Diamonds

Francisco Arrieta, Emily Schmidt and Lucia Camenisch

2022-12-07

```
library(data.table)     #for reading data.tables
library(kableExtra)     #for more elaborate tables
library(ggplot2)        #for making graphs
library(GGally)         #for making graphs
library(dplyr)          #for data manipulation
library(tidyr)          #for changing the shape and hierarchy of a data set
library(DataExplorer)   #for graphing missing value percentages
library(car)            #for statistic functions

source("VIF.R")
```

Data Exploration

```
diamonds <- fread("diamonds.csv", sep=",", header = T) # Load your data, diamonds.csv

diamonds$V1 <- NULL # Remove column 'V1' as it is similar to an ID variable - no additional meaning der

# Rename columns for more precise names
colnames(diamonds)[5] <- "depth_ratio" # depth to depth_ratio
colnames(diamonds)[8] <- "length" # x to length
colnames(diamonds)[9] <- "width"  # y to width
colnames(diamonds)[10] <- "depth" # z to depth

# add variable to confirm the depth ratio
#explain the 2 times x in the depth formula

dim(diamonds) # Dimensions of data
```

```
## [1] 53940    10
```

```
summary(diamonds) # Produce result summaries of all variables
```

```
##      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_ratio        table           price           length
##  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  
##      width            depth       
##  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  
```

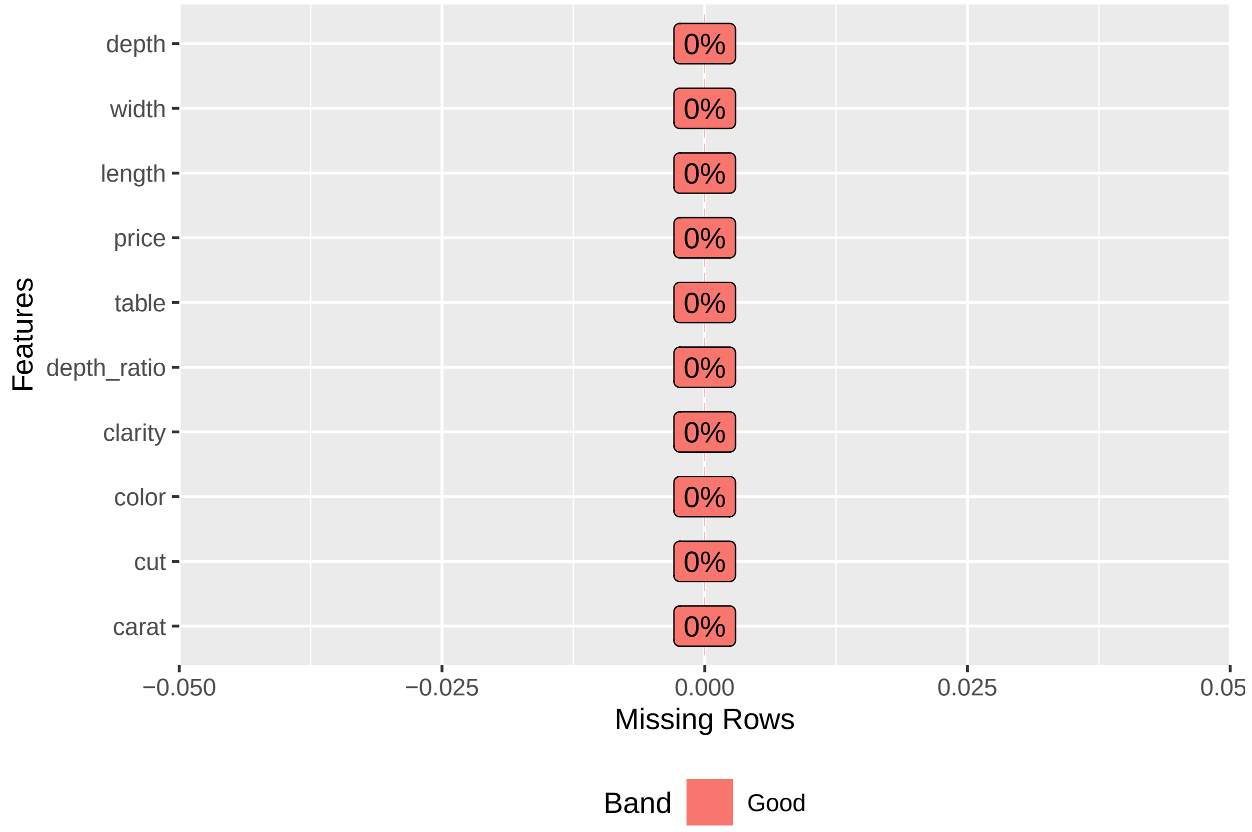
```
str(diamonds) # Type of variables
```

```
## Classes 'data.table' and '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_ratio: 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 ...
##  $ length     : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ width      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ depth      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Number of unique values in each variable
sapply(diamonds, function(x) length(unique(x)))
```

```
##       carat         cut       color     clarity depth_ratio       table 
##         273           5           7           8         184         127 
##       price      length       width       depth 
##       11602         554         552         375 
```

```
# Missing values analysis
plot_missing(diamonds) # Plots the percentages of missing values
```



```
# pairs(diamonds[, c(1, 5:10)])
```

```
# carat no problems


unique(diamonds$cut) # Review unique values for cut
```

```
## [1] "Ideal"     "Premium"   "Good"      "Very Good" "Fair"
```

```
diamonds$cut <- as.factor(diamonds$cut) # Factor the cut to five levels
diamonds$cut <- ordered(diamonds$cut, levels = c("Fair", "Good", "Very Good", "Premium", "Ideal")) # Or

unique(diamonds$color) # Review unique values for color
```

```
## [1] "E" "I" "J" "H" "F" "G" "D"
```

```
diamonds$color <- as.factor(diamonds$color) # Factor the color to seven levels
diamonds$color <- ordered(diamonds$color, levels = c("J", "I", "H", "G", "F", "E", "D")) # Ordered from

unique(diamonds$clarity) # Review unique values for clarity
```

```
## [1] "SI2"  "SI1"  "VS1"  "VS2"  "VVS2" "VVS1" "I1"   "IF"
```

```
diamonds$clarity <- as.factor(diamonds$clarity) # Factor the clarity to eight levels
diamonds$clarity <- ordered(diamonds$clarity, levels = c("I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS

# table is ok

# price is ok
# Remove values of 0 for for dimensions which includes zeros in length and width
nrow(diamonds[depth %in% 0,]) # Remove 20 rows due to depth = 0.0
```

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

```
diamonds <- diamonds[depth > 0, ] # Include only values with depth greater than zero

# Create formula to check the absolute value of length to width, comparison
diamonds[, subtraction := abs(length - width)]
nrow(diamonds[subtraction>10,]) # Remove 2 rows due their extreme subtraction value (~59 and ~26)
```

```
## [1] 2
```

```
diamonds <- diamonds[subtraction <= 10, ] # Include only values with subtraction less than ten

diamonds[, depth_check := round(100*(2*depth)/((length + width)), 1)]
diamonds[, diff := abs(depth_check-depth_ratio)]
# treshold at 0.3? anastasia
nrow(diamonds[diff > 0.3,]) # we remove 268 rows
```

```
## [1] 253
```

```
diamonds <- diamonds[diff <= 0.3,]
# hist(diamonds[diff >= 0.4 & diff < 1, diff], breaks = 50)

# Removed created columns needed to clean the data
diamonds[, subtraction := NULL]
diamonds[, depth_check := NULL]
diamonds[, diff := NULL]
# Total rows remove: 275 observations
```

```
# Reorder data table to group like variable types
diamonds <- diamonds[, c(7, 2:4, 1, 8:10, 5:6)]
```

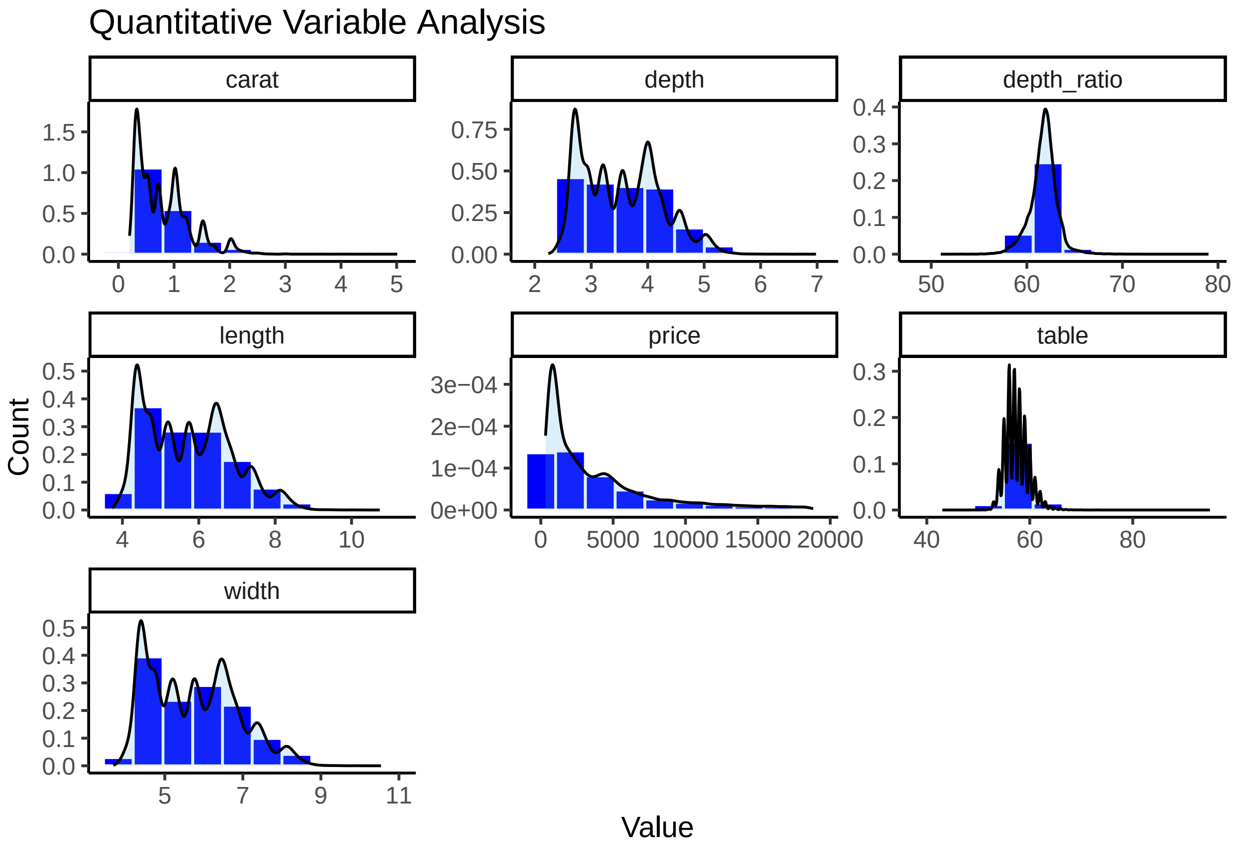
```
# Used ggpairs to create a scatterplot matrix
# ggpairs(diamonds[, c(1, 5:10)], title = "Scatterplot Matrix",
#           proportions = "auto",
#           columnLabels = c("Price", "Carat", "Length", "Width", "Depth","Depth Ratio","Table"),
#           upper = list(continuous = wrap('cor',size = 3)),) + theme_light()
```

```
diamonds %>% gather() %>% head() # Reshaping the data which means it collects a set of column names and
```

```
##     key value
## 1 price   326
## 2 price   326
## 3 price   327
## 4 price   334
## 5 price   335
## 6 price   336
```

```
histograms <- ggplot(gather(data = diamonds[, c(1, 5:10)]),aes(value)) +
  geom_histogram(aes(y=..density..),bins = 10, color = "white", fill = "blue") + # Creates bin sizing a
  geom_density(alpha= .2, fill="#56B4E9") +
  facet_wrap(~key,scales = "free") + # Converting the graphs into panels
  ggtitle("Quantitative Variable Analysis") + # Title name
  ylab("Count") + xlab("Value") + # Label names
  theme_classic() # A classic theme, with x and y axis lines and no grid lines
```

```
histograms
```

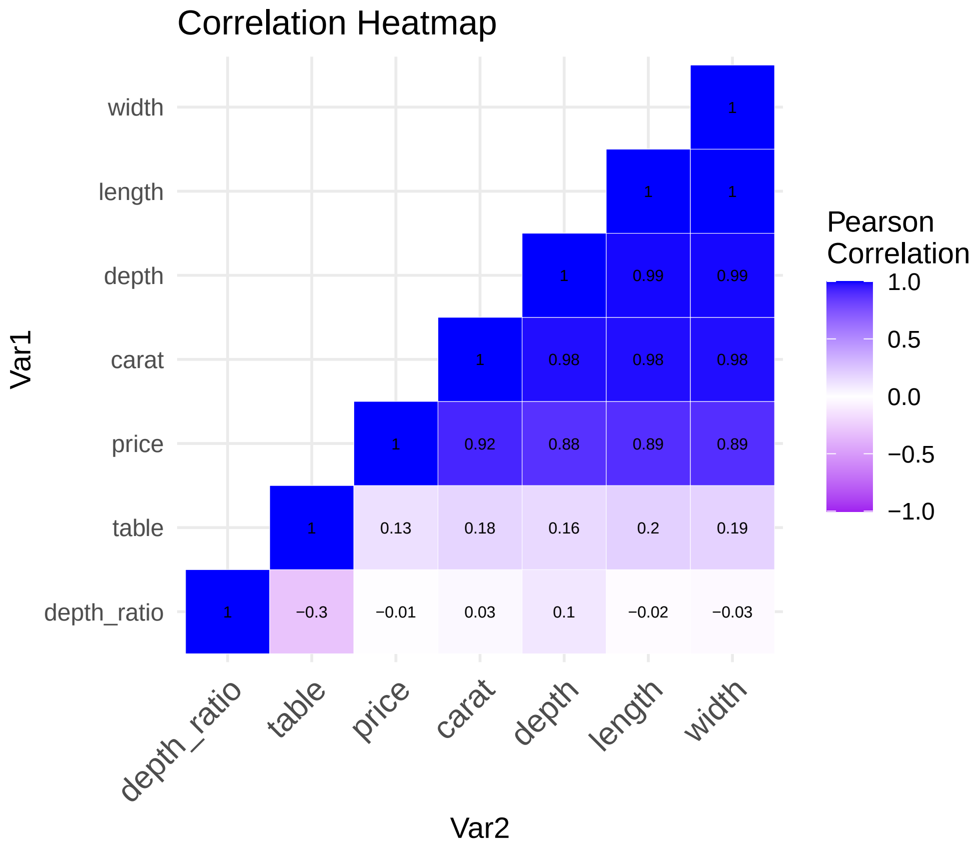


```
# Create heatmap to show variable correlation
cormat <- round(cor(diamonds[, c(1, 5:10)]),2) # Round the correlation coefficient to two decimal place

melted_cormat <- melt(cormat) # One way to reshape and elongate the data frame

# Get upper triangle of the correlation matrix
  get_upper_tri <- function(cormat){
    cormat[lower.tri(cormat)]<- NA
    return(cormat)
  }

# Rename the correlation coefficient value
upper_tri <- get_upper_tri(cormat)
upper_tri
```

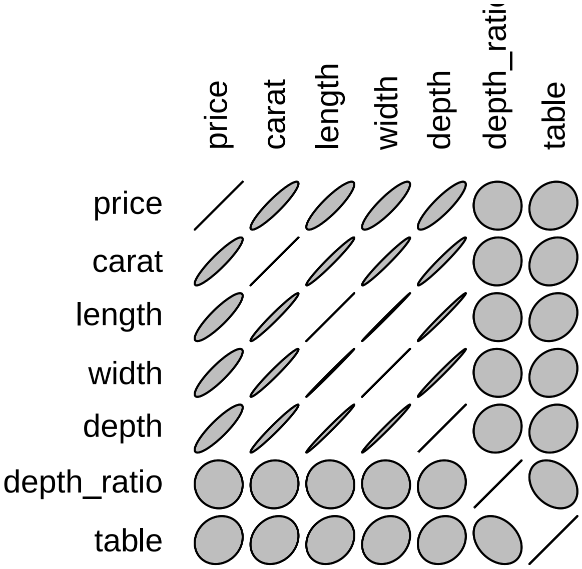
```
##             price carat length width depth depth_ratio table
## price           1  0.92   0.89  0.89  0.88       -0.01  0.13
## carat          NA  1.00   0.98  0.98  0.98        0.03  0.18
## length         NA    NA   1.00  1.00  0.99       -0.02  0.20
## width          NA    NA     NA  1.00  0.99       -0.03  0.19
## depth          NA    NA     NA    NA  1.00        0.10  0.16
## depth_ratio    NA    NA     NA    NA    NA        1.00 -0.30
## table          NA    NA     NA    NA    NA          NA  1.00
```

```
# Use correlation between variables as distance
reorder_cormat <- function(cormat){
```

```
dd <- as.dist((1-cormat)/2)
hc <- hclust(dd)
cormat <-cormat[hc$order, hc$order]
}

# Reorder the correlation matrix
cormat <- reorder_cormat(cormat)
upper_tri <- get_upper_tri(cormat)

# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)

# Create a ggheatmap with multiple characteristics
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "purple", high = "blue", mid = "white", midpoint = 0, limit = c(-1,1), spa
  ggtitle("Correlation Heatmap") + # Title name
  theme_minimal() + # Minimal theme, keeps in the lines
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1)) +
  coord_fixed() +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 2)

# Print the heat map
print(ggheatmap)
```



```
library(ellipse)
plotcorr(cor(diamonds[, -c(2:4)]))
```



```r
# set seed for reproducing the partition
set.seed(111)

# generating training set index
train.index <- sample(c(1:nrow(diamonds)), 0.5*nrow(diamonds))
# generating validation set index taken from the complementary of training set
valid.index <- sample(setdiff(c(1:nrow(diamonds)), train.index), 0.3*nrow(diamonds))
# defining test set index as complementary of (train.index + valid.index)
test.index <- as.numeric(setdiff(row.names(diamonds), union(train.index, valid.index)))

# creating data tables Train, Valid and Test using the indexes
Train <- diamonds[train.index, ]
Valid <- diamonds[valid.index, ]
Test <- diamonds[test.index, ]
```

```r
# diamonds_lm <- lm(price ~ carat + length + width + depth + depth_ratio + table, data = diamonds)
# diamonds_lm2 <- lm(price ~ carat + depth_ratio + table, data = diamonds)
# diamonds_lm3 <- lm(price ~ carat + depth + depth_ratio + table, data = diamonds)
#
# diamonds_vif <- vif(diamonds_lm)
# VIF(diamonds[, c(5:8)])
# diamonds_vif2 <- vif(diamonds_lm2)
#
# diamonds_vif3 <- vif(diamonds_lm3)
#
#
# summary(diamonds_vif)
```

Predictions

Conclusions

Final Title