Diamonds

Francisco Arrieta, Emily Schmidt and Lucia Camenisch

2022-12-20

Contents


```
library(data.table)     #for reading data.tables
library(ggplot2)        #for making graphs
library(tidyr)          #for changing the shape and hierarchy of a data set
library(ellipse)        #for mapping correlation
library(e1071)          #for skewness
library(caret)          #for preProcess() and accuracy()
library(fastDummies)    #for creating dummies
library(forecast)       # for accuracy() measures
library(kableExtra)     #for more elaborate tables
# library(GGally)         #for making graphs
# library(dplyr)          #for data manipulation
# library(DataExplorer)   #for graphing missing value percentages
# library(car)            #for statistic functions


source("VIF.R")
source("ProcStep.R")
source("GlobalCrit.R")
options(scipen = 999)
```

Data Exploration

```
diamonds <- fread("diamonds.csv") # 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
```

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

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

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

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

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

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

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

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

```r
# Factor the clarity to eight levels
diamonds$clarity <- as.factor(diamonds$clarity)
# Ordered from worst to best
diamonds$clarity <- ordered(diamonds$clarity,
                            levels = c("I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF"))
```

```r
# 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
```

```r
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)]
# Remove 2 rows due their extreme subtraction value (~59 and ~26)
nrow(diamonds[subtraction>10,])
```

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

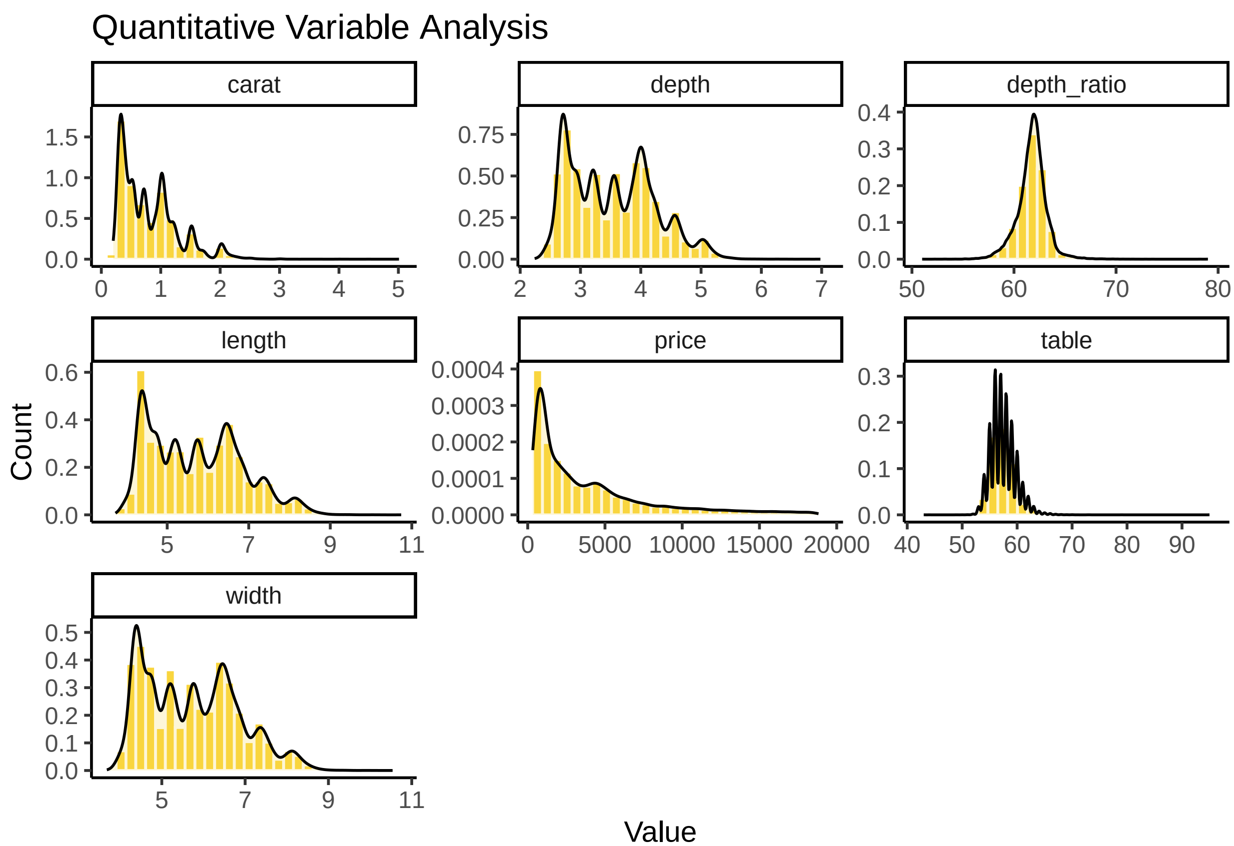
```r
# Include only values with subtraction less than ten
diamonds <- diamonds[subtraction <= 10, ]

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 253 rows
```

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

```r
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 removed: 275 observations
```

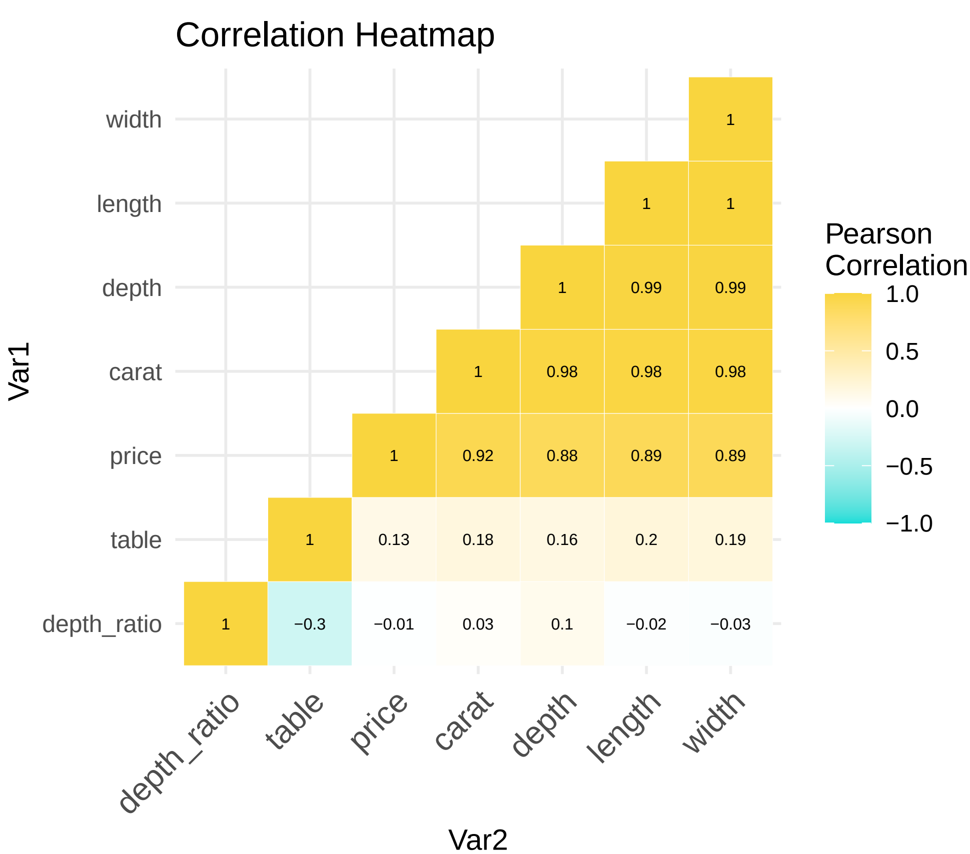
```
# Reorder data table to group like variable types
diamonds <- diamonds[, c(7, 2:4, 1, 8:10, 5:6)]
```

```
ggplot(gather(data = diamonds[, c(1, 5:10)]), aes(value)) +
  geom_histogram(aes(y = after_stat(density)),
#                  bins = 10,
                 color = "white",
                 fill = "#F9D53E") + # Creates bin sizing with colors
  geom_density(alpha = .2, fill = "#F9D53E") +
  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
```



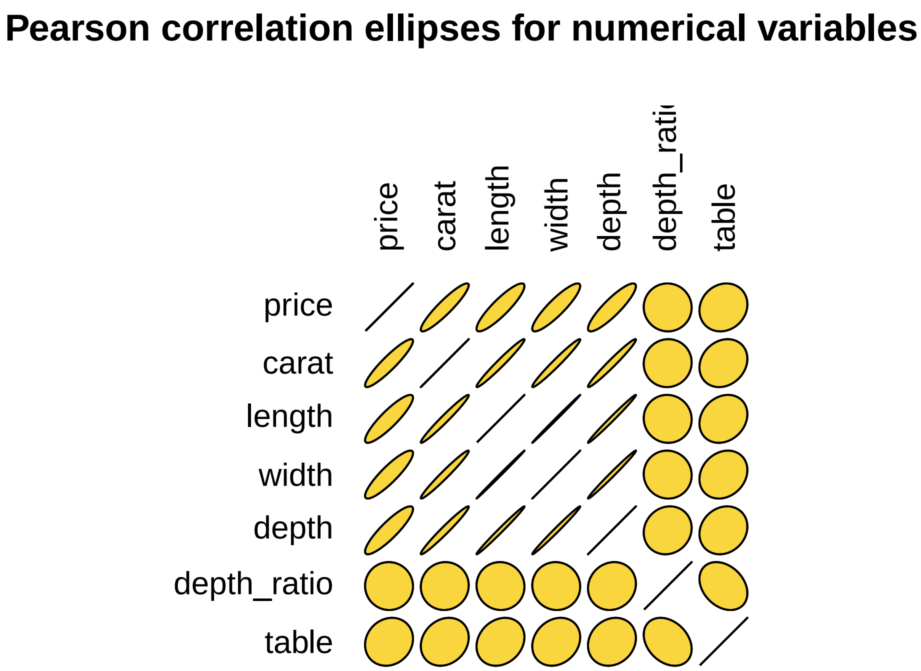

```
# Create heatmap to show variable correlation
# Round the correlation coefficient to two decimal places
cormat <- round(cor(diamonds[, c(1, 5:10)]), 2)

# 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]
return(cormat)
}
```

```r
# Reorder the correlation matrix
cormat <- reorder_cormat(cormat)

# Keeping only upper triangular matrix
# upper_tri returns TRUE/FALSE for each coordinate (TRUE -> part of upper triangle)
# multiplying will thus keep the upper triangle values and set the others to 0
cormat <- cormat*upper.tri(cormat, diag = TRUE)
# Values of the lower triangle (0) are replaced by NA
cormat[cormat == 0] <- NA

# Melt the correlation matrix
cormat <- reshape2::melt(cormat, na.rm = TRUE)

# Create a ggheatmap with multiple characteristics
ggplot(cormat, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "#15DDD8", high = "#F9D53E", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab", name="Pearson\nCorrelation") +
  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)
```




```r
rm(cormat, reorder_cormat)
```

```r
plotcorr(cor(diamonds[, -c(2:4)]), col = "#F9D53E",
         main = "Pearson correlation ellipses for numerical variables")
```




```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.integer(setdiff(row.names(diamonds), union(train.index, valid.index)))
```

Linear Regression

```r
# creating training and validation sets
Train_lr <- diamonds[train.index, ]
Valid_lr <- diamonds[valid.index, ]
```

```r
# using the VIF function from statistical modelling to check multicollinearity of predictors
VIF(y = diamonds$price, matx = diamonds[, -c(1)])
```

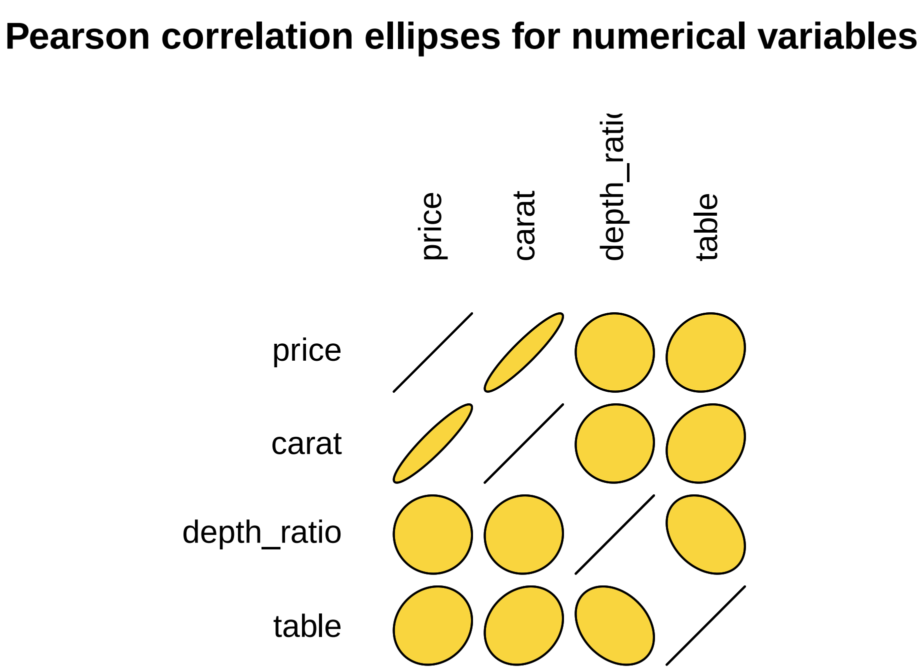
```
##
##                     GVIF Df GVIF^(1/(2*Df))
## cut              2.45522  4         1.11882
## color            1.18398  6         1.01417
```

```
## clarity        1.36848  7         1.02266
## carat         25.91780  1         5.09095
## length      1091.42000  1        33.03670
## width       1143.44000  1        33.81480
## depth       2008.33000  1        44.81440
## depth_ratio   31.99350  1         5.65628
## table          1.80396  1         1.34311
##
##   Mean: 165.105
```

```
# removing length, width and depth and computing VIF without them
VIF(y = diamonds$price, matx = diamonds[, -c(1, 6, 7, 8)])
```

```
##
##                 GVIF Df GVIF^(1/(2*Df))
## cut          1.93382  4         1.08593
## color        1.17045  6         1.01320
## clarity      1.30388  7         1.01913
## carat        1.32381  1         1.15057
## depth_ratio 1.38914  1         1.17862
## table        1.79505  1         1.33980
##
##   Mean: 1.98352
```

```
# plotting correlation ellipses of numerical variables with length, width and depth removed
plotcorr(cor(diamonds[, -c(2:4, 6:8)]), col = "#F9D53E",
         main = "Pearson correlation ellipses for numerical variables")
```



```r
# applying the skewness() function of every numerical variable from our training set
sapply(Train_lr[, c(1, 5:10)], skewness)
```

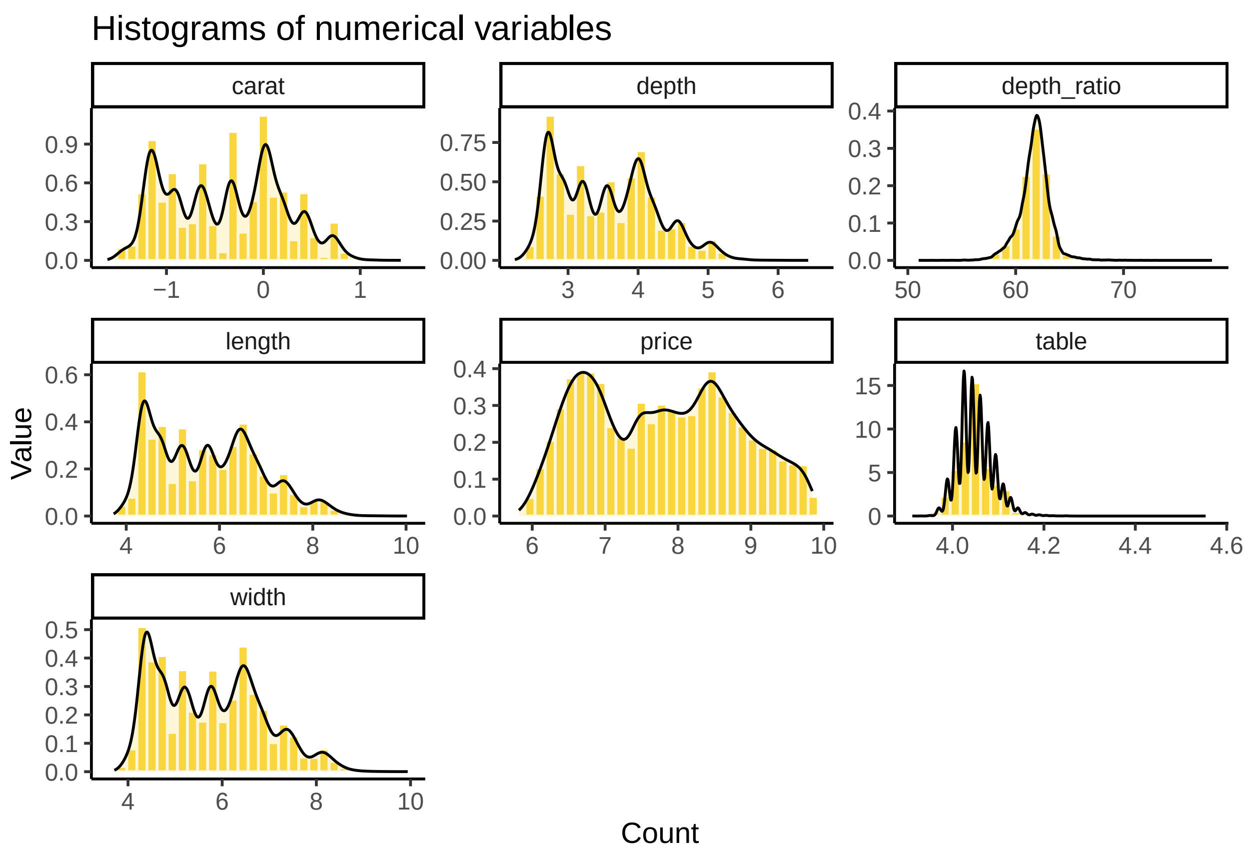
```
##       price       carat      length       width       depth depth_ratio
##  1.62597473  1.09675143  0.39846889  0.39220078  0.39210931  0.01824979
##       table
##  0.87288692
```

```r
# logarithmic transformation on price, carat and table
Train_lr$price <- log(Train_lr$price)
Train_lr$carat <- log(Train_lr$carat)
Train_lr$table <- log(Train_lr$table)

# recomputing the skewness of numerical variables to see the improvement
sapply(Train_lr[, c(1, 5:10)], skewness)
```

```
##       price       carat      length       width       depth depth_ratio
##  0.11305083  0.09668960  0.39846889  0.39220078  0.39210931  0.01824979
##       table
##  0.64541910
```

```r
# transforming in the validation set as well
Valid_lr$price <- log(Valid_lr$price)
Valid_lr$carat <- log(Valid_lr$carat)
Valid_lr$table <- log(Valid_lr$table)
```

```r
# computing the histograms of numerical variables now that they are unskewed

ggplot(gather(data = Train_lr[, c(1, 5:10)]), aes(value)) + # numerical vars of training set
  geom_histogram(aes(y = after_stat(density)),              # making histograms with color params
                 color = "white",
                 fill = "#F9D53E") +
  geom_density(alpha = .2, fill = "#F9D53E") +              # making density lines
  facet_wrap(~ key, scales = "free") +                      # multiple plots with facet
  labs(title = "Histograms of numerical variables",         # labels of the plot
       x = "Count",
       y = "Value") +
  theme_classic()                                           # aesthetic theme
```




```r
# we compute the the mean and std values based on training data (for numerical variables)
norm.values <- preProcess(Train_lr[, c(1, 5:10)], method=c("center", "scale"))

# we standardize the training and validation data
Train_lr[, c(1, 5:10)] <- predict(norm.values, Train_lr[, c(1, 5:10)])
Valid_lr[, c(1, 5:10)] <- predict(norm.values, Valid_lr[, c(1, 5:10)])
```

```r
# we compute the linear model using all predictors and display its summary
LM_complete = lm(price ~. , data = Train_lr)
summary(LM_complete)
```

```
##
## Call:
## lm(formula = price ~ ., data = Train_lr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04059 -0.08291  0.00028  0.08156  1.42487
##
## Coefficients:
##                Estimate Std. Error t value             Pr(>|t|)
## (Intercept) -0.0747373  0.0016289 -45.882 < 0.0000000000000002 ***
## cut.L        0.1168565  0.0037189  31.422 < 0.0000000000000002 ***
## cut.Q       -0.0342102  0.0030248 -11.310 < 0.0000000000000002 ***
## cut.C        0.0174498  0.0027023   6.457       0.0000000001084 ***
## cut^4        0.0003697  0.0020867   0.177              0.85937
## color.L      0.4366728  0.0028463 153.417 < 0.0000000000000002 ***
```

```
## color.Q     -0.0974458  0.0025856 -37.687 < 0.0000000000000002 ***
## color.C      0.0126123  0.0024180   5.216      0.0000001841568 ***
## color^4      0.0134193  0.0022210   6.042      0.0000000015434 ***
## color^5      0.0001336  0.0021044   0.063              0.94940
## color^6      0.0026592  0.0019108   1.392              0.16404
## clarity.L    0.9080235  0.0050105 181.224 < 0.0000000000000002 ***
## clarity.Q   -0.2486235  0.0046594 -53.360 < 0.0000000000000002 ***
## clarity.C    0.1370069  0.0039810  34.415 < 0.0000000000000002 ***
## clarity^4   -0.0685371  0.0031777 -21.568 < 0.0000000000000002 ***
## clarity^5    0.0290689  0.0025896  11.225 < 0.0000000000000002 ***
## clarity^6   -0.0022942  0.0022444  -1.022              0.30670
## clarity^7    0.0312779  0.0019833  15.771 < 0.0000000000000002 ***
## carat        0.9997686  0.0082204 121.621 < 0.0000000000000002 ***
## length       0.1775342  0.0266043   6.673      0.0000000000255 ***
## width       -0.0200231  0.0270778  -0.739              0.45963
## depth       -0.0702733  0.0359774  -1.953              0.05080 .
## depth_ratio  0.0117907  0.0045226   2.607              0.00914 **
## table        0.0011885  0.0010849   1.096              0.27330
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.131 on 26808 degrees of freedom
## Multiple R-squared:  0.9829, Adjusted R-squared:  0.9828
## F-statistic: 6.685e+04 on 23 and 26808 DF,  p-value: < 0.00000000000000022
```

```
# we use iterative search algorithms on the complete model: forward, backward and stepwise
# we display summaries of the three models obtained with iterative methods

LM_forward_complete = step(LM_complete, direction = "forward")
```

```
## Start:  AIC=-109064.2
## price ~ cut + color + clarity + carat + length + width + depth +
##     depth_ratio + table
```

```
summary(LM_forward_complete)
```

```
##
## Call:
## lm(formula = price ~ cut + color + clarity + carat + length +
##     width + depth + depth_ratio + table, data = Train_lr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04059 -0.08291  0.00028  0.08156  1.42487
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) -0.0747373  0.0016289 -45.882 < 0.0000000000000002 ***
## cut.L        0.1168565  0.0037189  31.422 < 0.0000000000000002 ***
## cut.Q       -0.0342102  0.0030248 -11.310 < 0.0000000000000002 ***
## cut.C        0.0174498  0.0027023   6.457      0.0000000001084 ***
## cut^4        0.0003697  0.0020867   0.177              0.85937
## color.L      0.4366728  0.0028463 153.417 < 0.0000000000000002 ***
## color.Q     -0.0974458  0.0025856 -37.687 < 0.0000000000000002 ***
## color.C      0.0126123  0.0024180   5.216      0.0000001841568 ***
```

```
## color^4      0.0134193  0.0022210   6.042      0.0000000015434 ***
## color^5      0.0001336  0.0021044   0.063              0.94940
## color^6      0.0026592  0.0019108   1.392              0.16404
## clarity.L    0.9080235  0.0050105 181.224 < 0.0000000000000002 ***
## clarity.Q   -0.2486235  0.0046594 -53.360 < 0.0000000000000002 ***
## clarity.C    0.1370069  0.0039810  34.415 < 0.0000000000000002 ***
## clarity^4   -0.0685371  0.0031777 -21.568 < 0.0000000000000002 ***
## clarity^5    0.0290689  0.0025896  11.225 < 0.0000000000000002 ***
## clarity^6   -0.0022942  0.0022444  -1.022              0.30670
## clarity^7    0.0312779  0.0019833  15.771 < 0.0000000000000002 ***
## carat        0.9997686  0.0082204 121.621 < 0.0000000000000002 ***
## length       0.1775342  0.0266043   6.673      0.0000000000255 ***
## width       -0.0200231  0.0270778  -0.739              0.45963
## depth       -0.0702733  0.0359774  -1.953              0.05080 .
## depth_ratio  0.0117907  0.0045226   2.607              0.00914 **
## table        0.0011885  0.0010849   1.096              0.27330
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.131 on 26808 degrees of freedom
## Multiple R-squared:  0.9829, Adjusted R-squared:  0.9828
## F-statistic: 6.685e+04 on 23 and 26808 DF,  p-value: < 0.00000000000000022
```

```r
LM_backward_complete = step(LM_complete, direction = "backward")
```

```
## Start:  AIC=-109064.2
## price ~ cut + color + clarity + carat + length + width + depth +
##     depth_ratio + table
##
##                Df Sum of Sq     RSS     AIC
## - width         1      0.01  459.84 -109066
## - table         1      0.02  459.85 -109065
## <none>                       459.83 -109064
## - depth         1      0.07  459.89 -109062
## - depth_ratio   1      0.12  459.95 -109059
## - length        1      0.76  460.59 -109022
## - cut           4     19.38  479.21 -107965
## - carat         1    253.72  713.54  -97277
## - color         6    428.53  888.36  -91407
## - clarity       7    874.13 1333.96  -80501
##
## Step:  AIC=-109065.7
## price ~ cut + color + clarity + carat + length + depth + depth_ratio +
##     table
##
##                Df Sum of Sq     RSS     AIC
## - table         1      0.02  459.86 -109066
## <none>                       459.84 -109066
## - depth         1      0.18  460.02 -109057
## - depth_ratio   1      0.30  460.14 -109050
## - length        1      0.75  460.59 -109024
## - cut           4     19.38  479.22 -107966
## - carat         1    256.52  716.35  -97173
## - color         6    428.96  888.80  -91395
## - clarity       7    877.59 1337.43  -80433
```

```
##
## Step:  AIC=-109066.4
## price ~ cut + color + clarity + carat + length + depth + depth_ratio
##
##               Df Sum of Sq     RSS     AIC
## <none>                      459.86 -109066
## - depth        1      0.18  460.04 -109058
## - depth_ratio  1      0.29  460.15 -109052
## - length       1      0.75  460.61 -109025
## - cut          4     24.58  484.44 -107677
## - carat        1    261.25  721.11  -96998
## - color        6    429.14  889.00  -91391
## - clarity      7    877.84 1337.70  -80430
```

```
summary(LM_backward_complete)
```

```
##
## Call:
## lm(formula = price ~ cut + color + clarity + carat + length +
##     depth + depth_ratio, data = Train_lr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04014 -0.08295  0.00025  0.08161  1.42593
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) -0.0741493  0.0015667 -47.329 < 0.0000000000000002 ***
## cut.L        0.1150923  0.0034602  33.262 < 0.0000000000000002 ***
## cut.Q       -0.0339481  0.0029582 -11.476 < 0.0000000000000002 ***
## cut.C        0.0163231  0.0025452   6.413      0.0000000001447 ***
## cut^4       -0.0001739  0.0020487  -0.085              0.93235
## color.L      0.4366337  0.0028441 153.523 < 0.0000000000000002 ***
## color.Q     -0.0974187  0.0025833 -37.711 < 0.0000000000000002 ***
## color.C      0.0125915  0.0024179   5.208      0.0000001927323 ***
## color^4      0.0133674  0.0022204   6.020      0.0000000017651 ***
## color^5      0.0001194  0.0021042   0.057              0.95475
## color^6      0.0026739  0.0019108   1.399              0.16171
## clarity.L    0.9075881  0.0049934 181.757 < 0.0000000000000002 ***
## clarity.Q   -0.2483814  0.0046523 -53.389 < 0.0000000000000002 ***
## clarity.C    0.1367692  0.0039749  34.408 < 0.0000000000000002 ***
## clarity^4   -0.0684754  0.0031760 -21.560 < 0.0000000000000002 ***
## clarity^5    0.0289678  0.0025881  11.193 < 0.0000000000000002 ***
## clarity^6   -0.0022952  0.0022444  -1.023              0.30649
## clarity^7    0.0313111  0.0019831  15.789 < 0.0000000000000002 ***
## carat        1.0002821  0.0081052 123.413 < 0.0000000000000002 ***
## length       0.1751101  0.0264602   6.618      0.0000000000371 ***
## depth       -0.0884193  0.0271461  -3.257              0.00113 **
## depth_ratio  0.0136166  0.0033272   4.093      0.0000427910418 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.131 on 26810 degrees of freedom
## Multiple R-squared:  0.9829, Adjusted R-squared:  0.9828
## F-statistic: 7.321e+04 on 21 and 26810 DF,  p-value: < 0.00000000000000022
```

```r
LM_stepwise_complete = step(LM_complete, direction = "both")
```

```
## Start:  AIC=-109064.2
## price ~ cut + color + clarity + carat + length + width + depth +
##     depth_ratio + table
##
##               Df Sum of Sq     RSS     AIC
## - width        1      0.01  459.84 -109066
## - table        1      0.02  459.85 -109065
## <none>                      459.83 -109064
## - depth        1      0.07  459.89 -109062
## - depth_ratio  1      0.12  459.95 -109059
## - length       1      0.76  460.59 -109022
## - cut          4     19.38  479.21 -107965
## - carat        1    253.72  713.54  -97277
## - color        6    428.53  888.36  -91407
## - clarity      7    874.13 1333.96  -80501
##
## Step:  AIC=-109065.7
## price ~ cut + color + clarity + carat + length + depth + depth_ratio +
##     table
##
##               Df Sum of Sq     RSS     AIC
## - table        1      0.02  459.86 -109066
## <none>                      459.84 -109066
## + width        1      0.01  459.83 -109064
## - depth        1      0.18  460.02 -109057
## - depth_ratio  1      0.30  460.14 -109050
## - length       1      0.75  460.59 -109024
## - cut          4     19.38  479.22 -107966
## - carat        1    256.52  716.35  -97173
## - color        6    428.96  888.80  -91395
## - clarity      7    877.59 1337.43  -80433
##
## Step:  AIC=-109066.4
## price ~ cut + color + clarity + carat + length + depth + depth_ratio
##
##               Df Sum of Sq     RSS     AIC
## <none>                      459.86 -109066
## + table        1      0.02  459.84 -109066
## + width        1      0.01  459.85 -109065
## - depth        1      0.18  460.04 -109058
## - depth_ratio  1      0.29  460.15 -109052
## - length       1      0.75  460.61 -109025
## - cut          4     24.58  484.44 -107677
## - carat        1    261.25  721.11  -96998
## - color        6    429.14  889.00  -91391
## - clarity      7    877.84 1337.70  -80430
```

```r
summary(LM_stepwise_complete)
```

```
##
## Call:
## lm(formula = price ~ cut + color + clarity + carat + length +
```

```
##     depth + depth_ratio, data = Train_lr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04014 -0.08295  0.00025  0.08161  1.42593
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) -0.0741493  0.0015667 -47.329 < 0.0000000000000002 ***
## cut.L        0.1150923  0.0034602  33.262 < 0.0000000000000002 ***
## cut.Q       -0.0339481  0.0029582 -11.476 < 0.0000000000000002 ***
## cut.C        0.0163231  0.0025452   6.413      0.0000000001447 ***
## cut^4       -0.0001739  0.0020487  -0.085              0.93235
## color.L      0.4366337  0.0028441 153.523 < 0.0000000000000002 ***
## color.Q     -0.0974187  0.0025833 -37.711 < 0.0000000000000002 ***
## color.C      0.0125915  0.0024179   5.208      0.0000001927323 ***
## color^4      0.0133674  0.0022204   6.020      0.0000000017651 ***
## color^5      0.0001194  0.0021042   0.057              0.95475
## color^6      0.0026739  0.0019108   1.399              0.16171
## clarity.L    0.9075881  0.0049934 181.757 < 0.0000000000000002 ***
## clarity.Q   -0.2483814  0.0046523 -53.389 < 0.0000000000000002 ***
## clarity.C    0.1367692  0.0039749  34.408 < 0.0000000000000002 ***
## clarity^4   -0.0684754  0.0031760 -21.560 < 0.0000000000000002 ***
## clarity^5    0.0289678  0.0025881  11.193 < 0.0000000000000002 ***
## clarity^6   -0.0022952  0.0022444  -1.023              0.30649
## clarity^7    0.0313111  0.0019831  15.789 < 0.0000000000000002 ***
## carat        1.0002821  0.0081052 123.413 < 0.0000000000000002 ***
## length       0.1751101  0.0264602   6.618      0.0000000000371 ***
## depth       -0.0884193  0.0271461  -3.257              0.00113 **
## depth_ratio  0.0136166  0.0033272   4.093      0.0000427910418 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.131 on 26810 degrees of freedom
## Multiple R-squared:  0.9829, Adjusted R-squared:  0.9828
## F-statistic: 7.321e+04 on 21 and 26810 DF,  p-value: < 0.00000000000000022
```

```r
# we use the GlobalCrit function from statistical modelling to reduce the number of predictors
# using global criterions: Mallows's Cp and AIC
GlobalCrit(LM_complete)
```

```
##
## ----------------------------------------------------
##   GLOBAL VARIABLE SELECTION PROCEDURE
##
##   ( Data =  Train_lr )
##
##   A = cut
##   B = color
##   C = clarity
##   D = carat
##   E = length
##   F = width
##   G = depth
##   H = depth_ratio
```

```
##   I = table
##
##   Models      |  Cp             |  AIC           |
##   ---------------------------------------------
##   ABCDEF      |      2.84  (10) |  32937.47  (10) |
##   ABCDEH      |      2.49  ( 9) |  32937.81  ( 9) |
##   ABCDEFG     |  -   0.63  ( 7) |  32940.94  ( 7) |
##   ABCDEFH     |  -   3.12  ( 5) |  32943.44  ( 5) |
##   ABCDEGH     |  -   6.12  ( 1) |  32946.43  ( 1) |
##   ABCDEFGH    |  -   4.80  ( 3) |  32945.11  ( 3) |
##   ABCDEFGI    |      0.80  ( 8) |  32939.51  ( 8) |
##   ABCDEFHI    |  -   2.18  ( 6) |  32942.50  ( 6) |
##   ABCDEGHI    |  -   5.45  ( 2) |  32945.77  ( 2) |
##   ABCDEFGHI   |  -   3.00  ( 4) |  32944.32  ( 4) |
##
## -----------------------------------------------------
```

```
##
## -----------------------------------------------------
##   GLOBAL VARIABLE SELECTION PROCEDURE
##
##   ( Data =  Train_lr )
##
##   A = cut
##   B = color
##   C = clarity
##   D = carat
##   E = length
##   F = width
##   G = depth
##   H = depth_ratio
##   I = table
##
##   Models      |  Cp             |  AIC           |
##   ---------------------------------------------
##   ABCDEF      |      2.84  (10) |  32937.47  (10) |
##   ABCDEH      |      2.49  ( 9) |  32937.81  ( 9) |
##   ABCDEFG     |  -   0.63  ( 7) |  32940.94  ( 7) |
##   ABCDEFH     |  -   3.12  ( 5) |  32943.44  ( 5) |
##   ABCDEGH     |  -   6.12  ( 1) |  32946.43  ( 1) |
##   ABCDEFGH    |  -   4.80  ( 3) |  32945.11  ( 3) |
##   ABCDEFGI    |      0.80  ( 8) |  32939.51  ( 8) |
##   ABCDEFHI    |  -   2.18  ( 6) |  32942.50  ( 6) |
##   ABCDEGHI    |  -   5.45  ( 2) |  32945.77  ( 2) |
##   ABCDEFGHI   |  -   3.00  ( 4) |  32944.32  ( 4) |
##
## -----------------------------------------------------
```

```r
# we compute the linear model obtained with global methods and display its summary
LM_CpAIC_complete = lm(price ~ . , data = Train_lr[, c(1:6, 8, 9)])
summary(LM_CpAIC_complete)
```

```
##
## Call:
## lm(formula = price ~ ., data = Train_lr[, c(1:6, 8, 9)])
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04014 -0.08295  0.00025  0.08161  1.42593
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) -0.0741493  0.0015667 -47.329 < 0.0000000000000002 ***
## cut.L        0.1150923  0.0034602  33.262 < 0.0000000000000002 ***
## cut.Q       -0.0339481  0.0029582 -11.476 < 0.0000000000000002 ***
## cut.C        0.0163231  0.0025452   6.413      0.0000000001447 ***
## cut^4       -0.0001739  0.0020487  -0.085              0.93235
## color.L      0.4366337  0.0028441 153.523 < 0.0000000000000002 ***
## color.Q     -0.0974187  0.0025833 -37.711 < 0.0000000000000002 ***
## color.C      0.0125915  0.0024179   5.208      0.0000001927323 ***
## color^4      0.0133674  0.0022204   6.020      0.0000000017651 ***
## color^5      0.0001194  0.0021042   0.057              0.95475
## color^6      0.0026739  0.0019108   1.399              0.16171
## clarity.L    0.9075881  0.0049934 181.757 < 0.0000000000000002 ***
## clarity.Q   -0.2483814  0.0046523 -53.389 < 0.0000000000000002 ***
## clarity.C    0.1367692  0.0039749  34.408 < 0.0000000000000002 ***
## clarity^4   -0.0684754  0.0031760 -21.560 < 0.0000000000000002 ***
## clarity^5    0.0289678  0.0025881  11.193 < 0.0000000000000002 ***
## clarity^6   -0.0022952  0.0022444  -1.023              0.30649
## clarity^7    0.0313111  0.0019831  15.789 < 0.0000000000000002 ***
## carat        1.0002821  0.0081052 123.413 < 0.0000000000000002 ***
## length       0.1751101  0.0264602   6.618      0.0000000000371 ***
## depth       -0.0884193  0.0271461  -3.257              0.00113 **
## depth_ratio  0.0136166  0.0033272   4.093      0.0000427910418 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.131 on 26810 degrees of freedom
## Multiple R-squared:  0.9829, Adjusted R-squared:  0.9828
## F-statistic: 7.321e+04 on 21 and 26810 DF,  p-value: < 0.00000000000000022
```

```r
# we define a training set without the correlated predictors (length, width, depth)
Train_minus_corr <- Train_lr[, -c(6:8)]

# we compute the linear model without correlated predictors and display its summary
LM_minus_corr = lm(price ~ ., data = Train_minus_corr)
summary(LM_minus_corr)
```

```
##
## Call:
## lm(formula = price ~ ., data = Train_minus_corr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98683 -0.08456 -0.00054  0.08206  1.42888
##
## Coefficients:
##               Estimate Std. Error  t value             Pr(>|t|)
## (Intercept) -0.0733380  0.0016248  -45.136 < 0.0000000000000002 ***
## cut.L        0.1174782  0.0037111   31.656 < 0.0000000000000002 ***
```

```
## cut.Q       -0.0331237  0.0029650  -11.172 < 0.0000000000000002 ***
## cut.C        0.0132619  0.0025653    5.170         0.0000002361 ***
## cut^4       -0.0022453  0.0020453   -1.098               0.2723
## color.L      0.4322410  0.0028233  153.096 < 0.0000000000000002 ***
## color.Q     -0.0958030  0.0025851  -37.059 < 0.0000000000000002 ***
## color.C      0.0133140  0.0024238    5.493         0.0000000399 ***
## color^4      0.0125884  0.0022260    5.655         0.0000000157 ***
## color^5     -0.0001023  0.0021099   -0.048               0.9613
## color^6      0.0028366  0.0019160    1.481               0.1388
## clarity.L    0.9068901  0.0050015  181.324 < 0.0000000000000002 ***
## clarity.Q   -0.2447781  0.0046522  -52.615 < 0.0000000000000002 ***
## clarity.C    0.1346227  0.0039816   33.811 < 0.0000000000000002 ***
## clarity^4   -0.0686749  0.0031846  -21.564 < 0.0000000000000002 ***
## clarity^5    0.0288881  0.0025955   11.130 < 0.0000000000000002 ***
## clarity^6   -0.0024999  0.0022505   -1.111               0.2667
## clarity^7    0.0320730  0.0019874   16.138 < 0.0000000000000002 ***
## carat        1.0865804  0.0009254 1174.133 < 0.0000000000000002 ***
## depth_ratio -0.0016945  0.0009461   -1.791               0.0733 .
## table       -0.0001141  0.0010774   -0.106               0.9157
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 26811 degrees of freedom
## Multiple R-squared:  0.9828, Adjusted R-squared:  0.9828
## F-statistic: 7.644e+04 on 20 and 26811 DF,  p-value: < 0.00000000000000022
```

```r
# we use iterative search algorithms on the model without corr: forward, backward and stepwise
# we display summaries of the three models obtained with iterative methods

LM_backward_minus_corr = step(LM_minus_corr, direction = "backward")
```

```
## Start:  AIC=-108920.2
## price ~ cut + color + clarity + carat + depth_ratio + table
##
##               Df Sum of Sq     RSS     AIC
## - table        1       0.0   462.4 -108922
## <none>                       462.4 -108920
## - depth_ratio  1       0.1   462.5 -108919
## - cut          4      20.1   482.6 -107784
## - color        6     429.9   892.3  -91293
## - clarity      7     882.5  1344.9  -80288
## - carat        1   23776.4 24238.8   -2687
##
## Step:  AIC=-108922.2
## price ~ cut + color + clarity + carat + depth_ratio
##
##               Df Sum of Sq     RSS     AIC
## <none>                       462.4 -108922
## - depth_ratio  1       0.1   462.5 -108921
## - cut          4      26.3   488.7 -107445
## - color        6     430.0   892.4  -91293
## - clarity      7     883.1  1345.5  -80277
## - carat        1   24022.4 24484.8   -2418
```

```r
summary(LM_backward_minus_corr)
```

```
##
## Call:
## lm(formula = price ~ cut + color + clarity + carat + depth_ratio,
##     data = Train_minus_corr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98668 -0.08451 -0.00056  0.08207  1.42904
##
## Coefficients:
##                Estimate Std. Error  t value             Pr(>|t|)
## (Intercept) -0.0733824  0.0015697  -46.750 < 0.0000000000000002 ***
## cut.L        0.1176217  0.0034546   34.048 < 0.0000000000000002 ***
## cut.Q       -0.0331066  0.0029605  -11.183 < 0.0000000000000002 ***
## cut.C        0.0133117  0.0025217    5.279         0.0000001310 ***
## cut^4       -0.0022101  0.0020180   -1.095               0.2734
## color.L      0.4322448  0.0028231  153.112 < 0.0000000000000002 ***
## color.Q     -0.0958088  0.0025845  -37.070 < 0.0000000000000002 ***
## color.C      0.0133162  0.0024237    5.494         0.0000000396 ***
## color^4      0.0125930  0.0022255    5.658         0.0000000154 ***
## color^5     -0.0001032  0.0021098   -0.049               0.9610
## color^6      0.0028362  0.0019159    1.480               0.1388
## clarity.L    0.9069041  0.0049997  181.393 < 0.0000000000000002 ***
## clarity.Q   -0.2447790  0.0046521  -52.616 < 0.0000000000000002 ***
## clarity.C    0.1346281  0.0039812   33.816 < 0.0000000000000002 ***
## clarity^4   -0.0686735  0.0031846  -21.565 < 0.0000000000000002 ***
## clarity^5    0.0288920  0.0025952   11.133 < 0.0000000000000002 ***
## clarity^6   -0.0024996  0.0022504   -1.111               0.2667
## clarity^7    0.0320720  0.0019873   16.138 < 0.0000000000000002 ***
## carat        1.0865705  0.0009207 1180.212 < 0.0000000000000002 ***
## depth_ratio -0.0016535  0.0008632   -1.916               0.0554 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 26812 degrees of freedom
## Multiple R-squared:  0.9828, Adjusted R-squared:  0.9828
## F-statistic: 8.047e+04 on 19 and 26812 DF,  p-value: < 0.00000000000000022
```

```r
LM_forward_minus_corr = step(LM_minus_corr, direction = "forward")
```

```
## Start:  AIC=-108920.2
## price ~ cut + color + clarity + carat + depth_ratio + table
```

```r
summary(LM_forward_minus_corr)
```

```
##
## Call:
## lm(formula = price ~ cut + color + clarity + carat + depth_ratio +
##     table, data = Train_minus_corr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98683 -0.08456 -0.00054  0.08206  1.42888
```

```
##
## Coefficients:
##               Estimate Std. Error  t value             Pr(>|t|)
## (Intercept) -0.0733380  0.0016248  -45.136 < 0.0000000000000002 ***
## cut.L        0.1174782  0.0037111   31.656 < 0.0000000000000002 ***
## cut.Q       -0.0331237  0.0029650  -11.172 < 0.0000000000000002 ***
## cut.C        0.0132619  0.0025653    5.170         0.0000002361 ***
## cut^4       -0.0022453  0.0020453   -1.098               0.2723
## color.L      0.4322410  0.0028233  153.096 < 0.0000000000000002 ***
## color.Q     -0.0958030  0.0025851  -37.059 < 0.0000000000000002 ***
## color.C      0.0133140  0.0024238    5.493         0.0000000399 ***
## color^4      0.0125884  0.0022260    5.655         0.0000000157 ***
## color^5     -0.0001023  0.0021099   -0.048               0.9613
## color^6      0.0028366  0.0019160    1.481               0.1388
## clarity.L    0.9068901  0.0050015  181.324 < 0.0000000000000002 ***
## clarity.Q   -0.2447781  0.0046522  -52.615 < 0.0000000000000002 ***
## clarity.C    0.1346227  0.0039816   33.811 < 0.0000000000000002 ***
## clarity^4   -0.0686749  0.0031846  -21.564 < 0.0000000000000002 ***
## clarity^5    0.0288881  0.0025955   11.130 < 0.0000000000000002 ***
## clarity^6   -0.0024999  0.0022505   -1.111               0.2667
## clarity^7    0.0320730  0.0019874   16.138 < 0.0000000000000002 ***
## carat        1.0865804  0.0009254 1174.133 < 0.0000000000000002 ***
## depth_ratio -0.0016945  0.0009461   -1.791               0.0733 .
## table       -0.0001141  0.0010774   -0.106               0.9157
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 26811 degrees of freedom
## Multiple R-squared:  0.9828, Adjusted R-squared:  0.9828
## F-statistic: 7.644e+04 on 20 and 26811 DF,  p-value: < 0.00000000000000022
```

```
LM_stepwise_minus_corr = step(LM_minus_corr, direction = "both")
```

```
## Start:  AIC=-108920.2
## price ~ cut + color + clarity + carat + depth_ratio + table
##
##               Df Sum of Sq     RSS     AIC
## - table        1       0.0   462.4 -108922
## <none>                       462.4 -108920
## - depth_ratio  1       0.1   462.5 -108919
## - cut          4      20.1   482.6 -107784
## - color        6     429.9   892.3  -91293
## - clarity      7     882.5  1344.9  -80288
## - carat        1   23776.4 24238.8   -2687
##
## Step:  AIC=-108922.2
## price ~ cut + color + clarity + carat + depth_ratio
##
##               Df Sum of Sq     RSS     AIC
## <none>                       462.4 -108922
## - depth_ratio  1       0.1   462.5 -108921
## + table        1       0.0   462.4 -108920
## - cut          4      26.3   488.7 -107445
## - color        6     430.0   892.4  -91293
## - clarity      7     883.1  1345.5  -80277
```

```
## - carat        1   24022.4 24484.8   -2418
```

```r
summary(LM_stepwise_minus_corr)
```

```
##
## Call:
## lm(formula = price ~ cut + color + clarity + carat + depth_ratio,
##     data = Train_minus_corr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98668 -0.08451 -0.00056  0.08207  1.42904
##
## Coefficients:
##               Estimate Std. Error  t value             Pr(>|t|)
## (Intercept) -0.0733824  0.0015697  -46.750 < 0.0000000000000002 ***
## cut.L        0.1176217  0.0034546   34.048 < 0.0000000000000002 ***
## cut.Q       -0.0331066  0.0029605  -11.183 < 0.0000000000000002 ***
## cut.C        0.0133117  0.0025217    5.279         0.0000001310 ***
## cut^4       -0.0022101  0.0020180   -1.095               0.2734
## color.L      0.4322448  0.0028231  153.112 < 0.0000000000000002 ***
## color.Q     -0.0958088  0.0025845  -37.070 < 0.0000000000000002 ***
## color.C      0.0133162  0.0024237    5.494         0.0000000396 ***
## color^4      0.0125930  0.0022255    5.658         0.0000000154 ***
## color^5     -0.0001032  0.0021098   -0.049               0.9610
## color^6      0.0028362  0.0019159    1.480               0.1388
## clarity.L    0.9069041  0.0049997  181.393 < 0.0000000000000002 ***
## clarity.Q   -0.2447790  0.0046521  -52.616 < 0.0000000000000002 ***
## clarity.C    0.1346281  0.0039812   33.816 < 0.0000000000000002 ***
## clarity^4   -0.0686735  0.0031846  -21.565 < 0.0000000000000002 ***
## clarity^5    0.0288920  0.0025952   11.133 < 0.0000000000000002 ***
## clarity^6   -0.0024996  0.0022504   -1.111               0.2667
## clarity^7    0.0320720  0.0019873   16.138 < 0.0000000000000002 ***
## carat        1.0865705  0.0009207 1180.212 < 0.0000000000000002 ***
## depth_ratio -0.0016535  0.0008632   -1.916               0.0554 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 26812 degrees of freedom
## Multiple R-squared:  0.9828, Adjusted R-squared:  0.9828
## F-statistic: 8.047e+04 on 19 and 26812 DF,  p-value: < 0.00000000000000022
```

```r
# we use the GlobalCrit function from statistical modelling to reduce the number of predictors
# using global criterions: Mallows's Cp and AIC
GlobalCrit(LM_minus_corr)
```

```
##
## -----------------------------------------------------
##   GLOBAL VARIABLE SELECTION PROCEDURE
##
##   ( Data =  Train_minus_corr )
##
##   A = cut
##   B = color
##   C = clarity
```

```
##   D = carat
##   E = depth_ratio
##   F = table
##
##   Models      |  Cp            |  AIC           |
##   -------------------------------------------------
##   BCD         |   1686.62  ( 8) |  31157.06  ( 8) |
##   ABCD        |  -   7.32  ( 2) |  32800.58  ( 2) |
##   ACDE        |  24920.99  (10) |  15162.85  ( 9) |
##   BCDE        |   1514.54  ( 7) |  31319.38  ( 7) |
##   BCDF        |   1498.62  ( 6) |  31334.46  ( 6) |
##   ABCDE       |  -   8.99  ( 1) |  32802.26  ( 1) |
##   ABCDF       |  -   5.79  ( 4) |  32799.06  ( 4) |
##   ACDEF       |  24919.19  ( 9) |  15162.82  (10) |
##   BCDEF       |   1158.93  ( 5) |  31658.16  ( 5) |
##   ABCDEF      |  -   7.00  ( 3) |  32800.27  ( 3) |
##
## -----------------------------------------------------
```

```
##
## -----------------------------------------------------
##   GLOBAL VARIABLE SELECTION PROCEDURE
##
##   ( Data =  Train_minus_corr )
##
##   A = cut
##   B = color
##   C = clarity
##   D = carat
##   E = depth_ratio
##   F = table
##
##   Models      |  Cp            |  AIC           |
##   -------------------------------------------------
##   BCD         |   1686.62  ( 8) |  31157.06  ( 8) |
##   ABCD        |  -   7.32  ( 2) |  32800.58  ( 2) |
##   ACDE        |  24920.99  (10) |  15162.85  ( 9) |
##   BCDE        |   1514.54  ( 7) |  31319.38  ( 7) |
##   BCDF        |   1498.62  ( 6) |  31334.46  ( 6) |
##   ABCDE       |  -   8.99  ( 1) |  32802.26  ( 1) |
##   ABCDF       |  -   5.79  ( 4) |  32799.06  ( 4) |
##   ACDEF       |  24919.19  ( 9) |  15162.82  (10) |
##   BCDEF       |   1158.93  ( 5) |  31658.16  ( 5) |
##   ABCDEF      |  -   7.00  ( 3) |  32800.27  ( 3) |
##
## -----------------------------------------------------
```

```r
# we compute the linear model obtained with global methods and display its summary
LM_CpAIC_minus_corr = lm(price ~ . , data = Train_lr[, c(1:5, 9)])
summary(LM_CpAIC_minus_corr)
```

```
##
## Call:
## lm(formula = price ~ ., data = Train_lr[, c(1:5, 9)])
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98668 -0.08451 -0.00056  0.08207  1.42904
##
## Coefficients:
##               Estimate Std. Error  t value             Pr(>|t|)
## (Intercept) -0.0733824  0.0015697  -46.750 < 0.0000000000000002 ***
## cut.L        0.1176217  0.0034546   34.048 < 0.0000000000000002 ***
## cut.Q       -0.0331066  0.0029605  -11.183 < 0.0000000000000002 ***
## cut.C        0.0133117  0.0025217    5.279         0.0000001310 ***
## cut^4       -0.0022101  0.0020180   -1.095               0.2734
## color.L      0.4322448  0.0028231  153.112 < 0.0000000000000002 ***
## color.Q     -0.0958088  0.0025845  -37.070 < 0.0000000000000002 ***
## color.C      0.0133162  0.0024237    5.494         0.0000000396 ***
## color^4      0.0125930  0.0022255    5.658         0.0000000154 ***
## color^5     -0.0001032  0.0021098   -0.049               0.9610
## color^6      0.0028362  0.0019159    1.480               0.1388
## clarity.L    0.9069041  0.0049997  181.393 < 0.0000000000000002 ***
## clarity.Q   -0.2447790  0.0046521  -52.616 < 0.0000000000000002 ***
## clarity.C    0.1346281  0.0039812   33.816 < 0.0000000000000002 ***
## clarity^4   -0.0686735  0.0031846  -21.565 < 0.0000000000000002 ***
## clarity^5    0.0288920  0.0025952   11.133 < 0.0000000000000002 ***
## clarity^6   -0.0024996  0.0022504   -1.111               0.2667
## clarity^7    0.0320720  0.0019873   16.138 < 0.0000000000000002 ***
## carat        1.0865705  0.0009207 1180.212 < 0.0000000000000002 ***
## depth_ratio -0.0016535  0.0008632   -1.916               0.0554 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 26812 degrees of freedom
## Multiple R-squared:  0.9828, Adjusted R-squared:  0.9828
## F-statistic: 8.047e+04 on 19 and 26812 DF,  p-value: < 0.00000000000000022
```

```
# we display a table of the predictors used in each model using kable()
# kable_styling() controls the parameter fullwidth for the total width of table

kable_styling(
  kable(
    # the data to display in the table: model names and crosses for presence of predictor
    data.table(Model = c("LM_complete", "LM_forward_complete", "LM_backward_complete",
                         "LM_stepwise_complete", "LM_CpAIC_complete",
                         "LM_minus_corr", "LM_forward_minus_corr", "LM_backward_minus_corr",
                         "LM_stepwise_minus_corr", "LM_CpAIC_minus_corr"),
              Cut           = c("X","X","X","X","X","X","X","X","X","X"),
              Color         = c("X","X","X","X","X","X","X","X","X","X"),
              Clarity       = c("X","X","X","X","X","X","X","X","X","X"),
              Carat         = c("X","X","X","X","X","X","X","X","X","X"),
              Length        = c("X","X","X","X","X"," "," "," "," "," "),
              Width         = c("X","X"," "," "," "," "," "," "," "," "),
              Depth         = c("X","X","X","X","X"," "," "," "," "," "),
              `Depth Ratio` = c("X","X","X","X","X","X","X","X","X","X"),
              Table         = c("X","X"," "," "," ","X","X"," "," "," ")
              ),
    align = 'lccccccccc',                                # alignment of each column
```

Table 1: Predictors used in each linear model

Model	Cut	Color	Clarity	Carat	Length	Width	Depth	Depth Ratio	Table
LM_complete	X	X	X	X	X	X	X	X	X
LM_forward_complete	X	X	X	X	X	X	X	X	X
LM_backward_complete	X	X	X	X	X		X	X	
LM_stepwise_complete	X	X	X	X	X		X	X	
LM_CpAIC_complete	X	X	X	X	X		X	X	
LM_minus_corr	X	X	X	X				X	X
LM_forward_minus_corr	X	X	X	X				X	X
LM_backward_minus_corr	X	X	X	X				X	
LM_stepwise_minus_corr	X	X	X	X				X	
LM_CpAIC_minus_corr	X	X	X	X				X	

```
    caption = "Predictors used in each linear model"),  # caption of the table
  full_width = FALSE)                                   # table isn't full width of page
```

In both cases, the forward selection doesn't discard any variables, whereas backward, stepwise and global selections all choose the same model with less variables than initially.

Thus, we have four different models emerging. We will keep `LM_complete`, `LM_CpAIC_complete`, `LM_minus_corr` and `LM_CpAIC_minus_corr` and remove the other models which are duplicates.

```
# we remove redundant models
rm(LM_forward_complete, LM_backward_complete, LM_stepwise_complete, LM_forward_minus_corr, LM_backward_
```

```
# predicting prices of validation set on the validation data

# we create a data table of predictions which contains predictions for the four models
# predictions are computed on validation set using predict()
LM_Predictions =
  data.table(
    LM_complete_pred = predict(object = LM_complete, newdata = Valid_lr),
    LM_CpAIC_complete_pred = predict(object = LM_CpAIC_complete, newdata = Valid_lr),
    LM_minus_corr_pred = predict(object = LM_minus_corr, newdata = Valid_lr),
    LM_CpAIC_minus_corr_pred = predict(object = LM_CpAIC_minus_corr, newdata = Valid_lr)
    )
```

```
# we have to scale back the price, to do so we fetch the mean and std value from norm.values

# we display all means and stds
norm.values$mean
```

```
##       price       carat      length       width       depth depth_ratio       table
##   7.7876577  -0.3941013   5.7331235   5.7353183   3.5414058  61.7640019   4.0504070
```

```
norm.values$std
```

```
##       price       carat      length       width       depth depth_ratio       table
##  1.01430151  0.58510643  1.12051551  1.11223832  0.69253953  1.41492652  0.03808011
```

```
# fetching for price
mean_price = norm.values$mean[1]
std_price = norm.values$std[1]

# scaling back (Y*mu + sigma), then exp() (we had transformed price with a log for skewness)
```

Table 2: Accuracy measures of linear models

Model	ME	RMSE	MAE	MPE	MAPE
LM_complete	36.36648	846.5888	408.5364	-0.8377036	10.33643
LM_CpAIC_complete	36.32774	845.5018	408.1349	-0.8408540	10.33659
LM_minus_corr	50.38994	810.1522	405.0486	-0.8420621	10.39559
LM_CpAIC_minus_corr	50.39878	810.1610	405.0616	-0.8418121	10.39568

```
LM_Predictions = LM_Predictions*std_price + mean_price
LM_Predictions = exp(LM_Predictions)
```

```
# taking real prices of validation data from diamonds (which has not been touched -> original scale)
LM_Predictions[, real_prices := diamonds[valid.index, price]]
```

```
# we compute accuracy measures for each model using accuracy() from the forecast package
# we convert each accuracy to a table (in order to put them together afterwards)

Acc1 = accuracy(object = LM_Predictions$LM_complete_pred, x = LM_Predictions$real_prices)
Acc1 = as.data.table(Acc1)
Acc2 = accuracy(object = LM_Predictions$LM_CpAIC_complete_pred, x = LM_Predictions$real_prices)
Acc2 = as.data.table(Acc2)
Acc3 = accuracy(object = LM_Predictions$LM_minus_corr_pred, x = LM_Predictions$real_prices)
Acc3 = as.data.table(Acc3)
Acc4 = accuracy(object = LM_Predictions$LM_CpAIC_minus_corr_pred, x = LM_Predictions$real_prices)
Acc4 = as.data.table(Acc4)

# we create a list of accuracy measures
Accs = list(Acc1, Acc2, Acc3, Acc4)
# we use rbindlist() from the data.table package to stack the four tables on top of eachother
Accs = rbindlist(Accs)
# now that Accs is created we can remove the four tables
rm(Acc1, Acc2, Acc3, Acc4)
# we add a column to Accs with the model names
Accs[, Model := c("LM_complete", "LM_CpAIC_complete", "LM_minus_corr", "LM_CpAIC_minus_corr")]
# we put the Model column first
Accs <- Accs[, c(6, 1:5)]

# displaying Accs as a table with kable() and adding a caption
# kable_styling controls width of table
kable_styling(kable(Accs, caption = "Accuracy measures of linear models"), full_width = FALSE)
```