Session II Recap

N.Paterno

7/9/2021

Solutions to 3.2.4 Exercises

Here is a recap of what we did during our session today. We began with the exercises from section 3.2.4 of R for Data Science. The solutions that we came up with are below with commentary.

1. Run `{ggplot(data = mpg)`. What do you see?

```
ggplot(data = mpg)
```


The resulting blank plot above is from not specifying a variable to be used for x and y in the aesthetics of the plot.

2. How many rows are in `mpg` ? How many columns?

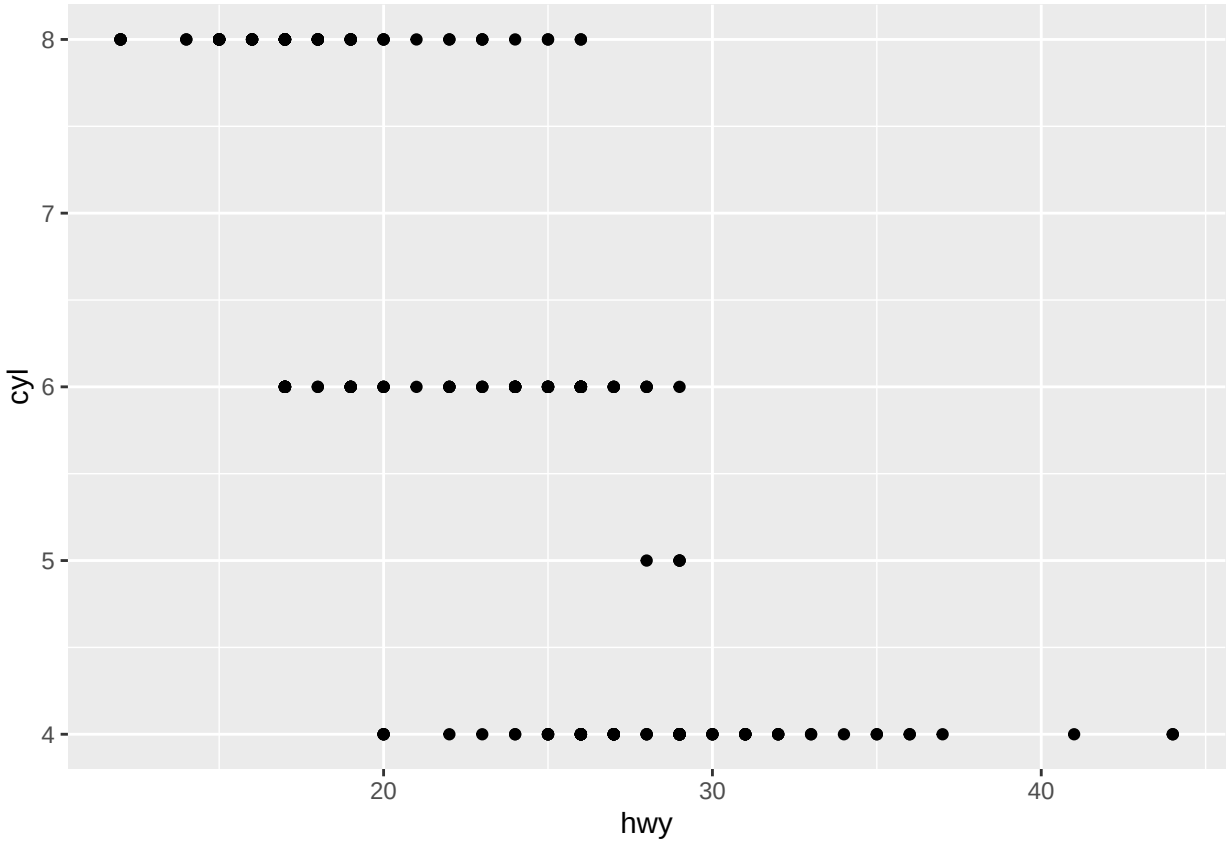
To solve this problem, we used the hint thats listed for problem 3. Typing `{r} ?mpg` into the console will pull up the help file for the data set. Part way down under "format" we see 234 rows and 11 variables (columns).

3. What does the `drv` variable describe?

From the help file, we see that this variable represents the type of drive train with levels of f (front-wheel), r (rear-wheel) and 4 (4wd).

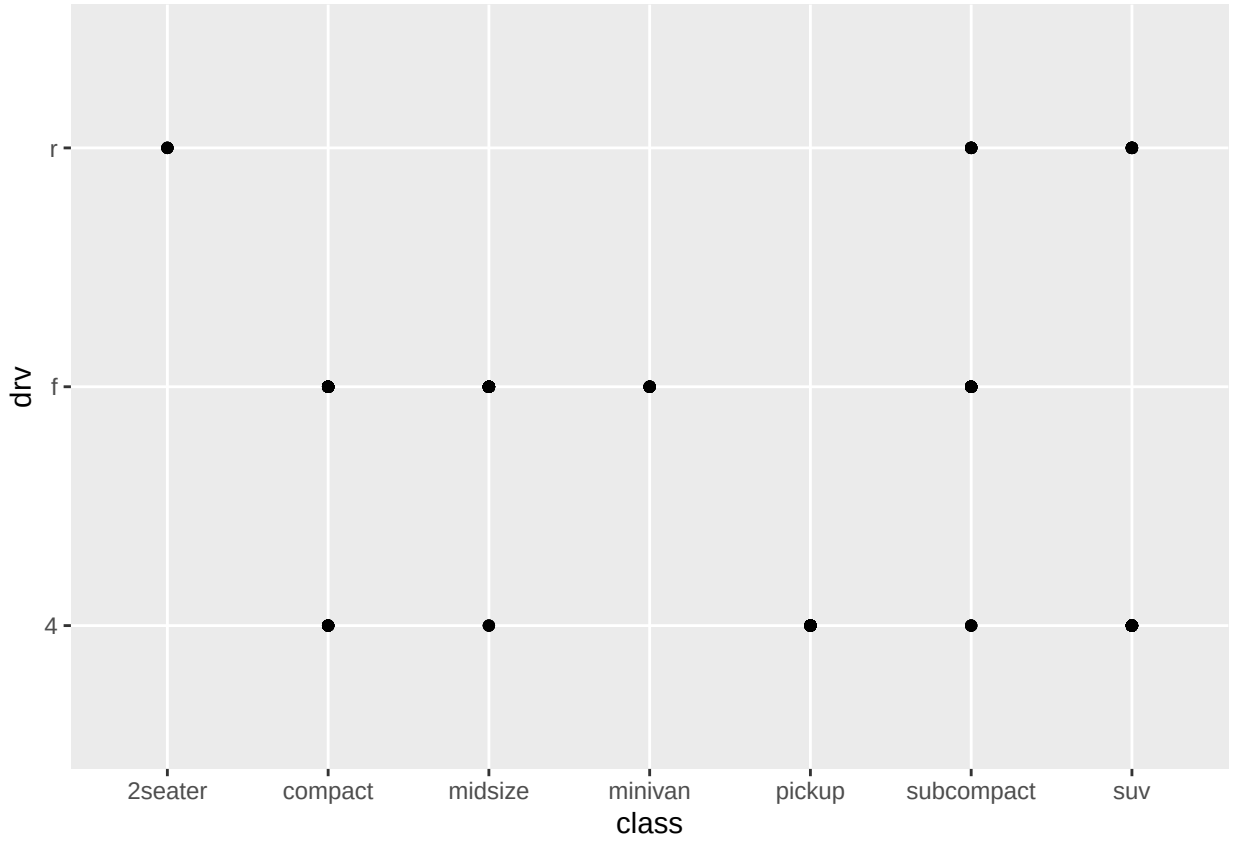
4. Make a scatterplot of `hwy` vs `cyl`.

```
## format:
## ggplot(data = name_of_dataset, aes(x = name_of_variable_1, y = name_of_variable_2)) +
## geom_{specifc_plot_type}()
ggplot(data = mpg, aes(x = hwy, y = cyl))+
  geom_point()
```



5. What happens if you make a scatterplot of `class` vs. `drv`? Why is the plot not useful?

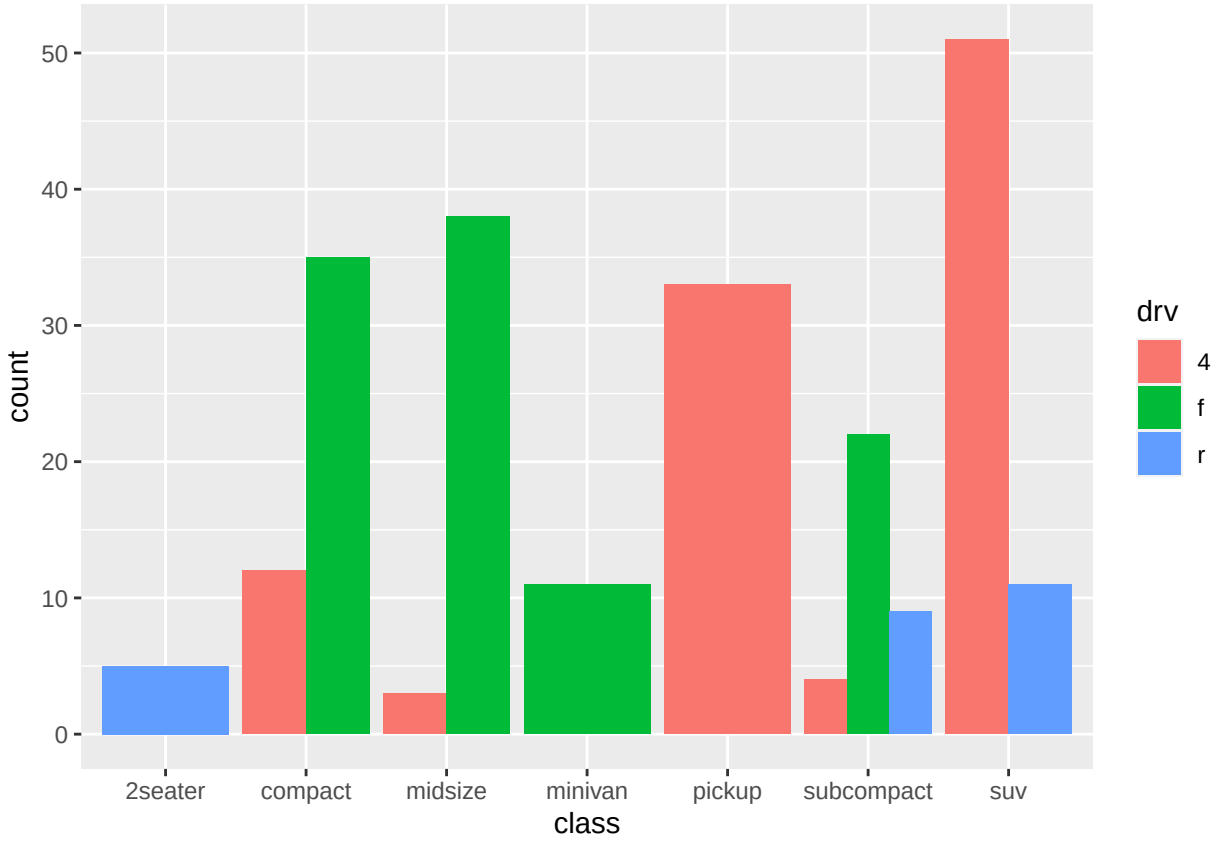
```
ggplot(data = mpg, aes(class, drv))+
  geom_point()
```



The plot above is not useful because it doesn't tell us much. All we can see is a yes/no plot of whether or not a specific class of vehicle has a specific drive train. For example, since `2seater` only has a dot in the `r` row, we know those vehicles only come with rear-wheel drive. We'll make a more informative bar plot below.

```
plot_data <- mpg %>% # creates a new data frame called plot_data
  select(c(class, drv)) # select is a part of {dplyr} it removes all variables except those selected

ggplot(data = plot_data, aes(x = class, fill = drv))+ # the 'fill' aesthetic changes the color of the b
  geom_bar(position = "dodge") # the dodge position places the bars side-by-side for each class instead
```

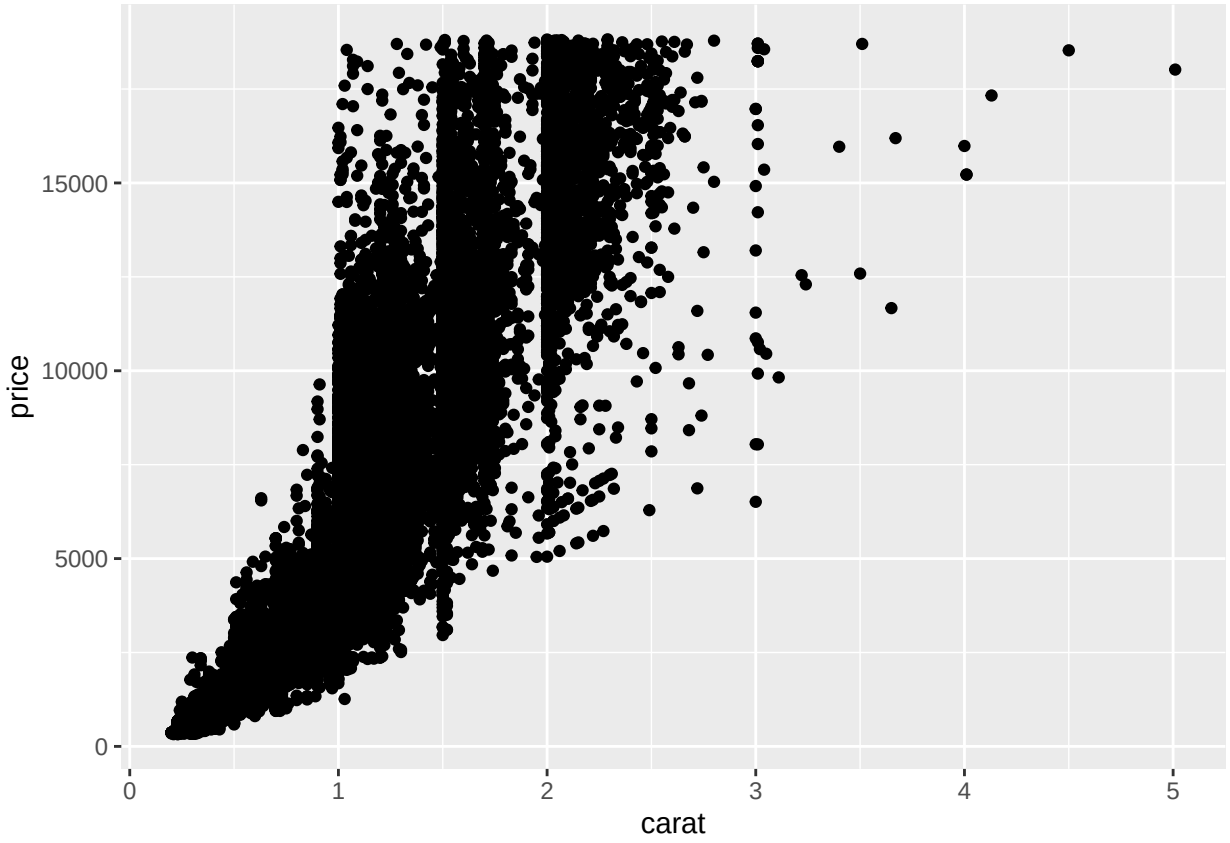


More Aesthetics

After this, we moved to the `diamonds` dataset and looked at some more of the aesthetic changes we can make. Below is a step by step progression of those changes.

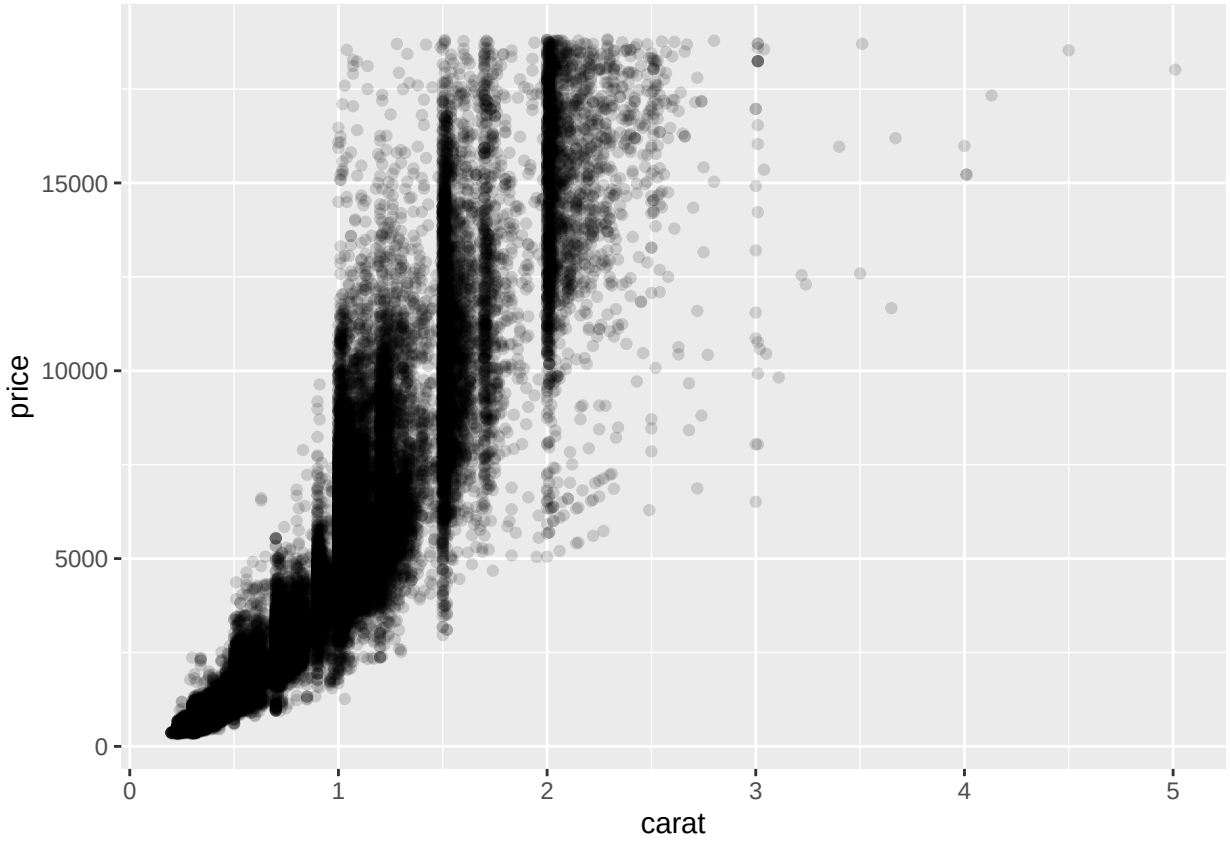
Base Plot

```
ggplot(diamonds, aes(carat, price))+
  geom_point()
```



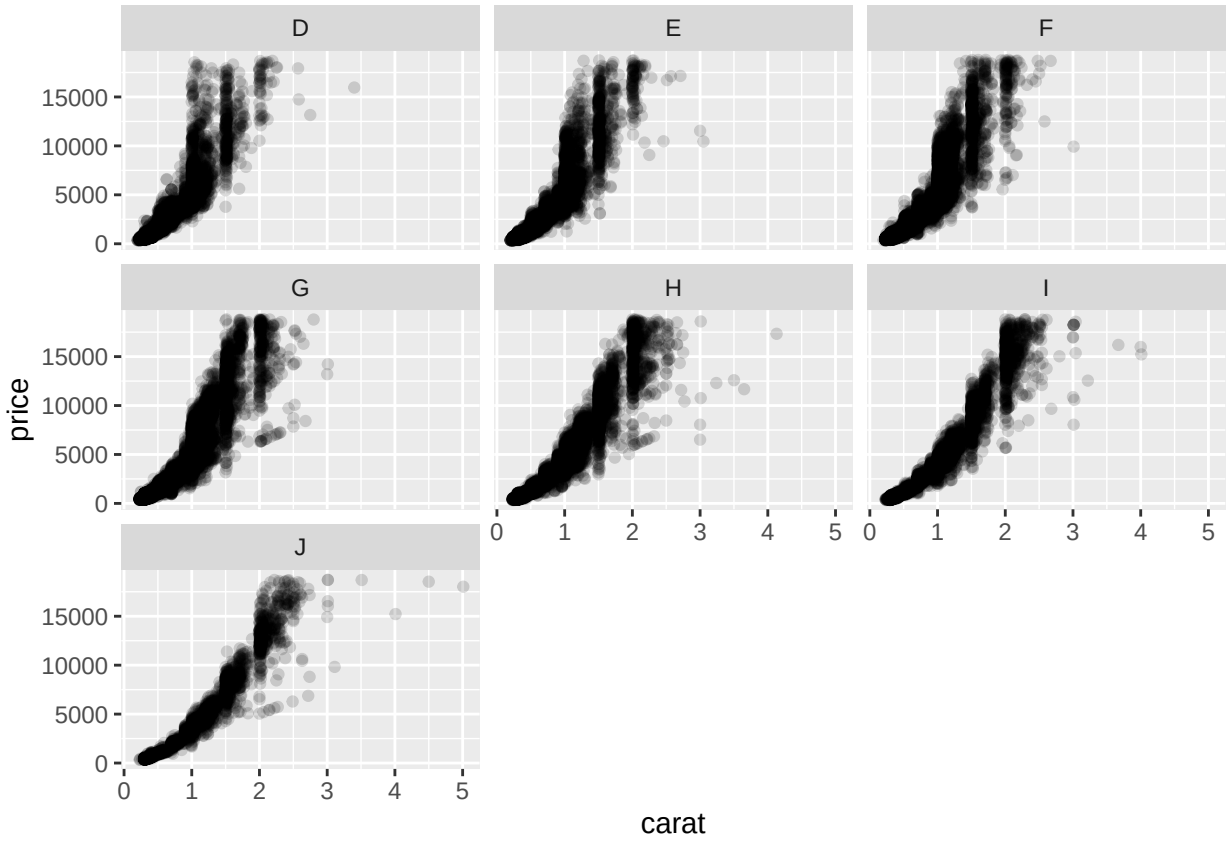
Transparency

```
ggplot(diamonds, aes(carat, price))+
  geom_point(alpha= 0.15) # closer to 0 = clear, closer to 1 = solid
```



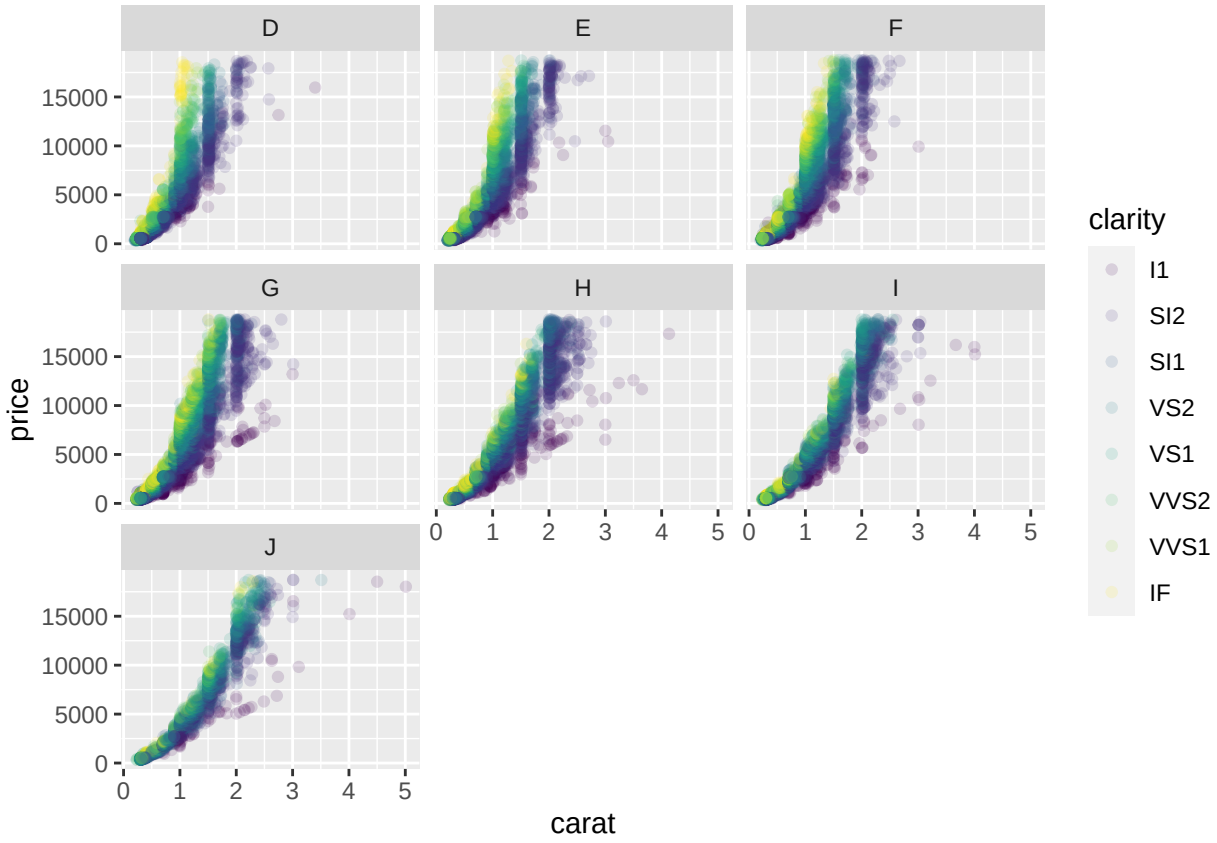
`facet_wrap`: Creating one sub-plot for each diamond color

```
ggplot(diamonds, aes(carat, price))+
  geom_point(alpha = 0.15)+
  facet_wrap(~color)
```



Changing the color of the points

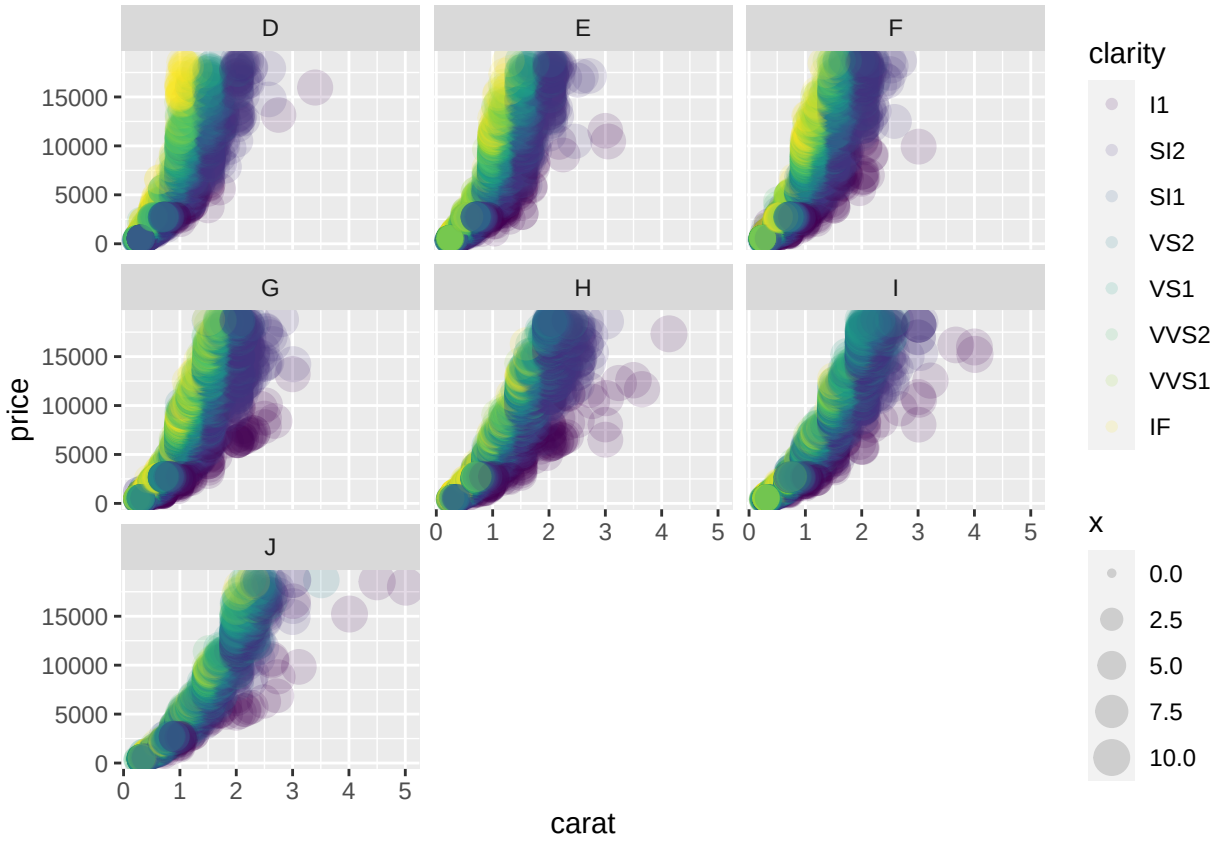
```
ggplot(diamonds, aes(carat, price, color = clarity))+
  geom_point(alpha = 0.15)+
  facet_wrap(~color)
```



Note: we can also specify a color to use with `color = "blue"` in the aesthetics.

Changing the size of the points

```
ggplot(diamonds, aes(carat, price, color = clarity, size = x))+
  geom_point(alpha = 0.15)+
  facet_wrap(~color)
```

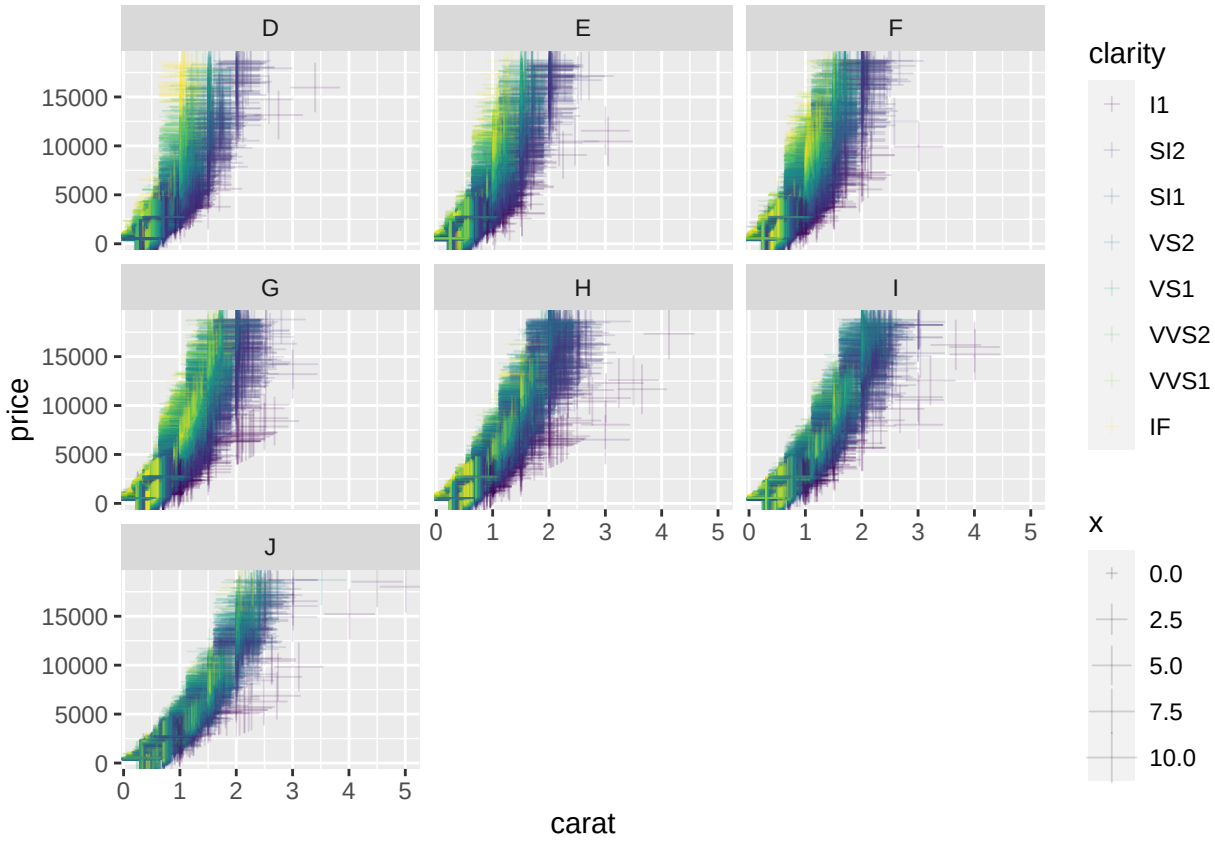


This will make the size of the dots equal to the length of the diamonds in mm. Note we can also specify a scaled size. The standard point is size 1. We can use `size = 2` to double the size of the points.

Changing shape

We didn't actually get to this part. The shape aesthetic can be specified using the key found at this website.

```
ggplot(diamonds, aes(carat, price, color = clarity, size = x))+
  geom_point(alpha = 0.15, shape = 3)+
  facet_wrap(~color)
```


D
E
F
G
H
I
J
price
carat
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
x
0.0
2.5
5.0
7.5
10.0