INFX 502 - Semester Project

Name: Pinky Sitikhu

ULID: C00477712

Department: Informatics (Masters')

Loading packages

```
# install.packages("moments")
library(moments)
library(ggcorrplot)
```

```
## Loading required package: ggplot2
```

```
library(readr)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v purrr   0.3.4      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Loading Dataset Since the dataset is in csv format, "read_csv" function is used to read the dataset. Further, to briefly observe the dataset, "head" function is used to preview few rows of our dataset.

```
data <- read_csv("realtor-data.csv", show_col_types = FALSE)
head(data)
```

```
## # A tibble: 6 x 12
##   status    price   bed  bath acre_~1 full_~2 street city  state zip_c~3 house~4
##   <chr>     <dbl> <dbl> <dbl>   <dbl> <chr>   <chr>  <chr> <chr>   <dbl>   <dbl>
## 1 for_sale 105000     3     2    0.12 Sector~ Secto~ Adju~ Puer~     601     920
## 2 for_sale  80000     4     2    0.08 Km 78 ~ Km 78~ Adju~ Puer~     601    1527
## 3 for_sale  67000     2     1    0.15 556G 5~ 556G ~ Juan~ Puer~     795     748
## 4 for_sale 145000     4     2    0.1  R5 Com~ R5 Co~ Ponce Puer~     731    1800
```

```
## 5 for_sale  65000     6     2    0.05 14 Nav~ 14 Na~ Maya~ Puer~     680      NA
## 6 for_sale 179000     4     3    0.46 Bo Cal~ Bo Ca~ San ~ Puer~     612    2520
## # ... with 1 more variable: sold_date <date>, and abbreviated variable names
## #   1: acre_lot, 2: full_address, 3: zip_code, 4: house_size
```

We can see the structure of the dataset using "str" function.

```
str(data)
```

```
## spec_tbl_df [923,159 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ status      : chr [1:923159] "for_sale" "for_sale" "for_sale" "for_sale" ...
##  $ price       : num [1:923159] 105000 80000 67000 145000 65000 179000 50000 71600 100000 300000 ...
##  $ bed         : num [1:923159] 3 4 2 4 6 4 3 3 2 5 ...
##  $ bath        : num [1:923159] 2 2 1 2 2 3 1 2 1 3 ...
##  $ acre_lot    : num [1:923159] 0.12 0.08 0.15 0.1 0.05 0.46 0.2 0.08 0.09 7.46 ...
##  $ full_address: chr [1:923159] "Sector Yahuecas Titulo # V84, Adjuntas, PR, 00601" "Km 78 9 Carr # 
##  $ street      : chr [1:923159] "Sector Yahuecas Titulo # V84" "Km 78 9 Carr # 135" "556G 556-G 16 St
##  $ city        : chr [1:923159] "Adjuntas" "Adjuntas" "Juana Diaz" "Ponce" ...
##  $ state       : chr [1:923159] "Puerto Rico" "Puerto Rico" "Puerto Rico" "Puerto Rico" ...
##  $ zip_code    : num [1:923159] 601 601 795 731 680 612 639 731 730 670 ...
##  $ house_size  : num [1:923159] 920 1527 748 1800 NA ...
##  $ sold_date   : Date[1:923159], format: NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   status = col_character(),
##   ..   price = col_double(),
##   ..   bed = col_double(),
##   ..   bath = col_double(),
##   ..   acre_lot = col_double(),
##   ..   full_address = col_character(),
##   ..   street = col_character(),
##   ..   city = col_character(),
##   ..   state = col_character(),
##   ..   zip_code = col_double(),
##   ..   house_size = col_double(),
##   ..   sold_date = col_date(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
```

In this dataset, status, street, city, and state variables are of character type. Similarly, zip_code is of numeric type. All these variables need to be converted into factor type.

```
data$status <- as.factor(data$status)
data$street <- as.factor(data$street)
data$city <- as.factor(data$city)
data$state <- as.factor(data$state)
data$zip_code <- as.factor(data$zip_code)
```

Now, checking whether they are changed or not:

```
str(data)
```

```
## spec_tbl_df [923,159 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ status      : Factor w/ 2 levels "for_sale","ready_to_build": 1 1 1 1 1 1 1 1 1 1 ...
##  $ price       : num [1:923159] 105000 80000 67000 145000 65000 179000 50000 71600 100000 300000 ...
##  $ bed         : num [1:923159] 3 4 2 4 6 4 3 3 2 5 ...
##  $ bath        : num [1:923159] 2 2 1 2 2 3 1 2 1 3 ...
##  $ acre_lot    : num [1:923159] 0.12 0.08 0.15 0.1 0.05 0.46 0.2 0.08 0.09 7.46 ...
##  $ full_address: chr [1:923159] "Sector Yahuecas Titulo # V84, Adjuntas, PR, 00601" "Km 78 9 Carr # 
##  $ street      : Factor w/ 110324 levels "0 0 Bay St /Judson St /Church St",..: 109072 106514 80404 
##  $ city        : Factor w/ 2542 levels "Abbot","Aberdeen",..: 15 15 1078 1765 1318 1947 415 1765 1765
##  $ state       : Factor w/ 18 levels "Connecticut",..: 10 10 10 10 10 10 10 10 10 10 ...
##  $ zip_code    : Factor w/ 3191 levels "601","602","603",..: 1 1 95 67 39 8 19 67 66 34 ...
##  $ house_size  : num [1:923159] 920 1527 748 1800 NA ...
##  $ sold_date   : Date[1:923159], format: NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   status = col_character(),
##   ..   price = col_double(),
##   ..   bed = col_double(),
##   ..   bath = col_double(),
##   ..   acre_lot = col_double(),
##   ..   full_address = col_character(),
##   ..   street = col_character(),
##   ..   city = col_character(),
##   ..   state = col_character(),
##   ..   zip_code = col_double(),
##   ..   house_size = col_double(),
##   ..   sold_date = col_date(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
```

check number of rows and columns in dataset

```
nrow(data)
```

```
## [1] 923159
```

```
ncol(data)
```

```
## [1] 12
```

There are more than 900k rows and 12 columns in the dataset.

From the given structure of the dataset, we can see that the variables are either character or numeric type. But, we need to check whether there are any missing values in the dataset.

Checking missing values

```
any(is.na(data))
```

```
## [1] TRUE
```

This indicates that there are some missing values in the dataset. So, we further check to see which of the columns has missing values. # find the columns containing missing values

```r
apply(is.na(data), 2, any)
```

```
##       status        price          bed         bath     acre_lot full_address 
##        FALSE         TRUE         TRUE         TRUE         TRUE        FALSE 
##       street         city        state     zip_code   house_size    sold_date 
##         TRUE         TRUE        FALSE         TRUE         TRUE         TRUE
```

We can see that most of the columns has missing values. But, let's check which column has most number of missing values

```r
colSums(is.na(data))
```

```
##       status        price          bed         bath     acre_lot full_address 
##            0           71       131703       115192       273623            0 
##       street         city        state     zip_code   house_size    sold_date 
##         2138           74            0          205       297843       466763
```

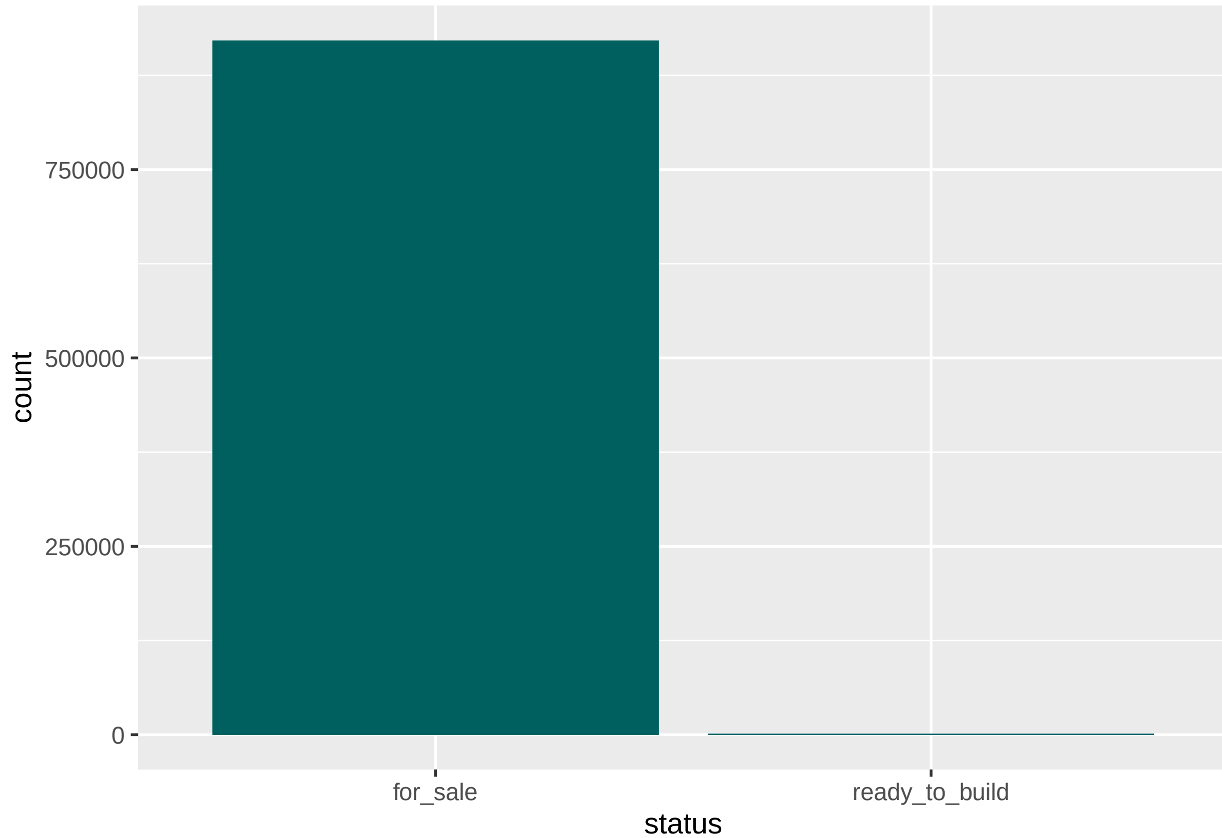
This shows that more than 400k observations do not have sold_date. There are certain columns which might not have significant influence/impact in the house prices. The columns like "status", "street", "full_address", "sold_date" can be removed. Before removing "status" column, let's observe the unique values and plot their distribution.

```r
unique(data$status)
```

```
## [1] for_sale       ready_to_build
## Levels: for_sale ready_to_build
```

Now, let's check the distribution of these two unique status.

```r
ggplot(data, aes(x=reorder(status, status, function(x)-length(x)))) +
geom_bar(fill='#006060') +  labs(x='status')
```



From the plot above, we can clearly see that there are huge number of observation that are "for_sale" as compared to "ready_to_build". Since there is great data disparity, we exclude this column from our further exploration.

So, removing the columns that has less or no influence in house prices.

```
new <- c("price", "bed", "bath", "acre_lot", "city", "state", "zip_code", "house_size")
df <- data[new]
head(df)
```

```
## # A tibble: 6 x 8
##    price   bed  bath acre_lot city          state       zip_code house_size
##    <dbl> <dbl> <dbl>    <dbl> <fct>         <fct>       <fct>         <dbl>
## 1 105000     3     2     0.12 Adjuntas      Puerto Rico 601             920
## 2  80000     4     2     0.08 Adjuntas      Puerto Rico 601            1527
## 3  67000     2     1     0.15 Juana Diaz    Puerto Rico 795             748
## 4 145000     4     2     0.1  Ponce         Puerto Rico 731            1800
## 5  65000     6     2     0.05 Mayaguez      Puerto Rico 680              NA
## 6 179000     4     3     0.46 San Sebastian Puerto Rico 612            2520
```

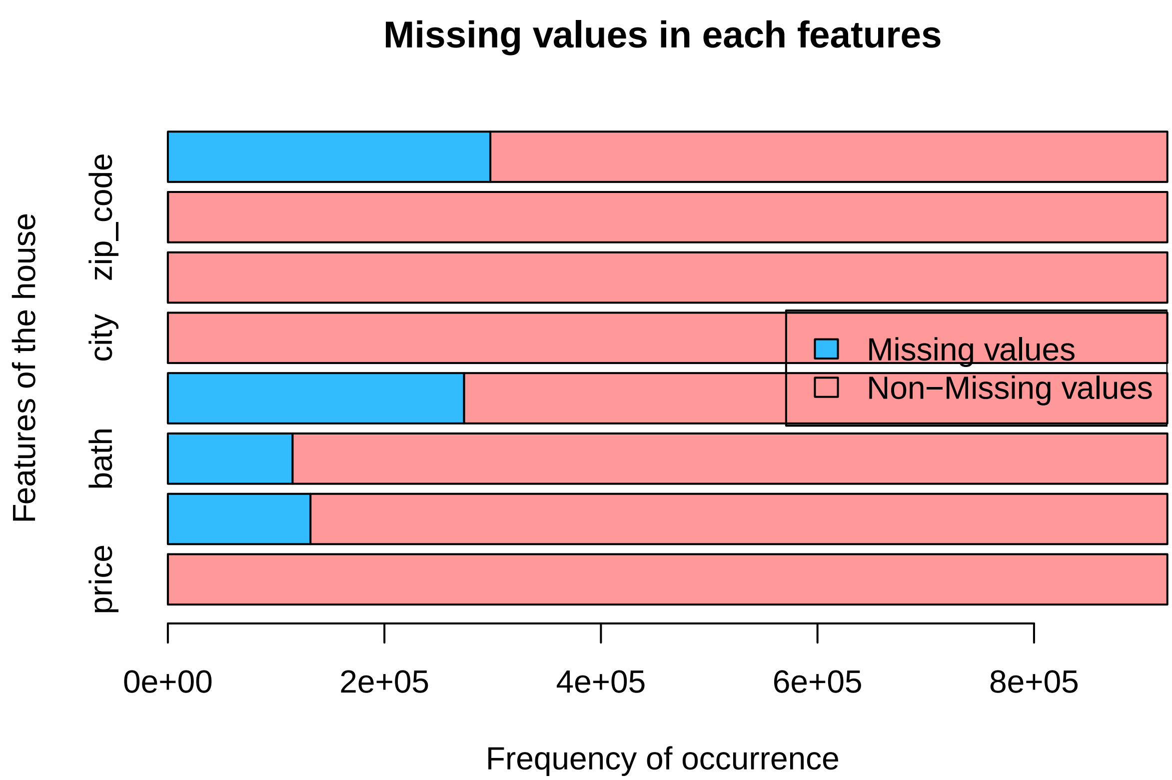
Visualize missing value count in the remaining columns

```
missing_count_func <- function(df){
  m<-c()
  for (i in colnames(df)){
    x<-sum(is.na(df[,i]))
```

```
    # count missing value
    m<-append(m,x)
    # count non-missing value
    m<-append(m,nrow(df)-x)
  }

  a<-matrix(m, nrow = 2)
  rownames(a)<-c("TRUE", "FALSE")
  colnames(a)<-colnames(df)
  return(a)
}

f=missing_count_func(df)
f
```

```
##        price    bed   bath acre_lot   city  state zip_code house_size
## TRUE      71 131703 115192   273623     74      0      205     297843
## FALSE 923088 791456 807967   649536 923085 923159   922954     625316
```

```
barplot(f, main = "Missing values in each features", xlab="Frequency of occurrence", ylab="Features of t
legend("right",c("Missing values","Non-Missing values"),
fill = c("#33bbff","#ff9999"))
```



There are various methods to handle NA or missing values, but in my use case, removing NA values and less significant columns sounds promising as there are more than 900k observations in the dataset. Also, using mean or median values or applying regression approach to fill missing values will not provide the accurate result for predicting prices.

```
nrow(df)
```

```
## [1] 923159
```

```
df <- na.omit(df)
```

Rechecking if any missing values remaining or not

```
any(is.na(df))
```

```
## [1] FALSE
```

Check number of rows remaining

```
nrow(df)
```

```
## [1] 421227
```

After removing all missing values from different columns, we have more than 400k observations left. Though we removed a lot of observations based on missing values, we proceed forward with 400k observations for further analysis (in this case).

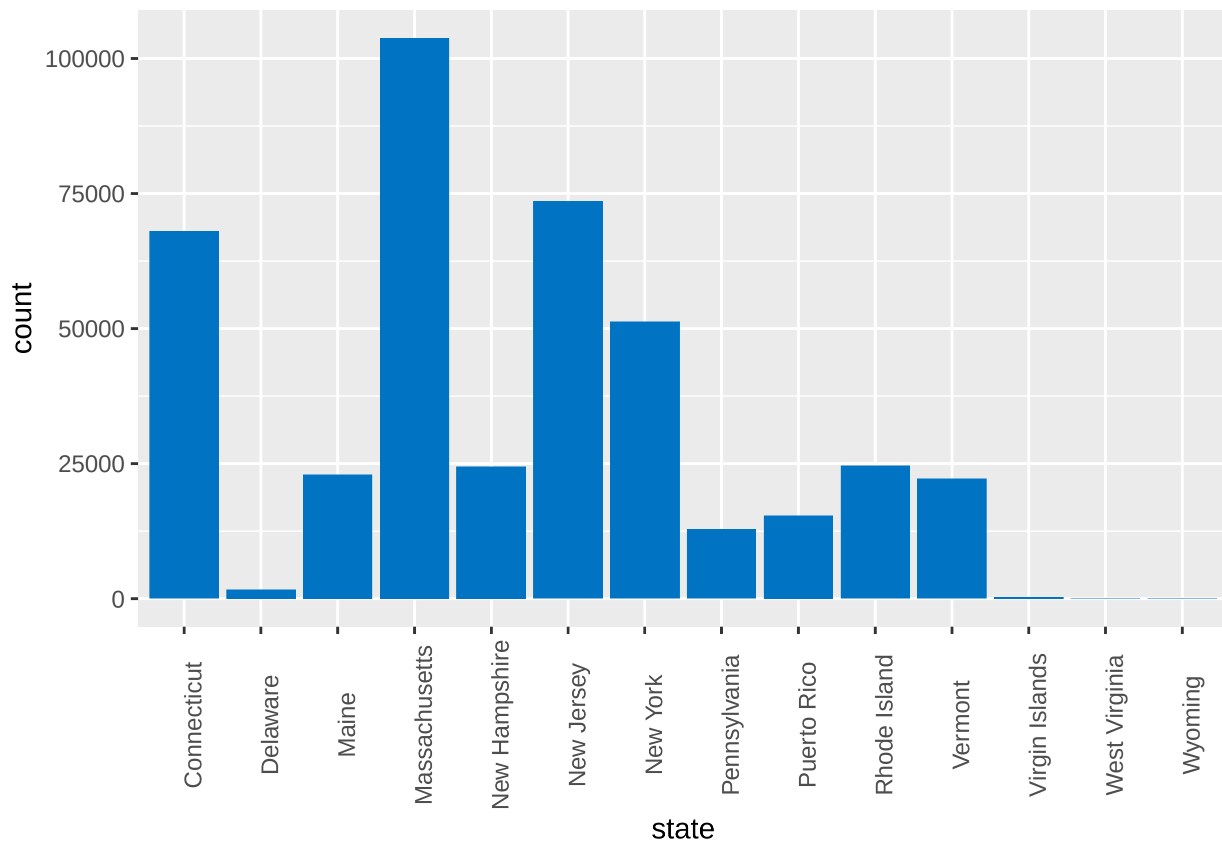
Analysis

Univariate Analysis

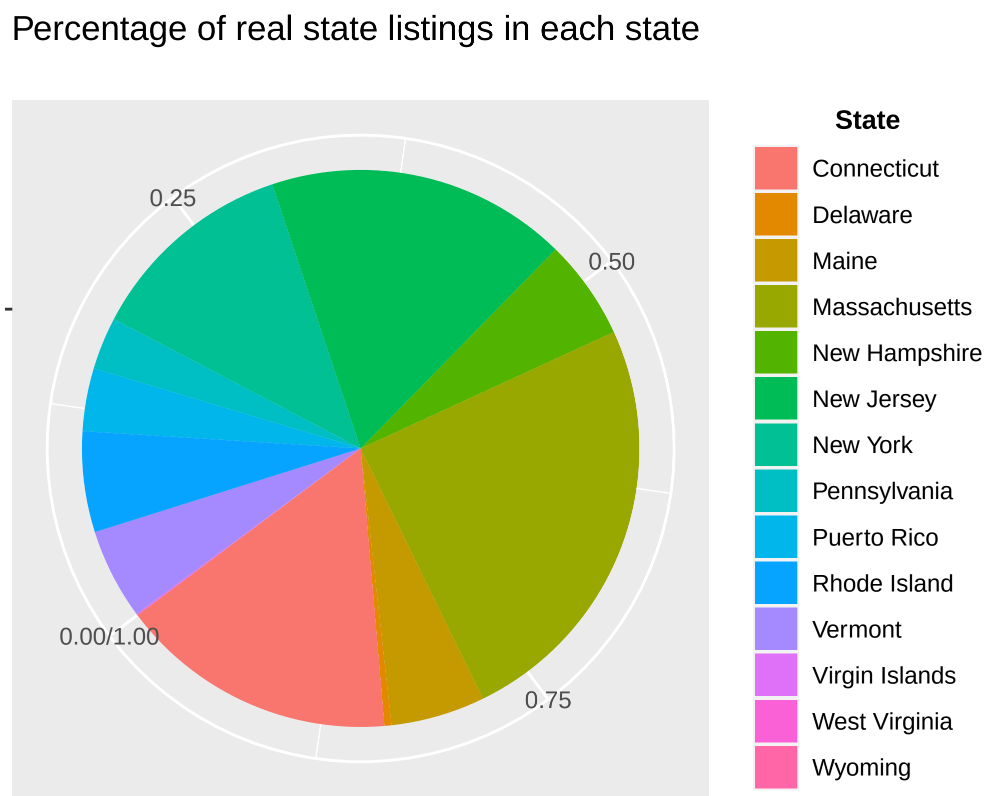
1. Exploration of categorical variables In the above variables, city, state, zip_code are categorical variable. First, I evaluate the distribution of the observations in terms of these variables. It would

```
ggplot(data=df, mapping=aes(x=state)) + geom_histogram(stat="count")+geom_bar(fill="#0073C2FF")+theme(a
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```r
df_state <- df %>%
  group_by(state) %>%
  count() %>%
  ungroup() %>%
  mutate(perc = `n` / sum(`n`)) %>%
  arrange(perc) %>%
  mutate(labels = scales::percent(perc))
ggplot(df_state, aes(x = "", y = perc, fill = state)) +
     geom_bar(width = 1, stat = "identity") +
     coord_polar(theta = "y", start = 180) +
     labs(x = "", y = "", title = "Percentage of real state listings in each state \n",
          fill = "State") +
     theme(plot.title = element_text(hjust = 0.5),
           legend.title = element_text(hjust = 0.5, face="bold", size = 10))
```




```
#+
#   geom_text(aes(label = paste(labels)),
#             position = position_stack(vjust = 0.5), size=2)
```

Interpretation

From the above chart, we can see that Massachusetts has a lot of houses listed (about 25%) which means a lot of properties from this state are for sale. It would be interesting to see in which cities and zip code there are more number of houses available for sale. Similarly, for states like Delaware, Virgin Islands, West Virginia and Wyoming, the number of houses on listing is very low, which looks negligible in above figure.

So, we can divide our analysis in two parts: price difference in the region that has a lot of listing vs in the region with less listing.

```
# mass_state <- df%>% filter(state == 'Massachusetts')
# mass_state_df <- mass_state %>%
#   group_by(city) %>%
#   count()
# mass_state_df<-mass_state_df[order(mass_state_df$n, decreasing=TRUE),]
# mass_state_df$n <- lapply(mass_state_df$n,as.numeric)
# head(mass_state_df)
```

2. Check how numerical variable is affecting the price

First, I observe the overall summary of the dataset for general overview.

```
summary(df)
```

```
##      price                bed              bath            acre_lot       
##  Min.   :      500   Min.   : 1.000   Min.   :  1.000   Min.   :0.00e+00  
##  1st Qu.:   285000   1st Qu.: 3.000   1st Qu.:  2.000   1st Qu.:1.10e-01  
##  Median :   459900   Median : 3.000   Median :  2.000   Median :2.60e-01  
##  Mean   :   784424   Mean   : 3.814   Mean   :  2.704   Mean   :9.44e+00  
##  3rd Qu.:   789000   3rd Qu.: 4.000   3rd Qu.:  3.000   3rd Qu.:8.90e-01  
##  Max.   :169000000   Max.   :99.000   Max.   :198.000   Max.   :1.00e+05  
##                                                                           
##             city                      state           zip_code     
##  Boston       : 13414   Massachusetts:103770   6010   :  1896  
##  New York City:  8136   New Jersey   : 73571   2895   :  1766  
##  Philadelphia :  7663   Connecticut  : 68033   1201   :  1731  
##  Staten Island:  7064   New York     : 51270   6790   :  1421  
##  Brooklyn     :  6878   Rhode Island : 24620   2127   :  1400  
##  Bronx        :  4394   New Hampshire: 24454   6082   :  1399  
##  (Other)      :373678   (Other)      : 75509   (Other):411614  
##    house_size     
##  Min.   :    122  
##  1st Qu.:   1333  
##  Median :   1894  
##  Mean   :   2412  
##  3rd Qu.:   2784  
##  Max.   :1450112  
## 
```

Exploring univariate numerical features

```
skewness(x=df$bed)
```

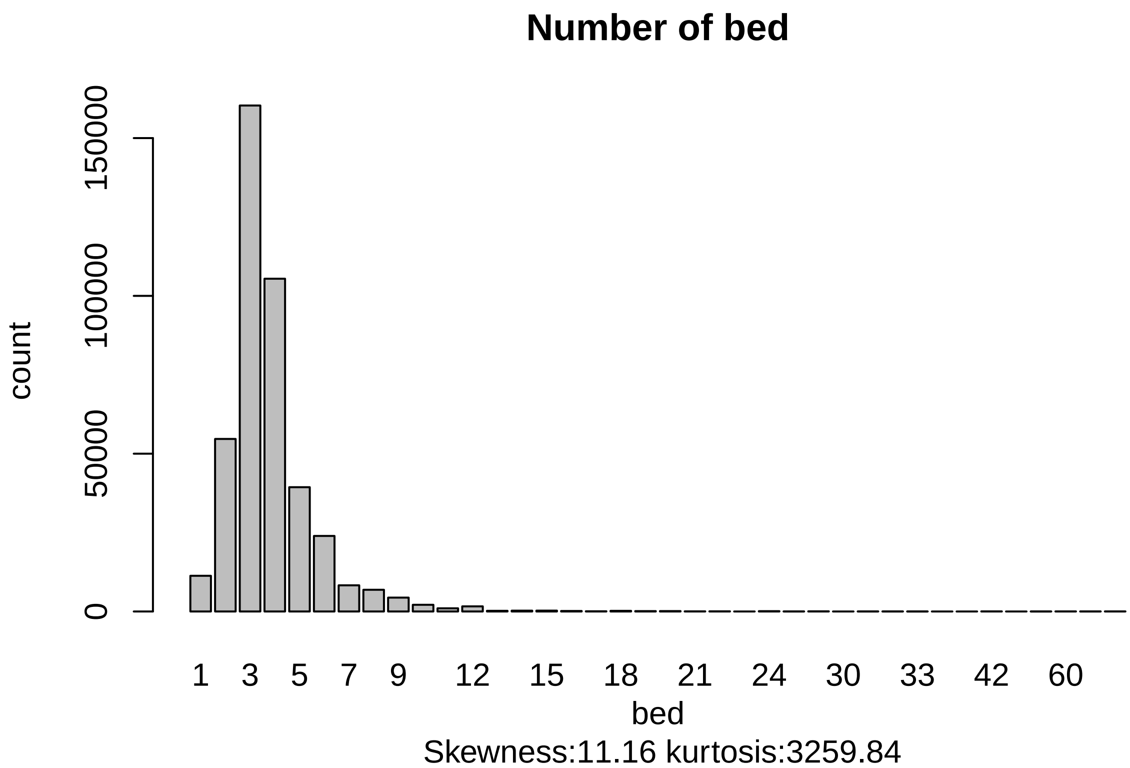
```
## [1] 11.15671
```

```
kurtosis(df$bed)
```

```
## [1] 359.5273
```

```
barplot(table(df$bed), main="Number of bed", xlab=paste0("bed", '\n', 'Skewness:', round(skewness(x=df$b
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): font
## width unknown for character 0x9
```




```
skewness(x=df$bath)
```

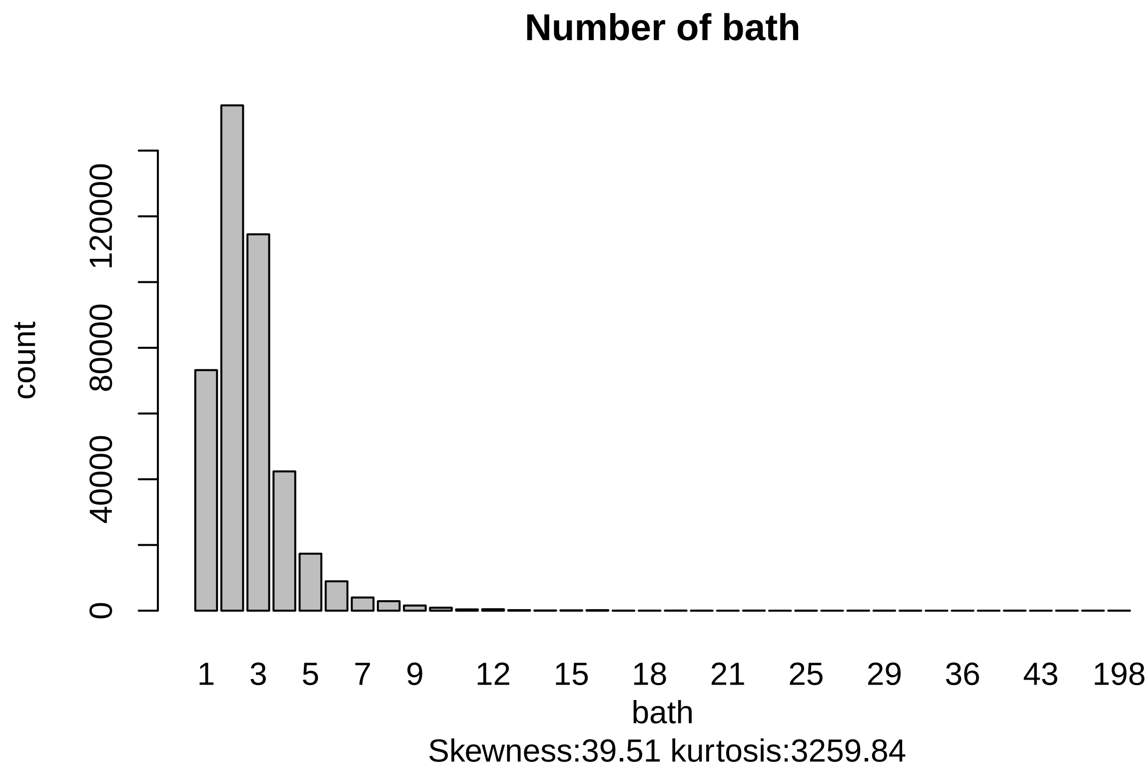
```
## [1] 39.51487
```

```
kurtosis(df$bath)
```

```
## [1] 3259.844
```

```
barplot(table(df$bath), main="Number of bath", xlab=paste0("bath", '\n', 'Skewness:', round(skewness(x=
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...): font
## width unknown for character 0x9
```



Skewness measures the asymmetry of a distribution around its mean and kurtosis measures how heavy the tails of a distribution is around its mean. From the bar plot, skewness and kurtosis values, it is clear that the bed and bath variables are right skewed distributions. Their median values are less than their means. The positive kurtosis value indicates that the tail is heavier than the normal distribution which means this data has more outliers than a normal distribution. This can be true, the maximum number of bed and bath are 99 and 198 respectively, which means the price, acre_lot and house_size need to be maximum in order to validate these numbers.

Outlier detection in bed variable

So, I extracted the potential outliers based on IQR critierion using the following function.

```
lowerOutlierLimit <- quantile(df$bed, probs=0.25, names=FALSE)-1.5*IQR(df$bed)
upperOutlierLimit <- quantile(df$bed, probs=0.75, names=FALSE)+1.5*IQR(df$bed)
bed_outliers<-df$bed[df$bed<lowerOutlierLimit | df$bed>upperOutlierLimit]
length(bed_outliers)
```

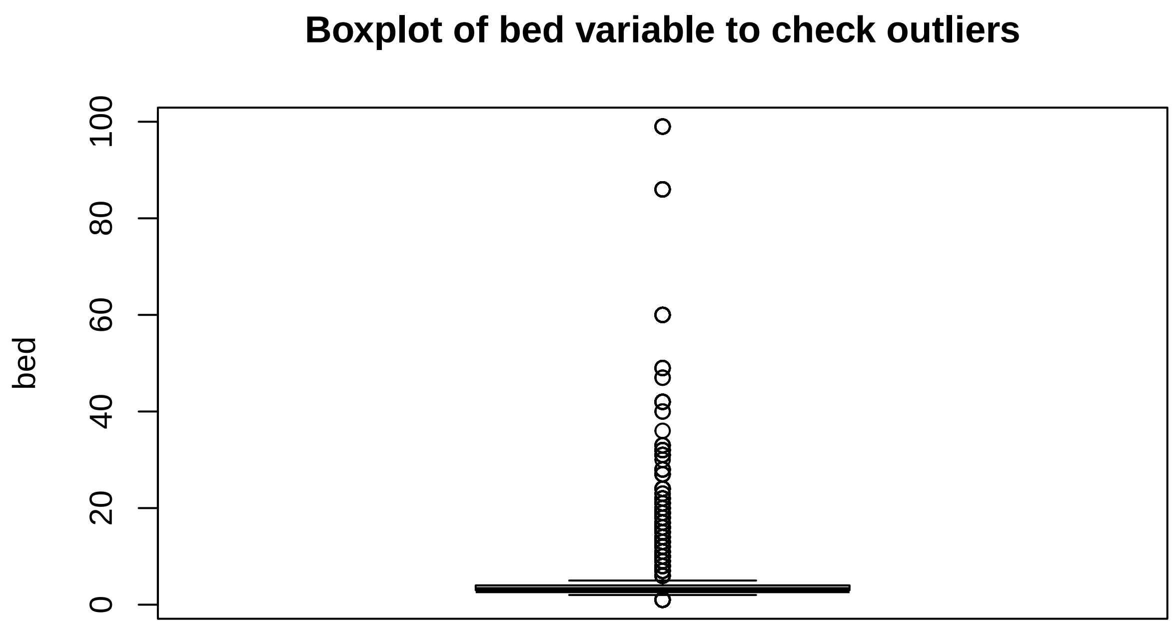
```
## [1] 61421
```

```
unique(bed_outliers)
```

```
##  [1]  6  1  9  7  8 12 13 10 11 33 24 28 14 18 20 16 15 19 17 40 21 86 31 27 42
## [26] 60 22 32 99 49 30 23 47 36
```

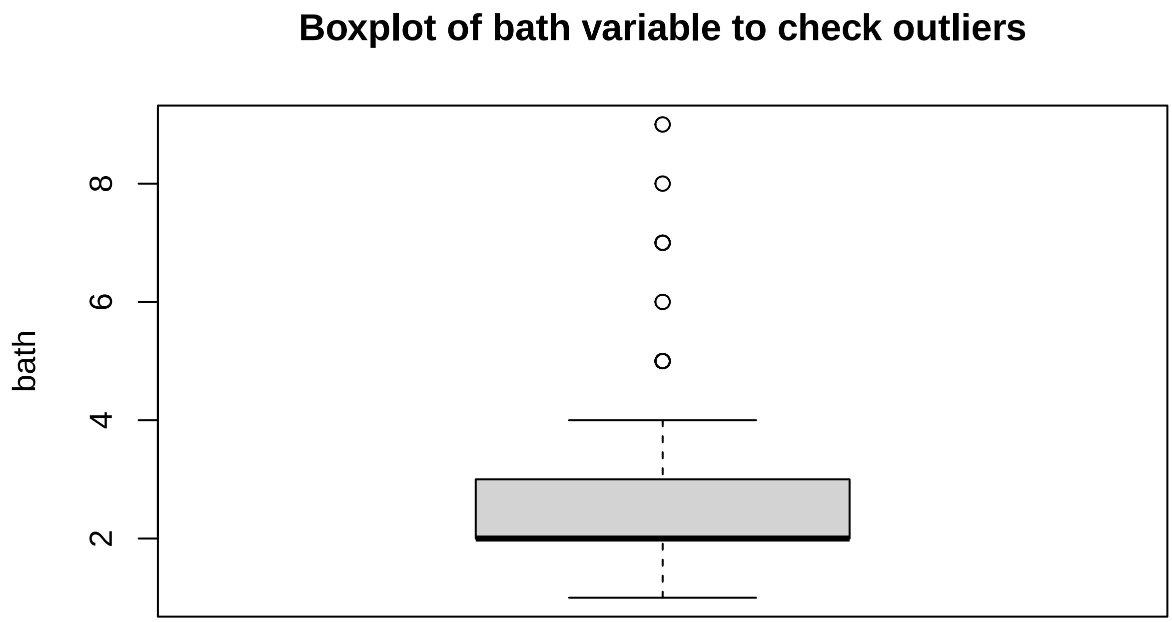
There are 61421 potential outliers in bed variable and the unique list of potential outliers is listed above. The potential outliers are even more clear from the box plot below. To check outliers in other variables, we can adopt the same approach.

```
boxplot(df$bed,
  ylab = "bed",
  main = "Boxplot of bed variable to check outliers"
)
```



For the boxplot of bath variables, we took the 128 samples of our dataset for better readability of image. These boxplots are useful as it shows the minimum, maximum, median, 1st quartile, 3rd quartile and outliers contained in the data.

```
dsample <- df[sample(nrow(df), 128), ]
boxplot(dsample$bath,
  ylab = "bath",
  main = "Boxplot of bath variable to check outliers"
)
```

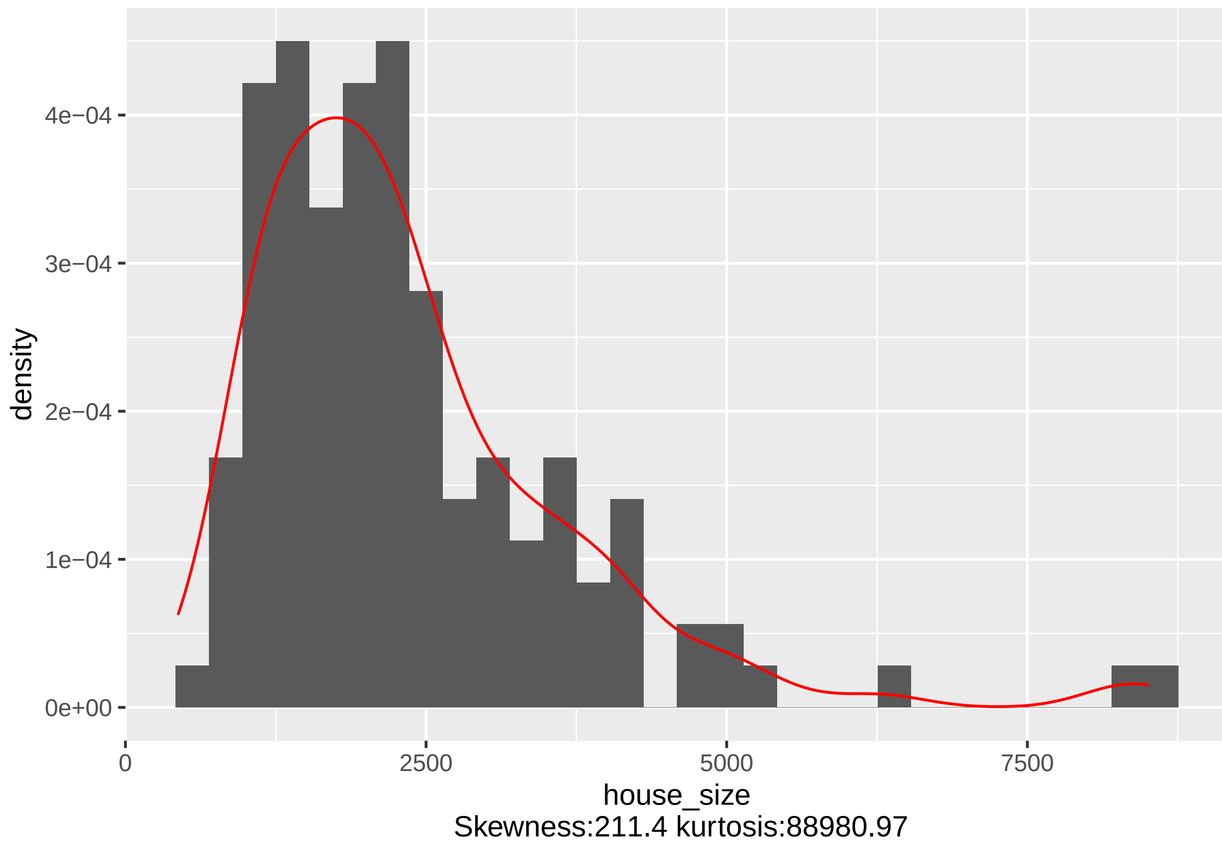


Based on the above observation, it seems like other variables also have skewed distribution. So, we explored their density plots to check the skewness of the house_size, acre_lot, and price variable. For these exploration, we took the small random sample of 128 observation and created the plots along with their skewness and kurtosis values. All these variables are right skewed with potential outliers within them.

```
dsample <- df[sample(nrow(df), 128), ]
ggplot(data=dsample, mapping=aes(x=house_size)) +
geom_histogram(aes(y=..density..), bins=30) + geom_density(color="red")+labs(x=paste0("house_size", '\n
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9
```

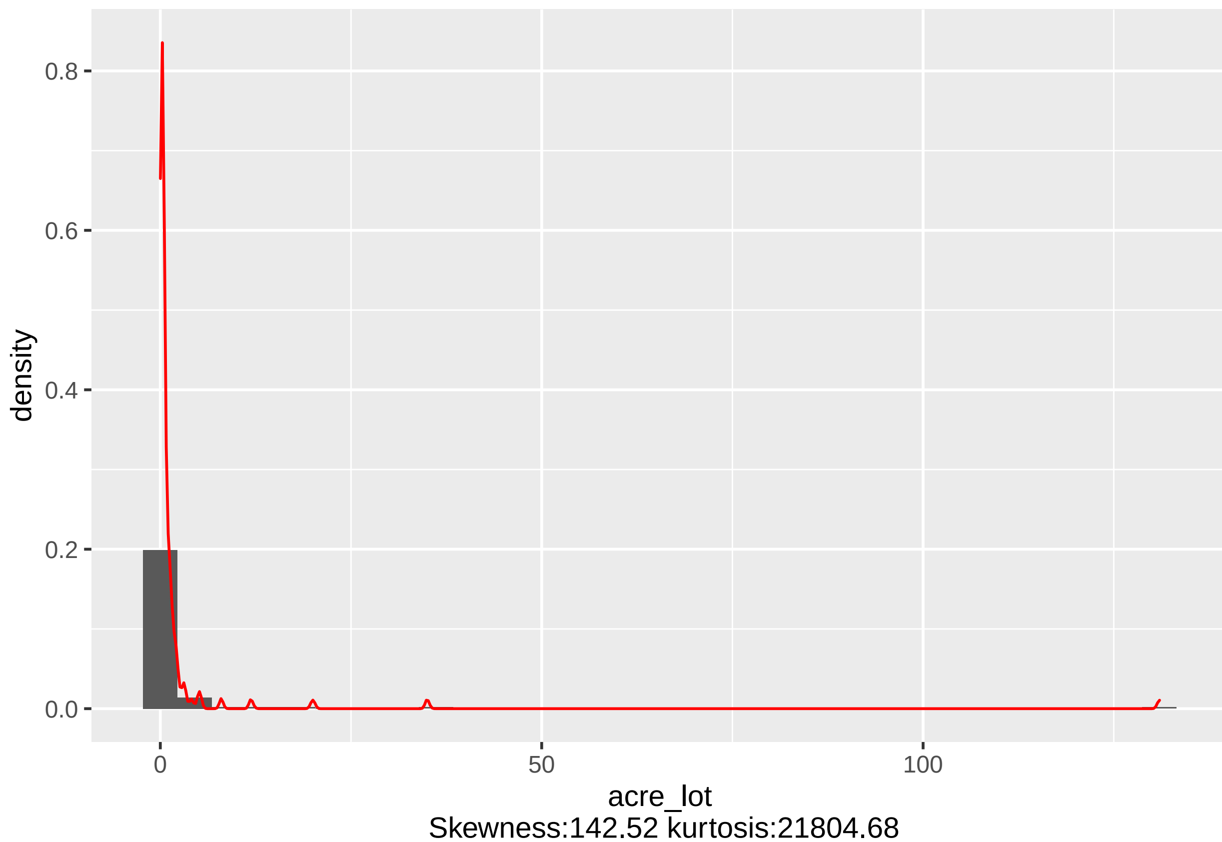
```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9
```




```
ggplot(data=dsample, mapping=aes(x=acre_lot)) +
geom_histogram(aes(y=..density..), bins=30) + geom_density(color="red")+labs(x=paste0("acre_lot", '\n',
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9
```

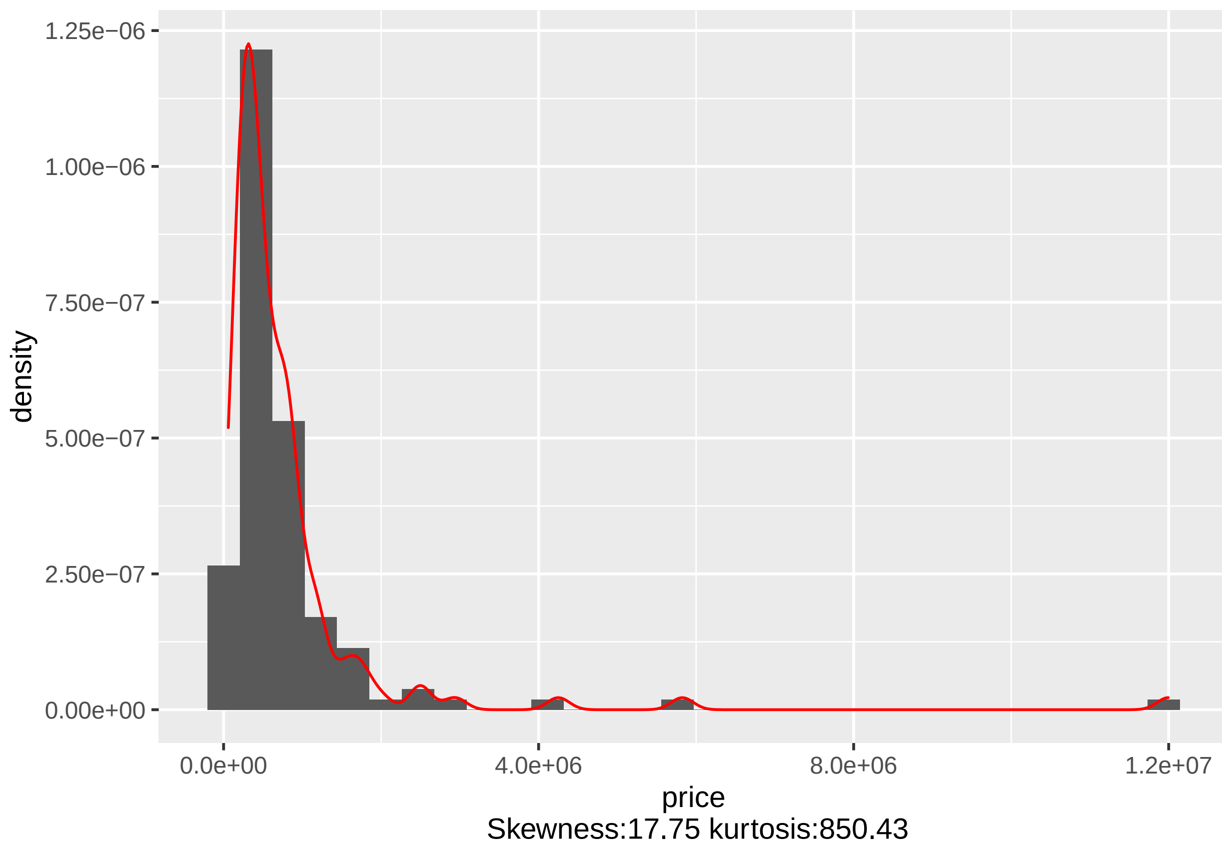
```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9
```




```
ggplot(data=dsample, mapping=aes(x=price)) +
geom_histogram(aes(y=..density..), bins=30) + geom_density(color="red")+labs(x=paste0("price", '\n', 'S
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
```

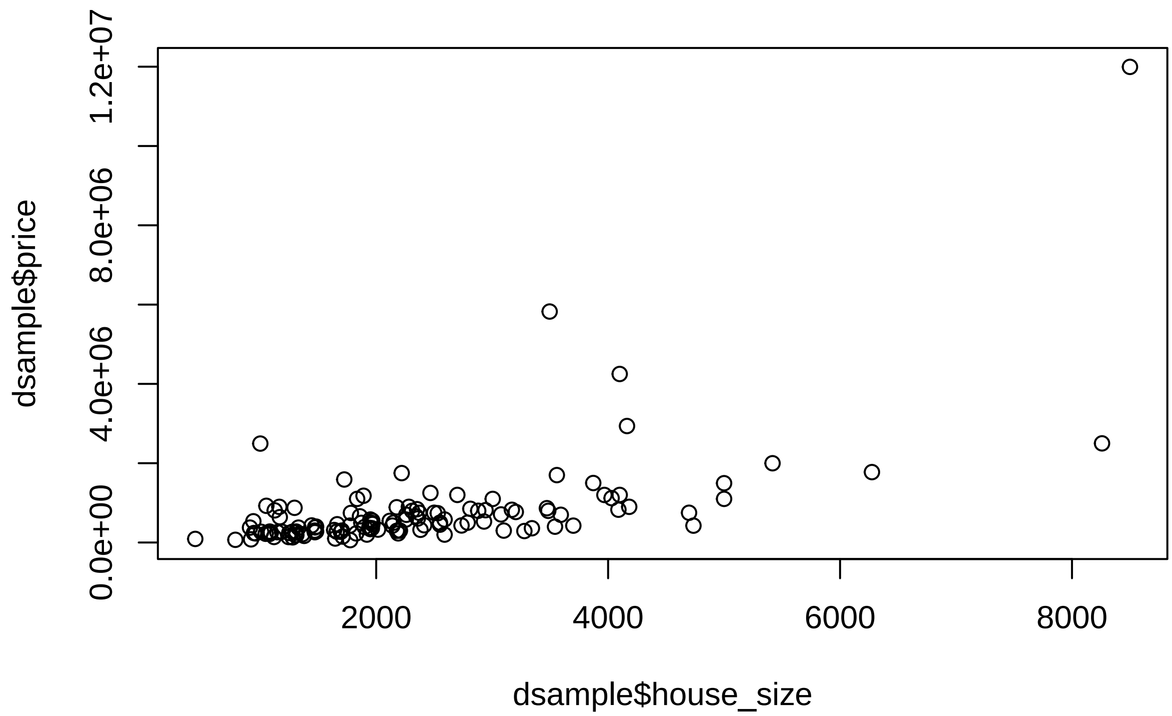
```
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9
```



Bivariate plots Now, we check what the factors affecting prices and find the relationship of other variables with price variables. Covariance, correlation and chi-square test are the common approaches/measures for bivariate data analysis. Covariance measures how two variable vary together, i.e if higher values of one variable is associated with the higher or lower values of the other variable. Positive covariance means both variable gets larger and smaller together and vice versa. Correlation measures the strength and direction of a linear relationship between two variables. A value close to 1 indicates very strong positive correlation and value close to -1 means strong negative correlation. And a value close to 0 indicates the lack of correlation between the two variables.

Now we check the relationship between house_size and price variable by checking their covariance, correlation and perform chi-square test.

```
plot(dsample$house_size, dsample$price)
```




```r
cov(df$house_size, df$price)
```

```
## [1] 1366950579
```

```r
cor(df$house_size, df$price)
```

```
## [1] 0.2770014
```

```r
chisq.test(df$house_size, df$price)
```

```
## Warning in chisq.test(df$house_size, df$price): Chi-squared approximation may be
## incorrect
```

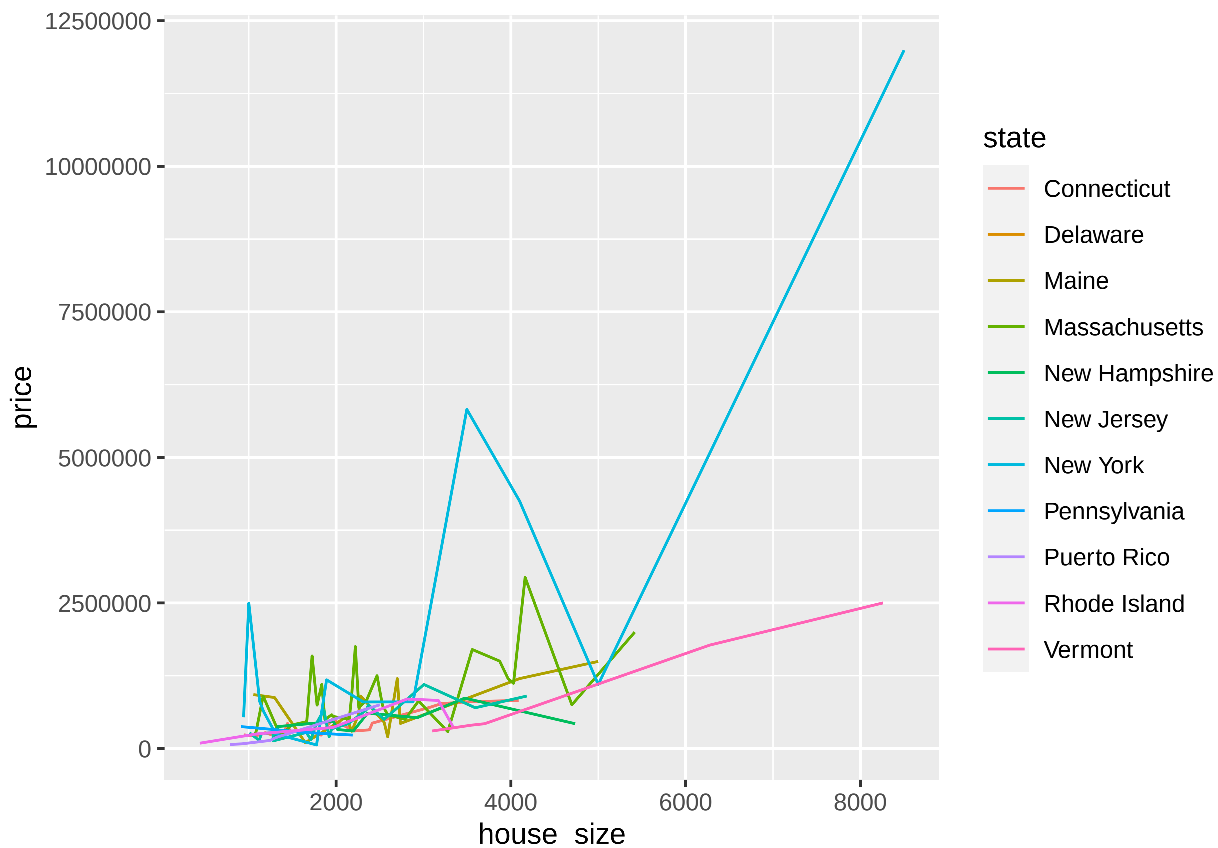
```
##
##  Pearson's Chi-squared test
##
## data:  df$house_size and df$price
## X-squared = 321627695, df = 26595132, p-value < 2.2e-16
```

We can see that the covariance is positive and higher value, which means if the house size increases, house price increases. Similarly, the correlation value is greater than 0, but not too close to 1, which means house size and price are somewhat correlated. Since the p-value is so smaller than 0.05, we have some evidence to reject the null hypothesis and assume that there is a relation between variables house size and price, and the relation has been explained by the covariance value.

relationship between bed, bath, acre_lot and price

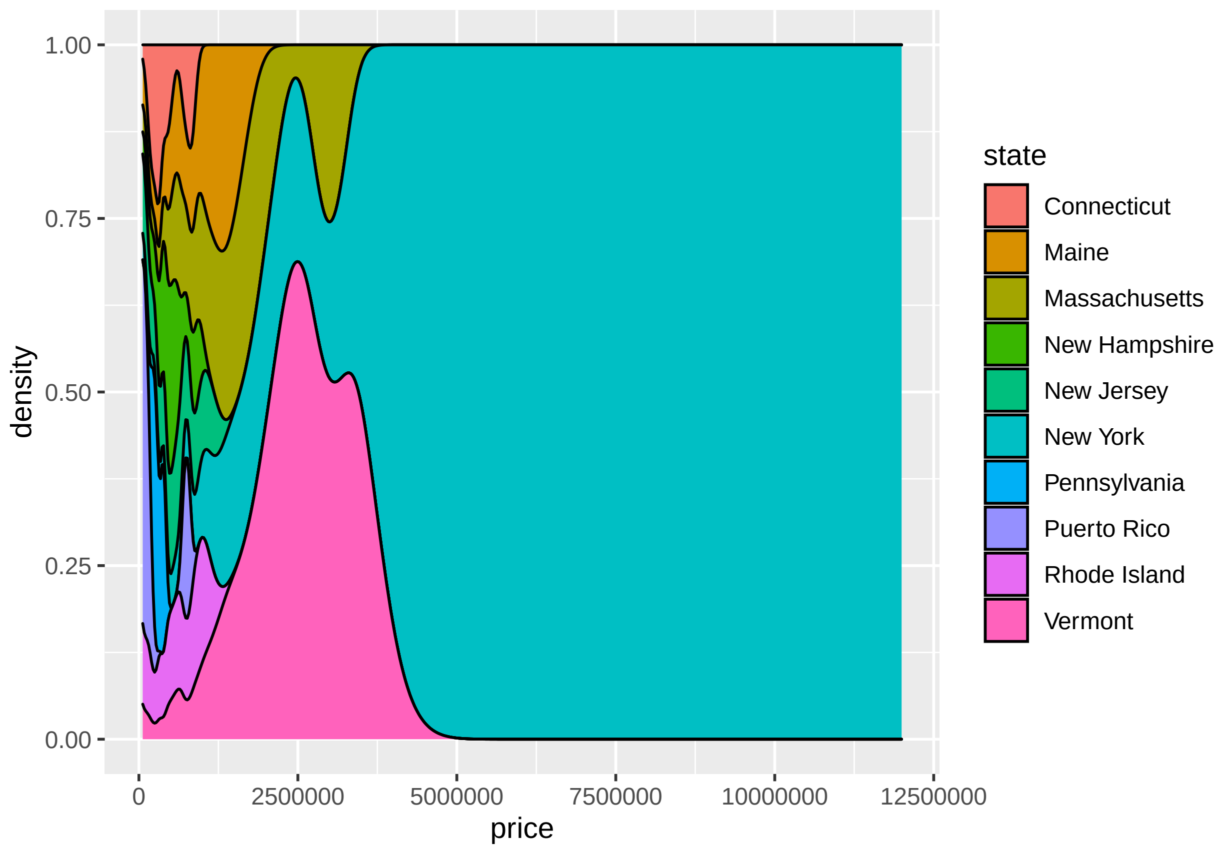
The following plots are created using the data sample and show the relationship between house size and its price for different state given in the dataset. The states like Massachusetts, New Hampshire which has highest price and includes more number of houses within the states. Similarly, the price of houses increases as the house size increase, which verifies the previous results.

```
ggplot(data=dsample, mapping=aes(x=house_size, y=price, color=state)) +
geom_line()
```



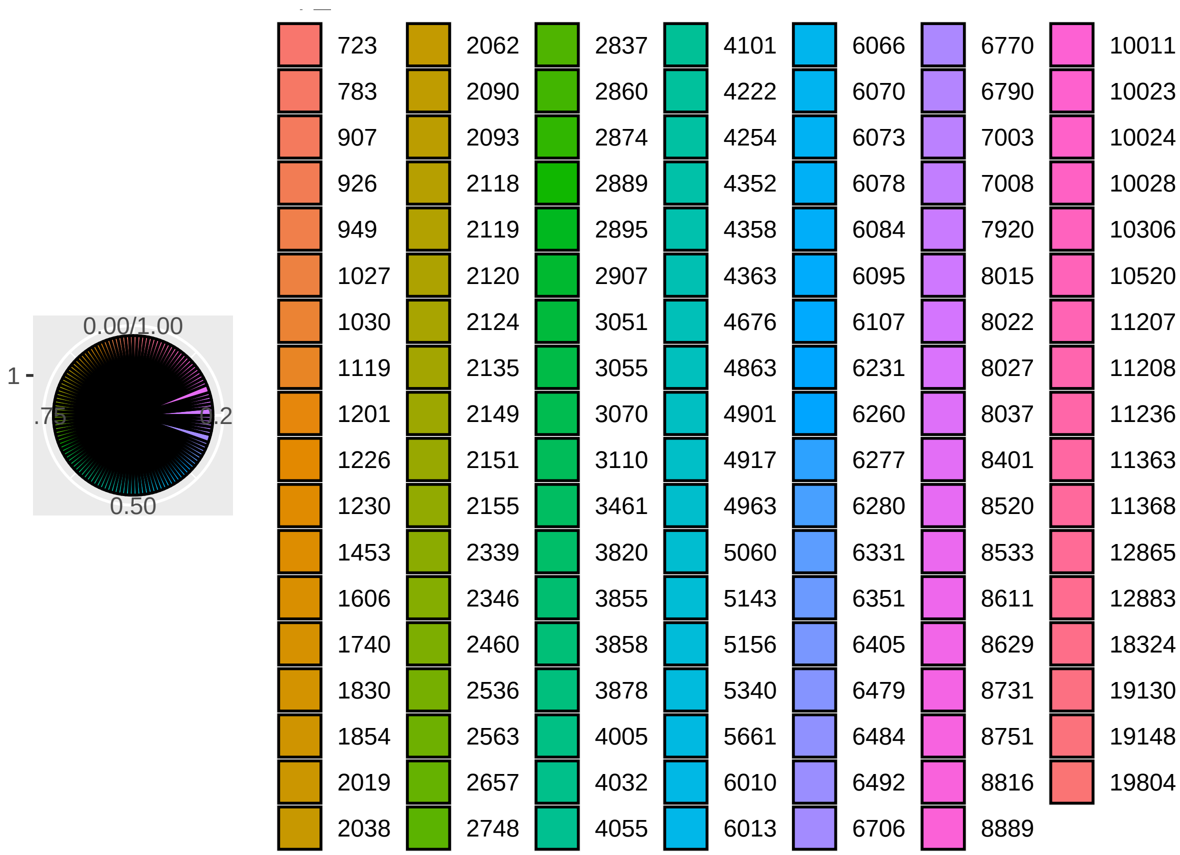

```
ggplot(data=dsample, mapping=aes(x=price, fill=state)) +
geom_density(position="fill")
```

```
## Warning: Groups with fewer than two data points have been dropped.

## Warning: Removed 1 rows containing missing values (position_stack).
```



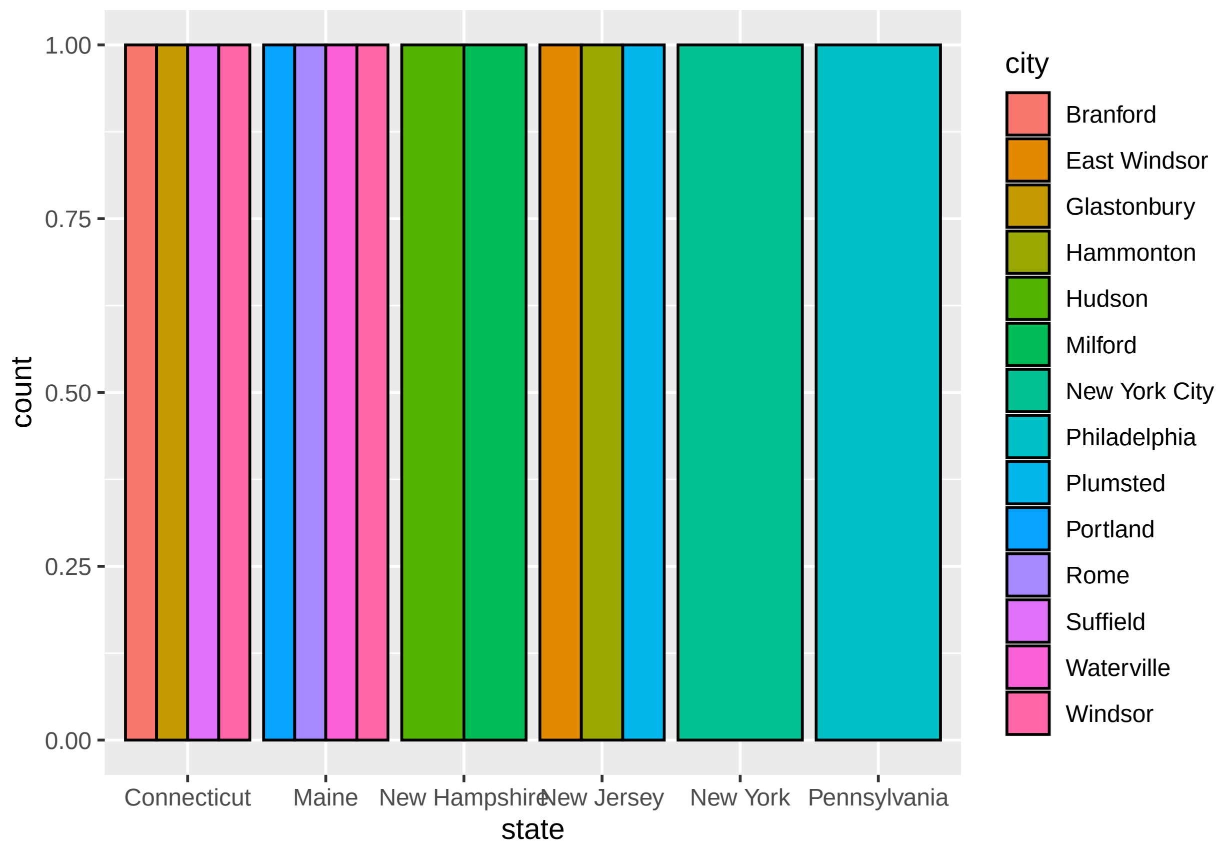
The following plot shows the distribution od data samples on the basis of its zip_code.

```
ggplot(data=dsample) + geom_bar(mapping=aes(x=factor(1), fill=zip_code), width=1,
    position="fill", color="black") + coord_polar(theta="y") + scale_y_continuous(
    name="") + scale_x_discrete(name="")
```



The following plot shows the distribution of number of cities in each state.

```
ggplot(data=dsample[1:15,], mapping=aes(x=state, fill=city)) +
geom_bar(color="black", position="dodge")
```

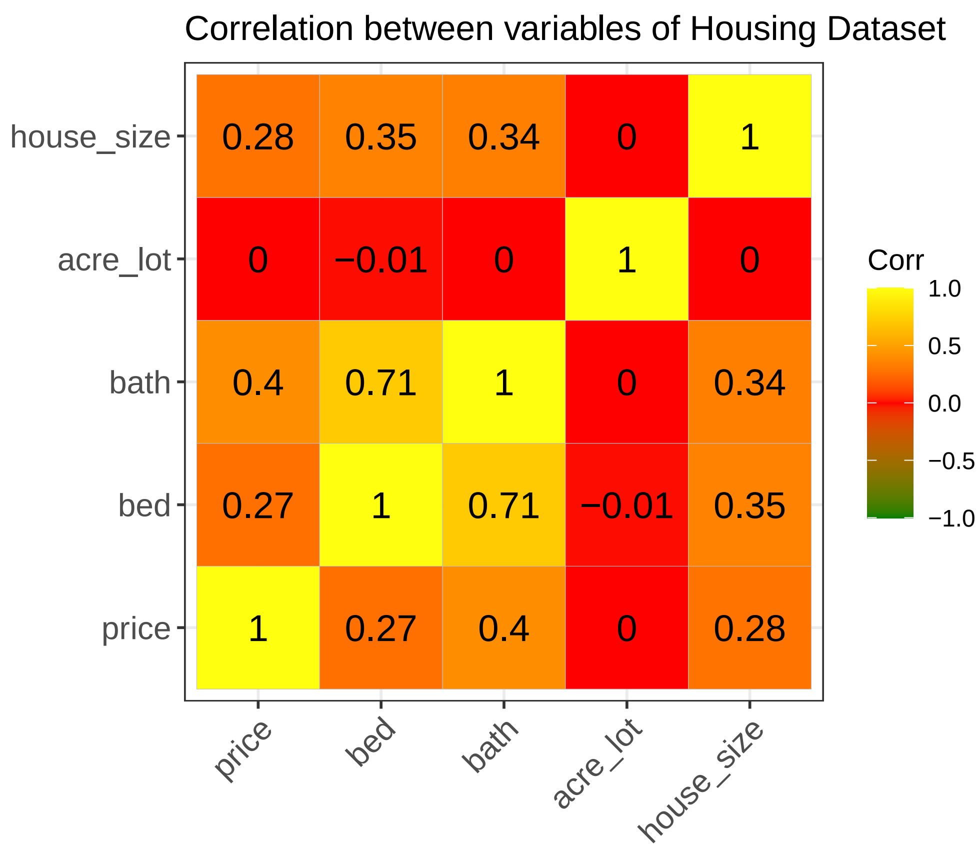


Check the correlation of each variable with another

```r
df_new <- subset(df, select= c(price, bed, bath, acre_lot, house_size))
cor(df_new)
```

```
##                   price          bed       bath      acre_lot   house_size
## price      1.0000000000  0.274215330  0.3989525  0.0004017237  0.277001388
## bed        0.2742153299  1.000000000  0.7106327 -0.0051144208  0.345525151
## bath       0.3989524971  0.710632667  1.0000000 -0.0017237000  0.341363458
## acre_lot   0.0004017237 -0.005114421 -0.0017237  1.0000000000 -0.001015233
## house_size 0.2770013885  0.345525151  0.3413635 -0.0010152331  1.000000000
```

```r
# Plot
ggcorrplot(round(cor(df_new),4),
           type = "full",
           lab = TRUE,
           lab_size = 5,
           colors = c("#008000", "#ff0001", "#ffff10"),
           title="Correlation between variables of Housing Dataset",
           ggtheme=theme_bw)
```



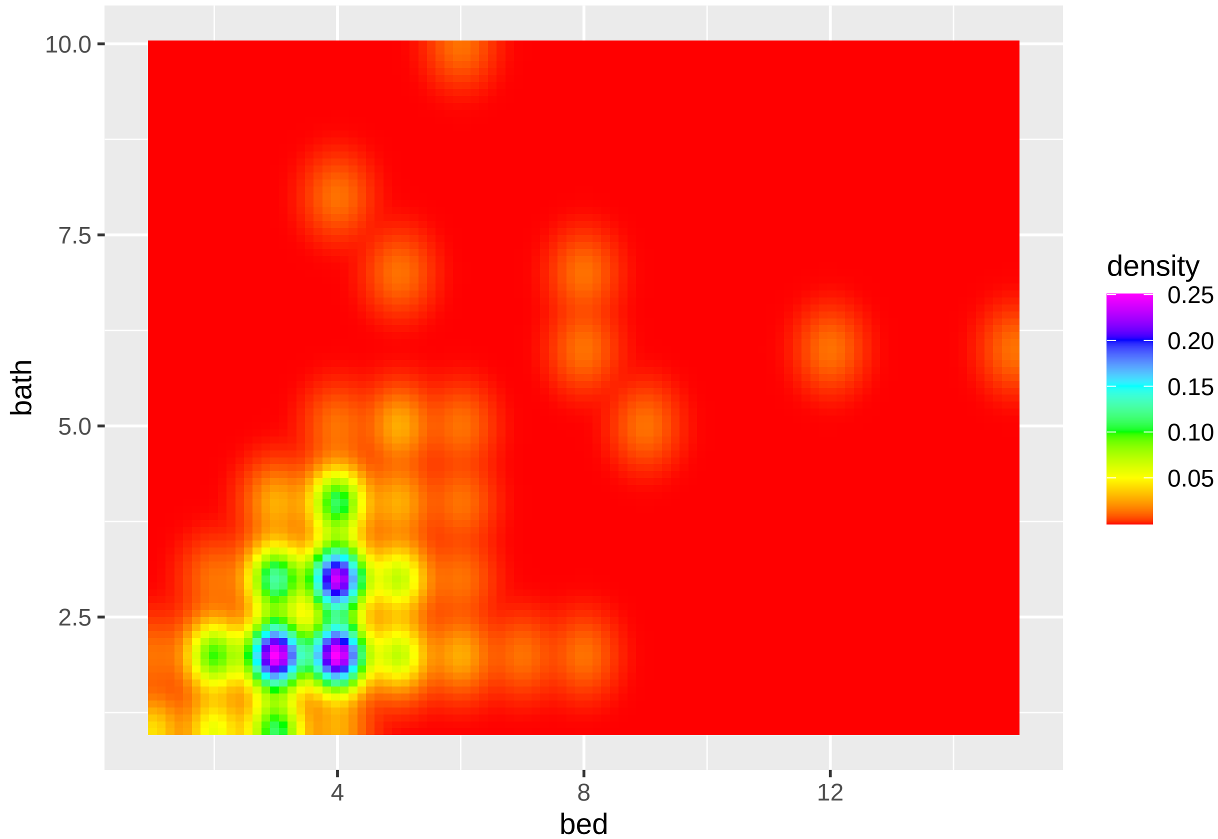
From this correlation plot, we can see that there is high correlation between number of beds and bath in a house. Variable acre_lot has very low correlation which is almost 0. There might be various reason of this. One of the reason might be the range of acre_lot is very low and below all the other variables. Normalization is to be done to make the scale of this variable similar to other variables. Another possible reason is there might be some non-linear relationship with this variable, and since correlation measures linear association between two given variables and cannot measure non-linear relation.

The other interesting thing is the correlation between bed and acre_lot is negative. A negative or inrverse correlation between two variables indicates that one variable increases while other decreases. Since there is not much description about what acre_lot means, as per the correlation value obtained it seems that as the number of bed increases, the acre_lot decreases. It can be interpreted as the size of empty land decreases as we create more bed for the house.

Beside that other variables like bed, bath, and house_size has positive correlation with price, which means as these variables increases price also increases.

Since we saw that bath and bed are highly correlated as compared to other variables, we plotted the joint density of bed and bath using stat_density2d. stat_density2d is used to estimate the joint density of two variables. From the heatmap created, we can see that the density is quite high in the left bottom corner, showing the correlation between bath and bed, which shows most of the houses have fairly equal number of bed and bath.

```
ggplot(data=dsample, mapping=aes(x=bed, y=bath))  + stat_density2d(geom="tile", contour=FALSE, aes(fill=
scale_fill_gradientn(colours = rainbow(6))
```


10.0
7.5
5.0
2.5
bath
4
8
12
bed
density
0.25
0.20
0.15
0.10
0.05