Data Analysis in R

Visualisation with ggplot2

Michael E DeWitt Jr

2018-11-15 (updated: 2018-11-14)

We've finally made it to graphing!

Statistical Graphics

Statistical Graphics

Understanding your data

Statistical Graphics

Understanding your data

Anomaly detection

Statistical Graphics

Understanding your data

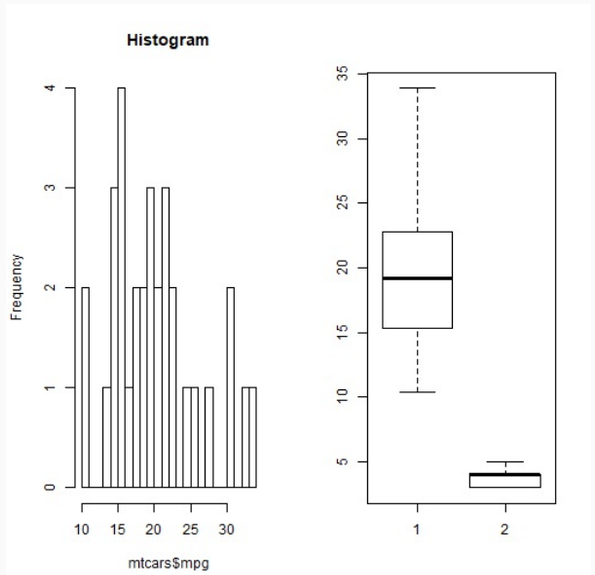
Anomaly detection

Communicating results!

Graphing in R

Base R has internal plotting/ graphing functionality

```
par(mfrow=c(1,2))
hist(mtcars$mpg, breaks = 30, main = "Histogram")
boxplot(mtcars$mpg, mtcars$gear)
```



Base R Graphing

Base R graphics are *pretty* good

Base R Graphing

Base R graphics are *pretty* good

Called pen and ink (e.g. each layer is drawn on top of the other)

Base R Graphing

Base R graphics are *pretty* good

Called pen and ink (e.g. each layer is drawn on top of the other)

Have to redraw entire graphic if you need to "go back"

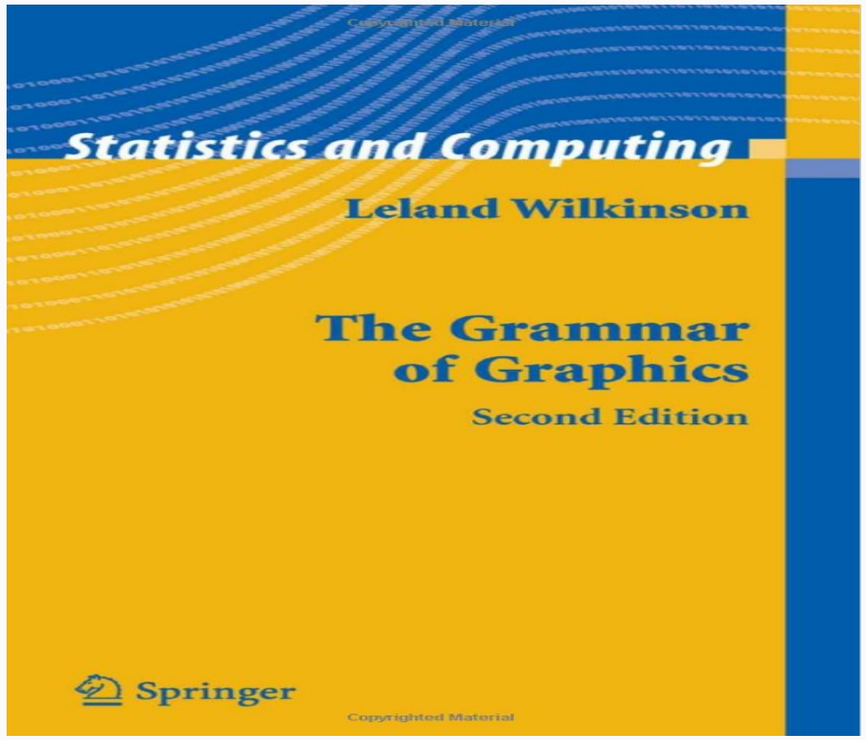
Enter `ggplot2`

The `ggplot2` package is based on the *Grammar of Graphics* by Leland Wilkinson

Grammar of Graphics proposes a philosophical underpinning for all statistical graphics

`ggplot2` is an implementation of the the grammar of graphics

Cheatsheet here



So what does it mean?

> In brief, the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system. Faceting can be used to generate the same plot for different subsets of the dataset. It is the combination of these independent components that make up a graphic.

Wickham *ggplot2*:Elegant Graphics for Data Analysis

The Grammar of Graphics isn't a Guide for Making *Good* Graphs

Good Graphics References Are All Around

Books

Edward Tufte's Books

Story Telling With Data

The Elements of Graphing Data

Data Visualization: A Practical Introduction (All ggplot2)

Blogs

Flowing Data

Our Wold in Data

FiveThirtyEight

Components of the Grammar

In the grammar of graphics there are the following components of a graphic

- **data** - what data are you trying to plot
- **mappings** - what aesthetic mapping are you trying to plot
- **layers**
    - geometries - bars, lines, points, etc
    - statistics - statistical summaries of the data (e.g. counts)
- **scales** - map color, size, shape
- **coordinate systems** - describe how the data are mapped
- **facets** - breaking the data into subsets of small multiples
- **themes** - the details of display like font size, color pallets, etc

Components Map Directly to `ggplot`

Each component has a function or argument in `ggplot2`

- **data** - `data`
- **mappings** - `aes(x, y, color, size, group)`
- **layers**
  - geometries - `geom_` (e.g. `geom_bar`)
  - statistics - `stat`
- **scales** - `scale_` (e.g.) `scale_colour_discrete`
- **coordinate systems** - `coord_polar`
- **facets** - `facet_`
- **themes** - `theme_minimal`

So Let's Start With a Graph

The `diamonds` set is automatically loaded with `ggplot2`

```
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

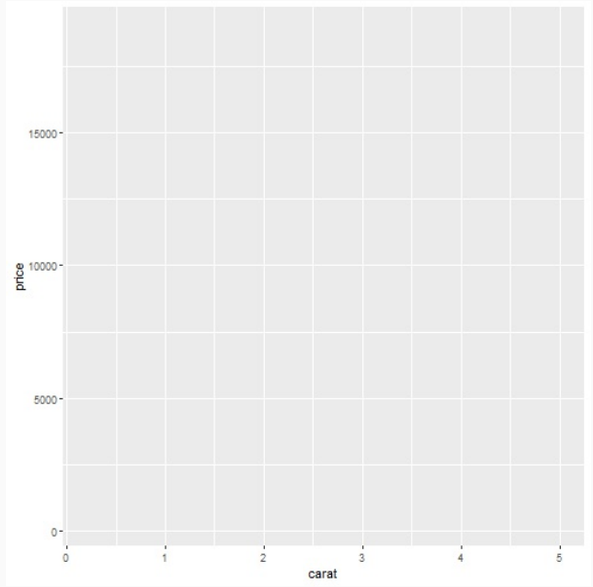
Building the Graph with `ggplot`

```
ggplot(data = diamonds)
```


Now we have initiated a graphical object

Now Add Our Aesthetics with `aes`

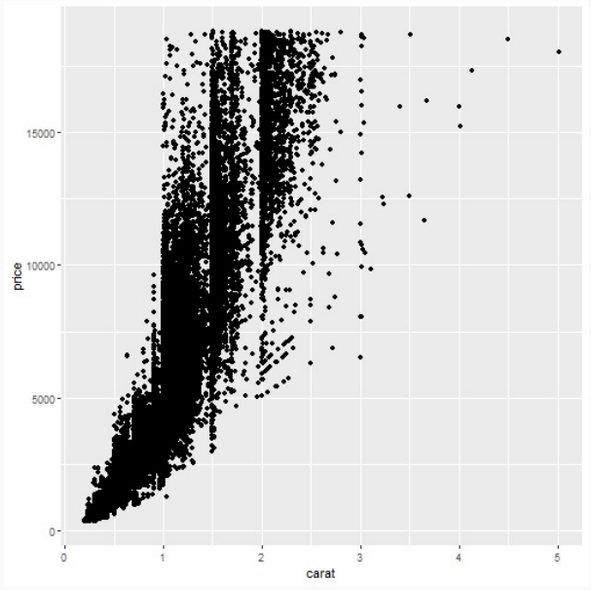
```
ggplot(data = diamonds, aes(x = carat, y= price))
```



Now Add Our Layer

`geom_point` will place a dot for each x-y pair

```
ggplot(data = diamonds, aes(x = carat, y= price))+
   geom_point()
```



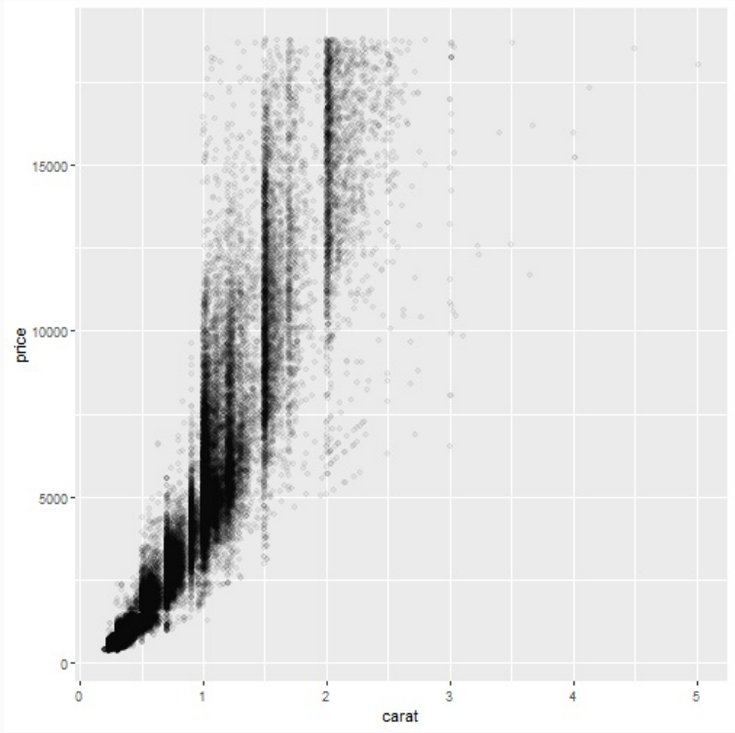
Can manipulate features of a given `geom`

`alpha` allows us to set the transparency

```
ggplot(data = diamonds,
       aes(x = carat, y= price))+
   geom_point(alpha = .05)
```

This is where we could change the point color with `colour = "red"`

We could change the marker type `shape = 5`

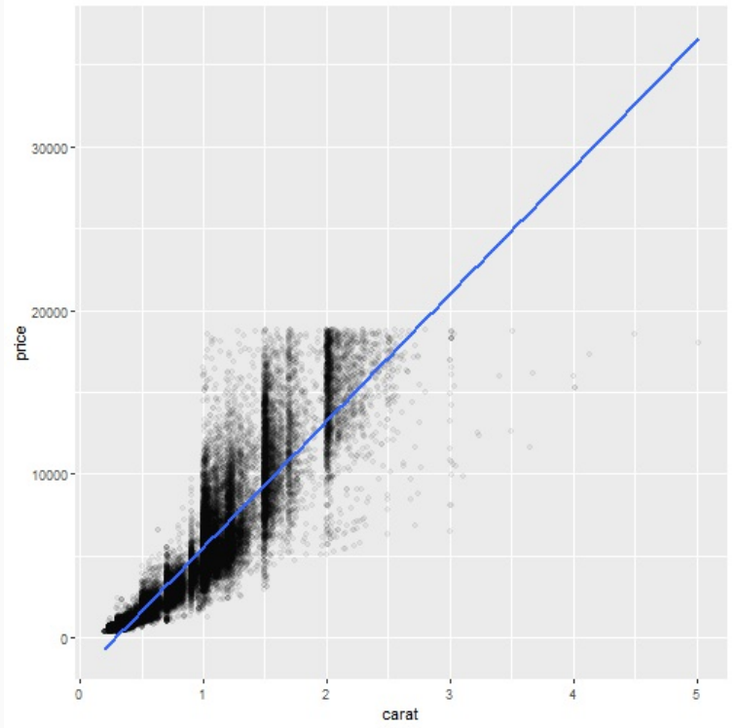


Now We Can Add a Statistic

```
p <- ggplot(data = diamonds,
            aes(x = carat, y= price))+
  geom_point(alpha = 0.05)+
   geom_smooth(method = "lm") # Our Statistic
```

In this case we are using a linear model, but other models can be used:

- Generalised linear models (logistic regression, poisson regression)
- LOESS
- GAMs

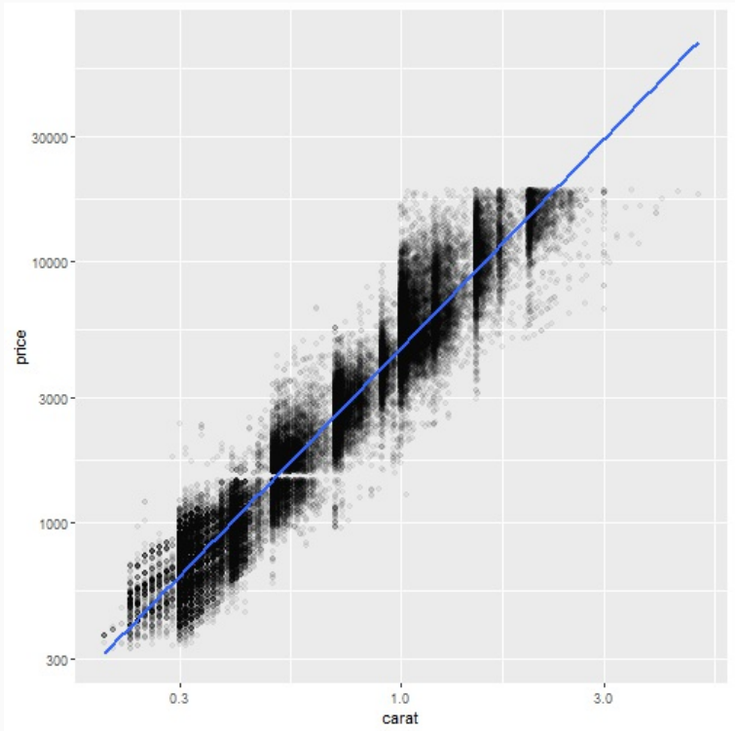


Modify the Scales with `scale_`

```
p <- p+
   scale_y_log10()+ # Scale for the Y-axis
   scale_x_log10() # Scale for X-axis
```

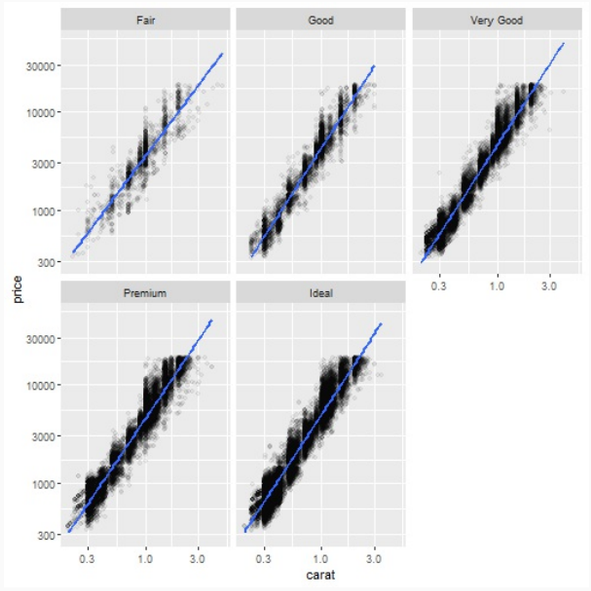
Through scale arguments we can change

- Other transformations like square root transformation
- Other "scales" like percents, discrete data, or dates
- Break points (e.g. what labels appear)
- Limits
- Control colours and some other properties



We Can Introduce Facets

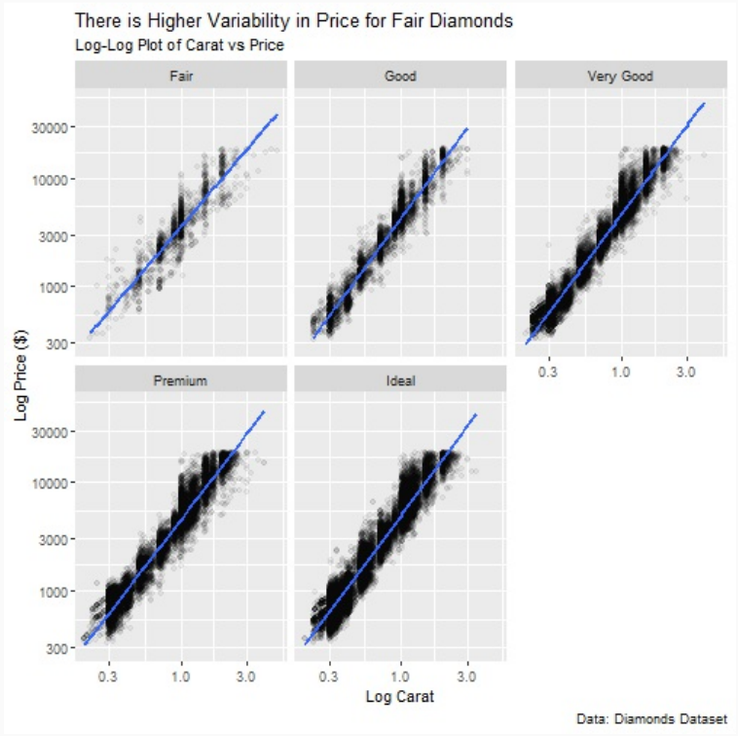
```
(p<- p+
   facet_wrap(~cut) # Facet by "cut"
 )
```



Now we can add some descriptors

```
p <- p +
  labs(
    title = "There is Higher Variability in Price for Fai
    subtitle = "Log-Log Plot of Carat vs Price",
    caption = "Data: Diamonds Dataset",
    y = "Log Price ($)",
    x = "Log Carat"
  )
```

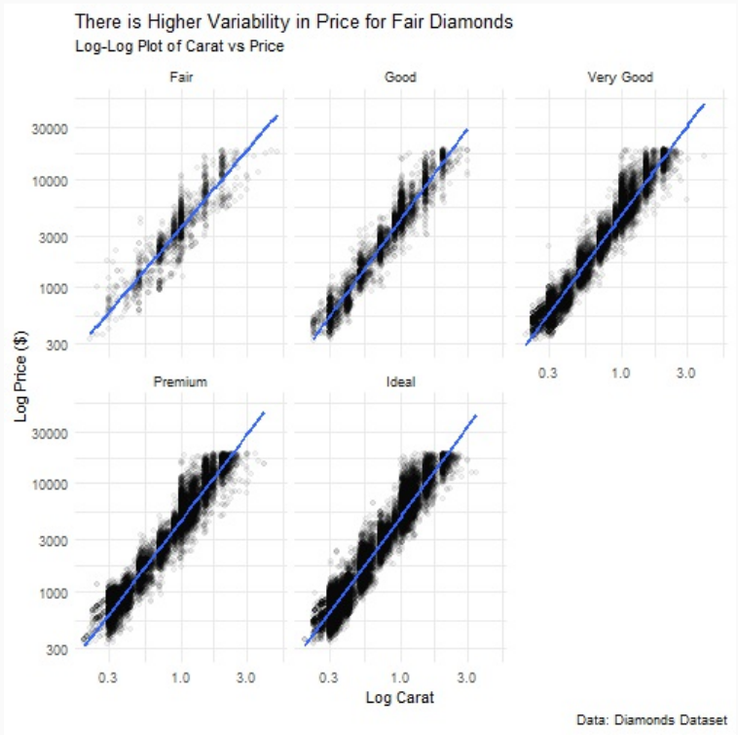
As you add additional aesthetics you can change the description in the `labs` arguments (e.g. if you added a colour aesthetic you could rename it here)



We can change the theme and a few details

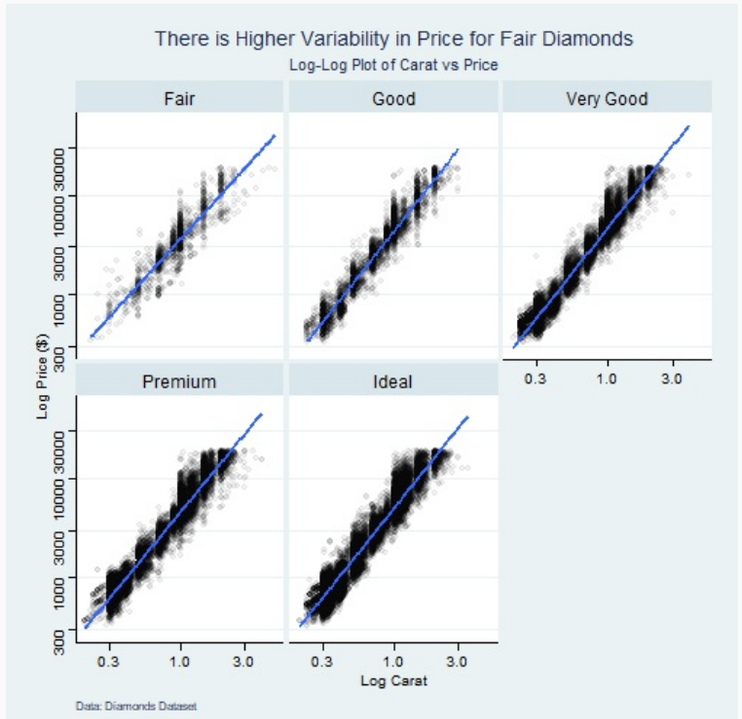
There are many default themes, but checkout ggthemes

```
p <- p+
   theme_minimal()
```



Or if it makes you more comfortable...

```
library(ggthemes)
p+
   theme_stata()
```

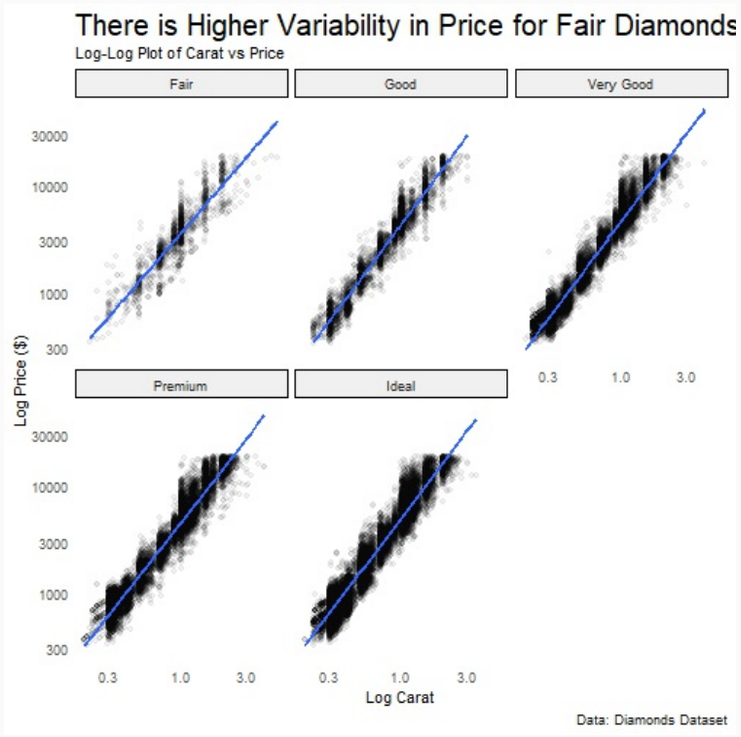


Or Customise Even Further

You can change *every* element of the graph

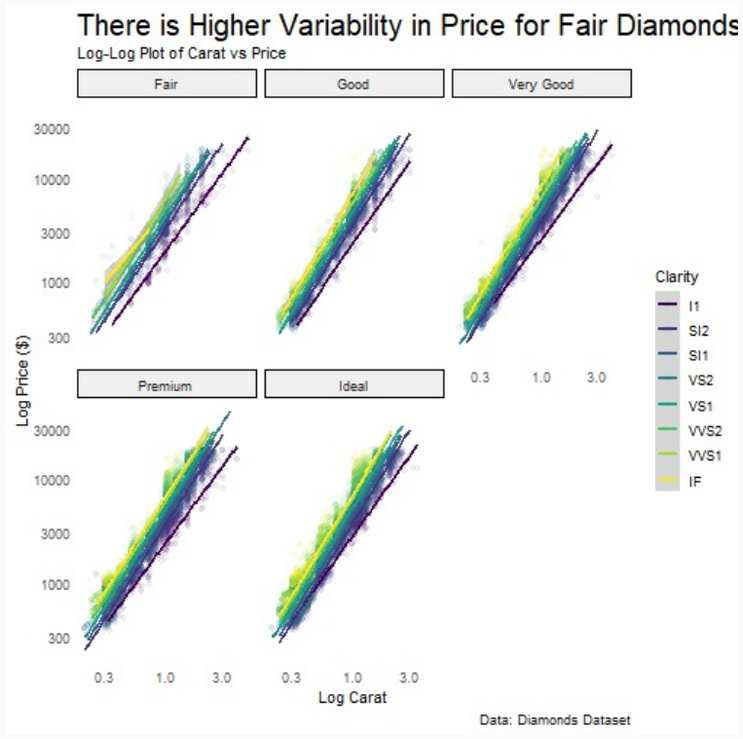
```
p <- p+
  theme(panel.grid =
          element_blank(),
        plot.title =
          element_text(size = 20),
        strip.background =
          element_rect(fill = "#F0F0F0"))
```

Check them all out here



Add Another Aesthetic

```
p<- p +
   aes(color = clarity)+
  labs(color = "Clarity")
```



And they layer and can be saved

You can layer additional components until you complete your message.

But...now you may want to save your image.

`ggave` has this functionality

Can save to pdf or png and specify the dimensions

```
p %>%
  ggsave(filename = "outputs/my_cool_plot.pdf")
```

Making More Complex Graphics

`ggplot2` has been extended in many ways

- Stitch plots together with `cowplot`
- Add GIS capabilities with `sf`
- Network analysis with `ggraph`
- interactive graphics with `plotly`

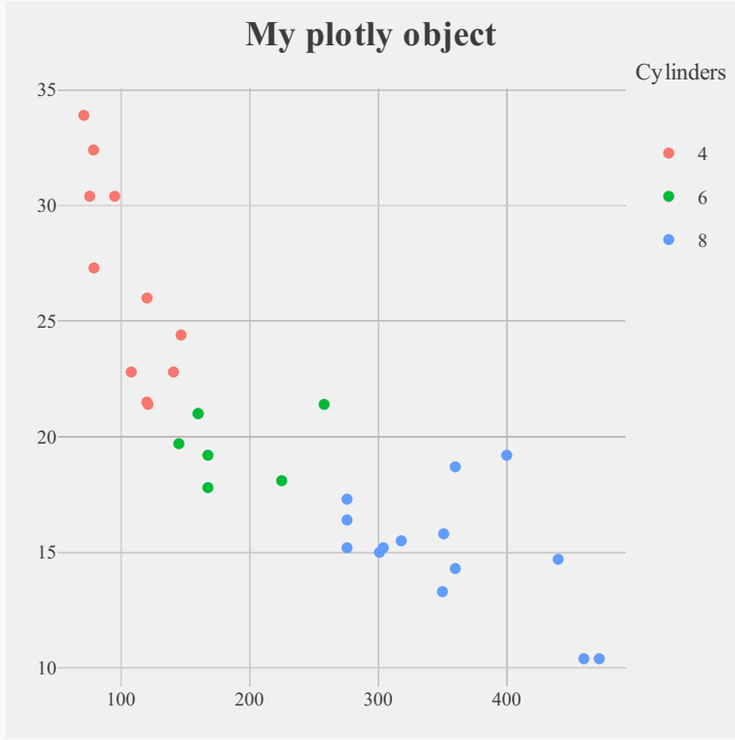
Interactivity with `plotly2`

The `plotly` library allows you to make interactive graphs with `ggplot2` objects.

These are great features on websites.

```
p2 <- ggplot(mtcars,
             aes(disp, mpg,
                 color= as.factor(cyl)))+
  geom_point()+
  labs(
    title = "My plotly object",
    color = "Cylinders"
  )+
  theme_fivethirtyeight()

library(plotly)
ggplotly(p2)
```



Recap

`ggplot2` is an implmentation of the *Grammar of Graphics*

It allows us to make publication quality graphics by mapping data to aesthetics via geometries and statistics.