UPPSALA
UNIVERSITET

Uppsala Child & Baby Lab

Introduction to Data Visualization with `ggplot2`

R for Psychology Research

Marcus Lindskog, Docent
2019-09-23

Overview

A note on Data Viz in R

A note on Data Viz in R

- A lot of the hype around R comes from the possibility to create nice looking graphs.
- There are many systems for making graphs in R.
- `ggplot2` is a very versatile system that will get you far.
- `ggplot2` also uses a coherent system that plays well with the rest of the `tidyverse`.
- A powerful system for Data Viz **cannot** make up for bad choices.
- A copy-paste approach will only get you so far.

Layered Grammar of Graphics

`ggplot2` and the Grammar of Graphics

- `ggplot2` implements the *grammar of graphics (GoG)*

- GoG is a coherent system for describing and building graphs.

- In GoG, a plot is built out of one or more **layers**.

- Each layer consists of one or more components:
  - data and aesthetic mappings.
  - geometric objects
  - scales
  - facet specifications
  - statistical transformations
  - coordinate system.
- `ggplot2` makes it easy to build graphs with these components.

The data of the day

The data of the day

- For the data viz examples that we will work with, we need a data set.
- I will use the `diamonds` data set that is included in the `ggplot2` package.
- An example of the data is shown in the table below.

Show 10 entries Search:

	carat	cut	color	clarity	depth	table	price
1	0.23	Ideal	E	SI2	61.5	55	326
2	0.21	Premium	E	SI1	59.8	61	326
3	0.23	Good	E	VS1	56.9	65	327
4	0.29	Premium	I	VS2	62.4	58	334
5	0.31	Good	J	SI2	63.3	58	335
6	0.24	Very Good	J	VVS2	62.8	57	336

Showing 1 to 6 of 6 entries Previous 1 Next

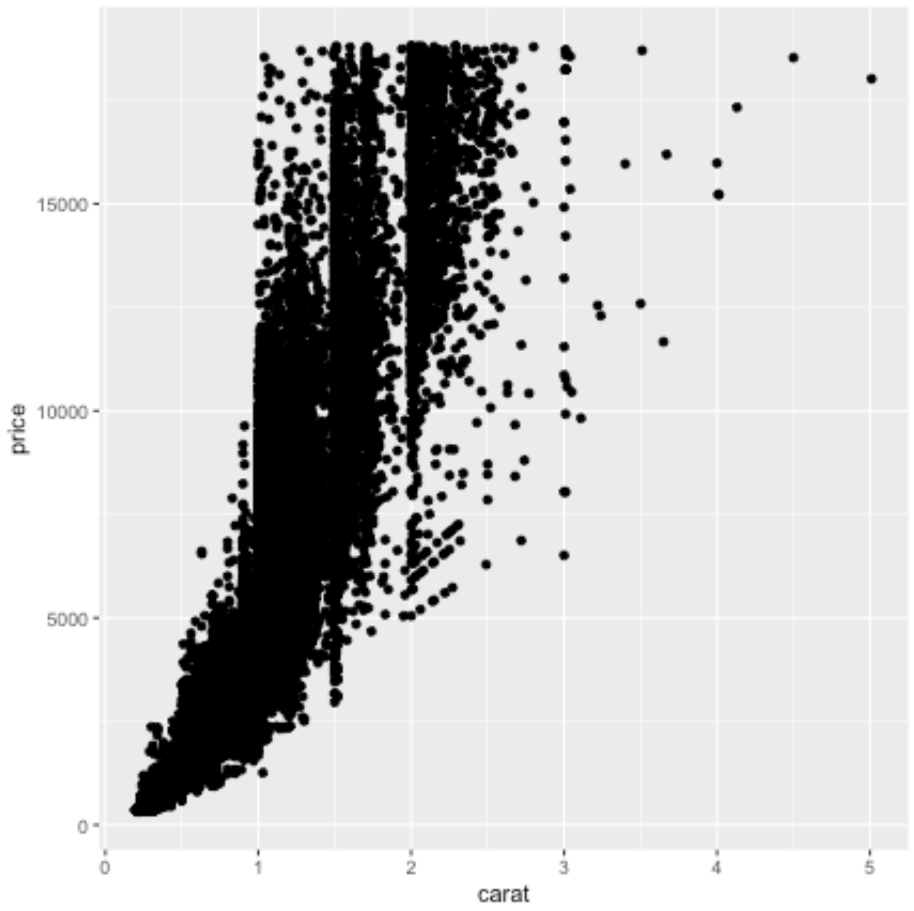
A simple plot with `ggplot2`

Creating a ggplot

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price))
```

Creating a ggplot

```
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price))
```



What did the code do?

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price))
```

- `ggplot()` - Created a coordinate system that we can draw layers to. This actually creates an empty graph.
- The first argument to `ggplot(data = diamonds)` is the data that we want to draw.
- The second row adds a layer to the plot. In this case a layer of *points*.

- The `geom_point()` function takes a **mapping** argument that defines how variables in the data set are mapped to visual properties in the layer.

- In our example, the visual properties are the x- and y-coordinates.

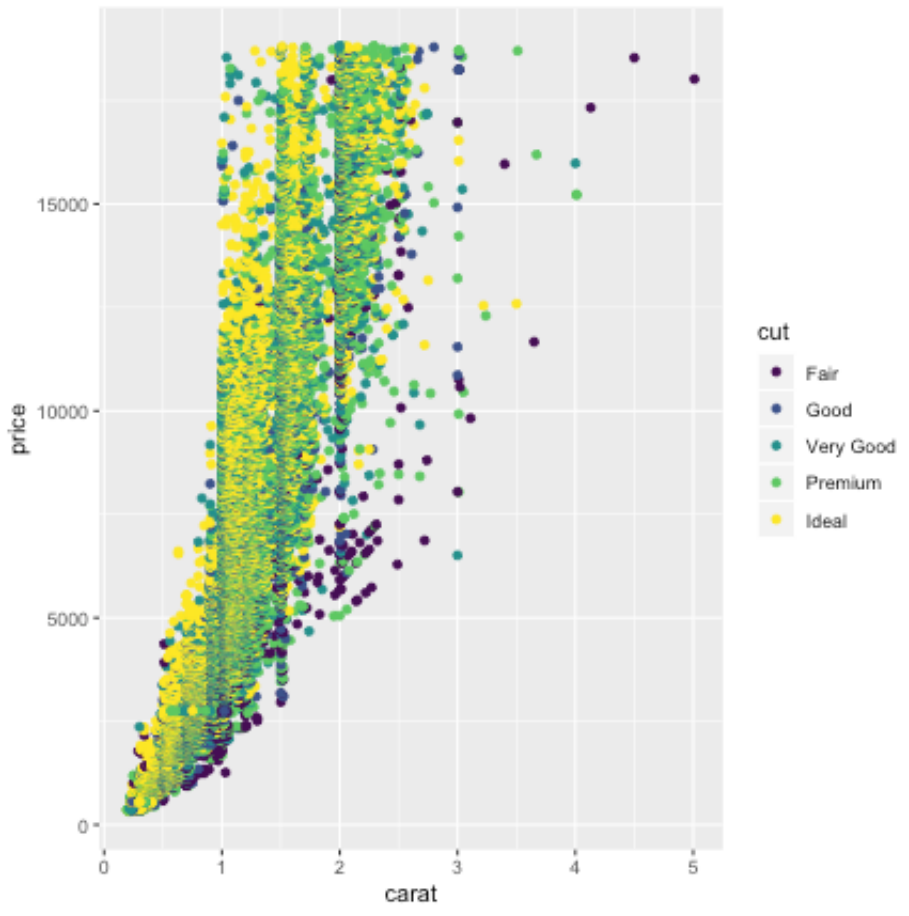
Aesthetics

Aesthetics mappings

- An aesthetic is a visual property of the objects in the plot.
- There are several aesthetics available in `ggplot2` and which are available also depends on the object you are trying to plot.
    - Size
    - Shape
    - Color
    - Alpha
- `ggplot2` will automatically assign a unique level of the aesthetic to each unique value of a variable (**scaling**)
- `ggplot2` will also add a legend to explain which level corresponds to which value.

Example - color

```
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price, color = cut))
```



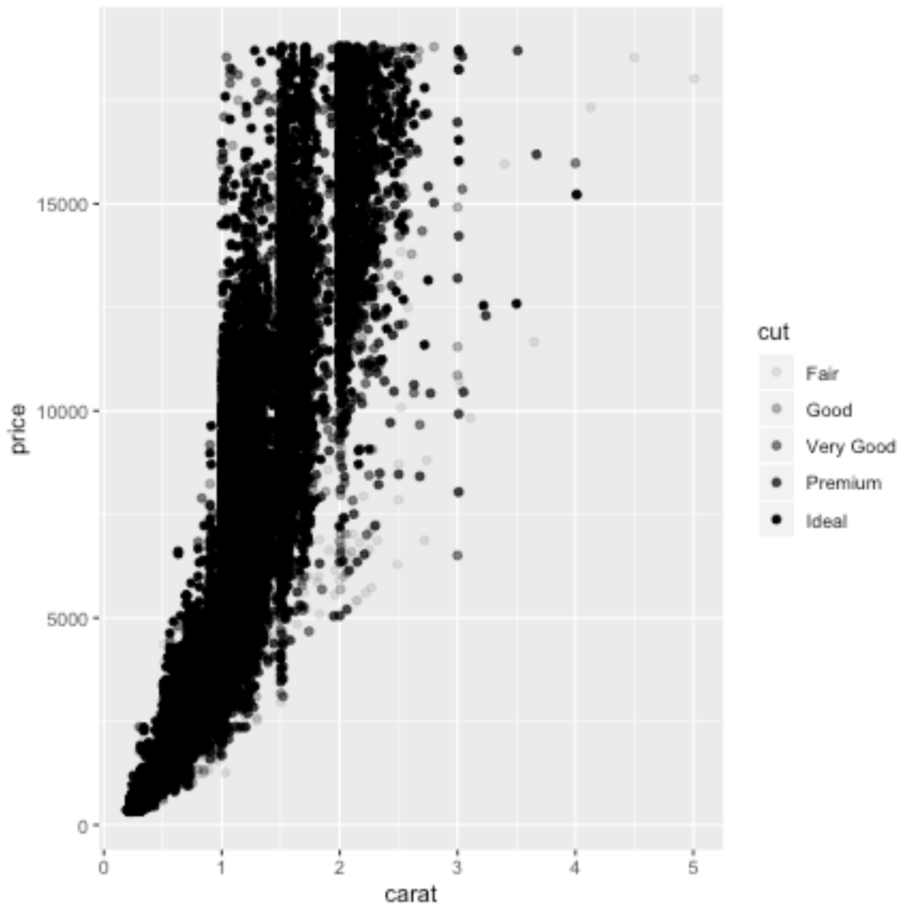
Example - shape

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price, shape = cut))
```

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

Example - alpha

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price, alpha = cut))
```



Setting aesthetics explicitly

- It is also possible to set an aesthetic for an object explicitly.
- This is then done **outside** of the `aes()` function.

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price), color = "blue")
```

Facets

Facets

- Using aesthetics, we can add variables to our plot.
- Another way to add variables, particularly categorical ones, is to use **facets**.
- Facets is a way to split the data into subgroups, and plot each subgroup in a separate graph.
- Importantly, using facets keeps the scaling the same in all graphs, which helps when comparing data.

Facet - `facet_wrap()`

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price)) +
  facet_wrap(~cut, nrow = 3)
```

Facet - `facet_grid()`

```
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price)) +
  facet_wrap(color~cut)
```

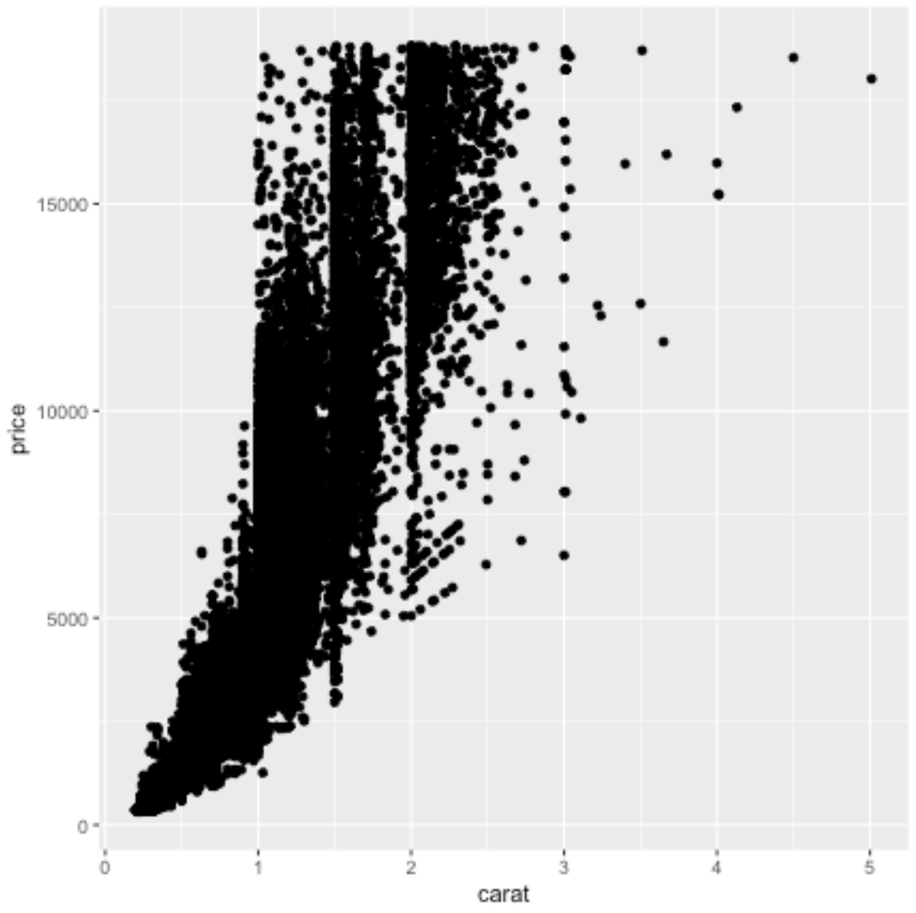
Facet - `facet_grid()`

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price)) +
  facet_wrap(.~cut)
```

Geoms

Geometric objects - Plot 1

```r
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price))
```



Geometric objects - Plot 2

```
# code chunk here
ggplot(data = diamonds) +
  geom_smooth(mapping = aes(x = carat, y = price))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Geometric objects

- Plot 1 and 2 describe the same data, but they are not identical.
- They use different **visual objects** to represent the data.
- In `ggplot2` visual objects are referred to as **geoms**.

- A geom is a geometrical object that the plot uses to represent the data.

- We change the geom(s) that we want to display by calling different geom functions.

- Every geom function takes a mapping argument. But not every aesthetic works with every geom.

Geom functions - Examples

- `geom_histogram()`
- `geom_point()`
- `geom_boxplot()`
- `geom_smooth()`
- `geom_bar()`
- `geom_errorbar()`
- `geom_pointrange()`
- `geom_rect()`
- `geom_hline()`
- `geom_vline()`
- `geom_violin()`

- `geom_text()`

- See also https://www.ggplot2-exts.org for extension packages.

Multiple geoms in the same plot

```
# code chunk here
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price)) +
  geom_smooth(mapping = aes(x = carat, y = price))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Geoms can share data and mapping

```r
# code chunk here
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

And they can have their own

```r
# code chunk here
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_point(aes(color = color)) +
  geom_smooth(aes(linetype = cut))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Geoms do not have to plot the same data

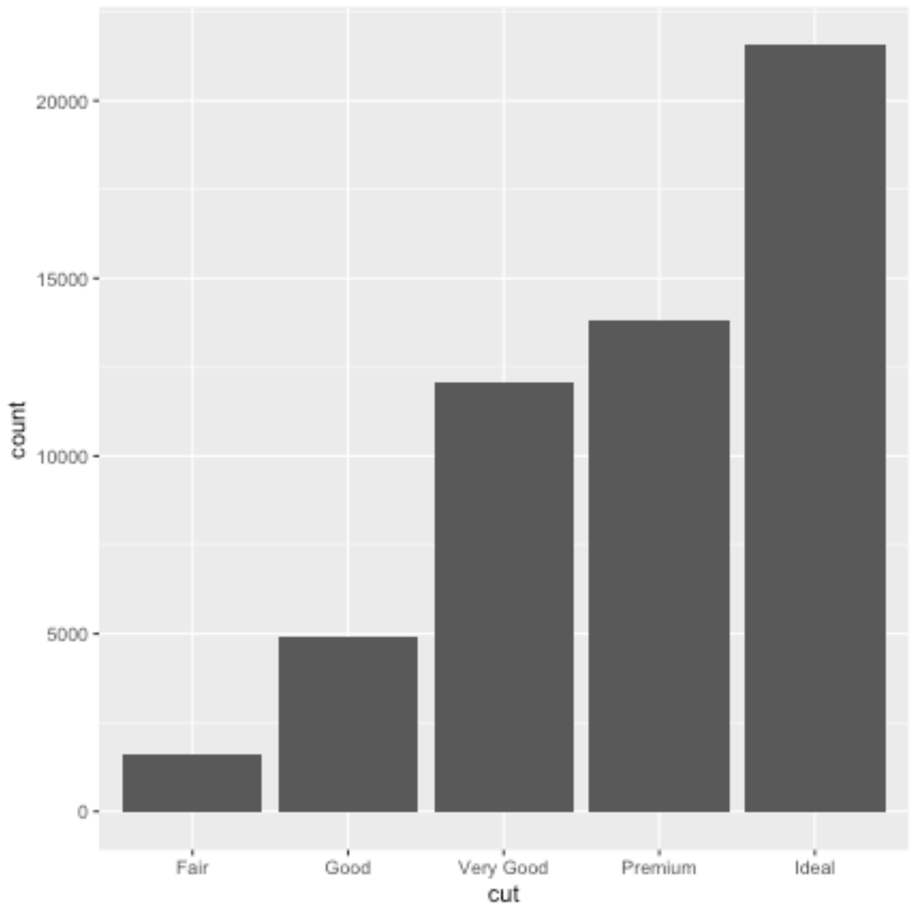
```r
# code chunk here
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
   geom_point(aes(color = color)) +
   geom_smooth(data = filter(diamonds, cut == "Fair"), se = FALSE)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

Stats

Statistical transformations

```
# code chunk here
ggplot(data = diamonds) +
   geom_bar(aes(x = cut))
```



Statistical transformations

- Some graphs, like scatter plots, plot the raw data from the data frame you pass to it.
- Other graphs, like bar graphs, calculate new values that are plotted. E.g.:
    - Bar charts, histograms, and frequency polygons bin the data and plot counts in each bin.
    - Smoothers, fit a model to the data and plot predictions from the model.
    - Box plots compute robust summary statistics that are displayed.

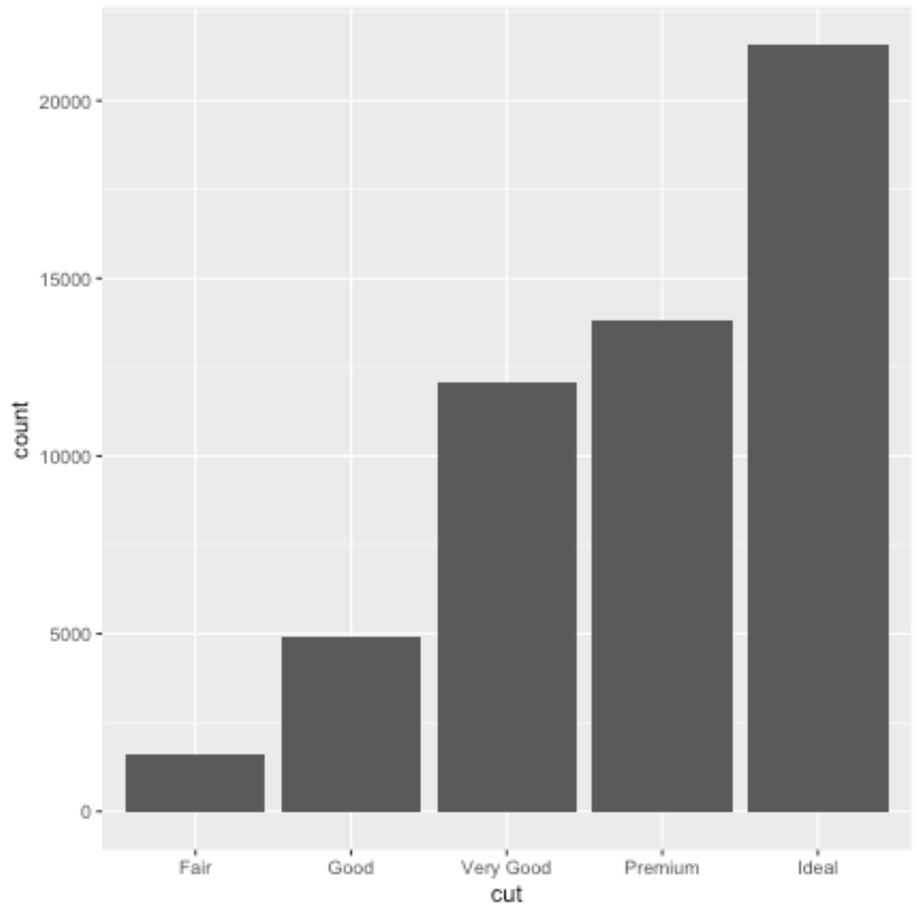
- In `ggplot2` vocabulary, the algorithm that transforms raw data to the new values that are plotted is called a **stat**.

- Every geom has a default stat. Conversely, every stat has a default geom.

- Hence, we can use geoms and stats interchangeably.

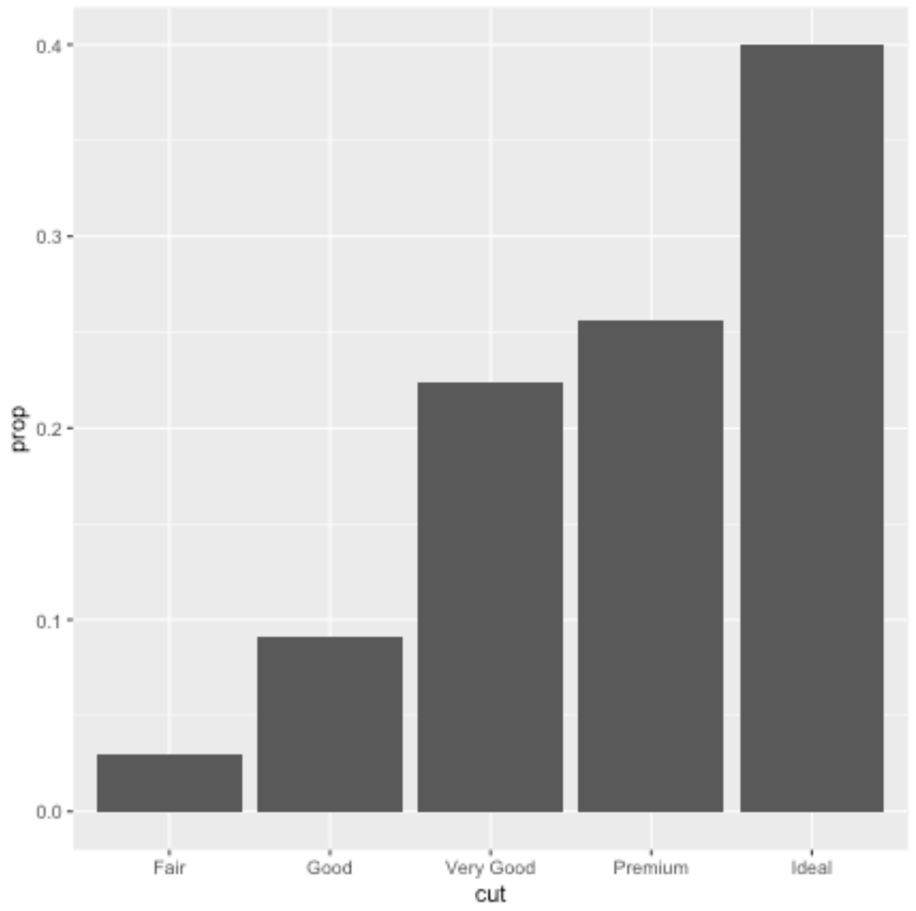
Example

```
# code chunk here
ggplot(data = diamonds) +
   stat_count(aes(x = cut))
```



Override the default stat

```
# code chunk here
ggplot(data = diamonds) +
   geom_bar(mapping = aes(x = cut, y = ..prop.., group = 1))
```



Coordinate Systems

Coordinate Systems

- `ggplot2` uses a Cartesian coordinate system by default.
- Most often, it is not necessary to change this default behavior. But sometimes it can be useful.
- We can also change the scale on which x, y are plotted.

Coordinate system example

```
# code chunk here
ggplot(data = diamonds) +
   geom_bar(aes(x = cut)) +
  coord_flip()
```

Coordinate system example - 2

```r
# code chunk here
ggplot(data = diamonds) +
   geom_bar(aes(x = cut)) +
  coord_flip() +
  coord_polar()
```

```
## Coordinate system already present. Adding new coordinate system, which will repla
```

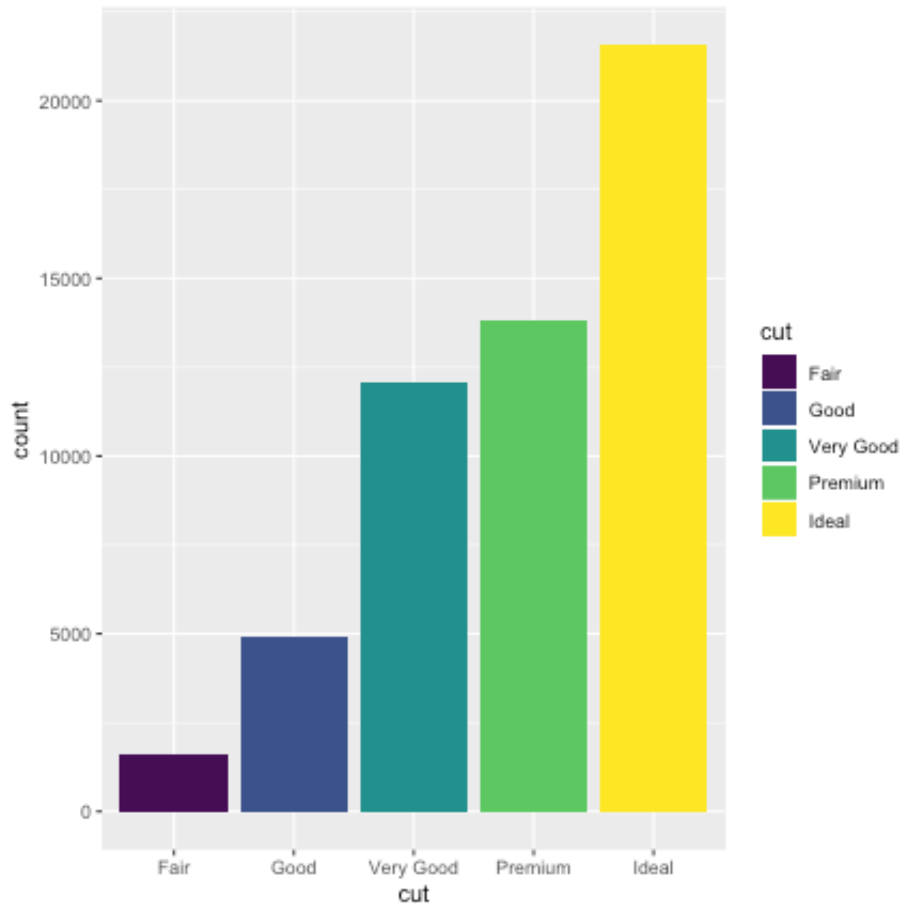
Scale example

```r
# code chunk here
ggplot(data = diamonds) +
   geom_point(aes(x = carat, y = price)) +
   scale_y_log10()
```

Customizing plots

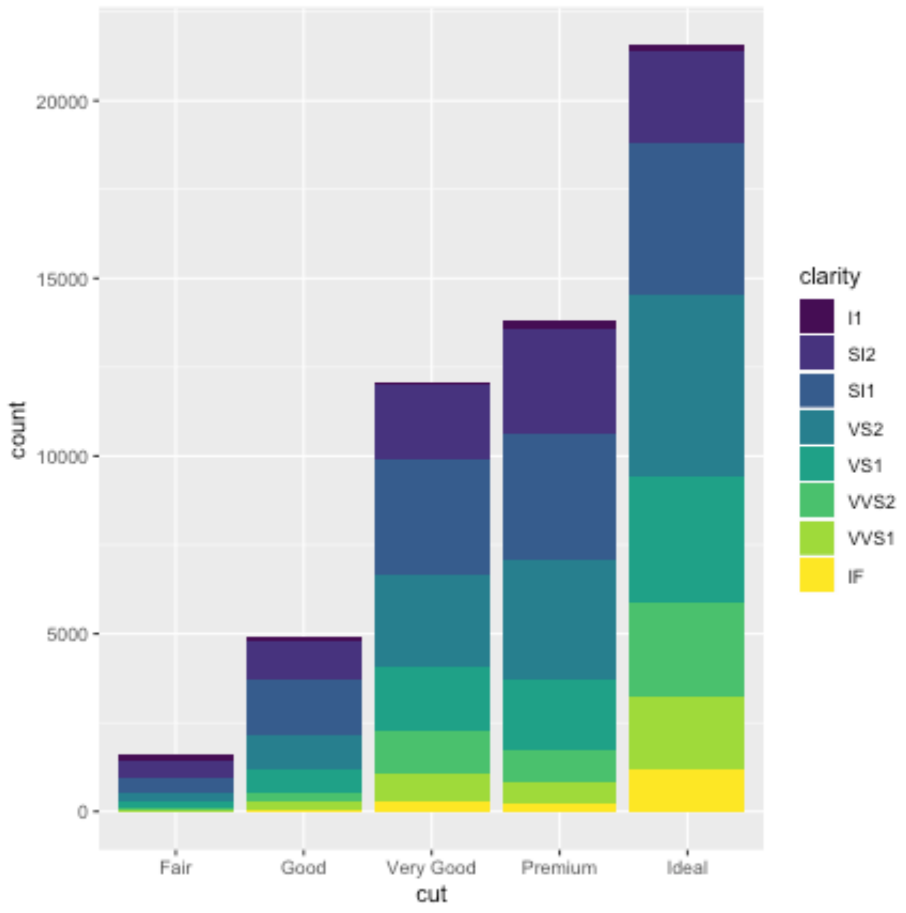
Positioning

```
# code chunk here
ggplot(data = diamonds) +
   geom_bar(aes(x = cut, fill = cut))
```



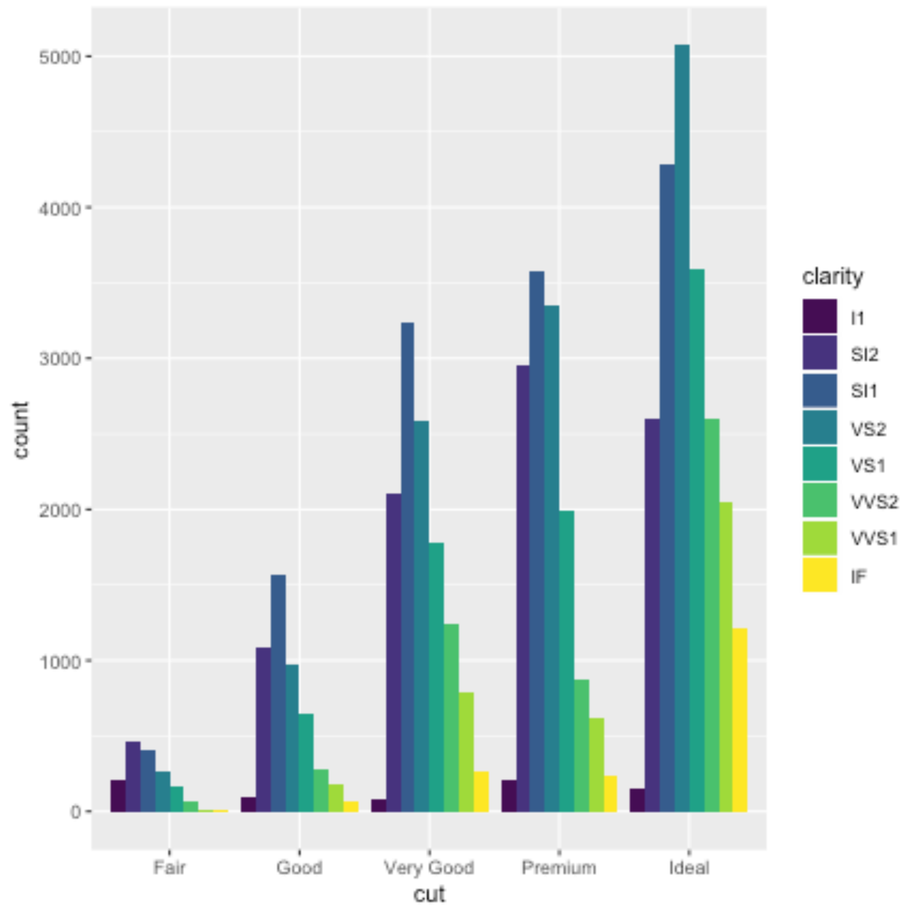
Positioning continued

```
# code chunk here
ggplot(data = diamonds) +
   geom_bar(aes(x = cut, fill = clarity))
```



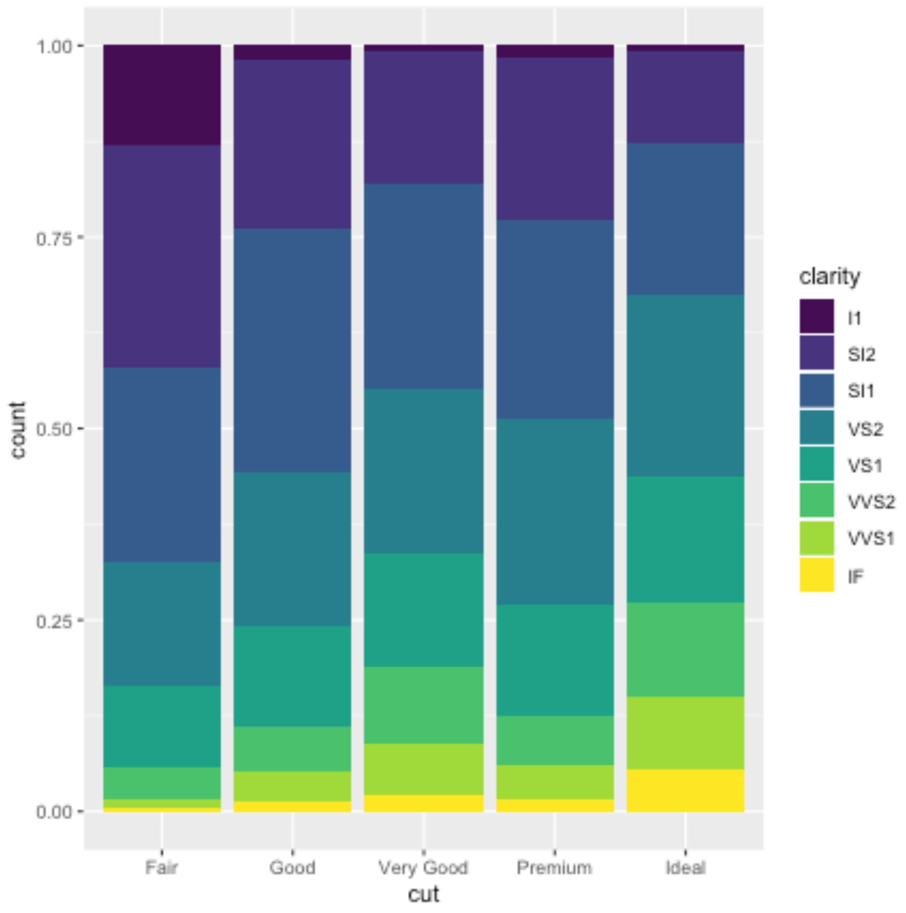
Positioning continued further

```r
# code chunk here
ggplot(data = diamonds) +
   geom_bar(aes(x = cut, fill = clarity), position = "dodge")
```



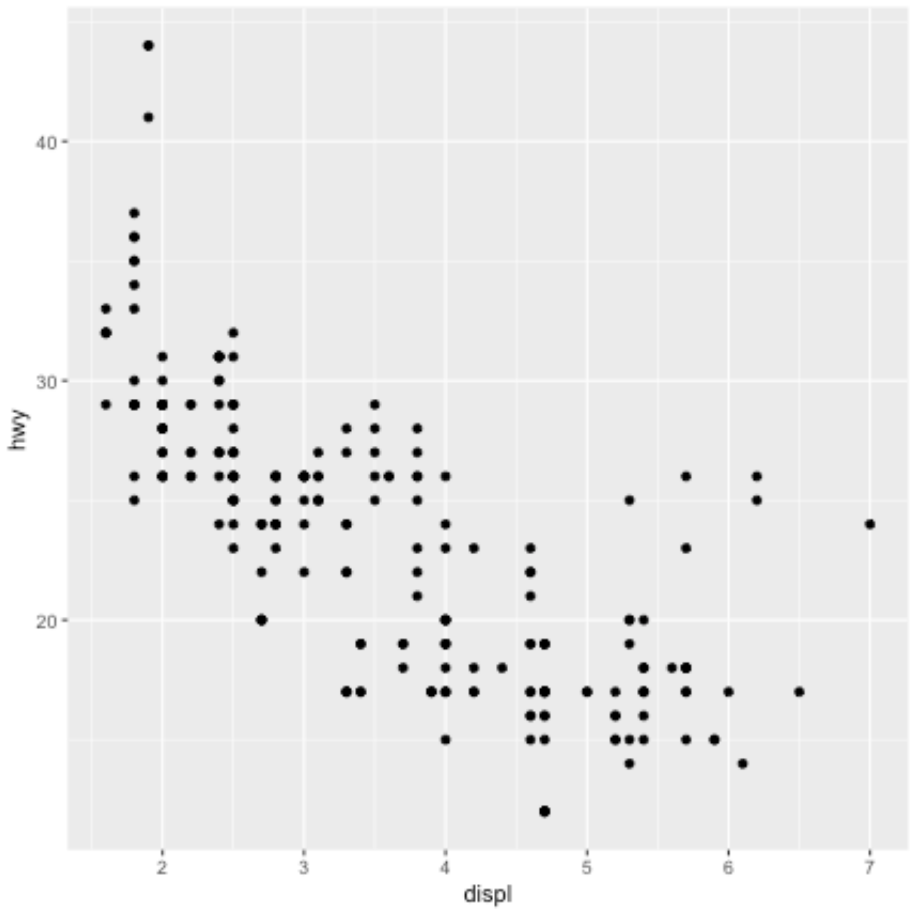
Positioning continued further still

```
# code chunk here
ggplot(data = diamonds) +
   geom_bar(aes(x = cut, fill = clarity), position = "fill")
```



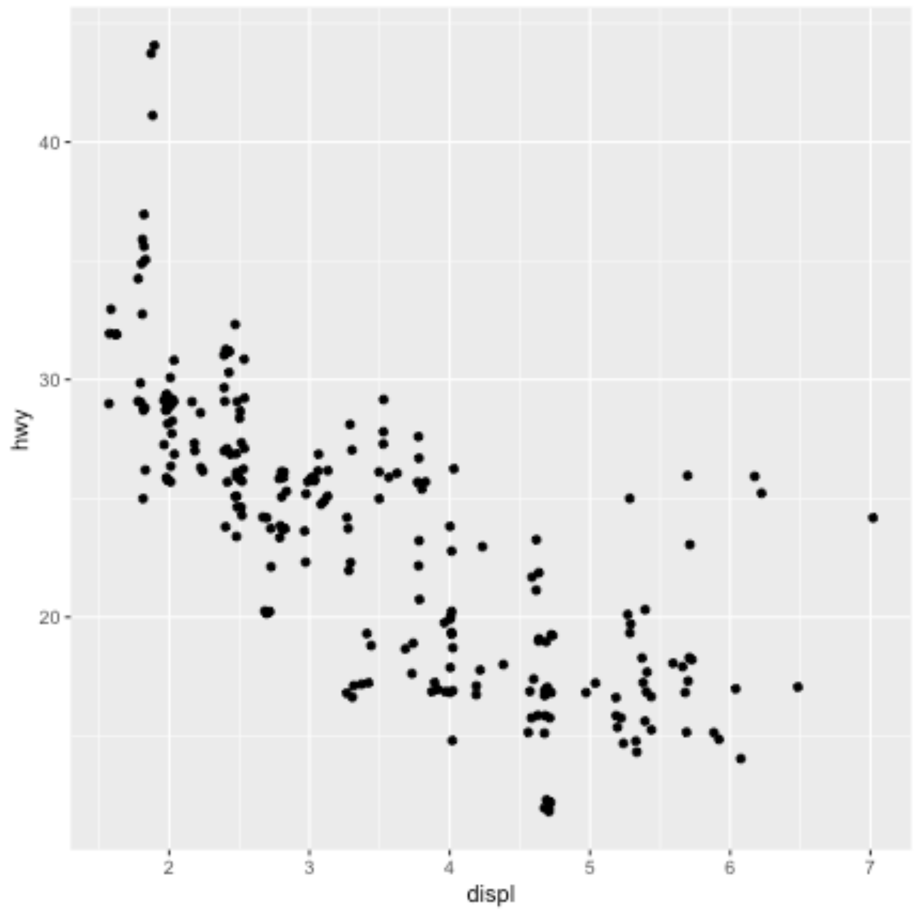
Positioning - we need some more

```
# code chunk here
ggplot(data = mpg) +
   geom_point(aes(x = displ, y = hwy))
```



Positioning - we need just a little more

```r
# code chunk here
ggplot(data = mpg) +
   geom_point(aes(x = displ, y = hwy), position = "jitter")
```

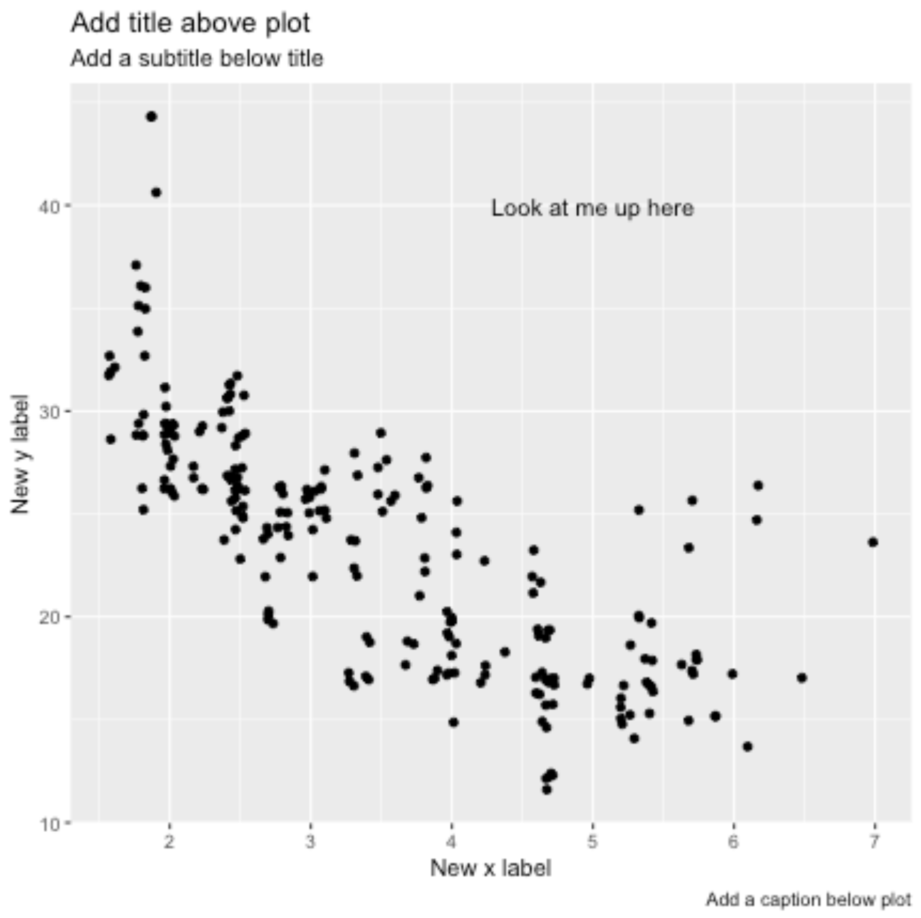


Labels

```r
p <- ggplot(data = mpg) +
   geom_point(aes(x = displ, y = hwy), position = "jitter")

p + labs(x = "New x label", y = "New y label",
         title = "Add title above plot",
         subtitle = "Add a subtitle below title",
         caption = "Add a caption below plot") +
  annotate(geom = "text", x = 5, y = 40, label  = "Look at me up here")
```

Labels

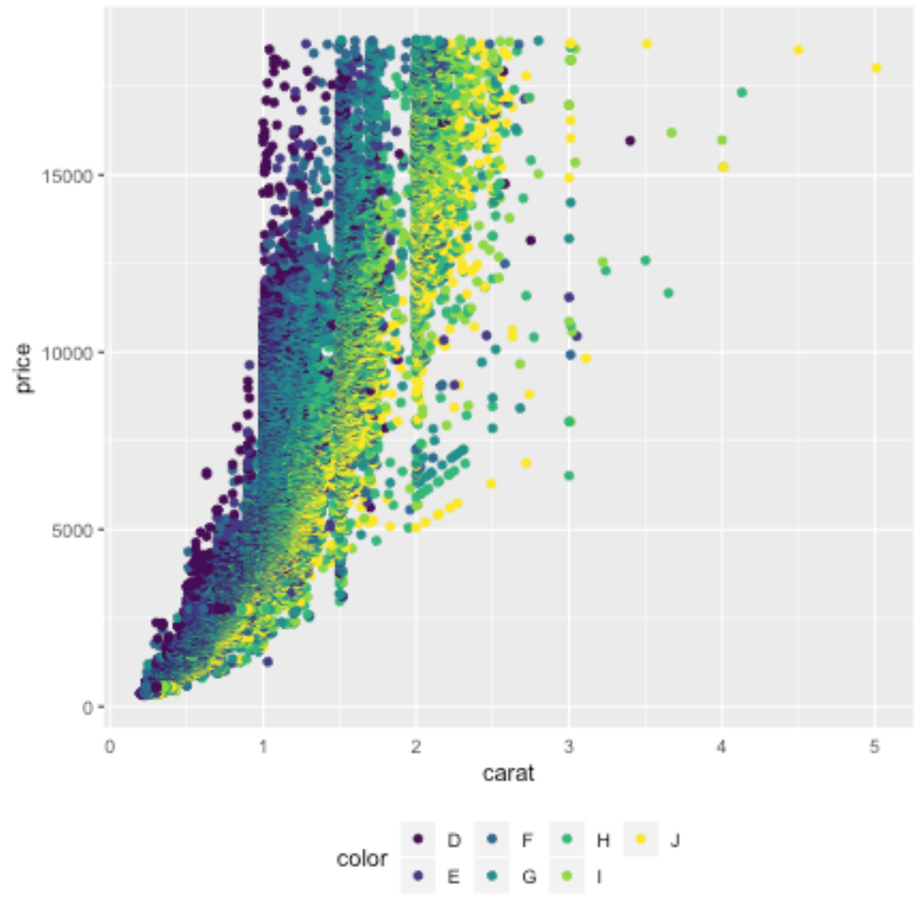

Add title above plot
Add a subtitle below title
Look at me up here
New y label
40
30
20
10
2
3
4
5
6
7
New x label
Add a caption below plot

Legends

```r
p <- ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_point(aes(color = color))

p + theme(legend.position = "bottom",
          axis.ticks.x = element_blank())
```

Legends

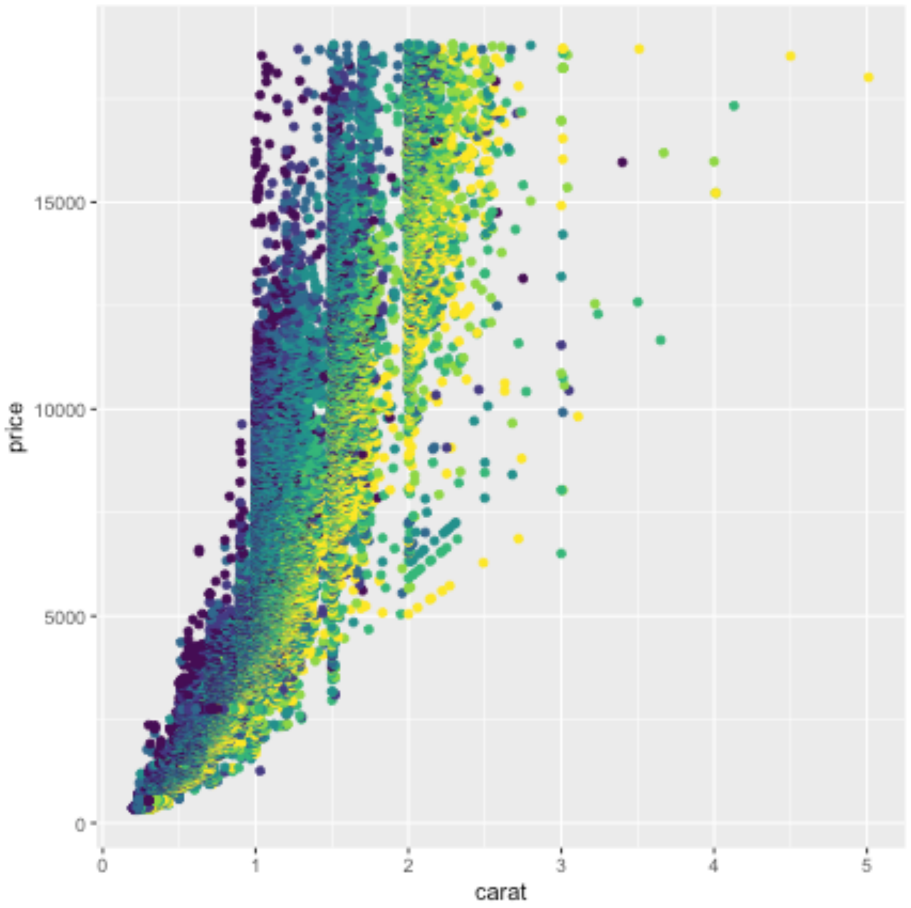


Legends

```r
p <- ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_point(aes(color = color))

p + guides(color = "none")
```

Legends

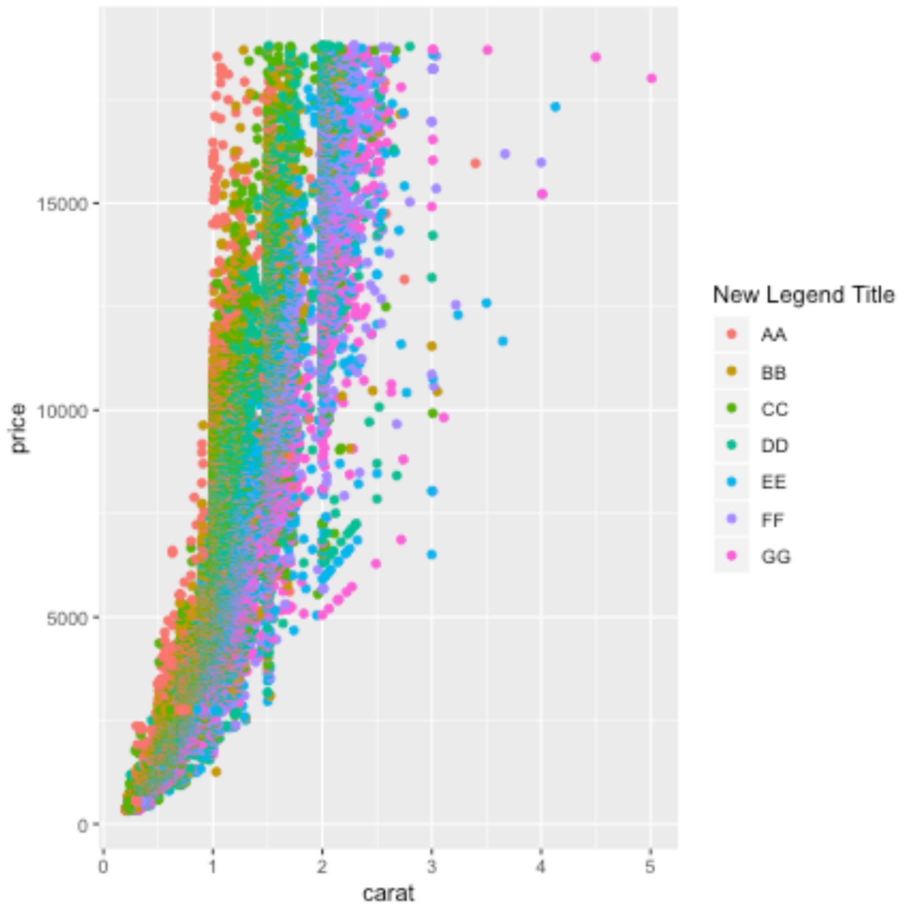


Legends

```r
p <- ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_point(aes(color = color))

p + scale_color_discrete(name = "New Legend Title",
                         labels = c("AA", "BB", "CC",
                                    "DD", "EE", "FF",
                                    "GG"))
```

Legends


price
15000
10000
5000
0
0
1
2
3
4
5
carat
New Legend Title
AA
BB
CC
DD
EE
FF
GG

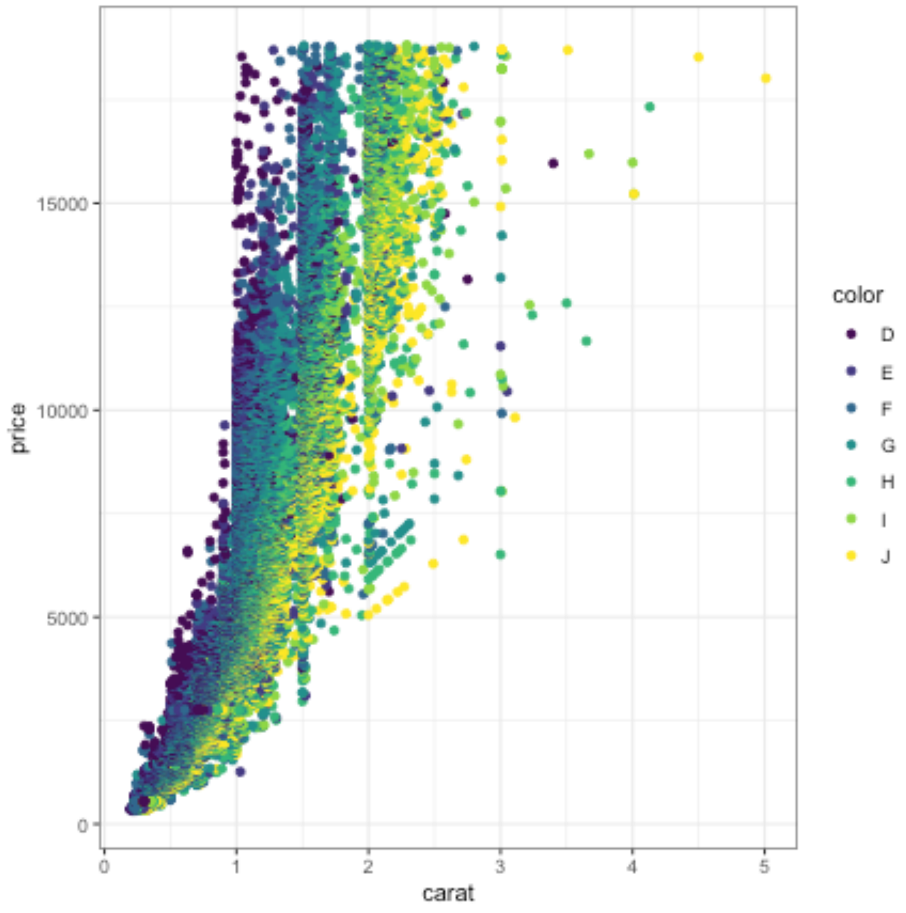
Themes

- The basic appearance of a plot can be changed using **themes**.
- There are a number of built in themes in `ggplot2`.
- Other themes are provided in various packages.

```
p <- ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_point(aes(color = color))

p + theme_bw()
```

Themes

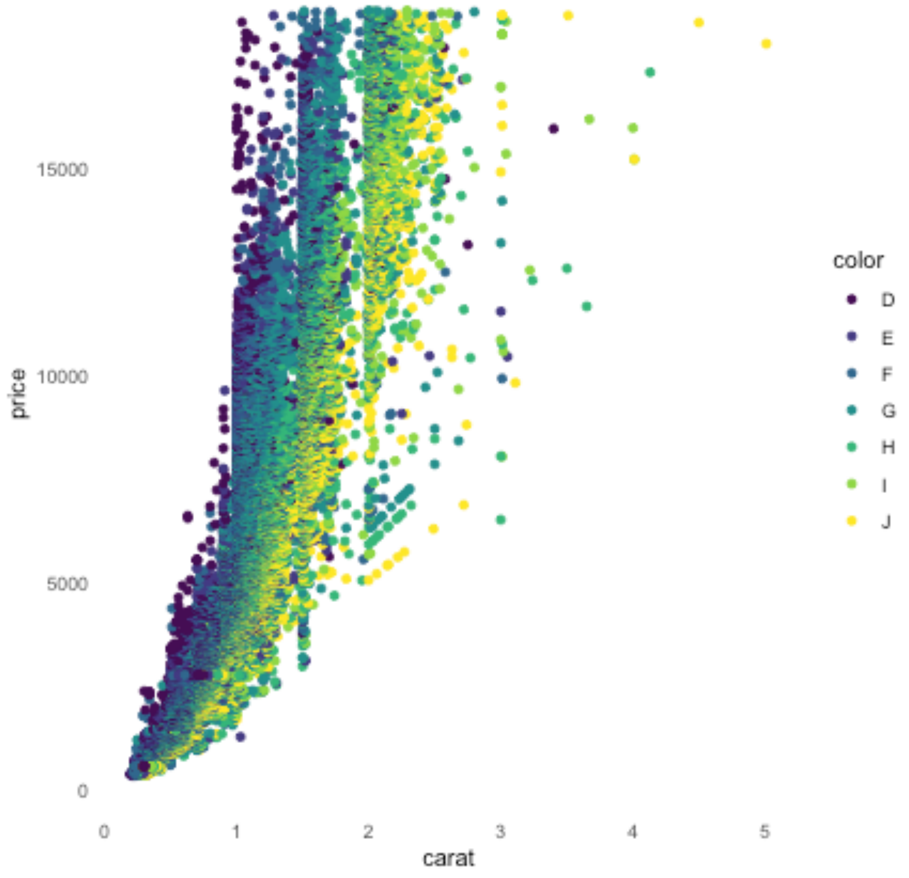


Themes

```
p <- ggplot(data = diamonds, mapping =
              aes(x = carat, y = price)) +
  geom_point(aes(color = color))

p + theme_bw() +
   theme_bw()+
  theme(panel.grid = element_blank(),
        axis.ticks  = element_blank(),
        panel.border = element_blank())
```

Themes



Week 5 - Exercises

- Using the `diamonds` data set in `ggplot2` produce the following seven types of graphs using `ggplot2`.
- Scatter plot, Bar plot, Box plot, Density plot, Dot plot, Histogram, and Violin plot.
- For each graph, be creative with adding variables, changing positions, adding labels, adding legends and adding and customizing themes.
- As long as you produce the seven graphs in a meaningful way, there is no right or wrong. However, I want you to play with as many features of the plot as possible such that you get comfortable with manipulating various aspects of your graph.
- You get special extra points for being creative!!!!
- If you produce boring graphs that don't explore the features of `ggplot2`, you will only be allowed to plot ugly graphs using SPSS for the rest of your life.

That's all folks!