Lab 6

Stats 32: Introduction to R for Undergraduates

Harrison Li

04/18/2024

Note: The content of this lab is partially borrowed from Kenneth Tay's course materials in the Autumn 2019 iteration of this course.

Let's return to the diamonds tibble from last class.

```
library(tidyverse)
```

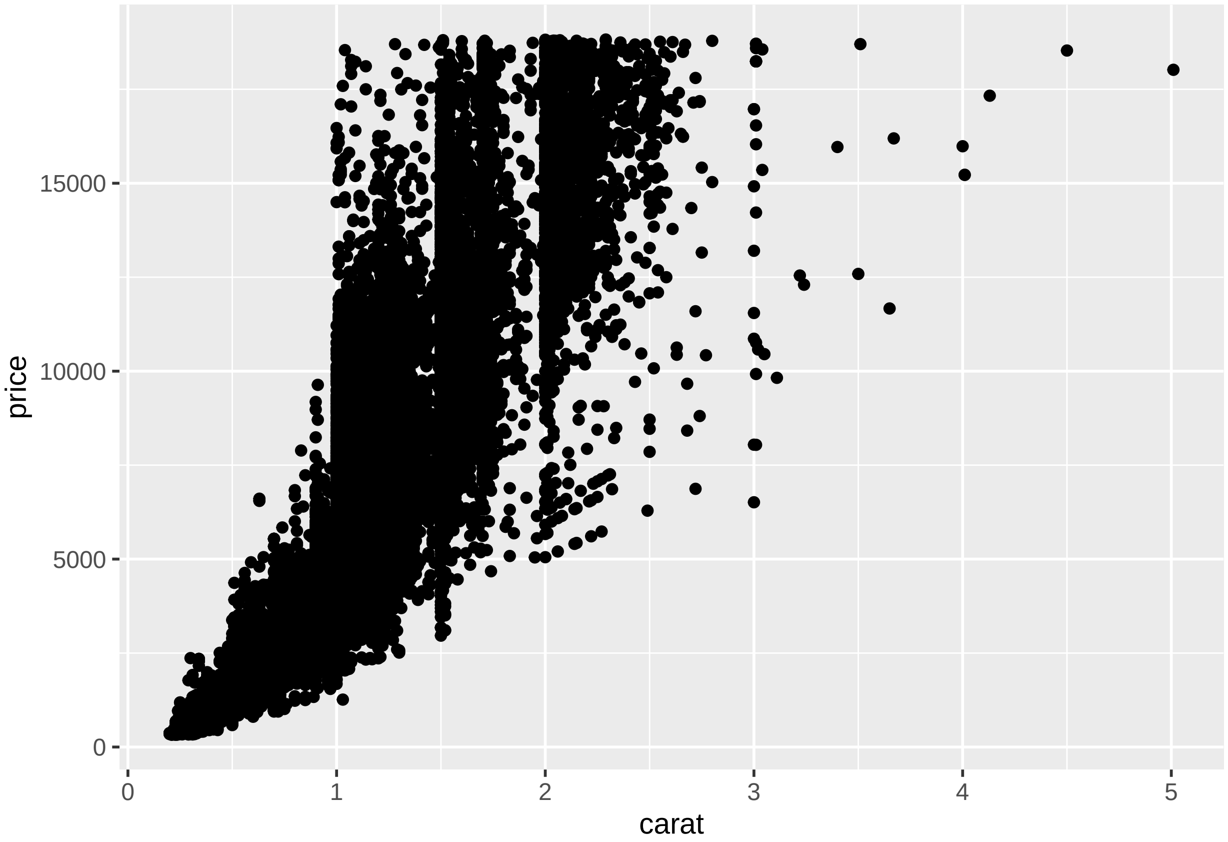
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

2-D visualizations

Since price is an important variable to consider when buying a diamond, we want to understand which characteristics of a diamond affect it and how.

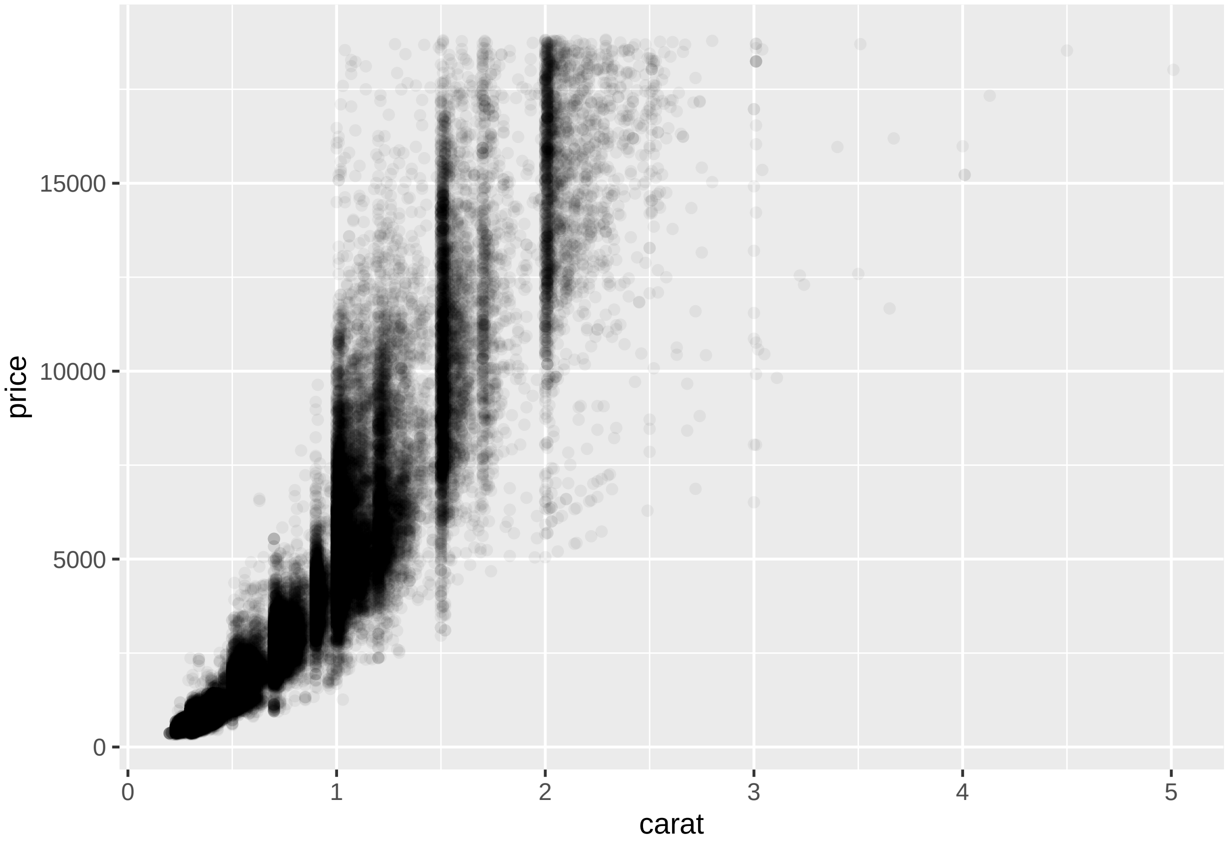
A first guess would be that the weight of a diamond, indicated by `carat`, would heavily influence price. Carat and price are both quantitative, so let's make a scatterplot of price vs. carat, using `geom_point()`:

```
diamonds %>%
    ggplot(mapping = aes(x = carat, y = price)) +
    geom_point()
```



Wow, what a mess! That's because we have so many data points being plotted over each other (this is called **overplotting**). Are there more diamonds in the 0-1 carat range or the 2-3 carat range? It's hard to tell. One way to address this is to modify the transparency of each point by adjusting "alpha". By default, `alpha = 1`, which represents being fully opaque. We can reduce alpha (`alpha = 0.05` means that 20 points are needed to get full opacity):

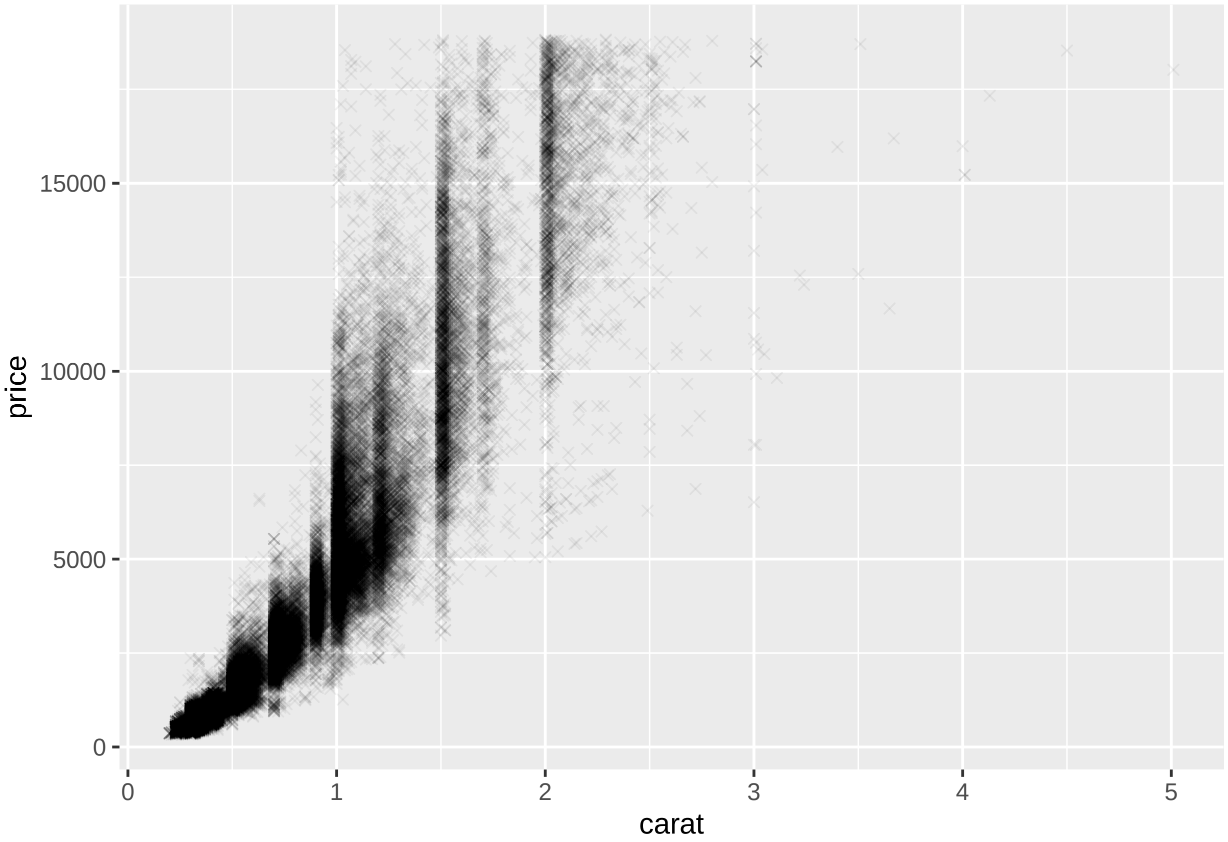
```
diamonds %>%
    ggplot() +
    geom_point(mapping = aes(x = carat, y = price), alpha = 0.05)
```



There's still a fair bit of overplotting going on, but some characteristics of the data become more obvious. For example, the carat size of diamonds seem to bunch up around certain values (e.g. just above 1, 1.5, 2). This may be worth investigating.

Instead of filled circles, we could change the shape of the points manually through the `shape` argument (see this reference for which symbols correspond to each `shape` value):

```
diamonds %>%
    ggplot() +
    geom_point(mapping = aes(x = carat, y = price), alpha = 0.05, shape = 4)
```

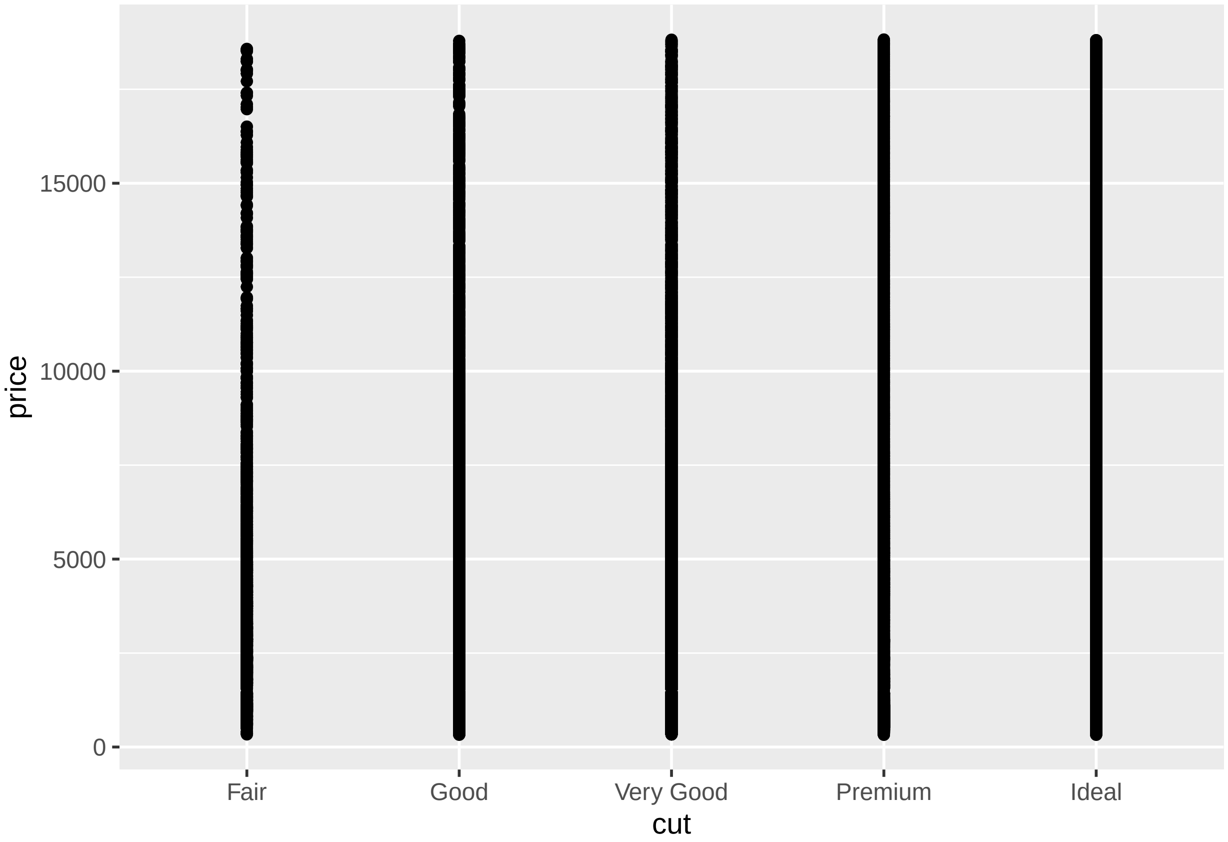


It’s debatable that changing the shape helped make the plot clearer.

Regardless, it’s clear that the heavier the diamond, the more expensive it is. At the same time, we see quite a wide spread of prices for diamonds of the same weight, indicating that there are probably other factors at play.

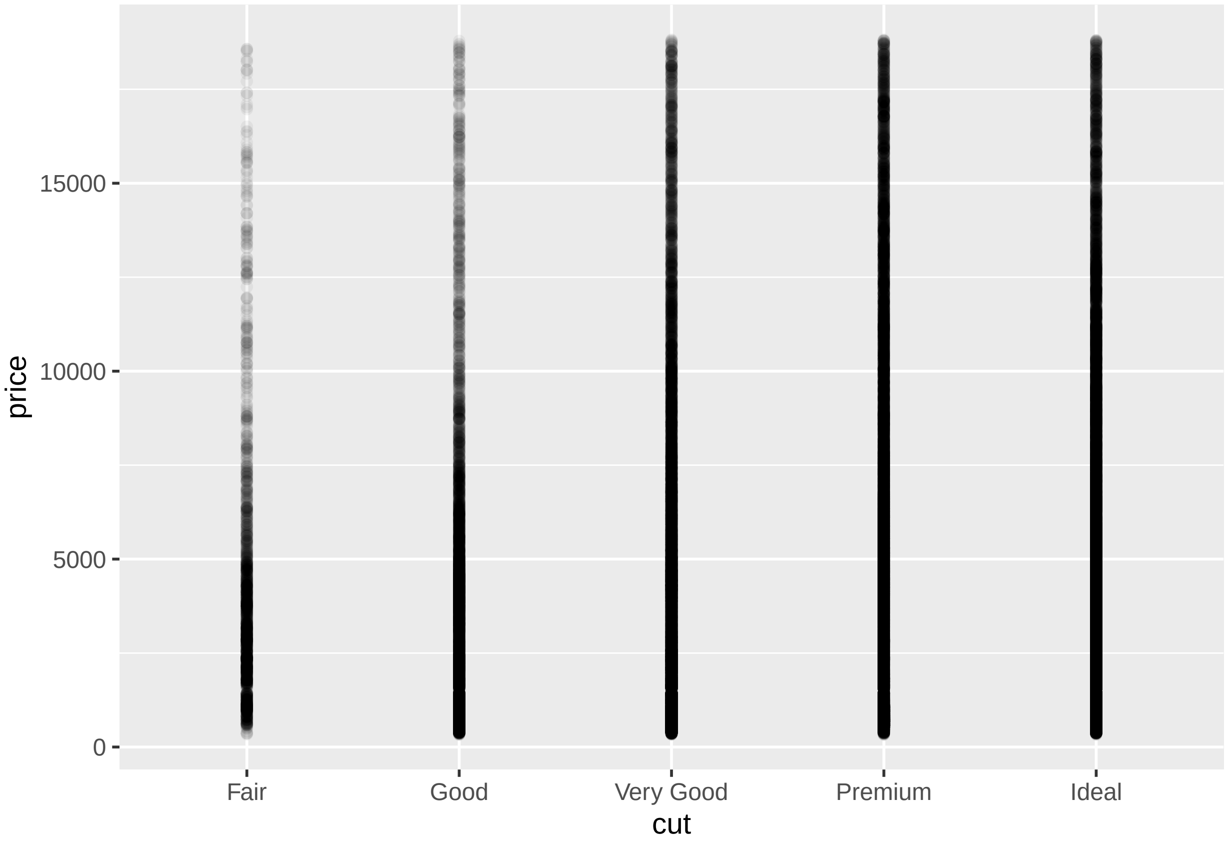
Looking at the dataset, we might guess that cut might be an important factor determining the price of a diamond as well. Let’s try a scatterplot:

```
diamonds %>%
  ggplot(diamonds, mapping = aes(x = cut, y = price)) +
  geom_point()
```



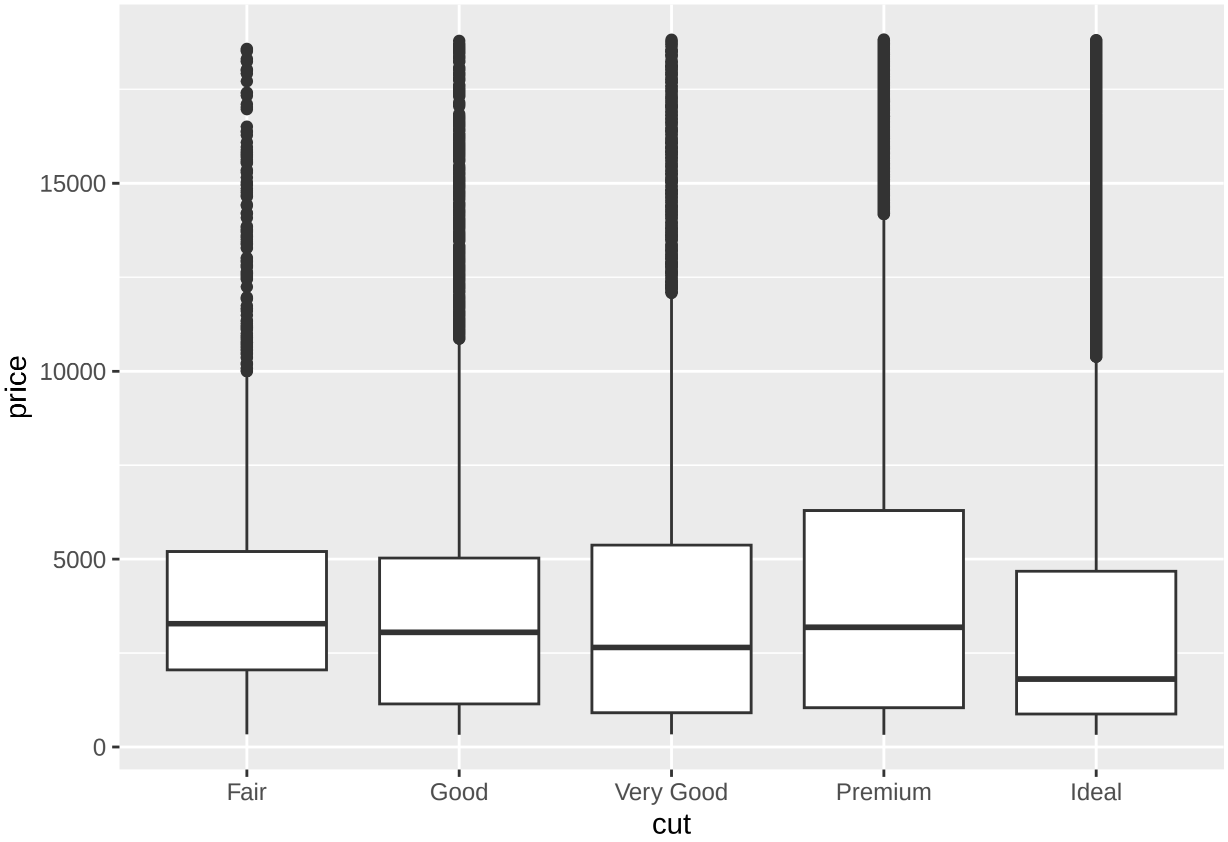
That's not informative at all! We again see a lot of overplotting going on (due to the sheer number of points). Let's use the alpha trick that we used previously:

```
diamonds %>%
  ggplot(mapping = aes(x = cut, y = price)) +
  geom_point(alpha = 0.05)
```



Still not great. As discussed in class, since `cut` is categorical, a better thing to do is multiple (side-by-side) boxplots:

```
diamonds %>%
  ggplot(aes(x = cut, y = price)) +
  geom_boxplot()
```



Interesting! The bulk of the distribution of prices is roughly the same, no matter what the cut is. In fact, from the boxplots, it seems the ideal cut diamonds tend to have the lowest prices!

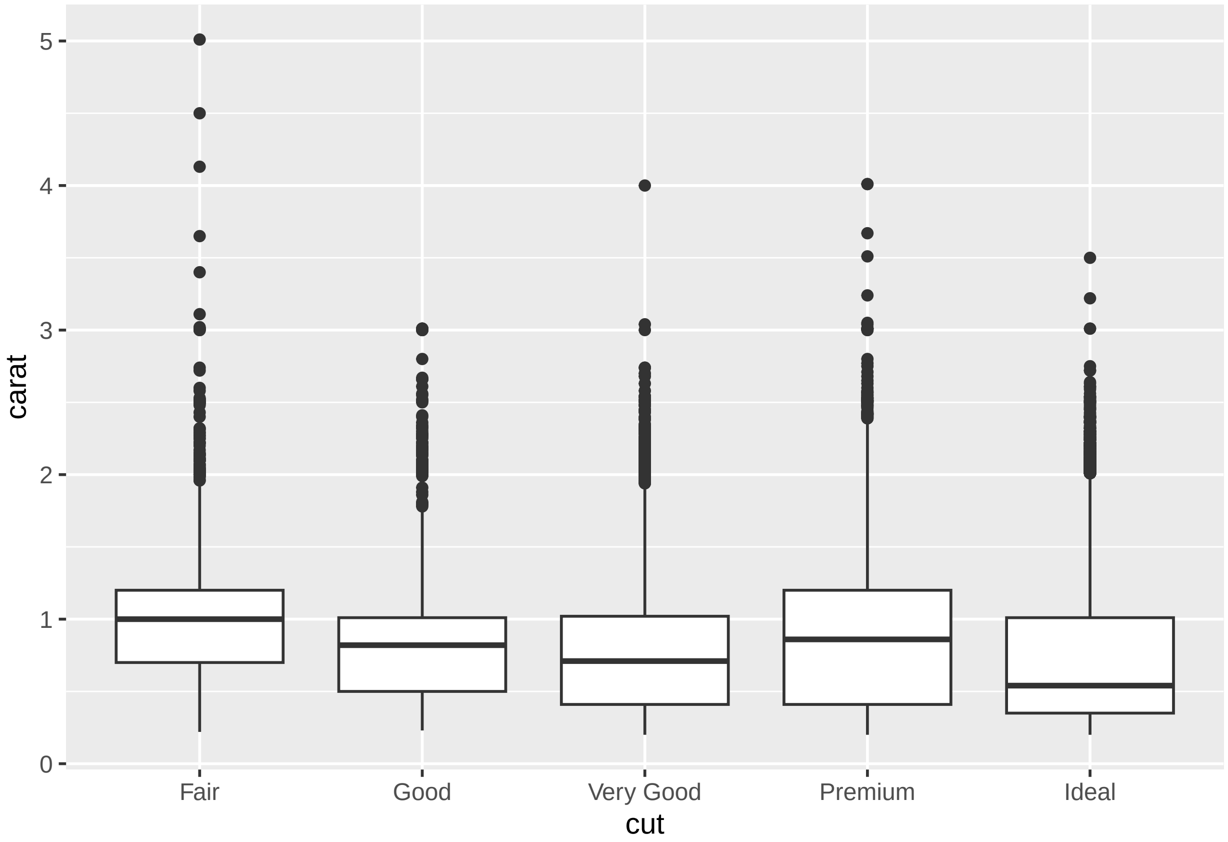
1. Make an appropriate visualization to show the distribution of prices broken down by the `color` of each diamond.

3 or more variables

It seems unintuitive that diamonds of ideal cut have lower prices. Could other variables in the data explain this? One possibility is that there just aren't many large diamonds of ideal cut: thus, a diamond of ideal cut tends to weigh less (smaller in carat size), and hence fetches a lower price.

For this, we can use side-by-side boxplots to look at how the distribution of size in carats varies by cut:

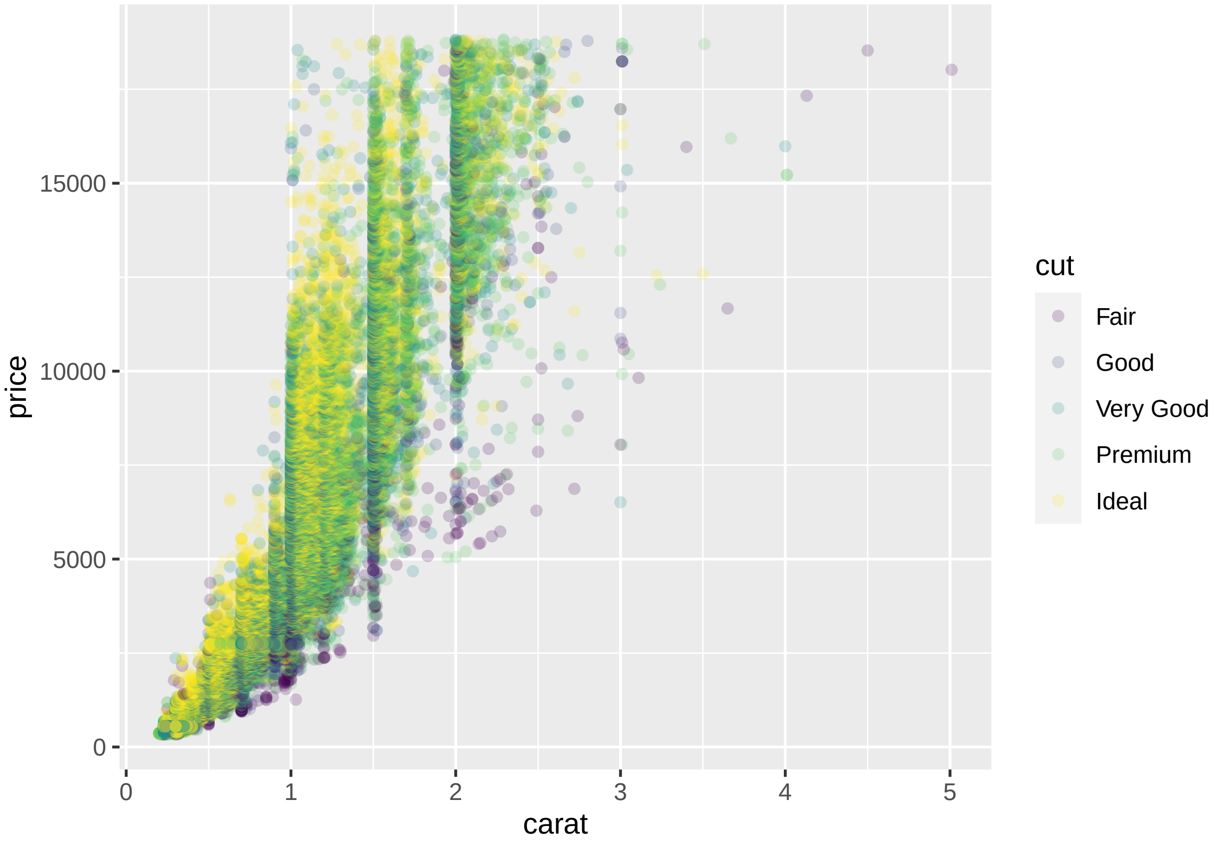
```
diamonds %>%
  ggplot(aes(x=cut, y=carat)) +
  geom_boxplot()
```



It does seem that the better cut diamonds tend to be a bit lighter (have smaller carat weights).

So now we instead try to understand how cut influences price, keeping carat weight fixed. We can do this by modifying our original scatterplot of price versus carat weight, but having the color of the dots indicate cut.

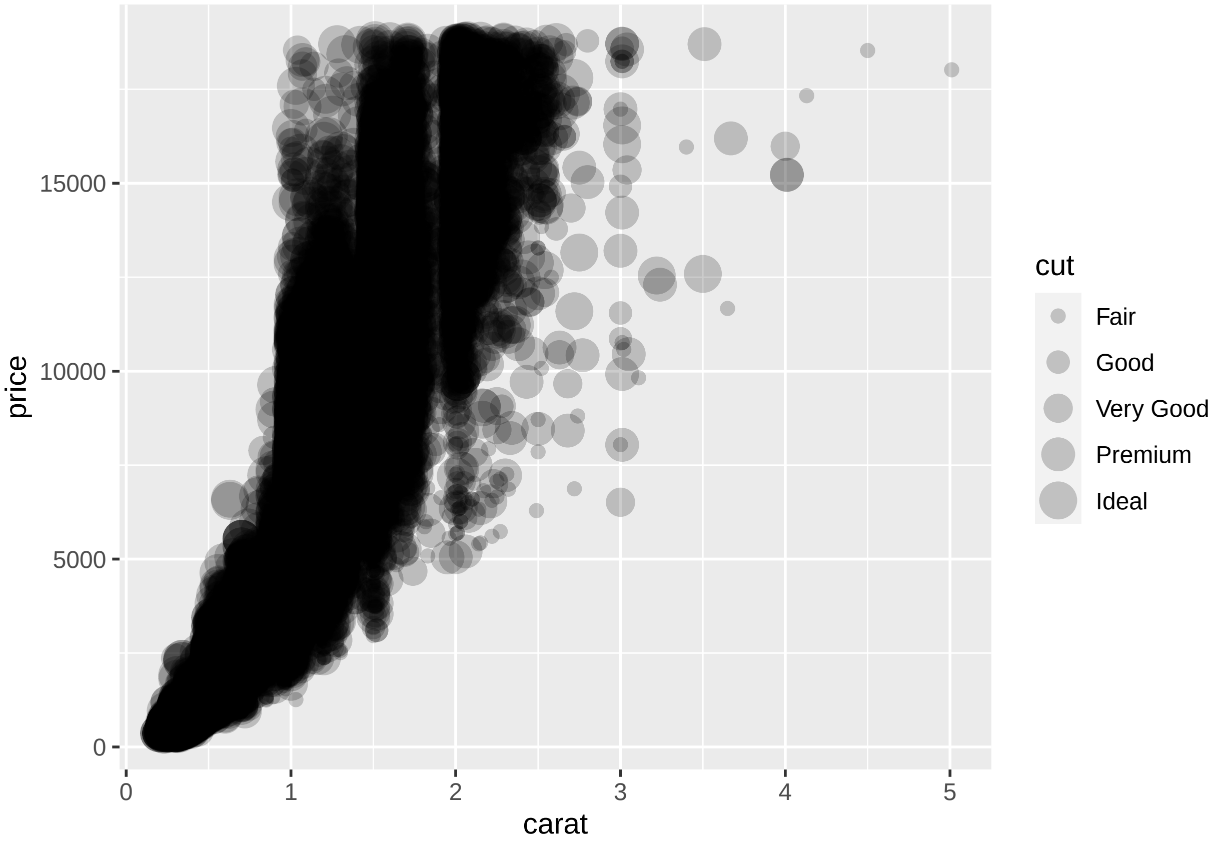
```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price, col = cut)) +
  geom_point(alpha = 0.2)
```



There seem to be clearly more yellow dots on the top side of the plot and more purple dots on the bottom plot, suggesting that within diamonds of a given size, those with a better cut indeed tend to be more expensive.

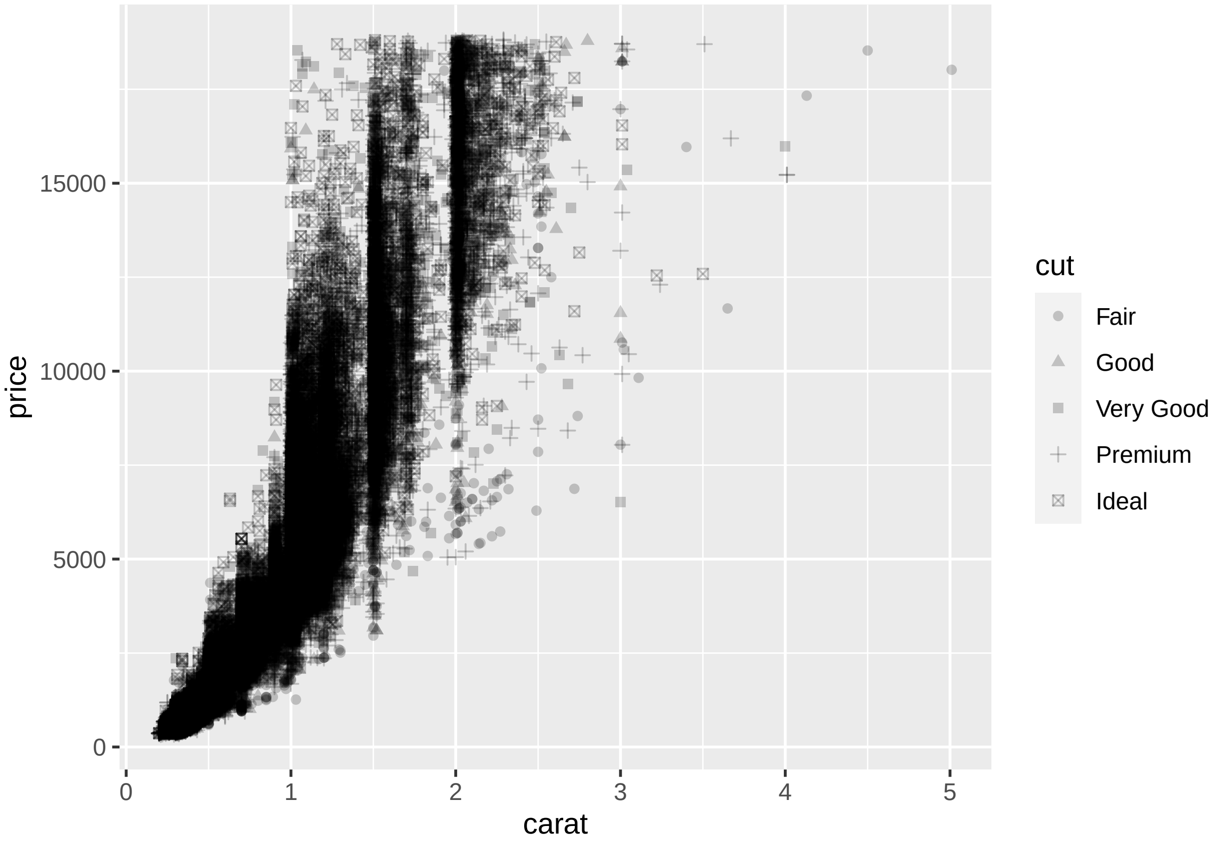
In this case, changing the color of the dots helped us to understand the data better. We could've also changed the transparency or shape, but these end up being less readable. Often it will take some trial and error (and some subjective human judgment) to get the best looking plot!

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price, size = cut)) +
  geom_point(alpha = 0.2)
```



```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price, shape = cut)) +
  geom_point(alpha = 0.2)
```

```
## Warning: Using shapes for an ordinal variable is not advised
```

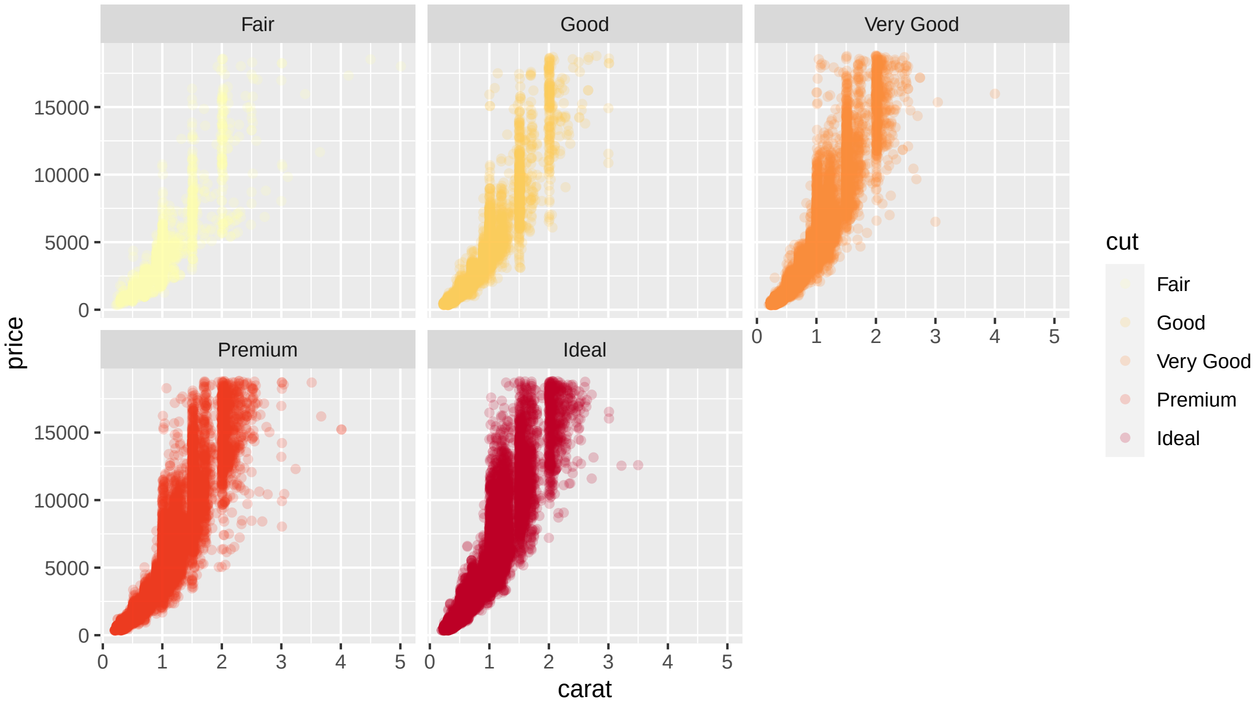


Faceting

There's still a fair amount of overplotting going on in our scatterplot with coloring by `cut`. Can we have separate graphs of price vs. carat for each `cut`?

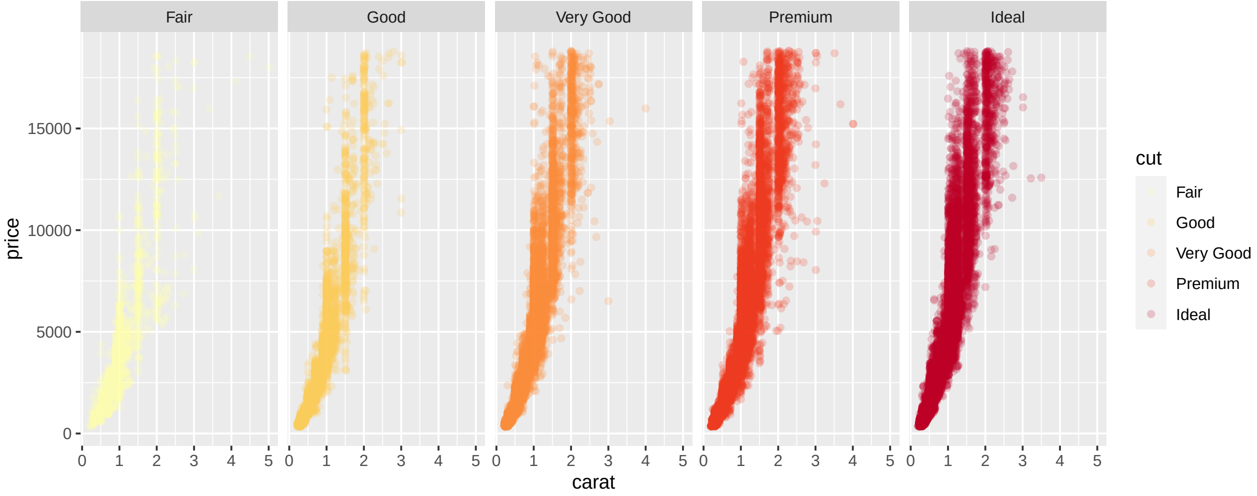
Yes! This is called splitting the plot into **facets**. R allows us to do this by using the function `facet_wrap()`. Use the following code to facet the plot by a single variable:

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price)) +
      geom_point(aes(color = cut), alpha = 0.2) +
      scale_colour_brewer(palette = "YlOrRd") +
      facet_wrap(~ cut)
```



By default, R put just 3 subplots in each row. We can change this by adding a `nrow` argument to `facet_wrap()`:

```
diamonds %>%
  ggplot(mapping=aes(x=carat, y=price)) +
     geom_point(aes(color=cut), alpha = 0.2) +
     scale_colour_brewer(palette = "YlOrRd") +
     facet_wrap(~ cut, nrow = 1)
```

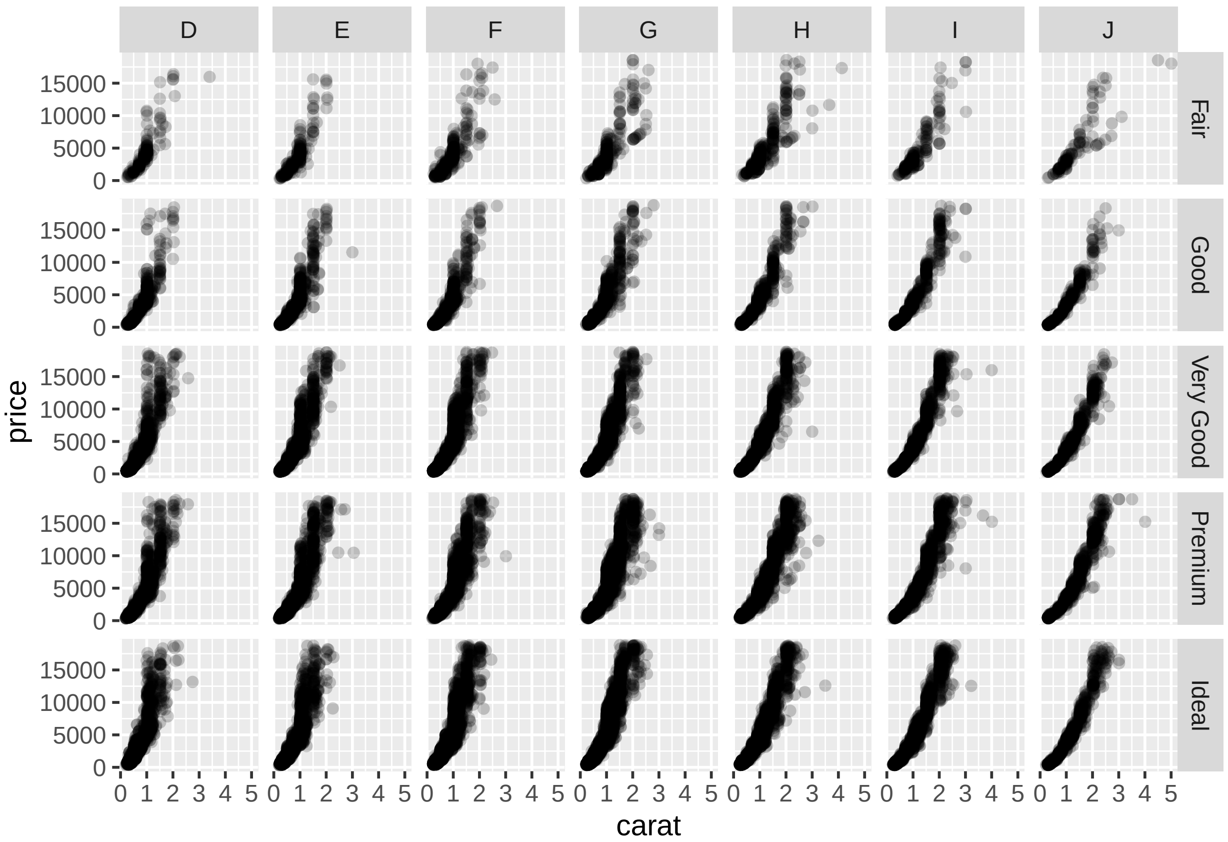


Faceting didn't help too much in this case, since the plots for the better cuts look very similar to one another.

As you can probably see by now, the possibilities are endless!

If we want to facet by more than 1 variable, we can do so with `facet_grid()`. The variable before the ~ sign will be split among the rows, while the variable after the ~ sign will be split among the columns:

```
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
    geom_point(alpha = 0.2) +
    facet_grid(cut ~ color)
```

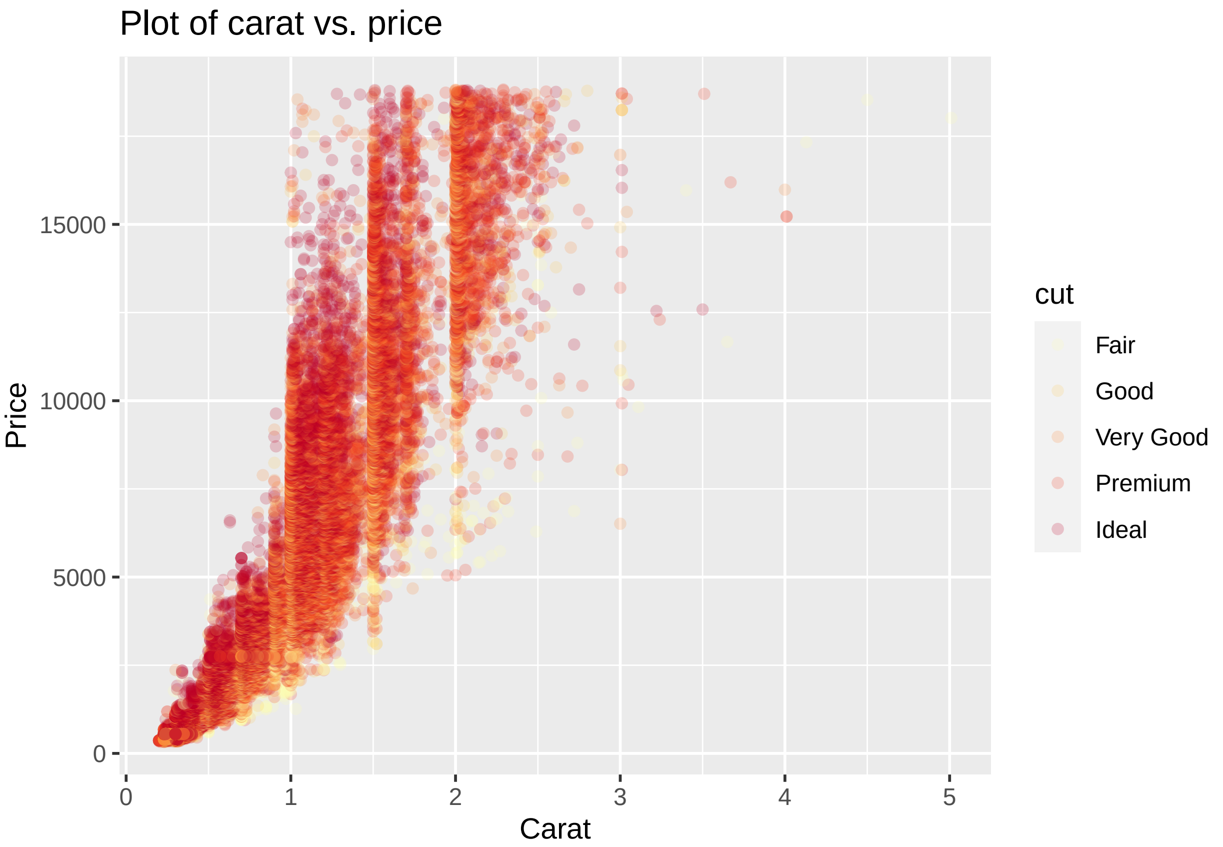


2. Create faceted scatterplots of `price` vs. `carat`, such that in each plot the points are colored by `color` and in each facet you have a different `cut`. Congratulations: You've just visualized 4 variables at a time - 2 categorical, 2 quantitative.

Layers and themes

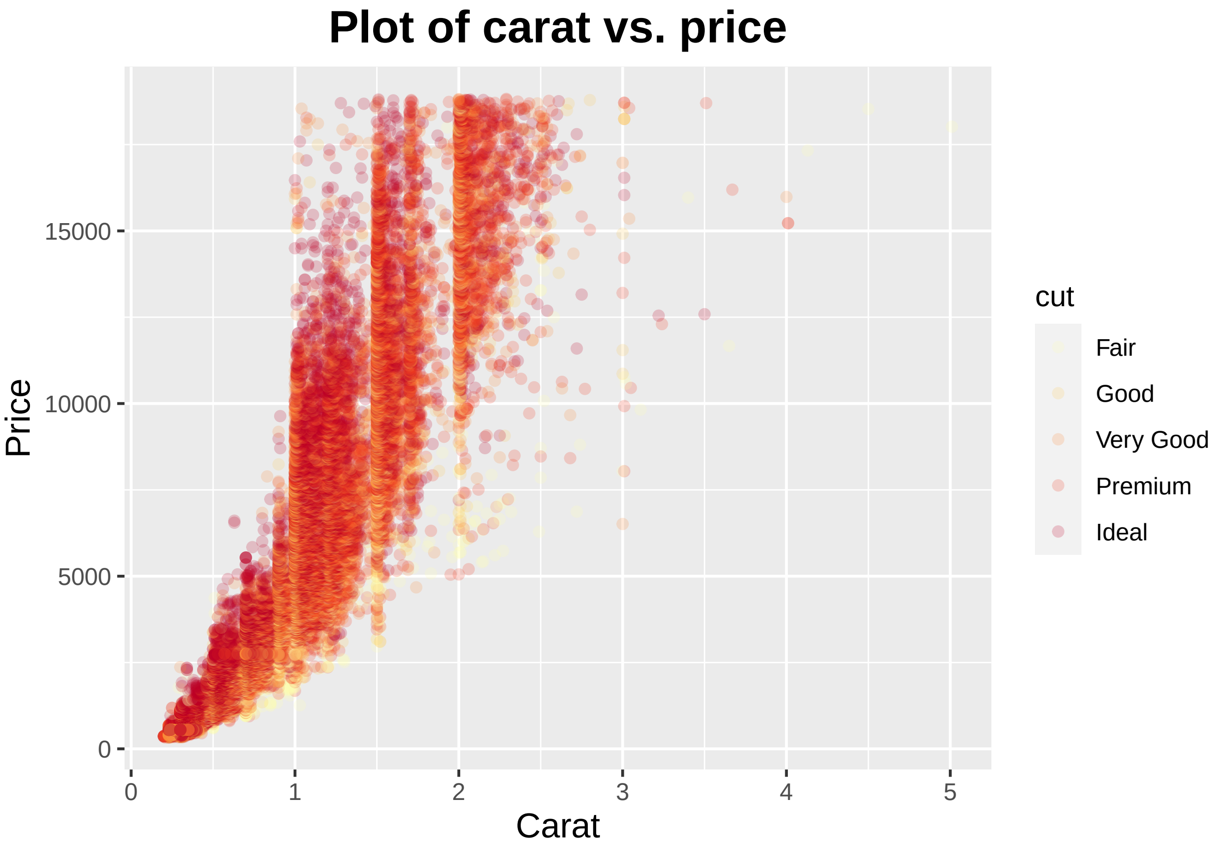
Let's say you're satisfied with the scatterplot of price vs. carat with color denoting cut, and that you want to share it with others. The first thing you should do is label your axes and give your plot a title by adding layers:

```
diamonds %>%
    ggplot(mapping = aes(x = carat, y = price, col = cut)) +
    geom_point(alpha = 0.2) +
    scale_colour_brewer(palette = "YlOrRd") +
    labs(x = "Carat", y = "Price", title = "Plot of carat vs. price")
```



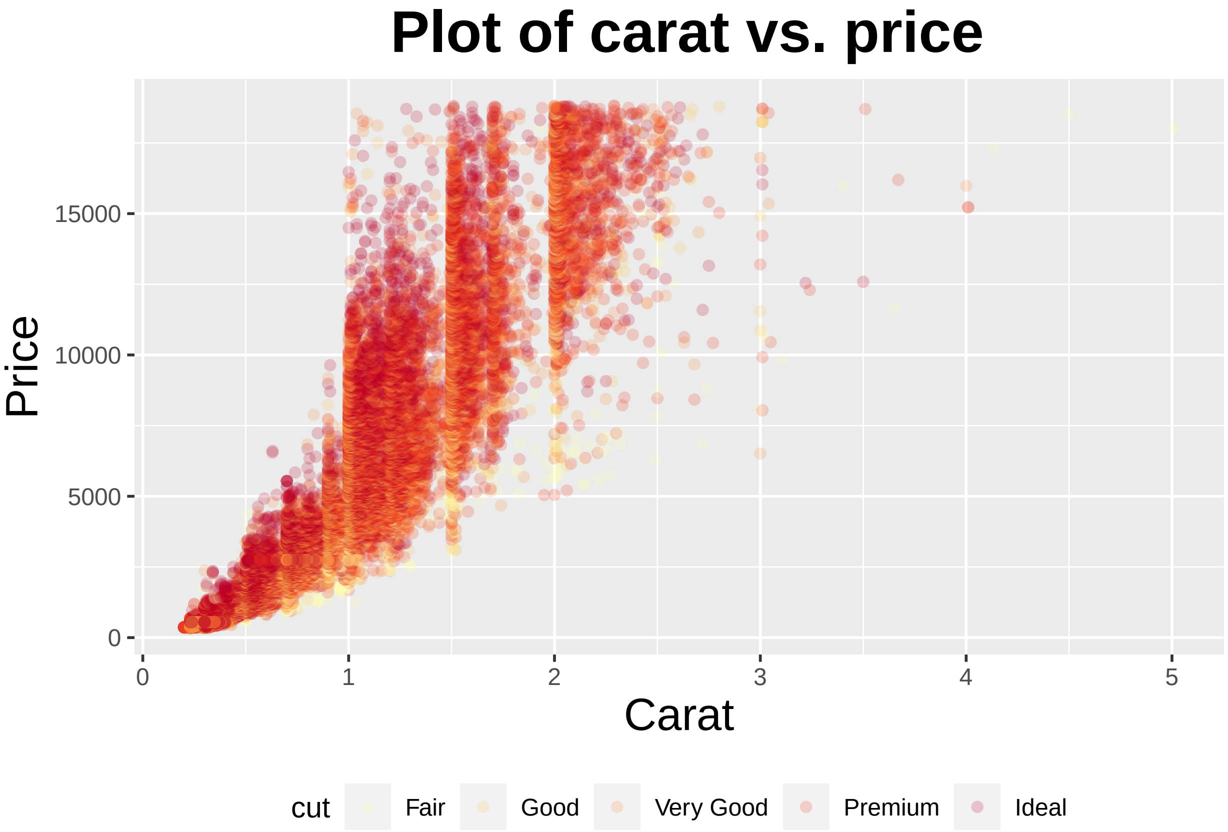
The size of the labels seems a bit small. We can adjust them using the `theme()` function. Let's centralize the plot title at the same time:

```
diamonds %>%
  ggplot(mapping = aes(x = carat, y = price, col = cut)) +
  geom_point(alpha = 0.2) +
  scale_colour_brewer(palette = "YlOrRd") +
  labs(x = "Carat", y = "Price", title = "Plot of carat vs. price") +
  theme(plot.title = element_text(size = rel(1.5), face = "bold", hjust = 0.5),
        axis.title = element_text(size = rel(1.2)))
```



We can move the legend around by setting a `legend.position` argument in `theme()` (possible options are “none”, “left”, “right”, “bottom”, “top”):

```
ggplot(data = diamonds, mapping = aes(x = carat, y = price, col = cut)) +
    geom_point(alpha = 0.2) +
    scale_colour_brewer(palette = "YlOrRd") +
    labs(x = "Carat", y = "Price", title = " Plot of carat vs. price") +
    theme(plot.title = element_text(size = rel(2), face = "bold", hjust = 0.5),
          axis.title = element_text(size = rel(1.5)),
          legend.position = "bottom")
```



For a full (long!) list of attributes which can be modified, see this reference. Don't try to memorize these. Just refer to them (or Stack Overflow) when you want to do something.

3. Add a title and better axis labels to your plot from question 1. Change the vertical axis so that the tick labels are at 0, 2000, 4000,. . . ,20000 instead of at 0, 5000,. . . Hint: look at the help page for `scale_y_continuous()`‘.

To save a plot, run the code generating that plot in the console (just clicking the run chunk button won't work), and then click on the Export button in the bottom right portion of the RStudio IDE, and click "Save as Image. . . " (or "Save as PDF. . . ") You can adjust the size of your image in the pop-up before saving it.

4. Save your plot for problem 3 in a directory where you won't lose it (no need to show any code for this part).