Data Import; Exploratory Data Analysis (EDA) in R

Based on lecture handouts originally written by Dr. David Gerard.

Learning Objectives

- Import data from CSV's,
- Working Directories
- Strategies for EDA
- Data Import Cheat Sheet.
- Readr Overview.

Part 1 (Data Import)

Working Directories

- The working directory is where R will look for and save things by default.
- When you specify to save a figure, save a file, or load some data, it will be with respect to the working directory.
- You can see where the current working directory is by `getwd()`, or by looking at the top of the console in RStudio.
- You can change the working directory by Session > Set Working Directory > Choose Directory. Or by CONTROL + SHIFT + H. Or you can use the `setwd()` command.
- A shortcut is to set the working directory to your source file location with Session > Set Working Directory > To Source File Location.
- When you read and write files/figures, you can then specify the path from the position of the working directory.
- Suppose we want to save the following figure:

  ```r
  suppressPackageStartupMessages(library(tidyverse))
  ```

  ```
  ## Warning: package 'tidyverse' was built under R version 4.0.5

  ## Warning: package 'tibble' was built under R version 4.0.5

  ## Warning: package 'tidyr' was built under R version 4.0.5

  ## Warning: package 'dplyr' was built under R version 4.0.4

  ## Warning: package 'forcats' was built under R version 4.0.5
  ```

  ```r
  data("mpg")
  pl <- ggplot(mpg, aes(x = hwy, y = cty)) +
    geom_point()
  ```

- To save `pl` in the current folder, we would use:

```
ggsave(filename = "./my_saved_plot.pdf", plot = pl)
```

- The "`.`" means "the current folder".
- To save `pl` in the folder one level up we would use:

```
ggsave(filename = "../my_saved_plot.pdf", plot = pl)
```

- The "`..`" means "go one level up".
- If we are in the analysis folder, and we want to save `pl` in the output folder, we would use:

```
  ggsave(filename = "../output/my_saved_plot.pdf", plot = pl)
```

- If we have a subfolder called "fig" within out current folder. We could save `pl` in "fig" with

```
  ggsave(filename = "./fig/my_saved_plot.pdf", plot = pl)
```

- **NEVER USE ABSOLUTE PATHS**. For example, you should never start the path from "C" if you use Windows. This makes your code non-transferable to other users.

readr

- To read a CSV (comma-separated values) file into R, use the `read_csv()` function from the readr package.

```
suppressPackageStartupMessages(library(tidyverse))
heights <- read_csv(file = "./heights.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   earn = col_double(),
##   height = col_double(),
##   sex = col_character(),
##   ed = col_double(),
##   age = col_double(),
##   race = col_character()
## )
```

- Use `read_tsv()` if columns are separated by tabs.
- Use `read_csv2()` if columns are separated by semicolons.
- Other file formats are listed in RDS.
- First export the Excel spreadsheet as a CSV. Then read the CSV file into R.
- You are using colors to represent meaningful information in Excel? Don't.
  - Edit the data so that the information is encoded by a new variable.
- If you don't know the format ahead of time, use `read_lines()` to print the first few lines.

```
read_lines(file = "./heights.csv", n_max = 10)
```

```
##  [1] "\"earn\",\"height\",\"sex\",\"ed\",\"age\",\"race\""
##  [2] "50000,74.4244387818035,\"male\",16,45,\"white\""
##  [3] "60000,65.5375428255647,\"female\",16,58,\"white\""
```

```
##  [4] "30000,63.6291977374349,\"female\",16,29,\"white\""
##  [5] "50000,63.1085616752971,\"female\",16,91,\"other\""
##  [6] "51000,63.4024835710879,\"female\",17,39,\"white\""
##  [7] "9000,64.3995075440034,\"female\",15,26,\"white\""
##  [8] "29000,61.6563258264214,\"female\",12,49,\"white\""
##  [9] "32000,72.6985437364783,\"male\",17,46,\"white\""
## [10] "2000,72.0394668497611,\"male\",15,21,\"hispanic\""
```

Special Considerations

- **Always check your data immediately after importing it**.
  - Check that the types are correct for each of the variables.
  - Check that the missing data were coded correctly.
  - Later on, when you notice something weird, consider that this might have resulted because of a problem during data import.

```
hate_crimes <- read_csv(file = "./hate_crimes2.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   state = col_character(),
##   median_house_inc = col_double(),
##   share_unemp_seas = col_double(),
##   share_pop_metro = col_double(),
##   share_pop_hs = col_double(),
##   share_non_citizen = col_double(),
##   share_white_poverty = col_double(),
##   gini_index = col_double(),
##   share_non_white = col_double(),
##   share_vote_trump = col_double(),
##   hate_crimes_per_100k_splc = col_double(),
##   avg_hatecrimes_per_100k_fbi = col_double()
## )
```

```
summarize_all(hate_crimes, class)
```

```
## # A tibble: 1 x 12
##   state     median_house_inc share_unemp_seas share_pop_metro share_pop_hs
##   <chr>     <chr>            <chr>            <chr>           <chr>
## 1 character numeric          numeric          numeric         numeric
## # ... with 7 more variables: share_non_citizen <chr>,
## #   share_white_poverty <chr>, gini_index <chr>, share_non_white <chr>,
## #   share_vote_trump <chr>, hate_crimes_per_100k_splc <chr>,
## #   avg_hatecrimes_per_100k_fbi <chr>
```

```
summarize_all(hate_crimes,funs(sum(is.na(.))))#sum all the NA's under each variable
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
```

```
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))

## # A tibble: 1 x 12
##   state median_house_inc share_unemp_seas share_pop_metro share_pop_hs
##   <int>            <int>            <int>           <int>        <int>
## 1     0                0                0               0            0
## # ... with 7 more variables: share_non_citizen <int>,
## #   share_white_poverty <int>, gini_index <int>, share_non_white <int>,
## #   share_vote_trump <int>, hate_crimes_per_100k_splc <int>,
## #   avg_hatecrimes_per_100k_fbi <int>
```

```
head(hate_crimes)
```

```
## # A tibble: 6 x 12
##   state      median_house_inc share_unemp_seas share_pop_metro share_pop_hs
##   <chr>                 <dbl>            <dbl>           <dbl>        <dbl>
## 1 Alabama               42278            0.06             0.64        0.821
## 2 Alaska                67629            0.064            0.63        0.914
## 3 Arizona               49254            0.063            0.9         0.842
## 4 Arkansas              44922            0.052            0.69        0.824
## 5 California            60487            0.059            0.97        0.806
## 6 Colorado              60940            0.04             0.8         0.893
## # ... with 7 more variables: share_non_citizen <dbl>,
## #   share_white_poverty <dbl>, gini_index <dbl>, share_non_white <dbl>,
## #   share_vote_trump <dbl>, hate_crimes_per_100k_splc <dbl>,
## #   avg_hatecrimes_per_100k_fbi <dbl>
```

- Sometimes the files code missing data other than `NA`. For example, it's common to use periods `.`, or in some genomic settings they use `-9` as missing.
- R won't know how to handle this without you telling it, so you'll have to know what the missing data encoding is and specify it with the `na` argument in `read_csv()`.
- readr will try to guess the type for each column (double, integer, character, logic, etc). Sometimes it guesses wrong. If it seems to be guessing wrong, use the `col_types` to explicitly specify the column types.
- Sometimes there are comments at the start of a data file. You can skip the first few lines before starting to read data with the `skip` argument.
- If the comments begin with a special character, you can use the `comment` argument.

Data Export

- You can write comma-separated and tab-separated files using `write_csv()`, `write_csv2()`, and `write_tsv()`.
- The defaults are usually fine.

Reading/Writing R Objects

- You can save and reload arbitrary R objects (data frames, matrices, lists, vectors) using `readRDS()` and `saveRDS()`.

Part 2 (Exploratory Data Analysis (EDA) in R)

We will use ggplot2 which is a R package dedicated to data visualization.

General Strategies

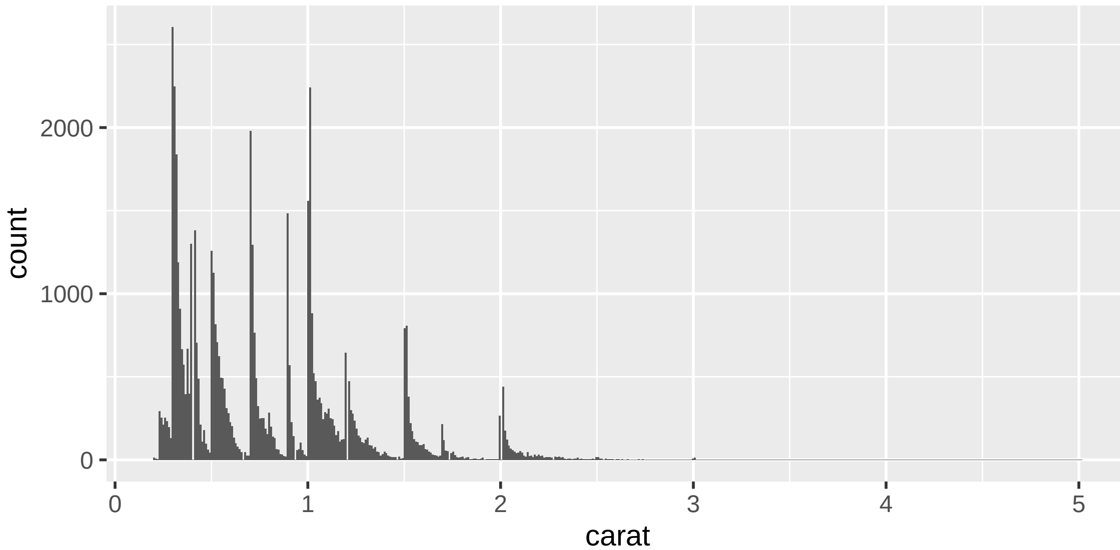
- Plot the distribution of every variable.
- Plot the bivariate distribution of every pair of variables (to find which variables are associated).
- Color code by variables to try and see if relationships can be explained.
- Calculate lots of summary statistics.
- Look at missingness.
- Look at outliers.
- EDA is about **curiosity**. Ask *many* questions, use *many* plots, investigate *many* aspects of your data. This will let you hone in on the few *interesting* questions you want to pursue deeper.

```
library(tidyverse)
data("diamonds")
```

Distribution of Every Variable:

- Quantitative: Use a histogram.
  - Look for modality. Indicates multiple groups of units. What can explain the modes? Can any of the other variables explain the modes?
  - Are certain values more likely than other values?
  - Look for skew.
  - `geom_histogram()`
  - Mean, median, standard deviation, five number summary.

  ```
  ggplot(data = diamonds, mapping = aes(x = carat)) +
    geom_histogram(bins = 500)
  ```




```
fivenum(diamonds$carat)
```

```
## [1] 0.20 0.40 0.70 1.04 5.01
```

```
mean(diamonds$carat)
```

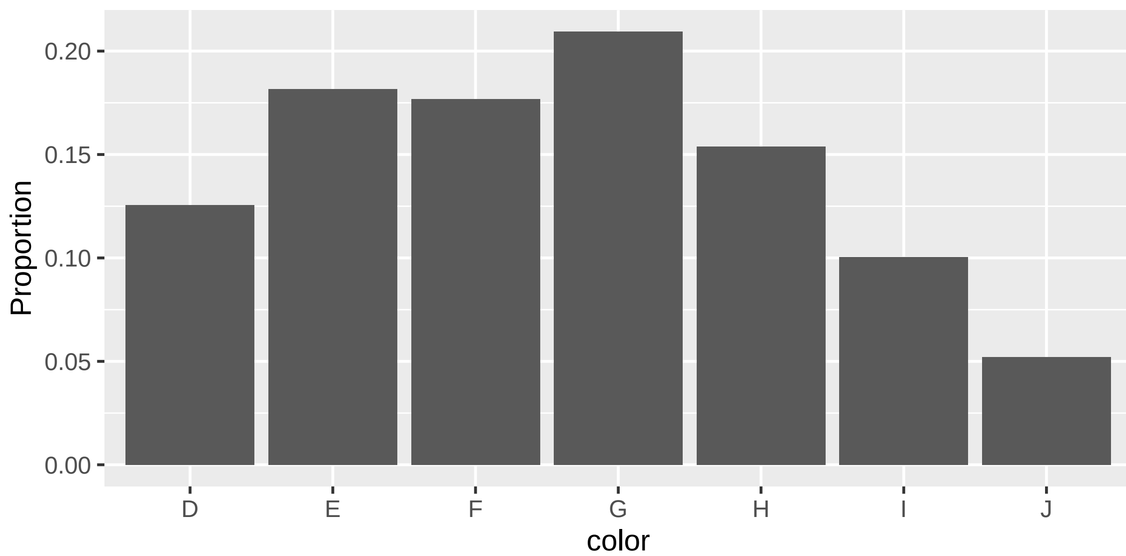
```
## [1] 0.7979397
```

```
sd(diamonds$carat)
```

```
## [1] 0.4740112
```

- Categorical: Use a bar chart. Or just a table of *proportions* (`table()` then `prop.table()`).
  - Absolute counts are sometimes interesting, but usually you want to look at the proportion of observations in each category.
  - Is there a natural ordering of the categories (bad, medium, good)?
  - Why are some categories more represented than others?
  - `geom_bar()`, `geom_col()`
  - Proportion of observations within each group.

```
ggplot(diamonds, aes(x = color, y = )) +
  geom_bar(aes(y = count / sum(..count..))) +
  ylab("Proportion")
```




```
table(diamonds$color)
```

```
##
##     D     E     F     G     H     I     J
##  6775  9797  9542 11292  8304  5422  2808
```

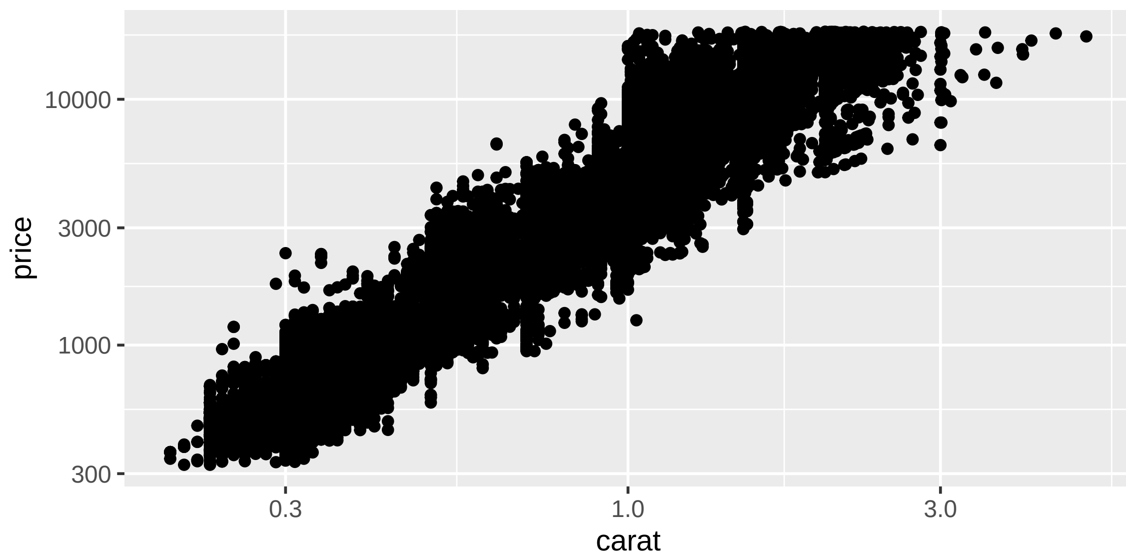
```
prop.table(table(diamonds$color))
```

```
##
##          D          E          F          G          H          I          J
## 0.12560252 0.18162773 0.17690026 0.20934372 0.15394883 0.10051910 0.05205784
```

Bivariate Distribution of Every Pair of Variables

- Quantitative vs Quantitative: Use a scatterplot
  - Is the relationship linear? Quadratic? Exponential?
  - Logging is useful tool to make some associations linear. If the relationship is (i) monotonic and (ii) curved, then try logging the x-variable *if the x-variable is all positive.* If it is also (iii) more variable at larger y-values, then try logging the y-variable *instead* of the x-variable *if the y-variable is all positive.* Try logging both if you still see curvature *if both variables are all positive.*
  - Ask if an observed association can be explained by another variable?
  - Correlation coefficient (only appropriate if association is linear).
  - Kendall's tau (always appropriate).

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() +
  scale_y_log10() +
  scale_x_log10()
```



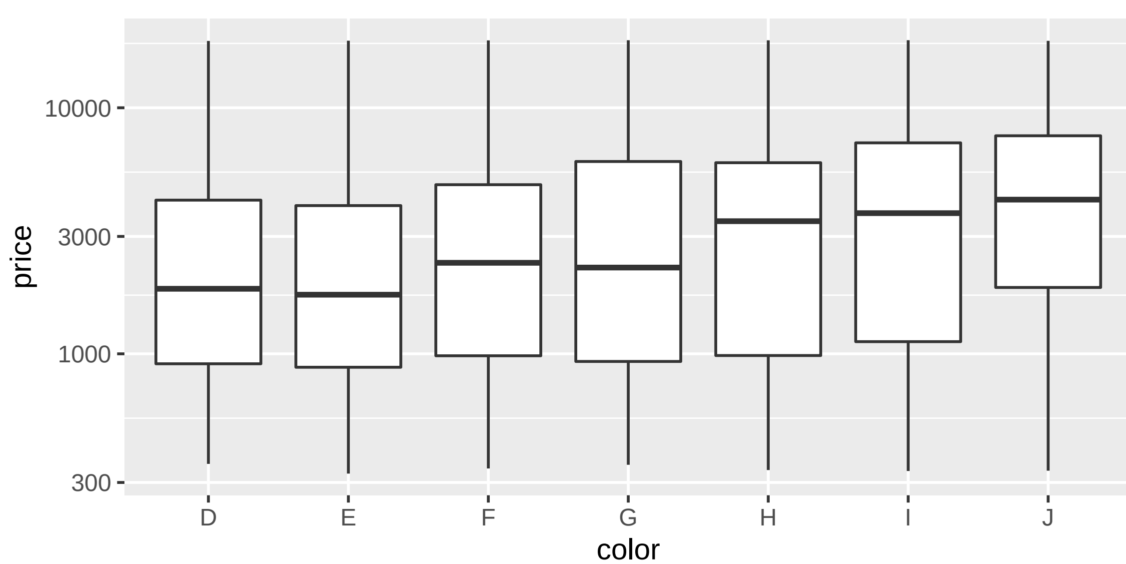

```
cor(diamonds$carat, diamonds$price)
```

```
## [1] 0.9215913
```

```
## cor(diamonds$carat, diamonds$price, method = "kendall")
```

- Categorical vs Quantitative: Use a boxplot
  - For which levels of the categorical variable is the quantitative variable higher or lower?
  - For which levels is the quantitative variable more spread out?
  - Aggregated means, medians, standard deviations, quantiles

```
ggplot(diamonds, aes(x = color, y = price)) +
  geom_boxplot() +
  scale_y_log10()
```



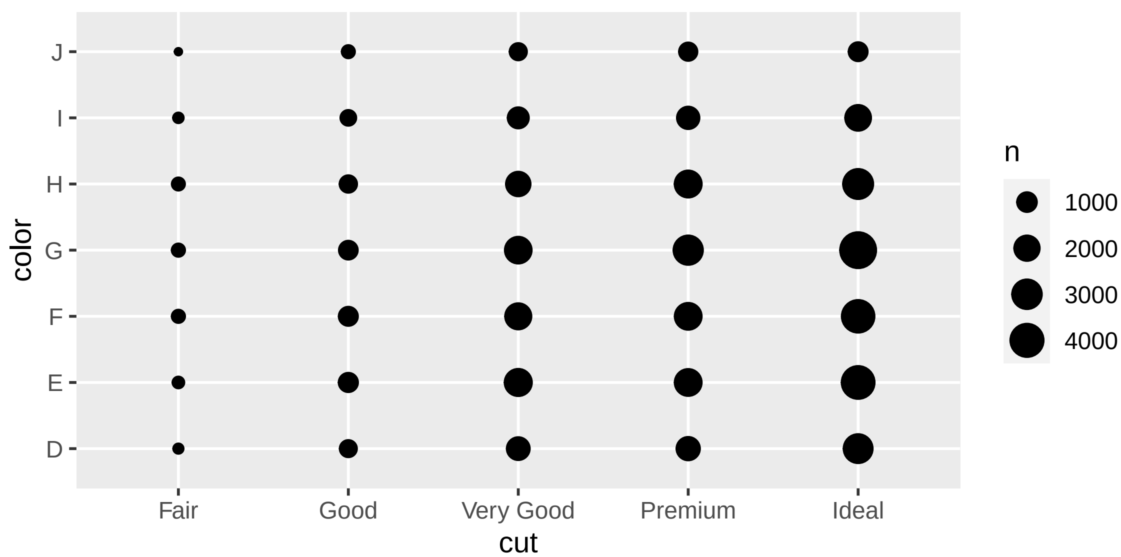

```
diamonds %>%
  mutate(logprice = log(price)) %>%
  group_by(color) %>%
```

```
  summarize(mean   = mean(logprice),
            sd     = sd(logprice),
            median = median(logprice),
            Q1     = quantile(logprice, 0.25),
            Q3     = quantile(logprice, 0.75))
```

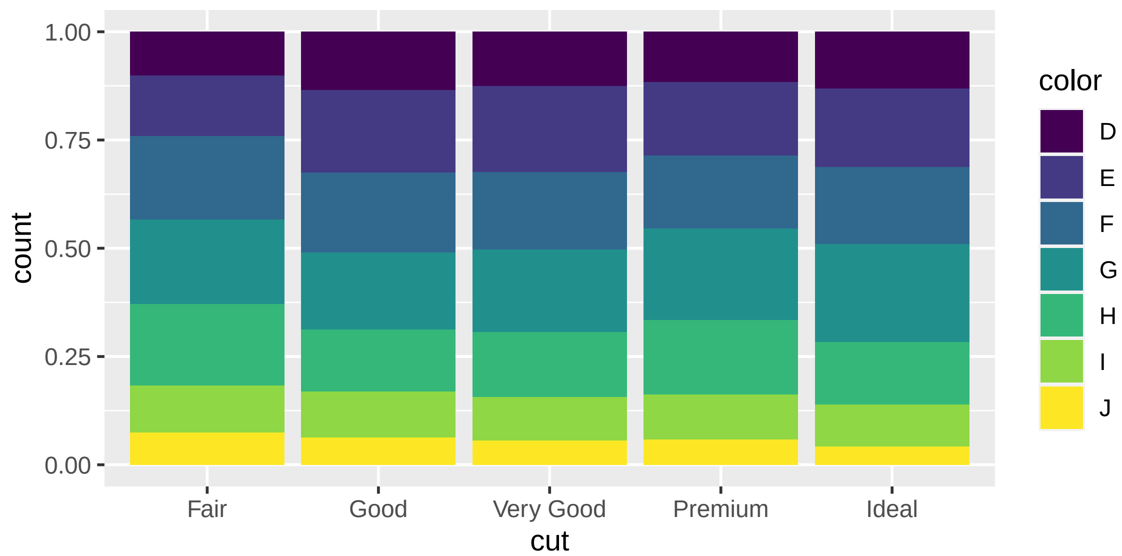
```
## # A tibble: 7 x 6
##   color  mean    sd median    Q1    Q3
##   <ord> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 D      7.62 0.926   7.52  6.81  8.35
## 2 E      7.58 0.925   7.46  6.78  8.29
## 3 F      7.76 0.968   7.76  6.89  8.49
## 4 G      7.79 1.03    7.72  6.84  8.71
## 5 H      7.92 1.06    8.15  6.89  8.70
## 6 I      8.02 1.11    8.22  7.02  8.88
## 7 J      8.15 1.04    8.35  7.53  8.95
```

- Categorical vs Categorical: Use a mosaic plot or a count plot
  - For which pairs of values of the categorical variables are there the most number of units?

```
## Only gives you the bivariate distribution
ggplot(diamonds, aes(x = cut, y = color)) +
  geom_count()
```




```
## Gives you the conditional distributions of color given cut
ggplot(diamonds, aes(x = cut, fill = color)) +
  geom_bar(position = "fill")
```




```r
## Gives you the conditional distributions of cut given color
ggplot(diamonds, aes(x = color, fill = cut)) +
  geom_bar(position = "fill")
```