Set 3: Exploratory Data Analysis and Base R Graphics

STAT GU4206/GR5206 *Statistical Computing & Introduction to Data Science*

Gabriel Young
Columbia University

September 20th, 2019

Last Time

- **Filtering**. Accessing elements of a structure based on some criteria. `v[v>5]`, `m[ m[,1]!=0, ]`.
- **Lists**. Elements can all be different types. Access like `l[[3]]`, `l$name`. Create with `list()`.
- `NA` **and** `NULL` **values**. `NA` is missing data and `NULL` doesn't exist.
- **Factors and Tables**. Factors is how `R` classifies categorical variables.
- **Dataframes**. Used for data that is organized with rows indicating cases and columns indicating variables.
- **Importing and Exporting Data in** `R`. Use `read.csv()` and `read.table()` depending on dataset type. The working directory.
- **Control Statements**. We studied iteration, `for` loops and `while` loops, and `if`, `else` statements.
- **Vectorized Operations**. To be used instead of iterations.

Check Yourself (Warm Up)

Iris

- Use the built-in `iris` dataset:
- This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

Check Yourself (Warm Up)

Tasks: Filtering Data

Use the built-in `iris` dataset:

- How many of the iris are species `versicolor` and have a petal width of less than or equal to 1.2?
- What is the mean petal length of the `setosa` species iris? (Do this with filtering and with `tapply()`.)
- Make a table of iris species for only those iris with sepal width greater than or equal to 3.0.
- Use the `ifelse()` command to create a new variable `Versicolor` that's an indicator variable (1 if the iris species is versicolor and 0 otherwise). Use the `table()` function to check your result.

Exploratory Data Analysis and R Graphics

Diamonds Dataset

- Download `diamonds.csv` from the Canvas page.
- Save to your computer and set your working directory to match that location.
- Run `diamonds <- read.csv("diamonds.csv", as.is = TRUE)`.

```
> diamonds          <- read.csv("diamonds.csv", as.is = T)
> diamonds$cut      <- factor(diamonds$cut)
> diamonds$color    <- factor(diamonds$color)
> diamonds$clarity <- factor(diamonds$clarity)
```

- Note that we could have just ran the line of code:
- `diamonds <- read.csv("diamonds.csv", as.is = FALSE)`.
- `as.is = FALSE` converts all character variables to factors.

Diamonds Dataset

Info on $\sim$ 54000 diamonds from `www.diamondse.info`.

Variables

- **Carat** – Weight of the diamond (0.2 - 5.01).
- **Color** – Diamond color from J (worst) to D (best).
- **Clarity** – A measurement of how clear the diamond is (I1 (worst), SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best)).
- **Cut** – Quality of the cut (Fair, Good, Very Good, Premium, Ideal).
- **Price** – Price in US dollars.

Diamonds Dataset

Code example.

Diamonds Dataset

Think by yourself for a few minutes: what are some interesting questions we could answer using this dataset?

Diamonds Dataset

Think by yourself for a few minutes: what are some interesting questions we could answer using this dataset?

Some ideas:

- What does the distribution of diamond prices look like? Symmetric? Skewed?
- How does a diamond's price relate to its weight?
- Does the relationship between the price and the weight change depending on the quality of the diamond's cut?

Exploratory Data Analysis [1]

Exploratory Data Analysis, or EDA for short, is exploring data in a systematic way.

It's an iterative process:

1. Generate questions about your data.
2. Search for answers by visualizing, transforming, and modelling your data.
3. Use what you learn to refine your questions and or generate new questions.

[1] EDA slides developed from G. Grolemund and H. Wickham.

Exploratory Data Analysis

EDA is a way for you to learn about and better understand your data.

Asking Questions

1. What type of **variation** occurs **within** my variables?
2. What type of **covariation** occurs **between** my variables?

We focus on each of these questions separately.

Variation

Variation is the tendency of measured values of a variable to change measurement-to-measurement.

- Visualizing the distribution of a variable is the best way to understand the patterns of a variable's variance.
- Visualize the distribution of a **categorical** variable using a bargraph.
- Visualize the distribution of a **continuous** variable using a histogram (or boxplot).

Bargraphs in R

Produce a bargraph using `barplot(heights, labels)` where `heights` is a vector of values for the heights of each bar and `labels` is an optional vector of labels for each bar.

- Can use `table()` as input for the bar heights.
- The order that `table()` uses is the order of the factor levels of its input.

Bargraphs in R

Plotting a Bargraph of Diamond Cut

```
> table(diamonds$cut)
```

```
     Fair      Good     Ideal   Premium Very Good 
     1610      4906     21551     13791     12082 
```

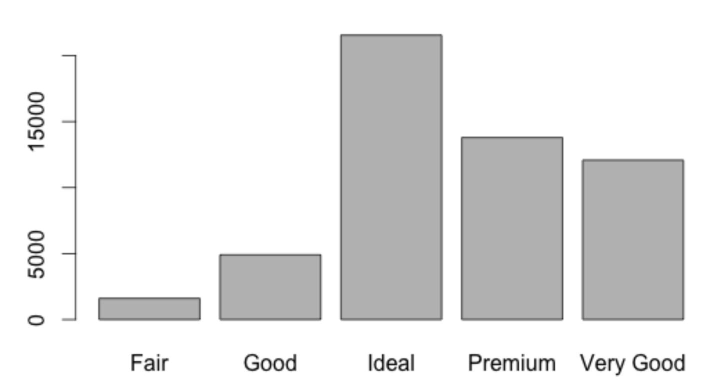
```
> names(table(diamonds$cut))
```

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

Bargraphs in R

Plotting a Bargraph of Diamond Cut

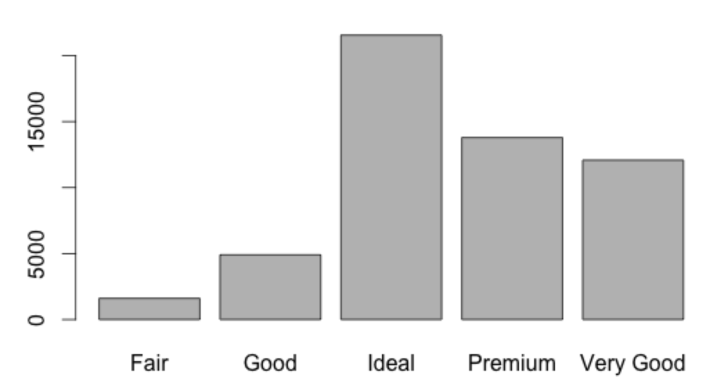
```
> barplot(height = table(diamonds$cut),
+         names.arg = names(table(diamonds$cut)))
```



Bargraphs in R

Plotting a Bargraph of Diamond Cut

```
> barplot(height = table(diamonds$cut),
+          names.arg = names(table(diamonds$cut)))
```



Oops! The order should be `Fair, Good, Very Good, Premium, Ideal`.

Bargraphs in R

Plotting a Bargraph of Diamond Cut

```
> levels(diamonds$cut)
```

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

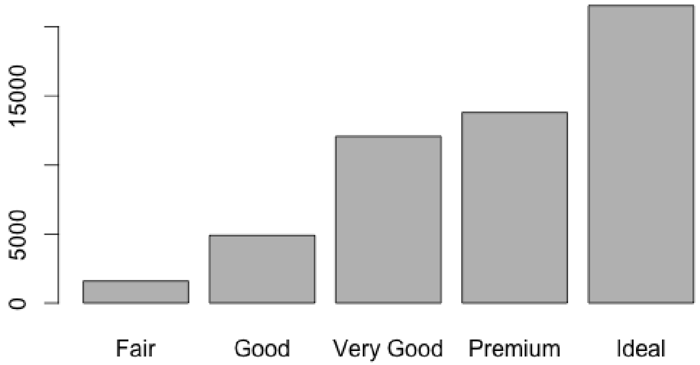
```
> diamonds$cut <- factor(diamonds$cut, level = c("Fair",
+                          "Good", "Very Good", "Premium",
+                          "Ideal"))
> levels(diamonds$cut)
```

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

Bargraphs in R

Plotting a Bargraph of Diamond Cut

```
> barplot(height = table(diamonds$cut),
+         names.arg = names(table(diamonds$cut)))
```



Histograms in R

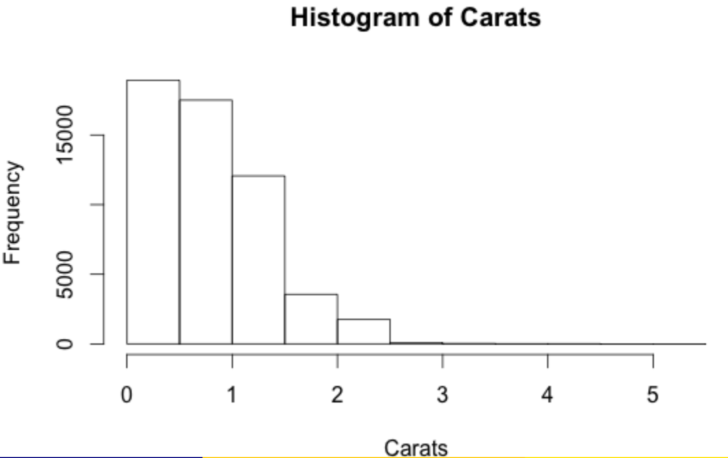
Produce a histogram using `hist(values)` where `values` is a vector of values.

- A histogram divides the x-axis into equally-spaced bins with the height of the bars used to indicate the number of observations falling within the bin.
- Change the width of the histogram's bins using `break =` argument. This specifies the number of bins.
- Make sure to explore different binwidths as different widths will display different patterns in the data.

Histograms in R

Plotting a Histogram of Diamond Cut

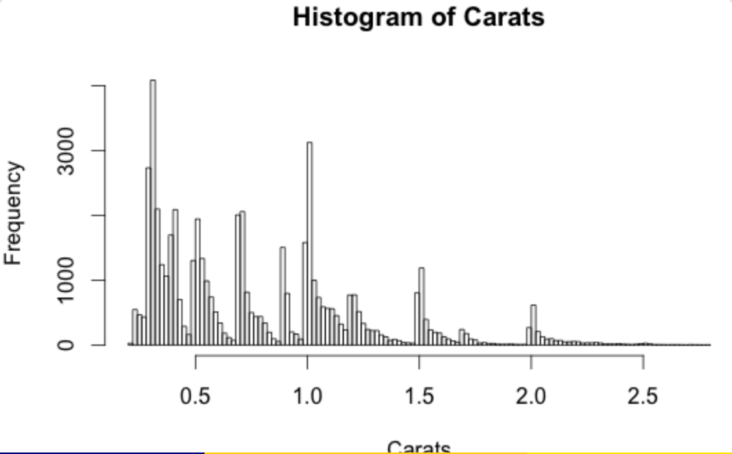
```
> hist(diamonds$carat, main = "Histogram of Carats",
+       xlab = "Carats")
```



Histograms in R

Plotting a Histogram of Diamond Cut

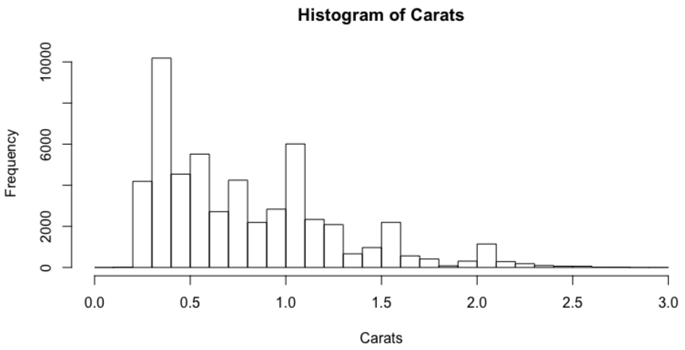
```
> hist(diamonds$carat[diamonds$carat < 3], breaks = 100,
+      main = "Histogram of Carats", xlab = "Carats")
```



Histograms in R

You Can Also Set The Breaks Manually

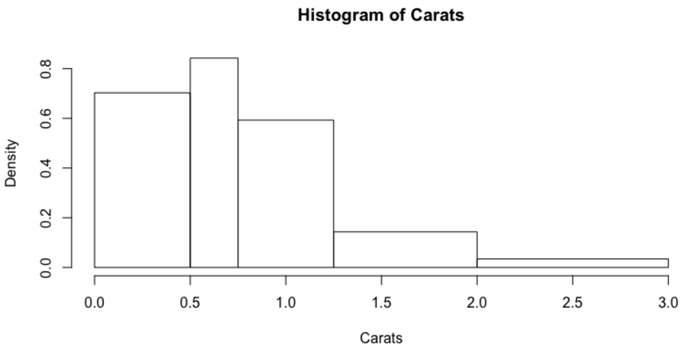
```
> hist(diamonds$carat[diamonds$carat < 3],
+       breaks = seq(0,3,by=.1),
+       main = "Histogram of Carats", xlab = "Carats")
```



Histograms in R

You Can Also Set The Breaks Manually

```
> hist(diamonds$carat[diamonds$carat < 3],
+       breaks = c(0,.5,.75,1.25,2,3),
+       main = "Histogram of Carats", xlab = "Carats")
```

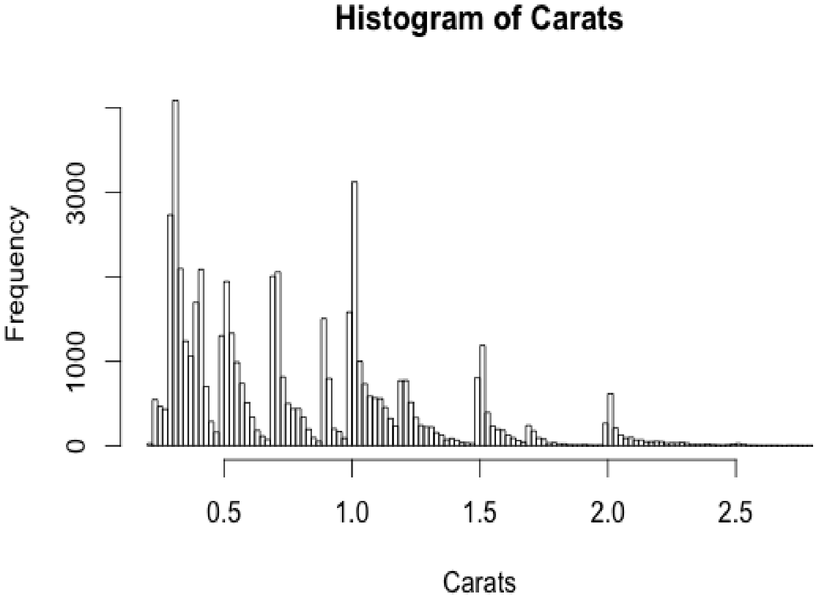


Visualizing Variation

What should we be looking for in these plots?

- Use the plots to create new questions.
  - What do you want to learn more about?
  - Are there any interesting patterns I want to explore.
- Use the plots to better understand the data.
  - Do these plots match my expectations? Why or why not?
  - Do the data cluster in interesting ways?
  - What are the typical values? Outliers? Why?
  - How could this be misleading?

Visualizing Variation



Covariation

Covariation is the tendency for the values of two or more variables to vary together in a related way.

- Visualizing the relationship between variables is the best way to spot covariation.
- Visualize the distribution of a **categorical** variable and a **continuous** variable using a boxplot.
- Visualize the distribution of two **continuous** variables using a scatterplot.

Boxplots in R

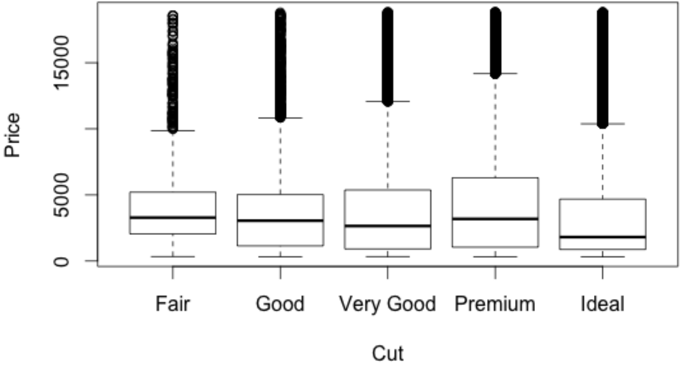
Produce a boxplot (box-and-whisker plot) using `boxplot(values ~ group)` where `values` is a vector of data to be split according to `group`.

- The box stretches from the 25^{th} percentile of the distribution to the 75^{th} percentile (the IQR).
- The line in the middle is the median.
- The 'whiskers' extend to 1.5 times the IQR on either end.

Boxplots in R

Plotting a Boxplot of Diamond Price by Cut

```
> boxplot(price ~ cut, data = diamonds, ylab = "Price",
+         xlab = "Cut")
```



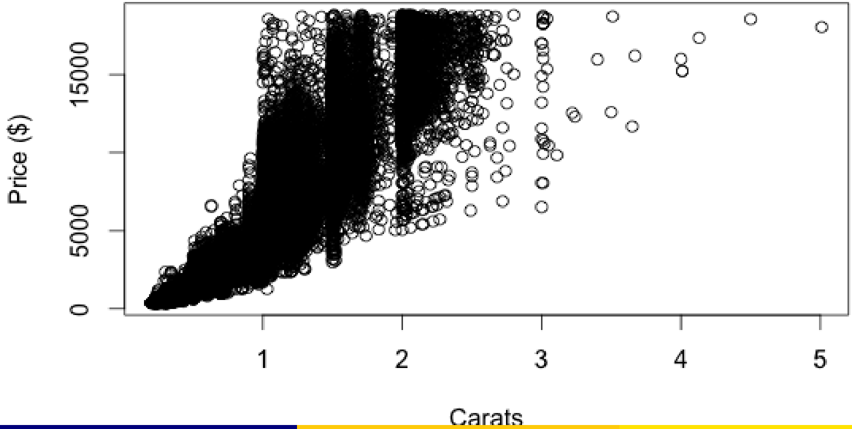
Scatterplot in R

Produce a scatterplot using `plot(x,y)` where `x` is a vector of x-values and `y` a vector of y-values.

Scatterplot in R

Plotting a Scatterplot of Diamond Price vs. Carat

```
> plot(diamonds$carat, diamonds$price, xlab = "Carats",
+       ylab = "Price ($)")
```



Visualizing Covariation

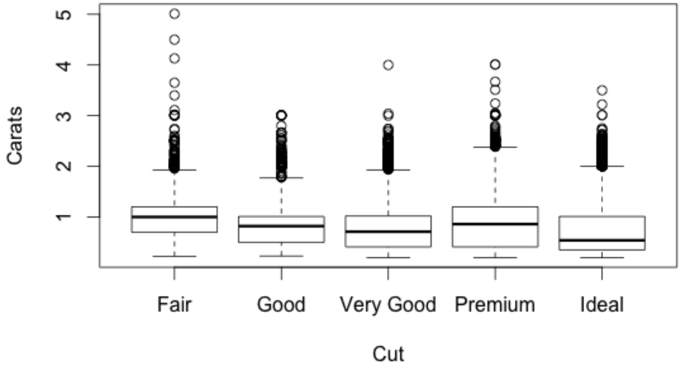
If a relationship exists between two variables it will show up as patterns in your plots.

Ask yourself the following questions.

- Is that pattern random (due to chance)?
- What relationship does the pattern imply?
- Is the relationship strong, weak, linear, non-linear, etc.?
- What other variables might affect the relationship?
- Does the relationship change if you look at individual subgroups of the data?

Boxplots in R

```
> boxplot(carat ~ cut, data = diamonds, ylab = "Carats",
+         xlab = "Cut")
```



Often Visualization Isn't Enough

- Difficult to understand the relationship between `price` and `cut` because `price` and `carat` and `carat` and `cut` are also related.
- Here we would need to use a model (last class, linear models) to consider all these relationships simultaneously.

Basics of Plotting

The `plot()` function.

- The foundation of many of `R`'s graphics functions.
- Often one builds up the graph in stages with `plot()` as a base.
- Each call to `plot()` begins a new graph window.
- Takes arguments, called *graphical parameters*, to change various aspects of the plot. (`?par`)

Building a Visualization: An Example

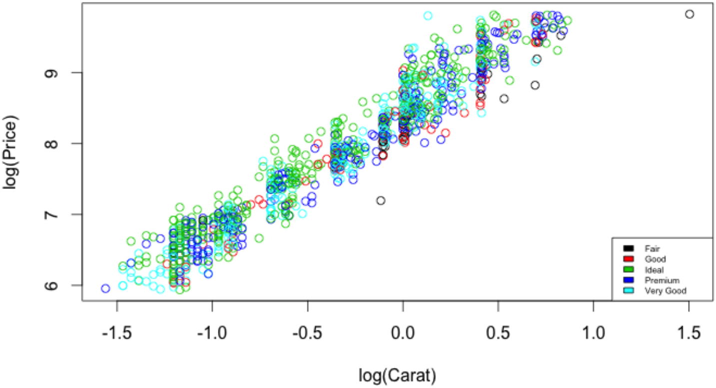
Back to the diamonds.

First, we create a smaller dataset from `diamonds` by randomly selecting 1000 rows.

```
> rows       <- dim(diamonds)[1]
> small_diam <- diamonds[sample(1:rows, 1000), ]
```

Building a Visualization: An Example

```
> plot(log(small_diam$carat), log(small_diam$price),
+      col = small_diam$cut)
> legend("bottomright", legend = levels(small_diam$cut),
+        fill = 1:length(levels(small_diam$cut)), cex = .5)
```



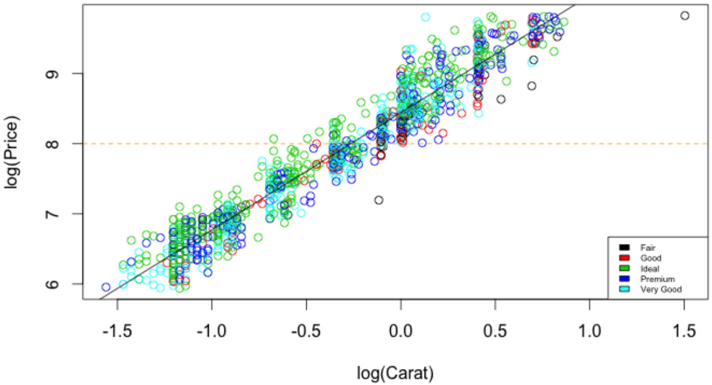
Building a Visualization: An Example

Adding Lines to a Scatterplot

- Add a straight line with `abline(int, slope)`.
  - `int` is the intercept of the line.
  - `slope` is the slope of the line.
- `lines()` can also be used.
  - Most simply, pass `lines()` `x` and `y` vectors and it connects the points.

Building a Visualization: An Example

```
> abline(8, 0, col = "orange", lty = 2)
> lm1 <- lm(log(small_diam$price) ~ log(small_diam$carat))
> abline(lm1)
```



Building a Visualization: An Example

Let's instead plot a regression line for each cut separately.

How do we do this?

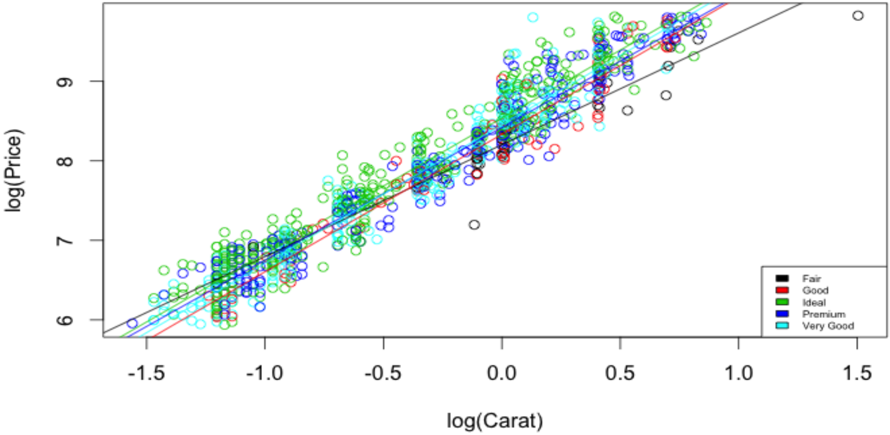
- Think about this task for a moment.

Building a Visualization: An Example

Let's instead plot a regression line for each cut separately.

```
> cuts        <- levels(small_diam$cut)
> col_counter <- 1
> for (i in cuts) {
+   this_cut    <- small_diam$cut == i
+   this_data   <- small_diam[this_cut, ]
+   this_lm     <- lm(log(this_data$price)
+                      ~ log(this_data$carat))
+   abline(this_lm, col = col_counter)
+   col_counter <- col_counter + 1
+ }
```

Building a Visualization: An Example



Building a Visualization: An Example

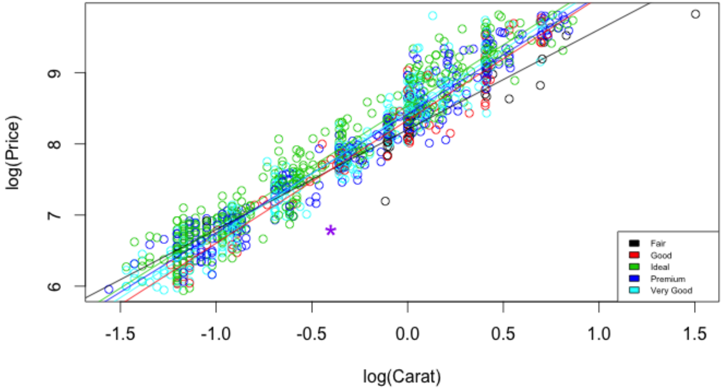
Adding Points to a Scatterplot

- Most easily done with `points(x,y)`.
- `(x,y)` is the location of the point to be added.
- `example(points)` could be helpful.

Building a Visualization: An Example

We add a new point for a diamond that is $898 and 0.67 carats.

```
> points(-0.4, 6.8,  pch = "*", col = "purple")
```



Building a Visualization: An Example

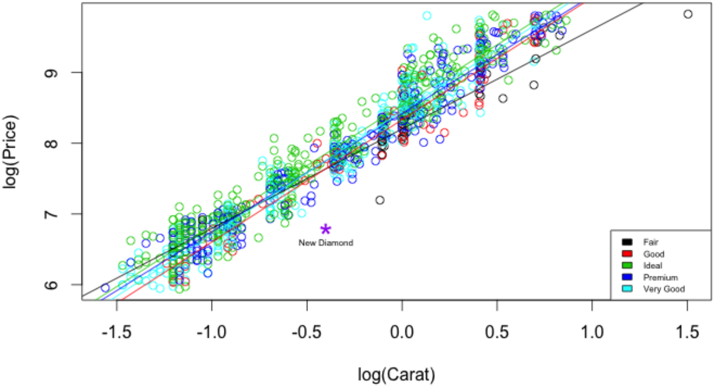
Adding Text to a Scatterplot

- Most easily done with `text(x,y, label)`.
  - `(x,y)` is the location of the text to be added.
  - `label` is the the text to be added at the specified location.
- The `locator()` function we saw in Lecture 1 can be useful here.

Building a Visualization: An Example

We add text to the new point we just added.

```
> text(-0.4, 6.8 - .2, "New Diamond", cex = .5)
```



Useful Graphical Parameters

The table below lists a selection of R's graphical parameters. More info at http://www.statmethods.net/advgraphs/parameters.html or using ?par.

Parameter	Description
pch	*Point Character*. Character of the points in the plot.
main	Title of the plot.
xlab, ylab	Axes labels.
lty	*Line Type*. E.g. 'dashed', 'dotted', etc.
lwd	*Line Width*. Line width relative to default = 1.
cex	*Character Expand*. Character size relative to default = 1.
xlim, ylim	The limits of the axes.
mfrow	Plot figures in an array (e.g. next to each other).
col	Plotting color.

Saving graphs to Files

Coding Example.

Optional Reading

- Chapter 2 (2.3.2, 2.3.5) An Introduction to Statistical Learning.
- Chapter 7 (Exploratory Data Analysis) in R for Data Science.

Moving on from Base R Graphics

- We will learn about more advanced plotting tools using the `ggplot2` package soon.
- Base `R` graphics are good when you want to produce something quick, like for EDA.
- `ggplot2` provides more sophisticated graphing tools for communicating your results.
- Note: Some practitioners still prefer Base `R` graphics over `ggplot`.