Kaggle - Zillow Home Prices EDA

Lilian Cheung

March 18, 2018

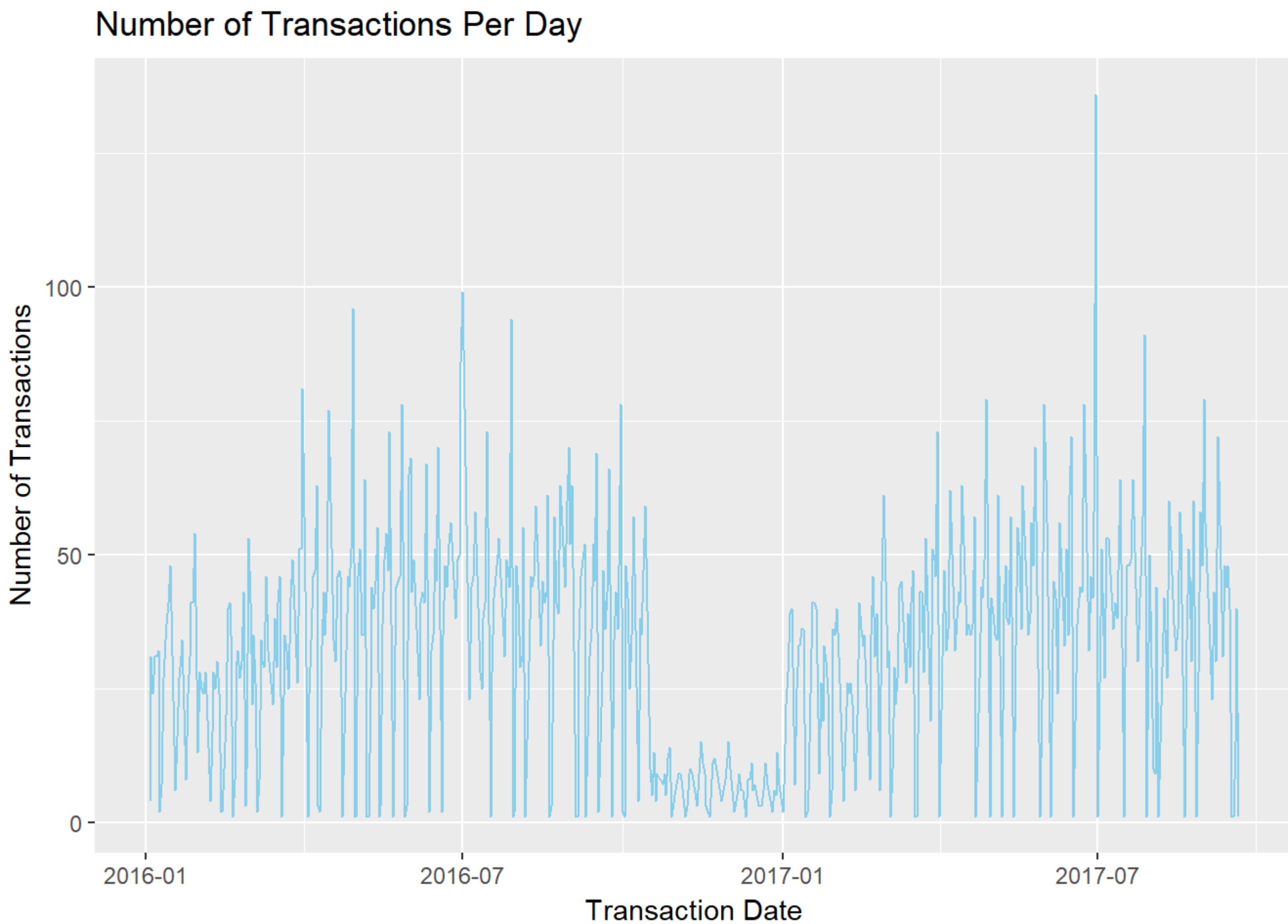
```r
runthis <- FALSE

if(runthis==TRUE) {
path_properties_2016 <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/properties_2016/propert
ies_2016.csv"

path_properties_2017 <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/properties_2017/propert
ies_2017.csv"

path_train_2016 <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/train_2016_v2/train_2016_v2.
csv"

path_train_2017 <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/train_2017/train_2017.csv"

## Training Data - 2016 (properties and logerrors of sales)
  properties_2016 <- fread( file=path_properties_2016, header=TRUE )
  train_2016 <- fread(file=path_train_2016, header=TRUE)

## Training Data - 2017 (properties and logerrors of sales)
  properties_2017 <- fread( file=path_properties_2017, header=TRUE )
  train_2017 <- fread(file=path_train_2017, header=TRUE)
}
```

```r
runthis <- FALSE

if (runthis==TRUE){
# Merge training data from 2016
  training_2016 <- merge( properties_2016, train_2016, by="parcelid" )

# Merge training data from 2017
  training_2017 <- merge( properties_2017, train_2017, by="parcelid")

# Combine 2016 and 2017 data into a dataframe
  training_all <- rbind(training_2016, training_2017)

# Save locally
  save_path <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/training_all.csv"
  write.csv( training_all, file=save_path, row.names=FALSE)
}
```

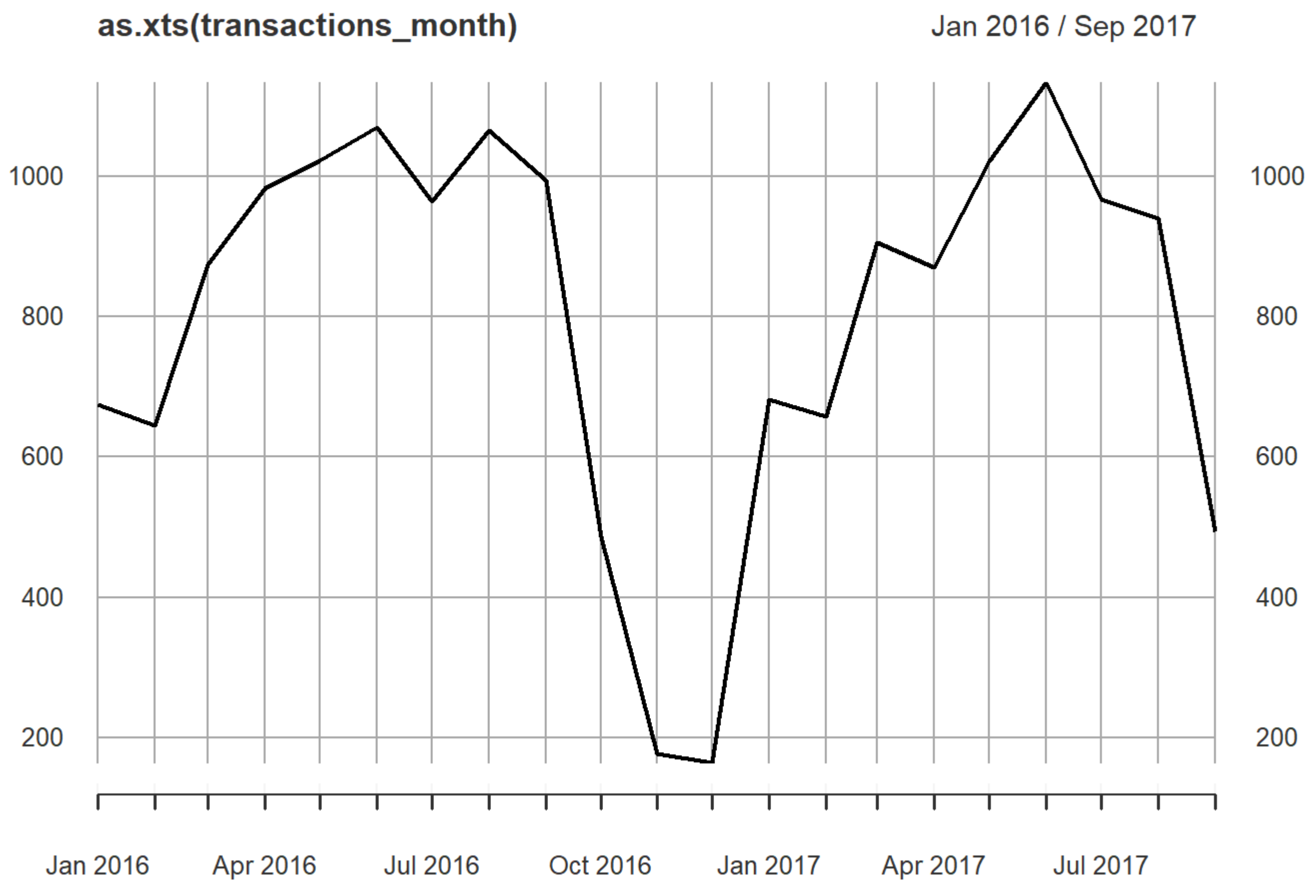
```r
# Read in dataset
  training_all_path <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/training_all.csv"
  training_all <- read.csv( file=training_all_path, header=TRUE)

## Sample some percent of dataset for EDA
  set.seed(123)
  percent <- 0.10
  training_sample <- sample_n( training_all, percent*nrow(training_all), replace=FALSE )

  ## remaining data
  # remaining_data <- dplyr::filter( training_all, !parcelid %in% training_sample[["parcelid"]] )

## Save locally
  save_path <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/training_sample.csv"
  write.csv( training_sample, file=save_path, row.names=FALSE)
```

```r
## Read in training sample
  read_path <- "C:/Lilian/Documents/Technical Training/R Projects/Kaggle/Zillow Home Prices/training_sample.csv"
  training_sample <- read.csv( file=read_path, header=TRUE)
  training_sample <- mutate(training_sample, transactiondate = as.Date(transactiondate))

## Reorder training sample by date
  training_sample <- arrange( training_sample, transactiondate )
```

```r
## Explore transaction counts in sample

# Number of transactions per day
  counts_transactions <- count( training_sample, transactiondate )

  n_transactions_plot <- ggplot( counts_transactions, aes(x = transactiondate, y = n) ) +
                         geom_line( color = "skyblue" ) +
                         xlab("Transaction Date") +
                         ylab("Number of Transactions") +
                         ggtitle("Number of Transactions Per Day")

  print( n_transactions_plot )
```



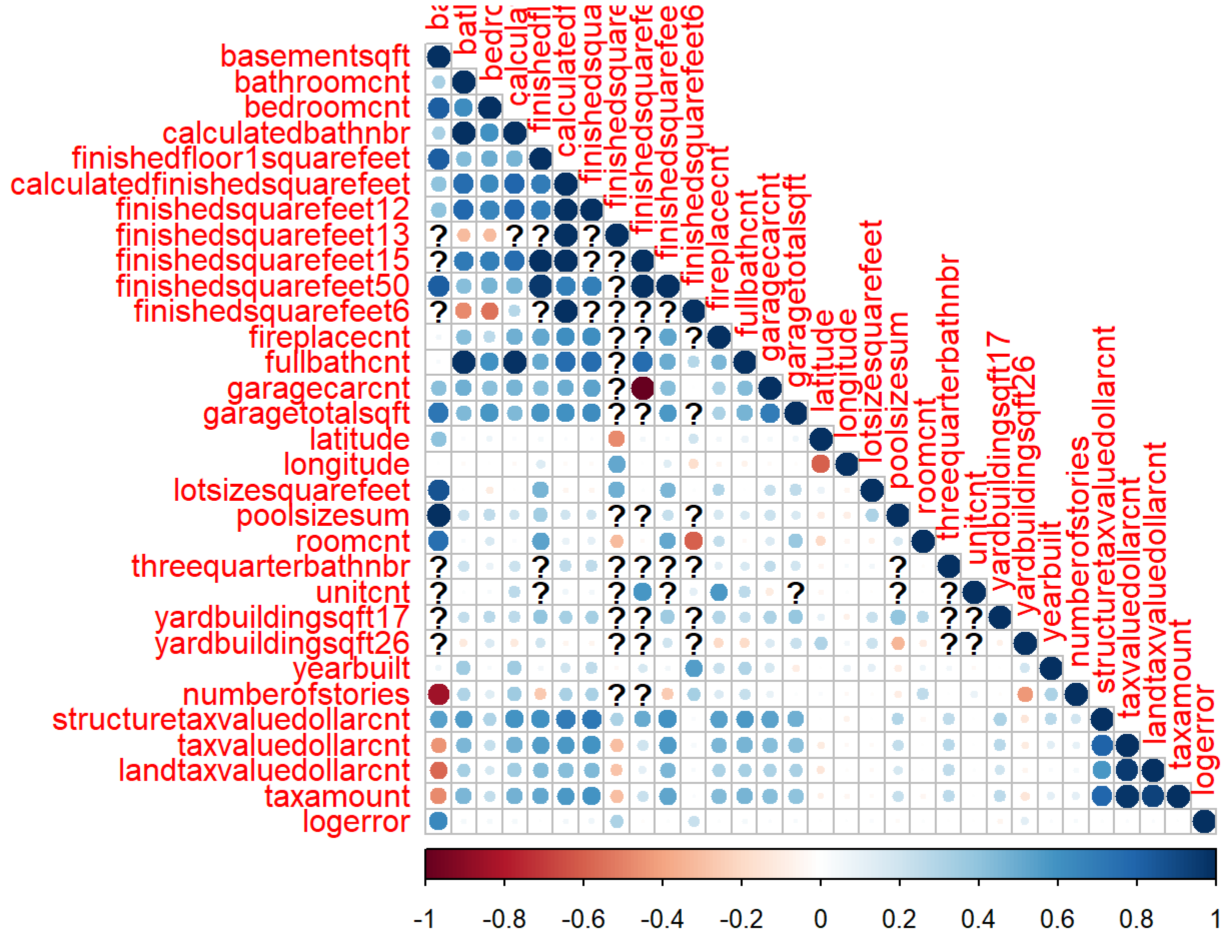

```
# Number of transactions per month in our dataset (note that October 2016 dips since part of the dataset is in Zillow's test
 dataset)
dt_training_sample <- data.table( training_sample )
counts_transactions_month <- dt_training_sample[ , .N, by = .(year(transactiondate), month(transactiondate)) ]

  # Graph (uses xts package)
  # Manipulate data into time series
  transactions_month <- counts_transactions_month[["N"]]
  transactions_month <- ts( counts_transactions_month[["N"]], frequency = 12, start = 2016)

  plot(as.xts(transactions_month), major.format = "%Y-%m", title="Number of Transactions per Month")
```


as.xts(transactions_month)
Jan 2016 / Sep 2017
1000
800
600
400
200
Jan 2016
Apr 2016
Jul 2016
Oct 2016
Jan 2017
Apr 2017
Jul 2017

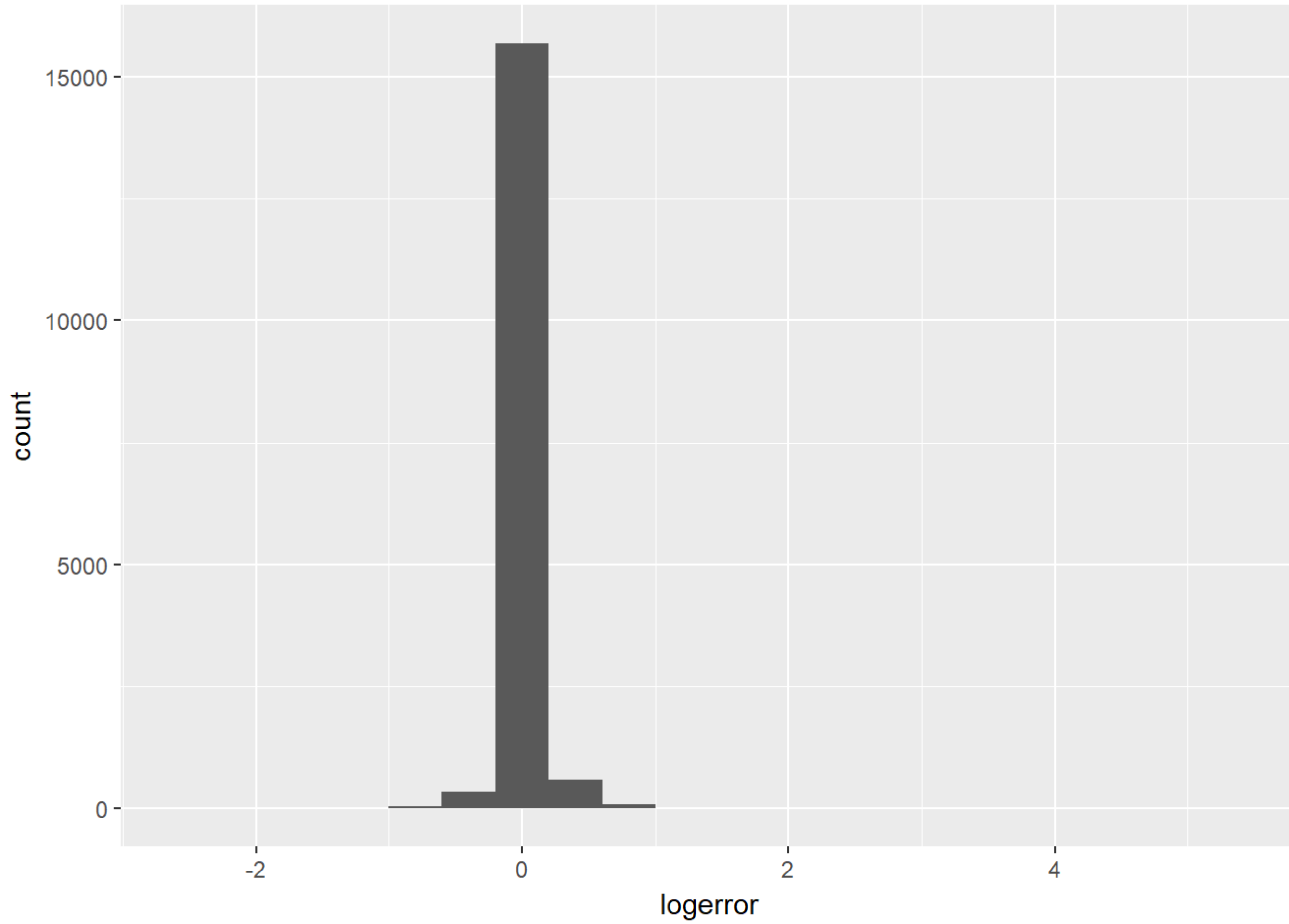
```
## Exploring the continuous predictors

## Convert variable types to factor as necessary
  to_factor <- c( "airconditioningtypeid", "architecturalstyletypeid",
        "buildingqualitytypeid", "buildingclasstypeid", "decktypeid",
        "fips", "heatingorsystemtypeid",
        "parcelid", "poolcnt", "pooltypeid10", "pooltypeid2", "pooltypeid7",
        "propertylandusetypeid", "rawcensustractandblock",
        "censustractandblock",
        "regionidcounty", "regionidcity", "regionidzip",
        "regionidneighborhood", "storytypeid", "typeconstructiontypeid",
        "assessmentyear", "taxdelinquencyyear")
  training_sample[ to_factor ] <- lapply( training_sample[to_factor], factor)

## Examine continuous variables
  num_columns <- sapply(training_sample, is.numeric)
  cts_training_sample <- training_sample[ num_columns ]

  # Correlogram
  cor_matrix <- cor( cts_training_sample, use='pairwise.complete.obs' )
```

```
## Warning in cor(cts_training_sample, use = "pairwise.complete.obs"): the
## standard deviation is zero
```

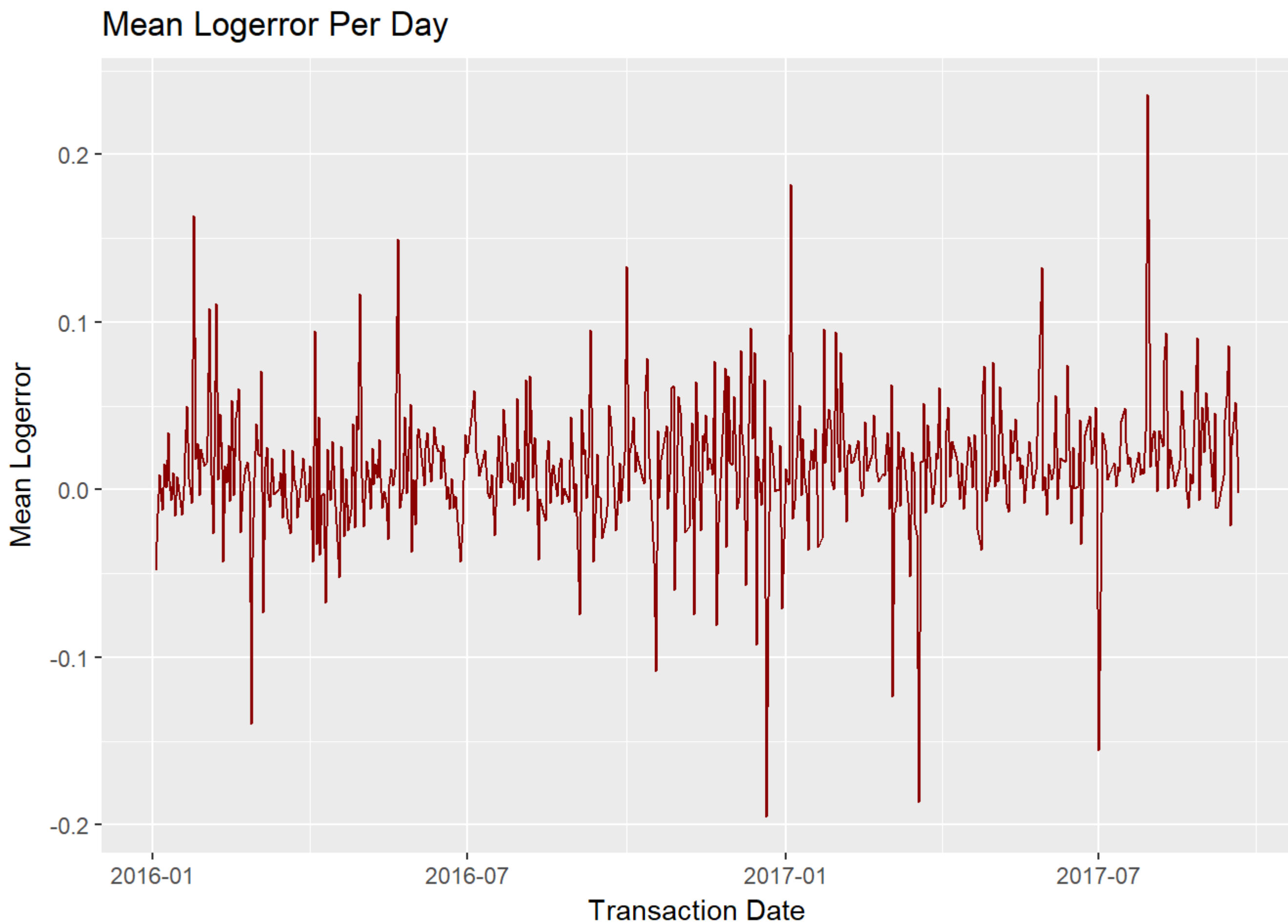
```
  corrplot(cor_matrix, type='lower')
```



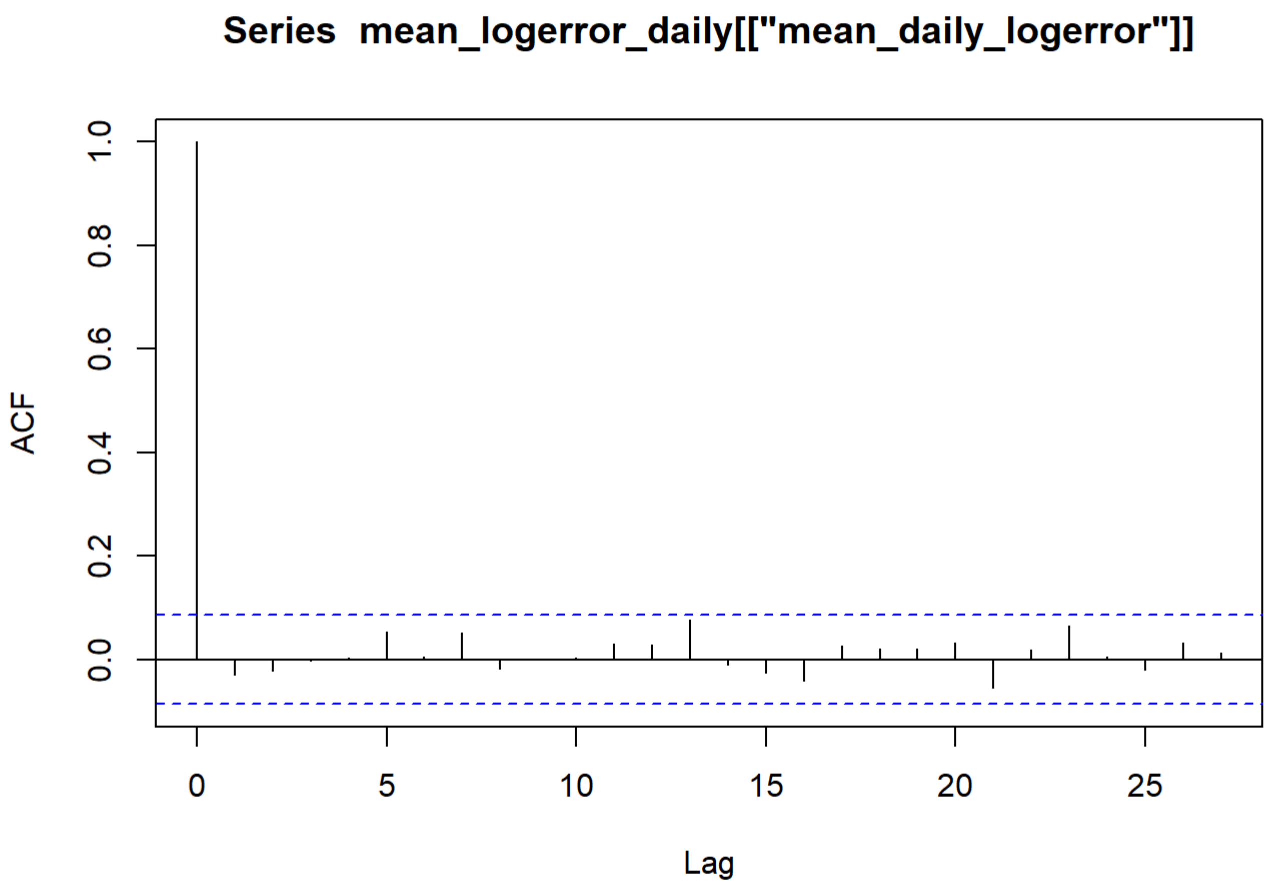

```
## Explore logerror, the response variable

  logerror_histogram <- ggplot( training_sample, aes(x = logerror) ) +
                        geom_histogram( bins = 20)
  logerror_histogram
```




```
## Plot of logerror per day
  mean_logerror_daily <- dt_training_sample[, mean(logerror), by = transactiondate]
    names(mean_logerror_daily) <- c("transactiondate", "mean_daily_logerror")

  daily_mean_logerror_plot <- ggplot( mean_logerror_daily, aes(x = transactiondate, y = mean_daily_logerror) ) +
                          geom_line( color = "darkred" ) +
                          xlab("Transaction Date") +
                          ylab("Mean Logerror") +
                          ggtitle("Mean Logerror Per Day")

  daily_mean_logerror_plot
```




```
## Does average logerror per day exhibit autocorrelation? No.
  acf( mean_logerror_daily[["mean_daily_logerror"]] )
```




```
## As expected, number of daily transactions does not have a high correlation with mean daily logerrors
  cor(x=mean_logerror_daily$mean_daily_logerror, y=counts_transactions$n)
```

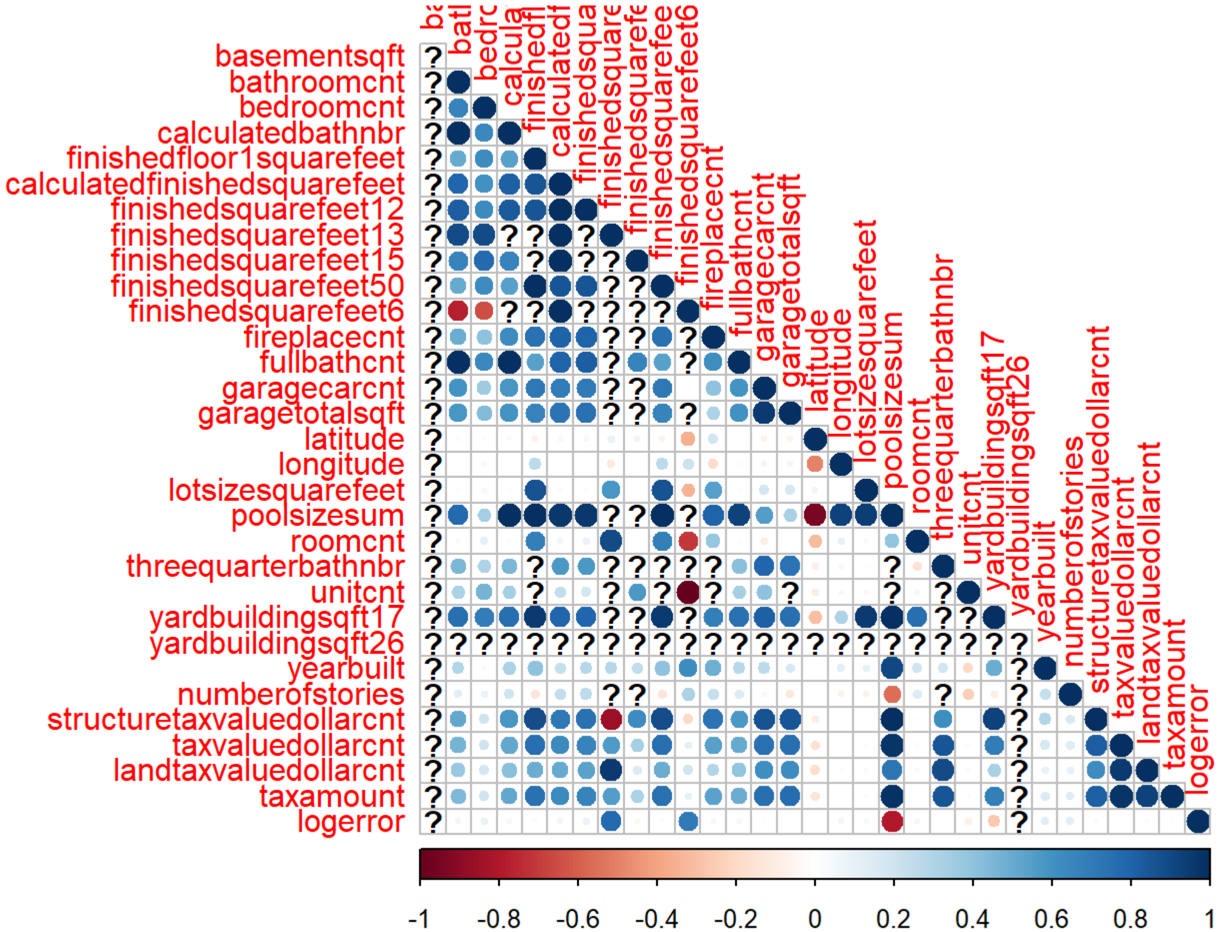
```
## [1] 0.05175558
```

```
  # 0.05
```

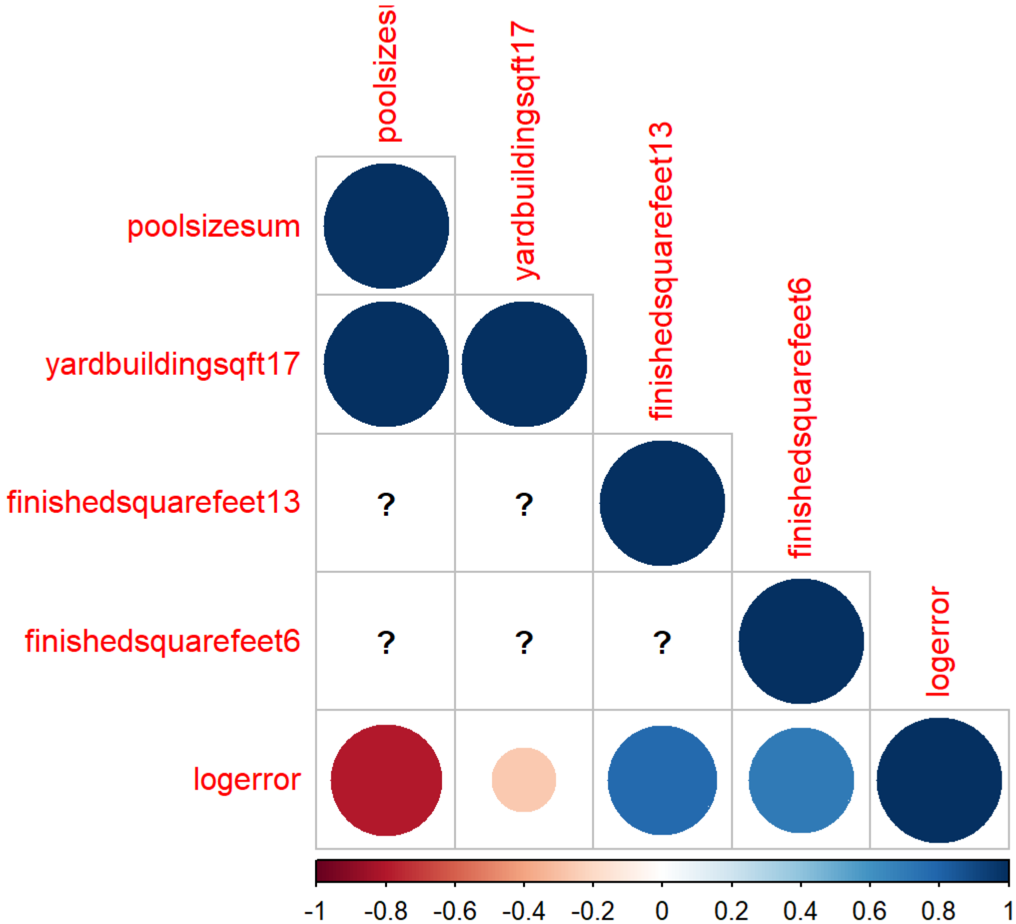
```
## Correlations between high logerrors and continuous features

## Arbitrarily define a high logerror as one that meets or exceeds 0.2
  high_logerror_sample <- dplyr::filter( training_sample, abs(logerror) >= 0.25 )

## Correlations between continuous variables in the high logerror sample
  num_columns <- sapply( high_logerror_sample, is.numeric)
  cts_high_logerror_sample <- high_logerror_sample[ num_columns ]

  # Correlogram
  cor_matrix <- cor( cts_high_logerror_sample, use='pairwise.complete.obs' )
```

```
## Warning in cor(cts_high_logerror_sample, use = "pairwise.complete.obs"):
## the standard deviation is zero
```

```
  corrplot(cor_matrix, type='lower')
```




```
## A closer look at columns of interest -- turns out most of these are missing, so this is not particularly helpful
  cols_of_interest <- c( "poolsizesum", "yardbuildingsqft17", "finishedsquarefeet13", "finishedsquarefeet6", "logerror")
  cor_matrix <- cor( cts_high_logerror_sample[, cols_of_interest], use='pairwise.complete.obs')
  corrplot(cor_matrix, type='lower')
```




```
# poolsizesum: total square footage of all pools on property
# yardbuildingsqft17: Patio in yard
# finishedsquarefeet13: Perimeter living area
# finishedsquarefeet6: Base unfinished and finished area

## These variables are mostly missing, so are not very helpful - e.g.
# table( is.na(cts_high_logerror_sample$poolsizesum) )
```

```
## ANOVA matrix?
```