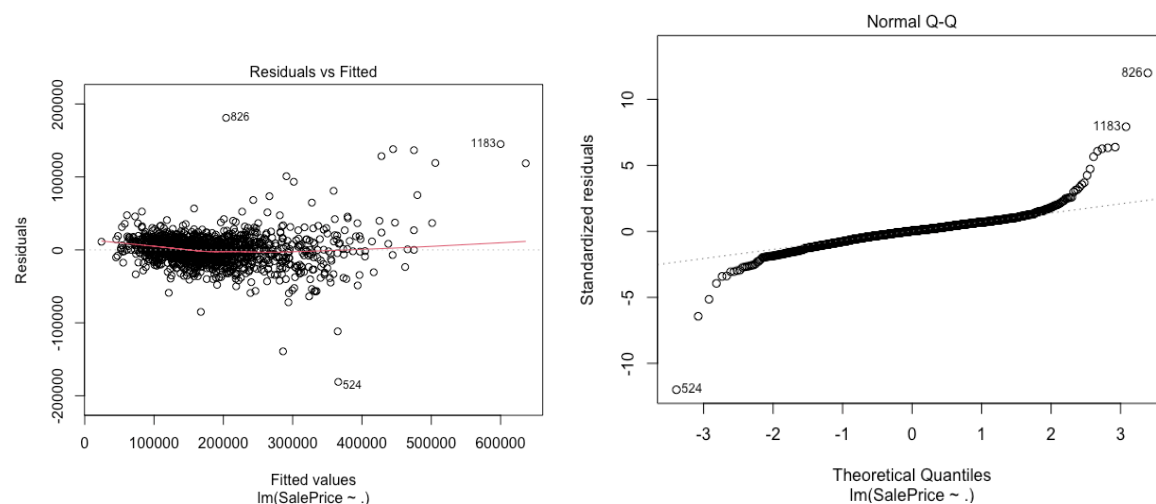
Residual standard error: 79230 on 1458 degrees of freedom
Multiple R-squared:  0.006062, Adjusted R-squared:  0.00538
F-statistic: 8.892 on 1 and 1458 DF, p-value: 0.002912
```

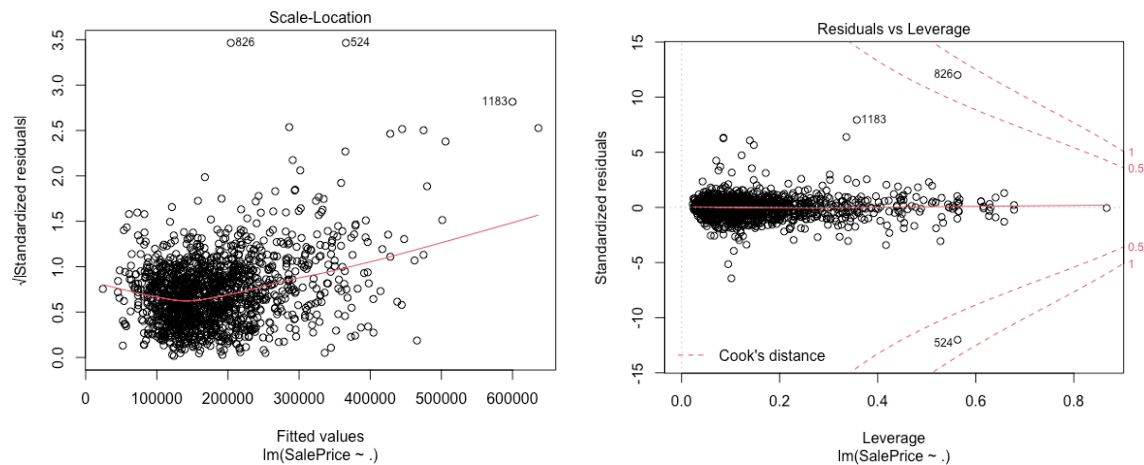
As per our regression summary statistics above, the model passes the joint F-test with a p-value (0.002912) that is lower than 0.05 which is the usual threshold. This indicates that the variable is non-zero and explains Y.

### Multiple linear regression

My second approach is to proceed with a multiple linear regression to predict the sale price using all of the variables with the train set.

I begin my modelling decision by analyzing the following plots:





Residuals vs Fitted: The data plotted should have no pattern. As you can see, the errors are not normally distributed.

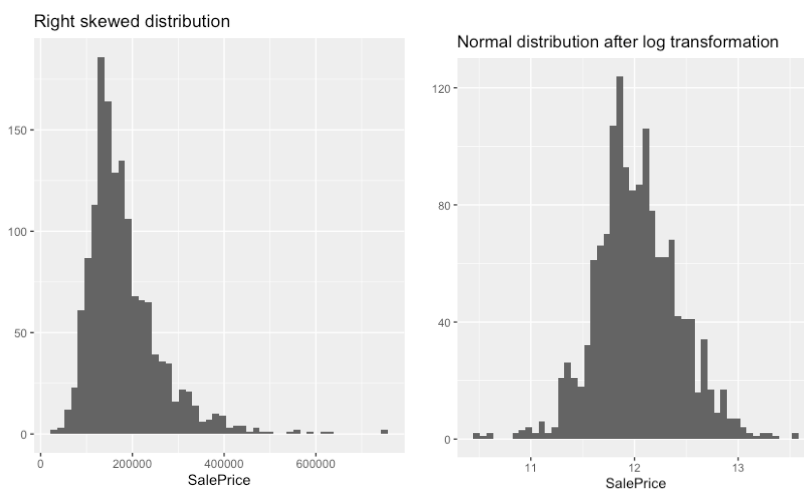
Normal Q-Q: The Q-Q plot gives insight into multivariate normality. As shown in the plot, the points are somewhat close to the diagonal line, which indicates there is a likelihood of having normality problems.

Scale-location: I used this plot to check if the points are spread randomly on either side (we want to see homoscedasticity). There is a cone shape pattern, which means there is evidence of heteroskedasticity.

Cook's Distance plot: You can see the dotted red lines which is a bad thing. Observations that are outside of those lines have high leverage and are worth investigating.

### Log-y linear regression

For the Log-y linear regression, I wanted to make the distribution of the target variable (SalePrice) from right skewed to normal, so I transformed the SalePrice variable by taking log in the model.

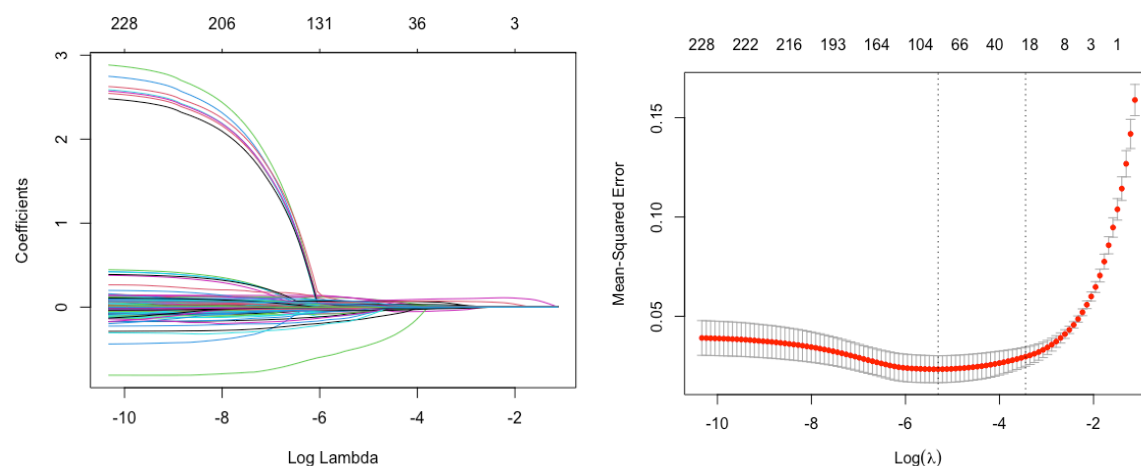


All three models have heteroskedasticity, multicollinearity and residuals that are not normally distributed. For that reason, another model should be built.

### Lasso regression

The next approach is to build a lasso regression. In order to evaluate the model before I use the whole training and predict the sale price with the testing set, I divided the train set into training (75% of the training data) and validation set (25% of the training data). I created a y variable and a matrix of x variables to ensure all interactions exist.

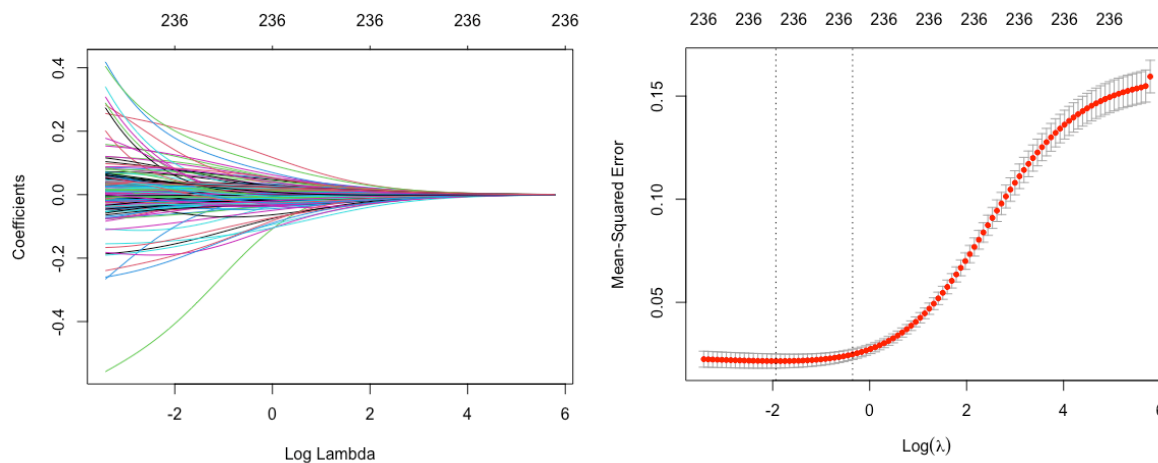
I begin my modelling decision by analyzing the following plot. As you can see, each curve corresponds with a coefficient. As lambda increases, the coefficient reduces, which means less variables are being used in the model. When lambda is small, you tend to include all of the variables in the model. If lambda is -2, that means only 2 variables are selected based on lambda. I chose the lambda with the smallest error and took the average of error using cross validation to select the best penalty lambda, which is 0.00299. As shown in the graph below, -5.306413 is the lowest point in lambda, which is also the best lambda with slightly more than 104 variables. I then estimated the model with the optimal penalty lambda. The variables with the 0 coefficient can be removed as lasso selected the best combination of variables for the model.



I predicted the SalePrice on the validation set and evaluated the model with the MSE and MAPE score. The MAPE is easier to interpret than the MSE score, as it indicates there is mean absolute percentage error of 11.11%.

### Ridge regression

My last approach is to build a ridge regression. I used the same steps as I did to build the lasso regression and the only difference is setting alpha to 0. As shown in the plot below on the left, the coefficients of ridge gradually converge to 0. As shown in the graph on the right below, -1.93394 is the lowest point in lambda, which is also the best lambda. I then estimated the model with the optimal penalty lambda and the ridge regression retains more variables than the lasso regression.



Lastly, I predicted the sale price using the ridge regression and evaluated the model with the MSE and MAPE score. The error for the ridge regression is 10.47% which is slightly lower than the MSE using the lasso regression.

For the final model selection, I compared the prediction error in the validation set and concluded the ridge regression is better because it has a lower minimum MSE. For that reason, I used the whole training set and final submission using ridge regression in the testing set to predict the sale prices.

```
Call: cv.glmnet(x = intrain_x, y = intrain_y, alpha = 0)
```

Measure: Mean-Squared Error

	Lambda	Index	Measure	SE	Nonzero
min	0.1474	84	0.01911	0.002886	229
1se	0.5950	69	0.02178	0.002968	229

## Appendix

### Leadership Board

Overview	Data	Code	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions	...
1534	Lubomír Straka						0.13612	3	23d
1535	Aditya Kumawat						0.13615	16	19d
1536	Brial Forestier						0.13616	3	1mo
1537	Vladimir Tugutov						0.13620	15	1mo
1538	Demo James						0.13621	45	2d
1539	Colin Chaigneau						0.13622	18	16d
1540	Jacquelinemak						0.13626	6	now

Your Best Entry ↑

Your submission scored 0.13626, which is an improvement of your previous score of 0.15088. Great job!

[Tweet this!](#)

### Regression Model

#### Ridge Regression

```
270 ridge.opt.fit <- glmnet(x = train_x, y = train_y, alpha = 0, lambda = ridge.penalty.lasso)
271 coef(ridge.opt.fit) #resultant model coefficients of the lasso.opt.fit model
272
273 test_x <- model.matrix(~., testing_set[-76])
274
275 SalePrice_Ridge <- exp(predict(ridge.opt.fit, s = ridge.penalty.lasso, newx = test_x))
276
```

