VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF APPLIED SCIENCES

PROBABILITY & STATISTICS (MT2013)

---

PROJECT

# Project 2 - Topic 1

---

Lecturer: Phan Thị Hường
Students: Nguyễn Minh Hùng - 1952737
Rousseau A. V. Thibaut - 1952001
Trần Quốc Duy - 1952214
Võ Lê Hải Đăng - 1852321
Bùi Trung Đức - 1852324

HO CHI MINH CITY, SEPTEMBER 2021

# Contents

# 1 Activity 1

This data set contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

Attribute Information:

- *price* - Price of each home sold
- *sqft_living* - Square footage of the apartments interior living space
- *floors* - Number of floors
- *condition* - An index from 1 to 5 on the condition of the apartment
- *sqft_above* - The square footage of the interior housing space that is above ground level
- *sqft_living15* - The square footage of interior housing living space for the nearest 15 neighbors

Steps:

1. Import data: **house price.csv**

2. Data cleaning: NA (Not Available)

3. Data visualization

   (a) Transformation
   (b) Descriptive statistics for each of the variables
   (c) Graphs: hist, boxplot, pairs.

4. Fitting linear regression models: We want to explore what factors may affect home prices in King County.

5. Predictions:

- Case 1:
  sqft_living15 = mean(sqft_living15), sqft_above = mean(sqft_above), sqft_living = mean(sqft_living), floor = 2, condition = 3.
- Case 2:
  sqft_living15 = max(sqft_living15), sqft_above = max(sqft_above), sqft_living = max(sqft_living), floor = 2, condition = 3.

## 1.1 Data Set Preparation (import data)

### 1.1.1 Necessary packages

Before exploring the data and building the models, we need to load some necessary packages for this data set:

```
library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(lubridate)

## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
had status 1
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(gridExtra)

##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
##
##     combine

library(caTools)
library(MASS)

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

library(caret)

## Loading required package: lattice

library(leaps)
```

### 1.1.2 Import data: house_price.csv

First we import the data set and get the data frame "house".

```
house = read.csv('house_price.csv')
head(house) # prints the first 6 rows of the data frame

##   X.2 X.1 X         id            date   price bedrooms bathrooms sqft_living
## 1   1   1 1 7129300520 20141013T000000  221900        3      1.00        1180
## 2   2   2 2 6414100192 20141209T000000  538000        3      2.25        2570
## 3   3   3 3 5631500400 20150225T000000  180000        2      1.00         770
## 4   4   4 4 2487200875 20141209T000000  604000        4      3.00        1960
## 5   5   5 5 1954400510 20150218T000000  510000        3      2.00        1680
## 6   6   6 6 7237550310 20140512T000000 1225000        4      4.50        5420
##   sqft_lot floors waterfront view condition grade sqft_above sqft_basement
## 1     5650      1          0    0         3     7       1180             0
## 2     7242      2          0    0         3     7       2170           400
## 3    10000      1          0    0         3     6        770             0
## 4     5000      1          0    0         5     7       1050           910
## 5     8080      1          0    0         3     8       1680             0
## 6   101930      1          0    0         3    11       3890          1530
##   yr_built yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1     1955            0   98178 47.5112 -122.257          1340       5650
## 2     1951         1991   98125 47.7210 -122.319          1690       7639
## 3     1933            0   98028 47.7379 -122.233          2720       8062
## 4     1965            0   98136 47.5208 -122.393          1360       5000
## 5     1987            0   98074 47.6168 -122.045          1800       7503
## 6     2001            0   98053 47.6561 -122.005          4760     101930
```

### 1.1.3 Data cleaning

Take a look at the data, we can see that the first four columns of the data set are just the numbers and IDs which will not affect the house price value, so lets remove it.

```
house = house[c(-1,-2,-3,-4)]
```

To check for NA values in the data, we can count the number of NA values.

```
# check for null values
sum(is.na(house))

## [1] 20
```

So there are 20 NA (Not Available) values in this table, so we can remove them using `na.omit()`.

```
house = na.omit(house)
```

After cleaning, we get the data frame with enough variables to prepare for the modeling: (first 6 rows)

```r
head(house)

##              date   price bedrooms bathrooms sqft_living sqft_lot floors
## 1 20141013T000000  221900        3      1.00        1180     5650      1
## 2 20141209T000000  538000        3      2.25        2570     7242      2
## 3 20150225T000000  180000        2      1.00         770    10000      1
## 4 20141209T000000  604000        4      3.00        1960     5000      1
## 5 20150218T000000  510000        3      2.00        1680     8080      1
## 6 20140512T000000 1225000        4      4.50        5420   101930      1
##   waterfront view condition grade sqft_above sqft_basement yr_built
## 1          0    0         3     7       1180             0     1955
## 2          0    0         3     7       2170           400     1951
## 3          0    0         3     6        770             0     1933
## 4          0    0         5     7       1050           910     1965
## 5          0    0         3     8       1680             0     1987
## 6          0    0         3    11       3890          1530     2001
##   yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1            0   98178 47.5112 -122.257          1340       5650
## 2         1991   98125 47.7210 -122.319          1690       7639
## 3            0   98028 47.7379 -122.233          2720       8062
## 4            0   98136 47.5208 -122.393          1360       5000
## 5            0   98074 47.6168 -122.045          1800       7503
## 6            0   98053 47.6561 -122.005          4760     101930
```

## 1.2 Data Visualization

### 1.2.1 Data transformation

As observed in the data, the data type of **date** is char vector, which is undefined for Regression, thus we have to decode and change it into 'date'. The library support for it is "lubridate".

```r
house$date = (substr(house$date, 1, 8))
house$date = ymd(house$date)
house$date = as.Date(house$date, origin = "1900-01-01")
head(house$date)

## [1] "2014-10-13" "2014-12-09" "2015-02-25" "2014-12-09" "2015-02-18"
## [6] "2014-05-12"
```

Note that:

- `substr()`: used to get the sub-string of all values in "date" column ("1" stand for start index, "8" stand for end index).

  For example, 20141013T000000 will become 20141013.

- `ymd()`: transform to year-month-date format.

- `as.Date()`: change to "date" data type.

The variable "price" which can vary a lot through many observations, so in case the histogram of the price variable is skewed, we will use the log transformation to another variable (log_price) to normalize the data.

```
log_price = log(house$price)
head(log_price)

## [1] 12.30998 13.19561 12.10071 13.31133 13.14217 14.01845
```

### 1.2.2 Descriptive statistics for each of the variables

Let it be fast and simple, we use function `summary()` to summarize the descriptive statistics for the data.

```
dim(house)      # dimention of the data frame

## [1] 21593    20

str(house)       # structure

## 'data.frame': 21593 obs. of  20 variables:
##  $ date         : Date, format: "2014-10-13" "2014-12-09" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
##  - attr(*, "na.action")= 'omit' Named int [1:20] 26 54 151 174 236 352 375 419 544 557 ...
##   ..- attr(*, "names")= chr [1:20] "26" "54" "151" "174" ...

summary(house)   # statistical summary

##       date                price            bedrooms        bathrooms
##  Min.   :2014-05-02   Min.   :  75000   Min.   : 0.000   Min.   :0.000
##  1st Qu.:2014-07-22   1st Qu.: 322000   1st Qu.: 3.000   1st Qu.:1.750
##  Median :2014-10-16   Median : 450000   Median : 3.000   Median :2.250
```

```
##  Mean   :2014-10-29   Mean   : 540068   Mean   : 3.371   Mean   :2.115
##  3rd Qu.:2015-02-17   3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500
##  Max.   :2015-05-27   Max.   :7700000   Max.   :33.000   Max.   :8.000
##   sqft_living       sqft_lot           floors        waterfront
##  Min.   :  290   Min.   :    520   Min.   :1.000   Min.   :0.000000
##  1st Qu.: 1427   1st Qu.:   5040   1st Qu.:1.000   1st Qu.:0.000000
##  Median : 1910   Median :   7620   Median :1.500   Median :0.000000
##  Mean   : 2080   Mean   :  15106   Mean   :1.494   Mean   :0.007549
##  3rd Qu.: 2550   3rd Qu.:  10687   3rd Qu.:2.000   3rd Qu.:0.000000
##  Max.   :13540   Max.   :1651359   Max.   :3.500   Max.   :1.000000
##       view          condition         grade          sqft_above
##  Min.   :0.0000   Min.   :1.000   Min.   : 1.000   Min.   : 290
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000   1st Qu.:1190
##  Median :0.0000   Median :3.000   Median : 7.000   Median :1560
##  Mean   :0.2342   Mean   :3.409   Mean   : 7.657   Mean   :1788
##  3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000   3rd Qu.:2210
##  Max.   :4.0000   Max.   :5.000   Max.   :13.000   Max.   :9410
##  sqft_basement        yr_built     yr_renovated       zipcode
##  Min.   :   0.0   Min.   :1900   Min.   :   0.0   Min.   :98001
##  1st Qu.:   0.0   1st Qu.:1951   1st Qu.:   0.0   1st Qu.:98033
##  Median :   0.0   Median :1975   Median :   0.0   Median :98065
##  Mean   : 291.4   Mean   :1971   Mean   :  84.3   Mean   :98078
##  3rd Qu.: 560.0   3rd Qu.:1997   3rd Qu.:   0.0   3rd Qu.:98118
##  Max.   :4820.0   Max.   :2015   Max.   :2015.0   Max.   :98199
##       lat             long        sqft_living15    sqft_lot15
##  Min.   :47.16   Min.   :-122.5   Min.   : 399   Min.   :   651
##  1st Qu.:47.47   1st Qu.:-122.3   1st Qu.:1490   1st Qu.:  5100
##  Median :47.57   Median :-122.2   Median :1840   Median :  7620
##  Mean   :47.56   Mean   :-122.2   Mean   :1987   Mean   : 12768
##  3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.:2360   3rd Qu.: 10083
##  Max.   :47.78   Max.   :-121.3   Max.   :6210   Max.   :871200
```

```
length(house)    # no. of columns in the data set
```

```
## [1] 20
```

```
colnames(house)    # name of columns
```

```
##  [1] "date"          "price"         "bedrooms"      "bathrooms"
##  [5] "sqft_living"   "sqft_lot"      "floors"        "waterfront"
##  [9] "view"          "condition"     "grade"         "sqft_above"
## [13] "sqft_basement" "yr_built"      "yr_renovated"  "zipcode"
## [17] "lat"           "long"          "sqft_living15" "sqft_lot15"
```

Since there are 20 variables in this data set, which will take a lot of time to process, we will choose only some predictors that are highly correlated with the dependent variable - price. To know the correlation coefficients we can use function `cor()`.

```
cor(house[-1], house$price)

##                      [,1]
## price          1.00000000
## bedrooms       0.30785621
## bathrooms      0.52503695
## sqft_living    0.70183491
## sqft_lot       0.08966225
## floors         0.25664737
## waterfront     0.26653704
## view           0.39733911
## condition      0.03651400
## grade          0.66714087
## sqft_above     0.60552743
## sqft_basement  0.32333245
## yr_built       0.05415078
## yr_renovated   0.12582504
## zipcode       -0.05317327
## lat            0.30694897
## long           0.02180145
## sqft_living15  0.58516115
## sqft_lot15     0.08233226
```

According to our correlation, price is highly correlated with bedrooms, bathroom, Sqft$_{l}iving, view, grade, sqft_{a}bove$

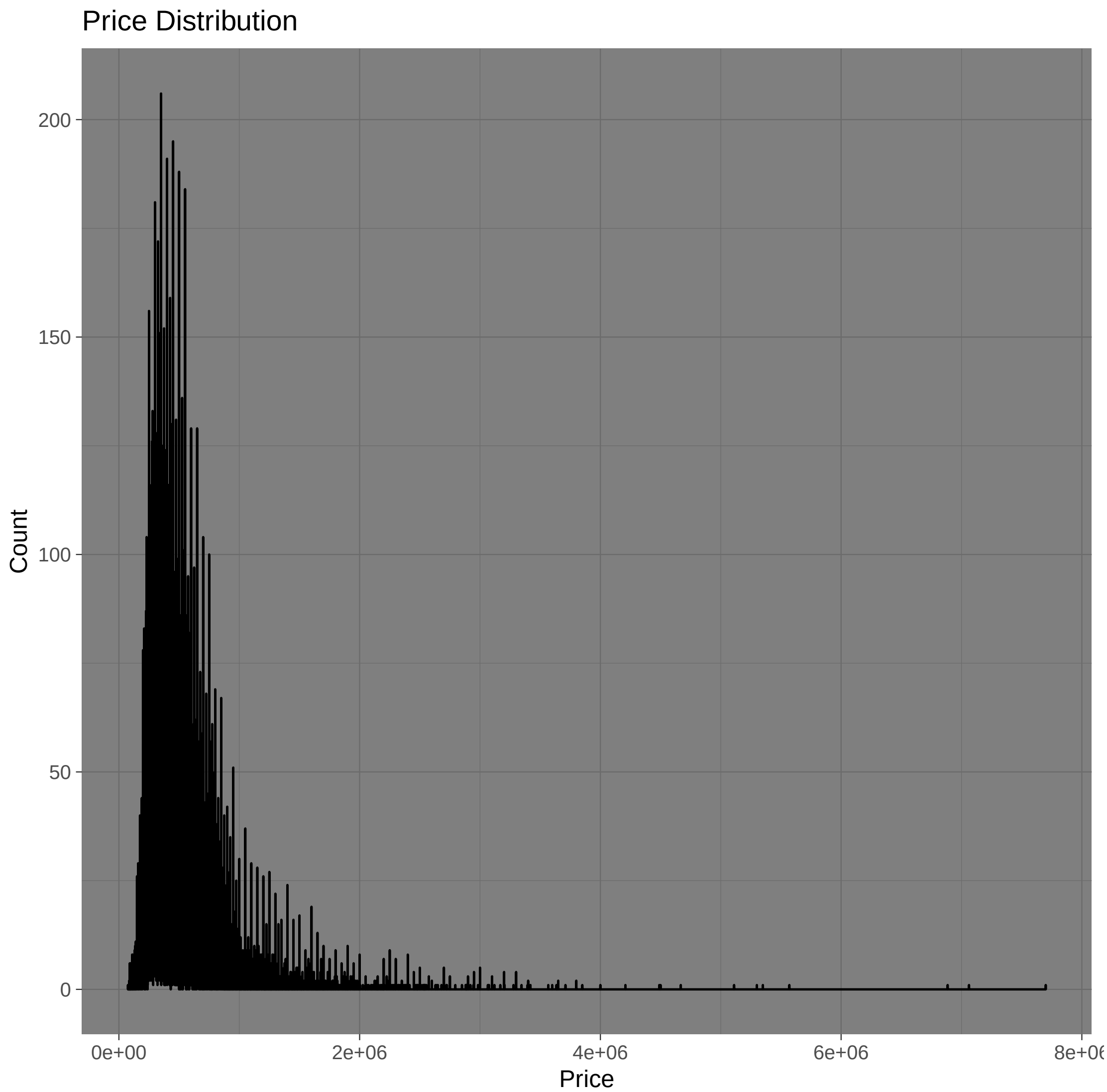
### 1.2.3 Histogram, Box plot and pairs

a) **Histogram for `house$price`**

A **histogram** represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.

In R, the library "ggplot2" support function **`ggplot()`** to illustrate the plot, especially **`geom_histogram()`**(using bars) and **`geom_freqpoly()`**(using lines).

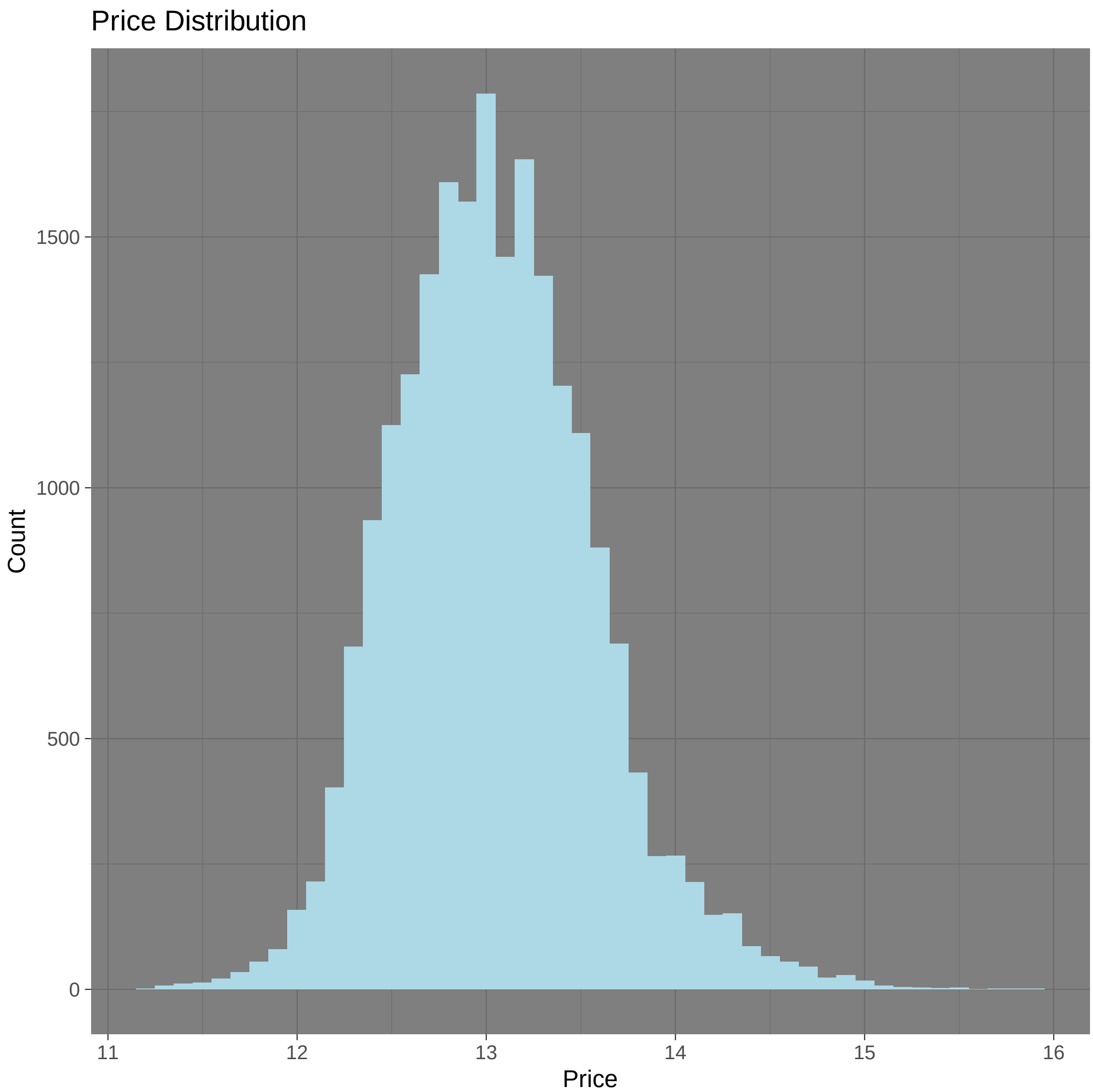
```
#PRICE DISTRIBUTION:
ggplot(house,aes(x=price))+
geom_freqpoly(binwidth = 500)+
theme_dark()+
xlab('Price')+
ylab('Count')+
ggtitle('Price Distribution')
```




```
# more number of houses cost in the range of 0 to 1.6 M
```

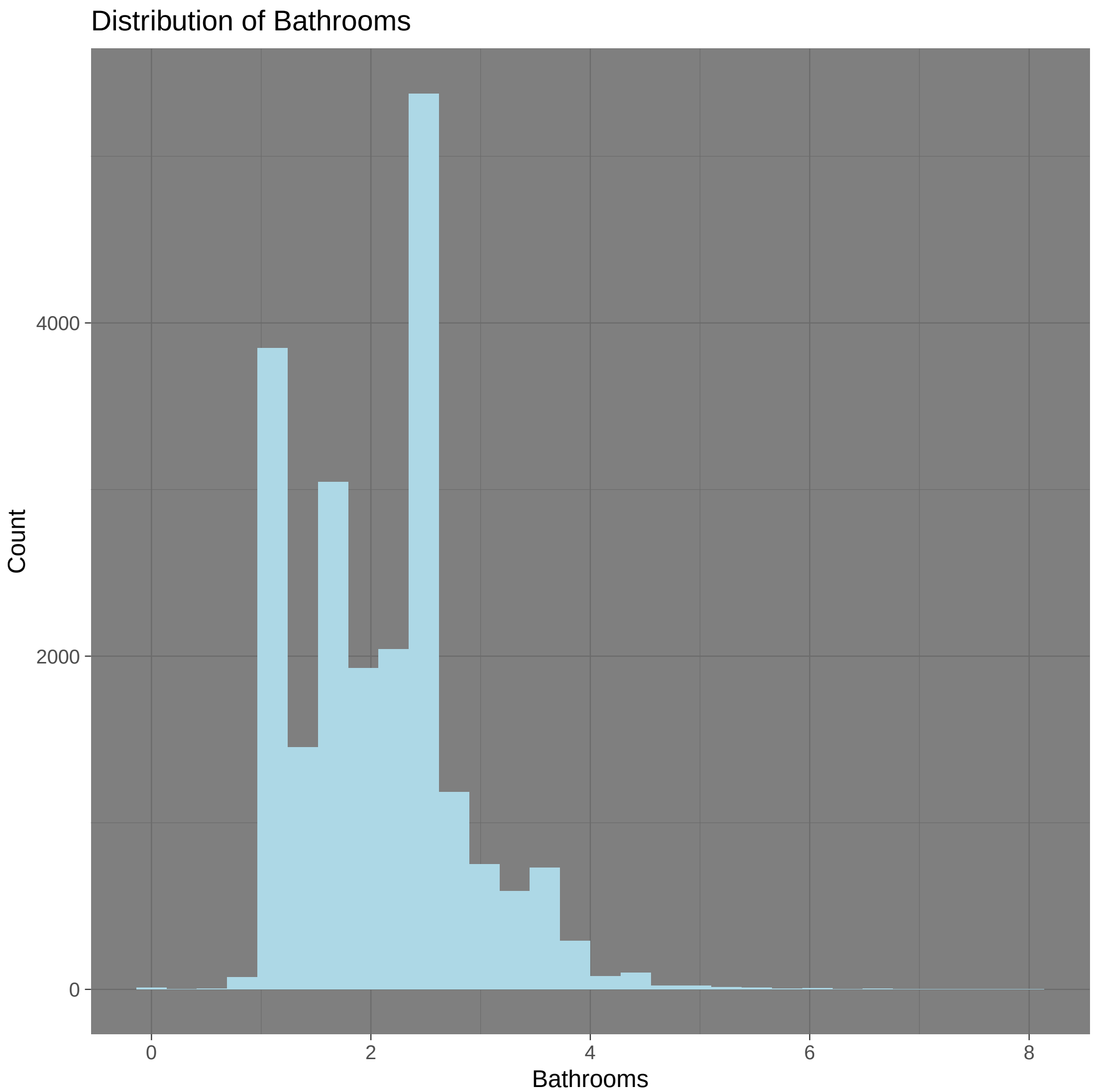
As can be seen, the graph aren't easy to imagine so we should normalize it using log transformation as we have discussed (done with variable log_price).

```
ggplot(house,aes(log_price))+
geom_histogram(fill='lightblue',binwidth = 0.10)+
theme_dark()+
xlab('Price')+
ylab('Count')+
ggtitle('Price Distribution')
```



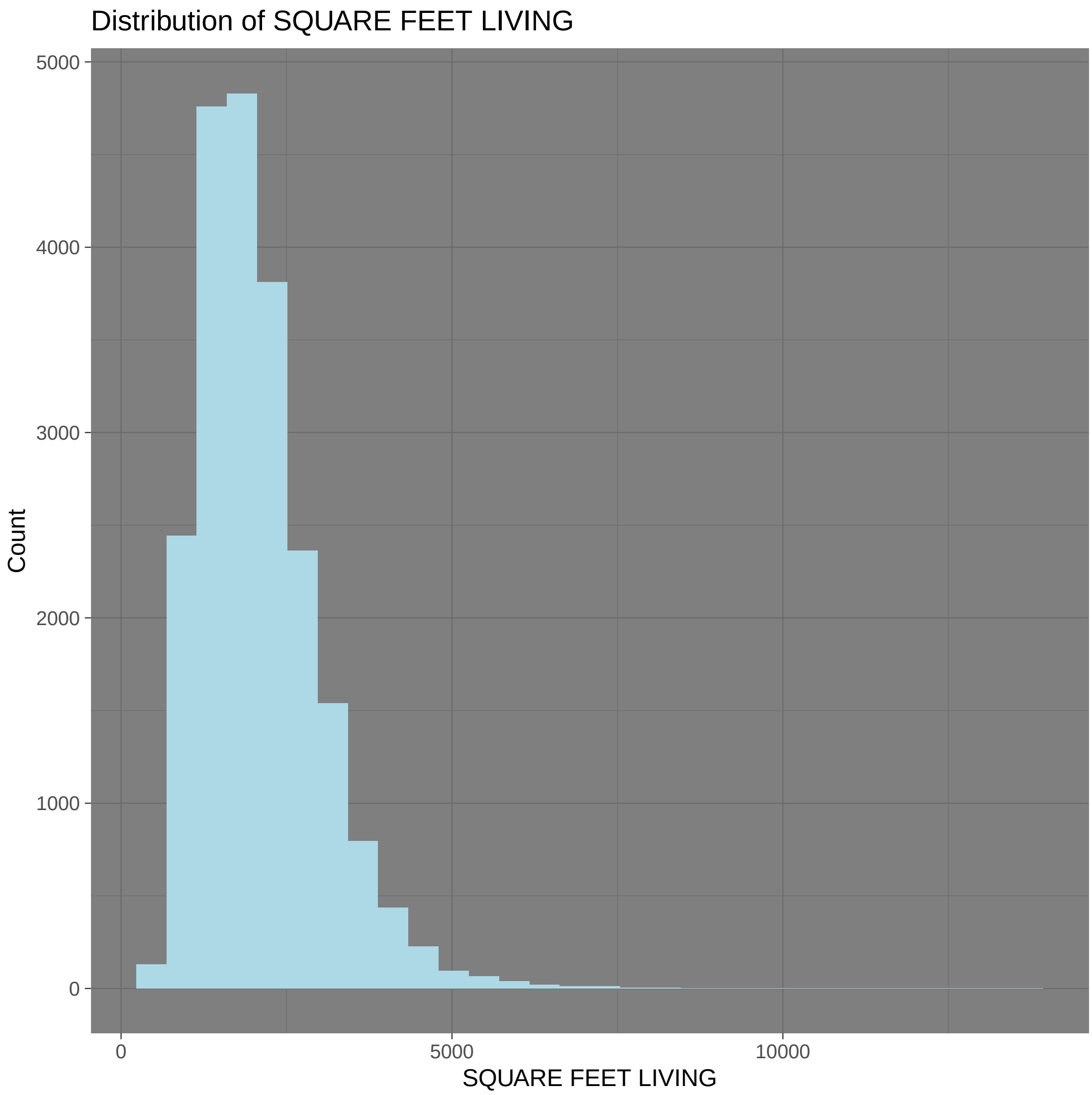

Histogram for bathroom:

```
#BATHROOM DISTRIBUTION: 0.53
ggplot(house,aes(bathrooms))+
  geom_histogram(fill='lightblue')+
  theme_dark()+
  xlab('Bathrooms')+
  ylab('Count')+
  ggtitle('Distribution of Bathrooms')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



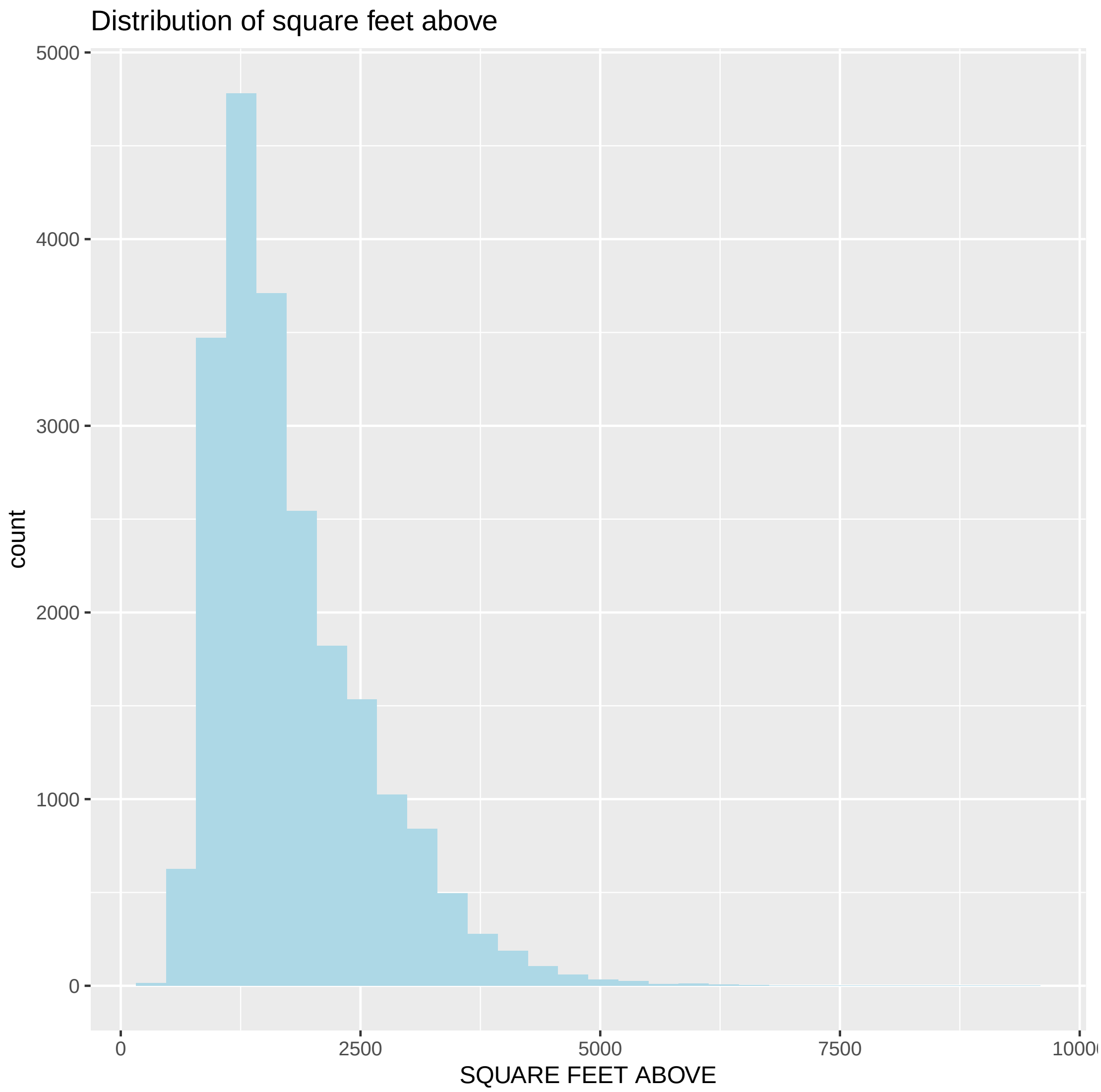

Histogram for sqrt_living:

```
#SQUARE FEET LIVING DISTRIBUTION: 0.70
log_size=log10(house$sqft_living)
ggplot(house,aes(sqft_living))+
  geom_histogram(fill='lightblue')+
  theme_dark()+
  xlab('SQUARE FEET LIVING')+
  ylab('Count')+
  ggtitle('Distribution of SQUARE FEET LIVING')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



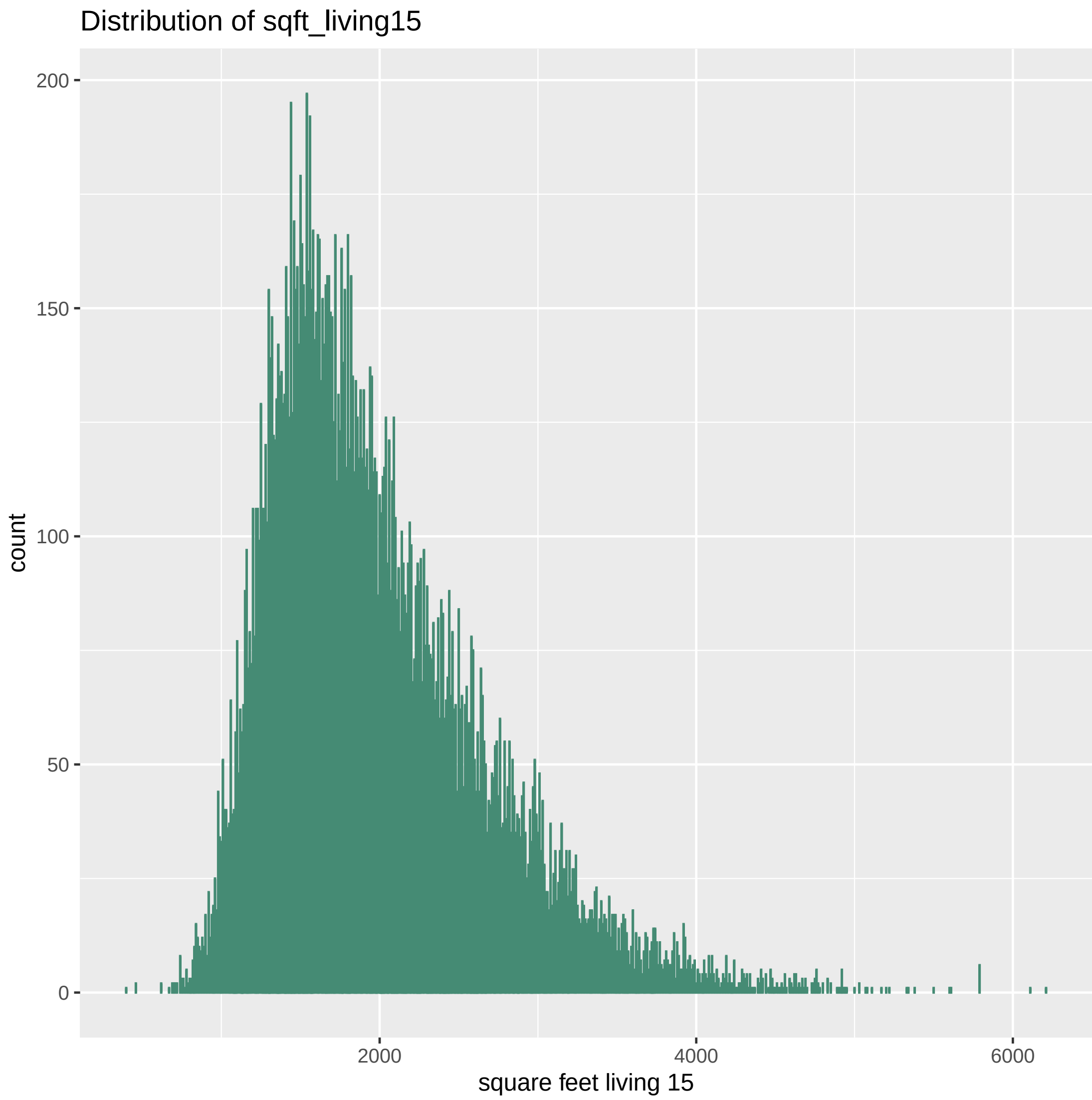

Histogram for sqrt_above:

```
#SQUARE FEET ABOVE DISTRIBUTION: 0.61
ggplot(house,aes(sqft_above))+
  geom_histogram(fill='lightblue')+
  xlab('SQUARE FEET ABOVE')+
  ggtitle("Distribution of square feet above")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




Histogram for square feet living 15:

```
#SQRT_LIVING 15 : 0.70
ggplot(house,aes(sqft_living15))+
  geom_bar(color = 'aquamarine4')+
  xlab('square feet living 15')+
  ggtitle("Distribution of sqft_living15 ")
```



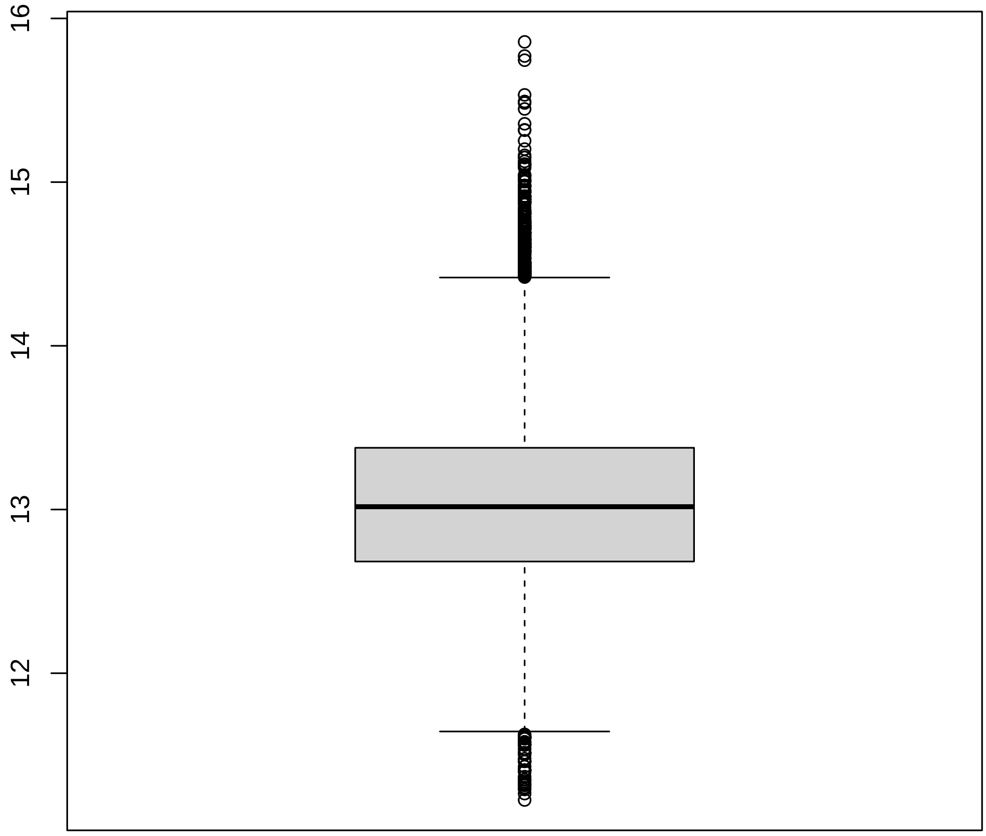

For the two categorical variables(view and grade) we draw box plots to understand the relationship.

b) **Boxplot:**

**Boxplots** are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing box plots for each of them.
Box plots are created in R by using the `boxplot()` function.

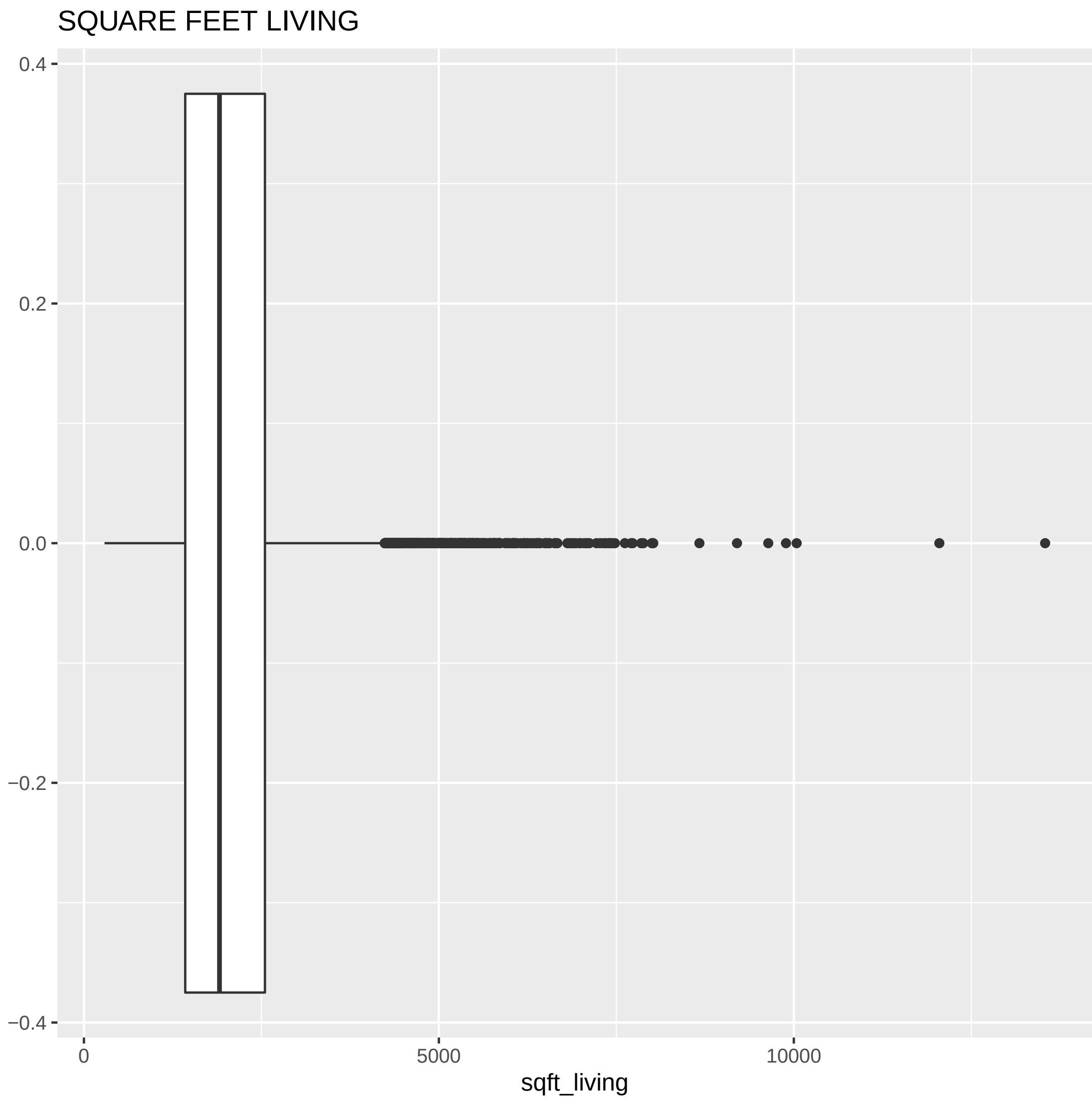
Boxplot for log_price:

```
boxplot(log_price)
```



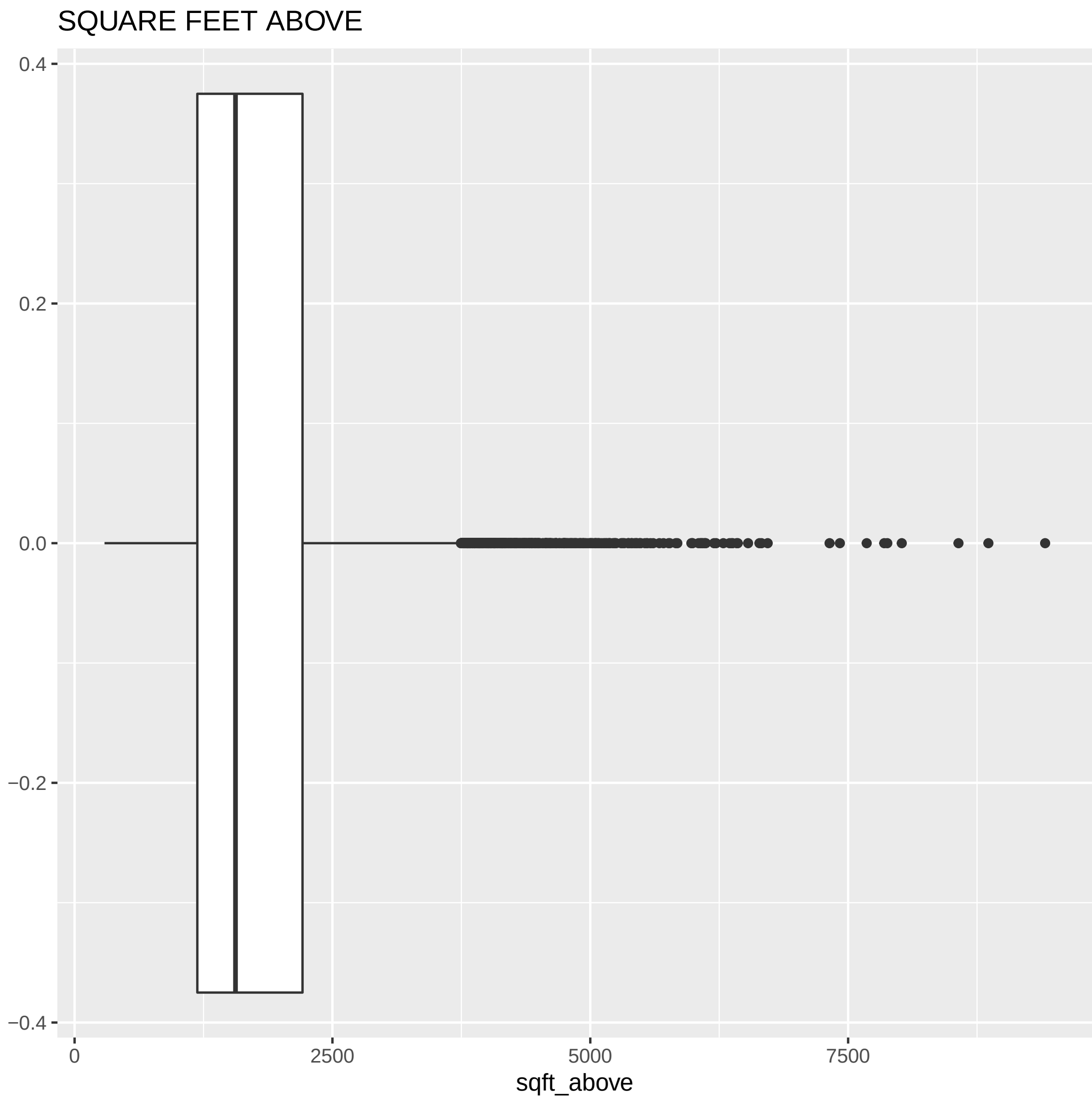

Boxplot for sqft_living:

```
ggplot(house,aes(sqft_living))+
geom_boxplot()+
ggtitle('SQUARE FEET LIVING')
```



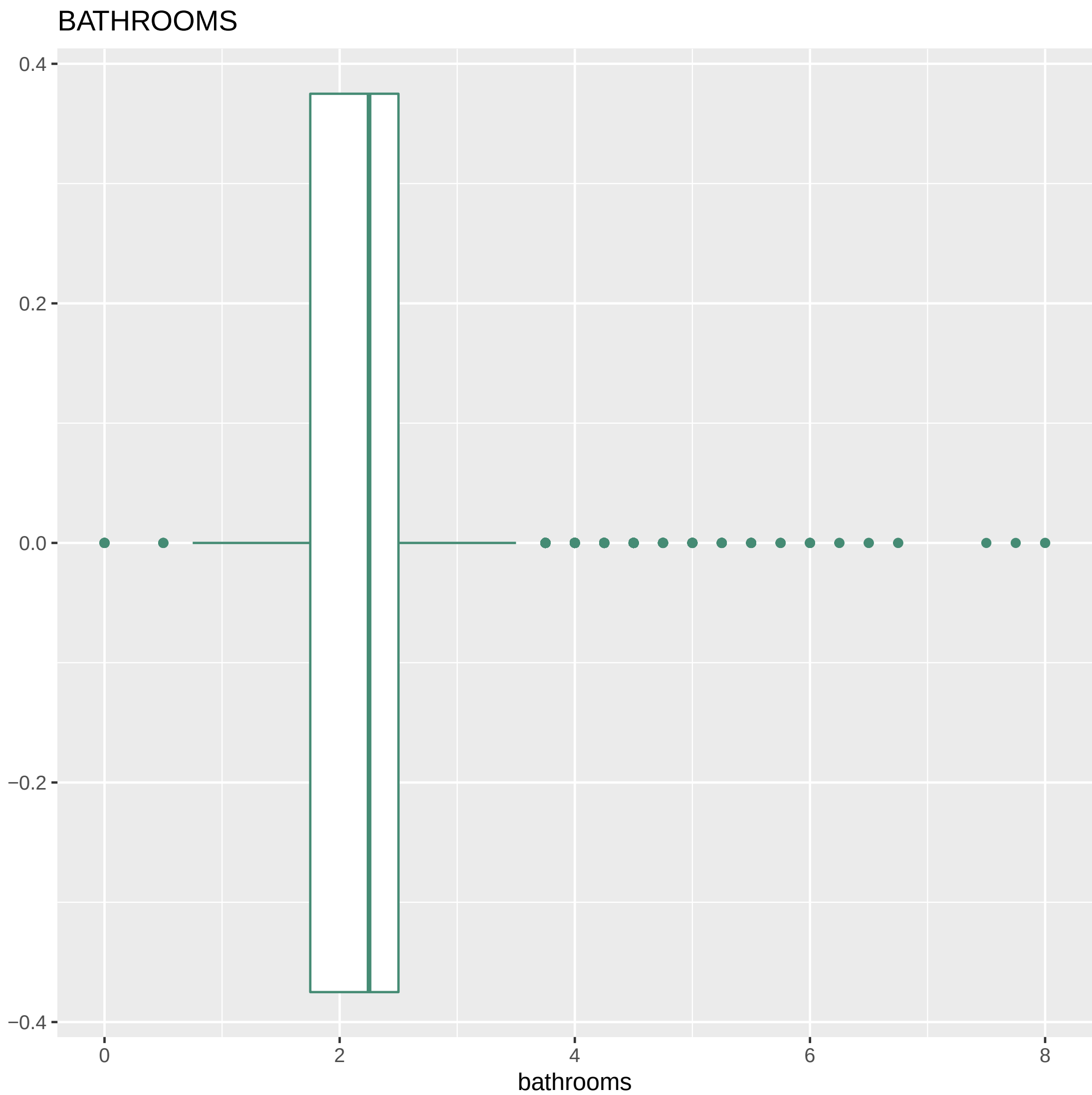

Boxplot for sqft_above:

```
ggplot(house,aes(sqft_above))+
geom_boxplot()+
ggtitle('SQUARE FEET ABOVE')
```



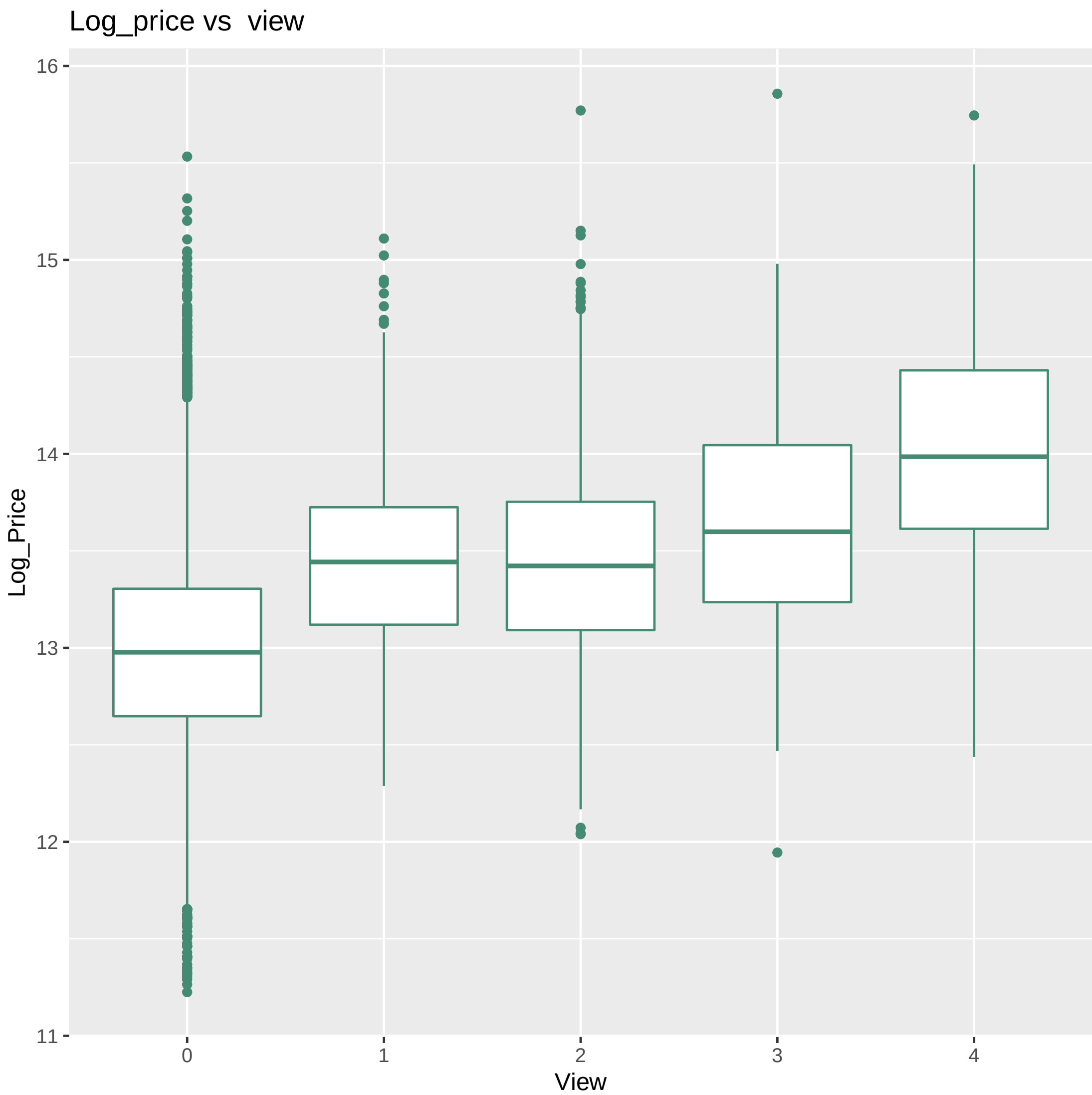

Boxplot for bathrooms:

```
ggplot(house,aes(bathrooms))+
geom_boxplot(color = 'aquamarine4')+
ggtitle("BATHROOMS")
```



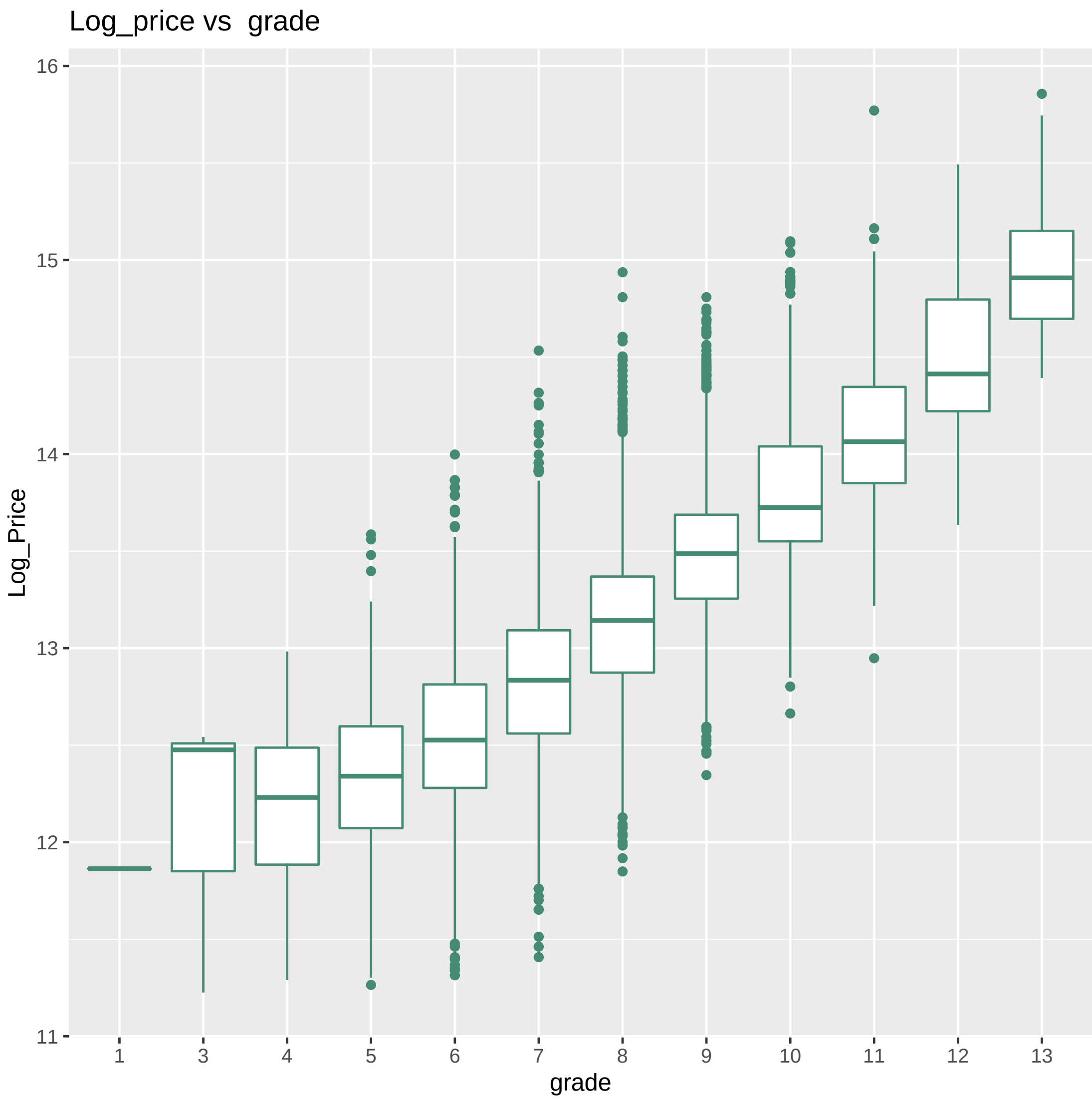

Boxplot for view:

```
ggplot(house,aes(x=factor(view),y= log_price))+
geom_boxplot(color = 'aquamarine4')+
geom_smooth(method = "lm")+
xlab('View')+
ylab('Log_Price')+
ggtitle("Log_price vs  view")

## `geom_smooth()` using formula 'y ~ x'
```



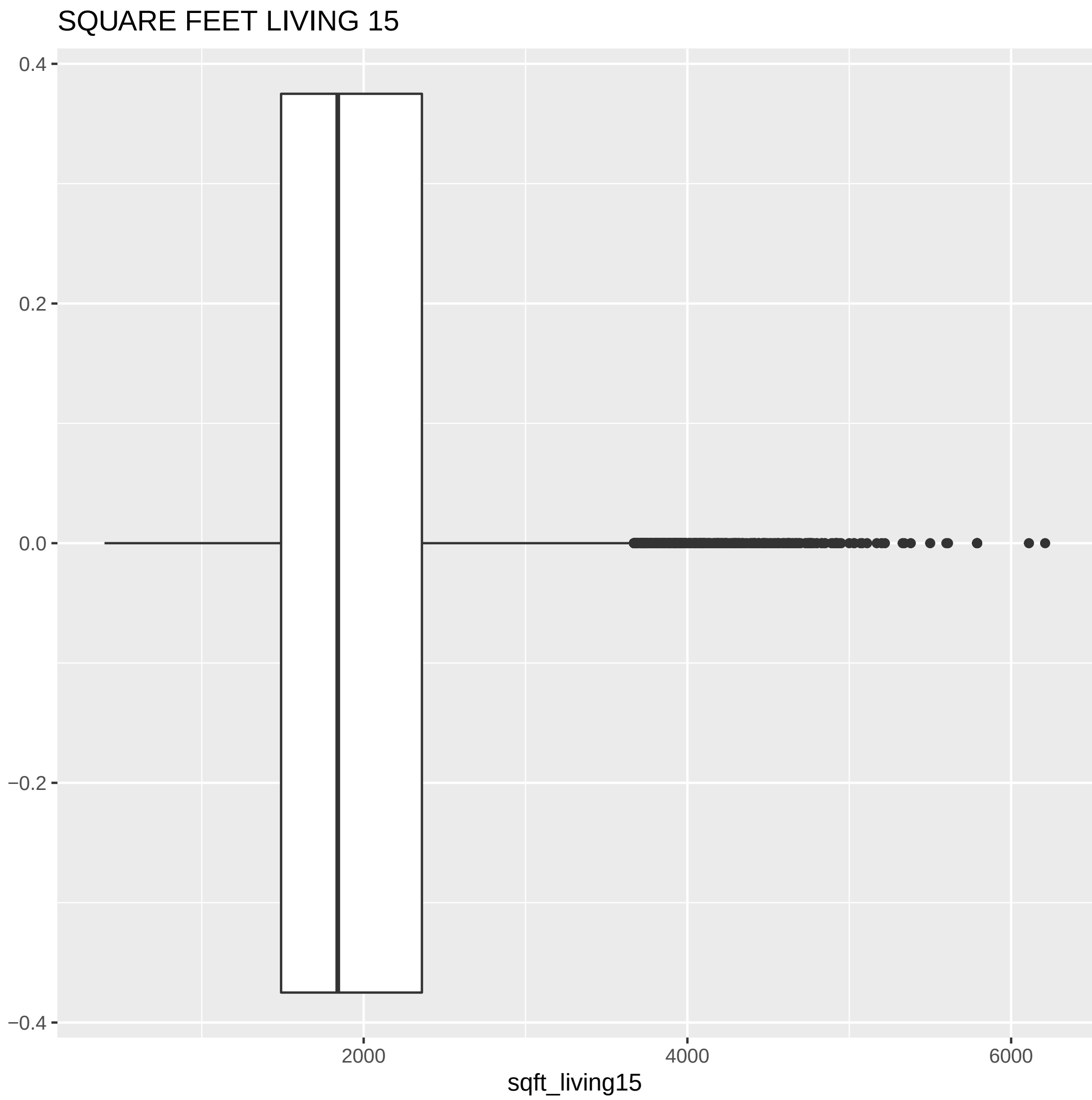

Boxplot for grade:

```
ggplot(house,aes(x=factor(grade),y= log_price))+
geom_boxplot(color = 'aquamarine4')+
geom_smooth(method = "lm")+
xlab('grade')+
ylab('Log_Price')+
ggtitle("Log_price vs  grade")

## `geom_smooth()` using formula 'y ~ x'
```




Boxplot for sqft_living15:

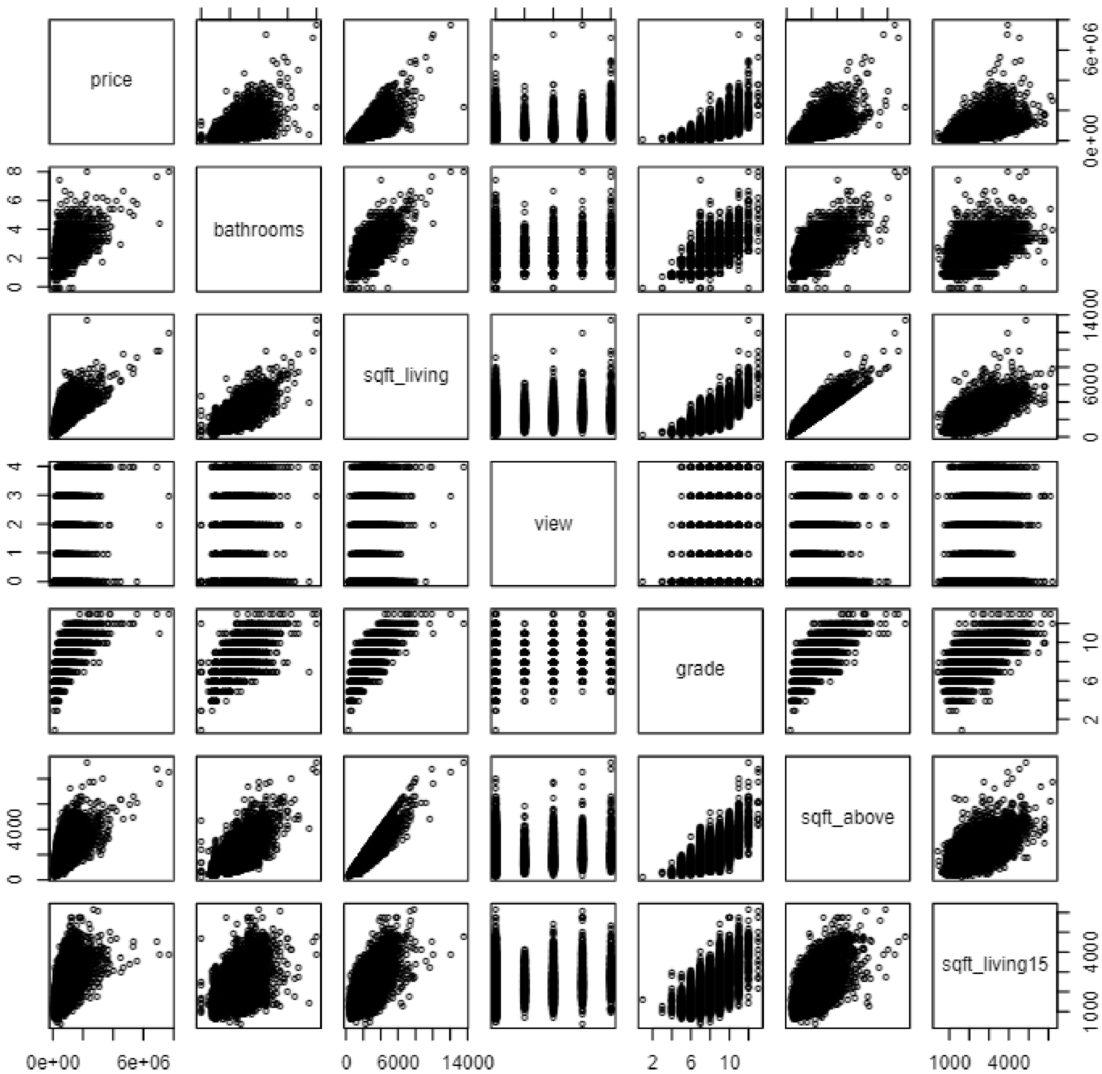
```
ggplot(house,aes(sqft_living15))+
geom_boxplot()+
ggtitle('SQUARE FEET LIVING 15')
```




b) **Pairs:**

The R function "pairs" returns a plot matrix, consisting of **scatter plots** for each variable-combination of a data frame. The basic R syntax for the pairs command is shown as follows.

```
pairs(~ price + bathrooms + sqft_living + view + grade +sqft_above+sqft_
living15,data = house)
```




## 1.3 Fitting linear regression models

### 1.3.1 Motivation

The data-set consists the prices and other attributes of almost 22,000 houses. It's a great data-set for us to build and train a model for the prediction of house price in the future. Hence, we need to explore whether sqft_living, bathrooms, bedrooms, floor,... may affect most on the house price. We extensively utilized the Multiple Linear Regression to analyze the relationship between the price and other attributes then applied this model for later prediction. The approach to this method will be discussed in this section.

### 1.3.2 MLR model

Regression analysis is a collection of statistical tools that are used to model and explore relationships between variables that are related in a nondeterministic manner.
**Multiple Linear Regression (MLR)** attempts to model a linear relationship between a dependent variable (response) and some independent variables (predictors/regressors). A model that might describe this relationship is

$$Y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon$$

where $\beta_0, \beta_1, ..., \beta_n$ are called partial regression coefficients since $\beta_i$ measures the change in Y per unit change in x_i when the other variables are kept constant.

### 1.3.3 Assumption

There are four assumptions associated with a linear regression model

- **Linearity**: The relationship between independent and dependent variables is linear. The linearity assumption can be checked with scatterplot of response versus regressor.
- **Independence**: There is no multicollinearity, i.e. dependencies between the independent variables. This assumption can be tested by correlation matrix or variance inflation factor. The magnitude of correlation coefficients greater than 0.8 or the VIF values exceed 10 indicates that multicollinearity is a problem.
- **Homoscedasticity**: The variance of residual is the same for any value of X. A scatterplot between residuals versus predicted values is a good way to check this assumption. There should be no pattern such as the cone-shaped pattern in the distribution.
- **Normality**: The errors between observed and predicted values should be normally distributed. We can check this assumption by looking at the histogram or Q-Q plot, or by applying goodness of fit on residuals.

### 1.3.4 Variables selection

We have chosen price as dependent variables, but the problem is selecting the independent variables. Picking all variables as regressors is unnecessarily costly and sometimes it has a negative effect on our model since some variables are irrelevant to the response or there is no linear relationship between them.
An important problem in many applications of regression analysis involves selecting the set of regressor variables to be used in the model.
In such a situation, we are interested in variable selection; that is, screening the candidate variables to obtain a regression model that contains the "best" subset of regressor variables. To keep model maintenance costs to a minimum and to make the model easy to use, we would like the model to use as few regressor variables as possible. Hence, price, bedrooms, bathrooms, sqft_living, view , grade, sqft_above, and sqft_living15 were considered for the full model based on above plots.

A linear model was fit to determine the relationship. The results are shown below.

```r
model <- dplyr::select(house, price, bedrooms, bathrooms, sqft_living, sqft_above, sqft_living15,
```

The model will be built using the stepwise regression method based on AIC. AIC stands for Akaike Information Criteria. It evaluates the quality of different models relative to the other models. The lower AIC is the better. The stepwise regression method will add or subtract each variable in each step based on whether the model produced has a higher or lower AIC. stepAIC will also remove multicollinearity, the situation in which two or more explanatory variables are highly related.
The R function step() can be used to perform variable selection. First, we have to create a linear model `null` to form a model with no regressor variable (or one highest correlation variable) and a `full` model using all regressor variables.

```r
null=lm(price~1, data = model)
full=lm(price~.,data = model)
summary(null)

##
## Call:
## lm(formula = price ~ 1, data = model)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -465068 -218068  -90068  104932 7159932
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   540068       2498   216.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 367100 on 21592 degrees of freedom

summary(full)

##
## Call:
## lm(formula = price ~ ., data = model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1260994  -125023   -19483    95556  4609109
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -4.802e+05  1.460e+04 -32.894  < 2e-16 ***
## bedrooms      -3.206e+04  2.211e+03 -14.499  < 2e-16 ***
## bathrooms     -2.057e+04  3.352e+03  -6.136 8.62e-10 ***
## sqft_living    2.296e+02  4.695e+00  48.911  < 2e-16 ***
## sqft_above    -4.528e+01  4.389e+00 -10.315  < 2e-16 ***
```

```
## sqft_living15  3.862e+00  3.896e+00   0.991    0.322
## grade          9.753e+04  2.404e+03  40.568  < 2e-16 ***
## view           8.858e+04  2.285e+03  38.763  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 238000 on 21585 degrees of freedom
## Multiple R-squared:  0.5797,Adjusted R-squared:  0.5796
## F-statistic:  4254 on 7 and 21585 DF,  p-value: < 2.2e-16
```

We can perform stepwise regression using the command:

```
step(null, scope = list(upper=full), data=model, direction="both")

## Start:  AIC=553357
## price ~ 1
##
##                   Df  Sum of Sq        RSS    AIC
## + sqft_living     1 1.4331e+15 1.4763e+15 538710
## + grade           1 1.2949e+15 1.6145e+15 540642
## + sqft_above      1 1.0668e+15 1.8426e+15 543496
## + sqft_living15   1 9.9621e+14 1.9132e+15 544308
## + bathrooms       1 8.0201e+14 2.1074e+15 546395
## + view            1 4.5933e+14 2.4501e+15 549649
## + bedrooms        1 2.7574e+14 2.6336e+15 551209
## <none>                         2.9094e+15 553357
##
## Step:  AIC=538710.3
## price ~ sqft_living
##
##                   Df  Sum of Sq        RSS    AIC
## + view            1 1.2348e+14 1.3528e+15 536826
## + grade           1 1.2104e+14 1.3553e+15 536865
## + bedrooms        1 4.0757e+13 1.4355e+15 538108
## + sqft_living15   1 2.0064e+13 1.4562e+15 538417
## + sqft_above      1 1.1897e+12 1.4751e+15 538695
## + bathrooms       1 1.4263e+11 1.4762e+15 538710
## <none>                         1.4763e+15 538710
## - sqft_living     1 1.4331e+15 2.9094e+15 553357
##
## Step:  AIC=536826.2
## price ~ sqft_living + view
##
##                   Df  Sum of Sq        RSS    AIC
## + grade           1 1.0823e+14 1.2446e+15 535028
## + bedrooms        1 2.7195e+13 1.3256e+15 536390
## + sqft_living15   1 1.1182e+13 1.3416e+15 536649
## + sqft_above      1 8.0050e+11 1.3520e+15 536815
## <none>                         1.3528e+15 536826
```

```
## + bathrooms      1 9.5085e+09 1.3528e+15 536828
## - view           1 1.2348e+14 1.4763e+15 538710
## - sqft_living    1 1.0972e+15 2.4501e+15 549649
##
## Step:  AIC=535027.6
## price ~ sqft_living + view + grade
##
##                   Df  Sum of Sq        RSS    AIC
## + bedrooms       1 1.3504e+13 1.2311e+15 534794
## + sqft_above     1 5.2224e+12 1.2394e+15 534939
## + bathrooms      1 4.7346e+12 1.2398e+15 534947
## <none>                        1.2446e+15 535028
## + sqft_living15  1 1.1774e+09 1.2446e+15 535030
## - grade          1 1.0823e+14 1.3528e+15 536826
## - view           1 1.1068e+14 1.3553e+15 536865
## - sqft_living    1 2.0616e+14 1.4507e+15 538335
##
## Step:  AIC=534794
## price ~ sqft_living + view + grade + bedrooms
##
##                   Df  Sum of Sq        RSS    AIC
## + sqft_above     1 6.1141e+12 1.2250e+15 534688
## + bathrooms      1 2.3101e+12 1.2288e+15 534755
## <none>                        1.2311e+15 534794
## + sqft_living15  1 3.7432e+09 1.2311e+15 534796
## - bedrooms       1 1.3504e+13 1.2446e+15 535028
## - grade          1 9.4543e+13 1.3256e+15 536390
## - view           1 1.0195e+14 1.3330e+15 536510
## - sqft_living    1 2.0217e+14 1.4333e+15 538075
##
## Step:  AIC=534688.5
## price ~ sqft_living + view + grade + bedrooms + sqft_above
##
##                   Df  Sum of Sq        RSS    AIC
## + bathrooms      1 2.1963e+12 1.2228e+15 534652
## + sqft_living15  1 1.1923e+11 1.2248e+15 534688
## <none>                        1.2250e+15 534688
## - sqft_above     1 6.1141e+12 1.2311e+15 534794
## - bedrooms       1 1.4396e+13 1.2394e+15 534939
## - view           1 8.7876e+13 1.3128e+15 536182
## - grade          1 1.0049e+14 1.3255e+15 536389
## - sqft_living    1 1.4682e+14 1.3718e+15 537131
##
## Step:  AIC=534651.7
## price ~ sqft_living + view + grade + bedrooms + sqft_above +
##     bathrooms
##
##                   Df  Sum of Sq        RSS    AIC
```

```
## <none>                        1.2228e+15 534652
## + sqft_living15  1 5.5659e+10 1.2227e+15 534653
## - bathrooms      1 2.1963e+12 1.2250e+15 534688
## - sqft_above     1 6.0004e+12 1.2288e+15 534755
## - bedrooms       1 1.1915e+13 1.2347e+15 534859
## - view           1 8.6849e+13 1.3096e+15 536131
## - grade          1 1.0197e+14 1.3247e+15 536379
## - sqft_living    1 1.4461e+14 1.3674e+15 537063
##
## Call:
## lm(formula = price ~ sqft_living + view + grade + bedrooms +
##     sqft_above + bathrooms, data = model)
##
## Coefficients:
## (Intercept)  sqft_living         view        grade     bedrooms   sqft_above
##  -480546.60       230.72     88853.10     98179.33    -32070.20       -44.56
##   bathrooms
##   -20812.10
```

According to this procedure, the best model is the one that includes the variables sqft_living, view, grade, bedrooms, sqft_above, bathrooms.

### 1.3.5 Conclusion

The most suitable model used for prediction is the one has the following regressors: sqft_living, view, grade, bedrooms, sqft_above, bathrooms. Hence, those variables are factors which may affect home price.
The linear model used for upcoming prediction:

```
finalmodel = lm(formula = price ~ sqft_living + view + grade + bedrooms + sqft_above + bathrooms,
summary(finalmodel)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + view + grade + bedrooms +
##     sqft_above + bathrooms, data = model)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1266314  -124874   -19527    95430  4598306
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.805e+05  1.460e+04 -32.924  < 2e-16 ***
## sqft_living  2.307e+02  4.566e+00  50.525  < 2e-16 ***
## view         8.885e+04  2.269e+03  39.156  < 2e-16 ***
## grade        9.818e+04  2.314e+03  42.429  < 2e-16 ***
## bedrooms    -3.207e+04  2.211e+03 -14.503  < 2e-16 ***
## sqft_above  -4.456e+01  4.330e+00 -10.292  < 2e-16 ***
```

```
## bathrooms   -2.081e+04  3.342e+03  -6.227 4.85e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 238000 on 21586 degrees of freedom
## Multiple R-squared:  0.5797,Adjusted R-squared:  0.5796
## F-statistic:  4962 on 6 and 21586 DF,  p-value: < 2.2e-16
```

Finally, our model equation can be written as follow:

```
predicted_price = 230.7*sqft_living + 88850*view + 98180*grade - 32070*
bedrooms - 44.56*sqft_above - 20810*bathrooms -480500
```

## 1.4 Predictions

### 1.4.1 Case 1

sqft_living15 = mean(sqft_living15), sqft_above = mean(sqft_above), sqft_living = mean(sqft_living), floor = 2, condition = 3.

```
predicted_price = (230.7*mean(house$sqft_living)
    + 88850*mean(house$view) + 98180*mean(house$grade)
    - 32070*mean(house$bedrooms) - 44.56*mean(house$sqft_above)
    - 20810*mean(house$bathrooms) - 480500)
predicted_price

## [1] 540088
```

The prediction for the price is: 540088

### 1.4.2 Case 2

sqft_living15 = max(sqft_living15), sqft_above = max(sqft_above), sqft_living = max(sqft_living), floor = 2, condition = 3.

```
predicted_price = (230.7*max(house$sqft_living)
    + 88850*max(house$view) + 98180*max(house$grade)
    - 32070*max(house$bedrooms) - 44.56*max(house$sqft_above)
    - 20810*max(house$bathrooms) - 480500)
predicted_price

## [1] 2630818
```

The prediction for the price is: 2630818

#### Evaluate a model using RMSE

Root Mean Square Error is one of the most popular metrics to evaluate a model. The smaller it is, the better predictions the model can produce. In addition, if the RMSE/mean is smaller than 1, that means the model has done a great job at predicting.

```
p = predict(finalmodel, house)
RMSE(p, house$price)/mean(house$price)

## [1] 0.4406235
```

# 2 Activity 2

This classic dataset contains the prices and other attributes of almost 54,000 diamonds.
**Content**

- price price in US dollars ($326-$18,823)
- carat weight of the diamond (0.2-5.01)
- cut quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- color diamond colour, from J (worst) to D (best)
- clarity a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))
- x length in mm (0-10.74)
- y width in mm (0-58.9)
- z depth in mm (0-31.8)
- depth total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43-79)
- table width of top of diamond relative to widest point (43-95)

The task is to Perform EDA and predict price of diamonds.

## 2.1 Data Set Preparation (Import data)

First, we need to load some necessary packages. We can simply install package "tidyverse" which includes all packages used for data manipulation and visualization.

```
library(tidyverse)

## - Attaching packages -------------------- tidyverse 1.3.1 -
## v tibble  3.1.2     v purrr   0.3.4
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## - Conflicts --------------------- tidyverse_conflicts() -
## x lubridate::as.difftime() masks base::as.difftime()
## x gridExtra::combine()     masks dplyr::combine()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x MASS::select()           masks dplyr::select()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()

library(psych)

##
## Attaching package: 'psych'
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

The data set is imported from file "diamonds.csv" to acquired the data frame "diamonds".

```
diamonds <- read.csv('diamonds.csv')
head(diamonds)   # To show the first 6 rows

##   X carat       cut color clarity depth table price    x    y    z
## 1 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

## 2.2 Data Cleaning

The first column of the data set is the ID numbers, which is not affect the diamonds' price, thus we will remove it.

```
diamonds <- diamonds[c(-1)]
head(diamonds)

##   carat       cut color clarity depth table price    x    y    z
## 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

Then, we count the number of NA values in the data set.

```
sum(is.na(diamonds))

## [1] 0
```

Since there is no NA value, we will move onto the next section, data visualization.

## 2.3 Data Visualization

### 2.3.1 Descriptive statistics for each of the variables

Let it be fast and simple, we use function `summary()` to summarize the descriptive statistics for the data.

```
dim(diamonds) # dimention of the data frame

## [1] 53940    10

str(diamonds) # structure

## 'data.frame': 53940 obs. of  10 variables:
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : chr  "Ideal" "Premium" "Good" "Premium" ...
##  $ color  : chr  "E" "E" "E" "I" ...
##  $ clarity: chr  "SI2" "SI1" "VS1" "VS2" ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...

summary(diamonds) # statistical summary

##      carat             cut               color             clarity
##  Min.   :0.2000   Length:53940       Length:53940       Length:53940
##  1st Qu.:0.4000   Class :character   Class :character   Class :character
##  Median :0.7000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.7979
```

```
##  3rd Qu.:1.0400
##  Max.   :5.0100
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##        y                z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
```

```r
length(diamonds) # no. columns in the data-set
```

```
## [1] 10
```

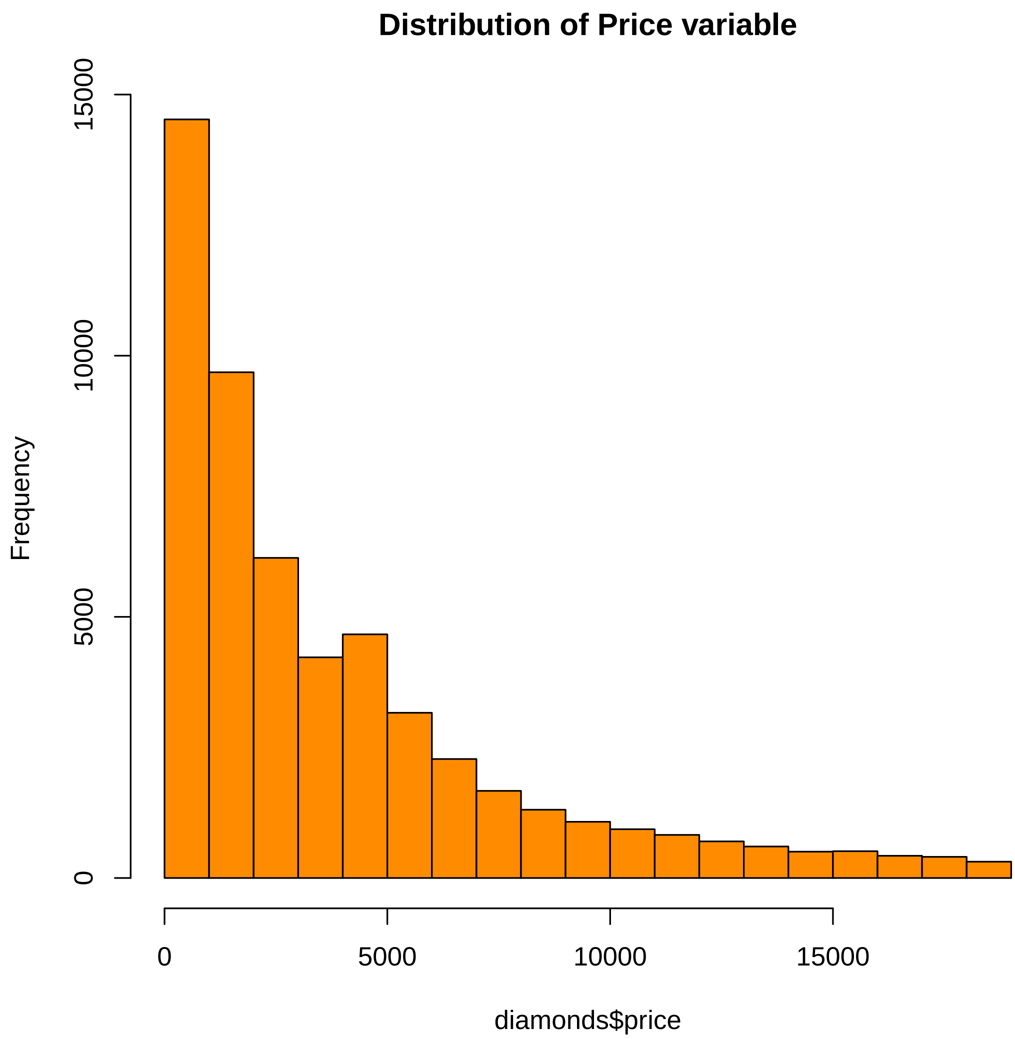
```r
colnames(diamonds) # name of columns
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

### 2.3.2 Histogram, Box plot and pairs

First of all, let's explore the distribution of our dependent variable price.

```r
hist(diamonds$price, main = "Distribution of Price variable", col = "darkorange")
```




Because the mean is greater than the median the distribution is right-skewed.

#### 2.3.2.a Definition

A **histogram** represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.
In R, the function `hist()`, `barplot()` are used to illustrate the plot.

**Boxplots** are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing box plots for each of them.

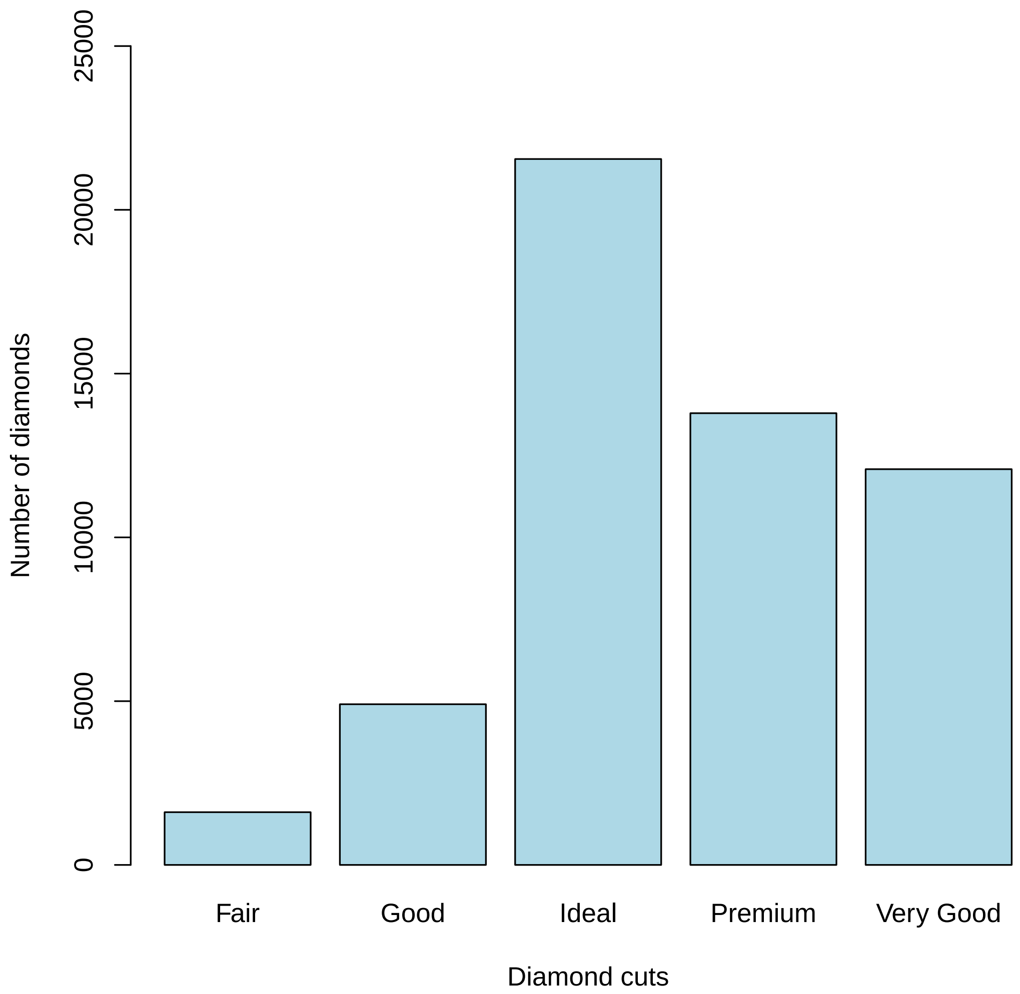
Box plots are created in R by using the `boxplot()` function, or `ggplot()` with `geom_boxplot()`.

The R function "pairs" returns a plot matrix, consisting of **scatter plots** for each variable-combination of a data frame. The basic R syntax for the pairs command is shown as follows.

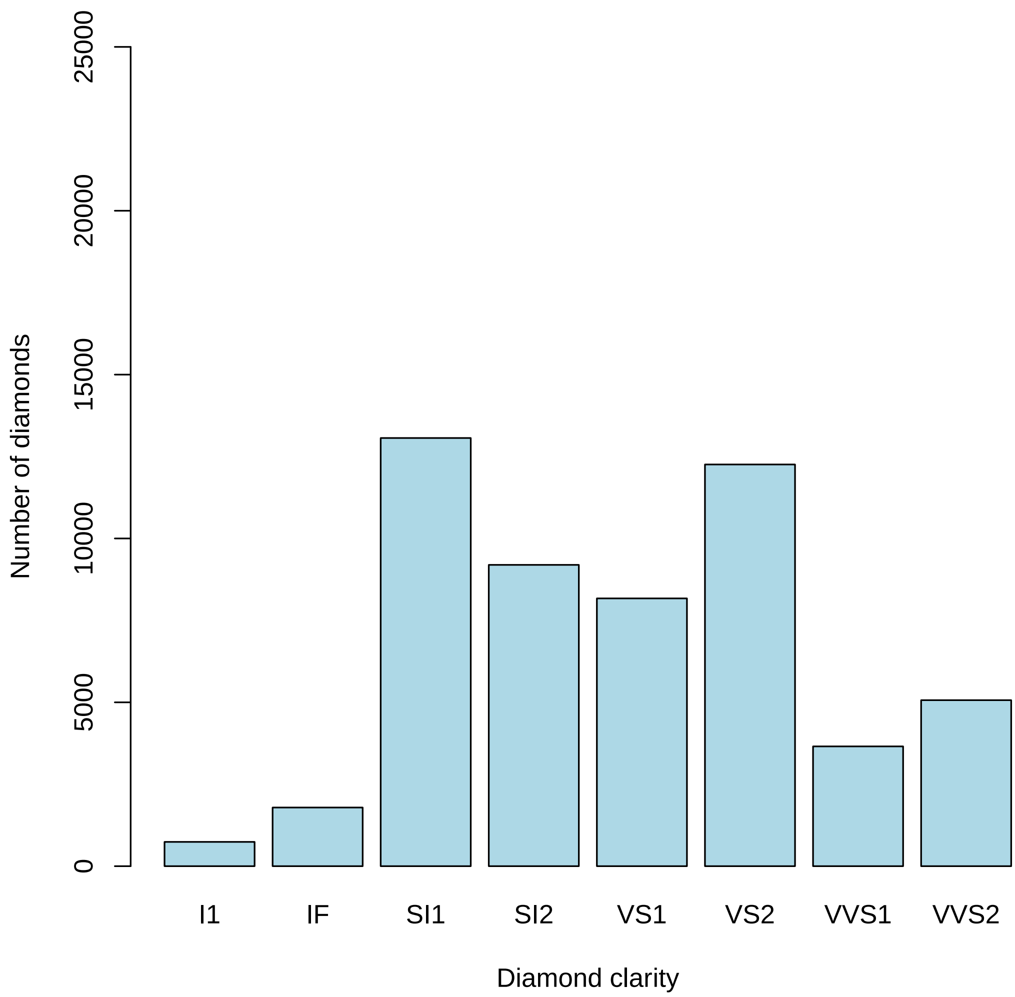
#### 2.3.2.b Factors variables

Some features (cut, color, clarity) are now **factors**. Let's have a look at how they are distributed.
Histogram:

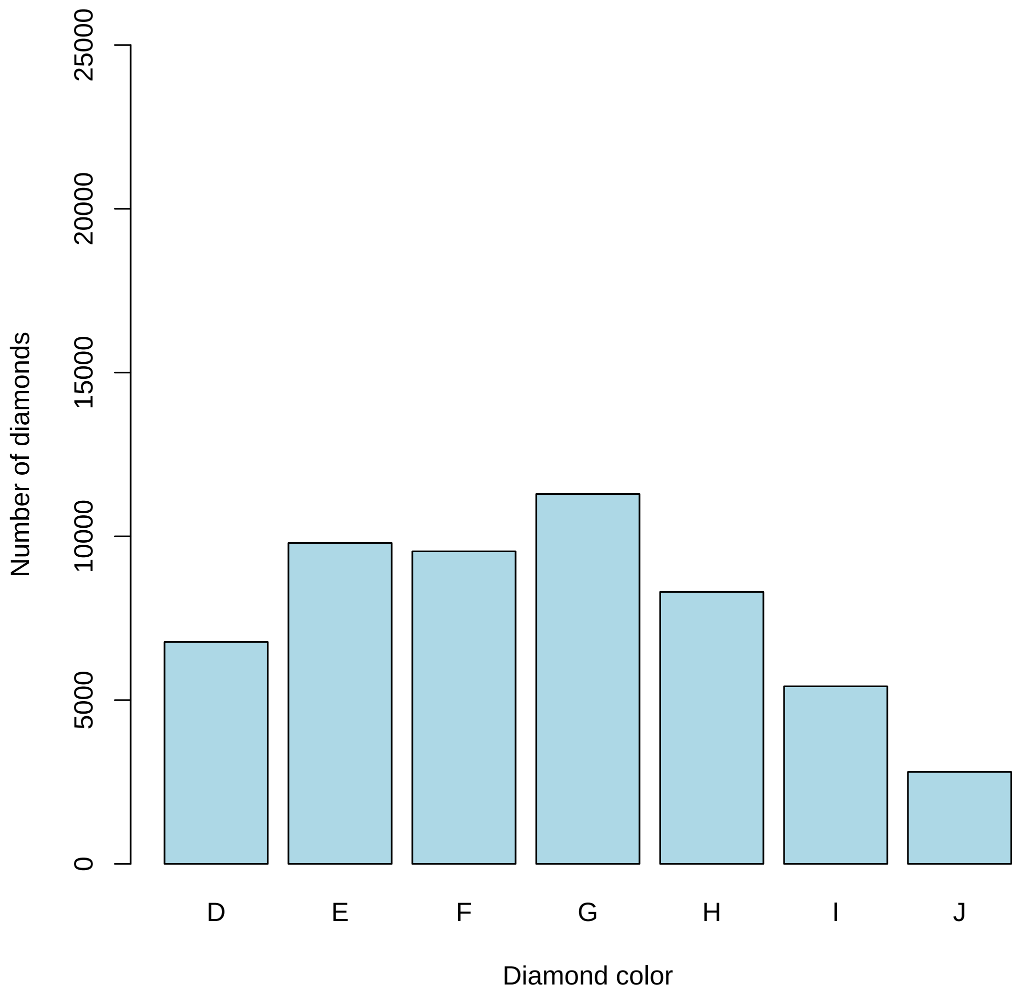
```
barplot(table(diamonds$cut),col="light blue",ylim=c(0,25000),ylab="Number of diamonds", xlab="Dia
```



```
barplot(table(diamonds$clarity),col="light blue",ylim=c(0,25000),ylab="Number of diamonds", xlab=
```

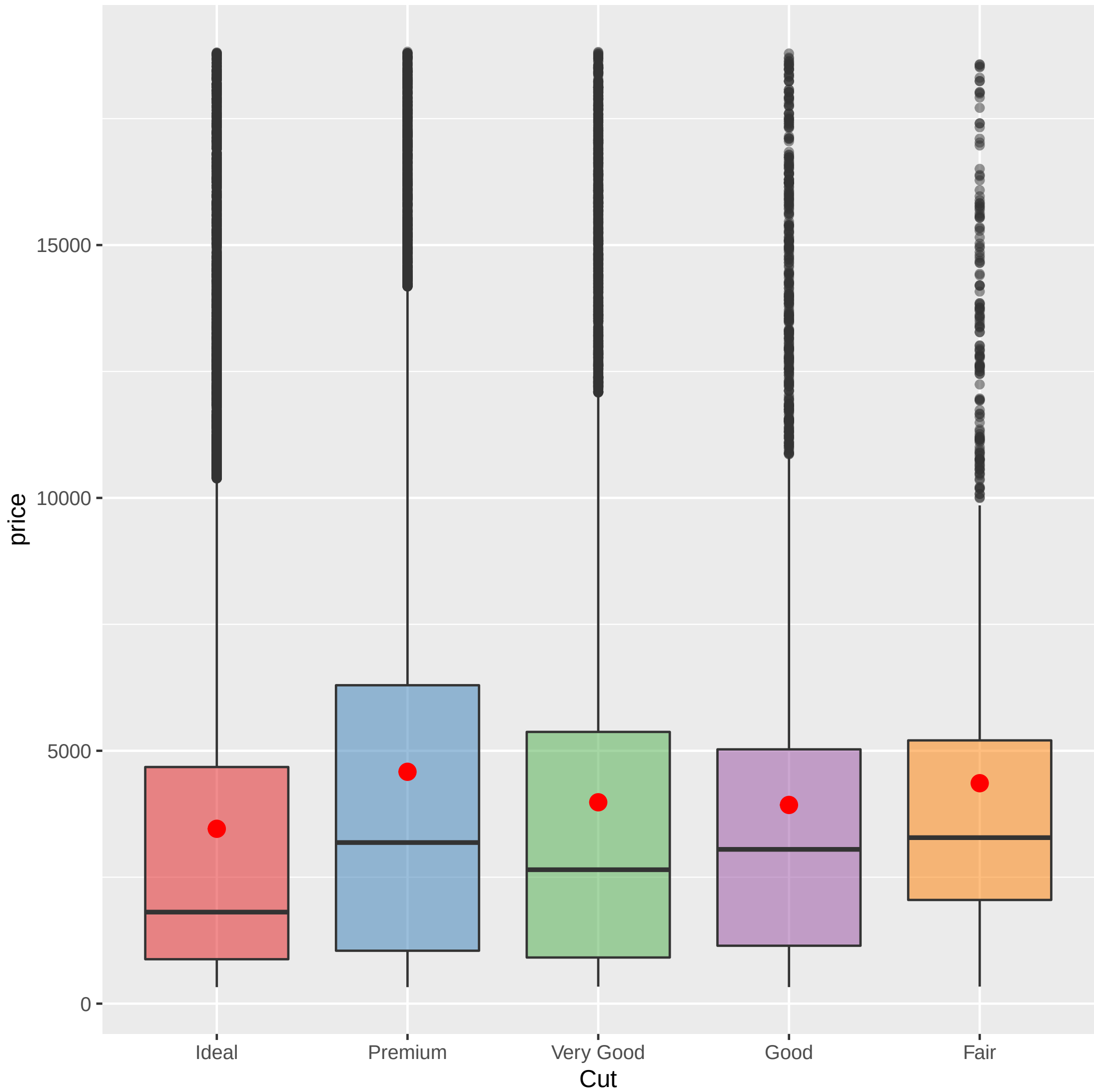



```
barplot(table(diamonds$color),col="light blue",ylim=c(0,25000),ylab="Number of diamonds", xlab="D
```

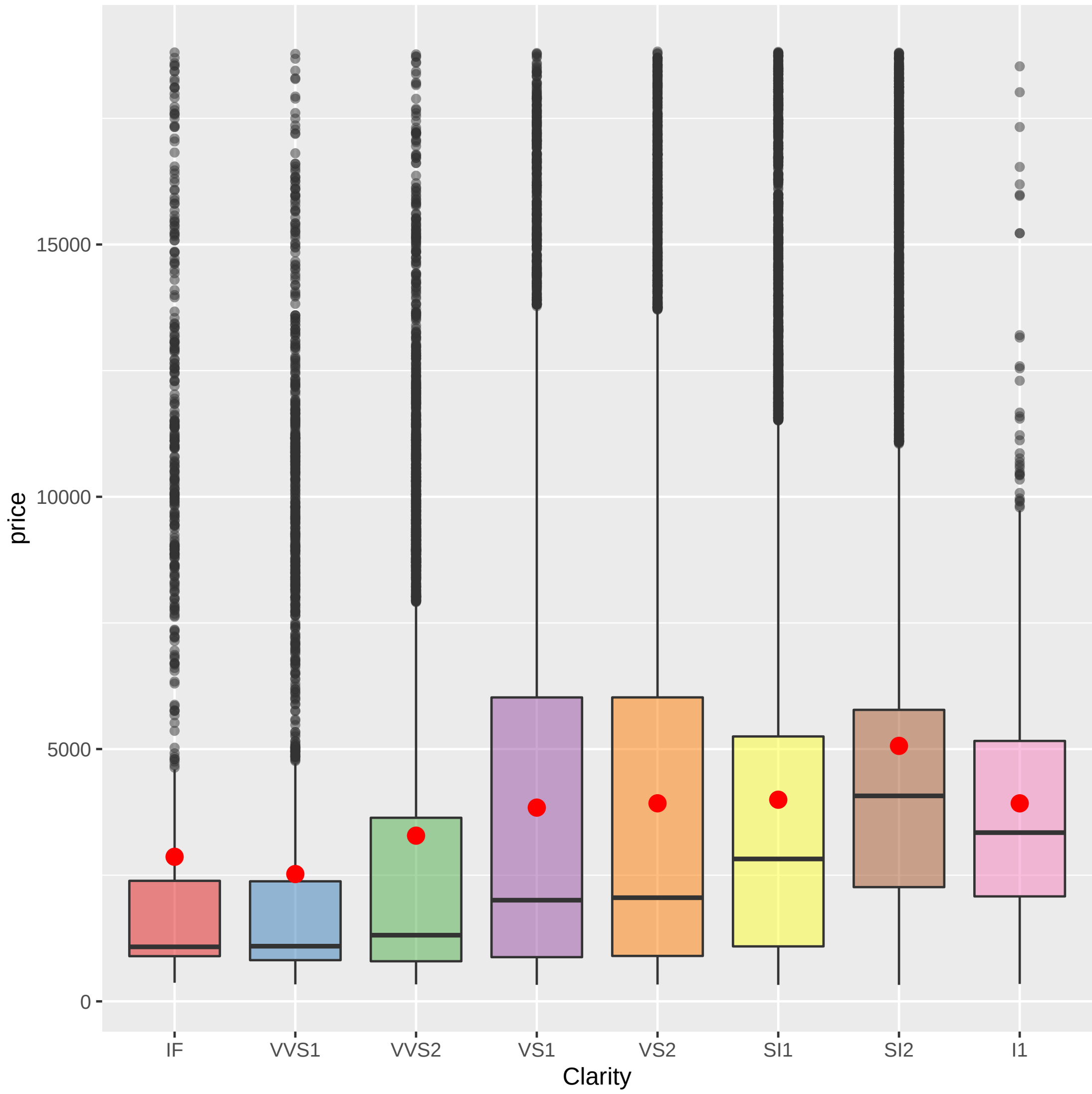



Boxplots:

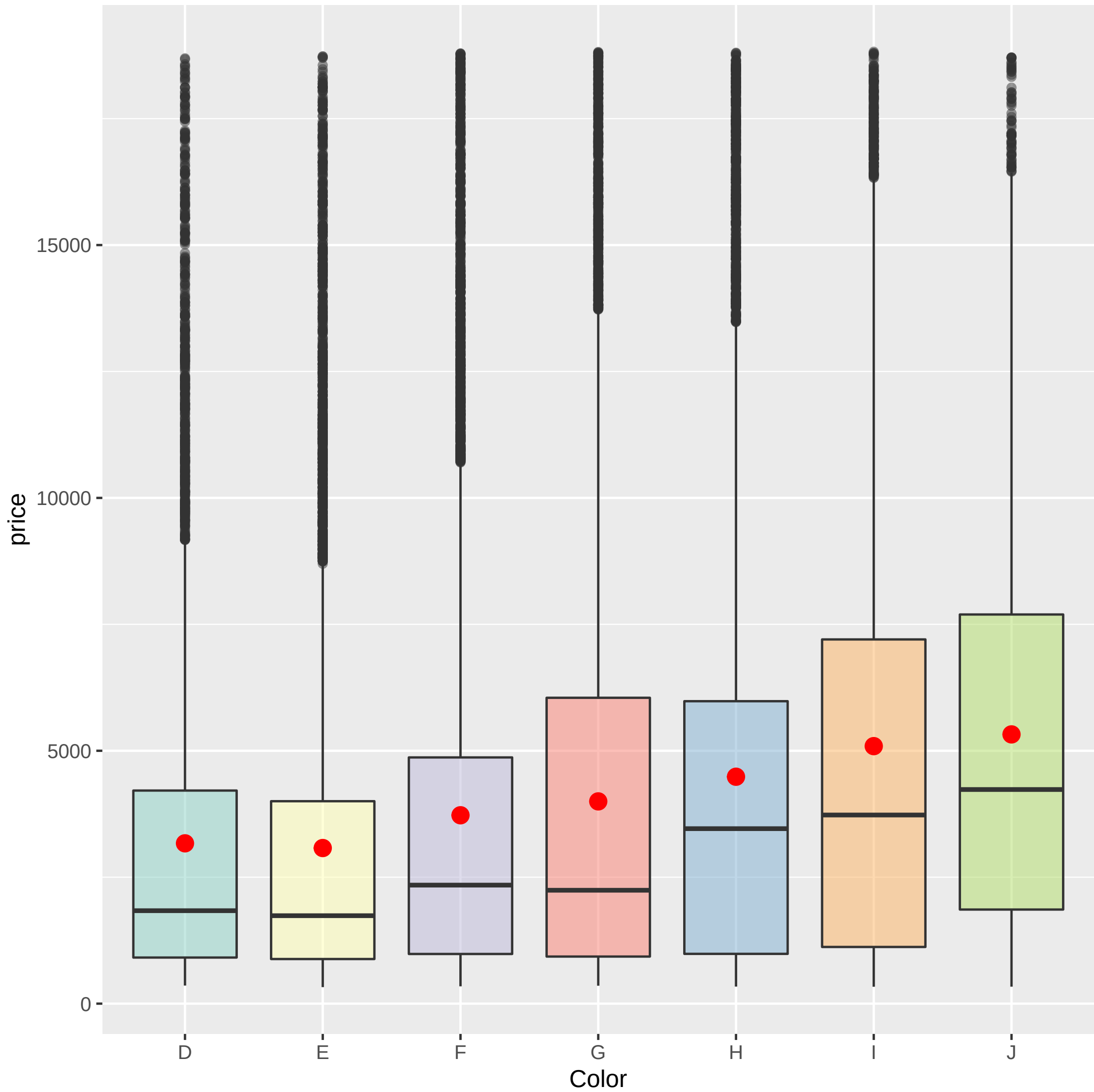
```
ggplot(diamonds, aes(x=Cut, y=price, fill=Cut)) +
geom_boxplot(alpha=0.5) +
stat_summary(fun=mean, geom="point", shape=20, size=5, color="red", fill="red") +
theme(legend.position="none") +
scale_fill_brewer(palette="Set1")
```




```
ggplot(diamonds, aes(x=Clarity, y=price, fill=Clarity)) +
geom_boxplot(alpha=0.5) +
stat_summary(fun=mean, geom="point", shape=20, size=5, color="red", fill="red") +
theme(legend.position="none") +
scale_fill_brewer(palette="Set1")
```




```
ggplot(diamonds, aes(x=Color, y=price, fill=Color)) +
geom_boxplot(alpha=0.5) +
stat_summary(fun=mean, geom="point", shape=20, size=5, color="red", fill="red") +
theme(legend.position="none") +
scale_fill_brewer(palette="Set3")
```

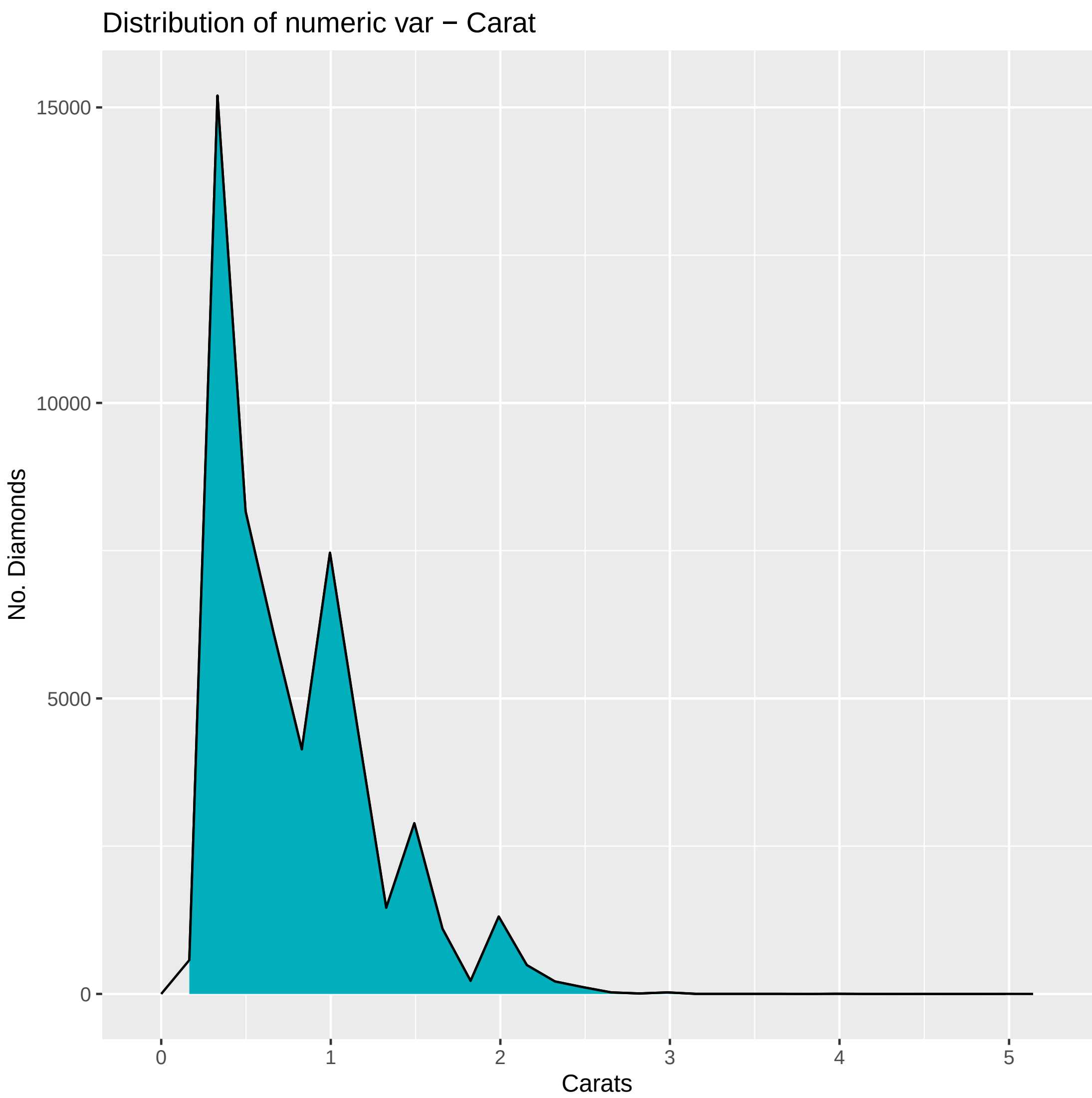



#### 2.3.2.c Numeric variables

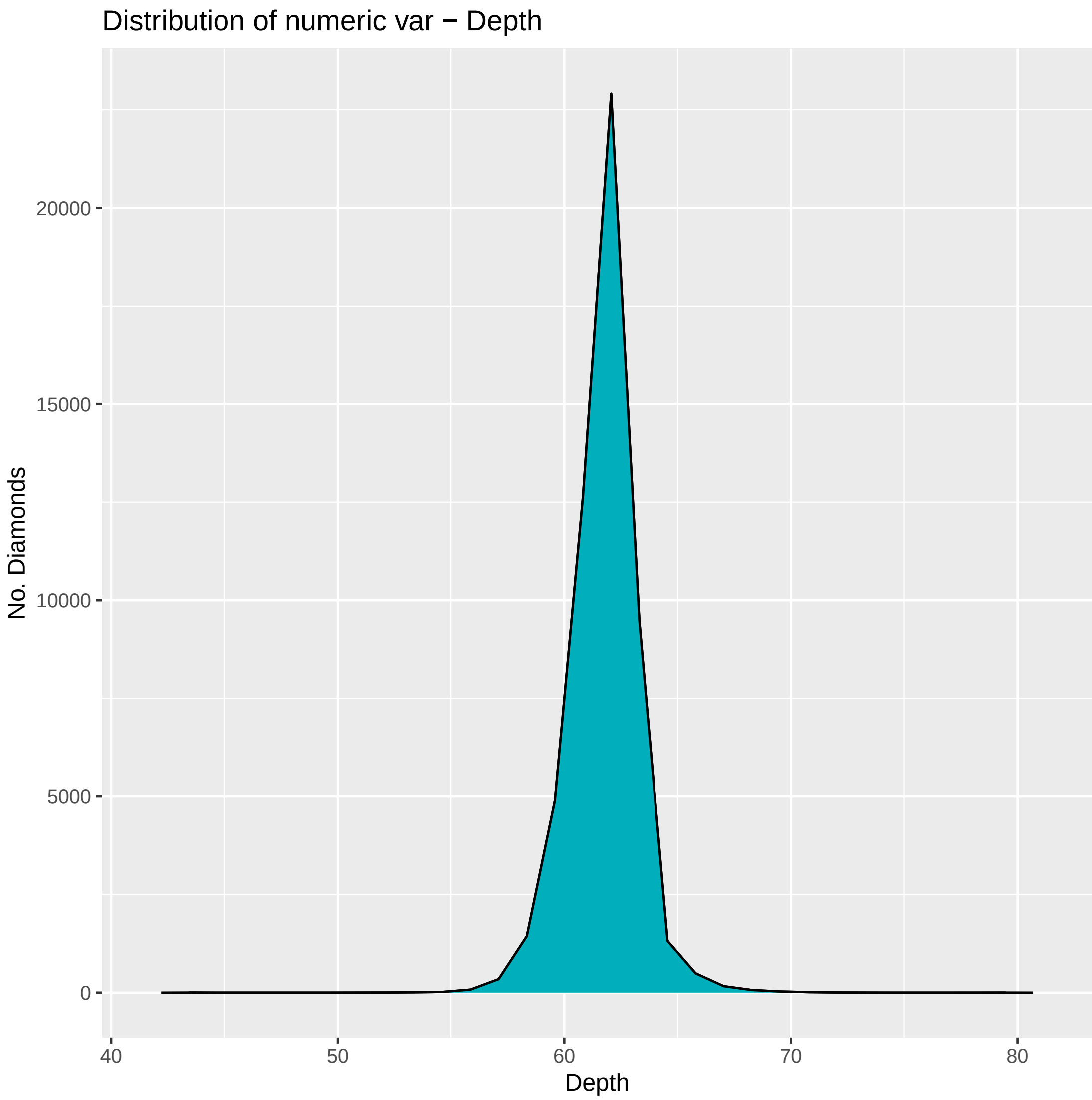
Carat, Depth, Table, x, y, z, are all numeric variables.
Histogram:

```
ggplot(diamonds,aes(x=carat))+
geom_freqpoly()+
geom_area(stat = "bin", color = "black", fill = "#00AFBB")+
xlab('Carats')+
ylab('No. Diamonds')+
ggtitle('Distribution of numeric var - Carat')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

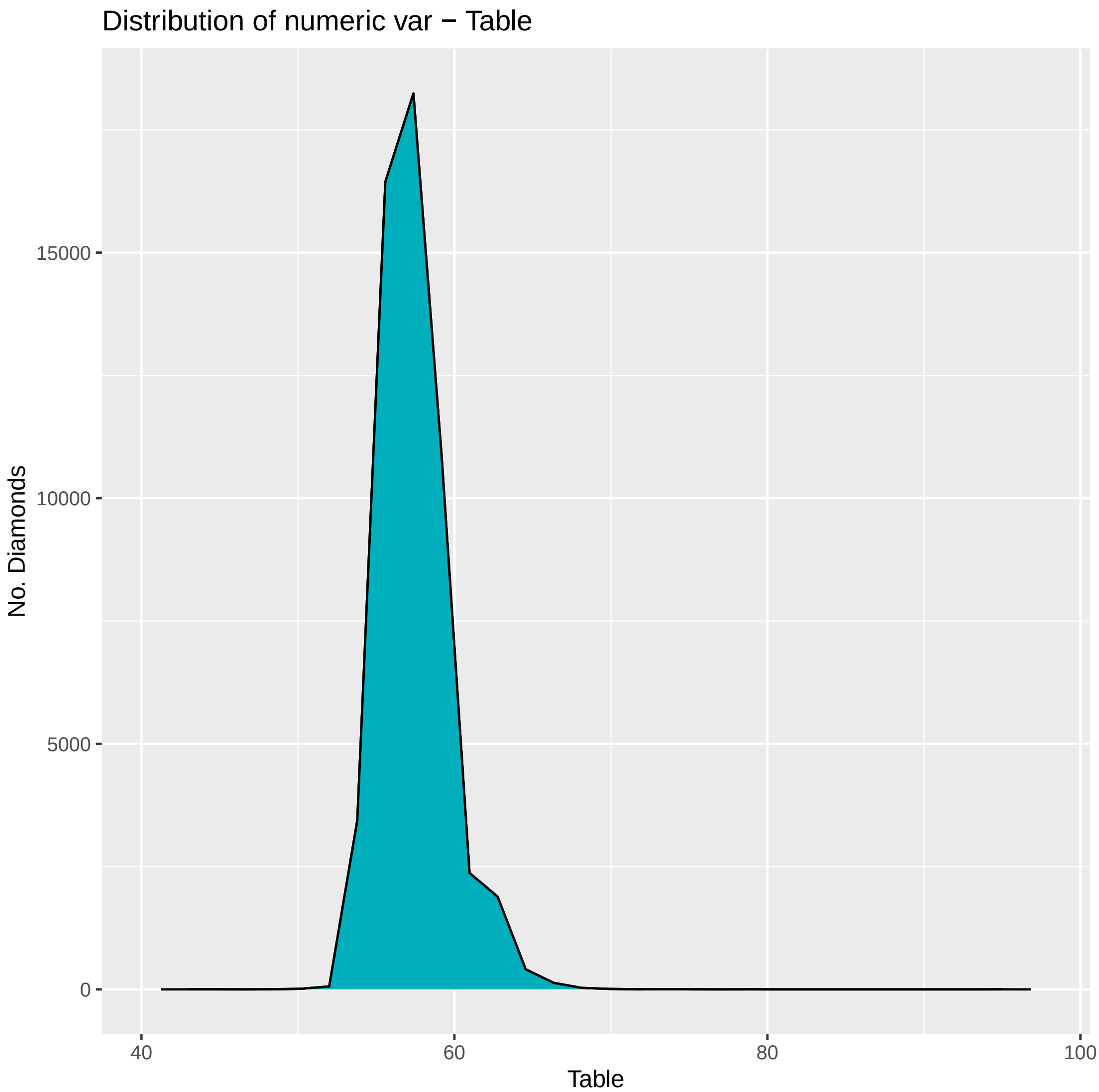
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



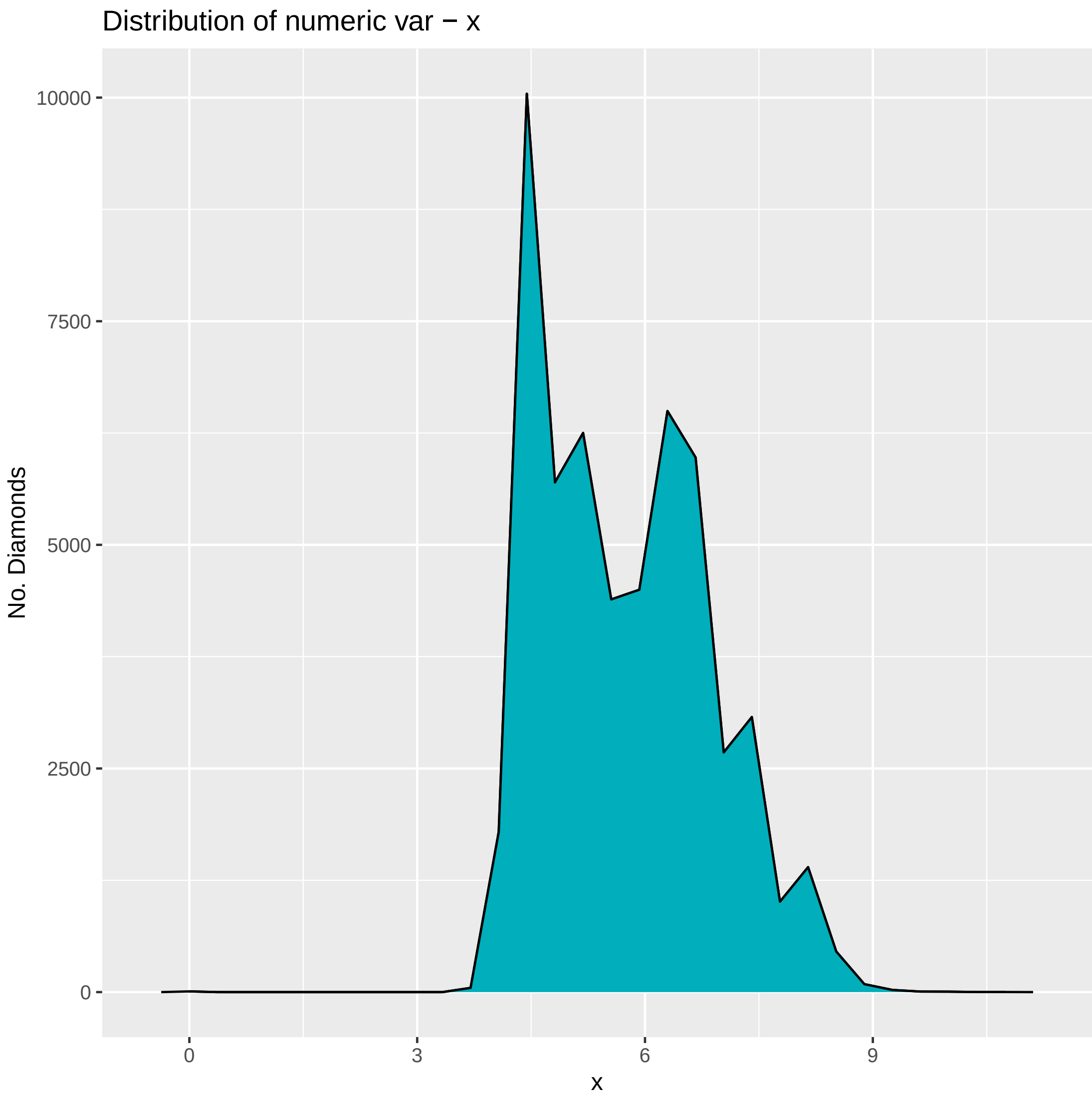

```
ggplot(diamonds,aes(x=depth))+
geom_freqpoly()+
geom_area(stat = "bin", color = "black", fill = "#00AFBB")+
xlab('Depth')+
ylab('No. Diamonds')+
ggtitle('Distribution of numeric var - Depth')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



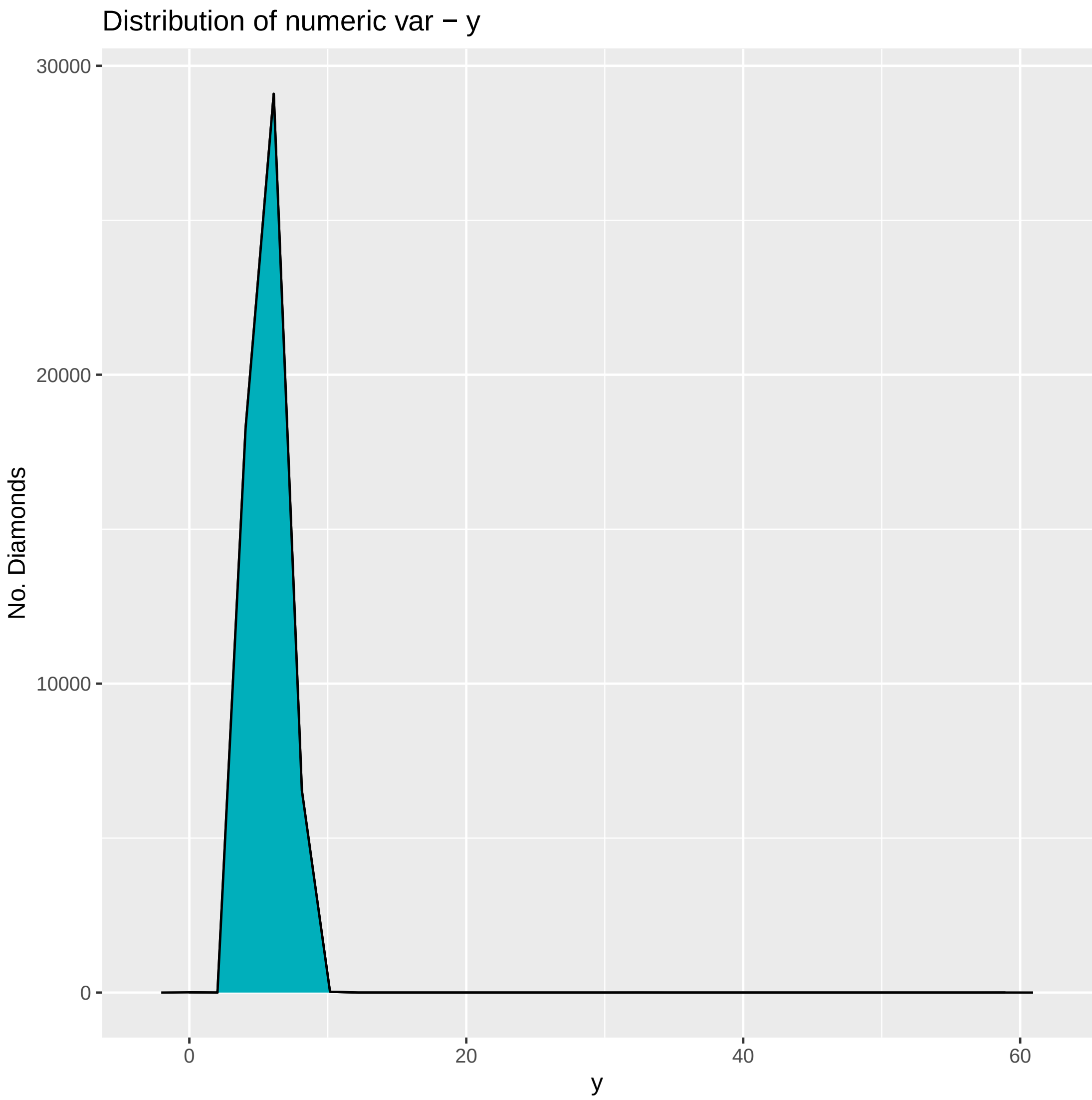

```r
ggplot(diamonds,aes(x=table))+
geom_freqpoly()+
geom_area(stat = "bin", color = "black", fill = "#00AFBB")+
xlab('Table')+
ylab('No. Diamonds')+
ggtitle('Distribution of numeric var - Table')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



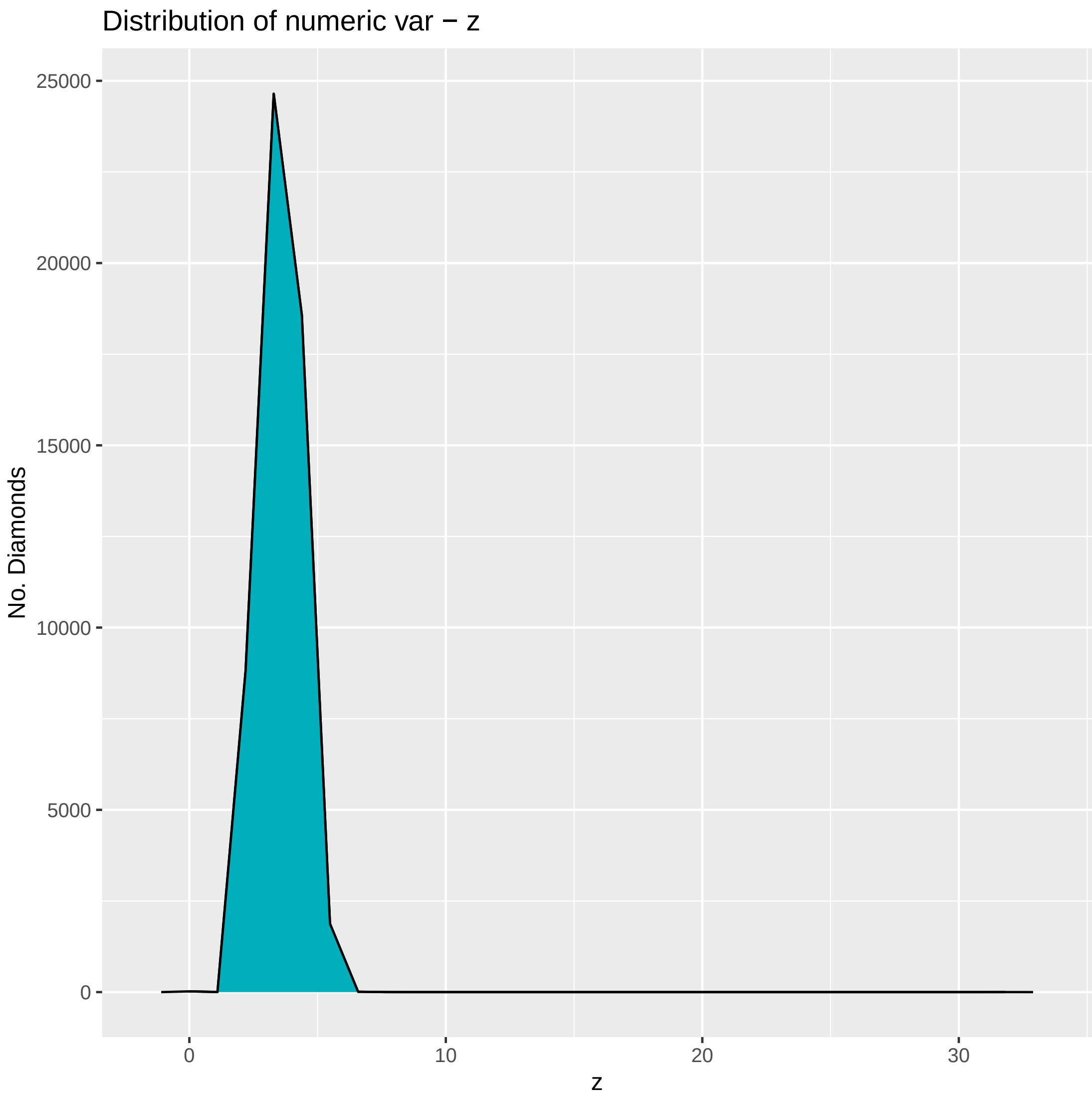

```r
ggplot(diamonds,aes(x=x))+
geom_freqpoly()+
geom_area(stat = "bin", color = "black", fill = "#00AFBB")+
xlab('x')+
ylab('No. Diamonds')+
ggtitle('Distribution of numeric var - x')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
ggplot(diamonds,aes(x=y))+
geom_freqpoly()+
geom_area(stat = "bin", color = "black", fill = "#00AFBB")+
xlab('y')+
ylab('No. Diamonds')+
ggtitle('Distribution of numeric var - y')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

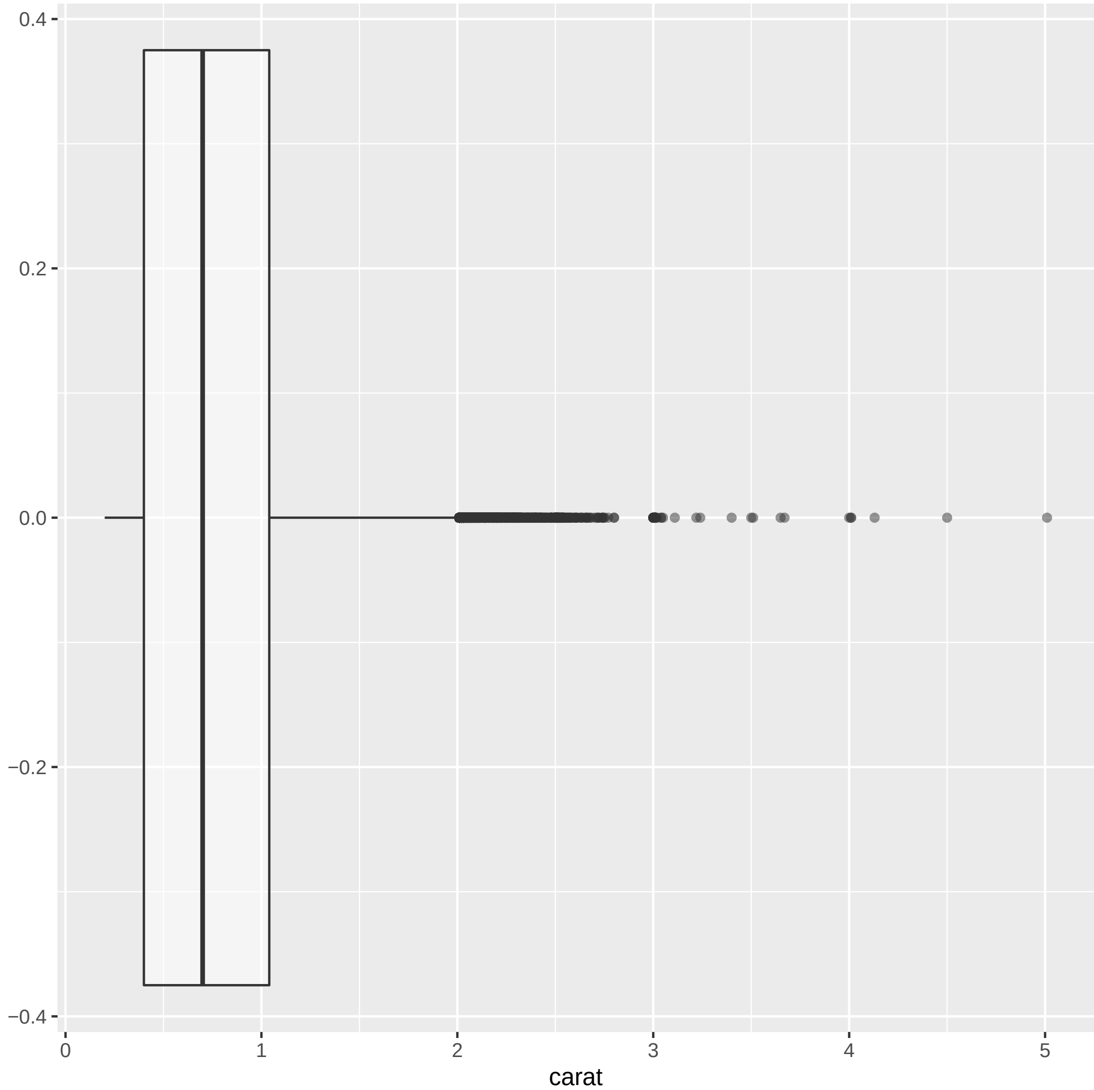



```
ggplot(diamonds,aes(x=z))+
geom_freqpoly()+
geom_area(stat = "bin", color = "black", fill = "#00AFBB")+
xlab('z')+
ylab('No. Diamonds')+
ggtitle('Distribution of numeric var - z')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

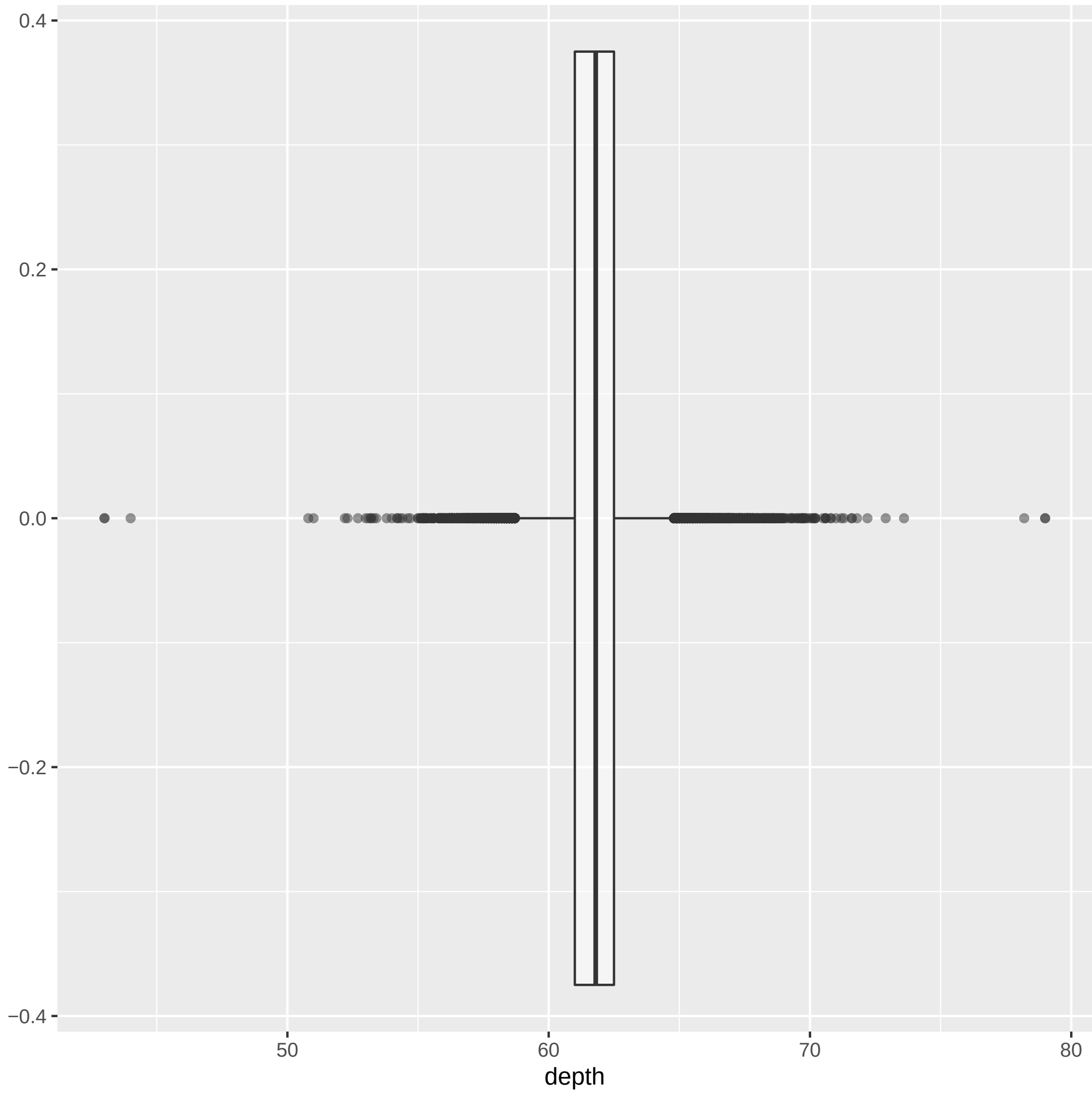



Boxplots:

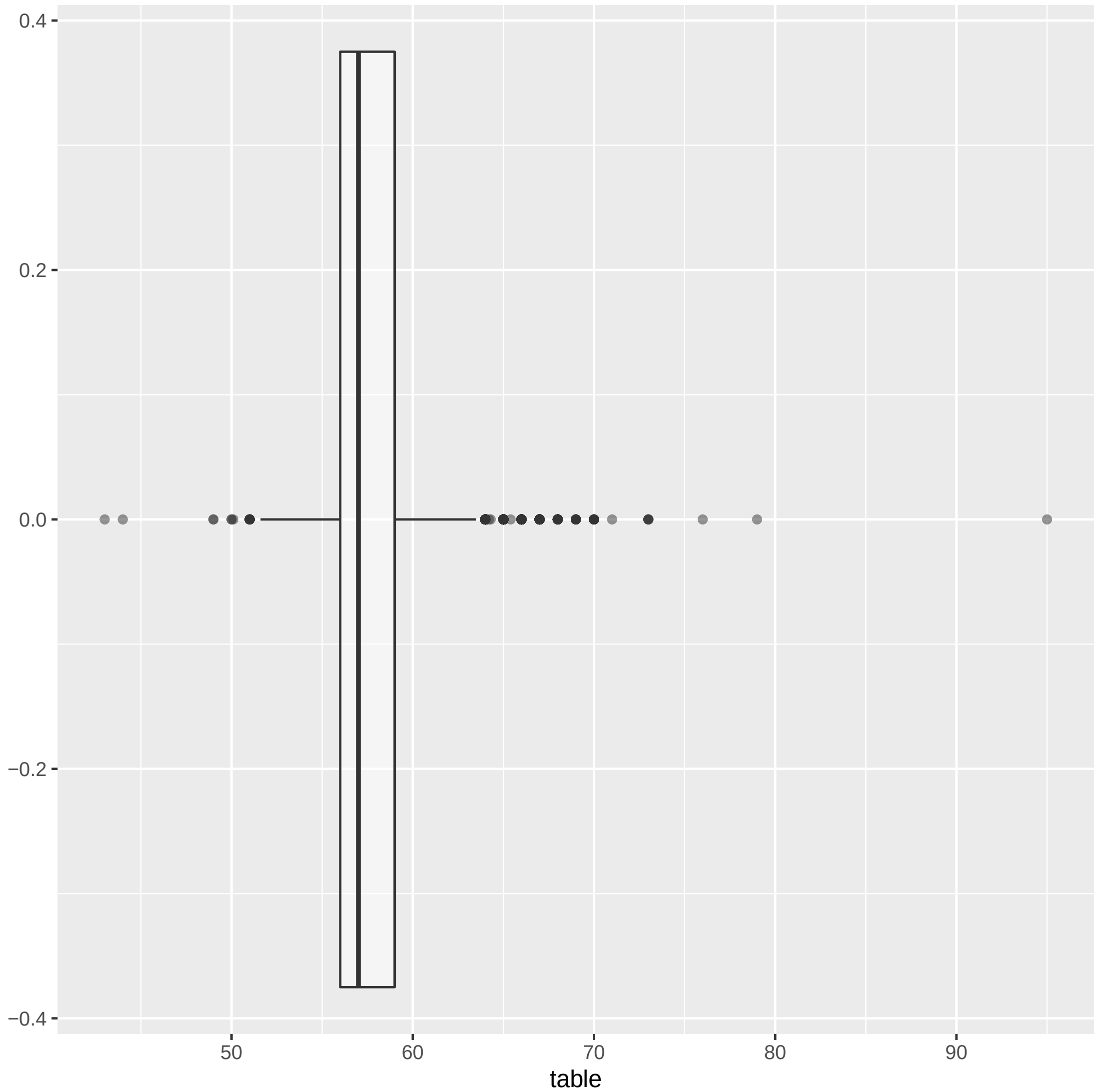
```
ggplot(diamonds, aes(x=carat)) +
geom_boxplot(alpha=0.5) +
theme(legend.position="none")
```



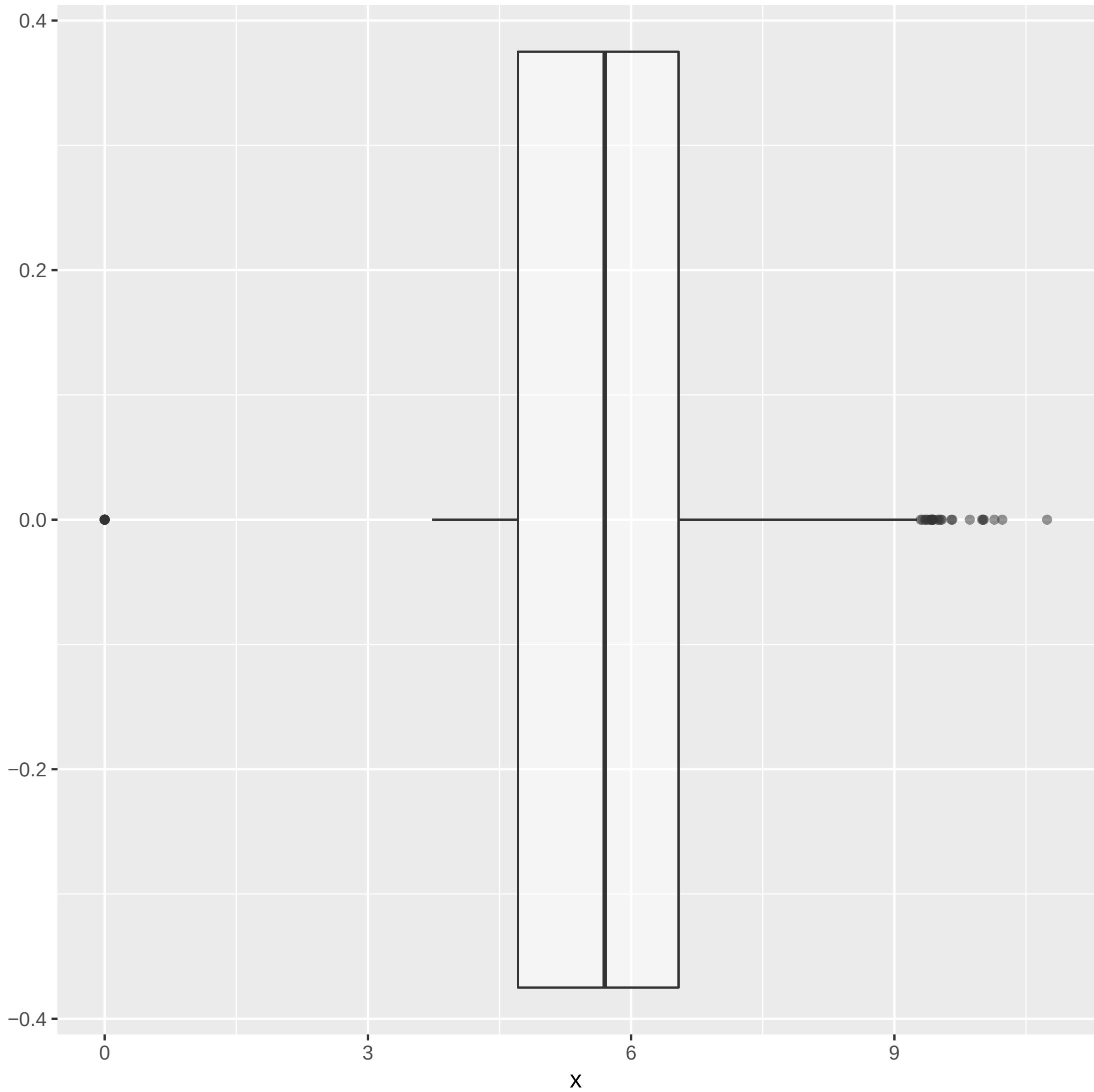

```
ggplot(diamonds, aes(x=depth)) +
geom_boxplot(alpha=0.5) +
theme(legend.position="none")
```



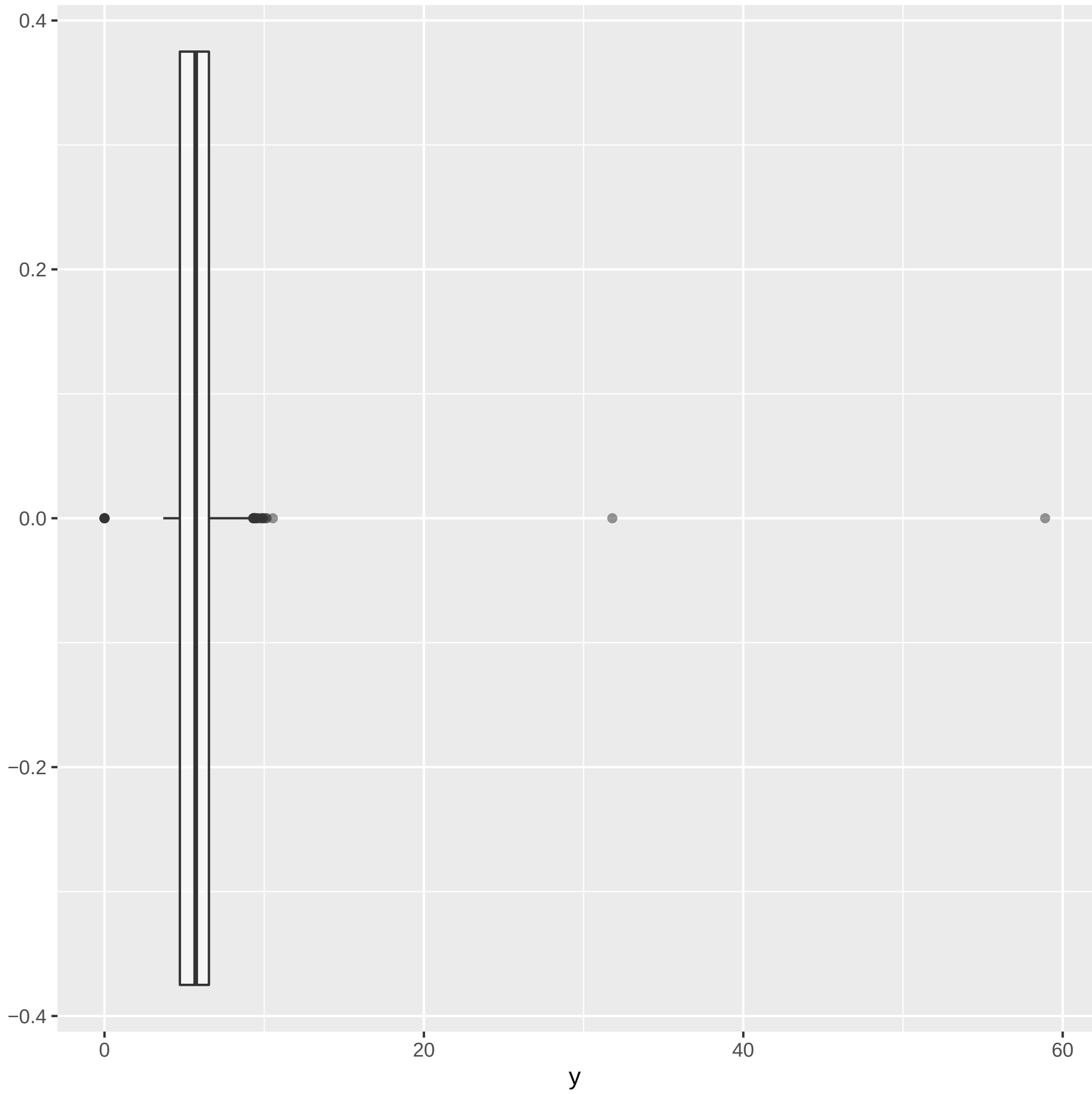

```
ggplot(diamonds, aes(x=table)) +
geom_boxplot(alpha=0.5) +
theme(legend.position="none")
```



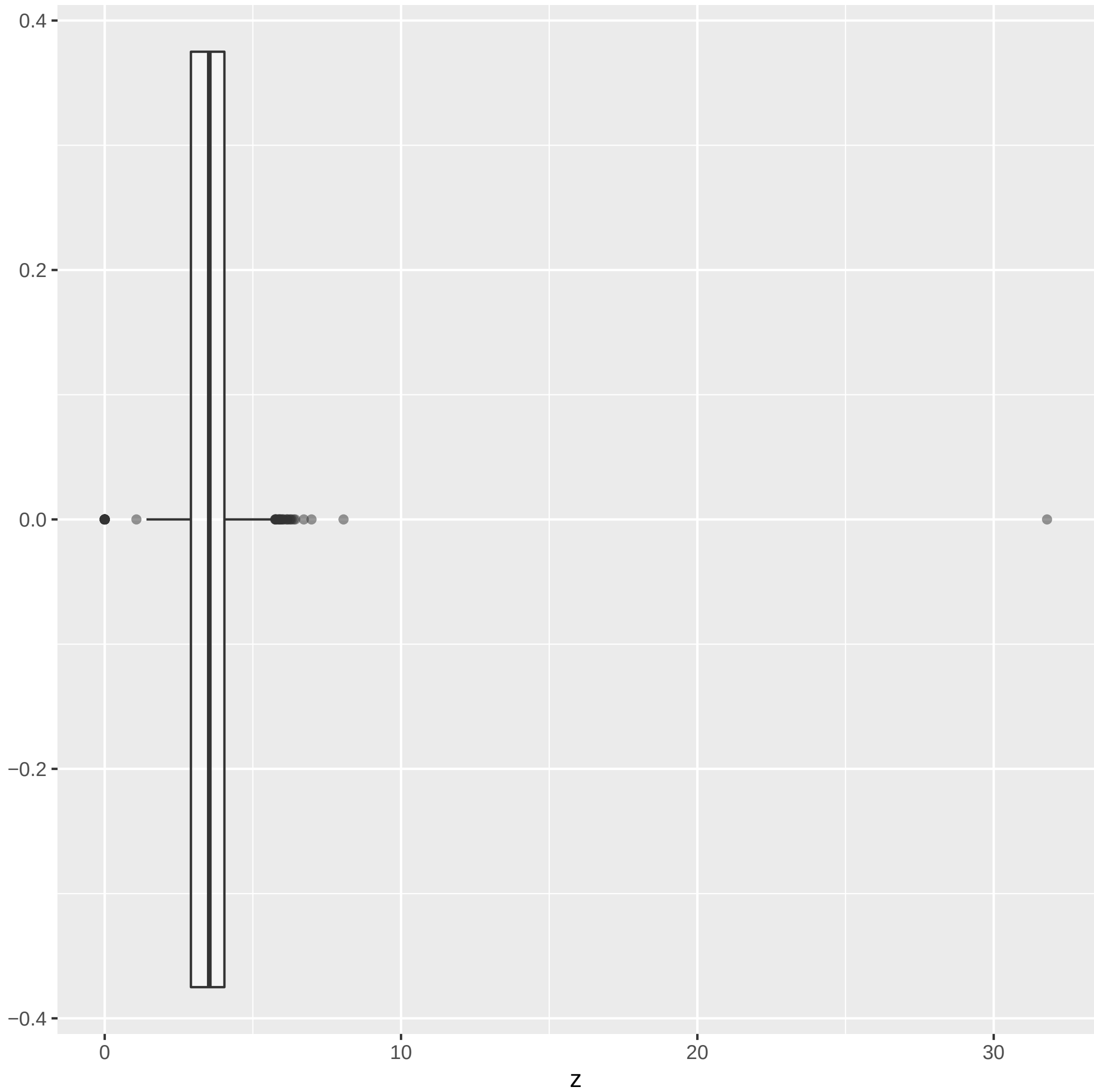

```
ggplot(diamonds, aes(x=x)) +
geom_boxplot(alpha=0.5) +
theme(legend.position="none")
```




```r
ggplot(diamonds, aes(x=y)) +
geom_boxplot(alpha=0.5) +
theme(legend.position="none")
```

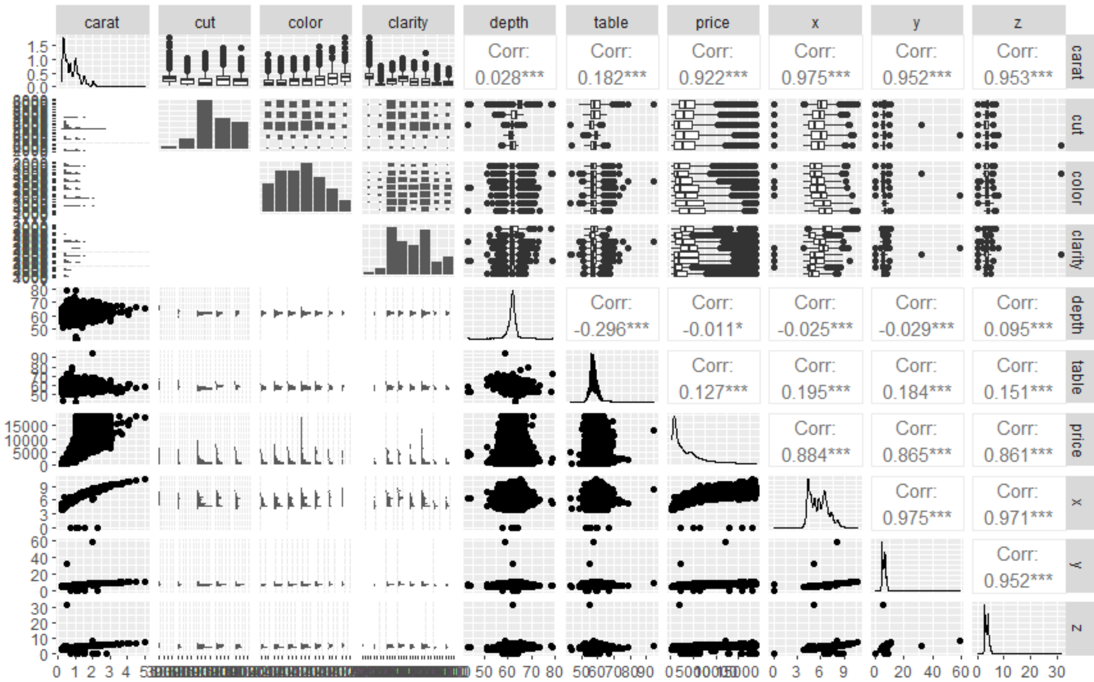



```
ggplot(diamonds, aes(x=z)) +
geom_boxplot(alpha=0.5) +
theme(legend.position="none")
```


0.4
0.2
0.0
−0.2
−0.4
0
10
20
30
z

**Pairs:**

```r
library(GGally)
ggpairs(diamonds[,1:10])
```




### 2.3.3 Remark

#### 2.3.3.a Data set

There are a total of 53,940 diamonds in the dataset with 10 features (carat, cut, colour, clarity, depth, table, price, x, y, and z)[4]. The variables cut, colour, and clarity, are ordered factor variables with the following levels. (Worst) ——————————> (best)

- **cut**: Fair, Good, Very Good, Premium, Ideal
- **colour**: J, I, H, G, F, E, D
- **clarity**: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF

Other observations include:
The ideal cut is the most prevalent one.
The maximum carat weight is 5.01.
The G color makes up the highest proportion on the color board.
The max price is $18,823.

#### 2.3.3.b Result

Looking at the chart, it is easy to see that carat, color, cut, clarity, depth, and table affect the price of a diamond. In addition, it also shows that carat and clarity are two of the factors that determine price volatility.

The graphs give an overview of particular patterns visualized in box plots and histograms. Generally, diamonds with better clarity, color, and cut cost less than diamonds with poorer characteristics. A good example of this is the price/carat vs clarity plot, let's first sort the clarity properties in order from worst to best ( I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF). It is clear that the diamond with the least clarity (I1) has the lowest price, this number grows up to a peak at VS1 and then declines as the clarity reaches maximum.

Another issue that deserves attention is that diamonds with attractive colors have a higher selling price, however, due to the smaller number of diamonds sold, their prices fluctuate in a disadvantageous direction. This can be seen clearly in diamond color distribution. For example, color D whose price although high has a less median price which states color D is not the most purchased diamond.

On the other hand, the cut is increasing in level, however, the Ideal cut shows a fall in price even though it is the most common cut. However, just because a diamond with an Ideal cut has a low price does not mean that all diamonds with a good cut are low priced. Moreover, we can see the price increase as the cut becomes better and finally decreases as the cut becomes best.

## 2.4 Fitting multiple linear regression model

### 2.4.1 Motivation

The data-set consists the prices and other attributes of almost 54,000 diamonds. It's a great data-set for us to build and train a model for the prediction of diamonds price in the future. Hence, we need to explore whether carat, cut, colour, clarity, ... may affect most on the diamonds price. We extensively utilized the Multiple Linear Regression to analyze the relationship between the price and other attributes then applied this model for later prediction. The approach to this method will be discussed in this section.

### 2.4.2 MLR model

Regression analysis is a collection of statistical tools that are used to model and explore relationships between variables that are related in a nondeterministic manner.
**Multiple Linear Regression (MLR)** attempts to model a linear relationship between a dependent variable (response) and some independent variables (predictors/regressors). A model that might describe this relationship is

$$Y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon$$

where $\beta_0, \beta_1, ..., \beta_n$ are called partial regression coefficients since $\beta_i$ measures the change in Y per unit change in x_i when the other variables are kept constant.

### 2.4.3 Assumption

There are four assumptions associated with a linear regression model

- **Linearity**: The relationship between independent and dependent variables is linear. The linearity assumption can be checked with scatterplot of response versus regressor.

- **Independence**: There is no multicollinearity, i.e. dependencies between the independent variables. This assumption can be tested by correlation matrix or variance inflation factor. The magnitude of correlation coefficients greater than 0.8 or the VIF values exceed 10 indicates that multicollinearity is a problem.
- **Homoscedasticity**: The variance of residual is the same for any value of X. A scatterplot between residuals versus predicted values is a good way to check this assumption. There should be no pattern such as the cone-shaped pattern in the distribution.
- **Normality**: The errors between observed and predicted values should be normally distributed. We can check this assumption by looking at the histogram or Q-Q plot, or by applying goodness of fit on residuals.

### 2.4.4 Procedure and Result

#### 2.4.4.a Variables Selection

We have chosen price as dependent variables, but the problem is selecting the independent variables. Picking all variables as regressors is unnecessarily costly and sometimes it has a negative effect on our model since some variables are irrelevant to the response or there is no linear relationship between them.
An important problem in many applications of regression analysis involves selecting the set of regressor variables to be used in the model.
In such a situation, we are interested in variable selection; that is, screening the candidate variables to obtain a regression model that contains the "best" subset of regressor variables. To keep model maintenance costs to a minimum and to make the model easy to use, we would like the model to use as few regressor variables as possible.
The first approach is to plot a correlation matrix to determine the relationship of these variables. In R, we use function `cor()` to create a correlation table, the default method is Pearson, but we can also compute Spearman coefficient.

```
diamonds.cor = cor(diamonds[c(-2,-3,-4,-11,-12,-13)], method = c("spearman"))
diamonds.cor

##            carat       depth      table      price           x           y
## carat 1.00000000  0.03010375  0.1949803 0.96288280  0.99611660  0.99557175
## depth 0.03010375  1.00000000 -0.2450611 0.01001967 -0.02344221 -0.02542522
## table 0.19498032 -0.24506114  1.0000000 0.17178448  0.20223061  0.19573406
## price 0.96288280  0.01001967  0.1717845 1.00000000  0.96319611  0.96271882
## x     0.99611660 -0.02344221  0.2022306 0.96319611  1.00000000  0.99789493
## y     0.99557175 -0.02542522  0.1957341 0.96271882  0.99789493  1.00000000
## z     0.99318344  0.10349836  0.1598782 0.95723227  0.98735532  0.98706751
##               z
## carat 0.9931834
## depth 0.1034984
## table 0.1598782
## price 0.9572323
## x     0.9873553
## y     0.9870675
## z     1.0000000
```

We can pick 5 numeric variables with highest correlation (|corr|>0.1) and drop the depth, however the variable z has strong correlation with other independent variables such as x, y, carat, which violates the independence assumption of linear regression. So, we'd better drop this feature out of the regressor set.
Therefore, we keep carat, table, x, y, cut, clarity, color to be used for the full model.

The model will be built using the stepwise regression method based on AIC. AIC stands for Akaike Information Criteria. It evaluates the quality of differrent models relative to the other models. The lower AIC is the better. The stepwise regression method will add or subtract each variable in each step based on whether the model produced has a higher or lower AIC. stepAIC will also remove multicollinearity, the situation in which two or more explanatory variables are highly related.
The R function step() can be used to perform variable selection. First, we have to create a linear model `null` to form a model with no regressor variable (or one highest correlation variable) and a `full` model using all regressor variables.

```
full <- lm(price~carat+table+x+y+cut+color+clarity,data = diamonds)
null <- lm(price~1,data = diamonds)
```

We can perform stepwise regression using the command:

```
step(null, scope = list(upper=full), data=diamonds, direction="both")

## Start:  AIC=894477.9
## price ~ 1
##
##            Df  Sum of Sq        RSS    AIC
## + carat     1 7.2913e+11 1.2935e+11 792389
## + x         1 6.7152e+11 1.8695e+11 812259
## + y         1 6.4296e+11 2.1552e+11 819929
## + color     6 2.6849e+10 8.3162e+11 892776
## + clarity   7 2.3308e+10 8.3517e+11 893007
## + table     1 1.3876e+10 8.4460e+11 893601
## + cut       4 1.1042e+10 8.4743e+11 893788
## <none>                   8.5847e+11 894478
##
## Step:  AIC=792389.4
## price ~ carat
##
##            Df  Sum of Sq        RSS    AIC
## + clarity   7 3.9082e+10 9.0264e+10 772998
## + color     6 1.2561e+10 1.1678e+11 786891
## + cut       4 6.1332e+09 1.2321e+11 789777
## + x         1 3.5206e+09 1.2583e+11 790903
## + table     1 1.4377e+09 1.2791e+11 791789
## + y         1 1.2425e+09 1.2810e+11 791871
## <none>                   1.2935e+11 792389
## - carat     1 7.2913e+11 8.5847e+11 894478
##
```

```
## Step:  AIC=772998.5
## price ~ carat + clarity
##
##             Df  Sum of Sq        RSS    AIC
## + color     6 1.6402e+10 7.3862e+10 762193
## + x         1 1.8542e+09 8.8410e+10 771881
## + cut       4 1.7808e+09 8.8483e+10 771932
## + y         1 7.4127e+08 8.9523e+10 772556
## + table     1 3.7751e+08 8.9886e+10 772774
## <none>                   9.0264e+10 772998
## - clarity   7 3.9082e+10 1.2935e+11 792389
## - carat     1 7.4490e+11 8.3517e+11 893007
##
## Step:  AIC=762193.4
## price ~ carat + clarity + color
##
##             Df  Sum of Sq        RSS    AIC
## + x         1 2.7337e+09 7.1128e+10 760161
## + cut       4 1.6992e+09 7.2163e+10 760946
## + y         1 1.1450e+09 7.2717e+10 761353
## + table     1 4.0965e+08 7.3452e+10 761895
## <none>                   7.3862e+10 762193
## - color     6 1.6402e+10 9.0264e+10 772998
## - clarity   7 4.2923e+10 1.1678e+11 786891
## - carat     1 7.3364e+11 8.0750e+11 891202
##
## Step:  AIC=760161.1
## price ~ carat + clarity + color + x
##
##             Df  Sum of Sq        RSS    AIC
## + cut       4 1.9182e+09 6.9210e+10 758694
## + table     1 2.7374e+08 7.0855e+10 759955
## + y         1 5.3543e+06 7.1123e+10 760159
## <none>                   7.1128e+10 760161
## - x         1 2.7337e+09 7.3862e+10 762193
## - color     6 1.7281e+10 8.8410e+10 771881
## - clarity   7 4.0940e+10 1.1207e+11 784670
## - carat     1 6.9076e+10 1.4020e+11 796764
##
## Step:  AIC=758694.4
## price ~ carat + clarity + color + x + cut
##
##             Df  Sum of Sq        RSS    AIC
## + table     1 9.9353e+06 6.9200e+10 758689
## <none>                   6.9210e+10 758694
## + y         1 9.8210e+05 6.9209e+10 758696
## - cut       4 1.9182e+09 7.1128e+10 760161
## - x         1 2.9528e+09 7.2163e+10 760946
```

```
## - color    6 1.7239e+10 8.6449e+10 770679
## - clarity  7 3.6397e+10 1.0561e+11 781474
## - carat    1 7.0221e+10 1.3943e+11 796473
##
## Step:  AIC=758688.7
## price ~ carat + clarity + color + x + cut + table
##
##           Df  Sum of Sq        RSS    AIC
## <none>                  6.9200e+10 758689
## + y        1 9.3965e+05 6.9199e+10 758690
## - table    1 9.9353e+06 6.9210e+10 758694
## - cut      4 1.6544e+09 7.0855e+10 759955
## - x        1 2.9005e+09 7.2101e+10 760901
## - color    6 1.7243e+10 8.6444e+10 770678
## - clarity  7 3.6397e+10 1.0560e+11 781471
## - carat    1 6.9847e+10 1.3905e+11 796327
##
## Call:
## lm(formula = price ~ carat + clarity + color + x + cut + table,
##     data = diamonds)
##
## Coefficients:
##  (Intercept)         carat     clarityIF    claritySI1    claritySI2
##    -3328.122     11080.367      5391.155      3677.600      2720.774
##   clarityVS1    clarityVS2   clarityVVS1   clarityVVS2        colorE
##     4604.142      4287.281      5043.203      4981.255      -208.970
##       colorF        colorG        colorH        colorI        colorJ
##     -274.615      -488.740      -988.230     -1473.790     -2377.066
##            x       cutGood      cutIdeal    cutPremium  cutVery Good
##     -954.607       691.234      1035.073       943.263       886.323
##        table
##       -7.441
```

According to this procedure, the best model is the one that includes the variable carat, clarity, color, x, cut, table.

#### 2.4.4.b Conclusion

The most suitable model used for prediction is the one has the following regressors: carat, clarity, color, x, cut, table. Hence, those variables are factors which may affect diamonds price.
The final model used to prediction:

```
final = lm(formula = price ~ carat + clarity + color + x + cut + table,
       data = diamonds)
summary(final)
```

```
##
## Call:
## lm(formula = price ~ carat + clarity + color + x + cut + table,
```

```
##     data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21098.0   -599.0   -179.3    382.4  10719.3
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -3328.122    175.557 -18.958   <2e-16 ***
## carat         11080.367     47.496 233.288   <2e-16 ***
## clarityIF      5391.155     51.059 105.586   <2e-16 ***
## claritySI1     3677.600     43.727  84.103   <2e-16 ***
## claritySI2     2720.774     43.905  61.970   <2e-16 ***
## clarityVS1     4604.142     44.615 103.197   <2e-16 ***
## clarityVS2     4287.281     43.936  97.580   <2e-16 ***
## clarityVVS1    5043.203     47.216 106.812   <2e-16 ***
## clarityVVS2    4981.255     45.920 108.477   <2e-16 ***
## colorE         -208.970     17.937 -11.650   <2e-16 ***
## colorF         -274.615     18.137 -15.141   <2e-16 ***
## colorG         -488.740     17.755 -27.527   <2e-16 ***
## colorH         -988.230     18.875 -52.355   <2e-16 ***
## colorI        -1473.790     21.209 -69.488   <2e-16 ***
## colorJ        -2377.066     26.191 -90.759   <2e-16 ***
## x              -954.607     20.080 -47.540   <2e-16 ***
## cutGood         691.234     32.957  20.974   <2e-16 ***
## cutIdeal       1035.073     31.099  33.283   <2e-16 ***
## cutPremium      943.263     30.342  31.088   <2e-16 ***
## cutVery Good    886.323     30.769  28.806   <2e-16 ***
## table            -7.441      2.674  -2.782   0.0054 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1133 on 53919 degrees of freedom
## Multiple R-squared:  0.9194,Adjusted R-squared:  0.9194
## F-statistic: 3.075e+04 on 20 and 53919 DF,  p-value: < 2.2e-16
```

**Residuals**: The maximum error of 10719.3 suggests that the model underpredicted the price by 10719.3 for at least one observation. 50% of errors fall within the first and third quartiles, between 179.3 over the price and 382.4 under the true value.

**P-value**: small p-values suggest that features are extremely unlikely to have no relationship to the dependent variable.

**Multiple R-squared** value of 0.9194 suggests that the model explains approx. 91.94% of the variation in the dependent variable.

Given the above, the model is performing well.

## 2.5 Making Prediction

The final step of building a Multiple Linear Regression model is testing. We have picked a sample of the data set as training set to estimate the regression coefficients. The remaining is now used as testing set to check the performance of our model.(ratio 3:1)

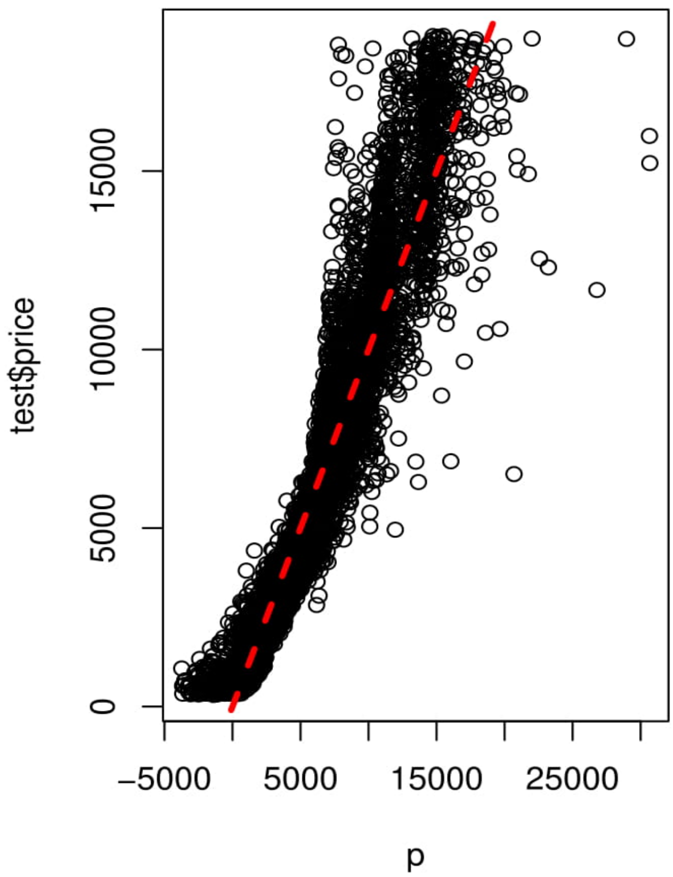
```
set.seed(101) # Set Seed so that same sample can be reproduced in future also
# Now Selecting 75% of data as sample from total 'n' rows of the data
sample <- sample.int(n = nrow(diamonds), size = floor(.75*nrow(diamonds)), replace = F)
train <- diamonds[sample, ]
test  <- diamonds[-sample, ]
train = lm(formula = price ~ carat + clarity + color + x + cut + table,
          data = train)
# Fit the MLR model
```

There are several model evaluation metrics for Regression and we used Root **Mean Square Error** to evaluate the model.
Mean Square Error is one of the most popular metrics to evaluate a model. The smaller it is, the better predictions the model can produce. In addition, if the RMSE/mean is smaller than 1, that means the model has done a great job at predicting.

```
p = predict(train, test)
RMSE(p, test$price)/mean(test$price)

## [1] 0.2929142
```

We can also plot to see how these predicted and actual data fitting.

```
plot(p, test$price)
abline(a=0, b=1, col="red", lwd=3, lty=2)
```




# References

[1] Douglas C. Montgomery Applied Statistics and Probability for Engineers 2014