Time Series Analysis of Zillow's Home Value Zestimate®

Predicting the Log Error of Zestimate® values

Roberto Cancel, Kiran Singh, and Tyler Wolff

University of San Diego

ADS 506: Applied Time Series Analysis

Erin Cooke

December 6, 2021

Abstract

Zillow.com is the United States' most visited real estate website – offering customers a wide range of real estate services, including its proprietary Zestimate®. Real estate professionals and homeowners have scrutinized Zillow's Zestimates® due to inaccuracies since they heavily depend on the amount, recency, and accuracy of the data provided for each market and property. The Zillow Prize competition was launched to help Zillow decrease the log error of Zestimates®, which, at the time of the contest, was 4.5% nationwide - larger error rates exist in certain markets. Our project aims to predict the log error of Zillow's Zestimates using various time series forecasting models for Los Angeles, Orange, and Ventura, California. Since we decided to include housing features in our model, an extensive data wrangling process was conducted to remove our dataset's millions of missing data points. A minimum missingness threshold of five percent was established and greatly reduced the missingness. Complete Case analysis was used to rationalize the remaining missingness. Three models were developed: OLS, SES, and ARIMA. The ARIMA (3,0,0) or AR(3) model resulted in the lowest MAE of 0.0123, the Kaggle competition criterion for model selection. Predictions of October, November, and December 2017 log errors were made using the ARIMA (3,0,0), and the values within the 95% confidence interval were within the range of previous values. Our data was the best fit to an AR(3) model indicates that predicted values are based on the past three values of log error.

Keywords: ARIMA, Time Series, Exponential Smoothing, Error, Housing, Zestimate, Kaggle

Table of Contents

Introduction

Zillow.com is the United States' most visited real estate website – offering customers a wide range of real estate services, including its proprietary Zestimate®. To predict the selling price of a home or Zestimate®, Zillow uses an Automated Valuation Model (AVM). Their AVM incorporates "public, MLS and user-submitted data into Zillow's proprietary formula, also taking into account home facts, location and market trends" (Zillow Group, 2021). Therefore, Zestimate® accuracies heavily depend on the amount, recency, and accuracy of the data provided for each market and property. Consequently, Zillow has been criticized by real estate professionals and homeowners due to inaccurate Zestimate®.

In 2017, Zillow launched Zillow Prize, a Kaggle-based competition to improve its Zestimate® algorithms, with a one-million grand prize for the team that most improved its Zestimate® accuracy, measured as the median margin of error. At the onset of the competition, the Zestimate® nationwide median margin of error was 4.5 percent. The competition consisted of two rounds. The first round, which was open to the public, challenged participants to predict the Zestimate® log error for real estate properties in Los Angeles, Orange, and Venture, California for six time points in 2016 and 2017: October, November, and December of 2016 and 2017. Equation (1) shows the log error calculation, defined as the difference between the log of the Zestimate and the log of the actual sale price. Logarithmic transformations are widely used in time series analysis to improve the normality of the data to help scale the data for analysis and model fitting. The second round, consisting of the top 100 teams, challenged participants to revamp the Zestimate® algorithms using various machine learning approaches and beat its benchmark model.

$$\log \text{error} = \log(Zestimate) - \log(SalePrice) \quad (1)$$

"The winning team's algorithm incorporated several sophisticated machine learning techniques, including using deep neural networks to directly estimate home values and remove outlier data points that fed into their algorithm. They also leveraged publicly-available, external data including rental rates,

commute times, and home prices, among other types of contextual information, such as road noise – all variables that factor into a home's estimated value" (Nielson, 2019). As a result, they beat the benchmark model by 13 percent, which increased the average Zestimate® for a typical home valued at $239,900 by $1,300.

Rather than predicting the log error for the six time points in the Zillow competition, this project aims to develop an algorithm to predict the log error of Zillow's Zestimate® in October, November, and December of 2017 after fitting on the 2016 data using various time series forecasting models. Our models will be evaluated using the predicted log error's mean absolute error (MAE), compared to the actual log error, per the scoring criterion in the Kaggle competition. Predicting error rates over time would assist Zillow in improving its model, particularly for time-related errors, and enhancing its user experience. In addition, further algorithm improvements can be made once the residual errors are reduced.

Dataset description

The datasets used were sourced from Kaggle as a part of the Zillow Prize: Zillow's Home Value Prediction Zestimate® competition. They consisted of a list of 2,985,217 properties and their home features in Los Angeles, Orange, and Ventura, California for 2016 and 2017, along with log error and transaction dates for homes sold from January 1, 2016 to September 25, 2017. A total of five files were imported: Properties_2016, Properties_2017, train_2016_v2, train_2017, and zillow_data_dictionary. Each property file contained 58 columns with property-specific home features such as parcel id, bathroom and bedroom counts, and square footage. In addition, each training file contained parcel id, Zestimate® log error, and transaction date. Zillow_data_dictionary consisted of a list of feature acronyms and descriptions. The training Properties_2016 and Properties_2017 had extensive missing home feature values - 85,129,239 and 84,835,659, respectively. Missing values will be addressed later in the methodology section.

Literature Review

Understanding the Zestimate® Historical Issues

Zillow ranks markets on a one to four-star scale based on the expected pricing reliability. Reliability is based on market sales and assessment data recency and the completeness of property-specific information within a market. Four-star markets represent the "best" Zestimates®, and one-star markets represent markets that are challenging to price or solely rely on the tax assessor's value. Corcoran and Liu (2014) compared differences between Zestimates® and actual sale prices of 2,005 single-family residential properties in suburban St. Paul, Minnesota, a four-star market, and St. Louis, Missouri a one-star market, for November 2013. As expected, "Zestimate® accuracy is significantly better in the four-stark market as compared with the one-star market, with a mean difference between Zestimates® and sales prices of 17.15 percent and 30.48 percent, respectively" (Corcoran & Liu, 2014). Finally, they explored price point mean errors for each market and determined that for all price points except $203,000 to $253,000, which had a four-star mean error rate of 9.53 and a one-star mean error rate of 12.38%, mean error rates in both markets exceed the 10 percent industry threshold. While this information is dated, it provides context for the real estate and homeowner concerns about Zillow's Zestimate® tool.

Time Series Analysis of Housing Prices

Bimali et al. (2021) aimed to create "an innovative solution proposed to facilitate land valuation based on recent sales, prediction of future price, and the effect of proposed development work on the land, so that real-estate customers and owners of real estate companies can be benefitted and make smarter property-related decisions." Without an AI real estate service, such as Zillow, they intended to identify a similar technology-driven methodology for the Sri Lankan market (Bimali et al., 2021). Therefore, they attempted to model their data, primarily from surveys and real estate websites, with six

different approaches: Multivariate Linear Regression, Artificial Neural Networks, LSTM – Recurrent Neural Network, Auto-Regressive Integrated Moving Average (ARIMA) model, and a *k*-nearest neighbors (KNN) algorithm. The LSTM and ARIMA models were tested on time series data, while the remaining were tested on cross-sectional data. "These models were evaluated in terms of mean absolute error (MAE), mean standard error (MSE), and root mean squared error (RMSE)" (Bimali et al., 2021). LTSM outperformed the ARIMA in predicting the current and future values; however, the authors noted trade-offs existed for future work, "depending on the dataset being used and its sample size" (Bimali et al., 2021). Nevertheless, based on the size and comprehensiveness of the data set used in this study, it is believed an ARIMA model will perform well for predicting Zestimate® log error.

Time Series Analysis Model for Estimating Housing Unit Price

Determining a realistic housing unit price in Ghana is an issue of great concern because there seems to be a large disparity between what sellers think a house is worth and what buyers are prepared to pay for a housing unit. Consequently, both sellers and buyers have to negotiate to find an acceptable price for both parties. Boye et al. (2017) developed Multiple Linear Regression Model (MLRM) to determine Housing Unit Price (HUP) for one-bedroom and two-bedroom housing units using selected Housing Unit Major Components (HUMC). Boye et al. (2018) also developed Principal Components Regression Model (PCRM) to determine HUP for one-bedroom and two-bedroom housing units using the same selected HUMC used by Boye et al. (2017). They build models, primarily from surveys and real estate websites, with six different approaches: Multivariate Linear Regression, Artificial Neural Networks, LSTM – Recurrent Neural Network, Auto-Regressive Integrated Moving Average (ARIMA) model, and a k-nearest neighbors (KNN) algorithm. The LSTM and ARIMA models were tested on time series data, while the remaining were tested on cross-sectional data. "These models were evaluated in terms of mean absolute error (MAE), mean standard error (MSE), and root mean squared error (RMSE). In this study, TSAM (Time series analysis model) has been

developed from observed housing unit prices over a period of 15 consecutive years, obtained from an estate development agency, to determine realistic HUP for one-bedroom and two-bedroom housing units.

Zillow Home Value Prediction using XGBOOST

A home purchase is normally the single most expensive purchase in one's lifetime. Things like Zestimates allow buyers and sellers to get a prediction of the current value of said home. The basic linear regression algorithm used achieves a 0.113 accuracy. Rolli (2019). In the paper they used multiple different machine learning algorithms such as linear regression, gradient boost, random forest, and XGBoost against a dataset of three counties in California. They were able to take the price the home was estimated at or the Zestimate and compare it to the selling price to determine if it was high or low. Many different things come into factor when building this type of algorithm such as the number of bedrooms, square footage, and location to name a few. After cleaning the data and running it through the algorithms they evaluated the results using the mean absolute error method. XGB performed the best, but the majority of the models had similar results. The final result was that the XGBoost method out performed the benchmarked Random Forest Regression model.

Methodology

Data Wrangling:

After importing the provided data sets into R, their contents were explored and analyzed. Since the project aimed to predict the log error of the Zestimate®, which could only be calculated for homes that were sold, it was determined that the training (2016) and testing (2017) data should be left joined with their respective property data by parcel id. This resulted in comprehensive training and testing data sets, indexed by parcel id, containing the log error, transaction date, and features for homes sold in 2016 and 2017.

While the transactional data was complete, housing feature columns contained extensive missing values - 2,537,678 missing home feature values for 2016 and 2,173,827 missing home feature values for 2017. Pratama et al. (2016) reviewed methods for handling missing data in times series analysis. Since the missing home features were Missing at random (MAR) or the "probability of missingness is depending only on the available information" (Pratama et al., 2016), the missing data mechanism is said to be ignorable. A missingness threshold of five percent was established for home features based on Schafer (1999)'s determination that a missing rate of five percent or greater impacted the usefulness and accuracy of statistical inferences. This also limits the number of observations lost during a complete case review. Thirty-five home features were removed based on this threshold, and 24 features were retained - reducing our missing values by 99.999% for each dataset. The remaining features contained redundancies: eight features described location, three features described property and land type identifiers, four features described bathroom counts, and three features described tax valuation. Using the descriptions from data_dict, the single-most comprehensive feature for each group was retained - which resulted in a reduction of an additional thirteen features and eleven features left for modeling.

Since the Ordinary Least Squares (OLS) model used home features for predicting log error and, therefore, could impacted by multicollinearity, multicollinearity will be addressed during the OLS model implementation and development if aliased features are detected. To avoid bias introduction via imputation, only complete cases were retained. By only retaining complete cases, our training set lost 3.61% of observations and the test set lost 3.14% of observations. Finally, the cleaned train and test sets were transformed into time series datasets with frequency of 365 and all parameters were transformed to a mean value per day to avoid multiple values per day and ensure stationarity.

***Data Exploration*:**

The mean Log error each day was plotted over time (see Appendix) and its distribution over time resembled white noise with a mean near .01 with possible conditional heteroscedasticity due to a non-constant variance of log error over time. Stationarity was confirmed by evaluating its trend with linear regression. Since the slope of 0.006 was statistically insignificant and the mean was consistent over time, the log error is deemed stationary. The autocorrelation function (ACF), which "measures the linear predictability of the series at time t, say x_t, using only the value x_s" (Shumway & Stoffer 2019), and partial autocorrelation function (PACF), which calculates "the correlation between x_{t+h} and x_t with the linear dependence of everything between them, namely $\{x_{t+1}, \ldots, x_{t+h-1}\}$, on each removed" (Shumway & Stoffer 2019), were plotted. Since the ACF and PACF tailed off over time, an autoregressive moving average (ARMA) model would likely fit the data well according to Table 4.1 in Shumway and Stoffer (2019).

Model Development:

To explore the ability to predict log error using housing features, Multiple linear regression, which is an extension of Ordinary Least Squares (OLS), was the first model developed. - see Equation 2. Backward selection, which creates a baseline model with all features and then iteratively removes features to create the most parsimonious or simple model, was deployed with the goal of minimizing Akaike's Information Criterion (AIC), which aims to balance the accuracy of the model's fit with the number of parameters included, as the penalty for feature selection and reduction.

$$x_t = \beta_0 + \beta_1 z_{t1} + \beta_2 z_{t2} + \cdots + \beta_1 z_{tq} + w_t , \quad (2)$$

where $\beta_0, \beta_1, \ldots, \beta_q$ are unknown fixed regression coefficients, and {wt} is white normal noise with variance σ_w^2.

The most influential parameters for predicting log error were property land-use type id, tax value dollar count, and bathroom count. Figure (in Appendix) shows that the predicted error hovered near the mean, likely due to the method chosen to ensure stationarity, and only 18.3% of the variance

of log error was explained. The plot of the residuals indicate the residuals are normally distributed or that conditional heteroscedasticity is not present. Since "linear regression models are often unsatisfactory for explaining all of the interesting dynamics of a time series" (Shumway & Stoffer 2019), Simple Exponential Smoothing (SES) for predicting log error was explored without housing features.

SES, a time series forecasting method for univariate data without a trend or seasonality, produces weighted averages of past values controlled by a smoothing parameter α. A smaller value for α indicates more weight is applied to older observations, while a larger value applies more weight to recent values. Unfortunately, due to technical limitations, the SES could not be used with the exogenous home features previously determined. It should be noted that SES is a very accurate univariate time series method for predicting log error over time.

"Adding nonstationary to ARMA models leads to the autoregressive integrated moving average (ARIMA) model" (Shumway & Stoffer 2019). While our statistically insignificant slope indicates stationarity, out of an abundance of caution an ARIMA, which integrates a differencing component to compensate for trends, was developed to predict log error over time using our OLS identified parameters. The model is presented in the form ARIMA (*p,d,q*), where *p* represents the autoregressive formula, *d* represents the amount of differencing applied to the time series, and *q* represents the moving average component (Shumway & Stoffer 2019). The auto.arima() function in R utilizes a variation of the Hyndman-Khandakar algorithm (Hyndman & Khandakar, 2008), which combines unit root tests, minimisation of the AICc and MLE to obtain the most appropriate ARIMA model. Once deployed, the auto.arima() function found that an AR(3), as initially hypothesized, model best fit the data. The residuals were, again, normally distributed.

Results

Model Selection:

Based on the Kaggle Zillow competition requirements, MAE is our model evaluation metric of choice. The ARIMA (3,0,0) model resulted in the lowest MAE at 0.0123. It should also be noted that the SES model could not be fitted to work with the housing features. The evaluation metrics for each model are available in the Appendix. The training and test data sets were then joined and October, November, and December data was filtered out. The auto.arima fit was used to forecast October, November, and December 2017 log errors. The model evaluation metrics are available near the end of the Appendix.

Preliminary Results:

The ARIMA (3,0,0) model was used to predict October, November, and December 2017 log errors. Within the 95% confidence interval, our predicted values appear to be in line with previous values.

Insights Gained:

Since the ARIMA (3,0,0) model resulted in the lowest MAE using housing features, our data resembles an AR(3) model - meaning that our predictions were based on the previous three values.

Discussion

Strengths & Limitations:

The current limitations with the dataset selected are the age of the data, and the amount of missing values. These limitations made it difficult to perform a complete time series analysis. The data set did yield tons of information that was of value, if it wasn't for the missing features the data would have been perfect for time series analysis.

Implications for Future Research:

For future research it would be best to get data from multiple different geographic locations. The current project looks at counties in California. It would be best practice to see how the Zestimates across the nation work, not just in California. With the current datasets being from 2016-2017, it would be a good idea to have more up to date data. It is important to note that based on the extent of missing

values and without using imputations it was difficult to perform time series analysis on this dataset. Overall, this problem set once expanded on could produce exponential results that benefit millions of Zillow users daily.

References

Bimali, Y., Rodrigo, S., Dharmaseelan, T., Thayalini, K., Gamage, A., & Rathnayaka, P. (2021, June 27). *A machine learning-based solution for predicting land values*. A Machine Learning-Based Solution for Predicting Land Values. Retrieved November 30, 2021, from https://www.researchgate.net/profile/Thenuka-Dharmaseelan/publication/352780787_A_Machine_Learning-Based_Solution_for_Predicting_Land_Values_E-Valuer_Land_value_predictor

Boye, P., Mireku-Gyimah, D., & Sadiq, H. (2019, March 19). *Time Series Analysis Model for Estimating Housing Unit price*. Time Series Analysis Model for Estimating the Price of a Housing Unit. Retrieved December 6, 2021, from http://www2.umat.edu.gh/gjt/index.php/gjt/article/view/172.

Corcoran, C., & Liu, F. (2014). Accuracy of Zillow's Home Value Estimates. *Real Estate Issues*, *39*(1), 46–49.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: TheforecastPackage Forr. *Journal of Statistical Software*, *27*(3). https://doi.org/10.18637/jss.v027.i03

Nielson, C. (2019, January 30). *What is zillow? - zillow*. Zillow Awards $1 Million to Team that Built a Better Zestimate. Retrieved December 3, 2021, from https://zillow.mediaroom.com/2019-01-30-Zillow-Awards-1-Million-to-Team-that-Built-a-Better-Zestimate.

Pratama, I., Permanasari, A. E., Ardiyanto, I., & Indrayani, R. (2016). A review of missing values handling methods on time-series data. *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*. https://doi.org/10.1109/icitsi.2016.7858189

Rolli, C. S. (2019, October 23). Zillow home value prediction using XGBOOST. Retrieved December 6, 2021, from https://scholarworks.calstate.edu/downloads/3197xr874.

Schafer, J. L. (1999). Multiple imputation: A Primer. *Statistical Methods in Medical Research*, *8*(1), 3–15. https://doi.org/10.1191/096228099671525676

Shumway, R., & Stoffer, D. (2019). *Time Series: A Data Analysis Approach Using R*. Chapman and Hall/CRC.

Zillow Group. (2021, October 26). *What is a zestimate? Zillow's zestimate accuracy*. Zillow. Retrieved December 3, 2021, from https://www.zillow.com/z/zestimate/.

Appendix (R Markdown)

Import Libraries

```
#Import Libraries
library(readr) # for quick importing of files
library(readxl) # for importing excel files
library(dplyr) # for easy data manipulation
library(ggplot2) # for generating plots
library(forecast) # for time series analysis
library(knitr) # for table generation and other functions
```

Import the data

```
#read csv and excel files. Suppress warnings due to size of raw datasets
properties_2016 <- read_csv("properties_2016.csv", col_types = cols())
properties_2017 <- read_csv('properties_2017.csv', col_types = cols())
data_dict <- read_excel("zillow_data_dictionary.xlsx")
train_2016 <- read_csv("train_2016_v2.csv", col_types = cols())
test_2017 <- read_csv("train_2017.csv", col_types = cols())
```

Exploratory Data Analysis

View the data

```
head(properties_2016) # 2016 parcel id and housing attributes (described in data_dict) for ALL
houses in this market

## # A tibble: 6 x 58
##   parcelid airconditioning~ architecturalst~ basementsqft bathroomcnt bedroomcnt
##      <dbl>            <dbl>            <dbl>        <dbl>       <dbl>      <dbl>
## 1 10754147               NA               NA           NA           0          0
## 2 10759547               NA               NA           NA           0          0
## 3 10843547               NA               NA           NA           0          0
## 4 10859147               NA               NA           NA           0          0
## 5 10879947               NA               NA           NA           0          0
## 6 10898347               NA               NA           NA           0          0
## # ... with 52 more variables: buildingclasstypeid <dbl>,
## #   buildingqualitytypeid <dbl>, calculatedbathnbr <dbl>, decktypeid <dbl>,
## #   finishedfloor1squarefeet <dbl>, calculatedfinishedsquarefeet <dbl>,
## #   finishedsquarefeet12 <dbl>, finishedsquarefeet13 <dbl>,
## #   finishedsquarefeet15 <dbl>, finishedsquarefeet50 <dbl>,
## #   finishedsquarefeet6 <dbl>, fips <chr>, fireplacecnt <dbl>,
## #   fullbathcnt <dbl>, garagecarcnt <dbl>, garagetotalsqft <dbl>, ...
```

Comment: Contains parcel id and home attribues for ALL houses in this market. We note extensive missing values in properties_2016.

```
head(properties_2017) # 2017 parcel id and housing attributes (described in data_dict)

## # A tibble: 6 x 58
##   parcelid airconditioning~ architecturalst~ basementsqft bathroomcnt bedroomcnt
##      <dbl>            <dbl>            <dbl> <lgl>              <dbl>      <dbl>
## 1 10754147               NA               NA NA                     0          0
## 2 10759547               NA               NA NA                     0          0
## 3 10843547               NA               NA NA                     0          0
```

```
## 4 10859147               NA               NA NA                      0          0
## 5 10879947               NA               NA NA                      0          0
## 6 10898347               NA               NA NA                      0          0
## # ... with 52 more variables: buildingclasstypeid <dbl>,
## #   buildingqualitytypeid <dbl>, calculatedbathnbr <dbl>, decktypeid <dbl>,
## #   finishedfloor1squarefeet <dbl>, calculatedfinishedsquarefeet <dbl>,
## #   finishedsquarefeet12 <dbl>, finishedsquarefeet13 <dbl>,
## #   finishedsquarefeet15 <dbl>, finishedsquarefeet50 <dbl>,
## #   finishedsquarefeet6 <dbl>, fips <chr>, fireplacecnt <dbl>,
## #   fullbathcnt <dbl>, garagecarcnt <dbl>, garagetotalsqft <dbl>, ...
```

Comment: Contains parcel id and home attributes for ALL houses in this market. We note extensive missing values in properties_2017.

```
head(train_2016)
```

```
## # A tibble: 6 x 3
##   parcelid logerror transactiondate
##      <dbl>    <dbl> <date>
## 1 11016594   0.0276 2016-01-01
## 2 14366692  -0.168  2016-01-01
## 3 12098116  -0.004  2016-01-01
## 4 12643413   0.0218 2016-01-02
## 5 14432541  -0.005  2016-01-02
## 6 11509835  -0.270  2016-01-02
```

Comment: Contains parcel id, log error, and transaction date - needs to be joined with properties_2016. After joining only the properties that sold (have logerror values) will be retained.

```
head(test_2017)
```

```
## # A tibble: 6 x 3
##   parcelid logerror transactiondate
##      <dbl>    <dbl> <date>
## 1 14297519  0.0256  2017-01-01
## 2 17052889  0.0556  2017-01-01
## 3 14186244  0.00538 2017-01-01
## 4 12177905 -0.103   2017-01-01
## 5 10887214  0.00694 2017-01-01
## 6 17143294 -0.0205  2017-01-01
```

Comment: Contains parcel id, log error, and transaction date - needs to be joined with properties_2017. After joining only the properties that sold (have logerror values) will be retained.

```
#since this is a dictionary, print as a table for easy reference
kable(data_dict%>%select(`Housing Feature` = Feature, Description),
      caption = 'Table 1: List of Attributes with description')
```

Table 1: List of Attributes with description

Housing Feature	Description
'airconditioningtypeid'	Type of cooling system present in the home (if any)

Housing Feature	Description
'architecturalstyletypeid'	Architectural style of the home (i.e. ranch, colonial, split-level, etc…)
'basementsqft'	Finished living area below or partially below ground level
'bathroomcnt'	Number of bathrooms in home including fractional bathrooms
'bedroomcnt'	Number of bedrooms in home
'buildingqualitytypeid'	Overall assessment of condition of the building from best (lowest) to worst (highest)
'buildingclasstypeid'	The building framing type (steel frame, wood frame, concrete/brick)
'calculatedbathnbr'	Number of bathrooms in home including fractional bathroom
'decktypeid'	Type of deck (if any) present on parcel
'threequarterbathnbr'	Number of 3/4 bathrooms in house (shower + sink + toilet)
'finishedfloor1squarefeet'	Size of the finished living area on the first (entry) floor of the home
'calculatedfinishedsquarefeet'	Calculated total finished living area of the home
'finishedsquarefeet6'	Base unfinished and finished area
'finishedsquarefeet12'	Finished living area
'finishedsquarefeet13'	Perimeter living area
'finishedsquarefeet15'	Total area
'finishedsquarefeet50'	Size of the finished living area on the first (entry) floor of the home
'fips'	Federal Information Processing Standard code - see https://en.wikipedia.org/wiki/FIPS_county_code for more details
'fireplacecnt'	Number of fireplaces in a home (if any)
'fireplaceflag'	Is a fireplace present in this home
'fullbathcnt'	Number of full bathrooms (sink, shower + bathtub, and toilet) present in home
'garagecarcnt'	Total number of garages on the lot including an attached garage
'garagetotalsqft'	Total number of square feet of all garages on lot including an attached garage
'hashottuborspa'	Does the home have a hot tub or spa
'heatingorsystemtypeid'	Type of home heating system

Housing Feature	Description
'latitude'	Latitude of the middle of the parcel multiplied by 10e6
'longitude'	Longitude of the middle of the parcel multiplied by 10e6
'lotsizesquarefeet'	Area of the lot in square feet
'numberofstories'	Number of stories or levels the home has
'parcelid'	Unique identifier for parcels (lots)
'poolcnt'	Number of pools on the lot (if any)
'poolsizesum'	Total square footage of all pools on property
'pooltypeid10'	Spa or Hot Tub
'pooltypeid2'	Pool with Spa/Hot Tub
'pooltypeid7'	Pool without hot tub
'propertycountylandusecode'	County land use code i.e. it's zoning at the county level
'propertylandusetypeid'	Type of land use the property is zoned for
'propertyzoningdesc'	Description of the allowed land uses (zoning) for that property
'rawcensustractandblock'	Census tract and block ID combined - also contains blockgroup assignment by extension
'censustractandblock'	Census tract and block ID combined - also contains blockgroup assignment by extension
'regionidcounty'	County in which the property is located
'regionidcity'	City in which the property is located (if any)
'regionidzip'	Zip code in which the property is located
'regionidneighborhood'	Neighborhood in which the property is located
'roomcnt'	Total number of rooms in the principal residence
'storytypeid'	Type of floors in a multi-story house (i.e. basement and main level, split-level, attic, etc.). See tab for details.
'typeconstructiontypeid'	What type of construction material was used to construct the home
'unitcnt'	Number of units the structure is built into (i.e. 2 = duplex, 3 = triplex, etc…)
'yardbuildingsqft17'	Patio in yard
'yardbuildingsqft26'	Storage shed/building in yard
'yearbuilt'	The Year the principal residence was built
'taxvaluedollarcnt'	The total tax assessed value of the parcel
'structuretaxvaluedollarcnt'	The assessed value of the built structure on the parcel
'landtaxvaluedollarcnt'	The assessed value of the land area of the parcel

Housing Feature	Description
'taxamount'	The total property tax assessed for that assessment year
'assessmentyear'	The year of the property tax assessment
'taxdelinquencyflag'	Property taxes for this parcel are past due as of 2015
'taxdelinquencyyear'	Year for which the unpaid propert taxes were due

Describe the data

```
# Print the size of each data set (rows then columns)
dim(properties_2016) # Contains 2985217 rows and 58 columns

## [1] 2985217      58

dim(properties_2017) # Contains 2985217 rows and 58 columns

## [1] 2985217      58

dim(data_dict) # Contains 58 rows and 2 columns

## [1] 58  2

dim(train_2016) # Contains 90275 rows and 3 columns

## [1] 90275     3

dim(test_2017) # Contains 77613 and 3 columns

## [1] 77613     3

# Confirm no duplicates in properties data sets
uniqueproperties2016 <- unique(properties_2016$parcelid)
length(uniqueproperties2016) # All unique properties

## [1] 2985217

uniqueproperties2017 <- unique(properties_2017$parcelid)
length(uniqueproperties2017) # All unique properties

## [1] 2985217

uniquesales2016 <- unique(train_2016$parcelid)
length(uniquesales2016) # 90275 - 90150 = 125 properties sold more than once in 2016

## [1] 90150

uniquesales2017 <- unique(test_2017$parcelid)
length(uniquesales2017)# 77613 - 77414 = 199 properties sold more than once in 2017

## [1] 77414

# Count missing values in each data set
sum(is.na(properties_2016))

## [1] 85129239

sum(is.na(properties_2017))

## [1] 84835659
```

```
sum(is.na(train_2016))

## [1] 0

sum(is.na(test_2017))

## [1] 0
```

Join Properties to their respective Train/Test Data Sets to retain only sold home data

```
sold_train <- left_join(train_2016, properties_2016, by = "parcelid") # to join by parcelid & only retain sold homes
sold_test <- left_join(test_2017, properties_2017, by = "parcelid") # to join by parcelid & only retain sold homes
```

Comment: Join the properties & train/test (2016/2017, respectively) by parcel id to retain only the houses sold (with logerror values) before handling missing data.

Confirm joined data sets only contain sold homes

```
dim(train_2016)

## [1] 90275     3

dim(sold_train)

## [1] 90275    60

dim(test_2017)

## [1] 77613     3

dim(sold_test)

## [1] 77613    60

head(sold_train) #visually inspect joined training data set

## # A tibble: 6 x 60
##   parcelid logerror transactiondate airconditioningtypeid architecturalstyletyp~
##      <dbl>    <dbl> <date>                          <dbl>                  <dbl>
## 1 11016594   0.0276 2016-01-01                          1                     NA
## 2 14366692  -0.168  2016-01-01                         NA                     NA
## 3 12098116  -0.004  2016-01-01                          1                     NA
## 4 12643413   0.0218 2016-01-02                          1                     NA
## 5 14432541  -0.005  2016-01-02                         NA                     NA
## 6 11509835  -0.270  2016-01-02                          1                     NA
## # ... with 55 more variables: basementsqft <dbl>, bathroomcnt <dbl>,
## #   bedroomcnt <dbl>, buildingclasstypeid <dbl>, buildingqualitytypeid <dbl>,
## #   calculatedbathnbr <dbl>, decktypeid <dbl>, finishedfloor1squarefeet <dbl>,
## #   calculatedfinishedsquarefeet <dbl>, finishedsquarefeet12 <dbl>,
## #   finishedsquarefeet13 <dbl>, finishedsquarefeet15 <dbl>,
## #   finishedsquarefeet50 <dbl>, finishedsquarefeet6 <dbl>, fips <chr>,
## #   fireplacecnt <dbl>, fullbathcnt <dbl>, garagecarcnt <dbl>, ...

head(sold_test) # visually inspect joined test data set

## # A tibble: 6 x 60
##   parcelid logerror transactiondate airconditioningtypeid architecturalstyletyp~
```

```
##      <dbl>    <dbl> <date>                          <dbl>                  <dbl>
## 1 14297519  0.0256  2017-01-01                        NA                     NA
## 2 17052889  0.0556  2017-01-01                        NA                     NA
## 3 14186244  0.00538 2017-01-01                        NA                     NA
## 4 12177905 -0.103   2017-01-01                        NA                     NA
## 5 10887214  0.00694 2017-01-01                         1                     NA
## 6 17143294 -0.0205  2017-01-01                        NA                     NA
## # ... with 55 more variables: basementsqft <lgl>, bathroomcnt <dbl>,
## #   bedroomcnt <dbl>, buildingclasstypeid <dbl>, buildingqualitytypeid <dbl>,
## #   calculatedbathnbr <dbl>, decktypeid <dbl>, finishedfloor1squarefeet <dbl>,
## #   calculatedfinishedsquarefeet <dbl>, finishedsquarefeet12 <dbl>,
## #   finishedsquarefeet13 <dbl>, finishedsquarefeet15 <dbl>,
## #   finishedsquarefeet50 <dbl>, finishedsquarefeet6 <dbl>, fips <chr>,
## #   fireplacecnt <dbl>, fullbathcnt <dbl>, garagecarcnt <dbl>, ...
```

Investigate and Handle Missing Values / Feature Redundancy

```
#Determine the extent of missing data in each data set
sum(is.na(sold_train)) # 2,537,678 missing values

## [1] 2537678

sum(is.na(sold_test)) # 2,173,827 missing values

## [1] 2173827
```

Comment: Extensive missing data exists - we will explore the extent of missing data per attribute.

```
summary(sold_train) # displays basic descriptive statistics and # of NA's per feature

##     parcelid            logerror        transactiondate
##  Min.   : 10711738   Min.   :-4.60500   Min.   :2016-01-01
##  1st Qu.: 11559500   1st Qu.:-0.02530   1st Qu.:2016-04-05
##  Median : 12547337   Median : 0.00600   Median :2016-06-14
##  Mean   : 12984656   Mean   : 0.01146   Mean   :2016-06-11
##  3rd Qu.: 14227552   3rd Qu.: 0.03920   3rd Qu.:2016-08-19
##  Max.   :162960842   Max.   : 4.73700   Max.   :2016-12-30
##
##  airconditioningtypeid architecturalstyletypeid  basementsqft
##  Min.   : 1.00         Min.   : 2.00            Min.   : 100.0
##  1st Qu.: 1.00         1st Qu.: 7.00            1st Qu.: 407.5
##  Median : 1.00         Median : 7.00            Median : 616.0
##  Mean   : 1.82         Mean   : 7.23            Mean   : 713.6
##  3rd Qu.: 1.00         3rd Qu.: 7.00            3rd Qu.: 872.0
##  Max.   :13.00         Max.   :21.00            Max.   :1555.0
##  NA's   :61494         NA's   :90014            NA's   :90232
##   bathroomcnt       bedroomcnt     buildingclasstypeid buildingqualitytypeid
##  Min.   : 0.000   Min.   : 0.000   Min.   :4           Min.   : 1.00
##  1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.:4           1st Qu.: 4.00
##  Median : 2.000   Median : 3.000   Median :4           Median : 7.00
##  Mean   : 2.279   Mean   : 3.032   Mean   :4           Mean   : 5.57
##  3rd Qu.: 3.000   3rd Qu.: 4.000   3rd Qu.:4           3rd Qu.: 7.00
##  Max.   :20.000   Max.   :16.000   Max.   :4           Max.   :12.00
##                                    NA's   :90259       NA's   :32911
##  calculatedbathnbr   decktypeid    finishedfloor1squarefeet
##  Min.   : 1.000    Min.   :66      Min.   :  44
##  1st Qu.: 2.000    1st Qu.:66      1st Qu.: 938
##  Median : 2.000    Median :66      Median :1244
##  Mean   : 2.309    Mean   :66      Mean   :1348
##  3rd Qu.: 3.000    3rd Qu.:66      3rd Qu.:1614
```

```
##  Max.   :20.000    Max.   :66      Max.   :7625
##  NA's   :1182      NA's   :89617   NA's   :83419
##  calculatedfinishedsquarefeet finishedsquarefeet12 finishedsquarefeet13
##  Min.   :    2               Min.   :    2        Min.   :1056
##  1st Qu.: 1184               1st Qu.: 1172        1st Qu.:1392
##  Median : 1540               Median : 1518        Median :1440
##  Mean   : 1773               Mean   : 1745        Mean   :1405
##  3rd Qu.: 2095               3rd Qu.: 2056        3rd Qu.:1440
##  Max.   :22741               Max.   :20013        Max.   :1584
##  NA's   :661                 NA's   :4679         NA's   :90242
##  finishedsquarefeet15 finishedsquarefeet50 finishedsquarefeet6
##  Min.   :  560        Min.   :  44         Min.   : 257
##  1st Qu.: 1648        1st Qu.: 938         1st Qu.:1112
##  Median : 2104        Median :1248         Median :2028
##  Mean   : 2380        Mean   :1356         Mean   :2303
##  3rd Qu.: 2862        3rd Qu.:1619         3rd Qu.:3431
##  Max.   :22741        Max.   :8352         Max.   :7224
##  NA's   :86711        NA's   :83419        NA's   :89854
##      fips            fireplacecnt    fullbathcnt      garagecarcnt
##  Length:90275       Min.   :1.00    Min.   : 1.000   Min.   : 0.00
##  Class :character   1st Qu.:1.00    1st Qu.: 2.000   1st Qu.: 2.00
##  Mode  :character   Median :1.00    Median : 2.000   Median : 2.00
##                     Mean   :1.19    Mean   : 2.241   Mean   : 1.81
##                     3rd Qu.:1.00    3rd Qu.: 3.000   3rd Qu.: 2.00
##                     Max.   :5.00    Max.   :20.000   Max.   :24.00
##                     NA's   :80668   NA's   :1182     NA's   :60338
##  garagetotalsqft  hashottuborspa heatingorsystemtypeid    latitude
##  Min.   :   0.0   Mode:logical   Min.   : 1.00         Min.   :33339295
##  1st Qu.:   0.0   TRUE:2365      1st Qu.: 2.00         1st Qu.:33811538
##  Median : 433.0   NA's:87910     Median : 2.00         Median :34021500
##  Mean   : 345.5                  Mean   : 3.93         Mean   :34005411
##  3rd Qu.: 484.0                  3rd Qu.: 7.00         3rd Qu.:34172742
##  Max.   :7339.0                  Max.   :24.00         Max.   :34816009
##  NA's   :60338                   NA's   :34195
##    longitude          lotsizesquarefeet    poolcnt       poolsizesum
##  Min.   :-119447865   Min.   :    167   Min.   :1       Min.   :  28.0
##  1st Qu.:-118411692   1st Qu.:   5703   1st Qu.:1       1st Qu.: 420.0
##  Median :-118173431   Median :   7200   Median :1       Median : 500.0
##  Mean   :-118198868   Mean   :  29110   Mean   :1       Mean   : 519.8
##  3rd Qu.:-117921588   3rd Qu.:  11686   3rd Qu.:1       3rd Qu.: 600.0
##  Max.   :-117554924   Max.   :6971010   Max.   :1       Max.   :1750.0
##                       NA's   :10150     NA's   :72374   NA's   :89306
##   pooltypeid10    pooltypeid2     pooltypeid7    propertycountylandusecode
##  Min.   :1       Min.   :1       Min.   :1       Length:90275
##  1st Qu.:1       1st Qu.:1       1st Qu.:1       Class :character
##  Median :1       Median :1       Median :1       Mode  :character
##  Mean   :1       Mean   :1       Mean   :1
##  3rd Qu.:1       3rd Qu.:1       3rd Qu.:1
##  Max.   :1       Max.   :1       Max.   :1
##  NA's   :89114   NA's   :89071   NA's   :73578
##  propertylandusetypeid propertyzoningdesc rawcensustractandblock
##  Min.   : 31.0         Length:90275       Length:90275
##  1st Qu.:261.0         Class :character   Class :character
##  Median :261.0         Mode  :character   Mode  :character
##  Mean   :261.8
##  3rd Qu.:266.0
##  Max.   :275.0
##
##   regionidcity    regionidcounty regionidneighborhood  regionidzip
##  Min.   :  3491   Min.   :1286   Min.   :  6952       Min.   : 95982
##  1st Qu.: 12447   1st Qu.:1286   1st Qu.: 46736       1st Qu.: 96193
```

```
##  Median : 25218   Median :3101   Median :118887       Median : 96393
##  Mean   : 33761   Mean   :2525   Mean   :190647       Mean   : 96586
##  3rd Qu.: 45457   3rd Qu.:3101   3rd Qu.:274800       3rd Qu.: 96987
##  Max.   :396556   Max.   :3101   Max.   :764167       Max.   :399675
##  NA's   :1803                    NA's   :54263        NA's   :35
##     roomcnt        storytypeid    threequarterbathnbr typeconstructiontypeid
##  Min.   : 0.000   Min.   :7       Min.   :1.00        Min.   : 4.00
##  1st Qu.: 0.000   1st Qu.:7       1st Qu.:1.00        1st Qu.: 6.00
##  Median : 0.000   Median :7       Median :1.00        Median : 6.00
##  Mean   : 1.479   Mean   :7       Mean   :1.01        Mean   : 6.01
##  3rd Qu.: 0.000   3rd Qu.:7       3rd Qu.:1.00        3rd Qu.: 6.00
##  Max.   :18.000   Max.   :7       Max.   :4.00        Max.   :13.00
##                   NA's   :90232   NA's   :78266       NA's   :89976
##     unitcnt       yardbuildingsqft17 yardbuildingsqft26   yearbuilt
##  Min.   :  1.00   Min.   :  25.0     Min.   :  18.0     Min.   :1885
##  1st Qu.:  1.00   1st Qu.: 180.0     1st Qu.: 100.0     1st Qu.:1953
##  Median :  1.00   Median : 259.5     Median : 159.0     Median :1970
##  Mean   :  1.11   Mean   : 310.1     Mean   : 311.7     Mean   :1969
##  3rd Qu.:  1.00   3rd Qu.: 384.0     3rd Qu.: 361.0     3rd Qu.:1987
##  Max.   :143.00   Max.   :2678.0     Max.   :1366.0     Max.   :2015
##  NA's   :31922    NA's   :87629      NA's   :90180      NA's   :756
##  numberofstories fireplaceflag  structuretaxvaluedollarcnt taxvaluedollarcnt
##  Min.   :1.00    Mode:logical   Min.   :    100            Min.   :      22
##  1st Qu.:1.00    TRUE:222       1st Qu.:  81245            1st Qu.:  199023
##  Median :1.00    NA's:90053     Median : 132000            Median :  342872
##  Mean   :1.44                   Mean   : 180093            Mean   :  457673
##  3rd Qu.:2.00                   3rd Qu.: 210534            3rd Qu.:  540589
##  Max.   :4.00                   Max.   :9948100            Max.   :27750000
##  NA's   :69705                  NA's   :380                NA's   :1
##  assessmentyear landtaxvaluedollarcnt   taxamount        taxdelinquencyflag
##  Min.   :2015   Min.   :      22      Min.   :    49.1   Length:90275
##  1st Qu.:2015   1st Qu.:   82228      1st Qu.:  2872.8   Class :character
##  Median :2015   Median :  192970      Median :  4542.8   Mode  :character
##  Mean   :2015   Mean   :  278335      Mean   :  5984.0
##  3rd Qu.:2015   3rd Qu.:  345420      3rd Qu.:  6901.1
##  Max.   :2015   Max.   :24500000      Max.   :321936.1
##                 NA's   :1             NA's   :6
##  taxdelinquencyyear censustractandblock
##  Min.   : 6.0       Min.   :6.037e+13
##  1st Qu.:13.0       1st Qu.:6.037e+13
##  Median :14.0       Median :6.038e+13
##  Mean   :13.4       Mean   :6.049e+13
##  3rd Qu.:15.0       3rd Qu.:6.059e+13
##  Max.   :99.0       Max.   :6.111e+13
##  NA's   :88492      NA's   :605
```

Comment: All missing values are property-specific attributes rather than transactional attributes. We will initially remove all property-specific attributes with 5% or more missing data before complete case analysis.

```
na_train <- data.frame(col = as.character(colnames(sold_train)),
                       pct_null = colSums(is.na(sold_train))*100/(colSums(is.na(sold_train))+colS
ums(!is.na(sold_train))))%>%
  filter(col != 'parcelid')

train <- sold_train[,colnames(sold_train) %in% na_train$col[na_train$pct_null < 5]] #retain at
tributes w/ <5% missing
```

```
test <- sold_test[,colnames(sold_test) %in% na_train$col[na_train$pct_null < 5]] #retain same attributes in test
removed <- sold_train[,colnames(sold_train) %in% na_train$col[na_train$pct_null > 5]]
```

Print Rationalized Features

```
colnames(removed) # 35 features removed
```

```
##  [1] "airconditioningtypeid"    "architecturalstyletypeid"
##  [3] "basementsqft"             "buildingclasstypeid"
##  [5] "buildingqualitytypeid"    "decktypeid"
##  [7] "finishedfloor1squarefeet" "finishedsquarefeet12"
##  [9] "finishedsquarefeet13"     "finishedsquarefeet15"
## [11] "finishedsquarefeet50"     "finishedsquarefeet6"
## [13] "fireplacecnt"             "garagecarcnt"
## [15] "garagetotalsqft"          "hashottuborspa"
## [17] "heatingorsystemtypeid"    "lotsizesquarefeet"
## [19] "poolcnt"                  "poolsizesum"
## [21] "pooltypeid10"             "pooltypeid2"
## [23] "pooltypeid7"              "propertyzoningdesc"
## [25] "regionidneighborhood"     "storytypeid"
## [27] "threequarterbathnbr"      "typeconstructiontypeid"
## [29] "unitcnt"                  "yardbuildingsqft17"
## [31] "yardbuildingsqft26"       "numberofstories"
## [33] "fireplaceflag"            "taxdelinquencyflag"
## [35] "taxdelinquencyyear"
```

Print Retained Features for further rationalization

```
colnames(train) # 24 retained features
```

```
##  [1] "logerror"                   "transactiondate"
##  [3] "bathroomcnt"                "bedroomcnt"
##  [5] "calculatedbathnbr"          "calculatedfinishedsquarefeet"
##  [7] "fips"                       "fullbathcnt"
##  [9] "latitude"                   "longitude"
## [11] "propertycountylandusecode"  "propertylandusetypeid"
## [13] "rawcensustractandblock"     "regionidcity"
## [15] "regionidcounty"             "regionidzip"
## [17] "roomcnt"                    "yearbuilt"
## [19] "structuretaxvaluedollarcnt" "taxvaluedollarcnt"
## [21] "assessmentyear"             "landtaxvaluedollarcnt"
## [23] "taxamount"                  "censustractandblock"
```

Address Redundcancies in Features

We notice redundancies in location data (fips, latitude & longitude, regionidcity, regionidcounty, rawcensustractandblock, censustractblock, and regionidzip) we will retain regionzip since it provides city/county information and lat/long are too property-specific.

We also notice redundancies in propertycountylandusecode, propertylandusetypeid, and landusetypeid. We will retain property landusetypeid.

We also we notice redundancies in bathroomcnt, calculatedbathnbr, and fullbathcnt. We will retain bathroomcnt.

We also notice redundancies in structuretaxvaluedollarcnt, taxvaluedollarcnt, and landtaxvaluedollarcnt. We will retain the cumulative taxvaluedollarcnt.

```
loc_red <- c("fips", "latitude", "longitude", "regionidcity", "regionidcounty", "rawcensustrac
tandblock", "censustractandblock", "propertycountylandusecode", "calculatedbathnbr", "fullbath
cnt", "structuretaxvaluedollarcnt", "landtaxvaluedollarcnt", "assessmentyear" )
train.df <- train[ , !(names(train) %in% loc_red)]
test.df <- test[ , !(names(test) %in% loc_red)]
```

Comment:

```
# Calculate the percentage change in missing data from removing features >5% missing & feature
reduction
sum(is.na(train.df))/sum(is.na(sold_train))-1 *100 # 6,613 missing values

## [1] -99.99943

sum(is.na(test.df))/sum(is.na(sold_test))-1 *100 # 4,343 missing values

## [1] -99.99962
```

Comment: Missing data is SIGNIFICANTLY reduced by removing attributes/features with >5% missing values

```
# Complete Cases
train_fin <- train.df[complete.cases(train), ]
test_fin <- test.df[complete.cases(test), ]

# Calculate % of observations retained
nrow(train_fin)/nrow(train) *100

## [1] 96.39989

nrow(test_fin)/nrow(test) *100

## [1] 96.86522
```

Comment: Complete Cases repesent 96.4% of our final training set and 96.9% of our final testing set.

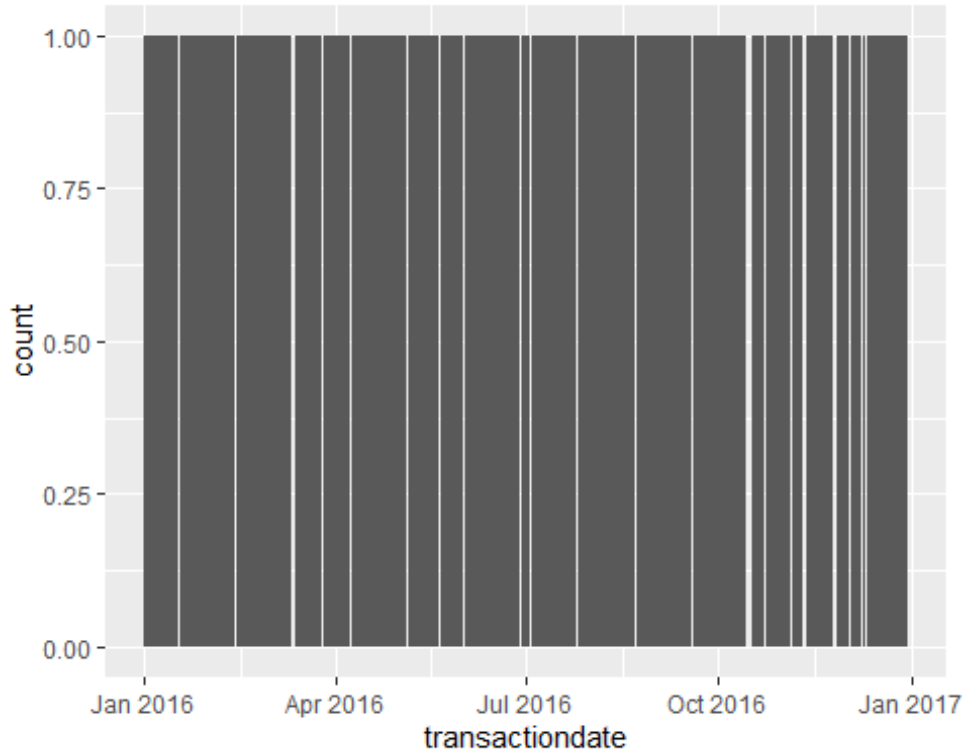
Since our data loss is minimal (~3-4%) we will proceed with complete case review rather than imputatino.

Transform Time variable

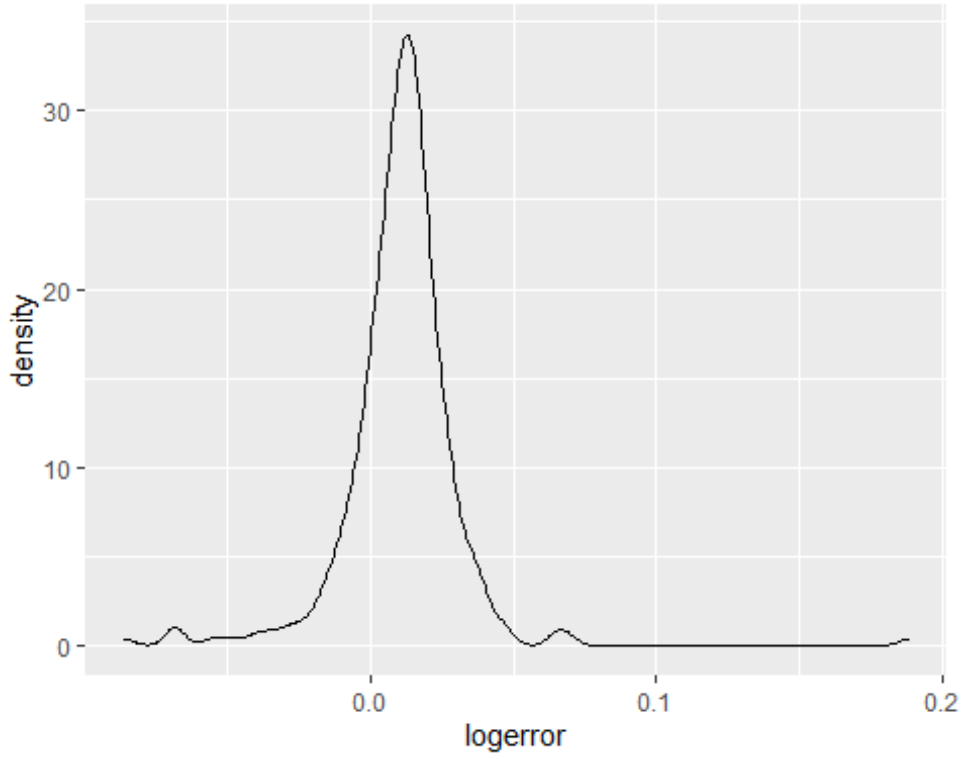
```
train.clean <- train_fin%>%
    group_by(transactiondate)%>%
  summarise_all(funs(mean)) # shows only the mean value per day rather than each value of the
day

test.clean <- test_fin%>%
  group_by(transactiondate)%>%
  summarise_all(funs(mean)) # shows only the mean value per day rather than each value of the
day
```

Outlier Detection

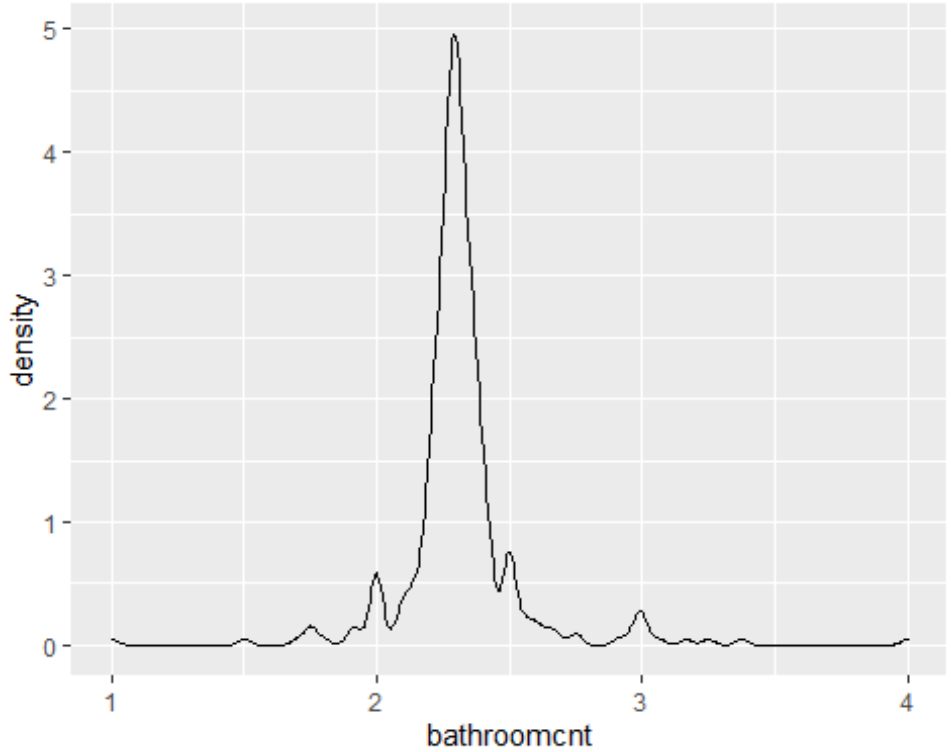
```
my_plots <- lapply(names(train.clean), function(var_x){
  p <-
    ggplot(train.clean) +
    aes_string(var_x)
  if(is.numeric(train.clean[[var_x]])) {
    p <- p + geom_density()
  } else {
    p <- p + geom_bar()
  }
})
my_plots

## [[1]]
```



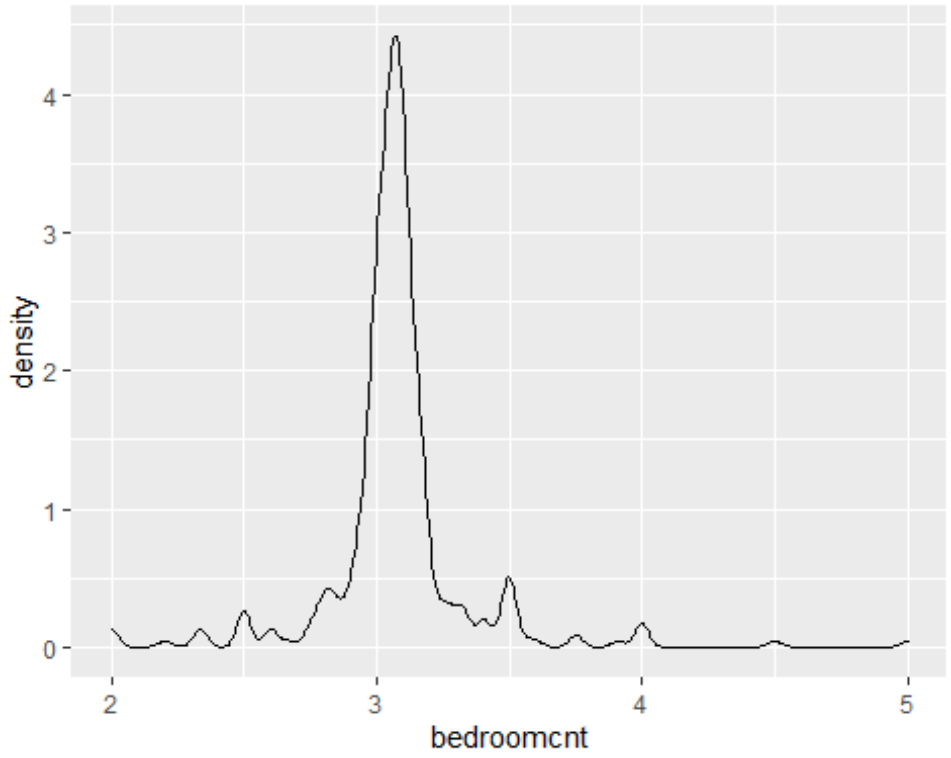

```
##
## [[2]]
```


30
20
10
0
density
0.0
0.1
0.2
logerror

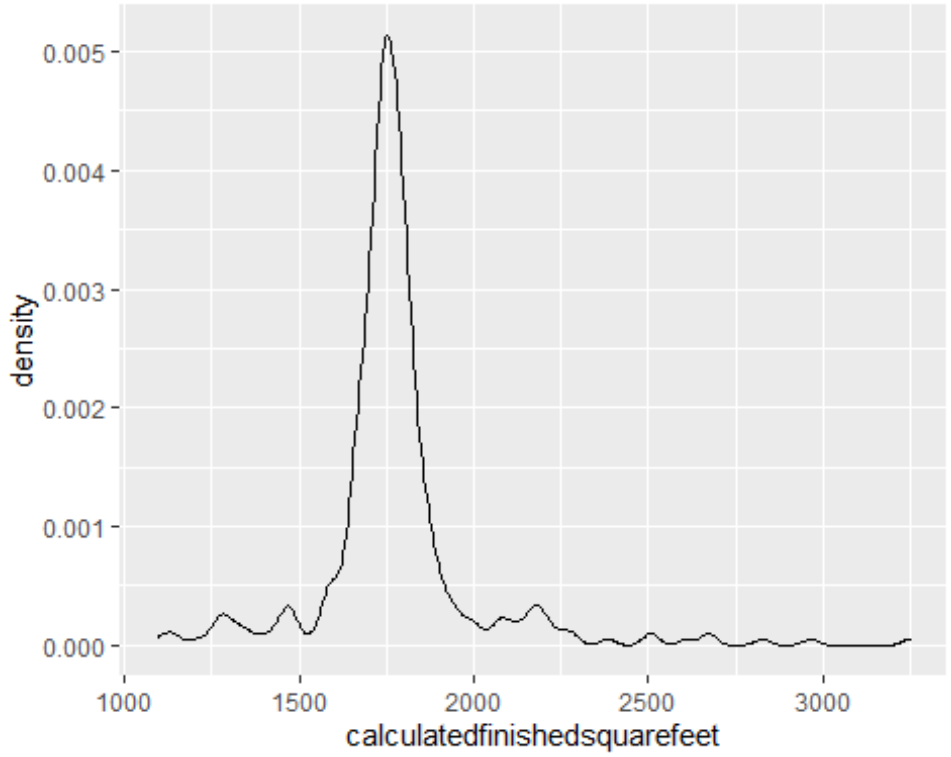
```
##
## [[3]]
```


5
4
3
2
1
0
density
1
2
3
4
bathroomcnt

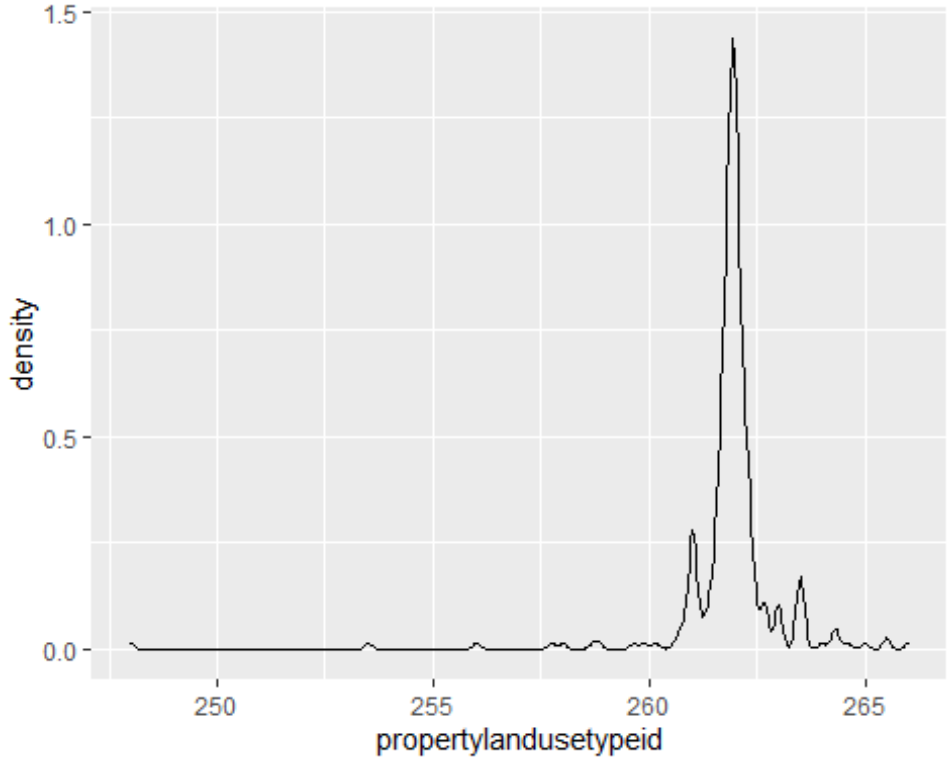
```
##
## [[4]]
```



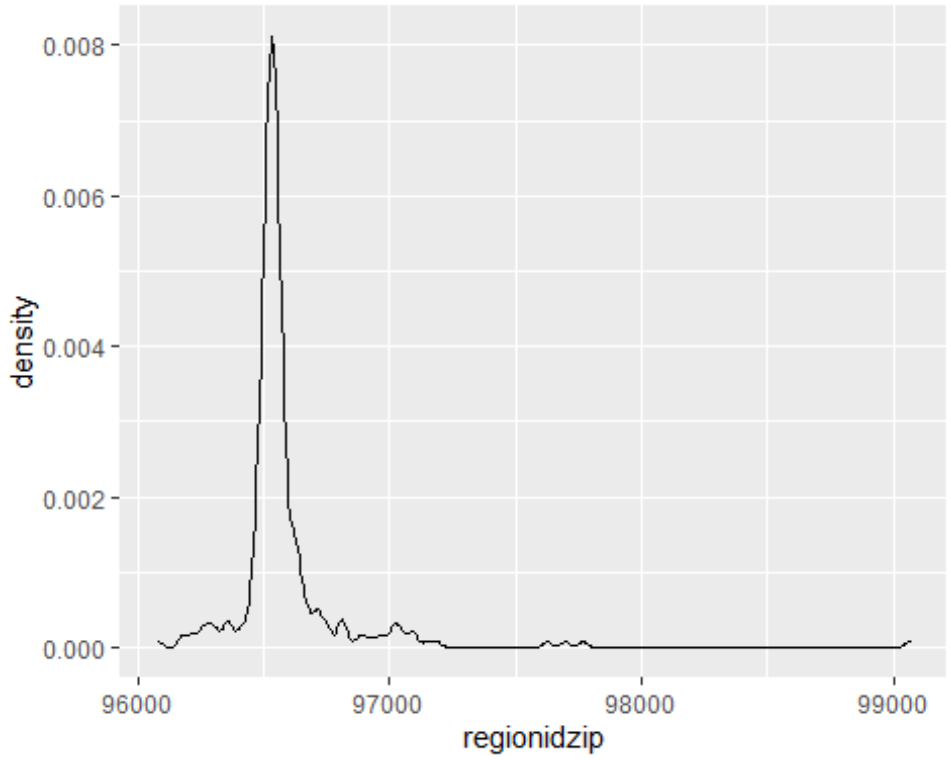

```
##
## [[5]]
```


0.005
0.004
0.003
0.002
0.001
0.000
density
1000
1500
2000
2500
3000
calculatedfinishedsquarefeet

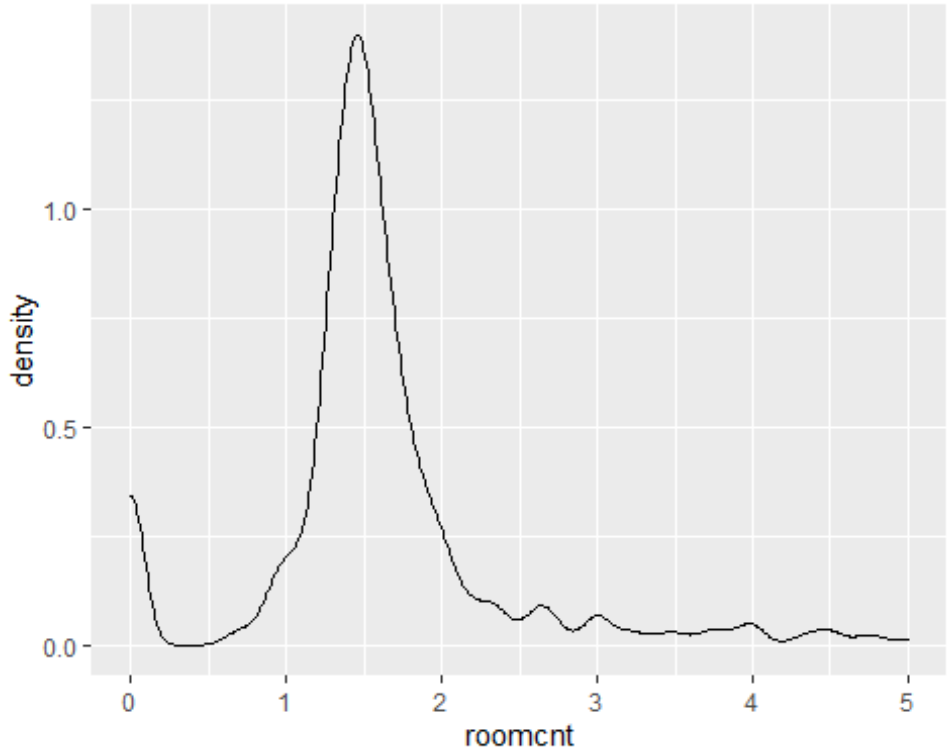
```
##
## [[6]]
```


1.5
1.0
0.5
0.0
density
250
255
260
265
propertylandusetypeid

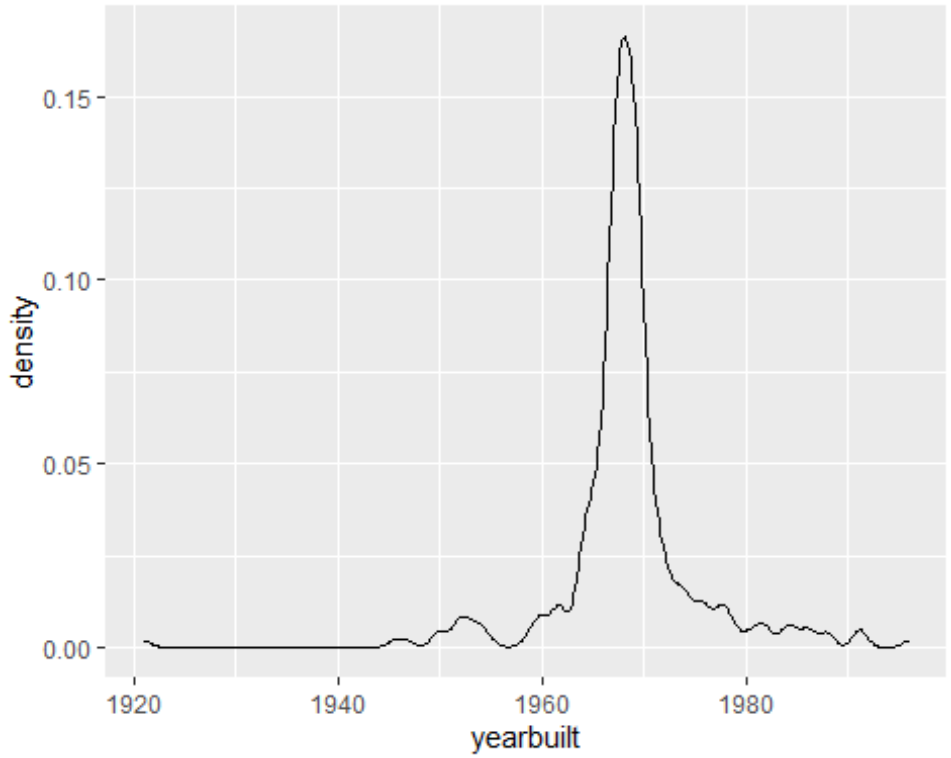
```
##
## [[7]]
```


0.008
0.006
0.004
0.002
0.000
density
96000
97000
98000
99000
regionidzip

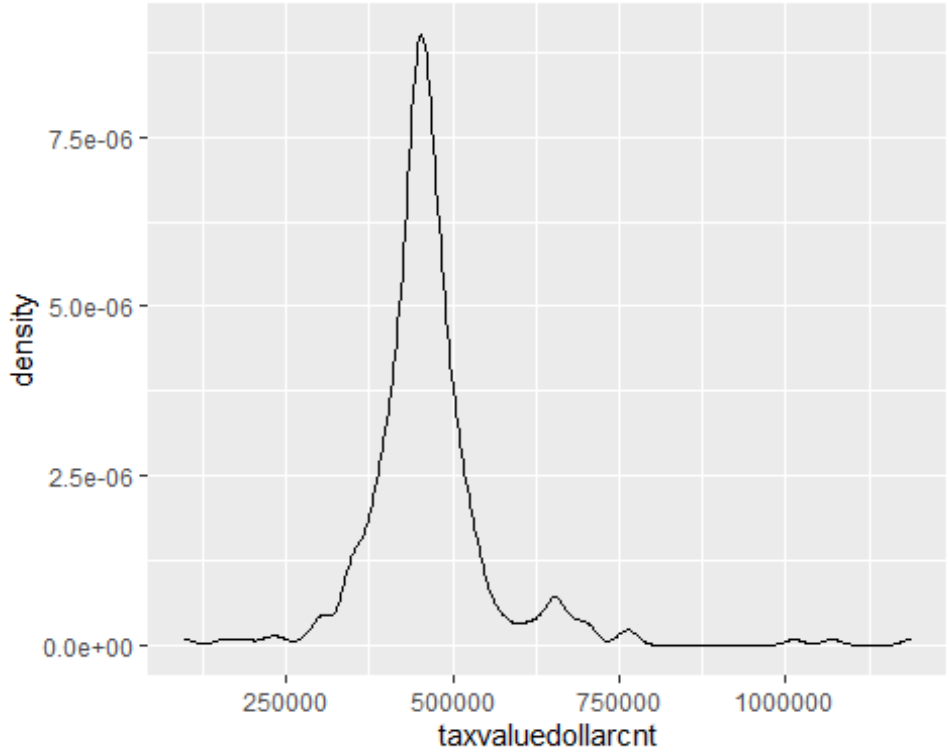
```
##
## [[8]]
```


density
1.0
0.5
0.0
0
1
2
3
4
5
roomcnt

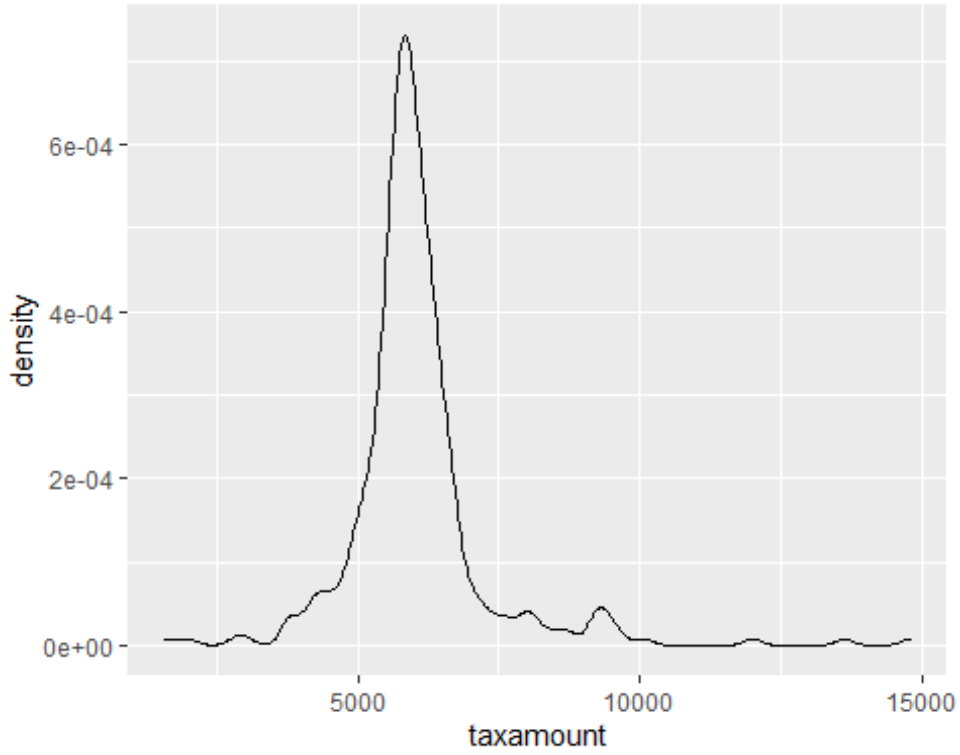
```
##
## [[9]]
```


density
0.15
0.10
0.05
0.00
1920
1940
1960
1980
yearbuilt

```
##
## [[10]]
```

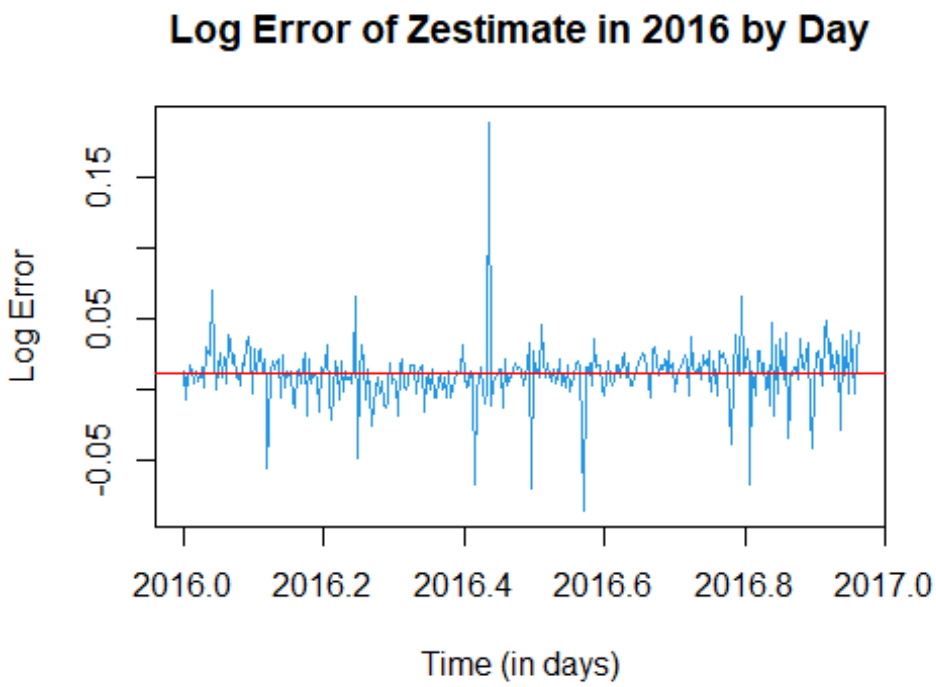

7.5e-06
5.0e-06
2.5e-06
0.0e+00
density
250000
500000
750000
1000000
taxvaluedollarcnt

```
##
## [[11]]
```



Evaluating the Log Error over time

```
zillow.ts <- ts(train.clean, frequency = 365, start=c(2016,1,1))
logerror <- zillow.ts[,'logerror']

test.ts <- ts(test.clean, frequency = 365, start=c(2017,1,1))
logerror_test <- test.ts[,'logerror']

ts.plot(logerror, ylab="Log Error",xlab = "Time (in days)", main="Log Error of Zestimate in 20
16 by Day", col=4)
logrerror_mean <- mean(logerror)
abline( h=logrerror_mean, col="red")
```



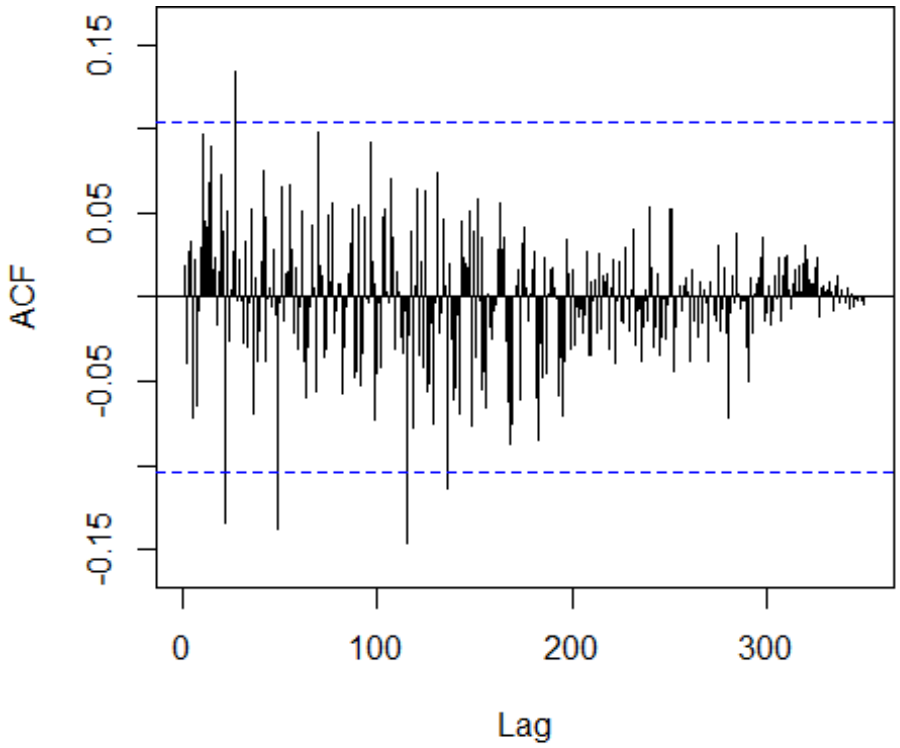
Comment: The Log Error resembles white noise with a mean near .01. Also the variance over time is not consistent (note postive and negative spikes centered around the mean) - indicating potential conditional heteroscedacity. We will confirm stationarity be evaluating its trend with linear regression.

```
#Confirming no trend exists
fit <- lm(logerror~time(zillow.ts), na.action=NULL)
summary(fit)

##
## Call:
## lm(formula = logerror ~ time(zillow.ts), na.action = NULL)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.097156 -0.006970  0.001248  0.008506  0.177795
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -10.828483   7.767627  -1.394    0.164
## time(zillow.ts)   0.005375   0.003852   1.395    0.164
##
## Residual standard error: 0.02012 on 350 degrees of freedom
## Multiple R-squared:  0.005533,   Adjusted R-squared:  0.002691
## F-statistic: 1.947 on 1 and 350 DF,  p-value: 0.1638
```

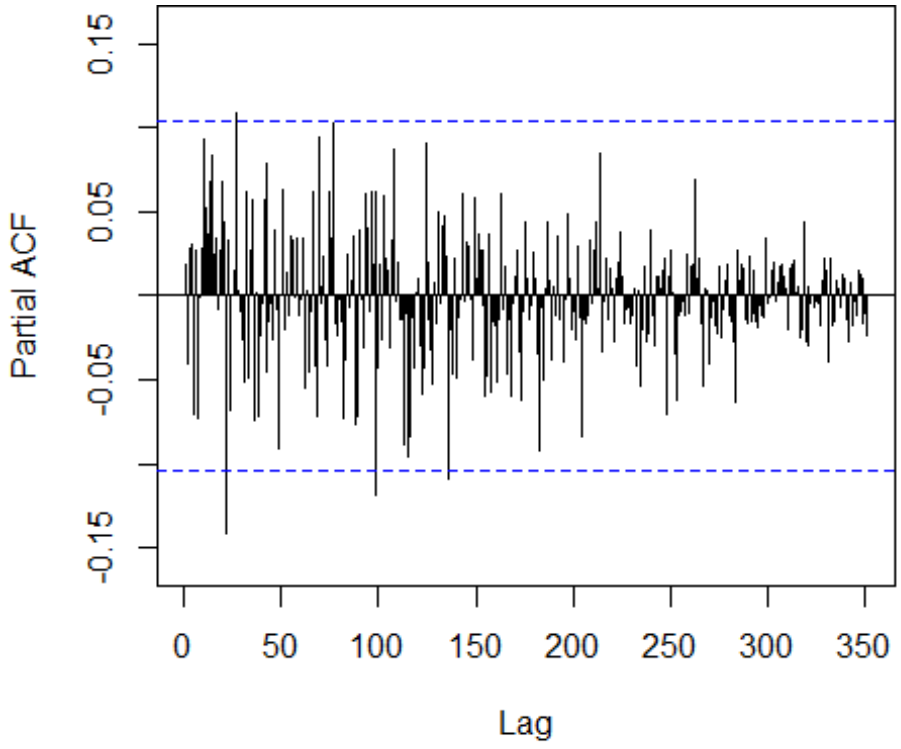
Comment: with a statistically insignificant slope coefficient of 0.005 - log error has no trend. This combined with it's consant mean indicates stationarity but with possible conditional heteroscedacity - which will be investigated addressed for the OLS and ARIMA models.

Evaluating Log Error over time

```
par(mar=c(5,4,0,2)+.01)
Acf(logerror, lag.max = 365)
```




```
par(mar=c(5,4,0,2)+.01)
Pacf(logerror)
```



Comment: The ACF and PACF tail off - suggesting an ARMA model - like AR(3).

Model Development:

Ordinary Least Squares with backward selection

```
library(olsrr)

##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers

library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:olsrr':
##
##     cement

## The following object is masked from 'package:dplyr':
##
##     select

# stepwise regression
model <- lm(logerror ~ ., data = train.clean%>%dplyr::select(-transactiondate))
ols_step_both_p(model,pent=.03, progress= FALSE)
```

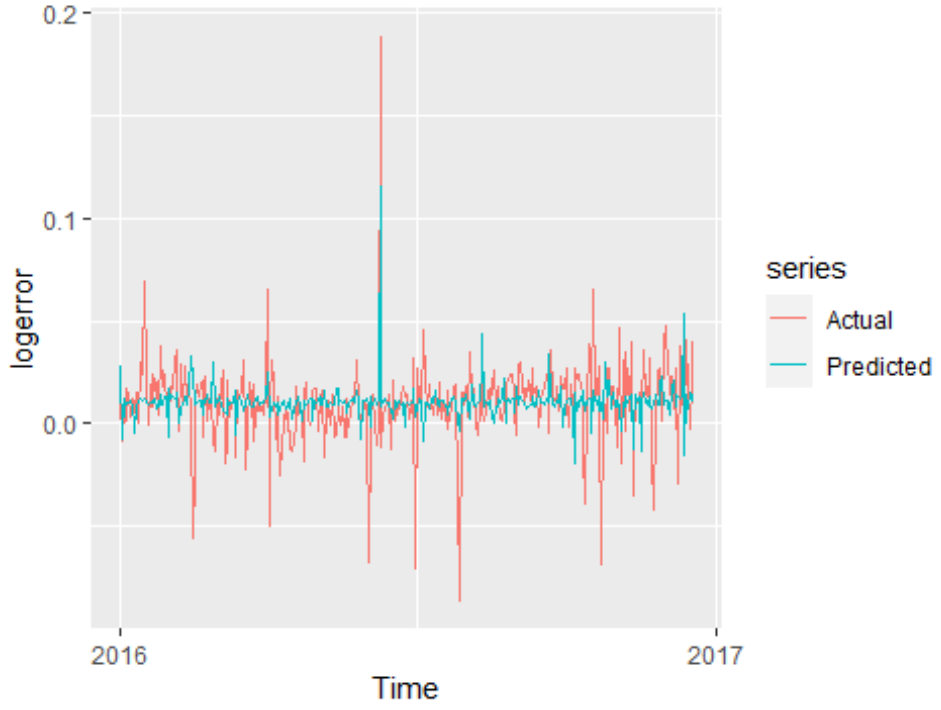
```
##
##                                     Stepwise Selection Summary
## ----------------------------------------------------------------------------------------------
---------
##                                  Added/                   Adj.
## Step          Variable           Removed     R-Square    R-Square     C(p)         AIC
RMSE
## ----------------------------------------------------------------------------------------------
---------
##    1    propertylandusetypeid    addition       0.144       0.141    17.9570    -1799.6400
0.0187
##    2         bathroomcnt         addition       0.156       0.151    14.8300    -1802.6073
0.0186
##    3      taxvaluedollarcnt      addition       0.182       0.175     5.4390    -1811.8982
0.0183
## ----------------------------------------------------------------------------------------------
---------
```

```
lm.fit <- tslm(logerror ~ propertylandusetypeid+bathroomcnt+taxvaluedollarcnt, data = zillow.t
s)
summary(lm.fit)
```

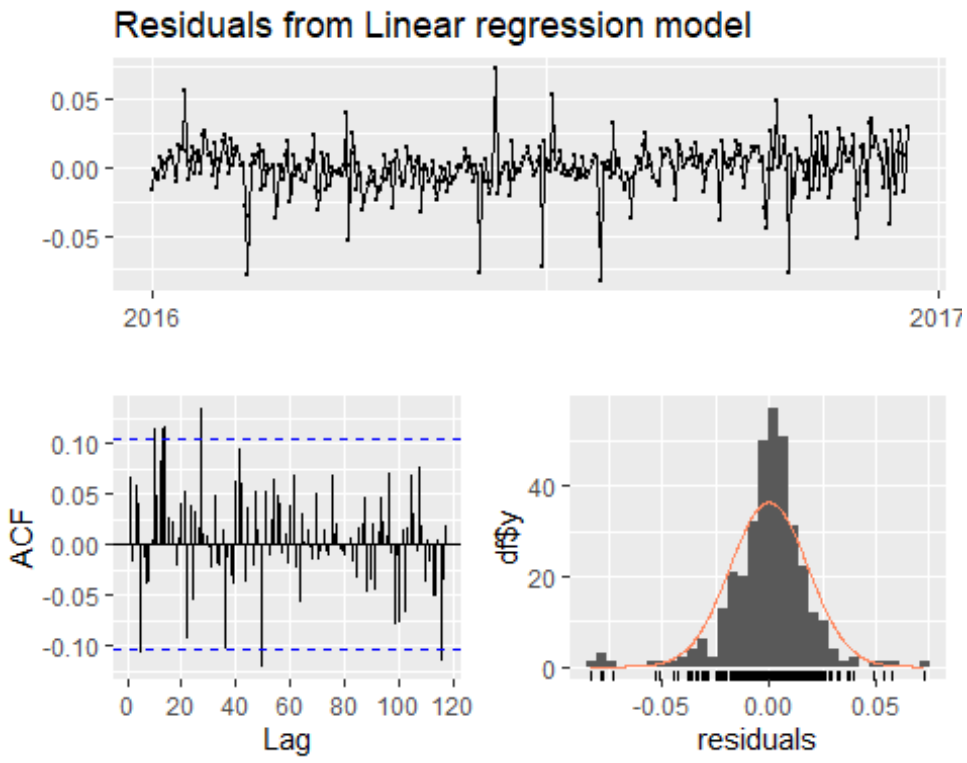
```
##
## Call:
## tslm(formula = logerror ~ propertylandusetypeid + bathroomcnt +
##     taxvaluedollarcnt, data = zillow.ts)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.083022 -0.007803  0.001703  0.009159  0.072464
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.415e+00  2.160e-01   6.551 2.06e-10 ***
## propertylandusetypeid -5.459e-03  8.143e-04  -6.705 8.17e-11 ***
## bathroomcnt            1.850e-02  5.053e-03   3.662 0.000289 ***
## taxvaluedollarcnt     -3.820e-08  1.134e-08  -3.368 0.000842 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01829 on 348 degrees of freedom
## Multiple R-squared:  0.1825, Adjusted R-squared:  0.1754
## F-statistic: 25.89 on 3 and 348 DF,  p-value: 3.848e-15
```

```
method[1] = 'Ordinary Least Squares'
accuracy[[1]] = data.frame(accuracy(lm.fit))
```

```
autoplot(logerror, series="Actual") +
    forecast::autolayer(fitted(lm.fit), series="Predicted")
```




```
checkresiduals(lm.fit)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 70
```

```
##
## data:  Residuals from Linear regression model
## LM test = 71.03, df = 70, p-value = 0.4432
```

Comment: The OLS model explained 19.27% of the variance in the data. The residual plot does not indicate obvious heteroscedasticity since they are approximately normally distributed with a slightly longer left tail. Simple Exponential Smoothing or ARIMA will likely outperform the OLS model.

Simple Exponential Smoothing

```
ses.fit <- ses(logerror)
summary(ses.fit)
```

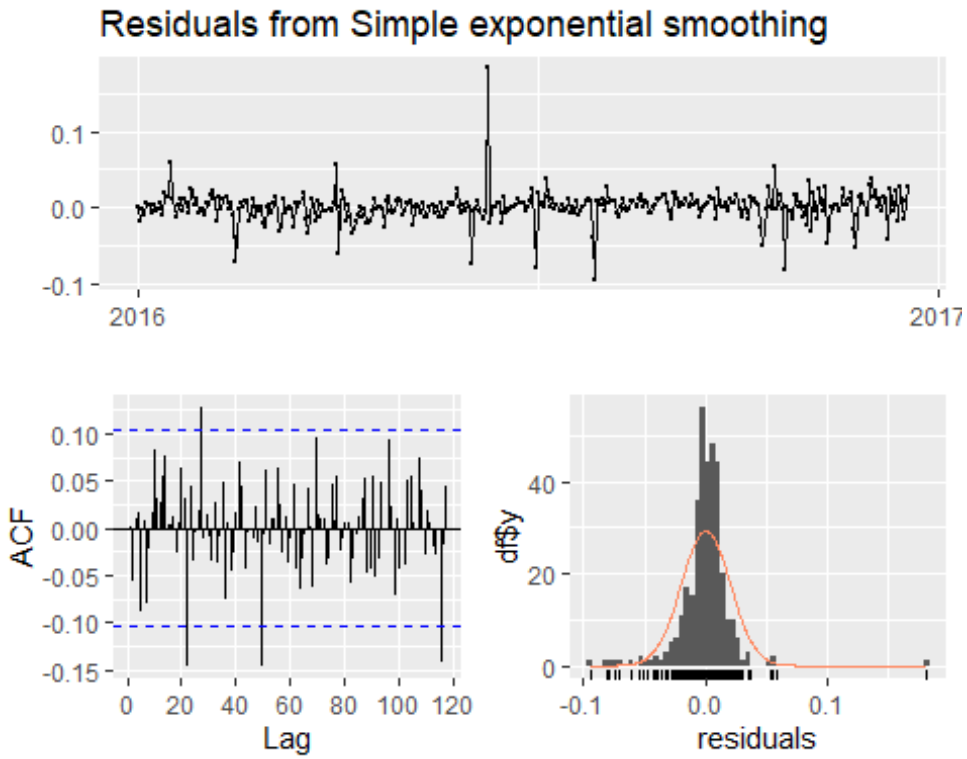
```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = logerror)
##
##   Smoothing parameters:
##     alpha = 0.0171
##
##   Initial states:
##     l = 0.0112
##
##   sigma:  0.0202
##
##       AIC      AICc       BIC
## -679.7478 -679.6789 -668.1570
##
## Error measures:
##                       ME       RMSE        MAE     MPE     MAPE MASE
## Training set 0.000435473 0.02012305 0.01211301 98.4266 216.6508  NaN
##                      ACF1
## Training set 0.002017516
##
## Forecasts:
##           Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 2016.9644     0.01384301 -0.01201930 0.03970531 -0.02570998 0.05339599
## 2016.9671     0.01384301 -0.01202306 0.03970908 -0.02571573 0.05340175
## 2016.9699     0.01384301 -0.01202683 0.03971284 -0.02572149 0.05340750
## 2016.9726     0.01384301 -0.01203059 0.03971660 -0.02572724 0.05341325
## 2016.9753     0.01384301 -0.01203435 0.03972036 -0.02573299 0.05341901
## 2016.9781     0.01384301 -0.01203811 0.03972412 -0.02573875 0.05342476
## 2016.9808     0.01384301 -0.01204187 0.03972788 -0.02574450 0.05343051
## 2016.9836     0.01384301 -0.01204563 0.03973164 -0.02575025 0.05343626
## 2016.9863     0.01384301 -0.01204939 0.03973540 -0.02575600 0.05344201
## 2016.9890     0.01384301 -0.01205315 0.03973916 -0.02576175 0.05344776
```

```
method[2] = 'Simple Exponential Smoothing'
accuracy[[2]] = data.frame(accuracy(ses.fit))
```

Comment:

The optimal simple exponential smoothing function for the zillow.ts data uses $\ell_t = .0171y_t + (1 - .0171)\ell_{t-1}$ and an initial ℓ value of .0112.

```
checkresiduals(ses.fit)
```



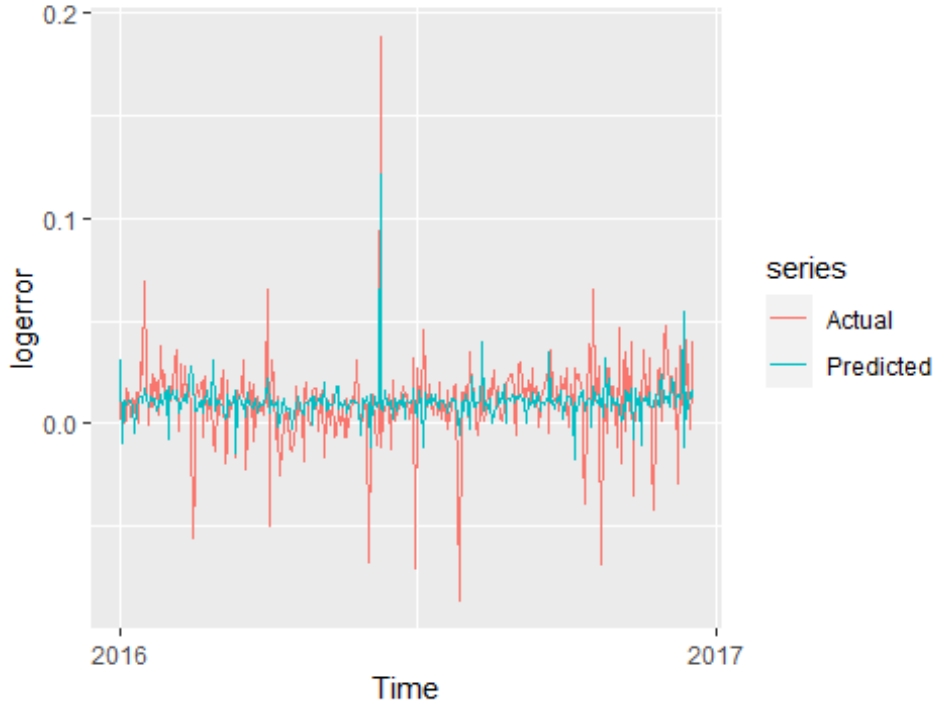

```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 64.334, df = 68, p-value = 0.6036
##
## Model df: 2.   Total lags used: 70
```

ARIMA

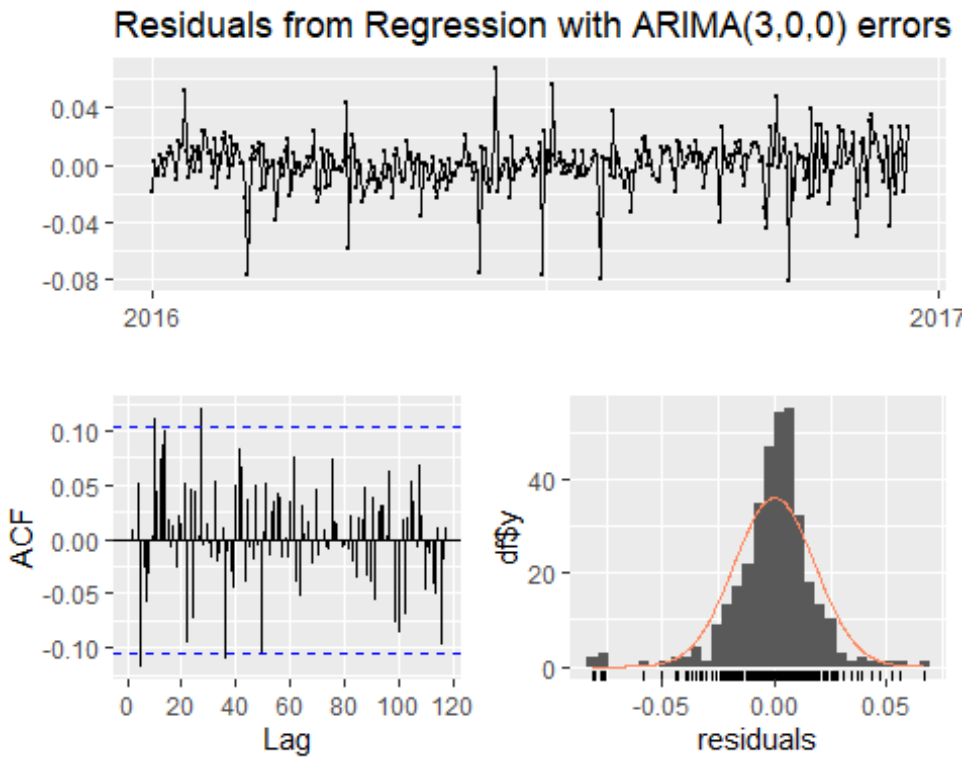
```
# Define xreg
xreg <- as.matrix(zillow.ts[,c('propertylandusetypeid','bathroomcnt','taxvaluedollarcnt')])

arima.fit <- auto.arima(logerror, xreg = xreg)

arima.fit <- Arima(logerror, xreg=xreg, order=c(3,0,0))
method[3] = 'ARIMA'
accuracy[[3]] = data.frame(accuracy(arima.fit))
```

Comment: The auto-arima indicated an ARIMA(3,0,0) model as the best fit.

```
autoplot(logerror, series="Actual") +
    forecast::autolayer(fitted(arima.fit), series="Predicted")
```




```
checkresiduals(arima.fit)
```




```
##
##  Ljung-Box test
```

```
##
## data:  Residuals from Regression with ARIMA(3,0,0) errors
## Q* = 66.356, df = 62, p-value = 0.3293
##
## Model df: 8.   Total lags used: 70
```

Summary Model Evaluation & Selection:

The fitted values of the ARIMA model follow a similar pattern to the linear model. Fitted values follow a similar, if more conservative, pattern as the observed values and, while there are some outlying residuals, the bulk of the values are clustered around zero.

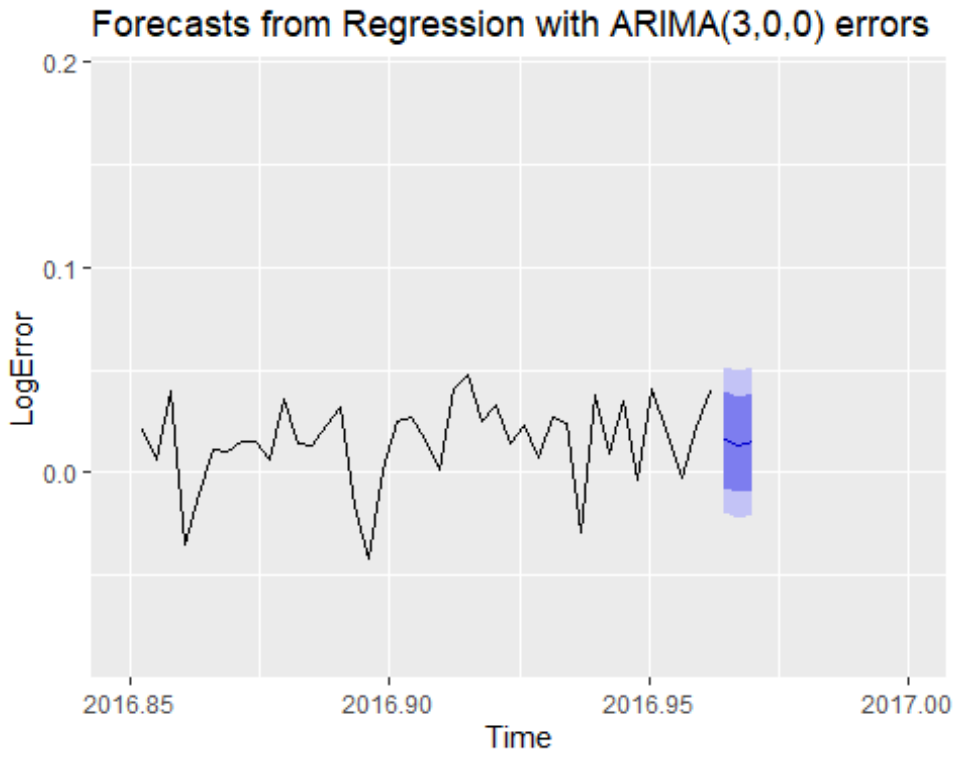
```
mod.health <- bind_rows(accuracy)%>%
  bind_cols(data.frame(method))%>%
  dplyr::select(ME, RMSE, MAE, method)

mod.health
```

```
##                              ME       RMSE        MAE
## Training set...1 -2.605300e-19 0.01819038 0.01243723
## Training set...2  4.354730e-04 0.02012305 0.01211301
## Training set...3  5.066436e-06 0.01802417 0.01231528
##                                  method
## Training set...1       Ordinary Least Squares
## Training set...2 Simple Exponential Smoothing
## Training set...3                       ARIMA
```

Comment: ARIMA outperformed OLS with an MAE of 0.0123 vers 0.0124. It should be noted due to technical issues, SES could not be fit with our housing features.

Forecasting October, November, December of 2017 using the ARIMA.Fit

```
test.zillow.ts <- ts(test.clean, frequency = 365, start=c(2017,1,1))
test.full <- ts(bind_rows(train.clean, test.clean)%>%
                  filter(transactiondate %in% c(as.Date('2016-10-01'),as.Date('2016-11-01'),as
.Date('2016-12-01'),
                                                as.Date('2017-10-01'),as.Date('2017-11-01'),as
.Date('2017-12-01'))),
                frequency = 365, start = c(2016,1,1))

forecast(arima.fit, xreg = test.full[,c('propertylandusetypeid','bathroomcnt','taxvaluedollarc
nt')])%>%
  autoplot()+
  labs(y = 'LogError')+
  xlim(2016.85,2017)

## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



Conclusion The ARIMA and multiple regression models appeared to perform best and th ARIMA model was used to make predictions about out of sample data points.