

# STAT394 Group Project Final Report

Ken MacIver, Tom Tribe, Jundi Yang, Mei Huang

2022-10-03

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	The variables . . . . .	2
<b>2</b>	<b>Methodology</b>	<b>3</b>
<b>3</b>	<b>Results</b>	<b>4</b>
3.1	Consolidated Exploratory Data Analysis (EDA) . . . . .	4
3.2	Categorical Summaries . . . . .	5
3.3	Make a data.frame version of dataset . . . . .	15
3.4	Determining whether the outliers are errors . . . . .	17
3.5	Colour-coded scatterplots . . . . .	19
<b>4</b>	<b>Simple linear regression using ‘carat’</b>	<b>23</b>
4.1	Testing to find the best regression model . . . . .	24
4.2	Fitting the best model . . . . .	29
4.3	Check Regression Assumptions . . . . .	30
4.4	Scaled multiple regression model . . . . .	31
<b>5</b>	<b>Principal Component Analysis</b>	<b>32</b>
5.1	Identify the extreme PC1 values . . . . .	35
5.2	PCA with Smaller Sample . . . . .	35
5.3	Principal Components Regression . . . . .	37
5.4	Factor Analysis . . . . .	40
<b>6</b>	<b>Linear Discriminant Analysis (LDA)</b>	<b>41</b>
6.1	Why prediction without the categorical variables is not working well . . . . .	43
6.2	Discriminant analysis using the reduced dataset (price, carat and clarity) . . .	45
<b>7</b>	<b>Cluster Analysis</b>	<b>49</b>
<b>8</b>	<b>Conclusions</b>	<b>54</b>

<b>9 Appendices</b>	<b>54</b>
9.1 Code for Producing KS Tests . . . . .	54
9.2 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Cut . . . . .	54
9.3 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Clarity . . . . .	54
9.4 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Color . . . . .	54
<b>Bibliography</b>	<b>54</b>

# 1 Introduction

Our multivariate statistical investigation uses the ‘Diamonds’ data set. This data set is a base data set in R (available in the `ggplot2` package) and is also freely available on Kaggle (“Diamonds Dataset, Kaggle.com” 2016). It contains information on ten different variables for 53940 diamonds. Of these ten variables three are categorical while 7 are numerical.

## 1.1 The variables

- Carat: a measure of the diamond’s weight. One carat equals 1/5 gram. In this data set there are diamonds with carat values ranging from 0.2 - 5.01.
- Cut: A diamond’s cut defines its proportions and its ability to reflect light. This variable has five levels: Fair (lowest quality), Good, Very Good, Premium and Ideal (highest quality).
- Color: A diamond’s color refers to how clear/colorless it is. This variable has seven levels: J (lowest quality), I, H, G, F, E, D (highest quality).
- Clarity: measures small imperfections on the surface and within the stone. This variable has eight levels: I1 (lowest clarity), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (highest clarity).
- x: length in mm. In this data set there are diamonds with a length ranging from 0 - 10.74.
- y: width in mm. In this data set there are diamonds with a width ranging from 0 - 58.9.
- z: depth in mm. In this data set there are diamonds with a depth ranging from 0 - 31.8.
- Depth: total depth percentage. This is calculated by dividing the total width by the total depth ( $depth = \frac{2 \times z}{(x+y)}$ ). Depth percentage impacts how light reflects off the diamond. In this data set we have diamonds with a depth percentage ranging from 43% - 79%.

- Table: The flat facet on the top of a diamond is called its table. Table is calculated by dividing table width by the total width. Table percentage impacts how light reflects off the diamond. In this data set there are diamonds with a table percentage ranging from 43% - 95%.
- Price: price of the diamond in United States dollars (USD). In this data set there are diamonds with a price ranging from \$326 - \$18823.

### 1.1.1 Why ‘diamonds’?

We selected the diamonds dataset because it was easy to understand what the variables were measuring. This was in contrast to some of the other ones we looked at which required domain specific familiarity, such as knowledge about biology or chemistry (or even worse, bio-chemistry!). This data set also allows us to undertake an investigation from which we glean insights about diamonds that have real-world application and use.

### 1.1.2 Aims

Our primary aim was to identify which variables best predictor the price of any given diamond. We hypothesised that increased diamond size (x, y and z) and weight will be positively correlated with diamond price. We also expected to see increased prices for diamonds with higher levels of the categorical variables. We were unsure how diamond depth and table percentage will relate to diamond price.

We also sought to explore the techniques of Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Cluster Analysis (CA) that we have learned in STAT394. The diamonds dataset has 280 unique possible interactions ( $5 \times 7 \times 8$ ) between different levels of the categorical variables. Our secondary aim was to discover whether one of these techniques (PCA, LDA or CA) could classify the data in a simpler manner.

## 2 Methodology

The diamonds data set was accessed via the kaggle.com website (“Diamonds Dataset, Kaggle.com” 2016) while all statistical analysis and reporting was completed using the computer software package R (R Core Team 2017), in RStudio (RStudio Team 2022) and using RMarkdown (Allaire et al. 2020). Github (“Github.com” 2022) was used a repository for storing all relevant documents and code during this project.

Upon loading the data set into R we first used the following code to delete unnecessary columns, ensure categorical variables were treated as factors with set levels and created a subset data set containing only the numeric variables.

```
# Read the data set into R
diamonds <- read.csv("./diamonds.csv", encoding = "UTF-8")
# Remove the index column
diamonds$X <- NULL
# Set categorical variables as factors and set levels
```

```

diamonds$cut <- factor(diamonds$cut,
                         levels = c("Fair", "Good", "Very Good", "Premium", "Ideal"))
diamonds$color <- factor(diamonds$color,
                          levels = c("J", "I", "H", "G", "F", "E", "D"))
diamonds$clarity <- factor(diamonds$clarity,
                            levels = c("I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF"))
# Make data frame of just the numerical variables
diamonds_num <- subset(diamonds, select = c(carat, depth, table, price, x, y, z))

```

When scaling numerical values in our data set we used the `scale()` function.

When taking smaller samples from our data set to improve the interpretability of visual plots we used a seed consisting of either the student identification number of the team member who wrote the code or 1234567890, and the pseudo random number generator Mersenne Twister.

To begin with, we undertook an in-depth Exploratory Data Analysis (EDA) of the data. The key results from the EDA can be found in the results section below.

## 3 Results

### 3.1 Consolidated Exploratory Data Analysis (EDA)

In this section we present key findings from the consolidated Exploratory Data Analysis.

#### 3.1.1 Summaries of Data

Table 1: Summary statistics for 'diamonds' (2 d.p.)

	carat	depth	table	price	x	y	z
sample size	53940.00	53940.00	53940.00	53940.00	53940.00	53940.00	53940.00
minimum	0.20	43.00	43.00	326.00	0.00	0.00	0.00
first quartile	0.40	61.00	56.00	950.00	4.71	4.72	2.91
median	0.70	61.80	57.00	2401.00	5.70	5.71	3.53
mean	0.80	61.75	57.46	3932.80	5.73	5.73	3.54
third quartile	1.04	62.50	59.00	5324.25	6.54	6.54	4.04
maximum	5.01	79.00	95.00	18823.00	10.74	58.90	31.80
IQR	0.64	1.50	3.00	4374.25	1.83	1.82	1.13
standard deviation	0.47	1.43	2.23	3989.44	1.12	1.14	0.71
skewness	1.12	-0.08	0.80	1.62	0.38	2.43	1.52
kurtosis	4.26	8.74	5.80	5.18	2.38	94.21	50.08

A brief look at the summary statistics of the numerical variables (table 1) shows that there are a range of different scales. We also see high skewness values for carat, price, y and z and high kurtosis values for all variables, particularly y and z. Both of these indicate non-normality of

our numeric variables. It is also interesting to note that at least one diamond has a zero value for x, y and z. These values are further investigated in the “Outliers and Unusual Points” section below.

## 3.2 Categorical Summaries

### 3.2.1 Tables of counts for the categorical variables

Table 2: Count of ‘color’

Level	Count
J	2808
I	5422
H	8304
G	11292
F	9542
E	9797
D	6775

Table 2 shows the number of diamonds in each level of the ‘color’ variable. As a reminder, ‘J’ is the lowest level and ‘D’ is the highest.

Table 3: Count of ‘cut’

Level	Count
Fair	1610
Good	4906
Very Good	12082
Premium	13791
Ideal	21551

Table 3 gives a breakdown of the how many diamonds are in each level of the ‘cut’ variable. As a reminder, ‘Fair’ is the lowest level and ‘Ideal’ is the highest. We can see that most are in the ‘Ideal’, with a substantial number also in the ‘Premium’ and ‘Very Good’.

Table 4: Count of 'clarity'

Level	Count
I1	741
SI2	9194
SI1	13065
VS2	12258
VS1	8171
VVS2	5066
VVS1	3655
IF	1790

Table 4 shows the number of diamonds in each level of the 'clarity' variable. As a reminder, 'I1' is the lowest level and 'IF' is the highest. ### Distribution of Numeric Variables

While examining histograms, Cullen and Frey Plots, skewness and kurtosis of the numerical variables, we observed evidence that the numerical variables were not normally distributed. The assumption of normality underlies many statistical procedures so we investigated this in more depth using Normal Quantile plots and goodness-of-fit tests.

```
par(mfrow=c(3,3))
qqnorm(diamonds$carat, xlab = "Observations",
        ylab = "Normal Quantiles", main = "Carat", col = "red")
qqline(diamonds$carat, col = "blue", lwd =2)
qqnorm(diamonds$depth, xlab = "Observations",
        ylab = "Normal Quantiles", main = "Depth", col = "red")
qqline(diamonds$depth, col = "blue", lwd =2)
qqnorm(diamonds$table, xlab = "Observations",
        ylab = "Normal Quantiles", main = "Table", col = "red")
qqline(diamonds$table, col = "blue", lwd =2)
qqnorm(diamonds$price, xlab = "Observations",
        ylab = "Normal Quantiles", main = "Price", col = "red")
qqline(diamonds$price, col = "blue", lwd =2)
qqnorm(diamonds$x, xlab = "Observations",
        ylab = "Normal Quantiles", main = "x", col = "red")
qqline(diamonds$x, col = "blue", lwd =2)
qqnorm(diamonds$y, xlab = "Observations",
        ylab = "Normal Quantiles", main = "y", col = "red")
qqline(diamonds$y, col = "blue", lwd =2)
qqnorm(diamonds$depth, xlab = "Observations",
        ylab = "Normal Quantiles", main = "z", col = "red")
qqline(diamonds$depth, col = "blue", lwd =2)
```

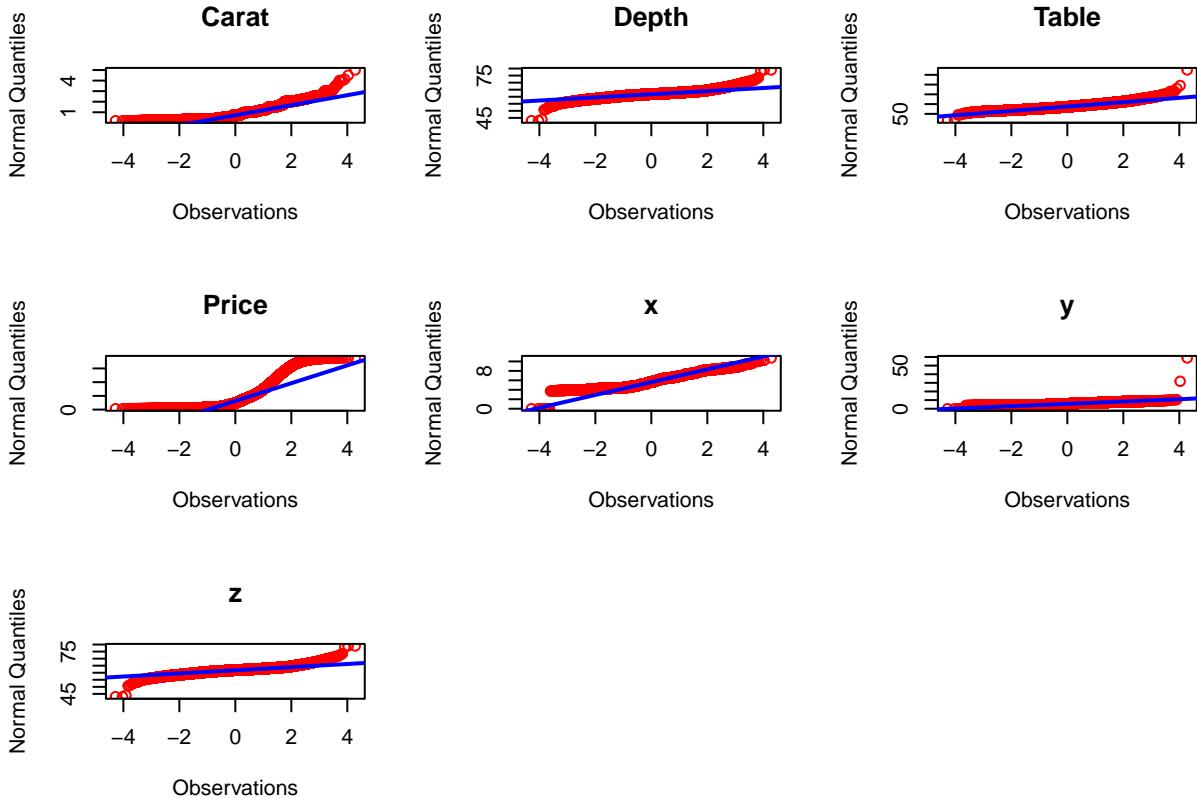


Figure 1: Normal QQ Plots of Numeric Variables

Despite the small size of the Normal QQ plots in figure 1, it is clear that each numeric variable deviates significantly from a normal distribution. We tested this using Kolmogorov-Smirnov goodness-of-fit test. The results from these tests are shown in the table below.

Table 5: Kolmogorov-Smirnov GOF results

Variable	Test_Statistic	p_value
Carat	0.122740	<2.2e-16
Depth	0.075871	<2.2e-16
Table	0.132250	<2.2e-16
Price	0.184670	<2.2e-16
x	0.093545	<2.2e-16
y	0.088528	<2.2e-16
z	0.089273	<2.2e-16

Table 5 shows that for all seven of the numeric variables, the Kolmogorov-Smirnov goodness-of-fit tests provided strong evidence for non-normality, as evidenced by the p-values which are all machine precision zero.

### 3.2.2 Correlation Matrix

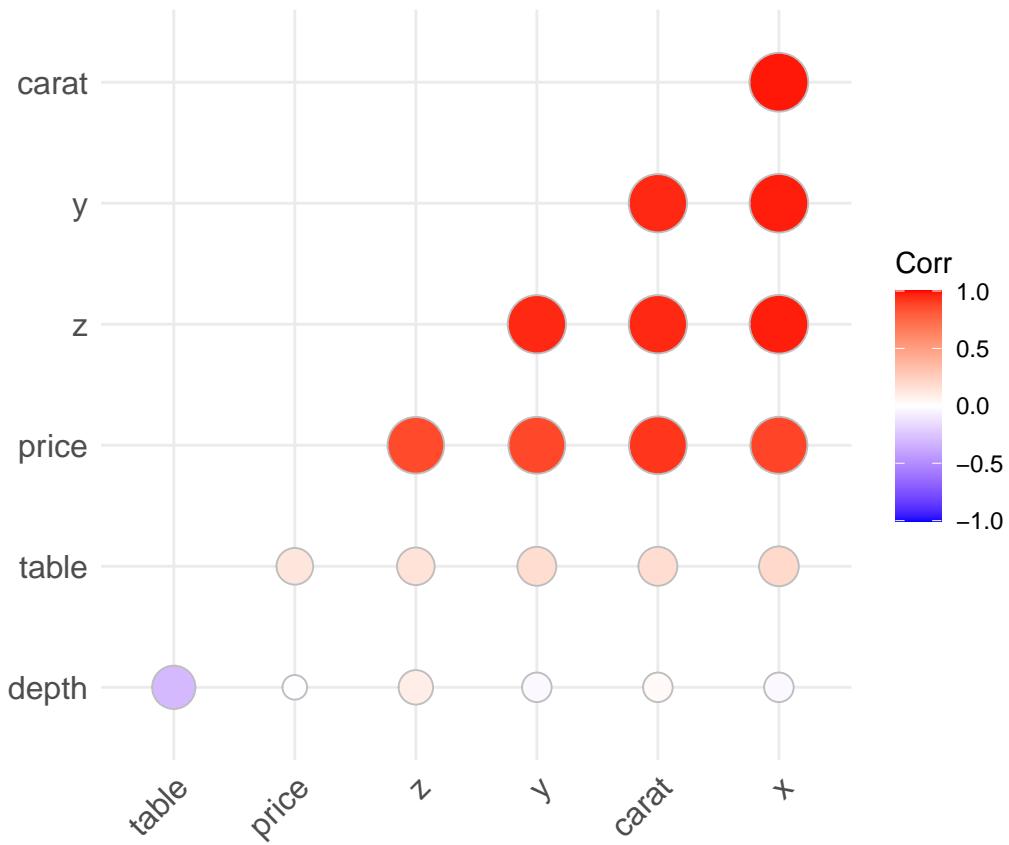


Figure 2: A Visualization of the Correlation Matrix

Figure 2 shows the correlation plot for the numerical variables in the diamonds data. ‘Carat’, ‘x’, ‘y’, ‘z’ and ‘price’ all show very strong correlations with each other, as evidenced by the large red dots. As “x”, “y” and “z” are all measures of size we should expect this and there may be some redundancy in these predictors. The ‘table’ variable is relatively uncorrelated with any of the others. ‘Depth’ and ‘table’ are negatively correlated (large purple dot), while depth is not correlated with any other variable. The strongest predictor of price is carat with length, width, depth also strongly correlated with price. Table is only very weakly correlated with price while depth is negatively correlated with price.

### 3.2.3 Price differences across Categorical Variables

As our leading question is investigating which variables are most predictive of price we decided to investigate if there are significant differences in price across levels of each categorical variable.

#### 3.2.3.1 Cut

## Fair Good Very Good Premium Ideal

```
## 4358.758 3928.864 3981.760 4584.258 3457.542
```

We see that the mean price of diamonds does differ for different levels of “cut”. However, we see that improved cut does not necessarily lead to increased price.

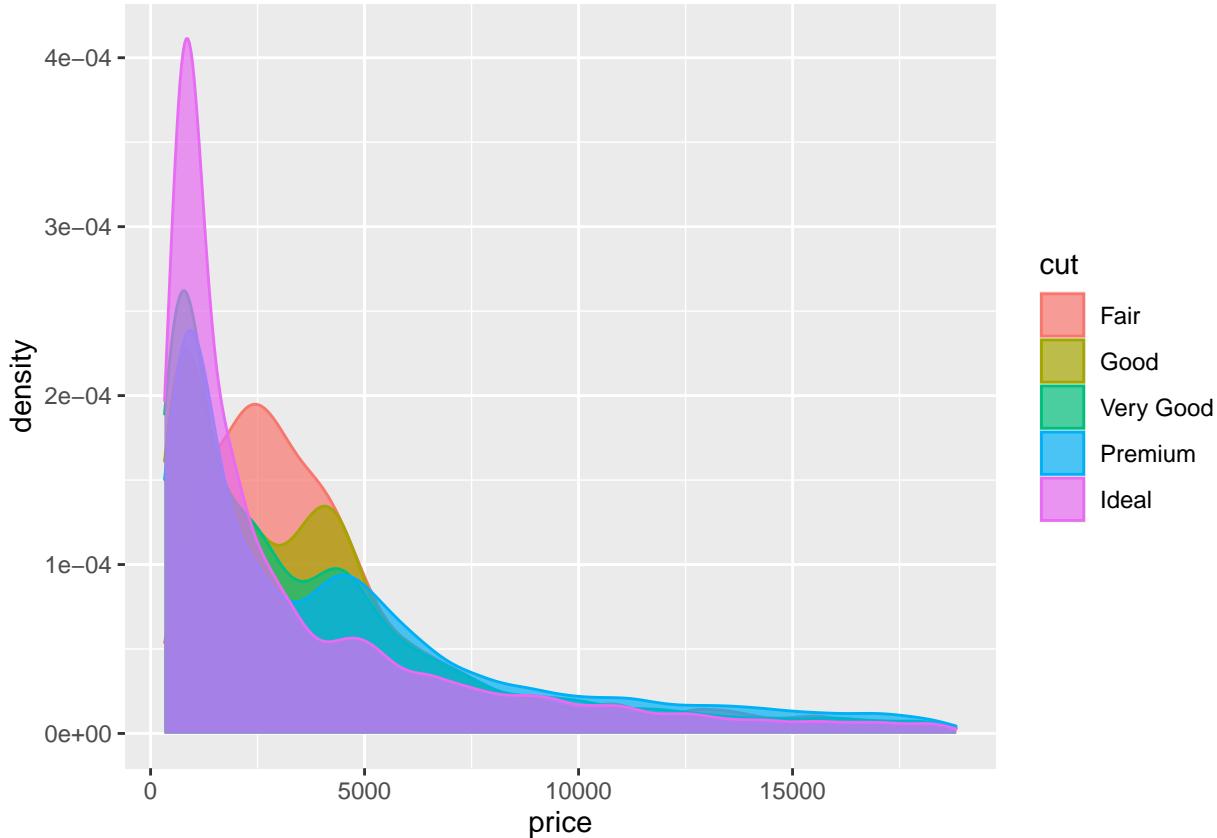


Figure ?? shows the density plots for the different levels of the ‘price’ variable. The legend on the right shows the level names and their order (‘fair’ = poorest, ‘ideal’ = best). Of note is the fact that the two best levels (premium and ideal) have significant peaks near the lower end of the price range compared with the other three. This is somewhat surprising, as intuitively one would imagine price to increase as the quality of cut increases. We conducted a Analysis of Variance to see if these differences in mean price across levels of cut were significant. we also conducted a Levene’s test to verify if the assumption of homogeneity of variance was violated. If significant we then implemented a non-parametric Kruskal Wallis Test. From the graph of the distributions of price for different levels of cut we can see that not all of them have a shape consistent with being normally distributed. A one Way ANOVA is reasonably robust to departures from normality, particularly as we have a very large sample. The results are summarised on the table below and the code for these tests may be found in the appendices.

### Table: ANOVA of Price by Cut

The ANOVA results above return a p-value of  $< 2.2\text{e-}16$ , meaning that there is strong evidence to suggest that mean price differs across levels of “cut”. The Tukey Test indicated that at the

Test	Test Statistic	p-value
ANOVA	175.7	< 2.2e-16
Levene's Test	123.6	< 2.2e-16
Kruskal Wallis	978.62	< 2.2e-16

5% significance level, the only pairs between which we do not see a significant difference in mean price are “very Good” and “Good” as well as “Premium” and “Fair”. Both the Levene’s test and Kruskal Wallis Test return significant results indicating that: a) the assumption of equal variance is violated and; b) that we have evidence of a significant difference in median prices for different levels of cut.

### 3.2.3.2 Clarity

```
##      I1      SI2      SI1      VS2      VS1      VVS2      VVS1      IF
## 3924.169 5063.029 3996.001 3924.989 3839.455 3283.737 2523.115 2864.839
```

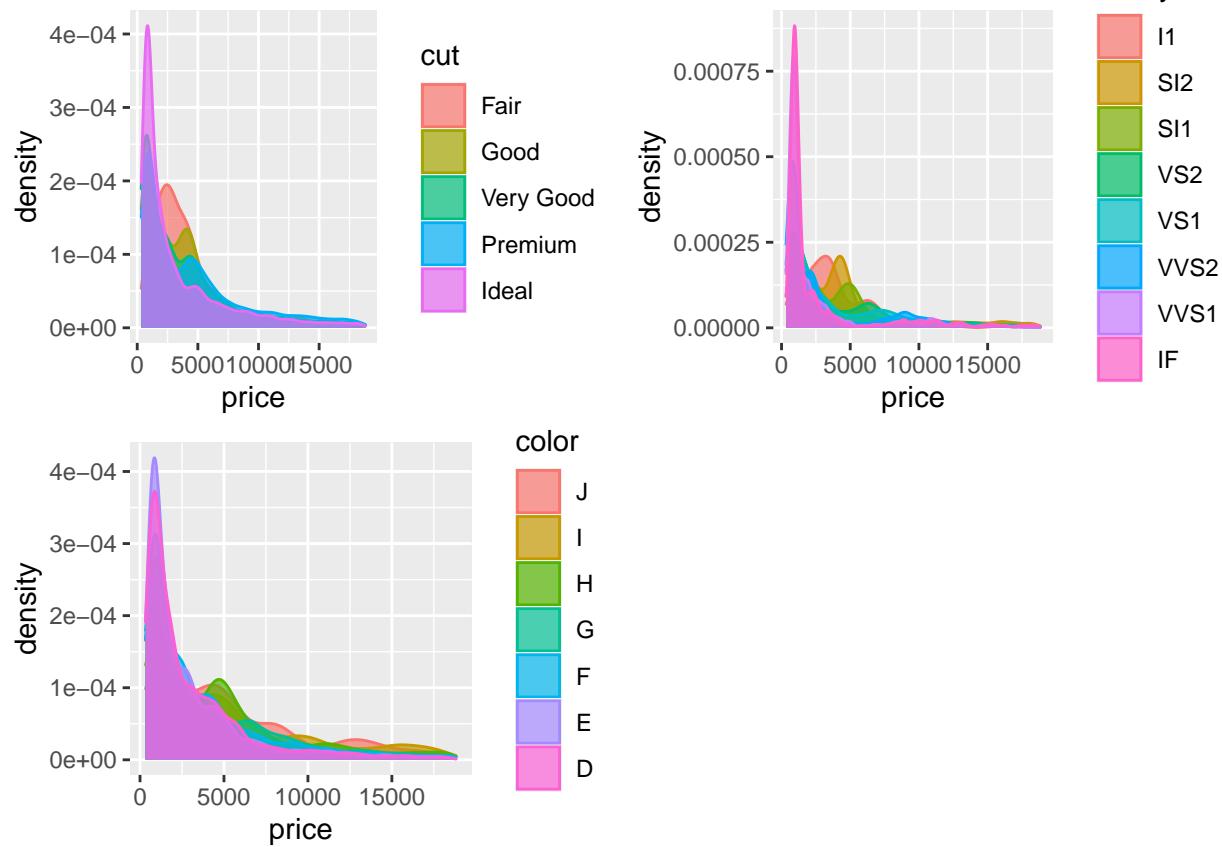
Again, we see an indication the the mean price of diamonds does vary across different levels of clarity. Interestingly the diamonds with the two highest clarities show the lowest mean price.

```
cutplot <- ggplot(subset(diamonds, cut == "Ideal" | cut == "Premium" | cut == "Good" | cut ==
  geom_density(aes(y = ..density..), alpha = .7)

clarityplot <- ggplot(subset(diamonds, clarity == "I1" | clarity == "SI2" | clarity == "SI1" |
  geom_density(aes(y = ..density..), alpha = .7)

colorplot <- ggplot(subset(diamonds, color == "J" | color == "H" | color == "I" | color == "G")
  geom_density(aes(y = ..density..), alpha = .7)

ggarrange(cutplot, clarityplot, colorplot, ncol = 2, nrow = 2)
```



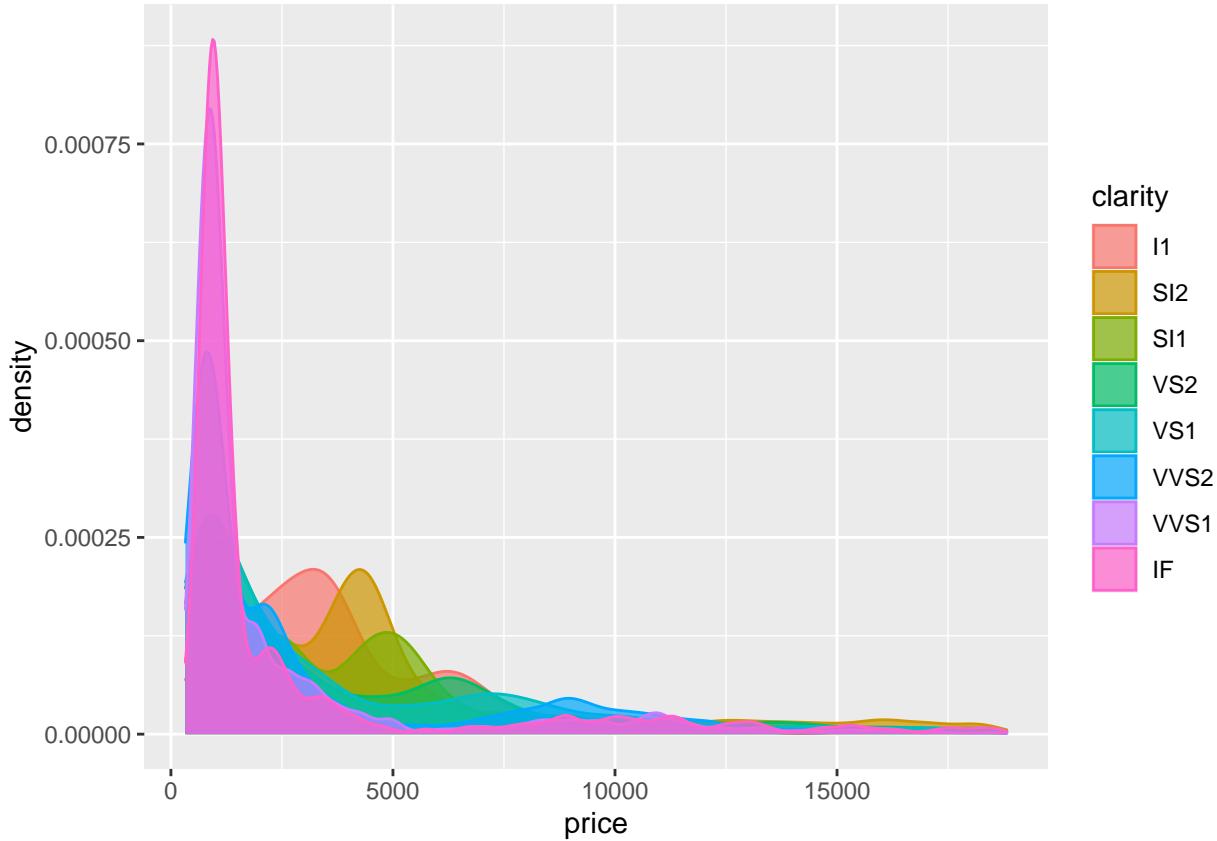


Figure ?? shows the densities of the different levels of ‘clarity’ for price. A prominent peak is visible near the left for the better quality levels (IF, best, pink; WS1 second best, purple; WS2, third best, blue). We conducted a Analysis of Variance to see if these differences in mean price across levels of clarity were significant. While we see potential evidence that the ANOVA assumptions of normality and equal variance may be violated, ANOVA is reasonably robust to these violations if the sample size is big enough. We also conducted a Levene’s test to verify if the assumption of homogeneity of variance was violated. If significant we then implemented a non-parametric Kruskal Wallis Test. The results are summarised on the table below and the code for these tests may be found in the appendices.

**Table: ANOVA of Price by Clarity**

Test	Test Statistic	p-value
ANOVA	215	< 2.2e-16
Levene’s Test	77.809	< 2.2e-16
Kruskal Wallis	2718.2	< 2.2e-16

The output from the ANOVA returns a p-value of  $< 2.2\text{e-}16$ , meaning there is strong evidence to suggest that mean price differs across levels of “clarity”. The output of the Levene’s test and Kruskal Wallis test above show that there is not equal variances between the levels, and that there is a significant difference between different levels of clarity. A tukey test showed that most pairwise combinations show a significant difference. There are only six that do not

show a difference and they are: SI1-I1; VS2-I1; VS1-I1; VS2-SI1; VS1-VS2; and IF-VVS1.

### 3.2.3.3 Color

```
##          J          I          H          G          F          E          D
## 5323.818 5091.875 4486.669 3999.136 3724.886 3076.752 3169.954
```

Again, we see an indication the the mean price of diamonds does vary across different levels of color. Interestingly the diamonds with the two highest color quality show the lowest mean price.

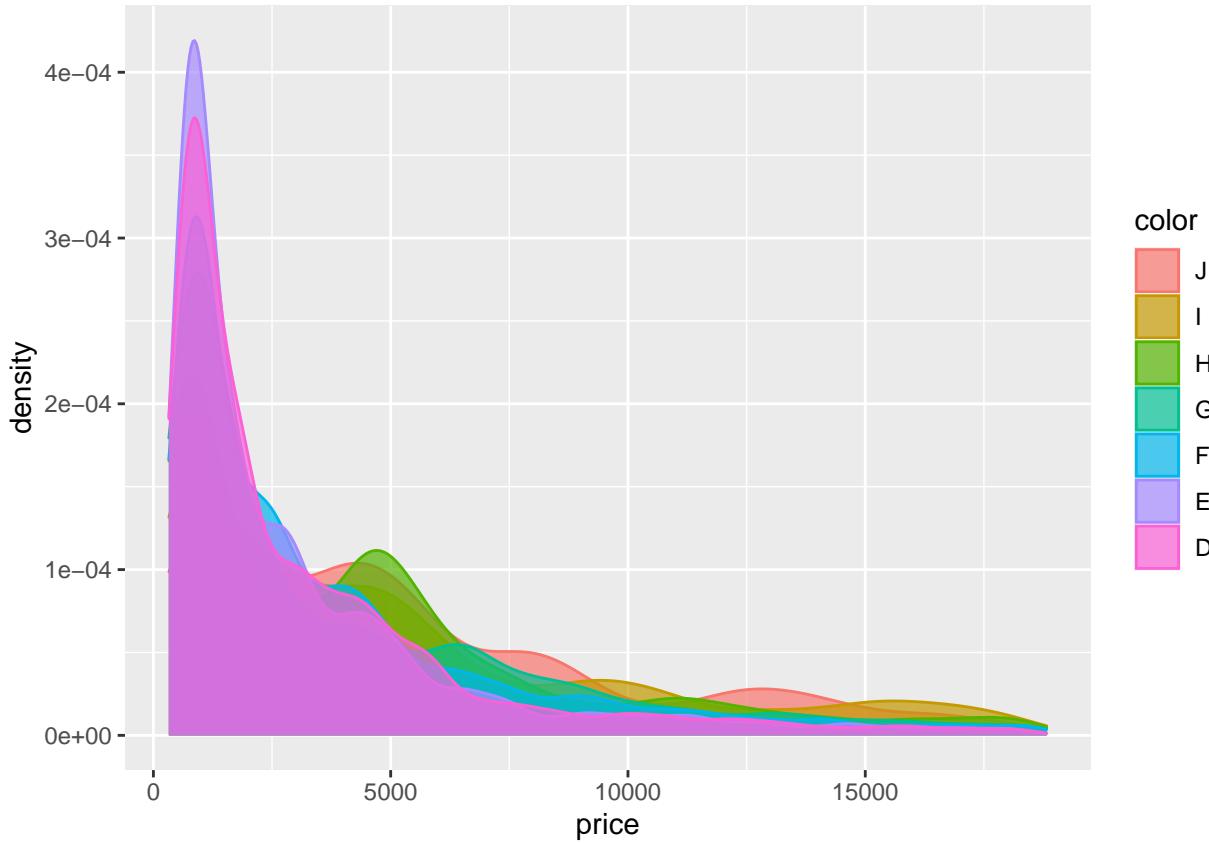


Figure 3: Densities of ‘price’ for the different levels of ‘color’

Figure 3 shows the density plots of ‘price’ for the different levels of the ‘color’ variable. There is a prominent peak on the left for the better quality levels of color (D in pink, E in purple, and F in blue), but otherwise the densities appear to be fairly similar. We will now test whether there are significant differences in mean price for different diamond colours. While we see potential evidence that the ANOVA assumptions of normality and equal variance may be violated, ANOVA is reasonably robust to these violations if the sample size is big enough. We also conducted a Levene’s test to verify if the assumption of homogeneity of variance was violated. If significant we then implemented a non-parametric Kruskal Wallis Test. The

results are summarised on the table below and the code for these tests may be found in the appendices.

**Table: ANOVA of Price by Clarity**

Test	Test Statistic	p-value
ANOVA	175.7	< 2.2e-16
Levene's Test	219.12	< 2.2e-16
Kruskal Wallis	1335.6	< 2.2e-16

We have strong evidence to suggest that mean price differs across levels of “color”. The output of the Levene’s test and Kruskal Wallis test above show that there is not equal variances between the levels, and that there is a significant difference in median price across different levels of clarity. The output from the Tukey Test showed significant differences in mean price for nearly all pairwise comparisons of diamond colors.

### 3.2.4 Outliers and Unusual Points

**3.2.4.1 Mahalanobis Distance** We decided to investigate outliers and unusual points in our dataset. These points may have increased leverage or influence when we come to using variables to predict price. We began by using the Mahalanobis Distance to identify surprising and very surprising points.

```
##  
##      Typical    Somewhat Surprising        very  
##      49611          1001       1489       1839
```

We see from the output above that while the vast majority of diamonds are typical there are a reasonable amount of “Surprising” and “Very Surprising” points in this dataset. We can see how many very surprising points (as identified with the Mahalanobis Distance) are members of each level of our categorical variables. This might help to indicate if any classes contain more surprising points than others.

```
##  
##      Fair      Good Very Good Premium     Ideal  
##      517       216     286      412      408  
  
##  
##      I1     SI2     SI1     VS2     VS1   VVS2   VVS1     IF  
##      209     570     280     251     231     103     105      90  
  
##  
##      J      I      H      G      F      E      D  
##      213    284    338    328    290    216    170
```

**3.2.4.2 Zero Values** In this section we identify any zero values and determine whether or not they are errors.

```
## [1] 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
```

```

## [1] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 3.73 3.73
## [1] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 3.68 3.71 3.71
## [1] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## [16] 0.00 0.00 0.00 0.00 0.00 1.07 1.41
## [1] 43.0 43.0 44.0 50.8 51.0 52.2 52.3 52.7 53.0 53.1
## [1] 43.0 44.0 49.0 49.0 50.0 50.0 50.1 51.0 51.0 51.0
## [1] 326 326 327 334 335 336 336 337 337 338

```

The output above shows that x, y and z all have a number of zero values, which presumably are errors.

### 3.2.5 Idendifying whether the zero observations are errors

## 3.3 Make a data.frame version of dataset

```

diamonds.df <- data.frame(diamonds)
diamonds.df[1:4,]

```

```

##   carat      cut color clarity depth table price     x     y     z surprise
## 1  0.23    Ideal    E    SI2   61.5     55   326 3.95 3.98 2.43 Typical
## 2  0.21 Premium   E    SI1   59.8     61   326 3.89 3.84 2.31 Typical
## 3  0.23     Good   E    VS1   56.9     65   327 4.05 4.07 2.31   very
## 4  0.29 Premium   I    VS2   62.4     58   334 4.20 4.23 2.63 Typical

```

```

# display the zero values for 'x'
diamonds.df[diamonds$x==0,]

```

```

##   carat      cut color clarity depth table price x     y     z surprise
## 11183  1.07    Ideal    F    SI2   61.6     56 4954 0 6.62 0   very
## 11964  1.00 Very Good   H    VS2   63.3     53 5139 0 0.00 0   very
## 15952  1.14      Fair   G    VS1   57.5     67 6381 0 0.00 0   very
## 24521  1.56    Ideal    G    VS2   62.2     54 12800 0 0.00 0   very
## 26244  1.20 Premium   D    VVS1  62.1     59 15686 0 0.00 0   very
## 27430  2.25 Premium   H    SI2   62.8     59 18034 0 0.00 0   very
## 49557  0.71     Good   F    SI2   64.1     60 2130 0 0.00 0   very
## 49558  0.71     Good   F    SI2   64.1     60 2130 0 0.00 0   very

```

```

# display the zero values for 'y'
diamonds.df[diamonds$y==0,]

```

```

##   carat      cut color clarity depth table price x  y  z surprise
## 11964  1.00 Very Good   H    VS2   63.3     53 5139 0 0 0   very
## 15952  1.14      Fair   G    VS1   57.5     67 6381 0 0 0   very
## 24521  1.56    Ideal    G    VS2   62.2     54 12800 0 0 0   very
## 26244  1.20 Premium   D    VVS1  62.1     59 15686 0 0 0   very

```

```

## 27430 2.25 Premium H SI2 62.8 59 18034 0 0 0 very
## 49557 0.71 Good F SI2 64.1 60 2130 0 0 0 very
## 49558 0.71 Good F SI2 64.1 60 2130 0 0 0 very

# display the zero values for 'z'
diamonds.df[diamonds$z==0,]

##      carat    cut color clarity depth table price     x     y     z surprise
## 2208 1.00 Premium G SI2 59.1 59 3142 6.55 6.48 0 very
## 2315 1.01 Premium H I1 58.1 59 3167 6.66 6.60 0 very
## 4792 1.10 Premium G SI2 63.0 59 3696 6.50 6.47 0 very
## 5472 1.01 Premium F SI2 59.2 58 3837 6.50 6.47 0 very
## 10168 1.50 Good G I1 64.0 61 4731 7.15 7.04 0 very
## 11183 1.07 Ideal F SI2 61.6 56 4954 0.00 6.62 0 very
## 11964 1.00 Very Good H VS2 63.3 53 5139 0.00 0.00 0 very
## 13602 1.15 Ideal G VS2 59.2 56 5564 6.88 6.83 0 very
## 15952 1.14 Fair G VS1 57.5 67 6381 0.00 0.00 0 very
## 24395 2.18 Premium H SI2 59.4 61 12631 8.49 8.45 0 very
## 24521 1.56 Ideal G VS2 62.2 54 12800 0.00 0.00 0 very
## 26124 2.25 Premium I SI1 61.3 58 15397 8.52 8.42 0 very
## 26244 1.20 Premium D VVS1 62.1 59 15686 0.00 0.00 0 very
## 27113 2.20 Premium H SI1 61.2 59 17265 8.42 8.37 0 very
## 27430 2.25 Premium H SI2 62.8 59 18034 0.00 0.00 0 very
## 27504 2.02 Premium H VS2 62.7 53 18207 8.02 7.95 0 very
## 27740 2.80 Good G SI2 63.8 58 18788 8.90 8.85 0 very
## 49557 0.71 Good F SI2 64.1 60 2130 0.00 0.00 0 very
## 49558 0.71 Good F SI2 64.1 60 2130 0.00 0.00 0 very
## 51507 1.12 Premium G I1 60.4 59 2383 6.71 6.67 0 very

```

The outputs above show that the zero values for ‘x’, ‘y’ and ‘z’ are all errors. We can tell this because these diamonds have carat, depth and price values, meaning that they cannot have zero length (x), width (y) and depth (z). Note that for a lot of the observations with zero values in one of these variables also have zero values in the others. The variable ‘y’ is a good example; the output shows that every observation with a zero value for ‘y’ also has zero values for ‘x’ and ‘z’.

### 3.3.1 Upper-value outliers

The outputs below show the upper values for the seven numerical variables. This output guides further investigation as to which are likely to be genuine and which are probably errors.

```

# show ten largest values for each of the seven numerical variables
sort(decreasing=T, diamonds$carat)[1:10]

```

```

## [1] 5.01 4.50 4.13 4.01 4.01 4.00 3.67 3.65 3.51 3.50

```

```

sort(decreasing=T, diamonds$x)[1:10]

## [1] 10.74 10.23 10.14 10.02 10.01 10.00 9.86 9.66 9.65 9.54

sort(decreasing=T, diamonds$y)[1:10]

## [1] 58.90 31.80 10.54 10.16 10.10 9.94 9.94 9.85 9.81 9.63

sort(decreasing=T, diamonds$z)[1:10]

## [1] 31.80 8.06 6.98 6.72 6.43 6.38 6.31 6.27 6.24 6.17

sort(decreasing=T, diamonds$depth)[1:10]

## [1] 79.0 79.0 78.2 73.6 72.9 72.2 71.8 71.6 71.6 71.3

sort(decreasing=T, diamonds$table)[1:10]

## [1] 95 79 76 73 73 73 71 70 70

sort(decreasing=T, diamonds$price)[1:10]

## [1] 18823 18818 18806 18804 18803 18797 18795 18795 18791 18791

```

The output above shows the ten largest values for each of the seven numerical variables. At a glance (informal inference) it appears that ‘carat’, ‘y’, ‘z’ and ‘table’ all have one or more values that are considerably higher than the rest, with y and z having maximums that are so extreme it is worth considering whether they are erroneous.

## 3.4 Determining whether the outliers are errors

### 3.4.1 The ‘x’ variable: probably no upper value errors

```

# display the values of 'x' that are greater than or equal to 10
diamonds.df[diamonds$x>=10,]

```

	carat	cut	color	clarity	depth	table	price	x	y	z	surprise
## 25999	4.01	Premium	I	I1	61.0	61	15223	10.14	10.10	6.17	very
## 26000	4.01	Premium	J	I1	62.5	62	15223	10.02	9.94	6.24	very
## 26445	4.00	Very Good	I	I1	63.3	58	15984	10.01	9.94	6.31	very
## 27131	4.13	Fair	H	I1	64.8	61	17329	10.00	9.85	6.43	very
## 27416	5.01	Fair	J	I1	65.5	59	18018	10.74	10.54	6.98	very
## 27631	4.50	Fair	J	I1	65.8	58	18531	10.23	10.16	6.72	very

The output above shows that the largest ‘x’ values are not outrageously larger than the rest. Additionally, the values of the other variables for the largest ‘x’ value are reasonably aligned in terms of magnitude, suggesting that these large ‘x’ values are genuine.

### 3.4.2 The ‘y’ variable: probably 2 upper value errors

```
# display the values of 'x' that are greater than or equal to 11
diamonds.df[diamonds$y>=10,]
```

```
##      carat      cut color clarity depth table price     x     y     z surprise
## 24068  2.00 Premium     H     SI2  58.9    57 12210 8.09 58.90 8.06   very
## 25999  4.01 Premium     I     I1   61.0    61 15223 10.14 10.10 6.17   very
## 27416  5.01 Fair       J     I1   65.5    59 18018 10.74 10.54 6.98   very
## 27631  4.50 Fair       J     I1   65.8    58 18531 10.23 10.16 6.72   very
## 49190  0.51 Ideal      E     VS1  61.8    55 2075  5.15 31.80 5.12   very
```

The output above suggests that the two extreme values for ‘y’ are almost certainly errors. They are 58.90 and 31.80; both are far larger than the next highest value, which is 10.54. It is very unlikely that diamonds with such enormous width values do not also have extreme length and depth values, or that they did not sell for more money.

### 3.4.3 The ‘z’ variable: probably 2 upper value errors

```
diamonds.df[diamonds$z>=8,]
```

```
##      carat      cut color clarity depth table price     x     y     z surprise
## 24068  2.00 Premium     H     SI2  58.9   57.0 12210 8.09 58.90 8.06   very
## 48411  0.51 Very Good   E     VS1  61.8   54.7 1970  5.12  5.15 31.80   very
```

Similarly with the extreme value of ‘z’ (31.80). The other dimensions of this diamond do not tally with the extreme depth value, nor does the relatively low price. The next highest value has been included for comparison (8.06). This is almost certainly an error.

### 3.4.4 The ‘carat’ variable: probably no upper value errors

```
diamonds.df[diamonds.df$carat>4,]
```

```
##      carat      cut color clarity depth table price     x     y     z surprise
## 25999  4.01 Premium     I     I1   61.0    61 15223 10.14 10.10 6.17   very
## 26000  4.01 Premium     J     I1   62.5    62 15223 10.02  9.94 6.24   very
## 27131  4.13 Fair       H     I1   64.8    61 17329 10.00  9.85 6.43   very
## 27416  5.01 Fair       J     I1   65.5    59 18018 10.74 10.54 6.98   very
## 27631  4.50 Fair       J     I1   65.8    58 18531 10.23 10.16 6.72   very
```

The output above shows that the highest value for ‘carat’ (5.01) is not outrageously higher than the next highest (4.50), suggesting that this is a genuine value. Also, the other variables are comparable between the highest and second highest, so it is likely that this value can be trusted.

### 3.4.5 The ‘table’ variable: probably no upper value errors

```
# display the values of 'table' that are greater than or equal to 73
diamonds.df[diamonds.df$table>=73,]
```

```
##      carat cut color clarity depth table price     x     y     z surprise
## 24933  2.01 Fair     F     SI1  58.6    95 13387 8.32 8.31 4.87   very
## 49376  0.70 Fair     H     VS1  62.0    73  2100 5.65 5.54 3.47   very
## 50774  0.81 Fair     F     SI2  68.8    79  2301 5.26 5.20 3.58   very
## 51343  0.79 Fair     G     SI1  65.3    76  2362 5.52 5.13 3.35   very
## 51392  0.71 Fair     D     VS2  55.6    73  2368 6.01 5.96 3.33   very
## 52861  0.50 Fair     E     VS2  79.0    73  2579 5.21 5.18 4.09   very
## 52862  0.50 Fair     E     VS2  79.0    73  2579 5.21 5.18 4.09   very
```

The output above for the highest values of ‘table’ show that the diamond with the largest value (95) was also considerably larger in other ways and was much more expensive. This suggests that this is a genuine value, rather than an error.

### 3.4.6 ‘depth’ and ‘price’: probably no upper value errors

```
sort(diamonds.df$depth, decreasing = TRUE)[1:10]
```

```
## [1] 79.0 79.0 78.2 73.6 72.9 72.2 71.8 71.6 71.6 71.3
```

```
sort(diamonds.df$price, decreasing = TRUE)[1:10]
```

```
## [1] 18823 18818 18806 18804 18803 18797 18795 18795 18791 18791
```

The output above shows that neither ‘depth’ nor ‘price’ have any extreme values. We can probably safely conclude that these are all genuine upper values.

## 3.5 Colour-coded scatterplots

### 3.5.1 Scatterplot of ‘Carat’ vs ‘Price’, colour-coded by ‘Cut’

```
# scatterplot of carat vs price, but also colouring the observations
# by 'cut'
ggplot(data=diamonds, aes(x=carat, y=price, color=cut))+
  geom_point()+
  guides(colour=guide_legend(reverse = T))
```

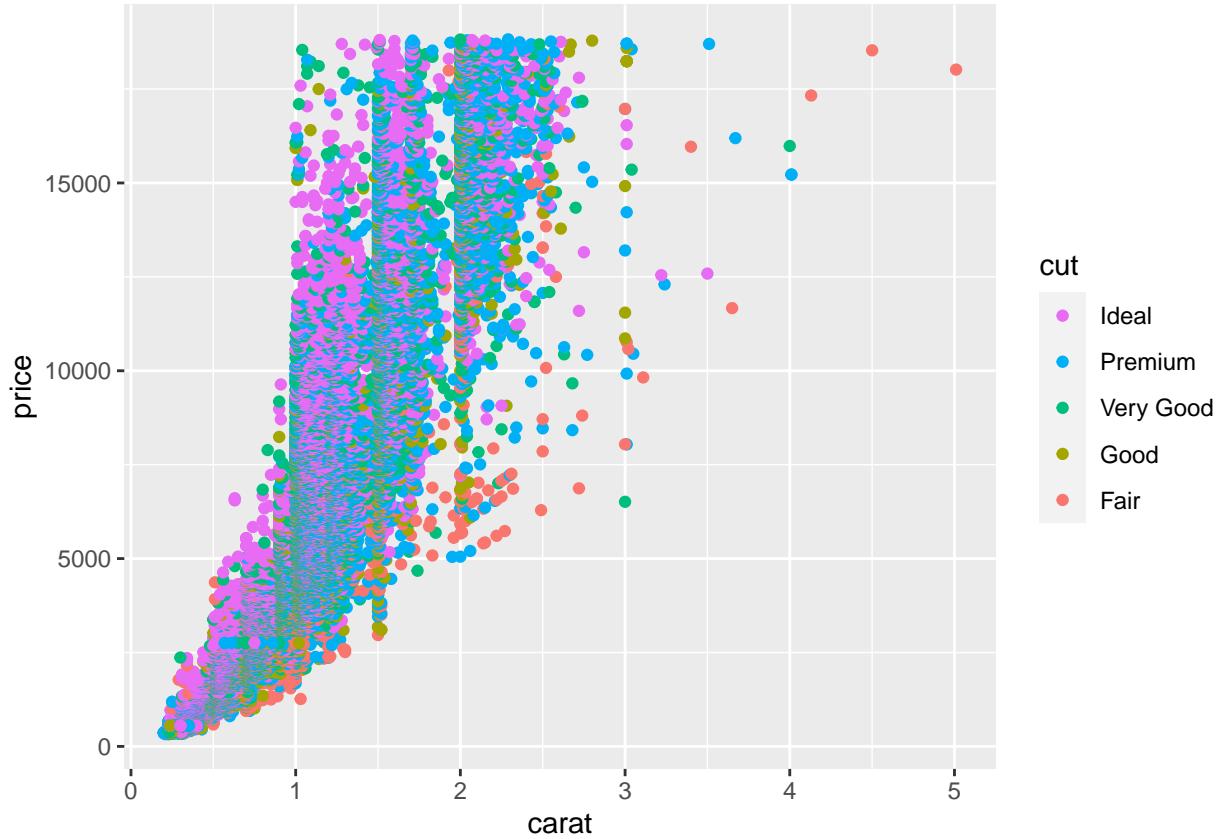


Figure 4: Carat vs Price, coloured by Cut

Figure 4 shows the scatterplot of ‘carat’ versus ‘price’ and coloured by ‘cut’.

### 3.5.2 Scatterplot of ‘Carat’ vs ‘Price’, colour-coded by ‘Clarity’

```
# scatterplot of carat vs price, but also colouring the observations
# by 'cut'
ggplot(data=diamonds, aes(x=carat, y=price, color=clarity))+
  geom_point()+
  guides(colour=guide_legend(reverse = T))
```

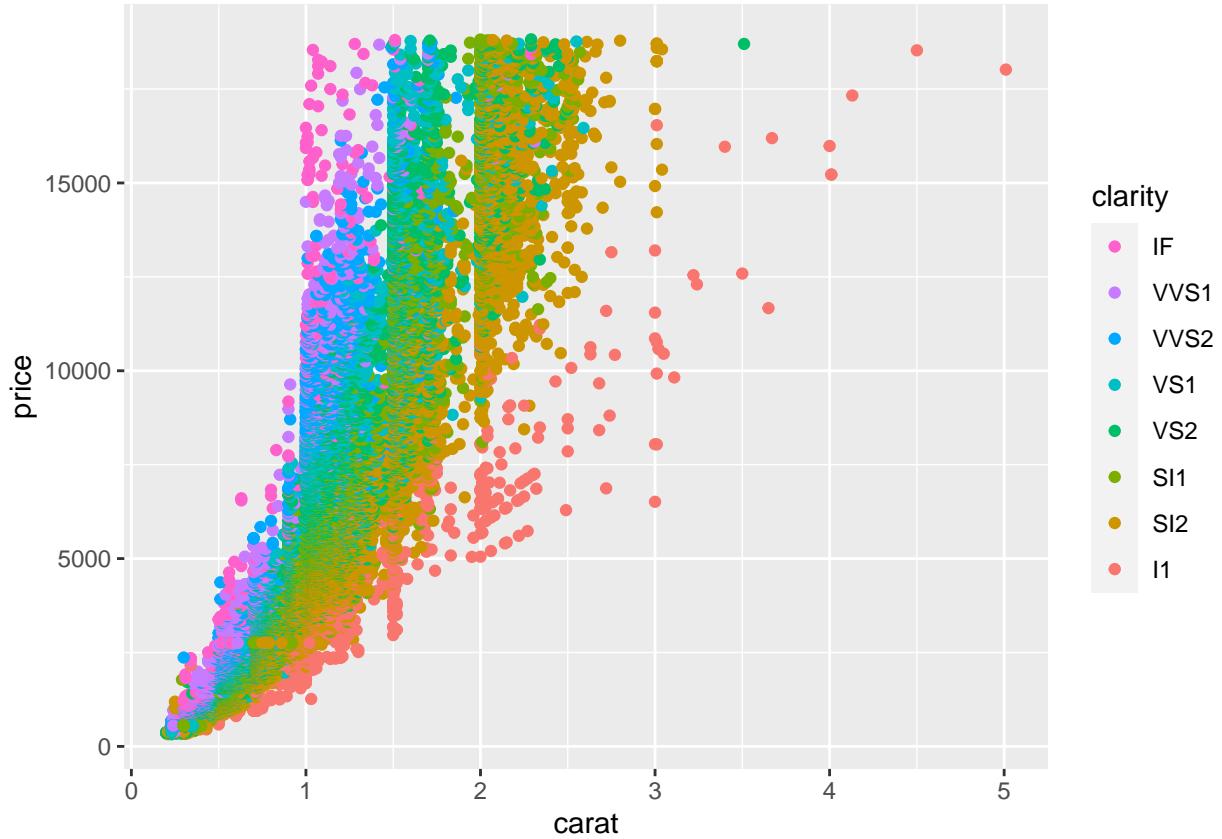


Figure 5: Carat vs Price coloured by Clarity

Figure 5 shows the scatterplot of ‘carat’ versus ‘price’ and coloured by ‘clarity’.

### 3.5.3 Scatterplot of ‘Carat’ vs ‘Price’, colour coded by ‘Color’

```
# scatterplot of carat vs price, but also colouring the observations
# by 'cut'
ggplot(data=diamonds, aes(x=carat, y=price, color=color))+
  geom_point()+
  guides(colour=guide_legend(reverse = T))
```

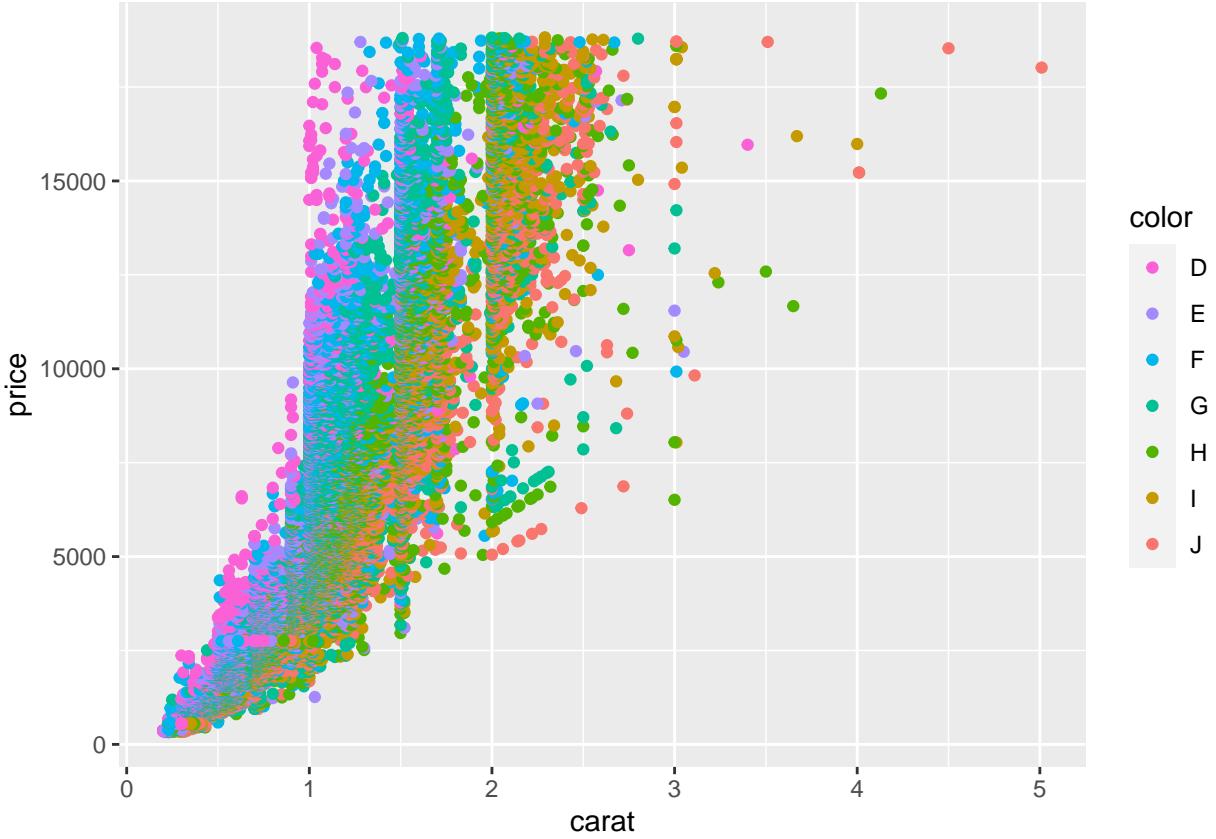


Figure 6: Carat vs Price coloured by Color

Figure 6 shows the scatterplot of ‘carat’ versus ‘price’ and coloured by ‘color’.

In the three scatterplots figures 4, 5 and 6 we see a clear trend of lighter diamonds (lower ‘carat’ values) that are most expensive (so, in the upper left of the plot) having the highest quality of ‘cut’, ‘clarity’ and ‘color’. This is no surprise, as we would expect the most expensive lighter diamonds to be of the best quality.

Conversely, we can also see some heavier diamonds which are of lower quality and lower price. Note in figure 4 that some of the lowest quality diamonds in terms of ‘cut’ (orange dots in the centre of the plot, on the vertical value of 2) are relatively cheap, despite being heavier than some of the lighter more expensive diamonds. But eventually the weight of the diamonds drives the price up regardless of the quality of ‘cut’, as we can see from the three orange dots in the top right of the plot. These are of the lowest quality cut, but are the three heaviest diamonds in the dataset, and so end up being expensive.

### 3.5.4 Interactions

As we have 3 categorical variables: color, clarity and cut with 7, 8 and 5 levels respectively we have 280 separate interactions in our data set. This makes it likely that there will be unknown classes in our data.

### 3.5.5 Simple and Multiple Regression

As we are interested in predicting price we thought it would be appropriate to do an initial multiple regression including all variables to see which are thought to be significant in predicting price when all other variables are included in the model. We have done this firstly with the raw variables and then with scaled versions of the variables.

## 4 Simple linear regression using ‘carat’

```
## [1] 0.9216
```

We theorised that ‘carat’ would be a very good predictor of ‘price’ based on the fact that ‘carat’ was the variable most highly correlated with ‘price’, with a correlation of 0.9216 between the two variables. Here we fit a simple linear regression model using ‘carat’ as the sole predictor variable.

```
# fit the linear regression model
carat.lm <- lm(price~carat, data = diamonds, x=T)

# display summary
summary(carat.lm)

## 
## Call:
## lm(formula = price ~ carat, data = diamonds, x = T)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18585.3   -804.8    -18.9    537.4   12731.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2256.36     13.06  -172.8   <2e-16 ***
## carat        7756.43     14.07   551.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1549 on 53938 degrees of freedom
## Multiple R-squared:  0.8493, Adjusted R-squared:  0.8493 
## F-statistic: 3.041e+05 on 1 and 53938 DF,  p-value: < 2.2e-16

# get anova table of linear regression model
anova(carat.lm)

## Analysis of Variance Table
##
## Response: price
```

```

##                Df      Sum Sq   Mean Sq F value    Pr(>F)
## carat           1 7.2913e+11 7.2913e+11 304051 < 2.2e-16 ***
## Residuals  53938 1.2935e+11 2.3980e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The output from the linear regression model above shows that ‘carat’ is a good predictor of ‘price’. The t-test statistic is  $3.0405091 \times 10^5$ , which is an enormous number, and the p-value is 0. So, with a p-value of zero, we have very strong evidence to say that ‘carat’ is a good predictor of ‘price’.

## 4.1 Testing to find the best regression model

### 4.1.1 The Akaike Information Criterion (AIC) test

We now use the Akaike Information Criterion (AIC) stepwise regression test in R to find the best regression model that has ‘price’ as the response variable. We are particularly interested in whether any other models outperform the simple ‘carat’ model.

```

# fit the linear regression model with all predictors
full.lm <- lm(price ~ ., data = diamonds)

# display the model summary
summary(full.lm)

```

```

##
## Call:
## lm(formula = price ~ ., data = diamonds)
##
## Residuals:
##      Min       1Q     Median      3Q      Max
## -16944.1   -570.5   -139.7    419.5   8938.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5828.406   392.009 -14.868 <2e-16 ***
## carat        9121.249   53.109 171.746 <2e-16 ***
## cutGood      1273.964   33.012  38.592 <2e-16 ***
## cutVery Good 1564.299   32.268  48.479 <2e-16 ***
## cutPremium   1626.320   32.373  50.237 <2e-16 ***
## cutIdeal     1623.098   33.122  49.003 <2e-16 ***
## colorI        887.044   24.837  35.714 <2e-16 ***
## colorH        1371.051   23.474  58.406 <2e-16 ***
## colorG        1827.242   22.940  79.654 <2e-16 ***
## colorF        1998.481   23.431  85.292 <2e-16 ***
## colorE        2047.623   23.543  86.975 <2e-16 ***

```

```

## colorD           2247.937    24.690   91.045 <2e-16 ***
## claritySI2      2927.291    41.490   70.554 <2e-16 ***
## claritySI1      3895.619    41.314   94.292 <2e-16 ***
## clarityVS2      4463.168    41.498  107.552 <2e-16 ***
## clarityVS1      4756.397    42.115  112.939 <2e-16 ***
## clarityVVS2     5087.090    43.342  117.372 <2e-16 ***
## clarityVVS1     5109.707    44.538  114.726 <2e-16 ***
## clarityIF        5402.446    48.149  112.202 <2e-16 ***
## depth            -9.174     4.329   -2.119  0.0341 *
## table            -43.533    2.753   -15.810 <2e-16 ***
## x                 -364.581   32.082  -11.364 <2e-16 ***
## y                  10.338    18.233   0.567  0.5707
## z                  8.243     31.589   0.261  0.7941
## surpriseSomewhat  1434.789   35.950  39.911 <2e-16 ***
## surpriseSurprising 1988.131   31.299  63.521 <2e-16 ***
## surpriseevery     2011.435   31.910  63.036 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1066 on 53913 degrees of freedom
## Multiple R-squared:  0.9287, Adjusted R-squared:  0.9287
## F-statistic: 2.7e+04 on 26 and 53913 DF, p-value: < 2.2e-16

# use the step() function to perform the stepwise AIC test
step(full.lm, direction = "both")

## Start:  AIC=752091.5
## price ~ carat + cut + color + clarity + depth + table + x + y +
##       z + surprise
##
##          Df  Sum of Sq      RSS      AIC
## - z         1 7.7325e+04 6.1220e+10 752090
## - y         1 3.6502e+05 6.1220e+10 752090
## <none>                6.1220e+10 752091
## - depth     1 5.1004e+06 6.1225e+10 752094
## - x         1 1.4664e+08 6.1367e+10 752219
## - table     1 2.8383e+08 6.1504e+10 752339
## - cut        4 3.0704e+09 6.4290e+10 754723
## - surprise   3 7.6369e+09 6.8857e+10 758426
## - color      6 1.4894e+10 7.6114e+10 763826
## - carat      1 3.3495e+10 9.4715e+10 775629
## - clarity     7 3.4434e+10 9.5654e+10 776149
##
## Step:  AIC=752089.6
## price ~ carat + cut + color + clarity + depth + table + x + y +

```

```

##      surprise
##
##          Df  Sum of Sq      RSS      AIC
## - y           1 4.1804e+05 6.1220e+10 752088
## <none>          6.1220e+10 752090
## + z           1 7.7325e+04 6.1220e+10 752091
## - depth       1 5.5742e+06 6.1226e+10 752092
## - x            1 1.9595e+08 6.1416e+10 752260
## - table        1 2.8390e+08 6.1504e+10 752337
## - cut           4 3.0706e+09 6.4291e+10 754721
## - surprise      3 7.6397e+09 6.8860e+10 758427
## - color          6 1.4895e+10 7.6115e+10 763824
## - carat          1 3.3509e+10 9.4729e+10 775635
## - clarity         7 3.4444e+10 9.5665e+10 776153
##
## Step: AIC=752087.9
## price ~ carat + cut + color + clarity + depth + table + x + surprise
##
##          Df  Sum of Sq      RSS      AIC
## <none>          6.1220e+10 752088
## + y           1 4.1804e+05 6.1220e+10 752090
## + z           1 1.3034e+05 6.1220e+10 752090
## - depth       1 5.6842e+06 6.1226e+10 752091
## - table        1 2.8444e+08 6.1505e+10 752336
## - x            1 3.0884e+08 6.1529e+10 752357
## - cut           4 3.0720e+09 6.4292e+10 754721
## - surprise      3 7.6394e+09 6.8860e+10 758425
## - color          6 1.4895e+10 7.6116e+10 763823
## - carat          1 3.3543e+10 9.4763e+10 775652
## - clarity         7 3.4452e+10 9.5673e+10 776156
##
## Call:
## lm(formula = price ~ carat + cut + color + clarity + depth +
##     table + x + surprise, data = diamonds)
##
## Coefficients:
## (Intercept)      carat      cutGood      cutVery Good
## -5849.415       9122.429      1274.554      1565.069
## cutPremium      cutIdeal      colorI       colorH
## 1626.305        1623.533      887.011      1371.040
## colorG          colorF       colorE       colorD
## 1827.224        1998.501      2047.676      2247.940
## claritySI2      claritySI1      clarityVS2      clarityVS1
## 2927.744        3896.077      4463.624      4756.964

```

```

##          clarityVVS2      clarityVVS1      clarityIF      depth
##          5087.604       5110.213       5403.030      -8.779
##          table           x surpriseSomewhat surpriseSurprising
##          -43.570        -349.684        1434.563      1987.869
##          surpriseevery
##          2011.350

```

Based on the output from the stepwise AIC regression comparison above, the best model which balances accuracy against parsimony is the one with predictors carat + cut + color + clarity + depth + table + x. In other words, the predictors that were dropped are y and z. The AIC value for this model is 758425 compared with 758426 for the next best. The difference between the two is only 1, but the principle of parsimony dictates that we select the simplest model, which means selecting the one that drops the variable ‘y’.

#### 4.1.2 Bayes Information Criterion (BIC) model comparison

```

# store the sample size in a variable
n <- length(diamonds$price)

# repeat the step test with the added 'k' argument to perform the BIC
step(full.lm, direction = "both", k=log(n))

## Start:  AIC=752331.7
## price ~ carat + cut + color + clarity + depth + table + x + y +
##         z + surprise
##
##          Df  Sum of Sq      RSS      AIC
## - z      1 7.7325e+04 6.1220e+10 752321
## - y      1 3.6502e+05 6.1220e+10 752321
## - depth  1 5.1004e+06 6.1225e+10 752325
## <none>          6.1220e+10 752332
## - x      1 1.4664e+08 6.1367e+10 752450
## - table  1 2.8383e+08 6.1504e+10 752570
## - cut    4 3.0704e+09 6.4290e+10 754928
## - surprise 3 7.6369e+09 6.8857e+10 758640
## - color   6 1.4894e+10 7.6114e+10 764012
## - carat   1 3.3495e+10 9.4715e+10 775860
## - clarity 7 3.4434e+10 9.5654e+10 776327
##
## Step:  AIC=752320.8
## price ~ carat + cut + color + clarity + depth + table + x + y +
##         surprise
##
##          Df  Sum of Sq      RSS      AIC
## - y      1 4.1804e+05 6.1220e+10 752310

```

```

## - depth      1 5.5742e+06 6.1226e+10 752315
## <none>          6.1220e+10 752321
## + z           1 7.7325e+04 6.1220e+10 752332
## - x           1 1.9595e+08 6.1416e+10 752482
## - table       1 2.8390e+08 6.1504e+10 752560
## - cut          4 3.0706e+09 6.4291e+10 754917
## - surprise     3 7.6397e+09 6.8860e+10 758631
## - color         6 1.4895e+10 7.6115e+10 764002
## - carat        1 3.3509e+10 9.4729e+10 775857
## - clarity       7 3.4444e+10 9.5665e+10 776322
##
## Step: AIC=752310.3
## price ~ carat + cut + color + clarity + depth + table + x + surprise
##
##             Df  Sum of Sq      RSS      AIC
## - depth      1 5.6842e+06 6.1226e+10 752304
## <none>          6.1220e+10 752310
## + y           1 4.1804e+05 6.1220e+10 752321
## + z           1 1.3034e+05 6.1220e+10 752321
## - table       1 2.8444e+08 6.1505e+10 752549
## - x           1 3.0884e+08 6.1529e+10 752571
## - cut          4 3.0720e+09 6.4292e+10 754908
## - surprise     3 7.6394e+09 6.8860e+10 758621
## - color         6 1.4895e+10 7.6116e+10 763992
## - carat        1 3.3543e+10 9.4763e+10 775866
## - clarity       7 3.4452e+10 9.5673e+10 776316
##
## Step: AIC=752304.4
## price ~ carat + cut + color + clarity + table + x + surprise
##
##             Df  Sum of Sq      RSS      AIC
## <none>          6.1226e+10 752304
## + depth      1 5.6842e+06 6.1220e+10 752310
## + y           1 5.2800e+05 6.1226e+10 752315
## + z           1 4.3708e+05 6.1226e+10 752315
## - table       1 2.9741e+08 6.1524e+10 752555
## - x           1 3.0951e+08 6.1536e+10 752566
## - cut          4 3.8813e+09 6.5107e+10 755576
## - surprise     3 7.9739e+09 6.9200e+10 758875
## - color         6 1.4904e+10 7.6130e+10 763991
## - clarity       7 3.4639e+10 9.5865e+10 776413
## - carat        1 3.6138e+10 9.7364e+10 777315
##
## Call:

```

```

## lm(formula = price ~ carat + cut + color + clarity + table +
##     x + surprise, data = diamonds)
##
## Coefficients:
## (Intercept)          carat       cutGood    cutVery Good
## -6603.96           9089.14      1292.27    1589.67
## cutPremium         cutIdeal      colorI      colorH
## 1653.71            1653.27      886.95     1370.89
## colorG             colorF       colorE      colorD
## 1827.02            1998.75      2048.11    2248.29
## claritySI2        claritySI1    clarityVS2  clarityVS1
## 2931.39            3898.97      4467.37    4761.35
## clarityVVS2       clarityVVS1   clarityIF    table
## 5092.38            5115.44      5409.38    -41.24
## x     surpriseSomewhat surpriseSurprising surpriseevery
## -336.70            1442.69      1996.17    2022.31

```

The model comparison using the Bayes Information Criterion (BIC) confirms that the best model is the one with variables carat + cut + color + clarity + depth + table + x (output above). The BIC value for this model is 758621, compared with 758629 for the next best.

## 4.2 Fitting the best model

The ‘best’ model (as found by the AIC test above) is fitted below.

```

best.lm <- lm(price ~ carat + cut + color + clarity +
                 depth + table + x, data = diamonds)

summary(best.lm)

##
## Call:
## lm(formula = price ~ carat + cut + color + clarity + depth +
##     table + x, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21385.0  -592.4  -183.7   376.5 10694.6
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.418     391.056  -0.009   0.993
## carat       11256.968    48.600 231.626 <2e-16 ***
## cutGood      580.240    33.572 17.283 <2e-16 ***
## cutVery Good  726.820    32.212 22.564 <2e-16 ***
## cutPremium   762.759    32.225 23.670 <2e-16 ***

```

```

## cutIdeal      833.260    33.396   24.951   <2e-16 ***
## colorI       903.322    26.337   34.299   <2e-16 ***
## colorH      1389.382    24.890   55.820   <2e-16 ***
## colorG      1887.561    24.313   77.637   <2e-16 ***
## colorF      2096.670    24.813   84.497   <2e-16 ***
## colorE      2160.267    24.922   86.682   <2e-16 ***
## colorD      2369.504    26.131   90.678   <2e-16 ***
## claritySI2   2702.077    43.812   61.674   <2e-16 ***
## claritySI1   3664.905    43.627   84.005   <2e-16 ***
## clarityVS2   4266.612    43.847   97.308   <2e-16 ***
## clarityVS1   4577.589    44.535   102.786   <2e-16 ***
## clarityVVS2  4950.168    45.847   107.972   <2e-16 ***
## clarityVVS1  5007.061    47.152   106.190   <2e-16 ***
## clarityIF     5344.338    51.015   104.761   <2e-16 ***
## depth        -66.769     4.091   -16.322   <2e-16 ***
## table        -26.457     2.911   -9.089   <2e-16 ***
## x            -1029.478    20.549   -50.098   <2e-16 ***
##
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1130 on 53918 degrees of freedom
## Multiple R-squared:  0.9198, Adjusted R-squared:  0.9198
## F-statistic: 2.944e+04 on 21 and 53918 DF,  p-value: < 2.2e-16

```

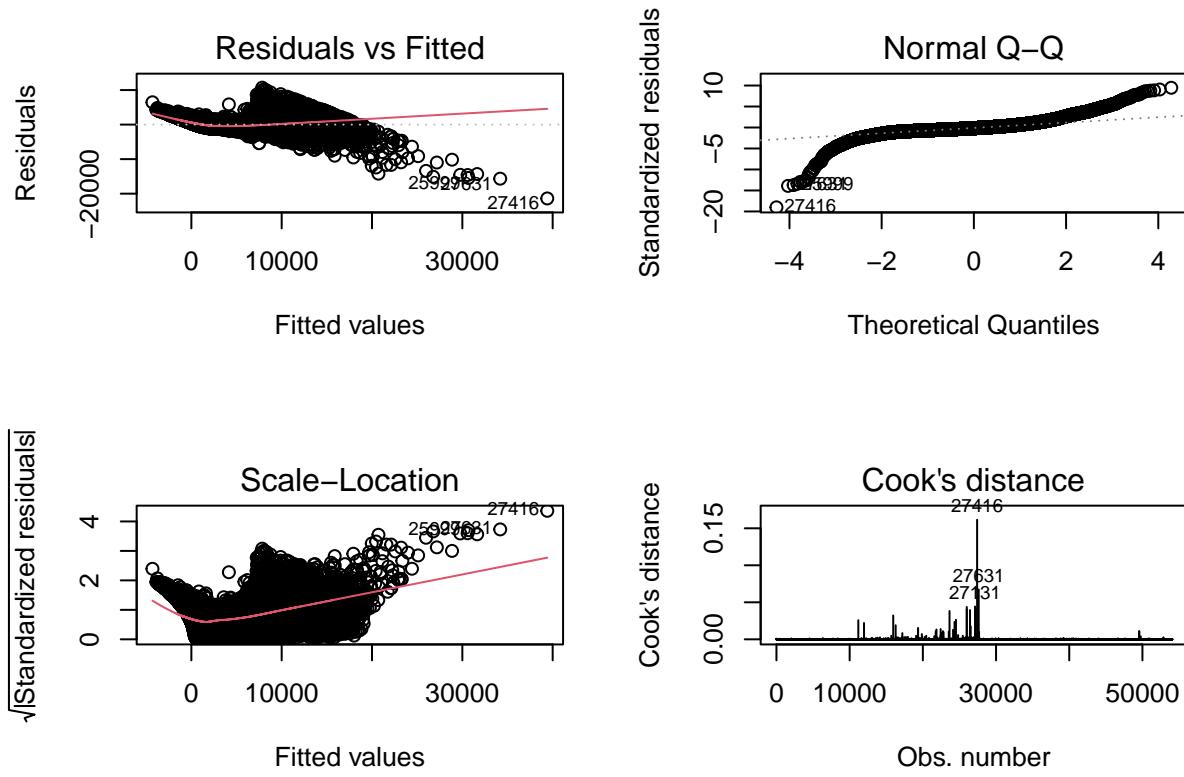
The summary output for the ‘best’ model above shows how the categorical variables, all of which measure diamond quality, add a lot to the accuracy of the model. For example, ‘Clarity’, which has 8 levels, adds from 2702.077 (lowest quality clarity) to 5344.338 (highest quality clarity) to price depending on which level the diamond is categorised at. This is a wide range and cannot be obtained with these categorical variables dropped from the model.

### 4.3 Check Regression Assumptions

```

par(mfrow=c(2,2))
plot(best.lm, 1)
plot(best.lm, 2)
plot(best.lm, 3)
plot(best.lm, 4)

```



By examining the residual plots we see that it is likely that the assumptions underlying a multiple linear regression such as linearity, homogeneity of the variance and normality of the residuals are violated. There also appears to be a number of points with high leverage and influence. We should therefore be cautious in concluding much from this regression model.

## 4.4 Scaled multiple regression model

In this version we scale the variables.

```
diamonds.lm <- lm(price ~ scale(carat) + scale(x) + scale(y) + scale(z) + scale(depth) +
summary(diamonds.lm)
```

```
##
## Call:
## lm(formula = price ~ scale(carat) + scale(x) + scale(y) + scale(z) +
##     scale(depth) + scale(table) + cut + clarity + color, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -21376.0   -592.4   -183.5    376.4  10694.2 
##
## Coefficients:
```

```

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2564.403    53.433 -47.993 <2e-16 ***
## scale(carat) 5335.934   23.050 231.494 <2e-16 ***
## scale(x)     -1131.028   36.903 -30.648 <2e-16 ***
## scale(y)      10.975    22.081   0.497   0.619
## scale(z)     -35.369    23.631  -1.497   0.134
## scale(depth) -91.410    6.496 -14.071 <2e-16 ***
## scale(table) -59.156    6.506  -9.092 <2e-16 ***
## cutGood       579.751   33.592  17.259 <2e-16 ***
## cutVery Good 726.783   32.241  22.542 <2e-16 ***
## cutPremium    762.144   32.228  23.649 <2e-16 ***
## cutIdeal      832.912   33.407  24.932 <2e-16 ***
## claritySI2    2702.586   43.818  61.677 <2e-16 ***
## claritySI1    3665.472   43.634  84.005 <2e-16 ***
## clarityVS2    4267.224   43.853  97.306 <2e-16 ***
## clarityVS1    4578.398   44.546 102.779 <2e-16 ***
## clarityVVS2   4950.814   45.855 107.967 <2e-16 ***
## clarityVVS1   5007.759   47.160 106.187 <2e-16 ***
## clarityIF     5345.102   51.024 104.757 <2e-16 ***
## colorI        903.154    26.337  34.292 <2e-16 ***
## colorH       1389.131    24.891  55.809 <2e-16 ***
## colorG       1887.359    24.313  77.628 <2e-16 ***
## colorF       2096.544    24.813  84.492 <2e-16 ***
## colorE       2160.280    24.922  86.683 <2e-16 ***
## colorD       2369.398    26.131  90.674 <2e-16 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1130 on 53916 degrees of freedom
## Multiple R-squared:  0.9198, Adjusted R-squared:  0.9198
## F-statistic: 2.688e+04 on 23 and 53916 DF,  p-value: < 2.2e-16

```

An initial multiple regression indicates that all variables but y and z are significant in the model (once the effect of the other predictors has been accounted for). This model accounts of 91.98% of the variation in the data.

## 5 Principal Component Analysis

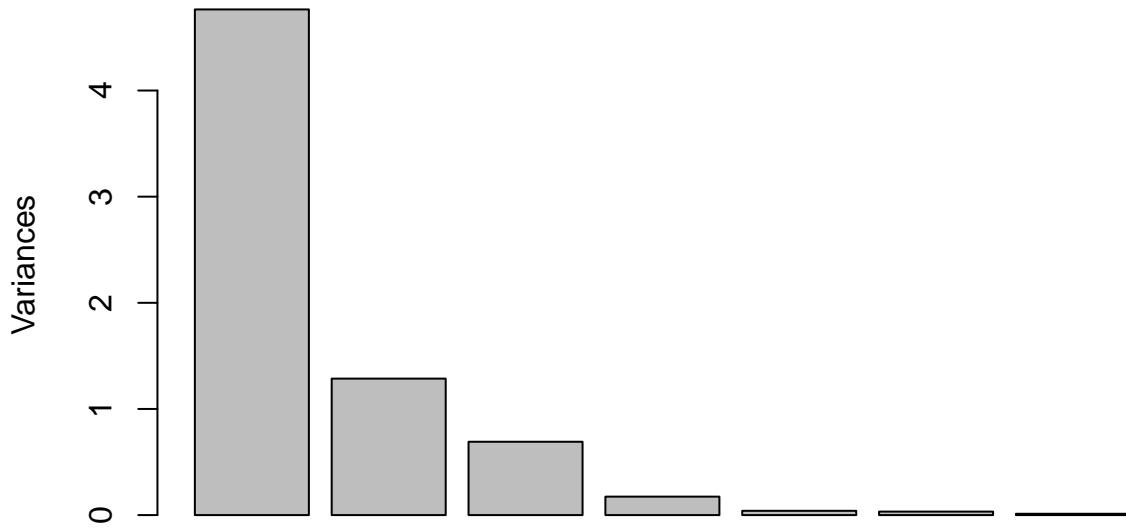
We will perform a PCA with the aim of being able to explain the variation in the data set with fewer dimensions. We also hope to then create a parsimonious model for predicting diamond price through principal components regression.

```

pca.diamonds <- prcomp(subset(diamonds.df, select = c(1,5:10)),
                        center=TRUE, scale. = TRUE, retx=TRUE)
plot(pca.diamonds)

```

## pca.diamonds



```
summary(pca.diamonds)
```

```
## Importance of components:  
##           PC1      PC2      PC3      PC4      PC5      PC6      PC7  
## Standard deviation 2.1826 1.1340 0.83115 0.41684 0.20077 0.18151 0.11135  
## Proportion of Variance 0.6806 0.1837 0.09869 0.02482 0.00576 0.00471 0.00177  
## Cumulative Proportion 0.6806 0.8642 0.96294 0.98776 0.99352 0.99823 1.00000
```

We see that the first principal component explains the majority of the variance (68%) in the dataset with second (18%) and third (10%) principal components explaining a noticeable amount. Components beyond the third principal component explain minimal variance (less than 4%). This indicates we can describe the data using only the first two or three principal components without losing much information. We will now examine which variables contribute most strongly to each Principal Component.

```
pca.diamonds$rotation
```

```
##           PC1      PC2      PC3      PC4      PC5  
## carat  0.4524454941 -0.034696011  0.005494814 -0.06835945  0.13399948  
## depth -0.0009161301 -0.730679714 -0.672829294 -0.04724800 -0.08873829  
## table  0.0995160875  0.675067376 -0.728069469 -0.05954060 -0.01037614  
## price  0.4255192667 -0.035257945  0.105449477 -0.84977817 -0.05377206  
## x      0.4532125054  0.003512550  0.039508824  0.24299509  0.08898016
```

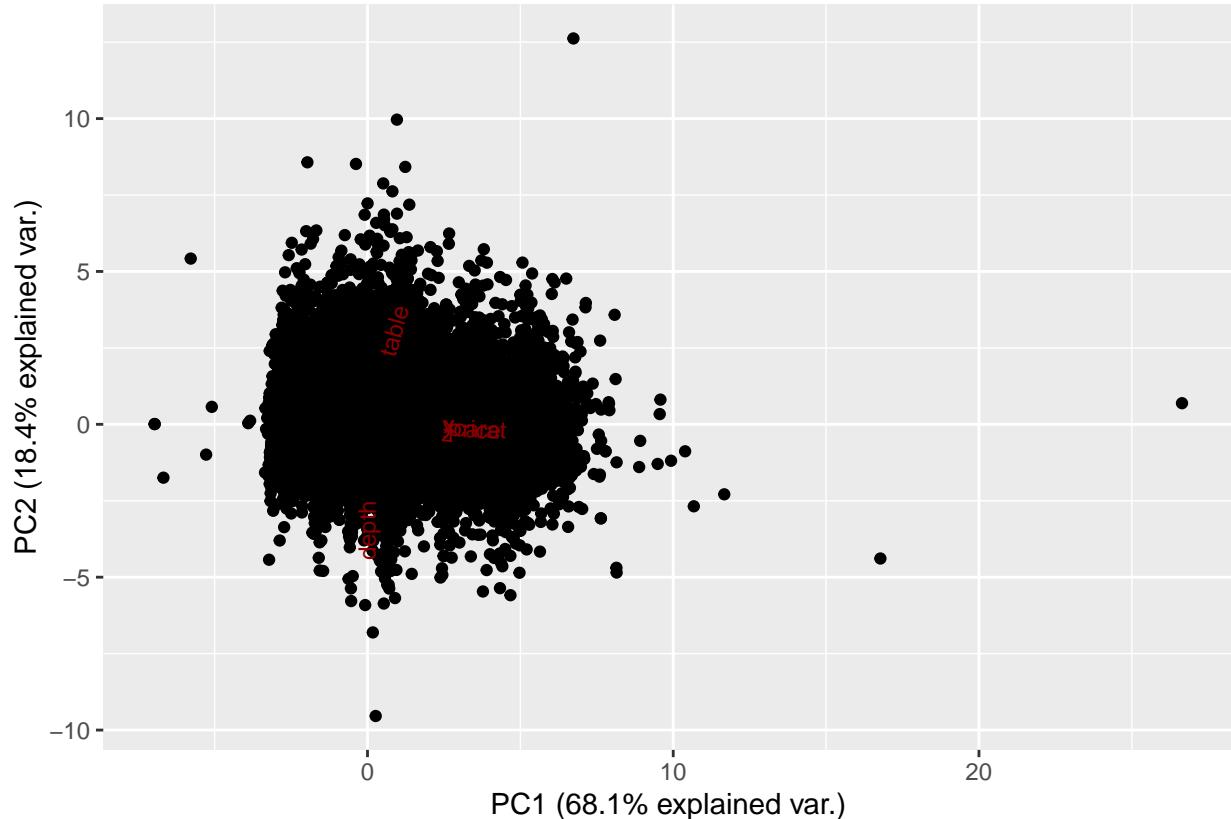
```

## y      0.4472649035  0.002157912  0.054188788  0.32846061 -0.77405793
## z      0.4459536619 -0.089035176 -0.039603439  0.31700727  0.60339656
##          PC6          PC7
## carat   0.76815114  0.425880295
## depth   0.01445027 -0.055600264
## table  -0.02526831 -0.002049255
## price  -0.27330947 -0.082814286
## x       0.19846061 -0.828658219
## y      -0.21526655  0.208857094
## z      -0.49867040  0.279957944

```

Looking at the eigenvectors we see that PC1 is primarily defined by carat, price, x, y and z. PC1 seems to be defined by dimension variables and, as we saw in the correlation matrix, price is closely associated with these variables. The variables with the most weight in PC2 are depth and table. We see a similar pattern in PC3 with depth and table having the largest weighting. This makes sense as in our examination of correlations between variables we saw that price, x, y, z and carat were all strongly correlated. The opposite loadings of depth and table in PC2 and PC3 reflect their negative correlation. A visualization of the PCA is shown in the Biplot below.

```
ggbiplot(pca.diamonds, obs.scale = 1, var.scale = 1)
```



Due to the vast number of data entries it is hard to interpret what is going on from the visual

display. We will not take a smaller sample of 1000 from the diamonds data and redo the PCA after first checking that the smaller sample has similar properties to the full sample. Also of note in this biplot are two significant outliers to the far right, which are extreme values of PC1. We shall identify those before creating the smaller sample.

## 5.1 Identify the extreme PC1 values

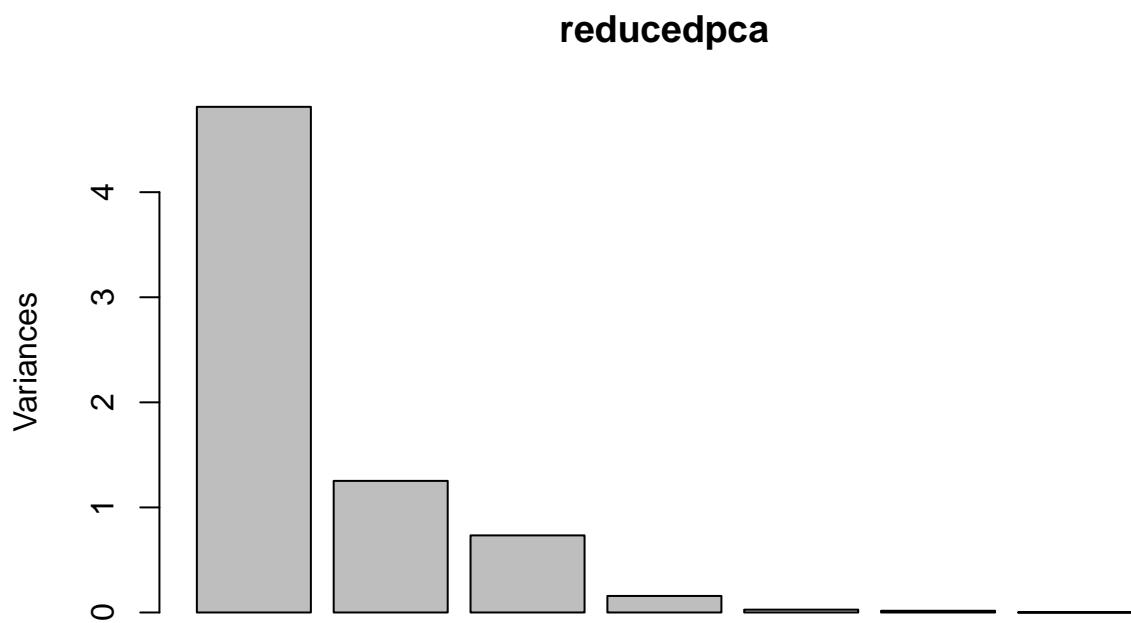
```
sort(decreasing=T,(pca.diamonds$x[,1]))[1:10]

##      24068      48411      27416      27631      49190      27131      25999      26000
## 26.641670 16.776354 11.669245 10.673276 10.389346 9.928569 9.581991 9.558664
##      26445      27680
## 9.491827 8.919222
```

## 5.2 PCA with Smaller Sample

```
set.seed(300525287, kind = "Mersenne-Twister")
smallersample <- diamonds[sample(nrow(diamonds), "1000"), ]
reducedpca <- prcomp(smaller.sample[,-c(2:4,11)], center = TRUE, scale = TRUE)

plot(reducedpca)
```



```

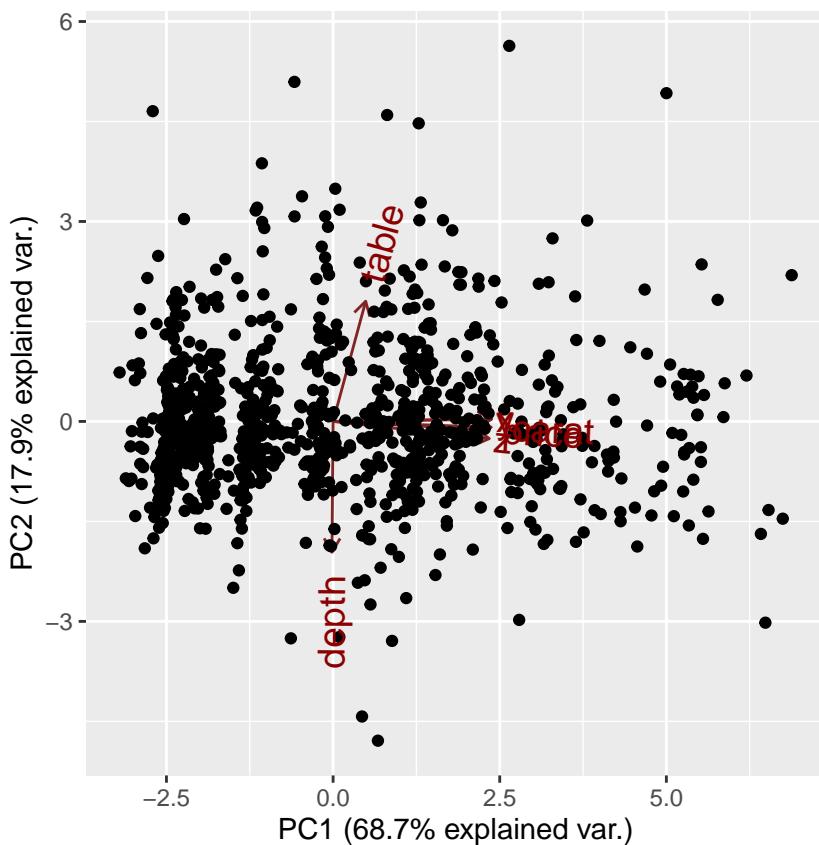
summary(reducedpca)

## Importance of components:
##          PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation   2.1937  1.1191  0.8565  0.39676 0.16404 0.12767 0.03195
## Proportion of Variance 0.6875  0.1789  0.1048  0.02249 0.00384 0.00233 0.00015
## Cumulative Proportion 0.6875  0.8664  0.9712  0.99368 0.99753 0.99985 1.00000

```

We see that the PCA with a sample size of 1000 has almost identical properties to the PCA of the whole dataset.

```
ggbiplot(reducedpca, obs.scale = 1, var.scale = 1, varname.size = 5)
```



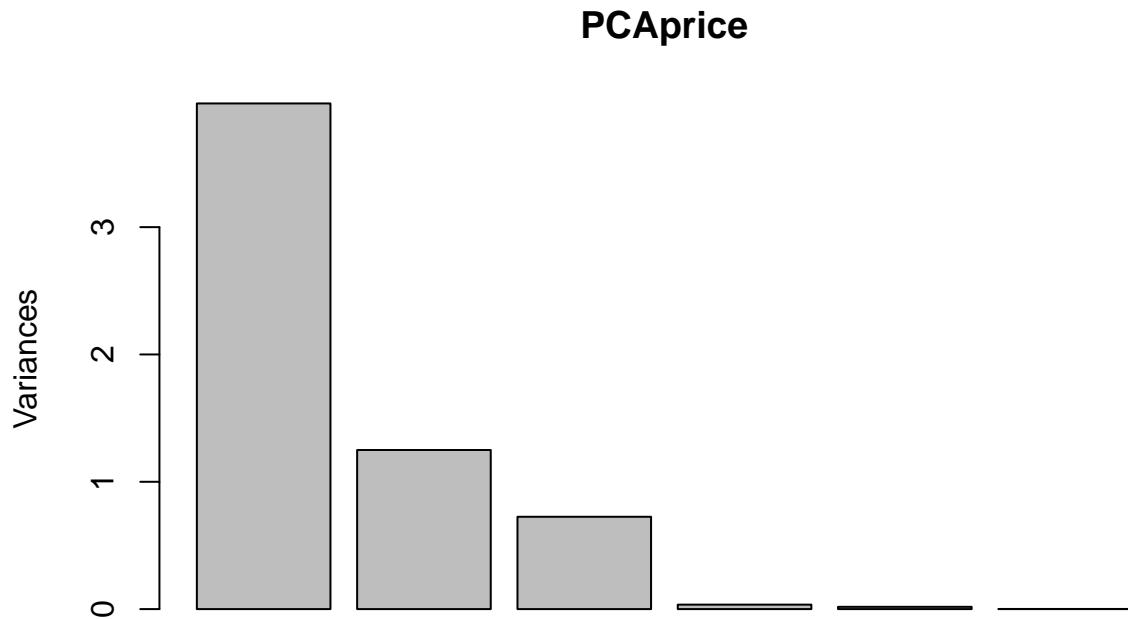
We see that carat, x, y, z and price are strongly influence the first PC. While depth and table strongly (and reasonably equally) influence the second PC. We see evidence of the negative correlation between table and depth and a close to zero correlation between these two variables and the variables that strongly influence PC1. We also have an indication that there is redundancy between carat, x, y, z and price. The percentage of variation retained by each Principal Component is included in the axis labels.

In this investigation we are interested in predicting the price of diamonds. To do this we will perform a regression with principal components with price as the dependent variable and the other numerical variables as explanatory variables.

### 5.3 Principal Components Regression

We will now perform a principal regression analysis with price as the response variable and all other numerical variables as the explanatory variables. We Will again use the smaller sample to increase the ease of interpretation of the visual plots.

```
PCAprice <- prcomp(smallerSample[, -c(2:4, 7, 11)], center = TRUE, scale = TRUE)
plot(PCAprice)
```



```
summary(PCAprice)
```

```
## Importance of components:
##                 PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation   1.9929  1.1178  0.8515  0.18700 0.13308 0.03283
## Proportion of Variance 0.6619  0.2083  0.1208  0.00583 0.00295 0.00018
## Cumulative Proportion 0.6619  0.8702  0.9910  0.99687 0.99982 1.00000
```

The plot above shows the proportion of variation explained by each of the principal components. The first three principal components explain nearly all (99%) the variation in the data.

```
PCAprice$rotation
```

```
##                 PC1          PC2          PC3          PC4          PC5
## carat    0.494554831 -0.0461150758  0.04445202 -0.806136953 -0.318086228
```

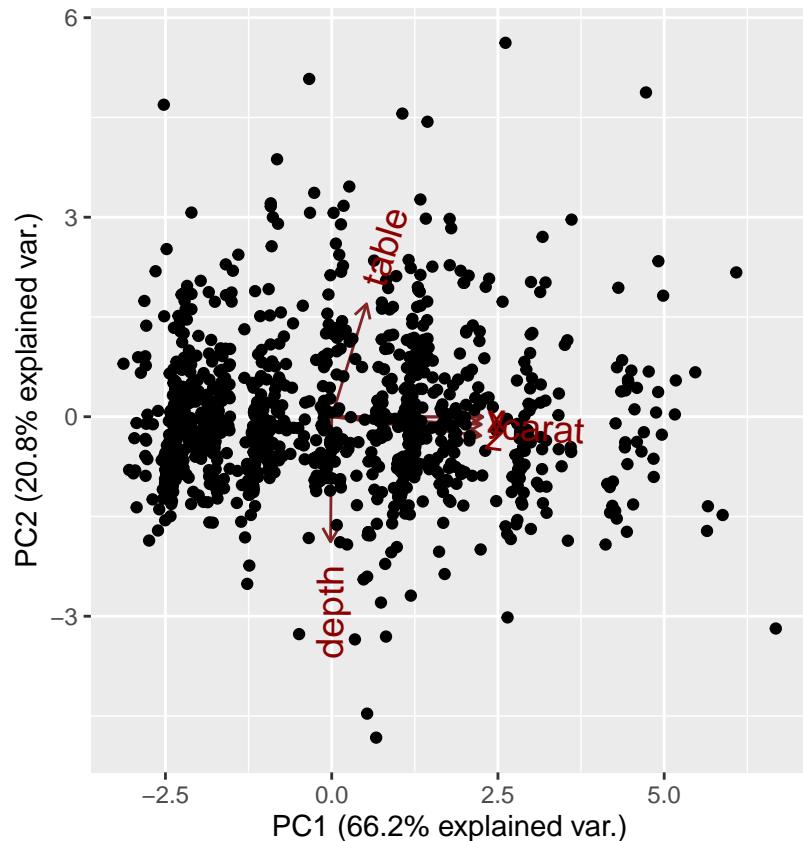
```

## depth -0.003656238 -0.7370594608 -0.66518876 -0.040970793 0.112128623
## table  0.115148753  0.6649799315 -0.73788433 -0.006134332 -0.004793799
## x      0.499625318 -0.0008101947  0.07042242  0.107199954  0.477863033
## y      0.499442121 -0.0022545940  0.07443970  0.134779378  0.482475245
## z      0.493026395 -0.1114123072 -0.02395964  0.564596906 -0.651989189
##          PC6
## carat   0.016652366
## depth   0.001569924
## table   0.002926009
## x      -0.711034219
## y      0.702896417
## z      -0.008867829

```

We see that in the first PC carat, x, y and z all have approximately equal weighting. Again this reflects the strong positive correlation between these two variables. In the second PC depth and table have the strongest weighting. We see that removing price from the PCA does not change much in terms of how much variation the first few components explain as well as which variables contribute most strongly to each component. This is because of how strongly price is correlated with the dimension variables

```
ggbiplot(PCAprice, obs.scale = 1, var.scale = 1, varname.size = 5)
```



Even with price not included we still see the same general structure as the PCA with price.

```

PCApricevars <- prcomp(smallerSample[, -c(2:4, 7, 11)], center = TRUE, scale = TRUE)$x
diamondPricePCA <- lm(smallerSample$price ~ PCApricevars[, 1] + PCApricevars[, 2] + PCApricevars[, 3] + PCApricevars[, 4] + PCApricevars[, 5] + PCApricevars[, 6])
summary(diamondPricePCA)

##
## Call:
## lm(formula = smallerSample$price ~ PCApricevars[, 1] + PCApricevars[, 2] + PCApricevars[, 3] + PCApricevars[, 4] + PCApricevars[, 5] + PCApricevars[, 6])
##
## Residuals:
##      Min        1Q    Median        3Q       Max 
## -14748.2   -604.6   -109.9    401.3    7965.2 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 3967.32    46.55   85.229 < 2e-16 ***
## PCApricevars[, 1] 1835.78    23.37   78.555 < 2e-16 ***
## PCApricevars[, 2] -208.03    41.66   -4.993 7.01e-07 ***
## PCApricevars[, 3]  444.25    54.69    8.122 1.35e-15 ***
## PCApricevars[, 4] -4228.97   249.04  -16.981 < 2e-16 ***
## PCApricevars[, 5] -2619.38   349.95  -7.485 1.58e-13 ***
## PCApricevars[, 6] 10444.39   1418.67   7.362 3.79e-13 ***

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1472 on 993 degrees of freedom
## Multiple R-squared:  0.8703, Adjusted R-squared:  0.8695 
## F-statistic:  1110 on 6 and 993 DF,  p-value: < 2.2e-16

```

We see that every single principal component is found to be significant in this model when the effects of the other principal components is taken into account. Therefore initially it seems we have failed to create a more parsimonious model for predicting price through using regression with principal components. However if we refit the model with just the first two principal components (see below) we still create a model that is able to explain over 81% of the variance in price with only two components. While this is not as good as the model with all the principal components (which explains 85%) it is still reasonably good and is much more simple. The original multiple regression model explained 91% of the variation in price with 9 explanatory variables while our PCR model explains 81% with just the first two principal components.

```

diamondPricePCA2 <- lm(smallerSample$price ~ PCApricevars[, 1] + PCApricevars[, 2])
summary(diamondPricePCA2)

```

```

##
## Call:
## lm(formula = smallerSample$price ~ PCApricevars[, 1] + PCApricevars[, 2])
## 
```

```

## lm(formula = smallersample$price ~ PCApricevars[, 1] + PCApricevars[, 2])
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -8846.0 -1162.0 -139.3  877.8 8890.8
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3967.32    56.28   70.49 < 2e-16 ***
## PCApricevars[, 1] 1835.78    28.26   64.97 < 2e-16 ***
## PCApricevars[, 2] -208.03    50.37  -4.13 3.94e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1780 on 997 degrees of freedom
## Multiple R-squared:  0.8096, Adjusted R-squared:  0.8092
## F-statistic: 2119 on 2 and 997 DF,  p-value: < 2.2e-16

```

It is likely that the remaining, unexplained variation could be explained by the categorical variables in the dataset. As Principal Components Analysis/Regression can only be used for numerical variables we had to leave out the variables of cut, clarity and color.

## 5.4 Factor Analysis

In our investigation of the principal components of the dataset we observed that the first two principal components explained the vast majority of the variation in the data. The first was dominated by the strong positive correlation between price and the dimension variables (x, y, z, carat) which we may call price + dimension. The second was the negative correlation between depth and table which we might combine into light performance as both variables are crucial in giving a diamond its “sparkle”. We hypothesise that these two factors (1. “Price + Dimension” 2. “Light Performance”) may explain the observations in the diamonds dataset. To investigate this we will conduct an exploratory factor analysis with two factors.

```

factanal(diamonds[,-c(2:4,11)], factors =2)

##
## Call:
## factanal(x = diamonds[, -c(2:4, 11)], factors = 2)
##
## Uniquenesses:
## carat depth table price      x      y      z
## 0.041 0.005 0.877 0.205 0.006 0.045 0.039
##
## Loadings:
##       Factor1 Factor2

```

```

## carat  0.976
## depth   0.114   0.991
## table   0.152  -0.316
## price   0.885  -0.113
## x       0.987  -0.140
## y       0.967  -0.141
## z       0.980
##
##          Factor1 Factor2
## SS loadings    4.641   1.141
## Proportion Var  0.663   0.163
## Cumulative Var  0.663   0.826
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 22188.2 on 8 degrees of freedom.
## The p-value is 0

```

As we hypothesized Carat, price, x, y and z are well explained by factor 1. Depth is very well explained by factor 2 but table is to a lesser degree. We see some evidence of the factor structure in our hypothesis with the “price + dimension” variables strongly implicated in Factor 1 and “Light conductance” variables most strongly explained by factor 2. However, we also see other original variables we did not expect in each of our factors. Factor 1 explains 66% of the variance in the data and Factor 2 explains an additional 16%. We also see high uniqueness ratings for table indicating that the two factors do not well account for it’s variance.

Overall the hypothesis test gives a highly significant result meaning that two factors is not sufficient to capture the full dimensionality of the data set.

## 6 Linear Discriminant Analysis (LDA)

As discussed previously the diamonds dataset has three categorical variables each with many levels. This means we have the possibility of 280 unique interactions between the levels of these categorical variables. We will attempt too use Linear Discriminant Analysis to classify observations more simply using one of the categorical variables at a time.

```

set.seed(300525287, kind = "Mersenne-Twister")
ind <- sample(c("Train", "Test"),
               nrow(smaller.sample),
               replace = TRUE,
               prob = c(.6, .4))
diamonds.Train <- smaller.sample[ind == "Train",]
diamonds.Test <- smaller.sample[ind == "Test",]
LDAct <- lda(cut ~ carat + depth + table + price
             + x + y + z, data = diamonds.Train)

```

```

LDAcolor <- lda(color ~ carat + depth + table + price
+ x + y + z, data = diamonds.Train)
LDAclarity <- lda(clarity ~ carat + depth + table + price
+ x + y + z, data = diamonds.Train)

```

We see that mean carat and mean price seem to differ the most across levels of “cut”. In LD1 x and y weight significantly more than the other variables.

```

library(ggord)

g1 <- ggord(LDAcut, diamonds.Train$cut, alpha = 0.5, )
g2 <- ggord(LDAcolor, diamonds.Train$color, alpha = 0.5 )
g3 <- ggord(LDAclarity, diamonds.Train$clarity, alpha = 0.5)
ggarrange(g1, g2, g3, ncol = 2, nrow =2)

```

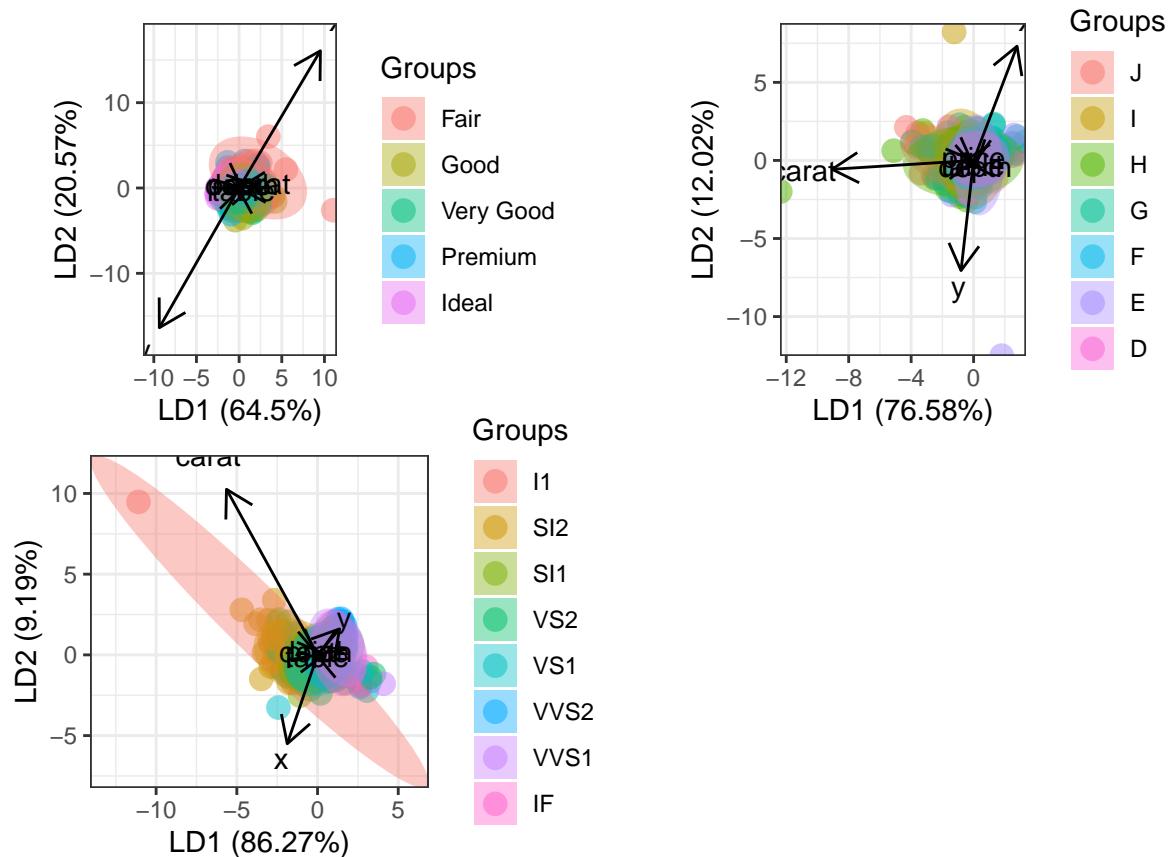


Figure 7: Ordination plot

All three ordination plots (figure 7) show a lot of overlap between different levels of the categorical variables. It seems unlikely that a linear discriminant function would have much success in partitioning the data according to these classes. We shall compute the confusion matrix for each LDA and assess the overall accuracy of each model.

```

RPCut <- predict(LDAcut, diamonds.Test)$class
RCMcut <- table(RPCut, actual = diamonds.Test$cut)
RPCcolor <- predict(LDAcolor, diamonds.Test)$class
RCMcolor <- table(RPCut, actual = diamonds.Test$color)
RPclarity <- predict(LDAclarity, diamonds.Test)$class
RCMclarity <- table(RPCut, actual = diamonds.Test$clarity)

Cutacc <- sum(diag(RCMcut))/sum(RCMcut)
Coloracc <- sum(diag(RCMcolor))/sum(RCMcolor)
Clarityacc <- sum(diag(RCMclarity))/sum(RCMclarity)

```

**Table: Classification Accuracy for Different LDA's**

Class	Accuracy
Cut	0.6382
Color	0.196
Clarity	0.1457

In the table above we see the only categorical variables the LDA has any success of classifying is "cut". For color and clarity the LDA performs very poorly indeed. We conclude that it is likely that the classes (particularly for clarity and color) are not linearly separable.

## 6.1 Why prediction without the categorical variables is not working well

### 6.1.1 Create modified version of dataset only containing carat values between 1 and 1.1

```

library(dplyr)
diamonds_car2 <- diamonds[,c(1,4,7)]
diamonds_car2[1:4,]

##   carat clarity price
## 1  0.23     SI2   326
## 2  0.21     SI1   326
## 3  0.23     VS1   327
## 4  0.29     VS2   334

diamonds_car19_21 <- filter(diamonds_car2, between(carat, 1, 1.1))
diamonds_car19_21[1:4,]

##   carat clarity price
## 1  1.01      I1   2781
## 2  1.01      I1   2788
## 3  1.01     SI2   2788
## 4  1.05     SI2   2789

```

```
# scatterplot of carat vs price, but also colouring the observations
# by 'cut'
ggplot(data=diamonds_car19_21, aes(x=carat, y=price, color=clarity))+ 
  geom_point()+
  guides(colour=guide_legend(reverse = T))+
  labs(title = "'Carat' vs 'price', coloured by 'clarity'")
```

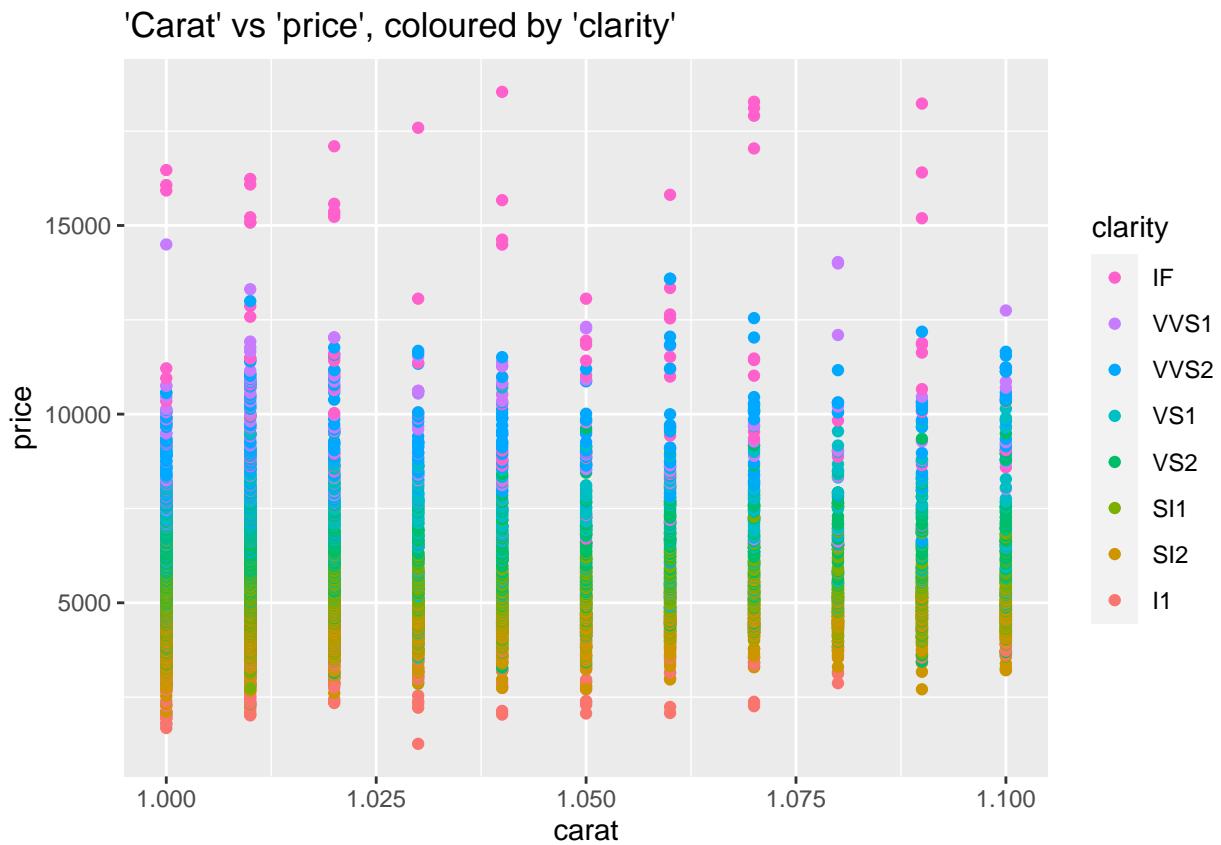


Figure 8: Carat (limited values) vs Price coloured by Clarity

Figure 8 shows a the scatterplot of ‘price’ vs ‘carat’ vs ‘clarity’, but just for values of ‘carat’ between 1 and 1.1, which helps us see the pattern more clearly. Firstly, note the vertical spread of the data points; most of the values begin at around 2500 USD and generally have extreme upper values larger than 15,000 USD, so a spread of approximately 10,000 USD. There are clear colour bands running horizontally, with the lowest quality ‘clarity’ diamonds near the bottom of the price range, while the highest values are near the top.

So, using ‘carat’ alone to predict these prices clearly is not enough; given the spread of price for any given value of carat, the categorical variables play an important in determining where in the range any given diamond will be. This was the reason our regression models that excluded the categorical variables did not perform nearly as well as those that included them.

## 6.2 Discriminant analysis using the reduced dataset (price, carat and clarity)

```
# split dataset into 'train' and 'test'  
set.seed(1234567890, kind = "Mersenne-Twister")  
ind <- sample(c("Tr", "Te"),  
              nrow(diamonds_car19_21), replace = TRUE, prob = c(0.6, 0.4))  
  
# display first few elements to check  
ind[1:15]  
  
## [1] "Te" "Tr" "Te" "Te" "Te" "Te" "Tr" "Te" "Te" "Te"  
## [11] "Te" "Tr" "Tr" "Te" "Te" "Te" "Te" "Tr" "Tr" "Te"  
## [21] "Te" "Te" "Te"  
  
# check that the proportions have come out as expected  
nrow(diamonds_car19_21)  
  
## [1] 7568  
  
table(ind)  
  
## ind  
##   Te      Tr  
## 3008 4560  
  
prop.table(table(ind))  
  
## ind  
##       Te      Tr  
## 0.397463 0.602537
```

The output above confirms that the random allocation of values to either the training or testing datasets produced the desired result of approximately 60% in the training set and 40% in the testing set.

### 6.2.1 Create training and test subsets

```
Train <- diamonds_car19_21[ind=="Tr",]  
Test <- diamonds_car19_21[ind=="Te",]
```

### 6.2.2 Create LDA and display summary

```
library(MASS)  
LDA.diam <- lda(clarity~carat+price,  
                  data=Train)
```

### 6.2.3 Boxplots of the LDA means

```
boxplot(t(LDA.diam$means), main=
  "Distribution of LDA means for each level of 'clarity'",
  xlab="Levels of 'clarity' (lowest on the left, highest on the right)",
  ylab="Count")
```

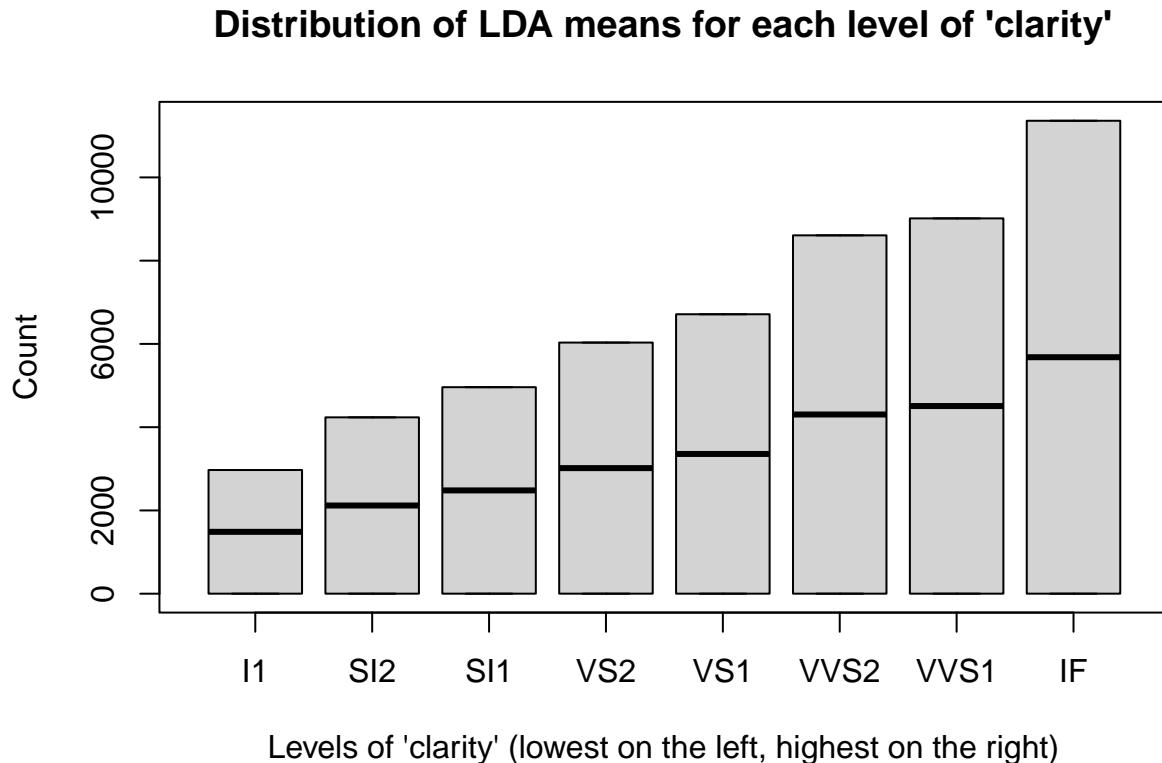


Figure 9: Distribution of LDA means

Figure 9 shows the distribution of the LDA means. It is obvious that there is an increasing number of diamonds of higher ‘clarity’ levels.

### 6.2.4 Create Prediction object

```
Pred <- predict(LDA.diam)
```

### 6.2.5 Biplot

```
library(klaR)
library(ggord)
```

```
library(devtools)
ggord(LDA.diam,Train$clarity)
```

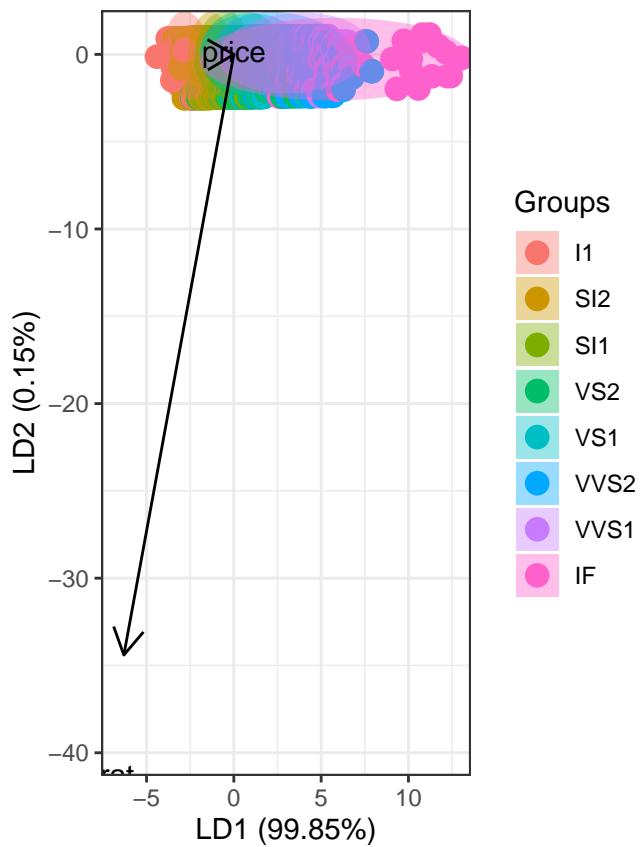


Figure 10: Biplot of ‘carat’, ‘price’ against ‘clarity’

Figure 10 shows the biplot for ‘carat’ versus ‘price’ against the different levels of ‘clarity’.

#### 6.2.6 Partition plot

```
partimat(clarity~carat+price, data = Train, method="lda")
```

## Partition Plot

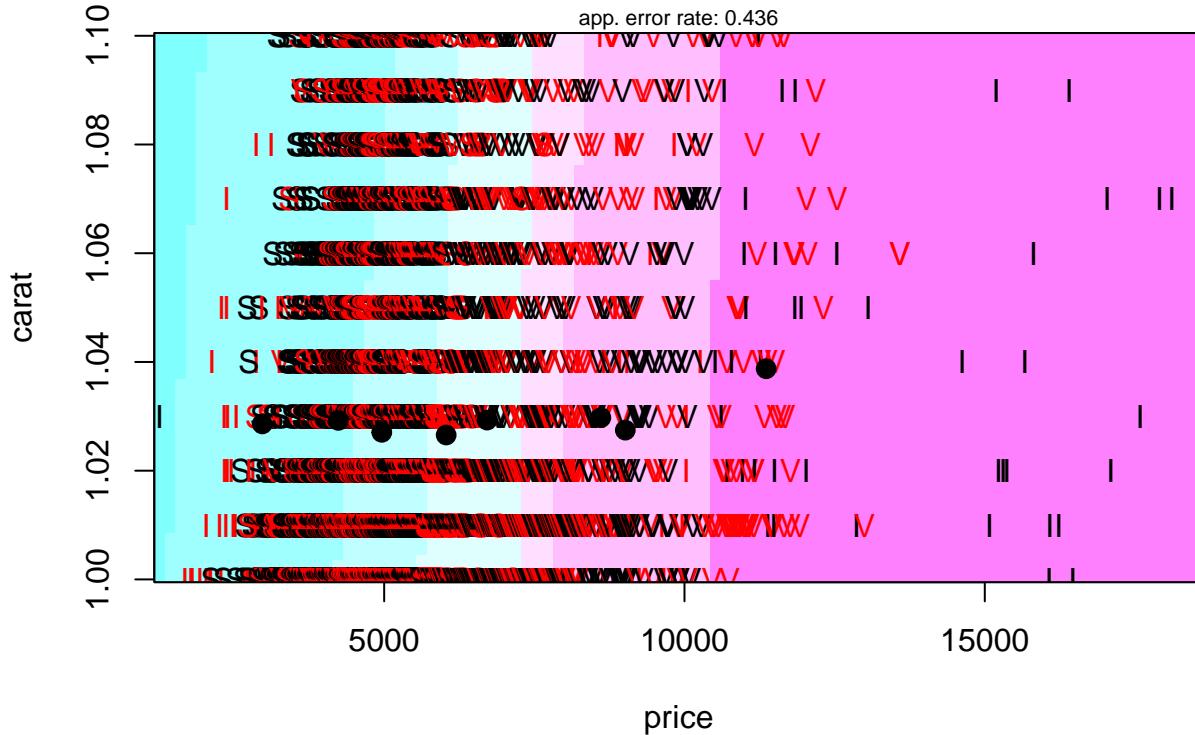


Figure 11: Partition plot for levels of ‘clarity’ as predicted by ‘carat’ and ‘price’

Figure 11 shows the partition plot for the discriminant analysis (DA) of the different levels of the ‘clarity’ variable as predicted by ‘carat’ and ‘price’. As a reminder, the levels are (from lowest to highest): Fair, Good, Very Good, Premium and Ideal.

```
RealisticPredicted <- predict(LDA.diam, Test)$class
RCM <- table(RealisticPredicted, Actual=Test$clarity)
RCM
```

	Actual							
##	I1	SI2	SI1	VS2	VS1	VVS2	VVS1	IF
##	I1	0	0	0	0	0	0	0
##	SI2	65	545	185	40	24	4	0
##	SI1	0	260	578	158	57	5	4
##	VS2	0	3	76	366	155	30	3
##	VS1	0	0	1	45	81	28	9
##	VVS2	0	1	0	12	43	118	31
##	VVS1	0	0	0	0	0	0	0
##	IF	0	0	0	0	1	14	17

The confusion matrix above shows that there were many mis-categorisations. There is a

pattern evident. The leading diagonal shows the correct predictions. There are many incorrect predictions for values immediately above and below the leading diagonal, meaning that the DA partitions struggled to separate adjacent categories.

```
prop_succ <- sum(diag(RCM))/sum(RCM)
prop_succ
```

```
## [1] 0.5698138
```

```
per_succ <- signif(prop_succ*100,4)
per_succ
```

```
## [1] 56.98
```

The output above shows that the percentage of successful predictions was 56.98%, indicating that this LDA is a mediocre predictor.

### 6.2.7 Summary of LDA

The relatively poor performance of LDA in this instance is perhaps not surprising given the amount of overlap we saw in figure 8 earlier (the scatterplot of ‘price’ vs ‘carat’ vs ‘clarity’ for ‘carat’ values between 1 and 1.1); there simply isn’t enough geometrical separation between the different levels of ‘clarity’ for the algorithm to accurately categorise the boundary cases.

## 7 Cluster Analysis

As LDA did not yield useful results in terms of classifying the diamonds data we also decided to conduct Cluster Analysis to see if the diamonds data set could be divided into groups. We used a K-means algorithm to achieve this. As we were initially unsure of how many clusters we should attempt to divide the data into we will first examine the sum of the squares of the intra-class distances for different numbers of clusters.

### 7.0.1 Optimal cluster number

```
fviz_nbclust((scale(smallerSample[,-c((2:4),11)])), kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2) +
  labs(subtitle = "Elbow method")
```

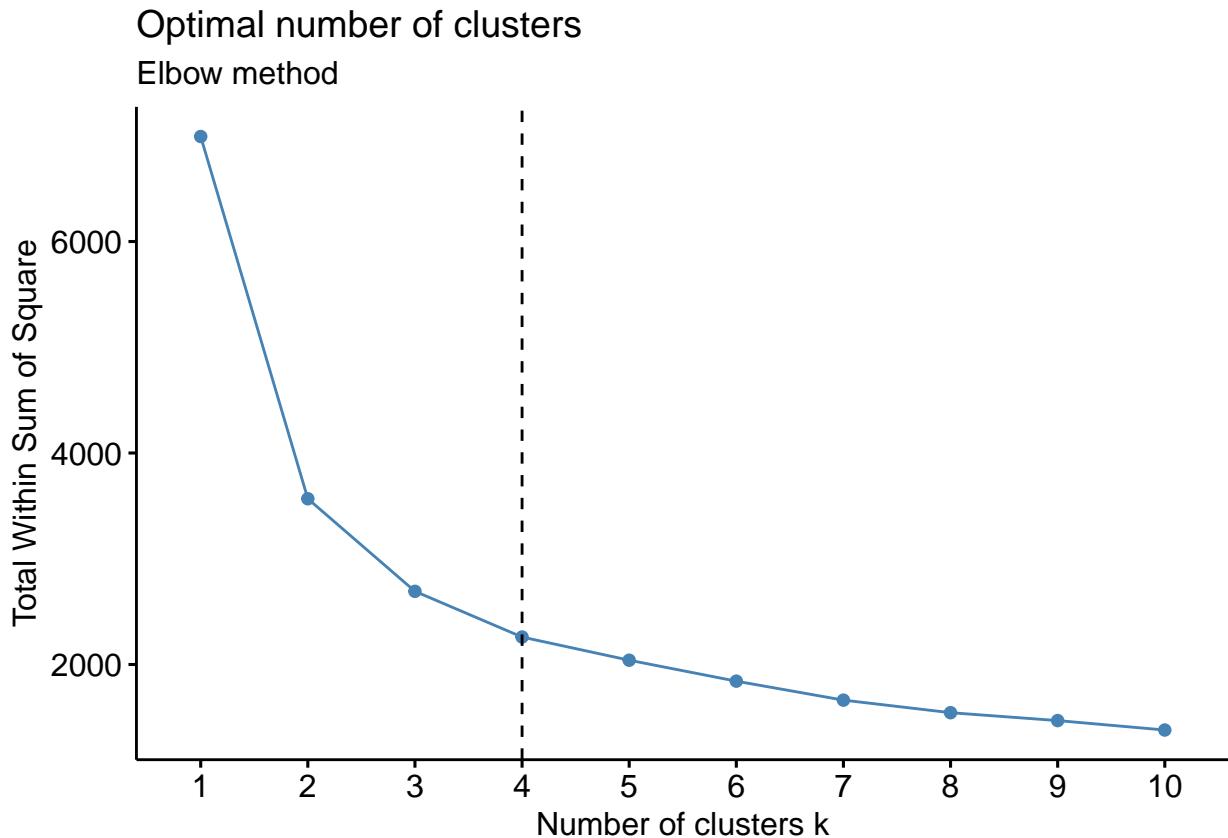


Figure 12: Optimal number of clusters

Figure 12 shows that when the elbow method is used, we see that 4 clusters is the optimum number.

### 7.0.2 kmeans function

```
clusters <- kmeans(scale(smaller.sample[,-c((2:4),11)]),
                      algorithm = "MacQueen", centers = 4,
                      iter.max = 100, nstart = 50)

# print clusters means
clusters$centers

##          carat      depth      table      price         x         y
## 1  0.3436636  0.4625156 -0.19171931  0.17367610  0.4846585  0.4876979
## 2 -0.8302592  0.0195466 -0.31324680 -0.71288391 -0.9107135 -0.9109156
## 3  1.9509740  0.1578009  0.05304506  2.09674493  1.7073677  1.7102105
## 4  0.2424997 -1.1838989  1.40775657  0.06611287  0.4472637  0.4388989
##          z
## 1  0.5477885
```

```

## 2 -0.9007307
## 3 1.7128313
## 4 0.2761387

# print first 20 values
clusters$cluster[1:20]

## 44678 2911 2255 27164 48502 47496 47160 11609 35207 33369 29772 33632 47294
##     4     4     1     3     2     2     1     1     2     2     2     2     2
## 39481 14569 25563 45445 23592 13383 1549
##     2     1     3     2     3     2     1

```

We create four clusters with sizes of 127, 455, 137 and 281 respectively. We see clear differences in the mean of each numeric variable across the four clusters. The within cluster sum of squares indicates that the clusters are not very compact. We shall now visualize the clusters with a scatterplot using the first two principal components of our dataset as the two dimensions. The first two principal components account for over 86% of the variance so this should be an appropriate visualization.

### 7.0.3 Cluster plot

```

fviz_cluster(clusters, data = (scale(smaller.sample[, -c((2:4), 11)])),
             geom = "point",
             ellipse.type = "norm",
             ggtheme = theme_bw()
)

```

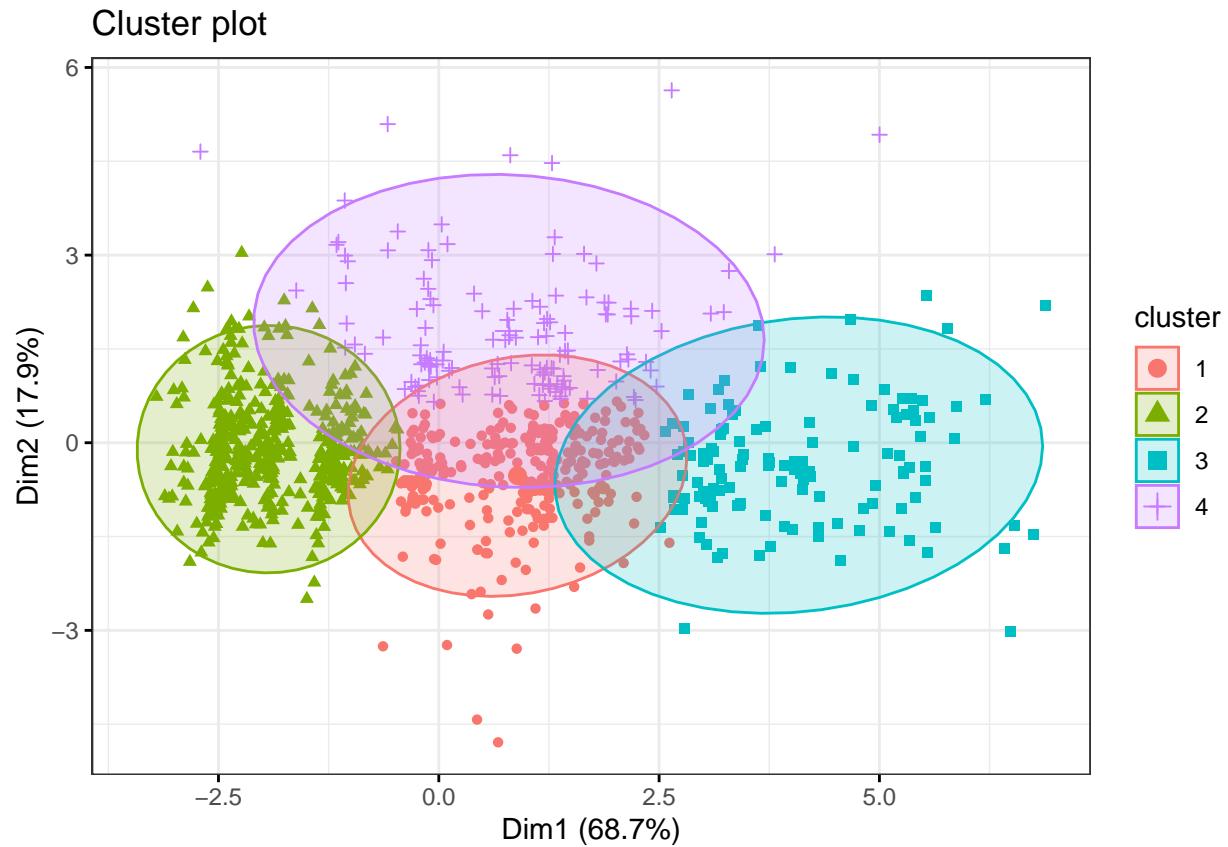


Figure 13: Cluster plot

In the cluster plot (figure 13) we see significant overlap between the four clusters as well as many observations that do not seem to fit well in any of the clusters. It seems that the diamond dataset does not easily split into distinct clusters.

#### 7.0.4 Silhouette plot

```
library(cluster)
sil <- silhouette(clusters$cluster, dist((scale(smaller.sample[,-c((2:4),11)]))))
fviz_silhouette(sil)

##   cluster size ave.sil.width
## 1       1  283      0.29
## 2       2  455      0.51
## 3       3  127      0.34
## 4       4  135      0.21
```

Clusters silhouette plot  
Average silhouette width: 0.38

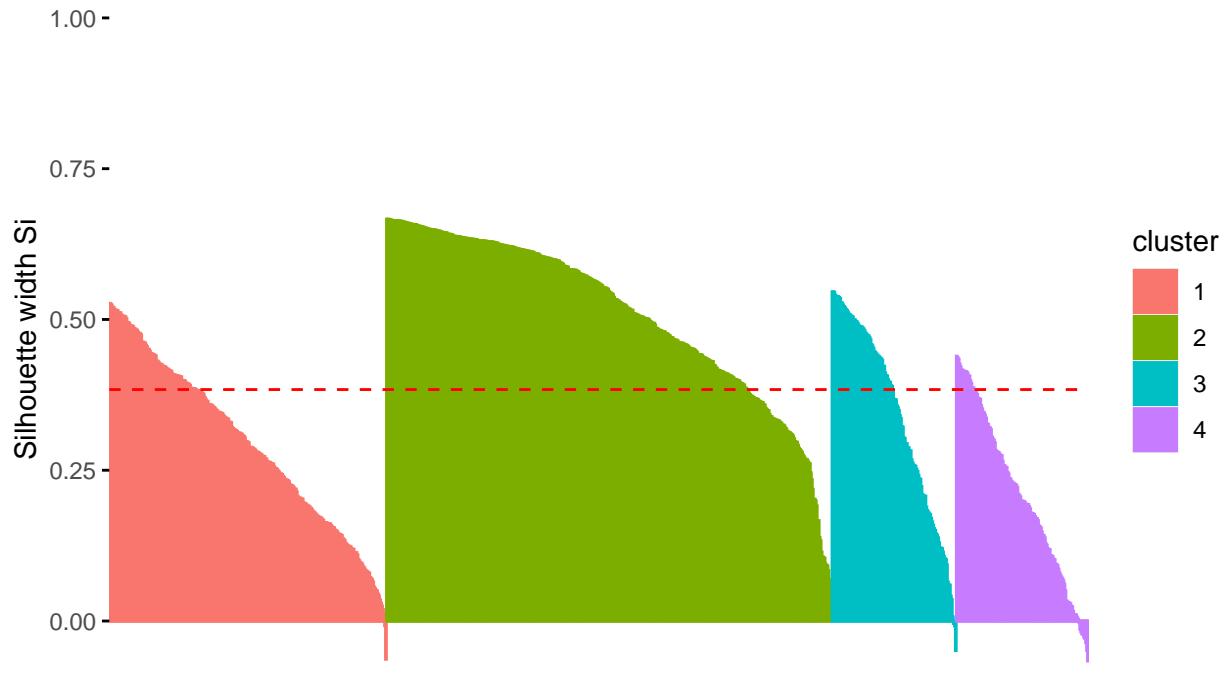


Figure 14: Clusters silhouette plot

The silhouette plot (figure 14) helps us to visualize how similar points each observation is to the cluster that it is assigned too. We see very few negative values indicating that very few observations have been assigned to the wrong cluster. However we do have a large proportion of values with low positive values indicating that these observations were close to the borderline between two clusters. We see that the average silhouette width is 0.38.

## 8 Conclusions

## 9 Appendices

- 9.1 Code for Producing KS Tests
- 9.2 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Cut
- 9.3 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Clarity
- 9.4 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Color

## Bibliography

- Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2020. *RMarkdown: Dynamic Documents for r*. <https://rmarkdown.rstudio.com>.
- “Diamonds Dataset, Kaggle.com.” 2016. <https://www.kaggle.com/datasets/shivam2503/diamonds>.
- “Github.com.” 2022. <https://github.com/arinicholas/STAT394-Group-Project>.
- R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.
- RStudio Team. 2022. *RStudio: Integrated Development Environment for r*. Boston, MA: RStudio, PBC. <http://www.rstudio.com/>.