STAT394 Group Project Final Report

Ken MacIver, Tom Tribe, Jundi Yang, Mei Huang

2022-09-27

Contents

1 DRAFT FINAL REPORT

2 DELETE FROM FINAL

2.1 Experimenting with the relative file pathways

Making sure the relative file pathways work according to Alejandro's file structure from his article Frery, Gomez, and Medeiros (2020). The bibliography file path in the YAML code above takes us down into the 'Bibliography' folder and then opens the 'MASTER.bib' file. That is working.

3 Introduction

Our multivariate statistical investigation uses the Diamonds data set (reference). This data set is a base data set in R and is also freely available on Kaggle (reference). It contains information on ten different variables for 53940 diamonds. Of these ten variables three are categorical while 7 are numerical.

3.1 Variables

Carat - Carat is a measure of the diamond's weight. One carat equals 1/5 gram. In this data set we have diamonds with a carat ranging from 0.2 - 5.01. Cut - A diamonds Cut defines its proportions and its ability to reflect light. This variable has five levels: Fair, Good, Very

Good, Premium Ideal. Color - A diamonds color refers to how clear/colorless a diamond is. This variable has seven levels: J (lowest quality), I, H, G, F, E, D (highest quality) Clarity - 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 - x is a diamonds length in mm. In this data set we have diamonds with a length ranging from 0 - 10.74 y - x is a diamonds width in mm. In this data set we have diamonds with a width ranging from 0 - 58.9
z - x is a diamonds depth in mm. In this data set we have diamonds with a depth ranging from 0 - 31.8 Depth - Depth measures a diamonds total depth percentage total depth percentage. This is calculate by dividing the total width by the total depth. Depth percentage impacts how light reflects of 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 in this data set measures table percentage which calculated by dividing table width by the total width. Table percentage impacts how light reflects of the diamond. In this data set we have diamonds with a table percentage ranging from 43% - 95% Price - The price of the diamond in US dollars. In this data set we have diamonds with a price ranging from $326 - $18823

We chose this data set as the variables were simple to understand conceptually and we hoped this would increase the ease to which we could make inferences about it compared to a data set on a subject we had little knowledge of. This data set also allows us to undertake an investigation from which we glean insights about diamonds that have real-world application and use.

Our study seeks to investigate which variables are most closely related to, and best predictors of, diamond price. We hypothesise that increased diamond size (x, y and z) and weight will be positively associated with diamond price. We also expect to see diamond price to be higher at the higher levels of the categorical variables compared to the lower levels. We are unsure how diamond depth and table percentage while relate to diamond price.

We also seek to classify.... (something about classes so we can use LDA/CA)

4 Methodology

The diamonds data set was accessed through Kaggle while all statistical analysis and reporting was completed in R/R Markdown. Github was used a repository for storing all relevant documents and code during this investigation.

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 of 123456789 and the pseudo random number generator Mersenne Twister.

To begin our investigation we first completed an in-depth Exploratory Data Analysis of the data set the consolidated results of which you can find in the results section. We used this to develop our leading questions that we attempt to answer using the multivariate statistical techniques of Principal Components Analysis, Factor Analysis, Linear Discriminant Analysis and Cluster Analysis.

5 Results

5.1 Consolidated EDA

We will begin by presenting results from our consolidated Exploratory Data Analysis that our relevant to our investigation.

5.1.1 Summaries of Data

Table 1: Summary statistics for 'diamonds' (3 s.f.)

	carat	depth	table	price	x	y	z
sample size	53940.000	53940.000	53940.000	53940.000	53940.000	53940.000	53940.000
minimum	0.200	43.000	43.000	326.000	0.000	0.000	0.000
first quartile	0.400	61.000	56.000	950.000	4.710	4.720	2.910
median	0.700	61.800	57.000	2401.000	5.700	5.710	3.530
mean	0.798	61.749	57.457	3932.800	5.731	5.735	3.539
third quartile	1.040	62.500	59.000	5324.250	6.540	6.540	4.040
maximum	5.010	79.000	95.000	18823.000	10.740	58.900	31.800
IQR	0.640	1.500	3.000	4374.250	1.830	1.820	1.130
standard deviation	0.474	1.433	2.234	3989.440	1.122	1.142	0.706
skewness	1.117	-0.082	0.797	1.618	0.379	2.434	1.522
kurtosis	4.256	8.739	5.801	5.177	2.382	94.206	50.082

A brief look at the summary statistics of the numerical variables shows that they all exist on very 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 as that is the minimum value for those variables. This will be further investigated in the outliers and unusual points section.

Categorical Summaries

Table: cut count

Cut	Count
Fair	1610
Good	4960
Ideal	21551
Premium	13791
Very Good	12082

The table above gives a breakdown of the how many diamonds are in each level of the 'cut' variable. We can see that most are in the 'Ideal', with a substantial number also in the 'Premium' and 'Very Good'.

5.1.2 Table of 'color' count

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

The table above shows the number of diamonds in each level of the 'color' variable.

Table: clarity count

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

The table above shows the counts for the different levels of the 'clarity' variable.

5.1.3 Distribution of Numeric Variables

While examining histograms, Cullen and Frey Plots, skewness and kurtosis of the numerical variables in our EDA we had reason to believe 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))  ## one row, two columns
qqnorm(diamonds$carat, xlab = "Observations", ylab = "Normal Quantiles", main = "Carat",
qqline(diamonds$carat, col = "blue", lwd =2)
qqnorm(diamonds$depth, xlab = "Observations", ylab = "Normal Quantiles", main = "Depth",
qqline(diamonds$depth, col = "blue", lwd =2)
qqnorm(diamonds$table, xlab = "Observations", ylab = "Normal Quantiles", main = "Table",
qqline(diamonds$table, col = "blue", lwd =2)
qqnorm(diamonds$price, xlab = "Observations", ylab = "Normal Quantiles", main = "Price",
qqline(diamonds$price, col = "blue", lwd =2)
qqnorm(diamonds$x, xlab = "Observations", ylab = "Normal Quantiles", main = "x", col = "
qqline(diamonds$x, col = "blue", lwd =2)
qqnorm(diamonds$y, xlab = "Observations", ylab = "Normal Quantiles", main = "y", col = "
qqline(diamonds$y, col = "blue", lwd =2)
qqnorm(diamonds$depth, xlab = "Observations", ylab = "Normal Quantiles", main = "z", col
qqline(diamonds$depth, col = "blue", lwd =2)
```

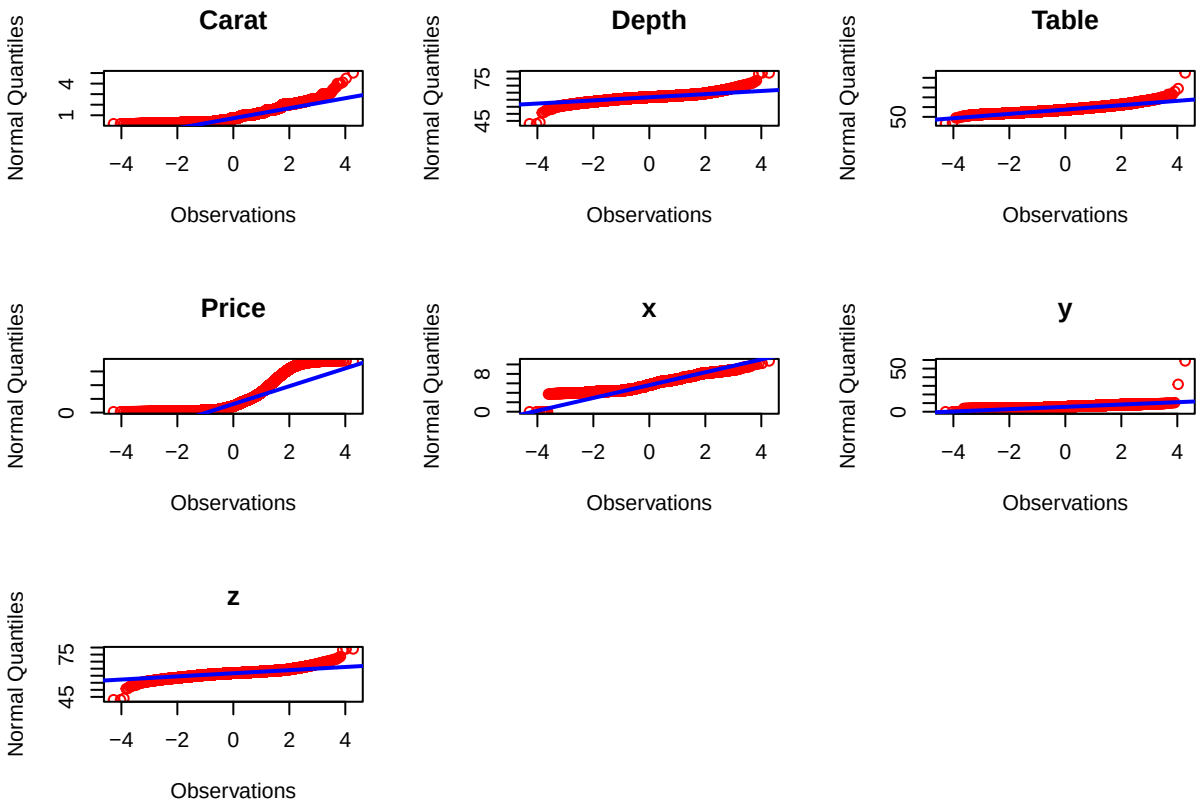

Figure 1: (#fig:normal QQ plots)Normal QQ Plots of Numeric Variables

Despite the small size of the Normal QQ plots shown above it is clear they indicate 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: Kolmogorov-Smirnov GOF Results

Variable	Test Statistic (D)	p-value
Carat	0.12274	$<$ 2.2e-16
Depth	0.075871	$<$ 2.2e-16
Table	0.13225	$<$ 2.2e-16
Price	0.18467	$<$ 2.2e-16
x	0.093545	$<$ 2.2e-16
y	0.088528	$<$ 2.2e-16
z	0.089273	$<$ 2.2e-16

The table above shows that for each numeric variable the KS goodness of fit test gave evidence that it did not follow a normal distribution

5.1.4 Correlation Matrix

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

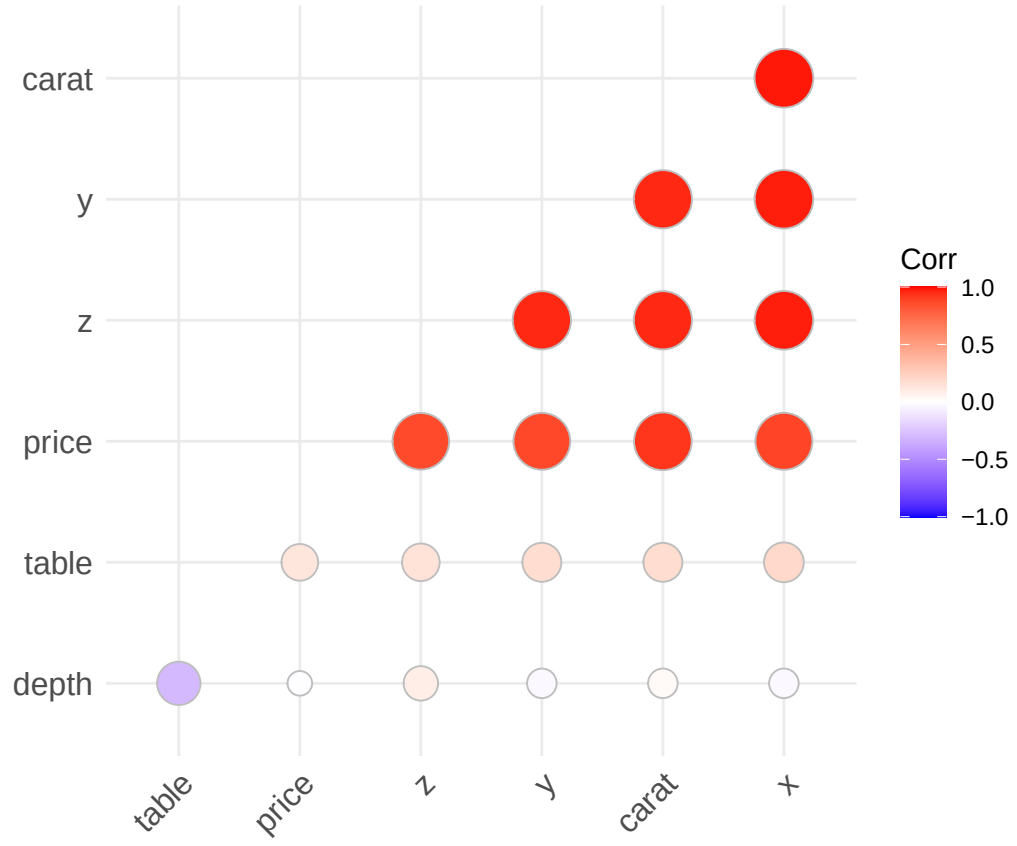

Figure 2: A Visualization of the Correlation Matrix

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.

5.1.5 Price differences across Categorical Variables

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

5.1.5.1 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.

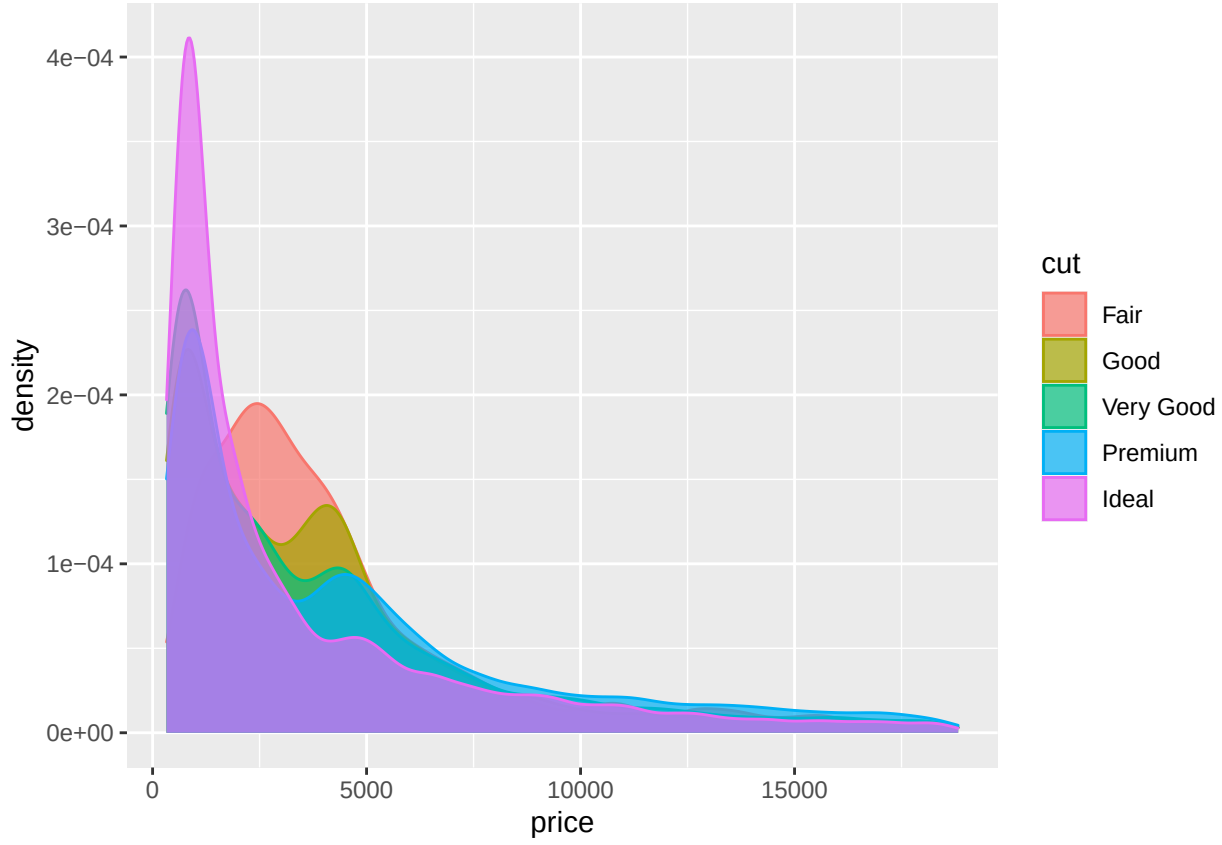

Figure **??**shows the density plots for the different levels of the 'price' variable. The legend on the right shows the level names and their order ('fair' = poorest, 'ideal' = best). Of note is the fact that the two best leves (premium and ideal) have significant peaks near the lower end of the price 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 verfiy 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 theses tests may be found in the appendices.

Table: ANOVA of Price by Cut

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

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

5.1.5.2 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.

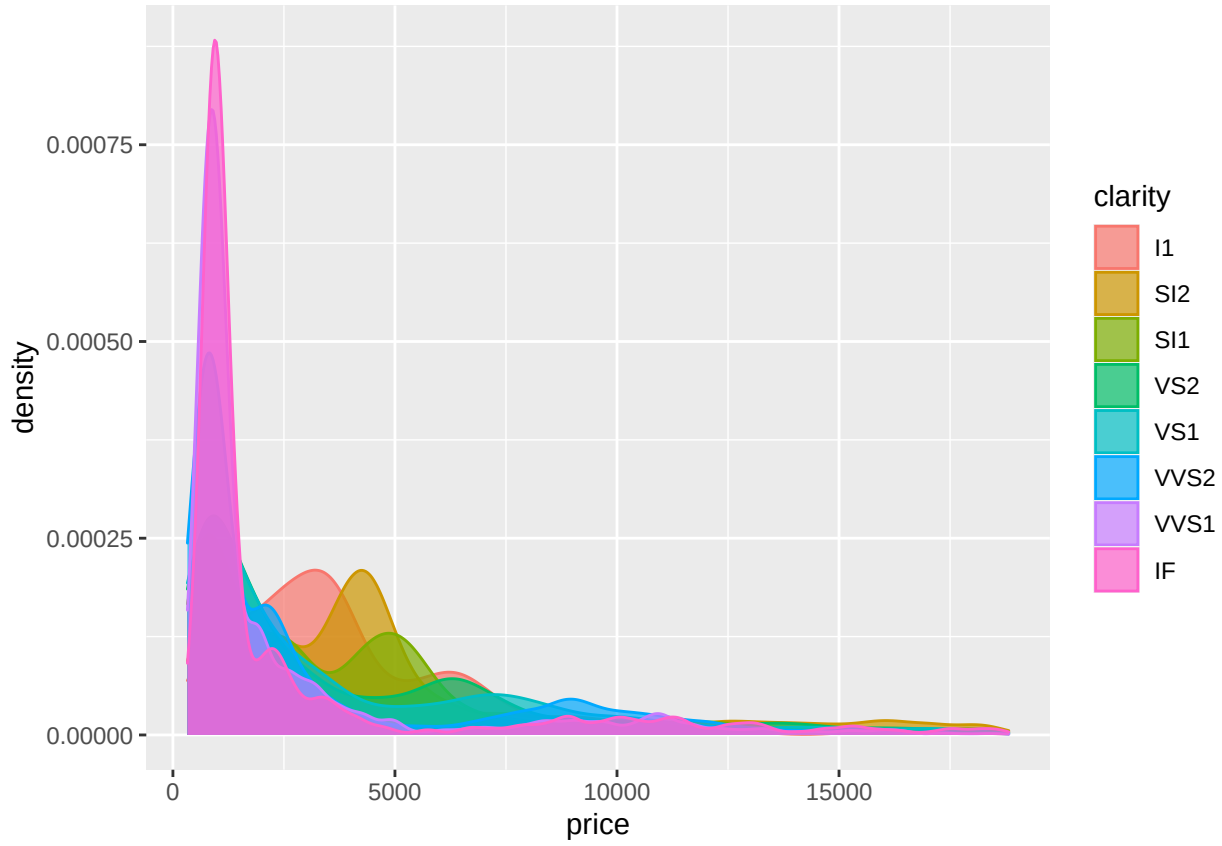

Figure **??** shows the densities of the different levels of 'clarity' for price. A prominent peak is visible near the left for the better quality levels (IF, best, pink; WS1 second best, purple; WS2, third best, blue). We conducted a Analysis of Variance to see if these differences in mean price across 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 verfiy 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 theses 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 frpm the ANOVA returns a p-value of < 2.2e-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.

5.1.5.3 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.

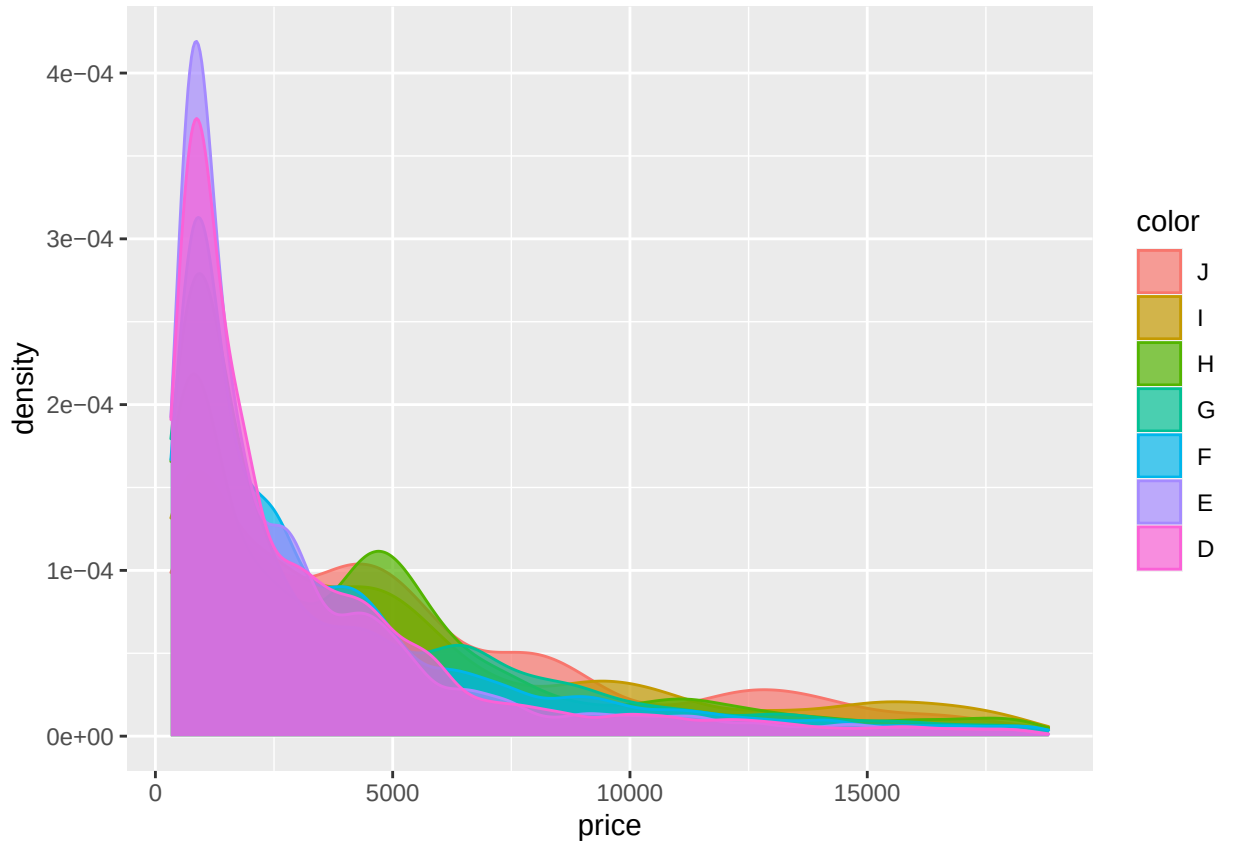

Figure 3: Densities of 'price' for the different levels of 'color'

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 verfiy 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 theses 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.

5.1.6 Outliers and Unusual Points

5.1.6.1 Mahalanobis Distance We decided too 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
```

5.1.6.2 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

##  [1] 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 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 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.

5.1.7 Idendifying whether the zero observations are errors

5.2 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'.

5.2.1 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 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.

5.3 Determining whether the outliers are errors

5.3.1 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.

5.3.2 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.

5.3.3 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.

5.3.4 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 comparible between the highest and second highest, so it is likely that this value can be trusted.

5.3.5 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.

5.3.6 '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.

5.3.7 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.

5.3.8 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.

6 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 0.9216between 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 3.0405091×10^5, 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'.

6.1 Testing to find the best regression model

6.1.1 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 ***
## surprisevery        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
```

```r
# 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
##       surprisevery
##           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'.

6.1.2 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        surprisevery
##            -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.

6.2 Fitting the best model

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

```r
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.

6.3 Scaled multiple regression model

In this version we scale the variables.

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

```
##
## 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.

7 Principal Component Analysis

7.1 Create the Principal Components object

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

```
## Standard deviations (1, .., p=7):
## [1] 2.1826394 1.1339612 0.8311506 0.4168373 0.2007666 0.1815120 0.1113495
##
## Rotation (n x k) = (7 x 7):
##                 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
```

7.2 Plot PC1 and PC2

```
# display the first 10 rows
pca.diamonds$x[1:10,]
```

```
##          PC1          PC2        PC3        PC4          PC5         PC6
## 1  -3.143427 -0.410702133  0.7320886 -0.4645845  0.014724366 0.15103754
## 2  -3.049109  2.285174750 -0.4267556 -0.6726753  0.074027479 0.13419752
## 3  -2.695201  4.972173987 -0.3512970 -0.5859113  0.097536644 0.07701253
## 4  -2.626978  0.008167717 -0.6577434 -0.3686689 -0.016694867 0.07865818
## 5  -2.428969 -0.466806778 -1.0762794 -0.2827061 -0.034419921 0.03743362
## 6  -3.024850 -0.476713889 -0.5338469 -0.5497805 -0.016881246 0.12372244
## 7  -3.018977 -0.220368177 -0.2971601 -0.5298647 -0.007222348 0.12374512
## 8  -2.951290 -0.629002434  0.5496498 -0.3761447  0.005197644 0.12899946
## 9  -2.959609 -0.441698785 -2.9291417 -0.7919847 -0.058598803 0.08368447
## 10 -2.851250  2.478264657 -0.2289987 -0.5447658  0.039101087 0.08510160
##           PC7
## 1  0.13160687
## 2  0.14522865
## 3  0.19592330
## 4  0.08804839
## 5  0.03719709
## 6  0.11166180
## 7  0.12336997
## 8  0.11760637
## 9  0.02350121
## 10 0.16735308
```

```
# calculate values for the scatterplot and screeplot
pca.var <- pca.diamonds$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100,1)
pca.var.per
```

```
## [1] 68.1 18.4  9.9  2.5  0.6  0.5  0.2
```

```
sum(pca.var.per[1:3])
```

```
## [1] 96.4
```

```
plot(pca.diamonds$x[,1], pca.diamonds$x[,2],
     main="Scatterplot of PC1 vs PC2",
      xlab=(paste("PC1 - ", pca.var.per[1], "%", sep="")),
      ylab=(paste("PC2 - ", pca.var.per[2], "%", sep="")))
```

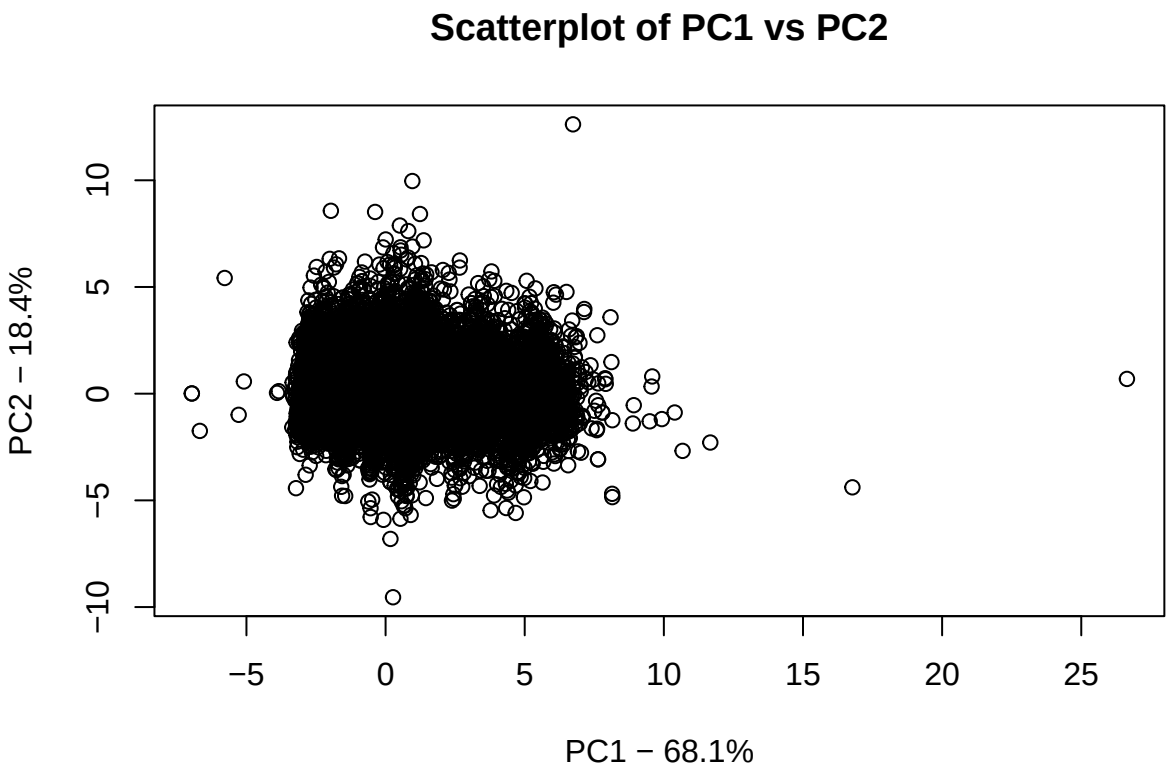

Figure 4: Scatterplot of PC1 vs PC2

Figure 4 shows the scatterplot of PC1 versus PC2. Of note are the two significant outliers to the far right, which are extreme values of PC1. The percentage of variation retained by each Principal Component is included in the axis labels.

8 DO THESE AFFECT THE QUALITY OF THE PCA?

8.1 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
```

The output above identifies the two upper extreme values of PC1.

8.2 Scree plot

```
barplot(pca.var.per, names.arg = pca.var.per,
        xlab = "Principal Components 1 to 7",
        ylab = "% information retained by each PC",
        main="Screeplot of the Principal Components")
```

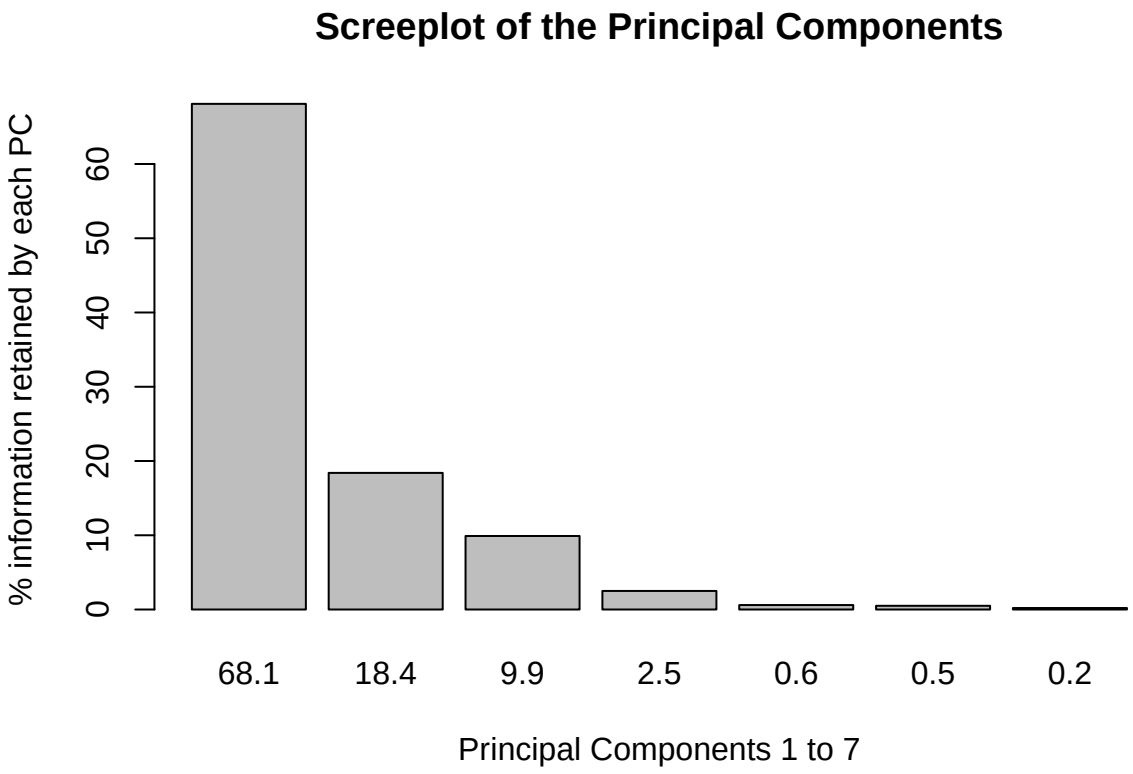

Figure 5: Scree plot of the Principal Components

Figure 5 shows the scree plot for the Principal Components of the diamonds dataset. PC1 contains 68.1% of the variance, PC2 contains 18.4%, while PC3 contains 9.9%. Between them they contain 96.4%, which is clearly very good.

```
## discover which measurements contribute most to PC1

loading_scores <- pca.diamonds$rotation[,1]
diam_scores <- abs(loading_scores) ## get the magnitudes
diam_score_ranked <- sort(diam_scores, decreasing=TRUE)
top_diam <- names(diam_score_ranked[1:7])

top_diam ## show the names
```

```
## [1] "x"     "carat" "y"     "z"     "price" "table" "depth"
```

```
pca.diamonds$rotation[top_diam,1]
```

```
##             x         carat             y             z         price
##  0.4532125054  0.4524454941  0.4472649035  0.4459536619  0.4255192667
##         table         depth
##  0.0995160875 -0.0009161301
```

From the output above, we can see that 'x' is the measurement that contributes the most to PC1, although 'carat' is virtually identical.

9 Factor analysis using an encoded version of the dataset

In this section we create a new version of the dataset which encodes the categorical variables as numerical. Then we repeat the Principal Component analysis and multiple linear regression, and also perform a factor analysis (FA).

Firstly, create a new version of the dataset in which the three categorical variables are transformed to numerical variables.

```
# create new version to transform

diamonds2 <- diamonds[,-11]

# transform the categorical variables to numerical
diamonds2$cut <-as.numeric(diamonds2$cut)
diamonds2$color <-as.numeric(diamonds2$color)
diamonds2$clarity <-as.numeric(diamonds2$clarity)
```

```
# display the first 10 rows to check that it has worked.
diamonds2[1:10,]
```

```
##    carat cut color clarity depth table price    x    y    z
## 1   0.23   5     6       2  61.5    55   326 3.95 3.98 2.43
## 2   0.21   4     6       3  59.8    61   326 3.89 3.84 2.31
## 3   0.23   2     6       5  56.9    65   327 4.05 4.07 2.31
## 4   0.29   4     2       4  62.4    58   334 4.20 4.23 2.63
## 5   0.31   2     1       2  63.3    58   335 4.34 4.35 2.75
## 6   0.24   3     1       6  62.8    57   336 3.94 3.96 2.48
## 7   0.24   3     2       7  62.3    57   336 3.95 3.98 2.47
## 8   0.26   3     3       3  61.9    55   337 4.07 4.11 2.53
## 9   0.22   1     6       4  65.1    61   337 3.87 3.78 2.49
## 10  0.23   3     3       5  59.4    61   338 4.00 4.05 2.39
```

9.1 Diamonds2: Create the Principal Components object

Repeat the Principal Component process.

```
pca.diamonds2 <- prcomp(diamonds2,
                        center=TRUE, scale. = TRUE, retx=TRUE)
pca.diamonds2
```

```
## Standard deviations (1, .., p=10):
##  [1] 2.2376039 1.1989134 1.1108068 0.9930729 0.8777714 0.5985341 0.3574292
##  [8] 0.2004193 0.1641863 0.1097430
##
## Rotation (n x k) = (10 x 10):
##                  PC1         PC2          PC3          PC4         PC5
## carat    0.44058696 -0.06121864  0.005471739 -0.007491061  0.04675840
## cut     -0.08471418 -0.63337908  0.371289014 -0.219790680 -0.24533138
## color   -0.13697759  0.15385990  0.039051358 -0.817733965  0.52073386
## clarity -0.18082577 -0.27907910  0.221225481  0.492722606  0.73670508
## depth    0.00770755 -0.11995053 -0.839364885  0.055131065  0.16173563
## table    0.10734253  0.67117485  0.303010229  0.180186684  0.05869215
## price    0.40320916 -0.14086013  0.086980720 -0.024865522  0.30297594
## x        0.44127077 -0.04619191  0.043477503 -0.038503875  0.01545886
## y        0.43502356 -0.05139248  0.046813782 -0.039289889  0.02019352
## z        0.43488042 -0.06359977 -0.060437800 -0.031942487  0.04103271
##                  PC6         PC7          PC8          PC9           PC10
## carat   -0.007226039  0.16507852 -0.062161510  0.732632672 -4.816069e-01
## cut     -0.586242953 -0.03891188  0.003075668  0.008101089 -5.784727e-03
## color   -0.024982756 -0.10575273 -0.006282563  0.062274993 -2.147671e-02
## clarity -0.009509136 -0.21924944 -0.010129357  0.080526733 -1.477656e-02
## depth   -0.490430051 -0.01184507  0.089821166  0.021116228  5.143468e-02
## table   -0.640145209 -0.01321672  0.009040473 -0.015244120 -5.923764e-05
## price   -0.024558139  0.75502142  0.040187144 -0.361085275  1.222473e-01
## x        0.033188395 -0.23625629 -0.078424497  0.303620494  8.026707e-01
## y        0.056306634 -0.38089786  0.746206330 -0.246564131 -1.951895e-01
## z       -0.019232551 -0.37511125 -0.650573243 -0.410829504 -2.595623e-01
```

9.2 Plot PC1 and PC2

```
# display the first 10 rows
pca.diamonds2$x[1:10,]
```

```
##             PC1         PC2        PC3        PC4        PC5        PC6
##  [1,] -3.050341 -0.39420679 -0.1927390 -1.6002064 -1.2127963  0.1242565
##  [2,] -2.974632  1.97032745  1.4206914 -0.6731925 -0.5923941 -0.4985339
##  [3,] -2.696672  4.19067649  3.2816737  0.5159932  0.5275511  0.4025552
##  [4,] -2.349596  0.25053442 -0.4652761  1.3670599 -1.1165900 -0.4577619
```

```
##  [5,] -1.699513  1.53347852 -1.9488706  1.6631150 -1.7631253  0.3168998
##  [6,] -2.802481  0.09735974 -0.9309265  2.6034187 -0.3114310  0.2037893
##  [7,] -2.990086  0.05982737 -0.4786499  2.4020302  0.3854108  0.3560497
##  [8,] -2.577326  0.24193988 -1.0246907  0.5364668 -1.1857323  1.0823687
##  [9,] -2.737687  3.04447457 -2.5660874  0.4147987  1.1233032 -0.7522592
## [10,] -2.686553  1.93665522  1.5287926  1.5336426 -0.4288914  0.2069841
##             PC7         PC8          PC9        PC10
##  [1,] 0.8215182  0.02265329 -0.030097115 -0.11662569
##  [2,] 0.8179704 -0.04258503 -0.001523237 -0.13615874
##  [3,] 0.5185189 -0.08956100  0.036186215 -0.19307214
##  [4,] 0.5937035  0.04463045 -0.103933463 -0.04477028
##  [5,] 0.8582091  0.06691603 -0.266614831  0.04371216
##  [6,] 0.6346084  0.04093700 -0.039523482 -0.06438437
##  [7,] 0.4400053  0.02133216  0.042814865 -0.09651627
##  [8,] 0.8320468  0.02768647 -0.108076544 -0.06627492
##  [9,] 0.6796937  0.07035403  0.020931183 -0.01939033
## [10,] 0.6499195 -0.01837892 -0.059030629 -0.13196199
```

```
# calculate values for the scatterplot and screeplot
pca.var2 <- pca.diamonds2$sdev^2
pca.var.per2 <- round(pca.var2/sum(pca.var2)*100,1)
pca.var.per2
```

```
##  [1] 50.1 14.4 12.3  9.9  7.7  3.6  1.3  0.4  0.3  0.1
```

```
sum(pca.var.per2[1:3])
```

```
## [1] 76.8
```

```
plot(pca.diamonds2$x[,1], pca.diamonds2$x[,2],
     main="Scatterplot of PC1 vs PC2",
      xlab=(paste("PC1 - ", pca.var.per2[1], "%", sep="")),
      ylab=(paste("PC2 - ", pca.var.per2[2], "%", sep="")))
```

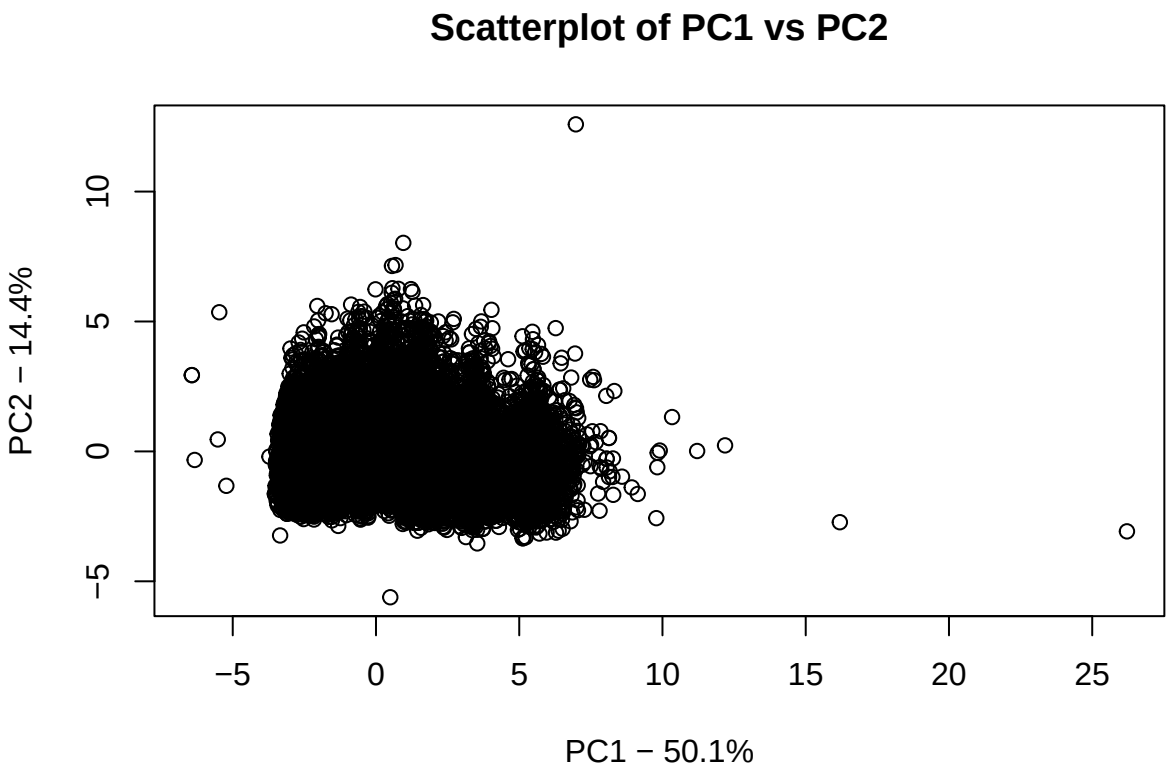

Figure 6: Scatterplot of PC1 vs PC2 (encoded version)

Figure 6 shows the scatterplot of PC1 versus PC2.

9.3 Identify the extreme PC1 values

```
sort(decreasing=T,(pca.diamonds2$x[,1]))[1:10]
```

```
##  [1] 26.211764 16.190542 12.183778 11.209583 10.334619  9.904397  9.832775
##  [8]  9.818712  9.784949  9.136873
```

The output above identifies the two upper extreme values of PC1.

9.4 Scree plot

```
barplot(pca.var.per2, names.arg = pca.var.per2,
        xlab = "Principal Components 1 to 10",
        ylab = "% information retained by each PC",
        main="Screeplot of the Principal Components")
```

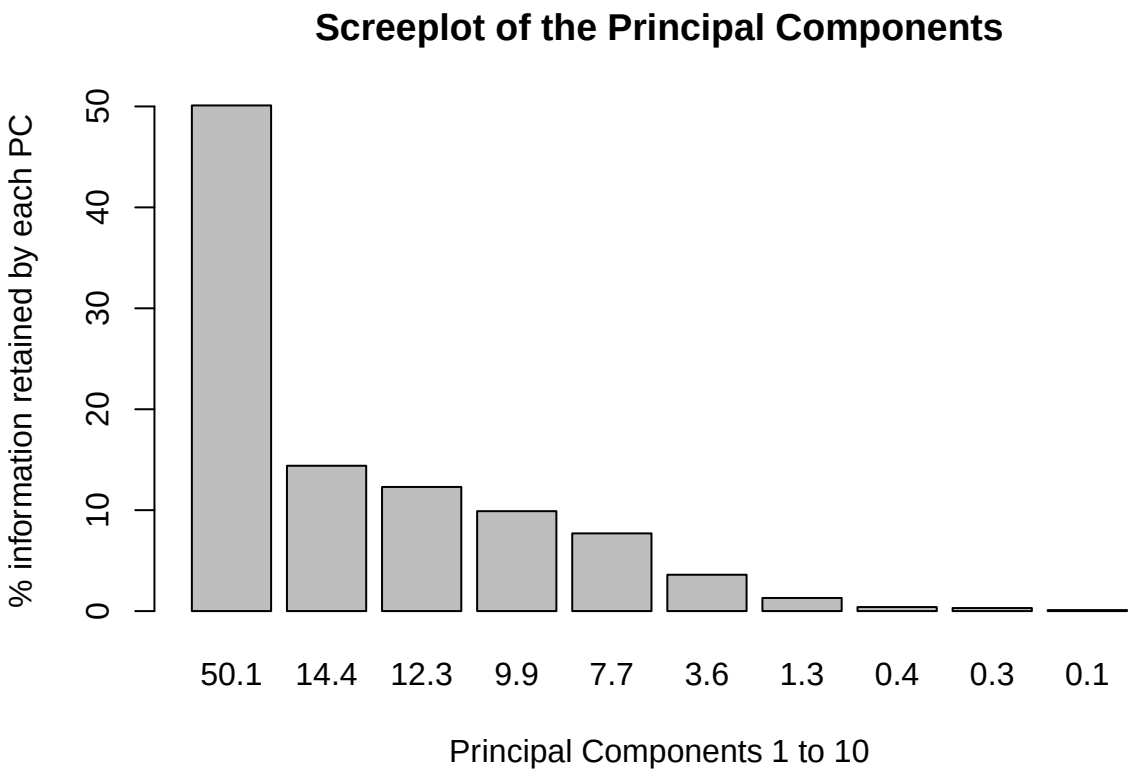

Figure 7: Scree plot of the Principal Components

Figure 7 shows the scree plot for the Principal Components of the diamonds dataset. PC1 contains 50.1% of the variance, PC2 contains 14.4%, while PC3 contains 12.3%. Between them they contain 76.8%, which is clearly very good.

```
## discover which measurements contribute most to PC1

loading_scores2 <- pca.diamonds2$rotation[,1]
diam_scores2 <- abs(loading_scores2) ## get the magnitudes
diam_score_ranked2 <- sort(diam_scores2, decreasing=TRUE)
top_diam2 <- names(diam_score_ranked2[1:10])

top_diam2 ## show the names
```

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

```
pca.diamonds2$rotation[top_diam2,1]
```

```
##           x       carat           y           z       price     clarity
##  0.44127077  0.44058696  0.43502356  0.43488042  0.40320916 -0.18082577
```

```
##       color       table         cut       depth
## -0.13697759  0.10734253 -0.08471418  0.00770755
```

From the output above, we can see that 'x' is the measurement that contributes the most to PC1, although 'carat' is virtually identical.

10 Factor analysis process

```
factanal(diamonds2, factors =2)
```

```
##
## Call:
## factanal(x = diamonds2, factors = 2)
##
## Uniquenesses:
##   carat     cut   color clarity   depth   table   price       x       y       z
##   0.029   0.963   0.903   0.693   1.000   0.953   0.005   0.006   0.045   0.051
##
## Loadings:
##         Factor1 Factor2
## carat    0.878   0.448
## cut             -0.191
## color   -0.145  -0.277
## clarity         -0.548
## depth
## table    0.106   0.189
## price    0.991   0.116
## x        0.828   0.555
## y        0.810   0.547
## z        0.805   0.548
##
##                Factor1 Factor2
## SS loadings      3.782   1.570
## Proportion Var   0.378   0.157
## Cumulative Var   0.378   0.535
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 58331.34 on 26 degrees of freedom.
## The p-value is 0
```

```
factanal(diamonds2, factors =4)
```

```
##
## Call:
## factanal(x = diamonds2, factors = 4)
```

```
##
## Uniquenesses:
##   carat     cut   color clarity   depth   table   price       x       y       z
##   0.027   0.482   0.898   0.676   0.005   0.384   0.005   0.005   0.045   0.039
##
## Loadings:
##         Factor1 Factor2 Factor3 Factor4
## carat    0.938           0.105   0.287
## cut             -0.218  -0.675  -0.119
## color   -0.191                  -0.254
## clarity -0.170          -0.188  -0.508
## depth            0.995
## table           -0.294   0.718
## price    0.995
## x        0.907                   0.401
## y        0.888                   0.395
## z        0.885                   0.407
##
##                Factor1 Factor2 Factor3 Factor4
## SS loadings      4.341   1.136   1.041   0.914
## Proportion Var   0.434   0.114   0.104   0.091
## Cumulative Var   0.434   0.548   0.652   0.743
##
## Test of the hypothesis that 4 factors are sufficient.
## The chi square statistic is 12923.39 on 11 degrees of freedom.
## The p-value is 0
```

11 Multiple regression using diamonds2 (numerically coded)

```
diam2.lm <- lm(price~.,data = diamonds2)
summary(diam2.lm)
```

```
##
## Call:
## lm(formula = price ~ ., data = diamonds2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23560.7   -629.7   -127.9    494.9   9903.1
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  2781.147    428.809   6.486 8.91e-11 ***
## carat       10743.908     51.837 207.263  < 2e-16 ***
## cut           120.750      5.715  21.130  < 2e-16 ***
## color         322.696      3.259  99.003  < 2e-16 ***
## clarity       501.856      3.523 142.450  < 2e-16 ***
## depth         -79.793      4.794 -16.644  < 2e-16 ***
## table         -26.760      2.948  -9.078  < 2e-16 ***
## x            -877.631     35.226 -24.914  < 2e-16 ***
## y              43.735     20.751   2.108   0.0351 *
## z             -29.335     36.017  -0.814   0.4154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1217 on 53930 degrees of freedom
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.907
## F-statistic: 5.845e+04 on 9 and 53930 DF,  p-value: < 2.2e-16
```

```
# use the step() function to perform the stepwise AIC test
step(diam2.lm, direction = "both")
```

```
## Start:  AIC=766375
## price ~ carat + cut + color + clarity + depth + table + x + y +
##     z
##
##           Df  Sum of Sq        RSS    AIC
## - z        1 9.8193e+05 7.9832e+10 766374
## <none>                  7.9831e+10 766375
## - y        1 6.5757e+06 7.9837e+10 766377
## - table    1 1.2198e+08 7.9953e+10 766455
## - depth    1 4.1009e+08 8.0241e+10 766649
## - cut      1 6.6088e+08 8.0492e+10 766818
## - x        1 9.1883e+08 8.0749e+10 766990
## - color    1 1.4509e+10 9.4339e+10 775381
## - clarity  1 3.0038e+10 1.0987e+11 783600
## - carat    1 6.3589e+10 1.4342e+11 797975
##
## Step:  AIC=766373.7
## price ~ carat + cut + color + clarity + depth + table + x + y
##
##           Df  Sum of Sq        RSS    AIC
## <none>                  7.9832e+10 766374
## + z        1 9.8193e+05 7.9831e+10 766375
## - y        1 6.0169e+06 7.9838e+10 766376
## - table    1 1.2159e+08 7.9953e+10 766454
## - depth    1 5.2801e+08 8.0360e+10 766727
```

```
## - cut      1 6.6188e+08 8.0493e+10 766817
## - x        1 1.3535e+09 8.1185e+10 767279
## - color    1 1.4508e+10 9.4339e+10 775379
## - clarity  1 3.0039e+10 1.0987e+11 783599
## - carat    1 6.3588e+10 1.4342e+11 797973

##
## Call:
## lm(formula = price ~ carat + cut + color + clarity + depth +
##     table + x + y, data = diamonds2)
##
## Coefficients:
## (Intercept)        carat          cut        color      clarity        depth
##     2882.48     10743.63       120.83       322.68       501.81       -81.49
##       table            x            y
##      -26.71      -893.26        41.45
```

The output above shows that the multiple linear regression model using the modified dataset with numerically coded categorical variables did not perform better than the one using the original data. The best model according to the AIC test for the numerically coded 'diamonds2' dataset contained eight predictor variables, compared with seven from the best original dataset.

12 Ken's Scatterplots (wasn't sure where would be best to place these- thought you'd have a better idea Ken)

12.1 Colour-coded scatterplots

12.1.1 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()
```

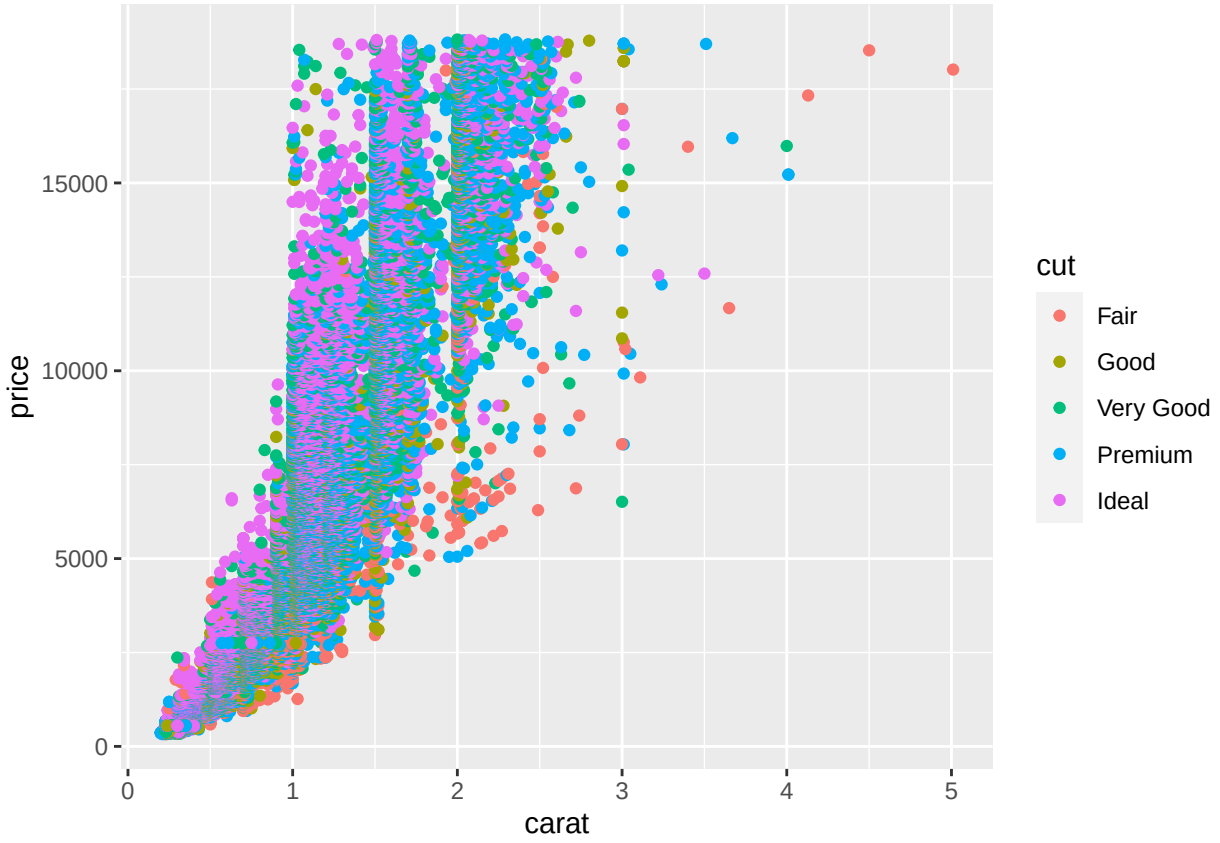

Figure 8: Carat vs Price, coloured by Cut

Figure 8 shows the scatterplot of 'carat' versus 'price' and coloured by 'cut'.

12.1.2 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()
```

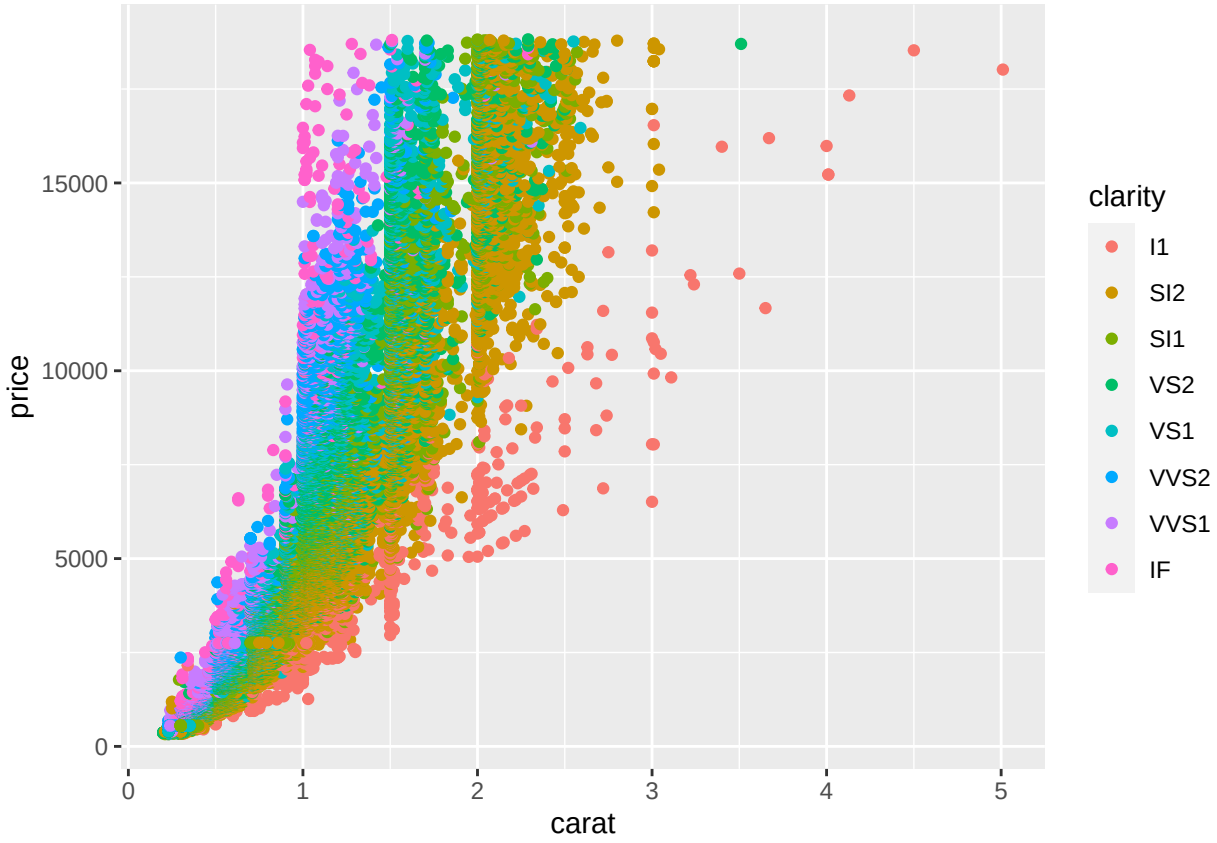

Figure 9: Carat vs Price coloured by Clarity

Figure 9 shows the scatterplot of 'carat' versus 'price' and coloured by 'clarity'.

12.1.3 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()
```

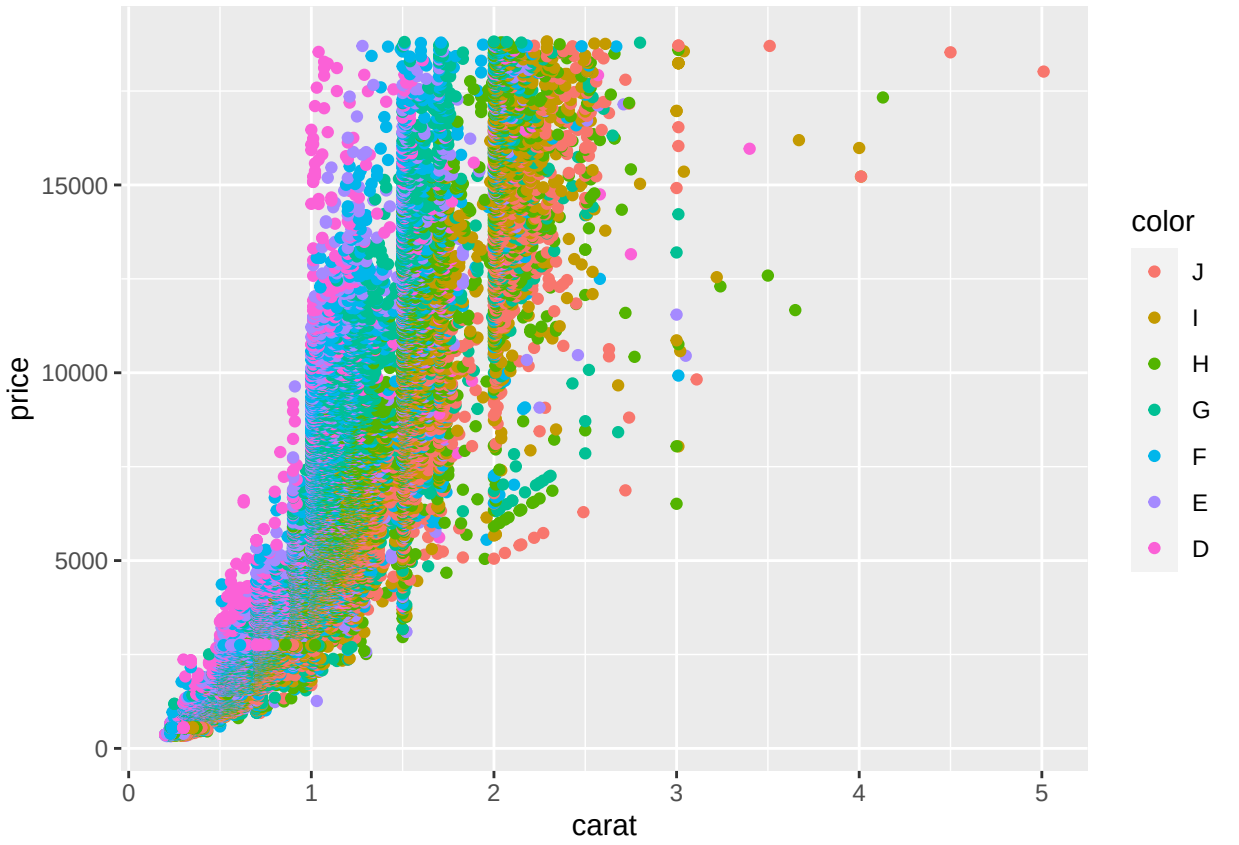

Figure 10: Carat vs Price coloured by Color

Figure 10 shows the scatterplot of 'carat' versus 'price' and coloured by 'color'.

In the three scatterplots figures 8, 9 and 10 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 8 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.

13 Conclusions

14 Bibiography

15 Appendices

15.1 Code for Producing KS Tests

15.2 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Cut

15.3 Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Clarity

Code for producing ANOVA/Levene's Test/Kruskal Wallis/Tukey of Price by Color

Frery, Alejandro C, Luis Gomez, and Antonio C Medeiros. 2020. "A Badging System for Reproducibility and Replicability in Remote Sensing Research." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13: 4988–95.