Lab 1: Installing Jupyter Notebook and Diving into ggplot

Derek Hansen (Credit to Yanxin Jin; all errors are my own)

Lab 1 Overview

1. Organization
2. Icebreaker
3. Getting set up with R and Jupyter
4. Additional R and ggplot basics
5. Troubleshooting (if time permits)

Office Hours

- Derek Hansen: M 9:00-10:30am
- Yanxin Jin: Tu 8:30-10:00am
- Jing Ouyang: Wed 12:00pm-1:30pm
- Brian Manzo: Th 9:00-10:30am

Contact

- Homework and lecture question: Piazza
  - All GSIs and Prof. Tan can answer
  - Option to be anonymous to other students (but GSIs and Prof. Tan will see your name)
- For other concerns related to the course: dereklh@umich.edu
  - Please put [STATS 306] in your heading

Lab

- 1.5 hours
- All notes from this Lab will be available on Github: https://github.com/dereklhansen/stats306_lab (https://github.com/dereklhansen/stats306_lab)
  - Jupyter notebook files (".ipynb") and PDFS (".pdf") available
- Recordings will be uploaded to Canvas
- 10 minute break halfway through
- Please keep your mic muted unless actively speaking
- No webcams required unless we are doing an interactive activity

Homework submission

- Write your homework in jupyter notebook and submit .ipynb file to the main course Canvas page (STATS 306 001)

Icebreaker

- Name?
- Major?
- Year?
- Fun fact

Getting set up with R and Jupyter

- I'm going to walk through an easy way to get up and running with R and Jupyter
- https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/ (https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/)

Additional R and ggplot basics

Here we'll demonstrate another dataset: the diamonds dataset.

```
In [6]: suppressMessages(library(tidyverse))

        head(diamonds)
```

A tibble: 6 × 10

carat	cut	color	clarity	depth	table	price	x	y	z
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

```
In [7]: summary(diamonds)
```

```
    carat                cut          color        clarity          depth      
Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065   Min.   :43.00  
1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258   1st Qu.:61.00  
Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80  
Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171   Mean   :61.75  
3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50  
Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00  
                                   J: 2808   (Other): 2531                  
    table           price             x                y         
Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000  
1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720  
Median :57.00   Median : 2401   Median : 5.700   Median : 5.710  
Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735  
3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540  
Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900  
                                                                 
      z         
Min.   : 0.000  
1st Qu.: 2.910  
Median : 3.530  
Mean   : 3.539  
3rd Qu.: 4.040  
Max.   :31.800  
```

R's built-in help

Just about everything in R is documented. Use the `help` function to open up a pop-up about the object in question.

```
In [8]: help(diamonds)
```

```
In [9]: ?diamonds
```

- Here the help for the diamonds dataset says: "A dataset containing the prices and other attributes of almost 54,000 diamonds".
- It also describes the variables in each of the columns.

Saving and loading R objects

- An important part of data analysis is saving your work so you can read it in later.
- The `saveRDS` function will save an object as a file
- The `readRDS` function will read a saved object from a file
- Use ".rds" or ".RDS" file extensions
- Note: "RDS" files can only be read by R!
- By default, saves in the current directory

In [10]:
```
saveRDS(diamonds, "diamonds.rds")
diamonds2 <- readRDS("diamonds.rds")
diamonds2
```

A tibble: 53940 × 10

carat	cut	color	clarity	depth	table	price	x	y	z
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48
0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47
0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53
0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
0.23	Very Good	H	VS1	59.4	61	338	4.00	4.05	2.39
0.30	Good	J	SI1	64.0	55	339	4.25	4.28	2.73
0.23	Ideal	J	VS1	62.8	56	340	3.93	3.90	2.46
0.22	Premium	F	SI1	60.4	61	342	3.88	3.84	2.33
0.31	Ideal	J	SI2	62.2	54	344	4.35	4.37	2.71
0.20	Premium	E	SI2	60.2	62	345	3.79	3.75	2.27
0.32	Premium	E	I1	60.9	58	345	4.38	4.42	2.68
0.30	Ideal	I	SI2	62.0	54	348	4.31	4.34	2.68
0.30	Good	J	SI1	63.4	54	351	4.23	4.29	2.70
0.30	Good	J	SI1	63.8	56	351	4.23	4.26	2.71
0.30	Very Good	J	SI1	62.7	59	351	4.21	4.27	2.66
0.30	Good	I	SI2	63.3	56	351	4.26	4.30	2.71
0.23	Very Good	E	VS2	63.8	55	352	3.85	3.92	2.48
0.23	Very Good	H	VS1	61.0	57	353	3.94	3.96	2.41
0.31	Very Good	J	SI1	59.4	62	353	4.39	4.43	2.62
0.31	Very Good	J	SI1	58.1	62	353	4.44	4.47	2.59
0.23	Very Good	G	VVS2	60.4	58	354	3.97	4.01	2.41
0.24	Premium	I	VS1	62.5	57	355	3.97	3.94	2.47
0.30	Very Good	J	VS2	62.2	57	357	4.28	4.30	2.67
0.23	Very Good	D	VS2	60.5	61	357	3.96	3.97	2.40
0.23	Very Good	F	VS1	60.9	57	357	3.96	3.99	2.42
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.70	Premium	E	SI1	60.5	58	2753	5.74	5.77	3.48
0.57	Premium	E	IF	59.8	60	2753	5.43	5.38	3.23
0.61	Premium	F	VVS1	61.8	59	2753	5.48	5.40	3.36
0.80	Good	G	VS2	64.2	58	2753	5.84	5.81	3.74
0.84	Good	I	VS1	63.7	59	2753	5.94	5.90	3.77
0.77	Ideal	E	SI2	62.1	56	2753	5.84	5.86	3.63

carat	cut	color	clarity	depth	table	price	x	y	z
<dbl>	<ord>	<ord>	<ord>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
0.74	Good	D	SI1	63.1	59	2753	5.71	5.74	3.61
0.90	Very Good	J	SI1	63.2	60	2753	6.12	6.09	3.86
0.76	Premium	I	VS1	59.3	62	2753	5.93	5.85	3.49
0.76	Ideal	I	VVS1	62.2	55	2753	5.89	5.87	3.66
0.70	Very Good	E	VS2	62.4	60	2755	5.57	5.61	3.49
0.70	Very Good	E	VS2	62.8	60	2755	5.59	5.65	3.53
0.70	Very Good	D	VS1	63.1	59	2755	5.67	5.58	3.55
0.73	Ideal	I	VS2	61.3	56	2756	5.80	5.84	3.57
0.73	Ideal	I	VS2	61.6	55	2756	5.82	5.84	3.59
0.79	Ideal	I	SI1	61.6	56	2756	5.95	5.97	3.67
0.71	Ideal	E	SI1	61.9	56	2756	5.71	5.73	3.54
0.79	Good	F	SI1	58.1	59	2756	6.06	6.13	3.54
0.79	Premium	E	SI2	61.4	58	2756	6.03	5.96	3.68
0.71	Ideal	G	VS1	61.4	56	2756	5.76	5.73	3.53
0.71	Premium	E	SI1	60.5	55	2756	5.79	5.74	3.49
0.71	Premium	F	SI1	59.8	62	2756	5.74	5.73	3.43
0.70	Very Good	E	VS2	60.5	59	2757	5.71	5.76	3.47
0.70	Very Good	E	VS2	61.2	59	2757	5.69	5.72	3.49
0.72	Premium	D	SI1	62.7	59	2757	5.69	5.73	3.58
0.72	Ideal	D	SI1	60.8	57	2757	5.75	5.76	3.50
0.72	Good	D	SI1	63.1	55	2757	5.69	5.75	3.61
0.70	Very Good	D	SI1	62.8	60	2757	5.66	5.68	3.56

- More useful functions from the "readr" package (part of the tidyverse) are `read_csv` and `write_csv`
- These read and write "Comma Separated Value" files, which are text files that can be read as spreadsheets into programs such as Excel
- A lot of datasets come in CSV format
- Pretty much all data analysis programs support CSV (R, Python, SAS, Stata, Excel, etc)
- You can specify the variable type of each column, or R will guess automatically

In [11]:

```
write_csv(diamonds, "diamonds.csv")
diamonds3 <- read_csv("diamonds.csv")
```

```
Parsed with column specification:
cols(
  carat = col_double(),
  cut = col_character(),
  color = col_character(),
  clarity = col_character(),
  depth = col_double(),
  table = col_double(),
  price = col_double(),
  x = col_double(),
  y = col_double(),
  z = col_double()
)
```

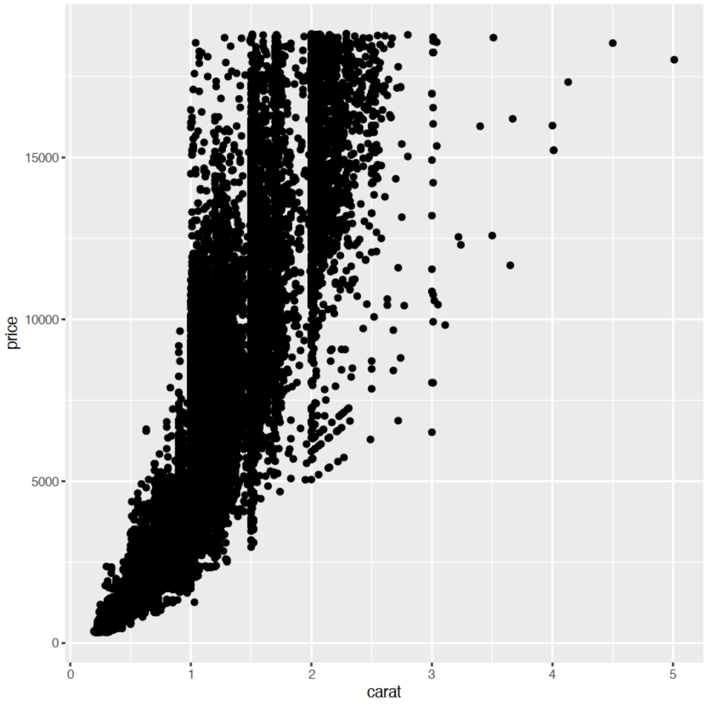
Many more file formats that R can read and write to:

- `read_tsv`, `read_delim` for text files with different text seperators
- `haven` package for reading to/from Stata, SAS, SDSS
- `readxl` package for reading to/from Excel
  - Generally better to just use CSV with Excel possible!

Creating the first plot

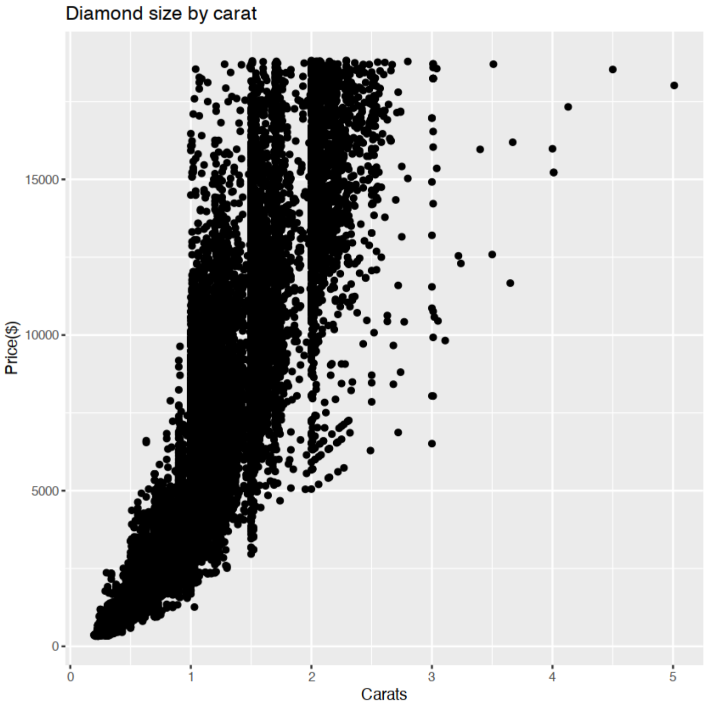
In [12]:

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



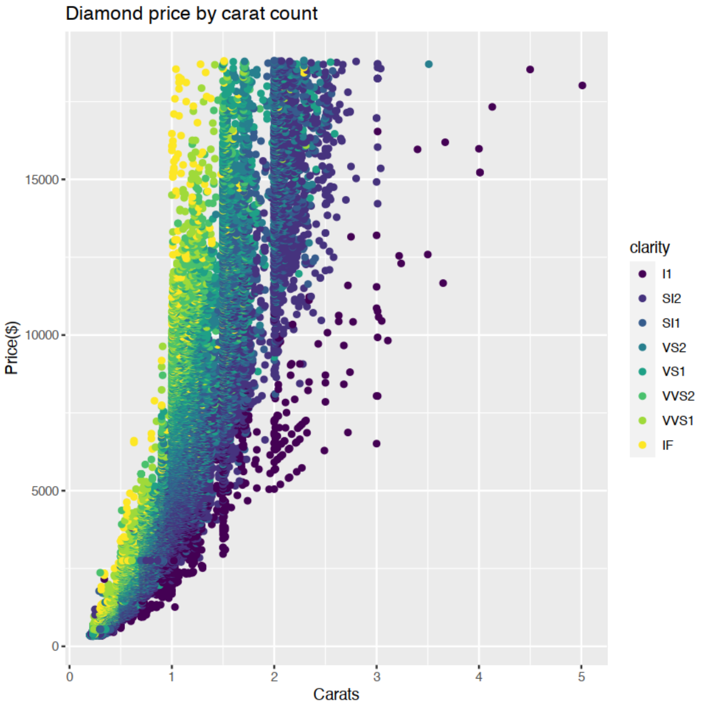
In [13]:

```
ggplot(data=diamonds) +
    geom_point(mapping = aes(x = carat, y = price)) +
    labs(x = 'Carats', y = 'Price($)') +
    ggtitle('Diamond size by carat')
```

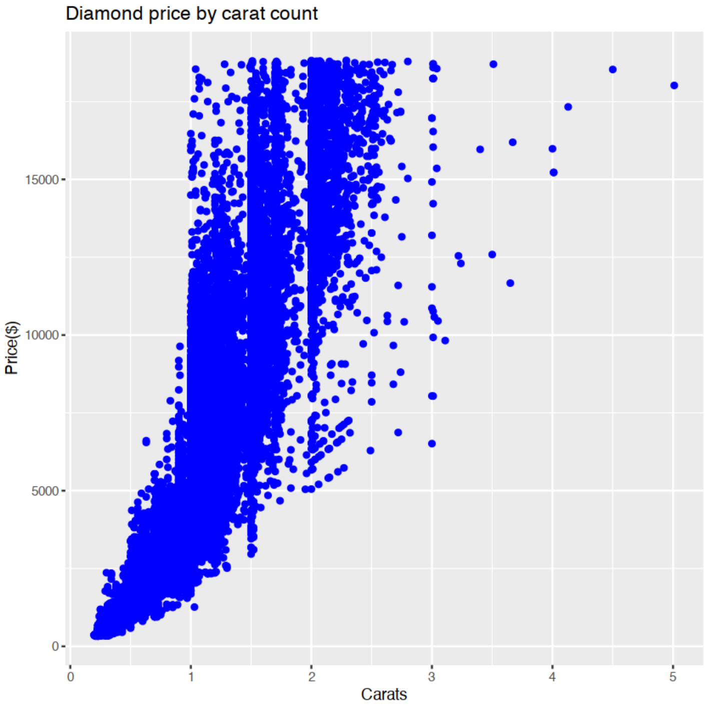


In [14]:

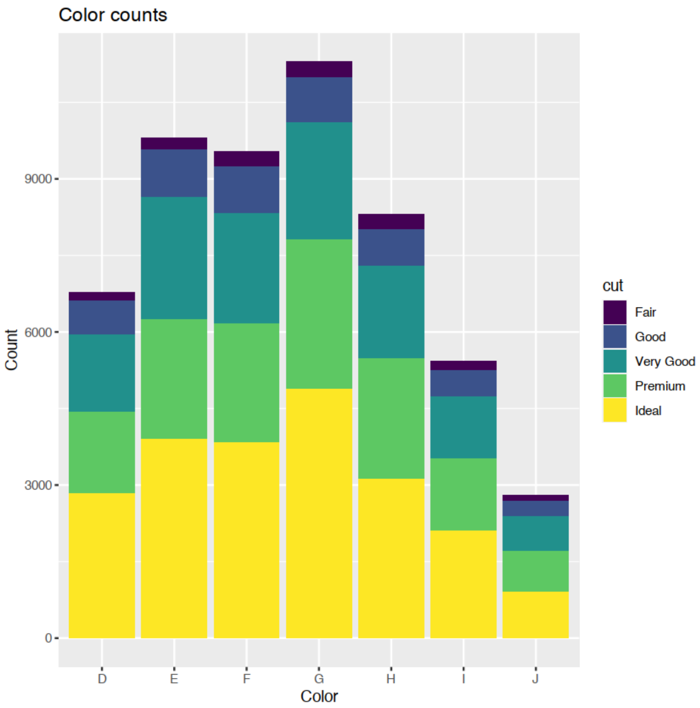
```
ggplot(data = diamonds) +
    geom_point(mapping = aes(x = carat, y = price, color = clarity)) +
    labs(x = 'Carats', y = 'Price($)') +
    ggtitle('Diamond price by carat count')
```



In [15]:
```
ggplot(data = diamonds) +
    geom_point(mapping = aes(x = carat, y = price), color = 'blue') +
    labs(x = 'Carats', y = 'Price($)') +
    ggtitle('Diamond price by carat count')
```



In [16]:
```
ggplot(data = diamonds) +
    # geom_point(mapping = aes(x = carat, y = price, color = clarity)) +
    geom_bar(mapping = aes(x = color, fill = cut)) +
    labs(x = 'Color', y = 'Count') +
    ggtitle('Color counts')
```



Package documentation

```
?head
?tail
?geom_bar

You can always google the package documentations as well. For example, can you find th
e documentation page for rnorm?
```

Exercise

```
1. What is the default value of the mean and standard deviation used by the ``rnorm''
function in R to generate a value from a normal distribution?
2. Create a boxplot of `price' grouped by the levels in the `cut' variable. (see Jupyt
er notebook for snippets for a hint)
```

In [17]:

```
Below are some imcomplete code snippets to help with the second exercise
```

```
Error in parse(text = x, srcfile = src): <text>:1:7: unexpected symbol
1: Below are
          ^
Traceback:
```

In [ ]:

```r
# boxplot helps to visualize the variability of a price for each cut
ggplot(data = diamonds) +
    geom_boxplot() +
    labs() +
    ggtitle()
```

Facets

If we want more segmented plots

In [ ]:

```r
ggplot(data = diamonds) +
    geom_point() +
    labs() +
    facet_grid() +
    ggtitle()
```

In [ ]: