Introduction to R

Alex Hollingsworth, Grant McDermott, and Kelli Marquardt

Indiana University | University of Oregon | Chicago FED

2021-07-13

This document was made using R-Markdown.

- R-Markdown is great because you can combine text with R code and results.
  - Text is formatted using markdown syntax,
  - It's saved as an `.Rmd` file.
- You can also just simply write `.R` code that is analogous to a `.do` file where there's a list of commands with comments. No need for this fancy `.Rmd` Markdown business.

This document was made using R-Markdown.

- It's pretty cool to integrate code with results right into the presentation.

Input

```
```{r}
 sin(3)
```
```

Output

```
sin(3)
```

```
## [1] 0.14112
```

Learn more about R-Markdown

- If you are curious about markdown formatting, you can check out this handy guide. https://www.markdownguide.org/basic-syntax/
- To learn more about R-Markdown, this "cheat-sheet" https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf, is a good starting place.

Running code

Code Chunks

This is a code chunk that will evaluate the command `sin(3)`

```
```{r}
 sin(3)
```
```

The output of this will appear as follows:

```
## [1] 0.14112
```

Inline Code

You can also create calculations inline for example `r sin(3)` will evaluate `sin(3)`. With output rendering like this 0.14112.

Loading packages

```r
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

You can install packages using the `install.packages()` syntax, but see the section on `pacman` below before worrying about that.

Viewing data

```
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  0.29 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
```

Viewing data

We can get a list of column names fairly easily too

```
names(diamonds)
```

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

Viewing data

```
diamonds %>% group_by(color) %>% summarise(mean(price))
```

```
## # A tibble: 7 x 2
##   color `mean(price)`
##   <ord>         <dbl>
## 1 D             3170.
## 2 E             3077.
## 3 F             3725.
## 4 G             3999.
## 5 H             4487.
## 6 I             5092.
## 7 J             5324.
```

Here the pipe command `%>%` will take the output from the command on the left and "pipes" it as input to the command on the right.

Viewing data

```
diamonds %>%
  group_by(color) %>%
  summarise(mean(price))
```

```
## # A tibble: 7 x 2
##   color `mean(price)`
##   <ord>         <dbl>
## 1 D             3170.
## 2 E             3077.
## 3 F             3725.
## 4 G             3999.
## 5 H             4487.
## 6 I             5092.
## 7 J             5324.
```

1. You can just move onto a new line without an error. This is neat because it allows us to write cleaner code and you don't need a delimiter or the `///` you may be used to from stata. So we can rewrite the above like this.
2. In more recent versions of R (4.1 and above), you don't need to have `dpylr` installed to pipe. You can do the same thing using `|>`.

Summary statistics table

First, load the model summary package

```
library(modelsummary)
```

Summary statistics table

Now we can use the `datasummary_skim` function.

```
datasummary_skim(diamonds)
```

	Unique (#)	Missing (%)	Mean	SD	Min	Median	Max	
carat	273	0	0.8	0.5	0.2	0.7	5.0	
depth	184	0	61.7	1.4	43.0	61.8	79.0	
table	127	0	57.5	2.2	43.0	57.0	95.0	
price	11602	0	3932.8	3989.4	326.0	2401.0	18823.0	
x	554	0	5.7	1.1	0.0	5.7	10.7	
y	552	0	5.7	1.1	0.0	5.7	58.9	
z	375	0	3.5	0.7	0.0	3.5	31.8	

Summary statistics table

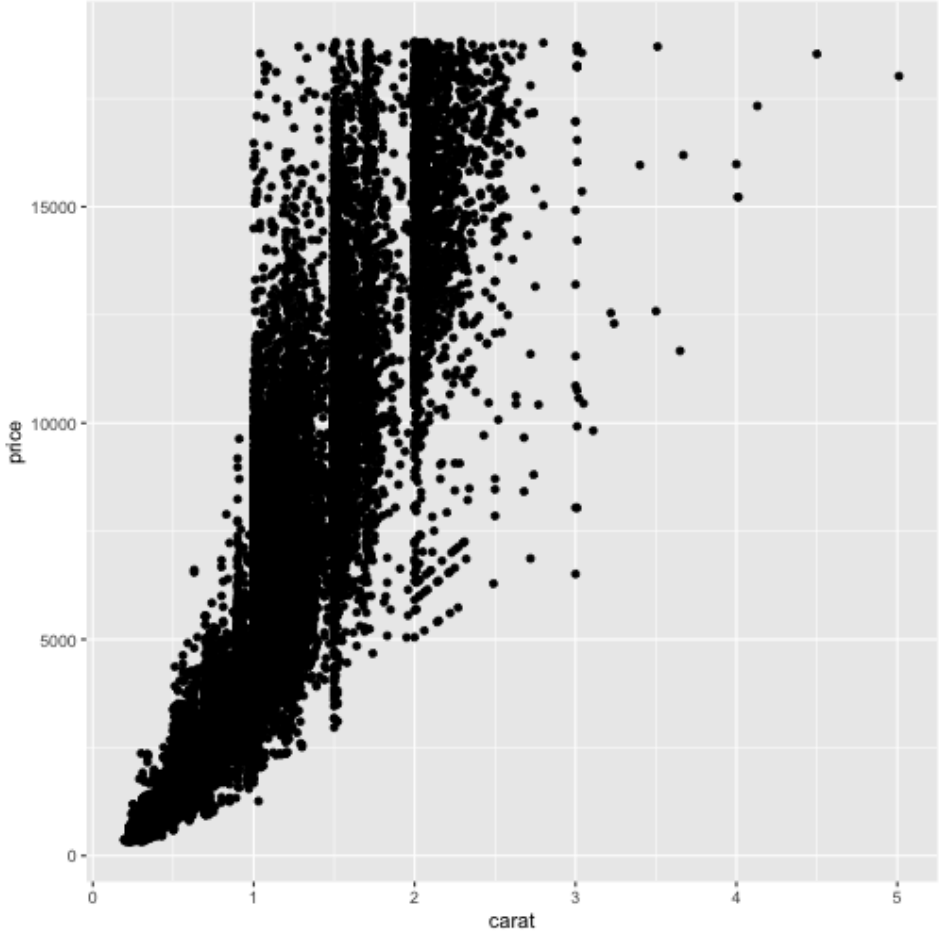
But not every variable is numeric. `datasummary` has this covered with the `type = "categorical"` sub-option.

```
datasummary_skim(diamonds, type = "categorical")
```

		N	%
cut	Fair	1610	3.0
	Good	4906	9.1
	Very Good	12082	22.4
	Premium	13791	25.6
	Ideal	21551	40.0
color	D	6775	12.6
	E	9797	18.2
	F	9542	17.7

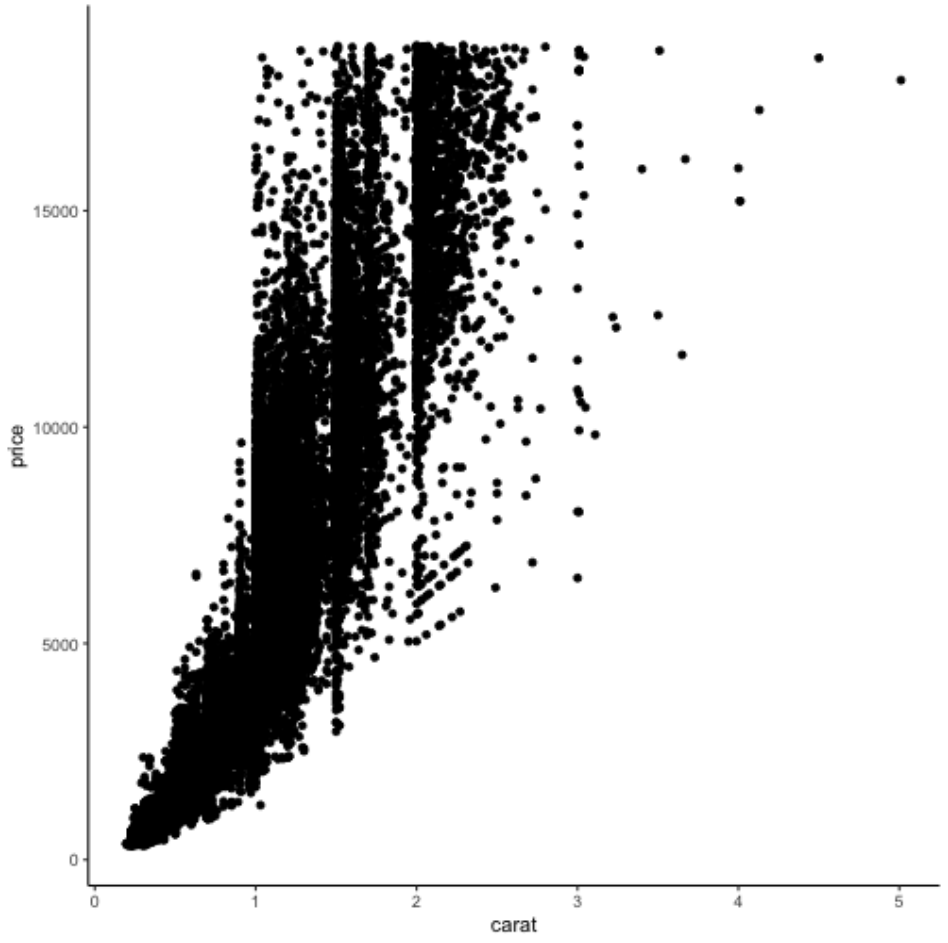
Graphing

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



Graphing

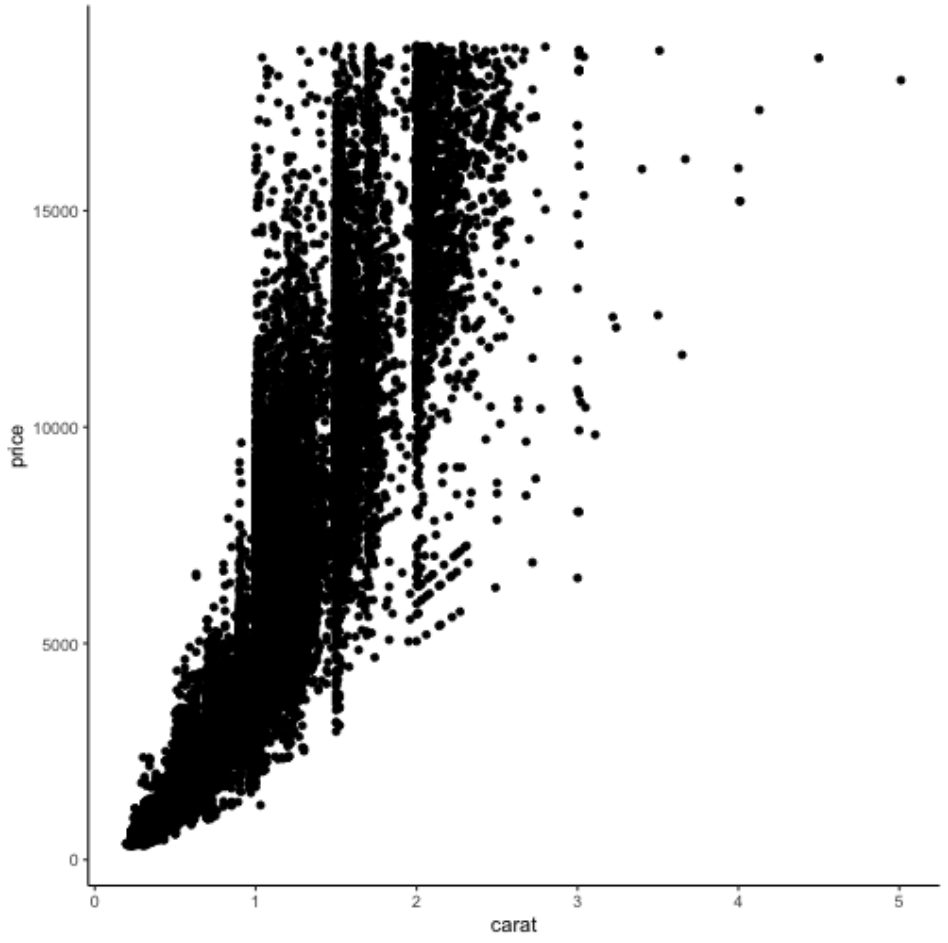
```
ggplot(data = diamonds, aes(y = price, x =
    geom_point() +
    theme_classic()
```



Graphing

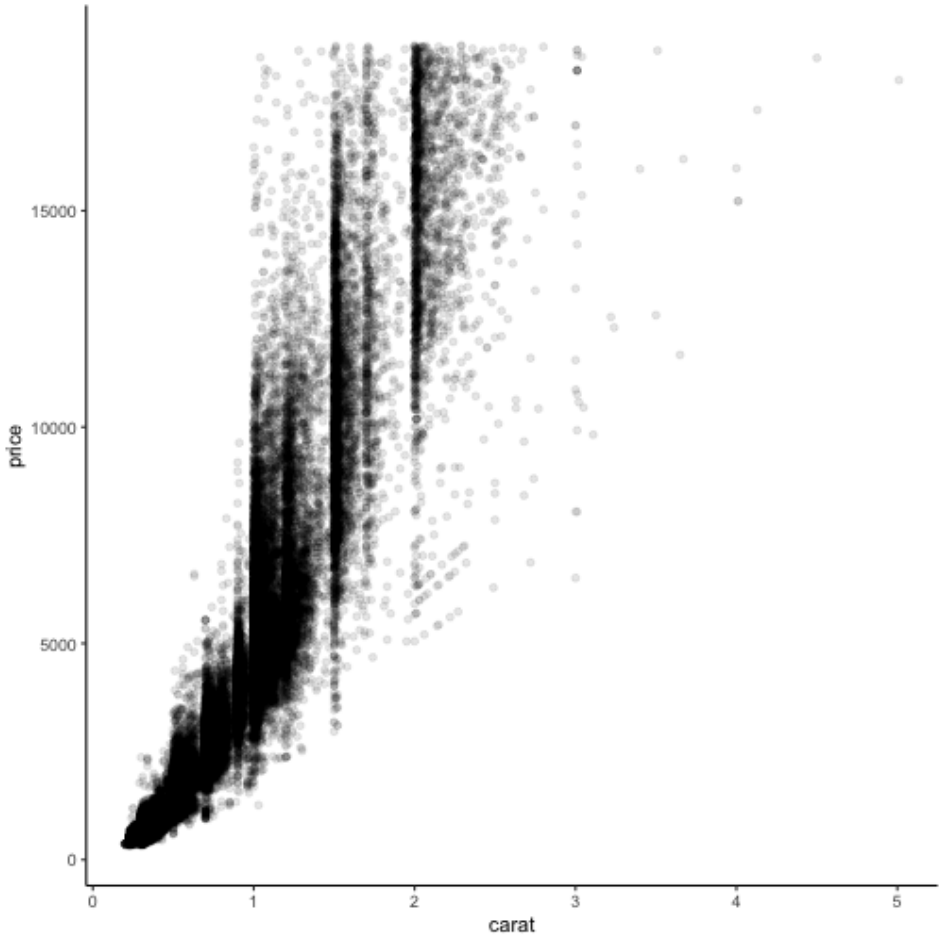
```
base_plot = ggplot(data = diamonds, aes(y =
    geom_point()

base_plot +
  {{theme_classic()}}
```



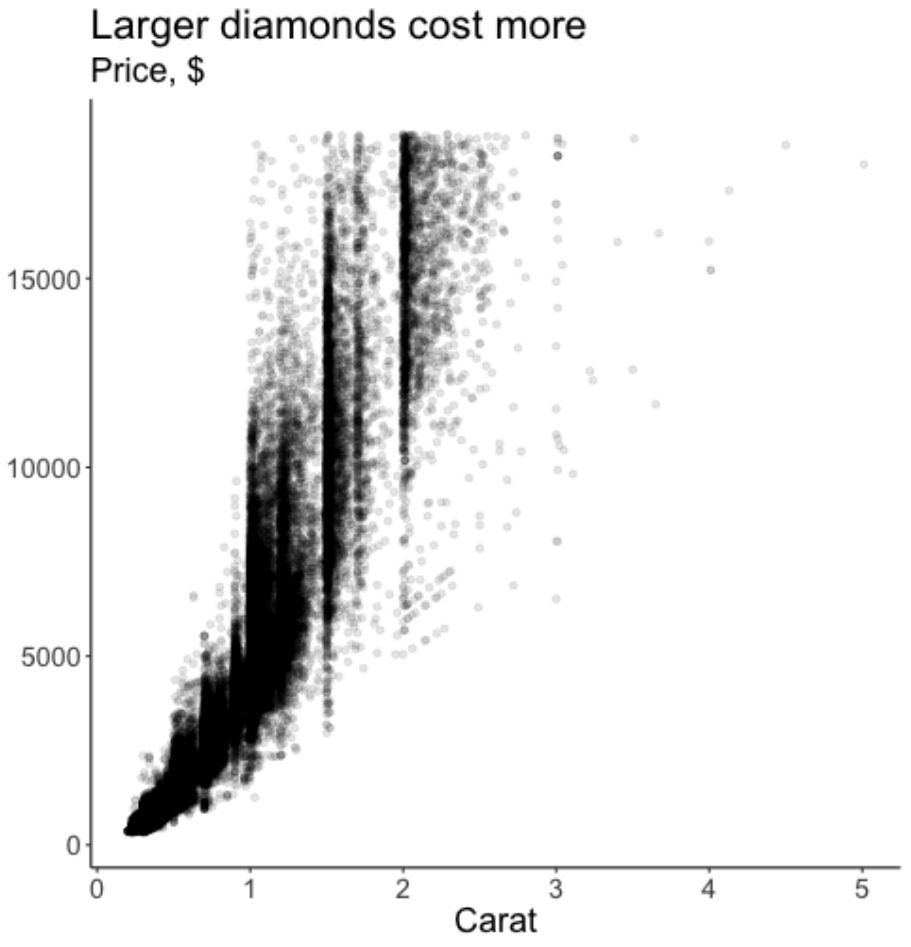
Graphing

```
ggplot(data = diamonds, aes(y = price, x = 
    geom_point(alpha = .1) +
    theme_classic()
```



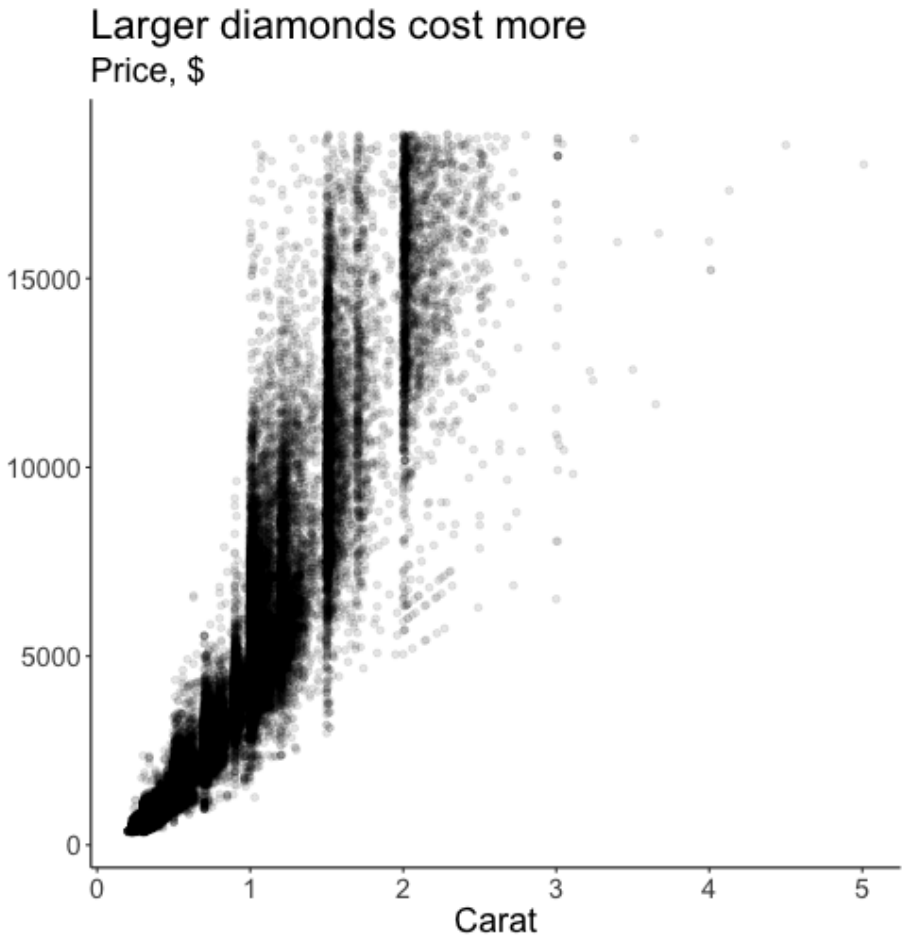
Graphing

```
ggplot(data = diamonds, aes(y = price, x = 
    geom_point(alpha = .1) +
    theme_classic() +
    theme(text = element_text(size = 18)) +
    labs(title = "Larger diamonds cost more
         subtitle = "Price, $",
         y = "",
         x = "Carat")
```



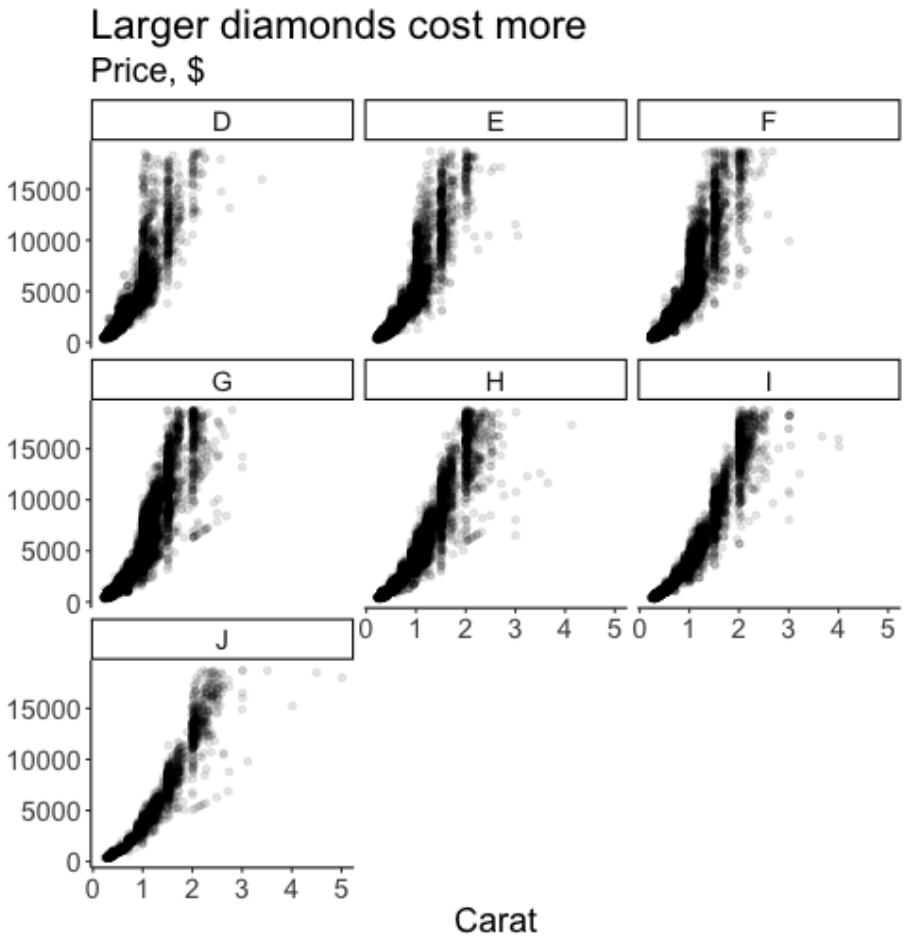
Graphing

```
ggplot(data = diamonds, aes(y = price, x = 
    geom_point(alpha = .1) +
    theme_classic() +
    theme(text = element_text(size = 18)) +
    labs(title = "Larger diamonds cost more
         subtitle = "Price, $",
         y = "",
         x = "Carat")
```



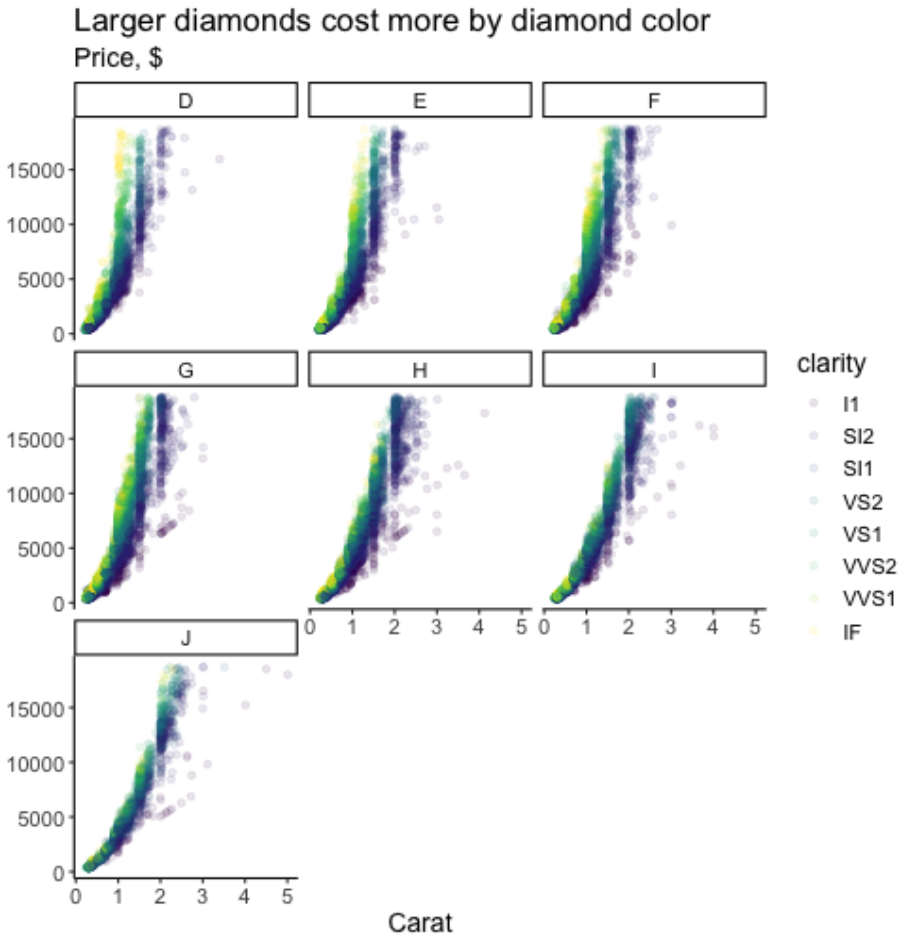
Graphing

```
ggplot(data = diamonds, aes(y = price, x =
    geom_point(alpha = .1) +
    facet_wrap(~color) +
    theme_classic() +
    theme(text = element_text(size = 18)) +
    labs(title = "Larger diamonds cost more
         subtitle = "Price, $",
         y = "",
         x = "Carat")
```



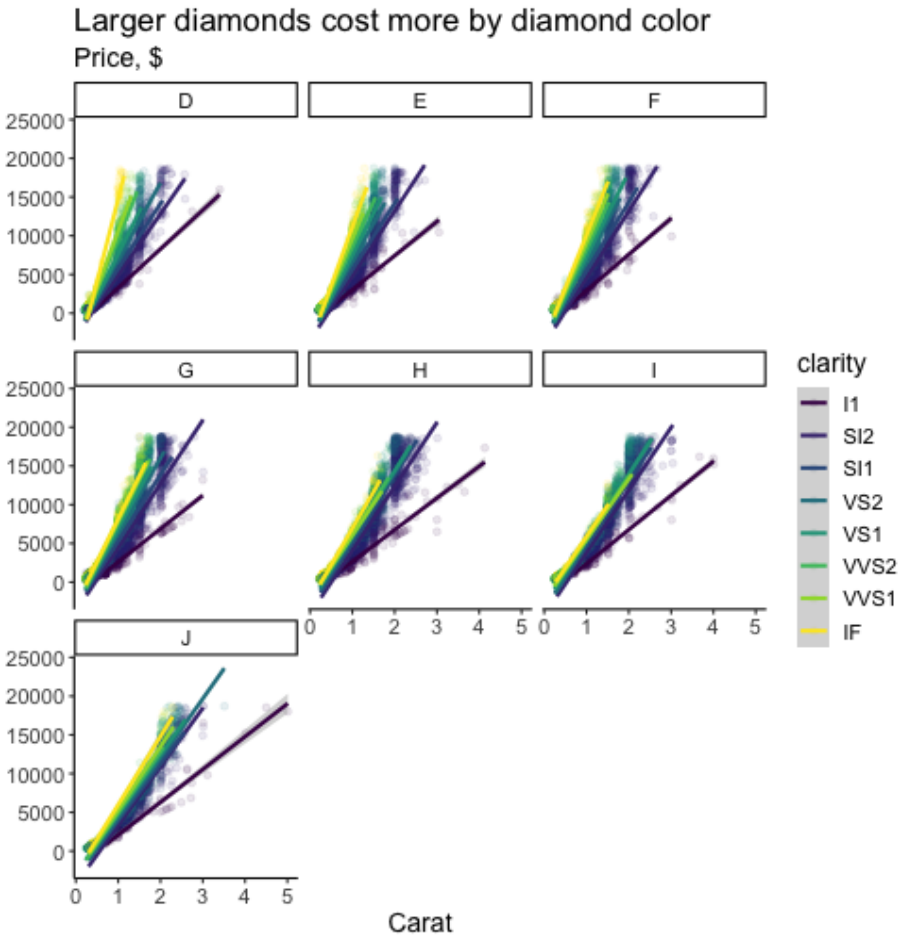
Graphing

```
ggplot(data = diamonds, aes(y = price, x =
    geom_point(alpha = .1) +
    facet_wrap(~color) +
    theme_classic() +
    theme(text = element_text(size = 14)) +
    labs(title = "Larger diamonds cost more
         subtitle = "Price, $",
         y = "",
         x = "Carat")
```



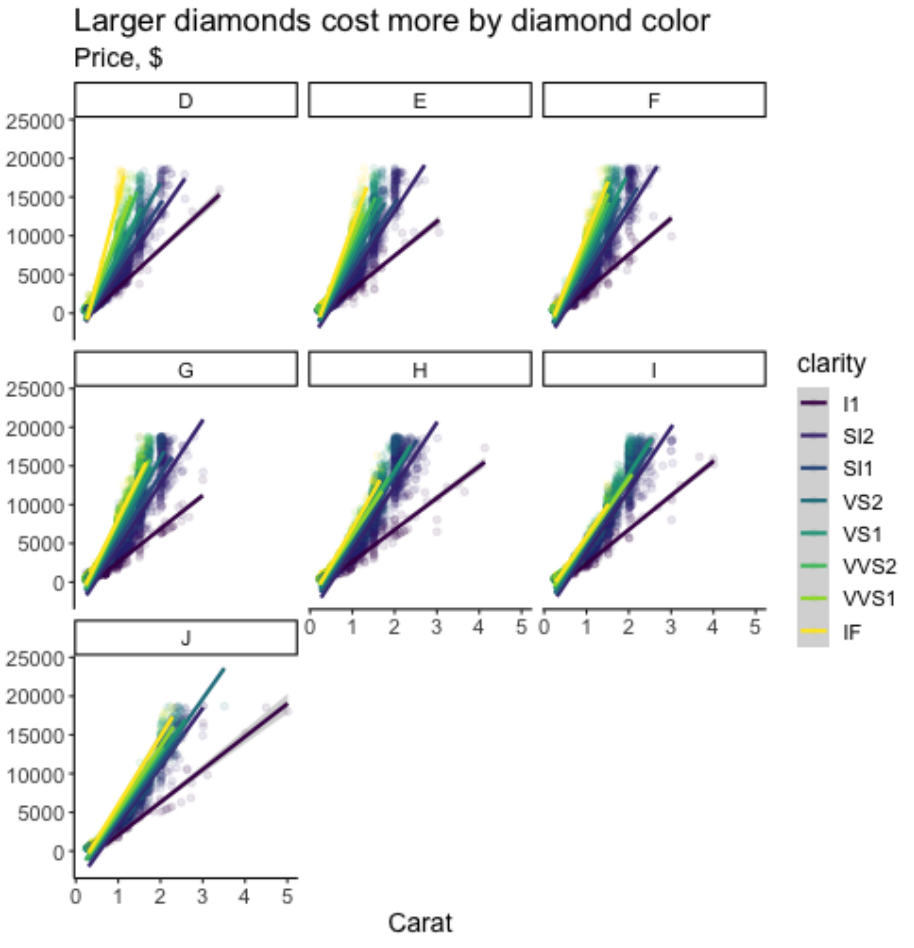
Graphing

```
ggplot(data = diamonds, aes(y = price, x = 
    geom_point(alpha = .1) +
    facet_wrap(~color) +
    geom_smooth(method = "lm") +
    theme_classic() +
    theme(text = element_text(size = 14)) +
    labs(title = "Larger diamonds cost more
          subtitle = "Price, $",
          y = "",
          x = "Carat")
```



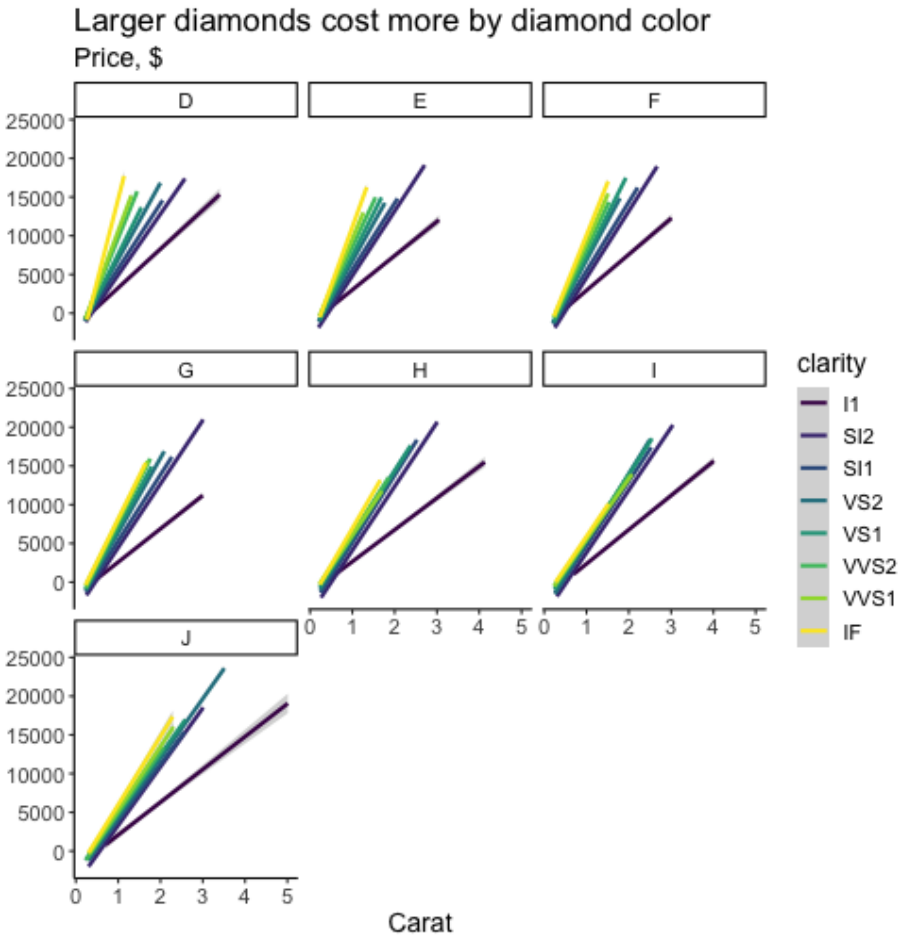
Graphing

```
ggplot(data = diamonds, aes(y = price, x = 
    geom_point(alpha = .1) +
    facet_wrap(~color) +
    geom_smooth(method = "lm") +
    theme_classic() +
    theme(text = element_text(size = 14)) +
    labs(title = "Larger diamonds cost more
         subtitle = "Price, $",
         y = "",
         x = "Carat")
```



Graphing

```
ggplot(data = diamonds, aes(y = price, x = 
    facet_wrap(~color) +
    geom_smooth(method = "lm") +
    theme_classic() +
    theme(text = element_text(size = 14)) +
    labs(title = "Larger diamonds cost more
         subtitle = "Price, $",
         y = "",
         x = "Carat")
```



Let's switch over to an R-Markdown document

Also look. Some latex

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$