Skills Problem set III

Earnest Salgado

21/04/2021

```
library(tidyverse)
library(dslabs)
library(ggplot2)
rm(list = ls())
knitr::opts_chunk$set(fig.width=6, fig.height=3)
```

This submission is my work alone and complies with the 30535 integrity policy.

Add your initials to indicate your agreement: **ES**

Add your collaborators: **GATW**

Late coins used this pset: 0. Late coins left: 9.

1 Exercises

1.1 Transformations: grouped operations with mutate() and filter() (20 points)

In this section, we will use the dataset gapminder in the dslabs package. This data includes health and income outcomes for 184 countries between 1960 and 2016.

1. Let's explore gdp growth:

a. Order the data by country. For each observation use lag() and group_by() to compute previous year gdp.

```
order_by_country <- gapminder %>%
  arrange(country) %>%
  group_by(country) %>%
  mutate(previousyr_gdp = lag(gdp)) %>%
  select(country,year,gdp,previousyr_gdp,continent,region)
```

b. Calculate gdp growth by country and year. Plot average annual growth by continent. Choose wisely your geom graph is as informative as possible. What do you learn from this plot?

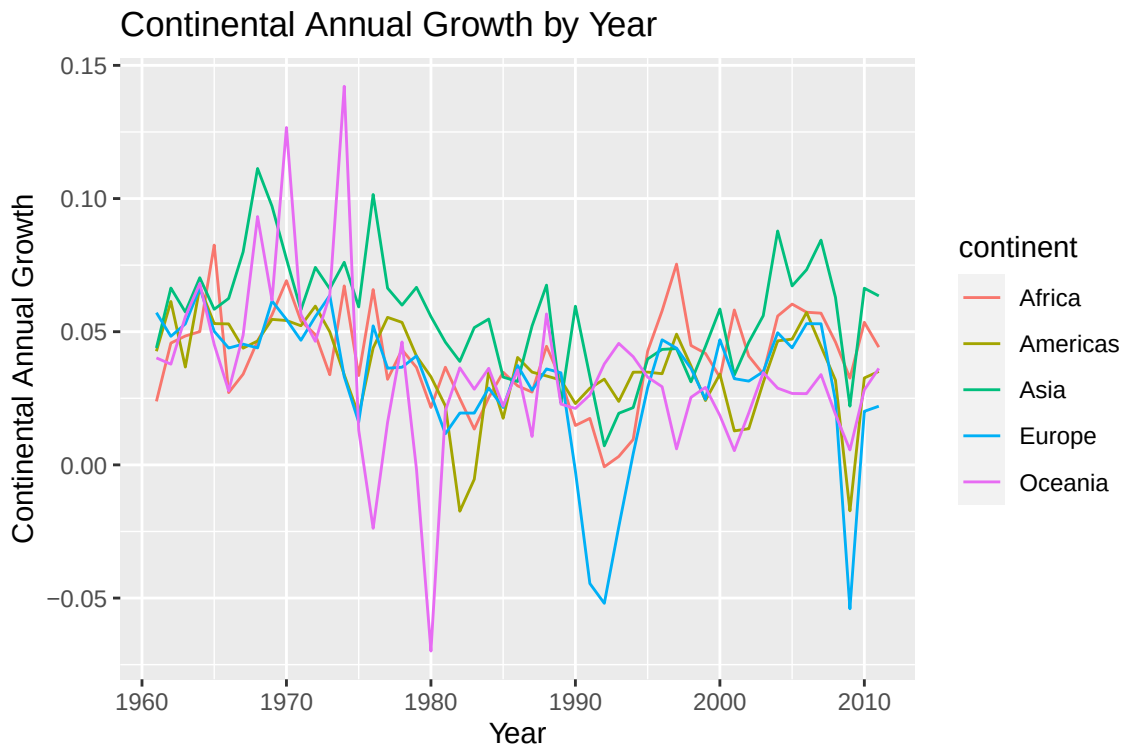
From this plot we learn broad patterns of annual gdp growth in terms of continents. Generally since 1960, gdp growth has held steady around 0.05 annually for all continents. Oceania has both the highest and lowest gdp growth swings between 1970 to 1980, then has a sideways pattern through 2010. Aside from them, Europe has experienced the largest dips in gdp growth, with two significant periods of negative growth in the early 1990s and late 2000s. Asia seems to have the highest trendline for gdp growth.

```r
growth_gdp <- order_by_country %>% mutate(
  country_annual_growth = (gdp - previousyr_gdp)/previousyr_gdp)

continent_growth <- growth_gdp %>%
  group_by(continent, year) %>%
  summarise(cont_annual_growth = mean(country_annual_growth, na.rm=TRUE)) %>%
  ungroup()

continent_growth <- na.omit(continent_growth)

ggplot(data = continent_growth,aes(x = year, y = cont_annual_growth, group = continent)) +
   geom_line(aes(color = continent)) + labs(x="Year", y="Continental Annual Growth") +
  ggtitle("Continental Annual Growth by Year")
```



2. Calculate the growth rate for each country (across all the years) relative to the mean growth of their region. Which 10 countries growth is the least compared to the mean in their region?

Compared to the mean of their regions, the 10 countries with the least growth are Guinea-Bissau, Cameroon, Cote d'Ivoire, Mali, Liberia, Congo Republic, Central African Republic, Gabon, Senegal, and New Caledonia.

```r
region_growth <- growth_gdp %>%
  group_by(region) %>%
  mutate(region_annual_growth = mean(country_annual_growth, na.rm=TRUE))

region_country <- region_growth %>%
  group_by(country) %>%
  summarise(region_mean= mean(region_annual_growth, na.rm=TRUE),
```

```
            country_mean = mean(country_annual_growth, na.rm=TRUE),
            relative_growth = country_mean - region_mean) %>%
  arrange(relative_growth) %>%
  head(10)

region_country %>% as_tibble()
```

```
## # A tibble: 10 x 4
##    country                   region_mean country_mean relative_growth
##    <fct>                           <dbl>        <dbl>           <dbl>
##  1 Serbia                         0.0334     -0.00695         -0.0403
##  2 Brunei                         0.0582      0.0207          -0.0375
##  3 Georgia                        0.0535      0.0197          -0.0338
##  4 Congo, Dem. Rep.               0.0417      0.00966         -0.0320
##  5 Haiti                          0.0339      0.00333         -0.0305
##  6 Djibouti                       0.0382      0.00950         -0.0287
##  7 Ukraine                        0.0167     -0.0108          -0.0275
##  8 Central African Republic       0.0417      0.0147          -0.0270
##  9 West Bank and Gaza             0.0535      0.0278          -0.0257
## 10 Croatia                        0.0334      0.00771         -0.0257
```

3. For each country, count the number of years they grew above their region mean (Hint: construct a boolean variable for every country-year measure whether it grew above the average). Show the 10 countries with more years above their region average.

```
growth_gdp <- na.omit(growth_gdp)

grew_above_avg <- growth_gdp %>%
  group_by(region) %>%
  mutate(region_annual_growth = mean(country_annual_growth, na.rm=TRUE),
         gdpgrowth_more = ifelse(country_annual_growth > region_annual_growth, 1, 0)) %>%
  ungroup() %>%
  group_by(country) %>%
  summarise(n = sum(gdpgrowth_more)) %>%
  arrange(desc(n)) %>%
  head(10)

grew_above_avg %>% as_tibble()
```

```
## # A tibble: 10 x 2
##    country                n
##    <fct>              <dbl>
##  1 China                 41
##  2 Botswana              39
##  3 Singapore             39
##  4 Chile                 37
##  5 Dominican Republic    36
##  6 Colombia              35
##  7 Malaysia              35
##  8 Australia             34
##  9 Costa Rica            34
## 10 Latvia                34
```

1.2 Exploratory Data Analysis (25 points)

1.2.1 EDA: Exploring variation

1. Explore the distribution of infant_mortality, using the whole sample

a. Describe the overall pattern. Does it fit your expectations? Does it make sense to plot all the years and countries at the same time?

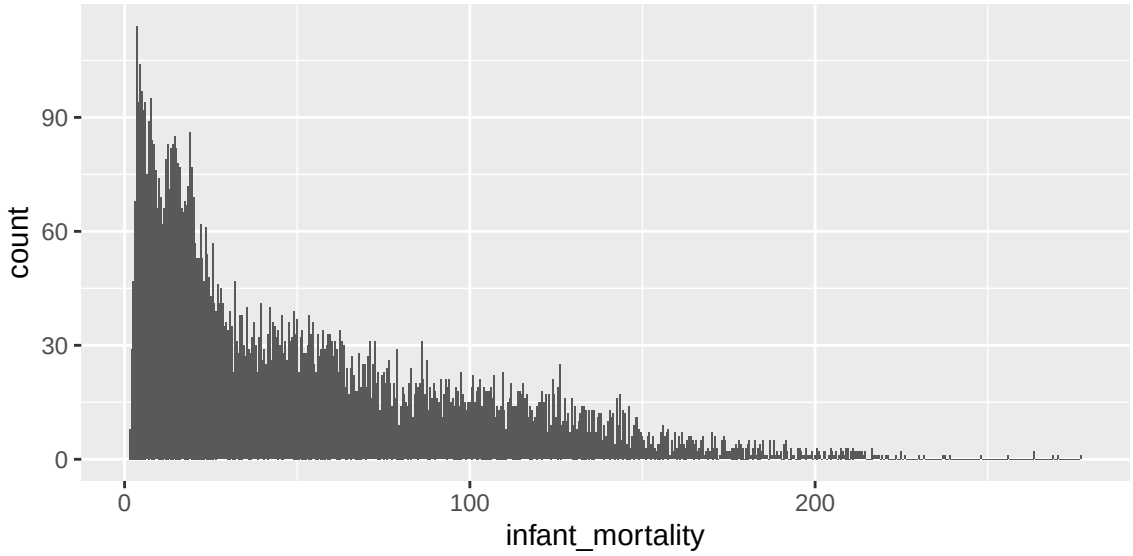
The average infant mortality value is 55. The overall pattern shows a decreasing infant mortality value overall for the whole sample, that is, there are fewer and fewer countries with higher infant mortality values.

Based on plotting infant_mortality vs year, it is not very informative to plot all the years and countries at once. The plot is unable to clearly show the data if we choose to include all the years for 280+ countries. For example, creating a plot grouped by continent is much simpler and clearer to identify data patterns:

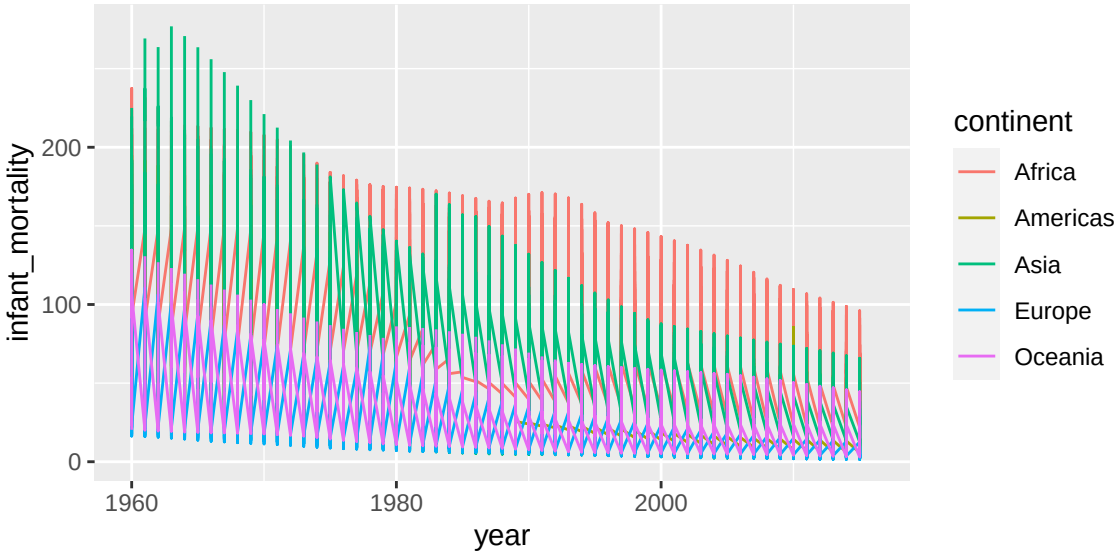
```
summary(gapminder)
```

```
##                 country           year      infant_mortality life_expectancy
##  Albania            :   57   Min.   :1960   Min.   :  1.50   Min.   :13.20
##  Algeria            :   57   1st Qu.:1974   1st Qu.: 16.00   1st Qu.:57.50
##  Angola             :   57   Median :1988   Median : 41.50   Median :67.54
##  Antigua and Barbuda:   57   Mean   :1988   Mean   : 55.31   Mean   :64.81
##  Argentina          :   57   3rd Qu.:2002   3rd Qu.: 85.10   3rd Qu.:73.00
##  Armenia            :   57   Max.   :2016   Max.   :276.90   Max.   :83.90
##  (Other)            :10203                  NA's   :1453
##    fertility       population             gdp               continent
##  Min.   :0.840   Min.   :3.124e+04   Min.   :4.040e+07   Africa  :2907
##  1st Qu.:2.200   1st Qu.:1.333e+06   1st Qu.:1.846e+09   Americas:2052
##  Median :3.750   Median :5.009e+06   Median :7.794e+09   Asia    :2679
##  Mean   :4.084   Mean   :2.701e+07   Mean   :1.480e+11   Europe  :2223
##  3rd Qu.:6.000   3rd Qu.:1.523e+07   3rd Qu.:5.540e+10   Oceania : 684
##  Max.   :9.220   Max.   :1.376e+09   Max.   :1.174e+13
##  NA's   :187     NA's   :185         NA's   :2972
##              region
##  Western Asia   :1026
##  Eastern Africa : 912
##  Western Africa : 912
##  Caribbean      : 741
##  South America  : 684
##  Southern Europe: 684
##  (Other)        :5586
```

```
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = infant_mortality), binwidth = 0.5)
```



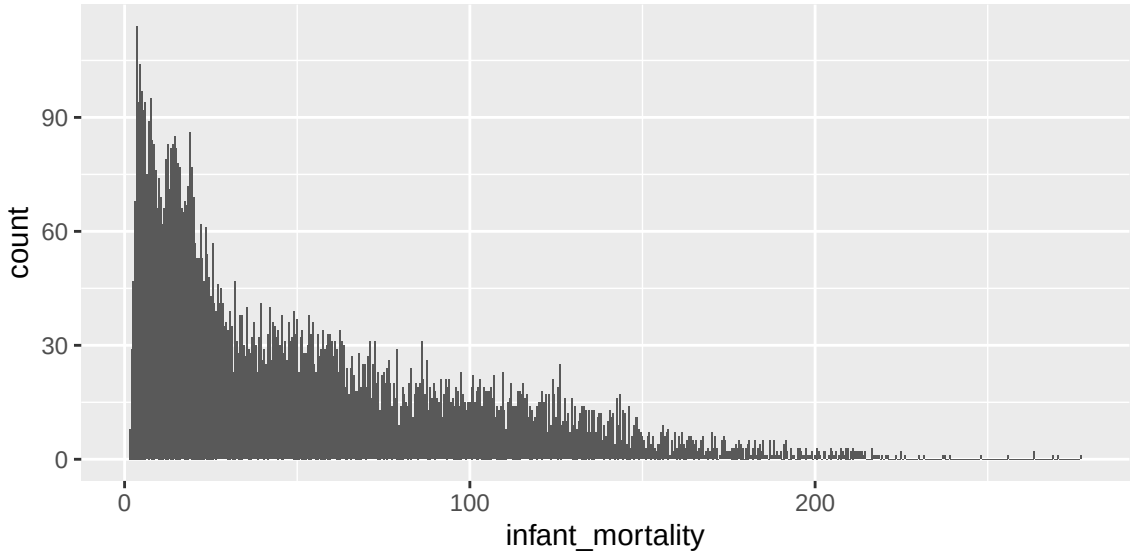

```
ggplot(data = gapminder,aes(x = year, y = infant_mortality, group = continent)) +
   geom_line(aes(color = continent))
```



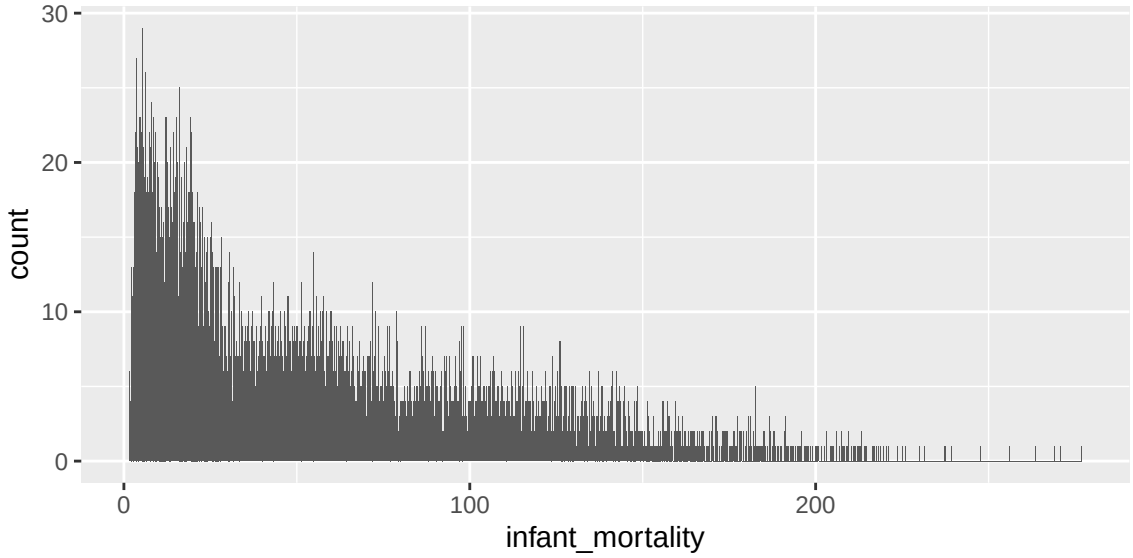
b. Play around with bin sizes. Do you discover anything unusual or surprising?

As we adjust bin sizes, the count of country/year pairs decreases when comparing the distribution plots of infant mortality, for example. In this case, bin size controls how many observations go in each unit of infant mortality.

```
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = infant_mortality), binwidth = 0.5)
```




```
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = infant_mortality), binwidth = 0.1)
```

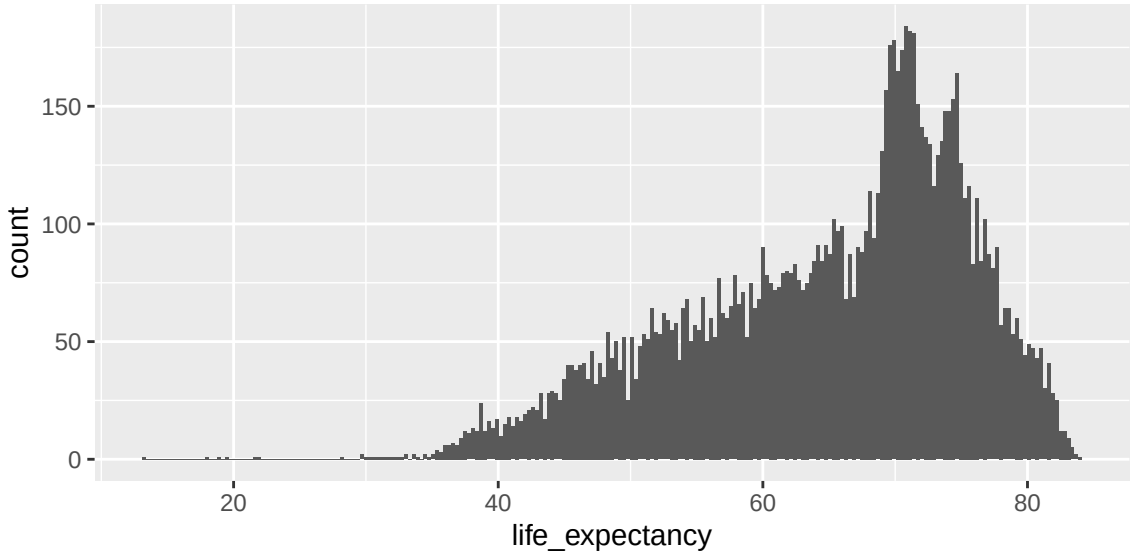


2. Explore the distribution of life_expectancy.

a. Describe the overall pattern using a graph.

From the graph below we see life expectancy distribution that has a left-skewed pattern. This means life expectancy overall for the dataset is elevated, showing many countries expect their populations to live into their 70s and 80s. Based on the graph alone, one can infer that the average life expectancy is above 60 years of age overall. We can confirm this by using summary().

```
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = life_expectancy), binwidth = 0.3)
```



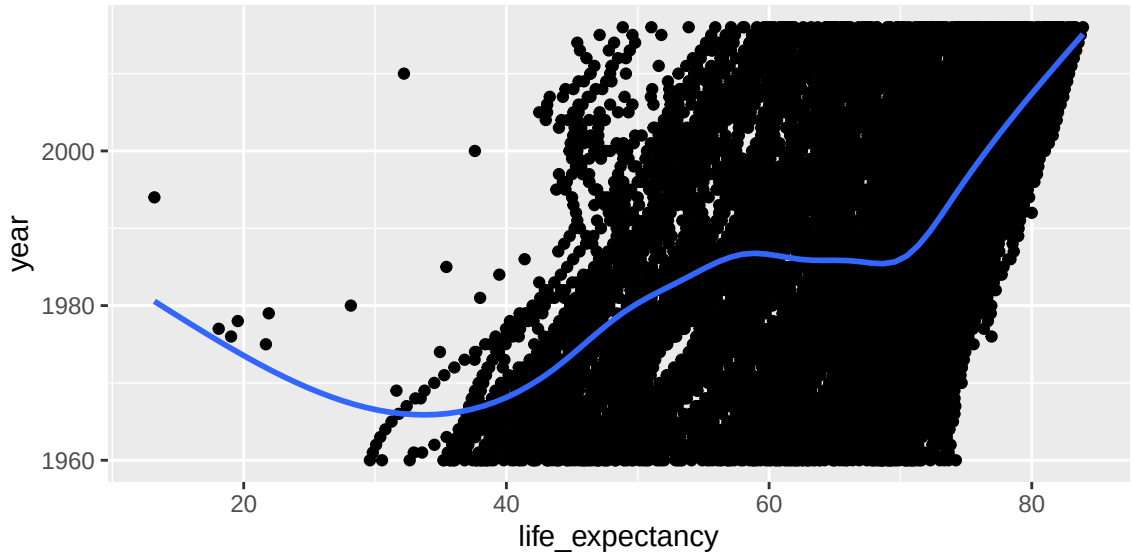

```
summary(gapminder$life_expectancy)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.20   57.50   67.54   64.81   73.00   83.90
```

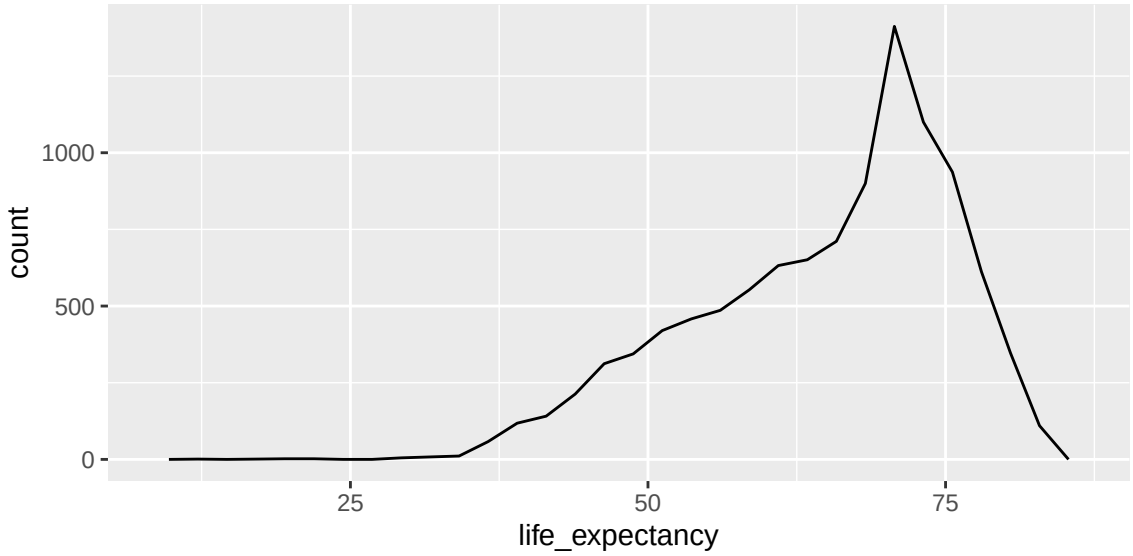
The following graphs examine life expectancy versus year, including a trendline, and shows the data in a couple other geom variations.

```
ggplot(data = gapminder,
       mapping = aes(x = life_expectancy, y = year)) +
      geom_point() +
      geom_smooth(se=FALSE)
```

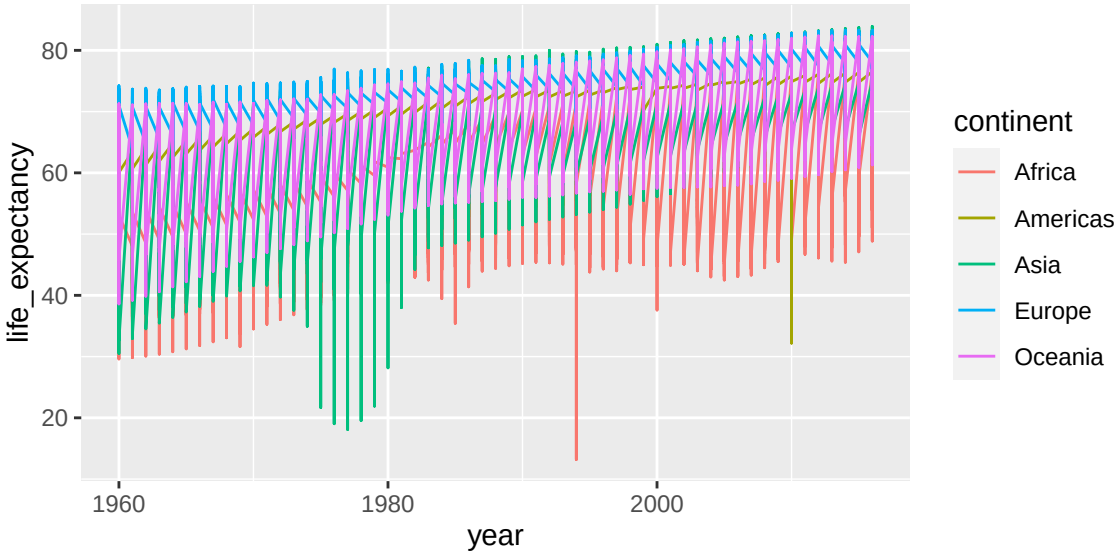
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
gapminder %>%
  ggplot(aes(x=life_expectancy)) +
  geom_freqpoly()
```




```
ggplot(data = gapminder,aes(x = year, y = life_expectancy, group = continent)) +
   geom_line(aes(color = continent))
```



The overall pattern for the years covered in the data is gradually increasing across continents. Specifically, the lower-end life expectancy is much more variable than the life expectancy max values. In terms of which continent has the highest life expectancy, Europe is highest and also the least variable, followed by the Americas. Asia and Africa has the most variable and lowest life expectancy patterns.

b. How many country-year pairs report a life expectancy under 40? How many over 80? Find the country-year pair with the highest life expectancy

```
countryyear_pairs <- gapminder %>%
  group_by(country, year) %>%
  filter(life_expectancy < 40)

nrow(countryyear_pairs)
```

```
## [1] 198
```

```
countryyear_pairs2 <- gapminder %>%
  group_by(country, year) %>%
  filter(life_expectancy > 80)

nrow(countryyear_pairs2)
```

```
## [1] 334
```

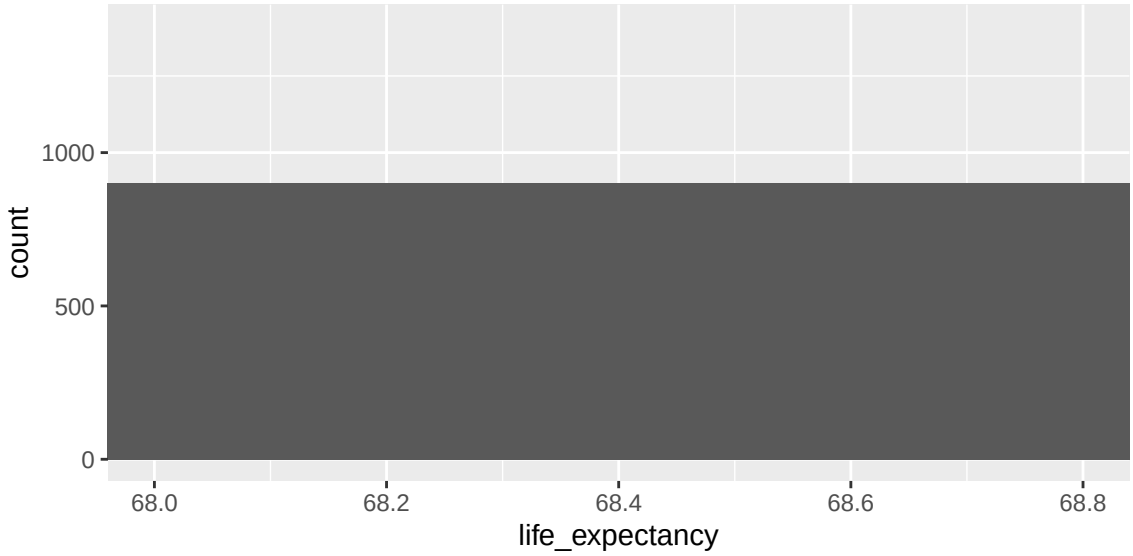
```
gapminder %>%
  arrange(desc(life_expectancy)) %>%
  head(1)
```

```
##            country year infant_mortality life_expectancy fertility population
## 1 Hong Kong, China 2016               NA            83.9        NA         NA
##   gdp continent       region
## 1  NA      Asia Eastern Asia
```

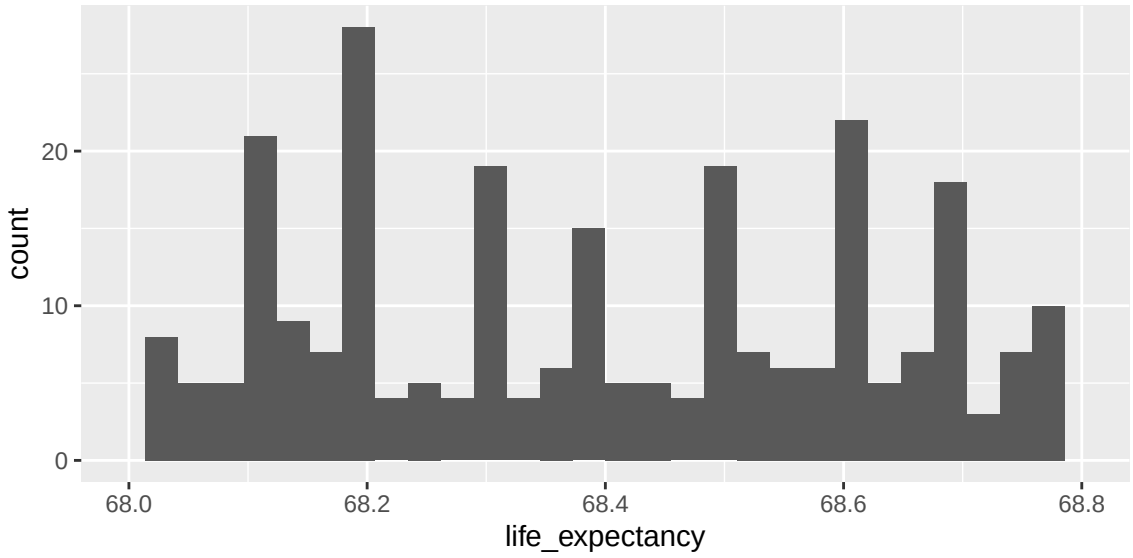
3. Compare and contrast coord_cartesian() vs xlim() or ylim() when zooming in on a histogram. What happens if you leave binwidth unset? What happens if you try and zoom so only half a bar shows?

We go through a series of different plots below and specifically find that with binwidth unset, xlim() does not show the half bar. However, when binwidth is set to a value, the plot generated using either coord_cartesian or xlim will be almost identical.

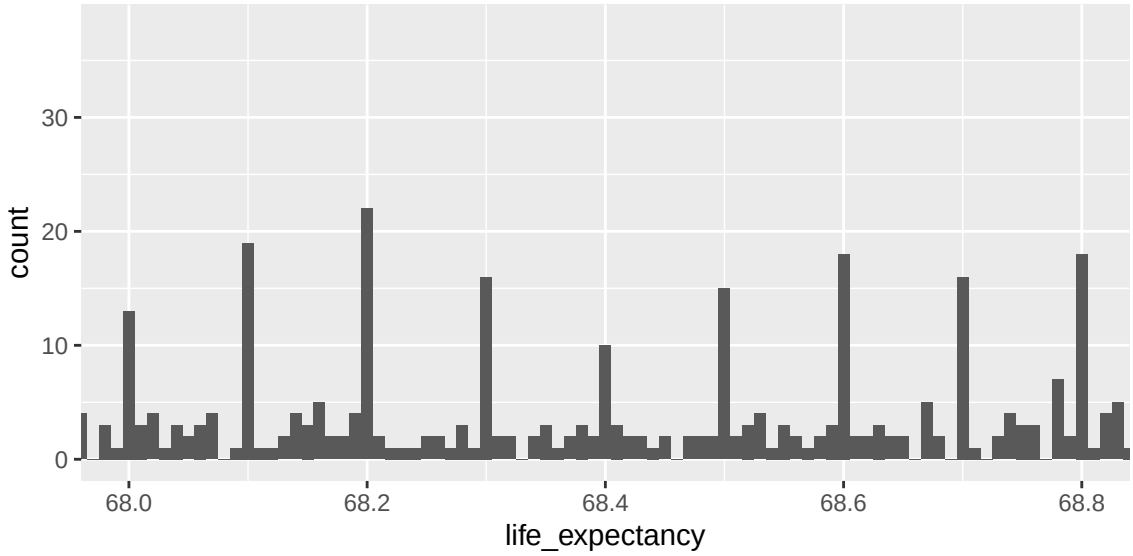
```
# Plot using coord_cartesian() with binwidth unset. We notice different a bar, essentially
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = life_expectancy)) +
  coord_cartesian(xlim = c(68, 68.8))
```



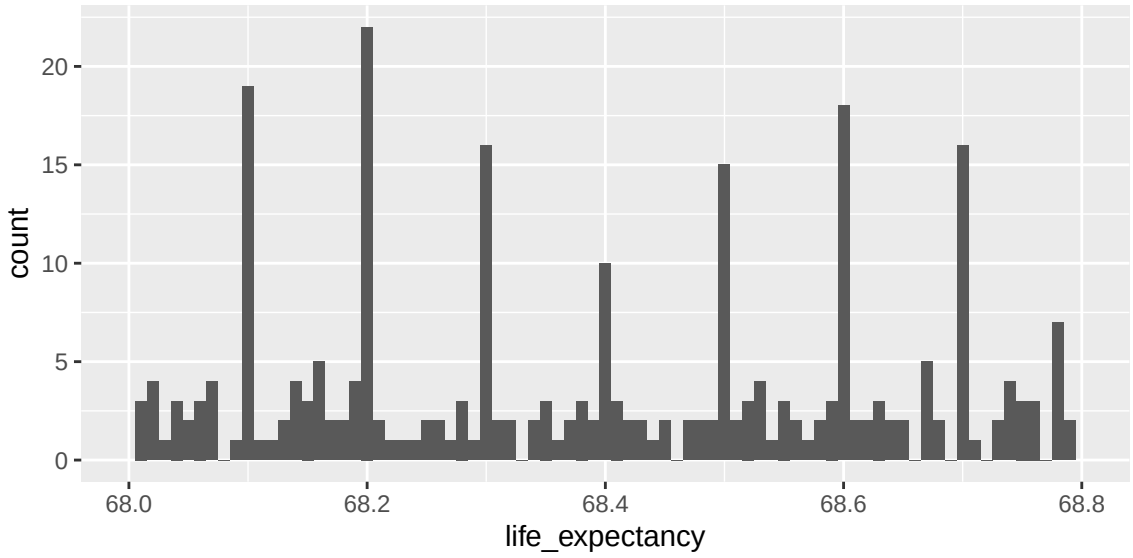

```
# Plot using xlim() with binwidth unset. We notice different bar heights within our xlim.
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = life_expectancy)) + xlim(c(68, 68.8))
```




```
# Plot using coord_cartesian() with binwidth set. We notice very different bar heights within our xlim.
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = life_expectancy), binwidth = 0.01) +
  coord_cartesian(xlim = c(68, 68.8))
```




```
# Plot using xlim() with binwidth set. We notice a very similar plot compared to using coord_cartesian(
ggplot(data = gapminder) +
  geom_histogram(mapping = aes(x = life_expectancy), binwidth = 0.01) +
  xlim(c(68, 68.8))
```



1.2.2 EDA: Navigating NAs (10 points)

1. What happens to missing values on a histogram? What happens to missing values in a bar chart? Why is there a difference?

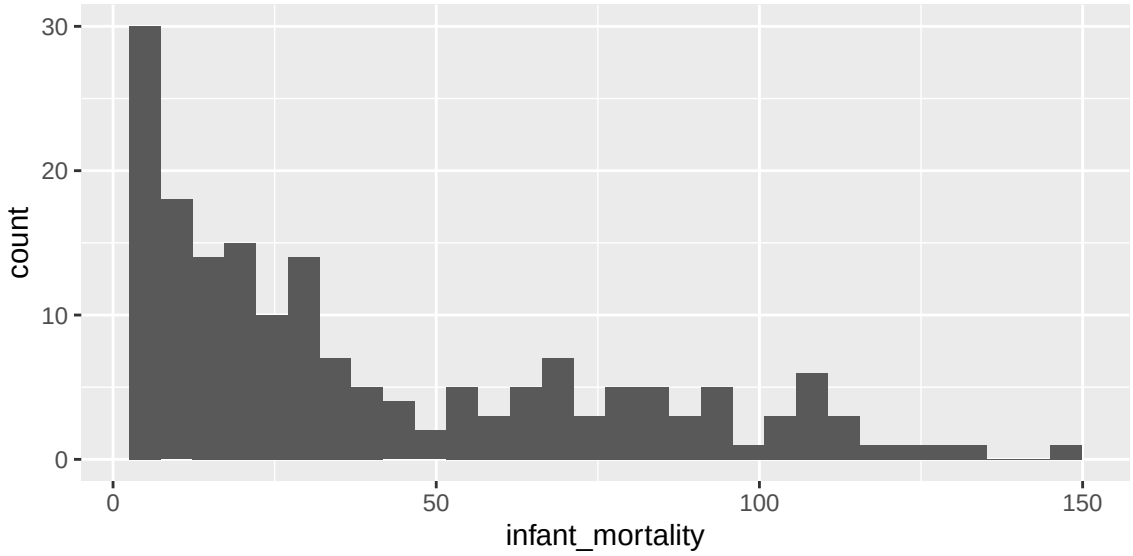
In a bar chart, NA is considered as just another category. In a histogram, NA is ignored because the x axis has order. geom_histogram() removed rows with NA values; Apparently geom_bar() doesn't remove NA, but rather treat it as another factor or category.

```r
gapminder_1999 <- gapminder %>%
  filter(year == "1999")

ggplot(data = gapminder_1999) +
  geom_histogram(aes(x = infant_mortality))
```

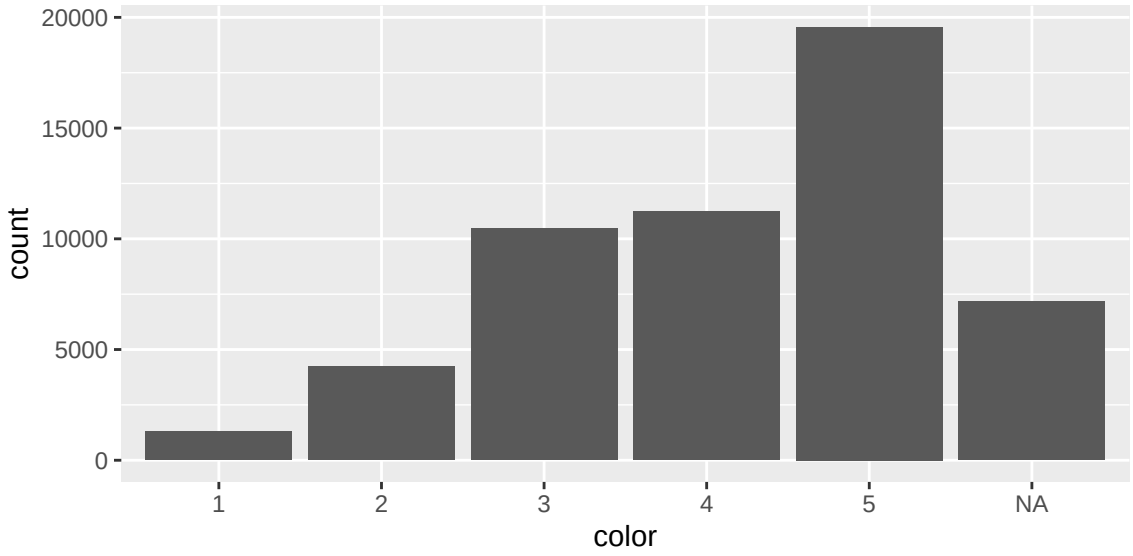
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 7 rows containing non-finite values (stat_bin).
```




```r
color_NA <- diamonds %>% mutate(color = as.factor(ifelse(y > 7, NA, cut)))

ggplot(data = color_NA) +
  geom_bar(aes(x = color))
```



2. What does na.rm = TRUE do in mean() and sum()?

Excluding NA values by using na.rm = TRUE will calculate the mathematical operation for all non-missing values in the dataset.

```
mean(gapminder$gdp, na.rm = TRUE)
```

```
## [1] 147954410013
```

```
sum(gapminder$gdp, na.rm = TRUE)
```

```
## [1] 1.120459e+15
```

2 Exploratory Data Analysis: Covariation (40 points)

We will use the diamonds data set for the next section

1. Looking at the table below it appears that fair is nearly the best cut of diamond and ideal the worst cut. But, the opposite is true!

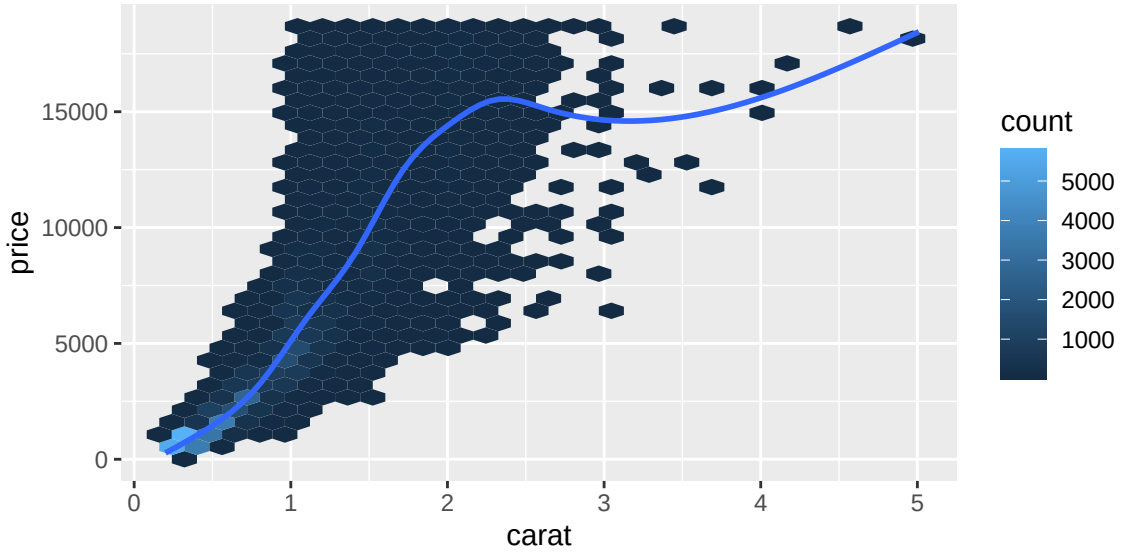
```
tibble(
  cut = c("Fair", "Good", "Very Good", "Premium", "Ideal"),
  mean_price = c(4359, 3929, 3982, 4584, 3458)
)
```

```
## # A tibble: 5 x 2
##   cut       mean_price
##   <chr>          <dbl>
## 1 Fair            4359
## 2 Good            3929
## 3 Very Good       3982
## 4 Premium         4584
## 5 Ideal           3458
```

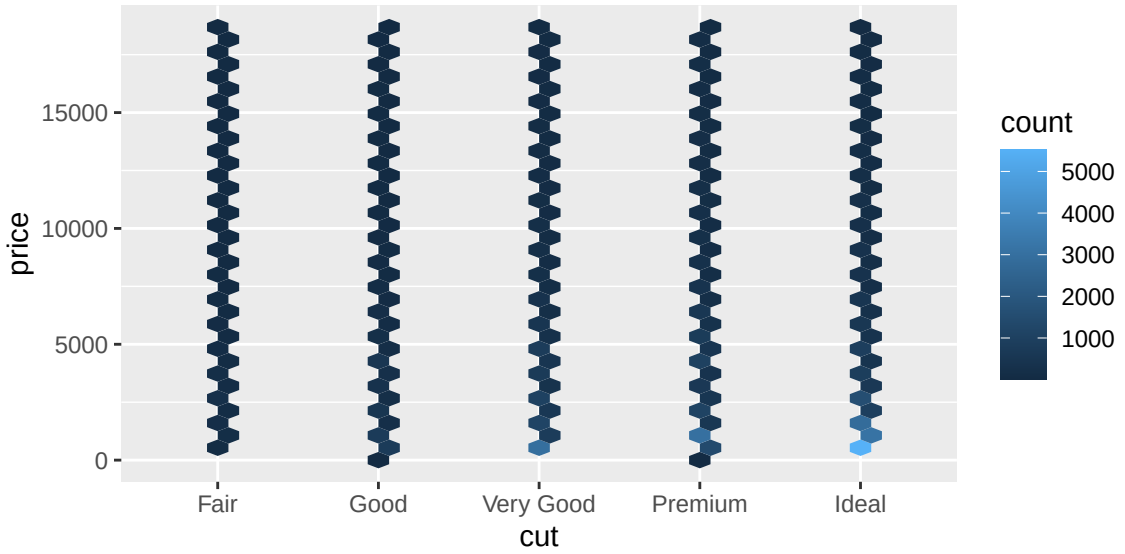
a. What variable in the diamonds dataset is most important for predicting the price of a diamond?

We observe that carat has the highest correlation to price compared to other variables. This makes it the most important variable in predicting price of diamonds. To make this comparison, we must convert the ordered categorical variables into the same type (numeric) as price.

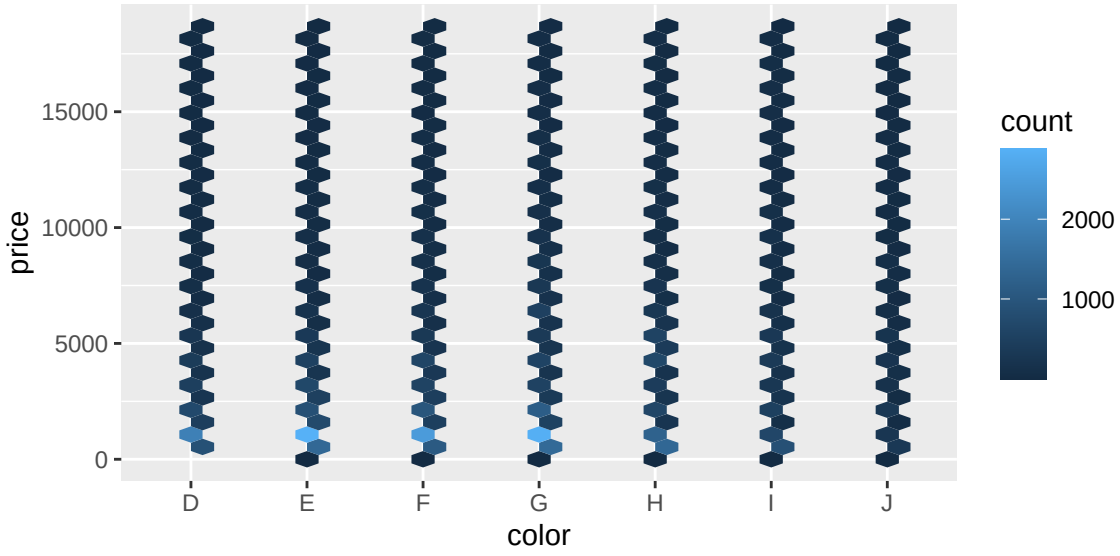
```
ggplot(data = diamonds,
       mapping = aes(x = carat, y = price)) +
      geom_hex() +
      geom_smooth(se=FALSE)
```



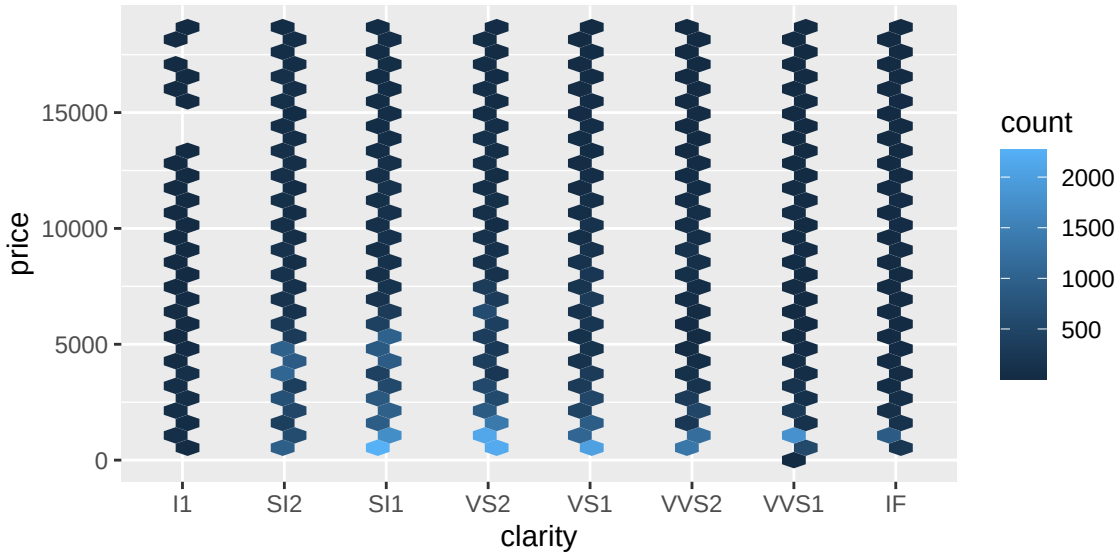

```
ggplot(data = diamonds,
       mapping = aes(x = cut, y = price)) +
      geom_hex() +
      geom_smooth(se=FALSE)
```



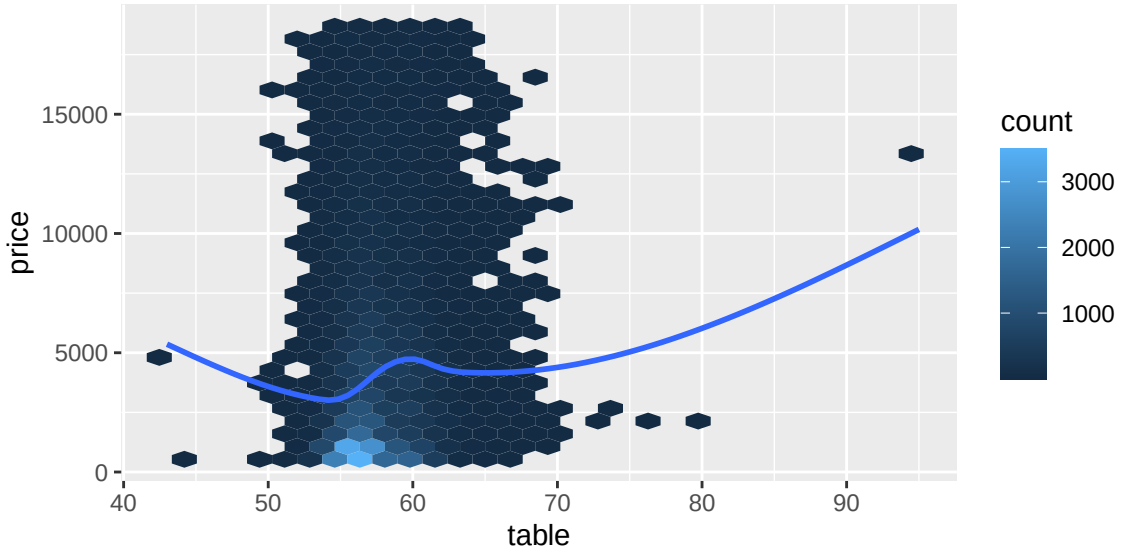

```
ggplot(data = diamonds,
       mapping = aes(x = color, y = price)) +
      geom_hex() +
      geom_smooth(se=FALSE)
```



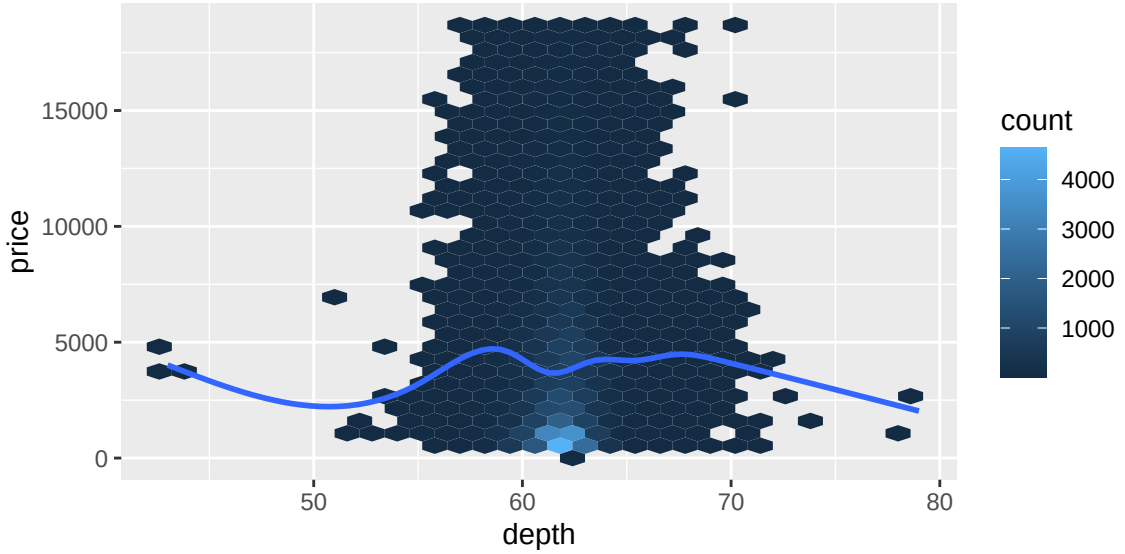

```
ggplot(data = diamonds,
       mapping = aes(x = clarity, y = price)) +
      geom_hex() +
      geom_smooth(se=FALSE)
```




```
ggplot(data = diamonds,
       mapping = aes(x = table, y = price)) +
      geom_hex() +
      geom_smooth(se=FALSE)
```



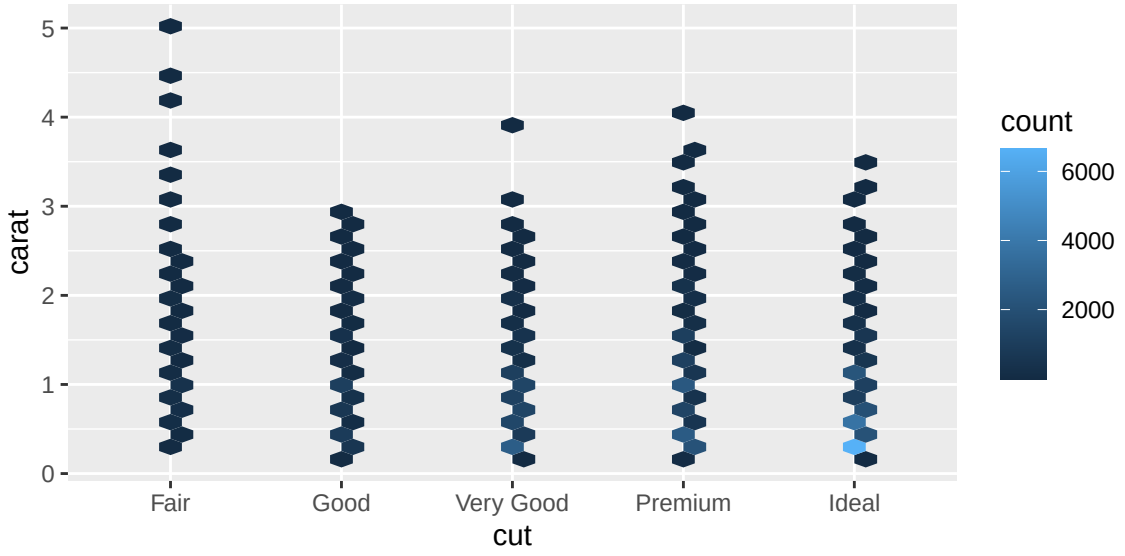

```
ggplot(data = diamonds,
       mapping = aes(x = depth, y = price)) +
      geom_hex() +
      geom_smooth(se=FALSE)
```



b. How is that variable correlated with cut?

When reading the graph below describing correlation of carat with the cut variable, we see a slightly negative correlation, if any correlation at all. This means that heavier diamonds are more likely to have a lower quality cut.

```
ggplot(data = diamonds,
       mapping = aes(x = cut, y = carat)) +
      geom_hex() +
      geom_smooth(se=FALSE)
```



c. Explain why the table above is misleading.

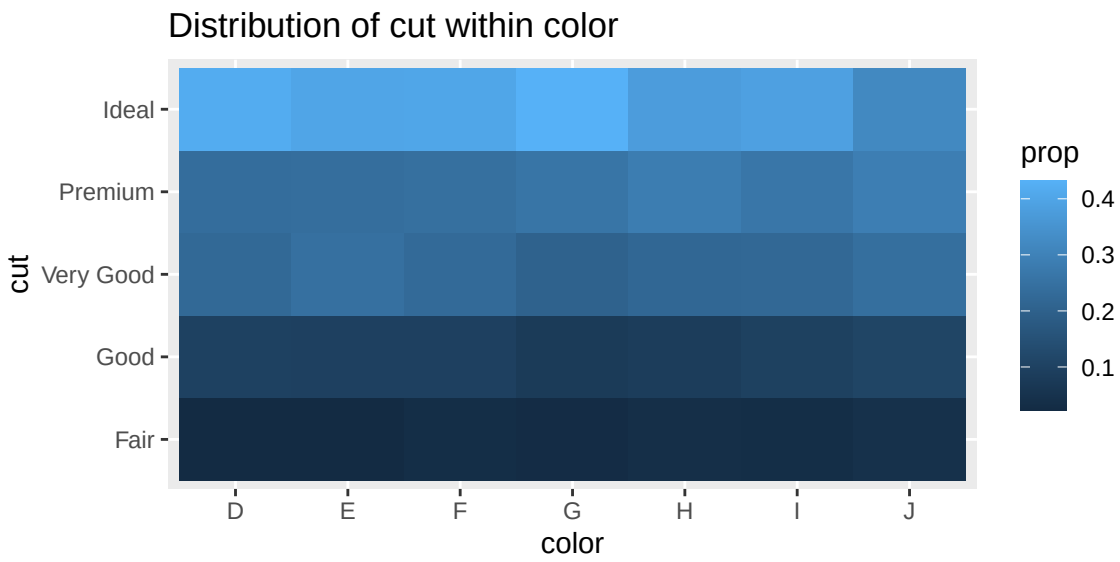
The table above is misleading because the mean price associated with different cuts is varied where the quality does not match their price levels. For example, fair cut diamonds are priced higher than good, very good, and ideal cut diamonds. Given the words chosen describing the diamond cut, one may expect the table to show mean price from lowest to highest, with fair being the lowest and ideal being the highest mean price.

2. Recreate the count dataframe (see textbook for plot in 7.5.2) with adjustments to more clearly show the distribution of cut within color.

a. Which cut is most common in every color category?

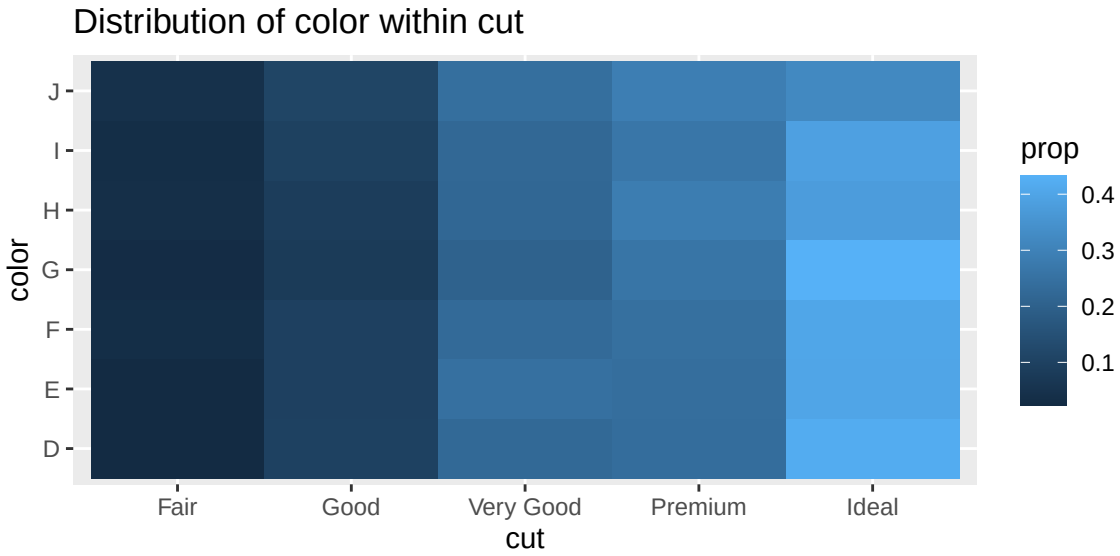
Ideal cut is the most common within color.

```
diamonds %>%
  count(color, cut) %>%
  group_by(color) %>%
  mutate(prop = n / sum(n)) %>%
  ggplot() +
      geom_tile(mapping = aes(x = color, y = cut, fill = prop)) +
  labs(title = 'Distribution of cut within color')
```



b. Repeat the exercise again to show distribution of colour within cut.

```
diamonds %>%
  count(color, cut) %>%
  group_by(color) %>%
  mutate(prop = n / sum(n)) %>%
  ggplot() +
      geom_tile(mapping = aes(x = cut, y = color, fill = prop)) +
  labs(title = 'Distribution of color within cut')
```

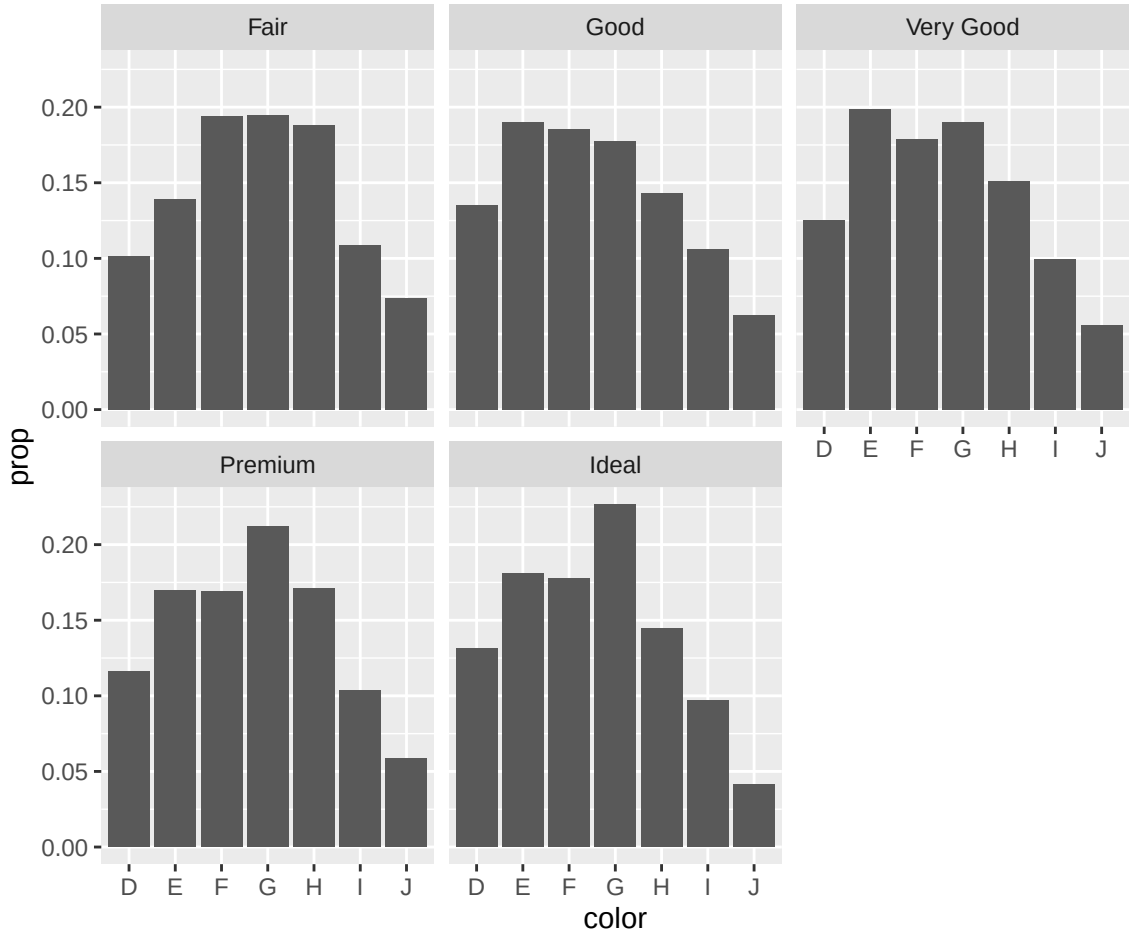


c. Using the dataframe you just produced as input, reproduce the following graph.

```
diamonds %>%
  count(color, cut) %>%
  mutate(prop = n / sum(n))
```

```
## # A tibble: 35 x 4
##    color cut           n    prop
##    <ord> <ord>     <int>   <dbl>
##  1 D     Fair        163 0.00302
##  2 D     Good        662 0.0123
##  3 D     Very Good  1513 0.0280
##  4 D     Premium    1603 0.0297
##  5 D     Ideal      2834 0.0525
##  6 E     Fair        224 0.00415
##  7 E     Good        933 0.0173
##  8 E     Very Good  2400 0.0445
##  9 E     Premium    2337 0.0433
## 10 E     Ideal      3903 0.0724
## # ... with 25 more rows
```

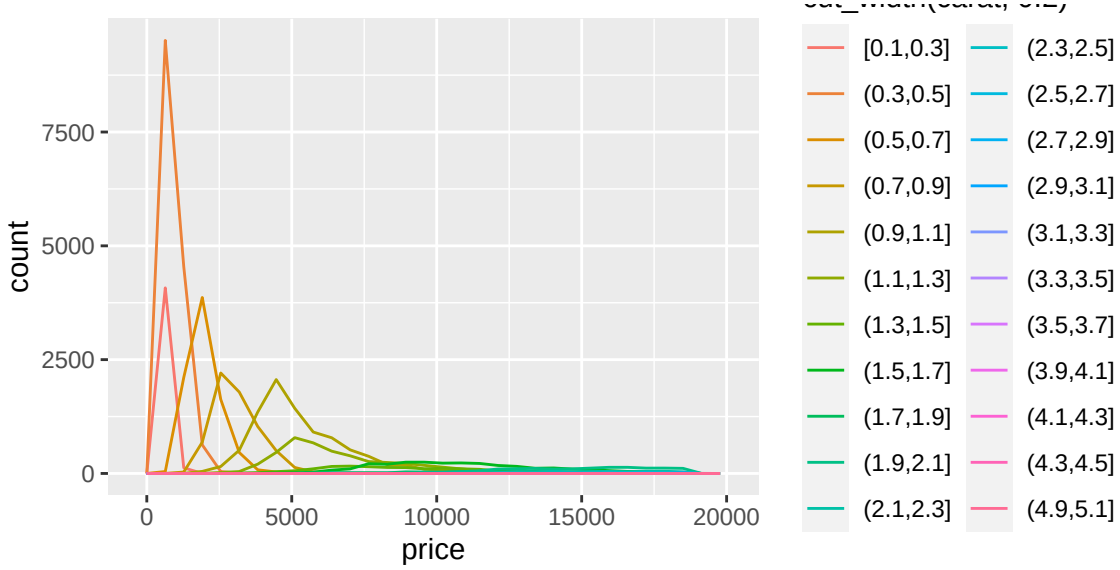
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = color, y = ..prop.., group = 1)) +
facet_wrap(vars(cut))
```



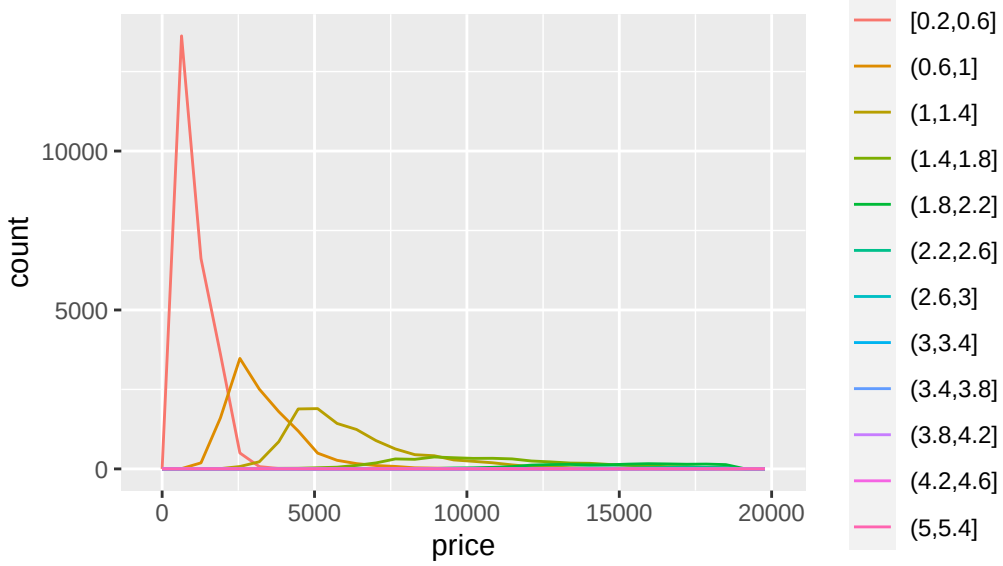
3. Instead of summarising the conditional distribution of price by carat size with a boxplot (see 7.5.3), you could use a frequency polygon, where you map binned carat data to the color aesthetic.

a. Make a frequency polygon using cut_width() and another using cut_number(). Adjust the parameters until you think the graphs are most useful.

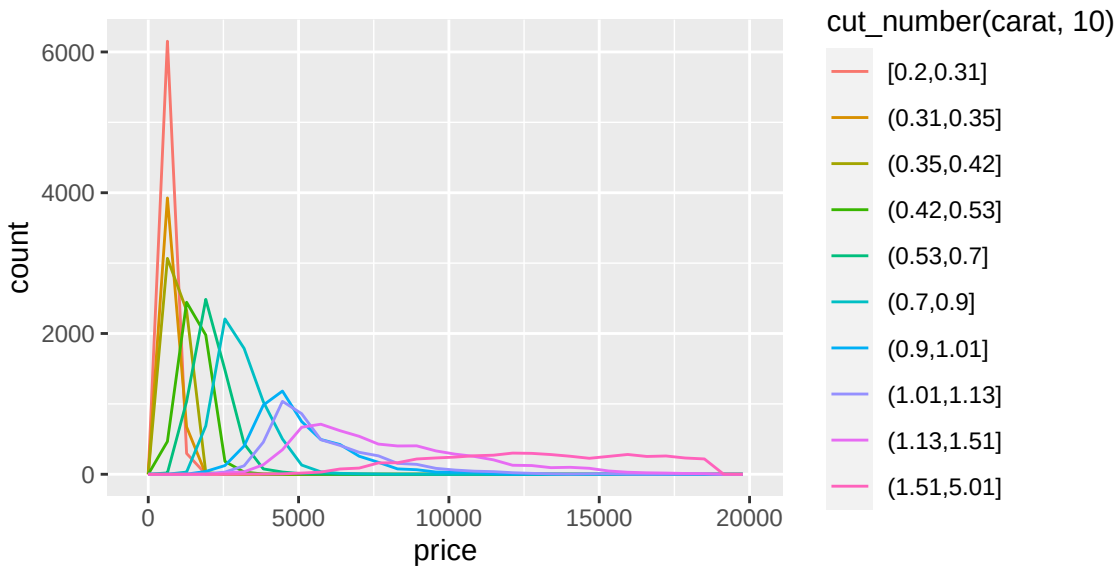
```
diamonds %>% ggplot() +
  geom_freqpoly(mapping = aes(x = price,
                              color = cut_width(carat, .2)), bins = 30)
```



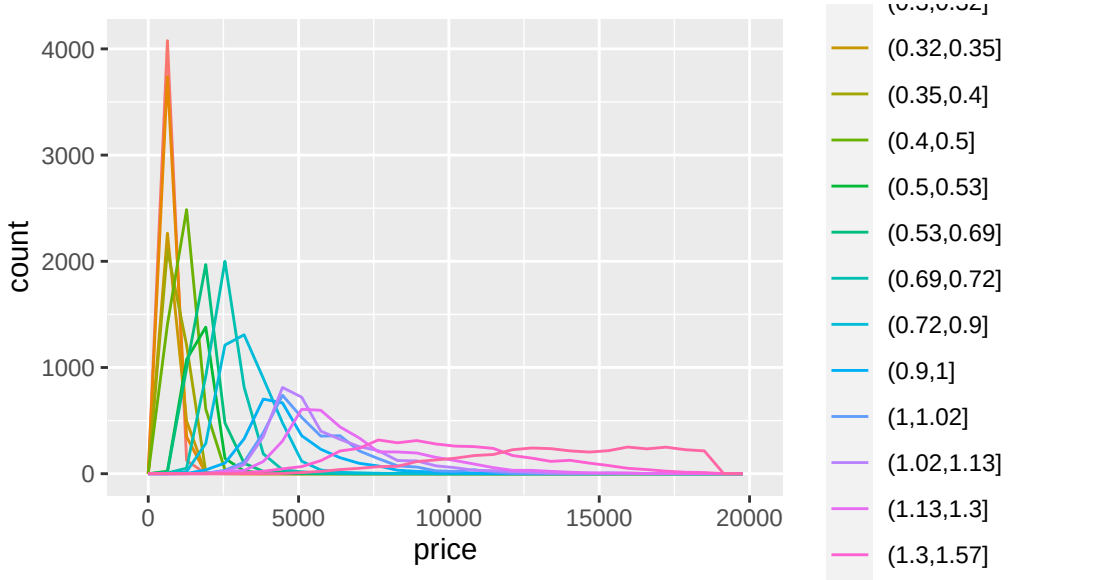

```
diamonds %>% ggplot() +
  geom_freqpoly(mapping = aes(x = price,
                              color = cut_width(carat, .4)), bins = 30)
```




```
diamonds %>% ggplot() +
  geom_freqpoly(mapping = aes(x = price,
                              color = cut_number(carat, 10)), bins = 30)
```




```
diamonds %>% ggplot() +
  geom_freqpoly(mapping = aes(x = price,
                              color = cut_number(carat, 15)), bins = 30)
```



b. How does this affect your visualisation of joint distribution of carat and price and your ability to interpret them?

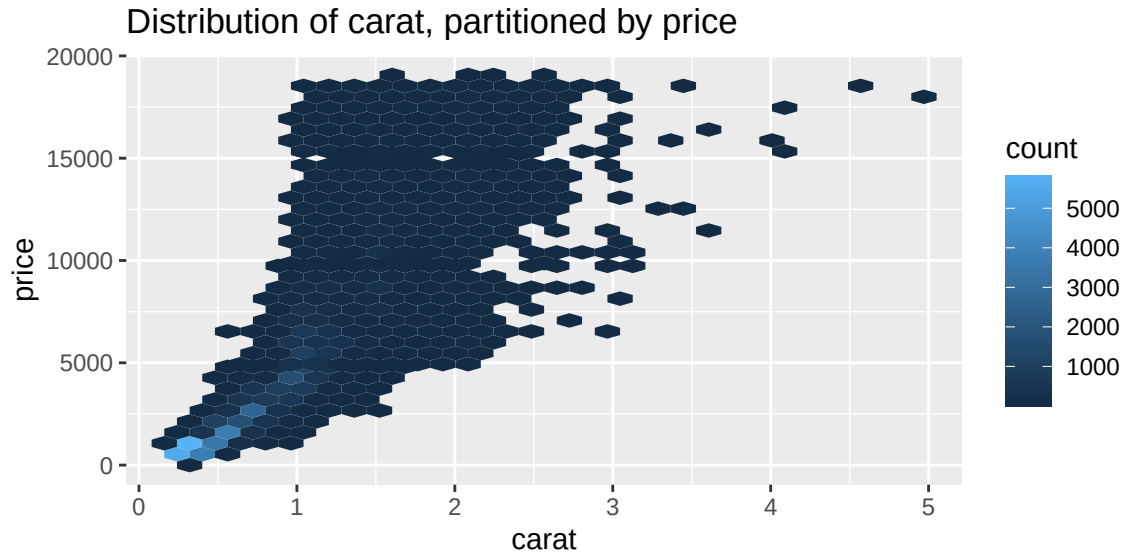
Setting the cut_width too small will have too many categories. Some of the categories will have very few observations, resulting in polygons that are flat and close to the x-axis. We compare the cut_width of .2 and .4.

Since most of the diamonds are below 1 carat, in both plots, the polygons of above 1 carat are flat, and some are not distinguishable from others. Conversely, cut_number() ensures the same number of observations in each group.

4. Visualize the distribution of carat, partitioned by price. The graph should have a clear story accentuated by a descriptive title.

Using geom_density() and partitioning by price with cut_width, it is not surprising to see that diamonds of higher carat are associated with higher price in general.

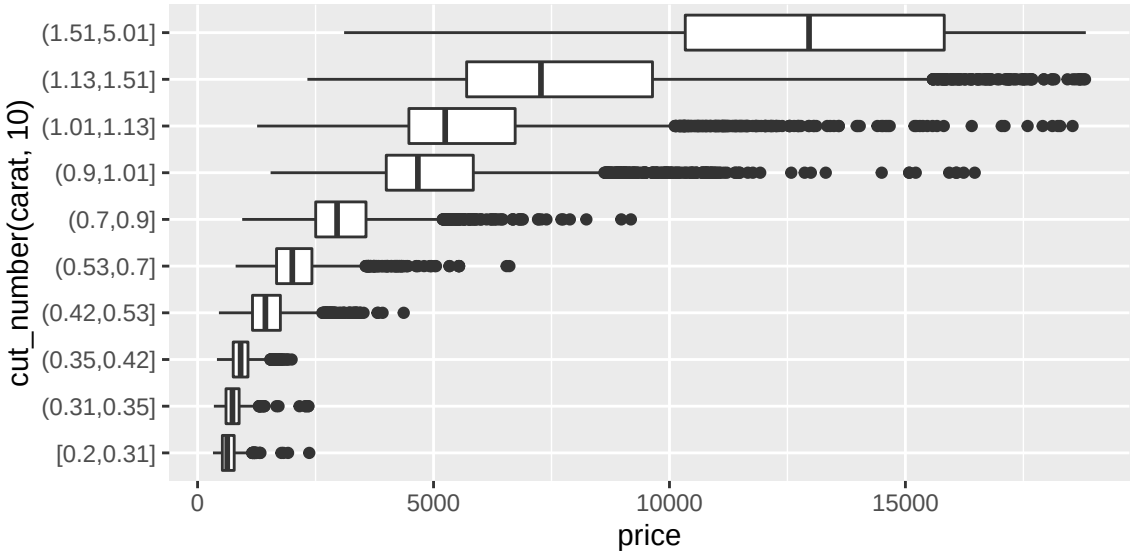
```
diamonds %>% ggplot() +
  geom_hex(mapping = aes(x = carat, y = price,
                         group = cut_width(price, 5000, boundary = 0))) +
  labs(title = 'Distribution of carat, partitioned by price')
```



5. How does the price distribution of very large diamonds compare to small diamonds? Is it as you expect, or does it surprise you? Why?

The price distribution (aka price range) of very large diamonds is much more variable than smaller diamonds. There could be other factors, such as cut, clarity, and color that have heavier influence on the price of larger diamonds.

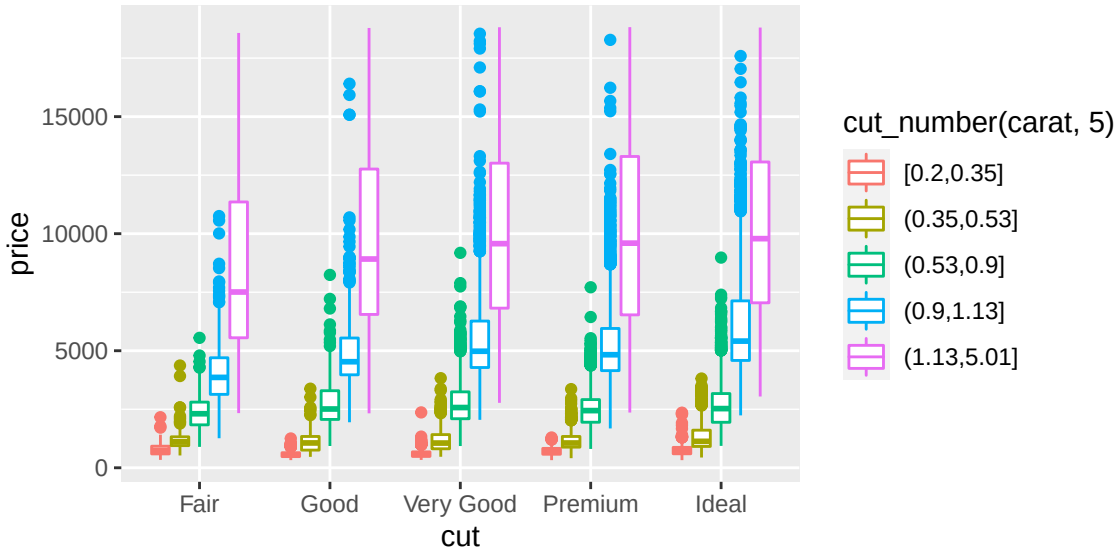
```
diamonds %>% ggplot +
  geom_boxplot(mapping = aes(x = cut_number(carat, 10),
                             y = price)) +
  coord_flip()
```



6. So far we've focused on visualizing covariation of two variables. We can add a third dimension in several ways. For example, we could map to a third aesthetic or add facets. Combine two of the techniques you've learned in this class to visualize the combined distribution of cut, carat, and price.

We can use boxplot:

```
diamonds %>% ggplot() +
  geom_boxplot(mapping = aes(x = cut, y = price,
                             color = cut_number(carat, 5)))
```

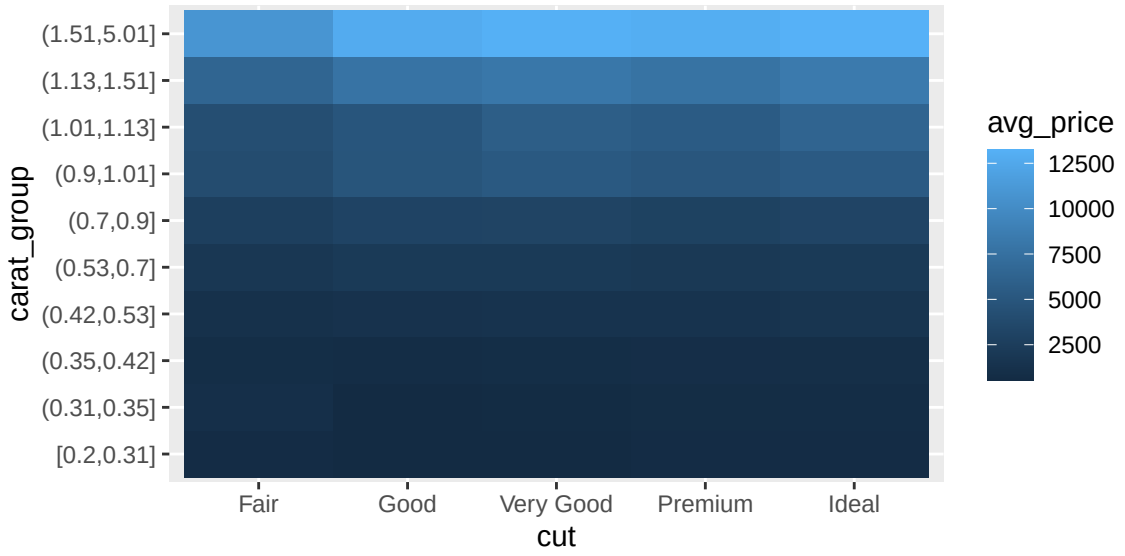


Heatmap:

```
diamonds %>% mutate(carat_group = cut_number(carat, 10)) %>%
  group_by(cut, carat_group) %>%
  summarize(avg_price = mean(price)) %>%
```

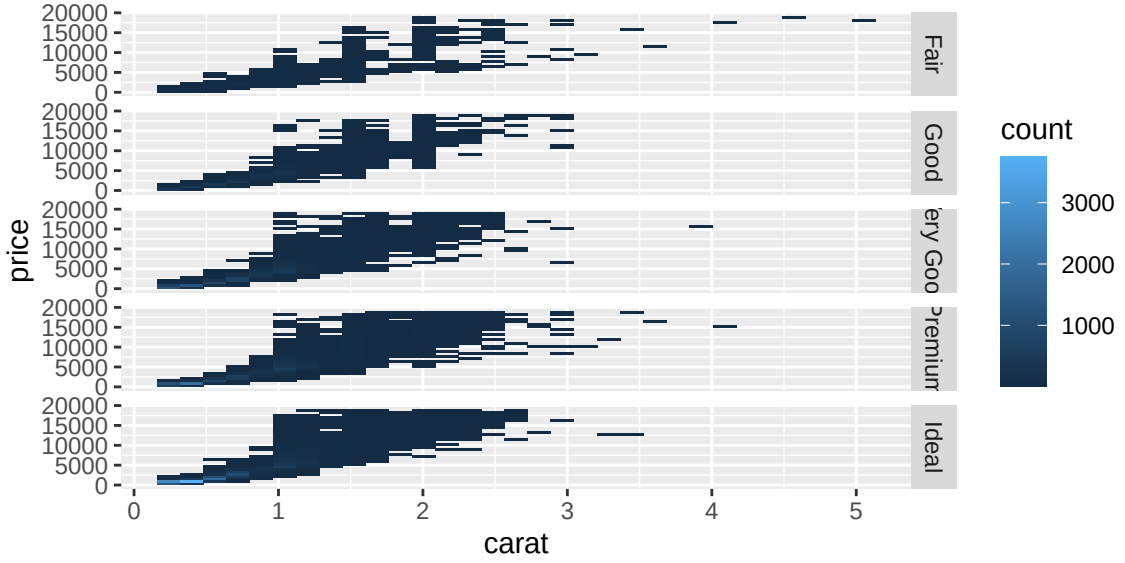
```
  ggplot() +
  geom_tile(mapping = aes(x = cut, y = carat_group,
                          fill = avg_price))
```

```
## `summarise()` has grouped output by 'cut'. You can override using the `.groups` argument.
```



Facetted geom_bin2d():

```
diamonds %>% ggplot() +
  geom_bin2d(mapping = aes(x = carat, y = price)) +
  facet_grid(vars(cut))
```

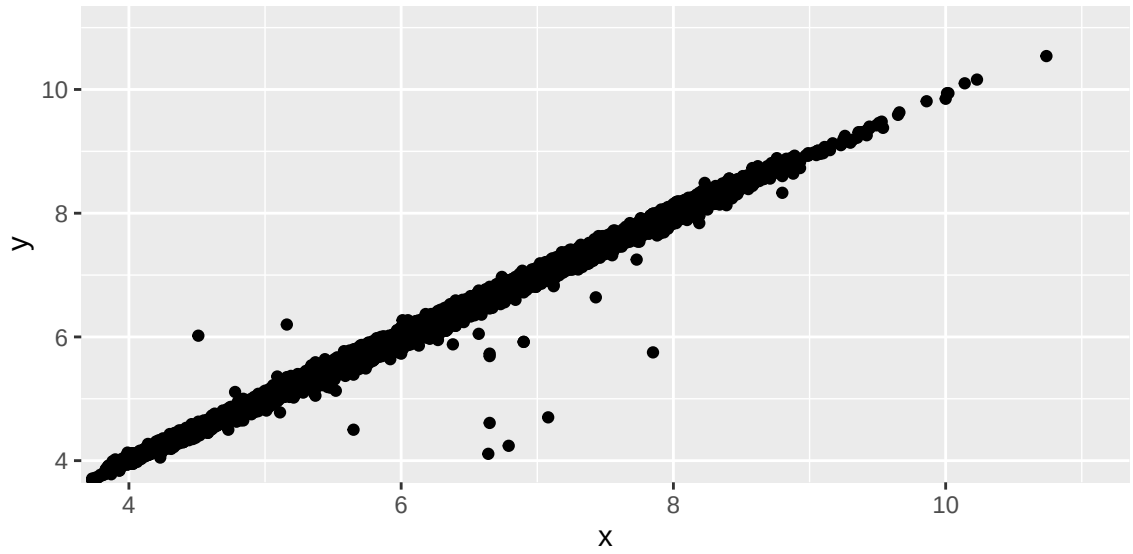


7. Two dimensional plots reveal outliers that are not visible in one dimensional plots. For example, some points in the plot have an unusual combination of x and y values, which makes the points outliers even

though their x and y values appear normal when examined separately. Consider the two plots below. Which geom is better for identifying outliers? Why?

Considering the two plots below, we are better equipped to identify individual outliers in our dataset if we use the scatterplot, or geom_line, rather than geom_histogram, which uses binned counts for allotting data. By using geom_line we're able to see exactly where an individual precisely point lies, which is not possible if it is contained within a bin.

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = x, y = y)) +
  coord_cartesian(xlim = c(4, 11), ylim = c(4, 11))
```




```
ggplot(data = diamonds) +
  geom_bin2d(mapping = aes(x = x, y = y)) +
  coord_cartesian(xlim = c(4, 11), ylim = c(4, 11))
```