Exploratory Data Analysis of 50000 Diamonds

Ali Ramezanian Nik

2023-03-26

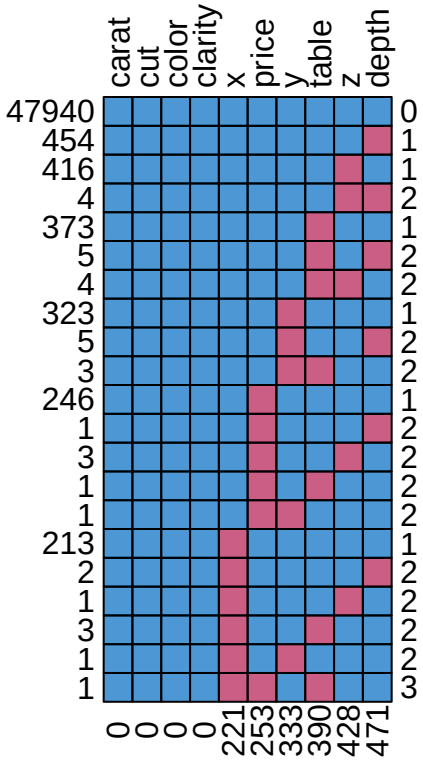
Introduction

This is an exploratory data analysis on a data set that is about the information of 50000 of diamonds. The provided data is about carat, price, depth, table, width, length, height, clarity, cut and color of diamonds. During this EDA we will compare these columns with each other and try to find the relations between these columns.
The first part of each EDA is data cleaning and we go in more details in the next part.

Cleaning NA's, Outliers and Incorrect Values from Diamonds

As we learnt from BDA course there is an important step in data analysis and it is data cleaning. So we start it now without killing time.
Let's first have an overview of missed Diamond indexes:

```
diamond_raw <- read.csv("DiamondData.csv")
devnull <- md.pattern.customized(diamond_raw, rotate.names = TRUE)
```



We can Understand from missing plot that there is only one row that has three missing columns. and there

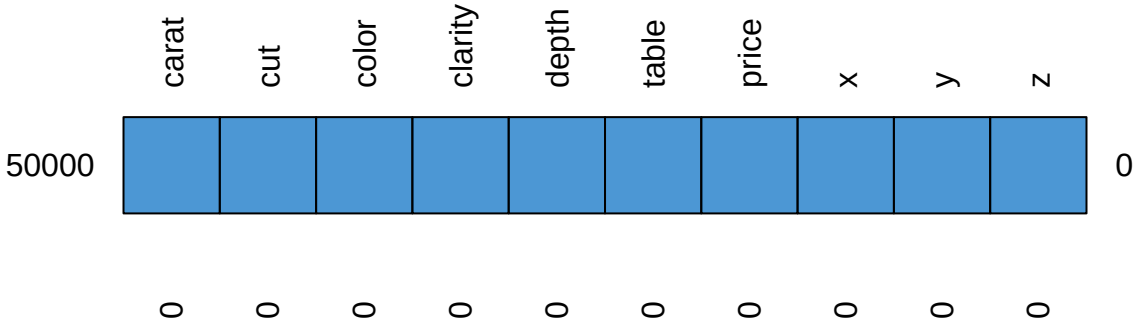
are many rows that has 1 or 2 missing columns.

Fill missing data

Now Let's fill the missing diamonds data with a famous method called PMM method.

```
diam <- mice(diamond_raw, method = 'pmm', printFlag = FALSE, )
diam <- complete(diam)
devnull <- md.pattern.customized(diam, rotate.names = TRUE)
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```



As You can see, all NA's has filled with "PMM" Method and MICE can approve that with its message.

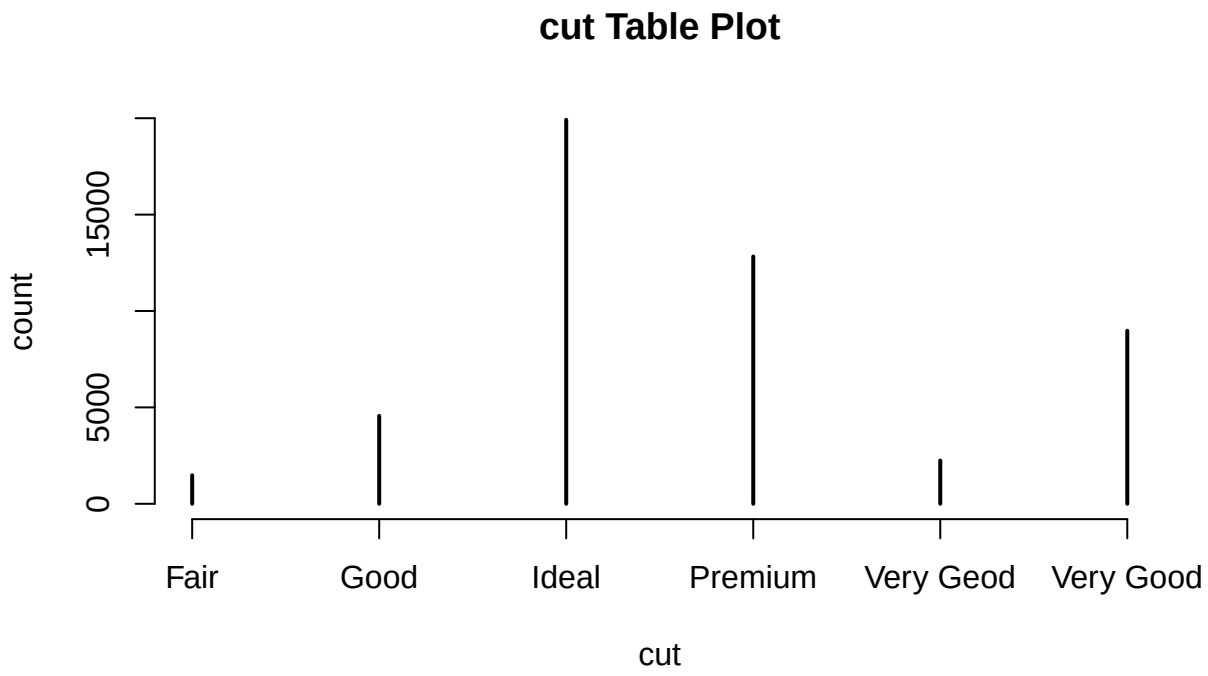
Viewing Tables to understand More about Diamonds dataset

In the previous step we removed all NA's from our data set and now we have pure data without anything missing. But Does This mean that our data does not have any outliers or misspelled words? For sure not. Let's view our qualitative columns to see whether we can hunt some smelly data.

Detective eye On 'Cut' column

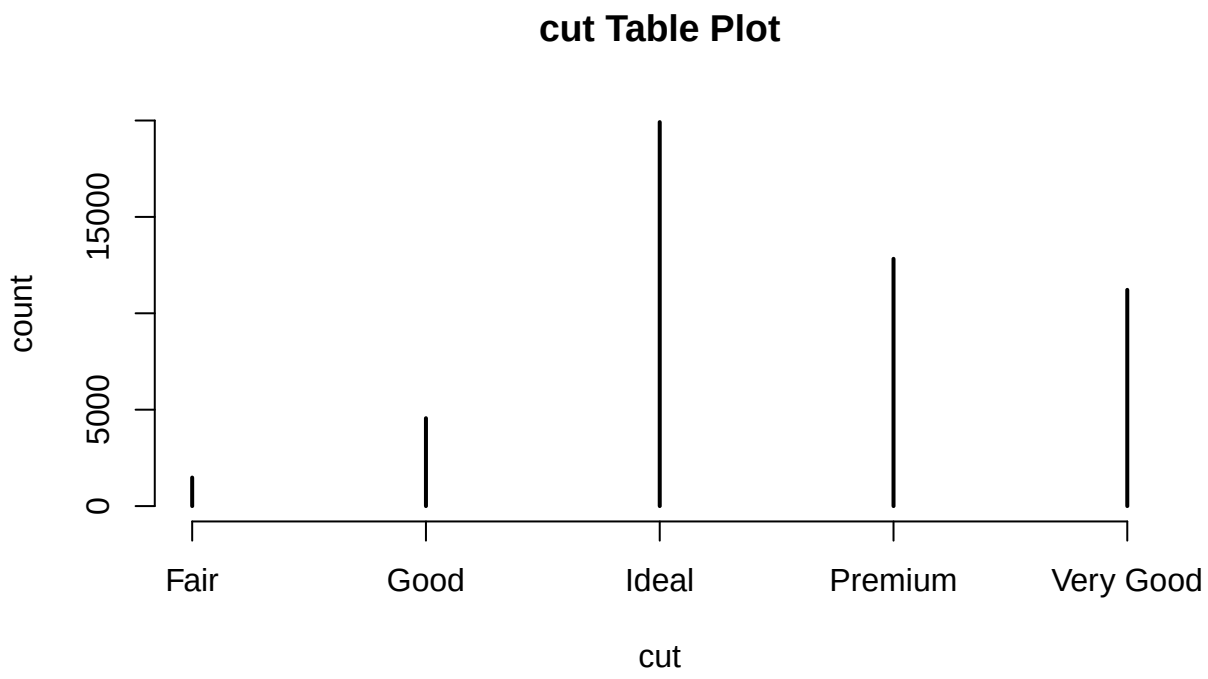
Let's view Cut table plot:

```
table.plot(diam, 'cut', "Table Plot")
```



There are only 5 legal labels for "cut" but here we can observe 6 labels and one
them are so strange: "Very Geod" ? It seems that the writer wanted to record "Very Good" but recorded "Very Geod". So let's correct that.

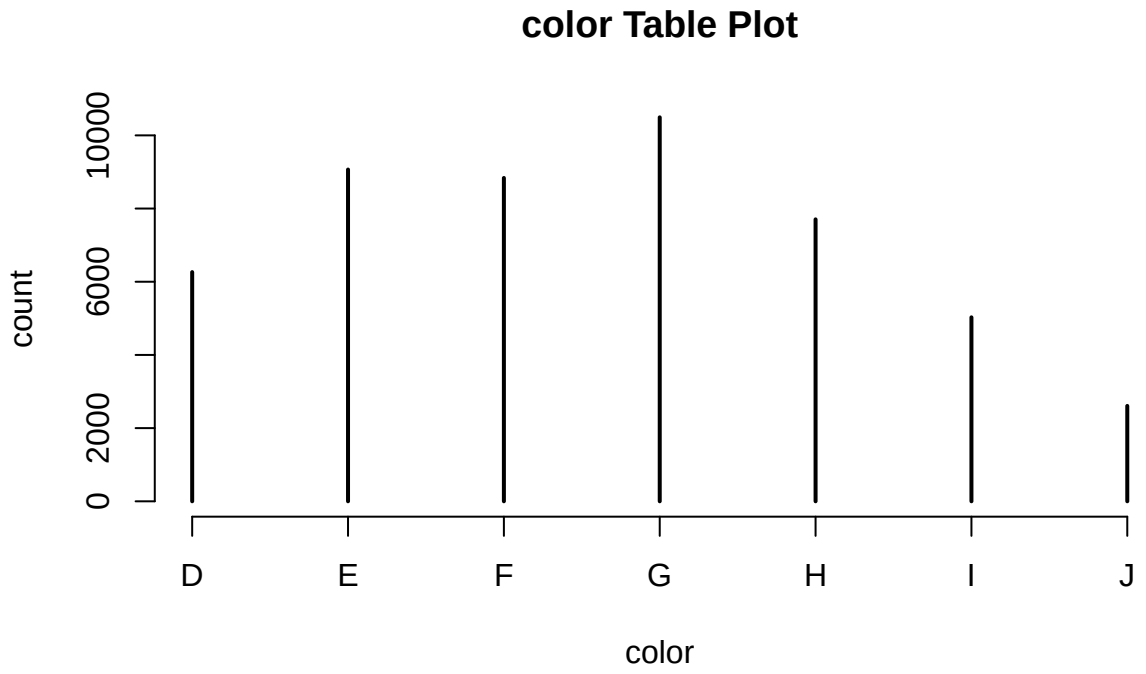
```
diam$cut <- str_replace(diam$cut, "Very Geod", "Very Good")
table.plot(diam, 'cut', "Table Plot")
```



Detective eye On 'Color' column

Let's view color table plot:

```
table.plot(diam, 'color', "Table Plot")
```

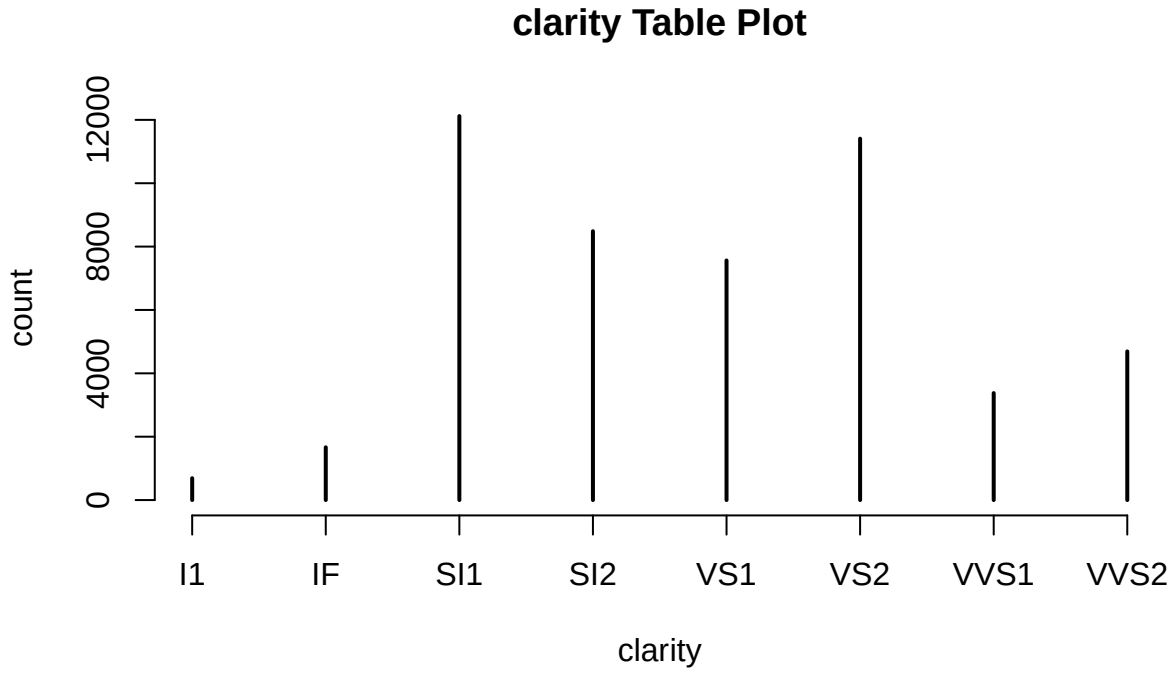


No error is observed within the color column.

Detective eye On 'Clarity' column

Let's view clarity table plot:

```
table.plot(diam, 'clarity', "Table Plot")
```

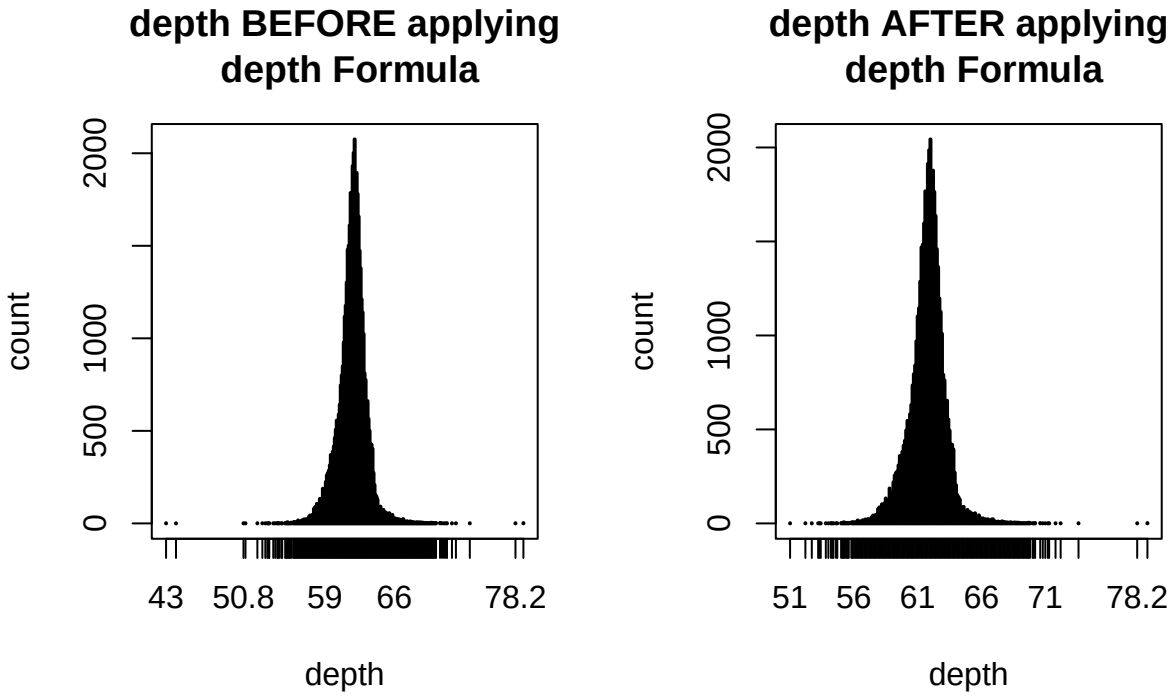


Now Let's check Quantitative columns for smelly data and remove them

Detective eye on 'Depth' column

We know the depth of each diamonds is dependant to its length, width, and height and the formula is: $depth = ((2 * z)/(x + y)) * 100$
So we should apply it to each row and delete rows that does not satisfy the formula.
but I decided to be not so harsh in deletion rows and I set some penalty for comparing the result with formuala (0.5).

```
par(mfrow = c(1, 2))
p1 <- table.plot(diam, 'depth', "BEFORE applying\n depth Formula")
diam <- apply.depth.formula(diam)
p2 <- table.plot(diam, 'depth', "AFTER applying\n depth Formula")
```

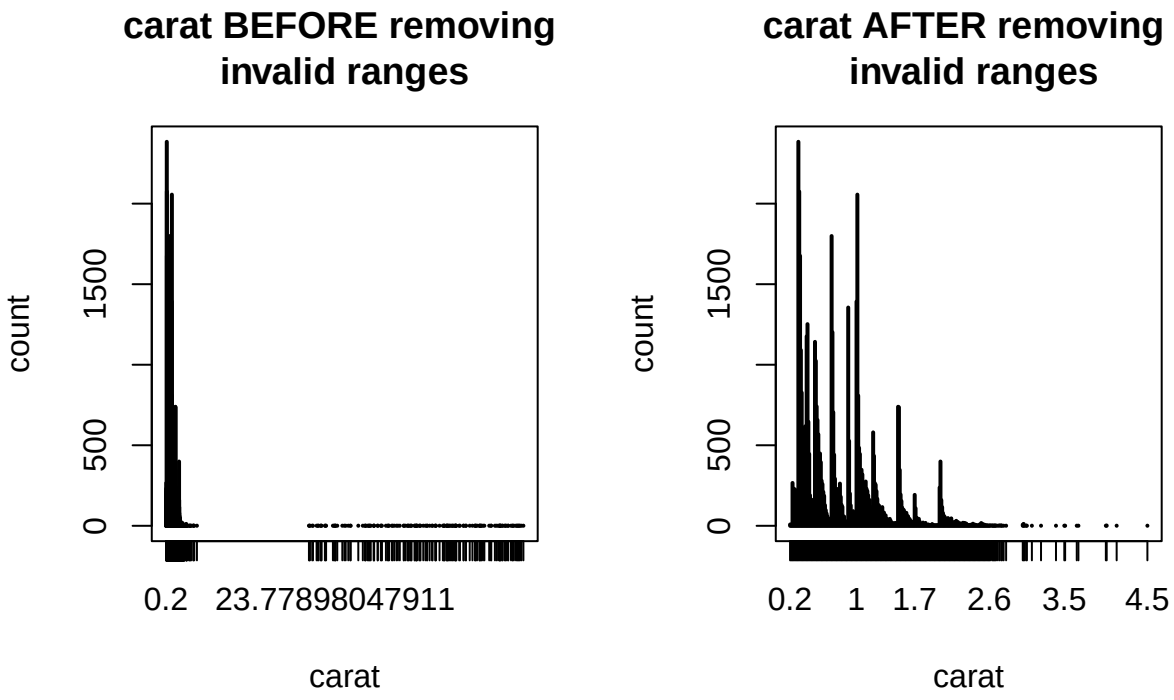


Some depth ranges has been removed and now depth starts from 51 instead of 43 but the upper range does not affected when we applied the depth formula.

Detective eye on 'Carat' column

We Know valid carat ranges are from 0.2 to 5.01
So we delete anything beyond and lower that range:

```
par(mfrow = c(1, 2))
p1 <-
  table.plot(diam, 'carat', "BEFORE removing \n invalid ranges ")
diam <- apply.carat.formula(diam)
p2 <- table.plot(diam, 'carat', "AFTER removing \n invalid ranges")
```

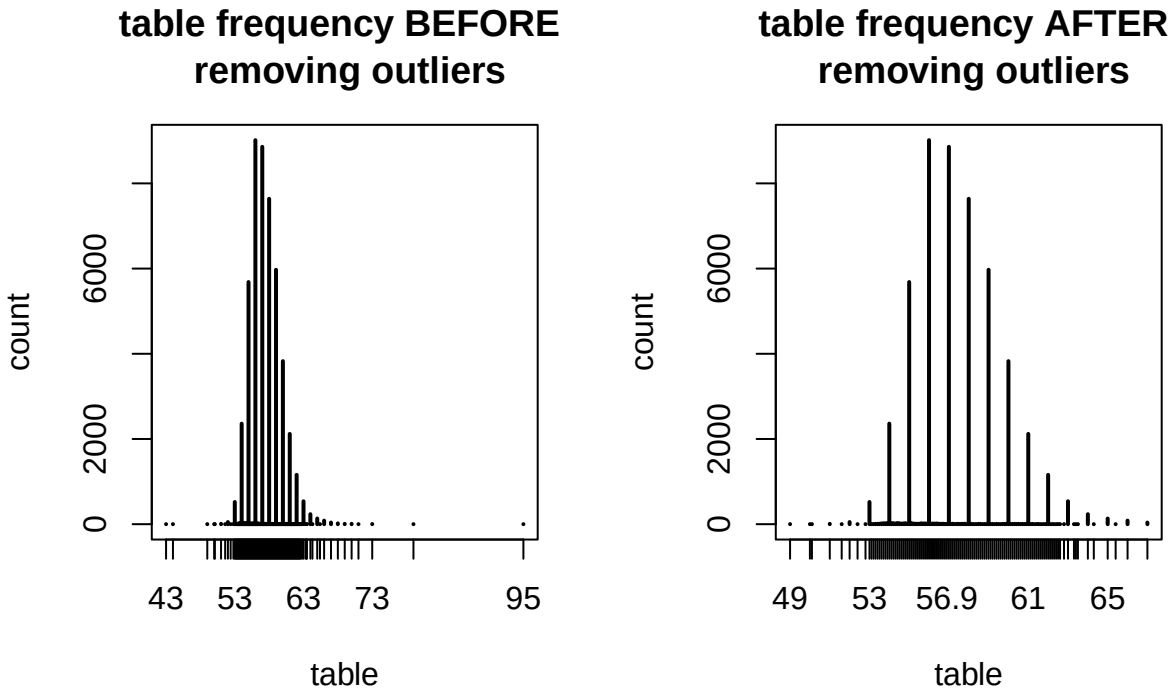


It is important to observer that there were so many outliers with high values that were removed after applying the range policy.

Detective eye on 'Table' column

There is no rule specified for Table values, So we should detect outliers based on interquartile method and remove them correspondingly:

```
par(mfrow = c(1, 2))
diam <- interquantile.method(diam, "table")
```



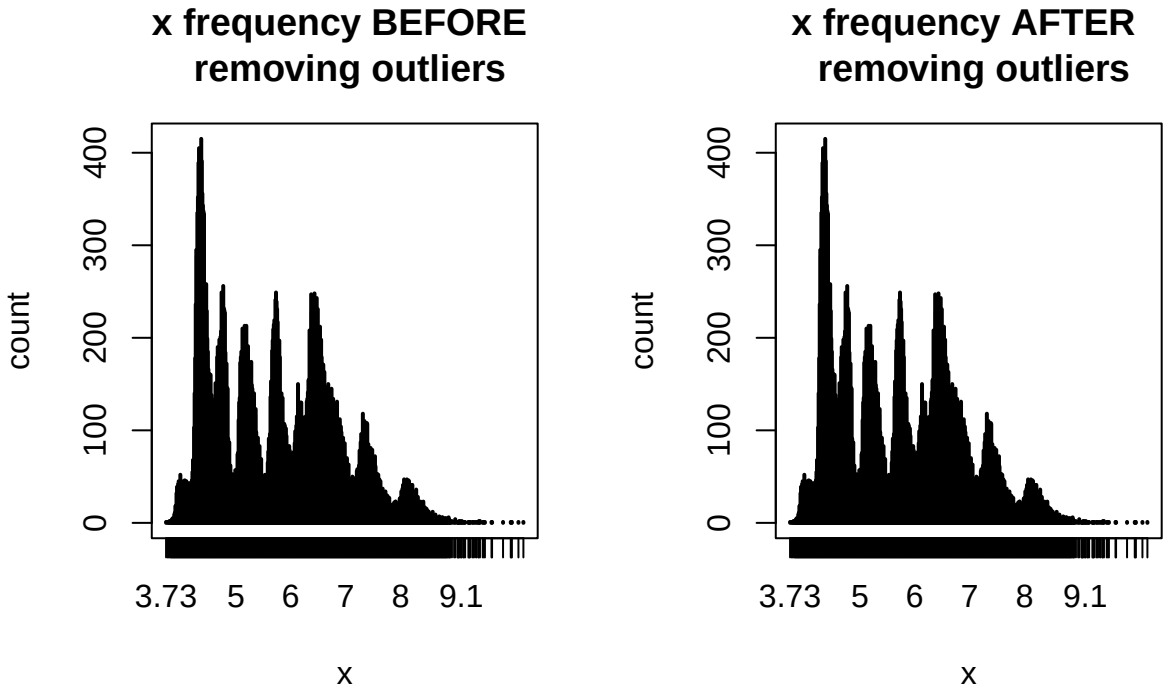
```
## [1] "46 of row deleted AFTER removing table outliers."
```

After removing table outliers range of values changed from around (43 to 95) to (49 to 65).

Detective eye on 'x, y and z' column

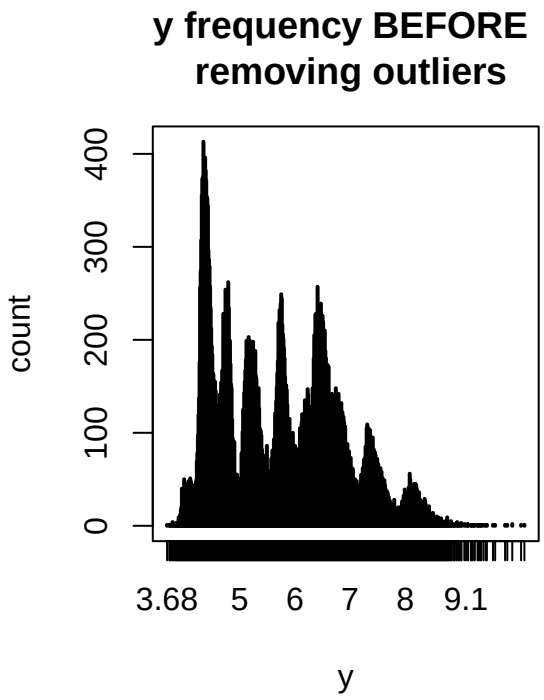
Let's first remove the outliers with Interquartile method and then compare the plots and decide whether deleting them are logical or not:

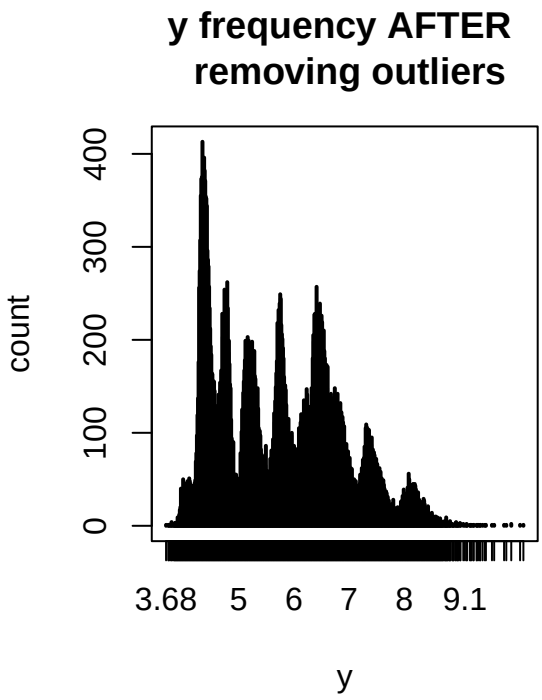
```
par(mfrow = c(1, 2))
diam <- interquantile.method(diam, "x")
```




```
## [1] "0 of row deleted AFTER removing x outliers."
```

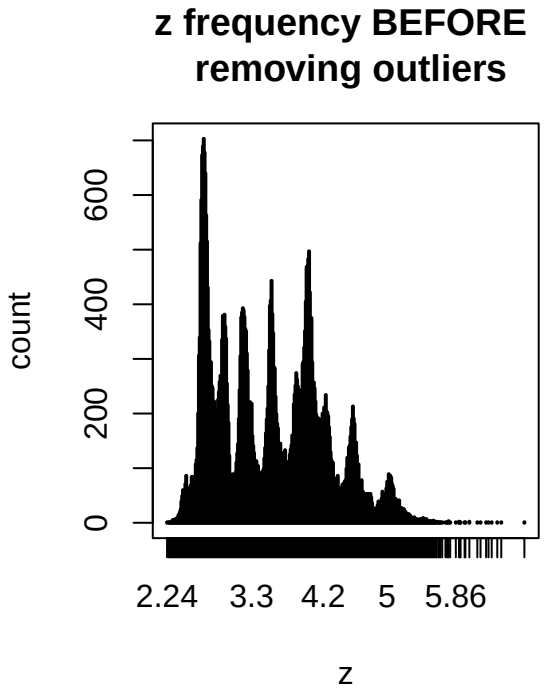
```
diam <- interquantile.method(diam, "y")
```

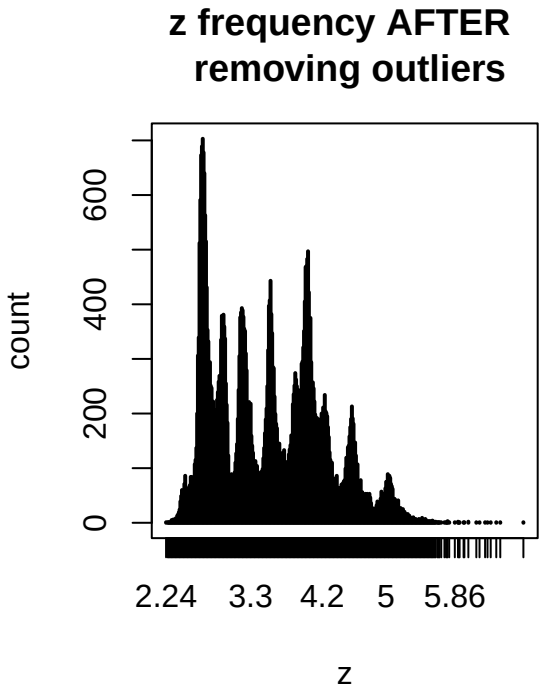





```
## [1] "0 of row deleted AFTER removing y outliers."
```

```
diam <- interquantile.method(diam, "z")
```





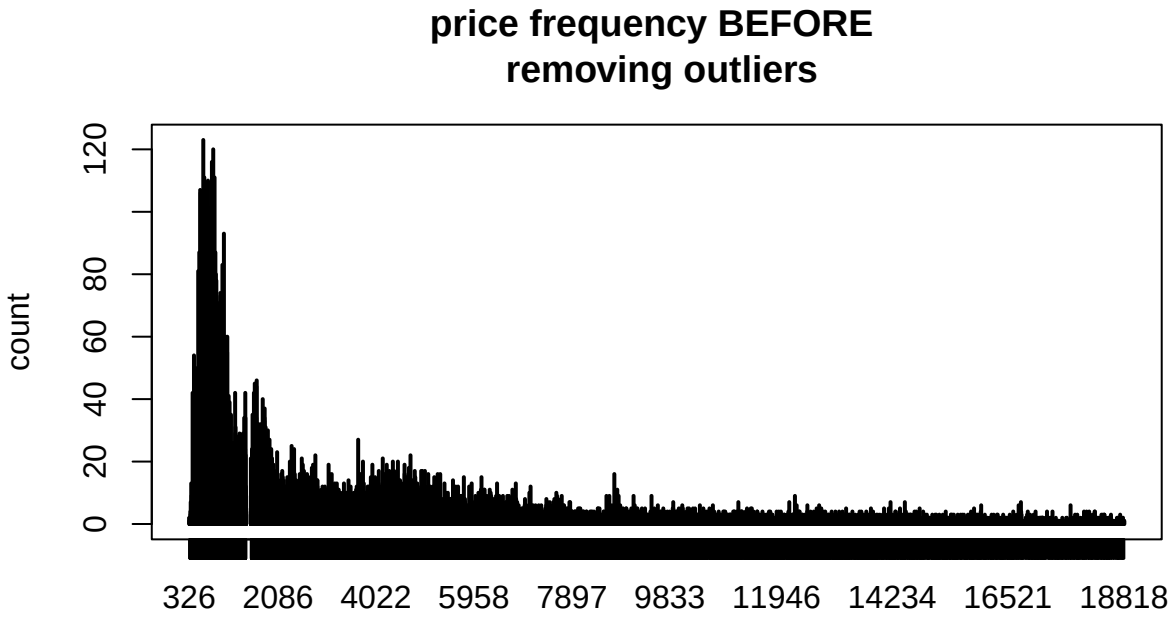

```
## [1] "0 of row deleted AFTER removing z outliers."
```

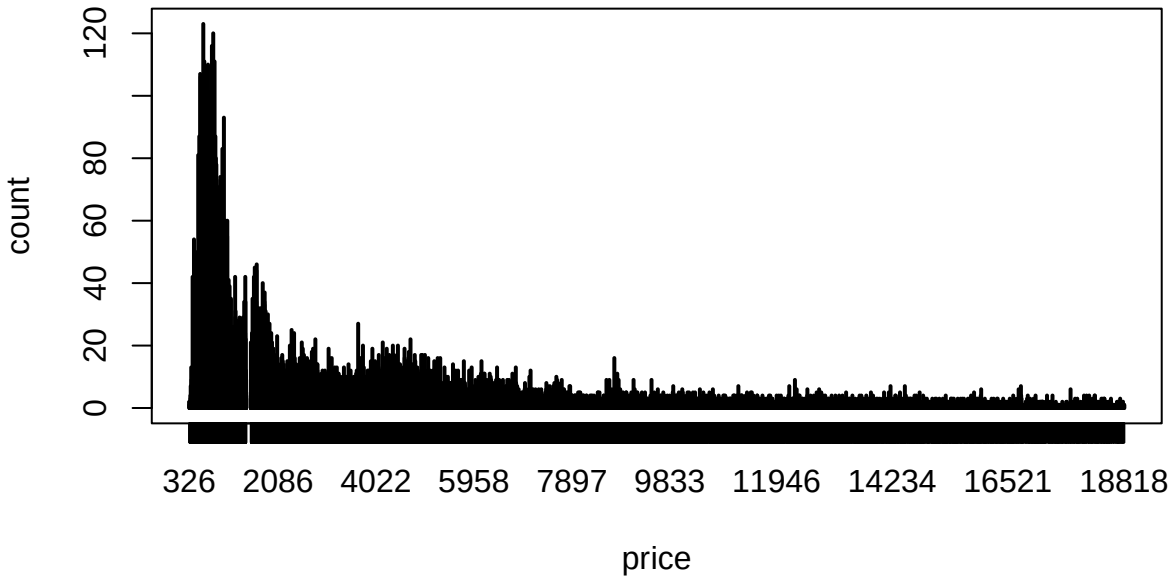
Interesting! No outliers were removed and why? because we already delete outliers like zero values columns when we applied “depth satisfication formula”

Detective eye on 'Price' column

We can not decide whether a price would be an outlier or not, because it is depend on the size and the person who paid, infact its depend on the shape and taste of buyer. So we set sieve_size of interquartile.method from 3 (default) to 4 to remove less indexes.

```
diam <- interquantile.method(diam, "price", sieve_size = 4)
```






```
## [1] "0 of row deleted AFTER removing price outliers."
```

No outliers are removed and its normal because we choosed a large sieve.

Summary of Variables

To Get Started Let's see the summary of our diamonds dataset.

```
summary(diam)
```

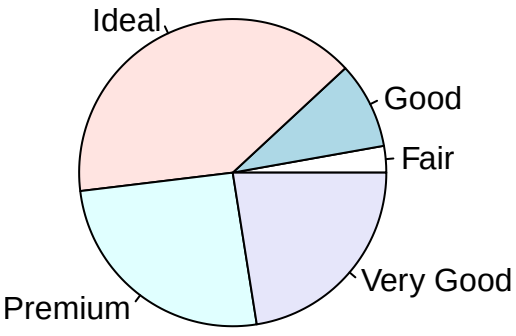
```
##      carat            cut               color             clarity
##  Min.   :0.2000   Length:49100       Length:49100       Length:49100
##  1st Qu.:0.4000   Class :character   Class :character   Class :character
##  Median :0.7000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.7964
##  3rd Qu.:1.0400
##  Max.   :4.5000
##      depth           table           price             x
##  Min.   :51.00   Min.   :49.00   Min.   :  326   Min.   : 3.730
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  947   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2397   Median : 5.690
##  Mean   :61.75   Mean   :57.45   Mean   : 3923   Mean   : 5.729
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5316   3rd Qu.: 6.540
##  Max.   :78.20   Max.   :67.00   Max.   :18823   Max.   :10.230
##        y               z
##  Min.   : 3.680   Min.   :2.240
##  1st Qu.: 4.720   1st Qu.:2.910
##  Median : 5.710   Median :3.520
##  Mean   : 5.731   Mean   :3.538
##  3rd Qu.: 6.530   3rd Qu.:4.030
##  Max.   :10.160   Max.   :6.720
```

From the summary of data we can observe that 3 columns are qualitative and other columns are quantitative. Also we can see the mean of X and Y is equal while Z has a different value. So the width and length are almost the same but height is different.

In This part We will plot pie-chart, bar-chart and histogram for some of our columns.

Pie-chart for cut column

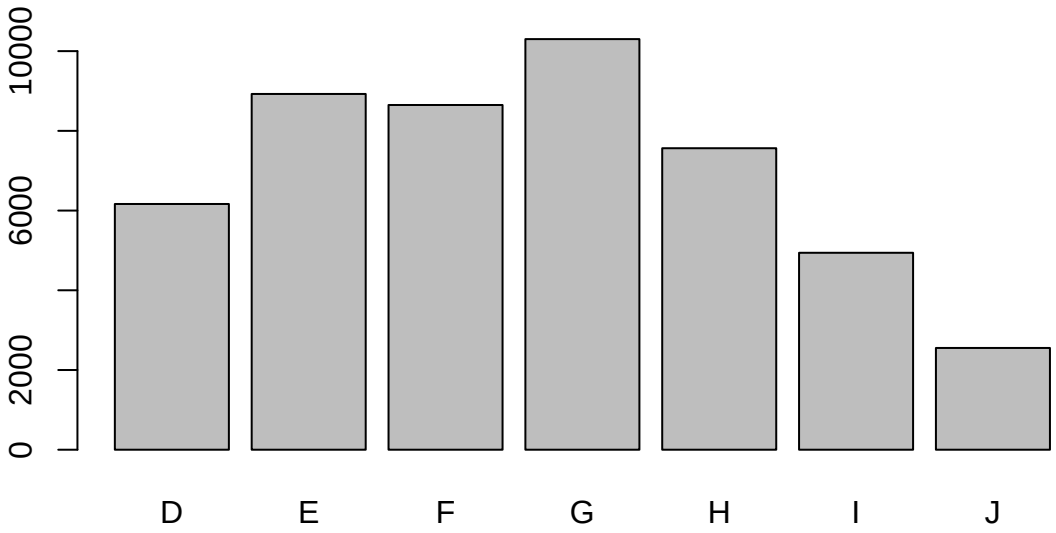
```
pie(table(diam$cut))
```



We can conclude that the Ideal and Premium cuts are more common in the dataset.

Bar-chart for color

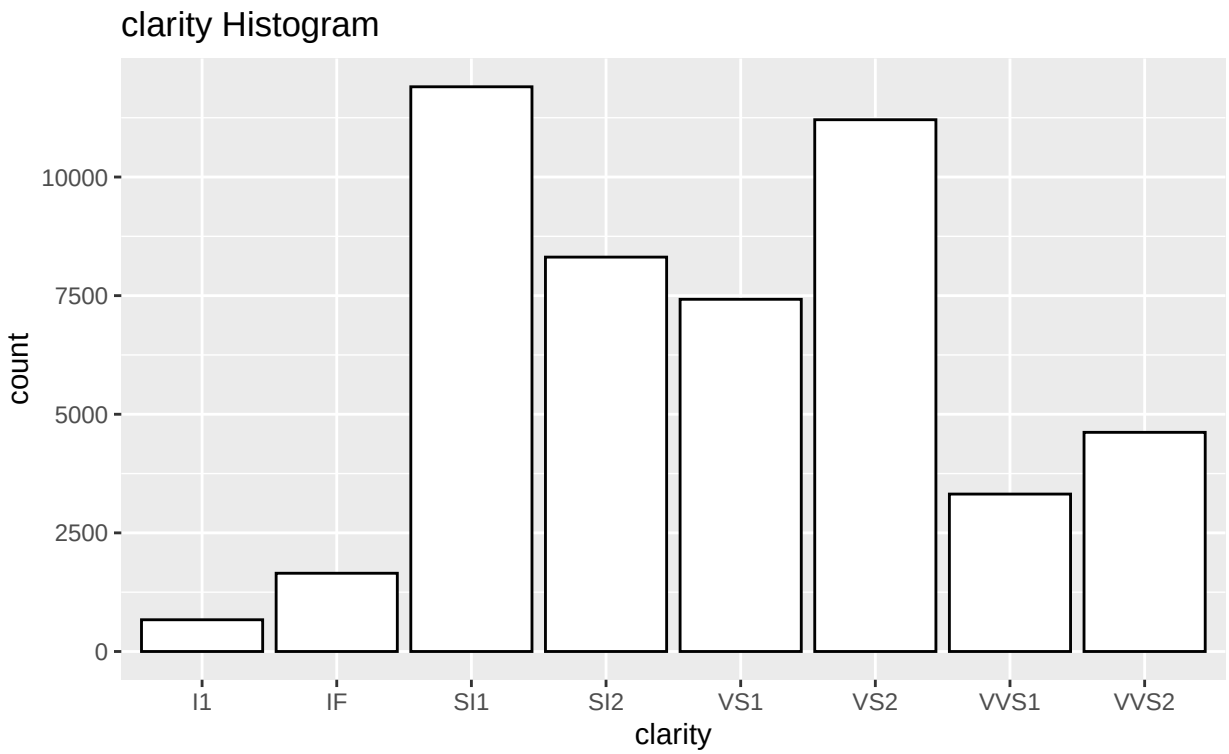
```
barplot(table(diam$color))
```



It is clear that the G color has dominant over other colors, so a considerable numbers of diamonds has average quality of color.

Histogram for clarity

```
ggtable.plot(diam, 'clarity', "Histogram")
```



From the top plot we cant conclude that there are only few diamonds that has very low clarity and very high clarity and other ones has an average clarity.

Analysis of Price

One of the most important attribute of each diamond is its price. So it is worth to analyze it in deep details.

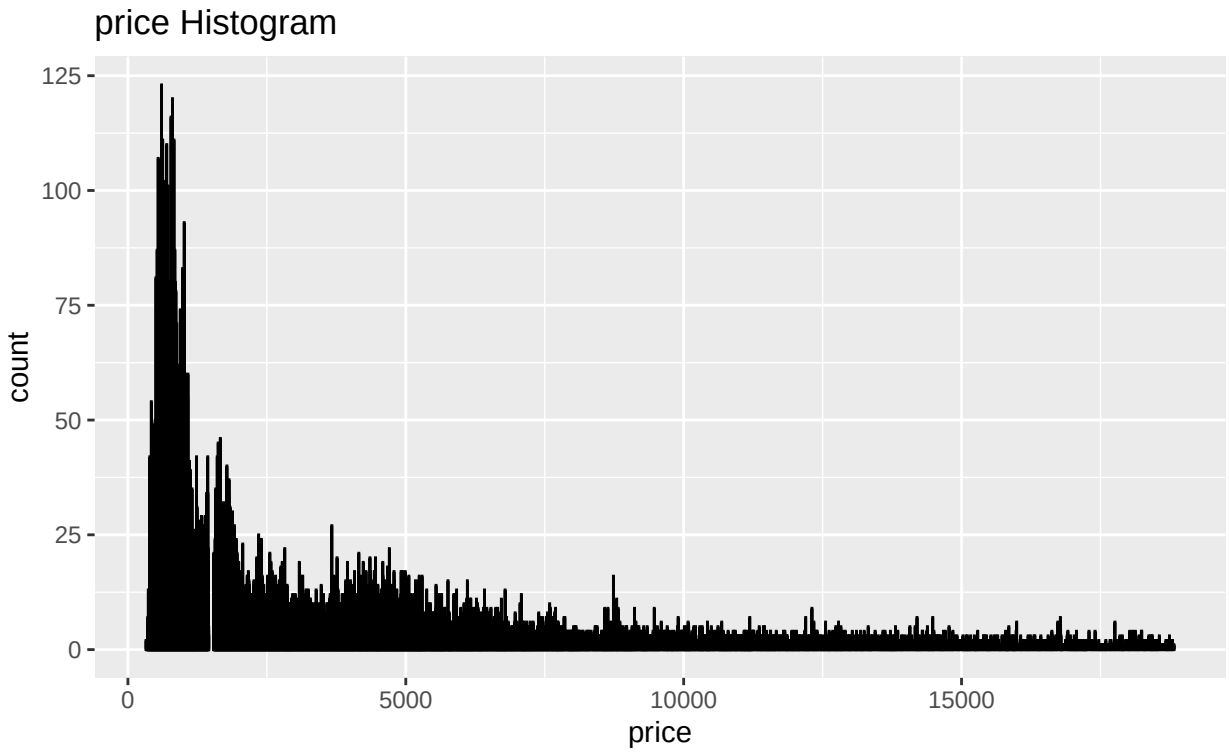
Histogram and Summarizing Price Column

By summarizing the price column and plotting its histogram, we can find its details.

```
summary(diam$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     947    2397    3923    5316   18823
```

```
price_histogram <- ggplot(diam, aes(x = price)) + geom_histogram()
ggtable.plot(diam, 'price', "Histogram")
```




```
price_variance <- var(x = diam$price)
standard_dev <- sqrt(price_variance)
standard_dev
```

```
## [1] 3980.598
```

We can conclude from the plot, mean, median and standard_deviation that most of the diamond prices is within 326 to mean+standard_deviation. Variance itself does not enlight many things about price except the sum of square of distance from mean. Also we can understand there is no more than 125 diamonds that has “near same prices”.

Group Diamonds by Price Column

We are going to add another column named price_group and we catogorize diamonds based on their prices.

```
diam$price_group <- with(diam, factor(
  findInterval(price, c(-Inf, quantile(
    price, probs = c(0.25, .5, .75)
  ), Inf)),
  labels = c("Low", "Medium", "High", "Very High")
))
```

```
tail(diam)
```

```
##       carat       cut color clarity depth table price    x    y    z
## 49995  0.50 Very Good     I     SI1  62.4    58  1048 5.00 5.06 3.14
## 49996  2.01   Premium     I     SI1  62.4    61 15707 8.01 7.99 4.99
## 49997  0.80   Premium     G      IF  62.6    58  4193 5.93 5.89 3.70
## 49998  0.90 Very Good     F     SI2  61.4    62  3770 6.11 6.20 3.78
## 49999  1.24     Ideal     F     SI2  61.4    58  6503 6.85 6.90 4.22
## 50000  0.70     Ideal     I    VVS1  61.4    55  2489 5.70 5.76 3.52
##       price_group
## 49995      Medium
## 49996   Very High
## 49997        High
## 49998        High
## 49999   Very High
## 50000        High
```

Now it's time to summarize each group of prices.

```
summarise(
  group_by(diam, price_group),
  mean = mean(price),
  standard_deviation = sd(price),
  min_value = min(price),
  max_value = max(price),
  count = n()
)
```

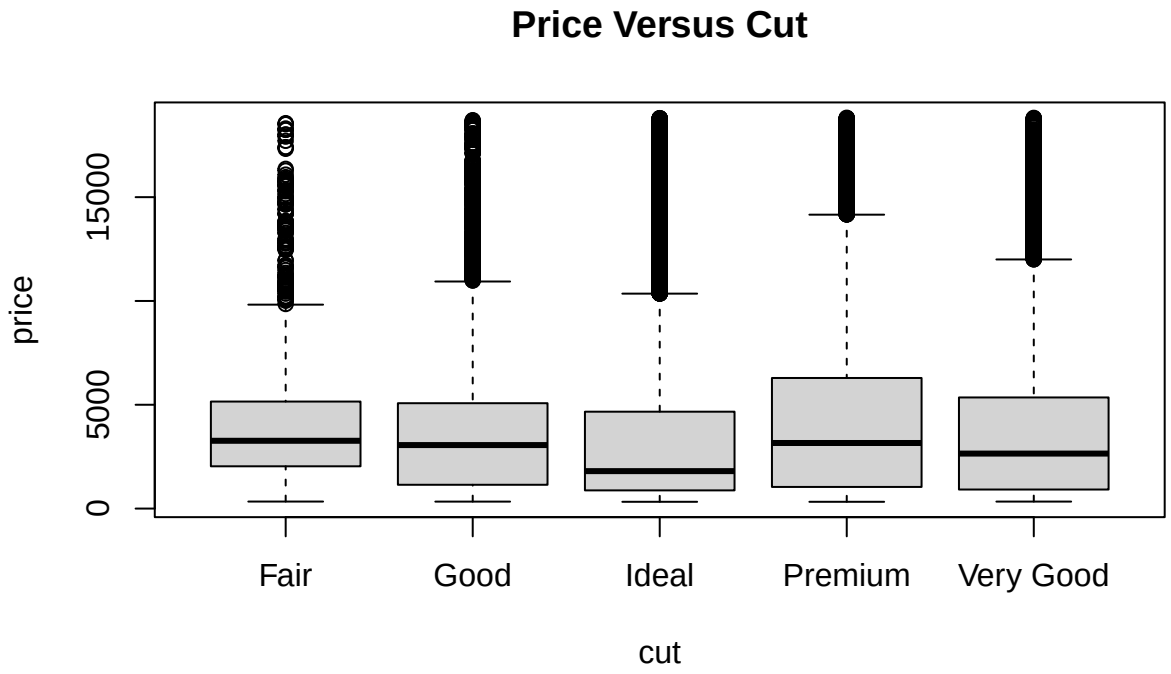
```
## # A tibble: 4 x 6
##   price_group  mean standard_deviation min_value max_value count
##   <fct>       <dbl>              <dbl>     <int>     <int> <int>
## 1 Low          687.               149.       326       946 12274
## 2 Medium      1556.               442.       947      2396 12275
## 3 High        3781.               856.      2397      5315 12275
## 4 Very High   9668.              3652.      5316     18823 12276
```

The above Table is the summarize of prices based on price_group.

Prices Versus Cuts

To analyze prices versus cuts, first we should observe the boxplot:

```
boxplot(price ~ cut, data = diam, main="Price Versus Cut")
```




```
summarise(
  group_by(diam, cut),
  mean = mean(price),
  standard_deviation = sd(price),
  min_value = min(price),
  max_value = max(price),
  count = n()
)
```

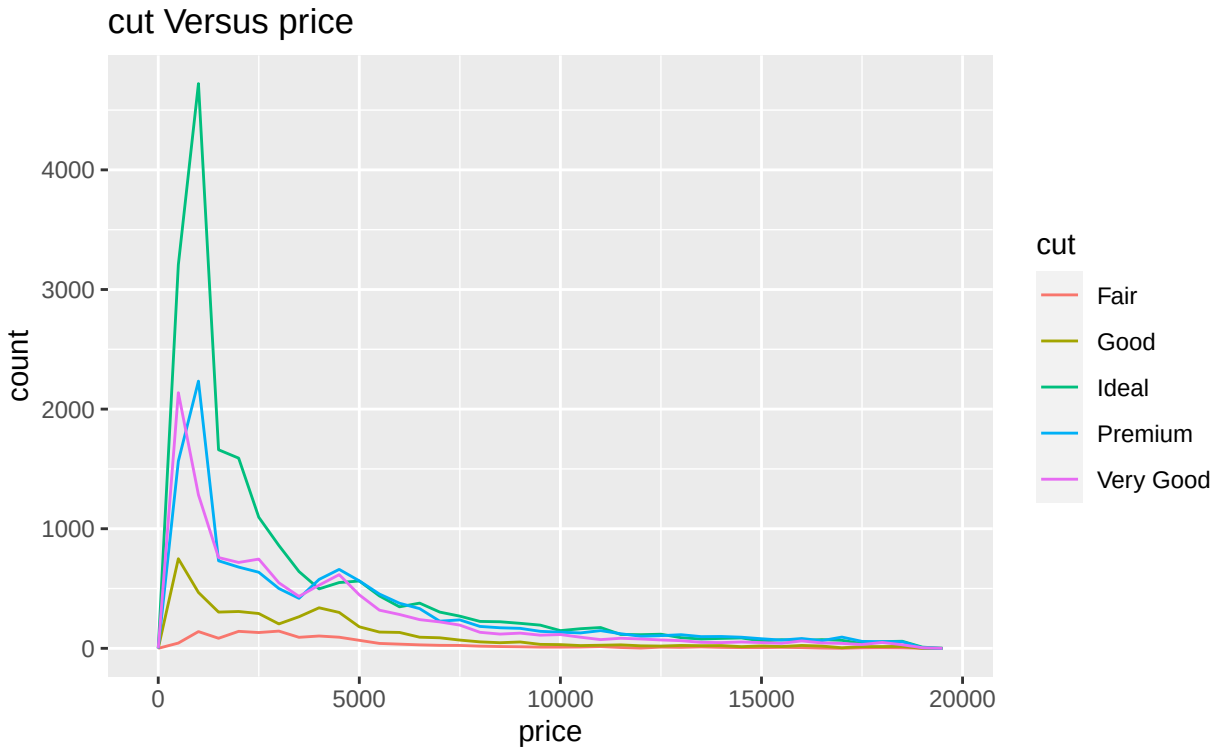
```
## # A tibble: 5 x 6
##   cut        mean standard_deviation min_value max_value count
##   <chr>     <dbl>              <dbl>     <int>     <int> <int>
## 1 Fair      4315.              3507.       337     18565  1375
## 2 Good      3945.              3684.       335     18707  4471
## 3 Ideal     3444.              3797.       326     18806 19645
## 4 Premium   4571.              4341.       326     18823 12564
## 5 Very Good 3981.              3928.       336     18818 11045
```

As we can observe from the box-plot and also the standard_deviation column of summarize table, premium and very-good cuts are more expensive but the good news is that even if you want a cheap (326$) diamond and still you want it has premium or very-good cut, you can buy it because as we can see from min_value of summarize table, there are variety cuts of diamonds with cheap prices.

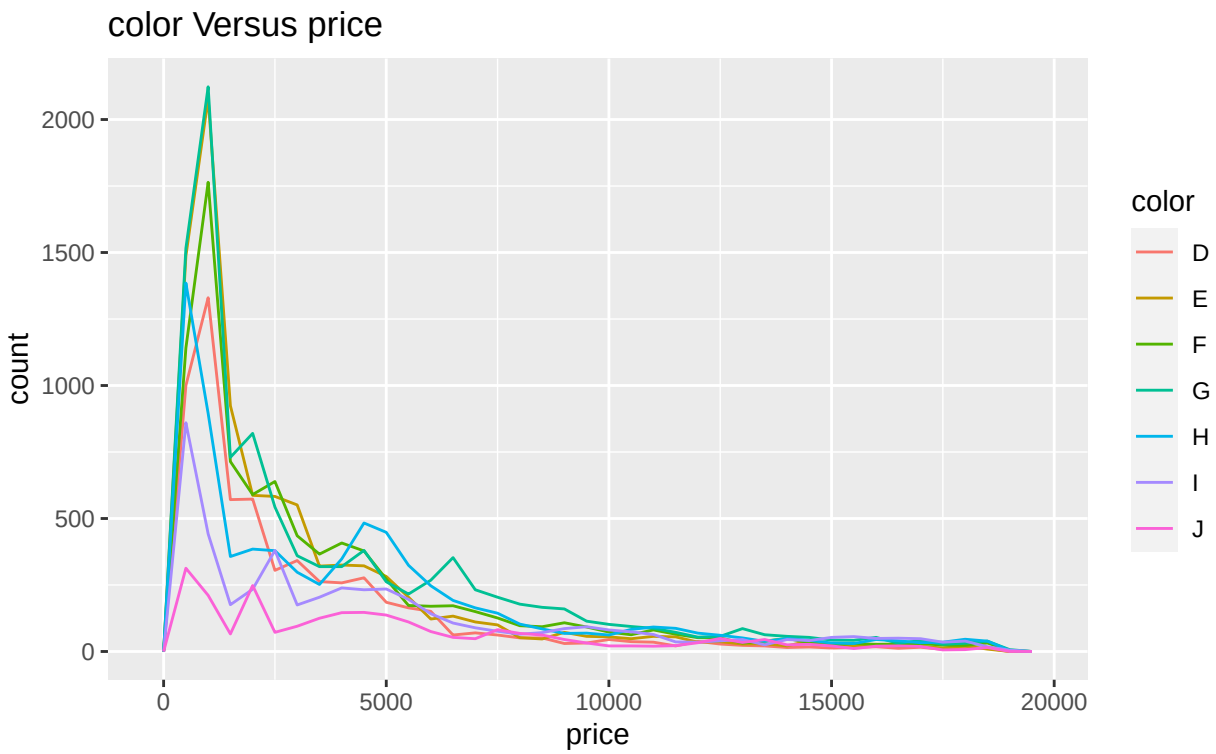
Columns Which Correlated with Price

To find out which columns are correlated with diamonds lets plot against other columns.

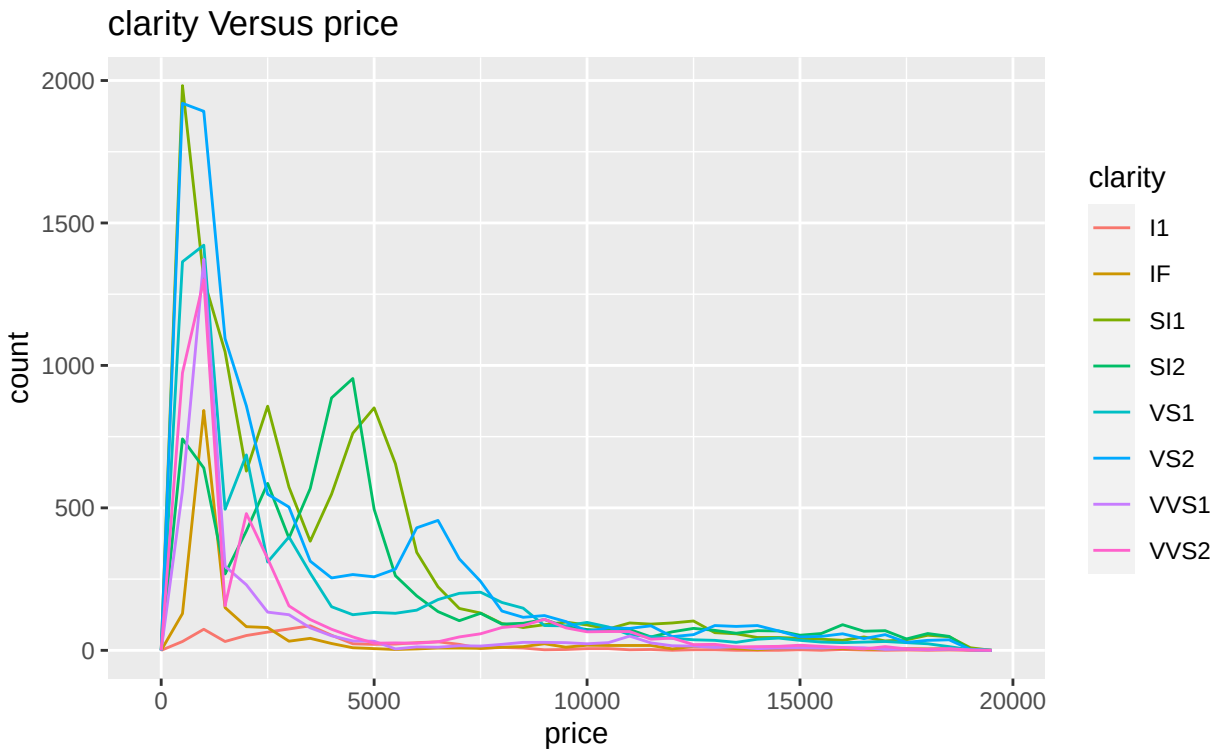
```
plot_against(diam, 'price', 'cut', quanitative = FALSE);
```



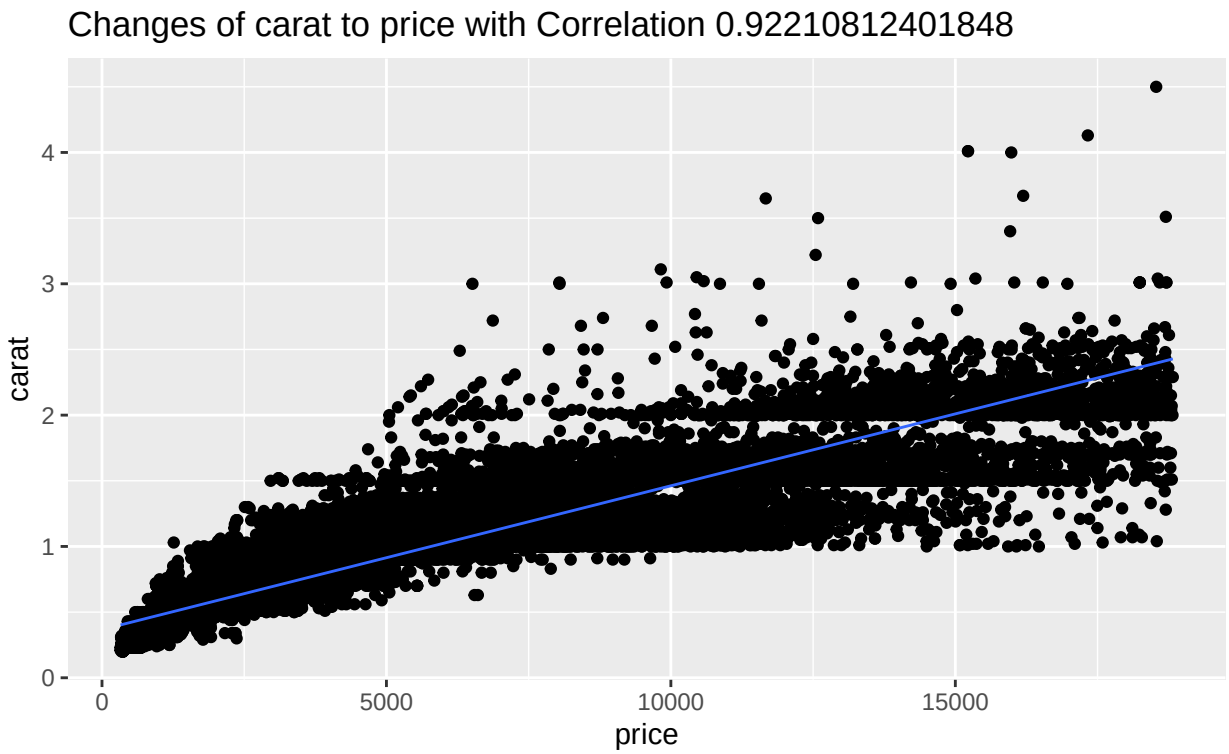

```
plot_against(diam, 'price', 'color', quanitative = FALSE);
```



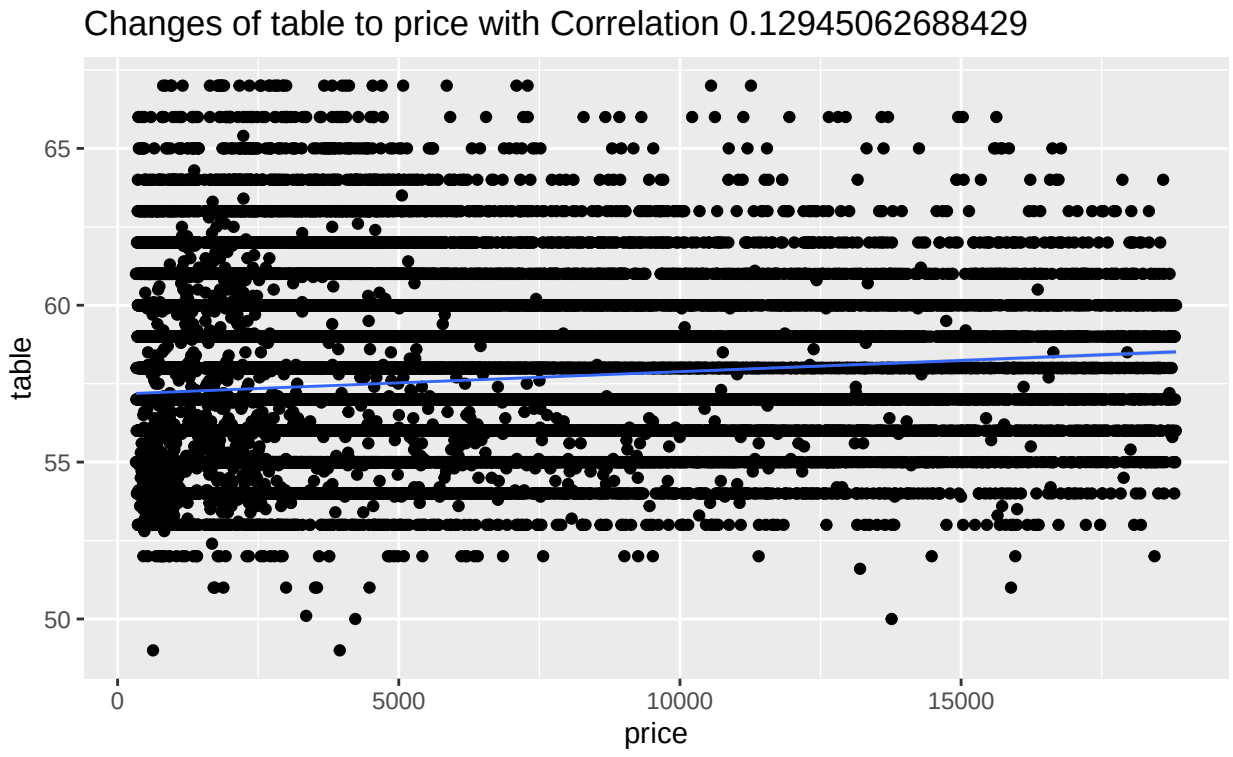

```
plot_against(diam, 'price', 'clarity', quanitative = FALSE);
```



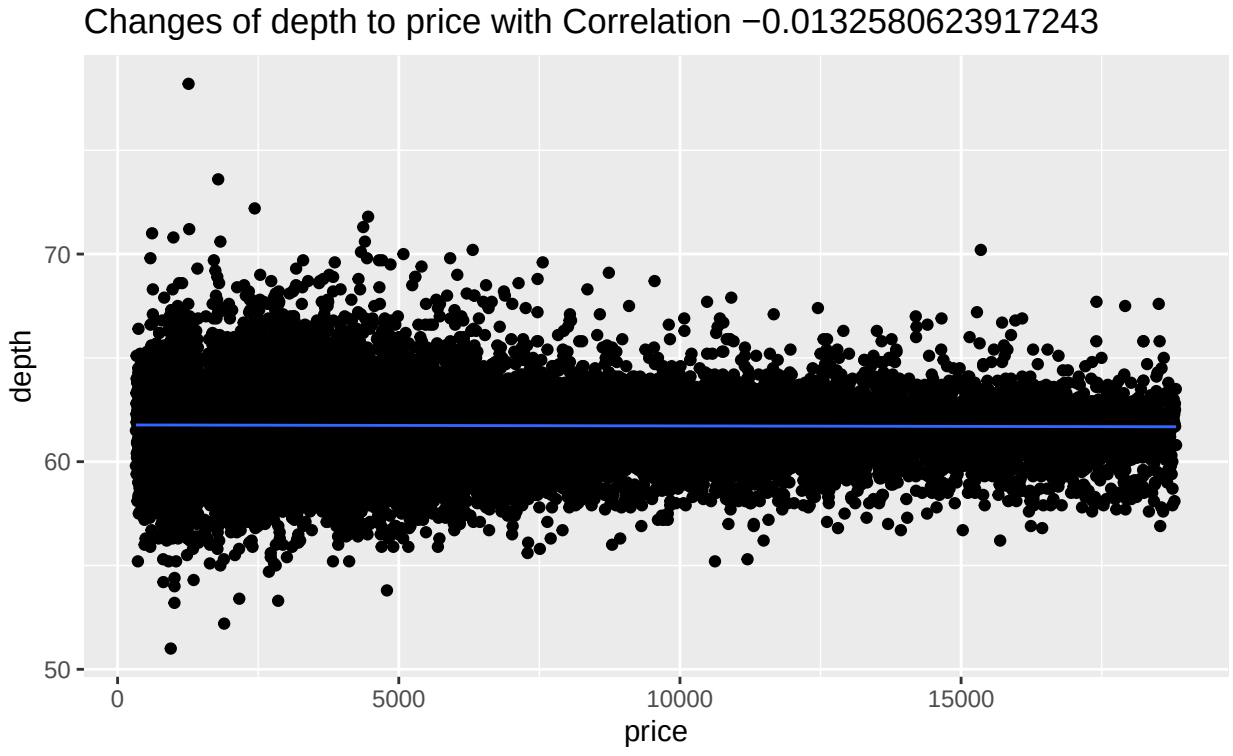

```
plot_against(diam, 'price', 'carat')
```



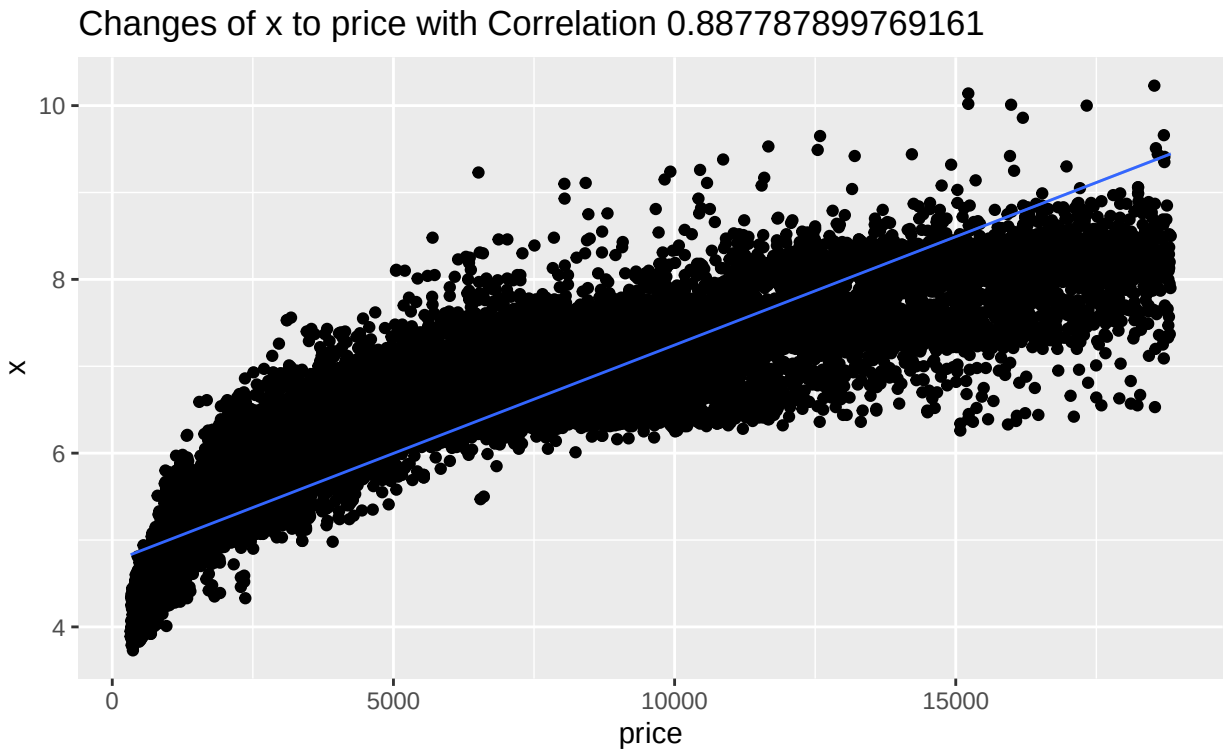

```
plot_against(diam, 'price', 'table')
```



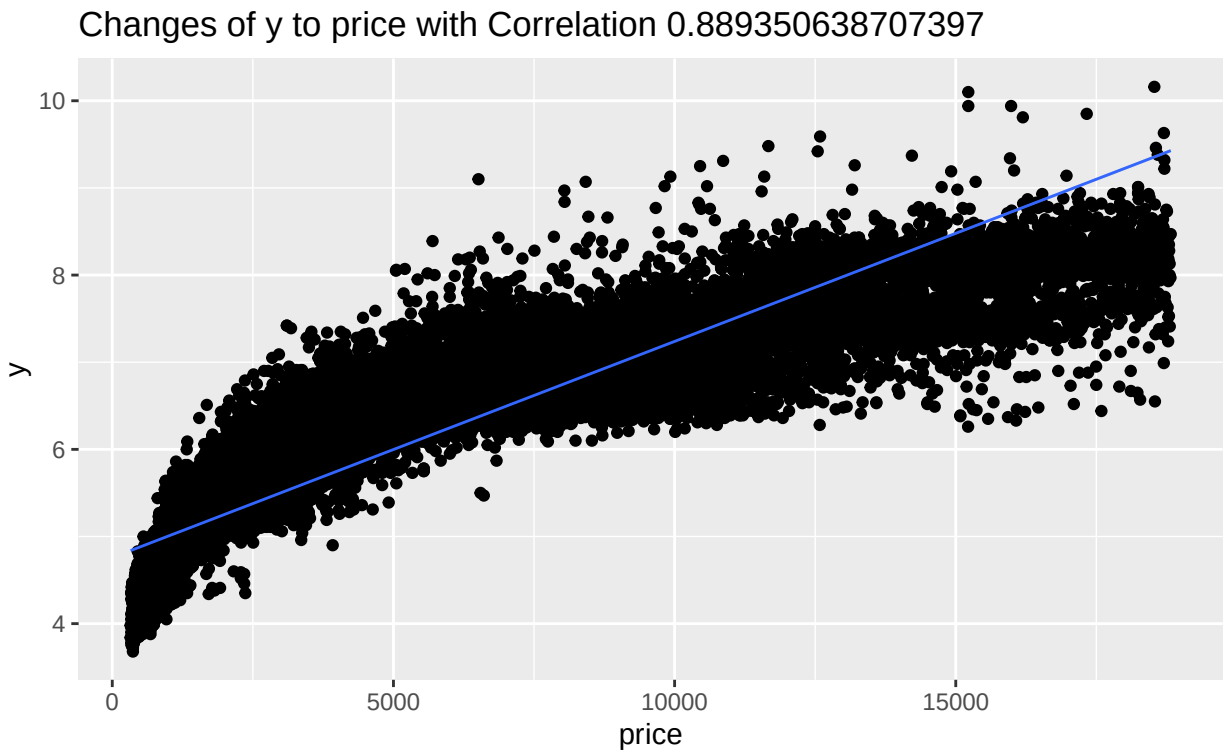

```
plot_against(diam, 'price', 'depth')
```



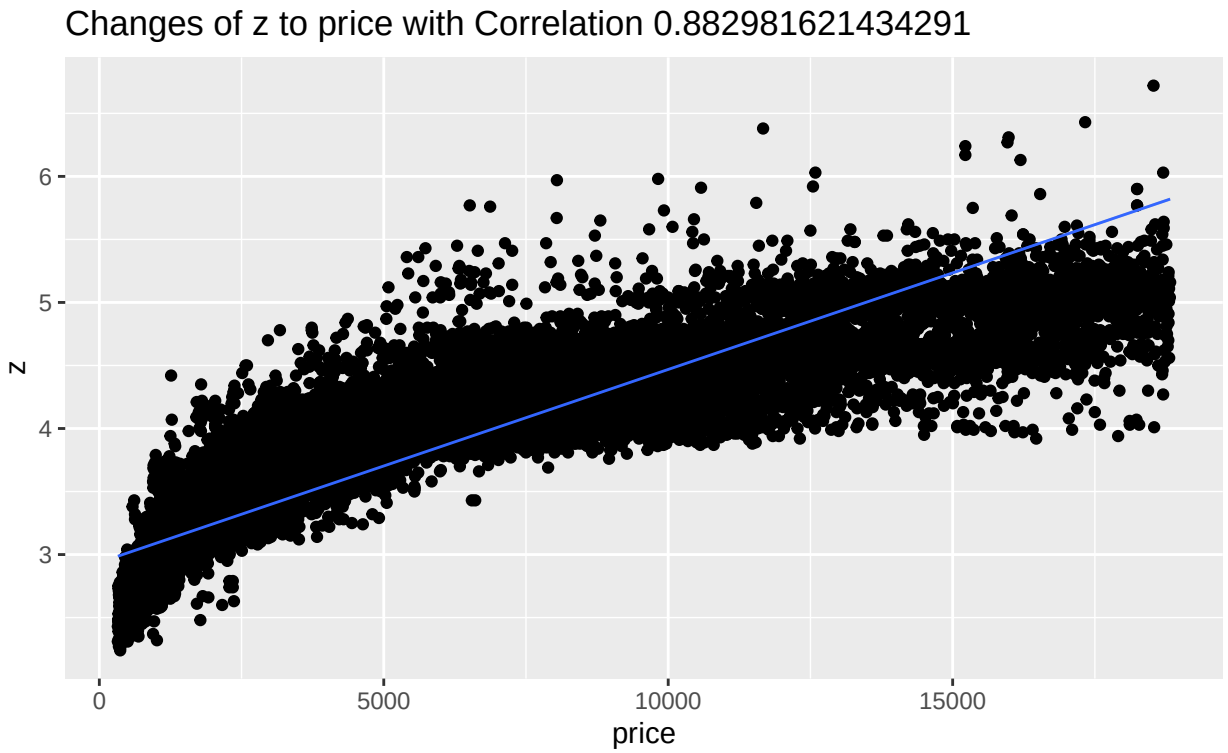

```
plot_against(diam, 'price', 'x')
```




```
plot_against(diam, 'price', 'y')
```




```
plot_against(diam, 'price', 'z')
```



As you can see from correlation results, the x, y, z and carat having more than 80% direct correlation. Also cut and color and clarity has some relation as we can see from the filled plot but I didn't find a method to calculate it by number.

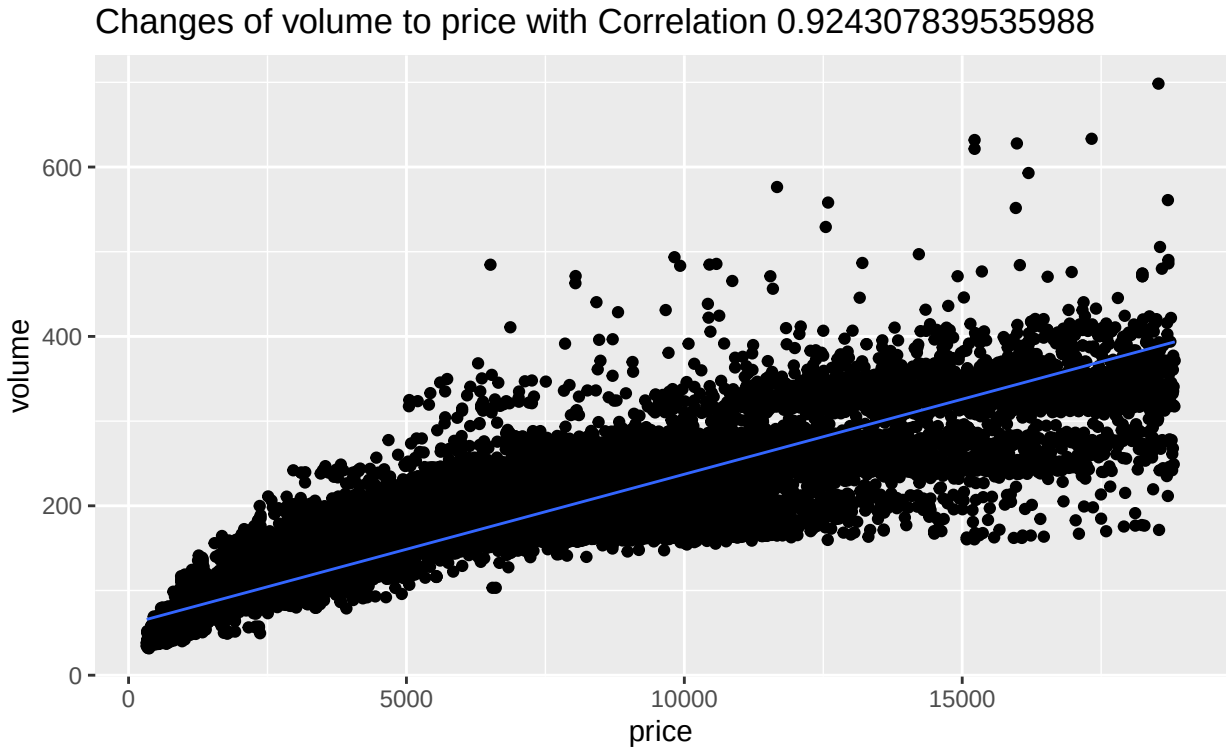
Analysis of Carat, Depth, Table, X, Y and Z

In this part we only focus on the carat, depth, table, x, y and z variables and compare those variables in more details and try to find out the relations between them.

Volume computation and comparison with price

Let's find the volume by multiplying dimensions and plot it against price.

```
diam$volume <- diam$x * diam$y * diam$z
plot_against(diam, 'price', 'volume')
```

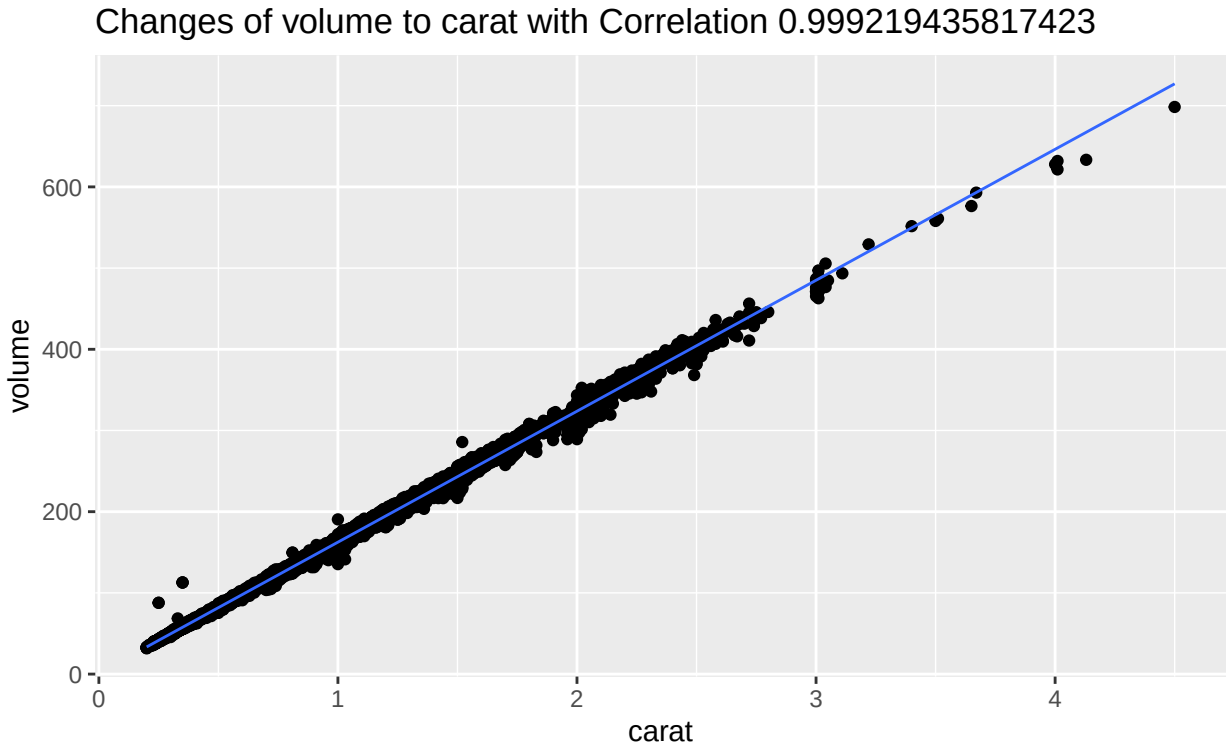


As we can see there is strong relation between volume and price. It was expected as x, y and z were also correlated.

Comparison of Volume with carat

Now, in this time let's compare it with carat and find the correlation:

```
plot_against(diam, 'carat', 'volume')
```

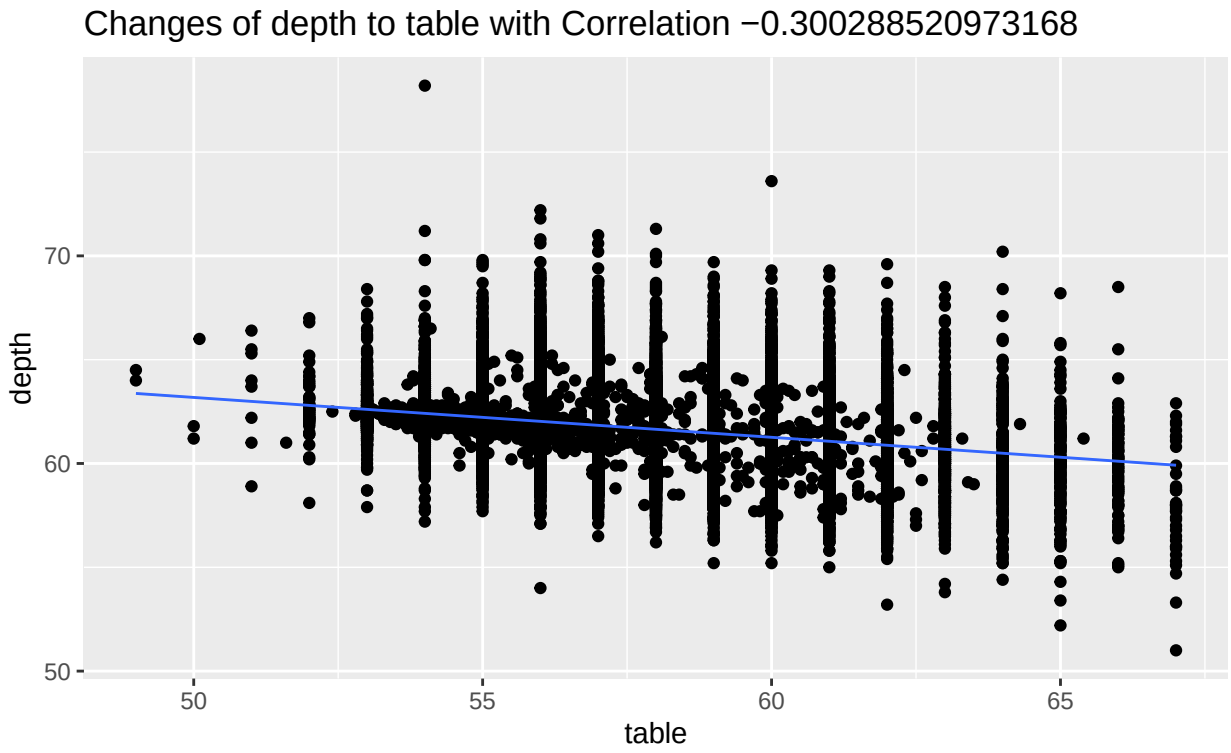


As we expected the carat and volume are correlated so strong.

Relation between table and depth

We shall find some relations between table and depth if plot them against each other and calculate the correlation:

```
plot_against(diam, 'table', 'depth')
```

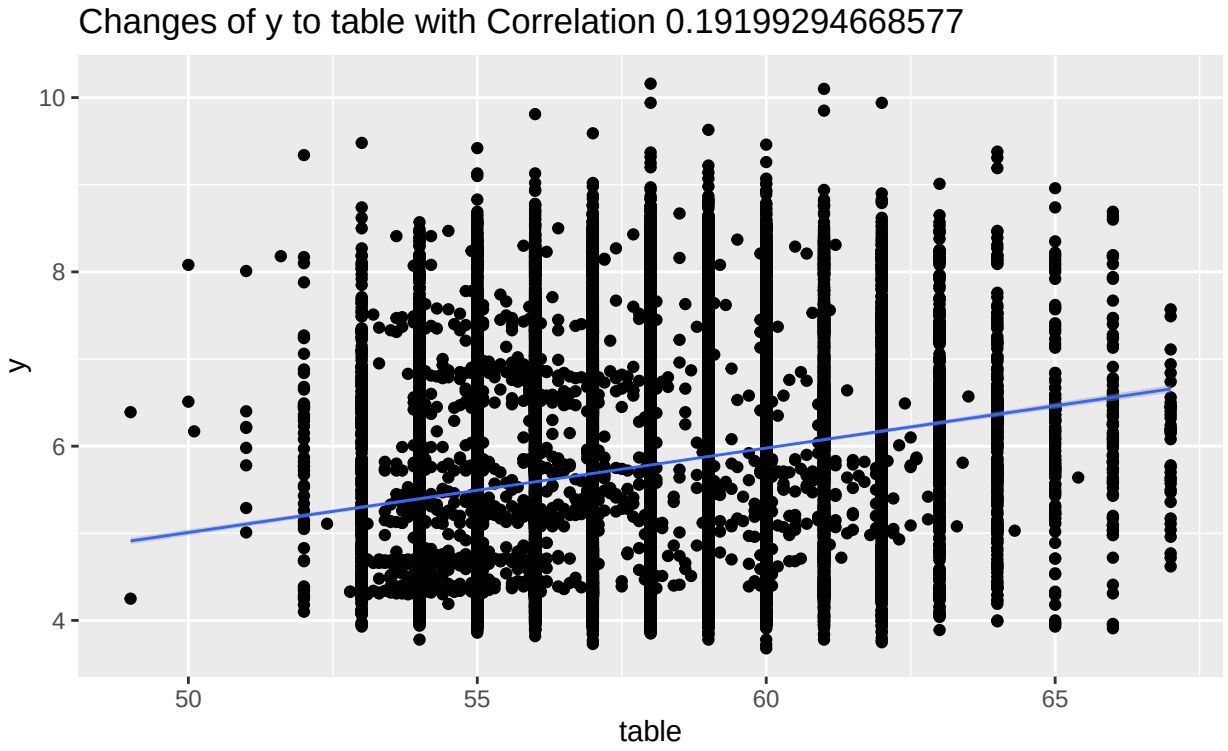


With the result calculation of correlation, we can say that they are weak correlated.

Relation between table and other variables

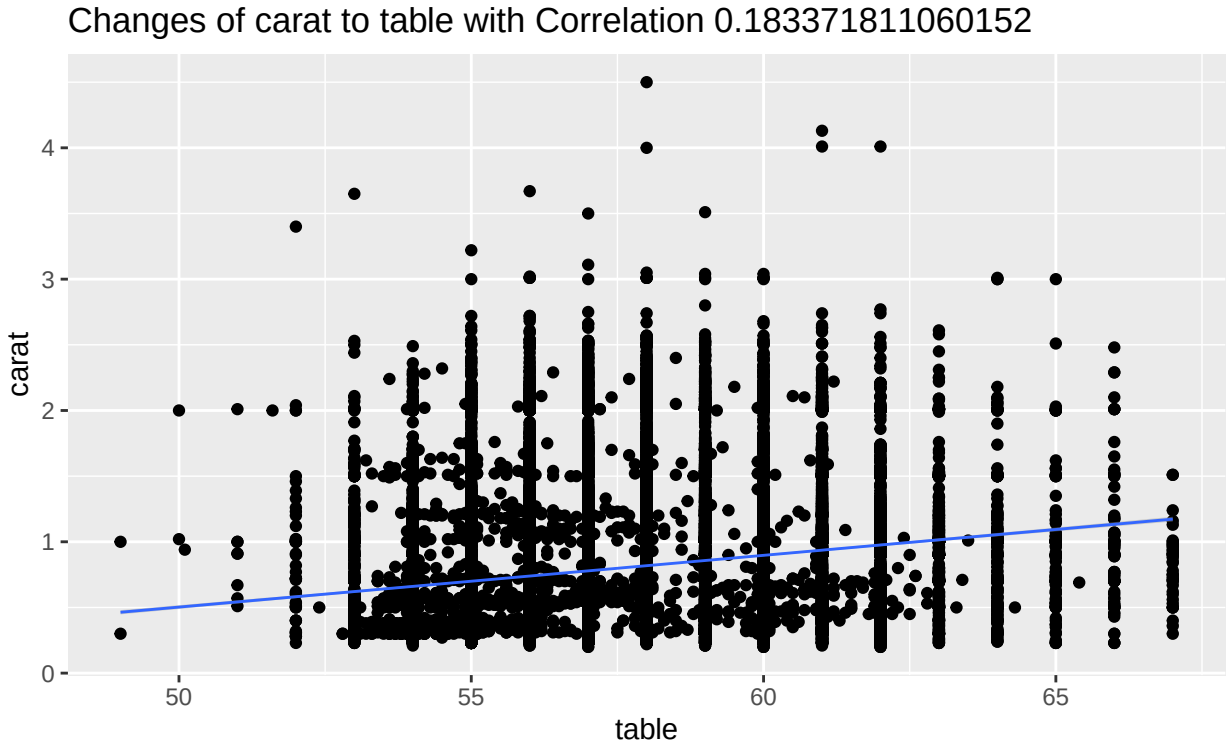
Let's first try to find a relation between our best guess and that's Y. Because table and Y are both is a variable of width. So let's start.

```
plot_against(diam, 'table', 'y')
```

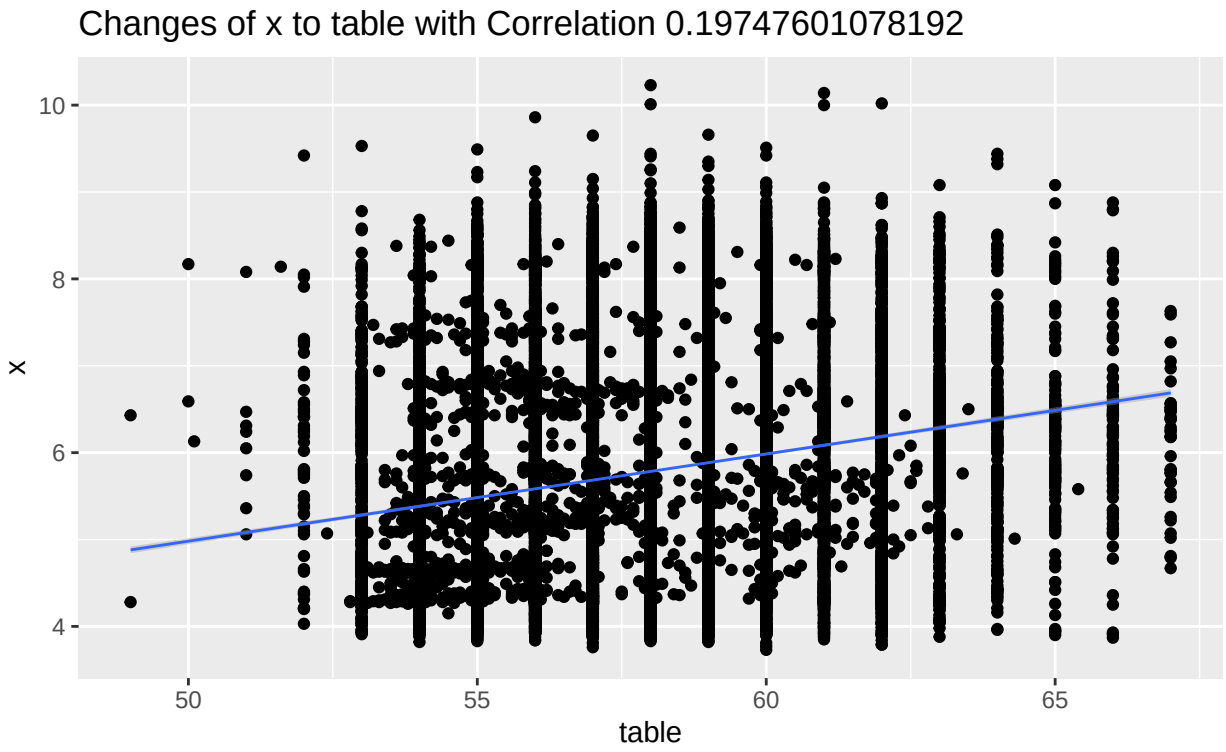


There is no correlation with our best guess and it seems that we will not find any relation with other variables. But let's just validate the latter statement by calculating the correlation.

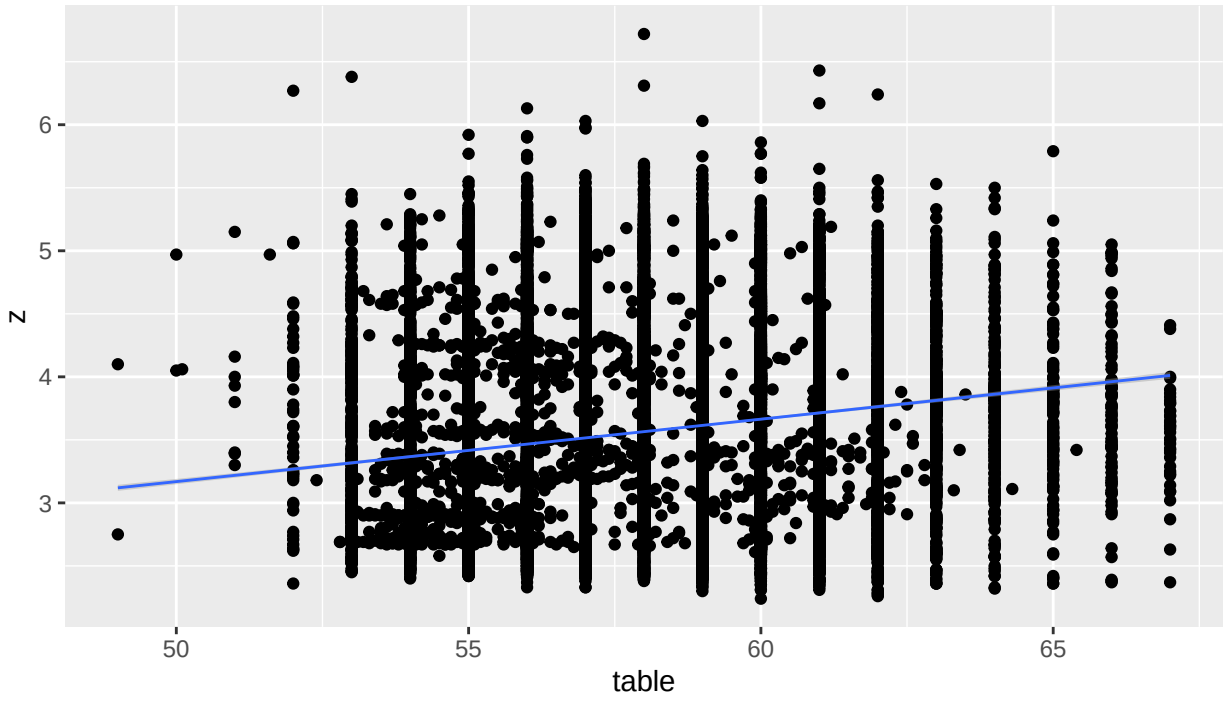
```
plot_against(diam, 'table', 'carat')
```



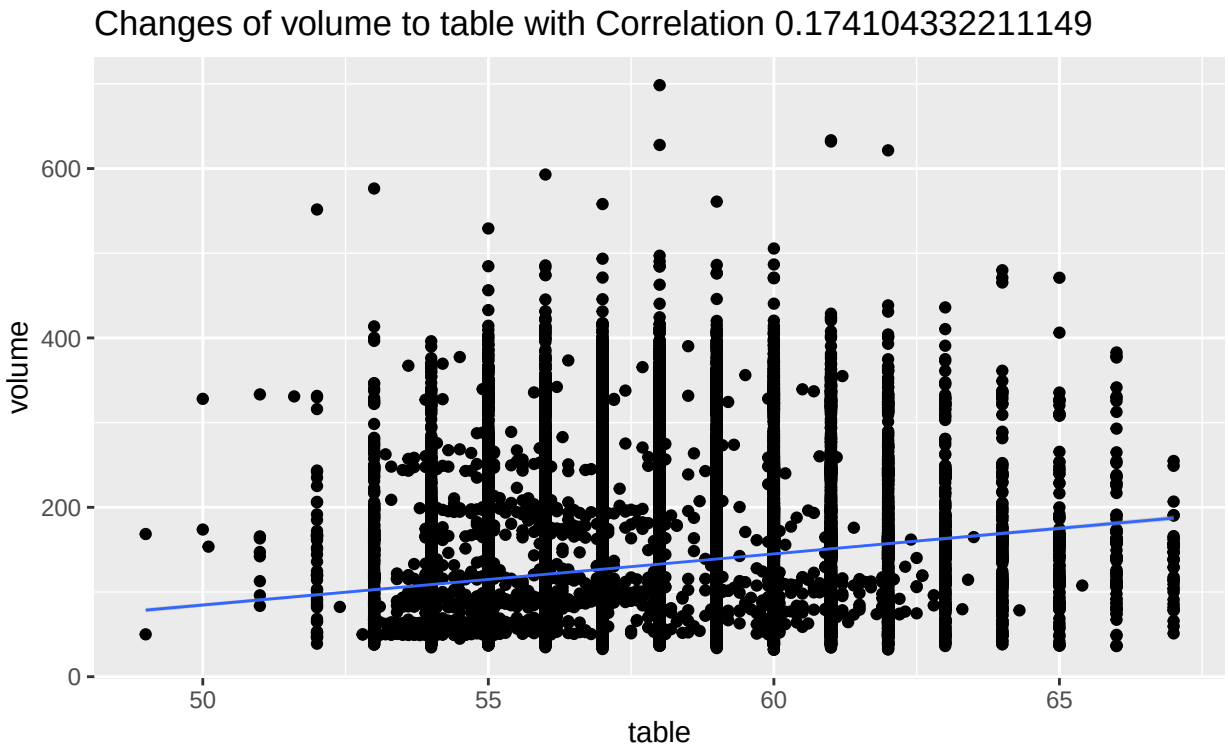
```
plot_against(diam, 'table', 'x')
```




```
plot_against(diam, 'table', 'z')
```



```
plot_against(diam, 'table', 'volume')
```



And we see, there is no correlation from the results.

Source Code

```
knitr::opts_chunk$set(warning=FALSE, message=FALSE)

install.packages("mice")
install.packages("ggplot2")
install.packages("stringr")
install.packages('dplyr')
install.packages('gridExtra')
install.packages('stats')
library('stats')
library("gridExtra")
library("stringr")
library("ggplot2")
library("mice")
library("dplyr")

apply.depth.formula <- function(df) {
  df <- subset(df, abs(depth - (((
    2 * z
  ) / (
    x + y
  )) * 100)) <= 0.5)
}
```

```
apply.carat.formula <- function(df) {
  df <- subset(df, df$carat >= 0.2 & df$carat <= 5.01)
}
ggtable.plot <- function(df , column, main_title) {
  p <-
    ggplot(df, aes(df[, column])) + ggtitle(paste(column, main_title)) +  xlab(column) + geom_histogram



  return(p)
}

table.plot <- function(df , column, main_title) {
  plot(
    table(df[, column]),
    main = paste(column, main_title),
    ylab = "count",
    xlab = column
  )
}

interquantile.method <- function(df , column, sieve_size = 3) {
  Q1 <- quantile(df[, column], .25)
  Q3 <- quantile(df[, column], .75)
  IQR <- IQR(df[, column])
  table.plot(df, column, "frequency BEFORE \n removing outliers")
  bef_dim <- dim(df)

  df <-
    subset(df, df[, column] > (Q1 - sieve_size * IQR) &
             df[, column] < (Q3 + sieve_size * IQR))

  table.plot(df, column, "frequency AFTER \n removing outliers")
  aft_dim <- dim(df)
  diff <- bef_dim - aft_dim
  print(paste(diff[1], "of row deleted AFTER removing", column, "outliers."))
  return(df)
}

plot_against <- function(df, a, b, quanitative = TRUE) {
  if (quanitative == FALSE) {
    ggplot(df, aes(df[, a], colour=df[, b])) + xlab(a) +   labs(colour=b) + ggtitle(paste(b, "Versus",
  } else {
    corl <- cor(df[, a], df[, b])
    ggplot(data = df, aes(df[, a], df[, b])) + geom_point() + geom_smooth(method = "lm", linewidth =
                                                                            .5) + labs(
                                                                              title = paste("Changes
                                                                              x = a,
                                                                              y = b,

                                                                            )
  }
}
```

```
md.pattern.customized <-
  function(x,
           plot = TRUE,
           rotate.names = FALSE) {
    if (!(is.matrix(x) || is.data.frame(x))) {
      stop("Data should be a matrix or dataframe")
    }
    if (ncol(x) < 2) {
      stop("Data should have at least two columns")
    }
    R <- is.na(x)
    nmis <- colSums(R)
    # sort columnwise
    R <- matrix(R[, order(nmis)], dim(x))
    pat <-
      apply(R, 1, function(x)
        paste(as.numeric(x), collapse = ""))
    # sort rowwise
    sortR <- matrix(R[order(pat),], dim(x))
    if (nrow(x) == 1) {
      mpat <- is.na(x)
    } else {
      mpat <- sortR[!duplicated(sortR),]
    }
    # update row and column margins
    if (all(!is.na(x))) {
      cat(" /\\     /\\\n{  `---'  }\n{  O   O  }\n==>  V <==")
      cat("  No need for mice. This data set is completely observed.\n")
      cat(" \\  \\|/  /\n  `-----'\n\n")
      mpat <- t(as.matrix(mpat, byrow = TRUE))
      rownames(mpat) <- table(pat)
    } else {
      if (is.null(dim(mpat))) {
        mpat <- t(as.matrix(mpat))
      }
      rownames(mpat) <- table(pat)
    }
    r <- cbind(abs(mpat - 1), rowSums(mpat))
    r <- rbind(r, c(nmis[order(nmis)], sum(nmis)))
    if (plot) {
      op <- par(mar = rep(0, 4))
      on.exit(par(op))
      plot.new()
      if (is.null(dim(sortR[!duplicated(sortR),]))) {
        R <- t(as.matrix(r[1:nrow(r) - 1, 1:ncol(r) - 1]))
      } else {
        if (is.null(dim(R))) {
          R <- t(as.matrix(R))
        }
        R <- r[1:nrow(r) - 1, 1:ncol(r) - 1]
      }
      if (rotate.names) {
        adj <- c(0, 0.5)
        srt <- 90
```

```
        length_of_longest_colname <- max(nchar(colnames(r))) / 2.6
        plot.window(
          xlim = c(-1, ncol(R) + 1),
          ylim = c(-1, nrow(R) + length_of_longest_colname),
          asp = 1
        )
      } else {
        adj <- c(0.5, 0)
        srt <- 0
        plot.window(
          xlim = c(-1, ncol(R) + 1),
          ylim = c(-1, nrow(R) + 1),
          asp = 1
        )
      }
      M <- cbind(c(row(R)), c(col(R))) - 1
      shade <- ifelse(R[nrow(R):1,], mdc(1), mdc(2))
      rect(M[, 2], M[, 1], M[, 2] + 1, M[, 1] + 1, col = shade)
      for (i in 1:ncol(R)) {
        text(i - .5,
             nrow(R) + .3,
             colnames(r)[i],
             adj = adj,
             srt = srt)
        text(i - .5,-1, nmis[order(nmis)][i], srt = 90)
      }
      for (i in 1:nrow(R)) {
        text(ncol(R) + .3, i - .5, r[(nrow(r) - 1):1, ncol(r)][i], adj = 0)
        text(-.3, i - .5, rownames(r)[(nrow(r) - 1):1][i], adj = 1)
      }
      return(r)
    } else {
      return(r)
    }
  }

# Cleaning NA's, Outliers and Incorrect Values from Diamonds
diamond_raw <- read.csv("DiamondData.csv")
devnull <- md.pattern.customized(diamond_raw, rotate.names = TRUE)

## Fill missing data
diam <- mice(diamond_raw, method = 'pmm', printFlag = FALSE, )
diam <- complete(diam)
devnull <- md.pattern.customized(diam, rotate.names = TRUE)

## Viewing Tables to understand More about Diamonds dataset
### Detective eye On 'Cut' column
table.plot(diam, 'cut', "Table Plot")

diam$cut <- str_replace(diam$cut, "Very Geod", "Very Good")
table.plot(diam, 'cut', "Table Plot")

### Detective eye On 'Color' column
table.plot(diam, 'color', "Table Plot")
```

```
### Detective eye On 'Clarity' column
table.plot(diam, 'clarity', "Table Plot")

### Detective eye on 'Depth' column
par(mfrow = c(1, 2))
p1 <- table.plot(diam, 'depth', "BEFORE applying\n depth Formula")
diam <- apply.depth.formula(diam)
p2 <- table.plot(diam, 'depth', "AFTER applying\n depth Formula")

### Detective eye on 'Carat' column
par(mfrow = c(1, 2))
p1 <-
  table.plot(diam, 'carat', "BEFORE removing \n invalid ranges ")
diam <- apply.carat.formula(diam)
p2 <- table.plot(diam, 'carat', "AFTER removing \n invalid ranges")

### Detective eye on 'Table' column
par(mfrow = c(1, 2))
diam <- interquantile.method(diam, "table")

### Detective eye on 'x, y and z' column
par(mfrow = c(1, 2))
diam <- interquantile.method(diam, "x")
diam <- interquantile.method(diam, "y")
diam <- interquantile.method(diam, "z")

### Detective eye on 'Price' column
diam <- interquantile.method(diam, "price", sieve_size = 4)

# Summary of Variables
summary(diam)

## Pie-chart for cut column
pie(table(diam$cut))

## Bar-chart for color
barplot(table(diam$color))

## Histogram for clarity
ggtable.plot(diam, 'clarity', "Histogram")

# Analysis of Price
## Histogram and Summarizing Price Column
summary(diam$price)
price_histogram <- ggplot(diam, aes(x = price)) + geom_histogram()
ggtable.plot(diam, 'price', "Histogram")
price_variance <- var(x = diam$price)
standard_dev <- sqrt(price_variance)
standard_dev

## Group Diamonds by Price Column
diam$price_group <- with(diam, factor(
  findInterval(price, c(-Inf, quantile(
```

```
    price, probs = c(0.25, .5, .75)
  ), Inf)),
  labels = c("Low", "Medium", "High", "Very High")
))
tail(diam)
summarise(
  group_by(diam, price_group),
  mean = mean(price),
  standard_deviation = sd(price),
  min_value = min(price),
  max_value = max(price),
  count = n()
)

## Prices Versus Cuts
boxplot(price ~ cut, data = diam, main="Price Versus Cut")
summarise(
  group_by(diam, cut),
  mean = mean(price),
  standard_deviation = sd(price),
  min_value = min(price),
  max_value = max(price),
  count = n()
)

## Columns Which Correlated with Price
plot_against(diam, 'price', 'cut', quanitative = FALSE);
plot_against(diam, 'price', 'color', quanitative = FALSE);
plot_against(diam, 'price', 'clarity', quanitative = FALSE);
plot_against(diam, 'price', 'carat')
plot_against(diam, 'price', 'table')
plot_against(diam, 'price', 'depth')
plot_against(diam, 'price', 'x')
plot_against(diam, 'price', 'y')
plot_against(diam, 'price', 'z')

# Analysis of Carat, Depth, Table, X, Y and Z
## Volume computation and comparison with price
diam$volume <- diam$x * diam$y * diam$z
plot_against(diam, 'price', 'volume')

## Comparison of Volume with carat
plot_against(diam, 'carat', 'volume')

## Relation between table and depth
plot_against(diam, 'table', 'depth')

## Relation between table and other variables
plot_against(diam, 'table', 'y')
```